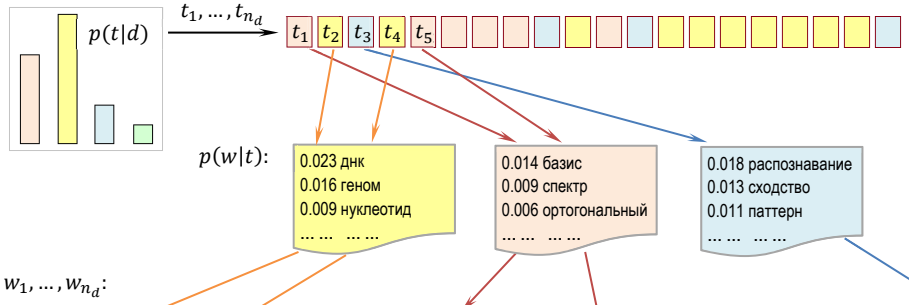
ISSN 2223-3792

Машинное обучение и анализ данных

Октябрь 2013 — Январь 2014 **Том 1, номер 6**



Разработан спектрально-аналитический подход к выявлению размытых протяженных повторов в геномных последовательностях. Метод основан на разномасштабном оценивании сходства нуклеотидных последовательностей в пространстве коэффициентов разложения фрагментов кривых GC- и GA-содержания по классическим ортогональным базисам. Найдены условия оптимальной аппроксимации, обеспечивающие автоматическое распознавание повторов различных видов (прямых и инвертированных, а также тандемных) на спектральной матрице сходства. Метод одинаково хорошо работает на разных масштабах данных. Он позволяет выявлять следы сегментных дупликаций и мегасателлитные участки в геноме, районы синтении при сравнении пары геномов. Его можно использовать для детального изучения фрагментов хромосом (поиска размытых участков с умеренной длиной повторяющегося паттерна).

Машинное обучение и анализ данных

Journal of Machine Learning and Data Analysis

Журнал «Машинное обучение и анализ данных» публикует новые теоретические и обзорные статьи с результатами научных исследований в области теоретических основ информатики и её приложений. Цель журнала — развитие теории машинного обучения, интеллектуального анализа данных и методов проведения вычислительных экспериментов. Принимаются статьи на английском и русском языках.

Журнал включен в российский индекс научного цитирования РИНЦ. Информация о цитировании статей находится на сайте Российского индекса научного цитирования. ISSN 2223-3792, номер свидетельства о регистрации ЭЛ № ФС 77-55486.

- Архив журнала http://www.ccas.ru/jmlda/
- Новостной сайт http://jmlda.org/
- Электронная система подачи статей http://jmlda.org/papers/

Тематика журнала:
— классификация, кластеризация, регрессионный анализ,
— алгебраический подход к проблеме синтеза корректных алгоритмов,
— многомерный статистический анализ,
— выбор моделей и сложность,
— предсказательное моделирование,
— статистическая теория обучения,
— методы прогнозирования временных рядов,
— методы обработки и распознавания сигналов,
— методы оптимизации в задачах машинного обучения и анализа данных,
— методы визуализации данных,
— обработка и распознавание речи и изображений,
— анализ и понимание текста,
— информационный поиск,
— прикладные задачи анализа данных.

Вычислительный центр Российской академии наук
Московский физико-технический институт
Факультет управления и прикладной математики
Кафедра «Интеллектуальные системы»

Москва, 2013

Содержание

Модификации EM-алгоритма для вероятностного тематического моделирования

К. В. Воронцов, А. А. Потапенко
voron@forecsys.ru, anya_potapenko@mail.ru
МФТИ, ВМК МГУ

Вероятностная тематическая модель (ВТМ) строит интерпретируемое представление коллекции текстовых документов, описывая каждый документ дискретным распределением на множестве тем, каждую тему — дискретным распределением на множестве терминов. Рассмотрен обобщенный EM-алгоритм с эвристиками сглаживания, сэмплирования, робастности и разреживания, позволяющий при различных сочетаниях этих эвристик получать как известные тематические модели PLSA (propabilistic latent semantic analysis), LDA (latent Dirichlet allocation), SWB (special words with background), так и новые. Предлагается упрощенный робастный алгоритм, который не требует ни дополнительных вычислительных затрат, ни хранения матрицы параметров шума, и хорошо сочетается с эвристикой разреживания. В экспериментах на двух коллекциях научных публикаций, англоязычной и русскоязычной, подбираются оптимальные сочетания стратегий разреживания и других эвристик. Показывается, что робастная модель без сглаживания позволяет разреживать искомые распределения на 99% без ухудшения качества (перплексии) модели.

Ключевые слова: *вероятностная тематическая модель, байесовский вывод, латентное размещение Дирихле, вероятностный латентный семантический анализ, EM-алгоритм.*

EM-like algorithms for probabilistic topic modeling

K. V. Vorontsov, A. A. Potapenko
Moscow Institute of Physics and Technology, Moscow State University

Probabilistic topic models discover a low-dimensional interpretable representation of text corpora by estimating a multinomial distribution over topics for each document and a multinomial distribution over terms for each topic. A unified family of expectation-maximization (EM) like algorithms with smoothing, sampling, sparsing, and robustness heuristics that can be used in any combinations is considered. The known models PLSA (probabilistic latent semantic analysis), LDA (latent Dirichlet allocation), SWB (special words with background), as well as new ones can be considered as special cases of the presented broad family of models. A new simple robust algorithm suitable for sparse models that do not require to estimate and store a big matrix of noise parameters is proposed. The present authors find experimentally optimal combinations of heuristics with sparsing strategies and discover that sparse robust model without Dirichlet smoothing performs very well and gives more than 99% of zeros in multinomial distributions without loss of perplexity.

Keywords: *probabilistic topic model, bayesian inference, latent dirichlet allocation, probabilistic latent semantic analysis, EM-alorithm.*

Введение

Тематическое моделирование (topic modeling) — одно из современных приложений машинного обучения к анализу текстов, активно развивающееся с конца 1990-х гг. *Вероятностная тематическая модель* коллекции текстовых документов определяет каждую тему как дискретное распределение на множестве терминов, каждый документ — как дис-

кретное распределение на множестве тем. Предполагается, что коллекция документов — это последовательность терминов, выбранных случайно и независимо из смеси этих распределений, и ставится задача восстановления компонент смеси по выборке.

Вероятностные тематические модели применяются для информационного поиска, выявления трендов в научных публикациях и новостных потоках, классификации и категоризации документов, изображений, аудио- и видеосигналов. Многочисленные разновидности и приложения BTM описаны в обзоре [8]. Большинство моделей разрабатываются на основе *латентного размещения Дирихле* LDA [5], с использованием математического аппарата графических моделей и байесовского вывода. Это современный активно развивающийся вероятностный инструментарий, находящий применения повсеместно в задачах анализа данных. Однако в тематическом моделировании он порождает две проблемы.

Во-первых, априорное распределение Дирихле не имеет лингвистических обоснований, не является моделью какого-либо языкового явления, и его применение продиктовано исключительно удобством аналитического интегрирования в байесовском выводе.

Во-вторых, одной из важнейших открытых проблем в теории BTM считается совмещение большого числа функциональных требований в одной модели [8]. Однако байесовский подход оказывается слишком сложным для совмещения более 2–3 требований.

Более простая классическая модель *вероятностного латентного семантического анализа* PLSA [12] не связана с какими-либо параметрическими априорными распределениями. Важной задачей представляется поиск обобщений или модификаций PLSA, имеющих адекватные лингвистические обоснования, обеспечивающие качество моделирования не хуже LDA и упрощающих построение многофункциональных моделей.

В предыдущих работах [1, 19] мы предложили обобщенный алгоритм тематического моделирования, совмещающий эвристики сглаживания, сэмплирования, частого обновления параметров, робастности и разреживания. Комбинируя эти эвристики практически в любых сочетаниях, возможно получать как известные модели PLSA, LDA, CVB0, SWB, так и новые. Эвристики робастности и разреживания имеют очевидный лингвистический смысл и позволяют отказаться от распределения Дирихле без потери качества модели. Было показано, что разреживание позволяет обращать в нуль 90%–95% значений в искомых дискретных распределениях, что позволяет эффективнее решать задачи тематического поиска и классификации больших коллекций текстовых документов. Данная работа является продолжением этих исследований и содержит следующие новые результаты.

1. Предлагаются стратегии постепенного разреживания, позволяющие достигать еще большей разреженности 99% без ухудшения качества модели. Исследуются границы применимости разреживания и его сочетаемость с другими эвристиками.

2. В робастном алгоритме предлагается эвристика постепенного увеличения априорных вероятностей шума и фона.

3. Предлагается упрощенный робастный алгоритм, который разделяет слова на тематические и шумовые без вычисления и хранения матрицы шума.

4. Предлагается новая эвристика разреживания условных распределений тем.

5. Сообщаются результаты экспериментов, в которых модель LDA не имеет значимого преимущества перед PLSA, что противоречит известным экспериментам Д. Блэя и др. [5]. Это объясняется тем, что в [5] сравнивались существенно разные реализации алгоритмов обучения этих двух моделей. Мы же сравниваем реализации, различающиеся только формулой байесовского сглаживания в LDA. Таким образом, мы обосновываем возможность отказа от избыточных вероятностных допущений, присущих LDA, и развития многофункциональных BTM на базе более простого математического аппарата PLSA.

Основные понятия и предположения

В данном разделе сведены основные предположения ВТМ. Предлагаемый набор гипотез отличается от общепринятого тем, в него введены гипотеза разреженности и гипотеза шумовых и фоновых слов. Общепринятая гипотеза об априорных распределениях Дирихле, наоборот, исключена из числа основных предположений и рассматривается далее как одна из дополнительных эвристик. Это отражает точку зрения авторов на то, какие предположения более адекватно описывают особенности текстов на естественном языке.

Пусть D — множество (коллекция) текстовых документов, W — множество (словарь) всех употребляемых в них терминов (слов или словосочетаний). Каждый документ $d \in D$ представляет собой последовательность n_d терминов $(w_1, \ldots, w_{n_d})$ из словаря W. Термин может повторяться в документе много раз.

В роли терминов могут выступать как отдельные слова, так и *ключевые фразы*, выделяемые с помощью тезаурусов [2], статистических критериев [3] или методов машинного обучения [18, 27]. Чтобы не различать формы (склонения, спряжения) слов, на стадии предварительной обработки данных производится либо *лемматизация* — приведение всех терминов к нормальной форме, либо *стемминг* — отбрасывание изменяемых частей слов. Стемминг лучше подходит для английского языка, лемматизация — для русского.

Гипотеза о вероятностном пространстве: *с каждым термином w в документе d может быть связана некоторая тема t из конечного множества тем T, которая не известна; коллекция документов образуется множеством троек (d, w, t), выбираемых случайно и независимо из дискретного распределения $p(d, w, t)$ на конечном множестве $D \times W \times T$.*

Документы $d \in D$ и термины $w \in W$ являются наблюдаемыми переменными, тема $t \in T$ является *латентной* (скрытой) переменной.

Гипотеза независимости элементов выборки: *ни порядок документов в коллекции, ни порядок терминов в документах не важны для выявления тематики.*

Предполагается, что тематику документа можно узнать даже после произвольной перестановки терминов, хотя для человека такой текст теряет смысл. Это предположение называют также гипотезой «мешка слов» (bag of words).

Приняв эту гипотезу, можно перейти к более компактному представлению документа как подмножества $d \subset W$, в котором каждому элементу $w \in d$ поставлено в соответствие число n_{dw} вхождений термина w в документ d.

Гипотеза условной независимости: *условные распределения вероятностей терминов в теме $p(w \mid d, t)$ одинаковы для всех документов $d \in D$ и равны $p(w \mid t)$.*

Это предположение допускает три эквивалентных представления:

$$p(w \mid d, t) = p(w \mid t); \qquad p(d \mid w, t) = p(d \mid t); \qquad p(d, w \mid t) = p(d \mid t)p(w \mid t). \tag{1}$$

Вероятностная модель порождения данных. Согласно определению условной вероятности, формуле полной вероятности и гипотезе условной независимости

$$p(w \mid d) = \sum_{t \in T} p(t \mid d)\, p(w \mid t), \tag{2}$$

где $p(t \mid d)$ и $p(w \mid t)$ — искомые распределения. Согласно модели порождения данных (2), коллекция D — это выборка наблюдений (d, w), генерируемых Алгоритмом 1. Процесс порождения последовательности слов документа показан на рис. 1.

Алгоритм 1 Порождение коллекции текстов с помощью вероятностной модели.

Вход: распределения $p(w\,|\,t)$, $p(t\,|\,d)$;
Выход: выборка пар (d_i, w_i), $i = 1, \ldots, n$;

1: **для всех** $d \in D$
2: задать длину n_d документа d;
3: **для всех** $i = 1, \ldots, n_d$
4: выбрать случайную тему t из распределения $p(t\,|\,d)$;
5: выбрать случайный термин w из распределения $p(w\,|\,t)$;
6: добавить в выборку пару (d, w), при этом тема t «забывается»;

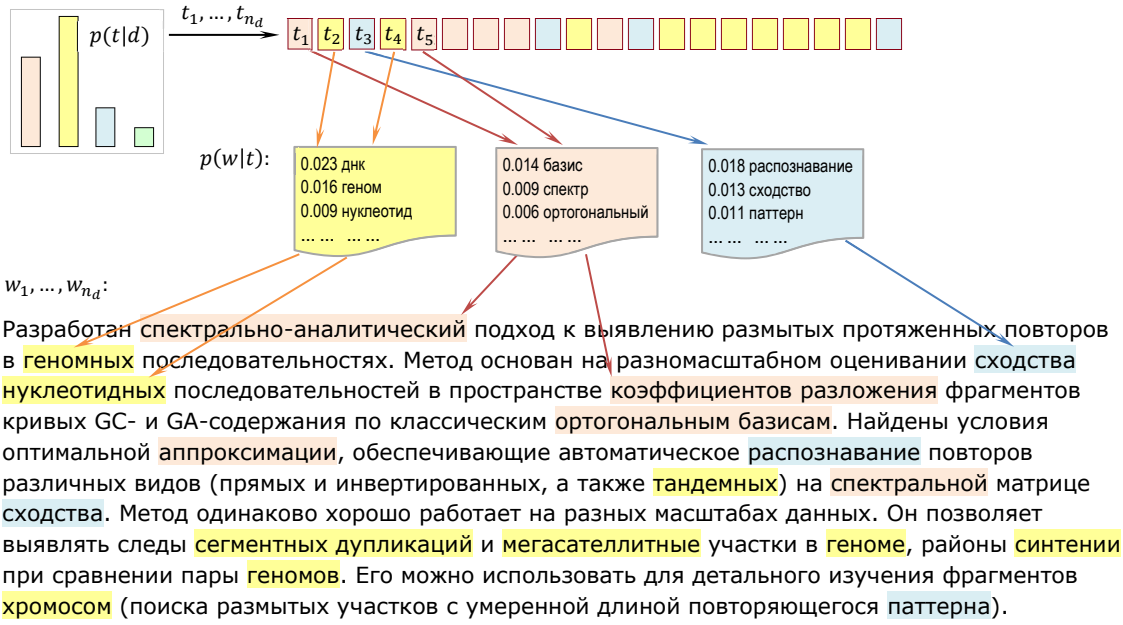


Рис. 1: Процесс порождения текстового документа вероятностной тематической моделью (2)

Построить *тематическую модель* коллекции документов D — значит найти множество тем T, распределения $p(w\,|\,t)$ для всех тем $t \in T$ и $p(t\,|\,d)$ для всех документов $d \in D$.

Распределения $p(t\,|\,d)$ — это сжатые тематические описания документов, которые предполагается использовать для дальнейшего решения задач информационного поиска, классификации, категоризации, аннотирования, суммаризации текстовых документов.

Гипотеза разреженности: *каждый документ d и каждый термин w связан с относительно небольшим числом тем t.*

Каждый документ относится лишь к небольшому числу тем (если же это энциклопедия, то ее лучше разбить на отдельные статьи). Каждая тема состоит из относительно небольшого числа терминов (в работах по филологии почти не встречаются термины из физики, химии, биологии, и многих других наук). Употребление термина в документе, как правило, связано только с одной темой. Таким образом, условные распределения $p(t\,|\,d)$, $p(w\,|\,t)$, $p(t\,|\,d, w)$ должны содержать значительную долю нулевых вероятностей.

Гипотеза шумовых и фоновых слов: *не все слова в тексте тематические.*

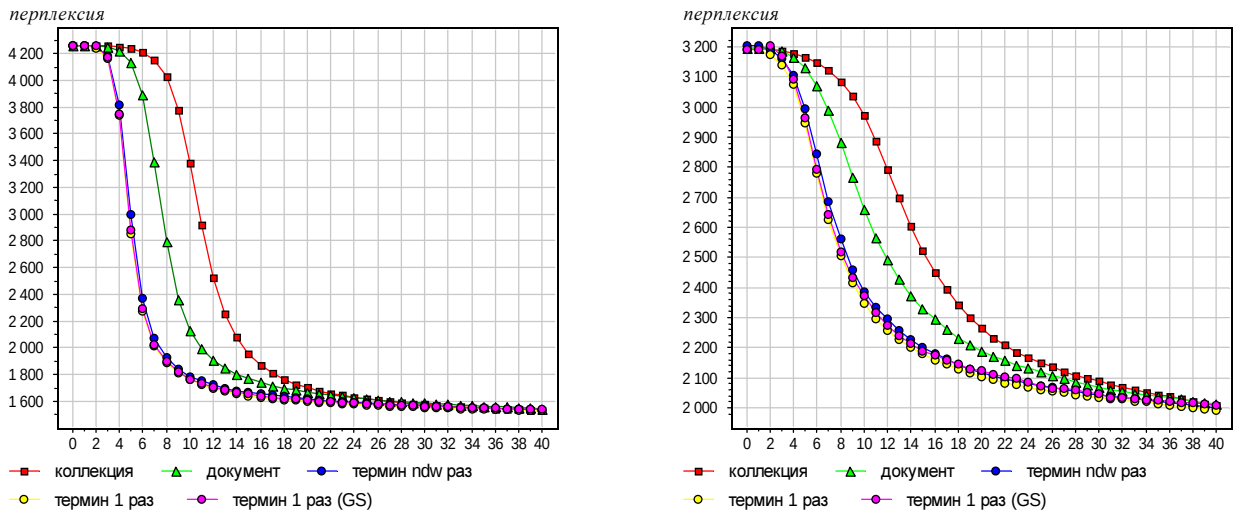


Рис. 2: Зависимость перплексии от числа итераций в стохастическом EM-алгоритме (SEM) при различной частоте обновления параметров φ_{wt}, θ_{td}: после каждого прохода коллекции, после каждого документа, после каждого термина (d, w) по всем n_{dw} его вхождениям, после каждого вхождения термина, GS — с предварительным уменьшением счетчиков как в алгоритме сэмплирования Гиббса. Параметры сглаживания: $\alpha_t = 0{,}5$, $\beta_w = 0{,}01$. Число тем $|T| = 100$

Различаются два вида нетематических слов: *шумовые* слова, специфичные для конкретного документа, и *фоновые* общеупотребительные слова, встречающиеся во многих документах. Фрагмент на рис. 1 содержит много слов, не относящихся ни к одной из тем.

Вообще, *тема* — это статистическое явление, связанное с совместным частым употреблением определенного набора терминов. Термины, употребляемые редко, статистически не значимы, не могут быть тематическими и должны относится к шумовым.

Гипотезы разреженности и шума–фона тесно связаны. Вместе они означают, что в каждом документе возможно отбросить относительно небольшую долю слов (скажем, треть или половину) так, чтобы появление оставшихся терминов описывалось распределениями $p(t \mid d)$, $p(w \mid t)$, разреженными гораздо сильнее (скажем, на 99%). В данной работе приводятся экспериментальные подтверждения этой гипотезы.

Частотные (выборочные) оценки вероятностей. Вероятности, связанные с наблюдаемыми переменными d и w, можно оценивать по выборке как частоты (здесь и далее выборочные оценки вероятностей p будем обозначать через $\hat{p}$):

$$\hat{p}(d,w) = \frac{n_{dw}}{n}, \qquad \hat{p}(d) = \frac{n_d}{n}, \qquad \hat{p}(w) = \frac{n_w}{n}, \qquad \hat{p}(w \mid d) = \frac{n_{dw}}{n_d}, \tag{3}$$

n_{dw} — число вхождений термина w в документ d;

$n_d = \sum\limits_{w \in W} n_{dw}$ — длина документа d в терминах;

$n_w = \sum\limits_{d \in D} n_{dw}$ — число вхождений термина w во все документы коллекции;

$n = \sum\limits_{d \in D} \sum\limits_{w \in d} n_{dw}$ — длина коллекции в терминах.

Вероятности, связанные со скрытой переменной t, также можно оценивать как частоты, если рассматривать коллекцию документов как выборку троек (d, w, t):

$$\hat{p}(t) = \frac{n_t}{n}, \qquad \hat{p}(w \mid t) = \frac{n_{wt}}{n_t}, \qquad \hat{p}(t \mid d) = \frac{n_{dt}}{n_d}, \qquad \hat{p}(t \mid d, w) = \frac{n_{dwt}}{n_{dw}}, \tag{4}$$

n_{dwt} — число троек, в которых термин w документа d связан с темой t;

$n_{dt} = \sum\limits_{w \in W} n_{dwt}$ — число троек, в которых термин документа d связан с темой t;

$n_{wt} = \sum\limits_{d \in D} n_{dwt}$ — число троек, в которых термин w связан с темой t;

$n_t = \sum\limits_{d \in D} \sum\limits_{w \in d} n_{dwt}$ — число троек, связанных с темой t.

В пределе $n \to \infty$ частотные оценки $\hat{p}(\cdot)$, определяемые формулами (3)–(4), стремятся к соответствующим вероятностям $p(\cdot)$.

Метод максимума правдоподобия. Для оценивания параметров тематической модели по коллекции документов D будем максимизировать правдоподобие (плотность распределения) выборки:

$$p(D) = C \prod_{d \in D} \prod_{w \in d} p(d, w)^{n_{dw}} = \prod_{d \in D} \prod_{w \in d} p(w \mid d)^{n_{dw}} \underbrace{C p(d)^{n_{dw}}}_{\text{const}} \to \max,$$

где C — нормировочный множитель, зависящий только от чисел n_{dw}. Отбросим множители C и $p(d)$, не влияющие на положение точки максимума, подставим выражение для $p(w \mid d)$ из (2) и введем для искомых величин обозначения $\theta_{td} = p(t \mid d)$, $\varphi_{wt} = p(w \mid t)$. Прологарифмировав правдоподобие, получим задачу максимизации

$$L(D; \Phi, \Theta) = \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \varphi_{wt} \theta_{td} \to \max_{\Phi, \Theta} \tag{5}$$

при ограничениях неотрицательности и нормировки:

$$\varphi_{wt} \geqslant 0, \quad \sum_{w \in W} \varphi_{wt} = 1; \qquad \theta_{td} \geqslant 0, \quad \sum_{t \in T} \theta_{td} = 1.$$

Перплексия (perplexity) является стандартным критерием качества тематических моделей [8]. Это мера несоответствия или «удивленности» модели $p(w \mid d)$ терминам w, наблюдаемым в документах коллекции, определяемая через логарифм правдоподобия (5):

$$\mathscr{P}(D; \Phi, \Theta) = \exp\Big(-\frac{1}{n} L(D; \Phi, \Theta)\Big) = \exp\Big(-\frac{1}{n} \sum_{d \in D} \sum_{w \in d} n_{dw} \ln p(w \mid d)\Big). \tag{6}$$

Чем меньше эта величина, тем лучше модель p предсказывает появление терминов w в документах d коллекции D.

Принято считать, что перплексия, вычисленная по той же коллекции D, по которой строилась модель $\Phi\Theta$, может быть подвержена эффекту переобучения и давать оптимистично заниженные оценки [5]. В наших экспериментах использовалась стандартная методика вычисления *контрольной перплексии* [4]. Коллекция документов изначально разбивалась на две части: обучающую D, по которой строилась модель, и контрольную D',

по которой вычислялась перплексия. После обучения модели векторы φ_t фиксировались, векторы θ_d контрольных документов $d \in D'$ оценивались по первой половине каждого документа, по вторым половинам вычислялась контрольная перплексия. Для разбиения на две половины последовательность терминов $\{w_1, \ldots, w_{n_d}\}$ каждого контрольного документа $d \in D'$ разбивалась после случайной перестановки на две части равной длины. Новые слова, ни разу не встретившиеся в обучающей коллекции D, но попавшие во вторую часть контрольного документа, игнорировались.

Численные эксперименты проводились на двух коллекциях, доступных на викистранице `www.MachineLearning.ru` «Коллекции документов для тематического моделирования». Для обеих коллекций производилась лемматизация и отбрасывались стоп-слова.

Коллекция RuDis содержит $|D| = 2000$ авторефератов диссертаций на русском языке; суммарная длина $n \approx 8.7 \cdot 10^6$, объем словаря $|W| \approx 3 \cdot 10^4$. Контрольная коллекция D' состоит из 200 авторефератов.

Коллекция NIPS содержит $|D| = 1566$ текстов статей научной конференции Neural Information Processing Systems на английском языке; суммарная длина $n \approx 2.3 \cdot 10^6$, объем словаря $|W| \approx 1.3 \cdot 10^4$. Контрольная коллекция D' состоит из 174 документов.

Тематические модели PLSA, LDA, SWB

Данный раздел носит обзорный характер. Вводятся классические тематические модели PLSA [12] и LDA [5]. Предлагается элементарная интерпретация M-шага EM-алгоритма для PLSA. Модель LDA рассматривается как легкое расширение PLSA. Рассматривается робастная модель PLSA-SWB (special words with background) с шумовыми и фоновыми словами. Она устраняет недостатки PLSA и позволяет отказаться от избыточных вероятностных допущений модели LDA [1]. В конце раздела предлагается новая *упрощенная робастная модель*, которая не требует ни дополнительных вычислений, ни памяти для хранения параметров шума и фона.

Вероятностный латентный семантический анализ (probabilistic latent semantic analysis, PLSA) был предложен Томасом Хофманном в [12] и основан на модели (2).

Для решения задачи максимизации правдоподобия (5) в PLSA применяется итерационный процесс, называемый *EM-алгоритмом*, в котором каждая итерация состоит из двух шагов — E (expectation) и M (maximization) [9].

На E-шаге по текущим значениям параметров φ_{wt}, θ_{td} с помощью формулы Байеса вычисляются условные распределения латентных тем $p(t \mid d, w)$ для каждого термина $w \in d$ в каждом документе d:

$$H_{dwt} = p(t \mid d, w) = \frac{p(w \mid t)p(t \mid d)}{p(w \mid d)} = \frac{\varphi_{wt}\theta_{td}}{\sum\limits_{s \in T} \varphi_{ws}\theta_{sd}}. \tag{7}$$

На M-шаге, наоборот, по условным вероятностям тем H_{dwt} вычисляется новое приближение параметров φ_{wt}, θ_{td}. Это легко сделать, если заметить, что величина

$$\hat{n}_{dwt} = n_{dw}p(t \mid d, w) = n_{dw}H_{dwt} \tag{8}$$

оценивает (не обязательно целое) число n_{dwt} вхождений термина w в документ d, связанных с темой t. Просуммировав $\hat{n}_{dwt}$ по документам d и по терминам w, получим оценки

$\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$, и через них, согласно (4), — частотные оценки условных вероятностей φ_{wt}, θ_{td}:

$$\varphi_{wt} = \frac{\hat{n}_{wt}}{\hat{n}_t}, \qquad \hat{n}_{wt} = \sum_{d \in D} n_{dw} H_{dwt}, \qquad \hat{n}_t = \sum_{w \in W} \hat{n}_{wt}; \tag{9}$$

$$\theta_{td} = \frac{\hat{n}_{dt}}{\hat{n}_d}, \qquad \hat{n}_{dt} = \sum_{w \in d} n_{dw} H_{dwt}, \qquad \hat{n}_d = \sum_{t \in T} \hat{n}_{dt}. \tag{10}$$

Оценки (9)–(10) являются решением задачи максимизации правдоподобия (5) при фиксированных H_{dwt}. Доказательство этого факта можно найти в [13, 1].

Число операций на каждом E- и M-шаге линейно по длине коллекции n и числу тем T.

Линейный проход по коллекции, то есть по всем терминам w во всех документах d, наиболее эффективен, если каждый документ d хранится в виде последовательности пар (w, n_{dw}). Такое представление возможно благодаря гипотезе «мешка слов».

Начальные приближения φ_t и θ_d можно задавать нормированными случайными векторами из равномерного распределения. Другая распространенная рекомендация — пройти по всей коллекции, выбрать для каждой пары (d, w) случайную тему t и вычислить частотные оценки (4) вероятностей φ_{wt} и θ_{td} для всех $d \in D$, $w \in W$, $t \in T$. В экспериментах эти два способа инициализации приводят к схожим результатам.

Недостатком PLSA является невозможность определить, какие из вероятностей θ_{td}, φ_{wt} равны нулю. Согласно формулам (7), (9), (10), если в начальном приближении $\theta_{td} = 0$ (тема t не представлена в документе d) или $\varphi_{wt} = 0$ (термин w не относится к теме t), то нулевое значение будет сохраняться на протяжении всех итераций. Аналогично, исходно ненулевые значения так и остаются ненулевыми. Таким образом, структура разреженности распределений не оптимизируется, а задается через начальное приближение.

Латентное размещение Дирихле. Другим недостатком PLSA принято считать высокую размерность пространства параметров, что может быть причиной переобучения [5]. В задачах машинного обучения для сокращения размерности обычно используется либо *отбор признаков*, приводящий к уменьшению числа параметров, либо *регуляризация* — наложение дополнительных ограничений на параметры. В частности, при *байесовской регуляризации* вводится априорное распределение вероятности в пространстве параметров.

Тематическая модель *латентного размещения Дирихле* (latent Dirichlet allocation, LDA) [5] основана на разложении (2) при дополнительном предположении, что векторы документов $\theta_d = (\theta_{td}) \in \mathbb{R}^{|T|}$ и векторы тем $\varphi_t = (\varphi_{wt}) \in \mathbb{R}^{|W|}$ порождаются распределениями Дирихле с параметрами $\alpha \in \mathbb{R}^{|T|}$ и $\beta \in \mathbb{R}^{|W|}$ соответственно:

$$\mathrm{Dir}(\theta_d; \alpha) = \frac{\Gamma(\alpha_0)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{td}^{\alpha_t - 1}, \qquad \alpha_t > 0, \qquad \alpha_0 = \sum_t \alpha_t, \qquad \theta_{td} > 0, \qquad \sum_t \theta_{td} = 1;$$

$$\mathrm{Dir}(\varphi_t; \beta) = \frac{\Gamma(\beta_0)}{\prod_w \Gamma(\beta_w)} \prod_w \varphi_{wt}^{\beta_w - 1}, \qquad \beta_w > 0, \qquad \beta_0 = \sum_w \beta_w, \qquad \varphi_{wt} > 0, \qquad \sum_w \varphi_{wt} = 1,$$

где $\Gamma(z)$ — гамма-функция. Векторы α и β называются *гиперпараметрами*.

Распределение Дирихле принято считать адекватным байесовским регуляризатором в задачах тематического моделирования.

Во-первых, это достаточно широкое параметрическое семейство распределений на единичном симплексе, то есть на множестве дискретных распределений. Если $\alpha_t = 1$ для

всех t, то распределение Дирихле переходит в равномерное. Математическое ожидание и дисперсия t-й координаты вектора θ_d равны, соответственно,

$$\mathsf{E}\theta_{td} = \int \theta_{td} \operatorname{Dir}(\theta_d; \alpha)\, d\theta_d = \frac{\alpha_t}{\alpha_0}, \qquad \mathsf{D}\theta_{td} = \frac{\alpha_t(\alpha_0 - \alpha_t)}{\alpha_0^2(\alpha_0 + 1)}. \tag{11}$$

Векторный параметр α определяет степень разреженности векторов θ_d, порождаемых распределением $\operatorname{Dir}(\theta; \alpha)$. Чем больше α_0, тем сильнее векторы θ_d концентрируется вокруг вектора математического ожидания $\mathsf{E}\theta_d$. Чем меньше α_t, тем сильнее значения θ_{td} концентрируются вокруг нуля. Чем меньше α_0, тем более разрежен вектор θ_d. Поэтому α_t называют *параметрами контраста*.

Во-вторых, модель LDA хорошо подходит для описания кластерных структур. Чем меньше значения гиперпараметров α и β, тем сильнее разрежено распределение Дирихле, и тем дальше отстоят друг от друга порождаемые им векторы. В частности, чем меньше α_0, тем сильнее различаются документы θ_d. Чем меньше β_0, тем сильнее различаются темы φ_t. Векторы $\varphi_t = p(w \mid t)$ в пространстве терминов $\mathbb{R}^{|W|}$ являются центрами тематических кластеров. Элементами кластеров являются эмпирические распределения документов $\hat{p}(w \mid d, t)$. Чем меньше значения гиперпараметров β_w, тем больше межкластерные расстояния по сравнению с внутрикластерными. Таким образом, гиперпараметры позволяют моделировать тематические кластерные структуры различной степени выраженности.

В-третьих, распределение Дирихле является сопряженным к мультиномиальному, что упрощает байесовский вывод апостериорных оценок вероятностей θ_{td} и φ_{wt}.

Недостатком априорного распределения Дирихле является отсутствие убедительных лингвистических обоснований. Предположение, что все векторы θ_d, $d \in D$ порождаются распределением Дирихле, причем одним и тем же, представляется весьма произвольным. То же можно сказать и о порождении векторов распределений φ_t для всех тем $t \in T$.

Второй недостаток заключается в том, что параметры θ_{td}, φ_{wt} и гиперпараметры α_t, β_w не могут обращаться в нуль, что противоречит гипотезе разреженности.

Чтобы получить оценки параметров θ_{td}, φ_{wt} в модели LDA, документ d рассматривается как выборка n_d пар тема–термин $X_d = \{(t_1, w_1), \ldots, (t_{n_d}, w_{n_d})\}$. В каждой паре тема t_i выбирается из дискретного распределения $p(t \mid d) = \theta_{td}$. Следовательно, вероятность встретить каждую из тем t ровно n_{td} раз подчиняется мультиномиальному распределению:

$$p(X_d|\theta_d) = \frac{n_d!}{\prod_t n_{td}!} \prod_t \theta_{td}^{n_{td}}.$$

Распределение Дирихле является *сопряженным* к мультиномиальному. Это означает, что при априорном распределении Дирихле $\theta_d \sim \operatorname{Dir}(\theta; \alpha)$ апостериорное распределение вектора θ_d принадлежит тому же семейству распределений, но с другим значением параметра: $\theta_d|X_d \sim \operatorname{Dir}(\theta; \alpha')$. Действительно, по формуле Байеса

$$p(\theta_d|X_d, \alpha) = \frac{p(X_d|\theta_d) \operatorname{Dir}(\theta_d; \alpha)}{p(X_d)} = C \prod_t \theta_{td}^{n_{td}} \theta_{td}^{\alpha_t - 1} = \operatorname{Dir}(\theta_d; \alpha'), \quad \alpha'_t = \alpha_t + n_{td},$$

где C — нормировочная константа, не зависящая от θ_d.

Оценим случайную величину θ_{td} ее математическим ожиданием (11) по апостериорному распределению:

$$p(t|d, X_d, \alpha) = \int p(t|d) p(\theta_d|X_d, \alpha)\, d\theta_d = \int \theta_{td} \operatorname{Dir}(\theta_d, \alpha')\, d\theta_d = \frac{n_{td} + \alpha_t}{n_d + \alpha_0}. \tag{12}$$

Заменив величину n_{td} ее оценкой $\hat{n}_{td}$, получим сглаженную байесовскую оценку параметра θ_{td} для ЕМ-алгоритма, альтернативную оценке максимума правдоподобия (10):

$$\theta_{td} = \frac{\hat{n}_{dt} + \alpha_t}{\hat{n}_d + \alpha_0}. \tag{13}$$

Аналогично выводится сглаженная байесовская оценка и для φ_{wt}, альтернативная (9):

$$\varphi_{wt} = \frac{\hat{n}_{wt} + \beta_w}{\hat{n}_t + \beta_0}. \tag{14}$$

Частотные оценки условных вероятностей (9)–(10) являются частным случаем сглаженных оценок (14)–(13), что позволяет использовать для обучения моделей LDA и PLSA один и тот же ЕМ-алгоритм. К этому же результату приводят методы сэмплирования Гиббса [20, 25] и вариационной байесовской аппроксимации [21].

Анализ известных алгоритмов обучения LDA показал, что все они являются модификациями ЕМ-алгоритма и отличаются, главным образом, формулой сглаживания частотных оценок вероятностей [4]. Оптимизация гиперпараметров [22, 23] еще сильнее нивелирует различия между алгоритмами. Согласно [11], максимизация апостериорной вероятности в модели LDA при $\alpha = 0$ и $\beta = 0$ приводит к формулам М-шага для модели PLSA.

Далее мы рассмотрим различные модификации ЕМ-алгоритма для обобщенной модели PLSA/LDA и покажем в экспериментах, что использование априорных распределений Дирихле не столь необходимо, как это принято считать. Эвристики разреживания и робастности, а также некоторые особенности реализации ЕМ-алгоритма могут гораздо сильнее влиять на вычислительную эффективность и качество тематической модели.

Робастная тематическая модель формализует предположение, что лишь некоторые слова в текстах относятся к каким-либо темам. Она представляет собой вероятностную смесь трех компонент — тематической, шумовой и фоновой [1]:

$$p(w \,|\, d) = \frac{Z_{dw} + \gamma\pi_{dw} + \varepsilon\pi_w}{1 + \gamma + \varepsilon}; \qquad Z_{dw} = \sum_{t \in T} \varphi_{wt}\theta_{td}. \tag{15}$$

Шумовая компонента $\pi_{dw} \equiv p_{\text{ш}}(w \,|\, d)$ — это слова, специфичные для конкретного документа d, либо редкие термины, относящиеся к темам, слабо представленным в данной коллекции. Отнесение шумовых слов к темам загрязняет распределения $\varphi_{wt} = p(w \,|\, t)$, увеличивает перплексию и искажает тематическую модель.

Фоновая компонента $\pi_w \equiv p_{\text{ф}}(w)$ — это общеупотребительные слова, в частности, стоп-слова, не отброшенные на стадии предварительной обработки. Фоновые слова имеют значимые вероятности во многих темах и только мешают различать темы.

Тематическая компонента Z_{dw} совпадает с моделью PLSA. Если она плохо объясняет избыточную частоту слова в документе, то слово относится к шуму или фону. Параметры γ и ε, ограничивающие долю таких слов, связаны с априорными вероятностями тематической, шумовой и фоновой компонент, равными $1.(1 + \gamma + \varepsilon)$, $\gamma.(1 + \gamma + \varepsilon)$, $\varepsilon.(1 + \gamma + \varepsilon)$ соответственно.

Похожая модель SWB на основе LDA предлагалась в [7]. Основное отличие нашей робастной модели от [7] в том, что она может сочетаться как с LDA, так и с PLSA, и не обязательно связана с сэмплированием Гиббса. Кроме того, мы не вводим априорных распределений Дирихле для параметров π_{dw}, π_w и (γ, ε), полагая, что фоновую и шумовую

компоненты гораздо логичнее разреживать, а не сглаживать. Параметры γ, ε логичнее фиксировать или определять по внешнему критерию качества, так как правдоподобие монотонно возрастает по γ и убывает по ε.

Задача максимизации правдоподобия (5) для модели (15) решена в [1]. По аналогии со стандартным ЕМ-алгоритмом, на Е-шаге для каждой пары (d, w) вычисляются по формуле Байеса условные вероятности тем $H_{dwt} = p(t \mid d, w)$,

$$H_{dwt} = \frac{\varphi_{wt}\theta_{td}}{Z_{dw} + \gamma\pi_{dw} + \varepsilon\pi_w}, \quad t \in T, \tag{16}$$

а также условные вероятности того, что слово w является шумом H_{dw} и фоном H'_{dw}:

$$H_{dw} = \frac{\gamma\pi_{dw}}{Z_{dw} + \gamma\pi_{dw} + \varepsilon\pi_w}; \qquad H'_{dw} = \frac{\varepsilon\pi_w}{Z_{dw} + \gamma\pi_{dw} + \varepsilon\pi_w}. \tag{17}$$

На М-шаге переменные θ_{td} и φ_{wt} вычисляются по прежним формулам (9) и (10) с единственным отличием, что теперь H_{dwt} вычисляются по новой формуле (16). Переменные π_{dw} и π_w вычисляются как частотные оценки условных вероятностей шума и фона:

$$\pi_{dw} = \frac{\nu_{dw}}{\nu_d}, \qquad \nu_{dw} = n_{dw}H_{dw}, \qquad \nu_d = \sum_{w \in d} \nu_{dw},$$
$$\pi_w = \frac{\nu'_w}{\nu'}, \qquad \nu'_w = \sum_{d \in D} n_{dw}H'_{dw}, \qquad \nu' = \sum_{w \in W} \nu'_w,$$

где ν_d и ν' — оценки числа шумовых слов в документе d и фоновых слов во всей коллекции. Эти формулы для π_{dw} и π_w называются *мультипликативным М-шагом*. Они порождают ту же проблему разреженности, что и переменные φ_{wt} и θ_{td}: если в начальном приближении значение π_{dw} или π_w не равно нулю, то оно так и останется ненулевым.

Формула *аддитивного М-шага*, полученная в [1] из условий Куна–Таккера задачи (5), приводит к автоматическому выбору структуры разреженности матрицы $(\pi_{dw})_{D \times W}$:

$$\pi_{dw} = \left(\frac{n_{dw}}{\nu_d} - \frac{Z_{dw} + \varepsilon\pi_w}{\gamma} \right)_+ . \tag{18}$$

Эта формула имеет прозрачную интерпретацию: если термин w в документе d встречается существенно чаще, чем предсказывают тематическая и фоновая компоненты модели, то его появление объясняется особенностями данного документа, и тогда $\pi_{dw} > 0$.

Упрощенная робастная модель. Недостатком предыдущей модели является необходимость подбирать параметры γ, ε и хранить параметры π_{dw}, число которых сопоставимо с размером коллекции. В качестве альтернативы рассмотрим упрощенную робастную модель, в которой фоновая компонента отсутствует, а шумовая компонента π_{dw} включается только когда $Z_{dw} = 0$, то есть когда термин w в документе d не является тематическим:

$$p(w \mid d) = \nu_d Z_{dw} + \big[Z_{dw} = 0\big]\pi_{dw}, \tag{19}$$

где параметр ν_d определяется из условия нормировки $\sum_{w \in W} p(w \mid d) = 1$.

Максимизация правдоподобия (5) снова приводит к частотным оценкам условных вероятностей (9)–(10), но теперь H_{dwt} и $\hat{n}_{dwt}$ оцениваются только по тематическим терминам:

$$\hat{n}_{dwt} = \big[Z_{dw} > 0\big] n_{dw} H_{dwt}.$$

Оптимальное значение π_{dw} достаточно определять только для тех (d,w), при которых $Z_{dw}=0$. Оно также выражается аналитически, $\pi_{dw}=n_{dw}/n_d$, что совпадает с частотной оценкой условной вероятности $p(w\,|\,d)$.

Нормировочный множитель ν_d равен доле тематических терминов в документе:

$$\nu_d=\sum_{w\in W}\big[Z_{dw}>0\big]\pi_{dw}=\frac{1}{n_d}\sum_{w\in d}\big[Z_{dw}>0\big]n_{dw}.$$

Заметим, что параметры π_{dw} и ν_d не нужны для вычисления тематической компоненты модели — матриц Φ и Θ, но могут понадобиться при вычислении перплексии (6), которая непосредственно зависит от $p(w\,|\,d)$.

Упрощенная робастная модель не требует дополнительных затрат памяти или времени. Поэтому в наших экспериментах она используется всегда, когда возможно обнуление тематической компоненты Z_{dw}, если явно не указано, что используется робастная модель (15).

Обобщенный EM-алгоритм и его модификации

В данном разделе рассматриваются рациональный, обобщенный и стохастический варианты EM-алгоритма [1]. Алгоритм сэмплирования Гиббса рассматривается как частный случай стохастического EM-алгоритма, применимый не только к модели LDA, но также и к PLSA. Тем самым мы показываем, что PLSA и LDA отличаются только эвристикой сглаживания, а многочисленные варианты EM-алгоритма одинаково применимы к обеим моделям. Предлагается несколько новых эвристик: принудительное разреживание условных распределений тем $p(t\,|\,d,w)$, постепенное увеличение априорных вероятностей шума и фона в робастной модели, простая и сложная стратегии разреживания матриц Φ и Θ. По сравнению с [1] существенно расширен состав экспериментов по подбору оптимального сочетания эвристик в EM-алгоритме.

Рациональный EM-алгоритм. Начнем с простой реорганизации шагов EM-алгоритма, чтобы избежать хранения трехмерной матрицы H_{dwt}. Заметим, что переменные $\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$ вычисляются на M-шаге в цикле по всем документам $d\in D$ и всем терминам $w\in d$. Внутри этого цикла переменные H_{dwt} будем вычислять непосредственно в тот момент, когда они понадобятся. Переменную $\hat{n}_d$ можно не вычислять, поскольку $\hat{n}_d=n_d$. Тогда E-шаг встраивается внутрь M-шага без дополнительных вычислительных затрат. Этот вариант EM-алгоритма будем называть *рациональным*, он показан в Алгоритме 2.

Условия остановки в данном алгоритме и всех последующих не сформулированы. В наших экспериментах 40 итераций всегда было достаточно для сходимости перплексии на обучающей и контрольной выборках.

Обобщенный EM-алгоритм. Известно, что на каждом M-шаге нет необходимости слишком точно решать задачу максимизации правдоподобия. Достаточно сместиться в направлении максимума и затем выполнить E-шаг. Этот вариант EM-алгоритма называется *обобщенным EM-алгоритмом* (generalized EM-algorithm, GEM) [9]. Другое обобщение состоит в том, что E-шаг выполняется только для части скрытых переменных, после этого M-шаг выполняется только для тех переменных, значения которых зависят от изменившихся скрытых переменных [17]. Для обобщенных EM-алгоритмов справедливы те же обоснования сходимости, что и для стандартного EM-алгоритма.

Алгоритм 2 PLSA-EM: рациональный EM-алгоритм для модели PLSA.

Вход: коллекция документов D, число тем $|T|$, начальные приближения Θ и Φ;
Выход: распределения Θ и Φ;

пока не выполнится критерий остановки, повторять итерации:
  обнулить $\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$ для всех $d \in D,\ w \in W,\ t \in T$;
  для всех $d \in D,\ w \in d$
    $Z := \sum_{t \in T} \varphi_{wt}\theta_{td}$;
    для всех $t \in T$ таких, что $\varphi_{wt}\theta_{td} > 0$
      увеличить $\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$ на $\delta = n_{dw}\varphi_{wt}\theta_{td}/Z$;
  $\varphi_{wt} := \hat{n}_{wt}/\hat{n}_t$ для всех $w \in W,\ t \in T$;
  $\theta_{td} := \hat{n}_{dt}/n_d$ для всех $d \in D,\ t \in T$;

Алгоритм 3 PLSA-GEM: обобщенный EM-алгоритм для модели PLSA.

Вход: коллекция документов D, число тем $|T|$, начальные приближения Θ и Φ;
Выход: распределения Θ и Φ;

обнулить $\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$, $\hat{n}_d$, n_{dwt} для всех $d \in D,\ w \in W,\ t \in T$;
пока не выполнится критерий остановки, повторять итерации:
  для всех $d \in D,\ w \in d$
    $Z := \sum_{t \in T} \varphi_{wt}\theta_{td}$;
    для всех $t \in T$ таких, что $n_{dwt} > 0$ или $\varphi_{wt}\theta_{td} > 0$
      $\delta := n_{dw}\varphi_{wt}\theta_{td}/Z$;
      увеличить $\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$, $\hat{n}_d$ на $(\delta - n_{dwt})$;
      $n_{dwt} := \delta$;
    если пора обновить параметры Φ, Θ **то**
      $\varphi_{wt} := \hat{n}_{wt}/\hat{n}_t$ для всех $w \in W,\ t \in T$ таких, что $\hat{n}_{wt}$ изменился;
      $\theta_{td} := \hat{n}_{dt}/\hat{n}_d$ для всех $d \in D,\ t \in T$ таких, что $\hat{n}_{dt}$ изменился;

Обобщенный EM-алгоритм для PLSA/LDA отличается от стандартного более частым обновлением параметров θ_{td} и φ_{wt} по текущим значениям счетчиков $\hat{n}_{wt}$ и $\hat{n}_{dt}$. Возможные варианты обновлений — после каждого документа или заданного числа документов, после каждого термина (d, w) или заданного числа терминов, после каждого вхождения термина.

В Алгоритме 2 обновления происходят после каждого прохода коллекции.

В Алгоритме 3 моменты обновления выбираются на шаге 9. В экспериментах на достаточно больших коллекциях частые обновления ускоряют сходимость, рис. 2.

При обновлении после каждого термина или каждого вхождения можно не хранить значения φ_{wt}, θ_{td}, а вычислять их каждый раз как частное двух счетчиков.

На первой итерации (т. е. при первом проходе коллекции) частые обновления не делаются, чтобы в счетчиках накопилась информация по всей коллекции. В противном случае оценки параметров θ_{td} и φ_{wt} по начальному фрагменту выборки могут оказаться хуже начального приближения. Начиная со второй итерации, для каждого термина (d, w) из счетчиков $\hat{n}_{wt}$ и $\hat{n}_{dt}$ вычитается n_{dwt} — то самое значение δ, которое было к ним прибавлено при обработке термина (d, w) на предыдущей итерации. Таким образом, счетчики $\hat{n}_{wt}$ и $\hat{n}_{dt}$ всегда содержат результат последнего однократного прохода всей коллекции.

Недостатком Алгоритма 3 является необходимость хранить массив значений n_{dwt}, $t \in T$ для каждого термина (d, w). Расход памяти объема $O(n|T|)$ может оказаться неприемлемым даже при небольшом числе тем. С другой стороны, согласно гипотезе разреженности, этот массив должен состоять преимущественно из нулей. Далее рассматриваются несколько альтернативных способов разреживания распределений $p(t \mid d, w)$.

Принудительное разреживание условных распределений тем. Стратегия *максимального разреживания* распределений $p(t \mid d, w)$ на первый взгляд представляется наиболее естественной: для каждого термина (d, w) игнорируются темы с наименьшими вероятностями, остаются только s тем с наибольшими значениями n_{dwt}.

Эксперименты показывают, что эта стратегия приводит к накоплению систематической ошибки и расходимости (рис. 3). На первых же итерациях возникает сильная (свыше 90%) разреженность распределений $\varphi_{wt} = p(w \mid t)$, которые к этому моменту еще не сошлись. Значения φ_{wt}, оказавшиеся равными нулю, далее так и остаются нулевыми. Эвристики сглаживания или включения разреживания с 10-й итерации не решают проблему.

Более удачной оказалась стратегия *постепенного разреживания*, когда в каждом распределении $p(t \mid d, w)$ обнуляется заданная доля r наименьших ненулевых значений и производится перенормировка. Эксперименты показали, что при $r \leqslant 0{,}2$ и включении разреживаний начиная с 10-й итерации расходимость не возникает и финальная перплексия мало отличается от случая $r = 0$. При этом постепенно увеличивается разреженность распределений θ_{td} (до 0,5 и выше) и распределений φ_{wt} (немного ниже 0,5).

Робастные алгоритмы более устойчивы к постепенному разреживанию распределений $p(t \mid d, w)$, для них параметр разреживания можно увеличивать до $r = 0{,}5$, при этом разреженность φ_{wt} достигает почти 0,9, разреженность θ_{td} достигает 0,7.

Стохастический ЕМ-алгоритм (stochastic EM-algorithm, SEM) [6] приводит к другой адекватной стратегии разреживания распределений $p(t \mid d, w)$. Для каждой пары (d, w) распределение $p(t \mid d, w)$ используется только для сэмплирования s случайных тем t_{dwi}, $i = 1, \ldots, s$, после чего оно «забывается». В формулах М-шага вместо распределения $p(t \mid d, w)$ используется его несмещенная эмпирическая оценка:

$$\hat{p}(t \mid d, w) = \frac{1}{s} \sum_{i=1}^{s} \big[t_{dwi} = t\big]. \tag{20}$$

Тем самым обеспечивается несмещенность оценок φ_{wt}, θ_{td} и сходимость ЕМ-алгоритма. Объем s сэмплируемых выборок является параметром метода.

Модификация Алгоритма 3, трансформирующая его в стохастический обобщенный ЕМ-алгоритм (PLSA-SGEM), состоит из трех изменений:

1. перед шагом 5 сэмплируется s тем t_{dwi}, $i = 1, \ldots, s$ из $p(t \mid d, w)$;
2. на шаге 5 цикл по всем $t \in T$ заменяется циклом по $t = t_{dwi}$, $i = 1, \ldots, s$;
3. на шаге 6 вычисляется $\delta := n_{dw}/s$.

Эксперименты показывают, что достаточно сэмплировать совсем небольшое число тем, около 5 тем обычно достаточно (табл. 1 и 2). Эта эвристика, названная *экономным сэмплированием* [1], сокращает затраты времени и памяти в тех случаях, когда средняя по коллекции величина n_{dw} превышает s.

В эксперименте проверялась также гипотеза, что число тем, связанных с парой (d, w), не должно превышать числа употреблений данного слова n_{dw}. Для этого производилось

сэмплирование $\min\{s, n_{dw}\}$ тем, однако результаты для этой эвристики немного хуже, чем при сэмплировании ровно s тем.

Робастная модель менее чувствительна к выбору параметра s (табл. 2).

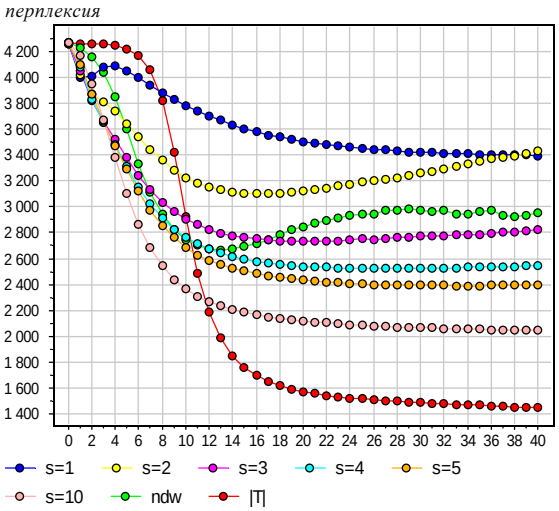
Качественные выводы, которые можно сделать по обучающей и по контрольной выборке, совпадают (табл. 1 и 2). В дальнейших экспериментах это тоже всегда так, но данные по обучающей выборке не показываются в таблицах и графиках.

Сэмплирование Гиббса. При $s = n_{dw}$ стохастический EM-алгоритм со сглаживанием (LDA-SEM) становится похож на *сэмплирование Гиббса* (Gibbs Sampling, GS) — один из основных методов обучения вероятностных тематических моделей [20, 25], см. Алгоритм 4. Алгоритмы LDA-SEM и LDA-GS отличаются несколькими деталями, которые, как показывают эксперименты, почти не влияют на качество модели.

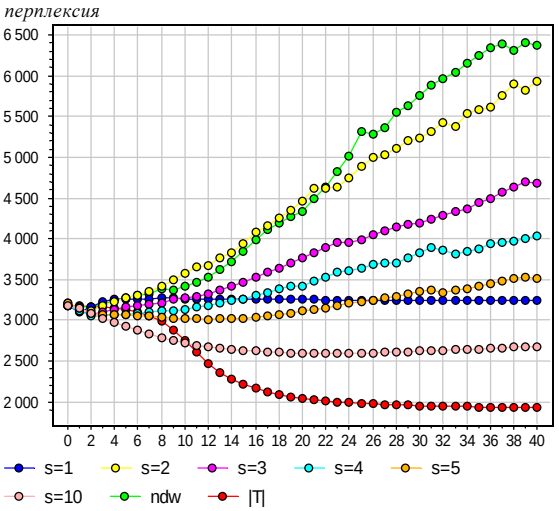
1. В LDA-GS число сэмплирований тем $s = n_{dw}$ для каждой пары (d, w). Согласно описанным выше экспериментам, можно также сэмплировать фиксированное число тем.

2. В LDA-GS параметры φ_{wt} и θ_{td} обновляются предельно часто — после обработки каждого вхождения термина w в документ d. В LDA-SEM обновления могут производиться с любой частотой. Эксперименты показывают, что частота обновления влияет только на скорость сходимости, но почти не влияет на значение контрольной перплексии в конце итераций, рис. 2. По результатам эксперимента можно рекомендовать обновления после каждого термина или после каждого вхождения термина, как в LDA-GS.

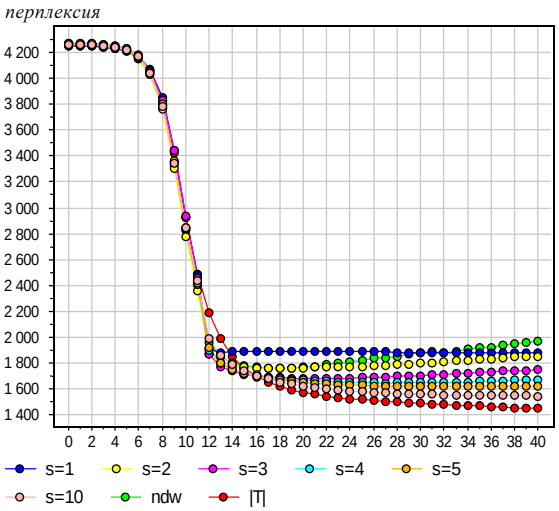
3. В LDA-GS перед сэмплированием счетчики уменьшаются на единицу (шаг 5). Тем самым при оценивании распределений не учитывается i-е вхождение термина w в документ d, для которого сэмплируется тема t_{dwi}. Из теории следует, что эта особенность исключительно важна [25]. Однако в экспериментах с коллекциями достаточно больших размеров оказывается, что она не влияет на качество модели — кривые «термин 1 раз» и «термин 1 раз (GS)» на рис. 2 практически совпадают. Можно одновременно уменьшать счетчики для старой темы и увеличивать для новой, как в Алгоритме 3.



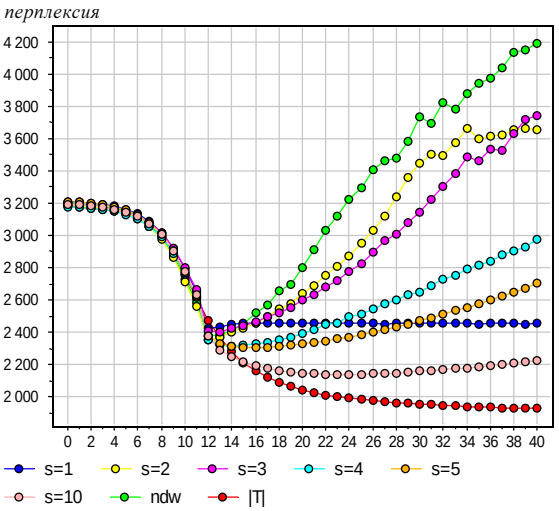
RuDis, разреживание с 1-й итерации



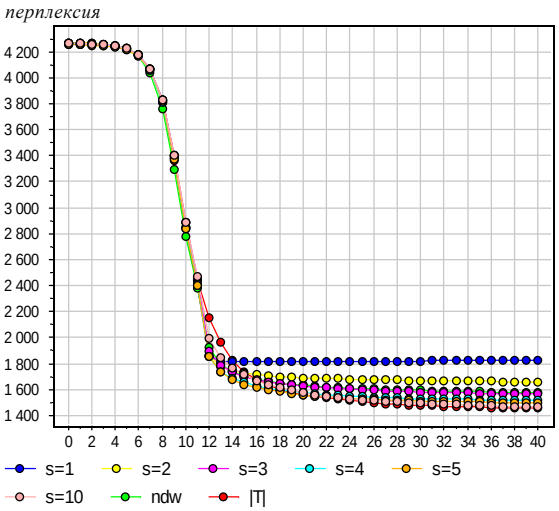
NIPS, разреживание с 1-й итерации



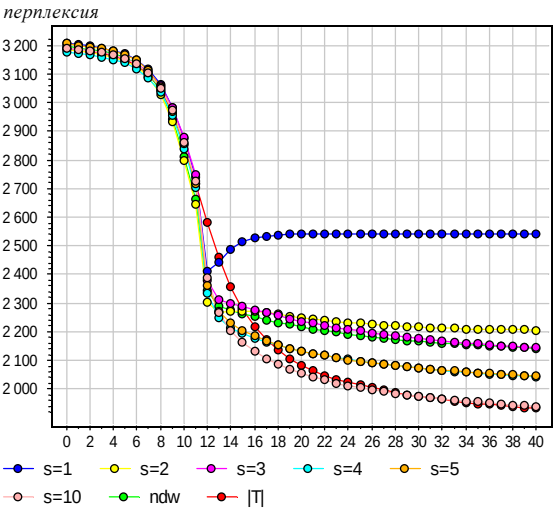
RuDis, разреживание с 10-й итерации



NIPS, разреживание с 10-й итерации



RuDis, с 10-й итерации, сглаживание LDA



NIPS, с 10-й итерации, сглаживание LDA

Рис. 3: Зависимость перплексии от числа итераций в рациональном EM-алгоритме при максимальном разреживании $p(t \mid d, w)$. Параметр разреживания: $s = 1, 2, 3, 4, 5, 10, n_{dw}$, при $s = |T|$ разреживания нет. Параметры сглаживания: $\alpha_t = 0.5$, $\beta_w = 0.01$. Число тем $|T| = 100$.

Таблица 1: Стохастический EM-алгоритм для модели LDA. Зависимость перплексии на обучении и контроле от объема s сэмплированной выборки (40 итераций, $\alpha_t = 0{,}5$, $\beta_w = 0{,}01$)

RuDis: s фиксирован

s	обуч.	конт.
n_{dw}	1367	1535
1	1707	1874
2	1547	1705
3	1463	1628
4	1407	1552
5	1383	1559
10	1295	1480

RuDis: $\min\{s, n_{dw}\}$

s	обуч.	конт.
n_{dw}	1367	1535
1	1724	1894
2	1575	1730
3	1507	1673
4	1479	1647
5	1459	1603
10	1418	1571

NIPS: s фиксирован

s	обуч.	конт.
n_{dw}	1506	2002
1	1796	2326
2	1616	2120
3	1513	2006
4	1473	1981
5	1430	1946
10	1326	1874

NIPS: $\min\{s, n_{dw}\}$

s	обуч.	конт.
n_{dw}	1506	2002
1	1791	2313
2	1647	2157
3	1591	2101
4	1562	2052
5	1547	2052
10	1517	2019

Таблица 2: Стохастический EM-алгоритм для робастной модели LDA. Зависимость перплексии на обучении и контроле от объема s сэмплированной выборки (40 итераций, $\alpha_t = 0{,}5$, $\beta_w = 0{,}01$)

RuDis: s фиксирован

s	обуч.	конт.
n_{dw}	717	794
1	777	857
2	754	830
3	736	815
4	728	804
5	724	799
10	715	789

RuDis: $\min\{s, n_{dw}\}$

s	обуч.	конт.
n_{dw}	717	794
1	773	850
2	748	821
3	737	811
4	731	807
5	728	801
10	722	800

NIPS: s фиксирован

s	обуч.	конт.
n_{dw}	1110	1363
1	1270	1544
2	1171	1442
3	1140	1414
4	1103	1375
5	1087	1352
10	1053	1317

NIPS: $\min\{s, n_{dw}\}$

s	обуч.	конт.
n_{dw}	1110	1363
1	1263	1530
2	1185	1464
3	1167	1441
4	1150	1423
5	1133	1398
10	1121	1393

Алгоритм 4 LDA-GS: сэмплирование Гиббса для тематической модели LDA.

Вход: коллекция D, число тем $|T|$, векторы гиперпараметров α, β;
Выход: распределения Θ и Φ;

обнулить $\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$ для всех $d \in D$, $w \in W$, $t \in T$;
пока не выполнится критерий остановки, повторять итерации:
  для всех $d \in D$, $w \in d$, $i = 1, \ldots, n_{dw}$
    если не первая итерация **то**
      $t := t_{dwi}$; уменьшить $\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$ на 1;
    сэмплировать тему t_{dwi} из $p(t \mid d, w) \propto (\hat{n}_{dt} + \alpha_t)(\hat{n}_{wt} + \beta_w)/(\hat{n}_t + \beta_0)$;
    $t := t_{dwi}$; увеличить $\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$ на 1;
$\varphi_{wt} = (\hat{n}_{wt} + \beta_w)/(\hat{n}_t + \beta_0)$ для всех $t \in T$, $w \in W$;
$\theta_{td} := (\hat{n}_{dt} + \alpha_t)/(n_d + \alpha_0)$ для всех $d \in D$, $t \in T$;

Таким образом, главной особенностью алгоритма сэмплирования Гиббса, как и стохастического EM-алгоритма, является *эвристика сэмплирования* — замена распределения тем $p(t \mid d, w)$ его разреженным эмпирическим аналогом (20). Хотя в литературе алгоритм

Алгоритм 5 PLSA-REM: робастный рациональный EM-алгоритм для модели PLSA.

Вход: коллекция D, число тем $|T|$, начальные приближения Θ, Φ, параметры γ, ε;
Выход: распределения: матрицы (φ_{wt}), (θ_{td}), (π_{dw}) и вектор (π_w);

инициализировать $\pi_{dw} := n_{dw}/n_d$; $\pi_w := n_w/n$; для всех $d \in D$, $w \in W$;
пока не выполнится критерий остановки, повторять итерации:
  обнулить $\hat{n}_{wt}$, $\hat{n}_t$, ν'_w, ν, ν' для всех $w \in W$, $t \in T$;
  для всех $d \in D$
    обнулить $\hat{n}_{dt}$, $\hat{n}_d$, ν_d для всех $w \in W$, $t \in T$;
    для всех $w \in d$
      $Z := \gamma\pi_{dw} + \varepsilon\pi_w + \sum_{t\in T} \varphi_{wt}\theta_{td}$;
      увеличить $\hat{n}_{wt}$, $\hat{n}_{dt}$, $\hat{n}_t$, $\hat{n}_d$ на $n_{dw}\varphi_{wt}\theta_{td}/Z$ для всех $t \in T$;
      увеличить ν_d, ν на $\nu_{dw} := n_{dw}\gamma\pi_{dw}/Z$;
      увеличить ν'_w, ν' на $n_{dw}\varepsilon\pi_w/Z$;
    $\theta_{td} := \hat{n}_{dt}/\hat{n}_d$ для всех $t \in T$;
    $\pi_{dw} := \nu_{dw}/\nu_d$ для всех $w \in d$;
  $\varphi_{wt} := \hat{n}_{wt}/\hat{n}_t$ для всех $w \in W$, $t \in T$;
  $\pi_w := \nu'_w/\nu'$ для всех $w \in W$;

сэмплирования Гиббса принято связывать с моделью LDA, он также в равной степени применим и к модели PLSA.

Эвристика *постепенного разреживания* является альтернативой сэмплированию. Обе эвристики легко встраиваются в любой EM-подобный алгоритм и сочетаются с другими эвристиками. Недостатком постепенного разреживания является необходимость хранения плотных массивов n_{dwt} на начальных итерациях.

В итоге рекомендуется либо рациональный EM-алгоритм с обновлением распределений Φ после каждого прохода коллекции, либо стохастический EM-алгоритм с экономным сэмплированием и обновлением после каждого термина, либо сэмплирование Гиббса.

Робастный EM-алгоритм. В отличие от рационального EM-алгоритма, n_{dw} вхождений термина w в документ d распределяются не только между темами $t \in T$, но также между шумовой и фоновой компонентами пропорционально вероятностям

$$\tilde{H}_{dw} = \big(\tfrac{1}{Z}\varphi_{wt}\theta_{td},\ t \in T;\ \ \tfrac{1}{Z}\gamma\pi_{dw};\ \ \tfrac{1}{Z}\varepsilon\pi_w\big),$$

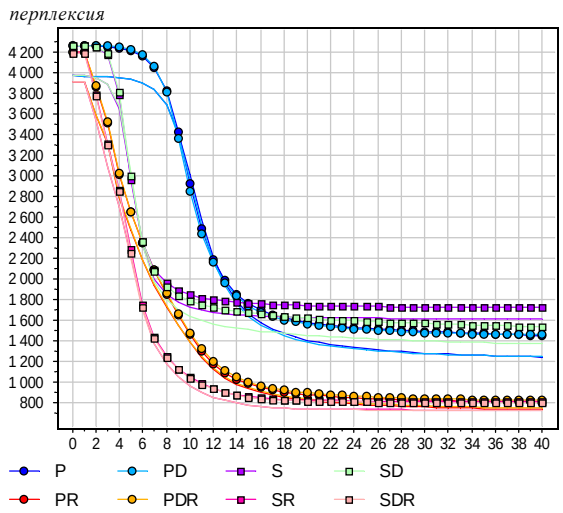
где Z — нормирующий множитель, см. Алгоритм 5.

В экспериментах использовался также стохастический робастный алгоритм с параметром сэмплирования $s = n_{dw}$. Все алгоритмы сравнивались в двух вариантах: с несмещенными оценками (9)–(10) и сглаженными оценками (13)–(14) параметров φ_{wt} и θ_{td}.

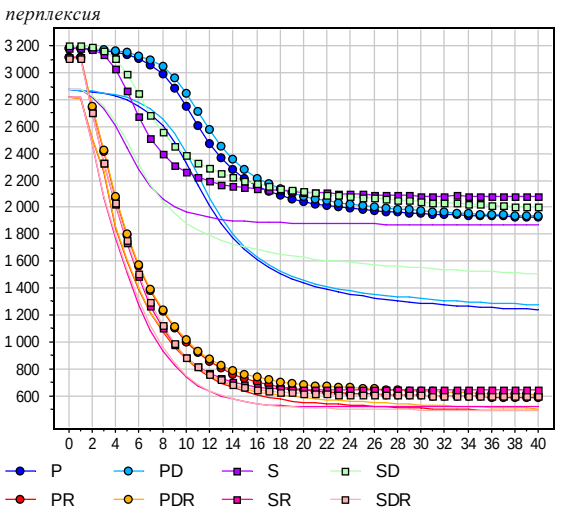
При вычислении перплексии на документах d контрольной выборки D' параметры φ_{wt} и π_w оценивались по обучающей выборке D, параметры θ_{td} и ν_d оценивались по первой половине документа d', параметры π_{dw} оценивались для каждой пары (d, w) согласно (18). Перплексия вычислялась по вторым половинам d'' контрольных документов.

Сравнение восьми алгоритмов, образуемых всеми комбинациями эвристик сглаживания, робастности и сэмплирования (рис. 4) позволяет сделать следующие выводы:

— для обеих задач робастные алгоритмы существенно превосходят неробастные и гораздо меньше переобучаются;



RuDis



NIPS

Рис. 4: Зависимость контрольной перплексии от числа итераций для всевозможных сочетаний эвристик: D — сглаживание Дирихле ($\alpha_t = 0{,}5$, $\beta_w = 0{,}01$); R — робастность ($\gamma = 0{,}3$, $\varepsilon = 0{,}01$); S — сэмплирование ($s = n_{dw}$), P — пропорциональное распределение (8); $|T| = 100$. Тонкие кривые без точек — перплексия обучающей выборки

Таблица 3: Контрольная перплексия $\mathscr{P}$ и оценки апостериорной вероятности шума $\hat{p}_{\text{ш}}$ и фона $\hat{p}_{\text{ф}}$ при различных значениях γ и ε (после 40 итераций, $|T| = 100$)

RuDis, $\varepsilon = 0{,}01$:

γ	$\mathscr{P}$	$\hat{p}_{\text{ш}}$
0	1540	0,000
0,001	1434	0,026
0,01	1277	0,090
0,05	1076	0,196
0,1	974	0,266
0,3	805	0,413
0,5	750	0,498

RuDis, $\gamma = 0{,}3$:

ε	$\mathscr{P}$	$\hat{p}_{\text{ф}}$
0	797	0,000
0,01	794	0,006
0,05	798	0,027
0,1	809	0,049
0,2	823	0,086
0,3	841	0,116
0,5	870	0,165

NIPS, $\varepsilon = 0{,}01$:

γ	$\mathscr{P}$	$\hat{p}_{\text{ш}}$
0	2001	0,000
0,001	1763	0,044
0,01	1381	0,152
0,05	991	0,296
0,1	818	0,377
0,3	604	0,527
0,5	525	0,598

NIPS, $\gamma = 0{,}3$:

ε	$\mathscr{P}$	$\hat{p}_{\text{ф}}$
0	598	0.000
0,01	596	0,005
0,05	605	0,023
0,1	613	0,043
0,2	630	0,079
0,3	640	0,109
0,5	668	0,157

— сэмплирование (20) немного хуже пропорционального распределения (8);
— сэмплирование без сглаживания может приводить к увеличению перплексии.

Величина переобучения (разность перплексии на обучающей и контрольной выборке) больше зависит от задачи, чем от алгоритма. Сравнение алгоритмов по перплексии на обучающей выборке приводит к тем же качественным выводам, что и их сравнение по перплексии на контрольной выборке. По всей видимости, для сравнения алгоритмов не нужна столь сложная методика разделения контрольных документов для вычисления перплексии; вполне достаточно вычислять перплексию только на обучающей выборке.

Возможны два варианта реализации M-шага — мультипликативный и аддитивный. В экспериментах на обеих задачах они не дают значимых различий перплексии.

Возможны два варианта определения роли каждого слова (d, w) при сэмплировании из распределения $\tilde{H}_{dw}$. В первом варианте роли распределяются между компонентами

тем, шума и фона «мягко», пропорционально их вероятностям, затем сэмплируются темы. Во втором варианте сэмплирование производится из всего распределения $\tilde{H}_{dw}$, в результате каждому слову «жестко» приписывается одна из трех взаимоисключающих ролей. В экспериментах эти два варианта также не дают значимых различий перплексии.

Зависимость перплексии от параметров γ и ε, как правило, монотонная, причем параметр γ гораздо сильнее влияет на перплексию, чем ε, см. табл. 3. С ростом γ перплексия уменьшается, так как компонента шума близка к униграммной модели документа, $\pi_{dw} \approx n_{dw}/n_d$, которая наиболее точно предсказывает вероятности слов $p(w \mid d)$, однако не является тематической. С ростом ε перплексия увеличивается, так как компонента фона близка к униграммной модели коллекции, $\pi_w \approx n_w/n$, которая хуже предсказывает вероятности слов $p(w \mid d)$, чем тематическая модель. Оценки апостериорных вероятностей шума $\hat{p}_{\text{ш}} = \nu/n$ и фона $\hat{p}_{\text{ф}} = \nu'/n$ также зависят от γ и ε монотонно. Следовательно, оптимальные значения параметров γ и ε должны определяться по внешним критериям качества той прикладной задачи, для решения которой строится тематическая модель.

На рис. 5 показаны зависимости перплексии и апостериорной вероятности шума от числа итераций при $\gamma = 0{,}0,\ 0{,}001,\ 0{,}3$ и при постепенном увеличении γ от $\gamma_0 = 0{,}001$ до $\gamma_1 = 0{,}3$ на первых $i_1 = 20$ итерациях:

$$\gamma = \gamma_0 + \frac{(\gamma_1 - \gamma_0)i^2}{i^2 + (i_1 - i)^2}.$$

Эвристика постепенного увеличения априорной вероятности шума позволяет достичь немного лучшего значения перплексии. Это можно объяснить тем, что шумовая компонента слишком агрессивно отбирает слова на первых же итерациях, когда тематическая компонента еще не успела сойтись.

Разреживающий ЕМ-алгоритм. Гипотеза разреженности предполагает, что коллекция порождается дискретными распределениями $\varphi_{wt} = p(w \mid t)$ и $\theta_{td} = p(t \mid d)$, в которых подавляющее большинство вероятностей равны нулю. Следствием этого является также и разреженность распределений $H_{dwt} = p(t \mid d, w)$. Обнуление значительной доли вероятностей φ_{wt} и θ_{td} позволяет ускорить ЕМ-алгоритм и хранить тематическую модель в более сжатом виде, открывая возможности для обработки очень больших коллекций.

Модель PLSA не оптимизирует структуру разреженности распределений и требует задавать ее через начальное приближение. Отдельные значения φ_{wt} и θ_{td} могут в ходе итераций сами собой приближаться к нулю, но, как правило, их доля недостаточна для получения выигрыша в производительности.

Модель LDA также не является разреженной — априорные распределения Дирихле запрещают вероятностям φ_{wt} и θ_{td} и гиперпараметрам β_w и α_t принимать нулевые значения. При стремлении гиперпараметров к нулю распределения Дирихле порождают векторы φ_t и θ_d, компоненты которых стремятся к нулю, но никогда не обращаются в нуль. Сглаженные оценки (14) и (13), используемые в LDA, менее разрежены, чем несмещенные частотные оценки (9) и (10), используемые в PLSA.

Известные подходы к разреживанию LDA требуют введения дополнительных параметров и усложнения ЕМ-алгоритма. В [10] предлагается хранить не сами значения φ_{wt} и θ_{td}, а только их разности с фоновыми распределениями. В [24] предполагается, что каждая тема описывается распределением Дирихле на подмножестве слов, заданном бинарными

переменными b_{wt} из распределения Бернулли. Сглаженность и разреженность регулируется независимо параметрами распределения Дирихле и распределения Бернулли. Недостатком данной модели является большое число дополнительных скрытых переменных, которые усложняют обучение. В [14] вводится распределение псевдо-Дирихле, которое строится путем расширения области определения распределения Дирихле и имеет ограниченную плотность, в то время как распределение Дирихле не ограничено в случае $\alpha < 1$, что и приводит к запрету нулевых значений φ_{wt} и θ_{td}.

В данной работе исследуются различные стратегии *принудительного разреживания*, когда в конце каждой итерации (полного прохода всей коллекции D) обнуляется некоторое количество наименьших значений φ_{wt} и θ_{td}. Эвристика разреживания не совместима со сглаживанием и применяется только к PLSA, т. е. при $\beta_w = 0,\ \alpha_t = 0$.

Предварительные эксперименты показали, что одновременное обнуление более 50% элементов является слишком сильным стрессом для модели и может вызывать расходимость ЕМ-алгоритма. Поэтому предлагается разреживать матрицы Φ и Θ постепенно, придерживаясь одной из следующих стратегий.

Простая стратегия: в каждом из распределений φ_t, θ_d обнуляется заданная доля r наименьших *ненулевых* значений. После обнуления производится перенормировка распределений. Число обнуляемых значений сокращается от итерации к итерации, поскольку доля берется от числа ненулевых значений. Обнуления прекращаются, когда в распределении остается $\lfloor r^{-1} \rfloor$ ненулевых значений. Недостатком этой стратегии является стремление к выравниванию доли ненулевых значений во всех распределениях, что представляется довольно странным ограничением.

Сложная стратегия устраняет этот недостаток. В каждом из распределений φ_t, θ_d обнуляется максимальное число наименьших значений, так, чтобы оно не превышало $r|W|$ и $r|T|$ соответственно, и сумма обнуляемых значений не превышала заданного порога R_φ или R_θ для распределений φ_t или θ_d соответственно.

Разреживания включаются, начиная с итерации i_0, чтобы в распределениях правильно выделились малые вероятности, и делаются не на каждой итерации, чтобы модель успевала восстановить адекватность. В экспериментах разреживания включались на итерациях с номерами $i = i_0 + k\delta,\ k = 1, 2, \ldots$, где i_0 и δ — параметры стратегии разреживания.

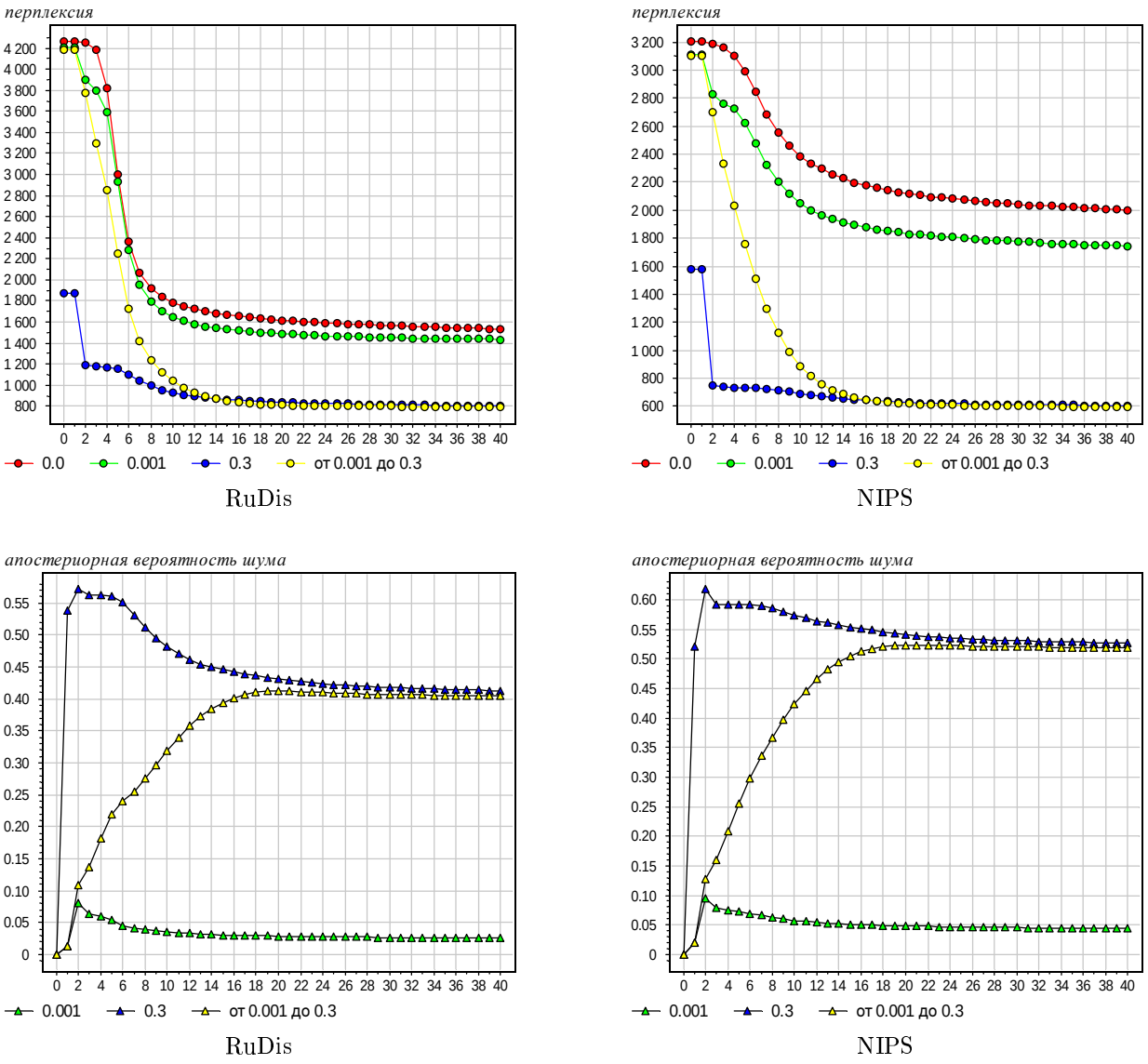


Рис. 5: Зависимость перплексии и апостериорной вероятности шума от числа итераций при $\gamma = 0.0,\ 0.001,\ 0.3$ и при постепенном увеличении γ от 0.001 до 0.3 на первых 20 итерациях. Остальные параметры: $\alpha_t = 0.5,\ \beta_w = 0.01,\ \varepsilon = 0.01,\ |T| = 100$.

Разреживание может приводить к обнулению распределения $p(t \mid d, w)$, тогда термин w интерпретируется как нетематический для документа d. Поэтому разреживание применяется совместно с робастной моделью (15), либо с упрощенной робастной моделью (19).

Результаты экспериментов приведены на рис. 6–8.

При совмещении упрощенной робастной модели, стохастического EM-алгоритма и разреживания достигается наименьшая перплексия и одновременно наибольшая разреженность матрицы Φ — до 99,4% для RuDis и 99,6% для NIPS (см. рис. 6).

В робастных алгоритмах с шумом и фоном разреживание почти не влияет на перплексию и позволяет достигать сопоставимой разреженности (см. рис. 7).

Под «агрессивным» разреживанием понимается уменьшение δ до 1 или уменьшение i_0 до 1 или применение сложной стратегии, когда доля обнуляемых значений не уменьшается с итерациями. При агрессивном разреживании или при использовании стохастического EM-алгоритма возможно разреживание распределений φ_t до 99%. При числе тем $T = 100$ это означает, что каждый термин в среднем относится только к одной теме.

При недостаточном априорном уровне шума $\gamma = 0{,}01$ агрессивное разреживание может приводить к расходимости EM-алгоритма (рис. 8). Тонкие кривые без точек, проходящие чуть ниже кривых контрольной перплексии, соответствуют перплексии на обучающей выборке. Они показывают, что расходимость возникает синхронно на контроле и обучении, причем на обучении расходимость даже более заметна и может быть легко обнаружена во время итераций EM-алгоритма.

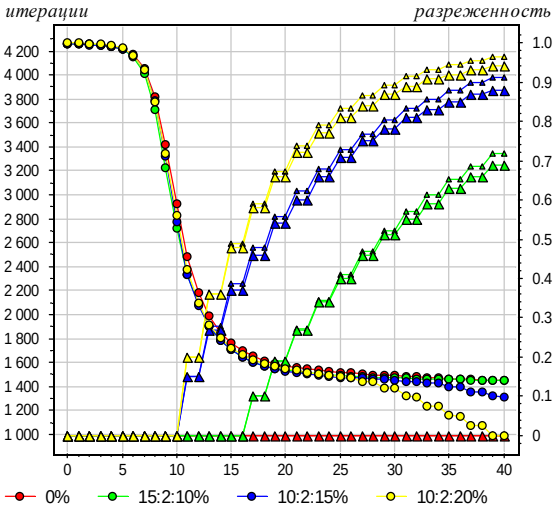
В экспериментах с упрощенной робастной моделью расходимость не наблюдалась.

О дилемме «сглаживание–разреживание»

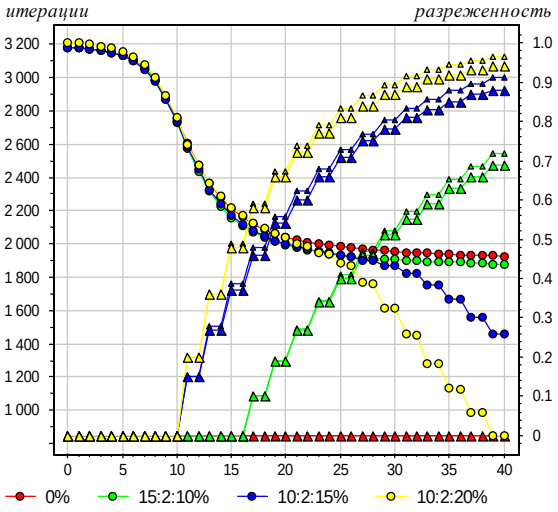
Среди рассмотренных эвристик только сглаживание и разреживание являются взаимно исключающими. В задачах машинного обучения им соответствуют два альтернативных подхода к понижению размерности — отбор признаков и регуляризация. Разреживание является частным случаем отбора признаков, который основан на предположении, что не все признаки несут полезную информацию. Сглаживание является следствием регуляризации, которая основана на предположении, что все признаки полезны и отбрасывать их нельзя, но необходимо ограничить их степени свободы.

В современных исследованиях по тематическому моделированию преобладают методы регуляризации. В данной работе мы показываем, что альтернативный подход к понижению размерности заслуживает не меньшего внимания. При этом разреживание особенно эффективно в сочетании с робастными тематическими моделями.

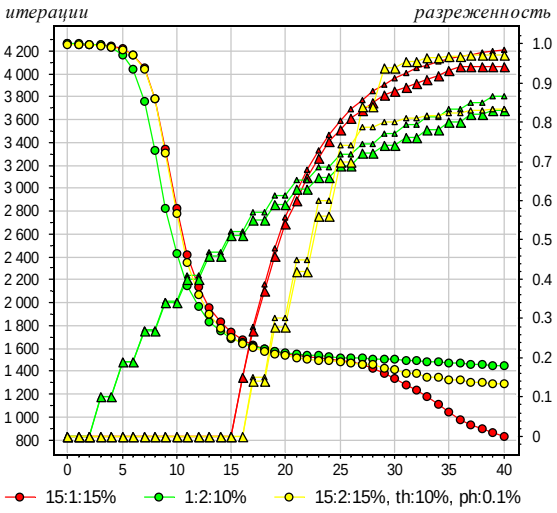
Согласно экспериментам, проведенным в [5], LDA обеспечивает существенно меньшие значения контрольной перплексии, чем PLSA. По аналогии с задачами классификации и регрессии отсюда был сделан стандартный вывод, что модель PLSA имеет слишком много параметров θ_{td}, φ_{wt}, и при отсутствии ограничений на них возникает переобучение. Байесовская регуляризация должна сокращать эффективную размерность и уменьшать переобучение. Однако более тщательное сравнение PLSA и LDA показывает, что регуляризация Дирихле в тематических моделях играет совсем другую роль.



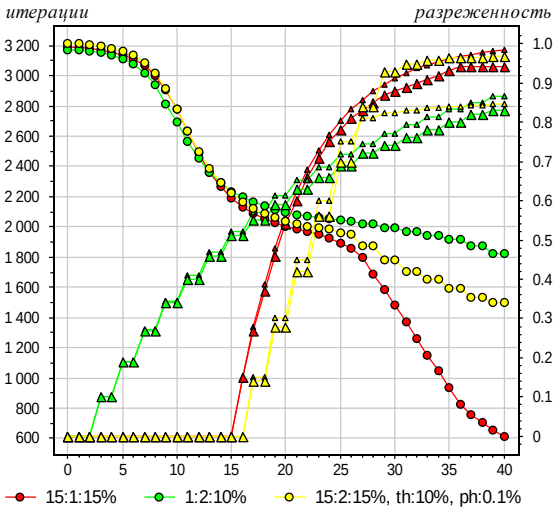
RuDis, разреживание через 2 итерации



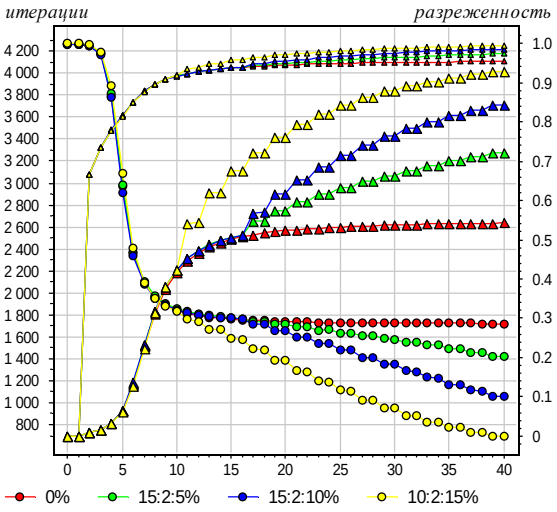
NIPS, разреживание через 2 итерации



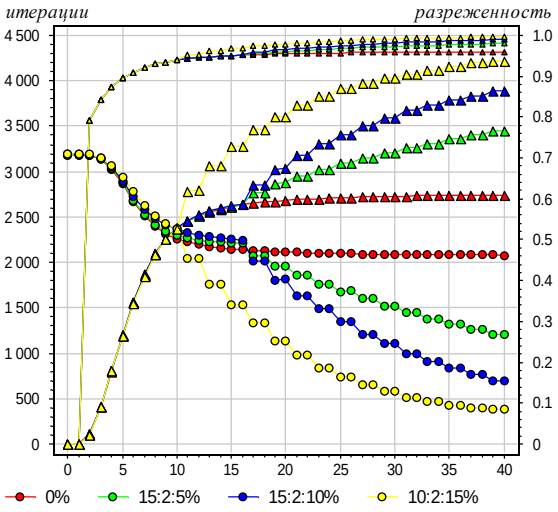
RuDis, агрессивное разреживание



NIPS, агрессивное разреживание

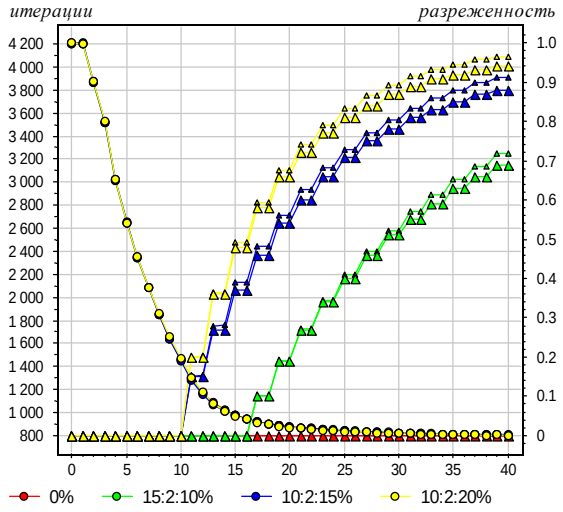


RuDis, SEM, через 2 итерации

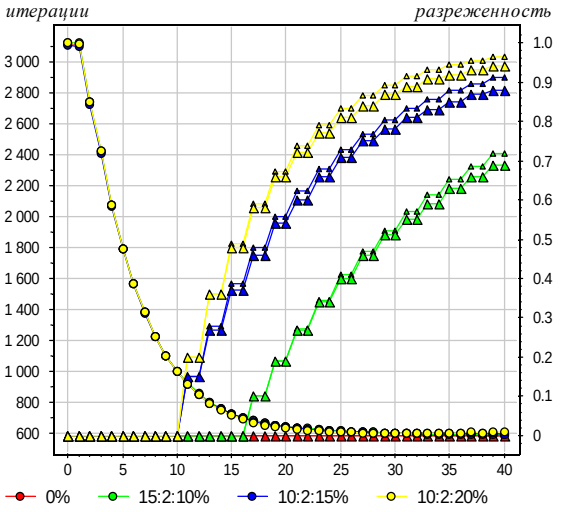


NIPS, SEM, через 2 итерации

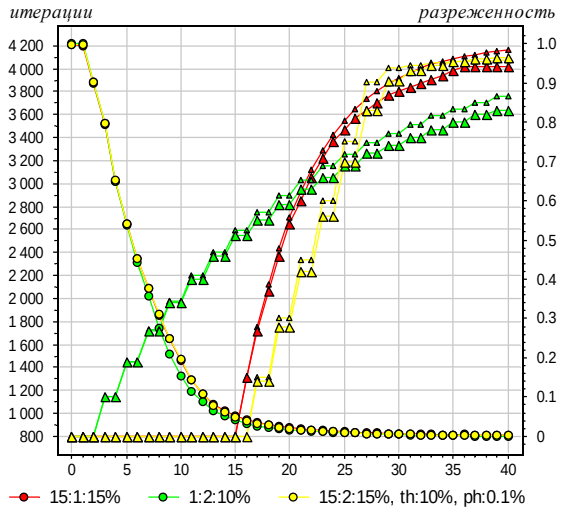
Рис. 6: Зависимость перплексии (○) и разреженности матриц Φ (▵) и Θ (△) от числа итераций для рационального и стохастического EM-алгоритма при различных параметрах разреживания, обозначаемых $i_0{:}\delta{:}r$, $\mathsf{th}{:}R_\theta$, $\mathsf{ph}{:}R_\varphi$. Число тем $|T| = 100$



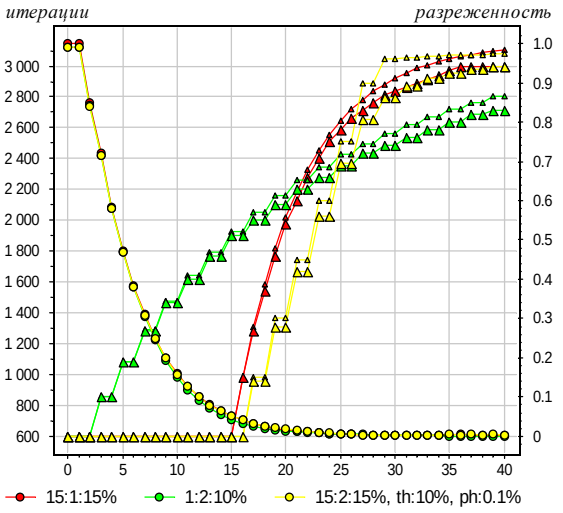
RuDis, разреживание через 2 итерации



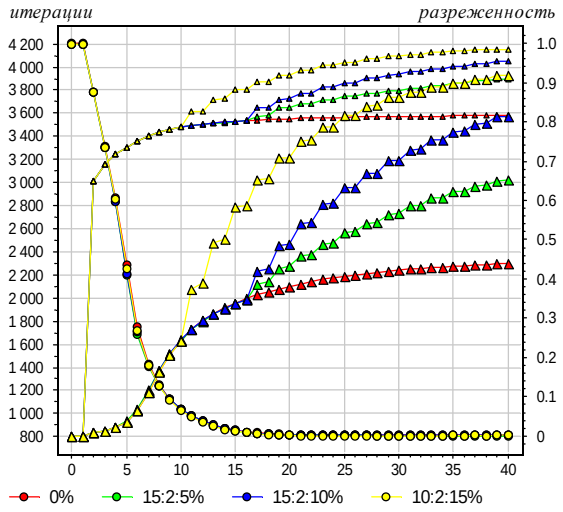
NIPS, разреживание через 2 итерации



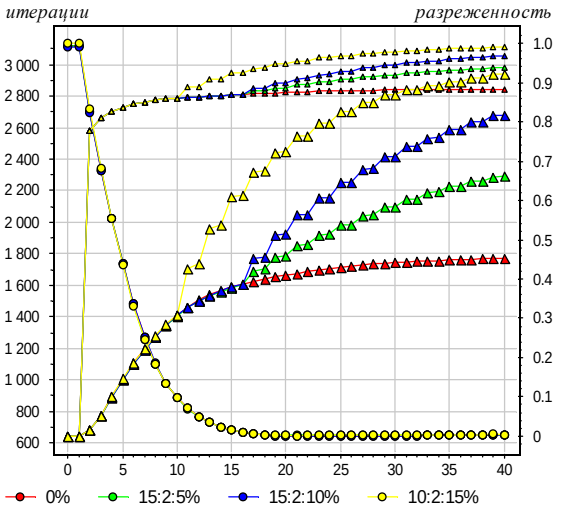
RuDis, агрессивное разреживание



NIPS, агрессивное разреживание

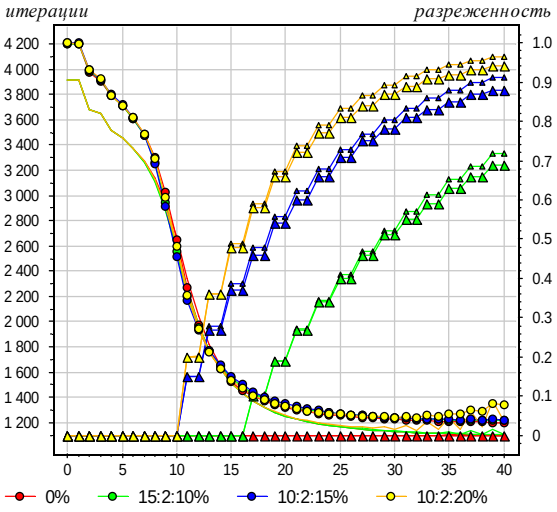


RuDis, SEM, через 2 итерации

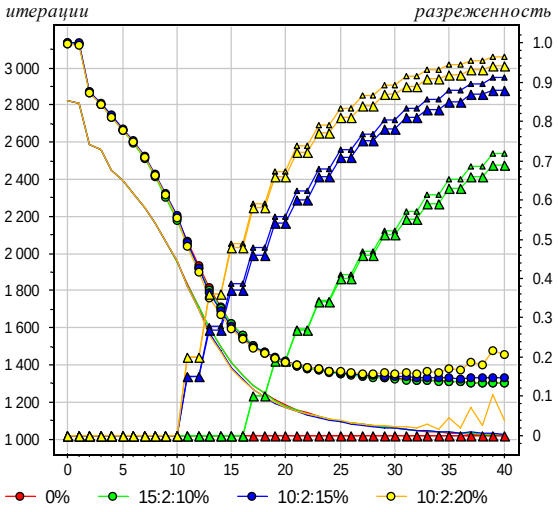


NIPS, SEM, через 2 итерации

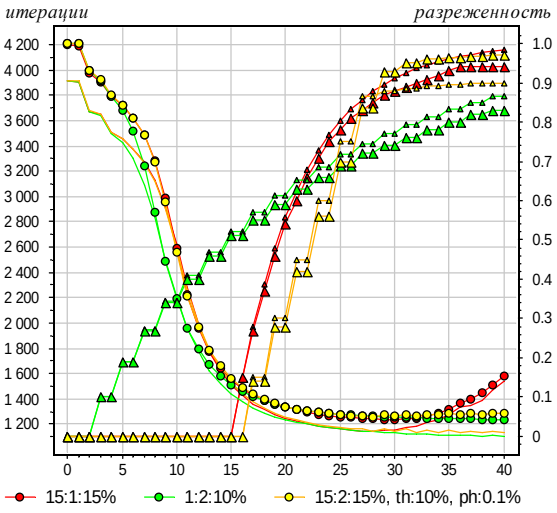
Рис. 7: Зависимость перплексии (○) и разреженности матриц Φ (△) и Θ (△) от числа итераций для рационального и стохастического робастного EM-алгоритма с параметрами робастности $\gamma = 0.3$, $\varepsilon = 0.01$ и параметрами разреживания $i_0{:}\delta{:}r$, $\mathsf{th}{:}R_\theta$, $\mathsf{ph}{:}R_\varphi$. Число тем $|T| = 100$



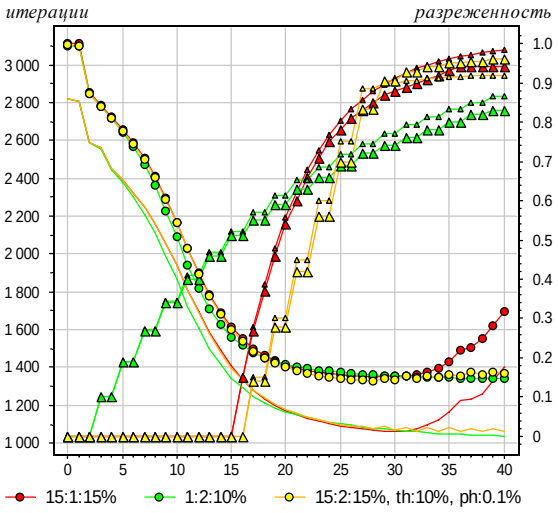
RuDis, разреживание через 2 итерации



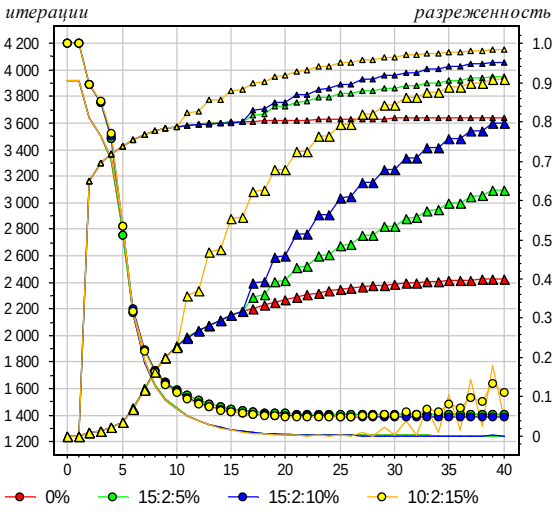
NIPS, разреживание через 2 итерации



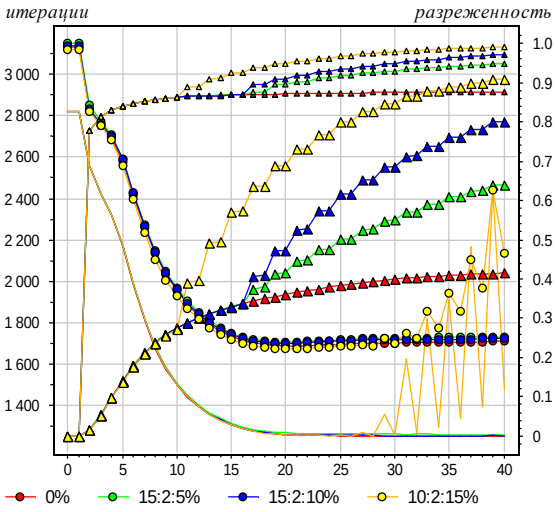
RuDis, агрессивное разреживание



NIPS, агрессивное разреживание



RuDis, SEM, через 2 итерации



NIPS, SEM, через 2 итерации

Рис. 8: Зависимость перплексии (○) и разреженности матриц Φ (△) и Θ (△) от числа итераций для рационального и стохастического робастного EM-алгоритма при малой априорной вероятности шума $\gamma = 0.01$, $\varepsilon = 0.01$, с параметрами разреживания $i_0{:}\delta{:}r$, $\mathsf{th}{:}R_\theta$, $\mathsf{ph}{:}R_\varphi$. Число тем $|T| = 100$

Регуляризация Дирихле приводит к сглаживанию частотных оценок условных вероятностей (13)–(14), что является единственным принципиальным отличием LDA от PLSA. В экспериментах оптимальные значения гиперпараметров α_t и β_w оказываются достаточно близкими к нулю [23]. Для большинства тем в документах $\alpha_t \ll n_{td}$; для большинства терминов в темах $\beta_w \ll n_{wt}$. Оценки параметров φ_{wt} и θ_{td} в PLSA и LDA заметно отличаются только для тем, очень редких в документе, и терминов, очень редких в теме. Они не несут статистически значимой информации о тематике. Их следовало бы проигнорировать, как шум, но вместо этого LDA, наоборот, повышает оценку их вероятности.

Утверждение о том, что LDA сокращает эффективную размерность пространства параметров [5], звучит неубедительно. PLSA и LDA оценивают параметры φ_{wt} и θ_{td} по одним и тем же формулам (13)–(14). Более того, в LDA вводятся дополнительные гиперпараметры α_t, β_w, которые также приходится оценивать [23].

Утверждение о том, что LDA гораздо меньше переобучается [5], не выдерживает аккуратной перепроверки. Качество тематических моделей принято сравнивать по контрольной перплексии, которая может резко повышаться при появлении в контрольных документах редких терминов, для которых модель предсказывает вероятность $p(w \mid d)$, близкую к нулю. В PLSA эта вероятность может оказаться равной нулю, тогда перплексия формально будет равна $+\infty$. Это выглядит как переобучение, однако по сути им не является, так как небольшую долю редких терминов вполне допустимо интерпретировать как нетематический шум. В LDA вероятности редких терминов не стремятся к нулю благодаря сглаженным оценкам φ_{wt} и θ_{td}. Модель LDA более толерантна к нетематическим терминам, но она не выделяет их в явном виде, как это делает робастная модель или SWB.

Если из контрольных документов убрать новые термины, то контрольные перплексии PLSA и LDA практически совпадают [1]. Недавние исследования [16, 26, 15], также подтверждают, что для больших коллекций нет существенных различий в качестве моделей PLSA и LDA. Реальные коллекции настолько велики, что переобучение не является проблемой для обеих моделей. Значимые отличия контрольной перплексии PLSA и LDA в ранних экспериментах [5], могут быть объяснены тем, что для них использовались существенно различные реализации алгоритмов обучения. В наших экспериментах использовался один и тот же алгоритм обучения для моделей PLSA и LDA, отличавшийся только сглаженными оценками в LDA. Сравнивать порождающие модели при существенно различных методах их обучения, вообще говоря, некорректно.

Таким образом, роль априорных распределений Дирихле оказывается весьма скромной — это не сокращение размерности и не уменьшение переобучения, а всего лишь более толерантное оценивание редких терминов, незначимых для выявления тематики. В то же время, регуляризация порождает свои проблемы. Она противоречит гипотезе разреженности и вводит гиперпараметры α_t и β_w, которые приходится подбирать. При появлении документов с новыми терминами w не ясно, как инициализировать β_w. Сглаженные оценки являются смещенными, в отличие от оценок максимума правдоподобия.

Заключение

Описан широкий класс методов тематического моделирования на базе обобщенного EM-алгоритма и эвристик сглаживания, сэмплирования, частого обновления параметров, робастности и разреживания, которые могут сочетаться в различных комбинациях.

В экспериментах на двух текстовых коллекциях получены следующие выводы.

1. Робастные алгоритмы с разреживанием являются лучшими по критерию контрольной перплексии и не требует введения априорных распределений Дирихле. Эвристика сглаживания для них оказывается избыточной.

2. Контрольная перплексия LDA лучше, чем у PLSA не потому, что PLSA переобучается, а потому, что LDA завышает оценки вероятности редких слов. При корректном сравнении на больших коллекциях перплексии PLSA и LDA практически не различаются.

3. Принудительное разреживание в робастных моделях PLSA позволяет обнулять до 99% параметров без ухудшения контрольной перплексии.

4. Упрощенная робастная модель с разреживанием, в отличие от модели SWB, четко выделяет в документах нетематические термины, не требует хранения параметров π_{dw}, не требует задания параметров γ и ε, и почти не увеличивает объем вычислений.

5. Наряду с сэмплированием Гиббса возможны и другие стратегии разреживания распределений $p(t \mid d, w)$, в частности, сэмплирование небольшого фиксированного числа s тем и постепенное разреживание путем обнуления небольшой доли наименьших вероятностей.

6. На достаточно больших коллекциях (10^6 терминов и более) обучающая и контрольная перплексия ведут себя практически одинаково и приводят к одинаковым качественным выводам. Таким образом, нет необходимости вычислять контрольную перплексию.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект № 11-07-00480) и программы ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения».

Авторы выражают глубокую признательность рецензентам за ценные замечания, способствовавшие улучшению изложения.

Литература

[1] К. В. Воронцов and А. А. Потапенко. Регуляризация, робастность и разреженность вероятностных тематических моделей. *Компьютерные исследования и моделирование*, 4(4):693–706, 2012.

[2] Н. В. Лукашевич. *Тезаурусы в задачах информационного поиска*. Издательство МГУ имени М. В. Ломоносова, 2011.

[3] С. В. Царьков. Автоматическое выделение ключевых фраз для построения словаря терминов в тематических моделях коллекций текстовых документов. *Естественные и технические науки*, 62(6):456–464, 2012.

[4] A. Asuncion, M. Welling, P. Smyth, and Y. W. Teh. On smoothing and inference for topic models. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2009.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[6] Gilles Celeux, Didier Chauveau, and Jean Diebolt. On stochastic versions of the EM algorithm. Technical Report RR-2514, INRIA, 1995.

[7] C. Chemudugunta, P. Smyth, and M. Steyvers. *Modeling general and specific aspects of documents with a probabilistic topic model*, volume 19, pages 241–248. MIT Press, 2007.

[8] Ali Daud, Juanzi Li, Lizhu Zhou, and Faqir Muhammad. Knowledge discovery through directed probabilistic topic models: a survey. *Frontiers of Computer Science in China*, 4(2):280–301, 2010.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society, Series B*, (34):1–38, 1977.

[10] Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. Sparse additive generative models of text. In *ICML'11*, pages 1041–1048, 2011.

[11] Mark Girolami and Ata Kabán. On an equivalence between PLSI and LDA. In *SIGIR'03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 433–434, 2003.

[12] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA, 1999. ACM.

[13] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, 2001.

[14] Martin O. Larsson and Johan Ugander. A concave regularization technique for sparse mixture models. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1890–1898, 2011.

[15] Yue Lu, Qiaozhu Mei, and ChengXiang Zhai. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. *Information Retrieval*, 14(2):178–203, 2011.

[16] Tomonari Masada, Senya Kiyasu, and Sueharu Miyahara. Comparing LDA with pLSI as a dimensionality reduction method in document clustering. In *Proceedings of the 3rd International Conference on Large-scale knowledge resources: construction and application*, LKR'08, pages 13–26. Springer-Verlag, 2008.

[17] Radford M. Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In Michael I. Jordan, editor, *Learning in graphical models*, pages 355–368. MIT Press, Cambridge, MA, USA, 1999.

[18] Pavel Pecina and Pavel Schlesinger. Combining association measures for collocation extraction. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 651–658. Association for Computational Linguistics, 2006.

[19] A. A. Potapenko and K. V. Vorontsov. Robust PLSA performs better than LDA. In *35th European Conference on Information Retrieval, ECIR-2013, Moscow, Russia, 24-27 March 2013*, pages 784–787. Lecture Notes in Computer Science (LNCS), Springer Verlag-Germany, 2013.

[20] Mark Steyvers and Tom Griffiths. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, 2004.

[21] Yee Whye Teh, David Newman, and Max Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *NIPS*, pages 1353–1360, 2006.

[22] Hanna Wallach. *Structured Topic Models for Language*. PhD thesis, Newnham College, University of Cambridge, 2008.

[23] Hanna Wallach, David Mimno, and Andrew McCallum. Rethinking LDA: Why priors matter. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1973–1981, 2009.

[24] Chong Wang and David M. Blei. Decoupling sparsity and smoothness in the discrete hierarchical dirichlet process. In *NIPS*, pages 1982–1989. Curran Associates, Inc., 2009.

[25] Yi Wang. Distributed Gibbs sampling of latent dirichlet allocation: The gritty details, 2008.

[26] Yonghui Wu, Yuxin Ding, Xiaolong Wang, and Jun Xu. A comparative study of topic models for topic clustering of chinese web news. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 5, pages 236–240, july 2010.

[27] Ziqi Zhang, José Iria, Christopher Brewster, and Fabio Ciravegna. A comparative evaluation of term recognition algorithms. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC08)*, 2008.

Автоматический анализ формы спирографических петель по их сигнатурам*

Манило Л. А., Немирко А. П., Саламонова И. С.
lmanilo@yandex.ru
СПбГЭТУ «ЛЭТИ»

Рассмотрены методы автоматического анализа спирографических петель в условиях искусственной вентиляции легких. Проведен анализ различных числовых характеристик петель «объем-давление», полученных по сигнатуре двумерных кривых. Показаны возможности применения динамического анализа спирограмм для оценки параметров вентиляции легких и ранней диагностики патологий.

Ключевые слова: *искусственная вентиляция легких, спирографическая петля, петля «объем-давление», анализ формы, сигнатура петли.*

Automatic analysis of form of spirographic loops on their signatures*

Manilo L. A., Nemirko A. P., Salamonova I. S.
SPbETU ¡¡LETI¿¿

The analysis of various numerical characteristics of loops “volume-pressure” obtained from a signature of two-dimensional curves is carried out. Possibilities of application of the dynamic analysis of spirograms for an assessment of parameters of ventilation of lungs and early diagnostics of pathologies are shown. Methods of the automatic analysis of loops in the conditions of mechanical ventilation are considered.

Keywords: *anapnotherapy, spirographic loop, pressure-volume loop, shape analysis, loop signature.*

Введение

Контроль состояния пациента в режиме искусственной вентиляции легких (ИВЛ) основан на динамической оценке ряда спирографических показателей, которые характеризуют эффективность газообмена. К ним в первую очередь относят такие параметры вентиляции, как жизненная емкость легких, сопротивление дыхательных путей, растяжимость легких. В современных аппаратах ИВЛ, обеспечивающих функцию «протезирования дыхания», требуется также реализация диагностических функций, позволяющих на ранних стадиях обнаруживать развитие состояний, угрожающих жизни пациента. К таким опасным состояниям можно отнести, например, отек легких, обструктивные нарушения в бронхолегочной системе пациента [1, 2].

Ранняя диагностика патологий органов дыхания возможна лишь в ходе непрерывного контроля за состоянием пациента. Она основана на автоматическом анализе основных параметров внешнего дыхания, а также обнаружении существенных отклонений в заданных режимах ИВЛ. Такой анализ можно осуществить по спирографическим кривым, включающим как скалярные функции (давление, поток и объем), так и двумерные функции, представленные в виде петель «объем–давление» (ОД) и «поток–объем» (ПО). В работах, посвященных автоматическому анализу спирограмм при ИВЛ, рассматриваются процедуры измерения параметров вентиляции лег-

Работа выполнена при финансовой поддержке Министерства образования и науки РФ, государственный контракт № 16.522.12.2016, и РФФИ, проекты № 12-01-00583, № 13-01-00540.

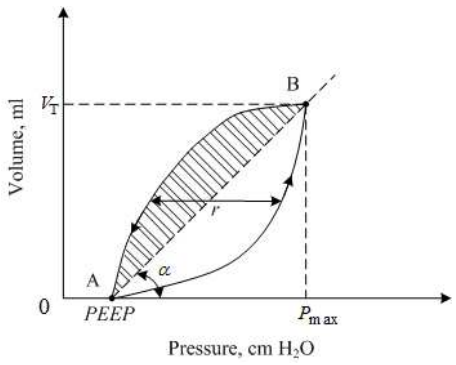


Рис. 1: Модельная петля «объем–давление»

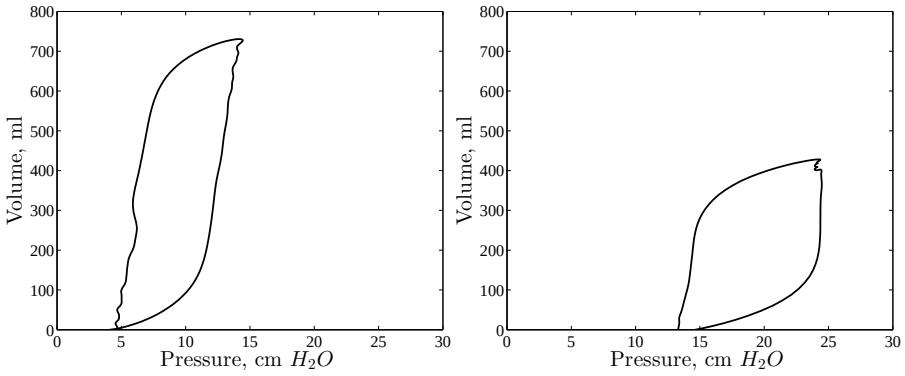
ких по скалярным кривым, однако петли привлекаются только для визуального пояснения изменений, происходящих в ходе мониторирования пациентов [3, 4, 5]. В то же время установлено, что этот вид спирограмм представляет наибольший практический интерес для задач мониторинга в процессе проведения респираторной поддержки [1].

Данное исследование направлено на разработку новых методов и алгоритмов непрерывного контроля функций внешнего дыхания при ИВЛ по спирографическим петлям, в частности, по петлям ОД. Новизна подхода заключается в использовании для распознавания существенных отклонений в заданных режимах ИВЛ двумерных кривых. Предполагается, что переход к автоматическому анализу петель дыхания позволит более точно регулировать параметры вентиляции и своевременно диагностировать развитие патологий. В задачу исследования входило формирование набора признаков, отражающих особенности изменения формы регистрируемых петель ОД, а также оценка возможности использования сигнатуры для обнаружения отклонений, вызванных развитием патологий органов дыхания.

Петли «объем-давление»

Петля ОД является графической формой описания функциональной зависимости дыхательного объема V от давления P в контуре системы дыхания. В ходе регистрации кривых петля образуется на протяжении каждого дыхательного цикла. Она одновременно отражает влияние двух физиологических параметров: растяжимости легких C (Compliance) и сопротивления дыхательных путей R (Resistance) [1, 2]. Типичная кривая ОД при ИВЛ с управляемым объемом для одного дыхательного цикла изображена на рис. 1. Точка A соответствует началу вдоха (концу выдоха), точка B — концу вдоха (началу выдоха), а величина V_T показывает дыхательный объем легких. Ввиду того, что существует запаздывание изменения величины объема V относительно давления P, график имеет вид петли гистерезиса. Нижняя ветвь петли связана с работой дыхания по растяжению эластичных тканей легких во время вдоха, а верхняя — с работой дыхания по преодолению сопротивления дыхательных путей на выдохе. Поскольку давление в конце выдоха должно поддерживаться на заданном уровне, кривая дыхания смещена вдоль оси давления P на положительную величину $PEEP$ (Positive End-Expiratory Pressure). Линия, соединяющая две характерные точки кривой (A, B), задает направление основной оси петли. Наклон оси, равный углу α, характеризует величину динамической растяжимости дыхательной системы $C = \operatorname{tg}\alpha$, а ширина петли r — величину сопротивления дыхательных путей R. При этом расстояния от оси петли до ее восходящей и нисходящей ветвей (на рис. 1 указано стрелками) отражают инспираторное R_i и экспираторное R_e сопротивления, соответственно. Кроме горизонтальной протяженности петли r с изменением величины сопротивления дыхательных путей связан другой интегральный параметр — площадь петли S_e в фазе выдоха (на рис. 1 отмечено штриховкой).

Наблюдение за динамикой формы петли, положением кривой на плоскости, углом наклона α и шириной петли r позволяет косвенно судить об изменениях основных спирометрических показателей C и R, описывающих работу дыхания пациента [1, 2]. На рис. 2 в качестве примера



(а) Норма

(б) Острый респираторный дистресс-синдром

Рис. 2: Пример спирограмм пациентов в режиме искусственной вентиляции легких

изображены две спирограммы пациентов, зарегистрированные в условиях клиники, для случаев нормы (рис. 2, а) и наличия острого респираторного дистресс-синдрома — ОРДС (рис. 2, б). Можно заметить, что при патологии изменяются форма, размеры петли, расположение ее на плоскости. На кривой ОД наблюдается отклонение петли к горизонтальной оси (снижается растяжимость легких C) и расширение ее вдоль оси давления (повышается сопротивление дыхательных путей R).

Анализ формы петель по их сигнатурам

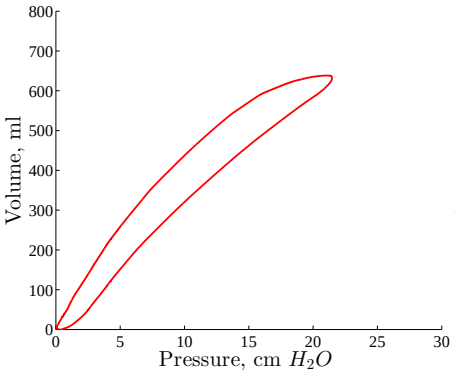
Существуют различные способы описания формы петель. Они основаны на использовании цепных кодов, аппроксимации фигур полиномами разных степеней, описании границ набором числовых признаков (площадь, длина, направление главных осей замкнутой фигуры и т. д.), представлении двумерных кривых в виде сигнатур [6]. Преимуществом сигнатур является то, что описание двумерных петель сводится к более простым одномерным функциям. В данной работе использован последний подход.

Наиболее простой способ получения сигнатуры петли состоит в построении зависимости расстояния от центроида петли до границ кривой в виде функции угла θ с применением равномерной дискретизации по углу (во всех экспериментах $\Delta\theta = 1^\circ$). Анализ изменений формы петель по сигнатуре предполагает независимость строящихся функций от поворота и размера петель. Первое условие достигается выбором фиксированной начальной точки отсчета сигнатуры. В качестве такой точки можно выбрать точку начала дыхательного цикла (точка A, рис. 1). Тогда можно проводить сравнение формы петель по скалярным кривым (сигнатуре), поскольку они будут синхронизированы с фазами дыхательного цикла. Инвариантность функции к размеру петли обеспечивается нормировкой отсчетов сигнатуры по максимальному значению или диапазону изменения ее ординат.

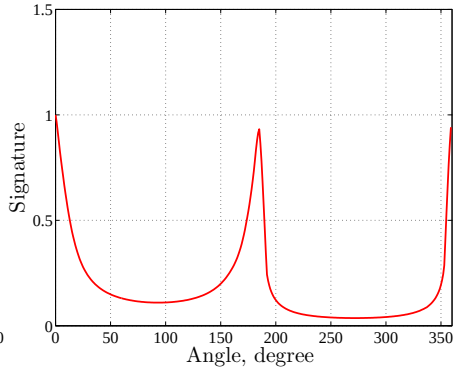
Используя цифровое изображение петли, а также функцию сигнатуры, можно вычислить ряд параметров, характеризующих форму и расположение петли на плоскости, провести сравнительный анализ спирограммы, регистрируемой в ходе наблюдения, и опорной петли. В качестве опорных петель необходимо выбрать те, которые отражают установленный врачом режим вентиляции легких. В процессе непрерывного контроля опорные кривые могут периодически обновляться, что связано с необходимостью корректировки задаваемых врачом параметров вентиляции.

Алгоритм динамического анализа петель «объем-давление» по их сигнатурам

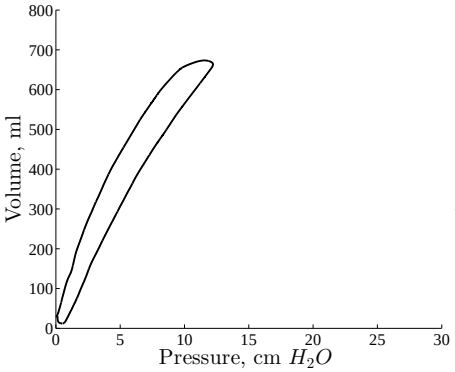
Алгоритм динамического анализа петель ОД и распознавания патологических отклонений в режиме вентиляции легких, можно представить следующим образом.



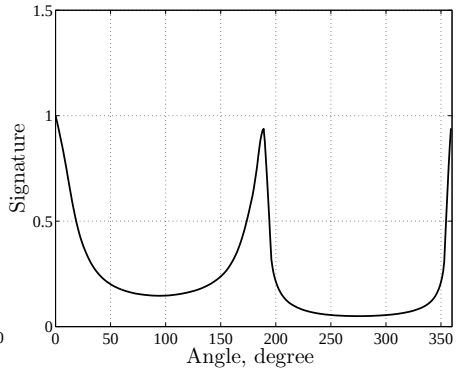
(а) Острый респираторный дистресс-синдром



(б) Сигнатура



(в) Норма



(г) Сигнатура

Рис. 3: Сравнение петель «объем-давление» в случае снижения величины Compliance ($\Delta\sigma = 0,08$; $\Delta\alpha = -15°$;$\Delta r = 0,05$; $\Delta S_e = 0,03$)

Вначале необходимо провести нормировку каждой из координатных осей (V и P) с тем, чтобы, во-первых, перейти к безразмерным величинам, и, во-вторых, получить расположение петель на плоскости в соответствии с графическим отображением линии нормальной динамической растяжимости легких. Как известно, эта линия должна располагаться так, чтобы значению растяжимости 65 мл/см вод. ст. соответствовал угол наклона α оси петли, равный 45° [2]. Поэтому при построении сигнатуры нами использована нормировка в виде $V_{\text{n}} = V/V^*$ и $P_{\text{n}} = P/P^*$, где нормирующие величины $V^* = 1300$ мл, $P^* = 20$ см. вод. ст.

Далее выполняются процедуры обработки нормированных петель и сигнатур, а также проводится сравнительный анализ параметров, характеризующих особенности формы спирографических кривых. В качестве этих параметров выбраны:

1) направление основной оси анализируемой петли ОД ($\operatorname{tg}\alpha$) или угловое отклонение от оси опорного дыхательного цикла ($\Delta\alpha$);

2) величина среднеквадратического отклонения σ_k спирограммы;

3) ширина петли r (или оценка площади петли ОД в фазе выдоха S_e);

4) выраженность артефактов L_a.

Включение режима слежения за спирограммой позволяет фиксировать отклонения этих признаков на величину, превышающую некоторый заданный в процентном отношении порог. Как показали эксперименты, сравнение петель целесообразно проводить по усредненным кривым,

полученным путем синхронного накопления двумерных функций. В этом случае значительно снижается влияние помех на качество анализа.

Наклон оси петли $\operatorname{tg}\alpha$ является оценкой динамической растяжимости легких $C = \operatorname{tg}\alpha$. Если измерять угловое отклонение $\Delta\alpha$, то можно оценивать степень изменения динамического Compliance. Введение порога на величину $\Delta\alpha$ позволяет обнаруживать патологические изменения, связанные с резким увеличением или уменьшением растяжимости легких.

Величина среднеквадратического отклонения сигнатуры используется в качестве оценки изменения формы петли относительно опорной кривой. Она вычисляется в виде:

$$\sigma_k = \sqrt{\frac{1}{n}\sum_{\theta}(r^*_\theta - r_{k\theta})^2},$$

где r^*_θ — отсчеты сигнатуры опорной петли; $r_{k\theta}$ — отсчеты сигнатуры для k-го порядкового дыхательного цикла; $\theta = 1^\circ, \ldots, 360^\circ$; n — размер выборки ($n = 360$).

Фактически данный параметр оценивает ошибку, связанную с отклонением в форме петель. Может быть использована и квадратичная форма данного критерия $(\sigma_k)^2$. Введение порога для величины σ_k позволяет выявить существенные отклонения в заданных режимах ИВЛ.

Сопротивление дыхательных путей в нашей работе предложено оценивать одним из двух способов. Первый основан на измерении ширины петли r на уровне середины осевой линии. Второй способ предполагает вычисление площади фигуры S_e, ограниченной осью петли и частью кривой в фазе выдоха. Учитывая трудности интерпретации измеренных отклонений в случае изменения размеров петли (что связано, например, с необходимостью увеличения V_T при ИВЛ), в работе предложено использовать дополнительную нормировку петель по максимальным значениям объема V_T и давления P_max опорной петли. В этом случае опорная петля, условно считающаяся «нормой», располагается по линии нормальной динамической растяжимости, а изменения параметров r и S_e подлежат сравнению даже при изменении параметров управления вентиляцией легких.

Показатель выраженности артефактов рассчитывался в виде $L_a = L^2/S$, где L — длина контура петли, S — площадь петли. Динамическая оценка его важна для выявления значительных колебаний контура петли, что связано с влиянием различных факторов (изменение тонуса мышц пациента, наличие препятствий в дыхательном контуре).

Экспериментальные исследования

В работе проведено моделирование алгоритма автоматического анализа петель ОД с применением специально разработанной программы, в основе которой лежит функция signature [7]. В качестве входных сигналов использовались:

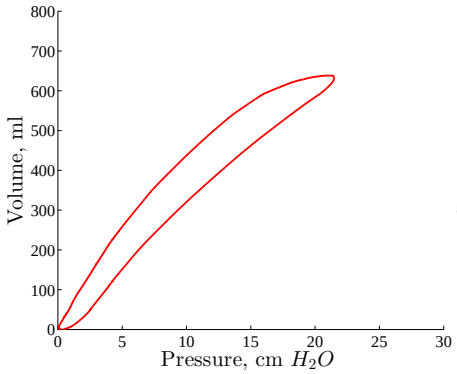
1) модельные кривые, полученные путем перевода спирографических изображений в цифровую форму (12 записей) [2];

2) сигналы, полученные на аппарате ИВЛ путем механического моделирования изменений растяжимости C и сопротивления R дыхательных путей (23 записи);

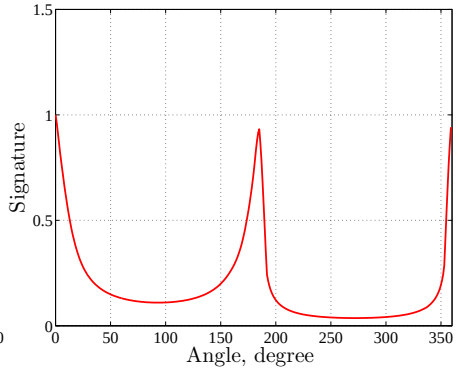
3) реальные записи спирограмм, зарегистрированные в условиях клиники в процессе респираторной поддержки пациентов (спирограммы 15 пациентов).

Реальные данные включали как выборку записей без патологий (5 записей), так и сигналы с тяжелыми нарушениями органов дыхания: ОРДС (10 записей) и обструкция дыхательных путей (5 записей). Эти нарушения были рассмотрены в качестве основных при разработке алгоритма распознавания патологий.

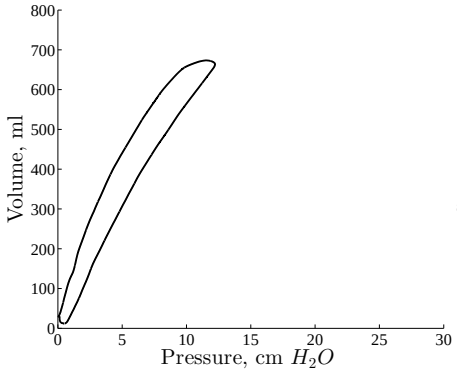
Процесс обработки изображений, исходно представленных в виде битовых матриц, включал в себя следующие этапы: получение описания изображения в виде траектории единичной толщины; приведение описания изображений к фактическому масштабу, указанному для исходных



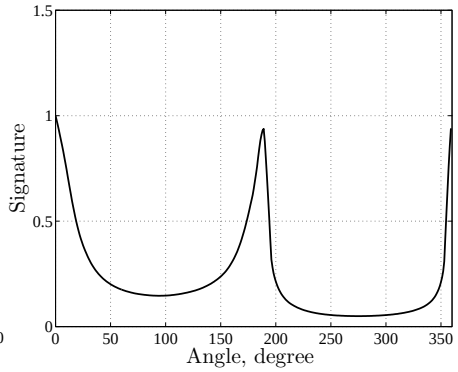
(а) Острый респираторный дистресс-синдром



(б) Сигнатура



(в) Норма



(г) Сигнатура

Рис. 4: Сравнение петель «объем-давление» в случае снижения величины Compliance ($\Delta\sigma = 0,08$; $\Delta\alpha = -15°$;$\Delta r = 0,05$; $\Delta S_e = 0,03$)

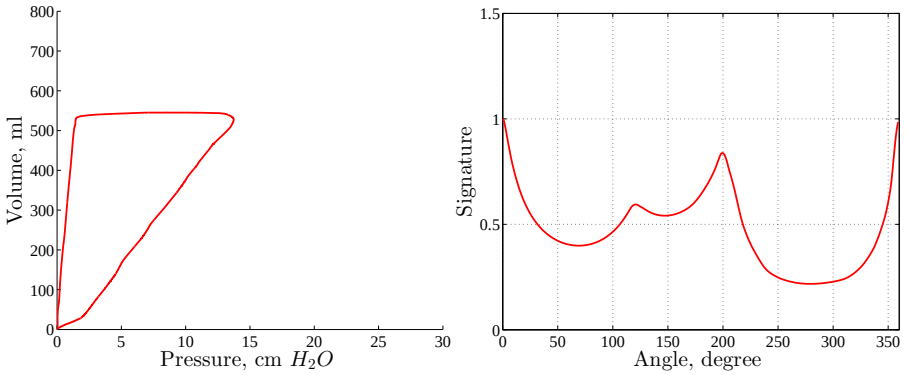
графиков; нахождение узловых точек кривых; интерполяция петель кубическими сплайнами и получение сигналов в виде последовательностей равномерно дискретизованных отсчетов. По двумерным нормализованным кривым определялись две опорные точки (начало и конец вдоха), по которым оценивался угол наклона петли α. Далее вычислялась функция сигнатуры и по ней путем сравнения одномерных описаний двух петель (анализируемой и опорной) определялись величины σ_k и r.

В качестве примера на рис. 3, 4 представлены реальные петли ОД. Слева приведены спирографические кривые, справа — сигнатуры петель. Нижние кривые соответствует нормальной форме спирограмм и являются опорными. Верхние петли, имеющие патологические отклонения, сравниваются с опорными.

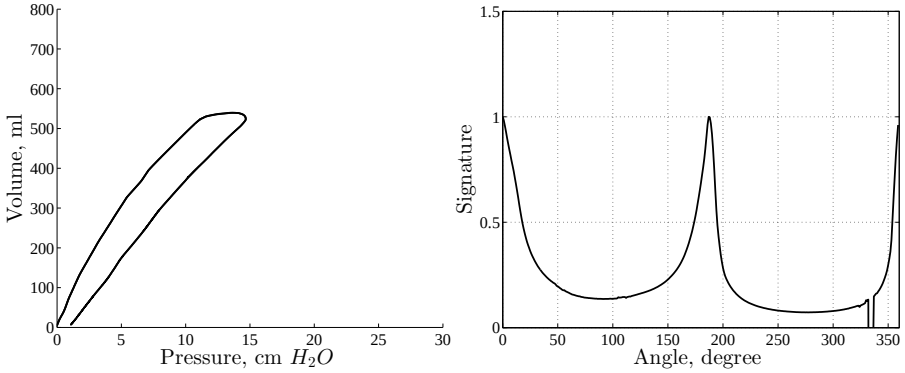
На рис. 3 показаны петли ОД для случая снижения величины Compliance, что вызывает поворот петли вправо. Это изменение обусловлено тем, что для достижения данного уровня объема дыхания (650 мл) требуется большее давление. Подобный вид петель можно наблюдать на поздних стадиях ОРДС. Наблюдаемое снижение величины Compliance может происходить как постепенно, при развитии легочных заболеваний, так и внезапно при острых нарушениях.

На рис. 4 показан случай расширения петли ОД, что вызвано повышением сопротивления дыхательных путей на выдохе, причем инспираторные участки двух кривых аналогичны. Возможными причинами таких отклонений может быть появление нарушений проходимости (обструкция дыхательных путей) [2].

Результаты сравнения числовых характеристик сигнатур анализируемых петель показывают, что в первом случае (рис. 3) наблюдается снижение растяжимости легких ($\Delta\alpha = -15°$), а



(а) Обструкция дыхательных путей (б) Сигнатура



(в) Норма (г) Сигнатура

Рис. 5: Сравнение петель «объем-давление» в случае повышения величины *Resistance* ($\Delta\sigma = 0,31$; $\Delta\alpha = 2°$; $\Delta r = 0,96$; $\Delta S_e = 0,08$)

во втором случае (рис. 4) наклон петли сохраняется($\Delta\alpha = 2°$), но изменяется форма петли. Чем больше отличия в форме петель, тем большие значения получает критерий среднеквадратического отклонения. В нашем случае величина $\Delta\sigma = 0,31$ превышает значение порога, равного 0,2. Это связано с резким расширением петли ($\Delta r = 0,96$) и увеличением ее площади ($\Delta S_e = 0,08$).

Распознавание этих опасных нарушений на ранних стадиях можно провести, используя полученные характеристики. Резкое уменьшение угла наклона α осевой линии петли ОД $\left(|\Delta\alpha| > 12°\right)$ свидетельствует об увеличении жесткости легких и указывает на развитие возможных осложнений (ОРДС). Резкое изменение величины σ, увеличение значений параметров r, или S_e, что можно представить в виде логического выражения $\Big((\Delta\sigma > 0,2) \bigwedge \big((\Delta r > 0,4) \bigvee (\Delta S_e > 0,05)\big)\Big)$, указывает на значительное расширение петли и свидетельствует о развитии обструкции дыхательных путей.

В работе проведены эксперименты по оценке эффективности распознавания различных форм петель по сигнатуре (реальные записи спирограмм) и обнаружению значимых изменений динамической растяжимости C и сопротивления R дыхательных путей (модельные сигналы). При моделировании спирограмм задавались разные значения Compliance и Resistance, что обеспечило возможность контроля изменений этих параметров. Результаты экспериментов на реальных записях показали, что во всех 15 случаях были безошибочно распознаны нарушения: ОРДС и обструкция дыхательных путей. В ходе экспериментов на модельных сигналах (30 испытаний) удалось безошибочно распознать кривые с измененной формой петли ОД, а также с отклонениями в динамической растяжимости легких. И только в двух случаях из 30 (6,7%) не удалось зафиксировать отклонения величины сопротивления дыхательных путей. Анализ записей показал, что

эти ошибки были связаны с заданием слишком малых величин управляемого потока. Полученные результаты свидетельствуют о возможности применения предложенных параметров сигнатуры петель для эффективного распознавания двух наиболее тяжелых патологий органов дыхания, а также косвенной оценки изменения основных спирографических показателей: Compliance и Resistance.

Кроме того, проведен анализ статистической зависимости между заданными величинами R и вычисленными оценками Resistance: r, r_e, S_e. Значения попарных коэффициентов корреляции k оказались достаточно высокими для всех трех параметров ($k > 0,85$), однако наибольшее значение коэффициента корреляции $k = 0,91$ получено для параметра r. Это указывает на то, что данный параметр является наиболее важным для контроля значимых изменений Resistance по петлям ОД. Поэтому в окончательное решающее правило для распознавания обструкции дыхательных путей вместо условий для r и S_e было включено только одно условие ($\Delta r > 0,4$).

Заключение

Таким образом, в работе предложены числовые характеристики спирографических петель, позволяющие обнаруживать существенные отклонения в заданных режимах ИВЛ и распознавать ранние формы развития патологий дыхательных путей. Кроме того, показана возможность косвенной оценки изменений динамической растяжимости и сопротивления дыхательных путей по числовым параметрам сигнатуры петель.

Разработан комплекс программ автоматического анализа спирограмм, который предполагается использовать в аппаратах ИВЛ для расширения их диагностических возможностей. Как показали исследования в клинических условиях, постоянный контроль за формой кривой ОД, ее наклоном и шириной являются важной составляющей респираторной поддержки, особенно у пациентов с развивающейся патологией легких.

Литература

[1] *Лебединский К. М., Мазурок В. А., Нефедов А. В.* Основы респираторной поддержки. СПб.: Изд. МАПО, 2006. 213 с.

[2] *Waugh J. B., Deshpande V. M., Brown M. K., Harwood R. J.* Rapid шnterpretation of mentilator цaveforms. New Jersey: Upper Saddle River, 2007. 151 pp.

[3] *Stenqvist O.* Practical assessment of respiratory mechanics // British J. Anaesthesia. 2003. No. 91(1). Pp. 92–105

[4] *Khirani S., Polese G., Aliverti A., at alias.* On-line monitoring of lung mechanics during spontaneous breathing: a physiological study // Respiratory Medicine. 2010. No. 104. Pp. 463–471

[5] *Karason S., Sondergaard S., Lundin S., Stenqvist O.* Continuous on-line measurements of respiratory system, lung and chest wall mechanics during mechanic ventilation // Intensive Care Medicine. 2001. No. 27. Pp. 1328–1339

[6] *Гонсалес Р., Вудс Р.* Цифровая обработка изображений. Пер. с англ. М.: Техносфера, 2005. 1072 с.

[7] *Гонсалес Р., Вудс Р., Эддинс С.,* Цифровая обработка изображений в среде MATLAB. Пер. с англ. М.: Техносфера, 2006. 616 с.

Информативность признаков для диагностики состояния подшипников на основе обнаружения локальных неоднородностей

Чувилина Е. В.
e.v.chuvilina@gmail.com
Рыбинский государственный авиационный технический университет имени П. А. Соловьева

Рассматривается задача диагностики состояния подшипников газотурбинных двигателей (ГТД) как задача распознавания образов на основе обнаружения локальных неоднородностей в вибросигнале. Предложены и исследованы ряд признаковых пространств, выделены наиболее информативные из них, имеющие линейную разделимость, а именно: изменение фрактальной размерности, векторы коэффициентов сноса, матрицы зависимости приращения от величины сигнала.

Ключевые слова: *вибродиагностика, локальные неоднородности, фрактальная размерность, вектор коэффициентов сноса, подшипники, информативность.*

Informative Features for Diagnostics of Bearings by Detection of Local Inhomogeneities

Chuvilina E. V.
P. A. Solovyev Rybinsk State Aviation Technical University

The task of diagnostics of bearings in a gasturbine engine is seen as a pattern recognition problem based on the detection of local inhomogeneities in the vibrosignal. A number of features are proposed and investigated; most informative ones having a linear separability, namely, fractal dimension, drift vector, and matrix of dependence increments from the signal, are allocated.

Keywords: *vibrodiagnostics, local inhomogenities, fractal dimension, vector of drift coefficients, bearings, informativity.*

Введение

Техническое состояние ГТД во многом определяет надежность и безопасность летательных аппаратов. По мере эксплуатации в деталях двигателя могут возникать дефекты. Взаимодействие отдельных узлов ГТД приводит к генерации сложных колебательных процессов, что позволяет контролировать и диагностировать состояние ГТД по вибрационным параметрам. Одним из критических элементов, от которых зависит работоспособность двигателя, являются подшипниковые узлы, поскольку они воспринимают большую часть статических и динамических усилий, возникающих в работающем механизме. Задача задачи диагностики технического состояния подшипников ГТД в настоящее время решена неудовлетворительно, оптимальное решение не найдено. В производстве для получения и исследования сигналов подшипников используются виброизмерительные приборы ИВУ-1 и MIC-200 [1], включающие в себя набор алгоритмов, таких как преобразование Фурье, вычисление корреляционных функций, статистических характеристик (среднеквадратичное значение, эксцесс, асимметрия), распознавание состояния подшипника выполняется оператором путем перебора получаемых характеристик. При диагностике этими приборами возникают ситуации, когда кондиционный подшипник относят к плохим. Возникает задача классифиции, при которой необоснованно снятые подшипники распознавались бы

как кондиционные. Практически все виброакустические методы контроля основаны либо на анализе самого сигнала, либо на анализе его частотных характеристик. В статье [2] представлены методы на основе декомпозиции вибросигнала по Wavelet-коэффициентам и на основе графического представления сигнала, которые хотя и помогают уточнить результаты анализа, но имеют скорее обощающий, чем различающий характер. В статье [3] предложен метод характерных последовательностей, имеющий высокую информативность, однако являющийся очень трудоемким и времезатратным.

Таким образом, возникает задача поиска наиболее информативных признаков, которые можно использовать для классификации подшипников ГТД.

Задача диагностики состояния подшипников газотурбинных двигателей

Рассматривается задача диагностики технического состояния межвальных подшипников ГТД (кондиционный, некондиционный) по оцифрованному вибросигналу, полученному прибором ИВУ-1. Для получения вибросигнала в зоне узла межвального подшипника на неработающем двигателе с помощью штанги внутрь вала турбины низкого давления (НД) устанавливается датчик, в процессе свободного вращения ротора измеряются амплитуды вибраций.

Пусть $T = 0, \pm 1, \pm 2, \ldots$ — бесконечная ось дискретного времени, $T_{t_0}^{t_0+N} = \{t \in T | t_0 \leqslant t < t_0 + N\}$ — некоторый ее фрагмент, быть может, неограниченный справа, если $N \to \infty$. Ставится задача принятия по наблюдаемой цифровой последовательности $S_{t_0}^{t_0+N}$ бинарного решения относительно исправности подшипника $S_{t_0}^{t_0+N} \to \{0, 1\}$. В некондиционных подшипниках возникают возмущения, отсутствующие в хороших: по мере появления дефектов на кинематических узлах подшипника, в вибросигнале появляются отдельные, короткие амплитудные пики, соответствующие моментам соударения дефектов; с развитием дефекта увеличиваются амплитуды пиков и их количество. По наличию и количеству неоднородностей в вибросигнале, соответствующих моментам появления дефектов, можно сформировать признак для определения состояния подшипника. Поэтому предлагается решать поставленную задачу на основе обнаружения локальных неоднородностей как изменений свойств вибросигнала подшипника.

Задача обнаружения изменения свойств случайных процессов сформулирована В. В. Моттлем [4] следующим образом. Наблюдаемый случайный процесс $S_{t_0}^{t_0+N} = (s_t, t_0 \leqslant t < t_0 + N)$ характеризуется скачкообразным изменяющимся значением параметра c:

$$c = \begin{cases} c_1, & t_0 \leqslant t < t_1; \\ c_2, & t_1 \leqslant t < t_2; \\ \ldots & \\ c_G, & t_{G-1} \leqslant t < t_G; \\ c_{G+1}, & t_G \leqslant t < t_0 + N. \end{cases}$$

Значение $G = 0$ будем интерпретировать как неизменное значение параметра в течение всего интервала наблюдения $c = c^1 = const, t_0 \leqslant t < t_0 + N$. Требуется, анализируя реализацию случайного процесса $S_{t_0}^{t_0+N}$, определить число G моментов скачкообразного изменения параметра, оценить моменты $t_1, \ldots t_G$. Скачкообразные изменения параметра c для вибросигнала подшипника будут соответствовать моментам возникновения дефектов. Некондиционным подшипникам соответствует более резкое изменение параметра c и большее количество G таких изменений, чем для кондиционных. Таким образом, по изменению параметра c можно судить о состоянии подшипника.

Обнаружение изменений любой функции распределения или какой-либо иной вероятностной характеристики может быть сведено к обнаружению изменения математического ожидания в некоторой новой случайной последовательности, сформированной из исходной (диагностическая последовательность) [5].

По реализации $S_{t_0}^{t_0+N}$ строится новая диагностическая последовательность F_0^{l-1} следующим образом.

Реализация $S_{t_0}^{t_0+N}$ представляется в виде последовательности перекрывающихся блоков длины b. Если новый блок начинается через каждые $b/2$ отсчетов, то их количество $l = [2N/b]$. Таким образом, реализация раскладывается на элементарные участки:

$$S_{t_0}^{t_0+N-1} \to [e_{t_0}^{t_0+b-1}, e_{t_0+b/2}^{(t_0+3b)/2-1}, \ldots, e_{t_0+ib/2}^{t_0+(i+2)b/2-1}, \ldots, e_{t_0+(l-1)b/2}^{t_0+N-1}]$$

Для каждого блока $e_{t_0+ib/2}^{t_0+(i+2)b/2-1}$ рассчитывается диагностический признак f_i. Получается последовательность

$$F_0^{l-1} = (f_i, 0 \leqslant i \leqslant l-1)$$

Моменты скачкообразного изменения значения диагностического признака F будут соответствовать моментам изменения параметра c в исходном процессе $S_{t_0}^{t_0+N-1}$ с точностью $b/2$. Таким образом, задача обнаружения локальных неоднородностей (изменения параметра c) в исходном процессе $S_{t_0}^{t_0+N-1}$ сводится к задаче обнаружения изменения свойств в F_0^{l-1}.

Состояние объекта (подшипника ГТД) может быть описано совокупностью определяющих его параметров (признаков). Однако не все признаки вносят одинаковый вклад в принятие решения. Важным этапом диагностики и классификации является выбор наиболее информативных признаков: система признаков должна обладать большой диагностической ценностью [6]. Использование неинформативных признаков снижает эффективность процесса диагностики.

В качестве диагностических признаков рассматриваются локальные фрактальные размерности, изменение фрактальной размерности, локальные матрицы зависимости величины приращения сигнала от самой величины сигнала, локальные векторы коэффициентов сноса, оценивается их информативность. Под локальностью понимается, что признаки рассчитываются не на всем сигнале, а на блоке сигнала.

Алгоритмы вычисления признаков на основе зависимости приращений от величины сигнала

Определение 1. *В точке t приращение сигнала — это разность между текущим и предыдущим значениями сигнала: $\delta_t = s_t - s_{t-1}$, $t = 1, \ldots, N$. Множество приращений для точек всего сигнала образуют последовательность приращений сигнала: $\Delta = \{\delta_1, \ldots, \delta_N\}$*

Множество значений всего сигнала S разбивается на n упорядоченных диапазонов, множество значений приращений разбивается на m упорядоченных диапазонов.

Разбиение множества значений сигнала:

$$\widetilde{S} = \{\widetilde{S}_1, \ldots, \widetilde{S}_n\}, \text{ где } \forall p = 1, \ldots, n-1 \; \forall s \in \widetilde{S}_p, s' \in \widetilde{S}_{p+1} \; s < s'$$

Разбиение множества приращений сигнала:

$$\widetilde{\Delta} = \{\widetilde{\Delta}_1, \ldots, \widetilde{\Delta}_m\}, \text{ где } \forall q = 1, \ldots, m-1 \; \forall \delta \in \widetilde{\Delta}_q, \delta' \in \widetilde{\Delta}_{q+1} \; \delta < \delta'$$

Таким образом, получается $n \times m$ пар диапазонов. В данной работе рассматривается разбиение множества значений и множества приращений сигнала на три равных диапазона: большие, средние и маленькие значения.

Вычисление матриц зависимости величины приращения от величины сигнала для блока сигнала

Для каждого блока вычисляется матрица H_i, содержащая количество точек, соответствующих парам диапазонов. Затем эта матрица нормируется по строкам.

Формально это можно записать следующим образом:

для $i = 0, \ldots, l-1$
  для $p = 1, \ldots, n$
    для $q = 1, \ldots, m$
      $A_{i\,pq} = \left|\{s_t \colon s_t \in \widetilde{S}_p, \delta_t \in \widetilde{\Delta}_q, t = t_0 + ib/2, \ldots, t_0 + (i+2)b/2 - 1\}\right|$
      $B_{ip} = \left|\{s_t \colon s_t \in \widetilde{S}_p, t = t_0 + ib/2, \ldots, t_0 + (i+2)b/2 - 1\}\right|$
      $H_{i\,pq} = A_{i\,pq}/B_{ip}$

Вычисление векторов коэффициентов сноса для блока сигнала

Определение 2. *Коэффициент сноса для диапазона значений сигнала — среднее значение приращения для точек блока, в которых значение сигнала соответствует рассматриваемому диапазону. Коэффициенты сноса блока для каждого диапазона значений образуют вектор коэффициентов сноса.*

Таким образом, для каждого блока рассчитывается вектор коэффициентов сноса V_i длины n.

Формально это можно записать следующим образом:

для $i = 0, \ldots, l-1$
  для $p = 1, \ldots, n$
    для $q = 1, \ldots, m$
      $A_{i\,pq} = \left|\{s_t \colon s_t \in \widetilde{S}_p, \delta_t \in \widetilde{\Delta}_q, t = t_0 + ib/2, \ldots, t_0 + (i+2)b/2 - 1\}\right|$
      $V_{i\,p} = \sum\limits_{q=1}^{q=m} \frac{A_{i\,pq}}{m}$

Алгоритм вычисления фрактальной размерности для блока сигнала

Компьютерные алгоритмы вычисления размерности Минковского [7] r обычно опираются на соотношение:

$$\log N(\varepsilon) = \log \varphi - \log \varepsilon$$

где Ф - константа, $N(\varepsilon)$ - минимальное число клеток со стороной ε, необходимых для покрытия фрактала (блока сигнала). График зависимости от $\log (N)_\varepsilon$ от $\log \varepsilon$ - прямая с угловым коэффициентом r.

Для определения неизвестных параметров φ и r необходимо оценить $N(\varepsilon)$ для нескольких значений ε. Если использовать клетки только двух размеров ε_1 и ε_2, то неизвестные φ и r можно определить из системы уравнений:

$$\log (N)_{\varepsilon_1} = \log \varphi - \log \varepsilon_1$$

$$\log (N)_{\varepsilon_2} = \log \varphi - \log \varepsilon_2$$

Тем не менее, учитывая, что величины $N(\varepsilon)$ могут быть найдены лишь приближенно, имеет смысл оценить $N(\varepsilon)$ для большого количества различных значений ε. В этом случае получится переопределенная система (число уравнений больше числа неизвестных), которая скорее всего не будет иметь точного решения. Далее определяются значения $\log\varphi$ и r, минимизирующие сумму квадратов отклонений.

Пусть $N_i(\varepsilon)$ – минимальное число клеток со стороной ε, необходимых для покрытия i-го блока, s_{max} и s_{min} – максимальное и минимальное значения в пределах всей реализации соответственно. Рассматриваются размеры клеток ε_k, $k = 0, \ldots, n$, где

$$\varepsilon_0 = (s_{max} - s_{min})/b,\ \varepsilon_k = 2^k(s_{max} - s_{min})/b,\ \varepsilon_n < b/3$$

Для каждого ε_k вычисляется $N_i(\varepsilon_k)$. Размерность i-го блока R_i оценивается как коэффициент наклона прямой, образованной графиком зависимости $\log N_i(\varepsilon)$ от $log\varepsilon$, который вычисляется по методу наименьших квадратов.

Формально алгоритм вычисления размерности можно записать следующим образом:

для $k = 0, \ldots, n$: $\varepsilon_n < b/3$
  $\varepsilon_k = 2^k(s_{max} - s_{min})/b$
  $N_i(\varepsilon_k)$=`CalcSquares(`$S_{t_0+ib/2}^{t_0+(i+2)b/2-1}$`,` ε_k`)`
R_i=`RbyMNK(`$\{N_i(\varepsilon_k)\}$`,` $\{\varepsilon_k\}$`)`

Здесь функция `CalcSquares(`$S_{t_0+ib/2}^{t_0+(i+2)b/2-1}$`,` ε_k`)` вычисляет минимальное количество клеток со стороной ε_k, необходимых для покрытия участка графика, соответствующего i-му блоку, функция `RbyMNK(`$\{N_i(\varepsilon_k)\}$`,` $\{\varepsilon_k\}$`)` вычисляет размерность блока по методу наименьших квадратов.

Алгоритм обнаружения локальных неоднородностей по фрактальной размерности

Изменение фрактальной размерности на соседних блоках $\Delta R_i = R_i - R_{(i-1)}$ сравнивается с величиной фрактальной размерности на $(i-1)$-м блоке. Считается, что обнаружена локальная неоднородность, если изменение размерности превышает $R_{(i-1)}$ более чем в α раз, где α — заданный порог:

$$\Delta R_i / R_{(i-1)} > \alpha.$$

Отдельно рассматривается как общее количество неоднородностей (Inh_{all}), так и количество неоднородностей, связанных с ростом (Inh_{up}) или уменьшением фрактальной размерности (Inh_{down}) – будем называть их *положительными* и *отрицательными*, соответственно.

$Inh_{all} = 0$
$Inh_{up} = 0$
$Inh_{down} = 0$
для $i = 0, \ldots, l$
  если $\Delta R_i / R_{(i-1)} > \alpha$ **то**
    если $R_i > R_{(i-1)}$ **то**
      $Inh_{down} = Inh_{down} + 1$
    иначе
      $Inh_{up} = Inh_{up} + 1$
    $Inh_{all} = Inh_{all} + 1$

Оцениваются информативности признаков Inh_{all}, Inh_{up}, Inh_{down} для разных размеров блока b и величины порога α. Выбирается та пара, при которых достигается максимальная информативность.

Оценка информативности признаков

Для оценки ценности предлагаемых диагностических признаков, рассчитывается их информативность I. Основной критерий информативности признакового пространства X вычисляется как отношение среднего межклассового расстояния к среднему внутриклассовому [8]:

$$I(X) = \frac{\overline{\rho_{IJ}}}{\overline{\rho_m}}$$

где ρ_m — *внутриклассовое расстояние* (среднее расстояние между объектами класса m), и ρ_{IJ} — *расстояние между классами* (среднее между расстояние между объектами разных классов). Такое определение понятия информативности может служить критерием информативности, поскольку характеризует компактность расположения объектов одного класса и удаленность объектов разных классов в признаковом пространстве. Используется евклидово расстояние, в многомерном пространстве оно равно

$$d(x_{I,k}, x_{J,l}) = \sqrt{\sum_{u=1}^{\dim X} (x_{I,k}(u) - x_{J,l}(u))^2},$$

где $\dim X$ — количество признаков, размерность пространства объектов, $d(x_{I,k}, x_{J,l})$ — евклидово расстояние между точками признакового пространства, которые соответствуют объектам k и l классов I и J, $x_{I,k}(u)$ — значение u-го признака k-го объекта класса I.

Внутриклассовое Евклидово расстояние :

$$\rho_m = \frac{2 \sum\limits_{1 \leqslant k < l \leqslant K_m} d(x_{m,k}, x_{m,l})}{K_m(K_m - 1)},$$

где K_m — количество объектов в классе m.

Евклидово расстояние между классами:

$$\rho_{I,J} = \frac{\sum\limits_{k=1}^{K_I} \sum\limits_{l=1}^{K_J} d(x_{I,k} - x_{J,l})}{K_I \cdot K_J}.$$

Отношение среднего межклассового Евклидова расстояния в признаковом пространстве к среднему внутриклассовому расстоянию в признаковом пространстве X:

$$I(X) = \frac{2 \sum\limits_{1 \leqslant I < J \leqslant M} \rho_{I,J}}{(M-1) \sum\limits_{m=1}^{M} \rho_m}.$$

Эксперимент

Для проведения эксперимента используется выборка вибросигналов подшипников трансмиссии ГТД, предоставленная «НПО «Сатурн» [1]. В нее входят 6 оцифрованных

вибросигналов подшипников: два кондиционных, два некондиционных и два необоснованно снятых. Размер выборки небольшой, но была необходимость получить результаты именно на этих данных. Частота дискретизации сигналов 10 кГц, длина реализаций 30000 отсчетов. Выборка может быть разбита на три класса: B — неисправные, C — кондиционные, (правильно распознанные с помощью прибора ИВУ-1) и N — необоснованно снятые (кондиционные, но ошибочно распознанные как плохие подшипники прибором ИВУ-1). Для решения задачи диагностики состояния подшипников можно рассматривать три класса (C, B, N) или два класса $(C \cup N, B)$. В данной работе рассматриваются оба случая.

Каждый вибросигнал разбивается на блоки длины b. Для каждого блока рассчитываются характеристики:

— матрица зависимости приращения от величины сигнала;
— вектор коэффициентов сноса;
— фрактальная размерность.

Для соседних блоков оцениваются:

— евклидово расстояние между матрицами зависимости приращения от величины сигнала;
— евклидово расстояние между векторами коэффициентов сноса;
— изменение фрактальной размерности;
— наличие локальной неоднородности по фрактальной размерности и определение отрицательная она или положительная.

Для всего сигнала оцениваются следующие диагностические признаки:

— среднее евклидово расстояние между матрицами зависимости приращения от величины сигнала;
— среднее евклидово расстояние между векторами коэффициентов сноса;
— средняя фрактальная размерность;
— среднее изменение фрактальной размерности;
— общее количество неоднородностей;
— количество положительный неоднородностей;
— количество отрицательных неоднородностей.

Параметры размер блока b и порог для определения неоднородности α используются для вычисления признаков:

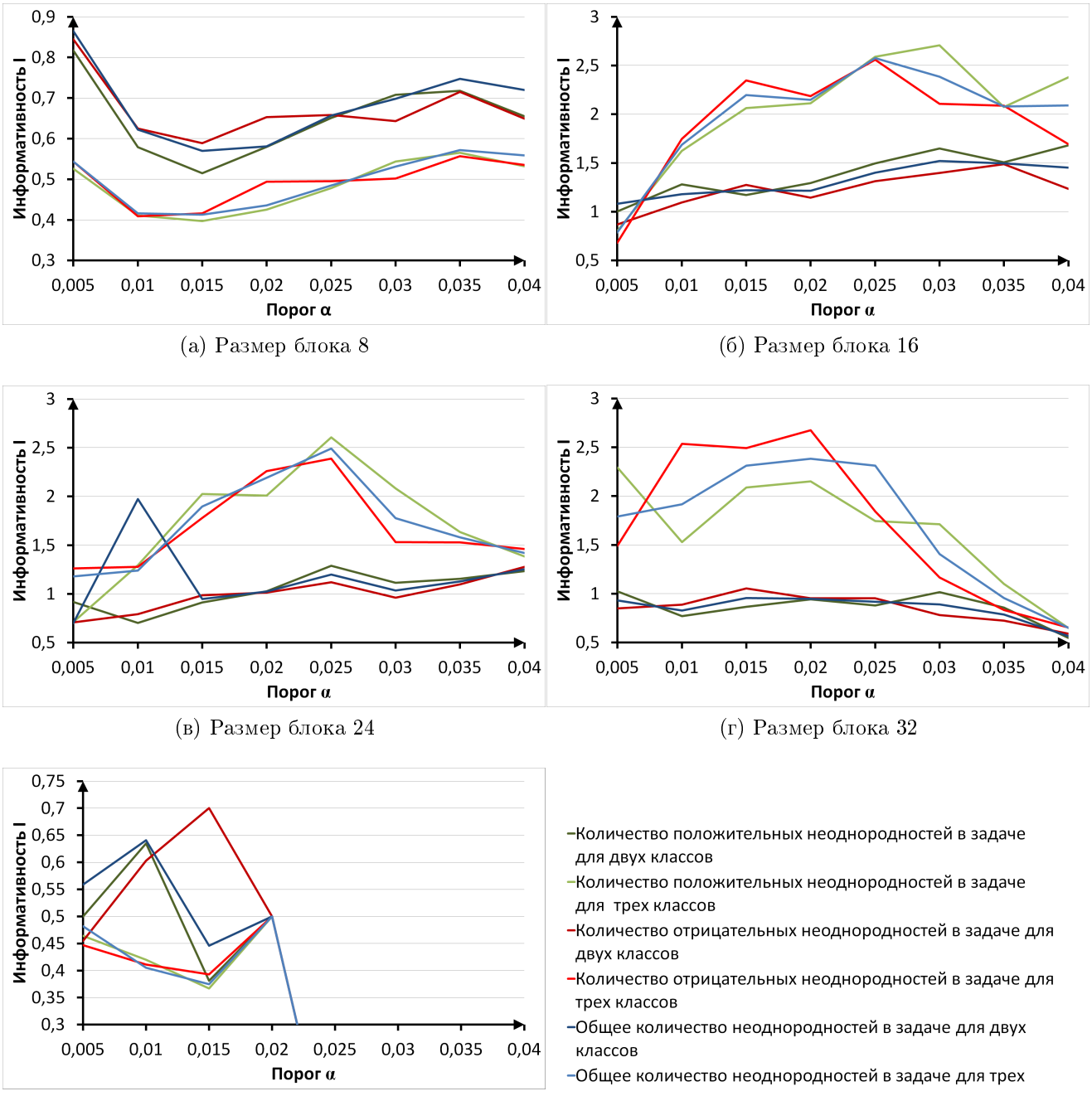
— общее количество неоднородностей;
— количество положительный неоднородностей;
— количество отрицательных неоднородностей.

Проводятся эксперименты с разными значениями b и α, рассчитывается информативность каждого из получаемых признаков в отдельности. Пара (b, α) определяется по результатам, имеющим наибольшую информативность.

На рис. 1 представлены графики зависимости информативности признака от величины порога при разных размерах блока. Видно, что при очень маленьких и слишком больших размерах блока информативность падает, информативность признаков при решении задачи на трех классах выше, чем при решении на двух классах. Наибольшая информативность достигается использовании признака «Количество положительных неоднородностей» при параметрах $b = 16, \alpha = 0,03$ при решении задачи на трех классах.

Только от порога b зависит вычисление признаков:

— среднее евклидово расстояние между матрицами зависимости приращения от величины сигнала;



(а) Размер блока 8

(б) Размер блока 16

(в) Размер блока 24

(г) Размер блока 32

(д) Размер блока 64

Рис. 1: Графики зависимости информативности признаков от величины порога

— среднее евклидово расстояние между векторами коэффициентов сноса;
— средняя фрактальная размерность;
— среднее изменение фрактальной размерности;

Графики зависимости информативности признака от величины порога при разных размерах блока (рис. 2) отражают, что наибольшая информативность достигается при использовании признака «Расстояние между векторами коэффициентов сноса» при $b = 24$ при решении задачи на трех классах $I = 2,267$.

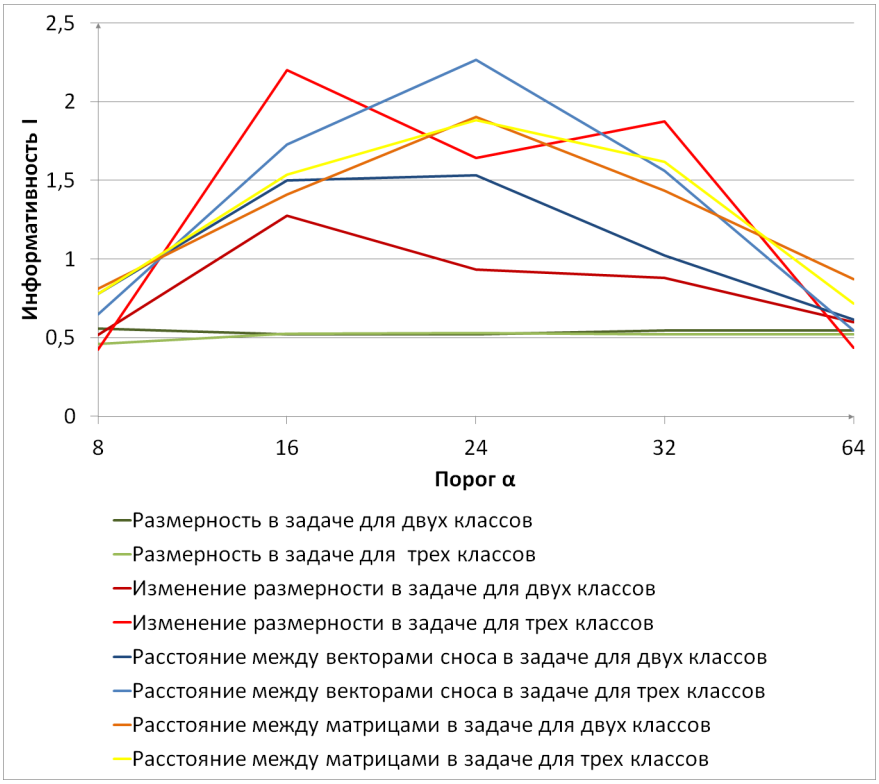


Рис. 2: График информативности в зависимости от размера блока b

При решении задачи на трех классах, признаки имеют большую информативность. С увеличением размера блока b изменения сглаживаются, их сложнее обнаружить, уменьшение размера блока b вызывает обнаружение большого количества ложных неоднородностей, что подтверждается информативностью признаков.

Можно выделить следующие признаки, подходящие для решения данной задачи:

1) количество «отрицательных» неоднородностей;
2) общее количество неоднородностей;
3) количество «положительных» неоднородностей;
4) расстояние между векторами коэффициентов сноса;
5) расстояние между матрицами зависимости приращения от величины сигнала.

Наибольшая информативность $I = 2,708$ достигается при использовании признака «Количество «положительных» неоднородностей» при размере блока $b = 16$ и пороге $\alpha = = 0,03$.

Информативность признаков, получаемых с помощью виброизмерительных приборов ИВУ-1 и MIC-200 приведены в табл. 1.

Таблица 1: Информативность диагностических признаков систем ИВУ-1 и MIC-200

Используемый виброприбор, признак	Информативность	
	2 класса	3 класса
ИВУ-1 (СКО)	0.692	0.572
MIC-200 (спектральные признаки)	0.519	0.733

Заключение

В работе предложены алгоритмы вычисления диагностических признаков для диагностики состояния подшипников ГТД по вибросигналу на основе обнаружения в нем локальных неоднородностей: фрактальная размерность сигнала, матрицы зависимости величины приращения от величины сигнала, векторы коэффициентов сноса. Вычислена их информативность, определены наиболее информативные признаки — количество «положительных», «отрицательных», общее количество неоднородностей, расстояние между векторами коэффициентов сноса, расстояние между матрицами зависимости приращения от величины сигнала. Предложен метод настройки алгоритмов вычисления диагностических признаков, найдены оптимальные параметры на обучающей выборке. Информативность предлагаемых признаков в ряде случаев выше, чем информативность СКО, полученное по сигналу с аналогового выхода ИВУ-1, и спектральных признаков, определяемые с помощью комплекса MIC-200 (ВДК-44), следовательно, целесообразно работать с предлагаемыми признаками. Помимо задачи диагностики их можно применять для решения задач обнаружения локальных неоднородностей в сигналах.

Литература

[1] *Шепель В. Т., Комаров Б. И., Грызлова Т. П.* Выбор признаков для диагностики технического состояния трансмиссионных подшипников ГТД // Авиационно-космическая техника и технология. 2005. № 8(24). С.97–100.

[2] *Облеухов А. А., Шалаев Д. С., Грызлова Т. П.* Анализ информативности Wavelet-представления вибросигналов в задаче диагностики состояния подшипников // Двигатели и энергоустановки аэрокосмических летательных аппаратов. 2011. № 10(87). С. 133–138.

[3] *Горшков А. П., Грызлова Т. П.* Система диагностики состояния сложных технических объектов по характерным последовательностям цифровых сигналов // Информационные технологии. 2008. № 9. С. 35–38.

[4] *Моттль В. В., Мучник И. Б.* Скрытые марковские модели в структурном анализе сигналов. М.: Физматлит, 1999. 352 с.

[5] *Бродский Б. Е., Дарховский Б. С., Каплан А. Я., Шишкин С. Л.* Непараметрическая сегментация электрических сигналов мозга // Автоматика и телемеханика. 1998. № 2. С. 23–32.

[6] *Биргер И. А.* Техническая диагностика. М.: Машиностроение, 1978. 240 с.

[7] *Кроновер Р. М.* Фракталы и хаос в динамических системах. Основы теории. М.: Постмаркет, 2000. 352 с.

[8] *Грызлова Т. П., Балыкина А. С.* Система оценки информативности диагностических признаков и признаковых пространств // Авиационно-космическая техника и технология. 2011. № 9(86). С. 148–154.

Об алгебро-логической коррекции в задачах распознавания по прецедентам*

Дюкова Е. В.[1]***, Любимцева М. М.***[2]***, Прокофьев П. А.***[3]
[1]edjukova@mail.ru; [2]m.lyubimtseva@gmail.com; [3]p_prok@mail.ru
[1,3]ВЦ РАН; [2]ВМК МГУ

Исследуются логические корректоры — модели распознающих алгоритмов, основанные на голосовании по корректным наборам элементарных классификаторов (эл.кл.). Вводится понятие *антимонотонного* корректного набора эл.кл. На базе антимонотонных корректных наборов эл.кл. построен логический корректор. Приведены результаты тестирования новой модели логического корректора на реальных даннных.

Ключевые слова: *распознающий алгоритм, логическая процедура распознавания, алгебро-логическая коррекция, элементарный классификатор, корректный элементарный классификатор, логический корректор, (монотонный) корректный набор элементарных классификаторов.*

Algebraic-logical correction in recognition problems*

Djukova E. V.[1]***, Lyubimtseva M. M.***[2]***, Prokofjev P. A.***[3]
[1,3]Dorodnitsyn Computing Centre, Russian Academy of Sciences; [2]Faculty of Computational Mathematics and Cybernatics, Moscow State University

A problem of constructing the correct recognition algorithms based on incorrect elementary classifiers (EC) is considered. A new type of correct sets of EC is suggested that is called antimonotonic. A new model of correct recognition algorithms based on antimonotonic correct sets of EC is constructed. This model is tested on real tasks.

Keywords: *recognition algorithm, logical recognition procedure, algebraic-logical correction, elementary classifier, correct elementary classifier, logical corrector, (monotonic) correct set of elementary classifiers.*

Введение

Рассматривается подход к задаче распознавания по прецедентам, базирующийся на применении аппарата дискретной математики (логических и алгебро-логических методов анализа данных). Важнейшими для этого направления исследований являются вопросы построения корректных распознающих алгоритмов, т. е. алгоритмов, правильно классифицирующих обучающие объекты.

Классические логические алгоритмы распознавания основаны на поиске в целочисленных данных конъюнктивных закономерностей или элементарных классификаторов (эл.кл.). В решающем правиле используется процедура голосования по найденным эл.кл. В случае целочисленной информации в роли эл.кл. выступают элементарные конъюнкции, определенные на наборах, являющихся признаковыми описаниями объектов. Элементарная конъюнкция считается корректной, если ее интервал истинности не содержит описаний обучающих объектов из разных классов. Корректность распознающего алгоритма обеспечивается корректностью используемых эл.кл. Основной задачей логического

Работа частично поддержана грантом РФФИ № 13-01-00787-а и грантом президента РФ НШ-4652.2012.1.

анализа данных в распознавании является поиск наиболее информативных корректных эл.кл.

Одним из новых направлений исследований в рассматриваемой области является синтез корректных логических распознающих процедур с использованием алгебраического подхода [1]. В данном случае в качестве базисных распознающих алгоритмов выступают эл.кл., не обязательно являющиеся корректными, а в качестве корректирующей функции берется булева функция. Голосование ведется по найденным корректным наборам эл.кл. Основной задачей алгебро-логической коррекции является поиск корректных наборов эл.кл. с хорошей распознающей способностью.

Идея алгебро-логического синтеза корректных процедур распознавания (логических корректоров) предложена в [2]. В указанной работе показано, что задача нахождения корректных наборов элементарных классификаторов равносильна задаче нахождения покрытий булевой матрицы, специальным образом построенной по обучающей выборке.

Подход развит в работах [3] и [4], в которых рассмотрены вопросы практического применения различных моделей логических корректоров.

В [4] на базе простейших эл.кл., порождаемых элементарными конъюнкциями ранга 1, построен алгоритм МОН. В качестве корректирующей булевой функции используется монотонная булевая функция. Для сокращения перебора при поиске корректных наборов эл.кл. с хорошей распознающей способностью использован генетический алгоритм из [5].

В [3] построен алгоритм LOBAGA, использующий произвольные эл.кл. в качестве базисных операторов. На этапе обучения алгоритм работает итеративно. На каждой итерации строится достаточно большой корректный набор эл.кл. с хорошей распознающей способностью, который называется локальным базисом. В рамках построенного локального базиса генетическим алгоритмом из [5] ищутся корректные наборы эл.кл. Каждому корректному набору эл.кл. приписывается вес, характеризующий его распознающую способность. Найденные корректные наборы эл.кл. пополняют семейство наборов эл.кл., построенное на предыдущих итерациях. Итерация заканчивается классификацией обучающих объектов путем голосования по всем найденным к данному моменту корректным наборам эл.кл. В результате происходит обновление веса каждого обучающего объекта, характеризующего типичность объекта для своего класса. Число итераций является параметром алгоритма.

В данной работе введено понятие антимонотонного корректного набора эл.кл. По аналогии с логическим корректором МОН из [4], использующим монотонные корректные наборы эл.кл., построен логический корректор АМОН, использующий антимонотонные корректные наборы эл.кл. Приведены результаты тестирования алгоритмов МОН, АМОН и LOBAGA на реальных данных. Тестируемые логические корректоры сравнивались с рядом распознающих алгоритмов из системы [6].

Основные понятия и обозначения

Рассматривается задача распознавания по прецедентам с непересекающимися классами $K_1, ..., K_l$, системой признаков $\{x_1, \ldots, x_n\}$ и набором обучающих объектов $\{S_1, \ldots, S_m\}$.

Пусть объект S имеет описание $(s_1, \ldots, s_n)$ в системе признаков $\{x_1, \ldots, x_n\}$ и пусть H — набор из r различных признаков $\{x_{j_1}, \ldots, x_{j_r}\}$. Тогда вектор $(s_{j_1}, \ldots, s_{j_r})$ называется подописанием объекта S по набору признаков H и обозначается (S, H).

Пусть далее $\sigma = (\sigma_1, \ldots, \sigma_r)$ — набор, в котором σ_q — допустимое значение признака x_{j_q} из H. Пара (H, σ) называется *элементарным классификатором* (эл. кл.). Число r называется *рангом эл. кл.* (H, σ).

Близость объекта S и эл.кл. (H, σ) оценивается величиной

$$B_{(H,\sigma)}(S) = \begin{cases} 1, \text{ если } (S, H) = \sigma, \\ 0, \text{ в противном случае.} \end{cases}$$

Пусть $U = \{(H^1, \sigma^1), \ldots, (H^d, \sigma^d)\}$ — набор эл.кл. Для объекта S через $\omega_U(S)$ обозначается вектор

$$(B_{(H^1,\sigma^1)}(S), \ldots, B_{(H^d,\sigma^d)}(S)),$$

называемый откликом набора эл.кл. U на объекте S.

Пусть T — подмножество множества обучающих объектов (*подвыборка*). Набор эл.кл. U называется *корректным для класса* K *(*$K \in \{K_1, \ldots, K_l\}$*) относительно подвыборки* T, если существует булева функция $F_K(t_1, \ldots, t_d)$, такая что для любой пары обучающих объектов $S_i \in T \cap K$ и $S_j \in T \setminus K$ выполняется одно из неравенств

$$F_K(\omega_U(S_i)) > F_K(\omega_U(S_j), \tag{1}$$
$$F_K(\omega_U(S_i)) < F_K(\omega_U(S_j). \tag{2}$$

Корректный для K набор U называется *монотонным*, если существует монотонная функция F_K, для которой всегда выполняется (1). Корректный для K набор U называется *антимонотонным*, если существует монотонная функция F_K, для которой всегда выполняется (2).

Алгоритмы МОН, АМОН и LOBAGA

На этапе обучения алгоритмов МОН, АМОН и LOBAGA обучающие объекты делятся на базовую подвыборку T_0 и настроечную подвыборку T_1. Базовая выборка используется для построения корректных набор эл.кл., настроечная выборка — для оценки распознающей способности наборов. Для каждого класса $K \in \{K_1, \ldots, K_l\}$ формируется семейство W_K, состоящее из корректных для K относительно T_0 наборов эл.кл. с хорошей распознающей способностью.

Алгоритм МОН. Алгоритм при обучении строит семейство W_K из монотонных корректных для K наборов эл.кл.

Определяется близость объекта S к обучающему объекту S_i по набору эл.кл. $U = \{(H^1, \sigma^1), \ldots, (H^d, \sigma^d)\}$

$$\delta_U(S, S_i) = \begin{cases} 1, & B_{(H_j,\sigma_j)}(S) \geqslant B_{(H_j,\sigma_j)}(S_i),\ j \in \{1, \ldots, d\}, \\ 0, & \text{иначе.} \end{cases}$$

Значение функционала

$$\Gamma'_K(U, S) = \frac{1}{|K \cap T_0|} \sum_{S_i \in K \cap T_0} \delta_U(S, S_i)$$

характеризует насколько объект S близок к классу K.

Распознающая способность корректного для K набора эл.кл. U оценивается величиной

$$\tau'_K(U) = \frac{1}{|T_1 \cap K|} \sum_{S_i \in T_1 \cap K} \Gamma'_K(U, S_i) - \frac{1}{|T_1 \setminus K|} \sum_{S' \in T_1 \setminus K} \Gamma'_K(U, S_i).$$

Пусть $\mathcal{U}^1$ — множество всех эл.кл. ранга 1, порождаемых обучающими объектами, и пусть L'_K — булева матрица, строящаяся по следующему правилу. Каждой строке матрицы L'_K соответствует пара обучающих объектов (S_i, S_j) из T_0, $S_i \in K$, $S_j \notin K$, каждому столбцу — эл.кл. (H, σ) из $\mathcal{U}^1$. Элемент матрицы L'_K, расположенный на пересечении строки (S_i, S_j) и столбца (H, σ), равен

$$B_{(H,\sigma)}(S_i) \wedge \neg B_{(H,\sigma)}(S_j).$$

Через $J(U)$ обозначим набор столбцов матрицы L'_K, порожденный эл.кл. из U. Набору столбцов $J(U)$ матрицы L'_K приписывается вес $\tau'_K(U)$.

Набор столбцов $J(U)$ называется покрытием булевой матрицы L, если в подматрице, составленной из столбцов $J(U)$ нет нулевой строки.

Утверждение 1 ([2]). *Набор эл.кл. $U \subseteq \mathcal{U}^1$ является монотонным корректным для класса K тогда и только тогда, когда набор столбцов $J(U)$ является покрытием матрицы L'_K.*

Таким образом, задача построения корректных для K наборов эл.кл. сводится в задаче построения покрытий матрицы L'_K с весами, близкими к максимальным. Для этого используется генетический алгоритм из [5]. Необходимое для формирования W_K число покрытий является параметром обучения.

Результатом обучения алгоритма МОН является совокупность построенных семейств наборов эл.кл. $W_{K_1}, \ldots, W_{K_l}$.

При распознавании объект S относится к классу с максимальным значением функционала

$$\Gamma'(W_K, S) = \frac{1}{|W_K|} \sum_{U \in W_K} \Gamma'_K(U, S),$$

В случае существования нескольких классов, для которых значение $\Gamma'(W_K, S)$ максимально, происходит отказ от распознавания.

Алгоритм АМОН. Алгоритм при обучении строит W_K из антимонотонных корректных для K наборов эл.кл.

Для оценки близости объекта S к классу K вместо $\Gamma'_K(U, S)$ используется функционал

$$\Gamma''_K(U, S) = \frac{1}{|K \setminus T_0|} \sum_{S_i \in K \setminus T_0} (1 - \delta_U(S, S_i)).$$

Функционал для оценки распознающей способности набора U и оценка за отнесение объекта S в классу K по семейству W_K строятся аналогично $\tau'_K(U)$ и $\Gamma'(W_K, S)$, и соответственно имеют вид

$$\tau''_K(U) = \frac{1}{|T_1 \cap K|} \sum_{S_i \in T_1 \cap K} \Gamma''_K(U, S_i) - \frac{1}{|T_1 \setminus K|} \sum_{S' \in T_1 \setminus K} \Gamma''_K(U, S_i),$$
$$\Gamma''(W_K, S) = \frac{1}{|W_K|} \sum_{U \in W_K} \Gamma''_K(U, S).$$

Для поиска антимонотонных корректных для K наборов строится булева матрица L''_K. Каждой строке L''_K соответствует пара обучающих объектов (S_i, S_j) из T_0, таких что $S_i \in$

K, $S_j \notin K$, каждому столбцу — эл.кл. $(H, \sigma) \in \mathcal{U}^1$. Элемент матрицы L''_K, находящийся в пересечении строки (S_i, S_j) и столбца (H, σ), равен

$$\neg B_{(H,\sigma)}(S_i) \wedge B_{(H,\sigma)}(S_j).$$

Утверждение 2. *Набор эл.кл. $U \subseteq \mathcal{U}^1$ является антимонотонным корректным для класса K тогда и только тогда, когда набор столбцов $J(U)$ является покрытием матрицы L''_K.*

Покрытию $J(U)$ матрицы L''_K приписывается вес $\tau''_K(U)$. Для поиска покрытий с весом, близким к максимальному, также используется генетический алгоритм [5].

Алгоритм LOBAGA. Алгоритм подробно описан в [3]. Строящиеся корректные наборы в отличие от алгоритмов МОН и АМОН состоят из эл.кл. произвольного ранга.

При инициализации берутся пустые семейства $W_{K_1}, \ldots, W_{K_l}$ и каждому обучающему объекту приписывается вес, равный $1/m$. В ходе последующих итераций веса объектов пересчитываются, и результирующий вес обучающего объекта характеризует типичность объекта для класса, к которому он принадлежит.

На каждой итерации специальным образом выбирается класс K, и формируется локальный базис $\mathcal{U}_K$ — достаточно большой корректный для K набор эл.кл. с хорошей распознающей способностью. В рамках локального базиса $\mathcal{U}_K$ генетическим алгоритмом ищется семество корректных наборов эл.кл. с распознающей способностью, близкой к максимальной. При вычислении распознающей способности корректных наборов эл.кл. учитываются веса объектов. Каждый найденный корректный набор эл.кл. U добавляется с весом α_U в семейство W_K. Вес α_U характеризует распознающую способность U. По найденным к настоящему моменту корректным наборам эл.кл. вычисляются оценки за отнесение каждого обучающего объекта к своему классу. На основании оценок пересчитываются веса объектов. Число итераций является параметром обучения.

Тестирование алгоритмов МОН, АМОН и LOBAGA

Алгоритмы МОН, АМОН и LOBAGA протестированы на 24 реальных задачах из области медицины, собранных в отделе Математических проблем распознавания и методов комбинаторного анализа ВЦ РАН [6], а также представленных в репозитории UCI. Характеристики задач представлены в табл. 1.

Тестирование проводилось по методу 10-fold cross-validation. В табл. 2 для каждого алгоритма указана величина

$$R = \frac{1}{10} \sum_{t=1}^{10} q_t,$$

где q_t — процент правильно распознанных объектов при тестировании с номером t.

Пусть для некоторых распознающих алгоритмов A_1 и A_2 на тестовой задаче Z получены соответственно результаты R_1 и R_2. Определим, что алгоритм A_1 на тестовой задаче Z показал лучшие результаты, чем алгоритм A_2, если $R_1 - R_2 \geqslant 3\%$. Будем говорить, что алгоритмы A_1 и A_2 на тестовой задаче Z показали сравнимые результаты, если $|R_1 - R_2| < 3\%$.

При тестировании МОН, АМОН и LOBAGA на задачах с действительнозначными данными (задачи 1–6, 15–24) предварительно осуществлялась корректная перекодировка методом из [7].

Проведено сравнение результатов тестирования алгоритмов МОН и АМОН на целочисленных задачах с представленными в [8] результатами тестирования других алгоритмов. Результаты счета для МОН приведены в табл. 3, а для АМОН — в таблице 4. В

Таблица 1: Характеристики задач

№	Задача	m	n	l	Объектов в классах	Тип данных
1	botwinimmuno	60	69	2	48/12	real
2	botwinSt	196	17	2	23/173	real
3	dorovski	33	12	2	16/17	real
4	ech_r	71	8	2	48/23	real
5	eco_l	144	7	4	76/33/24/11	real
6	Hepatit	155	19	2	32/123	real
7	input	344	9	2	218/126	int
8	manelis1	145	35	2	38/107	int
9	manelis2	107	35	2	35/72	int
10	manelis3	73	35	2	38/35	int
11	manelis4	110	35	2	38/72	int
12	matchak2	132	24	2	30/102	int
13	matchak3	269	24	2	51/218	int
14	matchak4	269	21	2	51/218	int
15	oil	114	5	3	60/15/39	real
16	patomorfoz	77	7	2	47/30	real
17	SARComa	80	18	2	40/40	real
18	sigapur	58	15	2	11/47	real
19	surv	77	8	2	52/25	real
20	botwinklbl	196	9	2	23/173	real
21	echu	131	9	2	89/42	real
22	heartUni	270	13	2	120/150	real
23	stupenexper	61	18	2	39/22	real
24	wineUni	178	13	3	59/71/48	real

табл. 5 указано число задач, на которых алгоритмы МОН и АМОН справились лучше (хуже/сравнимо), чем другие алгоритмы.

Значения $1,0$ или -1 в табл. 4 (табл. 3) указывают на то, что на задаче алгоритм МОН (АМОН) работает соответственно лучше, сравнимо или хуже по сравнению с другим указанным алгоритмом.

Для обозначения алгоритмов в табл. 3, 4 и 5 приняты следующие сокращения:

1) ЛДФ – линейный дискриминант Фишера,
2) АВО – алгоритм вычисления оценок,
3) ГМ – алгоритм построения решающих деревьев генетическим метод,
4) НС – нейронная сеть,
5) МБРД – метод бинарного решающего дерева,
6) МБС – метод k ближайших соседей,
7) ГТТ – стохастический алгоритм голосования по тупиковым тестам,
8) C5.0 – алгоритм построения решающих деревьев,
9) C5.0 boost – C5.0 с использованием бустинга,
10) C5.0 prun – C5.0 с использованием отсечения,
11) AGI.Bias, AGI.La.sum – алгоритмы из [8],
12) LAD Tree – бустинг над решающими деревьями,
13) Random Forest – баггинг над решающими деревьями,

Таблица 2: Результаты работы МОН, АМОН и LOBAGA

Задача	**МОН**	**АМОН**	**LOBAGA**
botwinimmuno	65,63%	68,75%	**69,32%**
botwinSt	**48,72%**	45,24%	47,54%
dorovski	46,32%	**58,64%**	57,32%
ech_r	56,88%	54,89%	**57,21%**
eco_l	46,35%	**47,14%**	46,74%
Hepatit	66,53%	72,43%	**73,89%**
input	98,06%	**98,46%**	98.03%
manelis1	75,59%	75,50%	**76,45%**
manelis2	61,79%	64,60%	**66,39%**
manelis3	75,41%	**81,02%**	80,44%
manelis4	75,88%	79,90%	**80,35%**
matchak2	63,73%	62,06%	**64,05%**
matchak3	**57,62%**	55,24%	56,75%
matchak4	54,39%	55,47%	**55,98%**
oil	60,60%	51,28%	**62,46%**
patomorfoz	70,89%	**76,35%**	75,40%
SARComa	47,50%	**50,00%**	**50,00%**
sigapur	50,00%	50,00%	**50,03%**
surv	47,58%	56,38%	**57,67%**
botwinklbl	50,00%	50,00%	**50,24%**
echu	75,96%	74,77%	**78,65%**
heartUni	75,42%	74,58%	**75,76%**
stupenexper	58,22%	**64,04%**	62,09%
wineUni	57,97%	59,53%	**60,48%**

14) Simple CART – аналог алгоритма CART построения решающих деревьев,

15) Simple CART prun – Simple CART с использованием отсечения,

16) J48 – аналог C4.5.

Тестирование алгоритмов МОН, АМОН и LOBAGA показало:

1) результаты АМОН на трети тестовых задач лучше, чем результаты МОН, и почти на всех остальных тестовых задачах результаты МОН и АМОН сравнимы;

2) алгоритм LOBAGA, благодаря сложным конструктивным особенностям, на всех задачах либо сравним с дающим лучшие результаты алгоритмом, либо работает лучше МОН и АМОН. При этом строящиеся корректоры LOBAGA состоят из наборов со значительно меньшим числом эл.кл. по сравнению со строящимися корректорами МОН и АМОН.

Так как на одних задачах хорошие результаты дает корректор МОН, а на других — корректор АМОН, и корректор LOBAGA позволяет с меньшим числом эл.кл. достичь сравнимых результатов, то по-видимому, целесообразно модифицировать схему построения корректора LOBAGA таким образом, чтобы в семейства корректных наборов эл.кл. входили как монотонные наборы, так и антимонотонные наборы.

Сравнение алгоритмов МОН и АМОН с другими алгоритмами на целочисленных задачах показало:

Таблица 3: Сравнение МОН с другими алгоритмами

Алгоритм \№ задачи	7	8	9	10	11	12	13	14
ЛДФ	0	1	-1	1	-1	1	1	0
ABO	0	1	-1	0	-1	1	1	1
AGI.bias	0	-1	0	-1	0	1	0	0
ГМ	0	0	-1	0	-1	1	1	1
НС	0	0	0	-1	-1	0	1	-1
МБРД	0	1	1	1	0	1	1	1
МБС	0	0	-1	1	0	1	0	-1
ГТТ	1	1	0	1	0	-1	-1	-1
C5.0	0	-1	1	1	0	1	1	0
AGI.La.sum	1	-1	0	-1	0	1	1	0
LAD Tree	1	0	-1	-1	0	1	1	1
Random Forest	0	0	-1	-1	-1	1	1	0
Simple CART	1	1	-1	1	-1	1	1	0
C5.0 boost	0	-1	0	0	0	1	1	0
C5.0 prun	0	-1	1	1	0	1	1	0
Simple CART prun	0	0	1	1	-1	1	1	1
J48	0	-1	1	1	0	1	1	1

Таблица 4: Сравнение АМОН с другими алгоритмами

Алгоритм \№ задачи	7	8	9	10	11	12	13	14
ЛДФ	0	1	0	1	-1	0	0	0
ABO	0	1	-1	1	-1	1	1	1
AGI.bias	0	-1	1	0	0	1	0	0
ГМ	0	0	0	1	0	1	1	1
НС	0	0	0	0	-1	0	0	-1
МБРД	1	1	1	1	0	1	1	1
МБС	0	0	-1	1	1	1	0	-1
ГТТ	1	1	0	1	1	-1	-1	-1
C5.0	1	-1	1	1	0	0	1	1
AGI.La.sum	1	-1	1	0	0	0	1	1
LAD Tree	1	0	-1	0	0	1	1	1
Random Forest	1	0	-1	0	0	1	0	1
Simple CART	1	1	-1	1	0	1	1	1
C5.0 boost	0	-1	1	1	0	0	0	0
C5.0 prun	1	-1	1	1	0	0	0	0
Simple CART prun	1	0	1	1	0	1	1	1
J48	0	-1	1	1	1	0	1	1

1) алгоритм МОН на большинстве задач работает хуже, чем НС, показал результаты, сравнимые с МБС и ГТТ, и относительно остальных алгоритмов отработал лучше на большинстве задач;

Таблица 5: Сравнение МОН и АМОН с другими алгоритмами в совокупности

	МОН			АМОН				МОН			АМОН		
	+1	-1	0	+1	-1	0		+1	-1	0	+1	-1	
ЛДФ	4	2	2	2	1	5	AGI.La.sum	3	2	3	4	1	3
АВО	4	2	2	5	2	1	LAD Tree	4	2	2	4	1	3
AGI.bias	1	2	5	2	1	5	Random Forest	2	3	3	3	1	4
ГМ	3	2	3	4	0	4	Simple CART	5	2	1	6	1	1
НС	1	3	4	0	2	6	C5.0 boost	2	1	5	2	1	5
МБРД	6	0	2	7	0	1	C5.0 prun	4	1	3	3	1	4
МБС	2	2	4	3	2	3	Simple CART prun	5	1	2	6	0	2
ГТТ	3	3	2	4	3	1	J48	5	1	2	5	1	2
C5.0	4	1	3	5	1	2							

2) алгоритм АМОН на большинстве задач отработал хуже, чем НС, и в сравнении с остальными алгоритмами отработал лучше на большинстве задач.

Заключение

В статье рассмотрен алгебро-логический подход к построению корректных распознающих алгоритмов, основанных на голосовании по корректным наборам эл.кл. Введено новое понятие *антимонотонного* корректного набора эл.кл., и построен новый логический корректор АМОН. Проведено тестирование логического корректора АМОН, а также предложенных ранее логических корректоров МОН и LOBAGA, на реальных данных. Лучшие результаты дает логический корректор LOBAGA. На целочисленных задачах логические корректоры МОН и АМОН, как правило, работают лучше других тестируемых в работе алгоритмов.

Литература

[1] *Журавлев Ю. И* Об алгебраическом-подходе к решению задач распознавания или классификации // Пробл. кибер. М: Наука, 1978. Вып. 33. С. 5–68.

[2] *Дюкова Е. В., Журавлев Ю. И., Рудаков К. В.* Об алгебро-логическом синтезе корректных процедур распознавания на базе элементарных алгоритмов // ЖВМ и МФ. 1996. Т. 36.

[3] *Dyukova E. V., Prokofjev P. A.* Models of recognition procedures with logical correctors // Pattern Recognition and Image Analysis. 2013. Vol. 23, No. 2. Pp. 235–244.

[4] *Dyukova E. V., Zhuravlev Yu. I., Sotnezov M. R.* Construction of an ensemble of logical correctors on the basis of elementary classifiers // Pattern Recognition and Image Analysis. 2011. Vol. 21, No. 4. Pp. 599–605.

[5] *Sotnezov R. M.* Genetic algorithms for problems of logical data analysis in discrete optimization and image recognition // Pattern Recognition and Image Analysis. 2009. Vol. 19, No. 3. Pp. 469–477.

[6] *Журавлев Ю. И., Рязанов В. В., Сенько О. В.* «Распознавание» Математические методы. Программная Система. Практические применения М.: Фазис, 2006. С. 176.

[7] *Дюкова Е. В., Сизов А. В., Сотнезов Р. М.* О корректном понижении значности данных в задачах распознавания // Доклады Всероссийской конференции «Математические методы распознавания образов» (ММРО-15). Петрозаводск, 11–17 сентября 2011 г. С. 80–83.

[8] *Genrikhov I. E.* Synthesis and analysis of recognizing procedures on the basis of full decision trees // Pattern Recognition and Image Analysis. 2011. Vol. 21, No. 1. Pp. 45–51.

Частичный синтаксический разбор текста на русском языке с помощью условных случайных полей

Кудинов М. С.
mikhailkudinov@gmail.com
Москва,
Вычислительный Центр им. А. А. Дородницына РАН.
Управление высокопроизводительных алгоритмов Исследовательского центра «Самсунг»

В статье изложен подход к поиску синтаксически связанных групп соседних слов (chunks) в русском тексте. Продемонстрирована принципиальная возможность и корректность постановки задачи выделения таких групп применительно к языку со свободным порядком слов. С использованием аппарата условных случайных полей определенный класс подобных групп можно выделить с F_1 мерой не менее 0.94. При этом обучающая выборка может быть получена путем обработки исходного текста синтаксическим анализатором без последующей ручной коррекции результатов. Тем не менее выделение достаточно длинных фрагментов текста оказывается затруднительным, а показатель F_1 меры, полученный в эксперименте, достаточно низким.

Ключевые слова: *графические вероятностные модели, поверхностный синтаксический разбор, обработка естественного языка.*

Shallow Parsing of Russian Text with Conditional Random Fields

Kudinov M. S.
Institution of Russian Academy of Sciences Dorodnicyn Computing Centre of RAS

The paper describes an aproach to chunking of sentences in Russian. Arguments in favor of the correctness and practicability of the chunking problem for a language with free word order are provided. An aproach based on conditional random fields provides detection of a certain class of chunks (base-NPs) with F_1 measure above 0.94 and the training set can be obtained from the raw text data processed by statistical parser without manual postprocessing. Meanwhile, detecting of longer phrases remains problematic and the F_1 measure in the corresponding experiment is relatively small.

Keywords: *probabilistic graphical models, shallow parsing, natural language processing.*

Введение

Решение задачи синтаксического разбора является одним их ключевых промежуточных пунктов в большом количестве задач, связанных с обработкой естественного языка. Важнейшими из них являются моделирование понимания речи и извлечение фактов из текста. Полный синтаксический разбор основан на аппарате синтаксических деревьев [1]. Однако полный синтаксический разбор имеет ряд недостатков, вследствие которых он не является одинаково пригодным для различных задач обработки естественного языка. К таким недостаткам можно отнести большой объем статистической модели (в случае, если анализатор основан на машинном обучении), большой расход памяти или низкое быстродействие, относительно невысокая точность разбора большинства современных решений. Для организации систем речевого диалога полный синтаксический разбор тем бо-

лее затруднителен, что в устной речи говорящий часто склонен организовывать свою речь не в форме предложений, а в форме более коротких фраз. Кроме того, полный синтаксический разбор является неустойчивым к незнакомым и ошибочно распознанным словам.

В работе [2] был предложен подход, основанный на выделении синтаксически связанных фрагментов текста (прямое заимствование «чанк» — chunk — пока не является распространенным в научной литературе, поэтому далее позволим себе использовать сокращение СФТ.) с помощью каскада конечных автоматов [3] с их последующим объединением в дерево на втором этапе работы алгоритма. Единственным формальным требованием к связанным фрагментам была нерекурсивность. Выделение связанных фрагментов именных групп без построения синтаксического дерева было обозначено в качестве самостоятельной задачи в работе [4]. В этой же работе впервые был предложен подход, основанный на машинном обучении. В работе [5] задача выделения СФТ была сведена к задаче маркировки последовательности слов с использованием СММ.

В работе [6] на примере решения задачи расстановки частеречных тегов, также являющейся задачей маркировки последовательностей, была продемонстрирована высокая эффективность марковской модели максимальной энтропии. В силу того, что ММЭ является дискриминационной моделью и позволяет использовать большое количество, вообще говоря, коррелированных признаков, ее использование значительно улучшило результаты распознавания, по сравнению с СММ.

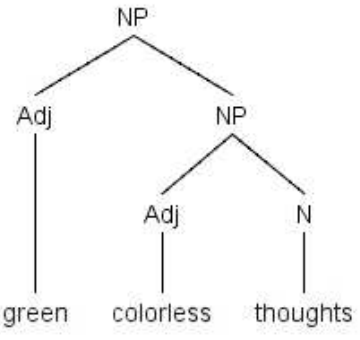
В работе [7] было предложено очередное улучшение модели маркировки последовательностей - условное случайное поле для линейной цепи (Linear-chain CRF, см. [8]). Авторы также использовали аппарат условных случайных полей для задачи расстановки частеречных тегов. Важнейшим преимуществом условных случайных полей перед марковской моделью максимальной энтропии является преодоление т.н. проблемы *смещения метки* (label bias problem), которая заключается в том, что в силу специфики коэффициента нормализации в каждом из факторов цепи для ММЭ состояния, имеющие высокую энтропию распределения последующих состояний имеют меньшие шансы быть выбранными в качестве меток, даже если на это указывают наблюдения [9].

Наконец, в статье [10] аппарат условных случайных полей был успешно применен к выделению СФТ в английском тексте. В качестве целевых меток использовались метки BIO, обозначающие начало фрагмента, середину фрагмента и нахождение вне фрагмента соответственно.

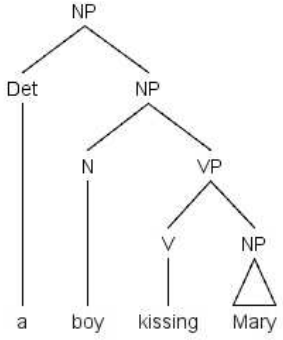
Использование методов машинного обучения в задачах выделения СФТ на материале языков с развитой морфологией (в том числе, славянских) и, как следствие, более свободным порядком слов на настоящий момент не распространено. Обзор методов поверхностного синтаксического разбора для чешского и польского языков изложены в работах [11], [12]. В обоих случаях использован подход, основанный на правилах.

При попытке описать существенный фрагмент грамматики языка разработка правил становится все более трудоемкой, а описание, иногда противоречивым, что приводит к необходимости ручного ранжирования. Этих проблем удается избежать, если использовать методы машинного обучения. Для начала, однако, необходимо уточнить формулировку задачи с учетом специфики языка.

Таким образом, в разд. 1 будут сформулированы и переформулированы некоторые базовые термины, необходимые для дальнейшего изложения. Далее в разд. 2 будут приведены данные, косвенно свидетельствующие о корректности и целесообразности решения задачи определения СФТ на русском материале в принципе, а также приведенных выше формулировок. В разд. 3 кратко излагается аппарат условных случайных полей. В



(а) Именная группа, являющаяся базовой именной группой



(б) Именная группа, не являющаяся базовой именной группой (нарушено условие нерекурсивности)

Рис. 1: Связь между проективностью и возможностью выделения базовых именных групп: построение базовой именной группы длинной более одного слова для дерева (б) невозможно

разд. 4 содержатся информация об обучающих и тестовых данных. В разд. 5 содержится информация о реализации алгоритмов и результаты экспериментов.

Базовые именные группы в русском языке

Центральным понятием в задаче выделения СФТ является понятие базовой группы, или base-XP. XP является обозначением синтаксической составляющей, принятым в англоязычных статьях. Так синтаксическая составляющая, возглавляемая вершиной-существительным, называется NP — noun phrase (именной группой); для глагола это VP — verb phrase (глагольная группа); синтаксическую составляющую безотносительно к ее типу принято обозначать как XP.

В работе [2], на возможные СФТ накладывалось следующее формальное ограничение: в их полном синтаксическом разборе не должно было присутствовать рекурсивных правил. Было продемонстрировано, что такие фрагменты выделяются значительно проще ввиду меньшего влияния синтаксической неоднозначности (attachment ambiguity). Таким образом, в первом приближении СФТ представляет собой вершину в синтаксическом дереве с зависимыми, соседними в линейном порядке, причем в полученном поддереве зависимостей не должно быть иных вершин, имеющих ту же часть речи, что и корень. Например:

green colorless thoughts является СФТ

a boy kissing Mary не является СФТ

В работе [4] речь идет об СФТ с вершиной-существительным. Для таких фрагментов было предложен термин базовой именной группы (Base-NP). В [1] отмечается, что строгого определения ни для СФТ (chunk), ни для базовой именной группы (Base-NP) не было предложено, однако требование нерекурсивности оставалось.

Поскольку в дальнейшем речь пойдет именно о базовой именной группе, сформулируем ее определение более строго:

Определение 1. *Последовательность слов в предложении S называется базовой именной группой, если (1) она является подпоследовательностью некоторой именной группы в дереве непосредственных составляющих; (2) в выводе данной подпоследовательности*

в грамматике языка G нетерминальный символ NP, соответствующий вершине именной группы, встречается единственный раз.

Корректность данного определения следует из определения именной группы (см. [13]).

Для того, чтобы дать определение базовой именной группы в русском языке, необходимо учесть тот факт, что английские определения имеют тенденцию помещаться слева в линейном порядке от вершины в синтаксическом дереве. В этом случае принято трактовать существительные как прилагательные:

USA Supreme Court

В русском подобные конструкции часто переводятся с помощью родительного падежа:

Верховный суд Российской федерации

Это подсказывает определение базовой именной группы в русском языке:

Определение 2. *Последовательность слов в предложении S на русском языке называется базовой именной группой, если 1) Она является подпоследовательностью некоторой именной группы в дереве непосредственных составляющих; 2) В дереве непосредственных составляющих для данной подпоследовательности существует не более одной именной группы, в т. ч. вершина данной группы стоит не в родительном падеже.*

Данное определение, безусловно, не покрывает всех возможных случаев употребления родительного падежа, но учитывает два наиболее важных случая употребления: 1) обозначение синтаксической зависимости между двумя существительными и 2) употребление родительного падежа в отрицательных предложениях.

Ниже будет показано, что выделение базовых именных групп согласно определению 2, действительно, позволяет выделять в тексте логически связанные фрагменты, однако вначале приведем экспериментальные свидетельства того, что более свободный порядок слов в русском языке не является «фатальным» для выделения связанных фрагментов текста.

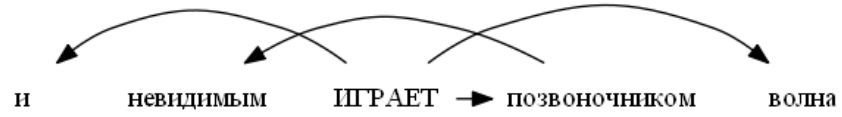
Эффекты свободного порядка слов

Одной из наиболее трудных проблем, которые ставит перед инженерами по обработке ЕЯ русский язык, является свободный порядок слов. Тем не менее имеется ряд как теоретических аргументов [13], так и эмпирических фактов, свидетельствующих об обратном. Так, известно, что большинство предложений на русском литературном языке проективны. Это, в свою очередь, означает, что в русских предложениях должна прослеживаться тенденция к сохранению базовых именных групп, хотя проективность сама по себе не гарантирует сохранение синтаксических групп. Для оценки эффектов от свободного порядка слов был поставлен эксперимент на небольшом корпусе интервью в электронных СМИ.

Обрабатываемые данные Обрабатываемый корпус составляли 500 предложений, взятых из интервью, опубликованных в интернет-изданиях. Каждый текст предварительно обрабатывался последовательно морфологическим анализатором TreeTagger [14], лемматизатором CSTlemma [15] и синтаксическим анализатором Malt Parser [16]. В результате был получен синтаксически аннотированный корпус, где каждому предложению было сопоставлено дерево зависимостей.

Экспериментальная методика Каждое синтаксическое дерево обрабатывалось алгоритмом, который восстанавливал порядок слов, соответствующий сильно проективной конструкции. Все предложения, подвергшиеся изменению, просматривались вручную для выяснения причин изменения исходного порядка слов. Просмотр выявил, что большая часть изменений (17 случаев) была вызвана ошибками в синтаксическом разборе Malt

(а) Дерево зависимостей и границы базовых именных групп для проективного предложения



(б) Дерево зависимостей для непроективного предложения

Рис. 2: Связь между проективностью и возможностью выделения базовых именных групп: построение базовой именной группы длинной более одного слова для дерева (б) невозможно

Parser'а. Из оставшихся 9 случаев 6 приходились на слабо-проективные конструкции с составным глагольным сказуемым (например, *захотел пойти*) и не могли оказать влияния на базовые именные группы. В результате лишь 3 случая приходились на действительно непроективные конструкции. Однако даже среди них дважды встретилась конструкция *друг к другу*, которая была разобрана как полноценное синтаксическое поддерево, что вообще говоря спорно.

Безусловно, экспериментальная выборка не может считаться репрезентативной, однако она позволяет предположить, что свободный порядок слов не является причиной для отказа от выделения СФТ именных групп. Одним из подходов к выделению таких фрагментов, показавший свою эффективность для многих языков со строгим порядоком слов, является подход, основанный на условных случайных полях для линейной цепи.

Условные случайные поля для линейных цепей

Условное случайное поле представляет собой частично ориентированную графическую вероятностную модель, а именно марковскую сеть, в которой при этом имеется условное распределение одного подмножества переменных (ненаблюдаемых) в зависимости от другого подмножества переменных (наблюдаемых). Условные случайные поля широко используются в сегментации изображений, распознавании действий, обработке естественного языка и других областях [9]. Общее определение для условных случайных полей таково:

Определение 3. *Условным случайным полем для переменных* $\boldsymbol{X} \cup \boldsymbol{Y}$ *называется неориентированный граф* H*, с множеством вершин* $V = \boldsymbol{X} \cup \boldsymbol{Y}$ *и множеством ребер с ассоциированными факторами* $\varphi_1(\boldsymbol{D}_1), \ldots, \varphi_m(\boldsymbol{D}_m)$ $(\varphi_i(\boldsymbol{D}_i) \geqslant 0)$*, причем* $\forall i \boldsymbol{D}_i \nsubseteq \boldsymbol{X}$*, а распределение вероятностей* $\mathsf{P}(\boldsymbol{Y} \mid \boldsymbol{X})$ *задается согласно формулам:*

$$\begin{aligned} \mathsf{P}(\boldsymbol{Y} \mid \boldsymbol{X}) &= \frac{1}{Z(\boldsymbol{X})} \tilde{\mathsf{P}}(\boldsymbol{Y}, \boldsymbol{X}) \\ \tilde{\mathsf{P}}(\boldsymbol{Y}, \boldsymbol{X}) &= \prod_{i=1}^{m} \varphi_i(\boldsymbol{D}_i) \\ Z(\boldsymbol{X}) &= \sum_{\boldsymbol{Y}} \tilde{\mathsf{P}}(\boldsymbol{Y}, \boldsymbol{X}) \end{aligned} \tag{1}$$

Тогда любые две вершины y_k, y_l *в* H *соединены неориентированным ребром тогда и только тогда, когда* $\exists \varphi_i \{y_k, y_l\} \subseteq \boldsymbol{D}_i$

В настоящей статье нас интересуют условные случайные поля определенного типа, а именно условные случайные поля для линейной цепи. Данную модель можно рассматривать как дискриминационный аналог скрытой марковской модели. Поэтому будем считать, что $\mathbf{x}$ есть последовательность слов в предложении, а $\mathbf{y}$ - последовательность меток BIO. Соответствующее условное случайное поле выглядит так, как показано на рис. 3.

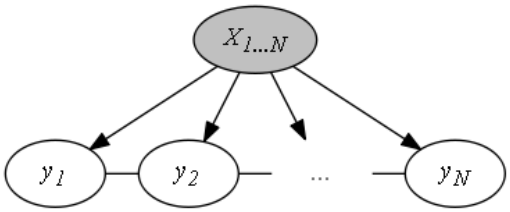


Рис. 3: Условное случайное поле для линейной цепи. Цветом обозначаются наблюдаемые значения. Дуги соответствуют факторам. Прописное X и направленные дуги указывают на то, что значение каждого фактора $\varphi_t^2(y_t, y_{t-1})$ пропорционально вероятности $p(y_t, y_{t-1}|X_{1...N})$, т.е. вероятности того, что данные скрытые состояния «участвовали» в генерации всей наблюдаемой последовательности. Подробнее о нотации см. [9]

В данной цепи имеются два типа факторов - одиночные (singleton) и парные (pairwise). Одиночные факторы $\varphi_t^1(y_t, \mathbf{x})$ задают влияние, которое оказывает наблюдаемая последовательность на метку y_t. Парные факторы $\varphi_t^2(y_t, y_{t-1})$ задают влияние соседних меток друг на друга. Важным отличием от СММ и ММЭ является то, что факторы вообще говоря не обязаны удовлетворять неравенству $\varphi(\mathbf{D}) \leqslant 1$. Аргумент $\mathbf{x}$ в факторе φ_t^1 указывает на то, что при вычислении его значений потенциально могут использоваться признаки любых элементов последовательности. Такими признаками, например, могут быть «слово x_t является существительным» или «слово x_t 'человек', слово x_{t+1} 'собака'». С другой стороны, φ_t^1 является функцией y_t, поскольку значение $\mathbf{x}$ постоянно.

Стандартным методом вывода в графических моделях является конструирование кластерного дерева с последующим применением алгоритма Витерби. В случае линейной цепи кластерное дерево также является цепью. Например, если в структуре на рис. 3 значения в каждом из одиночных факторов будут вычисляться только на основе признаков текущего наблюдения, то кластерное дерево примет следующий вид:

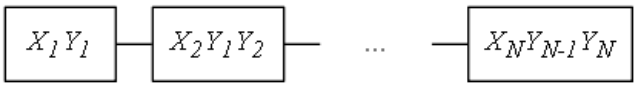


Рис. 4: Кластерное дерево, соответствующее условному случайному полю для линейной цепи с «окном наблюдения» равным 1

Фактор кластера ψ_t представляет собой произведение факторов, входящих в него. Таким образом, его область определения совпадает с областью определения парного фактора, в то время как при вычислении его значений также используются и различные признаки элементов наблюдаемой последовательности. Из этих соображений будем обозначать фактор кластера как $\psi_t(y_t, y_{t-1}, \mathbf{x})$.

Для вычисления значений фактора $\psi_t(y_t, y_{t-1}, \mathbf{x})$ на основе наблюдаемых признаков вводятся индикаторные функции признаков f_k и соответствующие веса λ_k. Тогда искомая

условная вероятность находится по следующей формуле:

$$\mathsf{P}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{X})} \exp\{\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x})\}, \quad (2)$$

где $\mathrm{Z}(\mathbf{X})$ – функция нормализации, которая вычисляется по формуле:

$$\mathsf{P}(\mathbf{y} \mid \mathbf{x}) = \sum_{y} \exp\{\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x})\} \quad (3)$$

На этапе обучения осуществляется подбор весов λ_k, на которых достигается максимум правдоподобия обучающей выборки. Обучение осуществляется с помощью различных вариаций градиентного подъема. Одним из наиболее эффективных является алгоритм L-BFGS. Исчерпывающая информация и ссылки содержатся в [9] и [8].

Используемые данные

Для экспериментов использовались обучающие выборки двух типов. Первая обучающая выборка была получена путем обработки неразмеченного текста из корпуса OpenCorpora морфологическим и синтаксическим анализатором (TreeTagger, CSTlemma, Malt Parser). В результате была получена исходная выборка с синтаксической разметкой в форме дерева зависимостей. Всего 46397 предложений. В качестве второй исходной выборки была использована часть синтаксически аннотированного корпуса русского языка SynTagRus ИППИ РАН. В обучающее множество было выделено всего 40976 предложений. SynTagRus также использует в качестве синтаксической аннотации структуру дерева зависимостей. Далее в каждом синтаксическом дереве выделялись базовые именные группы. Первое слово группы получало метку B, последующие — метку I; слова, не вошедшие ни в одну из базовых именных групп, получали метку O. Алгоритм извлечения базовых именных групп представлял собой простой обход дерева вглубь с объединением в одну базовую именную группу поддеревьев, удовлетворяющих следующим критериям:

1) Все узлы поддерева представляют собой подпоследовательность в линейном порядке слов без разрывов.
2) Вершиной каждого поддерева является слово-существительное в любом падеже.
3) В остальных узлах поддерева могут присутствовать только узлы со следующими характеристиками:
   (а) существительное в родительном падеже;
   (б) прилагательное или порядковое числительное в любом падеже;
   (в) наречие.
4) Знаки препинания отсутствуют в подпоследовательности.

Таким образом, были получены две обучающие выборки с *BIO*-разметкой. В тестовое множество было выделено 6310 предложений корпуса SynTagRus, обработанных тем же способом. Две указанных обучающих выборки и тестовая выборка были использованы в первой серии экспериментов.

Для второй серии экспериментов набор меток *BIO* был расширен двумя метками BH и IH с целью выделения вершины базовой именной группы. Методика подготовки обучающего и тестового множеств осталась прежней, за исключением незначительных необходимых изменений в алгоритме генерации меток на основе синтаксического дерева.

Целью третьей серии экспериментов было оценить возможности алгоритма в выделении более длинных фрагментов текста: рядов однородных членов предложения, конструкций с предлогом или союзом внутри, пунктуации. В этом случае ИГ, вообще говоря, допускает рекурсивную вложенность. Назовем такие фрагменты рекурсивными именными группами. Для этой задачи были проведены эксперименты только с выборкой на основе корпуса SynTagRus.

Эксперименты

Для работы с условными случайными полями была использована библиотека MALLET [17], реализованная на Java. В качестве признаков последовательности были использованы следующие признаки токенов: часть речи, падеж, число, род, заглавная буква. Символы пунктуации рассматривались как самостоятельная часть речи. Для каждого токена в качестве признаков также использовались признаки правого и левого соседа, а также все полученные конъюнкции признаков текущего и каждого из соседних токенов: $\{f_t^1 \& f_{t-1}^1$ $f_t^2 \& f_{t-1}^1 \ldots\}$.

Таким образом, в основном в качестве признаков использовались морфологические характеристики имен. Признак «слово начинается с заглавной буквы» было решено ввести для более качественной обработки имен и должностей: *Президент Российской Федерации Борис Борисович Гребенщиков*. Также было решено поставить отдельный эксперимент для моделей, использующих токен в качестве признака.

Стандартной оценкой качества поверхностного синтаксического разбора является заимствованный из информационного поиска показатель по F_1 мере, вычисляемый на основе доли правильно выделенных СФТ.

Рассмотрим коллекцию текстов, взятых в качестве тестовой выборки. Число СФТ, правильно распознанных алгоритмом, отнесенное к количеству элементов, возвращенных алгоритмом, называется точностью (Precision):

$$Precision = \frac{tp}{tp + fp}, \tag{4}$$

где tp (true positive) — количество СФТ правильно распознанных алгоритмом, fp (false positive) – число СФТ, ошибочно возвращенных алгоритмом.

Полнотой (Recall) называется следующая величина:

$$Recall = \frac{tp}{tp + fn}, \tag{5}$$

где fn (false negative) — число СФТ, ошибочно пропущенных алгоритмом.

Взвешенное среднее гармоническое этих величин называется F_1-мерой:

$$F_1 = \frac{1}{\frac{1}{P} * \frac{1}{2} + \frac{1}{R} * \frac{1}{2}} = \frac{2PR}{P + R} \tag{6}$$

, где P и R – соответственно точность и полнота.

Результаты экспериментов приведены в табл. 1 и 2.

Оценка точности, полноты и F_1-меры для выделения вершин СФТ проводилась аналогичным образом: оценивалась доля вершин, правильно определенных алгоритмом.

Из таблиц видно, что алгоритм демонстрирует сбалансировано высокий результат как по точности, так и по полноте, что соответствующим образом сказывается и на F_1-мере. Более того, расширение набора тегов BIO, используемого, например, для английского

Таблица 1: Результаты экспериментов. Базовые и рекурсивные ИГ.

Модель	Precision	Recall	F_1-мера
`SynTagRus. Базовая ИГ. Токены+`	0.9339	0.9307	0.9323
`SynTagRus. Базовая ИГ. Токены-`	0.9452	0.9427	0.9439
`OpenCorpora. Базовая ИГ. Токены+`	0.9229	0.9115	0.9172
`OpenCorpora. Базовая ИГ. Токены-`	0.9305	0.9238	0.9271
`SynTagRus. Рекурсивная ИГ. Токены+`	0.7521	0.7747	0.7632
`SynTagRus. Рекурсивная ИГ. Токены-`	0.7654	0.7536	0.7594

Таблица 2: Результаты экспериментов. Выделение вершин.

Модель	Precision	Recall	F_1-мера
`SynTagRus. Токены+`	0.9556	0.9514	0.9555
`SynTagRus. Токены-`	0.9648	0.9545	0.9596
`OpenCorpora. Токены+`	0.9601	0.9458	0.9529
`OpenCorpora. Токены-`	0.9523	0.9428	0.9510

языка [10] за счет дополнительных тегов для выделения вершин, нисколько не ухудшает результат, а качество их выделения по F_1-мере также оказывается высоким (см. табл. 2). Это позволяет реализовать возможность постановки всего СФТ именной группы в начальную форму, например, для обращения к базе данных. Также стоит отметить тот факт, что модель обученная на основе выборки, полученной при помощи автоматического синтаксического анализатора, показывает достаточно высокий результат (строки 3, 4 табл. 1). Таким образом, модель может быть обучена на корпусе текстов, собранном самостоятельно, что критично для языков, не имеющих больших открытых лингвистических корпусов (к ним относится и русский). Небольшое снижение всех показателей при добавлении признаков-токенов, по всей видимости, является результатом переобучения. Последним фактом, который нельзя не отметить, является значительное ухудшение как точности, так и полноты при попытке обработки более длинных фрагментов текста. В данном случае гораздо сильнее сказывается синтаксическая неоднозначность. К примеру, для правильного распознавания групп с предлогом внутри использование только морфологических характеристик представляется недостаточным: *[Международный суд по правам человека] в [Гааге] vs. [Международный суд по правам] [человека в Гааге] vs. [Международный суд] по [правам человека] в [Гааге] и т.д.* Вопреки ожиданиям, богатый набор морфологических характеристик словоформ в русском языке не дает желаемого эффекта.

Заключение

В работе была совершена попытка адаптации одной из техник синтаксического анализа, давно и эффективно применяющихся для языков с более фиксированным порядком слов. Несмотря на то, что ранее было принято считать, что свободный порядок слов, является препятствием для применения поверхностного синтаксического анализа, результаты экспериментов показывают, что метод работает достаточно надежно и может найти свое применение в тех задачах, где применение полного синтаксического анализа не требуется. Этими задачами могут быть поиск ключевых слов в документе или высказывании, поступающем на вход диалоговой системы. Метод позволяет выделять в тексте такие фрагменты, как: *[экологическая программа]*, *[Президиум Совета Министров СССР]*, *[бас-*

сейн Азовского моря]. Необходимым условием применения данного метода, является такое специфичное для данного языка определение базовой синтаксической группы, которое бы позволило находить максимально длинные нерекурсивные фрагменты текста, что и было сделано в статье.

Достоинством метода является его нетребовательность к качеству обучающей выборки: она может быть получена на основе выдачи доступного синтаксического анализатора. Безусловным недостатком является низкая точность выделения фрагментов, содержащих внутри предлог и/или союз: *[Комиссия ООН] по [правам человека]* вместо *[Комиссия ООН по правам человека]*. В целом в статье удалось показать, что при внесении необходимых поправок, зависящих от языка, метод работает не хуже, чем для языков с более строгим порядком слов.

Литература

[1] *Jurafsky D., Martin M.* Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition. 2nd ed. Upper Saddle River, New Jersey: Prentice-Hall, 2009. Pp. 427–458.

[2] *Abney S.* Parsing by chunks. Principle-based Parsing / Eds. R. Berwick, S. Abney, and C. Tenny. Kluwer Academic Publishers, 1991. Pp. 257–279.

[3] *Abney S.* Part-of-speech tagging and partial parsing // Corpus-based methods in language and speech processing. / Eds. S. Young, G. Bloothooft. Kluwer Academic Publishers, 1997. Pp. 124–136.

[4] *Ramshaw L, Marcus M.* Text chunking using transformation-based learning // 3rd Annual Workshop on Very Large Corpora Proceedings, 1995. Pp. 82–94.

[5] *Freitag D., and McCallum A.* Information extraction with HMM structures learned by stochastic optimization // 17th National Conference on Artificial Intelligence (AAAI) Proceedings. Austin, Texas, 2000.

[6] *McCallum A., Freitag M. and Pereira F* Maximum entropy Markov models forinformation extraction and segmentation // 17th International Conference on Machine Learning Proceedings. Stanford, California, 2000. Pp. 591–598.

[7] *Lafferty J., McCallum A. and Pereira F.* Conditional random fields: Probabilistic models for segmenting and labeling sequence data // 18th International Conference on Machine Learning Proceedings. Williamstown, Massachusetts, 2001. Pp. 282–289.

[8] *Sutton C., Mccallum A.* Introduction to conditional random fields for relational learning // Introduction to statistical relational learning / Eds. L. Getoor, B. Taskar. Cambridge, Massachusetts: MIT Press, 2006.

[9] *Koller D., Friedman N.* Probabilistic graphical models: Principles and techniques. Cambridge, Massachusetts: MIT Press, 2009. Pp. 943–961., 345–361., 561–564.

[10] *Sha F, Pereira F.* Shallow parsing with conditional random fields // Proceedings of HLT/NAACL, 2003. Pp. 213–220.

[11] *Dolezalova J, Petkevic V.* Shallow parsing of Czech sentence based on Correct morphological disambiguation // Linguistics Investigations into Formal Description of Slavic Languages / P. Kosta, L. Schürcks L., editors, , 2005.

[12] *Przepiorkowski A.* Slavic information extraction and partial parsing // Proceedings of the Workshop on Balto-Slavonic Natural Language Processing. Praga, 2005.

[13] *Тестелец Я. Г.* Введение в общий синтаксис. Москва: РГГУ, 2001. 800 с.

[14] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/` — TreeTagger — a language independent part-of-speech tagger, 2013.

[15] http://corpus.leeds.ac.uk/mocky/ — Russian statistical taggers and parsers, 2013.

[16] *Sharof S., Nivre, J.* The proper place of men and machines in language technology: Processing Russian without any linguistic knowledge. // Proceedings of Dialogue, Russian Conference on Computational Linguistics, 2011.

[17] *McCallum A.* MALLET: A Machine Learning for Language Toolkit. // http://mallet.cs.umass.edu MaltParser, 2002.

[18] *Маннинг К., Рагхаван П., Шютце Х.* Введение в информационный поиск. Москва: Вильямс, 2011. 528 с.

[19] *Грановский Д. В., Бочаров В. В., Бичинева С. В.* Открытый корпус: принципы работы и перспективы // http://opencorpora.org/doc/articles/2010_IMS.pdf

Иерархические структуры данных и решающие алгоритмы для классификации изображений*

Ланге М. М.,Ганебных С. Н.
lange_mm@ccas.ru
Москва, Вычислительный центр им. А.А. Дородницына РАН

Исследуется задача классификации объектов, заданных изображениями, в терминах соотношения вычислительной сложности и вероятности ошибки. Используя многоуровневую сеть эталонов, предлагаются алгоритмы иерархического поиска решения по критерию ближайшего эталона. Получены сравнительные оценки вычислительной сложности иерархических алгоритмов относительно алгоритма полного перебора эталонов. Приведены экспериментальные зависимости вычислительной сложности от вероятности ошибки распознавания подписей, жестов и лиц для решающих алгоритмов на основе иерархического поиска и полного перебора.

Ключевые слова: *изображение, классификация, эталонный объект, многоуровневая структура, решающий алгоритм, переборный поиск, иерархический поиск, вычислительная сложность, доля ошибок.*

Hierarchical Data Structures and Decision Algorithms for Efficient Image Classification*

Lange M. M., Ganebnykh S. N.
Dorodnicyn Computing Centre of RAS, Moscow, Russian Federation

The problem of image-based object recognition in terms of computational complexity as a function of error rate is studied. Using a multilevel network of the template objects, two fast guided search algorithms for the decision template are suggested. For the guided search and exhaustive search algorithms, comparative estimations of computational complexity are estimated. Experimental curves of computational complexity as the function of classification error rate are obtained for a common source of gestures, signatures, and faces using the guided and exhaustive search decision algorithms.

Keywords: *image, classification, template object, multilevel structure, decision algorithm, exhaustive search, guided search, computational complexity, error rate.*

Введение

В ряде приложений задачу распознавания образов целесообразно решать в терминах соотношения характеристик качества и вычислительной сложности, требуя минимизации вероятности ошибки при заданном ограничении на объем вычислений либо минимизации объема вычислений при допустимой вероятности ошибки. В такой постановке метрическая модель классификации имеет очевидное сходство с теоретико-информационной моделью кодирования источников с заданным критерием качества (Rate Distortion Coding) при наличии канала наблюдения [1]. В модели классификации канал наблюдения эквивалентен представлению, формирующему описания объектов источника (образов), в пространстве которых строится классификатор. Решение может быть получено в пространстве иерар-

Работа выполнена при финансовой поддержке РФФИ, проект № 12-01-00920-а.

хических описаний образов с многоуровневым разрешением [2, 3, 4]. Структура таких описаний удобна для сокращения объема вычислений и позволяет реализовать алгоритмы быстрого анализа геометрических форм. В настоящей работе исследуется задача распознавания образов, заданных изображениями, в пространстве древовидно структурированных представлений эллиптическими примитивами [5]. Класс допустимых объектов, которые допускают такой способ представления, включает всевозможные объекты в виде многосвязных двумерных твердых тел с однозначно идентифицируемой системой собственных координат. Допускаются объекты в форме линейчатых (line-based) и областных (region-based) тел с полутоновой окраской. К таким объектам относятся подписи, рукописные символы, отпечатки пальцев, жесты, лица, силуэты и многие другие. Для широкого класса источников древовидные представления обладают свойством универсальности, а структура таких представлений позволяет строить сети эталонов с многоуровневым разрешением, которые допускают применение быстрых решающих алгоритмов на основе процедуры направленного поиска [6].

Задача классификации в пространстве представлений образов

Пусть $\mathbf{A}$ – множество образов, заданных изображениями, в котором каждый образ $A \in \mathbf{A}$ имеет многоуровневое представление

$$A^L = (a^0, \ldots, a^l, \ldots, a^L), \tag{1}$$

в виде полного бинарного дерева [6] заданной глубины L. Представление a^l в (1) образовано набором из 2^l эллиптических примитивов, записанных в узлах l-го уровня, а последовательность представлений $A^l = (a^0, \ldots, a^l)$ образует поддерево глубины l в дереве A^L. Дерево A^L строится путем рекурсивного разбиения образа A на сегменты и аппроксимации сегментов в узлах с номерами $n = 0, \ldots, 2^{L+1} - 2$ эллиптическими примитивами вида

$$Q_n = (n, \mathbf{r}_n, \mathbf{u}_n, \mathbf{v}_n, z_n), \tag{2}$$

где $\mathbf{r}_n$ – вектор центра эллипса; $\mathbf{u}_n$, $\mathbf{v}_n$ – векторы большой и малой полуосей; z_n – средний уровень яркости пикселей в аппроксимируемом сегменте. Векторы $\mathbf{r}_n$, $\mathbf{u}_n$, $\mathbf{v}_n$ задаются в собственных координатах образа A. Каждый делимый сегмент с номером n дает пару потомков с номерами $2n + 1$, $2n + 2$. Если сегмент не может быть разбит на две части, то его потомки дублируют этот сегмент. Параметры каждого примитива (2) нормируются относительно соответствующих параметров корневого примитива с номером $n = 0$. Класс источников изображений и свойства представлений (1) определены в следующих утверждениях.

Утверждение 1. *Источник объектов, заданных на изображениях двумерными твердыми телами в виде наборов пикселей с идентифицируемой системой собственных координат, порождает множество образов* $\mathbf{A}$*, которые допускают многоуровневое представление вида* (1).

Утверждение 2. *При достаточно малом размере пикселей и большом числе уровней квантования яркостей, дерево примитивов* A^L *вида* (1) *практически инвариантно к преобразованиям поворота, смещения, изменения масштаба и уровня яркости образа* A.

Примеры многоуровневых представлений подписи, жеста руки и лица наборами эллиптических примитивов a^1, a^3, a^5, a^7, даны на рис. 1.

Для определения семейства мер различия объектов $A \in \mathbf{A}$, $\hat{A} \in \mathbf{A}$ по их представлениям A^L, $\hat{A}^L$ вида (1) определим функции различия соответственных примитивов $Q_n \in A^L$ и

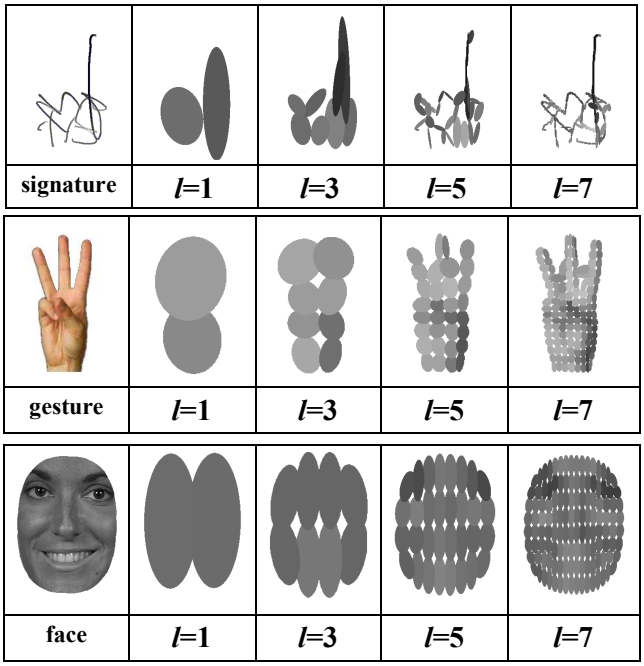


Рис. 1: Примеры многоуровневых представлений подписи, жеста руки и лица

$\hat{Q}_n \in \hat{A}^L$ (с одинаковыми номерами n) по трем группам параметров. Эти функции имеют следующий вид

$$\rho_1(Q_n, \hat{Q}_n) = \|\mathbf{r}_n - \hat{\mathbf{r}}_n\|,$$

$$\rho_2(Q_n, \hat{Q}_n) = \frac{1}{2}\left(\frac{\|\mathbf{u}_n - \hat{\mathbf{u}}_n\| + \|\mathbf{v}_n - \hat{\mathbf{v}}_n\|}{\max(\|\mathbf{u}_n\|, \|\hat{\mathbf{u}}_n\|)}\right),$$

$$\rho_3(Q_n, \hat{Q}_n) = |z_n - \hat{z}_n|.$$

Для пары объектов A, $\hat{A}$ мера различия l-го порядка по k-й группе параметров примитивов определяется на поддеревьях A^l, $\hat{A}^l$ и имеет вид

$$d_k^l(A, \hat{A}) = \sum_{n=1}^{2^{l+1}-2} w_n \rho_k(Q_n, \hat{Q}_n), \tag{3}$$

где $w_n = \lambda \lfloor \log_2(n+1) \rfloor 2^{-\lfloor \log_2(n+1) \rfloor}$ – весовой коэффициент n-го примитива с нормировочным множителем λ, обеспечивающим $\sum\limits_{n=1}^{2^{l+1}-2} w_n = 1$, $k = 1, 2, 3$, $l = 1, \ldots, L$. Суммирование мер (3) с весами $\omega_k : \omega_1 + \omega_2 + \omega_3 = 1$ дает обобщенную меру различия l-го порядка

$$d^l(A, \hat{A}) = \sum_{k=1}^{3} \omega_k d_k^l(A, \hat{A}), \tag{4}$$

где $l = 1, \ldots, L$, а веса ω_k, $k = 1, 2, 3$, определяются оценками вероятностей ошибок по мерам (3), полученными методом скользящего контроля на этапе обучения.

Будем считать, что множество $\mathbf{A} = \{\mathbf{A}_i\}_{i=1}^c$ содержит объекты, принадлежащие $c \geqslant 2$ классам. Для обучения используется обучающее множество

$$\mathbf{B} = \left\{\mathbf{B}_i = \{B_{ij}\}_{j=1}^M\right\}_{i=1}^c \subset \mathbf{A}, \tag{5}$$

в котором каждый кластер $\mathbf{B}_i \subset \mathbf{A}_i$ содержит M объектов. Множество (5) используется для оценивания весовых коэффициентов в мере (4) и для отбора эталонных объектов. В каждом кластере $\mathbf{B}_i$, $i = 1, \ldots, c$, формируется ансамбль наборов эталонов

$$\left(\hat{\mathbf{B}}_i^1, \ldots, \hat{\mathbf{B}}_i^m, \ldots, \hat{\mathbf{B}}_i^M\right), \tag{6}$$

где m – число эталонов в наборе $\hat{\mathbf{B}}_i^m$. Используя наборы эталонов $\hat{\mathbf{B}}_i^m$, $i = 1, \ldots, c$, $m = 1, \ldots, M$ на множестве представлений вида (1) вводятся разделяющие функции

$$g^l(A|\hat{\mathbf{B}}_i^m) = \max_{B \in \hat{\mathbf{B}}_i^m} K_{d^l}(A, B) \tag{7}$$

порядка $l = 1, \ldots, L$ с ядром $K_{d^l}(A, B)$ в форме невозрастающей функции от меры $d^l(A, B)$ вида (4). Решение относительно класса объекта $A \in \mathbf{A}$ строится на функциях (7) с параметрами $l = L$, $m = M$ и имеет вид

$$i^* = \arg\max_{i=1}^{c} g^L(A|\hat{\mathbf{B}}_i^M) \tag{8}$$

Решение (8) может быть получено с помощью алгоритма полного перебора эталонов (Exhaustive Search) при фиксированных значениях L и M, и с помощью алгоритмов иерархического поиска (Guided Search) решающих эталонов при значениях $l = 1, \ldots, L$ и $m = 1, \ldots, M$. В настоящей работе эффективность решающих алгоритмов исследована в терминах зависимости вычислительной сложности от вероятности ошибки для ядра $K_{d^l}(A, B) = 1 - d^l(A, B)$ и ES-алгоритма с параметрами $l = L$, $m = M$ и двух GS-алгоритмов с параметрами $m = M$, $l = 1, \ldots, L$ и $l = L$, $m = 1, \ldots, M$. Для указанных алгоритмов получены асимптотические оценки сложности при большом числе классов. По результатам тестирования построены зависимости вычислительной сложности от доли ошибок, которые демонстрируют сравнительные соотношения быстродействия и качества классификации для переборного и иерархических алгоритмов поиска решения.

Многоуровневая сеть эталонов

Ансамбли наборов эталонов вида (6), формируемые на кластерах обучающего множества (5), предлагается строить для каждого кластера независимо, в пространстве древовидных представлений в форме (1). На множестве представляющих деревьев с мерой (4) порядка L для каждого кластера $\mathbf{B}_i \subset \mathbf{B}$, $i = 1, \ldots, c$ выполняется дихотомическое разбиение на непересекающиеся сегменты $\mathbf{B}_{ij} : \mathbf{B}_i = \{B_{ij}\}_{j=1}^M$ и выбор в каждом сегменте $\mathbf{B}_{ij}$ эталонного объекта $\hat{B}_{ij} = \arg\min_{\hat{B} \in \mathbf{B}_{ij}} \max_{B \in \mathbf{B}_{ij}} d^L(B, \hat{B})$. На каждом шаге дихотомии выполняется разбиение сегмента $\mathbf{B}_{ij} \to (\mathbf{B}_{ij'}, \mathbf{B}_{ij''})$ имеющего наибольшее рассеяние $\max_{B \in \mathbf{B}_{ij}} d^L(B, \hat{B}_{ij})$ среди сегментов текущего разбиения $\mathbf{B}_i$. Сегменты $\mathbf{B}_{ij'}$ и $\mathbf{B}_{ij''}$ включают объекты, ближайшие к наиболее удаленным друг от друга опорным объектам $B_{ij'} \in \mathbf{B}_{ij}$ и $B_{ij''} \in \mathbf{B}_{ij}$. В результате такой процедуры формируется бинарное дерево, содержащее M уровней (рис. 2). Каждая вершина такого дерева соответствует сегменту $\mathbf{B}_{ij}$ и эталону $\hat{B}_{ij} \in \mathbf{B}_{ij}$, выбираемому в центре этого сегмента. Уровни дерева с номерами $k = 0, \ldots, M-1$ образуют наборы эталонов $\hat{\mathbf{B}}_i^m = \{\hat{B}_{ij}\}_{j=1}^m$ в ансамбле (6), где $m = k + 1$. Последний уровень дает набор эталонов $\hat{\mathbf{B}}_i^M = \{\hat{B}_{ij}\}_{j=1}^M$, совпадающий с кластером $\mathbf{B}_i$ в (5).

Поскольку построение ансамблей $(\hat{\mathbf{B}}_i^1, \ldots, \hat{\mathbf{B}}_i^m, \ldots, \hat{\mathbf{B}}_i^M)$, $i = 1, \ldots, c$ выполняется в пространстве древовидных представлений в форме (1), последовательность представлений

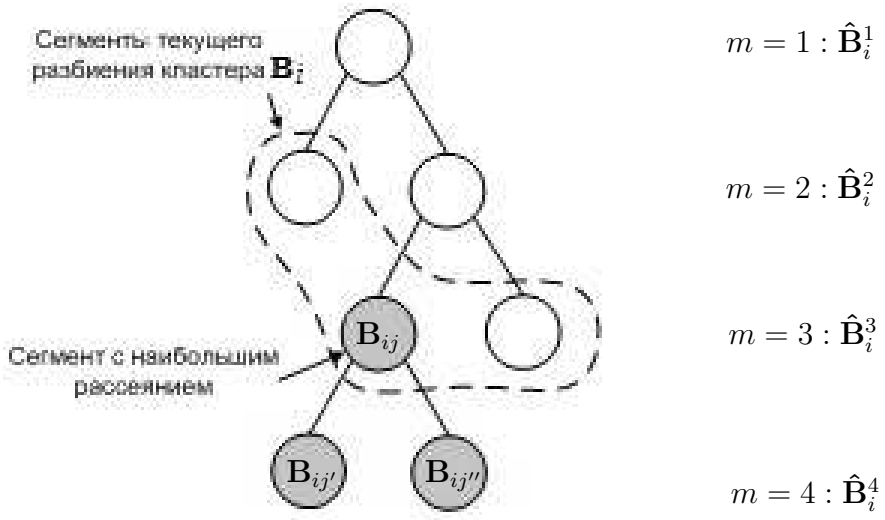


Рис. 2: Древовидная структура ансамбля наборов эталонов $(\hat{\mathbf{B}}_i^1, \ldots, \hat{\mathbf{B}}_i^m, \ldots, \hat{\mathbf{B}}_i^M)$ кластера $\mathbf{B}_i$

порядка $l = 1, \ldots, L$ для указанных ансамблей образует многоуровневую сеть эталонов. Такая сеть может быть записана в форме матрицы

$$\begin{array}{ccccc} \hat{\mathbf{B}}^{11} & \ldots & \hat{\mathbf{B}}^{l1} & \ldots & \hat{\mathbf{B}}^{L1} \\ \hat{\mathbf{B}}^{1m} & \ldots & \hat{\mathbf{B}}^{lm} & \ldots & \hat{\mathbf{B}}^{Lm} \\ \hat{\mathbf{B}}^{1M} & \ldots & \hat{\mathbf{B}}^{lM} & \ldots & \hat{\mathbf{B}}^{LM} \end{array} \tag{9}$$

в которой $\hat{\mathbf{B}}^{lm}$ – представление порядка l для множества наборов $\hat{\mathbf{B}}^m = \{\hat{\mathbf{B}}_i^m\}_{i=1}^c$, а строки соответствуют последовательностям представлений таких наборов при фиксированном m и $l = 1, \ldots, L$. В следующем разделе рассматриваются три алгоритма поиска решения (8) с использованием сети эталонов (9) и приводятся сравнительные оценки вычислительной сложности этих алгоритмов.

Алгоритмы поиска решения и оценки сложности

Поиск решения (8) с помощью ES-алгоритма выполняется на множестве $\hat{\mathbf{B}}^{LM}$ в сети эталонов (9). Два рассматриваемых GS-алгоритма используют для поиска решения последовательности множеств $\hat{\mathbf{B}}^{1M}, \ldots, \hat{\mathbf{B}}^{lM}, \ldots, \hat{\mathbf{B}}^{LM}$ и $\hat{\mathbf{B}}^{L1}, \ldots, \hat{\mathbf{B}}^{Lm}, \ldots, \hat{\mathbf{B}}^{LM}$, которые соответствуют M-й строке и L-му столбцу матрицы (9). Эти алгоритмы базируются на процедурах сужения зоны поиска на последовательных уровнях $l = 1, \ldots, L$ для первого алгоритма и $m = 1, \ldots, M$ для второго алгоритма. Стратегии сужения зоны поиска для двух рассматриваемых GS-алгоритмов определяются функциями

$$c_l = \lfloor c 2^{-\alpha(l-1)} \rfloor,\ l = 1, \ldots, L,\ \alpha = (L-1)^{-1} \log(c/c^*) \tag{10}$$

$$c_m = \lfloor c m^{-\beta} \rfloor,\ m = 1, \ldots, M,\ \beta = (\log M)^{-1} \log(c/c^*) \tag{11}$$

с параметром $c^* = 1, 2, \ldots$. В соответствии с (10) и (11) значения c_l и c_m определяют число классов, анализируемых на l-м и m-м уровнях, из которых отбираются c_{l+1} и c_{m+1} классов с наибольшими значениями разделяющих функций $g^l(A|\hat{\mathbf{B}}_i^M)$ и $g^L(A|\hat{\mathbf{B}}_i^m)$ вида (7). Отобранные классы анализируются на следующих уровнях с номерами $l+1$ и $m+1$

соответственно. На последних L-м и M-м уровнях вычисляется решение вида (8) по c^* классам, отобранным на предыдущих уровнях.

Вычислительная сложность решающих алгоритмов определяется числом пар соответственных примитивов в представлениях эталонов и представлении предъявляемого объекта, которые участвуют в вычислении разделяющих функций. Учитывая древовидность структур представления объектов и ансамблей наборов эталонов, реализация l-го уровня α-стратегии (10) требует просмотра 2^l пар примитивов на один эталон (вершин бинарных деревьев), а реализация m-го уровня β-стратегии (11) требует сравнения объекта с единственным эталоном в каждом классе при $m = 1$ и сравнения объекта с двумя эталонами в каждом классе при $m = 2, \ldots, M$. С учетом последнего замечания в следующих утверждениях сформулированы оценки вычислительной сложности GS-алгоритмов на основе α и β стратегий.

Утверждение 3. *Для фиксированных значений $M \geqslant 1$, $c^* \geqslant 1$ и $\alpha = \frac{\log(c/c^*)}{L-1} \geqslant 1$, GS-алгоритм поиска решения, использующий α -стратегию* (10), *обеспечивает вычислительную сложность*

$$C_\alpha^{\mathrm{GS}} = M \sum_{l=1}^{L} c_l 2^l \leqslant 2McL \tag{12}$$

При $c \to \infty$ полученная оценка дает $C_\alpha^{\mathrm{GS}} = O(c \log c)$.

Утверждение 4. *Для фиксированных значений $L \geqslant 1$, $c^* \geqslant 1$ и $\beta = \frac{\log(c/c^*)}{\log M} \geqslant 1$, GS-алгоритм поиска решения, использующий β-стратегию* (11), *обеспечивает вычислительную сложность*

$$C_\beta^{\mathrm{GS}} = 2(2^L - 1)(c_1 + 2\sum_{m=2}^{M} c_m) \leqslant 2(2^L - 1)(1 + 2\sum_{m=2}^{M} m^{-1})c \tag{13}$$

При $M \to \infty$ и, следовательно, при $c \to \infty$ полученная оценка дает

$$C_\beta^{\mathrm{GS}} \leqslant 2(2^L - 1)(2 \ln M + 2\gamma - 1)c = O(c \log c)$$

где $\gamma \approx 0.577$ – константа Эйлера [7].

Для сравнения вычислительная сложность ES алгоритма имеет вид

$$C^{\mathrm{ES}} = 2(2^L - 1)Mc \tag{14}$$

С учетом ограничений на параметры M, L, c, принятых в утверждениях 3 и 4, оценка (14) дает асимптотику $C^{\mathrm{ES}} = O(c^2)$ при $c \to \infty$. Из приведенных оценок следует, что при $\alpha = 1$, $\beta = 1$ и $c \to \infty$ иерархические алгоритмы обеспечивают вычислительный выигрыш $C^{\mathrm{ES}}/C^{\mathrm{GS}} = \Omega(c/\log c)$. Для ES алгоритма и GS алгоритма на основе α-стратегии этот результат получен в работе [8].

Экспериментальные оценки эффективности классификации

Повышение быстродействия классификатора за счет вычислительного выигрыша иерархических решающих алгоритмов (GS алгоритмов) по сравнению с переборным решающим алгоритмом (ES алгоритмом) оправдано при сопоставимых вероятностях ошибки классификации, т.е. при $\varepsilon^{\mathrm{ES}} \approx \varepsilon^{\mathrm{GS}}$. В этом разделе приводятся численные оценки долей ошибок $\varepsilon_\alpha^{\mathrm{GS}}(L, M)$, $\varepsilon^{\mathrm{ES}}(L, M)$ и вычислительных сложностей $C_\alpha^{\mathrm{GS}}(L, M)$, $C^{\mathrm{ES}}(L, M)$ при

Таблица 1: Вычислительная сложность и средняя доля ошибок для переборного ES алгоритма и иерархического GS алгоритма на основе α-стратегии при $M = 20$

L	1	2	3	4	5	6	7	8	9	10
$C_\alpha^{\mathrm{GS}}(L,M)$	3600	3760	4960	7360	11360	18160	30320	50320	87920	162720
$C^{\mathrm{ES}}(L,M)$	3600	10800	25200	54000	111600	226800	457200	918000	1839600	3682800
$\varepsilon_\alpha^{\mathrm{GS}}(L,M)$	0.1531	0.0698	0.0377	0.0257	0.0182	0.0145	0.0123	0.0109	0.0101	0.0096
$\varepsilon^{\mathrm{ES}}(L,M)$	0.1533	0.0687	0.0357	0.0247	0.0155	0.0122	0.0099	0.0083	0.0080	0.0078

значениях параметров $M = 20$, $L = 1, \ldots, 10$ и $c^* = 2$, и аналогичные оценки $\varepsilon_\beta^{\mathrm{GS}}(L,M)$, $\varepsilon^{\mathrm{ES}}(L,M)$, и $C_\beta^{\mathrm{GS}}(L,M)$, $C^{\mathrm{ES}}(L,M)$, при $L = 10$, $M = 1, \ldots, 20$ и $c^* = 2$. По найденным параметрическим оценкам получены численные зависимости $C_\alpha^{\mathrm{GS}}(\varepsilon)$, $C_\beta^{\mathrm{GS}}(\varepsilon)$, и $C^{\mathrm{ES}}(\varepsilon)$.

Эксперименты проводились с источником полутоновых изображений (8 бит/пиксель), включающих подписи (40 классов по 40 изображений, жесты руки (25 классов по 40 изображений и лица (25 классов по 40 изображений). Общее число классов $c = 90$, общее число изображений (объектов) равно 3600. Вычисление функций сложности выполнялось с использованием оценок (12), (13) и (14).

Для вычисления средних долей ошибок применялась схема кросс-тестирования на основе 100 кратного разбиения множества изображений источника на 2 части (100 times 2 fold cross-validation) [9]. На каждом разбиении формировались обучающая и тестовая выборки. Первая использовалась для получения оценок вероятности ошибки на этапе обучения $\varepsilon_k^{\mathrm{LOO}}(L,M)$, $k = 1,2,3$ методом leave-one-out [9] по каждой из трех компонент меры (4); вторая – для вычисления доли ошибок тестирования $\varepsilon(L,M)$ по каждому из трех рассматриваемых решающих алгоритмов. В качестве оценок весовых коэффициентов в мере (4) выбирались нормированные величины

$$\omega_k = \frac{\log \varepsilon_k^{\mathrm{LOO}}(L,M)}{\sum\limits_{k=1}^{3} \log \varepsilon_k^{\mathrm{LOO}}(L,M)}, \; k = 1,2,3.$$

Ошибки тестирования $\varepsilon(L,M)$ усреднялись по 100 разбиениям для получения соответствующих оценок $\varepsilon_\alpha^{\mathrm{GS}}(L,M)$, $\varepsilon_\beta^{\mathrm{GS}}(L,M)$ и $\varepsilon^{\mathrm{ES}}(L,M)$. Эксперименты выполнены на одном ядре процессора Intel Core i3 с тактовой частотой 3GHz, на компьютере с оперативной памятью объемом 3Gb и операционной системой MS Windows 7.

Вычислительная сложность и средняя доля ошибок тестирования приведены в табл. 1 и 2. Построенные по этим таблицам графики зависимостей вычислительной сложности от средней доли ошибок даны на рис. 3.

Из данных табл. 1 и графиков на рис. 3 следует, что при $L = 1$ реализуется наибольшая доля ошибок, равная 0.1533, и в этом случае α-стратегия не имеет вычислительного выигрыша относительно стратегии полного перебора; при $L = 10$ доля ошибок достигает наименьшего значения 0.0078, при котором вычислительный выигрыш не менее 22.6. Аналогично из табл. 2 и рис. 3 следует, что при $M = 1$ доля ошибок максимальна и равна 0.1520 и в этом случае β-стратегия не дает вычислительного выигрыша относительно стратегии перебора, а при $M = 20$ доля ошибок достигает наименьшего значения 0.0079, при котором вычислительный выигрыш оценивается снизу величиной 4.6.

Заключение

Исследованы соотношения вычислительной сложности от вероятности ошибки для алгоритмов распознавания двумерных объектов, заданных изображениями. Предложены

Таблица 2: Вычислительная сложность и средняя доля ошибок для переборного ES алгоритма и иерархического GS алгоритма на основе β-стратегии при $L = 10$

M	1	2	3	4	5	6	7	8	9	10
$C_\beta^{\mathrm{GS}}(L,M)$	184140	192324	225060	261888	302808	347820	380556	417384	450120	491040
$C^{\mathrm{ES}}(L,M)$	184140	368280	552420	736560	920700	1104840	1288980	1473120	1657260	1841400
$\varepsilon_\beta^{\mathrm{GS}}(L,M)$	0.1558	0.1306	0.0954	0.0798	0.0541	0.0361	0.0277	0.0194	0.0175	0.0155
$\varepsilon^{\mathrm{ES}}(L,M)$	0.1520	0.0497	0.0336	0.0249	0.0196	0.0161	0.0141	0.0124	0.0113	0.0104
M	11	12	13	14	15	16	17	18	19	20
$C_\beta^{\mathrm{GS}}(L,M)$	519684	560604	585156	609708	646536	675180	712008	732468	765204	789756
$C^{\mathrm{ES}}(L,M)$	2025540	2209680	2393820	2577960	2762100	2946240	3130380	3314520	3498660	3682800
$\varepsilon_\beta^{\mathrm{GS}}(L,M)$	0.0141	0.0132	0.0126	0.0122	0.0114	0.0111	0.0107	0.0106	0.0103	0.0103
$\varepsilon^{\mathrm{ES}}(L,M)$	0.0098	0.0093	0.0089	0.0085	0.0083	0.0082	0.0081	0.0080	0.0079	0.0079

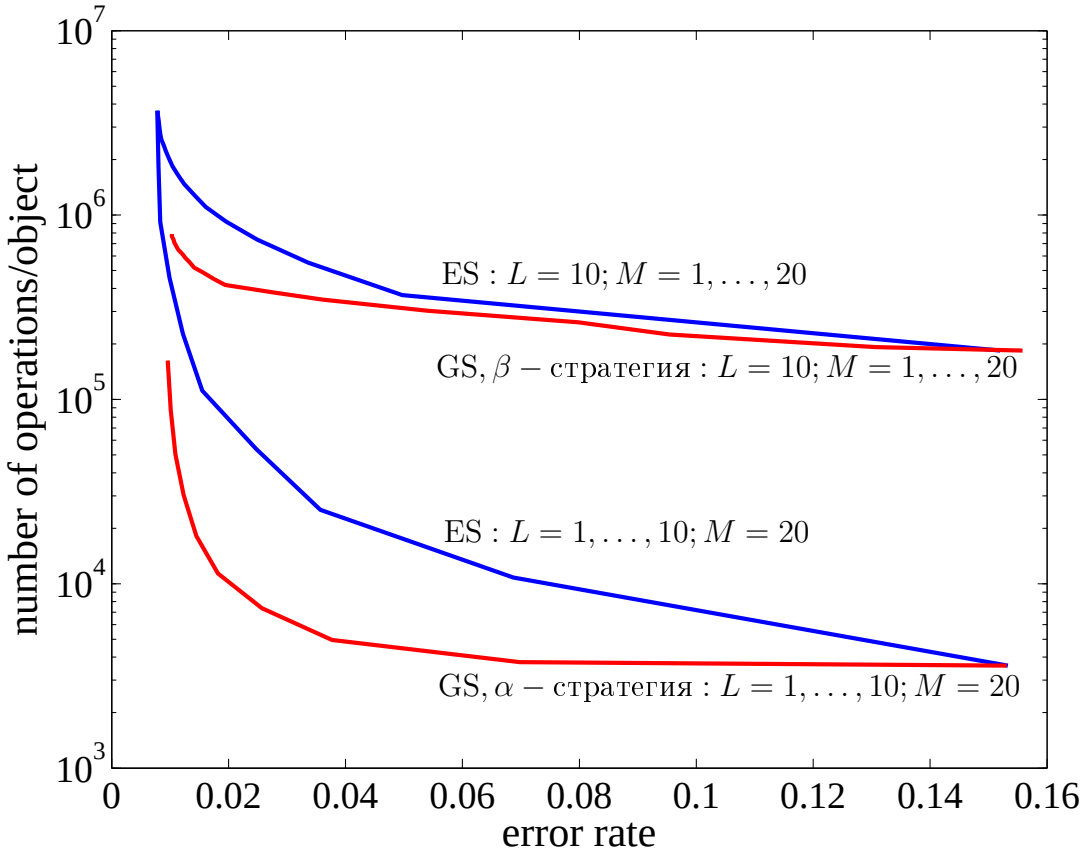


Рис. 3: Графики зависимостей вычислительной сложности (в операциях на объект) от средней доли ошибок для ES и GS алгоритмов

древовидные структуры для представления объектов и формирования ансамблей эталонов. Используя многоуровневую сеть эталонов, рассмотрены иерархические алгоритмы поиска решения по критерию ближайшего эталона и получены оценки вычислительной сложности этих алгоритмов. Рассмотренные иерархические алгоритмы базируются на двух стратегиях последовательного сужения зоны поиска решения в многоуровневой базе эталонов. При большом числе классов оценен вычислительный выигрыш предложенных иерархических алгоритмов относительно алгоритма полного перебора эталонов. Для переборного и иерархических решающих алгоритмов найдены экспериментальные зависи-

мости вычислительной сложности от доли ошибок распознавания. Указанные зависимости получены для составного источника, включающего изображения подписей, жестов руки и лиц.

В перспективе планируется построение решающего алгоритма на основе объединения рассмотренных стратегий сужения зоны поиска, и получение для него оценок сложности и вероятности ошибки. Кроме того, необходимо исследовать решающие алгоритмы с разделяющими функциями, построенными на других ядрах потенциального типа.

Литература

[1] *Dobrushin R. L., Tsybakov B. S.* Information transmission with additional noise // IRE Transactions on Information Theory. 1962. Vol. 8, No. 5. Pp. 293–304.

[2] *Rosenfeld A.* Quadtrees and Pyramids for Pattern Recognition and Image Analysis // 5th International Conference on Pattern Recognition Proceedings. 1980. Pp. 802–811.

[3] *Torsello A.* Matching hierarchical structures for shape recognition. PhD Thesis. University of York, 2004. 197 p.

[4] *Elfiky N. M., Khan F. S., Weijer J., Gonzalez J.* Discriminative compact pyramids for object and scene recognition // Pattern Recognition. 2012. Vol. 45, No. 4. Pp. 1627–1636.

[5] *Ganebnykh S. N., Lange M. M.* Classification of 2D Grayscale objects in a space of multiresolution representations // Pattern Recognition and Image Analysis. 2009. Vol. 19, No. 4. Pp. 591–602.

[6] *Ganebnykh S. N., Lange M. M.* Metric classifier using multilevel network of templates // Pattern Recognition and Image Analysis. 2012. Vol. 22, No. 2. Pp. 265–277.

[7] *Грехем Р. Л., Кнут Д. Э., Паташник О.* Конкретная математика. М.: Вильямс, 2010. 784 с.

[8] *Ланге М. М., Ганебных С. Н.* Классификация изображений в пространстве универсальных представлений с многоуровневым разрешением // Доклады 9-й Международной конференции «Интеллектуализация обработки информации» (ИОИ-2012). 2012. С. 464–467.

[9] *Bishop C. M.* Pattern recognition and machine learning. Springer, 2006. 738 p.

Computable Combinatorial Overfitting Bounds*

***K. V. Vorontsov*[1], *A. I. Frey*[2], *E. A. Sokolov*[3]**
voron@forecsys.ru, oleksandr.frei@gmail.com, sokolov.evg@gmail.com
[1]Dorodnitsyn Computing Centre, Russian Academy of Sciences; [2]Moscow Institute of Physics and Technology; [3]Moscow State University

Computable combinatorial data dependent on generalization bounds are studied. This approach is based on simplified probabilistic assumptions: it is assumed that the instance space is finite, the labeling function is deterministic, and the loss function is binary. A random walk across a set of linear classifiers with low error rate is used to compute the bound efficiently. The experimental evidence to confirm that this approach leads to practical overfitting bounds in classification tasks is provided.

Keywords: *probability of overfitting, empirical risk, combinatorial bounds, similaritiy between classifiers, cross-validation procedure.*

Вычислимые комбинаторные оценки вероятности переобучения*

***Воронцов К. В.*[1], *Фрей А. И.*[2], *Соколов Е. А.*[3]**
[1]Москва, Вычислительный Центр РАН; [2]Московский Физико-Технический Институт; [3]Московский Государственный Университет

В данной статье изучаются комбинаторные оценки обобщающей способности, вычислимые по обучающей выборке. Эти оценки основаны на упрощенной вероятностной модели, в которой рассматривается лишь конечная генеральная совокупность объектов и бинарная функция потерь. Для линейных классификаторов предлагается новый эффективный метод вычисления комбинаторных оценок, использующий случайные блужданий по множеству классификаторов с низким числом ошибок. В заключении приводится экспериментальное обоснование предлагаемого метода.

Ключевые слова: *вероятность переобучения, эмпирический риск, оценка вероятности переобучения, сходство алгоритмов, скользящий контроль.*

1 Introduction

Accurate bounding of overfitting is an active area of research starting with the pioneer work [1]. A widely adapted approach is based on a probabilistic framework where an instance space $\mathbb{X}$ (usually, of an infinite cardinality) is equipped with an unknown probability distribution. Consider an independent and identically distributed (i.i.d.) sample $X = \{x_1, \ldots, x_\ell\}$ from $\mathbb{X}$, a set A of all feasible classifiers (for example, all linear classifiers in the original feature space), and a learning algorithm μ which selects a specific classifier $a = \mu X$ from the set A based on the observed sample X. The goal of generalization bounds is to predict an average performance of the classifier a on the whole $\mathbb{X}$. The most generalization bounds are derived from various concentration inequalities [2] and take into account the dimensionality of A (such as

*This work was supported by the Russian Foundation for Basic Research (projects Nos. 11-07-00480 and 12-07-33099-mol-a-ved) and by the program "Algebraic and Combinatorial Methods in Mathematical Cybernetics and New Generation Information Systems" of the Department of Mathematical Sciences of the Russian Academy of Sciences.

Vapnik–Chervonenko (VC) dimension, fat-shattering dimension, etc.), properties of the learning algorithm μ (such as the local properties of empirical risk minimization [3]), and information drawn from the observed sample X (such as the normalized margin in margin-based bounds [4, 5]). Generalization bounds can be useful in structural risk minimization and in model selection, and, hope, some time in future they could replace costly cross-validation procedures.

Despite recent significant improvements [6], there is still a big gap between theory and practice. Even the latest PAC (Probably approximately correct) Bayesian bounds [7] vastly overestimate overfitting, especially when the cardinality of an instance space is small. Another difficulty is that intermediate bounds are usually expressed in terms of unobservable quantities and that makes impossible to measure and compare the factors of overestimation. Finally, many papers lack experimental evaluation. As a result, from practical perspective, the existing bounds are not well suitable for prediction and control of overfitting.

The present authors believe that a simplified probabilistic framework is quite sufficient for obtaining practical overfitting bounds. In this paper, it is assumed that the instance space $\mathbb{X}$ is finite, and for each object $x \in \mathbb{X}$ and classifier $a \in A$, there exists a deterministic binary loss $I(a, x) \in \{0, 1\}$ associated with classification of x as $a(x)$. Let assume that all partitions of the set $\mathbb{X}$ into an observed training sample X of size ℓ and a hidden test sample $\bar{X} = \mathbb{X} \setminus X$ of size k can occur with equal probability. Then, the overfitting probability can be defined by a purely combinatorial formula [8]:

$$Q_\varepsilon(\mu, \mathbb{X}) = \mathsf{P}\big[\nu(\mu(X), \bar{X}) - \nu(\mu(X), X) \geqslant \varepsilon\big] \tag{1}$$

where μ is the learning algorithm; $\nu(a, X)$ is the error rate of a classifier $a \in A$ on a sample X; and the square brackets denote a transformation of a logical value into a numerical one: [true] = = 1, [false] = 0. A definition similar to (1) first appears in [9] for a specific case of $k = 1$ and then in [10] (this time for an arbitrary k, but with significant notation differences). This definition closely resembles the procedure of complete cross-validation, which is known to provide sharp estimates of performance of a learning algorithm on data yet unknown during learning phase.

Definition (1) is not guarantied to be an upper bound for new objects beyond $\mathbb{X}$ (not even up to some probability). In this paper, the authors do not discuss how to mitigate this problem. It is just assumed that the lack of guaranties is acceptable in the same way as people normally accept results of a fare 10-fold cross-validation in experimental evaluations.

Within (1), an empirical risk minimization $\mu X = \arg\min_{a \in A} \nu(a, X)$ is used as a learning algorithm. This requires an explicit representation of the set of classifiers A, which might be of enormous cardinality (10^9 and higher) in real applications. In this paper, a special case of linear classifiers is studied and an efficient algorithm that samples a small set of classifiers (up to 10^4) to recover accurate overfitting bound (1) is presented. Also note that a direct computation of (1) is intractable because it involves a sum across all $\mathbb{C}^\ell_{\ell+k}$ subsets $X \subset \mathbb{X}$. For empirical risk minimization, efficient upper bounds on (1), obtained in [11] are used, and compared with Monte-Carlo estimate of (1).

Overfitting of logistic regression in experiments is studied on 15 datasets from the UCI repository [12]. The results confirm that the present approach provides sharp estimates of overfitting, which correlate with the actual overfitting, recovers a correct shape of the learning curves, and outperform the state-of-the-art of PAC-Bayesian bounds.

The rest of this paper is organized as follows. Section 2 contains a brief review of combinatorial bounds on (1). Section 3 describes the algorithm of sampling a representative set of linear classifiers. Section 4 provides experimental results and Section 5 concludes the paper.

2 Background

Let $\mathbb{X} = \{x_1, \ldots, x_L\}$ be a finite instance space and A be a set of classifiers. By $I\colon A \times \times X \to \{0,1\}$ denote a binary loss function such that $I(a,x) = 1$ if a classifier a produces an error on an object x. For further consideratio, there is no need to specify what is "classifier." Particularly, a regression function can also be a "classifier" if a binary loss function is used.

The binary vector $(a_i) \equiv \big(I(a,x_i)\big)_{i=1}^{L}$ of size L is called an *error vector* of the classifier a. Assume that all classifiers from A have pairwise different error vectors. The number of errors of a classifier a on a sample $X \subseteq \mathbb{X}$ is defined as $n(a,X) = \sum_{x\in X} I(a,X)$. The error rate is defined as $\nu(a,X) = (1/|X|)n(a,X)$. The subset $A_m = \{a \in A\colon n(a,\mathbb{X}) = m\}$ is called *m-layer* of classifiers.

A *learning algorithm* is a mapping $\mu\colon 2^{\mathbb{X}} \to A$ that takes a training sample $X \subseteq \mathbb{X}$ and gives a classifier $\mu X \in A$. The learning algorithm μ is called *empirical risk minimization* (ERM) whenever for all $X \in \mathbb{X}$ it satisfies $\mu X \in A(X)$ where

$$A(X) \equiv \operatorname{Arg}\min_{a\in A} n(a,X).$$

The choice of a classifier that minimizes empirical risk may be ambiguous because of discreteness of the function $n(a,X)$. An ERM algorithm μ is said to be *pessimistic* if

$$\mu X \in \operatorname{Arg}\max_{a\in A(X)} n(a,\bar{X}).$$

The pessimistic ERM cannot be implemented in practice because it looks into a hidden part of data $\bar{X}$ unknown at the learning stage. Nevertheless, pessimistic ERM is a very useful theoretical concept because it gives tight upper bounds of overfitting probability for any ERM.

Permutational probability. By $[\mathbb{X}]^\ell$ denote a set of all $\mathbb{C}_L^\ell = L!/(\ell!(L-\ell)!)$ samples $X \subset \mathbb{X}$ of size ℓ. Define a probability operator P and an expectation operator E for a predicate $\varphi\colon [\mathbb{X}]^\ell \to \{0,1\}$ and a real function $\psi\colon [\mathbb{X}]^\ell \to \mathbb{R}$:

$$\mathsf{P}\,\varphi = {\mathbb{C}_L^\ell}^{-1} \sum_{X\in[\mathbb{X}]^\ell} \varphi(X); \qquad \mathsf{E}\psi = {\mathbb{C}_L^\ell}^{-1} \sum_{X\in[\mathbb{X}]^\ell} \psi(X).$$

If the *discrepancy* $\delta(a,X) = \nu(a,\bar{X}) - \nu(a,X)$ is greater than a given nonnegative threshold ε, then the classifier $a = \mu X$ is said to be *overfitted*. The goal is to estimate the *probability of overfitting*:

$$Q_\varepsilon(\mu,\mathbb{X}) = \mathsf{P}\big[\delta(\mu,X) \geqslant \varepsilon\big].$$

where $\delta(\mu,X) = \delta(\mu X,X)$ for short.

The *inversion* of an upper bound $Q_\varepsilon \leqslant \eta(\varepsilon)$ is an inequality $\nu(\mu X,\bar{X}) - \nu(\mu X,X) \leqslant \varepsilon(\eta)$ that holds with probability at least $1-\eta$ where $\varepsilon(\eta)$ is the inverse function for $\eta(\varepsilon)$. The *median of an upper bound* $Q_\varepsilon \leqslant \eta(\varepsilon)$ is the inversion at $\eta = 1/2$.

Average train and test errors are defined as follows:

$$\nu_\ell(\mu,\mathbb{X}) = \mathsf{E}\nu(\mu X,X); \tag{2}$$

$$\bar{\nu}_\ell(\mu,\mathbb{X}) = \mathsf{E}\nu(\mu X,\bar{X}). \tag{3}$$

Hypergeometric distribution. For a classifier a such that $m = n(a,\mathbb{X})$, the probability to have s errors on a sample X is given by a hypergeometric function:

$$\mathsf{P}[n(a,X) = s] = \mathbb{C}_m^s \mathbb{C}_{L-m}^{\ell-s} {\mathbb{C}_L^\ell}^{-1} \equiv h_L^{\ell,m}(s)$$

where argument s runs from $s_0 = \max\{0, m-k\}$ to $s_1 = \min\{m, \ell\}$, and parameter m takes values $0, \ldots, L$. It is assumed that $\mathbb{C}_m^s = h_L^{\ell,m}(s) = 0$ for all other integers m, s.

Define the hypergeometric cumulative distribution function (left tail of the distribution):

$$H_L^{\ell,m}(z) = \sum_{s=s_0}^{\lfloor z \rfloor} h_L^{\ell,m}(s).$$

Consider a set $A = \{a\}$ containing a fixed classifier so that $\mu X = a$ for any X. Then the probability of overfitting Q_ε transforms into the probability of large deviation between error rates on two samples X and $\bar{X}$. If the number of errors $n(a, \mathbb{X})$ is known, then an exact Q_ε bound can be obtained.

Theorem 1 (FC-bound [11]). *For a fixed classifier (FC) a such that $m = n(a, \mathbb{X})$, any set $\mathbb{X}$, and any $\varepsilon \in [0, 1]$, the probability of overfitting is given by the left tail of the hypergeometric distribution:*

$$Q_\varepsilon(a, \mathbb{X}) = H_L^{\ell,m}\left(\tfrac{\ell}{L}(m - \varepsilon k)\right). \tag{4}$$

The hypergeometric distribution plays a fundamental role in combinatorial bounds. Together with union bound, Eq. (4) provides an upper estimate of $Q_\varepsilon(\mu, \mathbb{X})$ that holds for any learning algorithm μ.

Theorem 2 (VC-type bound [11]). *For any set $\mathbb{X}$, any learning algorithm μ, and any $\varepsilon \in [0, 1]$, the probability of overfitting is bounded by the sum of FC-bounds over the set A:*

$$Q_\varepsilon(\mu, \mathbb{X}) \leqslant \mathsf{P}\big[\max_{a \in A} \delta(a, X) \geqslant \varepsilon\big] \leqslant \sum_{a \in A} H_L^{\ell,m}\left(\tfrac{\ell}{L}(m - \varepsilon k)\right), \quad m = n(a, \mathbb{X}). \tag{5}$$

There are two reasons for looseness of (5). First, most classifiers in A are bad and should have vanishing probability to be obtained as a result of learning. Nevertheless, the uniform deviation bound ignores the learning algorithm μ. Second, similar classifiers share their contribution, which is ignored by union bound. Better bound should account for actual learning algorithm and similarity between classifiers.

Splitting and connectivity bounds. Define an order relation on classifiers $a \leqslant b$ as a natural order over their error vectors: $a_i \leqslant b_i$ for all $i = 1, \ldots, L$. Define a metric on classifiers as a Hamming distance between error vectors: $\rho(a, b) = \sum_{i=1}^{L} |a_i - b_i|$.

Theorem 3 (Splitting and connectivity (SC) bound [11]). *If learning algorithm μ is pessimistic ERM, then for any $\varepsilon \in [0, 1]$, the probability of overfitting is bounded by the weighted sum of FC-bounds over the set A:*

$$Q_\varepsilon(\mu, \mathbb{X}) \leqslant \sum_{a \in A} \mathbb{C}_{L-q-r}^{\ell-q} \mathbb{C}_L^{\ell\,-1} H_{L-q-r}^{\ell-q,\,m-r}\left(\tfrac{\ell}{L}(m - \varepsilon k)\right). \tag{6}$$

Here, $m = n(a, \mathbb{X})$; $q = q(a)$ is the upper connectivity; and $r = r(a)$ is the inferiority of a classifier a:

$$q(a) = \#\{b \in A \colon a < b \text{ and } \rho(a, b) = 1\};$$
$$r(a) = \#\{x \in \mathbb{X} \colon I(a, x) = 1 \text{ and } \exists\, b \in A \text{ such that } b \leqslant a \text{ and } I(x, b) = 0\}$$

where for any set S notation $\#S$ stands for cardinality of S.

Splitting and connectivity bound (6) turns into VC-type bound (5) when all $q(a)$ and $r(a)$ are set to zeros.

The weight $P_a = \mathbb{C}_{L-q-r}^{\ell-q}\mathbb{C}_L^{\ell}{}^{-1}$ in sum (6) is an upper bound on the probability $\mathsf{P}[\mu X = a]$ to get a given classifier a as a result of learning. This quantity decreases exponentially as the connectivity $q(a)$ or the inferiority $r(a)$ increase. This implies that approximate calculation of $Q_\varepsilon(\mu, \mathbb{X})$ requires knowledge not about the full set A, but only about few bottom layers of A. This fact motivates an algorithm presented in the next section.

3 Sampling Linear Classifiers

One has to deal with the set of all classifiers A to calculate bounds (5), (6), or to estimate $Q_\varepsilon(\mu, \mathbb{X})$ directly from definition (1). In this section, an efficient algorithm which samples a small set of classifiers (about 10^4) sufficient to recover accurate overfitting bound is described.

Consider binary classification problem with labels $y_i \in \{-1, +1\}$, $i = 1, \ldots, L$, assigned to objects $x_i \in \mathbb{X} \subset \mathbb{R}^d$, respectively. Consider a set of unbiased linear classifiers $a(x; w) = \operatorname{sign}\langle w, x\rangle$ where $w \in \mathbb{R}^d$ is a real vector of weights. A pair of classifiers (w_1, w_2) is called *neighbors* if their classification differs only by one object: $x \in \mathbb{X}$ such that $\operatorname{sign}(\langle w_1, x\rangle) \operatorname{sign}(\langle w_2, x\rangle) = -1$.

The immediate goal is to find all or some of neighbors of a given classifier w_0. Then, this procedure will be used to organize random walk on the graph $G = (A, E)$ where vertices correspond to classifiers in A and edges connect neighbor classifiers.

Finding neighbor classifiers along specific direction. *Dual transformation* D maps a point $x \in \mathbb{R}^d$ into hyperplane $D(x) = \{w \in \mathbb{R}^d \colon \langle w, x\rangle = 0\}$ and a hyperplane $h = \{x \in \mathbb{R}^d \colon \langle w, x\rangle = 0\}$ into point $D(h) = w$. Applying dual transformation D to finite set of points $\mathbb{X} \subset \mathbb{R}^d$ produces a set of hyperplanes $\mathbb{H} \equiv \{D(x_i)\}_{i=1}^L$. Each hyperplane $h_i \in \mathbb{H}$ divides $\mathbb{R}^d$ into two half-spaces:

$$h_i^+ = \{w \in \mathbb{R}^d \colon \operatorname{sign}\langle w, x_i\rangle = y_i\};$$
$$h_i^- = \{w \in \mathbb{R}^d \colon \operatorname{sign}\langle w, x_i\rangle = -y_i\}.$$

These half-spaces h_i^+ and h_i^- correspond to linear classifiers giving correct and incorrect answer on x_i, respectively. So, to find all classifiers with given error vector $I = (I_i)_{i=1}^L$, $I_i \in \{+, -\}$ where "+" corresponds to correct answer and "−" corresponds to incorrect, let just find the intersection of half-spaces $\bigcap_{i=1}^L h_i^{I_i}$. This intersection contains all linear classifiers with error vector I (and only them). So, a set of hyperplanes $\mathbb{H}$ dissects $\mathbb{R}^d$ into convex polytopes called *cells*, and the partition itself is called *an arrangement of hyperplanes* [13]. It can be shown that finding neighbors of classifier $w_0 \in \mathbb{R}^d$ is equivalent to finding cells adjacent to the cell of w_0 in arrangement $\mathbb{H}$.

In order to find a neighbor of the classifier w_0, let select an arbitrary vector $u \in \mathbb{R}^d$ and consider a parametric set of classifiers $\{w_0 + tu \colon t \geqslant 0\}$. This set corresponds to a ray in the space of classifiers which starts from w_0 and goes along the direction of u. An intersection of this ray with hyperplane $h_i \in \mathbb{H}$ is defined by condition $\langle w_0 + tu, x_i\rangle = 0$, e. g., for $t_i = -\langle w_0, x_i\rangle / \langle u, x_i\rangle$. Let $t_{(1)}$ and $t_{(2)}$ be the first and the second smallest positive values from $\{t_i\}$, $i = 1, \ldots, L$. Whenever $t_{(1)} \neq t_{(2)}$, one can conclude that $w' = w_0 + (t_{(1)} + t_{(2)})u/2$ defines an adjacent classifier along direction u.

Random walk on classifiers graph. Techniques of random walk [14, 15, 16] provide common approach to sample vertices from huge graphs. They are based on stationary distributions of Markov chains and have nice properties when the sample is large. The goal is to get

Algorithm 1 Random walk on classifiers graph

Require: starting point w_0; sample $\mathbb{X} \subset \mathbb{R}^d$; integer parameters N, m, n; float parameter $p \in (0, 1]$;
Ensure: set of classifiers A with unique error vectors.

Initialize concurrent random walk: $v_i = w_0$, $i = 1, \ldots, N$;
Create set $A := \varnothing$;
while $A.size() < n$
  for all $i \in 1, \ldots, N$
    Find neighbor v_i' of v_i along random direction $u \in \mathbb{R}^d$;
    if $n(v_i', \mathbb{X}) > n(v_i, \mathbb{X})$ **then**
      with probability $(1 - p)$ **continue**;
    else if $n(v_i', \mathbb{X}) > n(w_0, \mathbb{X}) + m$ **then**
      continue;
    $v_i = v_i'$;
    A.add(v_i);
return A

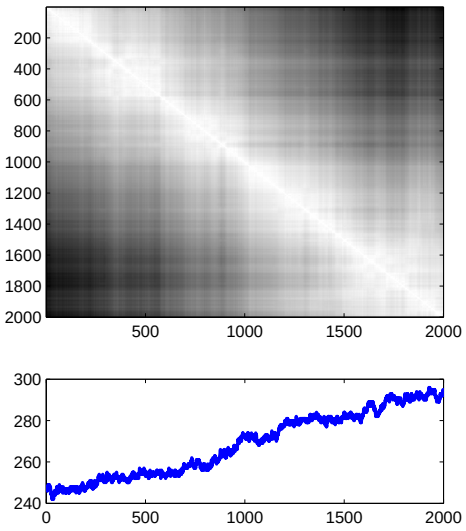


Fig. 1: Map of hamming distances between classifiers (top chart) and error profile (bottom chart) produced by a simple random walk

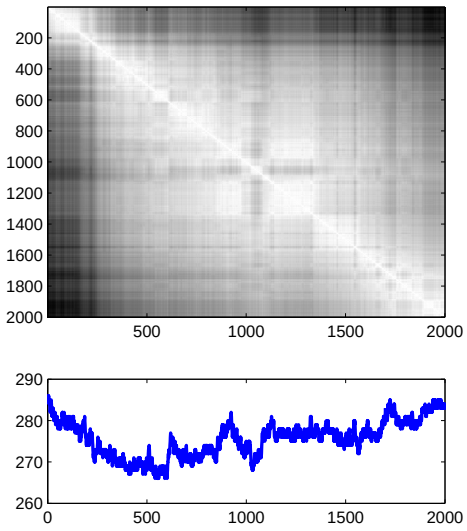


Fig. 2: Map of hamming distances between classifiers (top chart) and error profile (bottom chart) produced by random walk where a step to upper vertex is made with probability 0.5

a small sample sufficient to estimate overfitting from (1), (5), or (6). In this paragraph, it will be discussed how to organize such random walk on A based on procedure that finds a random neighbor for $w \in A$.

The algorithm is given in listing 1. It is controlled by the desired number of classifiers n, maximal number of layer m, the number of concurrent walks N, and the probability p of transition towards classifier with higher number of errors. The computational complexity of this algorithm is $O(Ldn)$.

To explain the necessity of the parameter p, the results of the simplest random walk with $n = 2000$ iterations are presented in Fig. 1. The bottom chart displays the number of errors $n(v_i, \mathbb{X})$ as a function of step. The upper chart displays a color map of pairwise hamming distances $\rho(v_i, v_j)$ between sampled vertices v_i and v_j. As a starting point, a classifier learned by logistic regression is used. It is natural to expect that it has relatively small number of errors which drifts upwards along random walk. This effect is undesired, because classifiers with high number of errors have too small chance to be selected by learning algorithm.

Figure 2 presents similar result for updated random walk where a step to upper vertex is made with probability $p = 0.5$. This enforces random walk to stay within the lower layers of the graph.

4 Experiment

The goal of the experiment on the benchmark datasets is twofold. First, it was checked whether combinatorial functionals $Q_\varepsilon(\mu, \mathbb{X})$ (1) and $\bar{\nu}_\ell(\mu, \mathbb{X})$ (3) together with algorithm 1 provide an accurate estimates of the overfitting on the holdout testing sample. Second, direct Monte-Carlo estimates of overfitting based on functional (1) were compared with VC-type bound (5), SC-bound (6), and with recent PAC-Bayesian dimension dependent (DD) and dimension independent (DI) margin bounds proposed in [7].

Table 1: Description of datasets

Dataset	#Examples	#Features	Dataset	#Examples	#Features
Sonar	208	60	Statlog	2310	19
Glass	214	9	Wine	4898	11
Liver dis.	345	6	Waveform	5000	21
Ionosphere	351	34	Pageblocks	5473	10
Wdbc	569	30	Optdigits	5620	64
Australian	690	6	Pendigits	10992	16
Pima	768	8	Letter	20000	16
Faults	1941	27			

Fifteen datasets from the UCI repository [12] have been used. If the dataset is a multiclass problem, the data were manually grouped into two classes since the binary classification problem has been studied. For preprocessing, objects with one or more missing features have been eliminated and all features have been normalized into $[0, 1]$ interval. A description of the datasets is given in Table 1 with number of examples after elimination.

In all experiments, the original dataset $\mathbb{X}$ was splitted into a training sample $\mathbb{X}_L$ and a testing sample $\mathbb{X}_K$. The training sample $\mathbb{X}_L$ is used to train a logistic regression and to calculate overfitting bounds. Then, the predictions of the bounds were compared with the actual error rate on $\mathbb{X}_K$.

In the first experiment, the learning curves of logistic regression, where L runs from 5% to 95% of the original dataset size with 5 percent steps were built. For each L, $M = 100$ splits $\mathbb{X} = \mathbb{X}^i_L \cup \mathbb{X}^i_K$, $i = 1, \ldots, M$, were generated and used to get Monte-Carlo estimates of train error rate $\nu_L(\mu_{\text{LR}}, \mathbb{X})$ from (2) and test error rate $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$ from (3) for logistic regression learning algorithm μ_{LR}:

$$\hat{\nu}_L(\mu_{\text{LR}}, \mathbb{X}) = \frac{1}{M}\sum_{i=1}^{M} \nu(\mu_{\text{LR}}\mathbb{X}^i_L, \mathbb{X}^i_L), \qquad \hat{\bar{\nu}}_L(\mu_{\text{LR}}, \mathbb{X}) = \frac{1}{M}\sum_{i=1}^{M} \nu(\mu_{\text{LR}}\mathbb{X}^i_L, \mathbb{X}^i_K).$$

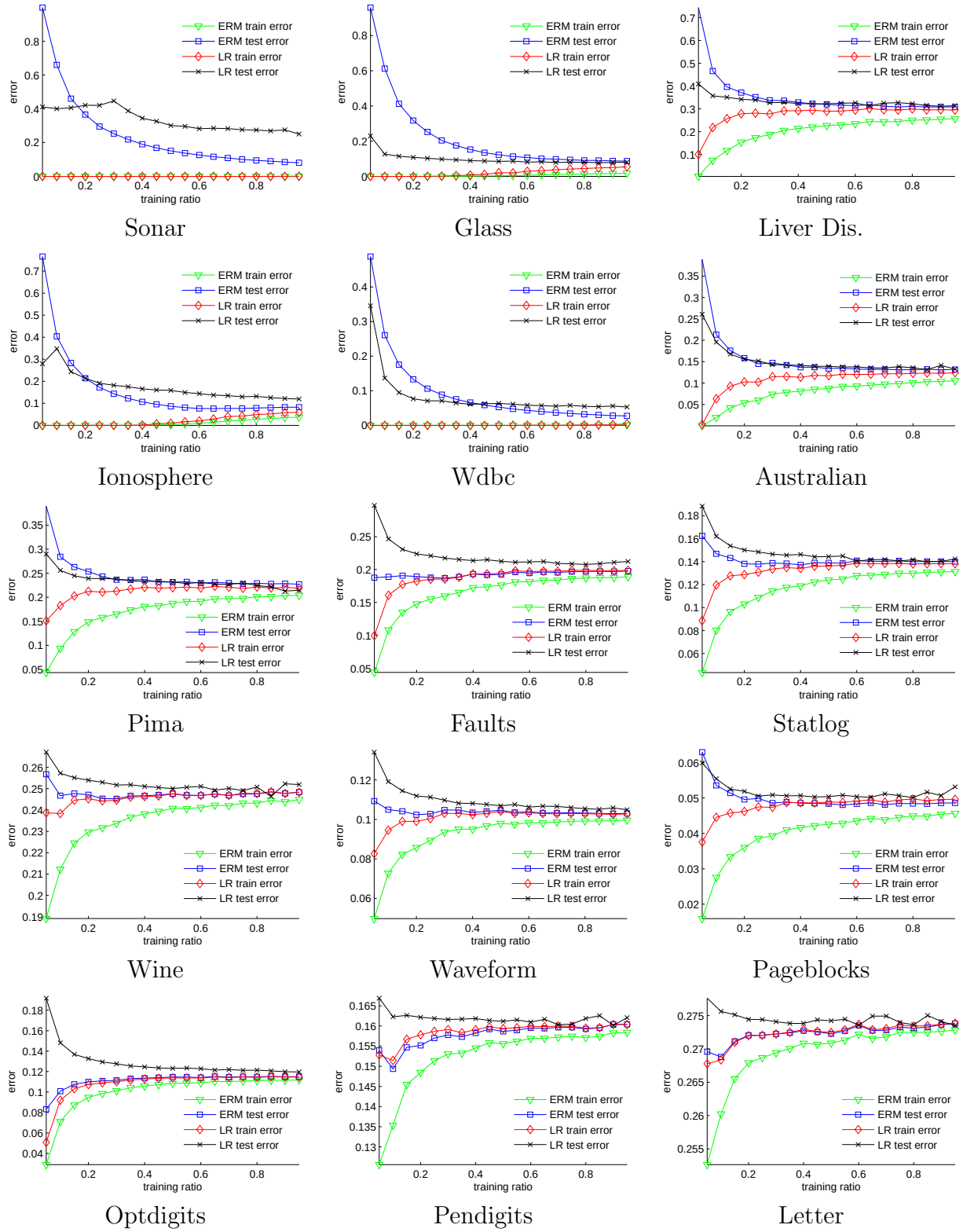


Fig. 3: Learning curves of logistic regression and ERM. The error ratio of logistic regression is estimated by Monte-Carlo method on splits of the original dataset $\mathbb{X} = \mathbb{X}_L \cup \mathbb{X}_K$. The error ratio of ERM is estimated on splits of the training set $\mathbb{X}_L = X_\ell \cup X_k$

After that, for each training sample $\mathbb{X}_L$, the classifiers were used and an average ERM errors were estimated: train error $\nu_\ell(\mu, \mathbb{X}_L)$ and test error $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$ where μ is ERM learning

Table 2: Comparison between real overfitting and various overfitting bounds: TrainErr stands for $\nu_L(\mu_{\text{LR}}, \mathbb{X})$, TestErr for $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$, Overfit is their difference, $\delta_\ell(\mu) \equiv \bar{\nu}_\ell(\mu, \mathbb{X}_L) - \nu_\ell(\mu, \mathbb{X}_L)$

	Monte-Carlo estimates				Generalization bounds			
Task	TrainErr	TestErr	Overfit	$\delta_\ell(\mu)$	VC	SC	PAC DI	PAC DD
Sonar	0.000	0.271	0.271	0.095	0.185	0.119	1.287	1.287
Glass	0.046	0.075	0.029	0.078	0.211	0.140	1.126	0.738
Liver dis.	0.299	0.314	0.015	0.060	0.261	0.209	1.207	1.067
Ionosphere	0.049	0.125	0.077	0.052	0.150	0.112	1.219	1.153
Wdbc	0.001	0.056	0.055	0.032	0.071	0.043	1.174	0.705
Australian	0.122	0.136	0.013	0.030	0.137	0.110	1.146	0.678
Pima	0.220	0.227	0.007	0.028	0.159	0.127	0.971	0.749
Faults	0.198	0.210	0.012	0.010	0.108	0.087	1.110	1.061
Statlog	0.138	0.142	0.005	0.010	0.096	0.082	1.102	0.747
Wine	0.248	0.250	0.002	0.004	0.134	0.109	0.776	0.637
Waveform	0.103	0.105	0.002	0.004	0.099	0.079	0.561	0.354
Pageblocks	0.050	0.050	0.001	0.004	0.073	0.057	0.737	0.186
Optdigits	0.115	0.121	0.006	0.004	0.102	0.084	1.068	0.604
Pendigits	0.160	0.161	0.001	0.002	0.127	0.103	0.774	0.432
Letter	0.274	0.274	0.001	0.001	0.165	0.137	0.818	0.636

algorithm. To sample classifiers on $\mathbb{X}_L$, algorithm 1 was launched with parameters $n = 8\,192$, $N = 64$, $m = 15$, $p = 0.8$, and classifier $\mu_{\text{LR}}\mathbb{X}_L$ was used as a starting point. To estimate $\nu_\ell(\mu, \mathbb{X}_L)$ and $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$, let again compute Monte-Carlo type estimates of definitions (2) and (3) by randomly generating $M' = 4\,096$ splits $\mathbb{X}_L = X_\ell^j \cup X_k^j$, $j = 1, \ldots, M'$, at constant ratio $\frac{\ell}{L} = 0.8$:

$$\hat{\nu}_\ell(\mu, \mathbb{X}_L) = \frac{1}{M'} \sum_{j=1}^{M} \nu(\mu X_\ell^j, X_\ell^j), \qquad \hat{\bar{\nu}}_\ell(\mu, \mathbb{X}_L) = \frac{1}{M'} \sum_{j=1}^{M} \nu(\mu X_\ell^j, X_k^j).$$

These estimates are then averaged over all partitions $\mathbb{X} = \mathbb{X}_L^i \cup \mathbb{X}_K^i$.

The four values (the actual train and test errors of logistic regression $\nu_L(\mu_{\text{LR}}, \mathbb{X})$ and $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$, ERM train error $\nu_\ell(\mu, \mathbb{X}_L)$, and ERM test error $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$) are charted as a functions of a training sample size ratio (Fig. 3) sorted according to sizes of datasets, from the smallest to the largest. Note that ERM test error might be either below or above actual test error rate of logistic regression because μ and μ_{LR} are quite different learning algorithms. However, from charts, one may conclude that $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$, estimated only based on $\mathbb{X}_L$ provides reasonably good estimate of actual test error rate $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$ and of the learning curve on test sample $\mathbb{X}_K$.

Now, let turn to comparison of different overfitting bounds. For each dataset, 5-fold cross validation were used and the results were averaged over 20 runs (for a total 100 runs). As before, training sample $\mathbb{X}_L$ was used to learn logistic regression, run algorithm 1, and estimate $\nu_\ell(\mu, \mathbb{X}_L)$ and $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$ based on 4 096 randomly generated splits $\mathbb{X}_L = X_\ell^j \cup X_k^j$. In addition, $\mathbb{X}_L$ was used to estimate overfitting $\bar{\nu}_\ell(\mu, \mathbb{X}_L) - \nu_\ell(\mu, \mathbb{X}_L)$ by medians of VC-type bound (5) and SC-bound (6) and to calculate DD-margin and DI-margin bounds from [7]. The results are presented in Table 2. Note that while all combinatorial bounds estimate overfitting, PAC DI and PAC DD are the upper bounds on the test error.

The key observation is that $\delta_\ell(\mu) \equiv \bar{\nu}_\ell(\mu, \mathbb{X}_L) - \nu_\ell(\mu, \mathbb{X}_L)$ is in the order of magnitude sharper than any of the other bounds. It works well for all datasets except Sonar (which is the smallest dataset in the selection described). Across combinatorial bounds, the SC-bound outperform VC-type bound, but still vastly overestimates the target quantity $\delta_\ell(\mu)$. All combinatorial bounds provide tighter estimates on overfitting and test error rate than PAC-Bayesian bounds.

Note that the VC-bound is estimated by a small subset of A obtained from a random walk. This is a "localized" VC-bound. The usual VC-bound estimated from VC-dimension d of a whole set A should be greater than 1 on all datasets.

5 Concluding Remarks

In this paper, new random walk technique is presented for efficient calculation of combinatorial data-dependent generalization bounds. Although combinatorial bounds are obtained for

empirical risk minimization under binary loss, it is shown that they provide sharp overfitting estimates for logistic regression. The bounds recover correct shape of the learning curves for logistic regression, correlate well with the actual overfitting, and outperform both classical VC-bound and recent state-of-the-art PAC-Bayesian bounds in experiments on 15 datasets from the UCI repository.

References

[1] *Vapnik V. N., Chervonenkis A. Y.* On the uniform convergence of relative frequencies of events to their probabilities // Theory of Probability and Its Applications. 1971. Vol. 16(2). Pp. 264–280.

[2] *Lugosi G.* On concentration-of-measure inequalities // Machine Learning Summer School, Australian National University, Canberra, 2003.

[3] *Koltchinskii V.* Local Rademacher complexities and oracle inequalities in risk minimization (with discussion) // The Annals of Statistics, 2006. Vol. 34. Pp. 2593-2706.

[4] *Koltchinskii V., Panchenko D.* Empirical margin distributions and bounding the generalization error of combined classifiers // The Annals of Statistics, 2002. Vol. 30(1). Pp. 1-50.

[5] *Koltchinskii V., Panchenko D.* Bounding the generalization error of convex combinations of classifiers: balancing the dimensionality and the margins // The Annals of Applied Probability, 2003. Vol. 13(1). Pp. 213-252.

[6] *Boucheron S., Bousquet O., Lugosi G.* Theory of classification: A survey of some recent advances // ESAIM: probability and statistics. 2005. Vol. 9(1). Pp. 323–375.

[7] *Jin C., Wang L.* Dimensionality dependent PAC-Bayes margin bound // Advances in Neural Information Processing Systems. 2012. Vol. 25. Pp. 1043–1051.

[8] *Vorontsov K. V.* Splitting and similarity phenomena in the sets of classifiers and their effect on the probability of overfitting // Pattern Recognition and Image Analysis. 2009. Vol. 19(3). Pp. 412–420.

[9] *Haussler D., Littlestone N., Warmuth M. K.* Predicting $\{0,1\}$-functions on randomly drawn points // Information and Computation, 1994. Vol. 115(2). Pp. 248–292, 1994.

[10] *Bax E.* Similar classifiers and VC error bounds, 1997.

[11] *Vorontsov K. V., Ivahnenko A. A.* Tight combinatorial generalization bounds for threshold conjunction rules. // 4-th International Conference on Pattern Recognition and Machine Intelligence (PReMI'11). Lecture Notes in Computer Science. Springer-Verlag, 2011. Pp. 66–73.

[12] *Asuncion A, Newman D. J.* UCI Machine Learning Repository // University of California, Irvine, School of Information and Computer Sciences, 2007.

[13] *Agarwal P. K., Sharir P.* Arrangements and their applications // Handbook of Computational Geometry, 1998. Pp. 49–119.

[14] *Avrachenkov K., Ribeiro B., Towsley D. (2010)* Improving random walk estimation accuracy with uniform restarts // Proc. of WAW 2010.

[15] *Ribeiro B., Towsley D.* Estimating and sampling graphs with multidimensional random walks // 10th Conference on Internet Measurement, 2010. Pp. 390–403.

[16] *Lee C., Xu X., Eun D.* Beyond random walk and Metropolis-Hastings samplers: Why you should not backtrack for unbiased graph sampling // ACM SIGMETRICS Performance Evaluation Review, 2012. Vol. 40(1). Pp. 319–330.

Численная реализация алгоритмов селективного комбинирования разнородных представлений объектов в задачах распознавания образов

***Н. А. Разин*[1], *В. В. Моттль*[2]**
nrmanutd@gmail.com[1], vmottl@yandex.ru[2]
МФТИ(ГУ)[1,2]; ВЦ РАН[2]

В работе рассматриваются методы решения задач беспризнакового распознавания образов в предположении, что объекты попарно сравниваются при помощи произвольной действительной функции. Такой подход является гораздо более общим, чем традиционный метод потенциальных функций (кернелов), требующий положительной полуопределенности матрицы функции сравнения объектов. Важное преимущество предлагаемых алгоритмов перед существующими заключается в том, что они хорошо распараллеливаются на современных многопроцессорных вычислительных системах, что позволяет использовать мощные кластеры для быстрого решения задач с большим объемом данных. Полученное ускорение по сравнению с наивными реализациями алгоритмов доходит до 25 раз на сравнительно слабой по мощности видеокарте NVidia GeForce 310M.

Ключевые слова: *беспризнаковое распознавание образов, машина релевантных векторов, алгоритмы умножения матриц, численные алгоритмы решения систем линейных уравнений, параллелизуемые алгоритмы.*

Numerical algorithms for selective multimodal pattern recognition

***N. A. Razin*[1], *V. V. Mottl* [2]**
MIPT[1,2]; Computing Centre of the Russian Academy of Sciences[2]

The authors address the problem of regression estimation under the assumption that pair-wise comparison of objects is arbitrarily scored by real numbers. Such a linear embedding is much more general than the traditional kernel-based approach, which demands positive semidefiniteness of the matrix of object comparisons. The advantage of proposed algorithms is their good scalability at multiprocessor systems, leading to use powerful clusters for solving pattern recognition problems for large data. The resulting computational speed for algorithms tested on usual video card NVidia GeForce 310M is 25 times faster compared to naive algorithms implementation.

Keywords: *machine learning, svm, rvm, gpu, cuda, parallel computations.*

Введение

Построение стандартного классификатора SVM (support vector machine) [1] сводится к решению задачи квадратичного программирования. Как следует из [2], появляются неквадратичные задачи SVM, которые сводятся к решению выпуклой задачи оптимизации, но скорость работы стандартных методов становится узким горлышком при построении разделяющей гиперплоскости. Один из известных подходов к построению стандартного классификатора SVM — использовать эвристический алгоритм SMO (sequential minimal optimization) [3]. Во-первых, SMO заточен под решение именно квадратичной задачи, к которой сводится классический SVM, а значит неквадратичные задачи [2] выпуклой оптимизации этим алгоритмом решать нельзя. Во-вторых,

SMO делает большое количество итераций, поэтому на очень больших объемах данных его ускорить не удастся.

Существующие способы, оптимизирующие скорость построения классификатора SVM, не годятся по следующим причинам:

— способы [3], [4], [5], [6] заточены под классический SVM с квадратичной задачей;

— способы класса [4] делают очень много «легковесных» итераций. Естественно, если выборка большая, такие алгоритмы будут работать очень долго. Задействовать мощности современных кластеров в случае с SMO либо с другим аналогичным алгоритмом, делающим много итераций, невыгодно.

Решение критерия с квадратично-модульной регуляризацией аналитически выписать не удается, во-первых, из-за модуля, а, во-вторых, из-за линейных ограничений. Поэтому для поиска точки оптимума используется двойственная задача и итерационные методы решения задач математического программирования. Двойственная задача к критерию с квадратично-модульной регуляризацией является стандартной задачей квадратичного программирования с линейными ограничениями. Для ее решения можно применять следующие классические методы численной условной оптимизации:

— метод проекции градиента;

— метод условного градиента;

— метод внутренней точки (метод штрафных функций).

Каждый из перечисленных методов обладает своими преимуществами и недостатками. Нас в первую очередь интересует скорость работы этих методов и требуемая для их работы память. Метод проекции градиента и метод условного градиента требуют линейных расходов по памяти в зависимости от размерности задачи. Однако их слабая сторона — количество итераций, которые эти методы делают. Очень часто даже в задачах небольшой размерности ($N \sim 100$) эти методы выполняют тысячи итераций. Естественно, что как бы отдельная итерация ни была ускорена, все равно принципиально большой скорости достигнуть не получится в силу того, что этих итераций выполняется очень много.

Метод внутренней точки в этом смысле обладает гораздо более высокими показателями скорости. Даже на больших объемах данных ($N \sim 1000$) этот метод выполняет десятки итераций. Однако каждая итерация вычислительно оказывается тяжелой. Связано это с тем, что метод внутренней точки сводит исходную задачу математического программирования к задаче безусловной оптимизации, добавляя к целевой функции штрафные функции, штрафующие выход переменных оптимизации за пределы области допустимых ограничений. Сама задача безусловной оптимизации решается методом Ньютона, что означает, что на каждой итерации нужно перемножить матрицы размеров $N \times N$ и решить систему линейных уравнений размеров $N \times N$. Первая из этих операций очень хорошо параллелится. Вторая операция тоже может быть распараллелена, но менее эффективно. Метод внутренней точки требует много памяти для хранения матриц размеров $N \times N$, то есть потребление памяти в этом случае растет квадратично от размерности задачи.

В данной работе предлагается подход к реализации обучения SVM, основанный на методе внутренней точки [7]. Во-первых, этим методом можно решать произвольную задачу выпуклой оптимизации, целевая функция которой дважды непрерывно дифференцируема. Во-вторых, метод в среднем делает существенно меньшее количество итераций по сравнению с SMO и имеет отличный потенциал для ускорения через распараллеливание. Ясно, что квадратичное использование памяти делает этот метод неприменимым при работе с очень большим количеством данных.

В рамках текущей работы авторы не столкнулись с задачами такой размерности, при которой становится невозможным применять метод внутренней точки.

Выпуклый критерий обучения — дважды регуляризованная машина опорных векторов

В данной статье мы исходим из классической версии SVM, предназначенной для отбора признаков и названной в [8] дважды регуляризованной машиной SVM.

Как и в классической SVM, предполагается, что объекты реального мира $\omega \in \Omega$ описываются k признаками $\mathbf{x}(\omega) = (x_1(\omega) \dots x_k(\omega))$, которых значительно больше, чем объектов N в обучающей выборке. Задача поиска оптимальной разделяющей гиперплоскости $\mathbf{a}^T\mathbf{x} + b = \sum_{l=1}^{k} a_l x_l + b \geqslant 0$ в $\mathbb{R}^k$ для заданной обучающей совокупности

$$\begin{cases} \sum\limits_{l=1}^{k} \left[\beta a_l^2 + \mu|a_l|\right] + \sum_{j=1}^{N} \delta_j \to \min\limits_{a_l, b, \delta_j}; \\ y_j(\sum\limits_{l=1}^{k} a_l x_{lj} + b) \geqslant 1 - \delta_j, j = 1, \dots, N, \\ \delta_j \geqslant 0, j = 1, \dots, N, \end{cases}$$

отличается от стандартной задачи обучения SVM более сложной регуляризацией, заключающейся в комбинировании норм L_1 и L_2 направляющего вектора с весами β, μ вместо чистой нормы L_2 в классическом случае.

Если $\mu = 0$, критерий превращается в классический SVM. Если же $\mu \to +\infty$, критерий становится сильно селективным, отбрасывая практически все признаки. Это означает, что, меняя параметр μ, можно менять характер обучения от сохранения всех признаков до исключения максимального их количества из решающего правила.

В отличие от классической машины SVM критерий (1) уже не является квадратичным, но остается выпуклым.

Машина RVM с произвольными функциями парного сравнения

Критерий обучения (1) остается полностью применимым в ситуации, когда представление объектов при помощи функцию сравнения $S(\omega', \omega'')$ более удобно, чем при помощи признакового описания $\mathbf{x}(\omega)$. Значения функции сравнения объекта $\omega \in \Omega$ на каждом из N объектов обучающей совокупности $x_l(\omega) = S(\omega_l, \omega)$ могут быть использованы как его вторичные признаки [9]. В этом случае мы получаем обобщенную версию RVM, которую следует назвать Relevance Object Machine, потому что нет никакой связи с представлением объектов при помощи векторов признаков.

Возможность представления объектов несколькими априори равновероятными функциями парного сравнения $S_i(\omega', \omega''), i = \overline{1, n}$ не влияет принципиально на критерий, кроме того, что количество вторичных признаков расширяется до nN:

$$\begin{cases} x_{il} = S_i(\omega_l, \omega) \text{для всех } \omega \in \Omega; \\ x_{il,j} = S_i(\omega_l, \omega_j) \text{для } j\text{-го объекта } \omega_j. \end{cases} \tag{1}$$

Прямое обобщение критерия (1) дает выпуклый критерий обучения, который отличается только количеством переменных:

$$\begin{cases} \sum\limits_{i=1}^{n} \sum\limits_{l=1}^{N} \left[\beta a_{il}^2 + \mu|a_{il}|\right] + \sum_{j=1}^{N} \delta_j \to \min\limits_{a_{il}, b, \delta_j}; \\ y_j(\sum\limits_{i=1}^{n} \sum\limits_{l=1}^{N} a_{il} x_{il,j} + b) \geqslant 1 - \delta_j, j = \overline{1, N}; \\ \delta_j \geqslant 0, j = \overline{1, N}. \end{cases}$$

Двойственная запись критерия и разбиение множества вторичных признаков

Теорема 1. *Оптимальная гиперплоскость* $(a_{il}, i = \overline{1,n}, l = \overline{1,N}, b)$ *определяется равенствами*

$$\begin{cases} \hat{a}_{il} = \dfrac{1}{2\beta}(s_{il} + \mu), s_{il} < -\mu; \\ \hat{a}_{il} = 0, -\mu \leqslant s_{il} \leqslant \mu; \\ \hat{a}_{il} = \dfrac{1}{2\beta}(s_{il} - \mu), s_{il} > \mu; \end{cases} \tag{2}$$

$$\hat{b} = -\frac{\sum_{j:0<\hat{\lambda}_j<1} \hat{\lambda}_j \sum_{l,i:\hat{\lambda}_l>0, \hat{a}_{il}\neq 0} \hat{a}_{il} x_{il,j} + \sum_{j:\hat{\lambda}_j=1} y_j}{\sum_{j:0<\hat{\lambda}_j<1} \hat{\lambda}_j}, \tag{3}$$

где

$$s_{il} = \sum_{j:\hat{\lambda}_j>0} y_j \hat{\lambda}_j x_{il,j},$$

$\{j : 0 < \hat{\lambda}_j < 1\}$ *и* $\{j : \hat{\lambda}_j = 1\}$ *– подмножества ненулевых решений* $\{j : \hat{\lambda}_j > 0\}$ *двойственной задачи выпуклого программирования:*

$$\begin{cases} W(\lambda_1, \ldots, \lambda_N | \mu) = \sum_{j=1}^N \lambda_j - \dfrac{1}{4\beta} \sum\limits_{i=1}^{n} \sum\limits_{l=1}^{N} \left\{ \min \begin{bmatrix} \mu + \sum\limits_{j=1}^{N} y_j \lambda_j x_{il,j} \\ 0 \\ \mu - \sum\limits_{j=1}^{N} y_j \lambda_j x_{il,j} \end{bmatrix} \right\}^2 \to \max\limits_{\lambda_1,\ldots,\lambda_N}; \\ \sum_{j=1}^N y_j \lambda_j = 0; \\ 0 \leqslant \lambda_j \leqslant 1,\ j = \overline{1,N}. \end{cases} \tag{4}$$

Доказательство. Доказательство основано на анализе седловой точки Лагранжиана исходной задачи оптимизации (2), в которой числа $(\lambda_1, \ldots, \lambda_N)$ являются множителями Лагранжа при неравенствах $y_j(\sum_{i=1}^n \sum_{l=1}^N a_{il} x_{il,j} + b) - 1 + \delta_j \geqslant 0$. ■

Итерационный алгоритм решения двойственной задачи

Утверждение 1. *Функция* $W(\lambda)$*в задаче*(4) *является выпуклой.*

Доказательство. В силу того, что $W(\lambda)$ является суммой двух выпуклых функций: линейной по λ и кусочно-квадратичной по λ, то отсюда с очевидностью следует, что $W(\lambda)$ - выпуклая функция. ■

Избавимся от условия-равенства в задаче (4), чтобы она полностью соответствовала постановке вида (9). Рассмотрим новые переменные η и матрицу перехода Z, что $\lambda = \mathbf{Z}\eta$. Тогда матрица Z выглядит так:

$$Z = \begin{pmatrix} -\dfrac{y_2}{y_1} & -\dfrac{y_3}{y_1} & \cdots & -\dfrac{y_N}{y_1} \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{5}$$

Обозначим $\tilde{\mathbf{x}}_{il} = \{y_1 x_{il,1}, ..., y_N x_{il,N}\}^T$.

Утверждение 2. *Двойственная задача* (4)*, записанная в переменных* η*, полностью соответствует постановке* (9).

Доказательство. Запишем задачу (4) в переменных η:

$$\begin{cases} W(\eta_1,\ldots,\eta_N|\mu) = \mathbf{1}^{\mathrm{T}} Z\eta - \frac{1}{4\beta}\sum_{i=1}^{n}\sum_{l=1}^{N}\left\{\min\begin{bmatrix}\mu+\tilde{\mathbf{x}}_{il}^{\mathrm{T}}Z\eta\\ 0\\ \mu-\tilde{\mathbf{x}}_{il}^{\mathrm{T}}Z\eta\end{bmatrix}\right\}^2 \to \max\limits_{\eta_1,\ldots,\eta_{N-1}}, \\ 0\leqslant (Z\eta)_j\leqslant 1,\ j=\overline{1,N} \end{cases} \tag{6}$$

Поскольку преобразование переменных линейно, то функция $W(\eta)$ остается выпуклой в силу теоремы 1. Ограничения-неравенства линейны, следовательно, вогнуты. ■

Утверждение 3. *Градиент и гессиан целевой функции $W(\eta)$ двойственной задачи 6 определяются выражениями:*

$$\nabla W(\eta) = \mathbf{Z}^{\mathrm{T}}\mathbf{1} - \frac{1}{2\beta}\sum_{i,l:|\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}\eta|>\mu}\left[\mathbf{Z}^{\mathrm{T}}\tilde{\mathbf{x}}_{il}\left(\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}\eta+\mu\,\mathrm{sign}(\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}\eta)\right)\right] \tag{7}$$

$$\nabla^2 W(\eta) = \frac{1}{2\beta}\mathbf{Z}^{\mathrm{T}}\left[\sum_{i,l:|\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}\eta|>\mu}\tilde{\mathbf{x}}_{il}\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\right]\mathbf{Z} \tag{8}$$

Доказательство. Заметим, что справедливо следующее равенство:

$$\min\begin{bmatrix}\mu+\tilde{\mathbf{x}}_{il}^{\mathrm{T}}Z\eta\\ 0\\ \mu-\tilde{\mathbf{x}}_{il}^{\mathrm{T}}Z\eta\end{bmatrix} = \min\begin{bmatrix}\mu-|\tilde{\mathbf{x}}_{il}^{\mathrm{T}}Z\eta|\\ 0\end{bmatrix} = \min\begin{bmatrix}\mu-\tilde{\mathbf{x}}_{il}^{\mathrm{T}}Z\eta\,\mathrm{sign}(\tilde{\mathbf{x}}_{il}^{\mathrm{T}}Z\eta)\\ 0\end{bmatrix}.$$

Подставляя это равенство в (6) и выполняя элементарные операции взятия первой и второй производных для функции многих переменных $W(\lambda)$, получаем требуемый результат. ■

Утверждения (2) и (3) дают нам возможность применить алгоритм [7] для решения двойственной задачи (6). После того, как найдены переменные η, мы выполняем обратное линейное преобразование к переменным λ. Решение исходной задачи получаем согласно теореме (1).

Алгоритм решения выпуклых задач селективного комбинирования потенциальных функций

Общая идея метода внутренней точки освещена в литературе [7], [10]. В данной работе будет использоваться модификация [7]. Исходная постановка задачи для этого метода такая:

$$\begin{cases}\min f(x);\\ c(x)\geqslant 0,\end{cases} \tag{9}$$

где $f(x) : R^n \to R$ — выпуклая функция, а $c(x) \geqslant 0$ означает, что каждая компонента $c_i : R^n \to R$ $(1 \leqslant i \leqslant m)$ неотрицательна и все c_i вогнуты. Штрафы выражаются логарифмическими функциями, а решение задачи безусловной минимизации ищется при помощи метода Ньютона.

Применяемый метод сначала сводит ту задачу, которую он решает, к двойственной, а потом оперирует в терминах этой двойственной задачи. Поэтому мы фактически будем решать задачу, двойственную к двойственной исходной задачи. После этого выписываются условия Каруша-Куна-Такера для новой двойственной задачи и полученная система решается итеративно при помощи метода Ньютона. Поскольку ограничений будет $2N$, то и двойственных переменных будет $2N$, а активных переменных останется $N-1$, потому что мы избавились от ограничения-равенства. Итого, переменных в двойственной задаче будет $3N-1$. Введем следующие обозначения:

— алгоритм оперирует значением функции $f(x)$, ее градиентом $\nabla f(x)$, гессианом $\partial^2 f(x)/(\partial x_i \partial x_j)$, значением ограничений $C(x)$ и их якобианом $\partial c_i(x)/\partial x_j$ в точке x;

— $N-1$ исходных переменных будем обозначать как и раньше: η.

— $2N$ двойственных переменных обозначим π;

— Ограничения-неравенства из (6) обозначим как вектор-столбец $\mathbf{c}(\eta) = \{c_1(\eta), ..., c_{2N}(\eta)\}^{\mathrm{T}}$, $\tilde{\mathbf{c}} = \{\underbrace{0, ..., 0}_{N}, \underbrace{1, ..., 1}_{N}\}^{\mathrm{T}}, \tilde{Z} = \left[\begin{smallmatrix} Z \\ -Z \end{smallmatrix}\right]$. Тогда

$$\mathbf{c}(\eta) = \tilde{Z}\eta + \tilde{\mathbf{c}}; \tag{10}$$

— $p = [\eta, \pi], \psi(p) = W(\eta) - \pi^T \mathbf{c}(\eta) - \mu \sum_{i=1}^{2N} \log(\pi_{(i)} c^2_{(i)}(\eta))$.

Теорема 2. *Алгоритм 1, построенный на базе модификации метода внутренней точки [7], корректно решает задачу (eqrefeq:svmmultimodalDoublyRegularizedSvm), т. е.:*

— *начальное приближение η^0 является внутренней точкой области, ограниченной условиями $c_{(l)}(\eta) > 0, l = \overline{1, 2N}$;*

— *градиент $\partial W(\eta)/(\partial \eta_i)$ и гессиан $\partial^2 W(\eta)/(\partial \eta_i \partial \eta_j)$ исходной функции $W(\eta)$ посчитаны корректно;*

— *якобиан ограничений $\partial c_{(i)}(\eta)/\partial \eta_j$ посчитан корректно;*

— *матрица вторых производных для каждого из ограничений $c_{(i)}(\eta)$ есть нулевая матрица.*

Доказательство. Напомним обозначения: условия-ограничения вычисляются согласно (10). Оценим элементы вектора $\mathbf{Z}\eta = (\lambda_1, ..., \lambda_N)^{\mathrm{T}}$:

$$\begin{cases} \lambda_1 = \sum_{i=1}^{N-1} -\frac{y_{i+1}}{y_1}\eta_i; \\ \lambda_i = \eta_{i-1}, i = 2, N. \end{cases} \tag{12}$$

Ясно, что для $i = 2, ..., N$ справедливо $0 < \lambda_i < 1$ в силу формулы (11). В силу предположения о том, что $n_{+1} \geqslant n_{-1}$ и $y_1 = -1$, имеем $\sum_{i=1}^{N-1} -(y_{i+1}/y_1)\eta_i = n_{+1}(3/(4n_{+1})) - (N - n_{+1})/(2(N - n_{+1})) = 1/4$, т.е. $0 < \lambda_1 < 1$. Совершенно аналогично доказывается, что $\mathbf{0} < -\mathbf{Z}\eta + \tilde{\mathbf{c}} < \mathbf{1}$. Остается заметить, что $\pi^0 = (1, ..., 1)^{\mathrm{T}} > \mathbf{0}^{\mathrm{T}}$, т. е. начальное приближение является внутренней точкой области, ограниченной условиями $c_{(l)}(\eta) > 0, l = \overline{1, 2N}$. Корректность градиента $\partial W(\eta)/\partial \eta_i$ и гессиана $\partial^2 W(\eta)/(\partial \eta_i \partial \eta_j)$ исходной функции $W(\eta)$ гарантируется **Утверждением** 3. Якобиан ограничений, поданных на вход, очевидным образом согласно (10) определяется как $\tilde{\mathbf{Z}}$. Матрица вторых производных $\partial^2 c_{(i)}(\eta)/(\partial \eta_i \partial \eta_j)$, очевидно, нулевая, так как все ограничения в рассматриваемом случае линейны. ■

Анализ сложности алгоритма решения выпуклых задач селективного комбинирования потенциальных функций

Предположим, что у нас в распоряжении имеется 1-процессорная вычислительная система. Это означает, что в один момент времени такая машина может выполнить одну элементарную арифметическую операций, т. е. сложение, вычитание, деление, умножение.

Пусть нам дана обучающая совокупность $\omega_j, j = \overline{1, N}$. Пусть нам заданы способы сравнения объектов $S_i(\omega', \omega''), i = \overline{1, n}$.

Каждый j-ый объект характеризуется вектором $\mathbf{x}_{il,j}, i = \overline{1, n}, l = \overline{1, N}$, т. е. nN числами. В итоге вся обучающая совокупность занимает память по порядку величины $O(nN^2)$. Перейдем к анализу времени работы алгоритма.

Алгоритм 1 Итерационный алгоритм решения задачи распознавания образов (2)

Вход: — обучающая выборка $x_{il,j}, i = \overline{1,n}, l = \overline{1,N}, j = \overline{1,N}, y_j = \pm 1$;

— $n_{+1} = |(j : y_j = 1)| \geqslant n_{-1} = |(j : y_j = -1)|, y_1 = -1$;

— регуляризующие коэффициенты $\beta > 0$, $\mu > 0$;

— параметры линейного поиска $0 < \nu < 1$, $0 < \alpha < 1$;

— параметры останова $\varepsilon_1 > 0, \varepsilon_2 > 0$.

Выход: $\hat{\mathbf{a}}_{il}, b$

1: начальное приближение $p^0 = [\eta^0, \pi^0]$:

$$\eta_i^0 = \begin{cases} 3/(4n_1), y_{i+1} = 1, \\ 1/(2(N - n_{+1})), y_{i+1} = -1 \end{cases} \tag{11}$$

$$\pi^0 = (1, \ldots, 1)$$

2: **повторять**

3: $\tilde{\mathbf{X}}(\eta^s) = \mathbf{Z}^T \left[\sum_{i,l:|\tilde{\mathbf{x}}_{il}^T \mathbf{Z}\eta^s|>\mu} \tilde{\mathbf{x}}_{il}\tilde{\mathbf{x}}_{il}^T\right] \mathbf{Z}$

4: $\nabla W(\eta^s) = -\mathbf{1}^T\mathbf{Z} + \tilde{\mathbf{X}}(\eta^s)\eta^s$

5: $\nabla^2 W(\eta^s) = \tilde{\mathbf{X}}(\eta^s)$

6: $C(\eta^{\mathbf{s}}) = diag(\tilde{Z}\eta + \tilde{\mathbf{c}})$, $\Theta = diag(\pi_1^s, \ldots, \pi_{2N}^s)$

7: $\rho^s = max(0, \frac{\|\eta^{sT}\tilde{Z}^T\pi^s\|}{2N})$

8: направление спуска $h^{s+1} = \left[d\eta^{s+1}, d\pi^{s+1}\right]$ вычисляем как решение системы линейный уравнений:

$$\begin{pmatrix} \nabla^2 W(\eta^s) & -\tilde{Z}^T \\ \Theta\tilde{Z} & C(\eta^s) \end{pmatrix} \begin{pmatrix} d\eta^{s+1} \\ d\pi^{s+1} \end{pmatrix} = \begin{pmatrix} -\nabla W(\eta^s) + \tilde{Z}\pi \\ \rho^s\mathbf{1}^T - C(\eta^s)\pi \end{pmatrix}$$

9: величину шага γ выбираем на основании алгоритма линейного поиска[10]: $\gamma_l = \nu^l\alpha$. Как только находится такое l, что $\psi(p^s + \gamma_l h^{s+1}) \leqslant \psi(p^s) + \omega\gamma_l\nabla\psi_\mu(p^s)^T h^{s+1}$, линейный поиск останавливается.

10: $\left[\eta^{s+1}, \pi^{s+1}\right] = [\eta^s, \pi^s] + \gamma_l h^{s+1} \in R^{3N-1}$.

11: **пока** $|W(\eta^s) - W(\eta^{s+1})| \geqslant \varepsilon_1$ **and** $|\nabla W(\eta^{s+1}) - \tilde{Z}^T\pi^{s+1}| \geqslant \varepsilon_2$ **and** $|C(\eta^{s+1})\pi^{s+1} - \rho^{s+1}\mathbf{1}^T| \geqslant \varepsilon_2$

12: $\lambda = \mathbf{Z}\eta^{\mathbf{s}}$

13: вычислить $\hat{\mathbf{a}}_{il}, b$ по формулам (2)

Утверждение 4. *Операция суммирования* $\sum_{i,l:|\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}\eta^s|>\mu} \tilde{\mathbf{x}}_{il}\tilde{\mathbf{x}}_{il}^{\mathrm{T}}$ *эквивалента операции умножения матриц* $\mathbf{X}^*\mathbf{X}^{*\mathrm{T}}$, *где* $\mathbf{X}^* = (\mathbf{x}_{i_1,l_1}, ..., \mathbf{x}_{i_m,l_m})$, $\{(i_k, l_k) : |\tilde{\mathbf{x}}_{i_k l_k}^T \mathbf{Z}\eta^s| > \mu\}$.

Доказательство. Пусть $\mathbf{X}^{**} = \sum_{i,l:|\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}\eta^s|>\mu} \tilde{\mathbf{x}}_{il}\tilde{\mathbf{x}}_{il}^{\mathrm{T}}$. Тогда $\mathbf{X}_{k,m}^{**} = \sum_{i,l=i_1,l_1}^{i_m,l_m} \mathbf{x}_{i,l,k}\mathbf{x}_{i,l,m}$, а это означает стандартную операцию умножения двух матриц, одна из которых состоит из векторов $\mathbf{x}_{i_1,l_1}, ..., \mathbf{x}_{i_m,l_m}$, а другая — ее транспонированная. ■

Сформулируем в терминах количества элементарных операций сложность каждого шага алгоритма 1:

1. Сначала необходимо сформировать матрицу объекты × признаки. Для этого нужно вычислить nN^2 чисел. Сложность, естественно, зависит от того, насколько долго вычисляются сами функции сравнения. Сложность этой операции $N^2\sum_{i=1}^n O(S_i)$, где $O(S_i)$ обозначает сложность вычисления i-ой функции парного сравнения.

2. Сложность вычисления $\tilde{\mathbf{x}}_{il,j}$ составляет $O(nN^2)$, так как сводится к выполнению $n \times N \times N$ операций умножения.

3. Начальное приближение, согласно шагу 1, генерирует $N-1$ число, используя каждый раз не более трех элементарных операций, так что сложность тут будет $O(N)$. Память такая же: $O(N)$.

4. В силу вида матрицы $\mathbf{Z}$ (5) операция умножения слева вектора на эту матрицу $\mathbf{x}^{\mathrm{T}}\mathbf{Z}$ эквивалентна сумме двух векторов: $(x_2, ..., x_N)^{\mathrm{T}} + x_1(-y_2/y_1, ..., -y_N/y_1)$, т.е. сложность получается $O(N)$.

5. В силу вида матрицы $\mathbf{Z}$ (5) операция умножения справа вектора на эту матрицу $\mathbf{Zx}$ эквивалентна умножению этого вектора на первый ряд этой матрицы и копированию элементов этого вектора в результирующий вектор, т. е. сложность получается $O(N)$.

6. В силу вида матрицы $\mathbf{Z}$ (5) операция умножения этой матрицы справа и слева на любую матрицу $\mathbf{Z}'$ сводится к умножению этой матрицы на первый ряд матрицы $\mathbf{Z}$ и копированию остальных элементов этой матрицы, т. е. сложность получается $O(N^2)$.

7. Анализ формулы шага 3 показывает, что на каждой итерации главного цикла необходимо знать для каждой пары i, l значение произведения $\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}$, которое никак не зависит от номера итерации и определяется только обучающей совокупностью. Поэтому можно один раз заранее вычислить все $\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}$ для всех i, l. Это занимает, с учетом вида матрицы $\mathbf{Z}$, $O(nN^2)$.

8. На шаге 3 необходимо для всех i, l вычислить произведение $\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}\eta^s$, заранее предвычисленных векторов у нас nN, тогда в данном случае сложность будет $O(nN^2)$, так как каждый раз сложность умножения равняется $O(N)$.

9. Далее на шаге 3 необходимо вычислить сумму $\sum_{i,l:|\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}\eta^s|>\mu} \tilde{\mathbf{x}}_{il}\tilde{\mathbf{x}}_{il}^{\mathrm{T}}$. Согласно **Утверждению** 4, это эквивалентно операции умножения матриц $nN \times k \times nN$, где $k = |i, l : |\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}\eta^s| > \mu|$, т. е. $k = 1, ..., nN$. То есть в худшем случае это $O(n^3N^3)$.

10. Шаг 4 ведет к сумме двух векторов, один из которых вычисляется за линейное время $O(N)$, а второй за квадратичное время $O(N^2)$.

11. Шаги 5–7 занимают линейное количество операций, т. е. $O(N)$

12. Решение системы линейных уравнений 8 делается в два этапа: сначала решается относительно $d\eta^{s+1}$ система с матрицей $(N-1) \times (N-1)$. Эту систему можно решить методом Гаусса за $O(N^3)$. На следующем этапе определяется уже $d\pi^{s+1}$, это делается за $O(N^2)$ операций.

13. Линейный поиск на шаге 9 выполняется за неопределенное количество операций, так как неизвестно заранее, на каком номере l функция $\psi(p^s)$ уменьшит свое значение. Допустим, что прошло l шагов. На каждом шаге требуется вычислить предыдущее значение $\psi(p^s)$, вычисленное на прошлой итерации, новое значение $\psi(p^s + \gamma_l h^{s+1})$ вычисляется, согласно формуле $\psi(p)$, за $O(nN)$, так как основные затраты тут пойдут на вычисление значения функции $W(\eta)$, остальные два слагаемых вычисляются за линейное время $O(N)$. Итого получается, что сложность шага 9 составляет $O(lnN)$.

14. Остальные шаги алгоритма 10–13 вычисляются за время $O(nN^2)$ за счет вычисления по формулам (2).

Количество операций до цикла 2 состоит из вычисления матрицы объекты × признаки, генерации начального приближения и предвычисления $\tilde{\mathbf{x}}_{il}^{\mathrm{T}}\mathbf{Z}$ для всех i, l: $O(nN^2) + O(N) + O(nN^2) = O(nN^2)$. Если предположить, что цикл 2 выполнялся m раз, а внутренний цикл линейного поиска выполнялся $l_1, ..., l_m$ раз соответственно, тогда сложность цикла 2 получается такой: $\sum_{i=1}^{m}\left[O(nN^2) + O(n^3N^3) + O(N^3) + O(l_i nN)\right] = O(mn^3N^3) + O(nN\sum_{i=1}^{m} l_i)$. Количество операций после алгоритма связано лишь с вычислением результата и равно $O(nN^2)$. В итоге общая сложность алгоритма, в предположении, что количество итераций равно m, а на каждой итерации выполнялся линейный поиск сложностью $l_i, i = 1, ..., m$,

$$\text{Complexity } = O(nN^2) + O(mn^3N^3) + O(nN\sum_{i=1}^{m} l_i) + O(nN^2) = \\ = O(mn^3N^3) + O(nN\sum_{i=1}^{m} l_i) \quad (13)$$

Соответственно, получается, что цикл 2 вносит максимальный вклад в количество операций алгоритма, так как на каждой итерации нужно выполнять умножение матрицы $nN \times k$ на саму себя транспонированную и решать систему линейных уравнений $N \times N$. Эти операции имеет смысл распараллелить. То, что алгоритм вынужден держать все nN^2 чисел, — особенность данного алгоритма и одновременно его самый главный недостаток, так как это существенно ограничивает объемы данных (количество объектов и количество функций парного сравнения), на которых можно было бы использовать данный алгоритм.

Модульная реализация распараллеливания

Согласно анализу сложности алгоритмов в разделах 8 наиболее трудоемкими операциями являются операции умножения двух матриц размеров $N \times N$ и решения системы линейных уравнений размеров $N \times N$. Именно эти две операции вносят максимальный вклад в итоговую сложность каждого из алгоритмов 1. Поэтому имеет смысл начать оптимизацию именно с этих операций. Согласно экспериментам, указанные выше алгоритмы тратят на умножение матриц и решение системы линейных уравнений 80% времени своей работы. Естественно, речь идет о реализации умножения матриц и решении системы линейных уравнений на однопроцессорной машине.

Умножение матриц

Обоснуем, почему распараллеливание умножения матриц лучше, чем существующие алгоритмы умножения. Кроме стандартного алгоритма умножения матриц $N \times N$ за время $O(N^3)$ известны также алгоритмы, работающие быстрее:

1. алгоритм Штрассена [11] умножает две матрицы размеров $N \times N$ за $O(N^{2.807})$;
2. алгоритм Копперсмита-Винограда [12], имеющий сложность умножения $O(N^{2.3755})$;
3. алгоритм Вильямс [13], немного обгоняющий алгоритм Копперсмита–Винограда, и имеющий сложность $O(N^{2.3727})$.

Представим, что у нас в распоряжении очень простая вычислительная система — видеокарта NVidia Geforce 310M, имеющая 16 GPU. Оценим, каких размеров должна быть матрица, чтобы алгоритм Штрассена дал выигрыш по сравнению с распараллеливанием на 16 ядер. Ответ прост: $N \geqslant 16^{1/(3-2.807)} \approx 1000000$. В оперативной памяти такая матрица занимала бы $8 \cdot 1000000^2/1024^2 \approx 8$ терабайт, что на текущий момент технически невозможно принципиально. Ясно, что при переходе к более мощным вычислительным системам выигрыш в скорости будет только увеличиваться при неизменных размерах матрицы. Что касается алгоритма Вильямс [13] и алгоритма Копперсмита–Винограда [12], то эти алгоритмы имеют очень большую константу

Алгоритм 2 Общая схема точных алгоритмов решения системы линейных уравнений $N \times N$

Вход: матрица A размеров $N \times N$, вектор b размеров $N \times 1$.
Выход: вектор $\mathbf{x}$ размеров $N \times 1 : A\mathbf{x} = \mathbf{b}$.

1: **для** $i = 1$ to N
;
2: обновить матрицу A, выполнив, в общем случае, для каждого метода(Гаусса-Жордана, разложение Холецкого и т.д.) $O(N^2)$ операций;
3: вычислить решение $\mathbf{x}$ на базе матрицы A, полученной после N итераций. Здесь сложность для каждого из методов не превосходит $O(N^2)$.

сложности, поэтому выигрывают у современных алгоритмов на матрицах, размеры которых превосходят современные технические возможности компьютеров. Поэтому в рамках данной работы параллельное умножение матриц на видеокарте — лучшее решение из всех возможных, исходя из количества арифметических операций. Если рассмотреть детали технической реализации, то необходимо учесть тот факт, что матрицу, загруженную в оперативную память компьютера, необходимо 1 раз скопировать в оперативную память видеокарты, что по порядку величины занимает $O(N^2)$ операций. Учитывая современные скорости передачи информации CPU $\rightarrow$ GPU и GPU $\rightarrow$ CPU [14], даже для матрицы 10000×10000 подобная операция копирования займет меньше секунды, что в разы меньше времени, которое тратится на операции умножения матриц на GPU. В рамках данной работы выполнено две реализации алгоритмов умножения матриц: для однопроцессорной машины(CPU) и для видеокарты(GPU). В качестве фреймворка, на базе которого выполнялась реализация алгоритмов, был взят OpenCL [15]. Это удобный современный фреймворк, сочетающий в себе возможность гибкой разработки на C++ и реализованный практически подо все современные видеокарты, в частности карты NVidia.

Решение системы линейных уравнений

Существующие способы решения системы линейных уравнений, матрицы которых не являются разреженными, делятся на точные(прямые) методы [16] и итерационные методы [16].

Точные методы решения системы линейных уравнений

Популярные точные методы для решения системы линейных уравнений $N \times N$:

1. метод Гаусса-Жордана [17], количество операций: $2N^3$;
2. решение системы линейных уравнений, используя разложение Холецкого [18], количество операций: $N^3/3$;
3. другие методы решения системы линейных уравнений через разложение матрицы в произведение более простых матриц [19]: LU-разложение, QR-разложение и т. д. Сложность таких методов по порядку величины составляет $O(N^3)$, однако эти методы работают дольше, чем через разложение Холецкого.

Каждый из перечисленных методов базируется на следующей схеме.

Согласно приведенному алгоритму 2 получается, что единственный вариант параллелизации заключается в распараллеливании каждой из N итераций. Здесь может быть два подхода. Первый подход заключается в том, чтобы на каждой из N итераций отправлять на видеокарту текущую матрицу, а потом забирать оттуда вычисленный результат. Такой подход имеет существенный недостаток: обновленную матрицу $N \times N$ придется копировать N раз во внутреннюю память карты. Уже при размерах $N = 10000$, согласно [14], на решение системы линейных уравнений потребуется ≈ 35 мин, что существенно дольше, чем просто решать систему линейных уравнений

через разложение Холецкого на однопроцессорной машине. Второй подход заключается в том, чтобы загрузить матрицу $N \times N$ на видеокарту целиком и проводить все вычисления на ней, однако даже в такой схеме не избежать N итераций, лежащих в основе данной схемы, что делает неэффективным подобный подход. Подтверждением являются попытки реализовать алгоритм разложения Холецкого на GPU [20], однако результаты пока слабые: алгоритм на видеокарте в 4 раза медленнее существующих аналогов, реализованных для однопроцессорной машины, причем потеря времени как раз уходит на копирование данных и прочие технические особенности существующих механизмов, обеспечивающих взаимодействие CPU и GPU. Итак, точные алгоритмы имеет смысл запускать на однопроцессорной машине, по крайней мере, в рамках технических ограничений, принятых в данной работе. Поэтому среди точных алгоритмов имеет смысл реализовать наиболее быстрый алгоритм решения системы линейных уравнений на базе разложения Холецкого [18].

Итерационные методы решения системы линейных уравнений

Наиболее популярные итерационные методы решения систем линейных уравнений:

1. метод Якоби [21];
2. метод Гаусса–Зейделя [21];
3. метод Релаксации [22];
4. метод сопряженных градиентов [10].

Для того чтобы первые три метода [21, 22] сходились, необходимо, чтобы спектральный радиус некоторой матрицы, зависящей от алгоритма, удовлетворял условию $\rho(\mathbf{B}) = \max\limits_{\lambda} |\lambda(\mathbf{B})| < 1$. Для каждого из алгоритмов экспериментально получен следующий факт: на задачах, которые рассматриваются в данной работе, спектральный радиус сходимости соответствующих матриц оказывается > 1, что приводит к тому, что итерационные алгоритмы не сходятся.

Метод сопряженных градиентов оказалось неэффективно использовать в прикладных задачах, рассмотренных в данной работе, поскольку этот метод почти всегда выполняет количество итераций, равное размерности матрицы (рис. 1). Поэтому в данной работе мы решили отказаться

Рис. 1: Зависимость количества итераций, выполняемых методом сопряженных градиентов при решении системы линейных уравнений 842×842 от номера итерации в методе внутренней точки. Видно, что чем выше номер итерации, тем хуже работает метод, в середине пути выходя на плато и делая сложность алгоритма решения системы линейных уравнений $O(N^3)$

от итерационных методов решения системы линейных уравнений.

Алгоритм 3 Алгоритм умножения двух матриц $n \times N$ и $N \times m$

Вход: матрица A размеров $n \times N$, Матрица B размеров $N \times m$.
Выход: матрица $\mathbf{C}$ размеров $n \times m : \mathbf{C} = \mathbf{AB}$.

для $i = 1$ to n
  для $j = 1$ to m
    выбрать, какое из двух ядер свободно
    обнулить текущую сумму для выбранного ядра: $S = 0$
    запустить на потоке на выбранном ядре вычисление суммы:
    для $k = 1$ to N
      $S = S + A[i][k]A[k][j]$
    записать результат $C[i][j] = S$ в соответствующую ячейку результирующей матрицы
    освободить занятый поток
вернуть результат: заполненная матрица C

Реализация алгоритма умножения матриц

Современные методы реализации алгоритмов [23] легко позволяют вынести такие операции в отдельные модули, чтобы потом один и тот же алгоритм можно было запустить на ноутбуке, на десктопе и на вычислительном кластере, указав каждый раз разную реализацию операций умножения матриц и решения системы линейных уравнений. Для достижения гибкости подобного уровня в данной работе все алгоритмы будут реализованы на языке C++, matlab. Данные языки, во-первых, поддерживаются практически на всех современных вычислительных системах, а во-вторых, позволяют применять стандартные паттерны проектирования [23], позволяющие переиспользовать код и легко подменять версии операций умножения и решения систем линейных уравнений в зависимости от вычислительной системы. Следует отметить важную разницу между центральным процессором портативного компьютера (CPU) и видеокартой (GPU). Центральный процессор портативного компьютера оснащен несколькими ядрами (обычно 2–4), на которых можно параллельно исполнять несколько вычислительных операций. Видеокарта является своеобразным миникластером — она устроена таким образом, что имеет на борту большое количество микропроцессоров (GPU), которых обычно сотни(на мощных видеокартах), что дает существенное преимущество в вычислительных мощностях GPU по сравнению с CPU.

Параллельное умножение матриц на CPU

Эксперименты проводились на машине, оснащенной 2-ядерным процессором Intel Core i7. Естественно, даже на однопроцессорной машине можно выиграть в скорости умножения вдвое, если распараллелить умножение на два ядра. Алгоритм 3 иллюстрирует механизм использования нескольких ядер при работе с CPU.

Параллельное умножение матриц на видеокарте NVidia GeForce 310M

Особенность реализации OpenCL под видеокарты Nvidia заключается в том, что максимальная эффективность работы видеокарты достигается тогда, когда размеры матриц кратны 32 [24]. Естественно, что в реальной жизни размеры матриц, которые нужно перемножать, вовсе необязательно кратны 32. С другой стороны, так как память на видеокарте ограничена, то очень большие матрицы там умножать тоже нельзя. Поэтому обе исходных матрицы, которые требуется умножить, необходимо разбить на блоки так, чтобы размер каждого блока был максимальным для данной видеокарты и кратен 32. Естественно, что останутся такие блоки, один из размеров которых будет $0 < s < 32$. В силу малого размера такого блока, можно легко задействовать стандартный CPU для перемножения по алгоритму 3. Согласно спецификации для видеокарты NVidia GeForce 310M необходимо передавать матрицы размеров не более 1024×1024.

Из курса линейной алгебры [17] известно следующее

Алгоритм 4 Алгоритм умножения двух матриц $n \times N$ и $N \times m$, где $n \mid 1024, N \mid 1024, m \mid 1024$, на видеокарте

Вход: матрица A размеров $n \times N$, Матрица $\mathbf{B}$ размеров $N \times m$, $n \mid 1024, N \mid 1024, m \mid 1024$
Выход: матрица $\mathbf{C}$ размеров $n \times m : \mathbf{C} = \mathbf{AB}$

1: разбить исходные матрицы на блоки 1024×1024. Это можно сделать, так как размеры этих матриц кратны 1024.
2: $n = 1024\tilde{n}, N = 1024\tilde{N}, m = 1024\tilde{m}$.
3: **для** $k = 1$ to $\tilde{n}$
4: **для** $l = 1$ to $\tilde{m}$
5: инициализировать блок матрицы $\mathbf{C}[k][l]$ нулями
6: **для** $p = 1$ to $\tilde{N}$
7: вычислить результат умножения блока $\mathbf{R} = \mathbf{A}[k][p]\mathbf{B}[p][l]$ на GPU, используя стандартный алгоритм умножения матриц на GPU через shared memory [24]
8: $\mathbf{C}[k][l] = \mathbf{C}[k][l] + \mathbf{R}$
перейти к вычислению следующего блока матрицы $\mathbf{C}[k][l]$
9: вернуть результат: заполненная матрица $\mathbf{C}$

Утверждение 5. *Результатом умножения двух блочных матриц*

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \dots & \mathbf{A}_{1M} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{L1} & \dots & \mathbf{A}_{LM} \end{pmatrix}$$

и

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_{11} & \dots & \mathbf{B}_{1N} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{M1} & \dots & \mathbf{B}_{MN} \end{pmatrix}$$

является матрица

$$\mathbf{C} = \begin{pmatrix} \sum_{k=1}^{M} \mathbf{A}_{1k}\mathbf{B}_{k1} & \cdots & \sum_{k=1}^{M} \mathbf{A}_{1k}\mathbf{B}_{kN} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{M} \mathbf{A}_{Lk}\mathbf{B}_{k1} & \cdots & \sum_{k=1}^{M} \mathbf{A}_{Lk}\mathbf{B}_{kN} \end{pmatrix}$$

при условии, что размеры всех блочных матриц $\mathbf{A}_{ik}, \mathbf{B}_{ik}$, а также матриц $\mathbf{A}, \mathbf{B}$, соответствуют правилам умножения матриц.

Алгоритм 4 описывает последовательность действий, необходимых для умножения двух матриц, когда их размеры кратны 1024. Алгоритм 5 обобщает ситуацию и описывает последовательность действий, необходимых для умножения матриц, размеры которых кратны 1024.

Оценим сложность алгоритма 4.

1. Шаги алгоритма 1–2 делаются за $O(1)$, так как это просто элементарные операции деления.

2. Шаг 5 заключается в том, что нужно выполнить инициализацию матрицы размерами $O(1024^2)$. Количество операций: 1024^2.

3. Шаг 7 заключается в том, чтобы скопировать два блока матрицы на устройство GPU - $O(1024^2)$, применить стандартный алгоритм умножения двух матриц [24] - $O(1024^3/k)$, где k – количество ядер на устройстве GPU, для видеокарты NVidia GeForce 310M это число равно 16.

4. Шаг 8 заключается в том, чтобы скопировать полученный результат умножения в финальную матрицу $\mathbf{C}$ — $O(1024^2)$.

Алгоритм 5 Алгоритм умножения двух матриц $n \times N$ и $N \times m$ на видеокарте

Вход: матрица A размеров $n \times N$, Матрица $\mathbf{B}$ размеров $N \times m$
Выход: матрица $\mathbf{C}$ размеров $n \times m : \mathbf{C} = \mathbf{AB}$

1: $n = 1024\tilde{n} + r_A, N = 1024\tilde{N} + r_{AB}, m = 1024\tilde{m} + r_B$
2: таким образом, мы получили разбиение двух исходных матриц на блоки:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{pmatrix}$$

размеры которых равны:

$$\begin{aligned} \mathbf{A}_{11} &= 1024\tilde{n} \times 1024\tilde{N}, \mathbf{A}_{12} = 1024\tilde{n} \times r_{AB}, \\ \mathbf{A}_{21} &= r_A \times 1024\tilde{N}, \mathbf{A}_{22} = r_A \times r_{AB} \\ \mathbf{B}_{11} &= 1024\tilde{N} \times 1024\tilde{m}, \mathbf{B}_{12} = 1024\tilde{m} \times r_B, \\ \mathbf{B}_{21} &= r_{AB} \times 1024\tilde{m}, \mathbf{B}_{22} = r_{AB} \times r_B \end{aligned}$$

3: произведение $A_{11}B_{11}$ вычисляем согласно алгоритму 4
4: все остальные произведения блоков $A_{ik} \times B_{lm}$ вычисляем по алгоритму 3
5: вернуть результат: заполненная матрица $\mathbf{C}$

Общее количество операций получается равным

$$\begin{aligned} \text{Complexity } &= \tilde{n}\tilde{m}\left(1024^2 + \tilde{N}\left(1024^2 + 1024^3/16 + 1024^2\right)\right) = \\ &= (66\tilde{N} + 1)nm = \left(\frac{33}{512} + \frac{1}{N}\right)nmN \end{aligned}$$

Память, необходимая для работы данного алгоритма:

$$\frac{\left(8nN + 8Nm + 8 \times 2 \times 1024^2\right)}{1024^2} = 8\left[\tilde{N}\left(\tilde{n} + \tilde{m}\right) + 2\right] \text{ мегабайта.}$$

Если матрицы таковы, что $\tilde{n} = \tilde{m} = \tilde{N} = 10$, то памяти такому алгоритму потребуется 1602 мегабайта $\approx$ 1,5 гигабайта.

Кроме умножения матриц $A_{11}B_{11}$ необходимо выполнить умножение остальных частей блочной системы:

$$\begin{aligned} \text{Complexity } &= nmr_{AB} + nNr_B + nr_{AB}r_B + r_ANm + r_Ar_{AB}m + r_ANr_B + r_Ar_{AB}r_B \leqslant \\ &\leqslant 1024\left(nm + nN + Nm\right) + 1024^2\left(n + m + N\right) + 1024^3 = O(nm + nN + Nm) \end{aligned}$$

Общее количество операций в алгоритме умножения матриц 5 получается равным

$$\text{Complexity } \leqslant \left(\frac{33}{512} + \frac{1}{N}\right)nmN + 1024\left(nm + nN + Nm\right) + 1024^2\left(n + m + N\right) + 1024^3$$

Очевидно, что алгоритм 5 требует столько же памяти, сколько и алгоритм 4.

Еще раз обратим внимание на алгоритм 1. Согласно шагу 3 и **Утверждению** 4, единственная операция умножения матриц в этом алгоритме, которая никак не упрощается, заключается в вычислении произведения $\mathbf{X}^*\mathbf{X}^{*\mathrm{T}}$, где $\mathbf{X}^*$ — некоторая матрица размеров $nN \times k$, $k = 1, ..., nN$. В этом случае очевидно, как применить алгоритм 5 к вычислению этого произведения: достаточно взять матрицу $\mathbf{X}^*$, транспонировать ее и применить алгоритм 5. Неоптимальность такого подхода заключается в том, что придется выделить память для того, чтобы хранить транспонированную матрицу $\mathbf{X}^*$, а затем скопировать туда транспонированную матрицу $\mathbf{X}^*$. Очевидно, что этого можно избежать, правильно организовав обращение к одной и той же области памяти,

Алгоритм 6 Алгоритм умножения $\mathbf{AA}^{\mathrm{T}}$ на видеокарте

Вход: матрица A размеров $n \times N$
Выход: матрица $\mathbf{C}$ размеров $n \times n : \mathbf{C} = \mathbf{AA}^T$
1: $\mathbf{C}$ = результат работы алгоритма 5 на матрицах $\mathbf{A}, \mathbf{A}^T$

где хранится матрица $\mathbf{X}^*$. Приведем алгоритм, выполняющий умножение $\mathbf{XX}^{\mathrm{T}}$ по поданной на вход матрице $\mathbf{X}$.

По количеству операций алгоритм 6 ничем не отличается от алгоритма 5, поэтому общее количество операций у этого алгоритма:

$$Complexity \leqslant \left(\frac{33}{512} + \frac{1}{N}\right) nmN + 1024\,(nm + nN + Nm) + \\ + 1024^2\,(n + m + N) + 1024^3$$

Память, необходимая для работы алгоритма 6, вдвое меньше памяти, требуемой для работы алгоритма 5, потому что не нужно хранить вторую матрицу:

$$\text{Memory } = \left(8nN + 8 \times 2 \times 1024^2\right)/1024^2 = 8\left[\tilde{N}\tilde{n} + 2\right] \text{ мегабайта}$$

Ускорение, полученное за счет распараллеливания умножения матриц

Проанализируем выигрыш, даваемый алгоритмом умножения матриц на GPU 5, по сравнению с алгоритмом умножения матриц на CPU 3. Для проведения экспериментов использовался ноутбук Sony Vaio, оснащенный процессором Core i7 и имеющий 4096 мегабайт оперативной памяти. Видеокарта NVIDIA GeForce 310M, имеющая 512 мегабайт памяти и 16 независимых GPU. Для каждого из чисел $\{2^n, n = \overline{5, 11}\}$ 100 раз генерируется две случайных матрицы, измеряется время работы каждого из двух алгоритмов 5 и 3, а потом усредняется для каждого размера матрицы. Результаты экспериментов можно видеть на рис. 2. Видно, что выигрыш при пере-

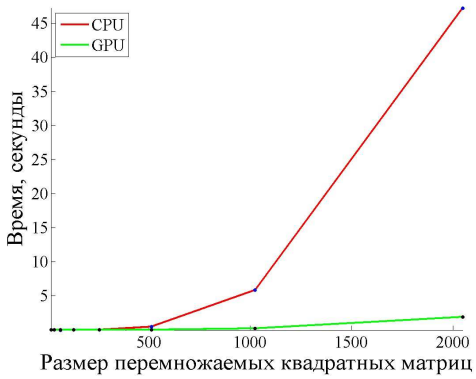


Рис. 2: Среднее время работы двух разных реализаций алгоритма умножения матриц — на видеокарте и на однопроцессорной машине — в зависимости от размера умножаемых матриц

множении матриц больших размеров доходит до 25 раз. Это связано не только с тем, что на видеокарте большее количество ядер, чем на центральном процессоре, но еще и с тем, что планировщик параллельных вычислений на видеокарте работает лучше и оптимальнее распределяет нагрузку, чем аналогичный планировщик на CPU.

Применение предложенного алгоритма на реальных задачах

Предложенный в работе алгоритм 1, использующий параллельный алгоритм умножения матриц 6 и решающий систему линейных уравнений на основе метода разложения Холецкого [18], успешно применен в задаче предсказания вторичной структуры белка в работе[25]. Математически строгие алгоритмы генерации и отбора признаков позволяют получать более высокое качество классификации. В работе [25] применен метод, основанный на RVM, к предсказанию вторичной структуры белка. Важной особенностью данного метода является то, что он позволяет автоматически отбирать наиболее информативные признаки из общего множества признаков. Средняя точность распознавания strand'ов оказалась равной примерно 75%, что находится на уровне уже существующих алгоритмов классификации. Особенно интересно то, что не было замечено переобучения, несмотря на то, что размерность векторов вторичных признаков в несколько раз выше размера обучающей совокупности. Разработанные алгоритмы позволяют существенно уменьшить количество аминокислотных фрагментов, необходимых для предсказания вторичной структуры белка с хорошей точностью. Предложенный в работе алгоритм 1 позволил ускорить этап обучения в задаче предсказания вторичной структуры белка [25] в 25 раз в сравнении с наивной реализацией данного алгоритма на однопроцессорной машине.

Заключение

В работе приведен численный алгоритм селективного комбинирования разнородных представлений объектов в задачах распознавания образов. Основное преимущество данного алгоритма перед существующими аналогами заключается в том, что он делает десятки итераций, каждая из которых хорошо распараллеливается. Данный алгоритм был реализован на видеокарте, что дало ускорение в 25 раз в сравнении с наивной реализацией такого же алгоритма на видеокарте. Недостатком данного алгоритма является то, что он требует квадратичного роста памяти в зависимости от количества объектов обучения, что существенно ограничивает применение данного алгоритма для обучения распознаванию на очень больших объемах данных(число объектов $N \gg 10^5$).

Литература

[1] V. Vapnik. *Statistical Learning Theory.* John-Wiley & Sons Inc., 1998.

[2] Татарчук А. И., Урлов Е. Н., and Моттль В. В. Метод опорных потенциальных функций в задаче селективного комбинирования разнородной информации при обучении распознаванию образов.

[3] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998. MSR-TR-98-14.

[4] Edgar Osuna, Robert Freund, and Federico Girosi. An improved training algorithm for support vector machines.

[5] Luca Zanni, Thomas Serafini, and Gaetano Zanghirati. Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research.*

[6] Edward Y. Chang, Kaihua Zhu, Hao Wang, and Hongjie Bai. Psvm: Parallelizing support vector machines on distributed computers.

[7] Paul Armand, Jean-Charles Gilbert, and Sophie Jan-Jegou. A feasible bfgs interior point algorithm for solving strongly convex minimization problems.

[8] L. Wang, J. Zhu, and H. Zou. The doubly regularized support vector machine. *Statistica Sinicia*, 16:589–615, 2006.

[9] Robert P. W. Duin, Elżbieta Pekalska, and Dick de Ridder. Relational discriminant analysis. *Pattern Recogn. Lett.*, 20(11-13):1175–1181, Nov 1999.

[10] Сухарев А. Г., Тимохов А. В., and Федоров В. В. *Курс методов оптимизации. Учеб. пособие.* ФИЗМАТЛИТ, М., 2-е edition, 2005.

[11] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354 – 356, 1969.

[12] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251 – 280, 1990.

[13] Virginia Vassilevska Williams. Breaking the coppersmith-winograd barrier. unpublished manuscript, 2011.

[14] Gpu memory transfer.

[15] Opencl: The open standard for parallel programming of heterogeneous systems, 2008.

[16] Dennis S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas (Second Edition)*. Princeton University Press, 41 William Street, Princeton, New Jersey 08540, 2009.

[17] Беклемишев Д. В. *Курс аналитической геометрии и линейной алгебры: Учеб. для вузов*. ФИЗМАТЛИТ, М., 11-е, испр. edition, 2006.

[18] Dariusz Dereniowski and Marek Kubale. Cholesky factorization of matrices in parallel and ranking of graphs. In *5th International Conference on Parallel Processing and Applied Mathematics. Lecture Notes on Computer Science 3019*, pages 985 —- 992. Springer-Verlag, 2004.

[19] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. 2000.

[20] Opencl accelerated cholesky factorization, 2011.

[21] Davod Khojasteh Salkuyeh. Generalized jacobi and gauss-seidel methods for solving linear system of equations. *NUMERICAL MATHEMATICS, A Journal of Chinese Universities (English Series)*, 16(2):164–170, 2007.

[22] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[23] Erich Gamma, Richard Helm, Ralph Johnson, and Vlissides John. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, USA, 1994.

[24] Nvidia opencl programming guide version 2.3, 2009.

[25] Nikolay Razin, Dmitry Sungurov, Vadim Mottl, Ivan Torshin, Valentina Sulimova, and Oleg Seredin. Multi–modal relevance vector machines for protein secondary structure prediction. In *PRIB-2012 proceedings*. Springer, 2012.

Комбинаторные оценки вероятности переобучения на основе кластеризации и покрытий множества алгоритмов*

А. И. Фрей, И. О. Толстихин
sashafrey@gmail.com, iliya.tolstikhin@gmail.com
Вычислительный Центр им. А. А. Дородницына РАН

В данной работе предлагается новая комбинаторная оценка вероятности переобучения, учитывающая сходство алгоритмов. Оценка основана на разложении множества алгоритмов на непересекающиеся подмножества (кластеры). Итоговая оценка учитывает сходство алгоритмов внутри каждого кластера и расслоение алгоритмов по числу ошибок между кластерами. Для оценки вероятности переобучения каждого кластера предлагается теоретико-групповой подход, основанный на учете симметрий. На примере задач из репозитория UCI показано, что предлагаемый метод в ряде случаев дает менее завышенную оценку вероятности переобучения по сравнению с известными ранее комбинаторными оценками.

Ключевые слова: *вероятность переобучения, эмпирический риск, оценка вероятности переобучения, сходство алгоритмов, частота ошибок алгоритма.*

Combinatorial bounds on probability of overfitting based on clustering and coverage of classifiers*

A. I. Frey, I. O. Tolstikhin
Computing Centre of RAS

The paper improves existing combinatorial bounds on probability of overfitting. A new bound is based on partitioning of a set of classifiers into nonoverlapping clusters and then embedding each cluster into a superset with known exact formula for the probability of overfitting. The key idea is to account for similarities between classifiers within each cluster. As a result, the new bound outperforms existing combinatorial bounds in the authors' experiments on real datasets from UCI repository.

Keywords: *probability of overfitting, empirical risk, combinatorial bounds, similaritiy between classifiers, error bounds.*

Введение

Решение задач классификации и прогнозирования можно рассматривать как задачу выбора по неполной информации. Качество алгоритма, выбранного по конечной обучающей выборке объектов, часто оказывается значительно хуже на независимой контрольной выборке. В таких случаях говорят, что произошло *переобучение* алгоритма [1, 2].

В комбинаторной теории оценок обобщающей способности [3] *вероятностью переобучения* называют долю разбиений генеральной выборки на обучающую и контрольную подвыборки фиксированной длины, при которых произошло переобучение. В [4] показа-

Работа поддержана РФФИ (проект № 11-07-00480, № 12-07-33099-мол-а-вед) и программой ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения».

но, что вероятность переобучения зависит от *профиля расслоения* множества алгоритмов и от *сходства алгоритмов* между собой.

Профилем расслоения называют распределение алгоритмов по числу ошибок на генеральной выборке. Алгоритмы с высоким числом ошибок имеют низкую вероятность реализоваться в результате обучения и потому дают незначительный вклад в вероятность переобучения. Данное явление подробно изучено в работах [5, 6, 7]. В дальнейших работах [8, 9] показано, что применение полученных оценок вероятности переобучения позволяет значительно улучшить качество композиций логических алгоритмов классификации на многих задачах из репозитория UCI.

Схожими называют алгоримы, хэмминговы расстояния между которыми малы. В [4] экспериментально показано, что сходство алгоритмов существенно уменьшает вероятность переобучения. Отметим, что большинство комбинаторных оценок вероятности переобучения основаны на анализе пар связных алгоритмов, т. е. различающихся только на одном объекте. Данный подход не позволяет в полной мере учесть сходство алгоритмов. Предложенный в [10] метод позволяет учесть сходство, но лишь в виде оценок худшего случая.

В данной работе связь между сходством алгоритмов и вероятностью переобучения будет изучена с помощью теоретико-группового подхода [11, 12, 13]. Чтобы устранить эффект расслоения в первую очередь будут рассмотрены модельные семейства, в которых все алгоритмы допускают равное число ошибок на полной выборке. На основе полученных результатов предлагается новая верхняя оценка вероятности переобучения, основанная на кластеризации алгоритмов с близкими векторами ошибок. Затем данная оценка обобщается на семейства с расслоением алгоритмов по числу ошибок. Эксперименты на 11 задачах из репозитория UCI показывают, что предлагаемый подход в ряде случаев уточняет все известные оценки вероятности переобучения.

Определения

Пусть задана генеральная выборка $\mathbb{X} = (x_1, \ldots, x_L)$, состоящая из L объектов. Произвольный алгоритм классификации, примененный к данной выборке, порождает бинарный вектор ошибок $a \equiv \big(I(a, x_i)\big)_{i=1}^{L}$, где $I(a, x_i) \in \{0, 1\}$ — бинарный индикатор ошибки алгоритма a на объекте x_i. В дальнейшем генеральная выборка $\mathbb{X}$ предполагается фиксированной, поэтому алгоритмы будут отождествляться с векторами их ошибок на выборке $\mathbb{X}$.

Для произвольной подвыборки $U \subseteq \mathbb{X}$ *число* и *частота* ошибок алгоритма a обозначаются, соответственно, через $n(a, U) = \sum\limits_{x_i \in U} I(a, x_i)$ и $\nu(a, U) = n(a, U)/|U|$.

Пусть $[\mathbb{X}]^\ell$ — множество всех разбиений генеральной выборки $\mathbb{X}$ на обучающую выборку X длины ℓ и контрольную выборку $\bar{X}$ длины $k = L - \ell$. *Методом обучения* называют отображение μ, которое произвольной обучающей выборке $X \in [\mathbb{X}]^\ell$ ставит в соответствие некоторый алгоритм $a = \mu(A, X)$ из заранее фиксированного множества $A \subset \mathbb{A}$, где $\mathbb{A} = \{0, 1\}^L$ — множество всех возможных бинарных векторов ошибок. Для произвольного разбиения $X \sqcup \bar{X} = \mathbb{X}$ *переобученностью* алгоритма $a = \mu(A, X)$ называют уклонение частот его ошибок на контроле и на обучении $\delta(a, X) = \nu(a, \bar{X}) - \nu(a, X)$.

Частоту ошибок $\nu(a, X)$ алгоритма a на обучающей выборке X часто называют *эмпирическим риском*. *Минимизация эмпирического риска* (МЭР) — это такой метод обучения, что для любой обучающей выборки X выбранный алгоритм $a = \mu(A, X)$ допускает наименьшее число ошибок на обучающей выборке X. Таким образом, для всех $X \in [\mathbb{X}]^\ell$ должно быть выполнено $\mu(A, X) \in A(X)$, где

$$A(X) \equiv \operatorname{Arg}\min_{a \in A} n(a, X). \tag{1}$$

При минимизации эмпирического риска может возникать неоднозначность — несколько алгоритмов из $A(X)$ могут иметь одинаковое число ошибок на обучающей выборке. Для устранения неоднозначности используется метод *пессимистической минимизации эмпирического риска*, выбирающий в $A(X)$ алгоритм с наибольшим числом ошибок на полной выборке. Пессимистический МЭР не может быть реализован на практике, так как он подглядывает в скрытую часть генеральной выборки. Тем не менее пессимистический МЭР является удобной теоретической конструкцией, поскольку он позволяет получать верхние оценки на вероятность переобучения любого МЭР.

Следуя слабой вероятностной аксиоматике [4], будем считать, что на множестве $[\mathbb{X}]^\ell$ всех C_L^ℓ разбиений $X \sqcup \bar{X}$ введено равномерное распределение вероятностей. Тогда вероятность переобучения $Q_\varepsilon(A)$ определяется как доля разбиений, при которых переобученность превышает заданный порог $\varepsilon \in (0,1]$:

$$Q_\varepsilon(A) = \mathsf{P}[\delta(\mu(A,X),X) \geqslant \varepsilon], \tag{2}$$

где $\mathsf{P}[\varphi] \equiv \frac{1}{C_L^\ell} \sum\limits_{X \in [\mathbb{X}]^\ell} \varphi(X)$, φ — произвольный предикат на множестве разбиений $[\mathbb{X}]^\ell$, а квадратные скобки переводят логическое выражение в число 0 или 1 по правилам [истина] $= 1$, [ложь] $= 0$. Заметим, что $1 - Q_\varepsilon(A)$ как функция порога ε есть функция распределения случайной величины $\delta(\mu(A,X),X)$, определенной на конечном вероятностном пространстве $\{[\mathbb{X}]^\ell, 2^{[\mathbb{X}]^\ell}, \mathsf{P}\}$, где P — равномерное распределение. В случаях, когда из контекста понятно, о каком множестве алгоритмов идет речь, будем опускать аргумент A и записывать вероятность переобучения как Q_ε.

Рассмотрим множество $A = \{a\}$, состоящее из одного алгоритма. Тогда $\mu X = a$ для любой выборки $X \in [\mathbb{X}]^\ell$. Это значит, что вероятность переобучения Q_ε преобразовалась в вероятность больших уклонений между частотами ошибок в выборках $X, \bar{X}$. Допустив, что число ошибок $n(a,\mathbb{X})$ нам известно, получим точное выражение для Q_ε.

Теорема 1 (FC-оценка [8]). *Для фиксированного алгоритма a, такого что $m = n(a,\mathbb{X})$, и любого $\varepsilon \in [0,1]$ вероятность переобучения определяется левым хвостом гипергеометрического распределения:*

$$Q_\varepsilon(a) = H_L^{\ell,m}\left(\tfrac{\ell}{L}(m - \varepsilon k)\right), \tag{3}$$

где $H_L^{\ell,m}(s) = \sum\limits_{t=0}^{s} h_L^{\ell,m}(t)$ — функция гипергеометрического распределения, $h_L^{\ell,m}(t) = C_m^t C_{L-m}^{\ell-t}/C_L^\ell$ — функция плотности гипергеометрического распределения [4].

Гипергеометрическое распределение играет важную роль во многих комбинаторных оценках. Оценка (3), примененная совместно с неравенством Буля, позволяет получить верхнюю оценку на $Q_\varepsilon(A)$, справедливую для любого метода обучения μ.

Теорема 2 (VC-оценка [8]). *Для любого метода обучения μ и любого $\varepsilon \in [0,1]$ вероятность переобучения ограничена суммой FC-оценок по множеству алгоритмов A:*

$$Q_\varepsilon(A) \leqslant \mathsf{P}\big[\max_{a \in A} \delta(a,X) \geqslant \varepsilon\big] \leqslant \sum_{a \in A} H_L^{\ell,m}\left(\tfrac{\ell}{L}(m - \varepsilon k)\right), \quad m = n(a,\mathbb{X}). \tag{4}$$

Назовем две причины завышенности оценки (4). Во-первых, большинство алгоритмов из A имеют высокую частоту ошибок и, следовательно, имеют исчезающе малую вероятность реализоваться в результате обучения. Тем не менее, оценка равномерного уклонения игнорирует это свойство метода обучения μ. Во-вторых, неравенство Буля игнорирует тот

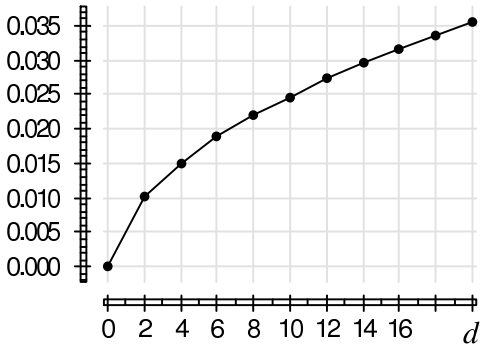


Рис. 1: Зависимость медианы распределения Q_ε от хэммингова расстояния $d = \rho(a_1, a_2)$ между векторами ошибок пары алгоритмов. $L = 100$, $\ell = 50$

факт, что алгоритмы с близкими векторами ошибок переобучаются в основном на одних и тех же разбиениях. Более точные оценки должны учитывать свойства метода обучения и сходство между алгоритмами.

Эффект сходства

Хэммингово расстояние между алгоритмами a_1 и a_2 называют величину

$$\rho(a_1, a_2) = \sum_{x_i \in \mathbb{X}} [a_1(x_i) \neq a_2(x_i)].$$

Рассмотрим простой пример, иллюстрирующий зависимость переобучения от хэммингова расстояния между алгоритмами семейства.

Эксперимент 1. Множество $A = (a_1, a_2)$ состоит из двух алгоритмов, допускающих по m ошибок на полной выборке. Векторы ошибок подобраны так, чтобы хэммингово расстояние $\rho(a_1, a_2)$ равнялось заранее фиксированному числу d. На рис. 1 приведена зависимость медианы распределения $Q_\varepsilon(A)$ от хэммингова расстояния между алгоритмами. Видно, что переобучение увеличивается с ростом $\rho(a_1, a_2)$.

Зададимся целью построить верхнюю оценку вероятности переобучения, учитывающую данный эффект. Допустим, что исходное множество алгоритмов A представлено в виде разбиения на непересекающиеся подмножества $A = A_1 \sqcup A_2 \sqcup \cdots \sqcup A_t$ так, что в каждое A_i попали лишь алгоритмы с близкими векторами ошибок. В данной ситуации будем называть множества A_i *кластерами* алгоритмов. Покажем, что задачу оценивания вероятности переобучения всего множества A можно свести к оцениванию вероятности переобучения отдельных кластеров.

Лемма 3. *Пусть множество алгоритмов A произвольным образом представлено в виде разбиения на непересекающиеся подмножества $A = A_1 \sqcup A_2 \sqcup \cdots \sqcup A_t$. Тогда вероятность переобучения пессимистического метода минимизации эмпирического риска оценивается сверху следующим выражением:*

$$Q_\varepsilon(A) \leqslant \sum_{i=1}^{t} Q_\varepsilon(A_t). \tag{5}$$

В дальнейшем лемма 3 будет играть ту же роль, что и неравенство Буля при выводе оценки (4). Преимущество данной леммы в том, что вместо суммирования по всем

алгоритмам $a \in A$ суммирование производится по кластерам произвольного разбиения $A = A_1 \sqcup A_2 \sqcup \cdots \sqcup A_t$.

Доказательство. Заметим, что достаточно доказать неравенство (5) для $t = 2$ (для произвольного числа кластеров неравенство доказывается индукцией по t). Обозначим через $\mu(A, X)$ алгоритм, выбранный пессимистическим методом минимизации эмпирического риска из множества A по обучающей выборке X. Рассмотрим произвольное разбиение $X \in [\mathbb{X}]^\ell$ и покажем следующее:

$$[\delta(\mu(A, X), X) \geqslant \varepsilon] \leqslant [\delta(\mu(A_1, X), X) \geqslant \varepsilon] + [\delta(\mu(A_2, X), X) \geqslant \varepsilon]. \tag{6}$$

Для разбиения X и множеств A_1, A_2, A множества $A_1(X), A_2(X), A(X)$ определены согласно (1). Обозначим через $n_1(X), n_2(X)$ и $n(X)$ число ошибок на обучающей выборке для алгоритмов из $A_1(X)$, $A_2(X)$ и $A(X)$, соответственно. Очевидно, что $n_1(X) \geqslant n(X)$ и $n_2(X) \geqslant n(X)$, но по крайней мере одно из этих неравенств обязательно обращается в равенство. Рассмотрим два случая: в первом одно неравенство строгое, во втором оба неравенства обращаются в равенство.

Случай 1. Пусть для определенности $n_1(X) > n(X)$. Тогда $A_2(X) = A(X)$, и следовательно $\mu(A_2, X) = \mu(A, X)$, откуда немедленно следует (6).

Случай 2. Из $n_1(X) = n_2(X) = n(X)$ следует, что $A(X) = A_1(X) \cup A_2(X)$, и, таким образом, либо $\mu(A, X) \in A_1(X)$, либо $\mu(A, X) \in A_2(X)$ (в зависимости от того, в какое из этих двух множеств попал алгоритм с наибольшим числом ошибок на полной выборке). Значит, вновь выполнено (6). ■

Для вывода верхних оценок вероятности переобучения каждого кластера удобно потребовать, чтобы внутри каждого кластера алгоритмы допускали равное число ошибок на полной выборке. Тогда можно воспользоваться следующей леммой из работы [13].

Лемма 4. *Пусть A_i, B — два множества алгоритмов, $A_i \subseteq B$, и все алгоритмы из B допускают равное число ошибок на полной выборке. Пусть метод обучения является минимизацией эмпирического риска. Тогда для всех $\varepsilon > 0$ выполнено неравенство $Q_\varepsilon(A_i) \leqslant Q_\varepsilon(B)$.*

Доказательство. Докажем утверждение для частного случая $B = A_i \cup \{b\}$. Рассмотрим произвольное разбиение $X \in [\mathbb{X}]^\ell$. Нас интересуют только разбиения с $\mu(B, X) = b$, потому что вклад остальных разбиений в вероятность переобучения не изменился. Пусть $a = \mu(A_i, X)$ — алгоритм, выбранный на разбиении X методом обучения из множества A_i. Поскольку μ является минимизацией эмпирического риска, получим $n(b, X) \leqslant n(a, X)$. Поскольку по условию алгоритмы a и b имеют равное число ошибок на полной выборке, уклонение частоты $\delta(b, X) \geqslant \delta(a, X)$. Следовательно, вклад каждого разбиения от добавления алгоритма b мог только увеличиться. ■

На выбор множества B накладывается несколько условий. Во-первых, оно должно состоять из алгоритмов с равным числом ошибок на полной выборке. Во-вторых, все алгоритмы должны иметь близкие векторы ошибок. В-третьих, оценка вероятности переобучения $Q_\varepsilon(B)$ должна быть вычислительно эффективной. Оказывается, следующее семейство удовлетворяет всем этим требованиям.

Определение 1. *Пусть a_0 — произвольный алгоритм с m ошибками, $r \leqslant m$ — натуральное число. Центральным слоем хэммингова шара называется множество:*

$$B_r^m(a_0) = \{a \in \mathbb{A} \colon \rho(a, a_0) \leqslant r \text{ и } n(a, \mathbb{X}) = m\}.$$

Данное множество состоит из алгоритмов хэммингова шара радиуса r с центром в a_0, допускающих на полной выборке столько же ошибок, сколько и центр шара. Вероятность

переобучения $Q_\varepsilon(B_r^m(a_0))$ зависит только от радиуса шара r и числа ошибок $n(a_0, \mathbb{X})$, поэтому в дальнейшем вместо $B_r^m(a_0)$ будет использоваться сокращенная запись B_r^m.

Эксперимент 2. Исследуем вероятность переобучения $Q_\varepsilon(B_r^m)$ численно с помощью метода Монте-Карло, сэмплируя (2) по 10 тыс. случайным подвыборкам $X \in [\mathbb{X}]^\ell$. Для сравнения мы рассмотрим еще одно модельное множество R_n^m, составленное из n алгоритмов, допускающих по m ошибок на полной выборке. Векторы ошибок всех алгоритмов из R_n^m сгенерированы случайно и независимо. В следующей таблице показано, при каком числе алгоритмов n в R_n^m медианы распределений $Q_\varepsilon(R_n^m)$ и $Q_\varepsilon(B_r^m)$ совпадают.

Таблица 1: Сравнение $|R_n^m|$ и $|B_r^m|$ при $L = 50$, $\ell = 25$, $m = 10$

r	$\lvert B_r^m\rvert$	$\lvert R_n^m\rvert$	δ
2	401	2	0.079
4	35 501	7	0.160
6	1 221 101	39	0.240
8	20 413 001	378	0.319

Из табл. 1 видно, что всего семь алгоритмов со случайными векторами ошибок могут переобучиться так же сильно, как и множество из десятков тысяч алгоритмов с близкими векторами ошибок. В следующем параграфе мы детально изучим это свойство множества B_r^m и приведем точную формулу для вероятности переобучения $Q_\varepsilon(B_r^m)$.

Центральный слой хэммингова шара

Точная формула вероятности переобучения $Q_\varepsilon(B_r^m)$ впервые приводится в работе [13].

Теорема 5 ([13]). *Вероятность переобучения рандомизированного метода минимизации эмпирического риска для m-го слоя хэммингова шара B_r^m радиуса r при $r \leqslant 2m$ и $n(a_0, \mathbb{X}) = m$ записывается в виде*

$$Q_\varepsilon(B_r^m) = H_L^{\ell,m}\left(\tfrac{\ell}{L}(m - \varepsilon k) + \left\lfloor r/2 \right\rfloor\right) \cdot [m \geqslant \varepsilon k], \tag{7}$$

где $H_L^{\ell,m}(s)$ — функция гипергеометрического распределения.

Доказательство этого утверждения ранее не публиковалось, и мы приводим его в настоящем параграфе. При доказательстве будет использоваться теоретико-групповой подход [11, 12, 13].

Пусть $S_L = \{\pi\colon \mathbb{X} \to \mathbb{X}\}$ — симметрическая группа из L элементов, действующая на генеральную выборку перестановками объектов. Действие произвольной $\pi \in S_L$ на алгоритм $a \in A$ определено перестановкой координат вектора ошибок: $(\pi a)(x_i) = a(\pi^{-1}x_i)$. Для произвольной выборки $X \in [\mathbb{X}]^\ell$ и множества алгоритмов $A \subset \{0,1\}^L$ действия πX и πA определены следующим образом: $\pi X = \{\pi x\colon x \in X\}$, $\pi A = \{\pi a\colon a \in A\}$.

Определение 2 ([11]). *Группой симметрий* $\mathrm{Sym}(A)$ *множества алгоритмов $A \subset \mathbb{A}$ называют его стационарную подгруппу:*

$$\mathrm{Sym}(A) = \{\pi \in S_L\colon \pi A = A\}.$$

Орбитой элемента m множества M, на котором действует группа G, называется подмножество $Gm = \{gm\colon g \in G\} \subseteq M$. Орбиты двух элементов m_1 и m_2 либо не пересекаются, либо совпадают. Это позволяет говорить о разбиении множества M на непересекающиеся орбиты: $M = Gm_1 \sqcup \ldots \sqcup Gm_k$. Множество орбит действия группы $\mathrm{Sym}(A)$ на A обозначим через $\Omega(A)$, и для каждой орбиты $\omega \in \Omega(A)$ обозначим через a_ω произвольного представителя этой орбиты. Аналогично, множество орбит действия $\mathrm{Sym}(A)$ на $[\mathbb{X}]^\ell$ обозначим через $\Omega([\mathbb{X}]^\ell)$, и для орбиты $\tau \in \Omega([\mathbb{X}]^\ell)$ обозначим ее представителя через X_τ.

Для широкого класса методов обучения группа $\mathrm{Sym}(A)$ позволяет учесть симметрии множества алгоритмов при выводе оценок вероятности переобучения. В частности, в работах [11, 12] используется *рандомизированный метод минимизации эмпирического риска*. Этот метод выбирает произвольный алгоритм из множества $A(X)$ случайно и равновероятно. Определение вероятности переобучения (2) приходится модифицировать, усреднив переобученность по множеству $A(X)$:

$$Q_\varepsilon(A) = \frac{1}{C_L^\ell} \sum_{X \in [\mathbb{X}]^\ell} \frac{1}{|A(X)|} \sum_{a \in A(X)} \big[\delta(a, X) \geqslant \varepsilon\big]. \tag{8}$$

В [12] показано, что с учетом группы симметрий $\mathrm{Sym}(A)$ вероятность переобучения (8) может быть переписана следующим образом:

$$Q_\varepsilon(A) = \frac{1}{C_L^\ell} \sum_{\omega \in \Omega(A)} |\omega| \sum_{X \in [\mathbb{X}]^\ell} \frac{[a_\omega \in A(X)]}{|A(X)|} [\delta(a_\omega, X) \geqslant \varepsilon].$$

Для вывода точной формулы $Q_\varepsilon(B_r^m)$ нам будет удобнее рассматривать действие группы симметрий на множестве $[\mathbb{X}]^\ell$ всех разбиений выборки на обучение и контроль.

Лемма 6. *Пусть для некоторой функции $f\colon 2^{\mathbb{A}} \times [\mathbb{X}]^\ell \to \mathbb{R}$ для всех $A \subset \mathbb{A}$, $X \in [\mathbb{X}]^\ell$ и всех $\pi \in \mathrm{Sym}(A)$ выполнено условие $f(A, X) = f(A, \pi X)$. Тогда справедливо следующее разложение:*

$$\sum_{X \in [\mathbb{X}]^\ell} f(A, X) = \sum_{\tau \in \Omega([\mathbb{X}]^\ell)} |\tau| f(A, X_\tau).$$

Доказательство. Доказательство с очевидностью следует из группировки равных слагаемых. ■

Нас интересует, какие функции удовлетворяют условию $f(A, X) = f(A, \pi X)$. Введем следующую классификацию:

— симметричной функцией *первого рода* будем называть $g\colon \mathbb{A} \times [\mathbb{X}]^\ell \to \mathbb{R}$, такую что для всех $\pi \in S_L$ выполнено $g(a, X) = g(\pi a, \pi X)$;
— симметричной функцией *второго рода* будем называть $G\colon 2^{\mathbb{A}} \times [\mathbb{X}]^\ell \to 2^{\mathbb{A}}$, такую что для всех $\pi \in S_L$ выполнено $\pi G(A, X) = G(\pi A, \pi X)$;
— симметричной функцией *третьего рода* будем называть $f\colon 2^{\mathbb{A}} \times [\mathbb{X}]^\ell \to \mathbb{R}$, такую что для всех $\pi \in S_L$ выполнено $f(A, X) = f(\pi A, \pi X)$.

В [12] было показано, что функции $n(a, X)$ и $\nu(a, X)$ являются симметричными функциями первого рода, а A и $A(X)$ — симметричными второго рода. Следующие две теоремы позволяют легко строить новые симметричные функции из уже имеющихся.

Теорема 7. *Пусть $g_1, g_2, \ldots, g_p$ — симметричные функции первого рода, $f_1, f_2, \ldots, f_p$ — симметричные функции третьего рода, $F\colon \mathbb{R}^p \to \mathbb{R}$ — произвольная функция многих переменных. Тогда $F(g_1, g_2, \ldots, g_p)$ — вновь симметричная функция первого рода, $F(f_1, f_2, \ldots, f_p)$ — симметричная функция третьего рода.*

Доказательство. Проведя элементарные выкладки, получим

$$F(\pi a, \pi X) \equiv F(g_1(\pi a, \pi X), \ldots, g_p(\pi a, \pi X)) = F(g_1(a, X), \ldots, g_p(a, X)) \equiv F(a, X),$$

и аналогично для функций третьего рода. ■

Теорема 8. *Пусть g — симметричная функция первого рода, G — симметричная функция второго рода. Тогда $f(A, X) \equiv |G(A, X)|$ и $f(A, X) \equiv \sum\limits_{a \in G(A,X)} g(a, X)$ — симметричные функции третьего рода.*

Доказательство. Заметим, что для любого $A \subset \mathbb{A}$ выполнено $|A| = |\pi A|$, поскольку π, как элемент группы, является биекцией. Следовательно, $|G(A, X)| = |\pi G(A, X)| = |G(\pi A, \pi X)|$.

Для функции $f(A, X) \equiv \sum\limits_{a \in G(A,X)} g(a, X)$ запишем следующую цепочку равенств:

$$\begin{aligned} f(\pi A, \pi X) &= \sum_{a \in G(\pi A, \pi X)} g(a, \pi X) = \sum_{a \in \pi G(A, X)} g(a, \pi X) = \\ &= \sum_{a \in G(A,X)} g(\pi a, \pi X) = \sum_{a \in G(A,X)} g(a, X) = f(A, X). \end{aligned}$$

■

Из приведенных выше теорем следует, что $\frac{1}{|A(X)|} \sum\limits_{a \in A(X)} [\delta(a, X) \geqslant \varepsilon]$ является симметричной функцией третьего рода, а следовательно вероятность переобучения можно факторизовать по действию группы симметрий на множестве разбиений выборки:

$$Q_\varepsilon(A) = \sum_{\tau \in \Omega([\mathbb{X}]^\ell)} \frac{|\tau|}{|A(X_\tau)|} \sum_{a \in A(X_\tau)} [\delta(a, X_\tau) \geqslant \varepsilon]. \tag{9}$$

Данную формулу можно упростить для случая, когда все алгоритмы из A имеют равное число ошибок на полной выборке.

Теорема 9. *Пусть все $a \in A$ имеют равное число ошибок на полной выборке. Тогда вероятность переобучения рандомизированного метода минимизации эмпирического риска записывается в виде:*

$$Q_\varepsilon(A) = \frac{1}{C_L^\ell} \sum_{\tau \in \Omega([\mathbb{X}]^\ell)} |\tau| \left[\min_{a \in A} n(a, X_\tau) \leqslant \frac{\ell}{L}(m - \varepsilon k) \right]. \tag{10}$$

Доказательство. Заметим, что все алгоритмы из $A(X_\tau)$ имеют одинаковое переобучение. Это следует из двух утверждений: во-первых, все алгоритмы из A имеют равное число ошибок на полной выборке, во-вторых, все алгоритмы из $A(X_\tau)$ имеют равное число ошибок на обучении. Значит (9) можно упростить следующим образом:

$$Q_\varepsilon(A) = \frac{1}{C_L^\ell} \sum_{\tau \in \Omega([\mathbb{X}]^\ell)} |\tau| \left[\delta(a', X_\tau) \geqslant \varepsilon \right],$$

где a' — произвольный алгоритм из $A(X_\tau)$. Из $a' \in A(X_\tau)$ следует, что $n(a', X_\tau) = \min\limits_{a \in A} n(a, X)$. Подставляя это выражение в определение уклонения частоты $\delta(a, X)$ и проводя элементарные выкладки, получаем (10). ■

Вернемся к выводу формулы вероятности переобучения для центрального слоя хэммингова шара B_r^m с центром в a_0. Обозначим через $X^m = \{x \in \mathbb{X}\colon a_0(x) = 1\}$ множество объектов, на которых ошибается алгоритм a_0, и $X^{L-m} = \{x \in \mathbb{X}\colon a_0(x) = 0\}$ — множество объектов, на которых a_0 не ошибается.

Лемма 10. *Группа $S_m \times S_{L-m}$, где S_m и S_{L_m} — симметрические группы перестановок, действующие на множествах X^m и X^{L-m}, соответственно, является подгруппой группы симметрий множества алгоритмов B_r^m.*

Доказательство. Доказательство следует из определения центрального слоя хэммингова шара и инвариантности хэммингова расстояния к действию группы S_L. ■

Лемма 11. *Орбиты $\tau \in \Omega([\mathbb{X}]^\ell)$ индексированы параметром $i = |X \cap X^m|$. Мощность орбиты τ_i записывается в виде $|\tau_i| = C_L^\ell h_L^{\ell,m}(i)$, где $h_L^{\ell,m}(i) = C_m^t C_{L-m}^{\ell-i}/C_L^\ell$ — функция плотности гипергеометрического распределения.*

Доказательство. Первое утверждение леммы непосредственно следует из структуры подгруппы симметрий, полученной в лемме 10. Мощность орбиты $|\tau_i| = C_L^\ell h_L^{\ell,m}(i)$ определяется числом способов выбрать i объектов из X^m и $\ell - i$ объектов из X^{L-m}. ■

Теперь мы можем приступить к доказательству теоремы 5.

Доказательство. Воспользовавшись теоремой 9 и леммой 11, получим

$$Q_\varepsilon(B_r^m) = \sum_{i=0}^{m} h_L^{\ell,m}(i) \left[\min_{a\in A} n(a, X_i) \leqslant \frac{\ell}{L}(m - \varepsilon k)\right].$$

Напомним, что по определению параметр i обозначает мощность множества $X \cap X^m$. Пусть $r' = \left\lfloor \frac{r}{2} \right\rfloor$. Тогда

$$\min_{a\in B_r^m} n(a, X_i) = \begin{cases} 0, \text{ при } i \leqslant r', \\ i - r', \text{ при } i > r'. \end{cases}$$

Следовательно,

$$Q_\varepsilon(B_r^m) = \begin{cases} 0, \text{ при } m < \varepsilon k, \\ H_L^{\ell,m}(\frac{\ell}{L}(m - \varepsilon k) + \left\lfloor r/2 \right\rfloor), \text{ при } m \geqslant \varepsilon k. \end{cases} \blacksquare$$

Из доказанной формулы следует, что вероятность переобучения $Q_\varepsilon(B_r^m)$ соответствует FC-оценке (3) для фиксированного алгоритма, смещенной вправо на $\Delta\varepsilon = \frac{L}{k\ell}\left\lfloor r/2 \right\rfloor$. Это объясняет, почему в эксперименте 2 медиана распределения $Q_\varepsilon(B_r^m)$ растет линейно с ростом радиуса r.

Учет расслоения

Результаты двух предыдущих параграфов позволяют сформулировать следующую оценку вероятности переобучения.

Теорема 12. *Пусть $A = A_1 \sqcup A_2 \sqcup \cdots \sqcup A_t$, и в каждом подмножестве A_i алгоритмы допускают равное число ошибок. Пусть каждое множество A_i вложено в центральный слой хэммингова шара $B_{r_i}^{m_i}$. Тогда*

$$Q_\varepsilon(A) \leqslant \sum_{i=1}^{t} Q_\varepsilon(B_{r_i}^{m_i}) = \sum_{i=1}^{t} \left\{ H_L^{\ell,m_i}(\tfrac{\ell}{L}(m_i - \varepsilon k) + \left\lfloor r_i/2 \right\rfloor) \cdot [m_i \geqslant \varepsilon k] \right\}. \tag{11}$$

Доказательство. Доказательство следует из лемм 3, 4 и формулы (7). ■

Оценка (11) учитывает сходство, но не расслоение алгоритмов по числу ошибок. Этот недостаток можно исправить, применив метод порождающих и запрещающих множеств [5]. Для этого метод необходимо обобщить, чтобы он был применим не к отдельным алгоритмам, а непосредственно к кластерам алгоритмов.

Гипотеза 1. Пусть множество алгоритмов A представлено в виде разбиения на непересекающиеся подмножества $A = A_1 \sqcup A_2 \sqcup \cdots \sqcup A_t$. Пусть выборка $\mathbb{X}$ и метод обучения μ таковы, что для каждого $i = 1, \ldots, t$ можно указать пару непересекающихся подмножеств $X_i \subset \mathbb{X}$ и $X'_i \subset \mathbb{X}$, удовлетворяющую условию

$$[\mu(A, X) \in A_i] \leqslant [X_i \subset X][X'_i \subset \bar{X}], \quad \forall X \in [\mathbb{X}]^\ell.$$

Пусть, кроме этого, все алгоритмы $a \in A_i$ не допускают ошибок на X_i и ошибаются на всех объектах из X'_i.

Множество X_i будем называть *порождающим*, множество X'_i — *запрещающим* для A_i. Гипотеза 1 означает, что результат обучения может принадлежать A_i только в том случае, если в обучающей выборке X находятся все порождающие объекты и ни одного запрещающего. Все остальные объекты $\mathbb{Y}_i \equiv \mathbb{X} \backslash X_i \backslash X'_i$ будем называть *нейтральными* для A_i.

Пусть $L_i = L - |X_i| - |X'_i|$, $\ell_i = \ell - |X_i|$, $k_i = k - |X'_i|$. Пусть $Q'_\varepsilon(A_i)$ есть вероятность переобучения на множестве нейтральных объектов $\mathbb{Y}_i$:

$$Q'_\varepsilon(A_i) = \frac{1}{C^{\ell_i}_{L_i}} \sum_{Y \in [\mathbb{Y}_i]^{\ell_i}} [\delta(\mu(A_i, Y), Y) \geqslant \varepsilon],$$

где $[\mathbb{Y}_i]^{\ell_i}$ — множество разбиений $\mathbb{Y}_i$ на обучающую выборку Y длины ℓ_i и контрольную выборку $\bar{Y}$ длины $k_i = L_i - \ell_i$.

Теорема 13 (Оценка расслоения-сходства). *Пусть выполнена гипотеза 1, а на разбиение $A = A_1 \sqcup A_2 \sqcup \cdots \sqcup A_t$ наложено дополнительное ограничение: внутри каждого кластера A_i все алгоритмы допускают равное число ошибок (обозначаемое через m_i). Тогда вероятность переобучения $Q_\varepsilon(A)$ ограничена сверху следующей оценкой:*

$$Q_\varepsilon(A) \leqslant \sum_{i=1}^{t} P_i \, Q'_{\varepsilon_i}(A_i), \tag{12}$$

где $P_i = \frac{C^{\ell_i}_{L_i}}{C^\ell_L}$, $\varepsilon_i = \frac{L_i}{\ell_i k_i} \frac{\ell k}{L} \varepsilon + \left(1 - \frac{\ell L_i}{L \ell_i}\right) \frac{m_i}{k_i} - \frac{|X'_i|}{k_i}$, $Q'_\varepsilon(A_i)$ — определенная выше вероятность переобучения на множестве нейтральных объектов.

Доказательство. Распишем определение вероятности переобучения:

$$\begin{aligned} Q_\varepsilon(A) &= \frac{1}{C^\ell_L} \sum_{X \in [\mathbb{X}]^\ell} [\delta(\mu(A, X), X) \geqslant \varepsilon] = \\ &= \frac{1}{C^\ell_L} \sum_{i=1}^{t} \sum_{X \in [\mathbb{X}]^\ell} [\mu(A, X) \in A_i][\delta(\mu(A_i, X), X) \geqslant \varepsilon] \leqslant \\ &\leqslant \frac{1}{C^\ell_L} \sum_{i=1}^{t} \sum_{X \in [\mathbb{X}]^\ell} [X_i \subset X][X'_i \subset \bar{X}][\delta(\mu(A_i, X), X) \geqslant \varepsilon]. \end{aligned}$$

Пусть $Y = X \setminus X_i$. Тогда $\sum\limits_{X \in [\mathbb{X}]^\ell}$ при условии $[X_i \subset X][X'_i \subset \bar{X}]$ можно заменить на суммирование по $Y \in [\mathbb{Y}_i]^{\ell_i}$.

$$Q_\varepsilon(A) \leqslant \frac{C_{L_i}^{\ell_i}}{C_L^\ell} \sum_{i=1}^{t} \frac{1}{C_{L_i}^{\ell_i}} \sum_{Y \in [\mathbb{Y}_i]^{\ell_i}} [\delta(\mu(A_i, X), X) \geqslant \varepsilon], \text{ где } X = Y \sqcup X_i. \tag{13}$$

Выразим условие $\delta(\mu(A_i, X), X) \geqslant \varepsilon$ в терминах Y. Обозначим $a = \mu(A_i, X)$, и пусть $n(a, Y) = s$. Тогда, используя условие $n(a, X_i) = 0$ и $n(a, X'_i) = |X'_i|$ из гипотезы 1, получим $n(a, X) = s$, $n(a, \bar{X}) = m_i - s$, $n(a, \bar{Y}) = m_i - |X'_i| - s$. Следовательно, условия переобучения для X и Y запишутся следующим образом:

$$[\delta(\mu(A_i, X), X) \geqslant \varepsilon] = \Big[s \leqslant \frac{\ell}{L}(m_i - \varepsilon k)\Big],$$
$$[\delta(\mu(A_i, Y), Y) \geqslant \varepsilon_i] = \Big[s \leqslant \frac{\ell_i}{L_i}\big(m_i - |X'_i| - \varepsilon_i k_i\big)\Big].$$

Пусть $\varepsilon_i = \frac{L_i}{\ell_i k_i}\frac{\ell k}{L}\varepsilon + \big(1 - \frac{\ell L_i}{L \ell_i}\big)\frac{m_i}{k_i} - \frac{|X'_i|}{k_i}$. Непосредственной проверкой убеждаемся, что $[\delta(\mu(A_i, X), X) \geqslant \varepsilon] = [\delta(\mu(A_i, Y), Y) \geqslant \varepsilon_i]$. Подставляя это в (13), получаем утверждение теоремы. ■

Покажем, как для произвольного разбиения $A = A_1 \sqcup A_2 \sqcup \cdots \sqcup A_t$ построить систему порождающих и запрещающих множеств. Следуя [5], введем на A отношение частичного порядка: $a \leqslant b$ тогда и только тогда, когда $I(a, x) \leqslant I(b, x), \forall x \in \mathbb{X}$. Определим $a < b$ если $a \leqslant b$ и $a \neq b$. Если $a < b$ и при этом $\rho(a, b) = 1$, то будем говорить, что a *предшествует* b, и записывать $a \prec b$.

Для отдельного алгоритма $a \in A$ порождающие и запрещающие множества определены в [5]:

$$\begin{aligned} X_a &= \{x \in X \colon \exists b \in A \colon a \prec b, I(a, x) < I(b, x)\}, \\ X'_a &= \{x \in X \colon \exists b \in A \colon b < a, I(b, x) < I(a, x)\}. \end{aligned} \tag{14}$$

Для кластера A_i положим

$$X_i = \bigcap_{a \in A_i} X_a, \quad X'_i = \bigcap_{a \in A_i} X'_a. \tag{15}$$

Лемма 14. *Множества X_i и X'_i, определенные в* (15)*, являются, соответственно, порождающим и запрещающим множествами для кластера A_i в смысле гипотезы 1.*

Доказательство. Для произвольного разбиения $X \in [\mathbb{X}]^\ell$ обозначим $a = \mu X$, и пусть $a \in A_i$. В [5] показано, что определенные в (14) множества X_a и X'_a являются порождающим и запрещающим множествами для алгоритма a, т. е. из условия $\mu X = a$ следует, что $X_a \subset X$ и $X'_a \subset \bar{X}$. Из определения X_i и X'_i следует, что $X_i \subset X_a$ и $X'_i \subset X'_a$. Следовательно, $X_i \subset X$ и $X'_i \subset \bar{X}$.

Условие «все алгоритмы $a \in A_i$ не допускают ошибок на X_i и ошибаются на всех объектах из X'_i» также следует из определения X_i и X'_i. ■

Полученные выше результаты позволяют уточнить оценку (11).

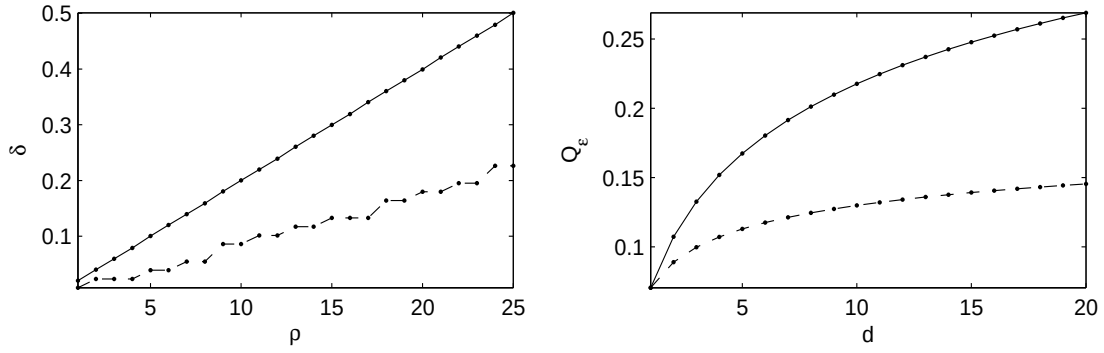


Рис. 2: Переобучение центрального слоя шара $B^m_{2\rho}$ (сплошная кривая) и локальной окрестности $\hat{B}^{m-\rho}_{2\rho,\rho}$ (пунктирная кривая) при $L = 200$, $\ell = 100$, $m = 50$. Рисунок слева отображает среднее уклонение частот ошибок на обучении и контроле в зависимости от параметра ρ. Рисунок справа отображает зависимость средней вероятности переобучения $\bar{Q}_\varepsilon(B, d)$ от параметра d при $\rho = 5$, $\varepsilon = 0,1$

Теорема 15. *В условиях теоремы 12, определений* (14) *и* (15) *для порождающих и запрещающих множеств справедлива следующая оценка вероятности переобучения:*

$$Q_\varepsilon(A) \leqslant \sum_{i=1}^{t} \frac{C^{\ell_i}_{L_i}}{C^{\ell}_{L}} Q'_{\varepsilon'}(B^{m'_i}_{r_i}) = \sum_{i=1}^{t} \Big\{ \frac{C^{\ell_i}_{L_i}}{C^{\ell}_{L}} H^{\ell_i, m'_i}_{L_i}(s(\varepsilon) + \lfloor r_i/2 \rfloor) \cdot [m_i \geqslant \varepsilon k] \Big\}, \tag{16}$$

где $s(\varepsilon) = \frac{\ell}{L}(m_i - \varepsilon k)$, $L_i = L - |X_i| - |X'_i|$, $\ell_i = \ell - |X_i|$, $m'_i = m_i - |X'_i|$.

Доказательство. Доказательство следует из теоремы 13, леммы 14 и формулы (7). ■

Главное отличие (16) от полученной ранее оценки (11) — в коэффициенте $\frac{C^{\ell_i}_{L_i}}{C^{\ell}_{L}}$, экспоненциально убывающем с ростом мощности порождающего и запрещающего множеств.

Локальная окрестность малой мощности

Отметим, что центральный слой хэммингова шара B^m_r дает достаточно грубую оценку вероятности переобучения для множества $A_i \subset B^m_r$. Во-первых, такая аппроксимация множества A_i не учитывает объекты выборки, лежащие глубоко внутри своего класса, и потому одинаково классифицируемые всеми алгоритмами кластера A_i. Во-вторых, при оценке не учитывается мощность кластера A_i, которая на реальных данных оказывается много меньше мощности множества B^m_r.

Определение 3. *Пусть все объекты генеральной выборки* $\mathbb{X}$ *разделены на три непересекающихся множества: надежно классифицируемые объекты* X_0*, ошибочно классифицируемые объекты* X_1 *и пограничные объекты* X_r*. Пусть* $|X_r| = r$ *и* $|X_1| = m$*,* ρ *— целочисленный параметр,* $\rho \leqslant r$*. Рассмотрим алгоритм* a_0*, допускающий* m *ошибок на* X_1 *и* ρ *ошибок на* X_r*. Локальной окрестностью алгоритма* a_0 *будем называть множество алгоритмов* $\hat{B}^m_{r,\rho} \subset \mathbb{A}$*, удовлетворяющее следующим условиям:*

- $\hat{B}^m_{r,\rho}$ *содержит все алгоритмы, допускающие ровно* ρ *ошибок на объектах из* X_r*,*
- *ни один алгоритм из* $\hat{B}^m_{r,\rho}$ *не ошибается на объектах из* X_0*,*
- *все алгоритмы из* $\hat{B}^m_{r,\rho}$ *ошибаются на всех объектах из* X_1*.*

На рис. 2 слева сравниваются вероятности переобучения центрального слоя хэммингова шара и локальной окрестности. Видно, что локальная окрестность дает меньшую

оценку вероятности переобучения. Следовательно, аппроксимация кластеров A_i с помощью локальных окрестностей дает более точную оценку вероятности переобучения.

Для дальнейшего уточнения оценки мы будем рассматривать кластер A_i как случайное подмножество некоторого объемлющего множества B (например, центрального слоя хэммингова шара или локальной окрестности).

Определение 4. *Средней вероятностью переобучения по подмножествам фиксированной мощности назовем следующую величину:*

$$\bar{Q}_\varepsilon(B,d) = \frac{1}{C^d_{|B|}} \sum_{A' \in [B]^d} Q_\varepsilon(A'), \tag{17}$$

где $[B]^d = \{A' \subset B\colon |A'| = d\}$ *— система подмножеств фиксированной мощности.*

На рис. 2 справа показан пример зависимости средней вероятности переобучения $\bar{Q}_\varepsilon(B,d)$ от параметра d для двух рассматриваемых нами семейств алгоритмов.

Теорема 16. *Пусть* B *— множество алгоритмов, допускающих равное число ошибок на полной выборке. Тогда выполнено следующее:*

$$\bar{Q}_\varepsilon(B,d) = 1 - \frac{1}{C^\ell_L} \sum_{\tau \in \Omega([\mathbb{X}]^\ell)} |\tau| \frac{C^d_{N_\varepsilon(B,X_\tau)}}{C^d_{|B|}}, \tag{18}$$

где $\Omega([\mathbb{X}]^\ell)$ *— множество орбит действия группы симметрий* A *на* $[\mathbb{X}]^\ell$, X_τ *— произвольный представитель орбиты* $\tau \in \Omega([\mathbb{X}]^\ell)$, $N_\varepsilon(B,X) = \sum\limits_{a \in B} [n(a,X) > s(\varepsilon)]$ *— число алгоритмов из* B*, непереобученных на разбиении* X, $s(\varepsilon) = \frac{\ell}{L}(m - \varepsilon k)$.

Доказательство. Запишем определение $\bar{Q}_\varepsilon(B,d)$ и воспользуемся тем, что $A' \subset B$ вновь является подмножеством слоя:

$$\bar{Q}_\varepsilon(B,d) = \frac{1}{C^d_{|B|}} \sum_{A' \in [B]^d} \frac{1}{C^\ell_L} \sum_{X \in [\mathbb{X}]^\ell} \left[\exists a \in A'\colon n(a,X) \leqslant s(\varepsilon)\right].$$

Переставим местами знаки суммирования и применим логическое отрицание:

$$\bar{Q}_\varepsilon(B,d) = 1 - \frac{1}{C^\ell_L} \frac{1}{C^d_{|B|}} \sum_{X \in [\mathbb{X}]^\ell} \underbrace{\sum_{A' \in [B]^d} \left[\forall a \in A'\colon n(a,X) > s(\varepsilon)\right]}_{F_\varepsilon(B,X)}.$$

Заметим, что выделенное в прошлой формуле выражение $F_\varepsilon(B,X)$ соответствует числу способов выбрать из B подмножество A' мощности d так, чтобы ни один из алгоритмов в A' не был переобученным. Обозначим через $N_\varepsilon(B,X)$ общее число алгоритмов в B, непереобученных на разбиении X. Тогда $F_\varepsilon(B,X)$ является числом сочетаний из $N_\varepsilon(B,X)$ по d:

$$\bar{Q}_\varepsilon(B,d) = 1 - \frac{1}{C^\ell_L} \frac{1}{C^d_{|B|}} \sum_{X \in [\mathbb{X}]^\ell} C^d_{N_\varepsilon(B,X)}.$$

По теоремам 7 и 8 функция $C^d_{N_\varepsilon(B,X)}$, где $N_\varepsilon(B,X) = \sum\limits_{a \in B} [n(a,X) > \varepsilon]$, является симметричной функцией третьего рода, и, следовательно, (18) факторизуется по действию группы

симметрий на множестве разбиений:

$$\bar{Q}_\varepsilon(B,d) = 1 - \frac{1}{C_L^\ell}\frac{1}{C_{|B|}^d}\sum_{\tau\in\Omega([\mathbb{X}]^\ell)} |\tau| C^d_{N_\varepsilon(B,X_\tau)}.$$

■

При больших значениях параметра d дробь $\frac{C^d_{N_\varepsilon(B,X)}}{C^d_{|B|}}$ приближенно равна $\left(\frac{N_\varepsilon(B,X_\tau)}{|B|}\right)^d$. Таким образом, вклад разбиения $(X,\bar{X})$ в вероятность переобучения полностью определяется мощностью d рассматриваемых подмножеств и числом алгоритмов из B, непереобученных на разбиении $(X,\bar{X})$.

Покажем, как вычислять оценку (18) на примере локальной окрестности $\hat{B}^m_{r,\rho}$.

Теорема 17. *Средняя вероятность переобучения случайного подмножества $A_i \subset \hat{B}^m_{r,\rho}$ фиксированной мощности d дается следующей формулой:*

$$\bar{Q}_\varepsilon(\hat{B}^m_{r,\rho},d) = 1 - \frac{1}{C_L^\ell}\sum_{i=0}^{\min(m,\ell)}\sum_{j=0}^{\min(r,\ell-i)} C_m^i C_r^j C_{L-m-r}^{\ell-i-j}\frac{C^d_{N_{i,j}}}{C^d_{|\hat{B}^m_{r,\rho}|}}, \tag{19}$$

где

$$N_{i,j} = \sum_{t=0}^{\min(j,\rho)} C_j^t C_{r-j}^{\rho-t}[i+t > s(\varepsilon)].$$

Доказательство. Рассмотрим три симметрические группы перестановок S_m, S_r и S_{L-m-r}, действующие на множествах X_1, X_r и X_0, соответственно. Группой симметрий множества алгоритмов $\hat{B}^m_{r,\rho}$ является декартово произведение $S_m \times S_r \times S_{L-m-r}$. Орбиты действия $\mathrm{Sym}(\hat{B}^m_{r,\rho})$ на $[\mathbb{X}]^\ell$ индексируются двумя параметрами, $i = |X\cap X_1|$ и $j = |X\cap X_r|$, где X — обучающая выборка. Мощность орбиты $\tau_{i,j}$ дается, соответственно, выражением $|\tau_{i,j}| = C_m^i C_r^j C_{L-m-r}^{\ell-i-j}$.

Разобравшись с симметриями, необходимо для представителя $X_{i,j} \in \tau_{i,j}$ вычислить величину $N_{i,j} = N(\hat{B}^m_{r,\rho}, X_{i,j})$ — количество алгоритмов из $\hat{B}^m_{r,\rho}$, непереобученных на разбиении $X_{i,j}$. Рассмотрим произвольный алгоритм $a \in \hat{B}^m_{r,\rho}$ и обозначим через t количество ошибок данного алгоритма на $X\cap X_r$. Тогда данный алгоритм делает $i+t$ ошибок на обучении, и, следовательно, условие того, что он не переобучен, записывается в виде $i+t > s(\varepsilon)$, где $s(\varepsilon) = \frac{\ell}{L}(m+\rho-\varepsilon k)$. Количество алгоритмов в $\hat{B}^m_{r,\rho}$ с данным значением параметра t равно $C_j^t C_{r-j}^{\rho-t}$. Суммируя по t, получим количество непереобученных алгоритмов:

$$N_{i,j} = \sum_{t=0}^{\min(j,\rho)} C_j^t C_{r-j}^{\rho-t}[i+t > s(\varepsilon)].$$

Тогда, по теореме 16, вероятность переобучения случайного подмножества $\hat{B}^m_{r,\rho}$, состоящего из d алгоритмов, дается следующей формулой:

$$\bar{Q}_\varepsilon(\hat{B}^m_{r,\rho},d) = 1 - \frac{1}{C_L^\ell}\sum_{i=0}^{\min(m,\ell)}\sum_{j=0}^{\min(r,\ell-i)} |\tau_{i,j}|\frac{C^d_{N_{i,j}}}{C^d_{|\hat{B}^m_{r,\rho}|}}, \tag{20}$$

где $\tau_{i,j}$ и $N_{i,j}$ определены выше. ■

Теорема 17 позволяет уточнить верхнюю оценку вероятности переобучения (16). Для этого будем оценивать $Q_\varepsilon(A_i)$ с помощью $\bar{Q}_\varepsilon(B^m_{r,\rho})$, где d — мощность A_i, m — число объектов, на которых ошибаются все $a \in A_i$, r — число таких объектов, для которых хотя бы два алгоритма из A_i дают разную классификацию, ρ — среднее число ошибок алгоритмов из A_i на множестве X_r. Отметим, что в данном случае оценка $Q_\varepsilon(A_i) \leqslant \bar{Q}_\varepsilon(\hat{B}^m_{r,\rho})$ является эвристикой.

Эксперимент

Проведем экспериментальное сравнение оценки расслоения-сходства (12) с тремя комбинаторными оценками (VC- и SC-оценками из [6], ES-оценкой из [14]) и двумя PAC-Bayesian оценками из [15].

В качестве исходных данных были взяты 11 задач из репозитория UCI [16]. Описание задач приводится в табл. 2. На этапе предобработки удалялись объекты с хотя бы одним пропущенным признаком. После этого каждый признак нормировался в интервал $[0, 1]$. Для многоклассовых задач целевые классы были вручную сгруппированы в два класса.

Таблица 2: Описание задач

Задача	#Объектов	#Признаков	Задача	#Объектов	#Признаков
Glass	214	9	Statlog	2310	19
Liver dis.	345	6	Wine	4898	11
Ionosphere	351	34	Waveform	5000	21
Australian	690	6	Pageblocks	5473	10
Pima	768	8	Optdigits	5620	64
Faults	1941	27			

Для каждой задачи мы выполняли процедуру пятикратной кросс-валидации, которая запускалась 100 раз для усреднения результатов. Таким образом, для каждой задачи мы генерировали $M = 500$ разбиений $\mathbb{X} = \mathbb{X}^i_L \sqcup \mathbb{X}^i_K$, $i = 1, \ldots, M$ и вычисляли оценку Монте-Карло для среднего уклонения частот ошибок логистической регрессии:

$$\hat{\delta}_L(\mu_{\text{LR}}, \mathbb{X}) = \frac{1}{M} \sum_{i=1}^{M} \nu(\mu_{\text{LR}}\mathbb{X}^i_L, \mathbb{X}^i_K) - \nu(\mu_{\text{LR}}\mathbb{X}^i_L, \mathbb{X}^i_L).$$

После этого каждая обучающая выборка $\mathbb{X}_L$ использовалась для вычисления комбинаторной оценки на уклонение частот ошибок МЭР. Множества алгоритмов A, из которого МЭР выбирал лучший алгоритм, генерировалось с помощью случайных блужданий по графу расслоения-связности линейных классификаторов [14]. В качестве начального приближения для случайного блуждания использовался алгоритм $\mu_{\text{LR}}\mathbb{X}_L$, настроенный логистической регрессией по обучающей выборке $\mathbb{X}_L$. Далее вновь использовался метод Монте-Карло: генерировались $M' = 4096$ случайных разбиений $\mathbb{X}_L = X^j_\ell \sqcup X^j_k$, $j = 1, \ldots, M'$ (при $\frac{\ell}{L} = 0.8$) и вычислялась следующая величина:

$$\hat{\delta}_\ell(\mu, \mathbb{X}_L) = \frac{1}{M'} \sum_{j=1}^{M'} \nu(\mu X^j_\ell, X^j_k) - \nu(\mu X^j_\ell, X^j_\ell),$$

где μ — метод минимизации эмпирического риска. В заключение эта величина усреднялась по всем разбиениям $\mathbb{X} = \mathbb{X}^i_L \sqcup \mathbb{X}^i_K$. Величины $\hat{\delta}_L(\mu_{\text{LR}}, \mathbb{X})$ и $\hat{\delta}_\ell(\mu, \mathbb{X}_L)$ соответствуют

Таблица 3: Сравнение фактического переобучения и различных оценок

	Ошибка	Переобучение		Комбинаторные оценки				PAC-Bayes	
Task	Тест	$\hat{\delta}_L(LR)$	$\hat{\delta}_\ell(\mu)$	VC	SC	ES	AF	DI	DD
glass	0.076	0.030	0.067	0.191	0.127	0.124	0.106	1.268	0.740
Liver dis.	0.315	0.017	0.046	0.249	0.192	0.146	0.161	1.207	1.067
Ionosphere	0.126	0.079	0.042	0.138	0.099	0.087	0.084	1.219	1.149
Australian	0.136	0.014	0.023	0.130	0.101	0.081	0.086	1.145	0.678
pima	0.227	0.007	0.021	0.151	0.117	0.090	0.098	0.971	0.749
faults	0.210	0.011	0.008	0.091	0.070	0.046	0.060	1.110	1.054
statlog	0.142	0.004	0.008	0.072	0.060	0.043	0.051	1.102	0.746
wine	0.250	0.002	0.003	0.061	0.047	0.032	0.040	0.776	0.637
waveform	0.105	0.003	0.003	0.043	0.033	0.023	0.023	0.561	0.354
pageblocks	0.051	0.001	0.003	0.030	0.022	0.016	0.018	0.739	0.186
Optdigits	0.121	0.006	0.003	0.043	0.034	0.023	0.026	1.068	0.604

реальному переобучению логистической регрессии и его идеальной оценке в рамках комбинаторного подхода.

В табл. 3 сравниваются следующие величины: $\hat{\delta}_L(\mu_{\text{LR}}, \mathbb{X})$ и $\hat{\delta}_\ell(\mu, \mathbb{X}_L)$; четыре верхние комбинаторные оценки величины $\hat{\delta}_\ell(\mu, \mathbb{X}_L)$, обозначенные как VC, SC, ES и AF; две PAC-Bayes оценки (обозначены как DD и DI). В отличие от комбинаторных оценок, ограничивающих уклонение частот ошибок, PAC-Bayes оценки справедливы непосредственно для частоты ошибок на контроле (приведена в столбце «Ошибка тест»). DD-оценка учитывает размерность пространства признаков и является более точной по сравнению с универсальной DI-оценкой, справедливой для любого числа признаков.

Комбинаторная SC-оценка соответствует оценке расслоения-связности из [6]. VC-оценка также приведена в [6], и она, в отличие от SC-оценки, не учитывает ни расслоение, ни связность. ES-оценка [14] основана на более тонком учете расслоения, при котором каждый алгоритм сравнивается со всем множеством найденных истоком графа расслоения-связности.

AF-оценка, предлагаемая нами в данной статье, получена из оценки расслоения-сходства (12). Чтобы полностью конкретизировать оценку (12), необходимо уточнить следующее: метод разбиения исходного множества алгоритмов A на кластеры, способ выбора порождающих и запрещающих множеств для каждого кластера, способ оценивания вероятности переобучения каждого кластера. В AF-оценке порождающие и запрещающие множества выбирались в соответствии с (15), для оценки вероятности переобучения каждого кластера используется формула (19), а представление множества алгоритмов A в виде $A = A_1 \sqcup \cdots \sqcup A_t$ производится с помощью иерархической кластеризации при выборе расстояния дальнего соседа. Исходная метрика на A определяется как хэммингово расстояние между векторами ошибок алгоритмов.

Из экспериментальных данных следует, что переобучение логистической регрессии хорошо приближается комбинаторной оценкой $\hat{\delta}_\ell(\mu, \mathbb{X}_L)$ для МЭР. Все четыре комбинаторные оценки существенно точнее обеих PAC-bayes оценок. Среди комбинаторных оценок наименее завышенной оказывается ES-оценка. За ней следует предложенная нами AF-оценка. Каждая из этих оценок существено уточняет SC-оценку расслоения-связности. Улучшение точности в ES- и AF-оценках основано на двух различных эффектах — более тонком учете расслоения в ES-оценке и учете сходства алгоритмов в AF-оценке. Представ-

ляется возможным, что объединение ES- и AF-оценок позволит добиться еще большего качества комбинаторных оценок вероятности переобучения.

Выводы

В данной работе предложена новая оценка вероятности переобучения, учитывающая расстояния между векторами ошибок алгоритмов. Оценка основана на разложении множества алгоритмов на кластеры, т. е. на непересекающиеся подмножества, состоящие из алгоритмов с похожими векторами ошибок. Для каждого такого кластера применяется верхняя оценка вероятности переобучения, учитывающая хэммингов диаметр кластера и его мощность. По аналогии с уже существующими оценками расслоения-связности доказана новая оценка, одновременно учитывающая и эффект расслоения по числу ошибок, и эффект сходства алгоритмов. Вывод данной оценки существенно опирается на теоретико-групповой подход. Эффективность полученных оценок продемонстрирована на примере 11 задач из репозитория UCI. Показано, что предлагаемый метод в ряде случаев дает более точную оценку переобучения по сравнению с уже известным оценками.

Литература

[1] *Вапник В. Н., Червоненкис А. Я.* Теория распознавания образов. М.: Наука. 1974.

[2] *Vapnik V.* Statistical learning theory. New York: Wiley. 1998.

[3] *Воронцов К. В.* Точные оценки вероятности переобучения // Докл. РАН. 2009. Т. 429, № 1. С. 15–18.

[4] *Vorontsov K. V.* Splitting and similarity phenomena in the sets of classifiers and their effect on the probability of overfitting // Pattern Recognition and Image Analysis. 2009. Vol. 19, №. 3. Pp. 412–420.

[5] *Vorontsov K. V.* Exact combinatorial bounds on the probability of overfitting for empirical risk minimization // Pattern Recognition and Image Analysis. 2010. Vol. 20, №. 3. Pp. 269–285.

[6] *Vorontsov K. V., Ivahnenko A. A., Reshetnyak I. M.* Generalization bound based on the splitting and connectivity graph of the set of classifiers // Pattern Recognition and Image Analysis: new information technologies (PRIA-10). St. Petersburg, Russian Federation. 2010.

[7] *Фрей А. И.* Вероятность переобучения плотных и разреженных многомерных сеток алгоритмов // Междунар. конф. ИОИ-8. М.: МАКС Пресс, 2010. С. 87–90.

[8] *Vorontsov K. V., Ivahnenko A. A.* Tight combinatorial generalization bounds for threshold conjunction rules // 4th International Conference on Pattern Recognition and Machine Intelligence. Lecture notes in computer science ser. Springer-Verlag, 2011. Pp. 66–73.

[9] *Фрей А. И., Ивахненко А. А, Решетняк И. М.* Применение комбинаторных оценок вероятности переобучения в простом голосовании конъюнкций // Междунар. конф. ИОИ-9. М.: МАКС Пресс, 2012. С. 86–89.

[10] *E. Bax.* Similar classifiers and VC error bounds. Tech. Rep. CalTech-CS-TR97-14. 1997.

[11] *Фрей А. И.* Точные оценки вероятности переобучения для симметричных семейств алгоритмов // Всеросс. конф. ММРО-14. М.: МАКС Пресс, 2009. С. 66–69.

[12] *Фрей А. И.* Точные оценки вероятности переобучения для симметричных семейств алгоритмов и рандомизированных методов обучения // Pattern Recognition and Image Analysis. 2010.

[13] *Толстихин И. О.* Вероятность переобучения плотных и разреженных семейств алгоритмов // Междунар. конф. ИОИ-8. М.: МАКС Пресс, 2010. С. 83–86.

[14] *Соколов Е. А., Воронцов К. В.* Минимизация вероятности переобучения для композиций линейных классификаторов малой размерности // Междунар. конф. ИОИ-9. М.: Торус Пресс, 2012. С. 82–85.

[15] Jin C., Wang L. Dimensionality dependent PAC-Bayes margin bound // *Advances in neural information processing systems*. 2012. Vol. 25. Pp. 1043–1051.

[16] *K. Bache, M. Lichman.* UCI machine learning repository // University of California, Irvine, School of Information and Computer Sciences. 2013.

Формирование признакового описания фактуры картин*

***Д. М. Мурашов*[1], *А. В. Березин*[2], *Е. Ю. Иванова*[2]**
d_murashov@mail.ru, berezin_alex@mail.ru
[1]Москва, Вычислительный центр им. А. А. Дородницына РАН; [2]Москва, Государственный Исторический музей

Рассматривается задача формирования признакового пространства для сравнения изображений в атрибуции произведений живописи. Предложено признаковое описание фактуры картин на основе характеристик хребтов полутонового рельефа, элементов структурного тензора и значений локальных волновых чисел. В отличие от известных разработок признаковое описание формируется только по информативным фрагментам изображений и не требует предварительной сегментации отдельных мазков кисти. Проведены вычислительные эксперименты. Предложенное признаковое описание позволит получить количественную характеристику стиля живописи автора и наряду с другими видами исследования картин сформировать атрибуционное заключение.

Ключевые слова: *фактура картин, признаки изображений, локальная ориентация хребтов изображения, структурный тензор, мера когерентности, преобразование Рисса, атрибуция картин.*

Composing feature description of paintings texture*

***D. M. Murashov*[1], *A. V. Berezin*[2], *Y. Yu. Ivanova*[2]**
[1]Moscow, Dorodnicyn Computing Centre of RAS; [2]Moscow, State Historical Museum

A task of composing a feature description of texture of paintings for the purposes of attribution is considered. A feature description for comparing texture fragments based on ridges of grayscale image relief, structure tensor, and local wave number is proposed. As compared to conventional techniques, the selected features are computed only in informative image fragments and do not require brush stroke segmentation. The results of computing experiments showed the efficiency of the proposed feature set. The feature set gives quantitative description of painter's artistic style and provides suitable accuracy of feature computing. The proposed feature set may be used as a part of technological description of fine art paintings for attribution.

Keywords: *texture of paintings, image features, local orientation of image ridges, structure tensor, coherency measure, Reisz transform, attribution of paintings.*

Введение

Рассматривается задача сравнения изображений для целей атрибуции произведений живописи. Под термином «атрибуция» понимается определение принадлежности неподписанного художественного произведения тому или иному автору, школе, времени, стране и т. д. [1].

Одно из направлений исследований в атрибуции связано с анализом цифровых изображений картин. Важнейшим техническим признаком картины является фактура. Понятие фактуры определено как видимое поверхностное строение картины [2]. Идея применения методов анализа изображений в атрибуции заключается в сравнении изображений

Работа выполнена при финансовой поддержке РФФИ, проект № 12-07-00668.

аутентичных картин и исследуемой картины по признакам, характеризующим индивидуальность художника.

В публикациях различных исследовательских групп предложено несколько подходов к решению рассматриваемой задачи. Один из подходов основан на переборном сравнении квадратных фрагментов, на которые разбиваются оба исследуемых изображения [3]. Признаками, как правило, служат коэффициенты ортогональных преобразований (в частности, вейвлет-преобразований). Такие методы требуют больших вычислительных затрат и весьма чувствительны к условиям получения изображений и параметрам аппаратных средств [4]. В работах [5] и [6] на изображениях находятся специфические объекты, по которым сравниваются изображения. Такими объектами являются отдельные мазки кисти художника. Однако выделить в автоматическом режиме отдельные мазки удается далеко не на всех полотнах. Это удается сделать, например, на картинах Ван Гога и некоторых других. Поэтому предпочтительно использовать признаки, для вычисления которых не требуется сегментация отдельных мазков.

Индивидуальность художника проявляется в особенностях мазков кисти, обусловленных системой наложения мазков, выбором кистей, нажимом. Индивидуальность выявляется также в подходе к проработке света, полутени и тени, в соотношении расположения мазков по границам формы и фона, а также по границам отдельных деталей [7].

В соответствии с рекомендациями искусствоведов предлагается в качестве образцов для сравнения использовать группу мазков, формирующих какую-либо деталь, или границы касания предметов друг с другом и фоном.

Задача формулируется следующим образом. Требуется сравнить изображения по набору признаков, вычисляемых на информативных фрагментах. Предполагается, что имеется два изображения (эталонное и тестовое), на которых присутствуют группы информативных фрагментов, принадлежащих некоторому классу (например, фрагменты на которых изображен нос или другая часть лица, фрагменты с деталями одежды). Каждый класс характеризуется определенным набором признаков. Число классов априори не известно. Необходимо: выделить на изображениях информативные объекты; классифицировать эти объекты и установить соответствие объектов одинаковых классов на изображениях; сравнить соответствующие фрагменты по выбранному набору признаков.

Решение задачи состоит из следующих этапов: (а) поиск однотипных информативных фрагментов на сравниваемых изображениях; (б) вычисление признаков; (в) сравнение найденных однотипных фрагментов; (г) заключение о схожести изображений по множеству найденных фрагментов.

Данная работа посвящена этапам (б) и (в) задачи сравнения изображений картин. Выбраны отличные от предложенных ранее элементы признакового описания, отражающего стиль живописи художника, и предлагается методика сравнения информативных фрагментов картин. Представлены результаты вычислительных экспериментов.

Исходные данные

Исходными данными являются цветные изображения фрагментов картин (портретов), выполненных разными художниками XVIII–XIX вв. Изображения фиксируются цифровой фотокамерой. Размеры фрагментов (например, лиц) составляют 4272×2848 пикселей. Размеры используемых в данной работе информативных областей составили около 1800×1000 пикселей. Разрешение составило 200 точек на 1 см, что соответствует качеству исходных данных для такого рода исследований. Так, в аналогичных работах [3], [4] и [6] иссле-

довались изображения с разрешением 196 точек на дюйм. На полученных изображениях корректировались искажения, обусловленные процессом съемки.

Признаковое описание фактуры картин

С учетом рекомендаций искусствоведов и анализа публикаций по тематике настоящего исследования будут использованы признаки, не связанные с сегментацией отдельных мазков кисти, которые будут извлекаться только из областей изображения, содержащих максимум информации о технике живописи художника. В представляемой работе предлагается использовать текстурные и частотные признаки информативных фрагментов. Для описания индивидуальных особенностей техники живописи предлагается использовать следующие признаки: локальная ориентация хребтов полутонового рельефа изображения, локальные признаки на основе структурного тензора (локальная ориентация и когерентность); локальные волновые числа. Локальная ориентация и когерентность характеризуют манеру живописи художника, специфическую для конкретных деталей картины. Волновое число связано с геометрическими характеристиками кисти, которую использовал автор в процессе формирования красочного слоя рассматриваемого фрагмента (толщина волоса, ширина кисти).

Гистограмма направлений хребтов мазков. В качестве характеристики группы мазков предлагается рассматривать гистограмму направлений хребтов мазков. Гистограмма направлений хребтов мазков может быть получена непосредственно из изображения и не требует операции сегментации отдельных мазков. Гистограмма формируется по значениям составляющих градиента уровней полутонов в точках, расположенных на хребтах информативных фрагментов. Хребты локализуются на основе метода, описанного в [8]. Множество точек, образующих хребет объекта полутонового изображения, дополнено параболическими и омбилическими точками поверхности, аппроксимирующей рельеф изображения.

Модель мазка представляется в виде поверхности, описываемой функцией $f(x,y)$, $f \in C^2(\mathbb{R}^2,\mathbb{R})$. Предполагается, что градиент $\boldsymbol{Df} \neq \mathbf{0}$, $\boldsymbol{Df} = (f_x, f_y)^\intercal$. Введем обозначения $\boldsymbol{N} = \boldsymbol{Df}/|\boldsymbol{Df}|$, $\boldsymbol{T} = \boldsymbol{Df}^\perp/|\boldsymbol{Df}|$, $\boldsymbol{Df}^\perp = (-f_x, f_y)^\intercal$, где $\boldsymbol{N}$ и $\boldsymbol{T}$ – нормальный и касательный векторы к линиям уровня (изофотам) f. Имеет место следующее выражение на основе матрицы Гессе $\boldsymbol{D^2 f}$ функци $f(x,y)$ [8]:

$$-\frac{1}{|\boldsymbol{D}f|}\begin{bmatrix} \boldsymbol{N}^\intercal\boldsymbol{D^2 f}\boldsymbol{N} & \boldsymbol{N}^\intercal\boldsymbol{D^2 f}\boldsymbol{T} \\ \boldsymbol{T}^\intercal\boldsymbol{D^2 f}\boldsymbol{N} & \boldsymbol{T}^\intercal\boldsymbol{D^2 f}\boldsymbol{T} \end{bmatrix} = \begin{bmatrix} g & \mu \\ \mu & k \end{bmatrix},$$

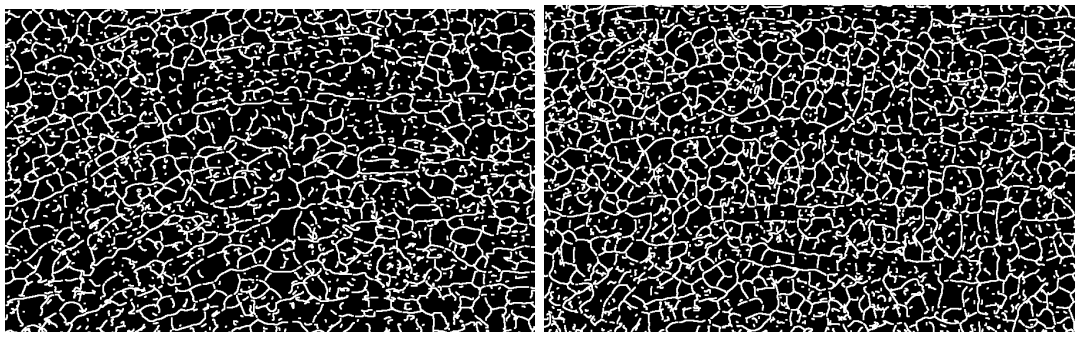
где $k = -\boldsymbol{T}^\intercal\left(\boldsymbol{D^2 f}/|\boldsymbol{Df}|\right)\boldsymbol{T}$ – кривизна изофоты, а $\mu = -\boldsymbol{T}^\intercal\left(\boldsymbol{D^2 f}/|\boldsymbol{Df}|\right)\boldsymbol{N}$ – кривизна линий потока яркости; $g = -\boldsymbol{N}^\intercal\left(\boldsymbol{D^2 f}/|\boldsymbol{Df}|\right)\boldsymbol{N}$ – мера изменения величины градиента вдоль линий потока. Точки, образующие хребты f удовлетворяют условиям: $\mu = 0$ и $k > \max(0, g)$. Примеры фрагментов красочного слоя и карт хребтов показаны на рис. 1 и рис. 2. Полученные хребты дефрагментируются на непересекающиеся связные компоненты.

Для получения гистограммы направлений хребтов вычисляется угол ориентации осей инерции связных компонент хребтов [9]:

$$\theta = \frac{1}{2}\arctan\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}},$$

(а) (б)

Рис. 1: Пример фрагментов изображений портретов разных авторов



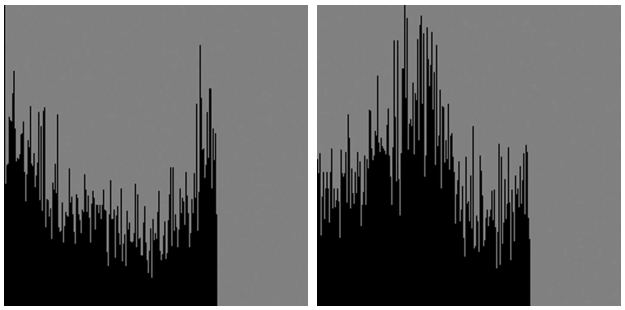
(а) (б)

Рис. 2: Изображения хребтов фрагментов, показанных на рис. 1

где $\mu_{i,j}$ — компоненты тензора инерции объекта

$$\boldsymbol{J} = \begin{bmatrix} \mu_{2,0} & -\mu_{1,1} \\ -\mu_{1,1} & \mu_{0,2} \end{bmatrix}.$$

При построении гистограммы направлений учитывалась длина (в пикселях) связных компонент хребтов. Полученные гистограммы представлены на рис. 3. Значения углов находятся в интервале от 0 до 180^o. Из рисунка видно различие распределений углов наклона осей инерции.



(а) (б)

Рис. 3: Гистограммы ориентации хребтов фрагментов, показанных на рис. 1

Признаки на основе структурного тензора. Другой признак, характеризующий локальную ориентацию текстуры картины, – локальная ориентация окрестности, определяемая структурным тензором (матрицей вторых моментов в точке $\boldsymbol{x}$, взвешенной оконной функцией):

$$\boldsymbol{\mu}_f(\boldsymbol{x}) = \int\limits_{\boldsymbol{z}\in\mathbb{R}^2} (\boldsymbol{Df}(\boldsymbol{z}))(\boldsymbol{Df}(\boldsymbol{z}))^{\mathsf{T}} w(\boldsymbol{x}-\boldsymbol{z})\, d\boldsymbol{z},$$

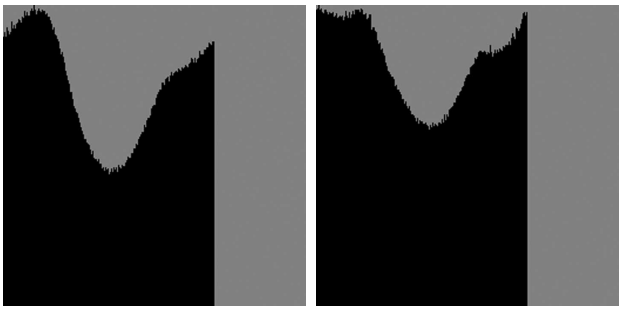
где $w(\boldsymbol{x}-\boldsymbol{z})$ – оконная гауссова функция [10]. Угол локальной ориентации φ $\boldsymbol{\mu}_f(\boldsymbol{x})$ определяется аналогично углу ориентации осей инерции θ:

$$\varphi = \frac{\pi}{2} + \frac{1}{2}\arctan\frac{2\mu_{f1,1}}{\mu_{f2,0}-\mu_{f0,2}},$$

где $\mu_{fi,j}$ – компоненты структурного тензора

$$\boldsymbol{\mu}_f = \begin{bmatrix} \mu_{f2,0} & -\mu_{f1,1} \\ -\mu_{f1,1} & \mu_{f0,2} \end{bmatrix}.$$

В [3] локальная ориентация текстуры определялась по энергии откликов 12 ориентированных фильтров Габора. Используемый в данной работе метод на основе матрицы вторых моментов позволяет оценить угловые величины с точностью в пределах одного градуса при размере окна 5 пикселей и $\sigma = 1$. Для фрагментов, приведенных на рис. 1, гистограммы локальной ориентации, полученные с использованием матрицы вторых моментов, показаны на рис. 4.



(а) (б)

Рис. 4: Гистограммы локальной ориентации текстуры для фрагментов, показанных на рис. 1

Локальная мера когерентности градиентов изображения вычисляется по формуле [9]:

$$c_c = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2},$$

где λ_1, λ_2 – собственные значения матрицы вторых моментов μ_f в точке $\boldsymbol{x}$. Локальная мера когерентности применяется для формирования маски областей, в которых будет вычисляться угол локальной ориентации φ текстуры.

Локальное волновое число. Значение локального волнового числа определяется из выражения [9]:

$$k_w = \frac{p(q_{1x}+q_{2y}) - q_1 p_x - q_2 p_y}{p^2 + q_1^2 + q_2^2},$$

где p, q_1 и q_2 – составляющие моногенного сигнала [11], получаемого из изображения преобразованием Рисса, а p_x, p_y, q_{1x} и q_{2y} – частные производные этих составляющих по пространственным координатам.

Сравнение фрагментов изображений

Для сравнения фрагментов изображений произведений живописи применяются методы статистического анализа [6], методы кластерного анализа и распознавания [3], [4]. Так как музеи, как правило, не располагают достаточным количеством картин одного автора, то затруднительно получить достаточное количество данных для обучения классификаторов. В этом случае для сравнения фактуры картин применяются статистические тесты. Например, в работе [6] при помощи перестановочного теста проверяется гипотеза о равенстве средних значений признаков, вычисленных по изображениям исследуемых картин. В представляемой работе Фрагменты изображений будут сравниваться с помощью статистических тестов и теоретико-информационной меры различия.

Для сравнения фрагментов по распределениям угла ориентации хребтов применялся тест Смирнова [12]. Полученные гистограммы являются приближениями плотности вероятности углов ориентации θ, построенными по выборкам их значений $\vartheta_{11}, ..., \vartheta_{1m}$ и $\vartheta_{21}, ..., \vartheta_{2n}$, вычисленных для однотипных фрагментов двух разных картин. Пусть $F(\theta_1)$ и $G(\theta_2)$ — эмпирические функции распределения, построенные по гистограммам рассматриваемых угловых величин для фрагментов двух картин. Предполагается, что выполнены все предположения о выборках случайных величин и их функциях распределения [13]. В качестве нулевой гипотезы H_0 выдвигается гипотеза об однородности функций распределения величин θ_1 и θ_2 против гипотезы неоднородности H_1. Статистикой критерия служит величина $D = \sup |F(\theta) - G(\theta)|$. Гипотезу H_0 следует отвергнуть, если $D > d$, где d – величина, зависящая от значения функции распределения Колмогорова при данных m и n и выбранном уровне значимости α.

Получены результаты тестирования признаков фактуры однотипных фрагментов двух портретов кисти Ф. Рокотова и двух портретов других авторов VIII–XIX веков. Значения статистики D критерия Смирнова для распределений угла ориентации хребтов однотипных фрагментов изображений приведены в табл. 1, а решения о принятии гипотезы H_0 при уровне значимости $\alpha = 0,01$ приведены в табл. 2. В таблицах через $R1$ и $R2$ обозначены два портрета кисти Ф. Рокотова, M и F – мужской и женский портреты кисти двух других художников. В табл. 2 единица означает, что гипотеза об однородности принимается, а ноль — гипотеза отвергается.

Теоретико-информационная мера различия формируется на основе дивергенции Кульбака-Лейблера [14] в виде следующего выражения:

$$D_{KL} = \frac{1}{2}\left(\sum_{u \in H} p_U(u) \log\left(\frac{p_U(u)}{q_U(u)}\right) + \sum_{u \in H} q_U(u) \log\left(\frac{q_U(u)}{p_U(u)}\right)\right),$$

где $p_U(u)$ и $q_U(u)$ – вероятности того, что значения углов ориентации векторов на сравниваемых фрагментах принимают значение u; H – алфавит U. Углы локальной ориентации вычисляются по компонентам структурного тензора только в областях высокого уровня когерентности.

Значения теоретико-информационной меры различия распределений направлений векторов, ортогональных векторам локальной ориентации текстуры, приведены в табл. 3. Данные, приведенные в таблице, показывают, что значения признака фактуры фрагмен-

Таблица 1: Значения статистики D для распределений угла ориентации хребтов

Картина	$R1$	$R2$	M	F
$R1$	0.	0.041	0.279	0.051
$R2$	0.041	0.	0.3	0.082
M	0.279	0.3	0.	0.237
F	0.051	0.082	0.237	0.

Таблица 2: Результат теста однородности распределений угла ориентации хребтов

Картина	$R1$	$R2$	M	F
$R1$	1	1	0	0
$R2$	1	1	0	0
M	0	0	1	0
F	0	0	0	1

Таблица 3: Значения меры различия D_{KL} распределений угла локальной ориентации текстуры

Картина	$R1$	$R2$	M	F
$R1$	0.	0.009	0.56	0.049
$R2$	0.009	0.	0.617	0.08
M	0.56	0.617	0.	0.4
F	0.049	0.08	0.4	0.

тов картин разных авторов различаются как минимум в 5 раз больше, чем значения признака на картинах одного автора.

Заключение

Предложено признаковое описание фрагментов картин на основе локальных пространственных и частотных характеристик изображений. Предложенные признаки на основе свойств хребтов полутонового рельефа изображения и элементов локального структурного тензора позволяют получить количественные характеристики фактуры картины, отражающей стиль живописи автора, а локальное волновое число позволяет оценить размеры кисти, использовавшейся художником. Используемые признаки обладают разделяющими свойствами. Вычисление признаков в отличие от известных подходов не требует предварительной сегментации отдельных мазков кисти и нечувствительно к вариациям условий получения изображений. Проведено тестирование предложенного признакового описания, которое показало его эффективность для сравнения фрагментов картин. Полученное описание фактуры позволит наряду с другими видами технико-технологических исследований сделать атрибуционное заключение. Дальнейшие исследования будут направлены на расширение признакового описания фактуры, увеличение базы изображений картин, накопление экспериментальных данных и построение алгоритма формирования решения о схожести манеры живописи.

Литература

[1] *Обухов Г. Г.* Краткий словарь терминов изобразительного искусства. М.: Советский художник, 1959.

[2] *Рыбников А. А.* Фактура классической картины. М.: Государственная Третьяковская Галерея, 1927.

[3] *Johnson C. R., Hendriks E., Berezhnoy I. J., Brevdo E., Hughes S. M., Daubechies I., Li J., Postma E., Wang J. Z.* Image processing for artist identification (Computerized analysis of Vincent van Gogh's painting brushstrokes) // Signal Processing Magazine, IEEE, 2008. Vol. 25, No. 4. Pp. 37–48.

[4] *Polatkan G., Jafarpour S., Brasoveanu A., Hughes S., Daubechies I.* Detection of forgery in paintings using supervised learning) // ICIP2009, IEEE, 2009. Pp. 2921–2924.

[5] *Sablatnig R. Kammerer P. Zolda E.* Structural analysis of paintings based on brush strokes. // Proc. of SPIE Scientific Detection of Fakery in Art, SPIE. 1998. V. 3315. Pp. 87–98.

[6] *Li J. Yao L. Hendriks E. Wang J. Z.* Rhythmic brushstrokes distinguish van Gogh from his contemporaries: Findings via automated brushstroke extraction // IEEE TPAMI. 2012. V. 34, No. 6. С. 1159–1176.

[7] *Игнатова Н. С.* Анализ фактуры живописного произведения. // Основы экспертизы произведений масляной живописи. Методические рекомендации. Вып. 1. М.: Всероссийикий художественный научно-реставрационный центр им. Академика И. Э. Грабаря, 1994. С. 15–26.

[8] *Eberly D.* Ridges in image and data analysis. Dordrecht/Boston/ London: Klewer Academic Publishers, 1996. 213 pp.

[9] *Jahne B.* Digital Image Processing. 6th ed. Berlin: Springer-Verlag, 2005. 584 pp.

[10] *Lindeberg T.* Scale-space theory in computer vision. The Kluwer International Series in Engineering and Computer Science. Dordrecht/Boston/ London: Klewer Academic Publishers, 1994. 423 pp.

[11] *Felsberg M. Sommer G.* The monogenic signal // IEEE Transactions on Signal Processing, 2001. Vol. 49, No. 12. Pp. 3136–3144.

[12] *ван дер Варден Б. Л.* Математическая статистика. М.: Издательство иностранной литературы, 1960. 435 с.

[13] *Лагутин М. Б.* Наглядная математическая статистика. М.: БИНОМ, 2011. 472 с.

[14] *Escolano F., Suau P., Bonev B.* Information theory in computer vision and pattern recognition. London: Springer Verlag, 2009.

Динамическая сегментация последовательности кадров*

Хашин С. И.
khash2@mail.ru, http://math.ivanovo.ac.ru/dalgebra/Khashin/
Ивановский государственный университет

Описан алгоритм сегментации, основанный не на одном кадре, а на паре соседних кадров из видеопоследовательности. По сравнению с обычной, статической сегментацией каждого кадра по отдельности качество значительно повышается. При сохранении той же погрешности количество сегментов удается сократить в несколько десятков раз. Типичное количество сегментов, нужных для получения приемлемой погрешности, уменьшается с $200-500$ до $5-15$.

Ключевые слова: *сегментация, последовательности кадров, клеточный автомат.*

Dynamic segmentation of frames sequences*

Khashin S. I.
Ivanovo State University

The image segmentation algorithm, based not on a single frame, but on the pair of frames from a video sequence is described. Compared to usual, static segmentation of each frame individually, the quality is improved drastically. While maintaining the same error, the number of segments can be reduced in tens times. Typical number of segments needed to produce an acceptable error is reduced from 200–500 to 5–15.

Keywords: *segmentation, frames sequences, cellular automaton.*

Введение

В основе алгоритмов сжатия видеоинформации лежит идея построения прогноза текущего кадра на основе предыдущего (или предыдущих). Во всех применяемых сегодня алгоритмах [1, 2, 3] для построения прогноза текущего кадра на основе предыдущего он разбивается на квадраты или прямоугольники небольшого размера (от 32×32 до 4×4) и для каждого из них находится *вектор движения*, с помощью этих векторов строится прогноз текущего кадра, находится *дифференциальный кадр, displaced frame difference (DFD)*, который тем или иным образом кодируется. Эти методы позволяют в несколько раз увеличить коэффициент сжатия без потери качества по сравнению со сжатием отдельных статических изображений.

Недостатком такого подхода является то, что движущиеся объекты плохо аппроксимируются прямоугольниками и для получения приемлемого результата приходится разбивать кадр на очень мелкие блоки.

В работах [4, 5, 6] предложен новый, объектно-ориентированный алгоритм сжатия. Его основа — сегментация предыдущего кадра, т. е. разбиение его не на прямоугольники, а на сегменты произвольной формы.

Различных алгоритмов сегментации существует достаточно много, см., например, [7, 8]. Но экспериментальная проверка показывает, что, с нашей точки зрения, различия меж-

Работа выполнена при финансовой поддержке РФФИ, проект № 11-07-00653.

ду ними не так велики: погрешность сегментации (см. ниже определение 9) различается на 10–15ожидать появления новых методов сегментации, существенно уменьшающих эту погрешность.

Для наших целей алгоритм сегментации должен быть управляемым, т. е. должна быть возможность строить сегментацию с заранее заданным (ориентировочным) количеством сегментов. Практика показала, что для кадров размера 720×480 для построения кадра-прогноза с той же погрешностью, какую дает стандарт $h264$ при обычной величине шума PSNR = 40 (Peak Signal-to-Noise Ratio) обычно требуется сегментация на 100–500 сегментов. Причем это количество слабо зависит от размера кадра. При переходе от размера 320×240 к 1920×1080 требуемое количество сегментов увеличивается лишь в 2–4 раза.

Оказывается алгоритм можно существенно улучшить, если сегментировать предыдущий кадр не сам по себе, а на основе целой цепочки из нескольких предыдущих кадров. В простейшем случае можно сегментировать пару предыдущих кадров. В этом случае многие визуально различные сегменты обладающие одинаковым (аффинным) движением будут объединены в один сегмент. В результате, для получения сегментации того же качества оказывается достаточным строить не несколько сотен сегментов, а не более одного десятка. Отметим, что *качество* сегментации в нашем случае — это не субъективная оценка, а вполне конкретное число (см. Определение 9).

Построению такого алгоритма сегментации и посвящена настоящая статья.

Основные определения

В настоящей работе мы будем рассматривать прямоугольные полноцветные (TrueColor) изображения.

Определение 1. *Изображением (кадром) размера $mx{\times}my$ будем называть набор из трех матриц (RGB) размера $mx{\times}my$. Обычно, но не обязательно, элементы этих матриц являются целыми числами из отрезка* [0..255]. *Эти три матрицы мы будем рассматривать как одно отображение $F : U \to \mathbb{R}^3$, где U — подмножество в $\mathbb{Z}^2$, состоящее из точек $(x, y) : 0 \leqslant x < mx,\ 0 \leqslant y < my$. Точки из U будем называть* пикселами *для отличия от остальных точек $\mathbb{R}^2$. Отображение F естественным образом продолжается до отображения $\mathbb{Z}^2 \to \mathbb{R}^3$. Кроме того, используя некоторую интерполяционную формулу (билинейную, бикубическую, сплайны и т. д.), функцию F можно продолжить до непрерывной кусочно-полиномиальной функции $\mathbb{R}^2 \to \mathbb{R}^3$, которую мы также будем обозначать буквой F.*

Определение 2. *Сегментацией $U = \cup U_i$ изображения будем называть разбиение области U на произвольные части $U_1, \ldots, U_N$, сегменты.*

Определение 3. *Размером $k(U_i)$ сегмента U_i будем называть количество пикселов в нем.*

Определение 4. *Пиксел $(x, y) \in U$ будем называть граничным для данной сегментации, если он принадлежит одному сегменту, а один из его четырех соседних — другому.*

Определение 5. *Два сегмента называются соседними, если в первом сегменте существует пиксел, один из соседних к которому принадлежит второму сегменту.*

Определение 6. *Аффинной сегментацией $S = \{U_i, A_i, i = 1, \ldots, N\}$ будем называть обычную сегментацию $U = \cup U_i$ снабженную набором аффинных преобразований $A_i : U_i \to \mathbb{R}^2$:*

$$A_i : \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} a_0 + a_1 x + a_2 y \\ a_3 + a_4 x + a_5 y \end{pmatrix}. \tag{1}$$

Для каждого пиксела (x, y) из области U обозначим через $S^*(x, y) \in \mathbb{R}^2$ точку $A_i(x, y)$, где i — номер сегмента, к которому принадлежит пиксел (x, y).

Определение 7. *а) Пусть f — функция $\mathbb{R}^2 \to \mathbb{R}^3$ и A — аффинное преобразование плоскости вида (1). Через $A^*(f)$ обозначим функцию $\mathbb{R}^2 \to \mathbb{R}^3$*

$$A^*(f)(x, y) = f(a_0 + a_1 x + a_2 y, a_3 + a_4 x + a_5 y)\,.$$

б) Пусть, дополнительно, (F_1, F_2) — два изображения (кадра) одинакового размера (то есть два отображения $U \to \mathbb{R}^3$). Для каждого пиксела (x, y) длину вектора $F_1(x, y) - A^*(F_2)(x, y)$ будем называть погрешностью преобразования A на паре кадров (F_1, F_2) в этом пикселе.

Определение 8. *Пусть S — аффинная сегментация области U. Тогда для произвольного изображения F через $S^*(F)$ обозначим новое изображение, строящееся по формулам $S^*(F)(x, y) = F(A_i^*(x, y))$, где i — номер сегмента, к которому принадлежит пиксел (x, y).*

Оценка качества сегментации в общем виде — сложная и, вообще говоря, очень субъективная задача. Если нас интересуют не отдельные изображения, а видеопоследовательности, цепочки изображений, мы можем предложить вполне естественное для данной ситуации понятие качества сегментации.

Определение 9. *Пусть (F_1, F_2) — два изображения одинакового размера и S — аффинная сегментация области U. Изображение $F_1 - S^*(F_2)$ будем называть разностным изображением. Среднеквадратичное значение длины вектора $|F_1 - S^*(F_2)|$ по всем пикселам области U будем называть погрешностью сегментации.*

Всюду в дальнейшем мы будем предполагать фиксированной пару кадров (F_1, F_2) одинакового размера $mx \times my$.

Алгоритм «А»

Обычно при обработке видеоданных [1, 2, 3, 9] используются те или иные методы нахождения движения. Все они являются различными вариантами одного и того же алгоритма Лукаса–Канады [10, 11] и находят движения лишь в виде сдвигов. Эти методы очень эффективны, но лишь для случая малых областей, сегментов. Во всех реализованных на сегодняшний день способах видеообработки это так и есть. Но в случае больших сегментов, занимающих значительную долю всего кадра, мы должны искать движения более общего вида, например, аффинных преобразований вида (1).

В работе [6] приведен алгоритм поиска таких преобразований, минимизирующих погрешность для данной пары кадров при фиксированной сегментации. Этот алгоритм заключается в следующем. Для каждого сегмента U_i рассмотрим аффинное преобразование A_i вида (1). Для него на сегменте U_i вычисляем величину

$$S = \sum_{(x,y)\in U_i} |F_1(x, y) - A_i^*(F_2)(x, y)|^2$$

и находим значения $(a_0, \ldots, a_5)$, при которых S достигает минимума. Описанный алгоритм является естественным обобщением стандартного алгоритма Лукаса-Канады на случай аффинных преобразований и практически так же эффективен и надежен. Разумеется, так как нам надо определить не две координаты вектора сдвига, а все шесть параметров

аффинного преобразования, он будет работать медленнее обычного алгоритма Лукаса–Канады. Однако это будет замедление в разы, а не на порядки. Будем называть этот процесс алгоритмом «А».

В задаче, исследуемой в настоящей работе, мы имеем возможность менять не только аффинные преобразования, но и саму сегментацию.

Клеточный автомат

Сегменты, получающиеся в процессе построения, иногда могут иметь очень хаотичную структуру. Для их сглаживания будем использовать клеточный автомат. При его описании «соседними» для данного пиксела считаются его восемь соседей. Таким образом, у углового пиксела области U — три соседних, у пиксела на границе кроме угловых — 5 соседних, у внутренних — по 8 соседей.

Один шаг работы клеточного автомата заключается в следующем. Перебираем все пикселы области U в псевдослучайном порядке. Если очередной пиксел принадлежит сегменту i, а более половины его соседей — к сегменту $j \neq i$, то относим этот пиксел к сегменту j, в противном случае ничего не делаем. Таким образом, за один шаг мы перебираем все пикселы кадра в псевдослучайном порядке.

Работа алгоритма зависит от выбранного способа псевдослучайного обхода всех $mx \times$ $\times\, my$ пикселов области U. Метод рандомизации, применяемый в нашей работе, основан на выборе простого числа p, несколько большего, чем $mx * my$ и первообразного корня a по модулю p. Как известно, в этом случае последовательность $1, a, a^2, \ldots, a^{p-1}$ является перестановкой чисел $1, \ldots, p-1$.

Применением клеточного автомата к данной сегментации будем называть результат последовательного выполнения описанных выше шагов клеточного автомата. Работа алгоритма прекращается, если на очередном шаге автомата сегментация не изменилась.

Хорошо известно, что описанный выше клеточный автомат закончит свою работу за конечное количество шагов.

Практика показывает, что описанный алгоритм дает хорошие результаты по сглаживанию сегментов.

Разделение сегмента

Пусть (F_1, F_2) — два изображения одинакового размера и U_1 — некоторый сегмент на U. Выделим из него подсегмент U_2 по следующему алгоритму. Вначале зафиксируем некоторую величину ожидаемого шума e, для типичных последовательностей кадров удобно будет взять $e = 2$.

1. Строим матрицу R, принимающую значения 1 в пикселах сегмента и 0 в остальных пикселах.
2. Сглаживаем матрицу R (см. ниже) с радиусом $(mx + my)/10$.
3. Находим пиксел (x_S, y_S), в котором эта сглаженная матрица принимает наибольшее значение. Если таких пикселов несколько, берем первый из них в лексикографическом порядке.
4. Обходим по спирали пикселы вокруг (x_S, y_S). Пикселы, принадлежащие сегменту U_1, добавляем в сегмент U_2 до тех пор, пока не получим 100 пикселов. Если во всем сегменте не более 100 пикселов, то такой сегмент не будем разбивать на части.
5. С помощью алгоритма «А» находим аффинное преобразование A_2, оптимальное для сегмента U_2. Величину погрешности преобразования A_2 на сегменте U_2 обозначим e_2. Если $e_2 < e$, положим $e_2 = e$.

6. Находим E — матрицу погрешностей преобразования A_2 на всей области U.
7. Сглаживаем матрицу E с радиусом 10.
8. В сегмент U_2 теперь отбираем те пикселы из исходного сегмента U_1, для которых величина сглаженной погрешности не превышает e_2.
9. Сглаживаем построенную область с помощью клеточного автомата.
10. Повторяем шаги $(5, 6, 7, 8, 9)$ до тех пор, пока сегмент U_2 не стабилизируется. Численные эксперименты показывают, что требуемое количество итераций от 3 до 6, причем более четырех повторений требуются крайне редко.

При разбиении сегмента используется некоторая версия алгоритма «сглаживания» матрицы. Опишем его более подробно.

Определение 10. *Сглаживанием матрицы R с радиусом k назовем ее свертку с ядром $Q(x, y) = q(|x|) \cdot q(|y|)$, где*

$$q(x) = \begin{cases} (k+1-x)/(k+1)^2 & x \leqslant k+1 \\ 0 & x > k+1 \end{cases},$$

т. е.

$$R'(x, y) = \sum_{dx,dy=-k}^{k} Q(dx, dy) \cdot R(x + dx, y + dy)$$

В частности, при сглаживании с радиусом 0 мы получаем исходную матрицу, при сглаживании с радиусом 1 — свертку с ядром:

$$\frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Можно использовать и более сложные методы сглаживания, например гауссово [7, 8], но это вряд ли существенно изменит результаты.

Улучшение сегментции

Пусть (F_1, F_2) — два изображения одинакового размера и $S = \{U_i, A_i, i = 1, \ldots, N\}$ — аффинная сегментация области U. Для ее улучшения применяем следующий алгоритм.

1. Для всех аффинных преобразований A_i строим E_i — матрицы погрешностей преобразований A_i на всей области U.
2. Сглаживаем матрицы E_i с радиусом 10.
3. В сегмент U_i отбираем те пикселы из U, для которых величина сглаженной погрешности наименьшая среди всех E_i.
4. Сглаживаем построенные сегменты с помощью клеточного автомата.
5. Удаляем пустые сегменты.
6. С помощью алгоритма «А» находим оптимальные аффинные преобразования для каждого вновь полученного сегмента U_i.
7. Повторяем шаги $(1 - -6)$ до тех пор, пока сегменты U_i почти не стабилизируются. Более точно — пока количество пикселов, перешедших в другой сегмент, в течение шага не станет меньше небольшой выбранной константы. В проведенных экспериментах константа выбиралась равной 10. Практика показала, что после четырех повторений практически всегда сегменты полностью перестают изменяться.

Динамическая сегментация

Пусть дана пара последовательных кадров (F_1, F_2) и пусть мы хотим разбить второй кадр F_2 на N сегментов.

Алгоритм динамической сегментации состоит из следующих шагов.

1. Начинаем с одного сегмента U_0, состоящего из всей области U и тождественного аффинному преобразованию A_0.
2. На каждом шаге выделяем из U_0 один сегмент с помощью алгоритма, описанного выше, до тех пор, пока не получим требуемое количество сегментов.
3. Улучшаем полученную сегментацию.

Результаты экспериментов

В данной задаче для оценки качества получающейся сегментации имеется уже готовый числовой параметр — погрешность сегментации (см. Определение 9).

Один сегмент, сдвиг

Для проверки работоспособности алгоритма берем следующую пару кадров: $f1$ — левые 500 столбцов стандартного изображения `lena.bmp` размера 512×512, $f2$ — правые 500 столбцов. Таким образом, $f2$ получается из $f1$ со сдвигом на 12 точек по горизонтали. В результате применения описанного алгоритма получаются два сегмента. Один содержит почти весь кадр, точнее говоря $99,66\%$ от площади кадра с аффинным преобразованием:

$$\begin{aligned} x' &= 11{,}998994 \quad +1{,}000002\,x + 0{,}000003\,y\,; \\ y' &= 0{,}000539 \quad\ -0{,}000001\,x + 0{,}999998\,y\,. \end{aligned}$$

Фактически оно совпадает со сдвигом на 12 точек влево. Второй сегмент появляется только из-за краевых эффектов. Общая погрешность преобразования оказывается равной $2,7$. На примере этой практически идеальной ситуации мы видим, каких показателей стоит ожидать от «хорошей» аффинной сегментации.

Несколько сегментов со спрайтами

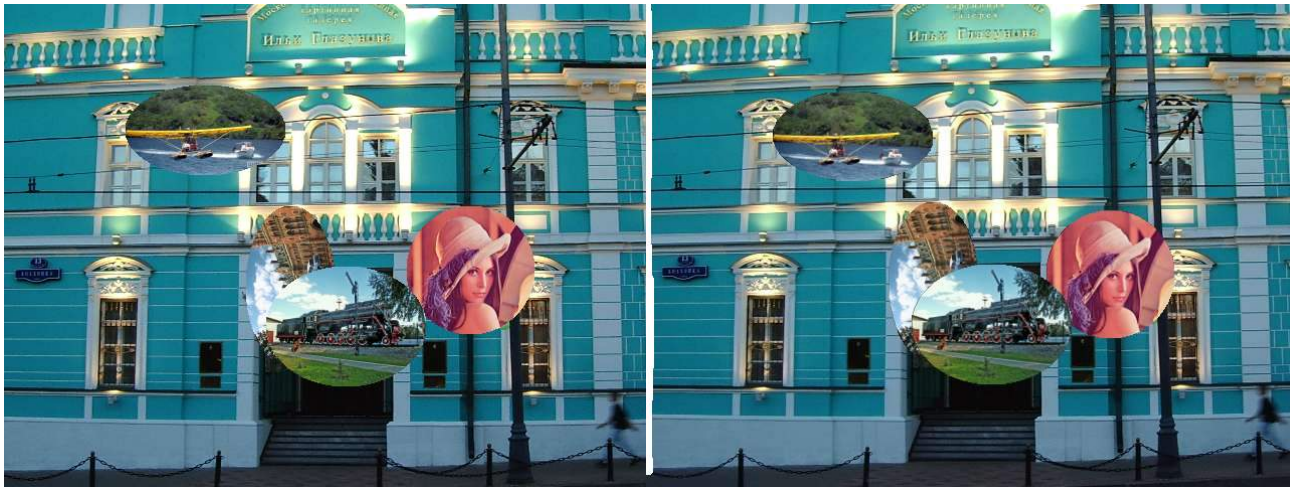
Здесь мы опять строим пару искусственных кадров. Каждый кадр размера 640×480 состоит из фона и четырех небольших спрайтов (эллипсы с полуосями 50..70). И для фона, и для каждого спрайта задано свое аффинное преобразование.

Алгоритм построил 8 сегментов, на рис. 2(а) разными цветами показаны разные сегменты, (б) — прорисованы их границы на исходном изображении. Погрешность сегментации оказывается равной $8,6$. Более подробное изучение дифференциального кадра показывает, что, как и следовало ожидать, наибольшая погрешность — в областях, которым нет соответствия на втором кадре, «uncovered background». Это случай тоже можно рассматривать как близкий к идеальному, все сегменты и аффинные преобразования оказались найдены верно.

Реальные видеопоследовательности

Описанные алгоритмы были применены к нескольким реальным видеопоследовательностям. Среди них стандартная видеопоследовательность `Foreman` размерности 352×288 и мультфильм `Factory` размерности 1920×1080.

В последовательности `Foreman` получается от 2 до 11 сегментов (на рис. 4, кадр № 173 — 10 сегментов. Слева разные сегменты показаны разным цветом, справа — на исходном изображении прорисованы границы сегментов), погрешность сегментации — от 5,6 до 9,8.



(а) (б)

Рис. 1: Два кадра со спрайтами

(а) (б)

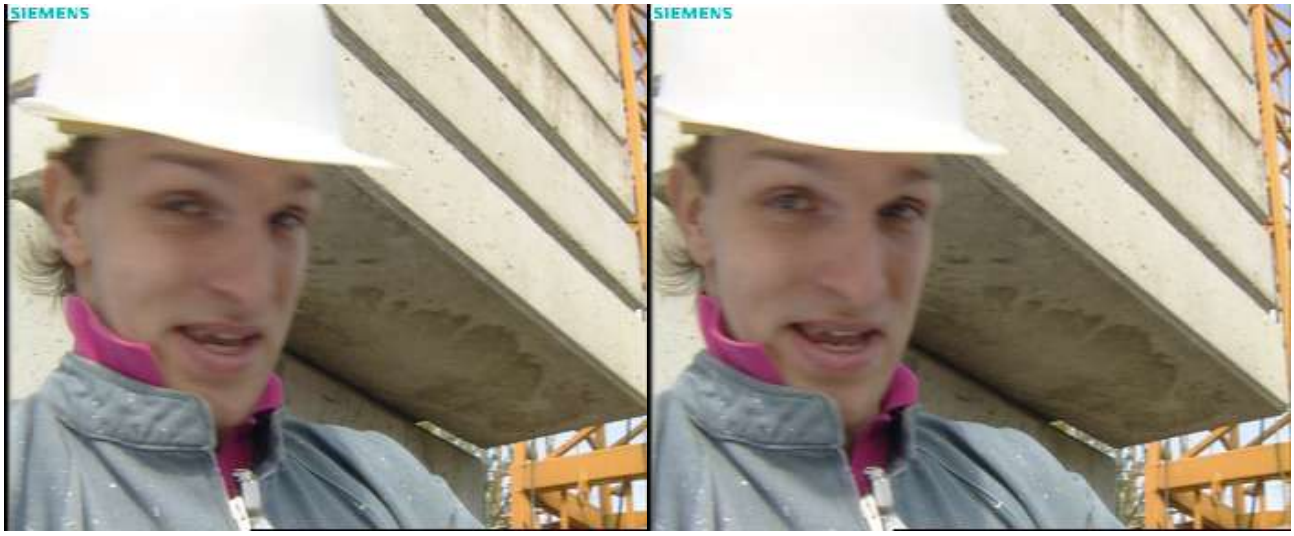
Рис. 2: Сегментация, 8 сегментов

Видно, что сегментация связана именно с различными движениями частей кадра. Например, если глаза двигаются синхронно со всем лицом, то они не выделяются в отдельные сегменты.

В последовательности `Factory` получается от 2 до 15 сегментов, погрешность сегментации — от 2,6 до 11,2. То есть при огромном увеличении площади кадра количество сегментов увеличивается незначительно.

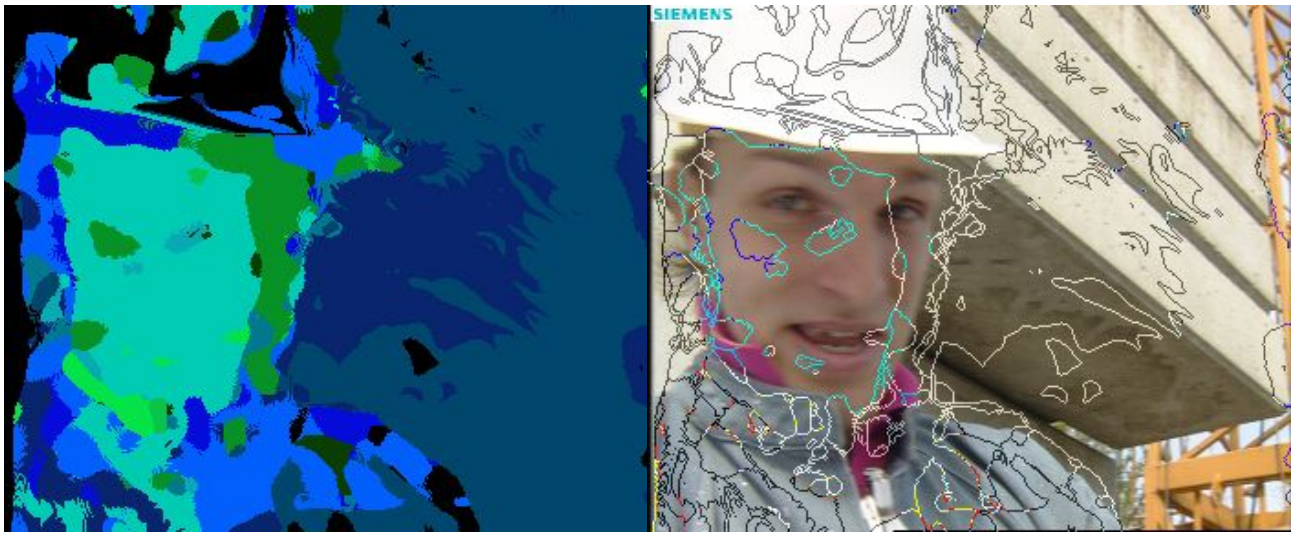
Заключение

В статье приводятся лишь первые экспериментальные результаты, полученные на сравнительно небольшом количестве примеров. Более полное экспериментальное исследование алгоритма, а также его оптимизация будут произведены позднее.



(а) (б)

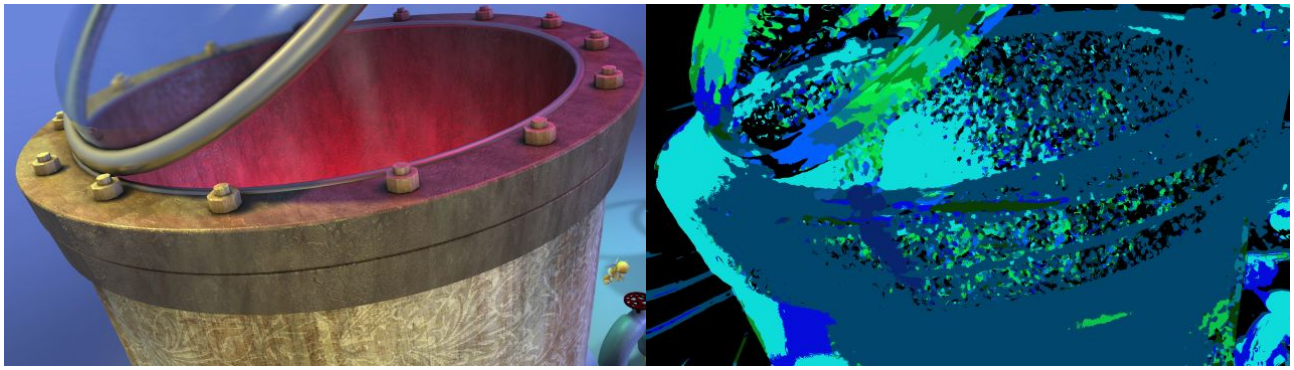
Рис. 3: Исходные кадры, номера $(172, 173)$



(а) (б)

Рис. 4: Сегментация на 10 сегментов

1. Предложенный метод дает весьма эффективный алгоритм сегментации: по сравнению со статической сегментацией при той же погрешности количество сегментов можно уменьшить в несколько десятков раз. Здесь требуется более детальное сравнение.
2. Алгоритм надежен: визуальное изучение получающихся сегментов показывает, что нет практически ни одной существенной ошибки.
3. Количество сегментов меняется в пределах от 2 до 20, причем оно мало меняется с ростом размера кадра.
4. Погрешность сегментации — от 1 до 14, причем она не изменяется с ростом размера кадра.
5. В реальных кадрах основной составляющей погрешности является не ошибки сегментации и нахождения движения, а «uncovered background» и цветовое различие между



(а) (б)

Рис. 5: Исходный кадр размера 1920×1080 и его сегментация на 15 сегментов

кадрами, т. е. причины, которые невозможно устранить выбором сегментации и аффинных преобразований.

Литература

[1] *ITU-T and ISO/IEC JTC 1* Generic coding of moving pictures and associated audio information. Part 2: Video // ITU-T Recommendation H.262 – ISO/IEC 13818-2 (MPEG-2), Nov. 1994.

[2] *ITU-T* Video coding for low bit rate communication // ITU-T Recommendation H.263; ver. 1, Nov. 1995; ver. 2, Jan. 1998; ver. 3, Nov. 2000.

[3] *ITU-T Rec. H.264 / ISO/IEC 11496-10.* Advanced video coding // Final Committee Draft, Document JVT-E022, Sept. 2002.

[4] *Хашин С. И.* Применение методов распознавания образов для сжатия видеоинформации // Докл. всеросс. конф. ММРО-13. — М.: МАКС Пресс, 2007. С. 420–424.

[5] *Хашин С. И.* Оценка качества сегментации изображения // Вестник ИвГУ. Вып. 2. — Иваново, 2010. "С. 112–118.

[6] *Хашин С. И.* Аффинная версия алгоритма Лукаса-Канады // Докл. всеросс. конф. ММРО-13. — М.: МАКС Пресс, 2011. "С. 459–462.

[7] *Гонсалес Р., Вудс Р.* Цифровая обработка изображений. — М.: Техносфера, 2006. — 1072 с.

[8] *Яне Б.* Цифровая обработка изображений. — М.: Техносфера, 2007. 583 с.

[9] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification // (ITU-TRec.H.264.ISO/IEC14496-10AVC) Joint Video Team (JVT), Mar. 2003. Doc. JVT-G050.

[10] *Lucas B. D., Kanade T.* An iterative image registration technique with an application to stereo vision // Imaging Understanding Workshop Proceedings. 1981. Pp. 121–130

[11] *Baker S., Gross R., Matthews I.* Lucas-Kanade 20 years on: A unifying framework // Int. J. Computer Vision, 2002. Vol. 56. Pp. 111–122.