Метод опорных векторов

Опорные вектора и расстояния (зазоры)

- Пусть $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ будет набором классифицируемых данных с *n* точками и размерностью *d*. Далее, предположим, что есть только две метки класса, то есть $y_i \in \{+1, -1\}$, обозначающих положительный и отрицательный классы.

Гиперплоскости

- Функция гиперплоскости:

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + \cdots + w_dx_d + b \qquad 21.1$$

- Здесь **w** – *d*-мерный вектор весов, а *b* – скаляр, называемый смещением. Для точек, лежащих на гиперплоскости, имеем

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0 \qquad 21.2$$

Таким образом, гиперплоскость определяется как множество всех таких точек, что

$$\mathbf{w}^T\mathbf{x} = -\mathrm{b}$$

Разделяющая гиперплоскость

- функция гиперплоскости $h(\mathbf{x})$ служит линейным классификатором или линейным дискриминантом, который предсказывает класс *y* для любой заданной точки **x** в соответствии с решающим правилом:

$$y = \begin{cases} +1 & \text{if } h(\mathbf{x}) > 0 \\ -1 & \text{if } h(\mathbf{x}) < 0 \end{cases} \qquad (21.3)$$

- Из 21.2: $h(\mathbf{a}_1) = \mathbf{w}^T\mathbf{a}_1 + b = 0$
  $h(\mathbf{a}_2) = \mathbf{w}^T\mathbf{a}_2 + b = 0$

$$\mathbf{w}^T(\mathbf{a}_1 - \mathbf{a}_2) = 0$$

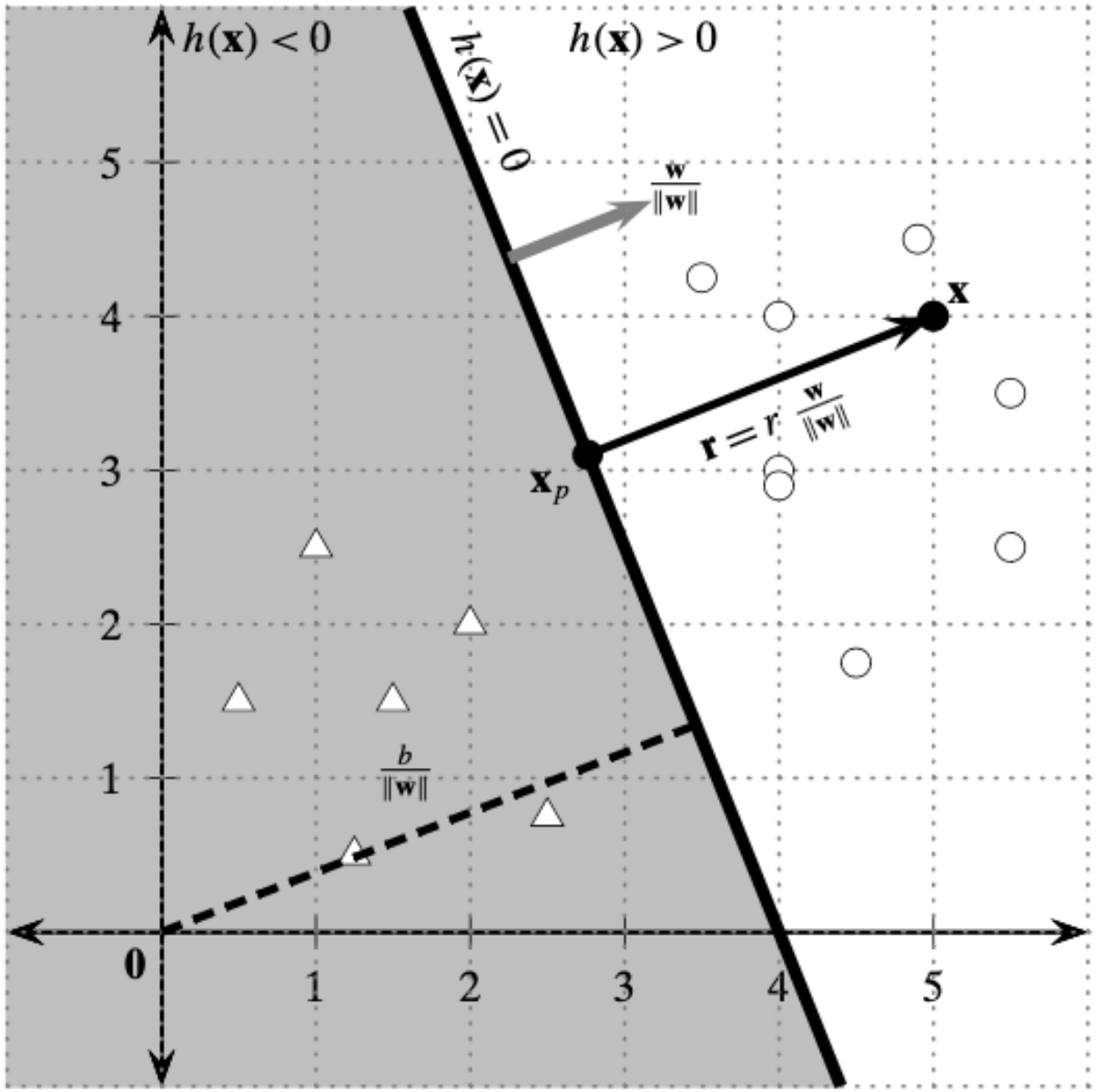
Расстояние от точки до гиперплоскости

- Рассмотрим точку $\mathbf{x} \in \mathbb{R}^d$ такую, что $\mathbf{x}$ не лежит на гиперплоскости. Пусть $\mathbf{x}_p$ – ортогональная проекция $\mathbf{x}$ на гиперплоскость, и пусть $\mathbf{r} = \mathbf{x} - \mathbf{x}_p$, тогда, как показано на рисунке 21.1, мы можем записать $\mathbf{x}$ как

$$\mathbf{x} = \mathbf{x}_p + \mathbf{r}$$

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \quad (21.4)$$

Геометрия разделяющей гиперплоскости в 2D



- Подставляя уравнение (21.4) в функцию гиперплоскости [Ур. (21.1)], получаем

$$h(\mathbf{x}) = h\left(\mathbf{x}_p + r\frac{\mathbf{w}}{\| \mathbf{w} \|}\right)$$

$$= \mathbf{w}^T\left(\mathbf{x}_p + r\frac{\mathbf{w}}{\| \mathbf{w} \|}\right) + b$$

$$= \underbrace{\mathbf{w}^T\mathbf{x}_p + b}_{h(\mathbf{x}_p)} + r\frac{\mathbf{w}^T\mathbf{w}}{\| \mathbf{w} \|}$$

$$= \underbrace{h\left(\mathbf{x}_p\right)}_{0} + r \| \mathbf{w} \|$$

$$= r \| \mathbf{w} \|$$

- Выражение для направленного расстояния от точки до гиперплоскости:

$$r = \frac{h(\mathbf{x})}{\| \mathbf{w} \|}$$

- Расстояние от точки **x** до гиперплоскости $h(\mathbf{x}) = 0$ задается как

$$\delta = yr = \frac{yh(\mathbf{x})}{\|\mathbf{w}\|} \text{ (21.5)}$$

- Для начала координат $\mathbf{x} = 0$ направленное расстояние равно

$$r = \frac{h(\mathbf{0})}{\| \mathbf{w} \|} = \frac{\mathbf{w}^T \mathbf{0} + b}{\| \mathbf{w} \|} = \frac{b}{\| \mathbf{w} \|}$$

Расстояние (зазор) и опорные векторы гиперплоскости

- Для обучающего набора размеченных точек, $\mathbf{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ с $y_i \in \{+1, -1\}$, и для разделяющей гиперплоскости $h(\mathbf{x}) = 0$, для каждой точки $\mathbf{x}_i$ мы можем найти её расстояние до гиперплоскости по формуле (21.5):

$$\delta_i = \frac{y_i h(\mathbf{x}_i)}{\| \mathbf{w} \|} = \frac{y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)}{\| \mathbf{w} \|}$$

По всем *n* точкам мы определяем границу линейного классификатора как минимальное расстояние точки от разделяющей гиперплоскости

$$\delta^* = \min_{\mathbf{x}_i}\left\{\frac{y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)}{\|\mathbf{w}\|}\right\} \quad (21.6)$$

- Опорный вектор $\mathbf{x}^*$ - точка, которая лежит точно на границе классификатора и, таким образом, удовлетворяет условию

$$\delta^* = \frac{y^*\left(\mathbf{w}^T\mathbf{x}^* + b\right)}{\| \mathbf{w} \|}$$

Каноническая гиперплоскость

- [Ур. (21.2)]. Умножение с обеих сторон на некоторый скаляр s дает эквивалентную гиперплоскость

$$sh(\mathbf{x}) = s\mathbf{w}^T\mathbf{x} + sb = (s\mathbf{w})^T\mathbf{x} + (sb) = 0$$

- Скаляр s выбирается так, чтобы абсолютное расстояние вектора поддержки от гиперплоскости равнялось 1

$$sy^*\left(\mathbf{w}^T\mathbf{x}^* + b\right) = 1$$

$$s = \frac{1}{y^*\left(\mathbf{w}^T\mathbf{x}^* + b\right)} = \frac{1}{y^*h(\mathbf{x}^*)} \quad (21.7)$$

- Расстояние (зазор) задается

$$\delta^* = \frac{y^*h(\mathbf{x}^*)}{\| \mathbf{w} \|} = \frac{1}{\| \mathbf{w} \|}$$

- Для канонической гиперплоскости, для каждого вектора поддержки $\mathbf{x}_i^*$ (с меткой y_i^*), мы имеем $y_i^* h(\mathbf{x}_i^*) = 1$, и для любой точки, которая не является опорным вектором мы имеем $y_i h(\mathbf{x}_i) > 1$, потому что, по определению, он должен быть дальше от гиперплоскости, чем опорный вектор
- Набор неравенств по всем *n* точкам в наборе данных **D**

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \text{ для всех точек } \mathbf{x}_i \in \mathbf{D} \quad (21.8)$$

Случай линейного разбиения

- Имея набор данных $\boldsymbol{D} = \{x_i, y_i\}_{\{i=1\}}^{n}$, где $\boldsymbol{x} \in \mathbb{R}^d, e \in \{+1, -1\}$, предположим, что точки можно линейно разбить, т.е. существует гиперплоскость, разбиение по которой безошибочно классифицирует каждую точку. Другими словами, все точки $y_i = +1$ лежат на одной стороне какой-либо гиперплоскости $(h(\boldsymbol{x}) > 0)$, а все $y_i = -1$ лежат на другой стороне той же гиперплоскости $(h(\boldsymbol{x}) < 0)$.

Гиперплоскость с максимальным зазором

- Основная идея SVM – выбор канонической гиперплоскости, заданной вектором весом $\boldsymbol{w}$ и ошибкой b, которая дает наибольший зазор среди всех возможных разделяющих гиперплоскостей. Если δ_h^* - зазор гиперплоскости $h(\boldsymbol{x}) = 0$, то задача найти оптимальную гиперплоскость h^*:

$$h^* = \arg\max_h\{\delta_h^*\} = \arg\max_{\{w,b\}}\left\{\frac{1}{\|w\|}\right\}$$

- Задача SVM – поиск гиперплоскости, максимизирующей зазор $\frac{1}{\|w\|}$ при n ограничениях из (21.8), а именно $y_i\ \left(\boldsymbol{w}^T\boldsymbol{x_i} + b\right) \geq 1$ для всех точек $x \in D$.

- Эквивалентная формулировка задачи условной минимизации:

Целевая функция: $\min\limits_{\{\mathbf{w},\mathrm{b}\}} \left\{\frac{\|\boldsymbol{w}\|^2}{2}\right\}$

Линейные ограничения: $y_i \ \left(\boldsymbol{w}^T \boldsymbol{x_i} + b\right) \geq 1, \forall \boldsymbol{x} \in \boldsymbol{D}$

- Поставленную задачу *прямой* выпуклой минимизации с n линейными ограничениями можно решить стандартными алгоритмами оптимизации, как показано в главе 21.5. Однако, чаще решается *двойственная* задача, которую можно получить с помощью множителей Лагранжа. Основная идея – ввести множитель Лагранжа α_i для каждого ограничения, которое удовлетворяет условиями Каруша-Куна-Такера для оптимального решения:

$$\alpha_i\left(y_i\left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) - 1\right) = 0$$

$$\text{и } \alpha \geq 0$$

- С учетом всех n ограничений, новая целевая функция – *Лагранжиан* – становится

$$\min_L L = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b \right) - 1 \right) \ (21.9)$$

- L минимизируется с учетом w и b, и взяв их нулевые значения, получаем

$$\frac{\partial}{\partial \mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \mathbf{0} \text{ или } \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \ (21.10)$$

$$\frac{\partial}{\partial b} L = \sum_{i=1}^{n} \alpha_i y_i = 0 \ (21.11)$$

- Подставляя эти уравнения в (21.9), можно получить двойственную целевую функцию Лангранжа, заданную исключительно в терминах множителей Лагранжа.

$$L_{dual} = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\left(\underbrace{\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i}_{\mathbf{w}}\right) - b\underbrace{\sum_{i=1}^{n}\alpha_i y_i}_{0} + \sum_{i=1}^{n}\alpha_i$$

$$= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n}\alpha_i$$

$$= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

- Таким образом, двойственная задача:

Целевая функция: $\max_{\alpha} L_{dual} = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \quad (21.12)$

Линейные ограничения: $\alpha_i \geq 0, \forall i \in \mathbf{D}$, and $\sum_{i=1}^{n}\alpha_i y_i = 0$

- где $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$ – вектор из множителей Лагранжа. L_{dual} – задача выпуклого квадратичного программирования (обратите внимание на множители $\alpha_i\alpha_j$), которая может быть решена стандартными методами оптимизации.

Вектор весов и ошибка

- Получив значения α_i для $i = 1, \dots, n,$ мы можем провести решение для вектора $\boldsymbol{w}$ и ошибки b. С учетом условия Каруша-Куна-Такера

$$\alpha_i\left(y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) - 1\right) = 0,$$

- что дает две возможности:

$$\alpha_i = 0, \text{ или}$$

$$y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) - 1 = 0, \text{ что предполагает } y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) = 1$$

- если $\alpha_i > 0$, то $y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) = 1$, и точка $\boldsymbol{x_i}$ должна быть опорным вектором. С другой стороны, если $y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) > 1$, т.е. точка – не опорный вектор, то $\alpha_i = 0$.

- Как только мы знаем α_i для всех точек, мы можем вычислить вектор весов $\boldsymbol{w}$ с помощью уравнения (21.10), но суммируя только опорные векторы:

$$\boldsymbol{w} = \sum_{i,\alpha_i>0} \alpha_i y_i \boldsymbol{x}_i \quad (21.13)$$

- Чтобы вычислить ошибку b, сначала вычисляется одно решение b_i для одного опорного вектора:

$$\alpha_i\left(y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1\right) = 0$$

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) = 1$$

$$b_i = \frac{1}{y_i} - \mathbf{w}^T\mathbf{x}_i = y_i - \mathbf{w}^T\mathbf{x}_i \quad (21.14)$$

- b берется как средняя ошибка по опорным векторам:

$$b = avg_{\alpha_i>0}\{b_i\} \, (21.15)$$

Классификатор SVM

- Имея оптимальную гиперплоскость $h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$, для каждой новой точки $\boldsymbol{z}$ можно предсказать её класс как

$$\hat{y} = sign\big(h(\boldsymbol{z})\big) = sign\big(\boldsymbol{w}^T \boldsymbol{z} + b\big) \; (21.16)$$

- где $sign$ – функция, возвращающая $+1$ для положительного аргумента и -1 для отрицательного.

SVM с мягким зазоров: случай линейной неразделимости

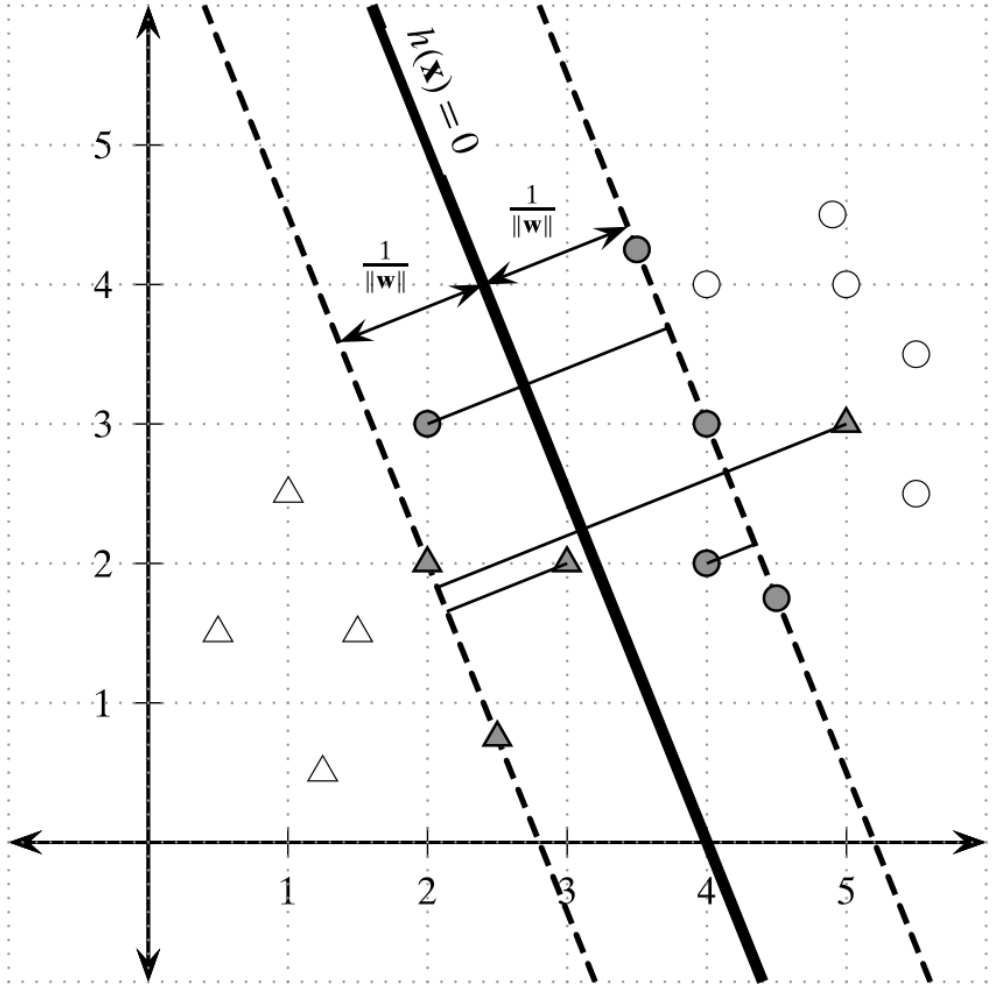


Рисунок 21.3. Гиперплоскость с мягким зазором: серые точки – опорные векторы. Показан зазор $\frac{1}{\|w\|}$

- Обработать неразделимые точки можно, введя *слабые переменные* ξ_i в уравнение (21.8) следующим образом:

$$y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \geq 1 - \xi_i,$$

- где $\xi_i \geq 0 -$ слабая переменная для точки $\boldsymbol{x}_i$, показывающая, насколько точка нарушает условие разбиение, т.е. точка не обязаны быть на расстоянии как минимум $\frac{1}{\|\boldsymbol{w}\|}$ от гиперплоскости.
- В случае линейной неразделимости (иначе – случай с *мягким зазором*), цель SVM – найти гиперплоскость с максимальным зазором и минимальными слабыми переменными. Тогда новая цель:

Целевая функция: $\min\limits_{\mathbf{w},b,\xi_i} \left\{\frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{n}\left(\xi_i\right)^k\right\}$ **(**21.17)

Линейные ограничения:

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \forall\mathbf{x}_i \in \mathbf{D}$$

$$\xi_i \geq 0 \forall\mathbf{x}_i \in \mathbf{D}$$

- Где C, k – ограничения, включающие в себя стоимость ошибочной классификации. Член $\sum_{i=1}^{n}\left(\xi_i\right)^k$ показывает *потерю*, т.е. отклонение от случая линейной разделимости.

- Скаляр C, выбирающийся эмпирически – *константа регуляризации*, контролирующая компромисс между максимизацией зазора (минимизацией $\frac{\|\boldsymbol{w}\|^2}{\mathbf{2}}$) или минимизацией потери (минимизацией суммы слабых членов $\sum_{i=1}^{n} (\xi_i)^k$)
- Константа k показывает вид потери. Обычно, k ставится в 1 или 2. Когда $k = 1$, это называется *петлевая функция потери* (hinge loss), и цель – минимизация суммы слабых переменных, когда $k = 2$, это *квадратичная функция потерь*, и цель – минимизация суммы квадратов слабых переменных

Петлевая функция потерь

- Предположив $k = 1$, можно вычислить Лагранжиан для задачи (21.17), введя множители Лагранжа α_i, β_i, которые удовлетворяют следующим условиям Каруша-Куна-Такера для оптимального решения:

$$\begin{matrix} \alpha_i\left(y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 + \xi_i\right) = 0 \text{ при } \alpha_i \geq 0 \\ \beta_i(\xi_i - 0) = 0 \text{ при } \beta_i \geq 0 \end{matrix} \quad (21.18)$$

- И Лагранжиан задается как:

$$L = \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i\left(y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 + \xi_i\right) - \sum_{i=1}^{n} \beta_i\xi_i \quad (21.19)$$

- Взятием частных производных по $\boldsymbol{w}, b, \xi$, и приравниванием их к 0, получаем:

$$\frac{\partial}{\partial \mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \mathbf{0} \text{ or } \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial}{\partial b} L = \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad (21.20)$$

$$\frac{\partial}{\partial \xi_i} L = C - \alpha_i - \beta_i = 0 \text{ or } \beta_i = C - \alpha_i$$

- Постановка этих значений в (21.19):

$$L_{\text{dual}} = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\left(\underbrace{\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}_{\mathbf{w}}\right) - b\underbrace{\sum_{i=1}^{n} \alpha_i y_i}_{0} + \sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \underbrace{(C - \alpha_i - \beta_i)}_{0} \xi_i$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

- И двойственная задача:

Целевая функция: $\max_{\alpha} L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ (21.21)

Линейные ограничения: $0 \leq \alpha_i \leq C, \forall i \in \mathbf{D}$ и $\sum_{i=1}^{n} \alpha_i y_i = 0$

- Ограничения на α_i теперь другие, т.к. мы требуем $\alpha_i + \beta_i = C$, где $\alpha_i \geq 0$ и $\beta_i \geq 0$, что предполагает $0 \leq \alpha_i \leq C$.

Вектор весов и потеря

- После решения для α_i, мы оказываемся в той же ситуации, что и раньше – точки $\alpha_i = 0 -$ не опорные вектора, и $\alpha_i > 0$ только для опорных векторов, составляющих все $\boldsymbol{x}_i$, для которых

$$y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) = 1 - \xi \quad (21.22)$$

- Заметим, что опорные вектора теперь включают как все точки вне зазора, для которых $\xi_i = 0$, так и точки $\xi_i > 0$.
- Как и раньше, вектор весов можно получить из опорных векторов:

$$\mathbf{w} = \sum_{i,\alpha_i>0} \alpha_i y_i \mathbf{x}_i \ (21.23)$$

- Можно получить решение для β_i с помощью уравнения (21.20):

$$\beta_i = C - \alpha_i$$

- Заменяя β_i в условиях Каруша-Куна-Такера (21.18), получаем:

$$(\mathrm{C} - \alpha_i)\xi_i = 0 \ (21.24)$$

- Т.е. для опорных векторов с $\alpha_i > 0$ возможны 2 случая:
- $\xi_i > 0$, что предполагает $C - \alpha_i = 0$, т.е. $\alpha_i = C$, или $C - \alpha_i > 0$, т.е. $\alpha_i < C$. В этом случае $\xi_i = 0$, и эти опорные вектора находятся вне зазора.
- С использованием опорных векторов вне зазора, для которых $0 < \alpha_i < C$ и $\xi_i = 0$, можно решить для b_i:

$$\alpha_i\left(y_i\left(\mathbf{w}^T\mathbf{x}_i + b_i\right) - 1\right) = 0$$

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b_i\right) = 1 \qquad (21.25)$$

$$b_i = \frac{1}{y_i} - \mathbf{w}^T\mathbf{x}_i = y_i - \mathbf{w}^T\mathbf{x}_i$$

- Когда оптимальная гиперплоскость определена, модель SVM предсказывает класс точки $\boldsymbol{z}$ следующим образом:

$$\hat{y} = \mathrm{sign}\left(h(\mathbf{z})\right) = \mathrm{sign}\left(\mathbf{w}^T\mathbf{z} + b\right)$$

Квадратичная потеря

- Для квадратичной потери имеем $k = 2$ в целевой функции (21.17). В этом случае, ограничение на положительность $\xi_i \geq 0$ можно отбросить, т.к. (1) сумма слабых членов $\sum_{i=1}^{n} \xi^2$ всегда положительна, и (2) возможные отрицательные значения слабых членов исключаются во время оптимизации, потому что выбор $\xi_i = 0$ приводит к уменьшению значения основной цели, и всё ещё удовлетворяет ограничению $y_i\left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) \geq 1 - \xi_i$ когда $\xi_i < 0$. Другими словами, процесс оптимизации заменит отрицательные значения слабых членов нулями. Таким образом, задача SVM для квадратичной потери задана следующим образом:

Целевая функция: $\underset{\mathbf{w},b,\xi_i}{min} \left\{ \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{n} (\xi_i)^2 \right\}$

Линейные ограничения: $y_i\left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 - \xi_i, \forall \mathbf{x}_i \in \mathbf{D}$

- И Лагранжиан:

$$L = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} \xi_i^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 + \xi_i \right) (21.26)$$

- Дифференцированием по $\boldsymbol{w}$, b и ξ_i и приравниванием соответствующих результатов к 0, получаем:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\xi_i = \frac{1}{2C} \alpha_i$$

- Подстановка результата обратно в (21.26) дает двойственную задачу:

$$L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \frac{1}{4C} \sum_{i=1}^{n} \alpha_i^2$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i^T \mathbf{x}_j + \frac{1}{2C} \delta_{ij} \right)$$

- где δ – *дельта-функция Кронекера*, заданная как $\delta_{ij} = 1$, где $i = 1$, и $\delta_{ij} = 0$ во всех остальных случаях.

- Таким образом, двойственная задача:

$$\max_{\alpha} L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i^T \mathbf{x}_j + \frac{1}{2C} \delta_{ij} \right) \quad (21.27)$$

$$\text{с ограничениями } \alpha_i \geq 0, \forall i \in \mathbf{D}, \text{ и } \sum_{i=1}^{n} \alpha_i y_i = 0$$

- Решая для α_i методами из раздела 21.5, можно получить вектор весов и ошибку следующим образом:

$$\mathbf{w} = \sum_{i, \alpha_i > 0} \alpha_i y_i \mathbf{x}_i$$

$$b = \text{avg}_{i, C > \alpha_i > 0} \{ y_i - \mathbf{w}^T \mathbf{x}_i \}$$

Ядерный SVM: нелинейный случай

- Чтобы применить трюк с ядром для нелинейной классификации SVM, мы должны показать, что для всех операций требуется только функция ядра:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- Пусть исходная база данных задана как $\mathbf{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$. Применяя ϕ к каждой точке, мы можем получить новый набор данных в пространстве признаков $\mathbf{D}_\phi = \{\phi(\mathbf{x}_i), y_i\}_{i=1}^n$.
- **Целевая функция**:

$$\min_{\mathbf{w},b,\xi_i} \left\{ \frac{\| \mathbf{w} \|^2}{2} + C \sum_{i=1}^{n} (\xi_i)^k \right\} \quad (21.28)$$

- **Линейные ограничения**:

$$y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \text{ and } \xi_i \geq 0, \forall \mathbf{x}_i \in \mathbf{D}$$

- где $\mathbf{w}$ – вектор весов, b – смещение, а ξ_i – переменные запаса (slack), все в пространстве признаков.

Функция потерь

- Для функции потерь двойственная задача Лагранжа [Ур. (21.21)] в пространстве признаков задается как

$$\begin{aligned} \max_{\boldsymbol{\alpha}} L_{dual} &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi\left(\mathbf{x}_j\right) \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \end{aligned} \tag{21.29}$$

- Учитываются ограничения $0 \leq \alpha_i \leq C$ и $\sum_{i=1}^{n} \alpha_i y_i = 0$. Двойственный лагранжиан зависит только от скалярного произведения двух векторов в пространстве признаков $\phi(\mathbf{x}_i)^T \phi\left(\mathbf{x}_j\right) = K\left(\mathbf{x}_i, \mathbf{x}_j\right)$, и, таким образом, мы можем решить задачу оптимизации, используя матрицу ядра $\mathbf{K} = \left\{K\left(\mathbf{x}_i, \mathbf{x}_j\right)\right\}_{i,j=1,\ldots,n}$.

Квадратичная функция потерь

- Для квадратичных потерь двойственная задача Лагранжа [уравнение (21.27)] соответствует замене ядра. Определим новую функцию ядра K_q следующим образом:

$$K_q(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j + \frac{1}{2C}\delta_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{2C}\delta_{ij}$$

- которая влияет только на диагональные элементы матрицы ядра K, так как $\delta_{ij} = 1,$ если $i = j$, и ноль в противном случае. Таким образом, двойственная задача Лагранжа имеет вид

$$\max_{\alpha} L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K_q(\mathbf{x}_i, \mathbf{x}_j) \quad (21.30)$$

- Учитываются ограничения $\alpha_i \geq 0$ и $\sum_{i=1}^{n} \alpha_i y_i = 0$. Вышеупомянутая оптимизация может быть решена с использованием того же подхода, что и для функции потерь, с простой заменой ядра.

Вектор веса и смещение

- Мы можем решить для $\mathbf{w}$ в пространстве признаков следующим образом:

$$\mathbf{w} = \sum_{\alpha_i > 0} \alpha_i y_i \phi(\mathbf{x}_i) \quad (21.31)$$

- Поскольку $\mathbf{w}$ использует $\phi(\mathbf{x}_i)$ напрямую, в общем случае мы не можем или не хотим вычислять $\mathbf{w}$ явно. Однако, как мы увидим далее, нет необходимости явно вычислять $\mathbf{w}$ для классификации точек.
- Как вычислить смещение с помощью операций ядра? Используя уравнение (21.25), мы вычисляем b как среднее по опорным векторам, которые находятся на грани, то есть с $0 < \alpha_i < C$ и $\xi_i = 0$:

$$b = \text{avg}_{i, 0<\alpha_i<C}\{b_i\} = \text{avg}_{i, 0<\alpha_i<C}\left\{y_i - \boldsymbol{w}^T \phi(\boldsymbol{x}_i)\right\} \quad (21.32)$$

- Подставляя $\mathbf{w}$ из уравнения. (21.31), получаем новое выражение для b_i:

$$b_i = y_i - \sum_{\alpha_j > 0} \alpha_j y_j \phi\left(\mathbf{x}_j\right)^T \phi(\mathbf{x}_i)$$
$$= y_i - \sum_{\alpha_j > 0} \alpha_j y_j K\left(\mathbf{x}_j, \mathbf{x}_i\right) \qquad (21.33)$$

- Обратите внимание, что b_i является функцией скалярного произведения двух векторов в пространстве признаков, и поэтому его можно вычислить с помощью функции ядра во входном пространстве.

Ядерный SVM классификатор

- Мы можем предсказать класс для новой точки $\mathbf{z}$ следующим образом:

$$\hat{y} = \text{sign}\left(\mathbf{w}^T \phi(\mathbf{z}) + b\right)$$

$$= \text{sign}\left(\sum_{\alpha_i > 0} \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{z}) + b\right)$$

$$= \text{sign}\left(\sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{z}) + b\right)$$

- Мы снова видим, что $\hat{y}$ использует только скалярные произведения в пространстве признаков.

Алгоритмы обучения SVM

- Для алгоритмов SVM в этом разделе вместо того, чтобы явно иметь дело со смещением b, мы отображаем каждую точку $\mathbf{x}_i \in \mathbb{R}^d$ в точку $\mathbf{x}'_i \in \mathbb{R}^{d+1}$ следующим образом:

$$\mathbf{x}'_i = (x_{i1}, \ldots, x_{id}, 1)^T \ (21.34)$$

- Кроме того, мы также отображаем весовой вектор в $\mathbb{R}^{d+1}$, где $w_{d+1} = b$, так что

$$\mathbf{w} = (w_1, \ldots, w_d, b)^T \ (21.35)$$

- Уравнение гиперплоскости [уравнение (21.1)] тогда задается следующим образом:

$$h(\mathbf{x}'): \mathbf{w}^T\mathbf{x}' = 0$$

$$h(\mathbf{x}'): \begin{pmatrix} w_1 & \cdots & w_d & b \end{pmatrix} \begin{pmatrix} x_{i1} \\ \vdots \\ x_{id} \\ 1 \end{pmatrix} = 0$$

$$h(\mathbf{x}'): w_1 x_{i1} + \cdots + w_d x_{id} + b = 0$$

- В обсуждении ниже мы предполагаем, что член смещения был включен в $\mathbf{w}$, и что каждая точка была отображена в $\mathbb{R}^{d+1}$ согласно уравнениям (21.34) и (21.35). Таким образом, последняя компонента $\mathbf{w}$ дает смещение b. Другим следствием отображения точек на $\mathbb{R}^{d+1}$ является то, что ограничение $\sum_{i=1}^{n} \alpha_i y_i = 0$ не применяется в двойственных формулировках SVM, приведенных в уравнениях (21.21), (21.27), (21.29) и (21.30), поскольку нет явного члена смещения b для линейных ограничений в целевой функции SVM, заданной в уравнении (21.17). Новый набор ограничений задается как

$$y_i \mathbf{w}^T \mathbf{x} \geq 1 - \xi_i$$

Двойственное решение: стохастический градиентный подъем

- Двойственная целевая функция оптимизации для функции потерь [Ур. (21.29)] имеет вид

$$\max_{\alpha} J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

- при ограничениях $0 \leq \alpha_i \leq C \; for \; all \; i = 1, \ldots, n$. Тут $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_n)^T \in \mathbb{R}^n$.
- Рассмотрим слагаемые в $J(\boldsymbol{\alpha})$, в которые входит множитель Лагранжа α_k:

$$J(\alpha_k) = \alpha_k - \frac{1}{2} \alpha_k^2 y_k^2 K(\mathbf{x}_k, \mathbf{x}_k) - \alpha_k y_k \sum_{\substack{i=1 \\ i \neq k}}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)$$

- Градиент или скорость изменения целевой функции при $\boldsymbol{\alpha}$ задается как частная производная от $J(\boldsymbol{\alpha})$ по $\boldsymbol{\alpha}$, то есть по каждому α_k:

$$\nabla J(\boldsymbol{\alpha}) = \left(\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_1}, \frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_2}, \ldots, \frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_n} \right)^T$$

- где k-я компонента градиента получается дифференцированием $J(\alpha_k)$ по α_k:

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_k} = \frac{\partial J(\alpha_k)}{\partial \alpha_k} = 1 - y_k \left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k) \right) \quad (21.36)$$

- Поскольку мы хотим максимизировать целевую функцию $J(\boldsymbol{\alpha})$, мы должны двигаться в направлении градиента $\nabla J(\boldsymbol{\alpha})$. Начиная с начального $\boldsymbol{\alpha}$, подход градиентного подъема последовательно обновляет его следующим образом:

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta_t \nabla \boldsymbol{J}(\boldsymbol{\alpha}_t)$$

- где $\boldsymbol{\alpha}_t$ – оценка на шаге t, η_t – размер шага.
- Правило обновления для k-го компонента задается как

$$\alpha_k = \alpha_k + \eta_k \frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_k} = \alpha_k + \eta_k \left(1 - y_k \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k) \right) \quad (21.37)$$

- где η_k – размер шага. Мы также должны обеспечить выполнение ограничений $\alpha_k \in [0, C]$. Таким образом, на шаге обновления выше, если $\alpha_k < 0$, мы сбрасываем его до $\alpha_k = 0$, а если $\alpha_k > C$, мы сбрасываем его до $\alpha_k = C$.

ALGORITHM 21.1. Dual SVM Algorithm: Stochastic Gradient Ascent

SVM-DUAL $(\mathbf{D}, K, C, \epsilon)$:

1 **foreach** $\mathbf{x}_i \in \mathbf{D}$ **do** $\mathbf{x}_i \leftarrow \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$ // map to $\mathbb{R}^{d+1}$

2 **if** *loss* = `hinge` **then**

3 $\quad \mathbf{K} \leftarrow \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,\ldots,n}$ // kernel matrix, hinge loss

4 **else if** *loss* = `quadratic` **then**

5 $\quad \mathbf{K} \leftarrow \{K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{2C}\delta_{ij}\}_{i,j=1,\ldots,n}$ // kernel matrix, quadratic loss

6 **for** $k = 1, \ldots, n$ **do** $\eta_k \leftarrow \frac{1}{K(\mathbf{x}_k, \mathbf{x}_k)}$ // set step size

7 $t \leftarrow 0$

8 $\boldsymbol{\alpha}_0 \leftarrow (0, \ldots, 0)^T$

9 **repeat**

10 $\quad \boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_t$

11 $\quad$ **for** $k = 1$ *to* n **do**

$\quad\quad$ // update kth component of $\boldsymbol{\alpha}$

12 $\quad\quad \alpha_k \leftarrow \alpha_k + \eta_k \left(1 - y_k \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right)$

13 $\quad\quad$ **if** $\alpha_k < 0$ **then** $\alpha_k \leftarrow 0$

14 $\quad\quad$ **if** $\alpha_k > C$ **then** $\alpha_k \leftarrow C$

15 $\quad \boldsymbol{\alpha}_{t+1} \leftarrow \boldsymbol{\alpha}$

16 $\quad t \leftarrow t + 1$

17 **until** $\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t-1}\| \leq \epsilon$

- Чтобы определить размер шага η_k, мы хотели бы выбрать его так, чтобы градиент при α_k стремился к нулю, что происходит, когда

$$\eta_k = \frac{1}{K(\mathbf{x}_k, \mathbf{x}_k)} \quad (21.38)$$

- Чтобы понять, почему, обратите внимание, что, когда обновляется только α_k, другие α_i не меняются. Таким образом, новый α имеет изменение только в α_k, и из уравнения (21.36) получаем

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_k} = \left(1 - y_k \sum_{i \neq k} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right) - y_k \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_k)$$

- Подставляя значение α_k из уравнения (21.37), имеем

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_k} = \left(1 - y_k \sum_{i \neq k} \alpha_i y_i \boldsymbol{K}(\mathbf{x}_i, \mathbf{x}_k)\right) - \left(\alpha_k + \eta_k \left(1 - y_k \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{K}(\mathbf{x}_i, \mathbf{x}_k)\right)\right) \boldsymbol{K}(\mathbf{x}_k, \mathbf{x}_k)$$

$$= \left(1 - y_k \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{K}(\mathbf{x}_i, \mathbf{x}_k)\right) - \eta_k \boldsymbol{K}(\mathbf{x}_k, \mathbf{x}_k) \left(1 - y_k \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right)$$

$$= \left(1 - \eta_k \boldsymbol{K}(\mathbf{x}_k, \mathbf{x}_k)\right) \left(1 - y_k \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{K}(\mathbf{x}_i, \mathbf{x}_k)\right)$$

- Подставляя η_k из уравнения (21.38), имеем

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial a_k} = \left(1 - \frac{1}{K(\mathbf{x}_k, \mathbf{x}_k)} K(\mathbf{x}_k, \mathbf{x}_k)\right)\left(1 - y_k \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right) = 0$$

- Метод последовательно обновляет $\boldsymbol{\alpha}$ и останавливается, когда изменение падает ниже заданного порога ϵ. Вычислительная сложность метода составляет $O(n^2)$ на итерацию.
- Обратите внимание, что после получения окончательного $\boldsymbol{\alpha}$ мы классифицируем новую точку $\mathbf{z} \in \mathbb{R}^{d+1}$ следующим образом:

$$\hat{y} = \text{sign}(h(\phi(\mathbf{z}))) = \text{sign}\left(\mathbf{w}^T \phi(\mathbf{z})\right) = \text{sign}\left(\sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{z})\right)$$

Основное решение: оптимизация Ньютона

- При $\mathbf{w}, \mathbf{x}_i \in \mathbb{R}^{d+1}$, как обсуждалось ранее, мы должны минимизировать целевую функцию:

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} (\xi_i)^k \quad (21.39)$$

- с учетом линейных ограничений

$$y_i\left(\mathbf{w}^T \mathbf{x}_i\right) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } i = 1, \ldots, n$$

- Преобразуя написанное выше, получаем выражение для $\xi_i \geq 1 - y_i\left(\mathbf{w}^T \mathbf{x}_i\right)$ и $\xi_i \geq 0$ откуда следует, что

$$\xi_i = \max\left\{0, 1 - y_i\left(\mathbf{w}^T \mathbf{x}_i\right)\right\} \quad (21.40)$$

- Подставляя уравнение (21.40) в целевую функцию [Ур. (21.39)], получаем

$$J(\mathbf{w}) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} max\left\{0, 1 - y_i\left(\mathbf{w}^T \mathbf{x}_i\right)\right\}^k$$

$$= \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{y_i\left(\mathbf{w}^T \mathbf{x}_i\right) < 1} \left(1 - y_i\left(\mathbf{w}^T \mathbf{x}_i\right)\right)^k \quad (21.41)$$

Квадратичная функция потерь

- Для квадратичных потерь $k = 2$ целевую функцию [Ур. (21.41)] можно записать как

$$J(\mathbf{w}) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{y_i(\mathbf{w}^T \mathbf{x}_i)<1} \left(1 - y_i\left(\mathbf{w}^T \mathbf{x}_i\right)\right)^2$$

- Градиент или скорость изменения целевой функции в $\mathbf{w}$ задается как частная производная $J(\mathbf{w})$ по отношению к $\mathbf{w}$:

$$\nabla_{\mathbf{w}} = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w} - 2C \left(\sum_{y_i(\mathbf{w}^T \mathbf{x}_i)<1} y_i \mathbf{x}_i \left(1 - y_i\left(\mathbf{w}^T \mathbf{x}_i\right)\right) \right)$$

$$= \mathbf{w} - 2C \left(\sum_{y_i(\mathbf{w}^T \mathbf{x}_i)<1} \mathbf{y}_i \right) + 2C \left(\sum_{\mathbf{x}_i} \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{w}$$

$$= \mathbf{w} - 2C\mathbf{v} + 2C\mathbf{S}\mathbf{w}$$

- где вектор $\mathbf{v}$ и матрица $\mathbf{S}$ заданы как

$$\mathbf{v} = \sum_{y_i(\mathbf{w}^T \mathbf{x}_i)<1} y_i \mathbf{x}_i \qquad \mathbf{S} = \sum_{y_i(\mathbf{w}^T \mathbf{x}_i)<1} \mathbf{x}_i \mathbf{x}_i^T$$

- Матрица Гессе определяется как матрица частных производных второго порядка $J(\mathbf{w})$ по $\mathbf{w}$, которая задается как

$$\mathbf{H}_{\mathrm{w}} = \frac{\partial \nabla_{\mathbf{w}}}{\partial \mathbf{w}} = \mathbf{I} + 2C\mathbf{S}$$

- Поскольку мы хотим минимизировать целевую функцию $J(\mathbf{w})$, мы должны двигаться в направлении, противоположном градиенту. Правило обновления оптимизации Ньютона для $\mathbf{w}$ задается как

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{H}_{\mathbf{w}_t}^{-1} \nabla_{\mathbf{w}_t} \quad (21.42)$$

- где $\eta_t > 0$ – скалярное значение, обозначающее размер шага на итерации t.
- Вычисление $\mathbf{S}$ требует $O(nd^2)$ шагов; вычисление градиента ∇, матрицы Гессе $\mathbf{H}$ и обновление весового вектора $\mathbf{w}_{t+1}$ занимает время $O(d^2)$; и инвертирование гессиана требует $O(d^3)$ операций для общей вычислительной сложности $O\left(nd^2 + d^3\right)$ на итерацию в худшем случае.
-

ALGORITHM 21.2. Primal SVM Algorithm: Newton Optimization, Quadratic Loss

SVM-PRIMAL $(\mathbf{D}, C, \epsilon)$:

foreach $\mathbf{x}_i \in \mathbf{D}$ **do**

  $\mathbf{x}_i \leftarrow \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$ // map to $\mathbb{R}^{d+1}$

$t \leftarrow 0$

$\mathbf{w}_0 \leftarrow (0, \ldots, 0)^T$ // initialize $\mathbf{w}_t \in \mathbb{R}^{d+1}$

repeat

  $\mathbf{v} \leftarrow \sum_{y_i(\mathbf{w}_t^T \mathbf{x}_i) < 1} y_i \mathbf{x}_i$

  $\mathbf{S} \leftarrow \sum_{y_i(\mathbf{w}_t^T \mathbf{x}_i) < 1} \mathbf{x}_i \mathbf{x}_i^T$

  $\nabla \leftarrow (\mathbf{I} + 2C\mathbf{S})\mathbf{w}_t - 2C\mathbf{v}$ // gradient

  $\mathbf{H} \leftarrow \mathbf{I} + 2C\mathbf{S}$ // Hessian

  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \mathbf{H}^{-1} \nabla$ // Newton update rule [Eq. (21.42)]

  $t \leftarrow t + 1$

until $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| \leq \epsilon$

Ядерные SVMs

- Пусть ϕ обозначает отображение из входного пространства в пространство признаков; каждая входная точка $\mathbf{x}_i$ отображается в характерную точку $\phi(\mathbf{x}_i)$. Пусть $K(\mathbf{x}_i, \mathbf{x}_j)$ обозначает ядерную функцию, а $\mathbf{w}$ обозначает весовой вектор в пространстве признаков. Гиперплоскость в пространстве признаков тогда задается как

$$h(\mathbf{x}): \mathbf{w}^T \phi(\mathbf{x}) = 0$$

- Используя уравнения (21.28) и (21.40), основная целевая функция в пространстве признаков может быть записана как

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{n} L\big(y_i, h(\mathbf{x}_i)\big) \quad (21.43)$$

- где $L\big(y_i, h(\mathbf{x}_i)\big) = max\{0, 1 - y_i h(\mathbf{x}_i)\}^k$ – функция потерь.
- Градиент в $\mathbf{w}$ задается как

$$\nabla_{\mathbf{w}} = \mathbf{w} + C \sum_{i=1}^{n} \frac{\partial L\big(y_i, h(\mathbf{x}_i)\big)}{\partial h(\mathbf{x}_i)} \cdot \frac{\partial h(\mathbf{x}_i)}{\partial \mathbf{w}}$$

$$\frac{\partial h(\mathbf{x}_i)}{\partial \mathbf{w}} = \frac{\partial \mathbf{w}^T \phi(\mathbf{x}_i)}{\partial \mathbf{w}} = \phi(\mathbf{x}_i)$$

- При оптимальном решении градиент обращается в нуль, т. Е. $\nabla_{\mathbf{w}} = 0$, что дает

$$\mathbf{w} = -C \sum_{i=1}^{n} \frac{\partial L\big(y_i, h(\mathbf{x}_i)\big)}{\partial h(\mathbf{x}_i)} \cdot \phi(\mathbf{x}_i) \qquad (21.44)$$

$$= \sum_{i=1}^{n} \beta_i \phi(\mathbf{x}_i)$$

- где β_i – коэффициент точки $\phi(\mathbf{x}_i)$ в пространстве признаков. Другими словами, оптимальный весовой вектор в пространстве признаков выражается как линейная комбинация точек $\phi(\mathbf{x}_i)$ в пространстве признаков.
- Используя уравнение (21.44) расстояние до гиперплоскости в пространстве признаков можно выразить как

$$y_i h(\mathbf{x}_i) = y_i \mathbf{w}^T \phi(\mathbf{x}_i) = y_i \sum_{j=1}^{n} \beta_j K\big(\mathbf{x}_j, \mathbf{x}_i\big) = y_i \mathbf{K}_i^T \boldsymbol{\beta} \quad (21.45)$$

- где $\mathbf{K} = \big\{K\big(\mathbf{x}_i, \mathbf{x}_j\big)\big\}_{i,j=1}^{n}$ – ядерная матрица $n \times n$, $\mathbf{K}_i$ – i-ая колонна $\mathbf{K}$ и $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)^T$ – вектор коэффициентов.

- Подстановка уравнений (21.44) и (21.45) в уравнение (21.43) с квадратичными потерями (k = 2) дает формулировку SVM с ядром исключительно в терминах матрицы ядра.

$$\min_{\beta} J(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j K(\mathbf{x}_i,\mathbf{x}_j) + C\sum_{i=1}^{n} max\{0,1-y_i\mathbf{K}_i^T\boldsymbol{\beta}\}^2$$

$$= \frac{1}{2}\boldsymbol{\beta}^T\mathbf{K}\boldsymbol{\beta} + C\sum_{y_i\mathbf{K}_i^T\beta<1}\left(1-y_i\mathbf{K}_i^T\boldsymbol{\beta}\right)^2$$

- Градиент $J(\boldsymbol{\beta})$ относительно $\boldsymbol{\beta}$ задается как

$$\nabla_\beta = \frac{\partial J(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \mathbf{K}\boldsymbol{\beta} - 2C\sum_{y_i\mathbf{K}_i^T\boldsymbol{\beta}<1} y_i\mathbf{K}_i\left(1-y_i\mathbf{K}_i^T\boldsymbol{\beta}\right)$$

$$= \mathbf{K}\boldsymbol{\beta} + 2C\sum_{y_i\mathbf{K}_i^T\boldsymbol{\beta}<1}\left(\mathbf{K}_i\mathbf{K}_i^T\right)\boldsymbol{\beta} - 2C\sum_{y_i\mathbf{K}_i^T\boldsymbol{\beta}<1}\mathbf{K}_i$$

$$= (\mathbf{K}+2C\mathbf{S})\boldsymbol{\beta} - 2C\mathbf{v}$$

- Вектор $\mathbf{v} \in \mathbb{R}^n$ и матрица $\mathbf{S} \in \mathbb{R}^{n \times n}$ заданы как

$$\mathbf{v} = \sum_{y_i \mathbf{K}_i^T \beta < 1} y_i \mathbf{K}_i$$

$$\mathbf{S} = \sum_{y_i \mathbf{K}_i^T \beta < 1} \mathbf{K}_i \mathbf{K}_i^T$$

- Кроме того, матрица Гессе имеет вид

$$\mathbf{H}_\beta = \frac{\partial \nabla_\beta}{\partial \beta} = \mathbf{K} + 2C\mathbf{S}$$

- Теперь мы можем минимизировать $J(\boldsymbol{\beta})$ с помощью оптимизации Ньютона, используя следующее правило обновления:

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \mathbf{H}_\beta^{-1} \nabla_{\boldsymbol{\beta}}$$

- Заметим, что если $\mathbf{H}_\beta$ сингулярна, т.е. если у нее нет обратной, мы добавляем к диагонали небольшой гребень, чтобы регуляризовать ее. То есть сделаем $\mathbf{H}$ обратимым следующим образом:

$$\mathbf{H}_\beta = \mathbf{H}_\beta + \lambda \mathbf{I}$$

- где $\lambda > 0$ – некоторое небольшое положительное значение гребня.
- После определения $\boldsymbol{\beta}$ любую контрольную точку $\mathbf{z}$ легко классифицировать следующим образом:

$$\hat{y} = \text{sign}\left(\mathbf{w}^T \phi(\mathbf{z})\right) = \text{sign}\left(\sum_{i=1}^{n} \beta_i \phi(\mathbf{x}_i)^T \phi(\mathbf{z})\right) = \text{sign}\left(\sum_{i=1}^{n} \beta_i K(\mathbf{x}_i, \mathbf{z})\right)$$

ALGORITHM 21.3. Primal Kernel SVM Algorithm: Newton Optimization, Quadratic Loss

SVM-PRIMAL-KERNEL $(\mathbf{D}, K, C, \epsilon)$:

foreach $\mathbf{x}_i \in \mathbf{D}$ **do**

  $\mathbf{x}_i \leftarrow \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$ // map to $\mathbb{R}^{d+1}$

$\mathbf{K} \leftarrow \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,\ldots,n}$ // compute kernel matrix

$t \leftarrow 0$

$\boldsymbol{\beta}_0 \leftarrow (0, \ldots, 0)^T$ // initialize $\boldsymbol{\beta}_t \in \mathbb{R}^n$

repeat

  $\mathbf{v} \leftarrow \sum_{y_i(\mathbf{K}_i^T \boldsymbol{\beta}_t) < 1} y_i \mathbf{K}_i$

  $\mathbf{S} \leftarrow \sum_{y_i(\mathbf{K}_i^T \boldsymbol{\beta}_t) < 1} \mathbf{K}_i \mathbf{K}_i^T$

  $\nabla \leftarrow (\mathbf{K} + 2C\mathbf{S})\boldsymbol{\beta}_t - 2C\mathbf{v}$ // gradient

  $\mathbf{H} \leftarrow \mathbf{K} + 2C\mathbf{S}$ // Hessian

  $\boldsymbol{\beta}_{t+1} \leftarrow \boldsymbol{\beta}_t - \eta_t \mathbf{H}^{-1} \nabla$ // Newton update rule

  $t \leftarrow t + 1$

until $\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\| \leq \epsilon$