Лекции по методу опорных векторов

К. В. Воронцов

21 декабря 2007 г.

Материал находится в стадии разработки, может содержать ошибки и неточности. Автор будет благодарен за любые замечания и предложения, направленные по адресу `voron@ccas.ru`. Перепечатка любых фрагментов данного материала без согласия автора является плагиатом.

Содержание

1 Метод опорных векторов (SVM)

Напомним основные обозначения.

Рассматривается задача обучения по прецедентам $\langle X, Y, y^*, X^\ell \rangle$, где X — пространство объектов, Y — множество ответов, $y^*\colon X \to Y$ — целевая зависимость, значения которой известны только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^\ell$, $y_i = y^*(x_i)$. Требуется построить алгоритм $a\colon X \to Y$, аппроксимирующий целевую зависимость на всём пространстве X.

§1.1 Метод опорных векторов в задачах классификации

Рассмотрим задачу классификации на два непересекающихся класса, в которой объекты описываются n-мерными вещественными векторами: $X = \mathbb{R}^n$, $Y = \{-1, +1\}$.

Будем строить линейный пороговый классификатор:

$$a(x) = \operatorname{sign}\left(\sum_{j=1}^{n} w_j x^j - w_0\right) = \operatorname{sign}\big(\langle w, x\rangle - w_0\big), \tag{1.1}$$

где $x = (x^1, \ldots, x^n)$ — признаковое описание объекта x; вектор $w = (w^1, \ldots, w^n) \in \mathbb{R}^n$ и скалярный порог $w_0 \in \mathbb{R}$ являются параметрами алгоритма.

Уравнение $\langle w, x\rangle = w_0$ описывает гиперплоскость, разделяющую классы в пространстве $\mathbb{R}^n$.

Хотя правило классификации в точности совпадает с моделью нейрона по МакКаллоку-Питтсу, критерий и методы настройки параметров в SVM радикально отличаются от персептронных (градиентных) методов обучения.

1.1.1 Понятие оптимальной разделяющей гиперплоскости

Предположим, что выборка линейно разделима, то есть существуют такие значения параметров w, w_0, при которых функционал числа ошибок

$$Q(w, w_0) = \sum_{i=1}^{\ell} \big[y_i(\langle w, x_i\rangle - w_0) < 0\big]$$

принимает нулевое значение. Но тогда разделяющая гиперплоскость не единственна, поскольку существуют и другие положения разделяющей гиперплоскости, реализующие то же самое разбиение выборки. Идея метода заключается в том, чтобы разумным образом распорядиться этой свободой выбора. Потребуем, чтобы разделяющая гиперплоскость максимально далеко отстояла от ближайших к ней точек обоих классов. Первоначально данный принцип классификации возник из эвристических соображений: вполне естественно полагать, что максимизация *зазора* (margin) между классами должна способствовать более уверенной классификации. В дальнейшем этот принцип получил мощное теоретическое обоснование [1, 8, 10].

Нормировка. Заметим, что параметры линейного порогового классификатора определены с точностью до нормировки: алгоритм $a(x)$ не изменится, если w и w_0 одновременно умножить на одну и ту же положительную константу. Удобно выбрать эту

константу таким образом, чтобы для всех пограничных (т. е. ближайших к разделяющей гиперплоскости) объектов x_i из X^ℓ выполнялись условия

$$\langle w, x_i \rangle - w_0 = y_i.$$

Сделать это возможно, поскольку при оптимальном положении разделяющей гиперплоскости все пограничные объекты находятся от неё на одинаковом расстоянии. Остальные объекты находятся дальше. Таким образом, для всех $x_i \in X^\ell$

$$\langle w, x_i \rangle - w_0 \begin{cases} \leqslant -1, & \text{если } y_i = -1; \\ \geqslant 1, & \text{если } y_i = +1; \end{cases} \tag{1.2}$$

Условие $-1 < \langle w, x \rangle - w_0 < 1$ задаёт полосу, разделяющую классы. Ни одна из точек обучающей выборки не может лежать внутри этой полосы. Границами полосы служат две параллельные гиперплоскости с направляющим вектором w. Точки, ближайшие к разделяющей гиперплоскости, лежат в точности на границах полосы. При этом сама разделяющая гиперплоскость проходит ровно по середине полосы.

Ширина разделяющей полосы. Чтобы разделяющая гиперплоскость как можно дальше отстояла от точек выборки, ширина полосы должна быть максимальной. Пусть x_- и x_+ — две произвольные точки классов -1 и $+1$ соответственно, лежащие на границе полосы. Тогда ширина полосы есть

$$\left\langle (x_+ - x_-), \frac{w}{\|w\|} \right\rangle = \frac{\langle w, x_+ \rangle - \langle w, x_- \rangle}{\|w\|} = \frac{(w_0 + 1) - (w_0 - 1)}{\|w\|} = \frac{2}{\|w\|}.$$

Ширина полосы максимальна, когда норма вектора w минимальна.

Итак, в случае, когда выборка линейно разделима, достаточно простые геометрические соображения приводят к следующей задаче: требуется найти такие значения параметров w и w_0, при которых норма вектора w минимальна при условии (1.2). Это задача квадратичного программирования. Она будет подробно рассмотрена в следующем разделе. Затем будет сделано обобщение на тот случай, когда линейной разделимости нет.

1.1.2 Линейно разделимая выборка

Построение оптимальной разделяющей гиперплоскости сводится к минимизации квадратичной формы при ℓ ограничениях-неравенствах вида (1.2) относительно $n+1$ переменных w, w_0:

$$\begin{cases} \langle w, w \rangle \to \min; \\ y_i\big(\langle w, x_i \rangle - w_0\big) \geqslant 1, \quad i = 1, \ldots, \ell. \end{cases} \tag{1.3}$$

По теореме Куна-Таккера эта задача эквивалентна двойственной задаче поиска седловой точки функции Лагранжа:

$$\begin{cases} \mathscr{L}(w, w_0; \lambda) = \dfrac{1}{2}\langle w, w \rangle - \displaystyle\sum_{i=1}^{\ell} \lambda_i \Big(y_i\big(\langle w, x_i \rangle - w_0\big) - 1 \Big) \to \min\limits_{w, w_0} \max\limits_{\lambda}; \\ \lambda_i \geqslant 0, \quad i = 1, \ldots, \ell; \\ \lambda_i = 0, \quad \text{либо} \quad \langle w, x_i \rangle - w_0 = y_i, \quad i = 1, \ldots, \ell; \end{cases}$$

где $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ — вектор двойственных переменных. Последнее из трёх условий называется *условием дополняющей нежёсткости*.

Необходимым условием седловой точки является равенство нулю производных Лагранжиана. Отсюда немедленно вытекают два полезных соотношения:

$$\frac{\partial \mathscr{L}}{\partial w} = w - \sum_{i=1}^{\ell} \lambda_i y_i x_i = 0 \qquad \Longrightarrow \qquad w = \sum_{i=1}^{\ell} \lambda_i y_i x_i; \tag{1.4}$$

$$\frac{\partial \mathscr{L}}{\partial w_0} = -\sum_{i=1}^{\ell} \lambda_i y_i = 0 \qquad \Longrightarrow \qquad \sum_{i=1}^{\ell} \lambda_i y_i = 0; \tag{1.5}$$

Из (1.4) следует, что искомый вектор весов w является линейной комбинацией векторов обучающей выборки, причём только тех, для которых $\lambda_i \neq 0$. Согласно условию дополняющей нежёсткости на этих векторах x_i ограничения-неравенства обращаются в равенства: $\langle w, x_i \rangle - w_0 = y_i$, следовательно, эти векторы находятся на границе разделяющей полосы. Все остальные векторы отстоят дальше от границы, для них $\lambda_i = 0$, и они не участвуют в сумме (1.4). Алгоритм (1.1) не изменился бы, если бы этих векторов вообще не было в обучающей выборке.

Опр. 1.1. *Если $\lambda_i > 0$ и $\langle w, x_i \rangle - w_0 = y_i$, то объект обучающей выборки x_i называется опорным вектором (support vector).*

Подставляя (1.4) и (1.5) обратно в Лагранжиан, получим эквивалентную задачу квадратичного программирования, содержащую только двойственные переменные:

$$\begin{cases} -\mathscr{L}(\lambda) = -\sum\limits_{i=1}^{\ell} \lambda_i + \dfrac{1}{2} \sum\limits_{i=1}^{\ell} \sum\limits_{j=1}^{\ell} \lambda_i \lambda_j y_i y_j (\langle x_i, x_j \rangle) \;\to\; \min\limits_{\lambda}; \\ \lambda_i \geqslant 0, \quad i = 1, \ldots, \ell; \\ \sum\limits_{i=1}^{\ell} \lambda_i y_i = 0. \end{cases} \tag{1.6}$$

Здесь минимизируется квадратичный функционал, имеющий неотрицательно определённую квадратичную форму, следовательно, выпуклый. Область, определяемая ограничениями неравенствами и одним равенством, также выпуклая. Следовательно, данная задача имеет единственное решение.

Допустим, мы решили эту задачу. Тогда вектор w вычисляется по формуле (1.4). Для определения порога w_0 достаточно взять произвольный опорный вектор x_i и выразить w_0 из равенства $w_0 = \langle w, x_i \rangle - y_i$. На практике для повышения численной устойчивости рекомендуется брать в качестве w_0 среднее по всем опорным векторам, а ещё лучше медиану:

$$w_0 = \operatorname{med}\big\{ \langle w, x_i \rangle - y_i \colon \lambda_i > 0,\ i = 1, \ldots, \ell \big\}. \tag{1.7}$$

В итоге алгоритм классификации может быть записан в следующем виде:

$$a(x) = \operatorname{sign}\left(\sum_{i=1}^{\ell} \lambda_i y_i \langle x_i, x \rangle - w_0 \right). \tag{1.8}$$

Обратим внимание, что реально суммирование идёт не по всей выборке, а только по опорным векторам, для которых $\lambda_i \neq 0$. Именно это свойство *разреженности* (sparsity) отличает SVM от других линейных разделителей — дискриминанта Фишера, логистической регрессии и однослойного персептрона.

Резюмируя, отметим, что пока остаются открытыми два вопроса: как быть, если классы линейно не разделимы, и как решить двойственную задачу (1.6)?

Начнём с первого вопроса. В следующем разделе рассматривается обобщение двойственной задачи на случай отсутствия линейной разделимости. После этого будет рассмотрен переход от скалярных произведений к произвольным ядрам — так называемый «kernel trick», позволяющий строить нелинейные разделители.

1.1.3 Линейно неразделимая выборка

Чтобы обобщить SVM на случай линейной неразделимости, позволим алгоритму допускать ошибки на обучающих объектах, но при этом постараемся, чтобы ошибок было поменьше. Введём набор дополнительных переменных $\xi_i \geqslant 0$, характеризующих величину ошибки на объектах x_i, $i = 1, \ldots, \ell$. Возьмём за отправную точку задачу (1.3); смягчим в ней ограничения-неравенства, и одновременно введём в минимизируемый функционал штраф за суммарную ошибку:

$$\begin{cases} \dfrac{1}{2}\langle w, w\rangle + C\sum\limits_{i=1}^{\ell}\xi_i \to \min\limits_{w, w_0, \xi}; \\ y_i\big(\langle w, x_i\rangle - w_0\big) \geqslant 1 - \xi_i, \quad i = 1, \ldots, \ell; \\ \xi_i \geqslant 0, \quad i = 1, \ldots, \ell. \end{cases} \tag{1.9}$$

К этой же оптимизационной задаче приводит ещё одна цепочка рассуждений. Вспомним, что в случае $Y = \{-1, +1\}$ *отступом* (margin) объекта x_i от границы классов называется величина

$$m_i = y_i\big(\langle w, x_i\rangle - w_0\big).$$

Алгоритм допускает ошибку на объекте x_i тогда и только тогда, когда отступ m_i отрицателен. Если $m_i \in (-1, +1)$, то объект x_i попадает внутрь разделяющей полосы. Если $m_i > 1$, то объект x_i классифицируется правильно, и находится на некотором удалении от разделяющей полосы. Запишем функционал числа ошибок алгоритма a на выборке X^ℓ в терминах отступов:

$$Q(a, X^\ell) = \sum_{i=1}^{\ell}[m_i < 0].$$

Заменим в этом функционале пороговую функцию потерь кусочно-линейной верхней оценкой: $[m_i < 0] \leqslant (1 - m_i)_+$, как показано на Рис. 1. Смысл этой замены в том, чтобы сделать функцию потерь чувствительной к величине ошибки и заодно ввести штраф за приближение объекта к границе классов.

Кроме того, добавим к функционалу Q штрафное слагаемое $\tau\|w\|^2$. В соответствии с принципом регуляризации некорректно поставленных задач по А. Н. Тихонову такая добавка означает, что среди всех векторов w, минимизирующих функционал Q, наиболее предпочтительны векторы с минимальной нормой. Регуляризация

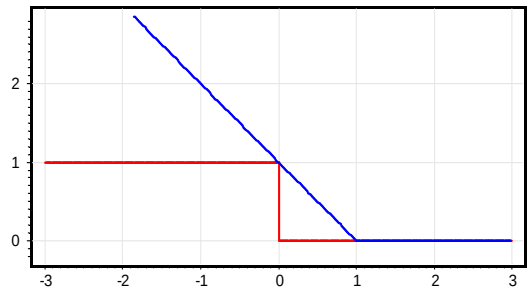

Рис. 1. Кусочно-линейная аппроксимация пороговой функции потерь: $[m_i < 0] \leqslant (1 - m_i)_+$.

часто применяется для настройки линейных моделей классификации и регрессии. При наличии шумовых и/или зависимых признаков она повышает устойчивость алгоритма по отношению к составу выборки и его обобщающую способность.

С учётом обеих модификаций функционал качества принимает вид:

$$Q(a, X^\ell) = \sum_{i=1}^{\ell} \big(1 - m_i\big)_+ + \tau \|w\|^2 \to \min_{w, w_0}. \tag{1.10}$$

Нетрудно показать, что задача минимизации данного функционала эквивалентна оптимизационной задаче с ограничениями (1.9), если взять параметр регуляризации $\tau = \frac{1}{2C}$. Таким образом, принцип оптимальной разделяющей гиперплоскости (или максимизации ширины разделяющей полосы) совпадает с принципом регуляризации по Тихонову.

Положительная константа C (или τ) является управляющим параметром метода и позволяет находить компромисс между максимизацией разделяющей полосы и минимизацией суммарной ошибки.

Вернёмся к задаче (1.9) и запишем её функцию Лагранжа:

$$\mathscr{L}(w, w_0, \xi; \lambda, \eta) = \frac{1}{2} \langle w, w \rangle - \sum_{i=1}^{\ell} \lambda_i \Big(y_i \big(\langle w, x_i \rangle - w_0 \big) - 1 \Big) - \sum_{i=1}^{\ell} \xi_i \big(\lambda_i + \eta_i - C \big),$$

где $\eta = (\eta_1, \ldots, \eta_\ell)$ — вектор переменных, двойственных к переменным $\xi = (\xi_1, \ldots, \xi_\ell)$. Как и в прошлый раз, условия Куна-Таккера сводят задачу к поиску седловой точки функции Лагранжа:

$$\begin{cases} \mathscr{L}(w, w_0, \xi; \lambda, \eta) \;\to\; \min\limits_{w, w_0, \xi} \max\limits_{\lambda, \eta}; \\ \xi_i \geqslant 0, \quad \lambda_i \geqslant 0, \quad \eta_i \geqslant 0, \quad i = 1, \ldots, \ell; \\ \lambda_i = 0, \quad \text{либо} \quad y_i \big(\langle w, x_i \rangle - w_0 \big) = 1 - \xi_i, \quad i = 1, \ldots, \ell; \\ \eta_i = 0, \quad \text{либо} \quad \xi_i = 0, \quad i = 1, \ldots, \ell; \end{cases}$$

В последних двух строках записаны условия дополняющей нежёсткости. Необходимым условием седловой точки является равенство нулю производных Лагран-

жиана. Отсюда получаются три полезных соотношения:

$$\frac{\partial \mathscr{L}}{\partial w} = w - \sum_{i=1}^{\ell} \lambda_i y_i x_i = 0 \qquad \Longrightarrow \qquad w = \sum_{i=1}^{\ell} \lambda_i y_i x_i; \tag{1.11}$$

$$\frac{\partial \mathscr{L}}{\partial w_0} = -\sum_{i=1}^{\ell} \lambda_i y_i = 0 \qquad \Longrightarrow \qquad \sum_{i=1}^{\ell} \lambda_i y_i = 0; \tag{1.12}$$

$$\frac{\partial \mathscr{L}}{\partial \xi_i} = -\lambda_i - \eta_i + C = 0 \qquad \Longrightarrow \qquad \eta_i + \lambda_i = C, \quad i = 1, \ldots, \ell. \tag{1.13}$$

Первые два соотношения в точности такие же, как и в линейно разделимом случае. Из третьего соотношения и неравенства $\eta_i \geqslant 0$ следует ограничение $\lambda_i \leqslant C$. Отсюда, и из условий дополняющей нежёсткости вытекает, что возможны только три допустимых сочетания значений переменных ξ_i, λ_i, η_i и отступов m_i.

Соответственно, все объекты x_i, $i = 1, \ldots, \ell$ делятся на следующие три типа:

1. $\lambda_i = 0$; $\eta_i = C$; $\xi_i = 0$; $m_i > 1$.

Объект x_i классифицируется правильно и находится далеко от разделяющей полосы. Такие объекты будем называть *периферийными*.

2. $0 < \lambda_i < C$; $0 < \eta_i < C$; $\xi_i = 0$; $m_i = 1$.

Объект x_i классифицируется правильно и лежит в точности на границе разделяющей полосы. Такие объекты, как и раньше, будем называть *опорными*.

3. $\lambda_i = C$; $\eta_i = 0$; $\xi_i > 0$; $m_i < 1$.

Объект x_i либо лежит внутри разделяющей полосы, но классифицируется правильно ($0 < \xi_i < 1$, $0 < m_i < 1$), либо попадает на границу классов ($\xi_i = 1$, $m_i = 0$), либо вообще относится к чужому классу ($\xi_i > 1$, $m_i < 0$). Во всех этих случаях объект x_i будем называть *нарушителем*.

В силу соотношения (1.13) в Лагранжиане обнуляются все члены, содержащие переменные ξ_i и η_i, и он принимает тот же вид, что и в случае линейной разделимости. Параметры разделяющей поверхности w и w_0, согласно формулам (1.11) и (1.12), также выражаются только через двойственные переменные λ_i. Таким образом, задача снова сводится к квадратичному программированию относительно двойственных переменных λ_i. Единственное отличие от линейно разделимого случая состоит в появлении ограничения сверху $\lambda_i \leqslant C$:

$$\begin{cases} -\mathscr{L}(\lambda) = -\sum\limits_{i=1}^{\ell} \lambda_i + \dfrac{1}{2} \sum\limits_{i=1}^{\ell} \sum\limits_{j=1}^{\ell} \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle \;\to\; \min\limits_{\lambda}; \\ 0 \leqslant \lambda_i \leqslant C, \quad i = 1, \ldots, \ell; \\ \sum\limits_{i=1}^{\ell} \lambda_i y_i = 0. \end{cases} \tag{1.14}$$

На практике для построения SVM решают именно эту задачу, а не (1.6), так как гарантировать линейную разделимость выборки в общем случае не представляется возможным. Этот вариант алгоритма называют *SVM с мягким зазором* (soft-margin SVM), тогда как в линейно разделимом случае говорят об *SVM с жёстким зазором* (hard-margin SVM).

Для алгоритма классификации сохраняется формула (1.8), с той лишь разницей, что теперь ненулевыми λ_i обладают не только опорные объекты, но и объекты-нарушители. В определённом смысле это недостаток SVM, поскольку нарушителями часто оказываются шумовые выбросы, и построенное на них решающее правило, по сути дела, опирается на шум.

Константу C обычно выбирают по критерию скользящего контроля. Это трудоёмкий способ, так как задачу приходится решать заново при каждом значении C.

Если есть основания полагать, что выборка почти линейно разделима, и лишь объекты-выбросы классифицируются неверно, то можно применить фильтрацию выбросов. Сначала задача решается при некотором C, и из выборки удаляется небольшая доля объектов, имеющих наибольшую величину ошибки ξ_i. После этого задача решается заново по усечённой выборке. Возможно, придётся проделать несколько таких итераций, пока оставшиеся объекты не окажутся линейно разделимыми.

1.1.4 Ядра и спрямляющие пространства

Существует ещё один подход к решению проблемы линейной неразделимости. Это переход от исходного пространства признаковых описаний объектов X к новому пространству H с помощью некоторого преобразования $\psi\colon X \to H$. Если пространство H имеет достаточно высокую размерность, то можно надеяться, что в нём выборка окажется линейно разделимой (легко показать, что если выборка X^ℓ не противоречива, то всегда найдётся пространство размерности не более ℓ, в котором она будет линейно разделима). Пространство H называют *спрямляющим*.

Если предположить, что признаковыми описаниями объектов являются векторы $\psi(x_i)$, а не векторы x_i, то построение SVM проводится точно так же, как и ранее. Единственное отличие состоит в том, что скалярное произведение $\langle x, x'\rangle$ в пространстве X всюду заменяется скалярным произведением $\langle \psi(x), \psi(x')\rangle$ в пространстве H. Отсюда вытекает естественное требование: пространство H должно быть наделено скалярным произведением, в частности, подойдёт любое евклидово, а в общем случае и гильбертово, пространство.

Опр. 1.2. *Функция $K\colon X \times X \to \mathbb{R}$ называется ядром (kernel function), если она представима в виде $K(x, x') = \langle \psi(x), \psi(x')\rangle$ при некотором отображении $\psi\colon X \to H$, где H — пространство со скалярным произведением.*

Постановка задачи (1.14), и сам алгоритм классификации (1.8) зависят только от скалярных произведений объектов, но не от самих признаковых описаний. Это означает, что скалярное произведение $\langle x, x'\rangle$ можно формально заменить ядром $K(x, x')$. Поскольку ядро в общем случае нелинейно, такая замена приводит к существенному расширению множества реализуемых алгоритмов $a\colon X \to Y$.

Более того, можно вообще не строить спрямляющее пространство H в явном виде, и вместо подбора отображения ψ заниматься непосредственно подбором ядра.

Можно пойти ещё дальше, и вовсе отказаться от признаковых описаний объектов. Во многих практических задачах объекты изначально задаются информацией об их попарном взаимоотношении, например, отношении сходства. Если эта информация допускает представление в виде двуместной функции $K(x, x')$, удовлетворяющей аксиомам скалярного произведения, то задача может решаться методом SVM. Для такого подхода недавно был придуман термин *беспризнаковое распознавание*

(featureless recognition), хотя многие давно известные метрические алгоритмы классификации (kNN, RBF и др.) также не требуют задания признаковых описаний.

Теорема Мерсера. Любая ли функция двух аргументов $K(x, x')$ может исполнять роль ядра? Следующая теорема даёт исчерпывающий ответ на этот вопрос и показывает, что класс допустимых ядер достаточно широк.

Теорема 1.1 (Мерсер, 1909 [4]). *Функция $K(x, x')$ является ядром тогда и только тогда, когда она симметрична, $K(x, x') = K(x', x)$, и неотрицательно определена: $\int_X \int_X K(x, x')g(x)g(x')dxdx' \geqslant 0$ для любой функции $g\colon X \to \mathbb{R}$.*

Существует эквивалентное определение неотрицательной определённости.

Опр. 1.3. *Функция $K(x, x')$ неотрицательно определена, если для любой конечной выборки $X^p = (x_1, \ldots, x_p)$ из X матрица $K = \|K(x_i, x_j)\|$ размера $p \times p$ неотрицательно определена: $z^\intercal K z \geqslant 0$ для любого $z \in \mathbb{R}^p$.*

Проверка неотрицательной определённости функции в практических ситуациях может оказаться делом нетривиальным. Часто ограничиваются перебором конечного числа функций, про которые известно, что они являются ядрами. Среди них выбирается лучшая, как правило, по критерию скользящего контроля. Очевидно, что это не оптимальное решение. На сегодняшний день проблема выбора ядра, оптимального для данной конкретной задачи, остаётся открытой.

Конструктивные способы построения ядер. Следующие правила порождения позволяют строить ядра в практических задачах.

1. Произвольное скалярное произведение $K(x, x') = \langle x, x' \rangle$ является ядром.
2. Константа $K(x, x') = 1$ является ядром.
3. Произведение ядер $K(x, x') = K_1(x, x')K_2(x, x')$ является ядром.
4. Для любой функции $\psi : X \to \mathbb{R}$ произведение $K(x, x') = \psi(x)\psi(x')$ является ядром.
5. Линейная комбинация ядер с неотрицательными коэффициентами $K(x, x') = = \alpha_1 K_1(x, x') + \alpha_2 K_2(x, x')$ является ядром.
6. Композиция произвольной функции $\varphi : X \to X$ и произвольного ядра K_0 является ядром: $K(x, x') = K_0(\varphi(x), \varphi(x'))$.
7. Если $s\colon X \times X \to \mathbb{R}$ — произвольная симметричная интегрируемая функция, то $K(x, x') = \int_X s(x, z)s(x', z)\, dz$ является ядром.
8. Функция вида $K(x, x') = k(x - x')$ является ядром тогда и только тогда, когда Фурье-образ $F[k](\omega) = (2\pi)^{\frac{n}{2}} \int_X e^{-i\langle \omega, x \rangle} k(x)\, dx$ неотрицателен.
9. Предел локально-равномерно сходящейся последовательности ядер является ядром.

10. Композиция произвольного ядра K_0 и произвольной функции $f\colon \mathbb{R} \to \mathbb{R}$, представимой в виде сходящегося степенного ряда с неотрицательными коэффициентами $K(x, x') = f\big(K_0(x, x')\big)$, является ядром. В частности, функции $f(z) = e^z$ и $f(z) = \frac{1}{1-z}$ от ядра являются ядрами.

Примеры ядер. Существует несколько «стандартных» ядер, которые при ближайшем рассмотрении приводят к уже известным алгоритмам: полиномиальным разделяющим поверхностям, двухслойным нейронным сетям, потенциальным функциям (RBF-сетям), и другим. Таким образом, ядра претендуют на роль универсального языка для описания широкого класса алгоритмов обучения по прецедентам. Наблюдается парадоксальная ситуация. С одной стороны, ядра — одно из самых красивых и плодотворных изобретений в машинном обучении. С другой стороны, до сих пор не найдено эффективного общего подхода к их подбору в конкретных задачах.

Пример 1.1. Пусть $X = \mathbb{R}^2$. Рассмотрим ядро $K(u, v) = \langle u, v\rangle^2$, где $u = (u_1, u_2)$, $v = (v_1, v_2)$, и попробуем понять, какое спрямляющее пространство и преобразование ψ ему соответствуют. Разложим квадрат скалярного произведения:

$$\begin{aligned} K(u, v) = \langle u, v\rangle^2 &= \langle (u_1, u_2), (v_1, v_2)\rangle^2 = \\ &= (u_1 v_1 + u_2 v_2)^2 = u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 u_2 v_2 = \\ &= \Big\langle (u_1^2, u_2^2, \sqrt{2}u_1 u_2), (v_1^2, v_2^2, \sqrt{2}v_1 v_2)\Big\rangle . \end{aligned}$$

Ядро K представляется в виде скалярного произведения в пространстве $H = \mathbb{R}^3$. Преобразование $\psi\colon \mathbb{R}^2 \to \mathbb{R}^3$ имеет вид $\psi\colon (u_1, u_2) \mapsto (u_1^2, u_2^2, \sqrt{2}u_1 u_2)$. Линейной поверхности в пространстве H соответствует квадратичная поверхность в исходном пространстве X. В частности, данное ядро позволяет линейно разделить внутреннюю и наружную часть эллипса, что невозможно в исходном двумерном пространстве.

Пример 1.2. Усложним ситуацию. Пусть теперь $X = \mathbb{R}^n$,

$$K(u, v) = \langle u, v\rangle^d .$$

Тогда компонентами вектора $\psi(u)$ являются различные произведения $(u_1)^{d_1} \cdots (u_n)^{d_n}$ при всевозможных целых неотрицательных $d_1, \ldots, d_n$, удовлетворяющих условию $d_1 + \cdots + d_n = d$. Число таких мономов, а следовательно и размерность пространства H, равно C^d_{n+d-1}. Пространство H изоморфно пространству всех полиномов, состоящих из мономов степени d от переменных $u_1, \ldots, u_n$.

Пример 1.3. Если $X = \mathbb{R}^n$,

$$K(u, v) = \big(\langle u, v\rangle + 1\big)^d,$$

то H — пространство всех мономов степени *не выше* d от переменных $u_1, \ldots, u_n$. В этом случае пространство H изоморфно пространству всех полиномов степени d. Линейная разделимость множеств в этом пространстве эквивалентна полиномиальной разделимости множеств в исходном пространстве X.

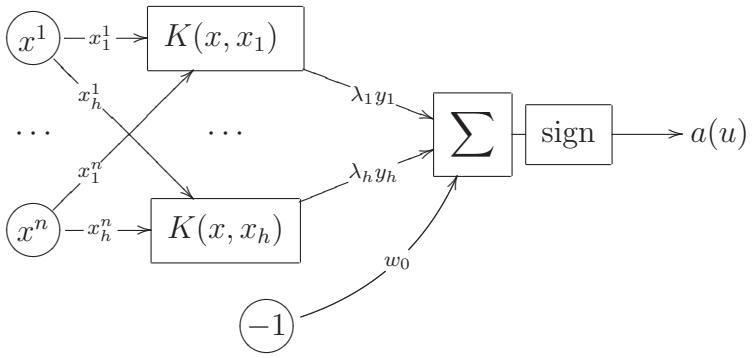

Рис. 2. Машина опорных векторов (SVM) как двухслойная нейросеть.

Связь SVM с двухслойными нейронными сетями. Рассмотрим структуру алгоритма $a(x)$ после замены в (1.8) скалярного произведения $\langle x_i, x\rangle$ ядром $K(x_i, x)$. Перенумеруем объекты так, чтобы первые h объектов оказались опорными. Поскольку $\lambda_i = 0$ для всех неопорных объектов, $i = h+1, \ldots, \ell$, алгоритм $a(x)$ примет вид

$$a(x) = \operatorname{sign}\left(\sum_{i=1}^{h} \lambda_i y_i K(x_i, x) - w_0\right).$$

Если $X = \mathbb{R}^n$, то алгоритм $a(x)$ можно рассматривать как двухслойную нейронную сеть, имеющую n входных нейронов и h нейронов в скрытом слое, см. Рис. 2. Сеть, настроенная методом SVM, имеет несколько замечательных особенностей.

Во-первых, число нейронов скрытого слоя определяется автоматически.

Во-вторых, векторы весов у нейронов скрытого слоя совпадают с признаковыми описаниями опорных объектов.

В-третьих, проясняется смысл двойственных переменных: λ_i — это вес, выражающий степень важности ядра $K(x_i, x)$.

Пример 1.4. Классическая нейронная сеть с сигмоидными функциями активации получится, если в качестве ядра взять функцию

$$K(u, v) = \operatorname{th}\big(k_0 + k_1 \langle u, v\rangle\big).$$

Данная функция удовлетворяет условиям Мерсера не при всех значениях параметров k_0 и k_1. В частности, она им не удовлетворяет при $k_0 < 0$ или $k_1 < 0$ [2]. Однако это не препятствует её успешному практическому применению. Вместо гиперболического тангенса $\operatorname{th} z$ часто используют также логистическую функцию $\sigma(z) = \frac{1}{1+e^{-z}}$.

Что плохого произойдёт, если функция $K(u, v)$ не будет удовлетворять условиям Мерсера? Постановка задачи квадратичного программирования (1.14) останется той же и в этом случае. Однако квадратичная форма утратит свойство неотрицательной определённости, минимизируемый функционал уже не будет выпуклым, и решение может оказаться не единственным. Самое неприятное то, что на границах гиперпараллелепипеда $0 \leqslant \lambda_i \leqslant C$ возникнет огромное количество локальных минимумов, и поиск решения среди них в общем случае потребует полного перебора. В этой ситуации многие методы квадратичного программирования будут выдавать какой-то локальный минимум, совсем не обязательно хороший.

Пример 1.5. Нейронная сеть с *радиальными базисными функциями* (radial basis functions, RBF) получится, если взять гауссовское ядро

$$K(u,v) = \exp\bigl(-\beta\|u-v\|^2\bigr),$$

где β — параметр. Ядро $K(x_i,x)$ вычисляет оценку близости объекта x к опорному объекту x_i. Чем ближе объекты, тем больше значение ядра. Выходной нейрон складывает все эти оценки, умножая их на коэффициенты $\lambda_i y_i$. При этом близости к опорным объектам класса $+1$ суммируются с положительными весами, а к объектам класса -1 — с отрицательными. Выходной нейрон производит голосование, сравнивая суммарные близости распознаваемого объекта x к обоим классам.

В разделе **??** рассматривался альтернативный метод обучения RBF-сетей, основанный на EM-алгоритме. Тогда гауссовские ядра играли роль компонент смеси вероятностных распределений. Центры ядер размещались не в опорных объектах, а в местах локальных сгущений плотности объектов. В этом и заключается основное отличие SVM-RBF от EM-RBF. Метод SVM сдвигает центры гауссианов ближе к границе классов, в результате форма разделяющей поверхности описывается более чётко. Таким образом, SVM-RBF лучше подходит для описания классов с границами сложной формы. С другой стороны, EM-RBF более устойчив к выбросам и предпочтителен в задачах с «размытыми» границами классов.

Преимущества SVM перед методом стохастического градиента.

- Вместо многоэкстремальной задачи решается задача квадратичного программирования, имеющая единственное решение. Методы оптимизации в этом случае существенно более эффективны.
- Автоматически определяется число нейронов скрытого слоя. Оно равно числу опорных векторов.
- Принцип оптимальной разделяющей гиперплоскости приводит к максимизации ширины разделяющей полосы между классами, следовательно, к более уверенной классификации. Градиентные нейросетевые методы выбирают положение разделяющей гиперплоскости произвольным образом, «как придётся».

Недостатки SVM.

- Метод опорных векторов неустойчив по отношению к шуму в исходных данных. Если обучающая выборка содержат шумовые выбросы, они будут существенным образом учтены при построении разделяющей гиперплоскости. Этого недостатка лишён *метод релевантных векторов* (relevance vector machine, RVM), см. **??**.
- До сих пор не разработаны общие методы построения спрямляющих пространств или ядер, наиболее подходящих для конкретной задачи. Построение адекватного ядра является искусством и, как правило, опирается на априорные знания о предметной области. На практике «вполне разумные» функции $K(x,x')$, выведенные из содержательных соображений, далеко не всегда оказываются положительно определёнными.

- В общем случае, когда линейная разделимость не гарантируется, приходится подбирать управляющий параметр алгоритма C.

1.1.5 Алгоритм настройки SVM

Двойственная задача (1.14) является задачей квадратичного программирования. Общие методы решения таких задач известны, но довольно трудоёмки, как в смысле реализации, так и по времени выполнения. Поэтому для обучения SVM применяются алгоритмы, учитывающие специфические особенности SVM. Специфика заключается в том, что число опорных векторов h, как правило, невелико, $h \ll \ell$, и эти векторы находятся поблизости от границы классов. Именно эти особенности и позволяют ускорить поиск опорных объектов. Специализированные алгоритмы настройки SVM успешно справляются с выборками из десятков тысяч объектов [5, 6].

Здесь мы рассмотрим не самый известный, но также довольно эффективный алгоритм — *последовательный метод активных ограничений* (incremental active set method, INCAS), предложенный в [3, 7].

Перепишем двойственную задачу (1.14) в матричных обозначениях. Введём матрицу $Q = \big(y_i y_j K(x_i, x_j)\big)_{i=1,\ell}^{j=1,\ell}$ размера $\ell \times \ell$ и три вектор-столбца длины ℓ: вектор ответов $y = (y_i)_{i=1,\ell}$, вектор двойственных переменных $\lambda = (\lambda_i)_{i=1,\ell}$ и вектор единиц $e = (1)_{i=1,\ell}$. Тогда задачу (1.14) можно переписать в виде

$$\begin{cases} \frac{1}{2}\lambda^\intercal Q\lambda - e^\intercal\lambda \to \min\limits_{\lambda}; \\ y^\intercal\lambda = 0; \\ 0 \leqslant \lambda \leqslant Ce. \end{cases} \tag{1.15}$$

Допустим, что решение λ ещё не известно, но зато известно разбиение множества объектов на три непересекающихся подмножества $\{1, \ldots, \ell\} = I_O \cup I_C \cup I_S$:

$$\begin{aligned} I_O &= \{i\colon \lambda_i = 0\} && \text{— периферийные объекты, } m_i > 1; \\ I_S &= \{i\colon 0 < \lambda_i < C\} && \text{— опорные объекты, } m_i = 1; \\ I_C &= \{i\colon \lambda_i = C\} && \text{— объекты-нарушители, } m_i < 1; \end{aligned}$$

где $m_i = y_i\big(\langle w, x_i\rangle - w_0\big)$ — отступ объекта x_i от границы классов.

Ограничения-неравенства $0 \leqslant \lambda_i \leqslant C$ становятся *активными*, то есть обращаются в равенства, на периферийных объектах ($i \in I_O$, $\lambda_i = 0$) и объектах-нарушителях ($i \in I_C$, $\lambda_i = C$). Соответствующие значения λ_i можно подставить обратно в (1.15) и получить оптимизационную задачу, зависящую только от части переменных λ_i, $i \in I_S$. Чтобы выписать эту задачу в матричном виде, введём трёхблочные обозначения для матрицы Q и векторов y, e, λ:

$$Q = \begin{pmatrix} Q_{SS} & Q_{SO} & Q_{SC} \\ Q_{OS} & Q_{OO} & Q_{OC} \\ Q_{CS} & Q_{CO} & Q_{CC} \end{pmatrix}; \quad y = \begin{pmatrix} y_S \\ y_O \\ y_C \end{pmatrix}; \quad e = \begin{pmatrix} e_S \\ e_O \\ e_C \end{pmatrix}; \quad \lambda = \begin{pmatrix} \lambda_S \\ 0 \\ Ce_C \end{pmatrix}.$$

В этих обозначениях задача (1.15) принимает вид

$$\begin{cases} \frac{1}{2}\lambda_S^\intercal Q_{SS}\lambda_S + Ce_C^\intercal Q_{CS}\lambda_S - e_S^\intercal\lambda_S \to \min\limits_{\lambda_S}; \\ y_S^\intercal\lambda_S + Ce_C^\intercal y_C = 0; \end{cases} \tag{1.16}$$

Если не обращать внимания на ограничения-неравенства, то это задача минимизации квадратичного функционала от $h = |I_S|$ переменных с одним линейным ограничением типа равенства. Она легко решается стандартными методами линейной алгебры и сводится, фактически, к обращению симметричной положительно определённой матрицы Q_{SS} размера $h \times h$.

Решение этой задачи даёт весь вектор λ, что позволяет вычислить параметры алгоритма w и w_0 по формулам (1.4) и (1.7). Теперь можно классифицировать объекты выборки x_i, вычислить отступы m_i, и проверить, правильно ли множество объектов было разбито на подмножества $I_O \cup I_C \cup I_S$. Если условия Куна-Таккера не нарушатся ни на одном объекте, значит, решение задачи (1.15) найдено. Если же обнаружится хотя бы одно противоречие, то соответствующий объект должен быть переведён из одного подмножества в другое.

Всего возможны четыре типа противоречий:

1. Если $i \in I_S$ и $\lambda_i \leqslant 0$, то объект x_i переводится из I_S в I_O.
2. Если $i \in I_S$ и $\lambda_i \geqslant C$, то объект x_i переводится из I_S в I_C.
3. Если $i \in I_O$ и $m_i \leqslant 1$, то объект x_i переводится из I_O в I_S.
4. Если $i \in I_C$ и $m_i \geqslant 1$, то объект x_i переводится из I_C в I_S.

После каждой модификации множеств I_O, I_C, I_S задача (1.16) решается заново. Итерационный процесс перевода объектов из одного множества в другое продолжается до тех пор, пока все объекты не будут удовлетворять условиям Куна-Таккера.

Описанный процесс является частным случаем *метода активных ограничений* (active sets method), который применяется для решения произвольных задач математического программирования с ограничениями-неравенствами. В линейном программировании он эквивалентен симплекс-методу. Сходимость данного метода в общем случае не гарантируется, однако в задачах настройки SVM зацикливания случаются крайне редко и легко предотвращаются с помощью нескольких удачных эвристик [3].

Неплохая эвристика заключается в том, чтобы на каждом шаге выбирать объект i, для которого условие Куна-Таккера нарушается сильнее всего. Это способствует увеличению скорости сходимости.

Начальное приближение. Количество итераций существенно зависит от того, насколько удачным окажется начальное приближение. На практике применяются различные приёмы, чтобы сразу поточнее «угадать» множество опорных векторов I_S.

Приём 1. Выбирается произвольная точка выборки, и находится ближайшая к ней точка другого класса. Для неё, в свою очередь, находится ближайшая точка в первом классе, и т. д. Этот итерационный процесс, как правило, сходится очень быстро к некоторой паре пограничных точек. В Алгоритме 1.1 эта пара принимается за начальное приближение I_S, но можно и продолжить процесс, построив несколько или даже все такие пары.

Приём 2. Строится несколько достаточно грубых линейных классификаторов. Для этого можно использовать однослойные персептроны, проводя небольшое число итераций методом стохастического градиента. Затем для каждой линейной разделяющей поверхности находится несколько ближайших к ней точек.

Алгоритм 1.1. Обучение SVM: последовательный метод активных ограничений

Вход:
X^ℓ — обучающая выборка;
C — параметр двойственной задачи;
Выход:
параметры линейного классификатора w, w_0;

1: начальное приближение:
$I_S =$ две ближайшие точки из разных классов;
$I_O =$ все остальные точки;
$I_C = \varnothing$;
2: **повторять**
3: **повторять**
4: решение оптимизационной задачи относительно λ_S:

$$\begin{cases} \frac{1}{2}\lambda_S^\intercal Q_{SS}\lambda_S + Ce_C^\intercal Q_{CS}\lambda_S - e_S^\intercal\lambda_S \to \min\limits_{\lambda_S}; \\ y_S^\intercal\lambda_S + Ce_C^\intercal y_C = 0; \end{cases}$$

5: **если** $|I_S| > 2$ и $\exists i$: $i \in I_S$ и $\lambda_i \leqslant 0$ **то** перевести i в I_O;
6: **если** $|I_S| > 2$ и $\exists i$: $i \in I_S$ и $\lambda_i \geqslant C$ **то** перевести i в I_C;
7: **пока** существует $i \in I_S$, который необходимо перевести в I_O или в I_C;
8: вычислить параметры алгоритма w и w_0 по формулам (1.4) и (1.7);
9: вычислить отступы m_i, $i \in I_0 \cup I_C$;
10: **если** $\exists i$: $i \in I_O$ и $m_i \leqslant 1$ **то** перевести i в I_S;
11: **если** $\exists i$: $i \in I_C$ и $m_i \geqslant 1$ **то** перевести i в I_S;
12: **пока** существует $i \in I_O \cup I_C$, который необходимо перевести в I_S;

Эффективность. Высокая эффективность Алгоритма 1.1 вытекает из двух фактов.

Во-первых, оптимизационная задача, решаемая на шаге 4, зависит только от матриц Q_{SS} и Q_{CS}. Значит, вычислять скалярные произведения $K(x, x')$ приходится только для пар объектов типа «опорный–опорный» и «опорный–нарушитель».

Во-вторых, множество I_S на каждом шаге изменяется только на один элемент. Это позволяет выполнять пересчёт обратной матрицы Q_{SS}^{-1} за $O(h^2)$ операций, тогда как обычное обращение потребовало бы $O(h^3)$ операций.

Преимущества метода INCAS.

- Метод позволяет решать задачи, в которых нет линейной разделимости, в том числе задачи с шумовыми выбросами.
- Метод особенно эффективен, когда число опорных векторов $h = |I_S|$ невелико.
- Метод хорошо приспособлен для решения таких задач, в которых обучающие объекты поступают по одному в режиме реального времени. Добавление нового объекта реализуется практически так же, как перевод объекта из одного подмножества в другое, и требует порядка $O(h^2)$ операций.

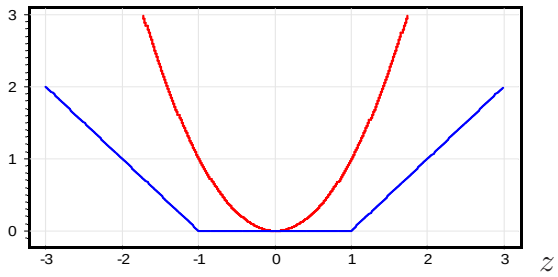

Рис. 3. Функции потерь в задачах регрессии: кусочно-линейная $|z|_\varepsilon$ при $\varepsilon = 1$ и квадратичная z^2.

Недостатки метода INCAS.

- Метод становится неэффективен, когда число опорных векторов велико. В то же время, это означает, что либо выборка существенно неразделима, либо ядро $K(x, x')$ выбрано неудачно. В обоих случаях такую задачу, скорее всего, нет смысла решать — надо менять ядро или саму постановку.

§1.2 Метод опорных векторов в задачах регрессии

В главе **??** мы уже рассматривали задачи многомерной линейной регрессии, предполагая, что $X = \mathbb{R}^n$, $Y = \mathbb{R}$, алгоритм имеет вид $a(x) = \langle w, x\rangle - w_0$, и для настройки параметров $w \in \mathbb{R}^n$ и $w_0 \in \mathbb{R}$ минимизируется квадратичный функционал. В случае гребневой регрессии (см. раздел **??**) вводится ещё и штрафное слагаемое, предотвращающее бесконтрольное увеличение коэффициентов w:

$$Q(a, X^\ell) = \sum_{i=1}^{\ell} \big(\langle w, x_i\rangle - w_0 - y_i\big)^2 + \tau\|w\|^2 \to \min_{w, w_0},$$

где τ — параметр регуляризации. Выбор именно квадратичной функции потерь обусловлен удобством решения задачи наименьших квадратов.

Однако в некоторых случаях более естественно использовать кусочно-линейную *функцию ε-чувствительности*, показанную на Рис 3: $|z|_\varepsilon = \max\{0, |z| - \varepsilon\}$, которая не считает за ошибки отклонения $a(x_i)$ от y_i, меньшие ε. Предполагается, что значение параметра ε задаёт эксперт, исходя из априорных соображений.

С этой функцией потерь функционал принимает вид

$$Q_\varepsilon(a, X^\ell) = \sum_{i=1}^{\ell} \big|\langle w, x_i\rangle - w_0 - y_i\big|_\varepsilon + \tau\,\langle w, w\rangle^2 \to \min_{w, w_0}. \tag{1.17}$$

Легко обнаруживается сходство данной задачи с задачей классификации (1.10). Покажем, что минимизация (1.17) эквивалентна некоторой задаче квадратичного программирования с линейными ограничениями типа неравенств. При этом также возникает двойственная задача, зависящая только от двойственных переменных; также достаточно оставить в выборке только опорные объекты; также решение выражается через скалярные произведения объектов, а не сами объекты; и также можно использовать ядра. Иными словами, SVM-регрессия отличается от SVM-классификации только в технических деталях, основные идеи остаются теми же.

Положим $C = \frac{1}{2\tau}$. Введём дополнительные переменные ξ_i^+ и ξ_i^-, значения которых равны потере при завышенном и заниженном ответе $a(x_i)$ соответственно:

$$\xi_i^+ = (a(x_i) - y_i - \varepsilon)_+, \quad \xi_i^- = (-a(x_i) + y_i - \varepsilon)_+, \quad i = 1, \ldots, \ell.$$

Тогда задача минимизации (1.17) может быть переписана в эквивалентной форме как задача квадратичного программирования с линейными ограничениями-неравенствами относительно переменных w_i, w_0, ξ_i^+ и ξ_i^-:

$$\begin{cases} \dfrac{1}{2}\langle w, w\rangle + C\sum\limits_{i=1}^{\ell}(\xi_i^+ + \xi_i^-) \to \min\limits_{w,w_0,\xi^+,\xi^-}; \\ y_i - \varepsilon - \xi_i^- \leqslant \langle w, x_i\rangle - w_0 \leqslant y_i + \varepsilon + \xi_i^+, \quad i = 1,\ldots,\ell; \\ \xi_i^- \geqslant 0, \quad \xi_i^+ \geqslant 0, \quad i = 1,\ldots,\ell. \end{cases} \tag{1.18}$$

Как и в предыдущих случаях, лагранжиан этой задачи выражается через двойственные переменные λ_i^+, λ_i^-, $i = 1,\ldots,\ell$, а скалярные произведения $\langle x_i, x_j\rangle$ можно заменить ядром $K(x_i, x_j)$. Опуская выкладки, представим результат:

$$\begin{cases} \mathscr{L}(\lambda^+,\lambda^-) = -\varepsilon\sum_{i=1}^{\ell}(\lambda_i^- + \lambda_i^+) + \sum_{i=1}^{\ell}(\lambda_i^- - \lambda_i^+)y_i - \\ \qquad\qquad - \frac{1}{2}\sum_{i,j=1}^{\ell}(\lambda_i^- - \lambda_i^+)(\lambda_j^- - \lambda_j^+)K(x_i,x_j) \;\to \max\limits_{\lambda^+,\lambda^-}; \\ 0 \leqslant \lambda_i^+ \leqslant C, \quad 0 \leqslant \lambda_i^- \leqslant C, \quad i = 1,\ldots,\ell; \\ \sum_{i=1}^{\ell}(\lambda_i^- + \lambda_i^+) = 0. \end{cases}$$

Все объекты x_i, $i = 1,\ldots,\ell$ делятся на следующие пять типов:

1. $|a(x_i) - y_i| < \varepsilon$; $\lambda_i^+ = \lambda_i^- = \xi_i^+ = \xi_i^- = 0$.

Ответ алгоритма $a(x_i)$ находится внутри отрезка $[y_i - \varepsilon, y_i + \varepsilon]$ и считается верным. Объект x_i не является опорным — вектор весов w не изменился бы, если бы этого объекта изначально не было в выборке.

2. $a(x_i) = y_i + \varepsilon$; $0 < \lambda_i^+ < C$; $\lambda_i^- = 0$; $\xi_i^+ = \xi_i^- = 0$.
3. $a(x_i) = y_i - \varepsilon$; $0 < \lambda_i^- < C$; $\lambda_i^+ = 0$; $\xi_i^+ = \xi_i^- = 0$.
4. $a(x_i) > y_i + \varepsilon$; $\lambda_i^+ = C$; $\lambda_i^- = 0$; $\xi_i^+ = a(x_i) - y_i - \varepsilon > 0$; $\xi_i^- = 0$.
5. $a(x_i) < y_i - \varepsilon$; $\lambda_i^- = C$; $\lambda_i^+ = 0$; $\xi_i^- = y_i - a(x_i) - \varepsilon > 0$; $\xi_i^+ = 0$.

Объекты типов 2–5 являются опорными и учитываются при определении вектора весов. При этом только на объектах типов 4 и 5 возникает ненулевая ошибка.

Уравнение регрессии также выражается через двойственные переменные:

$$a(x) = \sum_{i=1}^{\ell}(\lambda_i^- - \lambda_i^+)K(x_i, x) - w_0;$$

где параметр w_0 определяется из ограничений-неравенств, которые становятся равенствами на опорных объектах типа 2 и 3:

$$\langle w, x_i\rangle - w_0 = \begin{cases} y_i + \varepsilon, & \text{если } x_i \text{ — объект типа 2;} \\ y_i - \varepsilon, & \text{если } x_i \text{ — объект типа 3.} \end{cases}$$

Как и раньше, чтобы избежать численной неустойчивости, имеет смысл взять медиану множества значений w_0, вычисленных по всем опорным векторам.

В этом методе есть два управляющих параметра. Параметр точности ε задаётся из априорных соображений. Параметр регуляризации C подбирается, как правило, по скользящему контролю, что является вычислительно трудоёмкой процедурой.

Более обстоятельное изложение многочисленных особенностей SVM-регрессии можно найти в руководстве [9].

Список литературы

[1] *Bartlett P., Shawe-Taylor J.* Generalization performance of support vector machines and other pattern classifiers // Advances in Kernel Methods. — MIT Press, Cambridge, USA, 1998.
http://citeseer.ist.psu.edu/bartlett98generalization.html.

[2] *Burges C. J. C.* Geometry and invariance in kernel based methods // Advances in Kernel Methods / Ed. by B. Scholkopf, C. C. Burges, A. J. Smola. — MIT Press, 1999. — Pp. 89 – 116.

[3] *Fine S., Scheinberg K.* INCAS: An incremental active set method for SVM: Tech. rep.: 2002.
http://citeseer.ist.psu.edu/fine02incas.html.

[4] *Mercer J.* Functions of positive and negative type and their connection with the theory of integral equations // *Philos. Trans. Roy. Soc. London.* — 1909. — Vol. A, no. 209. — Pp. 415–446.

[5] *Osuna E., Freund R., Girosi F.* An improved training algorithm for support vector machines // Neural Networks for Signal Processing VII. IEEE Workshop. — 1997. — Pp. 276–285.
http://citeseer.ist.psu.edu/osuna97improved.html.

[6] *Platt J. C.* Fast training support vector machines using sequential minimal optimization // Advances in Kernel Methods / Ed. by B. Scholkopf, C. C. Burges, A. J. Smola. — MIT Press, 1999. — Pp. 185–208.

[7] *Scheinberg K.* An efficient implementation of an active set method for svms // *J. Mach. Learn. Res.* — 2006. — Vol. 7. — Pp. 2237–2257.

[8] *Shawe-Taylor J., Cristianini N.* Robust bounds on generalization from the margin distribution: Tech. Rep. NC2–TR–1998–029: Royal Holloway, University of London, 1998.
http://citeseer.ist.psu.edu/shawe-taylor98robust.html.

[9] *Smola A., Schoelkopf B.* A tutorial on support vector regression: Tech. Rep. NeuroCOLT2 NC2-TR-1998-030: 1998.
http://citeseer.ist.psu.edu/smola98tutorial.html.

[10] *Vapnik V., Chapelle O.* Bounds on error expectation for support vector machines // *Neural Computation.* — 2000. — Vol. 12, no. 9. — Pp. 2013–2036.
http://citeseer.ist.psu.edu/vapnik99bounds.html.