Применение обучения с подкреплением для одновременного выбора модели алгоритма классификации и ее структурных параметров*

В. А. Ефимова, А. А. Фильченков, А. А. Шалыто
efimova@rain.ifmo.ru; afilchenkov@corp.ifmo.ru; shalyto@mail.ifmo.ru
Университет ИТМО, Россия, г. Санкт-Петербург, Кронверкский проспект, 49

Существует множество алгоритмов машинного обучения, однако для эффективного решения задачи интеллектуального анализа данных необходимо не только выбрать один из них, но и настроить его структурные параметры. В настоящей работе ставится задача одновременного автоматического выбора алгоритма классификации и настройки его структурных параметров и предлагается ее решение на основе решения задачи о многоруком бандите. Описываются эксперименты, проведенные на множестве реальных наборов данных. Продемонстрировано, что предложенный подход обеспечивает более высокую точность классификации по сравнению с существующими методами.

Ключевые слова: *выбор алгоритма; настройка структурных параметров; настройка гиперпараметров; оптимизация; многорукий бандит; обучение с подкреплением*

DOI: 10.21469/22233792.2.2.09

1 Введение

Совместный выбор модели алгоритма и ее структурных параметров (гиперпараметров) для обработки набора данных считается трудной задачей, на текущий момент не получившей решения. Фактически на настоящий момент задача разбивается на две подзадачи, решаемые независимо: выбор алгоритма с фиксированными структурными параметрами из конечного множества (портфолио) алгоритмов и оптимизацию структурных параметров конкретной модели алгоритма. Для того чтобы снять терминологическую неопределенность, далее в этой работе *алгоритмами* будем называть алгоритмы машинного обучения, для которых заданы структурные параметры, если же структурные параметры не заданы, такой алгоритм будем называть *моделью алгоритма*.

Первая подзадача в подавляющем большинстве случаев решается с помощью перебора. В одних из первых работ [1, 2], посвященных проблематике выбора алгоритма из портфолио, исследовалось влияние задачи классификации на выбор алгоритма и использовались решающие правила. Другими решениями первой подзадачи выступают следующие подходы к выбору алгоритма: случайным образом; на основе каких-либо эвристических правил, выработанных исследователем; помощью *k-стратового скользящего контроля* (*k-fold cross-validation*) [3]. Последний из указанных подходов предполагает запуск всех алгоритмов и последующее сравнение, что требует значительного времени, а остальные подходы не универсальны, т. е. не могут быть применены во всех случаях. Существуют более эффективные подходы, такие как, например, мета-обучение [4]. Его целью является решение задачи выбора алгоритма из портфолио алгоритмов для решения поставленной задачи без непосредственного применения каждого из них [4]. Решение этой задачи в рамках

*Работа выполнена при финансовой поддержке Правительства Российской Федерации, грант 074-U01, и РФФИ, проект № 16-37-60115.

мета-обучения сводится к задаче обучения с учителем. Для этого используется заранее отобранное множество наборов данных D. Для каждого набора данных $d \in D$ вычисляется вектор мета-признаков, которые описывают свойства этого набора данных. Ими могут быть: число категориальных или численных признаков объеков в d, число возможных меток, размер d и многие другие [5]. Каждый алгоритм запускается на всех наборах данных из D. После этого вычисляется эмпирический риск, на основе которого формируются метки классов. Затем мета-классификатор обучается на полученных результатах. В качестве описания набора данных выступает вектор мета-признаков, а в качестве метки — алгоритм, оказавшийся самым эффективным с точки зрения заранее выбранной меры качества. Данная подзадача исследовалась во многих работах, например в [6, 7].

Вторая подзадача — оптимизация структурных параметров — состоит в поиске параметров, характеризующих модель алгоритма, при которых алгоритмы этой модели достигают наилучшего результата с точки зрения заранее выбранной меры качества. Например, для *метода опорных векторов* (SVM — support vector machine) структурным параметром выступает функция ядра, а при использовании *нейронной сети* — число скрытых слоев и число нейронов в них. Для ряда моделей (преимущественно статистических и регрессионных) структурные параметры подбирают аналитически или сводят задачу к более простой задаче оптимизации, как, например, это сделано в [8]. Однако в общем случае этот подход неприменим. В настоящее время разработаны алгоритмы, автоматически решающие данную задачу: *поиск по решетке* (*Grid Search*) [9], *случайный поиск* (*Random Search*) [10], *стохастический градиентный спуск* [11], *древовидный парзеновский оценщик* (*Tree-structured Parzen estimator*) [12] и *байесовская оптимизация*, к которой относится алгоритм *последовательной оптимизации по модели* (*Sequental Model-Based Optimization*, далее SMBO) [13]. В работе [14] был предложен алгоритм *последовательной конфигурации алгоритма по модели* (*Sequential model-based algorithm configuration*, далее SMBA), работающий на основе SMBO. Рассмотрим его работу. В некоторый момент времени для каждой модели алгоритма уже известно множество структурных параметров, с которыми на текущий момент она работает оптимальным образом. С помощью локального поиска в это множество добавляются наборы структурных параметров, отличающиеся от оптимальных в одной позиции и улучшающие эффективность алгоритма. Кроме того, в это множество добавляется некоторое число случайных наборов структурных параметров. Выбранные конфигурации (модели алгоритмов с заданными структурными параметрами) сортируются по *ожидаемому улучшению* (*expected improvement*), а после этого запускаются несколько лучших.

Единственным совмещающим решение двух подзадач и реализованным на данный момент подходом является полный перебор. Существует открытая библиотека для настройки алгоритмов машинного обучения Auto-WEKA [15]. Она позволяет автоматически выбирать из 27 базовых алгоритмов, 10 мета-алгоритмов и 2 ансамблевых алгоритмов лучший, одновременно настраивая его структурные параметры. Решение достигается полным перебором: оптимизация структурных параметров запускается на всех алгоритмах по очереди. Подробно работа библиотеки Auto-WEKA описана в [15]. Недостатком такого подхода является слишком большое время работы.

Решение проблемы одновременного выбора модели алгоритма и структурных параметров может также осуществляться на основе мета-обучения. Например, в [16] предложено 292 комбинации моделей алгоритмов и их структурных параметров для 6 изученных моделей алгоритмов. Однако выделение алгоритмов осуществляется эмпирическим путем, что обусловливает методологическую несостоятельность подхода для решения рассматри-

ваемой задачи, поскольку подход не гарантирует, что среди перебираемых решений будет оптимальное.

Цель данной работы — предложить метод одновременного выбора модели алгоритма и ее структурных параметров на основе конкурентного распределения времени между настройкой структурных параметров для различных моделей, который окажется эффективнее, чем существующие.

2 Постановка задачи

Для того чтобы поставить задачу одновременного выбора модели алгоритма и ее структурных параметров, сначала формально поставим две подзачади, перечисленные во Введении. Будем следовать обозначениям из [17].

Пусть A — модель алгоритма, характеризующаяся структурными параметрами $\lambda = \{\lambda_1, \ldots, \lambda_m\}, \lambda_1 \in \Lambda_1, \ldots, \lambda_m \in \Lambda_m$. Тогда с ней связано пространство структурных параметров $\Lambda = \Lambda_1 \times \cdots \times \Lambda_m$. За A_λ обозначим алгоритм, т. е. модель алгоритма, для которой задан вектор структурных параметров $\lambda \in \Lambda$.

Для обеих подзадач необходимо зафиксировать меру качества работы алгоритма. За меру качества в данной работе примем *отложенный эмпирический риск* (*hold-out empirical risk*). Эффективность алгоритма A_λ оценивается с помощью разделения набора данных на обучающую и тестовую выборку с последующим подсчетом эмпирического риска, достигаемого на тестовой выборке:

$$Q(A_\lambda, D) = \frac{1}{|D|} \sum_{x \in D} [A_\lambda(x) \neq y(x)],$$

где $[A_\lambda(x) \neq y(x)]$ — функция потерь для задачи классификации, определяющая размер ошибки при запуске алгоритма A_λ на объекте x набора данных D.

Подзадача выбора лучшего алгоритма из портфолио формулируется следующим образом. Дано некоторое множество алгоритмов с фиксированными структурными параметрами $\mathcal{A} = \{A^1_{\lambda_1}, \ldots, A^m_{\lambda_m}\}$ и обучающая выборка $D = \{d_1, \ldots, d_n\}$. Здесь $d_i = (x_i, y_i) \in X \times Y$, где X — множество признаков, описывающих объекты, а Y — конечное множество меток. Требуется выбрать алгоритм $A^*_{\lambda^*}$, который окажется наиболее эффективным с точки зрения меры качества Q. В рассматриваемом случае это сводится к задаче минимизации эмпирического риска:

$$A^*_{\lambda_*} \in \arg\min_{A^j_{\lambda_j} \in \mathcal{A}} Q(A^j_{\lambda_j}, D).$$

Подзадача оптимизации структурных параметров заключается в подборе таких $\lambda^* \in \Lambda$, при которых заданная модель алгоритма A будет наиболее эффективна. Запишем в виде формулы:

$$\lambda^* \in \arg\min_{\lambda \in \Lambda} Q(A_\lambda, D).$$

Структурные параметры для модели алгоритма подбирается в процессе их настройки. Формализуем это понятие. *Процесс настройки структурных параметров* для алгоритма A^i:

$$\pi_i(t, A^i, \{\lambda_j\}_{j=0}^k) \to \lambda^i_{k+1} \in \Lambda^i.$$

Его можно описать как работу алгоритма настройки структурных параметров, запущенного на некоторой модели алгоритма A^i с ограничением по времени t и хранящего историю

изменения вектора лучших найденных на данный момент (за k итераций) структурных параметров $\{\lambda_j\}_{j=0}^{k}$.

В данной работе предлагается одновременно решать задачи выбора модели алгоритма и оптимизации структурных параметров: дан набор моделей алгоритмов $\mathcal{A} = \{A^1, \ldots, A^k\}$, с каждым из которых связано пространство структурных параметров $\Lambda^1, \ldots, \Lambda^k$ соответственно. Необходимо найти алгоритм $A^*_{\lambda^*}$, минимизирующий эмпирический риск:

$$A^*_{\lambda^*} \in \underset{A^j \in \mathcal{A}, \lambda \in \Lambda^j}{\arg\min} Q(A^j_\lambda, D).$$

Предположим, что дан один или некоторое множество алгоритмов настройки структурных параметров для моделей алгоритмов из $\mathcal{A}$, задающих соответствующие процессы π_i. Тогда предыдущая задача сводится к запуску этих процессов. Но возникает новая задача — задача минимизации времени, потраченного на поиск лушчего алгортима. С практической точки зрения больший интерес представляет похожая задача, а именно: задача поиска лучшего алгоритма за фиксированное время. Эта задача и решается в данной работе. Формализуем ее.

Пусть задан некоторый временной бюджет T на поиск лучшего алгоритма $A^*_{\lambda^*}$. Требуется разбить его на интервалы $T = t_1 + \cdots + t_m$ таким образом, что при запуске процессов π_i с ограничением по времени t_i соответственно получим минимальный эмпирический риск:

$$\min_j Q(A^j_{\lambda_j}, D) \xrightarrow[(t_1,\ldots,t_m)]{} \min,$$

где $A^j \in \mathcal{A}, \lambda_j = \pi_j(t_j, A^j, \varnothing)$ и $t_1 + \cdots + t_m = T$; $t_i \geqslant 0 \, \forall i$.

3 Предлагаемый подход

В рассматриваемой задаче ключевым ресурсом является время T, отведенное на оптимизацию структурных параметров. Для удобства разобьем его на q небольших равных интервалов t, которые будем называть *временными бюджетами*. Теперь перейдем к задаче распределения временных бюджетов. Рассматриваемую задачу можно представить в следующем виде: через равные временны́е интервалы необходимо решать, какой процесс настройки структурных параметров будет выполняться в течение следующего интервала.

Заведомо неизвестно, какое качество будут показывать алгоритмы той или иной модели при решении конкретной задачи. Выделяя время только на модель, алгоритмы которой показывают лучшие результаты («эксплуатируя» соответствующую модель), можно упустить хорошие алгоритмы других моделей. Наоборот: при равномерном распределении времени между моделями настройка структурных параметров для моделей алгоритмов, неэффективных при решении данной задачи, занимает слишком много времени. Поэтому задача представляет собой поиск компромисса между исследованием (поочередным запуском настройки структурных параметров для всех моделей алгоритмов) и эксплуатацией (запуском настройки только для одной модели алгоритма). Его поиск производится с помощью обучения с подкреплением, частным случаем которого является задача о многоруком бандите [18]. Остальные подвиды обучения с подкреплением решают задачу в начальных предположениях, несколько отличающихся от условий задачи одновременного выбора модели алгоритма и ее структурных параметров, например они предполагают, что среда изменяется в зависимости от выбранного действия. Поскольку в данной задаче это не так, остальные виды обучения с подкреплением рассматриваться не будут.

В задаче о многоруком бандите рассматривается бандит с N ручками, дернув за любую из которых можно получить выигрыш некоторого размера, определяемый случайным

распределением, ассоциированным с данной ручкой. В каждый момент времени k агент выбирает ручку a_i и получает выигрыш $r(i,k)$. Цель агента — минимизировать суммарные потери (по сравнению с лучшей стратегией) за конечное время T. В данной работе использовались следующие алгоритмы решения задачи о многоруком бандите, описанные в [18]:

1) ε-жадный — на каждой итерации находит для каждой ручки a средний выигрыш $\bar{r}_{a,t}$, затем с вероятностью $1-\varepsilon$ выбирает ручку с максимальным средним выигрышем, а с вероятностью ε выбирает случайную ручку. Если перейти к пределу, то каждая ручка будет выбрана бесконечное число раз, так что средние выигрыши с вероятностью 1 сойдутся к настоящему выигрыш;
2) UCB1 — на стадии инициализации агент выбирает все ручки по очереди. Далее на итерации t выбирает ручку a_t для которой выполняется:

$$a_t = \underset{i=1,\ldots,N}{\arg\max}\, \bar{r}_i + \sqrt{\frac{2\ln t}{n_i}},$$

где $\bar{r}_i$ — средний выигрыш при выборе ручки i; n_i — число раз, которое выбиралась ручка i. Стоит отметить, что данный алгоритм прост в реализации;
3) Softmax — на стадии инициализации агент выбирает все ручки по очереди. Далее на итерации t выбирает ручку a_i с вероятностью:

$$p_{a_i} = \frac{e^{\bar{r}_i/\tau}}{\sum_{j=1}^{N} e^{r_j/\tau}},$$

где τ — положительный параметр, называемый температурой. При $\tau \to 0$ Softmax подобен жадному алгоритму.

В качестве ручек будем рассматривать процессы настройки структурных параметров $\{\pi_i(t, A^i, \{\lambda_k\}_{k=0}^q) \to \lambda_{q+1}^i \in \Lambda^i\}_{i=0}^m$ для множества алгоритмов $\mathcal{A} = \{A_{\lambda_1}^1, \ldots, A_{\lambda_m}^m\}$. После выбора ручки $i = a_k$ на итерации k будем выдавать процессу π_{a_k} некоторый промежуток времени t на настройку структурных параметров, в конце которого получим вектор структурных параметров λ_k^i. После завершения работы выбранного процесса оценку результата произведем, вычисляя эмпирический риск для процесса π_i на итерации k как $Q(A_{\lambda_k^i}^i, D)$.

Функцию выигрыша $r(i,k)$ можно определить двумя способами. В первом (простейшем) случае наградой будет выступать разница между оптимальным эмпирическим риском, найденным в ходе предыдущих итераций, и текущим эмпирическим риском.

Для второго способа воспользуемся особенностями работы алгоритма SMAC, описанного во Введении. Для нашей цели воспользуемся математическим ожиданием эмпирического риска на шаге k, с помощью которого вычисляется ожидаемое улучшение: $E_t(Q(A_{\lambda_k^i}^i, D))$, где $Q(A_{\lambda_k^i}^i, D)$ — эмпирический риск, достигаемый процессом p_i на наборе данных D в момент времени k.

Заметим, что процесс π_i решает задачу минимизации эмпирического риска, тогда как задача о многоруком бандите — задачи о максимизации выигрыша. Поэтому функцию среднего выигрыша определим следующим образом:

$$\bar{r}_{i,(k)} = \frac{Q_{\max} - E_{(k)}(Q(A_{\lambda_k^i}^i, D))}{Q_{\max}},$$

где $Q_{\max}$ — максимальный эмпирический риск, достижимый при решении данной задачи.

4 Экспериментальное исследование

Выше было рассмотрены существующие подходы к решению задачи. Поскольку на данный момент наиболее полно задачу можно решить с помощью библиотеки Auto-WEKA, для сравнения выбрана именно она.

Исследования проводились на десяти наборах реальных данных, находящихся в репозитории UCI и доступных по ссылке `http://www.cs.ubc.ca/labs/beta/Projects/autoweka/datasets/`, описание которых представлено в табл. 1. Предложенный подход позволяет использовать любой метод настройки структурных параметров, но для более корректного сравнения с библиотекой Auto-WEKA был выбран метод, реализованный в ней, а именно: SMBO. Рассматриваются 6 известных моделей алгоритмов классификации: *метод ближайших соседей (kNN)*, *метод опорных векторов*, *логистическая регрессия* (*Logistic Regression*), *случайный лес* (*Random Forest*), *персептрон* (*Perceptron*), *дерево принятия решений* (*C4.5 Decision Tree*). В табл. 2 приведено число категориальных и численных структурных параметров для каждой из рассмотренных моделей алгоритмов классификации.

В поставленной задаче временной бюджет выделяется на ее полное решение, тогда как в предложенном алгоритме производится разбиение общего временно́го отрезка на равные более мелкие отрезки, которые по одному выдаются процессам на каждой итерации.

Таблица 1 Описание использованных наборов данных

Название набора данных	Число категориальных признаков	Число численных признаков	Число классов	Число объектов в обучающей выборке	Число объектов в тестовой выборке
Dexter	0	20000	2	420	180
German Credit	13	7	2	700	300
Dorothea	0	100000	2	805	345
Yeast	0	8	10	1039	445
Secom	0	590	2	1097	470
Semeion	0	256	10	1116	477
Car	6	0	4	1210	518
KR-vs-KP	36	0	2	2238	958
Waveform	0	40	3	3500	1500
Shuttle	38	192	2	35000	15000

Таблица 2 Число категориальных и численных структурных параметров, настраиваемых у моделей алгоритмов, между которыми производился выбор

Модель алгоритма	Категориальные	Численные
Метод ближайших соседей	4	1
Метод опорных векторов	4	6
Логистическая регрессия	0	1
Случайный лес	2	3
Перцептрон	5	2
Дерево принятия решений C4.5	6	2

Таблица 3 Сравнение предложенной активной стратегии с библиотекой Auto-WEKA по наименьшему достигнутому эмпирическому риску Q

Набор данных	AutoWEKA	UCB1	0,4-жадный	0,6-жадный	Softmax	$\text{UCB1}_{E(Q)}$	$\text{Softmax}_{E(Q)}$
Car	0,3305	**0,1836**	**0,1836**	**0,1836**	**0,1836**	**0,1836**	**0,1836**
Yeast	34,13	**29,81**	**29,81**	**33,65**	**29,81**	**29,81**	**29,81**
KR-vs-KP	0,2976	**0,1488**	**0,1488**	**0,1488**	**0,1488**	**0,1488**	**0,1488**
Semeion	4,646	**1,786**	**1,786**	**1,786**	**1,786**	**1,786**	**1,786**
Shuttle	**0,00766**	0,0115	0,0115	**0,00766**	0,0115	**0,0076**	**0,0076**
Dexter	7,143	2,38	2,381	2,381	2,381	2,381	**0,16**
Waveform	11,28	**8,286**	**8,286**	**8,286**	**8,286**	**8,286**	**8,286**
Secom	4,545	**3,636**	4,545	4,545	**3,636**	**3,636**	**3,636**
Dorothea	6,676	4,938	4,958	4,938	4,938	4,32	**2,469**
German Credits	19,29	**14,29**	**14,29**	15,71	**14,29**	**14,29**	**14,29**

Авторами было проведено исследование поведения предложенного алгоритма для различных значений временно́го бюджета, выделяемого на одну итерацию, чтобы найти значение, при котором результат оптимален. Были рассмотрены временны́е бюджеты от 10 до 60 с, с шагом в 3 с. Предложенный алгоритм был запущен на трех наборах данных из описанных выше: Car, German Credits и KRvsKP. Для решения задачи о многоруком бандите использовались алгоритмы UCB1, 0,4-жадный, 0,6-жадный и Softmax. Запуск каждой конфигурации производился трижды. В результате был выбран временной бюджет в 30 с на итерацию.

Рассматривалась работа предложенного алгоритма с алгоритмами решения задачи о многоруком бандите UCB1, 0,4-жадный, 0,6-жадный, Softmax с наивной функцией выигрыша, а также алгоритмами $\text{UCB1}_{E(Q)}$, $\text{Softmax}_{E(Q)}$ с описанной выше функцией выигрыша. На итерацию выделялось по 30 с, а общий временной бюджет составил 3 ч (10 800 с). Каждый из алгоритмов запускался 12 раз со случайными начальными значениями генератора псевдослучайных чисел, которые требует библиотека. Время работы библиотеки Auto-WEKA тоже ограничивалось 3 ч, а выбор производился из тех же алгоритмов классификации. Результаты приведены в табл. 3. Жирным выделены наименьшие значения в строке как оптимальные для задачи минимизации эмпирического риска.

Как видно из таблицы, предложенный метод оказался не хуже, а во многих случаях и существенно лучше библиотеки Auto-WEKA на всех десяти наборах данных, так как позволил достичь меньшей ошибки. Результаты для различных алгоритмов решения задачи о многоруком бандите не сильно отличаются, однако наименьшей ошибки достигли алгоритмы $\text{UCB1}_{E(Q)}$ и $\text{Softmax}_{E(Q)}$, использующие предложенную функцию выигрыша. Как показали эксперименты, предложенный метод позволяет улучшить существующее решение задачи совместного выбора алгоритма классификации и его структурных параметров, так как поиск производится на всем пространстве структурных параметров каждой модели алгоритма. Существенно, что за фиксированное время предложенный метод позволяет найти решения, по мере качества не уступающие конфигурациям, найденным с помощью Auto-WEKA.

Для того чтобы показать статистическую значимость того, что ариации предложенного метода достигли меньшей ошибки, был проведен тест Уилкоксона. Он применим для оценки полученных результатов, так как имеется 10 наборов данных, на каждом из ко-

торых запускались библиотека Auto-WEKA и описанные выше вариации предложенного алгоритма. Получили 6 проверок критерия: сравнение библиотеки Auto-WEKA с каждым вариантом. При $n = 10$ выборках значимые результаты получаются при сумме нетипичных рангов $T < T_{0,01} = 5$. Так как в данном случае решается задача минимизации, проверили критерий для лучшего из 12 запусков на каждом наборе данных. Для ε-жадных алгоритмов получили $T = 3$, для остальных — $T = 1$, что показывает статистическую значимость полученных результатов.

5 Заключение

В данной работе был предложен и исследован метод решения актуальной задачи совместного выбора алгоритма классификации и его структурных параметров, основанный на сведении задачи к задаче о многоруком бандите. Кроме того, предложена функция выигрыша, позволяющая лучше адаптировать решение задачи о многоруком бандите для применения к решению данной задачи. Как показали эксперименты, предложенная стратегия позволяет улучшить существующее решение задачи совместного выбора алгоритма классификации и его структурных параметров, так как поиск производится на всем пространстве структурных параметров каждой модели алгоритма.

Метод можно улучшить, изначально отсортировав процессы настройки структурных параметров с помощью мета-обучения. Его также можно улучшить, если с помощью мета-обучения подбирать в пару алгоритму классификации алгоритм настройки структурных параметров. Кроме того, можно ввести контекст процесса настройки структурных параметров и воспользоваться алгоритмом для решения задачи о многоруком контекстном бандите. В качестве контекста можно использовать значение эмпирического риска, полученного при запуске алгоритма классификации на наборах данных, отобранных с помощью мета-обучения.

Литература

[1] *Rendell L., Cho H.* Empirical learning as a function of concept character // Mach. Learn., 1990. Vol. 5. P. 267–298.

[2] *Aha D. W.* Generalizing from case studies: A case study // 9th Conference (International) on Machine Learning Proceedings, 1992. P. 1–10. doi: 10.1016/B978-1-55860-247-2.50006-1.

[3] *Rodrigues J. D., Perez A., Lozano J. A.* Sensitivity analysis of k-fold cross validation in prediction error estimate // IEEE Trans. Pattern Anal., 2010. Vol. 32. No. 3. P. 569–575.

[4] *Giraud-Carrier C., Vilalta R., Brazdil P.* Introduction to the special issue on meta-learning // Mach. Learn., 2004. Vol. 54. No. 3. P. 187–193. doi: 10.1023/B:MACH.0000015878.60765.42.

[5] *Filchenkov A., Pendryak A.* Datasets meta-feature description for recommending feature selection algorithm // Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference Proceedings. — IEEE, 2015. P. 11–18. doi: 10.1109/AINL-ISMW-FRUCT.2015.7382962.

[6] *Lim T.-S., Loh W.-Y., Shih Y.-S.* A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms // Mach. Learn., 2000. Vol. 40. No. 3. P. 203–228. doi: 10.1023/A:1007608224229.

[7] *Ali S., Smith K. A.* On learning algorithm selection for classification // Appl. Soft Comput., 2006. Vol. 6. No. 2. P. 119–138. doi: 10.1016/j.asoc.2004.12.002.

[8] *Зайцев А. А., Стрижов В. В., Токмакова А. А.* Оценка гиперпараметров линейных регрессионных моделей методом максимального правдоподобия // Информационные технологии, 2013. Вып. 2. С. 11–15.

[9] *Bergstra J., Bengio Y.* Random search for hyper-parameter optimization // J. Mach. Learn. Res., 2012. Vol. 13. No. 1. P. 281–305.

[10] *Hastie T., Tibshirani R., Friedman J.* The elements of statistical learning: Data mining, inference and prediction // Math. Intell., 2005. Vol. 27. No. 2. P. 83–85. doi: 10.1007/978-0-387-84858-7.

[11] *Bottou L.* Online learning and stochastic approximations // On-Line Learning Neural Networks, 1998. Vol. 17. No. 9. P. 142–177.

[12] *Bergstra J., Bardenet R., Bengio Y.* Algorithms for hyper-parameter optimization // Advances in Neural Information Processing Systems Proceedings, 2011. P. 2546–2554.

[13] *Snoek J., Larochelle H., Adams R. P.* Practical Bayesian optimization of machine learning algorithms // Advances in Neural Information Processing Systems Proceedings, 2012. P. 2951–2959.

[14] *Hutter F., Hoos H. H., Leyton-Brown K.* Sequential model-based optimization for general algorithm configuration. — University of British Columbia, Computer Science, 2010. TR-2010-10.

[15] *Thornton C., Hutter F., Hoos H.* Auto-WEKA: Automated selection and hyper-parameter optimization of classification algorithms // 19th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining Proceedings, 2013. doi: 10.1145/2487575.2487629.

[16] *Leite R., Brazdil P., Vanschoren J.* Selecting classification algorithms with active testing // Machine learning and data mining in pattern recognition. — Springer, 2012. P. 117–131.

[17] *Воронцов К. В.* Математические методы обучения по прецедентам (теория обучения машин). `http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf`.

[18] *Sutton R. S., Barto A. G.* Reinforcement learning: An introduction. — Cambridge, MA, USA: MIT Press, 1998. 342 p.

Поступила в редакцию 18.08.2016

Reinforcement-based simultaneous classification model and its hyperparameters selection*

V. A. Efimova, A. A. Filchenkov, and A. A. Shalyto

`efimova@rain.ifmo.ru; afilchenkov@corp.ifmo.ru; shalyto@mail.ifmo.ru`

ITMO University, 49 Kronverksky pr., St. Petersburg, Russia

Many algorithms for data analysis exist, especially for the classification problem. To solve a data analysis problem, a proper algorithm should be chosen, and also, its hyperparameters should be selected. These two problems, algorithm selection and hyperparameter optimization, are commonly solved independently. The full-model selection process requires unacceptable time budgets. Thus, this is one of the factors preventing the spread of automated model selection methods. The goal of this work is to suggest a method for simultaneous algorithm and its parameters selection to reduce full-model election time. In order to do so, this problem was reduced to a multiarmed bandit problem. An algorithm is presented as an arm and algorithm of hyperparameters search during a fixed time is presented as the corresponding arm play. Also, several reward functions are described. The experiments have been held on 10 popular labeled datasets from the UCI repository. To compare the proposed method, 10 several well-known classification algorithms from WEKA library and algorithm for hyperparameter optimization

*The research was supported by the Russian Government (grant 074-U01) and the Russian Foundation for Basic Research (project No. 16-37-60115).

from Auto-WEKA library have been used. The proposed method has been compared with the brute force search implemented in WEKA library and a random time budget assignment policy. The results show significant time reduction of selecting proper algorithm and its hyperparameters for processing given dataset. The proposed method often produces classification results much better than Auto-WEKA state-of-the-art automatic algorithm selection and hyperparameter optimization tool.

Keywords: *algorithm selection; hyperparameter optimization; multiarmed bandit; reinforcement learning*

DOI: 10.21469/22233792.2.2.09

References

[1] Rendell, L., and H. Cho. 1990. Empirical learning as a function of concept character. *Mach. Learn.* 5:267–298.

[2] Aha, D. W. 1992. Generalizing from case studies: A case study. *9th Conference (International) on Machine Learning Proceedings*. 1–10. doi: 10.1016/B978-1-55860-247-2.50006-1.

[3] Rodrigues, J. D., A. Perez, and J. A. Lozano. 2010. Sensitivity analysis of k-fold cross validation in prediction error estimate. *IEEE Trans. Pattern Anal.* 32(3):569–575.

[4] Giraud-Carrier, C., R. Vilalta, and P. Brazdil. 2004. Introduction to the special issue on meta-learning. *Mach. Learn.* 54(3):187–193. doi: 10.1023/B:MACH.0000015878.60765.42.

[5] Filchenkov, A., and A. Pendryak. 2015. Datasets meta-feature description for recommending feature selection algorithm. *Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference Proceedings*. IEEE. 11–18. doi: 10.1109/AINL-ISMW-FRUCT.2015.7382962.

[6] Lim, T.-S., W.-Y. Loh, and Y.-S. Shih. 2000. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach. Learn.* 40(3):203–228. doi: 10.1023/A:1007608224229.

[7] Ali, S., and K. A. Smith. 2006. On learning algorithm selection for classification. *Appl. Soft Comput.* 6(2):119–138. doi: 10.1016/j.asoc.2004.12.002.

[8] Zaytsev, A. A., V. V. Strijov, and A. A. Tokmakova. 2013. Estimation regression model hydroparameters using maximum likelihood. *Informatsionnye Tekhnologii* [Information Technologies] 2:11–15.

[9] Bergstra, J., and Y. Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* 13(1):281–305.

[10] Hastie, T., R. Tibshirani, and J. Friedman. 2005. The elements of statistical learning: data mining, inference and prediction. *Math. Intell.* 27(2):83–85. doi: 10.1007/978-0-387-84858-7.

[11] Bottou, L. 1998. Online learning and stochastic approximations. *On-Line Learning Neural Networks* 17(9):142–177.

[12] Bergstra, J., R. Bardenet, and Y. Bengio. 2011. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems Proceedings*. 2546–2554.

[13] Snoek, J., H. Larochelle, and R. P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems Proceedings*. 2951–2959.

[14] Hutter, F., H. H. Hoos, and K. Leyton-Brown. 2010. *Sequential model-based optimization for general algorithm configuration*. University of British Columbia, Computer Science. TR-2010-10.

[15] Thornton, C., F. Hutter, and H. Hoos. 2013. Auto-WEKA: Automated selection and hyperparameter optimization of classification algorithms. *19th ACM SIGKDD Conference (International) on Knowledge Discovery and Data Mining Proceedings*. doi: 10.1145/2487575.2487629.

[16] Leite, R., P. Brazdil, and J. Vanschoren. 2012. Selecting classification algorithms with active testing. *Machine learning and data mining pattern recognition*. Springer. 117–131.

[17] Vorontsov, K. V. *Matematicheskie metody obucheniya po pretsedentam (teoriya obucheniya mashin)* [Mathematical methods of training on precedents (machine learning theory)]. `http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf` (accessed November 22, 2016).

[18] Sutton, R. S., and A. G. Barto. 1998. *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press. 342 p.

Received August 18, 2016