БИОФИЗИКА, 2018, том 63, вып. 6, с. 1226–1237

ДИСКУССИИ

УДК 577.3

АНАЛИЗ МЕТОДОВ НАСТРОЙКИ МАШИН ОПОРНЫХ ВЕКТОРОВ ДЛЯ ЗАДАЧИ БИНАРНОЙ КЛАССИФИКАЦИИ

© 2018 г. Н.О. Кадырова, Л.В. Павлова

Институт прикладной математики и механики Санкт-Петербургского политехнического университета Петра Великого, 195251, Санкт-Петербург, Политехническая ул., 29

E-mail: natalia.kadyrova@gmail.com

Поступила в редакцию 19.07.18 г.

Методы, основанные на машинах опорных векторов (Support Vector Machines, SVM), позволяют обрабатывать данные большого объема и большой размерности, в том числе и плохо структурированные, что особенно важно при решении задач прогнозирования в области биоинформатики. Данная работа посвящена определяющему этапу в построении машин опорных векторов – выбору модели. Проведен анализ методов настройки машин опорных векторов для задачи бинарной классификации. Рассмотрены и реализованы два альтернативных подхода к определению наилучших значений гиперпараметров: градиентный и метод поиска с чередующимися окрестностями. Проведено сравнение результатов, полученных с использованием указанных методов на эталонных данных различной природы. Подобрана стратегия перевыбора для корректного оценивания качества построенной модели в процессе настройки машины. Результаты проведенного исследования продемонстрировали возможность выработки эффективной методики построения машин опорных векторов для задач бинарной классификации.

Ключевые слова: машина опорных векторов, настройка машины, метод поиска с чередующимися окрестностями, градиентный метод настройки, стратегия вложенного перевыбора.

DOI: 10.1134/S0006302918060236

Одной из наиболее популярных современных методологий машинного обучения с учителем, которая относится к инструментарию интеллектуального анализа данных, является *машина опорных векторов*, известная в англоязычной литературе под названием Support Vector Machine (SVM). Это новый тип методов статистического оценивания функций (или новый вид машин обучения).

Практическое использование SVM-подхода стремительно расширяется и связано с его способностью обрабатывать громадные объемы информации и с его гибкостью при моделировании данных различной природы, в том числе плохо структурированных, что особенно важно при решении задач прогнозирования в области биоинформатики [1–4]. Использование подходящих ядерных функций в качестве количественной меры подобия двух объектов дает возможность построения нелинейных решающих правил на основе линейных алгоритмов, а также позволяет применять SVM-методы при анализе данных, не имеющих ясного векторного представления. SVM-методы активно применяются в различных прикладных областях, где нужно решать задачи кластеризации, классификации, прогнозирования, принятия решений.

В основе SVM-методов лежит индукционный принцип структурной минимизации риска, который обеспечивает согласованность между качеством обучения и сложностью класса аппроксимирующих функций. Принцип структурной минимизации риска оснащает SV-машины способностью к обобщению, которая и является целью статистического обучения. SVM-алгоритмы практически не требуют дополнительной априорной информации (например, функционального вида искомого решающего правила) и преодолевают «проклятие размерности».

Основные принципы и идеи современного подхода к задачам бинарной классификации и восстановления регрессии, основанного на машинах опорных векторов, рассмотрены нами в работе [5]. Основные алгоритмы построения машин опорных векторов для задач бинарной классификации и восстановления регрессии и исследование их сравнительной эффективности представлены в работах [6,7].

Сокращения: SVM – машины опорных векторов (Support Vector Machines), VNS – метод поиска с чередующимися окрестностями (Variable Neighborhood Search).

Данная работа посвящена важной и актуальной задаче выбора модели, которая является определяющим этапом построения машины опорных векторов. Процесс построения машины опорных векторов состоит из двух основных этапов: настройки машины (выбора модели) и определения оптимальных параметров настроенной машины (обучения машины). Настройка машины предполагает поиск таких значений гиперпараметров, параметров регуляризации и ядра, которые приводят к наиболее высокому качеству машины. Это определяющий и самый дорогой в вычислительном отношении этап построения SVM. Исследователи, применяющие SVM-методы, крайне заинтересованы в его автоматизации. Настройка машины – нетривиальный процесс, для которого до сих пор не построено теоретическое обоснование. В последние годы предложен целый ряд подходов к этой проблеме, но единая методика еще не выработана.

Настройка машины может проводиться на основе различных методов оптимизации: методов поиска на решетке, методов случайного поиска, эволюционных методов на основе генетических алгоритмов, методов, основанных на использовании градиента функции валидации.

Общепринятый метод настройки параметров – поиск на решетке (grid search) [8]. Сначала используют грубую решетку, затем для полученной области пространства гиперпараметров, отвечающей приемлемому качеству машины, выбирают более мелкое разбиение. Этот метод настройки является весьма затратным по времени и применим только в случае малого количества гиперпараметров.

В работе [9] был предложен алгоритм настройки Grid-Quadtree, который интегрирует структуру поиска quadtree в поиск на решетке. Такой метод существенно превосходит стандартный поиск на решетке по количеству вычислительных операций.

В последнее десятилетие появились работы, посвященные настройке SVM на основе различных методов случайного поиска, например [10–14].

Поиск по образцу (pattern search) состоит в варьировании с определенным шагом одного из гиперпараметров в окрестности его начального значения при фиксированных остальных. Процедура проводится для всех гиперпараметров, шаг поиска постепенно уменьшается. Настройка SVM с использованием поиска по образцу предложена в работе [10]. Метод имитации отжига (simulated annealing) основан на имитации физического процесса, происходящего при кристаллизации вещества, в том числе при отжиге металлов. Применение метода не гарантирует нахождения глобального экстремума целевой функции, но при корректной генерации случайных точек во время поиска, как правило, происходит улучшение ее значения. В работе [14] для настройки SVM применяли метод имитации отжига. Различные модификации стохастического метода поиска с чередующимися окрестностями, описанные в работах [15–17], быстро и эффективно находят значения гиперпараметров, улучшающие способность SV-классификатора к прогнозированию.

Последовательная оптимизация параметров (Sequential Parameter Optimization, SPO) – метаалгоритм, состоящий из двух этапов. На первом этапе («построение модели») машина, для гиперпараметров которой осуществляется поиск, строится несколько раз для каждого набора значений множества расчетных точек. Таким образом, снижается влияние шума. На втором этапе («использование и улучшение модели») для наилучшего набора значений, полученного на первом этапе, на основе определенной функции полезности находится множество дополнительных расчетных точек. Строящаяся машина проверяется и на них. Программное обеспечение метода, описанного в работе [18], реализовано в пакете Sequential Parameter Optimization Toolbox (SPOT) для R [19].

В работе [20] предложен итерационный метод оптимизации, основанный на оценивании значения целевой функции для набора ожидаемых решений (Efficient Parameter Selection via Global Optimization, EPSGO). Для этого используется гауссов процесс с параметрами, полученными методом максимального правдоподобия. Метод рекомендован только для пространства гиперпараметров небольшой размерности.

Методы настройки, основанные на генетических алгоритмах [21,22], моделируют эволюционный процесс: на существующую популяцию (набор значений гиперпараметров), которая размножается, воздействуют мутации, затем происходит естественный отбор на основе оптимизации валидационной функции. Примером подобных методов является алгоритм эволюционной стратегии адаптации матрицы ковариаций (Covariance Matrix Adaptation Evolution Strategy, CMA-ES) [22]. На каждой итерации алгоритма имеется набор из n «предков». Для каждого «предка» моделируется множество «потомков» с использованием матрицы ковариаций, построенной с помощью эволюционной стратегии адаптации. Из них n лучших на следующем шаге становятся «предками». На каждом шаге матрица ковариации пополняется ин-

формацией о текущем наилучшем значении вектора гиперпараметров, поэтому поиск с большой вероятностью происходит в его окрестности. Метод эволюционной стратегии адаптации матрицы ковариаций представляет собой достаточно громоздкую процедуру, поскольку является самонастраивающимся.

Подход к настройке машины опорных векторов, использующий градиент функции валидации по гиперпараметрам, предложен в работах [23–25]. Для использования градиентного метода обычно приходится производить сглаживание функции валидации, той или иной характеристики качества машины. Удобно использовать сигмоидальное сглаживание функций, однако при этом возникает необходимость оценивания дополнительных параметров. Поскольку валидационная функция зависит от гиперпараметров неявно, вычисление ее градиента представляет собой еще одну нетривиальную задачу.

С целью выработки эффективной методики построения машин опорных векторов для задачи бинарной классификации из ряда предложенных в литературе выбраны и реализованы два альтернативных подхода: стохастический метод поиска с чередующимися окрестностями [13], не требующий дифференцируемости целевой функции по гиперпараметрам, и метод, основанный на вычислении градиента функции валидации, отражающей качество машины [24]. Оба подхода не ограничивают число настраиваемых параметров и определяют их наилучшие значения при небольшом количестве прогонов SVM-алгоритма. Выбор методов настройки определялся, прежде всего, их вычислительной эффективностью, универсальностью и простотой, отсутствием дополнительных параметров и существенных ограничений.

МЕТОД ПОИСКА С ЧЕРЕДУЮЩИМИСЯ ОКРЕСТНОСТЯМИ

Метод поиска с чередующимися окрестностями (Variable Neighborhood Search, VNS) – метаэвристика, относящаяся к числу алгоритмов, не требующих существования производных оптимизируемых функций. Метаэвристики – стратегии, управляющие процессом поиска решения, которые позволяют эффективно исследовать пространство поиска для нахождения (почти) оптимальных решений. Метод VNS относится к методам локального поиска, которые начинают свою работу со стартового значения, на каждом шаге заменяя результат лучшим решением, найденным в окрестности текущего решения.

Метод с чередующимися окрестностями использует введение структуры окрестностей, чтобы избежать локальных экстремумов. Он заключается в последовательном движении по определенному множеству значений гиперпараметров и поиске оптимального решения в соседстве с текущим наилучшим значением. Предполагается последовательное исследование заявленных соседств для достижения наилучшего результата. При этом размер соседств изменяется, чтобы избежать ловушек локального экстремума.

Пусть $\boldsymbol{\theta}$ – вектор гиперпараметров размерности p; Θ – множество допустимых значений $\boldsymbol{\theta}$. Тогда класс ядер K может быть представлен в виде $K = \{K(\boldsymbol{\theta}): \boldsymbol{\theta} \in \Theta\}$. Например, для гауссового ядра $\boldsymbol{\theta} = (C, \sigma)$. Поскольку в теории машин опорных векторов не делается никаких предположений о характере распределения исходных данных, значение вероятности корректной классификации вновь поступающих объектов $a(\cdot)$ вычислить невозможно. Получим оценку качества модели $\hat{a}(K(\boldsymbol{\theta}))$ с помощью процедуры перекрестной проверки. Используя метод VNS, описанный в работе [13], найдем значение вектора гиперпараметров $\boldsymbol{\theta} \in \Theta$, при котором достигается наилучшее качество классификации.

Метод VNS в случайном порядке исследует окрестности текущего решения и заменяет его, если находится лучшее решение. Поиск стартует с первого соседства. Если сгенерированное случайным образом решение оказывается не лучше текущего, то выбирается следующее соседство. Однако в случае, когда новое решение приводит к построению классификатора более высокого качества, счетчик итераций увеличивается, а исследование продолжается снова с первого соседства, но построенного уже относительно нового решения.

Алгоритм 1. Алгоритм VNS для настройки машины опорных векторов.

Входные данные:

– множество ядер $K = \{K(\boldsymbol{\theta}): \boldsymbol{\theta} \in \Theta\}$;

– максимальное число итераций m;

– структура соседств: $\{N_1, N_2, \ldots, N_{K_{\max}}\}$, где $K_{\max}$ – количество соседств;

$N_k(\tilde{\boldsymbol{\theta}}) = \{\boldsymbol{\theta} \in \Theta: \|\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_\infty \le r_k\}$, r_k – радиус соседства N_k, $k = 1, \ldots, K_{\max}$.

Инициализация переменных:

Выбрать начальное значение $\tilde{\boldsymbol{\theta}} \in \Theta$;

установить счетчик соседств $k \leftarrow 1$ и счетчик итераций iter $\leftarrow 0$.

Шаг 1. Повтор, пока (iter < m и $k < K_{\max}$).

Шаг 1.1. Перемешивание.

Выбор случайного решения $\boldsymbol{\theta}'$ внутри соседства k, построенного относительно текущего решения $\tilde{\boldsymbol{\theta}}$, $\boldsymbol{\theta}' \in N_k(\tilde{\boldsymbol{\theta}})$.

Шаг 1.2. Смена соседства.

Если $\hat{a}(\boldsymbol{\theta}') > \hat{a}(\tilde{\boldsymbol{\theta}})$, тогда $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}'$ и возврат к первому соседству $k \leftarrow 1$,

иначе $k \leftarrow k + 1$.

Шаг 1.3. iter $\leftarrow$ iter + 1.

Шаг 2. Если iter = m, то STOP и финальное решение $\tilde{\boldsymbol{\theta}}$,

иначе $k \leftarrow 1$ и переход на шаг 1.

Модификация алгоритма 1, осуществляющая после шага «1.1. Перемешивание» поиск локального минимума для рассматриваемого соседства, представляет собой широко известный и наиболее часто используемый вариант метода с чередующимися окрестностями, VNS-Glob, описанный в работе [15].

Алгоритм 2. VNS-Glob для настройки машины опорных векторов.

Входные данные:

– множество ядер $K = \{K(\boldsymbol{\theta}): \boldsymbol{\theta} \in \Theta\}$;

– максимальное число итераций m;

– структура соседств: $\{N_1, N_2, \ldots, N_{K_{\max}}\}$, где $K_{\max}$ – количество соседств,

$N_k(\tilde{\boldsymbol{\theta}}) = \{\boldsymbol{\theta} \in \Theta: \|\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_\infty \le r_k\}$, r_k – радиус соседства N_k, $k = 1, \ldots, K_{\max}$.

Инициализация переменных:

Выбрать начальное значение $\tilde{\boldsymbol{\theta}} \in \Theta$;

установить счетчик соседств $k \leftarrow 1$ и счетчик итераций iter $\leftarrow 0$.

Шаг 1. Повтор, пока (iter < m и $k < K_{\max}$)

Шаг 1.1. Перемешивание: Выбор случайного решения $\boldsymbol{\theta}'$ внутри соседства k, построенного относительно текущего решения $\tilde{\boldsymbol{\theta}}$, $\boldsymbol{\theta}' \in N_k(\tilde{\boldsymbol{\theta}})$.

Шаг 1.2. Применение локального спуска. В качестве начальной точки для него используется полученное на предыдущем шаге значение $\boldsymbol{\theta}'$. В результате спуска находится $\boldsymbol{\theta}''$ – локальный экстремум.

Шаг 1.3. Смена соседства: Если $\hat{a}(\boldsymbol{\theta}'') > \hat{a}(\tilde{\boldsymbol{\theta}})$, тогда $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}''$ и возврат к первому соседству $k \leftarrow 1$, иначе $k \leftarrow k + 1$.

Шаг 1.4. iter $\leftarrow$ iter + 1.

Шаг 2. Если iter = m, то STOP и финальное решение $\tilde{\boldsymbol{\theta}}$, иначе $k \leftarrow 1$ и переход на шаг 1.

Согласно алгоритму 2, после выбора случайным образом значения в окрестности текущего результата внутри рассматриваемого соседства будет проведен поиск локального экстремума. Если полученный локальный экстремум не лучше текущего решения, то процедура повторяется для следующего соседства. В противном случае поиск вновь стартует из первого соседства.

VNS-Glob конкурентоспособен в тех случаях, когда размерность элементов множества Θ невелика. Однако для ситуации, когда p достаточно большое, например, для класса ядер $K = \{\Sigma_{j=1}^{R}\mu_j K_j: \mu_j \ge 0, K_j \in K_j, \forall j = 1, \ldots, R\}$, модифицированный метод вряд ли достигнет области подходящих решений. В этом случае используется метод поиска с чередующимися окрестностями без модификаций.

Важным этапом рассмотренных алгоритмов является выбор нового значения в окрестности текущего решения. С этой точки зрения полезны модификации VNS, предложенные в работе [17], где очередное значение на шаге «Перемешивание» определяется согласно определенному вероятностному распределению, например распределению Гаусса.

Некоторые SVM-модели можно рассматривать как вложенные, т.е. имеющие вложенную ядерную структуру $K_{(1)} \subset K_{(2)} \subset \ldots \subset K_{(M)}$, где M – уровень вложенности. Предложенный в работе [13] вложенный вариант VNS, настраивающий такую модель, при поиске наилучшего значения для внешней модели $K_{(m+1)}$ использует решение, найденное для внутренней модели $K_{(m)}$.

ГРАДИЕНТНЫЙ МЕТОД

Эффективный метод настройки машины опорных векторов, использующий градиент функции валидации, предложен в работе [24].

Рассмотрим двойственную задачу стандартного SVM-обучения для бинарной классификации с петлевой функцией потерь: найти $\min_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = -\Sigma_{i=1}^{l}\alpha_i + \frac{1}{2}\Sigma_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ при ограничениях $0 \le \alpha_i \le C$,

$$\Sigma_{i=1}^{l}\alpha_i y_i = 0. \quad (1)$$

Здесь $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l) \in R^n \times \{-1; +1\}$ – тренировочная последовательность объема l; $\alpha_i \ge 0$ – множители Лагранжа; $C > 0$ – параметр регуляризации, контролирующий ширину полосы; $K(\boldsymbol{x}, \boldsymbol{x}')$ – положительно определенная ядерная функция.

Оптимальная решающая функция представляет собой линейную комбинацию значений ядра на тренировочных векторах: $f(\boldsymbol{x}) = \Sigma_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) - b$, где $b \in R$ – параметр сдвига.

Петлевая функция потерь имеет вид $L(f_i, y_i) = \max\{0, 1 - f_i y_i\}$, где f_i – значение решающего правила для элемента $\boldsymbol{x}_i$.

Условия оптимальности для двойственной задачи (1) приводят к системе линейных алгебраических уравнений относительно параметров α и b:

$$\boldsymbol{P}\begin{pmatrix}\alpha\\ b\end{pmatrix} = \boldsymbol{q}, \tag{2}$$

где матрица $\boldsymbol{P}$ и вектор $\boldsymbol{q}$ в общем случае зависят от вектора гиперпараметров θ. Будем предполагать, что ядро и элементы $\boldsymbol{P}$ и $\boldsymbol{q}$ – непрерывно дифференцируемые по гиперпараметрам функции.

Рассмотрим $\boldsymbol{P}$ и $\boldsymbol{q}$ для петлевой функции потерь. После построения SV-классификатора индексы тренировочного множества можно разделить на три группы:

$$I_0 = \{i: \alpha_i = 0\},\ I_c = \{i: \alpha_i = C\},\ I_u = \{i: 0 < \alpha_i < C\}.$$

Пусть α_0, α_c, α_u, $\boldsymbol{y}_c$, $\boldsymbol{y}_u$, $\boldsymbol{e}_c$, $\boldsymbol{e}_u$, $\boldsymbol{Q}_{uc}$, $\boldsymbol{Q}_{uu}$ обозначают векторы и матрицы для соответствующих множеств индексов. Здесь $\boldsymbol{e}$ – векторы из единиц, $\boldsymbol{Q}$ – матрицы из элементов $y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Тогда систему (2) можно представить в следующем виде:

$$\alpha_0 = \boldsymbol{0},\ \alpha_c = C\boldsymbol{e}_c,\ \begin{pmatrix}\boldsymbol{Q}_{uu} & -\boldsymbol{y}_u\\ -\boldsymbol{y}_u^{\mathrm{T}} & \boldsymbol{0}\end{pmatrix}\begin{pmatrix}\alpha_u\\ b\end{pmatrix} = \begin{pmatrix}\boldsymbol{e}_u - \boldsymbol{Q}_{uc}\alpha_c\\ \boldsymbol{y}_c^{\mathrm{T}}\alpha_c\end{pmatrix}.$$

Пусть $\{\tilde{\boldsymbol{x}}_j, \tilde{\boldsymbol{y}}_j\}_{j=1}^{\tilde{l}}$ – валидационная последовательность (подмножество данных, с помощью которого проверяется качество очередной построенной в процессе настройки SV-машины), а $\tilde{K}_{ji} = K(\tilde{\boldsymbol{x}}_j, \boldsymbol{x}_i)$ определяет значение ядерной функции для элемента j из этой последовательности и элемента i из тренировочного множества. Тогда значение решающего правила для элемента $\tilde{\boldsymbol{x}}_j$:

$$\tilde{f}_j = \Sigma_i \alpha_i y_i \tilde{K}_{ji} - b,$$

может быть представлено в виде

$$\tilde{f}_j = \boldsymbol{v}_j^{\mathrm{T}}\beta,$$

где β – вектор, содержащий и α, и b, а $\boldsymbol{v}_j$ состоит из $y_i\tilde{K}_{ji}$ для всех $i = 1, \ldots, l$ и -1 в качестве последнего элемента (что соответствует элементу b вектора β).

Проблему выбора модели, т.е. определения оптимальных элементов вектора гиперпараметров, сформулируем в виде:

$$\text{найти } \theta^* = \operatorname*{argmin}_{\theta} g(\tilde{f}_1, \ldots, \tilde{f}_{\tilde{l}}), \tag{3}$$

где g – дифференцируемая функция валидации от откликов $\tilde{f}_j$, которые неявно зависят от θ. Будем вычислять градиент $\nabla_\theta g$.

Обозначим через τ некоторый гиперпараметр, являющийся элементом вектора θ, а частную производную какой-либо величины ϑ по параметру τ – через $\dot{\vartheta}$.

Чтобы найти $\dot{\beta}$, нужно продифференцировать (2) по τ: $\boldsymbol{P}\dot{\beta} + \dot{\boldsymbol{P}}\beta = \dot{\boldsymbol{q}}$, отсюда

$$\dot{\beta} = \boldsymbol{P}^{-1}(\dot{\boldsymbol{q}} - \dot{\boldsymbol{P}}\beta). \tag{4}$$

Теперь рассмотрим $\dot{g}$:

$$\dot{g} = \Sigma_{j=1}^{\tilde{l}} \frac{\partial g}{\partial \tilde{f}_j}\dot{\tilde{f}}_j,$$

где $\dot{\tilde{f}}_j = \boldsymbol{v}_j^{\mathrm{T}}\dot{\beta} + \dot{\boldsymbol{v}}_j^{\mathrm{T}}\beta$.

Часто ранг $\boldsymbol{P}$ намного меньше ее размерности, например, в случае петлевой функции потерь. Тогда вместо соотношения (4), чтобы избежать вычисления $\boldsymbol{P}^{-1}$, можно использовать соотношение

$$\boldsymbol{P}\dot{\beta} = (\dot{\boldsymbol{q}} - \dot{\boldsymbol{P}}\beta) \tag{5}$$

для получения приближенного значения вектора $\dot{\beta}$, например, методом сопряженных градиентов. Поскольку правая часть соотношения (5) зависит от параметра τ, по которому происходит дифференцирование, необходимо находить решение задачи (3) для каждого элемента θ, что тоже заметно увеличивает вычислительную сложность при большом количестве гиперпараметров. Однако оказывается, что для вычисления полного градиента g относительно всех элементов θ, достаточно решить *только* одну систему линейных уравнений.

Определим коэффициент перед множителем $\dot{\tilde{f}}_j$, необходимый для вычисления $\dot{g}$, обозначив его через ψ_j:

$$\psi_j = \frac{\partial g}{\partial \tilde{f}_j}, \tag{6}$$

тогда

$$\dot{g} = \Sigma_{j=1}^{\tau}\psi_j\dot{\tilde{f}}_j = \Sigma_{j=1}^{\tau}\psi_j(\boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{P}^{-1}(\dot{\boldsymbol{q}} - \dot{\boldsymbol{P}}\beta) + \dot{\boldsymbol{v}}_j^{\mathrm{T}}\beta) = \quad (7)$$
$$= \boldsymbol{d}^{\mathrm{T}}(\dot{\boldsymbol{q}} - \dot{\boldsymbol{P}}\beta) + (\Sigma_{j=1}^{\tau}\psi_j\dot{\boldsymbol{v}}_j)^{\mathrm{T}},$$

где вектор $\boldsymbol{d}$ – решение уравнения

$$\boldsymbol{P}^{\mathrm{T}}\boldsymbol{d} = (\Sigma_{j=1}^{\tau}\psi_j\boldsymbol{v}_j). \quad (8)$$

Преимущество данного преобразования в том, что вектор $\boldsymbol{d}$ не зависит от параметра τ, по которому происходит дифференцирование. Таким образом, $\boldsymbol{d}$ необходимо найти лишь один раз.

В случае петлевой функции потерь вычисления $\dot{\boldsymbol{P}}\beta$ для части $\boldsymbol{P}$, соответствующей α_0, можно проигнорировать. Пусть ($\boldsymbol{d}_0$, $\boldsymbol{d}_u$, $\boldsymbol{d}_c$, d_b) и ($\boldsymbol{r}_0$, $\boldsymbol{r}_u$, $\boldsymbol{r}_c$, r_b) обозначают соответствующие части векторов $\boldsymbol{d}$ и $\Sigma_{j=1}^{\tau}\psi_j\boldsymbol{v}_j$, тогда выражение (7) может быть представлено в эквивалентном виде:

$$\begin{pmatrix}\boldsymbol{Q}_{uu} & -\boldsymbol{y}_u\\ -\boldsymbol{y}_u^{\mathrm{T}} & \boldsymbol{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{d}_u\\ d_b\end{pmatrix} = \begin{pmatrix}\boldsymbol{r}_u\\ r_b\end{pmatrix},\ \boldsymbol{d}_c = \boldsymbol{r}_c - \boldsymbol{Q}_{uc}^{\mathrm{T}}\boldsymbol{d}_u + \boldsymbol{y}_c d_b,\ \boldsymbol{d}_0 = 0.$$

Итак, для вычисления полного градиента g по θ следует:

1. Вычислить ψ_j по формуле (6).

2. Решить систему линейных алгебраических уравнений (8) и получить вектор $\boldsymbol{d}$.

3. Для каждой компоненты вектора θ найти значение $\dot{g}$ по формуле (7).

Вычисление $\dot{\boldsymbol{P}}\beta$ производится для каждого параметра отдельно. В задачах с большим количеством гиперпараметров этот этап расчетов – самая ресурсоемкая часть. Заметим, что для $\tau = C$ значение $\dot{\boldsymbol{P}}\beta$ очевидно, а для $\tau = \sigma$ можно сэкономить вычислительные затраты, кэшируя значения ядерной матрицы.

После вычисления градиента фунции валидации задача поиска гиперпараметров может быть решена квазиньютоновскими методами, например, с помощью алгоритма Бройдена–Флетчера–Гольдфарба–Шанно, для работы которого требуются только значения функции и ее градиента. Рекомендуется заканчивать процесс оптимизации при выполнении условия:

$$|g(\theta^{(k+1)}) - g(\theta^{(k)})| \le 10^{-3}|g(\theta^{(k)})|,$$

где $\theta^{(k+1)}$ и $\theta^{(k)}$ – результаты последовательных итераций оптимизационного процесса.

Представим функцию валидации как функцию от элементов матрицы несоответствий:

$$g = g(tp, fp).$$

Таблица 1. Матрица несоответствий

True ↓ \ Pred→	+1	–1
+1	tp	$\tau_+ - tp$
–1	fp	$\tau_- - tp$

Табл. 1 содержит матрицу несоответствий, где tp – количество истинно положительных образцов, полученных с помощью выбранного решающего правила, fp – количество ложно положительных образцов, τ_+ – количество положительных меток и τ_- – количество отрицательных меток в валидационной последовательности.

Пусть $u(z)$ – функция Хевисайда. Определим

$$\tilde{u}_j = u(\tilde{y}_j\tilde{f}_j) =$$
$$= \begin{cases}1, \text{ если образец правильно классифицирован}\\ 0 \text{ при ошибочной классификации.}\end{cases}$$

Тогда $tp = \Sigma_{j:\,\tilde{y}_j=+1}\tilde{u}_j$, $fp = \Sigma_{j:\,\tilde{y}_j=-1}(1-\tilde{u}_j)$.

Пусть функция валидации представляет собой долю неправильно классифицированных образцов: $Er = \dfrac{\tau_+ - tp + fp}{\tau}$. Для использования градиентного метода необходимо произвести сглаживание функции валидации. Пусть $\tilde{s}_j$ определяет сигмоидальное приближение $\tilde{u}_j$:

$$\tilde{s}_j = \frac{1}{1 + \exp(-\gamma_1\tilde{y}_j(\tilde{f}_j - \gamma_0))},$$

где $\gamma_1 > 0$ – сигмоидальный параметр масштаба, влияющий на близость приближения $\tilde{s}_j$ к $\tilde{u}_j$ и, следовательно, контролирующий степень сглаживания. В общем случае γ_0 и γ_1 могут быть функциями от значений валидационной функции.

Рассмотрим функцию валидации при таком сглаживании: $g = g(\tilde{s}_1, \ldots, \tilde{s}_\tau)$. Тогда значение ψ_j может быть получено следующим образом:

$$\psi_j = \frac{\partial g}{\partial \tilde{s}_j}\frac{\partial \tilde{s}_j}{\partial \tilde{f}_j} + \left(\Sigma_r\frac{\partial g}{\partial \tilde{s}_r}\frac{\partial \tilde{s}_r}{\partial \gamma_0}\right)\frac{\partial \gamma_0}{\partial \tilde{f}_j} + \left(\Sigma_r\frac{\partial g}{\partial \tilde{s}_r}\frac{\partial \tilde{s}_r}{\partial \gamma_1}\right)\frac{\partial \gamma_1}{\partial f_j},$$

где частные производные $\tilde{s}_j$ по $\tilde{f}_j$, γ_0, γ_1:

$$\frac{\partial \tilde{s}_j}{\partial \tilde{f}_j} = \tilde{s}_j(1-\tilde{s}_j)\gamma_1\tilde{y}_j,\ \frac{\partial \tilde{s}_r}{\partial \gamma_0} = -\tilde{s}_r(1-\tilde{s}_r)\gamma_1\tilde{y}_r,$$

$$\frac{\partial \tilde{s}_r}{\partial \gamma_1} = \tilde{s}_r(1-\tilde{s}_r)\tilde{y}_r(\tilde{f}_r-\gamma_0).$$

С учетом сигмоидальной аппроксимации имеем:

$$tp \approx \Sigma_{j:\,\tilde{y}_j=+1}\tilde{s}_j,\ fp \approx \Sigma_{j:\,\tilde{y}_j=-1}(1-\tilde{s}_j).$$

Частную производную g можно представить в виде:

$$\frac{\partial g}{\partial \tilde{s}_j} = \frac{\partial g}{\partial tp}\frac{\partial tp}{\partial \tilde{s}_j} + \frac{\partial g}{\partial fp}\frac{\partial fp}{\partial \tilde{s}_j}. \quad (9)$$

Чтобы вычислить уравнение (9), потребуются частные производные g по tp и fp. Соответствующие частные производные для функции валидации:

$$\frac{\partial Er}{\partial tp} = -\frac{1}{\tilde{l}},\ \frac{\partial Er}{\partial fp} = \frac{1}{\tilde{l}}.$$

Определим параметры γ_0 и γ_1 согласно прямому методу сигмоидального сглаживания:

$$\gamma_0 = 0,\ \gamma_1 = \frac{t}{c}.$$

Здесь ρ – выборочное стандартное отклонение откликов $\{\tilde{f}_j\}$, соответствующих элементам $\tilde{\boldsymbol{x}}_j$ валидационной последовательности, т.е. $\rho = \sqrt{\frac{\Sigma_k(\tilde{f}_k-\mu)^2}{\tilde{l}}}$; μ – их выборочное среднее, $\mu = \frac{1}{\tilde{l}}\Sigma_k\tilde{f}_k$; t – константа, для которой эвристически было установлено значение $t = 10$, используемая для вычисления шага функции. Тогда

$$\frac{\partial \gamma_0}{\partial \tilde{f}_j} = 0,\ \frac{\partial \gamma_1}{\partial \tilde{f}_j} = -\frac{t}{\tilde{l}\rho^3}(\tilde{f}_j-\mu).$$

Выбрав критерий качества классификатора, функцию валидации, сгладив ее, вычислив градиент сглаженной функции по гиперпараметрам и определившись со стандартными методами решения системы линейных алгебраических уравнений и градиентным методом оптимизации, можем успешно решать поставленную задачу настройки машины.

ИНИЦИАЛИЗАЦИЯ ОПТИМИЗАЦИОННОГО ПРОЦЕССА ДЛЯ НАСТРОЙКИ МАШИНЫ

Выбор начальной точки для метода, оптимизирующего гиперпараметры машины опорных векторов, чрезвычайно важен. В случае двух гиперпараметров возможен ее осмысленный выбор. В работе [26] предпринято исследование асимптотического поведения ошибки обобщения (точнее, ее оценок, таких как leave-one-out-ошибка или ошибка, полученная с помощью процедуры перекрестной проверки) для стандартного классификатора с гауссовым ядром. Для этого случая показано, что пространство гиперпараметров, параметра регуляризации C и параметра ядра σ^2 имеет достаточно определенные очертания. Чтобы избежать дополнительных ограничений в задаче оптимизации ($C > 0$ и $\sigma^2 > 0$), а также для устойчивости процесса оптимизации работают с логарифмами исходных параметров.

В пространстве поиска значений логарифмов гиперпараметров может быть выделена область, которая не содержит значения, ведущие к недообучению или к переобучению. Пусть $C_{\lim} = 2\frac{l_1}{l}$, где l_1, l_2 – количества образцов тренировочной последовательности соответствующего класса, удовлетворяющие условию $l_1 > l_2 + 1 > 2$. Показано, что при малом σ^2 и $C \geq \tilde{C}_{\lim}$ SV-классификатор перестает зависеть от C. Поэтому контур области подходящих значений параллелен оси $\log C$ в случае малого значения σ^2 и больших C. При $\sigma^2 \to \infty$ SV-классификатор с гауссовым ядром ведет себя подобно линейному классификатору. Пусть $\tilde{C}$ – наилучшее значение параметра регуляризации для линейного классификатора. Показано, что $\log\sigma^2 = \log C - \log\tilde{C}$.

Таким образом, для осмысленного выбора начальной точки процесса оптимизации гиперпараметров машины можно использовать следующую процедуру поиска области подходящих значений параметров:

1. Найти наилучшее значение параметра регуляризации $\tilde{C}$ для линейного классификатора.

2. Построить область пар (C, σ^2), удовлетворяющих выражению $\log\sigma^2 = \log C - \log\tilde{C}$ при найденном фиксированном значении $\tilde{C}$.

В результате подобной процедуры количество переборов уменьшится на порядок по сравнению с общепринятым поиском на решетке.

СТРАТЕГИИ ФОРМИРОВАНИЯ ВЫБОРОК ПРИ НАСТРОЙКЕ МАШИНЫ ОПОРНЫХ ВЕКТОРОВ

Процесс построения машины опорных векторов по существу состоит из двух этапов: настройки машины (model selection) и определения оптимальных параметров настроенной машины (model evaluation). Настройка машины состоит в поиске таких значений гиперпараметров, т.е. параметров регуляризации и ядра, которые приводят к наиболее высокому качеству машины. Корректно оценить качество построенной модели в процессе настройки машины позволяет правильно подобранная стратегия перевыбора (resampling technique).

Можно разбить исходные данные на два непересекающихся множества, производить обучение машины по одному из них, а на другом тестировать построенную машину (процедура hold-out). При этом возникают следующие проблемы: во-первых, требуется большой объем исходной выборки для получения статистически обоснованных результатов; во-вторых, невозможно определить дисперсию и устойчивость характеристик модели в связи с изменениями в тренировочном множестве. Для решения этих проблем служат различные методы перевыбора: процедура k-кратной перекрестной проверки; процедура бутстреп (bootstrap), использующая случайный выбор с возвращением; процедура subsampling, использующая случайный выбор без возвращения.

Для того чтобы избежать явления переобучения при построении машин опорных векторов, рекомендуется применять стратегию вложенного перевыбора [27], состоящую в следующем: в то время, как качество модели оценивается во внешнем цикле, каждая тренировочная последовательность снова подвергается перевыбору во внутреннем цикле для настройки гиперпараметров.

Итак, для успешного построения машины опорных векторов и получения корректной оценки ее качества следует:

1. Разбить исходные данные на три непересекающихся множества: тренировочное для обучения машины; валидационное для настройки гиперпараметров; тестовое, предназначенное исключительно для оценивания качества уже настроенной модели.

2. Выбрать ядро и зафиксировать значения гиперпараметров.

3. Обучить модель на тренировочном множестве.

4. Определить качество модели на валидационном множестве.

5. Повторить шаги 2–4, используя различные значения гиперпараметров (и/или ядра).

6. Выбрать лучшую модель и обучить ее на данных из тренировочного и валидационного множеств.

7. Оценить качество построенной машины по данным из тестового множества.

При использовании k-кратной перекрестной проверки или бутстрепа шаги 3 и 4 нужно повторить для каждого варианта перевыбора.

РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

Для решения задачи бинарной классификации применен стандартный SVM-подход, использовано гауссово ядро и петлевая функция потерь. В качестве функции валидации использована доля ошибок классификации.

Реализованы в среде MatLab два различных подхода к решению задачи настройки параметров: градиентный метод (далее Grad) и метод с чередующимися окрестностями без модификаций (далее VNS). Их исследование проведено на эталонных данных различной природы: Splice – данные о границах сплайсинга нуклеотидных последовательностей (3175 образцов, 60 признаков) и German – данные кредитования немецкого банка (1000 образцов, 24 признака) [28]. Каждая выборка случайным образом поделена на два непересекающихся множества – тренировочное и тестовое, предназначенное исключительно для оценивания качества уже настроенной машины: Splice (1000 + 2175), German (700 + 300). Такое разбиение выборок обусловлено возможностью сравнения полученных результатов с результатами, приведенными в литературе [13,24].

Чтобы избежать переобучения и получить (почти) несмещенную оценку характеристики качества машины, использована стратегия вложенного перевыбора с двумя вложенными процедурами перекрестной проверки: пятикратной во внешнем цикле и четырехкратной во внутреннем.

На каждом шаге внешнего цикла настроенные во внутреннем цикле гиперпараметры были использованы для построения машины, которую тестировали на соответствующей части исходного тренировочного множества. Полученные оценки доли ошибок усредняли, а также использовали для вычисления стандартного отклонения оценки характеристики качества машины.

Значения гиперпараметров, полученные на внешнем цикле, при которых строилась модель с наименьшей оценкой доли ошибки классифи-

Таблица 2. Сравнение результатов классификации

Данные	VNS ([13])	VNS	Grad ([24])	Grad
German	0,2133	0,2067	–	0,2067
Splice	0,0993	0,0956	0,0817	0,0970

кации, использовали для построения машины опорных векторов по всему исходному тренировочному множеству. Качество таким образом построенной машины проверяли на исходном тестовом множестве, элементы которого не участвовали в ее настройке и обучении.

В табл. 2 приведена доля ошибок классификации, полученная на тестовом множестве. Эксперимент проводили при числе итераций, равном 54, количестве соседств, равном 25, радиусе соседства, равном его номеру (для VNS) из начальной точки (–3, 2). Выбор условий проведения эксперимента, как и разбиение выборок, обусловлен возможностью сравнения полученных результатов с результатами, приведенными в литературе.

В табл. 3 приведены результаты работы SV-классификаторов до и после очистки данных от шумящих признаков. Для очистки данных применяли процедуру, основанную на использовании линейного дискриминанта Фишера [29]. После очистки данных Splice осталось 33 признака, после очистки данных German – 15 признаков. Представленные результаты подтверждают устойчивость метода опорных векторов к шуму. При этом исключение шумящих признаков приводит к существенному сокращению времени вычислений.

В табл. 4 представлены результаты анализа поведения методов настройки при выборе начальных приближений из благоприятной области значений гиперпараметров, которая не содержит значения, ведущие к недообучению или к переобучению. Эту область определяли с привлечением линейного классификатора [26]. При выборе стартовой точки из благоприятной области качество построенного SV-классификатора практически не меняется и соответствует уровню, достигнутому в работах [13,24]. Результаты экспериментов демонстрируют устойчивость данных методов настройки к начальному приближению в случае его выбора из благоприятной области.

Проведено исследование с целью определения объема тренировочного множества, при котором получаемые значения гиперпараметров еще остаются конкурентноспособными. В табл. 5 приведены результаты экспериментов для данных Splice. С уменьшением размера тренировочной последовательности все еще достигается приемлемый результат, при этом время расчетов по отношению ко времени расчетов по полной выборке заметно сокращается.

ЗАКЛЮЧЕНИЕ

Рассмотрен один из современных подходов из области машинного обучения с учителем к решению задачи бинарной классификации, относящийся к технологии интеллектуального анализа данных.

Проведен анализ методов настройки машин опорных векторов для задачи бинарной классификации.

Таблица 3. Влияние процедуры очистки от шума на качество классификации

	C	σ	Оценка доли ошибки после настройки	Доля ошибки при тестировании	Число признаков
VNS	Splice				
	51,0797	6,5225	0,1750 ± 0,0490	0,0989	60
	36,1479	2,8694	0,1520 ± 0,0326	0,0846	33 (60)
	German				
	1090,9233	36,7607	0,2114 ± 0,0487	0,2067	24
	42,0485	10,1265	0,2086 ± 0,0442	0,2000	15 (24)
Grad	Splice				
	1,2040	1,3236	0,1350 ± 0,0352	0,0970	60
	1,000	1,000	0,1060 ± 0,0325	0,0851	33 (60)
	German				
	2,6096	2,7211	0,2643 ± 0,0311	0,2067	24
	2,7183	2,7183	0,2571 ± 0,0303	0,2233	15 (24)

Таблица 4. Влияние выбора начальной точки на результаты настройки машины

Начальная точка	C	σ	Оценка доли ошибки после настройки	Доля ошибки при тестировании
Splice (градиентный метод)				
[0, 0]	1,2040	1,3236	0,1350 ± 0,0352	0,0970
[1, 0]	2,7013	1,0164	0,1000 ± 0,0255	0,0878
Splice (метод VNS)				
[0, 0]	36,1479	2,8694	0,1520 ± 0,0326	0,0846
[1, 0]	22,4270	5,3391	0,1520 ± 0,0358	0,0864
German (градиентный метод)				
[–1, 0]	0,6288	1,6656	0,2614 ± 0,0275	0,2467
[1, 1]	2,7183	2,7183	0,2571 ± 0,0303	0,2233
German (метод VNS)				
[1, 1]	34,0947	7,4899	0,2071 ± 0,0491	0,2033
[–1, 0]	42,0485	10,1265	0,2086 ± 0,0442	0,2000

Таблица 5. Влияние объема тренировочной последовательности на результаты настройки машины

Объем тренировочной последовательности	C	σ	Оценка доли ошибки после настройки	Доля ошибки при тестировании	Относительное время расчетов
Splice (градиентный метод)					
1000	1,4933	1,6159	0,1390 ± 0,0439	0,0984	1,000
800	1,5236	2,0120	0,1462 ± 0,0432	0,1025	0,7918
600	2,8805	2,0044	0,1750 ± 0,0283	0,1053	0,4747
400	2,2378	1,8995	0,1800 ± 0,0326	0,1264	0,2555
Splice (метод VNS)					
1000	22,4270	5,3391	0,1520 ± 0,0358	0,0864	1,000
800	11,4291	4,6458	0,1387 ± 0,0366	0,0832	0,5576
600	7,3313	2,6268	0,1783 ± 0,0304	0,1103	0,5225
400	34,3733	4,4700	0,1900 ± 0,0609	0,1149	0,2709

Из ряда подходов, предложенных в литературе, выбраны и реализованы два альтернативных подхода настройки машин: градиентный и подход, не требующий определения градиента валидационной функции по гиперпараметрам, – метод поиска с чередующимися окрестностями.

Определена область в пространстве гиперпараметров, которая не содержит значения, ведущие к недообучению или к переобучению машины. Выбор начальной точки для инициализации процессов оптимизации осуществлен из этой области.

Подобрана стратегия перевыбора для корректного оценивания качества построенной модели в процессе настройки машины.

Проведено сравнение результатов, полученных с использованием указанных методов на эталонных данных различной природы.

Показано, что SV-классификатор, как и ожидалось, устойчив к шуму.

Исследовано влияние объема тренировочной последовательности на результат настройки машины.

Результаты проведенного исследования продемонстрировали возможность выработки эффективной методики построения машин опорных векторов для задач бинарной классификации.

Работа с реальными данными требует дальнейшего обобщения рассмотренных в работе подходов к задаче настройки машины опорных

векторов. Отметим некоторые моменты дальнейшей работы в этой области:

– большое количество гиперпараметров. Методы настройки, использованные в работе, подходят для ядер со многими параметрами;

– различные функции валидации. Методы настройки, использованные в работе, подходят для различных валидационных функций.

– многокритериальная настройка машин.

ВЫВОДЫ

Из целого ряда подходов к настройке машин опорных векторов выбраны, реализованы и исследованы два альтернативных подхода: градиентный [13] и метод, не требующий определения градиента валидационной функции по гиперпараметрам, – метод поиска с чередующимися окрестностями [24]. Выбранные методы настройки допускают большое количество гиперпараметров и работают с различными функциями валидации. Результаты проведенного в настоящей работе исследования продемонстрировали возможность выработки эффективной методики построения машин опорных векторов для задач бинарной классификации.

Эффективность выработанной методики построения машин опорных векторов, согласно анализу результатов проведенных в работе исследований, определяют:

– выбор начальной точки для инициализации процессов оптимизации настройки машины;

– существенная экономия времени расчетов, благодаря менее затратным (по сравнению с общепринятым поиском на решетке) методам настройки машины;

– существенное сокращение объема тренировочной последовательности при настройке машины.

СПИСОК ЛИТЕРАТУРЫ

1. G. Schweikert, A. Zien, G. Zeller, et al., Genome Res. **19**, 2133 (2009). http://www.genome.org/cgi/doi/10.1101/gr.090597.108.
2. Ch. Cui, Ch. Fang, and K. Han, BMC Bioinformatics **13** (Suppl. 7) S5 (2012). http://www.biomedcentral.com/1471-2105/13/S7/S5.
3. S. Madhavan, Y. Gusev, T. G. Natarajan, et al., Front. Genetics **4**, Article 236 (2013).
4. A. T. Bari, M. Golam, R. Reaz, et al., MATCH Commun. Math. Comput. Chem. **71** (1), 241 (2014).
5. Н. О. Кадырова и Л. В. Павлова, Биофизика **59** (3), 446 (2014).
6. Н. О. Кадырова и Л. В. Павлова, Биофизика **60** (1), 18 (2015).
7. Н. О. Кадырова и Л. В. Павлова, Биофизика **60** (6), 1085 (2015).
8. Ch. Hsu, Ch. Chang, and Ch. Lin, http://www.csie.ntu.edu.tw/ (2010).
9. M. Beltrami, A. C. L. da Silva, Appl. Math. Sci. **9** (2), 75 (2015).
10. M. Momma and K. P. Bennett, in *Proc. of SIAM Conf. on Data Mining* **132**, 261 (2002).
11. Zh. Yan, Yu. Yang, and Yu. Ding, Int. J. Signal Processing, Image Processing and Pattern Recognition **6** (5), 437 (2013).
12. E. Carrizosa, B. Martín-Barragán, and D. R. Morales (2012), https://www.sbs.ox.ac.uk/sites/default/files/SBS_working_papers/Neighborhood_final.pdf.
13. E. Carrizosa, B. Martín-Barragán, and D. R. Morales, Computers & Operations Res. **43**, 328 (2014).
14. J. Xu, Metallurgical and Mining Industry **1**, 97 (2016).
15. M. Drazić, V. Kovacevic-Vujcić, M. Cangalović, and N. Mladenović, In *Global Optimization* (Springer US, 2006), pp. 135–154.
16. P. Hansen, N. Mladenović, and J. A. M. Pérez, Ann. Operat. Res. **175** (1), 367 (2010).
17. N. Mladenović, в сб. *Материалы международной конференции «Дискретная оптимизация и исследование операций»* (Новосибирск, 2013), сс. 17–23.
18. P. Koch, B. Bischl, O. Flasch, et al., Evolutionary Intelligence 5 (3), 153 (2012).
19. T. Bartz-Beielstein, J. Ziegenhirt, W. Konen, et al. (2011), http://CRAN.Rproject.org/package=SPOT.
20. H. Fröhlich and A. Zell, IEEE Neural Networks **3**, 1431 (2005).
21. A. Perolini, Int. J. Computer, Electrical, Automation, Control and Information Engineering **4** (4), 625 (2010).
22. S. Lessmann, R. Stahlbock, and S. F. Crone, IEEE Neural Networks **6**, 3063 (2006).
23. O. Chapelle, V. Vapnik, O. Bousquet, et al., Machine Learning **46** (1), 131 (2002).
24. S. S. Keerthi, V. Sindhwani, O. Chapelle, In *Advances in Neural Information Processing Systems* (2006), pp. 673–680.
25. T. Glasmachers and Ch. Igel, IEEE Trans. Pattern Analysis and Machine Intelligence **32** (8), 1522 (2010).
26. S. S. Keerthi and C. J. Lin, Neural computation **15** (7), 1667 (2003).
27. B. Bischl, O. Mersmann, and H. Trautmann, In *WEMACS 2010, held in conjunction with PPSN 2010,* 11 (2010), pp. 14–31.
28. http://archive.ics.uci.edu/ml/index.php.
29. S. M. Weiss and N. Indurkhya, *Predictive data mining: a practical guide* (Morgan Kaufmann Publ. Inc., 1998).

Analysis of Methods for Tuning Support Vector Machine for Binary Classification

N.O. Kadyrova and L.V. Pavlova

Institute of Applied Mathematics and Mechanics, Peter the Great St. Petersburg Polytechnical University, Polytechnicheskaya ul. 29, St. Petersburg, 195251 Russia

With the methods based on support vector machines it is possible to handle voluminous, high-dimensional and also poorly structured datasets, it is especially important in finding a solution for prediction in bioinformatics. In this paper, we discuss a key stage in designing support vector machines, namely, how to choose the model. Methods for tuning support vector machine have been analyzed for binary classification. Two alternative approaches, the gradient based method and the derivative-free method of stochastic search, are considered and implemented for determination of the best values of hyperparameters. The quality performance of the support-vector-classifiers obtained with the use of the mentioned methods is investigated based on benchmark data. The nested resampling technique is used to improve assessing model accuracy. The results show the effectiveness of chosen model selection methods for binary classification.

Keywords: support vector machine, model selection, variable neighborhood search, gradient-based method, nested resampling technique