*БИОФИЗИКА, 2015, том 60, вып. 6, с. 1085–1098*

**МОЛЕКУЛЯРНАЯ БИОФИЗИКА**

*УДК 577.3*

# СРАВНИТЕЛЬНАЯ ЭФФЕКТИВНОСТЬ АЛГОРИТМОВ ПОСТРОЕНИЯ МАШИН ОПОРНЫХ ВЕКТОРОВ ДЛЯ ЗАДАЧИ ВОССТАНОВЛЕНИЯ РЕГРЕССИИ

**© 2015 г. Н.О. Кадырова, Л.В. Павлова**

*Санкт-Петербургский политехнический университет, Институт прикладной математики и механики, 195251, Санкт-Петербург, ул. Политехническая, 29*

*E-mail: office@spbstu.ru*

Поступила в редакцию 06.07.15 г.

Методы построения машин опорных векторов не требуют дополнительной априорной информации и позволяют обрабатывать крупные объемы данных большой размерности, что особенно важно для многих проблем вычислительной биологии. Представлен обзор основных алгоритмов построения машин опорных векторов для задачи восстановления регрессии и проведено исследование их сравнительной эффективности. Критический анализ результатов этого исследования дал возможность выделить среди этих алгоритмов наиболее эффективные. Приведено описание рекомендуемых алгоритмов, достаточное для их практической реализации.

*Ключевые слова: восстановление регрессии, машины опорных векторов, SVR-алгоритмы, сравнительная эффективность SVR-алгоритмов.*

Основные принципы и идеи современного подхода к задачам бинарной классификации и восстановления регрессии, основанного на машинах опорных векторов (Support Vector Machines – SVM), рассмотрены нами в работе [1]. Основные алгоритмы построения машин опорных векторов для задачи бинарной классификации и исследование их сравнительной эффективности представлены в работе [2]. SVM-методы относятся к технологии, названной интеллектуальным анализом данных, и замечательны тем, что практически не требуют дополнительной априорной информации, не чувствительны к выбросам и шуму и позволяют обрабатывать крупные объемы данных большой размерности.

Новый индукционный принцип для обучения по конечным выборкам, структурная минимизация риска, приводит к задаче минимизации регуляризованного риска. Ряд алгоритмов построения машин опорных векторов для задачи восстановления регрессии (SVM for Regression – SVR) непосредственно решает эту задачу, используя ядерные машины. Например, алгоритм Naive Online Risk Minimization Algorithm (NORMA) [3], подробно рассмотренный нами для построения SV-классификаторов [2].

Стандартные постановки задачи бинарной классификации и задачи восстановления регрессии с ε-нечувствительной функцией потерь позволяют свести исходную задачу минимизации верхней границы ожидаемого риска к задаче квадратичного программирования с ограничениями типа равенств относительно двойственных переменных, полностью определяющих искомое решающее правило или восстанавливаемую зависимость соответственно. Непосредственное решение двойственной задачи квадратичного программирования обычно невозможно из-за ее громадной размерности. Основные подходы к построению непрямого решения этой задачи опираются на свойство разреженности, присущее SV-машинам. Ряд SVR-алгоритмов основан на идее декомпозиции, т.е. разбиении исходной задачи квадратичного программирования на серию подзадач существенно меньшей размерности. Например, к ним относятся алгоритмы SVM_Torch [4] и Sequential minimal optimization (SMO) [5].

Стандартная постановка задачи восстановления регрессии с квадратичной функцией потерь приводит к SV-машинам, не обладающим свойством разреженности, что с практической точки зрения неприемлемо. Для получения разреженной аппроксимации полного ядерного

Сокращения: SVM – support vector machines, SVR – support vector machines for regression, SMO – sequantional minimal optimization, ККТ – условия Каруша–Куна–Таккера, SOR – successive overrelaxation, IU_R – incremental updating, GSLS – greedy spare least-squares, KRLS – kernel recursive least-squares, ALD – approximate linear dependence, MAE – mean absolute error, RMSE – root mean square error.

разложения применяют алгоритмы последовательного наращивания множества опорных векторов, такие как Greedy Sparse Least Squares (GSLS) [6,7] и Kernel Recursive Least Squares (KRLS) [8].

В настоящей работе рассмотрены основные алгоритмы построения машин опорных векторов для задачи восстановления регрессии и исследована их сравнительная эффективность.

## ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ ВОССТАНОВЛЕНИЯ РЕГРЕССИИ

Рассмотрим двойственную задачу SVM-обучения для восстановления регрессии с ε-нечувствительной функцией потерь [1]:

найти

$$\max_{\alpha^{(*)}} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)k(\boldsymbol{x}_i,\boldsymbol{x}_j) - \varepsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{l}y_i(\alpha_i-\alpha_i^*) \quad (1)$$

при ограничениях

$$\begin{cases}\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0,\\ \alpha_i,\alpha_i^*\in[0,C]\,\forall i.\end{cases}$$

Здесь $(\boldsymbol{x}_1,y_1)$, … , $(\boldsymbol{x}_l, y_l) \in R^n \times R$ – тренировочная последовательность; ε и $C$ – известные параметры, такие что ошибка, меньшая ε, не штрафуется, а положительный параметр $C$ определяет компромисс между допустимой величиной ошибки (превышающей ε) и сложностью решающей функции. Ядерная функция, соответствующая скалярному произведению в пространстве признаков, – $k(\boldsymbol{x},\boldsymbol{x}')$. Искомый вектор двойственных переменных $\boldsymbol{\alpha}$ состоит из двух подвекторов $(\alpha_1,\alpha_2,\ldots,\alpha_l)$ и $(\alpha_1^*,\alpha_2^*,\ldots,\alpha_l^*)$, где $l$ – объем тренировочной последовательности. Здесь и далее $(\alpha_i^{(*)})$ обозначает $\alpha_i$ и (или) $\alpha_i^*$ в зависимости от контекста.

Оптимальная решающая функция для задачи восстановления регрессии в этом случае имеет вид [1]:

$$f(\boldsymbol{x})=\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)k(\boldsymbol{x}_i,\boldsymbol{x})+b. \quad (2)$$

## АЛГОРИТМ SEQUENTIAL MINIMAL OPTIMIZATION

Алгоритм Sequential Minimal Optimization (SMO) – SVM-алгоритм, впервые предложенный для задач классификации в работах [9,10], использует крайнюю форму декомпозиции задачи (1): последовательность подзадач квадратичного программирования размерности два. Вычислительные проблемы, которые могут возникать при реализации SMO, полностью связаны с матрицей скалярных произведений $\boldsymbol{K} = \{k(\boldsymbol{x}_i, \boldsymbol{x}_j)\}$. В работах [5,11] SMO-алгоритм Platt'a [9] обобщен на случай SVR. Заметим, что такое обобщение применимо только к случаю ε-нечувствительной функции потерь, так как для большинства других выпуклых функций потерь не существует явного решения подзадачи квадратичного программирования размерности два.

SMO-алгоритм представляет собой итеративный процесс, каждый шаг которого состоит из двух этапов: выбор рабочего множества объема два $\{\alpha_i^{(*)},\alpha_j^{(*)}\}$ (эвристический метод); решение задачи квадратичного программирования для выбранного рабочего множества (аналитический метод). Когда не остается двух подходящих для рабочего множества двойственных переменных, исходная задача оптимизации (1) оказывается решенной.

**Условия оптимальности и условия остановки алгоритма.** Для получения удобных условий остановки SMO-алгоритма введем величины:

$$F_i=y_i-\sum_{j=1}^{l}(\alpha_j-\alpha_j^*)k(\boldsymbol{x}_i,\boldsymbol{x}_j),\ \ i=1,\ldots,l.$$

Выпишем Лагранжиан и условия оптимальности Каруша–Куна–Таккера (ККТ) для двойственной задачи (1) и упростим эти условия согласно возможным сочетаниям значений переменных $\alpha_i$ и $\alpha_i^*$ для всех $i$. Поскольку при выполнении условий оптимальности для любого $i$ величины $\alpha_i$ и $\alpha_i^*$ не могут быть отличными от нуля одновременно, возможны всего пять случаев:

$$\begin{aligned}&1.\ \alpha_i=\alpha_i^*=0 \ \leftrightarrow\ -\varepsilon\le(F_i-\beta)\le\varepsilon;\\ &2.\ 0<\alpha_i<C \ \leftrightarrow\ F_i-\beta=\varepsilon;\\ &3.\ 0<\alpha_i^*<C \ \leftrightarrow\ F_i-\beta=-\varepsilon;\\ &4.\ \alpha_i=C \ \leftrightarrow\ F_i-\beta\ge\varepsilon;\\ &5.\ \alpha_i^*=C \ \leftrightarrow\ F_i-\beta\le-\varepsilon.\end{aligned} \quad (3)$$

Здесь через β обозначен множитель Лагранжа при ограничении типа равенств двойственной задачи (1).

Определим следующие множества индексов при фиксированном значении вектора $\boldsymbol{\alpha}$:

$$I_{0a}=\{i:0<\alpha_i<C\};\ I_{0b}=\{i:0<\alpha_i^*<C\};$$
$$I_1=\{i:\alpha_i=0,\alpha_i^*=0\};\ I_2=\{i:\alpha_i=0,\alpha_i^*=C\};$$
$$I_3=\{i:\alpha_i=C,\alpha_i^*=0\};\ I_0=I_{0a}\cup I_{0b}.$$

Введем величины:

$$\tilde{F}_i=\begin{cases}F_i+\varepsilon,\ i\in I_{0b}\cup I_2,\\ F_i-\varepsilon,\ i\in I_{0a}\cup I_1,\end{cases}$$

$$\bar{F}_i=\begin{cases}F_i-\varepsilon,\ i\in I_{0a}\cup I_3,\\ F_i+\varepsilon,\ i\in I_{0b}\cup I_1.\end{cases}$$

Тогда условия (3) можно переписать в виде:

$$\beta\ge\tilde{F}_i\ \ \forall i\in I_0\cup I_1\cup I_2;$$
$$\beta\le\bar{F}_i\ \ \forall i\in I_0\cup I_1\cup I_3.$$

Определим

$$b_{\text{up}}=\min\{\bar{F}_i: i\in I_0\cup I_1\cup I_3\},$$
$$b_{\text{low}}=\max\{\tilde{F}_i: i\in I_0\cup I_1\cup I_2\}.$$

Тогда условия оптимальности будут выполнены для такого значения вектора $\boldsymbol{\alpha}$, при котором

$$b_{\text{low}}\le b_{\text{up}}. \tag{4}$$

Будем говорить, что пара индексов $\{i,j\}$ определяет нарушение при заданном $\boldsymbol{\alpha}$, если выполнено одно из условий:

$$i\in I_0\cup I_1\cup I_2 \quad \text{и}\ j\in I_0\cup I_1\cup I_3\ \text{и}\ \tilde{F}_i>\bar{F}_j, \tag{5а}$$

$$i\in I_0\cup I_1\cup I_3 \quad \text{и}\ j\in I_0\cup I_1\cup I_2\ \text{и}\ \bar{F}_i<\tilde{F}_j. \tag{5б}$$

Условия оптимальности будут выполнены для данного $\boldsymbol{\alpha}$ тогда и только тогда, когда не останется ни одной пары индексов $\{i,j\}$, определяющей нарушение при этом значении $\boldsymbol{\alpha}$.

При вычислениях используем положительный параметр толерантности $\tau > 0$. Условие (4) примет вид

$$b_{\text{low}}\le b_{\text{up}}+2\tau, \tag{6}$$

а для определения нарушения заменим условия (5а) и (5б) соответственно на

$$i\in I_0\cup I_1\cup I_2, \quad j\in I_0\cup I_1\cup I_3\ \text{и}\ \tilde{F}_i>\bar{F}_j+2\tau; \tag{7а}$$

**Таблица 1.** Допустимый интервал для $\alpha_i^{(*)}$

| | $\alpha_j$ | $\alpha_j^*$ |
|---|---|---|
| $\alpha_i$ | $L = \max(0,\ \gamma-C)$<br>$H = \min(C,\ \gamma)$ | $L = \max(0,\ \gamma)$<br>$H = \min(C,\ C+\gamma)$ |
| $\alpha_i^*$ | $L = \max(0,\ -\gamma)$<br>$H = \min(C,\ -\gamma+C)$ | $L = \max(0,\ -\gamma-C)$<br>$H = \min(C,\ -\gamma)$ |

$$i\in I_0\cup I_1\cup I_3, \quad j\in I_0\cup I_1\cup I_2\ \text{и}\ \bar{F}_i<\tilde{F}_j-2\tau. \tag{7б}$$

SMO-алгоритм использует условие (6) (или (7)) для проверки оптимальности $\boldsymbol{\alpha}$. Заметим, что при этом не требуется знать параметр сдвига $b$. Кроме того, благодаря такому способу проверки условий ККТ, при заданном первом члене рабочего множества второй его член находится автоматически.

**Выбор рабочего множества из двух элементов.** На каждом шаге SMO-алгоритм выбирает две переменные $\{\alpha_i^{(*)},\alpha_j^{(*)}\}$ двойственной задачи (1) для совместной оптимизации. Выбор такого рабочего множества осуществляется аналогично SMO-алгоритму для задачи классификации [10].

**Решение задачи квадратичного программирования для выбранного рабочего множества.** Рабочее множество состоит из двух переменных $\{\alpha_i^{(*)},\alpha_j^{(*)}\}$ с разными индексами $i$ и $j$. Таким образом, имеем четыре различных состава рабочего множества, каждый из которых нужно рассматривать отдельно. Ограничение типа равенств задачи (1) можно использовать для исключения одной из переменных (пусть это будет переменная с индексом $j$):

$$(\alpha_i-\alpha_i^*)+(\alpha_j-\alpha_j^*)= \tag{8}$$
$$=(\alpha_i^{(\text{old})}-\alpha_i^{*(\text{old})})+(\alpha_j^{(\text{old})}-\alpha_j^{*(\text{old})})\equiv\gamma,\ \gamma=\text{const}.$$

Ограничения типа неравенств задачи (1) для переменной с индексом $j$ приводят к определенным ограничениям для переменной с индексом $i$, для которой решаем аналитически соответствующую задачу квадратичного программирования. Допустимый интервал для оптимизируемой $i$-й переменной в зависимости от состава рабочего множества представлен в табл. 1. Целевая функция $W(\alpha_i^{(*)},\alpha_j^{(*)})$ для задачи квадратичного программирования размерности два имеет вид:

$$-\frac{1}{2}\begin{pmatrix}\alpha_i-\alpha_i^*\\ \alpha_j-\alpha_j^*\end{pmatrix}^T\begin{pmatrix}k(\boldsymbol{x}_i,\boldsymbol{x}_i) & k(\boldsymbol{x}_i,\boldsymbol{x}_j)\\ k(\boldsymbol{x}_i,\boldsymbol{x}_j) & k(\boldsymbol{x}_j,\boldsymbol{x}_j)\end{pmatrix}\begin{pmatrix}\alpha_i-\alpha_i^*\\ \alpha_j-\alpha_j^*\end{pmatrix}+ \quad (9)$$
$$+ v_i(\alpha_i-\alpha_i^*)+v_j(\alpha_j-\alpha_j^*)-\varepsilon(\alpha_i+\alpha_i^*+\alpha_j+\alpha_j^*),$$

где $v_i = y_i + b - \sum_{\substack{p=1\\ p\neq i,j}}^{l}(\alpha_p-\alpha_p^*)k(\boldsymbol{x}_i,\boldsymbol{x}_p)$.

Нужно найти значения $(\alpha_i^{(*)},\alpha_j^{(*)})$, доставляющие максимум $W(\alpha_i^{(*)},\alpha_j^{(*)})$ при ограничениях:

$$\begin{cases}(\alpha_i-\alpha_i^*)+(\alpha_j-\alpha_j^*)=\gamma,\\ \alpha_i,\alpha_i^*,\alpha_j,\alpha_j^*\in[0,C].\end{cases}$$

Используя ограничение типа равенств, исключаем переменные $\alpha_j,\alpha_j^*$. Учитываем, что $\alpha_j\alpha_j^* = \alpha_i\alpha_i^* = 0$, и решаем аналитически задачу квадратичного программирования для переменной с индексом $i$. Значение переменной с индексом $j$ вычисляем опираясь на полученное в процессе оптимизации значение переменной с индексом $i$ и используя соотношение (8).

Значения элементов рабочего множества, доставляющие безусловный максимум целевой функции $W(\alpha_i^{(*)},\alpha_j^{(*)})$ в зависимости от состава рабочего множества, представлены в табл. 2. Здесь $\eta = k(\boldsymbol{x}_1,\boldsymbol{x}_1) + k(\boldsymbol{x}_2,\boldsymbol{x}_2) - 2k(\boldsymbol{x}_1,\boldsymbol{x}_2)$ – вторая производная целевой функции по $\alpha_i^{(*)}$.

Окончательно получим значения $\alpha_i^{(*)}$ и $\alpha_j^{(*)}$, доставляющие условный максимум целевой функции $W(\alpha_i^{(*)},\alpha_j^{(*)})$, учитывая ограничения типа неравенств задачи (1) согласно допустимым интервалам для $\alpha_i^{(*)}$ (табл. 1). При этом значение переменной $\alpha_j^{(*)}$ вычисляем опираясь на полученное в процессе условной оптимизации значение переменной $\alpha_i^{(*)}$.

Если в тренировочной последовательности имеются хотя бы два одинаковых вектора $\boldsymbol{x}$, то $\eta$ может стать равной нулю. SMO-алгоритм будет работать и в этом случае, вычисляя $W_L$ и $W_H$ – значения целевой функции $W(\alpha_i^{(*)},\alpha_j^{(*)})$ (9) на концах сегмента прямой, которую определяет линейная зависимость между $\alpha_i^{(*)}$ и $\alpha_j^{(*)}$. $L$ и $H$ – границы для $\alpha_i^{(*)}$ (табл. 1). Если $W_L < W_H - \tau$, то $\alpha_i^{(*)\text{new}} = L$. Если $W_L > W_H + \tau$, то $\alpha_i^{(*)\text{new}} = H$. Если значения целевой функции $W_L$ и $W_H$ отличаются друг от друга меньше чем на $\tau$, то текущее значение $\alpha_i^{(*)}$ не меняется, и выбирается новое рабочее множество.

Структура SMO-алгоритма для задачи восстановления регрессии аналогична структуре SMO-алгоритма для задачи классификации [10,11].

## АЛГОРИТМ SUCCESSIVE OVERRELAXATION

Алгоритм Successive Overrelaxation (SOR) [12,13] для решения задачи бинарной классификации допускает простое обобщение для задачи восстановления регрессии при использовании ε-нечувствительной функции потерь. Добавление к целевой функции исходной задачи SVR-обучения дополнительного слагаемого $b^2/2$ позволяет (как и в случае задачи классификации) получить двойственную задачу без ограничений типа равенств, решение которой эквивалентно решению системы линейных алгебраических уравнений. Алгоритм SOR строит решение этой системы методом последовательной верхней релаксации.

Оптимальная решающая функция для задачи восстановления регрессии имеет вид (2), где

$$b = \sum_{i=1}^{l}(\alpha_i-\alpha_i^*).$$

Введем обозначения:

$$\boldsymbol{H}=\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}},\ \text{где}\ \boldsymbol{Z}=\begin{bmatrix}d_1\boldsymbol{\varphi}(\boldsymbol{x}_1)\\ \dots\\ d_l\boldsymbol{\varphi}(\boldsymbol{x}_l)\\ d_{l+1}\varphi(\boldsymbol{x}_1)\\ \dots\\ d_{2l}\boldsymbol{\varphi}(\boldsymbol{x}_l)\end{bmatrix},\ \boldsymbol{d}=\begin{bmatrix}1\\ \dots\\ 1\\ -1\\ \dots\\ -1\end{bmatrix},\ \boldsymbol{E}=\boldsymbol{d}\boldsymbol{d}^{\mathrm{T}},$$

$$\boldsymbol{e}=\begin{bmatrix}y_1-\varepsilon\\ \dots\\ y_l-\varepsilon\\ -y_1-\varepsilon\\ \dots\\ -y_l-\varepsilon\end{bmatrix},\quad \boldsymbol{\alpha}=\begin{bmatrix}\alpha_1\\ \dots\\ \alpha_l\\ \alpha_1^*\\ \dots\\ \alpha_l^*\end{bmatrix},\quad \boldsymbol{M}=\boldsymbol{H}+\boldsymbol{E}.$$

Тогда упомянутую выше двойственную задачу без ограничений типа равенств можно записать в виде:

$$\text{найти}\ \min_{\boldsymbol{\alpha}}\frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\alpha}-\boldsymbol{e}^{\mathrm{T}}\boldsymbol{\alpha}$$

$$\text{при ограничениях}\ 0\le\boldsymbol{\alpha}\le C\boldsymbol{I}, \quad (10)$$

где $\boldsymbol{I}$ – единичная матрица размерности $2l \times 2l$.

**Таблица 2.** Значения элементов РМ, доставляющие безусловный максимум функции $W(\alpha_i^{(*)},\alpha_j^{(*)})$

| РМ | $\alpha_i^{(*)new}$ | $\alpha_j^{(*)new}$ |
|---|---|---|
| $\alpha_i,\ \alpha_j$ | $\alpha_i^{old} - (F_j - F_i)/\eta$ | $\alpha_j^{old} - (\alpha_i^{new} - \alpha_i^{old})$ |
| $\alpha_i,\ \alpha_j^*$ | $\alpha_i^{old} - (F_j - F_i + 2\varepsilon)/\eta$ | $\alpha_j^{*old} + (\alpha_i^{new} - \alpha_i^{old})$ |
| $\alpha_i^*,\ \alpha_j$ | $\alpha_i^{*old} + (F_j - F_i - 2\varepsilon)/\eta$ | $\alpha_j^{old} + (\alpha_i^{*new} - \alpha_i^{*old})$ |
| $\alpha_i^*,\ \alpha_j^*$ | $\alpha_i^{*old} + (F_j - F_i)/\eta$ | $\alpha_j^{*old} - (\alpha_i^{*new} - \alpha_i^{*old})$ |

Приравнивая нулю градиент целевой функции задачи (10), приходим к системе линейных алгебраических уравнений: $\boldsymbol{M}\boldsymbol{\alpha} = \boldsymbol{e}$, которую решаем методом последовательной верхней релаксации.

*Структура SOR-алгоритма.*

1. Выбираем значение $\omega \in (0,2)$.

2. Выбираем начальное приближение $\boldsymbol{\alpha}^0 \in R^{2l}$. В качестве $\boldsymbol{\alpha}^0$ обычно берут нулевой вектор.

3. Зная $\boldsymbol{\alpha}^k$, вычисляем компоненты вектора $\boldsymbol{\alpha}^{k+1}$ по формуле

$$\alpha_i^{(k+1)} =$$

$$= \left( \omega \left( e_i - \sum_{j=1}^{i-1} \boldsymbol{m}_{ij}\alpha_j^{(k+1)} - \sum_{j=i+1}^{l} \boldsymbol{m}_{ij}\alpha_j^{(k)} \right) / \boldsymbol{m}_{ii} + (1-\omega)\alpha_i^{(k)} \right)_{\#},$$

$$i = \overline{1,2l},$$

где $(\alpha_i)_{\#} = \begin{cases} 0, \text{ если } \alpha_i \le 0 \\ \alpha_i, \text{ если } 0 \le \alpha_i \le C. \\ C, \text{ если } \alpha_i \ge C \end{cases}$

Вычисления проводим, пока $\|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\| > \tau$, где $\tau$ – заданный уровень толерантности (обычно $\tau = 10^{-3}$).

## АЛГОРИТМ INCREMENTAL UPDATING

Алгоритм Incremental Updating (IU_R) – SVM-алгоритм для задачи восстановления регрессии, используемый в условиях, когда тренировочные образцы поступают по одному, в online-режиме. Пошаговое SVR-обучение на новых данных путем отбрасывания всех членов предшествующей тренировочной последовательности, кроме полученных опорных векторов, дает только приближенное решение. В работе [14,15] предложен точный online-метод для задачи классификации (IU-SVM-алгоритм), который обновляет решение задачи обучения при поступлении каждого нового образца. Обобщение этого алгоритма для задачи восстановления регрессии в случае ε-нечувствительной функции потерь рассмотрено в работе [16].

После добавления нового тренировочного образца IU_R-алгоритм обновляет уже построенное оптимальное решение SVR-задачи обучения. За конечное число шагов алгоритм приводит к выполнению условий ККТ как для всех предшествующих данных, так и для вновь поступившего образца. Это достигается путем изменения ключевых переменных системы за счет наибольшего возможного приращения двойственной переменной, соответствующей новому образцу.

**Условия оптимальности.** Пусть $\boldsymbol{Q}$ – симметричная положительно определенная $l \times l$ матрица с элементами $Q_{ij} = k(\boldsymbol{x}_i,\boldsymbol{x}_j)$. Включим в целевую функцию двойственной задачи для восстановления регрессии в случае ε-нечувствительной функции потерь (1) ограничение типа равенств:

$$W = \frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)Q_{ij} +$$

$$+ \varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) - \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) + b\sum_{i=1}^{l}(\alpha_i - \alpha_i^*).$$

Введем обозначения:

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_{j=1}^{l} Q_{ij}(\alpha_j - \alpha_j^*) - y_i + \varepsilon + b, \ i = 1,\ldots,l;$$

$$g_i^* = \frac{\partial W}{\partial \alpha_i^*} = -g_i + 2\varepsilon; \ \beta_i = (\alpha_i - \alpha_i^*), \ i = 1,\ldots l.$$

До поступления следующего нового образца условия ККТ выполнены для всех $l$ образцов тренировочной последовательности, т.е. по $l$ образцам построена SV-машина. Согласно условиям ККТ, векторы тренировочной последовательности можно разделить на четыре категории:

$S = \{\boldsymbol{x}_i : g_i = 0$ или $g_i^* = 0\}$ – множество опорных векторов, лежащих на границах ε-полосы;

$E = \{\boldsymbol{x}_i : g_i < 0\}$ – множество опорных векторов, нарушающих верхнюю границу ε-полосы;

$E^* = \{\boldsymbol{x}_i : g_i^* < 0\}$ – множество опорных векторов, нарушающих нижнюю границу ε-полосы;

$R = \{\boldsymbol{x}_i : g_i > 0$ и $g_i^* > 0\}$ – множество векторов, не являющихся опорными (лежащих внутри ε-полосы).

Соответствующие этим множествам векторов множества индексов (будем обозначать их строчными буквами) индуцируют некоторое разбиение матрицы $\boldsymbol{Q}$, вектора $\boldsymbol{y}$, вектора коэффициентов $\boldsymbol{\beta}$ и градиента целевой функции $\boldsymbol{g}$. Будем использовать для таких разбиений нижние индексы $s$, $e$, $e^*$, $r$.

Сформулируем условия оптимальности для задачи с целевой функцией $W$ и ограничениями (1) в терминах $g_i$ и $g_i^*$:

$$\begin{gathered} g_i > 2\varepsilon \to g_i^* < 0,\ \beta_i = -C,\ i \in e^*; \\ g_i > 2\varepsilon \to g_i^* = 0,\ -C < \beta_i < 0,\ i \in s; \\ 0 < g_i < 2\varepsilon \to 0 < g_i^* < 2\varepsilon,\ \beta_i = 0,\ i \in r; \\ g_i = 0 \to g_i^* = 2\varepsilon,\ 0 < \beta_i < C,\ i \in s; \\ g_i < 0 \to g_i^* > 2\varepsilon,\ \beta_i = C,\ i \in e. \end{gathered} \qquad (11)$$

Пусть поступил новый тренировочный образец $(\boldsymbol{x}_c, y_c)$. Вычислим для него $g_c$ и $g_c^*$. Если обе эти величины положительны, то $\boldsymbol{x}_c$ оказался внутри ε-полосы, т.е. $\boldsymbol{x}_c$ нужно поместить в множество $R$. В противном случае положим $\beta_c = 0$ и будем тщательно модифицировать это значение, увеличивая его при $g_c < 0$ или уменьшая при $g_c^* < 0$, пока значения $g_c$, $g_c^*$, $\beta_c$ не станут соответствовать условиям ККТ (11) (т.е. $\boldsymbol{x}_c$ нужно будет поместить в одно из множеств: $S$, $E$ или $E^*$).

Рассмотрим, каким образом изменение коэффициента $\beta_c$ (на величину $\Delta\beta_c$) влияет на значения величин $g_i$, $g_i^*$, $\beta_i$, соответствующих векторам тренировочной последовательности, в предположении, что состав множеств $S$, $E$, $E^*$, $R$ при этом не меняется. Проводя рассуждения, аналогичные случаю классификации [15], получим правила изменения ключевых переменных системы:

1. Для приращений коэффициентов, соответствующих векторам из $S$, и для $b$ получим

$$\Delta b = \delta_0 \Delta\beta_c \qquad (12)$$

$$\Delta\beta_j = \delta_j \Delta\beta_c \quad \forall j \in S,$$

где $\boldsymbol{\delta}$ – вектор чувствительностей коэффициентов $\beta_j$, $j \in S$, и сдвига $b$ к изменению $\beta_c$ имеет структуру $[\delta_0\ \boldsymbol{\delta}_s^{\mathrm{T}}]$ и определяется следующим образом:

$$\boldsymbol{\delta} = -\begin{bmatrix} 0 & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & \boldsymbol{Q}_{ss} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ \boldsymbol{Q}_{cs}^{\mathrm{T}} \end{bmatrix}.$$

Обозначим первый матричный сомножитель в этом выражении через $\mathfrak{R}$.

2. Для приращений величин, соответствующих векторам, не принадлежащим множеству $S$, получим:

$$\begin{bmatrix} \Delta g_c \\ \Delta g_i \end{bmatrix} = \gamma \Delta\beta_c \quad \forall i \notin S, \qquad (13)$$

где вектор чувствительностей $\boldsymbol{\gamma}$ определяется следующим образом:

$$\gamma_i = Q_{ic} + \sum_{j \in S} Q_{ij}\delta_j + \delta_0 \quad \forall i \notin S,\ i = c.$$

Легко видеть, что $\Delta g_i^* = -\Delta g_i$, так что соотношение (13) задает и изменение $\boldsymbol{g}^*$, индуцируемое изменением $\beta_c$. Итак, обновление машины опорных векторов, вызванное новым членом тренировочной последовательности, должно контролироваться соотношениями чувствительности (12) и (13).

**Верхний предел для величины $\beta_c$.** IU_R-алгоритм определяет наибольшее возможное приращение $\Delta\beta_c$, при котором некоторый вектор стремится мигрировать в другое (соседнее) множество. Чтобы учесть подобные структурные изменения, рассмотрим возникающие при этом ситуации.

1. Некоторый вектор перемещается из $E$ в $S$.

Это перемещение необходимо, если вследствие обновления согласно (13) соответствующее $g_i$ для $i \in E$ становится равным нулю. Тогда наибольшее допустимое изменение величины $\beta_c$, которое не вынуждает векторы множества $E$ перемещаться в $S$, будет иметь вид:

$$\Delta\beta_c^{(4)} = \min_{i \notin \varepsilon} \frac{-g_i}{\gamma_i}.$$

2. Некоторый вектор перемещается из $S$ в $E$.

Это перемещение необходимо, если вследствие обновления согласно (12) соответствующее $\beta_i$ (которое было положительным) для $i \in S$ становится равным $C$. Тогда наибольшее допустимое изменение величины $\beta_c$, которое не

вынуждает векторы множества $S$ перемещаться в $E$, будет иметь вид:

$$\Delta\beta_c^{(1)} = \min_{i \in s} \frac{C - \beta_i}{\delta_i}.$$

3. Некоторый вектор перемещается из $S$ в $R$.

Это перемещение необходимо, если вследствие обновления согласно (12) соответствующее $\beta_i$ (которое не лежало на границах) для $i \in S$ становится равным нулю. Тогда наибольшее допустимое изменение величины $\beta_c$, которое не вынуждает векторы множества $S$ перемещаться в $R$, будет иметь вид:

$$\Delta\beta_c^{(2)} = \min_{i \in s} \frac{-\beta_i}{\delta_i}.$$

4. Некоторый вектор перемещается из $R$ в $S$.

Это перемещение необходимо, если вследствие обновления согласно (13) одна из соответствующих величин, $g_i$ или $g_i^*$ для $i \in R$ становится равной нулю. Тогда наибольшее допустимое изменение величины $\beta_c$, которое не вынуждает векторы множества $R$ перемещаться в $S$, будет иметь вид:

$$\Delta\beta_c^{(5)} = \min\left\{\min_{i \in R} \frac{-g_i}{\gamma_i}, \min_{i \in R} \frac{-g_i^*}{\gamma_i}\right\}.$$

5. Некоторый вектор перемещается из $S$ в $E^*$.

Это перемещение необходимо, если вследствие обновления согласно (12) соответствующее $\beta_i$ (которое было отрицательным) для $i \in S$ становится равным $-C$. Тогда наибольшее допустимое изменение величины $\beta_c$, которое не вынуждает векторы множества $S$ перемещаться в $E^*$, будет иметь вид:

$$\Delta\beta_c^{(3)} = \min_{i \in s} \frac{-C - \beta_i}{\delta_i}.$$

6. Некоторый вектор перемещается из $E^*$ в $S$.

Это перемещение необходимо, если вследствие обновления согласно (13) соответствующее $g_i^*$ для $i \in E^*$ становится равным нулю. Тогда наибольшее допустимое изменение величины $\beta_c$, которое не вынуждает векторы множества $E^*$ перемещаться в $S$, будет иметь вид:

$$\Delta\beta_c^{(6)} = \min_{i \in e^*} \frac{g_i^*}{\gamma_i}.$$

7. Величина $g_c$ становится нулем. Если обновление возможно, т.е. $\gamma_c > \tau$, то наибольшее возможное приращение $\beta_c$ имеет вид:

$$\Delta\beta_c^{(7)} = \min\left\{\frac{-g_c}{\gamma_c}, \frac{g_i^*}{\gamma_c}\right\}.$$

8. $\beta_c$ достигает верхней границы, $C$.

В этом случае наибольшее возможное приращение $\beta_c$ таково:

$$\Delta\beta_c^{(8)} = C - \beta_c.$$

9. $\beta_c$ достигает нижней границы, $-C$.

В этом случае наибольшее возможное приращение $\beta_c$ таково:

$$\Delta\beta_c^{(9)} = -C - \beta_c.$$

Наименьшее из перечисленных девяти приращений $\Delta\beta_c$ и представляет собой наибольшее возможное приращение $\beta_c$, которое будем обозначать $\Delta\beta_c^{\max}$.

**Рекурсивное обновление матрицы $\mathfrak{R}$.** После того как наибольшее возможное приращение $\beta_c$ определено, производится обновление переменных $\boldsymbol{\beta}_s$, $b$, $\boldsymbol{g}$ согласно формулам (12) и (13). Матрица $\mathfrak{R}$ тоже должна быть пересчитана для того, чтобы учесть новый состав множества $S$. Рассмотрим эффективное обновление этой матрицы.

1. Некоторый вектор $\boldsymbol{x}_k$ добавляется к множеству $S$.

Применение формулы Шермана–Моррисона–Вудбери для обращения блочной матрицы [17] приводит к соотношению

$$\mathfrak{R} \leftarrow \begin{bmatrix} \mathfrak{R} & \boldsymbol{\eta}_k \\ \boldsymbol{\eta}_k^T & Q_{kk} \end{bmatrix}^{-1} = \begin{bmatrix} \mathfrak{R} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + \frac{1}{Q_{kk} - \boldsymbol{\eta}_k^T \mathfrak{R} \boldsymbol{\eta}_k} \begin{bmatrix} \delta_k \\ 1 \end{bmatrix} [\delta_k^{\mathrm{T}} 1],$$

где $\boldsymbol{\eta}_k = \begin{bmatrix} 1 \\ \boldsymbol{Q}_{ks}^T \end{bmatrix}$, $\delta_k = -\mathfrak{R}\boldsymbol{\eta}_k$.

2. Некоторый вектор $\boldsymbol{x}_k$ удаляется из множества $S$.

Тогда матрица $\mathfrak{R}$ конструируется следующим образом:

$$\mathfrak{R}_{ij} \leftarrow \mathfrak{R}_{ij} - \mathfrak{R}_{kk}^{-1} \mathfrak{R}_{ik} \mathfrak{R}_{kj}, \forall \mathfrak{R}_{ij} \in \mathfrak{R}; i \in s \cup \{0\}, i,j \neq k,$$

где индекс 0 соответствует $b$.

При поступлении первого вектора в множество $S$ инициализируем матрицу $\mathfrak{R}$:

$$\mathfrak{R} = \begin{bmatrix} -Q_{cc} & 1 \\ 1 & 0 \end{bmatrix}.$$

**Структура IU_R-алгоритма.** Пусть на основе тренировочной последовательности объема $l$ построена SV-машина. При поступлении нового образца $\{\boldsymbol{x}_c, y_c\}$ используем имеющиеся решение $\{\beta_i^l, b^l\}$, $i = 1,\ldots, l$, и матрицу $\mathfrak{R}$ для получения нового оптимального решения $\{\beta_i^{l+1}, b^{l+1}\}$, $i = 1,\ldots, l+1$.

Этапы IU_R-алгоритма ($l \leftarrow l+1$):

1. Первоначально положим $\beta_c = 0$.

2. Если $g_c > 0$ и $g_c^* > 0$, то $r \leftarrow r \cup \{c\}$ и SVR-решение остается прежним. Конец процедуры.

3. Если $g_c \le 0$, выполняем:

увеличиваем $\beta_c$, обновляем величины $g_i$, $g_i^*$, $\beta_i$ в соответствии с уравнениями (12) и (13), пока одно из следующих условий не будет выполнено:

$g_c = 0$, тогда добавляем $\boldsymbol{x}_c$ к множеству $S$, обновляем матрицу $\mathfrak{R}$. Конец процедуры;

$\beta_c = C$, тогда добавляем $\boldsymbol{x}_c$ к множеству $E$. Конец процедуры;

некоторый вектор мигрирует из/в множества $R$, $E$ или $E^*$ в/из множество $S$.

Тогда обновляем соответствующие множества и матрицу $\mathfrak{R}$.

Иначе (т.е. $g_c^* \le 0$) выполняем:

уменьшаем $\beta_c$, обновляем величины $g_i$, $g_i^*$, $\beta_i$ в соответствии с уравнениями (12) и (13), пока одно из следующих условий не будет выполнено:

$g_c = 0$, тогда добавляем $\boldsymbol{x}_c$ к множеству $S$, обновляем матрицу $\mathfrak{R}$. Конец процедуры;

$\beta_c = -C$, тогда добавляем $\boldsymbol{x}_c$ к множеству $E^*$. Конец процедуры;

некоторый вектор мигрирует из/в множества $R$, $E$ или $E^*$ в/из множество $S$.

Тогда обновляем соответствующие множества и матрицу $\mathfrak{R}$.

Возврат к пункту 3.

IU_R-алгоритм всегда сходится за конечное число шагов [15].

## АЛГОРИТМ GREEDY SPARSE LEAST-SQUARES

Итерационный алгоритм Greedy Sparse Least-Squares (GSLS) [6,7] основан на использовании квадратичной функции потерь. Стандартная постановка SVR-задачи в этом случае имеет вид:

$$\text{найти} \min_{w,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{l}\sum_{i=1}^{l}\xi_i^2, \tag{14}$$

при ограничениях

$$y_i = \langle \boldsymbol{w}, \boldsymbol{\varphi}(\boldsymbol{x}_i) \rangle + b + \xi_i,\ \xi_i \sim N(0,\sigma).$$

Здесь $\boldsymbol{\varphi}$ – отображение исходных векторов в пространство признаков. Решение задачи (14)

$$f(\boldsymbol{x}) = \sum_{i=1}^{l}\alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

оказывается абсолютно плотным: $\alpha_I \ne 0$ для всех $i$ [1]. Алгоритм GSLS обеспечивает разреженность решения за счет включения на каждой итерации в ядерное разложение, $\boldsymbol{w} = \Sigma_i \alpha_i \boldsymbol{\varphi}(\boldsymbol{x}_i)$, образца, минимизирующего регуляризованный эмпирический риск.

Пусть вектор весов $\boldsymbol{w}$ достаточно хорошо аппроксимируется взвешенной суммой ограниченного подмножества $S$ векторов тренировочной последовательности, т.е. $\boldsymbol{w} \approx \Sigma_i \beta_i \boldsymbol{\varphi}(\boldsymbol{x}_i)$, $i \in S \subset \{1,2,\ldots l\}$, где объем $S$, $|S|$, значительно меньше $l$. Тогда целевая функция (14) может быть аппроксимирована функцией

$$L(\beta, b) = \frac{1}{2}\sum_{i,j\in S}\beta_i\beta_j k_{ij} + \frac{C}{l}\sum_{i=1}^{l}\left(y_i - \sum_{j\in S}\beta_j k_{ij} - b\right)^2. \tag{15}$$

где $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Приравняв частные производные функции $L$ по $\beta_i$, $i \in S$ и $b$ нулю, получим систему $|S|+1$ линейных алгебраических уравнений с $|S|+1$ неизвестными:

$$\sum_{i\in S}\beta_i\sum_{j=1}^{l}k_{ij} + lb = \sum_{j=1}^{l}y_j \quad \text{и}$$

$$\sum_{i\in S}\beta_i\left\{\frac{l}{2C}k_{ir} + \sum_{j=1}^{l}k_{jr}k_{jt}\right\} + b\sum_{i=1}^{l}k_{ir} = \sum_{i=1}^{l}y_i k_{ir},\ \forall r \in S,$$

или в матричном виде:

$$\boldsymbol{H}\begin{bmatrix} b \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} l & \boldsymbol{q}^{\mathrm{T}} \\ \boldsymbol{q} & \boldsymbol{P} \end{bmatrix}\begin{bmatrix} b \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \Sigma_{k=1}^{l} y_k \\ c \end{bmatrix}, \tag{16}$$

где

$$\boldsymbol{P} = \left[\frac{l}{2\gamma}k_{ij} + \sum_{r=1}^{l} k_{rj}k_{ri}\right]_{i,j\,\in\,S},$$

$$\boldsymbol{q} = \left(\sum_{j=1}^{l} k_{ij}\right)_{i\,\in\,S}, \boldsymbol{c} = \left(\sum_{j=1}^{l} y_j k_{ij}\right)_{i\,\in\,S}.$$

На нулевом шаге алгоритма система (16) представляет собой уравнение относительно параметра сдвига $b$.

Затем на каждом $m$-м шаге алгоритма ($l$–$m$+1) раз решаем систему (16) размерности ($m$+1). В множество опорных векторов (с индексами из $S$) включаем тренировочный вектор, минимизирующий целевую функцию (15). Рекурсивное обращение блочной матрицы проводим следующим образом:

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{d} \\ \boldsymbol{d}^{\mathrm{T}} & g \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} + \frac{1}{h}\boldsymbol{A}^{-1}\boldsymbol{d}\boldsymbol{d}^{\mathrm{T}}\boldsymbol{A}^{-1} & -\frac{1}{h}\boldsymbol{A}^{-1}\boldsymbol{d} \\ -\frac{1}{h}\boldsymbol{d}^{\mathrm{T}}\boldsymbol{A}^{-1} & \frac{1}{h} \end{bmatrix},$$

где $h = g - \boldsymbol{d}^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{d}$.

Обучение машины завершаем, когда $|S|$ достигает заранее заданного размера или когда величина приращения целевой функции (15) становится меньше заданного порогового значения $\tau$.

## АЛГОРИТМ KERNEL RECURSIVE LEAST SQUARES

Алгоритм Kernel Recursive Least Squares (KRLS) – online-алгоритм восстановления регрессии, основанный на использовании квадратичной функции потерь [8]. Алгоритм конструирует разреженное решение по мере поступления новых членов тренировочной последовательности. Вновь поступивший вектор добавляется в словарь, т.е. становится опорным, если он не является приближенной линейной аппроксимацией векторов, составляющих словарь к моменту его поступления.

**Процедура построения разреженного решения (составления словаря).** Пусть в момент времени $t$ поступил новый образец $(\boldsymbol{x}_t, y_t)$, и пусть к этому моменту был составлен словарь из $m_{t-1}$ векторов:

$$D_{t-1} = \{\tilde{\boldsymbol{x}}_j\}_{j=1}^{m_{t-1}},$$

где $\tilde{\boldsymbol{x}}_j$ – векторы, включенные в словарь, $\{\boldsymbol{\varphi}(\tilde{\boldsymbol{x}}_j)\}_{j=1}^{m_{t-1}}$ – соответствующие им линейно независимые векторы в пространстве признаков.

Вектор $\boldsymbol{x}_t$, поступивший в момент времени $t$, не включается в словарь, если найдется такой вектор коэффициентов $\mathbf{a}_t$ размерности $m_{t-1}$, для которого выполнено условие приближенной линейной зависимости (approximate linear dependence, ALD):

$$\delta_t \le \nu, \text{ где } \delta_t = \min_{\mathbf{a}_t} \left|\left| \sum_{j=1}^{m_{t-1}} \mathrm{a}_{tj}\boldsymbol{\varphi}(\tilde{\boldsymbol{x}}_j) - \boldsymbol{\varphi}(\boldsymbol{x}_t) \right|\right|^2, \quad (17)$$

где $\nu$ – параметр, определяющий степень разреженности решения.

Введем обозначения:

$$[\tilde{\boldsymbol{K}}_{t-1}]_{i,j} = k(\tilde{\boldsymbol{x}}_i,\tilde{\boldsymbol{x}}_j),\ (k_{t-1}(\boldsymbol{x}_t))_i = k(\tilde{\boldsymbol{x}}_i,\boldsymbol{x}_t),$$
$$k_{tt} = k(\boldsymbol{x}_t,\boldsymbol{x}_t),\ i, j = 1,\ldots,m_{i-1}.$$

После возведения в квадрат и замены скалярного произведения ядерной функцией $\langle\boldsymbol{\varphi}(\boldsymbol{x}),\boldsymbol{\varphi}(\boldsymbol{x}')\rangle = k(\boldsymbol{x},\boldsymbol{x}')$ выражение (17) примет вид:

$$\delta_t = \min_{\mathbf{a}_t}\{\mathbf{a}_t^{\mathrm{T}}\tilde{\boldsymbol{K}}_{t-1}\mathbf{a}_t - 2\mathbf{a}_t^{\mathrm{T}}\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) + k_{tt}\}. \quad (18)$$

Минимизация $\delta_t$ в представлении (18) дает оптимальный вектор коэффициентов

$$\tilde{\mathbf{a}}_t = \tilde{\boldsymbol{K}}_{t-1}^{-1}\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t), \quad (19)$$

тогда условие ALD принимает вид:

$$\delta_t = k_{tt} - \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x})^{\mathrm{T}}\tilde{\mathbf{a}}_t \le \nu. \quad (20)$$

Пусть $\boldsymbol{A}_t$ – матрица коэффициентов, построенная к моменту времени $t$, $[\boldsymbol{A}_t]_{ij} = \mathrm{a}_{ij}$ Если для $\boldsymbol{x}_i$, $i \in \{1,2,\ldots,\ t\}$ выполнено условие ALD, $i$-ю строку матрицы $\boldsymbol{A}_t$ определяет вектор $\tilde{\mathbf{a}}_t$, вычисленный по формуле (19). Если для $\boldsymbol{x}_i$ условие ALD не выполнено, т.е. $\boldsymbol{\varphi}(\boldsymbol{x}_i)$ не является линейной комбинацией векторов словаря, то $i$-ю строку матрицы $\boldsymbol{A}_t$ определяет вектор (0,…,0,1) длины $m_i$, где $m_i$ – число опорных векторов после поступления $\boldsymbol{x}_i$. Отметим, что в силу последовательной природы алгоритма, $[\boldsymbol{A}_t]_{ij} = 0$ для $j > m_i$.

Таким образом, для каждого $i$-го вектора, $i = 1,2,\ldots,\ t$, выполнено условие:

$$\boldsymbol{\varphi}(\boldsymbol{x}_i) = \sum_{j=1}^{m_i} a_{ij}\boldsymbol{\varphi}(\tilde{\boldsymbol{x}}_j) + \boldsymbol{\varphi}_i^{\mathrm{res}},\ (\|\boldsymbol{\varphi}_i^{\mathrm{res}}\|^2 \le \nu),$$

где $\boldsymbol{\varphi}_i^{\mathrm{res}}$ – вектор остаточных компонент. Если $\nu$ достаточно мало, то

$$\varphi(\boldsymbol{x}_i) \approx \sum_{j=1}^{m_i} a_{ij}\varphi(\tilde{\boldsymbol{x}}_j).$$

Введем матрицу $\boldsymbol{\Phi}_t = [\varphi(\boldsymbol{x}_1),\ldots, \varphi(\boldsymbol{x}_t)]$. Тогда

$$\boldsymbol{\Phi}_t \approx \tilde{\boldsymbol{\Phi}}_t \boldsymbol{A}_t^{\mathrm{T}}. \quad (21)$$

Ядерная матрица может быть аппроксимирована следующим образом:

$$\boldsymbol{K}_t \approx \boldsymbol{A}_t \tilde{\boldsymbol{K}}_t \boldsymbol{A}_t^{\mathrm{T}},$$

где $[\boldsymbol{K}_t]_{ij} = k(\boldsymbol{x}_i,\boldsymbol{x}_j)$ – элемент ядерной матрицы, построенной для всех поступивших к моменту времени $t$ образцов.

Алгоритм KRLS строит линейную регрессию в пространстве признаков в форме $f(\boldsymbol{x}) = \langle \boldsymbol{w},\varphi(\boldsymbol{x})\rangle$, где параметром является вектор $\boldsymbol{w}$, включающий и весовые коэффициенты, и сдвиг $b$. Соответствующий вектор отображений имеет вид $(\varphi^{\mathrm{T}},1)^{\mathrm{T}}$.

В каждый момент времени $t$ требуется минимизировать накопленную квадратичную ошибку:

$$L(\boldsymbol{w}) = \sum_{i=1}^{t}(f(\boldsymbol{x}_i) - y_i)^2 = \|\boldsymbol{\Phi}_t^{\mathrm{T}}\boldsymbol{w} - \boldsymbol{y}_t\|^2, \quad (22)$$

где $\boldsymbol{y}_t = (y_1,\ldots y_t)^{\mathrm{T}}$. Решением этой задачи является вектор $\boldsymbol{w}$:

$$\boldsymbol{w}_t = \underset{\boldsymbol{w}}{\operatorname{argmin}}\|\boldsymbol{\Phi}_t^{\mathrm{T}}\boldsymbol{w} - \boldsymbol{y}_t\|^2 = (\boldsymbol{\Phi}_t^{\mathrm{T}})^*,$$

здесь $\boldsymbol{B}^*$ обозначает матрицу, псевдообратную к матрице $\boldsymbol{B}$. Согласно теореме репрезентативности [18], вектор $\boldsymbol{w}$ может быть представлен в виде:

$$\boldsymbol{w}_t = \sum_{I=1}^{t}\alpha_t\varphi(\boldsymbol{x}_i) = \boldsymbol{\Phi}_t\boldsymbol{\alpha}, \quad (23)$$

где $\boldsymbol{\alpha} = (\alpha_1,\ldots \alpha_t)^{\mathrm{T}}$. Подставляя (23) в (22) и минимизируя по $\boldsymbol{\alpha}$ функцию $\|\boldsymbol{K}_t\boldsymbol{\alpha} - \boldsymbol{y}_t\|^2$, получаем оптимальный вектор $\boldsymbol{\alpha}_t = \boldsymbol{K}_t^*\boldsymbol{y}_t$. Используя представление (21) в (23), получаем:

$$\boldsymbol{w}_t = \boldsymbol{\Phi}_t\boldsymbol{\alpha}_t = \tilde{\boldsymbol{\Phi}}_t\boldsymbol{A}_t^{\mathrm{T}}\boldsymbol{\alpha}_t = \tilde{\boldsymbol{\Phi}}_t\tilde{\boldsymbol{\alpha}}_t, \quad (24)$$

здесь $\tilde{\boldsymbol{\alpha}}_t = \boldsymbol{A}_t^{\mathrm{T}}\boldsymbol{\alpha}_t$ – «редуцированный» вектор коэффициентов, где только $m$ компонент, соответствующих опорным векторам, отличны от 0. С учетом (24) накопленная функция потерь (22) принимает вид:

$$L(\tilde{\boldsymbol{\alpha}}) = \|\boldsymbol{\Phi}_t^{\mathrm{T}}\tilde{\boldsymbol{\Phi}}_t\tilde{\boldsymbol{\alpha}} - \boldsymbol{y}_t\|^2 = \|\boldsymbol{A}_t\tilde{\boldsymbol{K}}_t\tilde{\boldsymbol{\alpha}} - \boldsymbol{y}_t\|^2,$$

а ее минимизация приводит к решению

$$\tilde{\boldsymbol{\alpha}}_t = (\boldsymbol{A}_t\tilde{\boldsymbol{K}}_t)^*\boldsymbol{y}_t = \tilde{\boldsymbol{K}}_t^{-1}\boldsymbol{A}_t^*\boldsymbol{y}_t. \quad (25)$$

При online режиме поступления данных в каждый момент времени $t$ имеет место один из двух случаев:

1. $\delta_t \le \nu$, т.е. для $\varphi(\boldsymbol{x}_t)$ выполнено условие ALD.

Тогда вычисляем вектор $\tilde{\boldsymbol{a}}_t$ по формуле (19), словарь и ядерную матрицу для образцов, входящих в словарь, оставляем неизменными, т.е. $D_t = D_{t-1}$, $\tilde{\boldsymbol{K}}_t = \tilde{\boldsymbol{K}}_{t-1}$ и $m_t = m_{t-1}$. Тогда $\boldsymbol{A}_t = [\boldsymbol{A}_{t-1}^{\mathrm{T}},\tilde{\boldsymbol{a}}_t]^{\mathrm{T}}$, $\boldsymbol{A}_t^{\mathrm{T}}\boldsymbol{A}_t = \boldsymbol{A}_{t-1}^{\mathrm{T}}\boldsymbol{A}_{t-1} + \tilde{\boldsymbol{a}}_t\tilde{\boldsymbol{a}}_t^{\mathrm{T}}$.

Введем обозначения: $\boldsymbol{P}_t = (\boldsymbol{A}_t^{-1}\boldsymbol{A}_t)^{-1}$. Применим рекурсивное правило обращения матрицы [19] для вычисления матрицы $P_t$:

$$\boldsymbol{P}_t = \boldsymbol{P}_{t-1} - \frac{\boldsymbol{P}_{t-1}\tilde{\boldsymbol{a}}_t\tilde{\boldsymbol{a}}_t^{\mathrm{T}}\boldsymbol{P}_{t-1}}{1 + \tilde{\boldsymbol{a}}_t^{\mathrm{T}}\boldsymbol{P}_{t-1}\tilde{\boldsymbol{a}}_t} \quad (26)$$

и выведем правило обновления вектора $\tilde{\boldsymbol{\alpha}}_t$:

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{K}}_t^{-1}\boldsymbol{P}_t\boldsymbol{A}_t^{\mathrm{T}}\boldsymbol{y}_t = \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\boldsymbol{K}}_t^{-1}\boldsymbol{q}_t(y_t - \tilde{\boldsymbol{k}}_{t-1}^{\mathrm{T}}(\boldsymbol{x}_t)\tilde{\boldsymbol{\alpha}}_{t-1}), \quad (27)$$

где $\boldsymbol{q}_t = \dfrac{\boldsymbol{P}_{t-1}\tilde{\boldsymbol{\alpha}}_t}{1 + \tilde{\boldsymbol{a}}_t^{\mathrm{T}}\boldsymbol{P}_{t-1}\tilde{\boldsymbol{a}}_t} \equiv \boldsymbol{P}_t\tilde{\boldsymbol{a}}_t$, а $\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) = \tilde{\boldsymbol{K}}_t\tilde{\boldsymbol{a}}_t$ согласно (19).

2. $\delta_t > \nu$, т.е. для $\varphi(\boldsymbol{x}_t)$ не выполнено условие ALD.

Тогда $D_t = D_{t-1}\cup\{\boldsymbol{x}_t\}$, $m_t = m_{t-1} + 1$, а матрица $\tilde{\boldsymbol{K}}_t$ расширяется:

$$\tilde{\boldsymbol{K}}_t = \begin{bmatrix} \tilde{\boldsymbol{K}}_{t-1} & \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \\ \tilde{\boldsymbol{K}}_{t-1}^{T}(\boldsymbol{x}_t) & k_{tt} \end{bmatrix} \rightarrow$$
$$\rightarrow \tilde{\boldsymbol{K}}_t^{-1} = \frac{1}{\delta_t}\begin{bmatrix} \delta_t\tilde{\boldsymbol{K}}_{t-1}^{-1} + \tilde{\boldsymbol{a}}_t\tilde{\boldsymbol{a}}_t^{\mathrm{T}} & -\tilde{\boldsymbol{a}}_t \\ -\tilde{\boldsymbol{a}}_t^{\mathrm{T}} & 1 \end{bmatrix}, \quad (28)$$

где

$$\tilde{\boldsymbol{a}}_t = \tilde{\boldsymbol{K}}_{t-1}^{-1}\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \text{ и } \boldsymbol{A}_t = \begin{bmatrix} \boldsymbol{A}_{t-1} & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix},$$
$$\boldsymbol{A}_t^{\mathrm{T}}\boldsymbol{A}_t = \begin{bmatrix} \boldsymbol{A}_{t-1}^{\mathrm{T}}\boldsymbol{A}_{t-1} & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix}, \boldsymbol{P}_t = \begin{bmatrix} \boldsymbol{P}_{t-1} & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix}, \quad (29)$$

здесь 0 – нулевой вектор соответствующей размерности.

Выведем правило обновления для $\tilde{\boldsymbol{\alpha}}_t$:

**Таблица 3.** Выбор параметра ε на примере SOR-алгоритма

| ε | SV/BSV | StDev |
|---|---|---|
| 0,10 | 265/0 | 0,282 |
| 0,20 | 247/0 | 0,446 |
| **0,24** | **234/0** | **0,475** |
| 0,25 | 232/0 | 0,484 |

$$\tilde{\alpha}_t = \tilde{K}_t^{-1}(A_t^{\mathrm{T}}A_t)^{-1}A_t^{\mathrm{T}}y_t = \quad (30)$$
$$= \begin{bmatrix} \tilde{\alpha}_{t-1} - (\tilde{a}_t/\delta_t)(y_t - \tilde{k}_{t-1}^{\mathrm{T}}(x_t)\tilde{\alpha}_{t-1}) \\ (1/\delta_t)(y_t - \tilde{k}_{t-1}^{\mathrm{T}}(x_t)\tilde{\alpha}_{t-1}) \end{bmatrix},$$
$$\tilde{k}_{t-1}(x_t) = \tilde{K}_t\tilde{a}_t.$$

*Структура KRLS-алгоритма.*

I. Инициализация:

Задаем ν, вводим $(x_1, y_1)$ и определяем начальные значения

$$\tilde{K}_1 = [k_{11}],\ \tilde{K}_1^{-1} = [1/k_{11}],$$
$$\alpha_1 = (y_1/k_{11}),\ P_1 = [1],\ m = 1.$$

II. Для $t = 2,3\ldots$

1) вводим очередной образец $(x_t, y_t)$ и вычисляем

$$(\tilde{k}_{t-1}(x_t))_t = k(\tilde{x}_t, x_t), i = 1,\ldots,m;$$

2) проверяем условие ALD: по формулам (19) и (20) вычисляем $\tilde{a}_t$, $\delta_t$.

Если $\delta_t \le \nu$: $D_t = D_{t-1}$;

$$q_t = \frac{P_{t-1}\tilde{a}_t}{1 + \tilde{a}_t^{\mathrm{T}}P_{t-1}\tilde{a}_t};$$

по формуле (26) обновляем $P_t$;

по формуле (27) вычисляем $\tilde{\alpha}_t$.

Иначе: $D_t = D_{t-1}\cup\{x_t\}$;

по формуле (28) вычисляем $\tilde{K}_t^{t-1}$;

по формуле (29) вычисляем $P_t$;

по формуле (30) вычисляем $\tilde{\alpha}_t$;

$$m = m + 1.$$

Результат работы алгоритма после поступления нового образца $(x_t, y_t)$: $D_t$, $\tilde{\alpha}_t$.

## РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

Исследована сравнительная эффективность алгоритмов решения задачи восстановления регрессии SMO, SOR и IU, основанных на ε-нечувствительной функций потерь, и алгоритмов GSLS и KRLS, основанных на квадратичной функций потерь. Численные эксперименты проведены на эталонных данных, представленных двумя выборками различной природы: Boston housing (506 образцов, 13 признаков) и Prototask (5192 образцов, количество признаков $n = 8$) [20].

**Таблица 4.** Подбор параметра σ при фиксированном $C$

| σ | CPU-time | SV | Testing error RMSE $m = 100$ |
|---|---|---|---|
| 1,0 | 7,531 | 67 | 0,812 |
| 1,05 | 4,437 | 52 | 0,793 |
| 1,10 | 2,765 | 41 | 0,795 |
| **1,15** | **2,078** | **35** | **0,774** |
| 1,14 | 2,578 | 39 | 0,778 |
| 1,13 | 2,390 | 38 | 0,782 |
| 0,90 | 15,234 | 92 | 0,864 |

**Таблица 5.** Уточнение параметра σ при фиксированном $C$ при увеличении объема ТП

| $l$ | σ | CPU-time | SV/BSV | $m$ | Testing error MAE |
|---|---|---|---|---|---|
| 300 | 1,30 | 1,641 | 219/0 | 100 | 0,634 |
| 1000 | 1,30 | 92,625 | 750/0 | 300 | 0,638 |
| | **1,03** | **68,312** | 772/0 | | **0,614** |
| 2000 | 1,03 | 1194,312 | 1501/0 | 3192 | 0,594 |
| | 1,10 | 578,719 | 1503/0 | | 0,610 |
| | **1,07** | **661,484** | **1493/0** | | **0,599** |

Для оценки качества SVR-машин использованы следующие характеристики:

– средняя абсолютная ошибка (mean absolute error, MAE):

$$MAE = \frac{\sum_{i=1}^{m}|f(x_i) - y_i|}{m};$$

– корень из среднеквадратичной ошибки (root mean square error, RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m}(f(x_i) - y_i)^2}{m}},$$

где $m$ – объем тестовой последовательности; $(x_i, y_i)$ – ее образцы; $f(x)$ – SVR-машина, по-

**Таблица 6.** Характеристики SVR-алгоритмов

| Алгоритмы | C | σ | CPU time | SV/BSV | Testing error | |
|---|---|---|---|---|---|---|
| | | | | | RMSE | MAE |
| Prototask, $l$ = 2000, $m$ = 3192 | | | | | | |
| ε-нечувствительная функция потерь, ε = 0,24 | | | | | | |
| SMO | 100 | 1,07 | 661,484 | 1493/0 | – | 0,599 |
| SOR (ω = 0,1) | 100 | 1,1 | 12778,625 | 1507/0 | – | 0,600 |
| IU | 100 | 1,1 | 444,750 | 1503 / 0 | – | 0,580 |
| квадратичная функция потерь | | | | | | |
| GSLS (γ = 2C/$l$) | 1 | 1,21 | 263,453 | 34/– | 0,731 | – |
| KRLS | – | 7 | 4,656 | 134/– | 0,640 | – |
| Boston housing, $l$ = 400, $m$ = 106 | | | | | | |
| ε-нечувствительная функция потерь, ε = 0,1 | | | | | | |
| SMO | 100 | 1 | 6,422 | 274/0 | – | 0,320 |
| SOR (ω = 0,25) | 100 | 1,1 | 190,578 | 266/0 | – | 0,337 |
| IU | 100 | 1,14 | 2,406 | 266/0 | – | 0,338 |
| квадратичная функция потерь | | | | | | |
| GSLS (γ = 2C/$l$) | 50 | 0,86 | 7,391 | 52/– | 0,400 | – |
| KRLS | – | 12 | 0,172 | 70/– | 0,340 | – |

строенная по тренировочной последовательности.

Для оценки согласия экспериментальных данных с построенной SVR-машиной использована стандартная ошибка модели (StDev):

$$StDev = \sqrt{\frac{\sum_{i=1}^{m}(f(\boldsymbol{x}_i) - y_i)^2}{l - p}},$$

где $\{\boldsymbol{x}_i, y_i\}$, $i = 1,\ldots,l$ – тренировочная последовательность, $f(\boldsymbol{x})$ – построенная SVR-машина, $p$ – число свободных параметров машины.

Исследования проведены при следующих условиях:

данные стандартизованы;

использовано гауссово ядро;

подбор параметров регуляризации ($C$) и ядра (σ), т.е. настройка SVR-машины, выполнен на грубой решетке с последующим тщательным уточнением на участке, соответствующем лучшим значениям принятых оценок качества;

для настройки использована процедура 10-кратной перекрестной проверки.

В табл. 3–5 приведены результаты настройки SVR-машин (данные Prototask), где:

CPU-time, с – время, затраченное на построение SVR-машины;

SV – общее число опорных векторов;

BSV* – число связанных опорных векторов, для которых $\alpha_i = C$ [2];

Testing error – значения RMSE и MAE, полученные на тестовой последовательности.

Для алгоритмов SMO, SOR и IU, основанных на ε-нечувствительной функции потерь, предварительно было подобрано значение параметра ε. В табл. 3 приведены результаты выбора ε на примере SOR-алгоритма. Построение SVR-машины проведено по тренировочной последовательности объема $l$ = 300 при значениях параметров регуляризации и ядра, полученных с применением процедуры 10-кратной перекрестной проверки ($C$ = 100, σ = 1,1). Для ε = 0,24, отобранном для проведения дальнейших экспериментов, было вычислено значение MAE = 0,604 на основании процедуры 10-кратной перекрестной проверки.

Результаты численных экспериментов выявили высокую чувствительность всех алгоритмов к значению параметра ядра σ и показали, что для широкого диапазона значений параметра регуляризации $C$ качество SVR-машины практически не меняется. В табл. 4 приведены результаты тщательного подбора параметра σ

*Алгоритмы, основанные на квадратичной функций потерь, не выделяют связанные опорные векторы.

при фиксированном $C$ на примере GSLS-алгоритма. Построение SVR-машины проведено по тренировочной последовательности объема $l$ = 300 при значении параметра регуляризации $C$ = 600 для $\tau$ = 0,001. Качество SVR-машин проверено на тестовой последовательности объема $m$ = 100.

Исследования показали, что при увеличении объема тренировочной последовательности уточнение значения параметра $\sigma$ улучшает качество работы SVR-машины и существенно сокращает время, затраченное на ее построение. Табл. 5 содержит результаты «подстройки» параметра $\sigma$ на примере SMO-алгоритма при $\varepsilon$ = 0,24 для значения $C$ = 100 при увеличении объема $l$ тренировочной последовательности. Качество машин проверено на тестовой последовательности объема $m$.

При проведении численных экспериментов по исследованию сравнительной эффективности алгоритмов выборки были разделены на две части: тренировочную (объема $l$) и тестовую (объема $m$). Полученные результаты приведены в табл. 6.

Отметим основные преимущества рассмотренных алгоритмов:

IU-алгоритм является точным и самым быстрым среди SVR-алгоритмов, основанных на $\varepsilon$-нечувствительной функций потерь. Позволяет работать с данными, поступающими в online-режиме;

SMO – самый быстрый из итерационных алгоритмов. С увеличением объема тренировочной последовательности время работы алгоритма растет существенно медленнее, чем в реализациях других алгоритмов;

SOR-алгоритм при эффективной организации вычислительного процесса допускает проведение обучения на тренировочной последовательности огромных объемов. Предварительная настройка параметра $\omega$, регулирующего скорость сходимости, позволяет значимо сокращать время работы SOR-алгоритма;

GSLS – быстрый алгоритм, использующий квадратичную функцию потерь, отличается вычислительной простотой и способностью восстанавливать зависимость по небольшому числу опорных векторов;

KRLS – очень быстрый алгоритм, основанный на использовании квадратичной функции потерь. Требует настройки только параметров ядра. Позволяет работать с данными, поступающими в online-режиме.

Согласно результатам экспериментов, каждый алгоритм при работе с каждой выборкой требует тщательного подбора параметров. Все исследованные алгоритмы более чувствительны к значению параметра ядра $\sigma$, нежели к значению параметра регуляризации $C$. При настройке SVR-машины следует ориентироваться не только на показатели качества, но и на среднее время работы алгоритма: удачный выбор параметров приводит к существенному сокращению времени построения SVR-машины, не ухудшая при этом качество обучения.

## СПИСОК ЛИТЕРАТУРЫ


1. Н. О. Кадырова и Л. В. Павлова, Биофизика **59** (3), 446 (2014).
2. Н. О. Кадырова и Л. В. Павлова, Биофизика **60** (1), 18 (2015).
3. J. Kivinen, A. Smola, and R. Williamson, IEEE Transactions on Signal Processing **52** (8), 2165 (2004).
4. R. Collobert and S. Bengio, J. Machine Learning Res. **1**, 143 (2001).
5. G. Flake and S. Lawrence, Machine Learning **46** (1–3), 271 (2002).
6. G. Cawley and N. Talbot, *ICANN* (Springer-Verlag, 2002).
7. G. Cawley and N. Talbot, Neurocomputing **48**, 1025 (2002).
8. Y. Engel, S. Mannor, and R. Meir, in *The report of Interdisciplinary center for neural computation*, (2003), p. 34.
9. J. Platt, in *Advances in Neural Information Processing Systems* (MIT Press, 1999), Vol. 11, p. 557.
10. S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, Technical Report CD–99–14, P. 1 (1999).
11. S. Shevade, S. Keerthi, C. Bhattacharyya, and K. Murthy, IEEE Transactions on Neural Networks **11** (5), 1188 (2000).
12. O. Mangasarian and D. Musicant, IEEE Transactions on Neural Networks **10** (5), 1032 (1999).
13. Y. Quan, J. Yang, L.-X. Yao, and C.-Z. Ye, J. Software **15** (2), 200 (2004).
14. G. Cauwenberghs and T. Poggio, in *Advances in Neural Information Processing Systems* (MIT Press, 2001), Vol. 13, p. 409.
15. P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller, OAI-PMH server at eprints.pascal-network.org. (2005).
16. M. Martin, *ECML* (Springer-Verlag, 2002), p. 282.
17. G.H. Golub and C. F. van Loan, *Matrix Computations*, 3rd (John Hopkins University Press, Baltimore, London, 1996).
18. G. S. Kimeldorf and G. Wahba, J. Math. Anal. Applic. **33**, 82 (1971).
19. L. Sharf, *Statistical signal processing* (Addison-Wesley, 1991).
20. http://archive.ics.uci.edu/ml/datasets.html.

# Comparative Efficiency of Algorithms Based on Support Vector Machines for Regression

**N.O. Kadyrova and L.V. Pavlova**

*Institute of Applied Mathematics and Mechanics, St. Petersburg State Polytechnical University, ul. Polytechnicheskaya 29, St. Petersburg, 195251 Russia*

Methods of construction of support vector machines do not require additional a priori information and can be used to process large scale data set. It is especially important for various problems in computational biology. The main set of algorithms of support vector machines for regression is presented. The comparative efficiency of a number of support-vector-algorithms for regression is investigated. A thorough analysis of the study results found the most efficient support vector algorithms for regression. The description of the presented algorithms, sufficient for their practical implementation is given.

*Key words: regression, support vector machines, algorithms based on support vector machines for regression, comparative efficiency of support-vector-algorithms for regression*