УДК 519.6

АЛГОРИТМ КЛАССФИКАЦИИ ПОТОКОВЫХ СИГНАЛОВ НА ОСНОВЕ ПОСЛЕДОВАТЕЛЬНОЙ МАШИНЫ ОПОРНЫХ ВЕКТОРОВ

А. В. Ковальчук[1], *Н. С. Беллюстин*[2]

[1]Институт прикладной физики РАН, Нижний Новгород

[2]ФГБНУ Научно-исследовательский радиофизический институт, Нижний Новгород

В работе предлагается и исследуется метод классификации сигналов, который при работе с потоком данных в реальном времени осуществляет изменение параметров классификации по вновь поступающим данным – этим обеспечивается высокая эффективность классификации. Предложенный метод реализуется на модификации известного алгоритма «машины опорных векторов», базовый вариант которого для работы в реальном времени непригоден из-за высоких требований к вычислительным ресурсам. Разработанный алгоритм последовательного «дообучения» машины опорных векторов позволяет существенно уменьшить время «обучения» и количество опорных векторов. На данных по распознаванию рукописных цифр показано, что ошибка разработанного алгоритма классификации сигналов возрастает несущественно. Сформулированы условия оптимальной ориентации гиперплоскостей в многомерном пространстве признаков и оптимальной величины зазора между ними при формировании двухпорогового (тернарного) классификатора.

Ключевые слова: Машина опорных векторов, классификация потоковых сигналов, тернарный классификатор.

Введение

Вопросы оптимальной обработки сложных и многомерных сигналов обсуждаются в литературе достаточно давно, и в настоящее время они играют ключевую роль в самых различных научных направлениях. Особенно важным типом таких сигналов являются двумерные изображения, возникающие при регистрации электромагнитных волн различных частот, например, радиолокационные изображения, оптические фотографии или рентгеновские снимки. Анализ сигнала всегда должен в итоге сформировать ответ на простой по структуре вопрос: присутствует ли в сигнале (на изображении) целевой объект или нет. Потому задача сводится к созданию классификатора, который должен отнести сигнал к одному из двух возможных классов – с объектом или без него. Поскольку каждый пример сигнала можно отождествлять с точкой в многомерном пространстве признаков (или же вектором из начала координат в эту точку), то задача классификации сводится к разделению многомерного

пространства признаков на области с помощью разделяющей гиперплоскости или более сложных гиперповерхностей.

Исследования последних лет показали, что вариант разделяющей гиперплоскости оказался во многих случаях достаточно эффективным, что подтверждается практикой применения его в одном из наиболее эффективных статистических методов классификации – «машине опорных векторов» (support vector machine, SVM) [1]. Алгоритм использует для обучения массив имеющихся экспериментальных данных, а затем проводит оптимальную разделяющую гиперплоскость в многомерном пространстве таким образом, чтобы ширина зазора, в котором находятся «опорные вектора» обоих классов, оказалась минимально возможной. Алгоритмы SVM нашли применение в задачах классификации данных с пропусками (полуавтоматическая классификация с обучением). В частности, примерами успешно решенных прикладных задач могут служить системы классификации радиолокационных данных или системы поддержки принятия решений в медицине, где часть поступающих сигналов классифицируется специалистом, а остальная часть – классификатором [2].

Однако сложным вопросом для теории алгоритмов SVM до сих пор остаются вопросы последовательного обучения в динамических условиях, когда последовательно пополняются имеющиеся для обучения экспериментальные данные и требуется осуществлять дополнительное обучение модели с учетом вновь поступивших данных. Стандартный SVM-подход требует на каждом последовательном шаге заново проводить полное обучение всей системы по всем имеющимся данным, а в случае работы в режиме постоянной готовности (online) временные затраты на такое обучение могут стать недопустимо высокими.

В связи с этим для последовательных алгоритмов SVM важна разработка приближенных динамических вариантов SVM, позволяющих при небольшой потере качества работы существенно уменьшить требуемую память и время переобучения по новым данным. Эти вопросы впервые обсуждались в работах [3, 4]. Идея последовательных динамических алгоритмов SVM, использующих методы нелинейной динамики, может состоять в использовании шагов введения и выведения входных векторов во множество опорных векторов, формирующих решение. В работе [5] предложен простой и быстрый геометрический алгоритм последовательного поиска граничных векторов и формирования классифицирующей гиперплоскости. Другие методы направлены на сокращение количества опорных векторов с некоторой потерей в качестве решения [6].

В настоящей работе предлагается упрощенная схема дополнительного обучения системы, в которой используются нелинейные правила итерационного перебора, нелинейные ядровые функции и нейродинамическое обучение системы в реальном времени. Такой подход на основе методов нелинейной динамики может приближенно решить рассматриваемую оптимизационную задачу при существенном сокращении необходимых вычислительных ресурсов. На примере задачи распознавания рукописных цифр подтверждена эффективность предложенного нового варианта последовательного алгоритма SVM, реализующего непрерывное дополнительное обучение классификатора. Предлагаемая модификация алгоритма значительно сокращает время на расчеты при сопоставимом качестве распознавания, получаемом с применением прямого алгоритма ν-SVM [7]. Приведены также экспериментальные результаты их сравнения.

Предлагаемый алгоритм не только обеспечивает эффективное разделение пространства признаков на классы, но и дает важную количественную характеристику степени перекрытия разделяемых классов – долю обучающих примеров, попадающих в зазор между двумя гиперплоскостями. Этот параметр позволяет сравнивать между собой различные варианты наборов признаков по их эффективности в отношении разделения двух рассматриваемых классов.

1. Машина опорных векторов

Рассмотрим обобщенную форму тернарного классификатора и сформулируем оптимизационную задачу для поиска его параметров.

1.1. Оптимизационная задача. Пусть имеется набор объектов обучающей выборки $S = \{x_1, ..., x_N\}$ и соответствующий им набор меток $L = \{y_1, ..., y_N\}$. Здесь $x_i \in X$– прецедент пространства X, в котором задано скалярное произведение; $y \in \{-1, 1\}$метка класса. Гиперплоскость в пространстве X может быть описана как

$$\left\{x \in X \,\middle|\, H_{(w,b)}(x) = \langle w, x\rangle - b = 0\right\},$$

где $w \in X, \|w\| = 1, b \in \mathfrak{R}^d$, а пара $(w, b) \in X \times \mathfrak{R}^d$ задает гиперплоскость в пространстве X. В этом представлении вектор w ортогонален гиперплоскости и является нормалью к ней.

Предположим, что задана нормаль w и соответствующее ей множество гиперплоскостей H_w. Тогда найдется пара гиперплоскостей $H_{(w,b^+)}, H_{(w,b^-)} \in H_w$, таких что

$$\begin{aligned} &(w, b^+) : H_{w,b^+}(x_i) \geq 0, \forall i \in S^+, \\ &(w, b^-) : H_{w,b^-}(x_i) < 0, \forall i \in S^-, \end{aligned} \tag{1.1}$$

где $S^+ = \{i \in \mathrm{Z}_+ \,|\, y_i = 1\}$, $S^- = \{j \in \mathrm{Z}_+ \,|\, y_j = -1\}$ – индексы объектов разных классов. Таким образом, граница между множествами примеров разного класса может быть описана параллельными гиперплоскостями (рис. 1). Очевидно, что выбор параметра b неоднозначен и условиям (1.1) удовлетворяет бесконечное множество пар гиперплоскостей. Для решения вопроса выбора гиперплоскости потребуется понятие оптимальной разделяющей гиперплоскости и зазора [8].

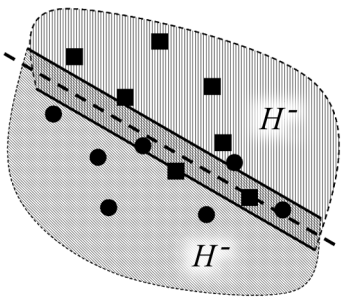


Рис. 1. Геометрическая интерпретация на плоскости классифицирующей гиперплоскости и зазора

Определение (зазор гиперплоскостей). Для гиперплоскостей (1) величина $\rho_{(n,b^+,b^-)} = b^+ - b^-$ называется *зазором гиперплоскостей.*

Можно выделить два важных для классификации объекта случая. В случае *положительной* величины зазора множества объектов, образуемые индексами S^+ и S^-, разделены в пространстве X, и существует гиперплоскость, корректно классифици-

рующая объекты. При *отрицательной* величине зазора гиперплоскости в пространстве X описывают три непересекающихся подмножества (см. рис. 1):

H^+ – содержит только примеры с меткой $(+1)$;

H^- – содержит только примеры с меткой (-1);

H^0 – может содержать примеры обоих классов, в этом случае объект $x \in H^0$ классифицируется неверно.

Существуют интуитивные и теоретически подкрепленные основания полагать, что больший зазор гиперплоскостей увеличивает обобщающую способность классификатора [9]. Согласно этим аргументам, оптимальную разделяющую гиперплоскость следует выбирать по максимальной величине зазора. Для поиска параметров гиперплоскостей с максимальным зазором может быть сформулирована следующая оптимизационная задача:

$$\max_{w,b^+,b^-} (b^+ - b^-),$$
$$s.t. \begin{cases} \langle w, x\rangle - b^+ \geq 0, & \forall i : y_i = 1, \\ \langle w, x\rangle - b^- < 0, & \forall i : y_i = -1, \\ \|w\|^2 = 1. \end{cases} \tag{1.2}$$

Учитывая, что обучающая выборка может содержать объекты с ошибочными метками классов, ошибки в измерениях значений объектов и т.д., в задачу (1.2) вводятся так называемые корректирующие коэффициенты ξ [8]. При этом требуется, чтобы количество таких ошибок было как можно меньше. Оптимизационная задача (1.2) тогда может быть представлена следующим образом:

$$\max_{w,b^+,b^-} \big(b^+ - b^- - \sum_{i=1\ldots N} c_i\xi_i\big),$$
$$s.t. \begin{cases} y_i\Big(\langle w, x\rangle - \dfrac{1+y_i}{2}b^+ + \dfrac{y_i-1}{2}b^-\Big) \geq -\eta\xi_i, \\ \xi_i \geq 0, \\ \|w\|^2 = 1. \end{cases} \tag{1.3}$$

Задача (1.3) является задачей нелинейного программирования, поскольку ограничивающие условия целевой функции имеют квадратичную функцию. Здесь c, η – параметры регуляризации, области значений которых будут рассмотрены ниже. Поиск решения задачи в прямой постановке представляет известную сложность, поэтому рассмотрим двойственную оптимизационную задачу, эквивалентную исходной. Функция Лагранжа оптимизационного функционала (1.3)

$$L(w, b^+, b^-, \xi, \alpha, \lambda, \mu) = \sum_{i=1\ldots N} c_i\xi_i - \big(b^+ - b^-\big) + \frac{\lambda}{2}\Big(\|w\|^2 - 1\Big) -$$
$$- \sum_{i=1\ldots N} \alpha_i \left(y_i\left(\langle w, x\rangle - \frac{1+y_i}{2}b^+ + \frac{y_i-1}{2}b^-\right) + \eta\xi_i\right) - \sum_{i=1\ldots N} \mu_i\xi_i, \tag{1.4}$$

где $\lambda \neq 0, \alpha_i \geq 0, \mu_i \geq 0$ – коэффициенты Лагранжа. Эту функцию необходимо минимизировать по прямым переменным (w, b^+, b^-, ξ) и максимизировать по двойственным переменным (α, λ, μ). Эти условия приводят к следующим выражениям:

$$\begin{aligned}
&\frac{\partial L}{\partial w} = \lambda w - \sum_{i=1\ldots N} \alpha_i y_i x_i = 0 \Rightarrow w = \frac{1}{\lambda} \sum_{i=1\ldots N} \alpha_i y_i x_i, \\
&\frac{\partial L}{\partial b^+} = -1 + \sum_{i=1\ldots N} \alpha_i y_i \frac{1+y_i}{2} = 0 \Rightarrow \sum_{i=1\ldots N} \alpha_i = 1, \forall i : y_i = 1, \\
&\frac{\partial L}{\partial b^-} = 1 - \sum_{i=1\ldots N} \alpha_i y_i \frac{y_i - 1}{2} = 0 \Rightarrow \sum_{i=1\ldots N} \alpha_i = 1, \forall i : y_i = -1, \\
&\frac{\partial L}{\partial \xi_i} = c_i - \eta\alpha_i - \mu_i = 0 \Rightarrow \alpha_i \leq \frac{c_i}{\eta}, \\
&\lambda^2 = \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle.
\end{aligned} \tag{1.5}$$

С учетом всех подстановок (1.5) функция Лагранжа (1.4) принимает вид $L = -\sqrt{\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle}$, и двойственная задача оптимизации формулируется как

$$\begin{aligned}
&\min \Big(W(\alpha) = \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \Big), \\
&s.t. \begin{cases} \sum\limits_{i=1\ldots N} \alpha_i = 1, \quad \forall i : y_i = 1, \\ \sum\limits_{i=1\ldots N} \alpha_i = 1, \quad \forall i : y_i = -1, \\ 0 \leq \alpha_i \leq \dfrac{c_i}{\eta}. \end{cases}
\end{aligned} \tag{1.6}$$

1.2. Решающая функция. Тернарный классификатор. В классической формулировке двухклассовой задачи классификации требуется отыскать решающую функцию $f_2 : X \to \{-1, 1\}$. В двойственной постановке функция принятия решения может быть записана как

$$f_2(x) = \operatorname{sgn}\Big(\frac{1}{\lambda} \sum_{1\ldots N} \alpha_i y_i \langle x_i, x \rangle - \frac{b^+ + b^-}{2} \Big), \tag{1.7}$$

где параметры b^+, b^- определяются из условия дополняющей нежесткости Каруша–Куна–Таккера (ККТ) [10]

$$b^+ = \max_{i \in S^+ : \alpha_i < \frac{c_i}{\eta}} (\langle w, x_i \rangle), \quad b^- = \min_{i \in S^- : \alpha_i < \frac{c_i}{\eta}} (\langle w, x_i \rangle).$$

Случай линейно разделимых данных, а также случай слабого пересечения классов достаточно редко встречается в практических задачах. Напротив, более интересен случай частичной вложенности множеств объектов разных классов в пространстве признаков X, которому соответствует отрицательная величина зазора гиперплоскостей. Поэтому в некоторых задачах более разумно использовать тернарные

функции принятия решения $f_3 : X \to \{-1, 0, 1\}$ [11]

$$f_3(x) = \begin{cases} 1, & \frac{1}{\lambda}\sum\limits_{1\ldots N} \alpha_i y_i \langle x_i, x\rangle - \max\left(b^+, b^-\right) \geq 0, \\ -1, & \frac{1}{\lambda}\sum\limits_{1\ldots N} \alpha_i y_i \langle x_i, x\rangle - \min\left(b^+, b^-\right) < 0, \\ 0, & \text{otherwise.} \end{cases} \tag{1.8}$$

Согласно (1.8), любому объекту ставится в соответствие либо метка класса $\{-1, 1\}$, либо метка 0 – отказ от принятия решения. Нетрудно заметить, что неразмеченными остаются попавшие в зазор объекты, которые считаются неизвестными. Если количество неизвестных объектов достаточно велико, для контроля качества классификатора (1.8) значения ошибки распознавания недостаточно. Поэтому вводится дополнительный параметр, называемый *ошибкой зазора*, который определяет долю объектов, попавших в зазор. Ошибка зазора формально есть: $E_m = (1/N)\,|\{i\,|f_3(x_i) = 0\}|$. Для оценки качества тернарного классификатора более удобно использовать эффективность классификатора, определяемую как $R_m = 1 - E_m$, где E_m– ошибка зазора.

1.3. Выбор параметров. Вопрос о выборе параметров в методе SVM подробно рассмотрен в работе [7], где впервые предложена формулировка задачи ν-SVM. Здесь сформулируем сначала важные следствия предлагаемого алгоритма (1.6), а затем покажем его эквивалентность известному ранее алгоритму ν-SVM (см. ниже п. 1.4).

Перед тем, как уточнить связь параметров машины опорных векторов, обобщим оптимизационную задачу (1.3) на случай построения нелинейных разделяющих поверхностей с помощью так называемого *ядрового перехода* [6,7]. Двойственная задача оптимизации (1.6) зависит от объектов обучения только в виде попарных скалярных произведений $\langle \Phi(x), \Phi(y)\rangle$. Решающие правила (1.7), (1.8) также зависят только от скалярных произведений между распознаваемым объектом и объектами обучения $\langle \Phi(x_i), \Phi(x)\rangle$. Предположим далее, что скалярное произведение в пространстве X известно как функция K в исходном пространстве $\mathfrak{R}^d$: $\langle \Phi(x), \Phi(y)\rangle = K(x, y)$. Оптимизационная задача (1.6) с учетом ядровой функции может быть записана как

$$\begin{gathered} \min\left(W(\alpha) = \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j K_{i,j}\right), \\ s.t. \begin{cases} \sum\limits_{\forall i: y_i = 1} \alpha_i = 1, \\ \sum\limits_{\forall i: y_i = -1} \alpha_i = 1, \\ 0 \leq \alpha_i \leq \dfrac{c_i}{\eta}. \end{cases} \end{gathered} \tag{1.9}$$

Для определения связи между параметрами введем понятие *ошибок обучения*.

Определение (доля ошибок обучения). Доля ошибок обучения для решающей функции $f_3(x)$ (без учета ошибки зазора) определяется как доля объектов обучения, для которых $\xi_i > 0$, $R_e = (1/N)\,|\{i\,|\xi_i > 0\}|$.

Тогда справедлива следующая теорема:

Теорема (свойства параметра η для классификации). Предположим, что параметры функции принятия решения $f_3(x)$ соответствуют решению оптимизационной задачи (1.9). Тогда

1) η является верхней оценкой на долю ошибок обучения,

2) η является нижней оценкой на долю опорных векторов

при нормировочном условии

$$\begin{cases} \sum\limits_{\forall i: y_i = 1} c_i = 1, \\ \sum\limits_{\forall i: y_i = -1} c_i = 1. \end{cases}$$

Доказательство.

1) Из условий ККТ, если $\xi_i > 0$, то $\mu_i = 0$ и $\alpha_i = c_i/\eta$. В худшем случае только доля $\eta = \sum_{D^+ \in S^+} c_i = \sum_{D^- \in S^-} c_i$ объектов обучения будет удовлетворять условиям (1.8) на вес α_i целевой функции с учетом нормировки. При этом оставшиеся примеры имеют нулевые веса α_i, выпадая из множества опорных векторов.

2) Согласно п. 1, минимальное количество опорных векторов определяется максимальной оценкой на долю ошибочных объектов, так как веса остальных векторов нулевые.

1.4. О связи с методом ν-SVM. Рассмотрим двойственную оптимизационную задачу в постановке ν-SVM

$$\min\left(W(\alpha) = \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j K_{i,j}\right),$$
$$s.t. \begin{cases} \sum\limits_{i=1}^{N} y_i \alpha_i = 0, \\ \sum\limits_{i=1}^{N} \alpha_i \geq \nu, \\ 0 \leq \alpha_i \leq c_i. \end{cases} \tag{1.10}$$

Соответствующие условия Лагранжа и условия дополняющей нежесткости ККТ для (1.10) могут быть записаны как

$$\begin{array}{ll} \alpha_i + \mu_i = c_i, \sum\limits_{i=1}^{N} y_i \alpha_i = 0, & \sum\limits_{i=1}^{N} \alpha_i - \delta = \nu, \\ \delta\rho = 0, \mu_i \zeta_i = 0, & \sum\limits_{i=1 \ldots N} \alpha_i \left(y_i \left(\langle w, x\rangle + b\right) - \rho + \xi_i\right) = 0, \end{array} \tag{1.11}$$

где δ, μ_i, α_i – коэффициенты Лагранжа. Между задачами (1.9) и (1.10) существует связь, которую можно сформулировать следующим образом.

Теорема (об эквивалентности). Если метод ν-SVM приводит к такому решению, что $\rho > 0$, тогда задачи (1.9) и (1.10) эквивалентны и их решения совпадают при условии, что параметр η равен ν.

Доказательство. Заметим, что при $\rho > 0$ из условий ККТ следует равенство $\sum_{i=1}^{N} \alpha_i = \nu$. Нетрудно убедиться, что это равенство справедливо и для (1.9) заменой $\alpha_i = \beta_i/2\eta$ и сложением ограничивающих равенств. Разность этих равенств дает эквивалентное условие $\sum_{i=1}^{N} y_i\beta_i = 0$ на веса разных классов. Задачи (1.9) и (1.10) имеют совпадающие условия, при равных параметрах $\eta = \nu$ их решения совпадают.

Параметры гиперплоскости (1.7) могут быть выражены через параметры ν-SVM: $b^+ = \rho - b$, $b^- = -\rho - b$, при соблюдении условий теоремы. Решающая функция запишется как $f_2(x) = \operatorname{sgn}\left(\sum_{1\ldots N} \alpha_i y_i K(x_i, x) + b\right)$ с величиной зазора гиперплоскости $\rho = (b^+ - b^-)/2$.

2. Метод последовательной оптимизации

Наиболее известными методами решения проблемы SVM (проблемы квадратичного программирования) является метод сопряженных градиентов [9] и метод последовательной минимальной оптимизации (sequential minimal optimization, SMO) [10]. Эти методы основаны на поиске оптимального сдвига вдоль заранее выбранного направления. Каждый сдвиг происходит относительно начального вектора α вдоль направления h так, чтобы удовлетворять ограничениям задачи. Новый вектор запишется как $\alpha^* = \alpha + \lambda^* h$, где

$$\begin{aligned} &\lambda^* = \arg\max W(\alpha + \lambda h), \\ &s.t. 0 \le \lambda \le \phi(\alpha, h). \end{aligned} \tag{2.1}$$

Верхняя граница $\phi(\alpha, h)$ определяется условиями оптимизации квадратичной задачи. Платт отметил, что расчеты величины сдвига становятся намного быстрее, если направление поиска в основном содержит нулевые коэффициенты [10]. Он предложил оставить лишь пару ненулевых коэффициентов, таких чтобы $\sum_k h_k = 0$. Алгоритм SMO использует направление поиска h, в котором отличны от нулевых лишь пара коэффициентов: $(+1)$ и (-1). Отметим, что такой выбор автоматически удовлетворяет условию ограничения на сумму весовых коэффициентов α, а ограничение на величину α можно получить выбором сдвига λ.

В рассматриваемой задаче (1.8) расчеты сдвига возможно производить сразу для четырех коэффициентов направления без существенной потери скорости. Ограничения здесь разделены для примеров разных классов и поэтому точное решение может быть найдено для совместных пар из разных классов. Новый вектор весов тогда можно записать как

$$\alpha^* = \alpha + \vartheta^* u + \varphi^* v, \tag{2.2}$$

где $u = Z_{i,j}$ и $u = Z_{t,p}$ $i, j \in S^+$ $t, p \in S^-$. Здесь $Z_{x,y}$ – разреженный вектор коэффициентов $(+1)$, (-1) с индексами x, y, соответственно. В этом случае, задача

квадратичной оптимизации запишется в следующей форме:

$$(\vartheta, \varphi) = \arg\max W\left(\alpha + \vartheta u + \varphi v\right),$$

$$s.t. \quad \begin{array}{l} 0 \le \vartheta \le \min \begin{pmatrix} \dfrac{c_i}{\eta} - \alpha_i \\ \alpha_j \end{pmatrix} = \vartheta_{\max}, \\ 0 \le \varphi \le \min \begin{pmatrix} \dfrac{c_t}{\eta} - \alpha_t \\ c\alpha_p \end{pmatrix} = \varphi_{\max}. \end{array} \tag{2.3}$$

Вычисляя экстремум целевой функции $W\left(\alpha + \vartheta u + \varphi v\right)$ по параметрам (ϑ, φ), получим

$$\begin{cases} \hat{\vartheta} = \dfrac{-\langle vg\rangle T + \langle ug\rangle V}{2\left(UV - TT\right)}, \\ \hat{\varphi} = \dfrac{-\langle ug\rangle T + \langle vg\rangle U}{2\left(UV - TT\right)}, \end{cases} \quad \begin{array}{l} U = \sum\limits_i \sum\limits_j u_i u_j y_i y_j K_{i,j}, \\ V = \sum\limits_i \sum\limits_j v_i v_j y_i y_j K_{i,j}, \\ T = \sum\limits_i \sum\limits_j u_i v_j y_i y_j K_{i,j}, \end{array} \tag{2.4}$$

где $g_k = \frac{\partial W(\alpha)}{\partial \alpha_k} = \sum\limits_i \alpha_i y_k y_i K_{k,i}$ – градиент целевой функции. Применяя разреженную форму для направлений u и v, оптимальные значения величины шага можно записать как

$$\begin{cases} \hat{\vartheta} = \dfrac{-T(g_t - g_p) + V(g_i - g_j)}{2\left(UV - TT\right)}, \\ \hat{\varphi} = \dfrac{-T(g_i - g_j) + U(g_t - g_p)}{2\left(UV - TT\right)}, \end{cases} \quad \begin{array}{l} U = K_{i,i} + K_{j,j} - 2K_{i,j}, \\ V = K_{t,t} + K_{p,p} - 2K_{t,p}, \\ T = -K_{i,t} + K_{i,p} + K_{j,t} - K_{j,p}. \end{array} \tag{2.5}$$

С учетом ограничений оптимальные значения шага достигаются при значениях

$$\vartheta^* = \min\left(\vartheta_{\max}, \hat{\vartheta}\right),$$

$$\varphi^* = \min\left(\varphi_{\max}, \hat{\varphi}\right).$$

На практике алгоритм SMO обычно используют с небольшим допуском $\tau > 0$ [12,13]. Выбираются только такие направления поиска, чтобы $\langle h, g\rangle \ge \tau$ и $\vartheta_{\max} > 0$, $\varphi_{\max} > 0$. При таком выборе появляется возможность увеличения целевой функции (2.4) вдоль направления h. Поэтому для определения направления поиска будем выбирать такие пары, что

$$\begin{array}{ll} i, j \in S^+, & u = Z_{i,j}, \\ t, p \in S^-, & v = Z_{t,p}, \end{array} \Leftrightarrow \begin{cases} \alpha_i < \dfrac{c_i}{\eta}, \quad \alpha_j > 0, \\ \alpha_t < \dfrac{c_t}{\eta}, \quad \alpha_p > 0, \\ g_i + g_t - g_j - g_p \ge \tau. \end{cases} \tag{2.6}$$

Среди возможных квартетов индексов объектов (i, j, t, p), удовлетворяющих условиям (2.6), наиболее эффективен квартет с максимальным направленным градиентом $\langle h, g \rangle$. Этот выбор описан в контексте понятия оптимальной гиперплоскости в работах [8,9].

2.1. Последовательный алгоритм машины опорных векторов, работающий в режиме online (OnSVM). Предлагаемый алгоритм OnSVM основан на идее последовательной оптимизации параметров задачи (1.9) для каждого, вновь добавленного, объекта x_k. Можно выделить три следующих основных шага алгоритма:

- Insert: добавление объекта во множество опорных векторов $S' = \{S, x_k\}$;
- Solve: поиск τ-четверки и формирование для неё оптимального решения;
- Remove: удаление объекта из множества S при достижении условий исключения.

2.1.1. Шаг Insert. В отличие от постановки задачи классической SVM, потоковое добавление объектов во множество S приводит к изменению ограничивающих условий на коэффициент α (1.9). В самом деле, поскольку нормировочный коэффициент c зависит от общего числа объектов, то добавление нового объекта снижает верхнюю границу $\alpha \le \alpha_{bound} = c/\eta$. Первый шаг Insert состоит из операций ограничения параметров α, присвоении новому объекту начального значения и пересчета градиентов g.

Шаг 1. Добавление объекта во множество опорных векторов

Insert
if $y_k = +1$ then
$N^+ \leftarrow N^+ + 1;\ \alpha^+_{bound} \leftarrow \frac{1}{\eta N^+};\ \alpha_s \leftarrow \min\left(\alpha_s, \alpha^+_{bound}\right);\ \alpha_k \leftarrow 1 - \sum_{i \in S^+} \alpha_i;\ S^+ \leftarrow S^+ \cup \{k\}$
else
$N^- \leftarrow N^- + 1,\ \alpha^-_{bound} \leftarrow \frac{1}{\eta N^-};\ \alpha_s \leftarrow \min\left(\alpha_s, \alpha^-_{bound}\right),\ \alpha_k \leftarrow 1 - \sum_{i \in S^-} \alpha_i;\ S^- \leftarrow S^- \cup \{k\}$
endif
$g_s \leftarrow -\sum_{i \in S} y_i y_s \alpha_i K_{i,s},\ \forall s \in S$

Здесь: N^+, N^- – количество объектов с положительными и отрицательными метками, соответственно; α^+_{bound}, α^-_{bound} – верхние границы. Начальное значение α выбрано таким, чтобы удовлетворять ограничительным равенствам. Этот выбор может не соответствовать оптимальному решению. Сдвиг верхней границы также может повлиять на отклонение от оптимума. Поэтому следующий шаг – это поиск оптимального совместного решения для измененного множества опорных векторов S.

2.1.2. Шаг Solve. На шаге Solve формируется τ-четверка опорных векторов из S с максимальным направленным градиентом. В случае, если τ-четверка не может быть найдена, этот шаг должен быть пропущен. Оптимальное решение для τ-четверки можно получить, используя соотношение (2.6). На этом этапе также определяются необходимые параметры смещения гиперплоскостей b.

Для τ-четверки отбираются те вектора i, которые не достигли ограничений на α_i в соответствующих направлениях поиска решения.

Шаг 2. Поиск совместного решения для τ-четверки

Solve
$i \leftarrow \underset{s\in S^+,\alpha_i<\alpha^+_{bound}}{\arg\max}\ (g_s);\ j \leftarrow \underset{s\in S^+,\alpha_j>0}{\arg\min}\ (g_s);\ t \leftarrow \underset{s\in S^-,\alpha_t<\alpha^-_{bound}}{\arg\max}\ (g_s);\quad p \leftarrow \underset{s\in S^-,\alpha_p>0}{\arg\min}\ (g_s);$
$\delta \leftarrow g_i - g_j + g_t - g_p$

if $\delta > \tau$ then
$\vartheta \leftarrow \min\left(\hat{\vartheta}, \alpha^+_{bound} - \alpha_i, \alpha_j\right);\ \varphi \leftarrow \min\left(\hat{\varphi}, \alpha^-_{bound} - \alpha_t, \alpha_p\right);\ \left\{\hat{\vartheta}, \hat{\varphi}\right\}$-as defined in (2.5, 2.6)
$\alpha_i \leftarrow \alpha_i + \vartheta,\ \alpha_j \leftarrow \alpha_j - \vartheta;\ \alpha_t \leftarrow \alpha_t + \varphi,\ \alpha_p \leftarrow \alpha_p - \varphi;$
$b^+ \leftarrow -\underset{s\in S^+,\alpha_s<\alpha^+_{bound}}{\max}\ (g_s);\ b^- \leftarrow \underset{s\in S^-,\alpha_s<\alpha^-_{bound}}{\max}\ (g_s);$

endif

2.1.3. Шаг Remove. Заключительный шаг Remove необходим для освобождения множества S опорных векторов от векторов, не участвующих в формировании параметров разделяющей гиперплоскости j: $\alpha_j = 0$, $g_j < 0$.

Шаг 3. Исключение опорного вектора

Remove
$i \leftarrow \underset{s\in S^+,\ \alpha_s=0}{\arg\min}\ (g_s);\quad \text{if } g_i < 0 \text{ then } S^+ = S^+ - \{i\}$
$j \leftarrow \underset{s\in S^-,\ \alpha_s=0}{\arg\min}\ (g_s);\quad \text{if } g_j < 0 \text{ then } S^- = S^- - \{j\}$

Шаг Remove исключает из каждого подмножества S^+, S^- не более одного опорного вектора, удовлетворяющего условиям исключения. Решение принимается по минимальному значению градиента g.

2.1.4. Алгоритм OnSVM. Алгоритм обучения OnSVM состоит из трех следующих процедур:

- Initialization – сбор минимального числа необходимых опорных векторов;
- Process – потоковая обработка примеров объектов, включая поиск оптимального решения и удаления опорных векторов;
- Finalize – поиск окончательного решения.

Алоритм OnSVM

Initialize	Process	Finalize
Repeat at least 4 times: Fetch example k Call Insert	Repeat a predefined number of times: Fetch an example k Run Insert Run Solve Run Remove	Repeat while $\delta > \tau$: Run Solve Run Remove

Алгоритм OnSVM может быть использован как алгоритм последовательного обучения. В этом случае модель границы готова к распознаванию объектов сразу после процедуры Initialize. Обучение последующих примеров проводится процеду-

рой Process. Процедура Finalize введена для окончательной оптимизации опорных векторов, нарушающих τ-условие.

На практике последовательные алгоритмы обучаются на основе эпох. Каждая эпоха состоит из случайно отобранных примеров объектов. После окончания заданного количества итераций по эпохам выполняется процедура Finalize. Эмпирические данные свидетельствуют, что обученный на одной эпохе классификатор почти не отличается от решения SVM.

Рассмотренный алгоритм основан на простом и эффективном методе SMO. Вопрос сходимости этого метода в его обобщенной формулировке – обобщенный метод последовательной минимальной оптимизации (generalized sequential minimal optimization, GSMO) – подробно рассмотрен в работе [14]. Легко убедиться, что алгоритм OnSVM является алгоритмом GSMO и удовлетворяет условиям *теоремы о сходимости*. Согласно выводам, сделанным в работе, для заданного положительного $\tau > 0$ алгоритм OnSVM в представленной формулировке сходится за конечное число шагов k.

2.2. Увеличение производительности. Существенный недостаток алгоритма OnSVM заключается в необходимости пересчета градиентов g при добавлении объекта. Пересчет градиентов существенно увеличивает количество вычислений алгоритма обучения, замедляя работу шага Insert. Причина состоит в зависимости верхней границы α_{bound} от количества обучаемых векторов.

Заметим, что сдвиг верхней границы влияет на те опорные вектора, величины весов α которых достигли границы, а также весов, достаточно близких к границе. Таким образом, опорные вектора из множества S можно разделить на 3 подмножества:

- подмножество S_0 – векторы с немодифицируемыми весами;
- подмножество S_1 – векторы с граничными весами;
- подмножество S_2 – векторы с весами не дальше от границы α'_{bound} чем на величину сдвига $\Delta\alpha = \alpha'_{bound} - \alpha''_{bound}$ и оставшиеся векторы S_0.

Здесь α'_{bound} и α''_{bound} – верхние границы до добавления вектора и после, соответственно. Очевидно, что граничные опорные векторы имеют одинаковую величину сдвига, чтобы удовлетворить граничным условиям после добавления вектора. Это свойство можно использовать для уменьшения количества вычислений ядровой функции.

Введем дополнительную переменную $\Delta g_i = \sum\limits_{j \in S_1} y_i y_j K_{i,j}$ как сумму ядровых функций по отношению к опорному вектору с индексом i. Тогда градиент этого вектора запишется как

$$g''_i = g'_i + \Delta\alpha\Delta g + \sum_{j \in S_2} \left(\alpha'_j - \alpha''_{bound}\right) y_i y_j K_{i,j}, \forall i \in S. \tag{2.7}$$

Вычисление ядровых функций здесь необходимо выполнить лишь для опорных векторов из множества S_2, а значения Δg_i могут быть вычислены заранее. Заметим также, что подмножество S_2 – переходное: $S_0 \to S_2 \to S_1$. Такой переход требует изменения переменной Δg_i

$$\Delta g''_i = \Delta g'_i + y_i y_j K_{i,j}, \quad \forall i \in S_1, \quad j \in S_2. \tag{2.8}$$

Кроме прямого перехода опорного вектора в подмножество S_1, возможен и обратный переход. Такая ситуация возникает во время поиска оптимального совместного решения для τ-четверки на шаге Solve. При обратном переходе, $S_1 \to S_0$, имеем

$$\Delta g_i'' = \Delta g_i' - y_i y_j K_{i,j}, \quad \forall i \in S_1, \quad j \notin S_1. \tag{2.9}$$

На шаге Solve также изменяются веса α в случае существования τ-четверки. Для сохранения действительных значений градиента, на шаге Solve вводим дополнительный пересчет градиента

$$g_s'' = g_s' - \vartheta y_i y_s K_{i,s} + \vartheta y_j y_s K_{j,s} - \varphi y_p y_s K_{p,s} + \varphi y_t y_s K_{t,s}, \forall s \in S,\ \{i,j,p,t\} \in \tau - quad. \tag{2.10}$$

Предлагаемые изменения существенно уменьшают количество вычислений ядровых функций. Так, для алгоритма OnSVM оценка на количество вызовов ядровой функции составляет $C(K) \sim N^2$. С учетом ускоряющих изменений количество вызовов может быть уменьшено до линейной зависимости и в общем случае составляет $C^{OnSVM}(K) \sim N + o(N^2) \ll N^2$. Здесь N – количество опорных векторов в множестве S.

3. Вычислительный эксперимент

Апробация предложенного алгоритма проводилась на данных MINIST ручного начертания цифр, доступных на сайте http://ftp.ics.uci.edu/pub/machine-learning-databases/. По этим данным была сформирована задача классификации начертания каждой цифры от всех остальных цифр в наборе 0...9. Данные разделялись на тренировочную и тестовую группы в соотношении 2 : 1. В экспериментах использовались данные Optdigits[1] и Pendigits[2], заметно различающиеся по сложности разделения данных на классы. На этих данных предложенный здесь алгоритм OnSVM сравнивался с известной ранее реализацией ν-SVM из открытой библиотеки машинного обучения LIBSVM 3.10 при ядровой функции гауссова типа $K_{RBF}(x,y) = \exp\left(-\gamma \|x-y\|^2\right)$. Значения использованных параметров: $\eta = 0.01$, $\gamma = 0.0005$, $\tau = 0.0001$.

В табл. 1 приведены результаты сравнения алгоритмов OnSVM и ν-SVM на базе Optdigits; в табл. 2 – на данных Pendigits. Здесь: «Num» – класс распознавания; «nSV» – количество опорных векторов; «nSV Max» – пиковое количество опорных векторов на эпохе обучения; E_2 – ошибка классификации на тестовых примерах в режиме бинарного классификатора (1.7); E_3 – ошибка тернарного классификатора (1.8); R_3 – эффективность тернарного классификатора.

В табл. 1 и 2 приведены также результаты для ошибок распознавания в режиме тернарного классификатора (1.8). Возрастание обобщающей способности алгоритма в этом режиме работы классификатора может значительно повышаться за счет ухудшения эффективности классификатора – уменьшения количества примеров, по которым классификатор принимает решение о присвоении метки класса.

[1] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/optdigits/

[2] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/pendigits/

Таблица 1

Сравнительные результаты работы алгоритма
для набора Optdigits

№№	OnSVM					LIBSVM	
	nSV	nSV Max	E_2,%	E_3, %	R_3, %	nSV	E_2,%
0	58	60	0.0	0.00	96.17	80	0.05
1	104	109	0.89	0.05	93.88	149	0.67
2	99	100	0.11	0.00	96.00	126	0.17
3	119	126	0.72	0.00	93.9	158	0.67
4	100	101	0.33	0.00	95.33	133	0.17
5	107	121	0.45	0.11	94.6	139	0.45
6	63	66	0.22	0.00	95.66	84	0.17
7	83	88	0.72	0.00	95.83	105	0.72
8	164	171	0.95	0.00	90.49	203	0.95
9	175	183	1.28	0.06	93.77	230	0.95
Total	107.2	112.5	0.57	0.02	94.6	140.7	0.5

Таблица 2

Сравнительные результаты работы алгоритма
для набора Pendigits

№№	OnSVM					LIBSVM	
	nSV	nSV Max	E_2, %	E_3, %	R_3, %	nSV	E_2, %
0	1036	1046	0.60	0.00	74.80	1267	0.60
1	963	991	0.40	0.00	71.40	1267	0.34
2	1168	1285	0.11	0.00	79.54	1143	0.11
3	827	959	0.28	0.03	86.00	872	0.28
4	957	979	0.34	0.00	75.85	1218	0.37
5	884	906	0.48	0.00	74.70	1157	0.43
6	858	880	0.17	0.00	75.73	1186	0.14
7	934	966	1.03	0.00	73.10	1186	1.03
8	1117	1126	0.11	0.00	73.79	1354	0.11
9	1032	1046	0.49	0.08	72.73	1291	0.51
Total	978.1	1018.4	0.4	0.01	75.76	1149.1	0.39

Параметр «nSV Max» позволяет оценить максимальный объем памяти, необходимый для кеширования ядровой матрицы. Кеширование ядровых функций значительно ускоряет процесс обучения, но размер памяти, требующийся для кеша, пропорционален квадрату количества обучаемых векторов. В алгоритме OnSVM необходимый размер памяти пропорционален параметру «nSV Max». В проведенных экспериментах этот параметр не отличался от общего количества опорных векторов более чем на 5%.

На рис. 2 и 3 показаны результаты усреднения измерений параметров десяти классификаторов на наборе данных Pendigits для каждой из цифр. Рис. 2 показы-

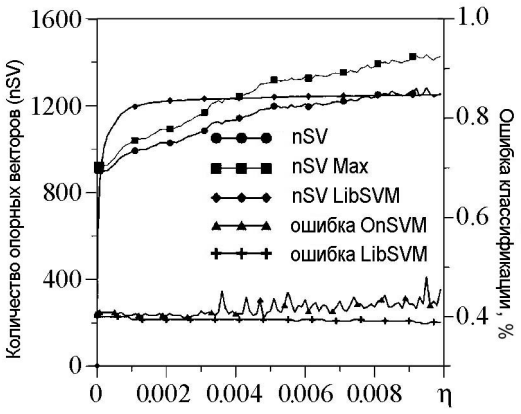


Рис. 2. Количество поддерживающих векторов и ошибки систем OnSVM и LIBSVM в зависимости от параметра η

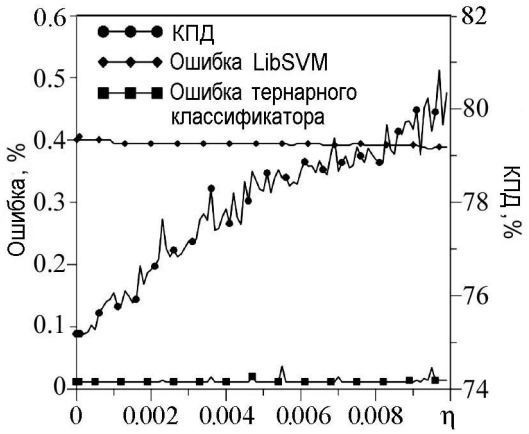


Рис. 3. Зависимость эффективности и уровня ошибок от параметра η

вает зависимость количества опорных векторов от параметра η в сравнении с реализацией LIBSVM и зависимость ошибки E_2 от η рассматриваемых алгоритмов. Рис. 3 показывает зависимости ошибки E_3 и эффективности R_3 от параметра алгоритма η.

Табл. 3 – сводная таблица сравнения рассматриваемых методов на базах данных Optdigits и Pendigits, а также на искусственно синтезированной базе данных Waveform, которая была получена в результате 100 экспериментов, в каждом из которых использовались обучающая и тестовая базы из 4000 и 1000 примеров, соответственно, при значениях параметров алгоритма $\eta = 0.001$, $\gamma = 0.005$, $\tau = 0.001$. Здесь же приведено количество примеров для каждой из трех рассмотренных групп.

Таблица 3

Сводная сравнительная таблица

	OnSVM		LIBSVM		Datasize	
Dataset	nSV	E_2(%)	nSV	E_2(%)	Train	Test
Pendigits	978.1	0.4	1149.1	0.39	7494	3823
Optdigits	107.2	0.57	140.7	0.5	3823	1797
Waveform[3]	377	12.5	444	12.9	4000	1000

Заключение

Проведенное исследование позволяет сделать следующие выводы.

- Вычислительные эксперименты подтвердили, что решение предлагаемого алгоритма OnSVM сходится к решению ν-SVM и позволяет достичь приемлемого качества решения за один проход по обучающим примерам.

[3] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/waveform/

- Количество опорных векторов у алгоритма OnSVM в среднем меньше, чем у точного решения SVM при эквивалентном качестве и при сокращении времени расчетов в десятки раз. В некоторых случаях удается получить значительно более компактные решения.
- Результаты работы бинарных классификаторов для алгоритмов OnSVM и LIBSVM достаточно близки, однако, применение тернарного классификатора в алгоритме OnSVM позволяет существенно уменьшить ошибки распознавания за счет некоторого падения эффективности.
- Поскольку при кешировании значений ядровой матрицы рост требуемой памяти пропорционален квадрату числа обучающих векторов, то предложенный последовательный алгоритм нуждается в меньшем количестве памяти при тех же временных затратах.

Перечисленные результаты позволяют использовать алгоритм в задачах классификации потоковых данных с пропусками в режиме попеременных классификации и дополнительного обучения при обязательной первичной инициализации алгоритма минимальным набором примеров. Алгоритм может быть использован в системах, для которых требуется возможность дополнительного обучения по вновь поступающим данным.

Работа выполнена при поддержке Программы Президиума РАН «Нелинейная динамика в математических и физических науках», грант РНФ № 14-11-00693.

Библиографический список

1. *Cristianini N., J. Shawe-Taylor J.* An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, 2000.
2. *Макаренко А.В., Правдивцев А.В., Воловик М.Г.* К вопросу о моделировании и анализе ИК-термокарт головного мозга человека // Изв. вуз. ПНД. 2011. Т. 19, № 6. С. 145.
3. *Cauwenberghs G., Poggio T.* Incremental and Decremental Support Vector Machine Learning // Advances in Neural Information Processing Systems. 2000. P. 409.
4. *Bordes A., Ertekin S., Weston J., Bottou L.* Fast kernel classifiers with online and active learning // Journal of Machine Learning Research. 2005. Vol. 6. P. 1579.
5. *Roobaert D.* DirectSVM: A Simple Support Vector Machine Perceptron // VLSI Signal Processing. 2002. P. 147.
6. *Orabona F., Castellini C., Caputo B., Jie L., Sandini G.* On-line independent support vector machines // Pattern Recognition. 2010. Vol. 43. P. 1402.
7. *Schoelkopf B., Smola A.* New Support Vector Algorithms // NeuroCOLT Technical Report NC2-TR-1998-031. 1998.
8. Алгоритмы и программы восстановления зависимостей / Под ред. В.Н. Вапника. М.: ФИЗМАТЛИТ, 1984.
9. *Vapnik V.* Estimation of Dependences Based on Empirical Data. Berlin: Springer-Verlag, 1982.

10. *Platt J.* Fast training of support vector machines using sequential minimal optimization / Eds B. Scholkopf, C.J.C. Burges, A.J. Smola. Advances in Kernel Methods – Support Vector Learning. Cambridge, MA: MIT Press, 1999.
11. *Растригин Л.А., Эренштейн Р.Х.* Метод коллективного распознавания. Энергоиздат, 1981. 80 с.
12. *Chang C.-C., Lin C.-J.* LIBSVM: A Library for Support Vector Machines. Technical report // Computer Science and Information Engineering. National Taiwan University, 2001–2004. http://www.csie.ntu.edu.tw/_cjlin/libsvm.
13. *Collobert R., Bengio S.* SVMTorch: Support vector machines for large-scale regression problems // Journal of Machine Learning Research. 2001. Vol. 1. P. 143.
14. *Keerthi S.S., Gilbert E.G.* Convergence of a generalized SMO algorithm for SVM classifier design // Machine Learning. 2002. Vol. 46. P. 351.

CLASSIFICATION ALGORITHM OF STREAMING SIGNALS BASED ON THE ONLINE SUPPORT VECTOR MACHINE

A. V. Kovalchuk[1], *N. S. Bellyustin*[2]

[1]Institute of Applied Physics RAS, Nizhny Novgorod

[2]FSBSI Radiophysical Scientific-Research Institute, Nizhny Novgorod

The work proposed a modification of support vector machines (SVM) to train and classify in real time (online) streams of data. The algorithm is tested on the data handwriting figures and shown that its error is comparable to SVM direct solution error. Speed and support vectors number of proposed SVM algorithm is smaller than in other known SVM implementations. Finally, a ternary classificator for 2-class problem is proposed which shows better results than binary.

Keywords: Support vector machine, streaming signal classification, online learning, ternary classifier.

References

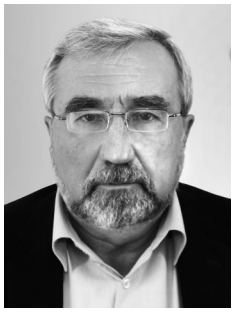
1. *Cristianini N., J. Shawe-Taylor J.* An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, 2000.
2. *Makarenko A.V., Pravdivtsev A.V., Volovik M.G.* Problems of modeling and analysis of infrared thermo maps human brain // Izvestiya VUZ. Applied Nonlinear Dynamics. 2011. Vol. 19, № 6. P. 145. (in Russian).
3. *Cauwenberghs G., Poggio T.* Incremental and Decremental Support Vector Machine Learning // Advances in Neural Information Processing Systems, 2000. P. 409.
4. *Bordes A., Ertekin S., Weston J., Bottou L.* Fast kernel classifiers with online and active learning // Journal of Machine Learning Research. 2005. Vol. 6. P. 1579.
5. *Roobaert D.* DirectSVM: A Simple Support Vector Machine Perceptron. VLSI Signal Processing, 2002. P. 147.

6. *Orabona F., Castellini C., Caputo B., Jie L., Sandini G.* On-line independent support vector machines // Pattern Recognition. 2010. Vol. 43. P. 1402.
7. *Schoelkopf B., Smola A.* New Support Vector Algorithms // NeuroCOLT Technical Report NC2-TR-1998-031. 1998.
8. Algorithms and programs for dependencies reconstraction / Ed. V.N. Vapnik. M.: Physmatlit, 1984 (in Russian).
9. *Vapnik V.* Estimation of Dependences Based on Empirical Data. Berlin: Springer-Verlag, 1982.
10. *Platt J.* Fast training of support vector machines using sequential minimal optimization / Eds B. Scholkopf, C.J.C. Burges, A.J. Smola. Advances in Kernel Methods – Support Vector Learning. Cambridge, MA: MIT Press, 1999.
11. *Rastrigin L.A., Erenstein R.H.* The method of collective recognition. Energoizdat, 1981. 80 p. (in Russian).
12. *Chang C.-C., Lin C.-J.* LIBSVM: A Library for Support Vector Machines. Technical report // Computer Science and Information Engineering. National Taiwan University, 2001–2004. http://www.csie.ntu.edu.tw/_cjlin/libsvm.
13. *Collobert R., Bengio S.* SVMTorch: Support vector machines for large-scale regression problems // Journal of Machine Learning Research. 2001. Vol. 1. P. 143.
14. *Keerthi S.S., Gilbert E.G.* Convergence of a generalized SMO algorithm for SVM classifier design // Machine Learning. 2002. Vol. 46. P. 351.

Поступила в редакцию 19.11.2015

Ковальчук Андрей Викторович – родился в Нижнем Новгороде (1982). Окончил Нижегородский государственный университет им. Лобачевского (2006). После окончания ННГУ работает в ИПФ РАН. Область научных интересов: динамика в нейронных сетях, машинное обучение, компьютерное зрение, оптоакустика. Опубликовал 30 научных работ по направлениям, указанным выше.

603950 Нижний Новгород, ГСП-120, ул. Ульянова, д. 46
Институт прикладной физики РАН
E-mail: aka.xzib1t@gmail.com



Беллюстин Николай Сергеевич – родился в Горьком (1950). Окончил с отличием Горьковский государственный университет (1972). После окончания работает в Научно-исследовательском радиофизическом институте на инженерных и научных должностях, в настоящее время – ведущий научный сотрудник. Защитил докторскую диссертация по радиофизике (2002). В область научных интересов входят исследования волн в плазме и других средах, искусственные нейронные сети, компьютерное зрение. Автор 150 научных работ.

603950 Нижний Новгород, ул. Большая Печерская, д. 25/12а
ФГБНУ Научно-исследовательский радиофизический институт
E-mail: bellyustin@mail.ru