ВВЕДЕНИЕ В КЛАССИФИКАЦИЮ

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург

2 марта 2019 г.

Random facts:

- 2 марта 1930 г. в газете «Правда» опубликована статья Иосифа Сталина «Головокружение от успехов» о «перегибах на местах», допущенных при коллективизации
- 2 марта 1949 г. на самолёте Lucky Lady II завершил первый беспосадочный полёт вокруг земного шара экипаж из 13 мужчин
- 2 марта 1989 г. во время телемоста Киев—Тбилиси А.М. Кашпировский произвёл дистанционное обезболивание хирургических операций на брюшной полости двум пациенткам
- 2 марта 2017 г. на конференции в Москве были официально добавлены в таблицу Менделеева впервые синтезированные в Дубне 115, 117 и 118 элементы: московий, теннессин и оганесон

Логистическая регрессия

В прошлый раз

- Итак, мы рассмотрели логистический сигмоид:

$$p(C_1 \mid x) = \frac{p(x \mid C_1)p(C_1)}{p(x \mid C_1)p(C_1) + p(x \mid C_2)p(C_2)} = \frac{1}{1 + e^{-a}} = \sigma(a),$$

$$\text{где } a = \ln \frac{p(x \mid C_1)p(C_1)}{p(x \mid C_2)p(C_2)}, \qquad \sigma(a) = \frac{1}{1 + e^{-a}}.$$

- Вывели из него LDA и QDA, обучили их методом максимального правдоподобия, а потом отвлеклись на naive Bayes.

Два класса

- Возвращаемся к задаче классификации.
- Два класса, и апостериорное распределение – логистический сигмоид на линейной функции:

$$p(C_1 \mid \phi) = y(\phi) = \sigma(w^\top \phi), \quad p(C_2 \mid \phi) = 1 - p(C_1 \mid \phi).$$

- *Логистическая регрессия* – это когда мы напрямую оптимизируем w.

Два класса

- Для датасета $\{\phi_n, t_n\}$, $t_n \in \{0, 1\}$, $\phi_n = \phi(x_n)$:

$$p(t \mid w) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1 - t_n}, \quad y_n = p(C_1 \mid \phi_n).$$

- Ищем параметры максимального правдоподобия, минимизируя $-\ln p(t \mid w)$:

$$E(w) = -\ln p(t \mid w) = -\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right].$$

Два класса

- Пользуясь тем, что $\sigma' = \sigma(1 - \sigma)$, берём градиент (похоже на перцептрон):

$$\nabla E(w) = \sum_{n=1}^{N} (y_n - t_n)\phi_n.$$

- Если теперь сделать градиентный спуск, получим как раз разделяющую поверхность.
- Заметим, правда, что если данные действительно разделимы, то может получиться жуткий оверфиттинг: $\|w\| \to \infty$, и сигмоид превращается в функцию Хевисайда. Надо регуляризовать.

IRLS

- В логистической регрессии не получается замкнутого решения из-за сигмоида.
- Но функция $E(w)$ всё равно выпуклая, и можно воспользоваться методом Ньютона-Рапсона – на каждом шаге использовать локальную квадратичную аппроксимацию к функции ошибки:

$$w^{\text{new}} = w^{\text{old}} - H^{-1}\nabla E(w),$$

где H (Hessian) – матрица вторых производных $E(w)$.

IRLS

- Замечание: давайте применим Ньютона-Рапсона к обычной линейной регрессии с квадратической ошибкой:

$$\nabla E(w) = \sum_{n=1}^{N} \left(w^\top \phi_n - t_n\right) \phi_n = \Phi^\top \Phi w - \Phi^\top t,$$

$$\nabla\nabla E(w) = \sum_{n=1}^{N} \phi_n \phi_n^\top = \Phi^\top \Phi,$$

и шаг оптимизации будет

$$w^{\text{new}} = w^{\text{old}} - \left(\Phi^\top \Phi\right)^{-1} \left[\Phi^\top \Phi w^{\text{old}} - \Phi^\top t\right] =$$

$$= \left(\Phi^\top \Phi\right)^{-1} \Phi^\top t,$$

т.е. мы за один шаг придём к решению.

IRLS

- Для логистической регрессии:

$$\nabla E(w) = \sum_{n=1}^{N} (y_n - t_n)\, \phi_n = \Phi^\top (y - t)\,,$$

$$H = \nabla\nabla E(w) = \sum_{n=1}^{N} y_n(1 - y_n)\phi_n \phi_n^\top = \Phi^\top R \Phi$$

для диагональной матрицы R с $R_{nn} = y_n(1 - y_n)$.

IRLS

- Формула шага оптимизации:

$$w^{\text{new}} = w^{\text{old}} - \left(\Phi^\top R \Phi\right)^{-1} \Phi^\top (y - t) = \left(\Phi^\top R \Phi\right)^{-1} \Phi^\top R z,$$

  где $z = \Phi w^{\text{old}} - R^{-1}(y - t)$.
- Получилось как бы решение взвешенной задачи минимизации квадратического отклонения с матрицей весов R.
- Отсюда название: iterative reweighted least squares (IRLS).

Несколько классов

- В случае нескольких классов

$$p(C_k \mid \phi) = y_k(\phi) = \frac{e^{a_k}}{\sum_j e^{a_j}} \text{ для } a_k = w_k^\top \phi.$$

- Опять выпишем максимальное правдоподобие; во-первых,

$$\frac{\partial y_k}{\partial a_j} = y_k \left([k = j] - y_j \right).$$

Несколько классов

- Теперь запишем правдоподобие – для схемы кодирования 1-of-K будет целевой вектор t_n и правдоподобие

$$p(T \mid w_1, \dots, w_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k \mid \phi_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

для $y_{nk} = y_k(\phi_n)$; берём логарифм:

$$E(w_1, \dots, w_K) = -\ln p(T \mid w_1, \dots, w_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}, \text{ и}$$

$$\nabla_{w_j} E(w_1, \dots, w_K) = -\sum_{n=1}^{N} \left(y_{nj} - t_{nj}\right) \phi_n.$$

Несколько классов

- Оптимизировать опять можно по Ньютону-Рапсону; гессиан получится как

$$\nabla_{w_k} \nabla_{w_j} E(w_1, \dots, w_K) = -\sum_{n=1}^{N} y_{nk} \left([k = j] - y_{nj} \right) \phi_n \phi_n^\top.$$

Пробит-регрессия

- А что если у нас другая форма сигмоида?
- Мы по-прежнему в той же постановке: два класса, $p(t = 1 \mid a) = f(a)$, $a = w^\top \phi$, f – функция активации.
- Давайте установим функцию активации с порогом θ: для каждого ϕ_n, вычисляем $a_n = w^\top \phi_n$, и

$$\begin{cases} t_n = 1, & \text{если } a_n \geq \theta, \\ t_n = 0, & \text{если } a_n < \theta. \end{cases}$$

Пробит-регрессия

- Если θ берётся по распределению $p(\theta)$, это соответствует

$$f(a) = \int_{-\infty}^{a} p(\theta)\mathrm{d}\theta.$$

- Пусть, например, $p(\theta)$ – гауссиан с нулевым средним и единичной дисперсией. Тогда

$$f(a) = \Phi(a) = \int_{-\infty}^{a} N\left(\theta \mid 0, 1\right)\mathrm{d}\theta.$$

Пробит-регрессия

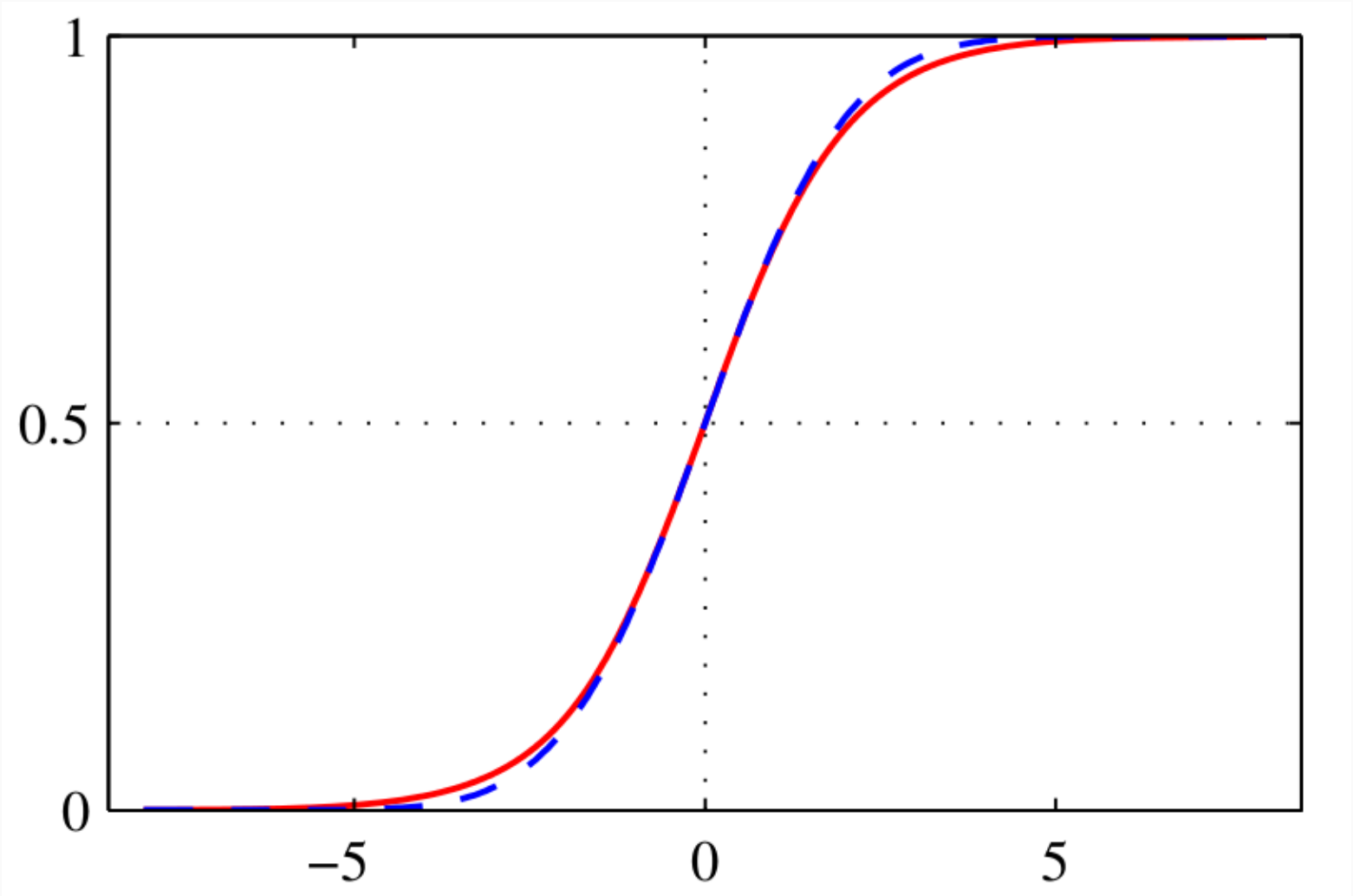
- Это называется *пробит-функцией* (probit); неэлементарная, но тесно связана с

$$\operatorname{erf}(a) = \frac{2}{\sqrt{\pi}} \int_0^a e^{-\frac{\theta^2}{2}} \mathrm{d}\theta :$$

$$\Phi(a) = \frac{1}{2}\left[1 + \frac{1}{\sqrt{2}}\operatorname{erf}(a)\right].$$

- Пробит-регрессия – это модель с пробит-функцией активации.

σ и Φ



Лапласовская аппроксимация и байесовская логистическая регрессия

Лапласовская аппроксимация

- Небольшое лирическое отступление: как приблизить сложное распределение простым?
- Например, как приблизить гауссианом возле максимума? (естественная задача)
- Рассмотрим пока распределение от одной непрерывной переменной $p(z) = \frac{1}{Z} f(z)$.

Лапласовская аппроксимация

- Первый шаг: найдём максимум z_0.
- Второй шаг: разложим в ряд Тейлора

$$\ln f(z) \approx \ln f(z_0) - \frac{1}{2} A(z - z_0)^2, \text{ где } A = -\frac{d^2}{dz^2} \ln f(z) \mid_{z=z_0} .$$

- Третий шаг: приблизим

$$f(z) \approx f(z_0) e^{-\frac{A}{2}(z-z_0)^2},$$

и после нормализации это будет как раз гауссиан.

Лапласовская аппроксимация

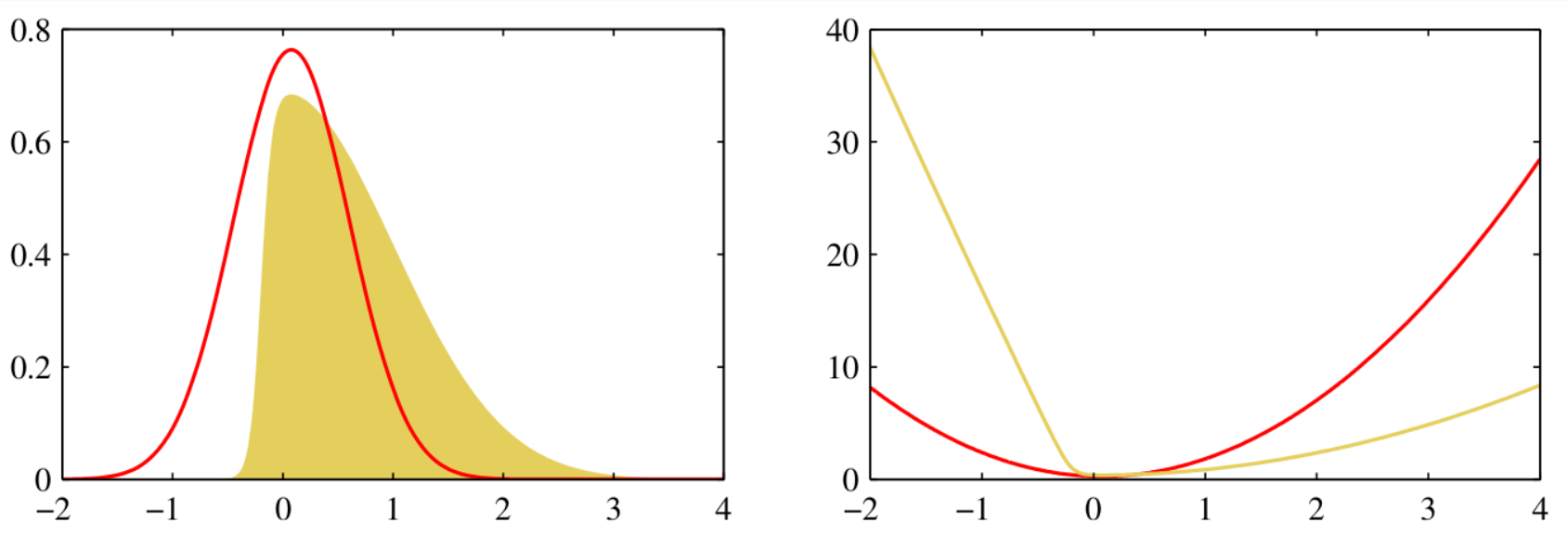
- Это можно обобщить на многомерное распределение $p(z) = \frac{1}{Z} f(z)$:

$$f(z) \approx f(z_0) e^{-\frac{1}{2}(z-z_0)^\top A (z-z_0)},$$

$$\text{где } A = -\nabla\nabla \ln f(z) \,|_{z=z_0} \,.$$

Упражнение. Какая здесь будет нормировочная константа?

Лапласовская аппроксимация



Сравнение моделей по Лапласу

- Вооружившись лапласовской аппроксимацией, давайте применим её сначала к выбору моделей.
- Напомним: чтобы сравнить модели из множества $\{M_i\}_{i=1}^L$, по тестовому набору D оценим апостериорное распределение

$$p(M_i \mid D) \propto p(M_i)p(D \mid M_i).$$

- Если модель определена параметрически, то $p(D \mid M_i) = \int p(D \mid \theta, M_i)p(\theta \mid M_i)d\theta$.
- Это вероятность сгенерировать D, если выбирать параметры модели по её априорному распределению; знаменатель из теоремы Байеса:

$$p(\theta \mid M_i, D) = \frac{p(D \mid \theta, M_i)p(\theta \mid M_i)}{p(D \mid M_i)}.$$

Сравнение моделей по Лапласу

- Мы раньше приближали фактически кусочно-постоянной функцией.
- Теперь давайте гауссианом приблизим; возьмём интеграл:

$$Z = \int f(z)dz \approx \int f(z_0)e^{-\frac{1}{2}(z-z_0)^\top A(z-z_0)}dz = f(z_0)\frac{(2\pi)^{M/2}}{|A|^{1/2}}.$$

- А у нас $Z = p(D)$, $f(\theta) = p(D \mid \theta)p(\theta)$.

Сравнение моделей по Лапласу

- Получаем

$$\ln p(D) \approx \ln p(D \mid \theta_{\text{MAP}}) + \ln P(\theta_{\text{MAP}}) + \frac{M}{2} \ln(2\pi) - \frac{1}{2} \ln |A|.$$

- $\ln P(\theta_{\text{MAP}}) + \frac{M}{2} \ln(2\pi) - \frac{1}{2} \ln |A|$ – *фактор Оккама.*
- $A = -\nabla\nabla \ln p(D \mid \theta_{\text{MAP}}) p(\theta_{\text{MAP}}) = -\nabla\nabla \ln p(\theta_{\text{MAP}} \mid D).$

Сравнение моделей по Лапласу

- Получаем

$$\ln p(D) \approx \ln p(D \mid \theta_{\text{MAP}}) + \ln P(\theta_{\text{MAP}}) + \frac{M}{2}\ln(2\pi) - \frac{1}{2}\ln|A|.$$

- Если гауссовское априорное распределение $p(\theta)$ достаточно широкое, и A полного ранга, то можно грубо приблизить (докажите это!)

$$\ln p(D) \approx \ln p(D \mid \theta_{\text{MAP}}) - \frac{1}{2}M \ln N,$$

  где M – число параметров, N – число точек в D, а аддитивные константы мы опустили.
- Это *байесовский информационный критерий* (Bayesian information criterion, BIC), он же *критерий Шварца* (Schwarz criterion).

Байесовская логистическая регрессия

- Теперь давайте обработаем логистическую регрессию по-байесовски.
- Логистическую регрессию так просто не выпишешь, как линейную – точного ответа из произведения логистических сигмоидов не получается.
- Будем приближать по Лапласу.

Байесовская логистическая регрессия

- Априорное распределение выберем гауссовским:

$$p(w) = N(w \mid \mu_0, \Sigma_0).$$

- Тогда апостериорное будет

$$\begin{aligned} p(w \mid t) \propto & p(w)p(t \mid w), \text{ и} \\ \ln p(w \mid t) = & -\frac{1}{2}\left(w - \mu_0\right)^\top \Sigma_0^{-1}\left(w - \mu_0\right) \\ & + \sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\right] + \text{const}, \\ \text{где } y_n = & \sigma(w^\top \phi_n). \end{aligned}$$

Байесовская логистическая регрессия

- Чтобы приблизить, сначала находим максимум w_{MAP}, а потом матрица ковариаций – это матрица вторых производных

$$\Sigma_N = -\nabla\nabla \ln p(w \mid t) = \Sigma_0^{-1} + \sum_{n=1}^{N} y_n(1 - y_n)\phi_n\phi_n^\top.$$

- Наше приближение – это

$$q(w) = N(w \mid w_{\text{MAP}}, \Sigma_N).$$

Байесовская логистическая регрессия

- Теперь можно описать байесовское предсказание:

$$p(C_1 \mid \phi, t) = \int p(C_1 \mid \phi, w) p(w \mid t) dw \approx \int \sigma(w^\top \phi) q(w) \mathrm{d}w.$$

- Заметим, что $\sigma(w^\top \phi)$ зависит от w только через его проекцию на ϕ.
- Обозначим $a = w^\top \phi$:

$$\sigma(w^\top \phi) = \int \delta(a - w^\top \phi) \sigma(a) \mathrm{d}a.$$

Байесовская логистическая регрессия

- $\sigma(w^\top \phi) = \int \delta(a - w^\top \phi)\sigma(a)\mathrm{d}a$, а значит,

$$\int \sigma(w^\top \phi)q(w)dw = \int \sigma(a)p(a)\mathrm{d}a,$$

$$\text{где } p(a) = \int \delta(a - w^\top \phi)q(w)\mathrm{d}w.$$

- $p(a)$ – это маргинализация гауссиана $q(w)$, где мы интегрируем по всему, что ортогонально ϕ.

Байесовская логистическая регрессия

- $p(a)$ – это маргинализация гауссиана $q(w)$, где мы интегрируем по всему, что ортогонально ϕ.
- Значит, $p(a)$ – тоже гауссиан; найдём его моменты:

$$\begin{aligned}\mu_a =& \mathrm{E}[a] = \int a p(a) \mathrm{d}a = \int q(w) w^\top \phi \mathrm{d}w = w_{\mathrm{MAP}}^\top \phi, \\ \sigma_a^2 =& \int \left(a^2 - \mathrm{E}[a]\right)^2 p(a) \mathrm{d}a = \\ =& \int q(w) \left[(w^\top \phi)^2 - (\mu_N^\top \phi)^2\right]^2 \mathrm{d}w = \phi^\top \Sigma_N \phi.\end{aligned}$$

- Итого получили, что

$$p(C_1 \mid t) = \int \sigma(a) p(a) \mathrm{d}a = \int \sigma(a) N(a \mid \mu_a, \sigma_a^2) \mathrm{d}a.$$

Байесовская логистическая регрессия

- $p(C_1 \mid t) = \int \sigma(a) N(a \mid \mu_a, \sigma_a^2) \mathrm{d}a.$
- Этот интеграл так просто не взять, потому что сигмоид сложный, но можно приблизить, если приблизить $\sigma(a)$ через пробит: $\sigma(a) \approx \Phi(\lambda a)$ для $\lambda = \sqrt{\pi/8}$.

Упражнение. Докажите, что для $\lambda = \sqrt{\pi/8}$ у σ и Φ одинаковый наклон в нуле.

Байесовская логистическая регрессия

- А если мы перейдём к пробит-функции, то её свёртка с гауссианом будет просто другим пробитом:

$$\int \Phi(\lambda a) N(a \mid \mu, \sigma^2) \mathrm{d}a = \Phi\left(\frac{\mu}{\sqrt{\frac{1}{\lambda^2} + \sigma^2}}\right).$$

Упражнение. Докажите это.

Байесовская логистическая регрессия

- В итоге получается аппроксимация

$$\int \sigma(a) N(a \mid \mu, \sigma^2) \mathrm{d}a \approx \sigma\left(\kappa(\sigma^2)\mu\right),$$

$$\text{где } \kappa(\sigma^2) = \frac{1}{\sqrt{1 + \frac{\pi}{8}\sigma^2}}.$$

Байесовская логистическая регрессия

- И теперь, собирая всё вместе, мы получили распределение предсказаний:

$$
\begin{aligned}
p(C_1 \mid \phi, t) &= \sigma\left(\kappa(\sigma_a^2)\mu_a\right), \text{ где} \\
\mu_a &= w_{\text{MAP}}^\top \phi, \\
\sigma_a^2 &= \phi^\top \Sigma_N \phi, \\
\kappa(\sigma^2) &= \frac{1}{\sqrt{1 + \frac{\pi}{8}\sigma^2}}.
\end{aligned}
$$

- Кстати, разделяющая поверхность $p(C_1 \mid \phi, t) = \frac{1}{2}$ задаётся уравнением $\mu_a = 0$, и тут нет никакой разницы с просто использованием w_{MAP}. Разница будет только для более сложных критериев.

SVM и задача линейной классификации

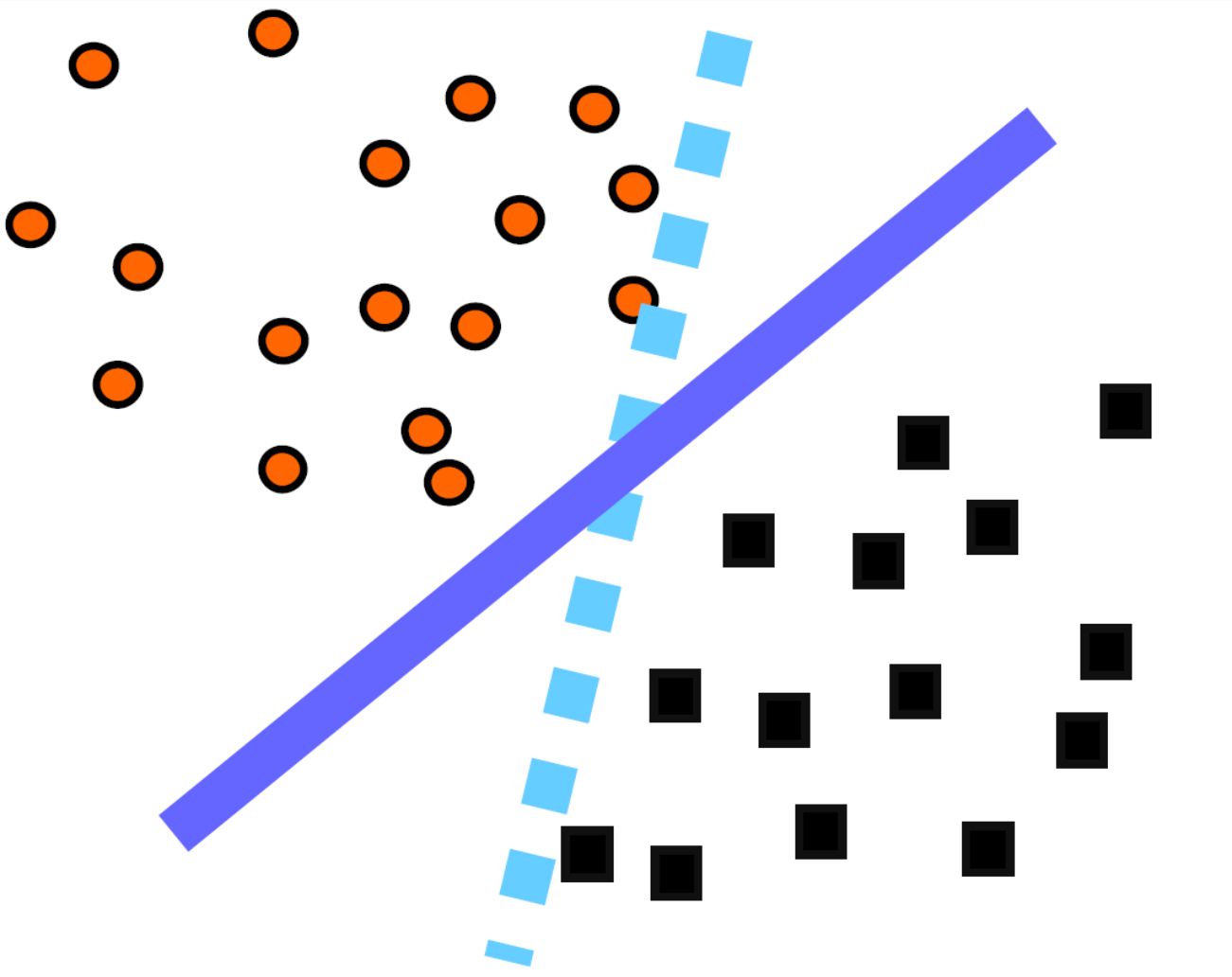
Постановка задачи

- Метод опорных векторов решает задачу классификации.
- Каждый элемент данных — точка в n–мерном пространстве $\mathbb{R}^n$.
- Формально: есть точки x_i, $i = 1..m$, у точек есть метки $y_i = \pm 1$.
- Мы интересуемся: можно ли разделить данные $(n-1)$–мерной гиперплоскостью, а также хотим найти эту гиперплоскость.
- Это всё?

Постановка задачи

- Нет, ещё хочется научиться разделять этой гиперплоскостью *как можно лучше*.
- То есть желательно, чтобы два разделённых класса лежали как можно дальше от гиперплоскости.
- Практическое соображение: тогда от небольших возмущений в гиперплоскости ничего не испортится.

Пример



Выпуклые оболочки

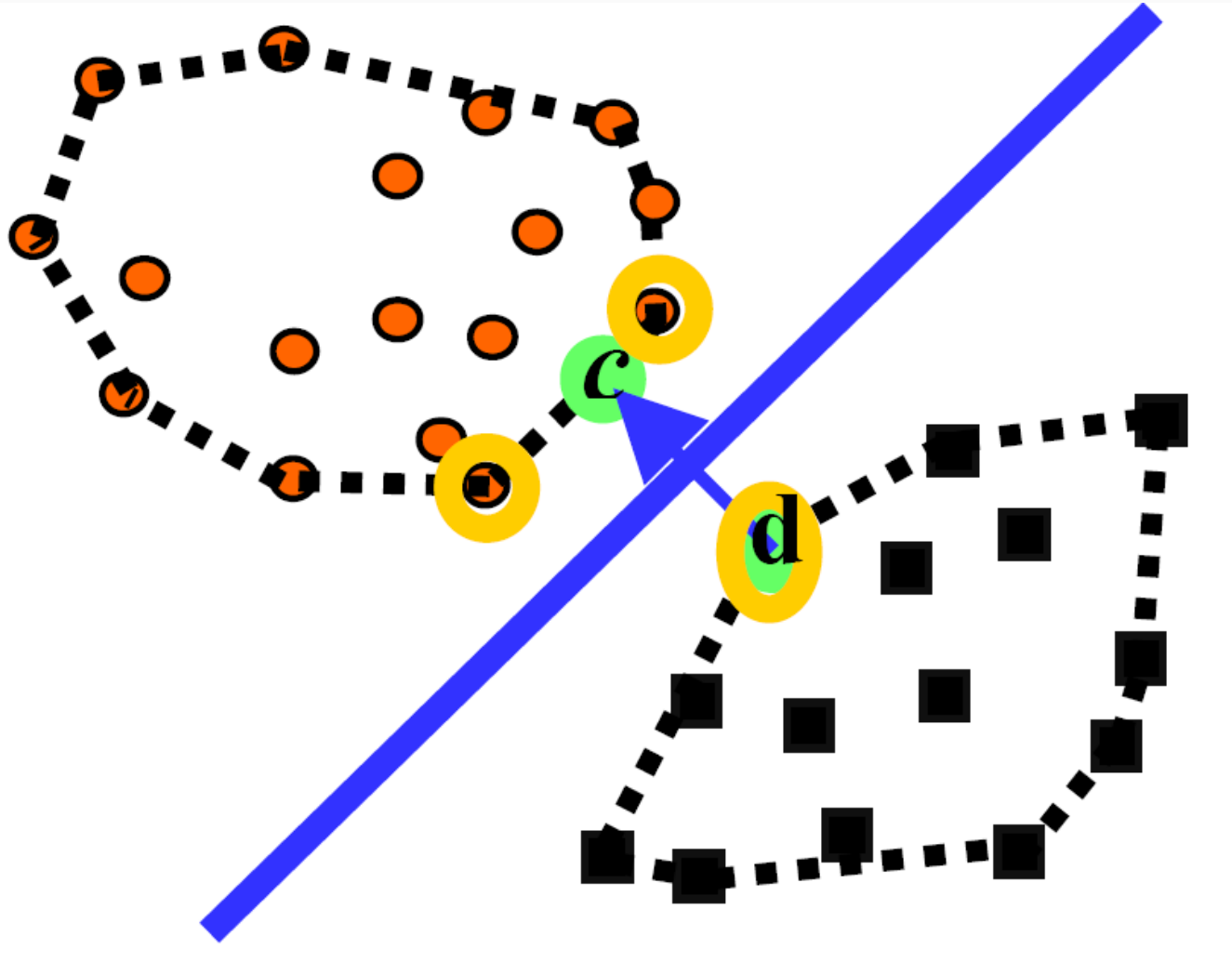
- Один подход: найти две ближайшие точки в выпуклых оболочках данных, а затем провести разделяющую гиперплоскость через середину отрезка.
- Формально это превращается в задачу квадратичной оптимизации:

$$\min_{\alpha} \left\{ ||c-d||^2, \text{ где } c = \sum_{y_i=1} \alpha_i x_i, d = \sum_{y_i=-1} \alpha_i x_i \right\}$$

$$\text{при условии } \sum_{y_i=1} \alpha_i = \sum_{y_i=-1} \alpha_i = 1, \alpha_i \geq 0.$$

- Эту задачу можно решать общими оптимизационными алгоритмами.

Пример



Максимизация зазора

- Другой подход: максимизировать *зазор* (margin) между двумя параллельными опорными плоскостями, затем провести им параллельную на равных расстояниях от них.
- Гиперплоскость называется *опорной* для множества точек X, если все точки из X лежат под одну сторону от этой гиперплоскости.
- Формально: расстояние от точки до гиперплоскости $y(x) = w^\top x + w_0 = 0$ равно $\frac{|y(x)|}{\|w\|}$.

Максимизация зазора

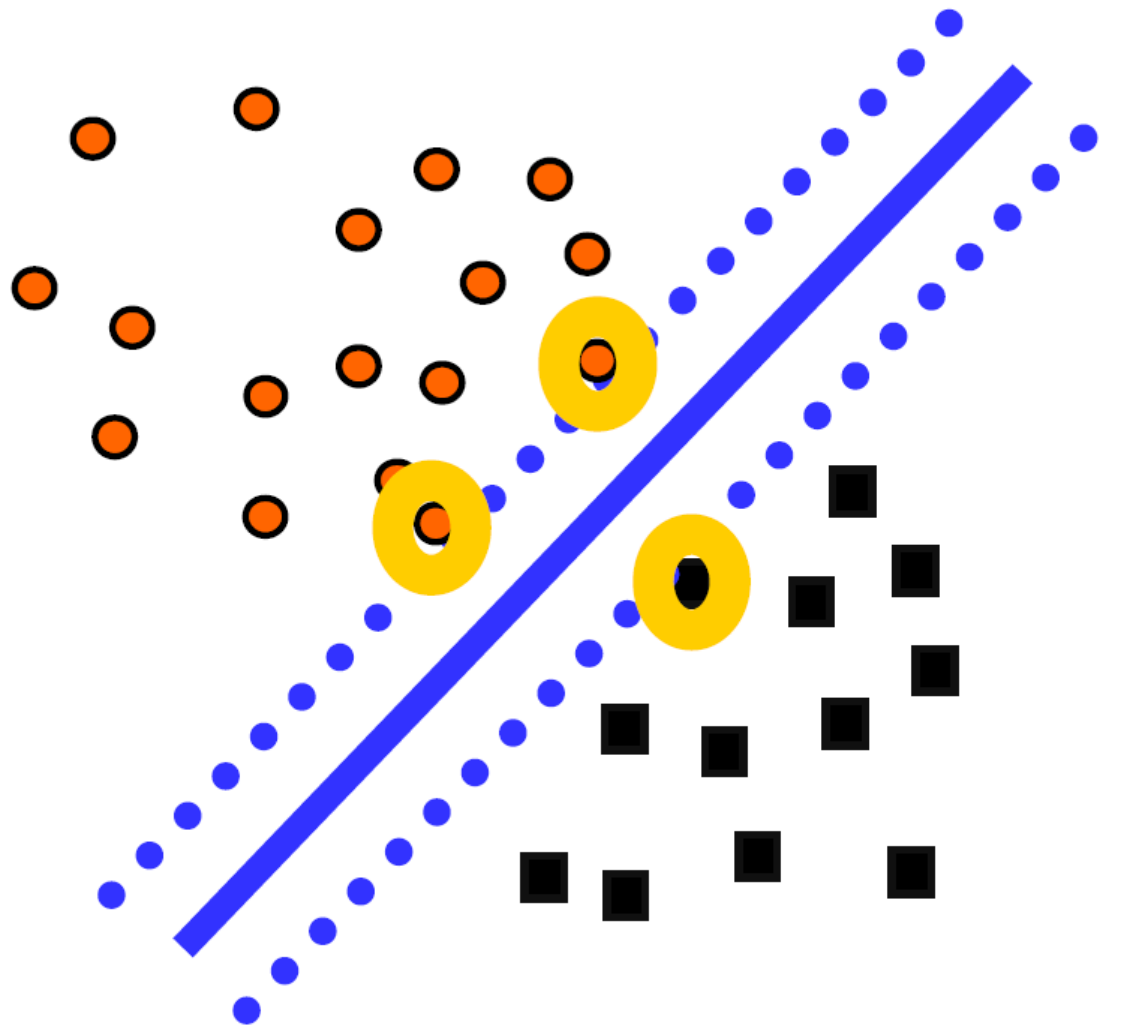
- Расстояние от точки до гиперплоскости $y(x) = w^\top x + w_0 = 0$ равно $\frac{|y(x)|}{\|w\|}$.
- Все точки классифицированы правильно: $t_n y(x_n) > 0$ ($t_n \in \{-1, 1\}$).
- И мы хотим найти

$$\arg\max_{w,w_0} \min_n \frac{t_n y(x_n)}{\|w\|} = \\ = \arg\max_{w,w_0} \left\{ \frac{1}{\|w\|} \min_n \left[t_n (w^\top x_n + w_0) \right] \right\}.$$

Максимизация зазора

- $\arg\max_{w,w_0} \left\{ \frac{1}{\|w\|} \min_n \left[t_n(w^\top x_n + w_0) \right] \right\}$. Сложно.
- Но если перенормировать w, гиперплоскость не изменится.
- Давайте перенормируем так, чтобы $\min_n \left[t_n(w^\top x_n + w_0) \right] = 1$.

Пример



Максимизация зазора

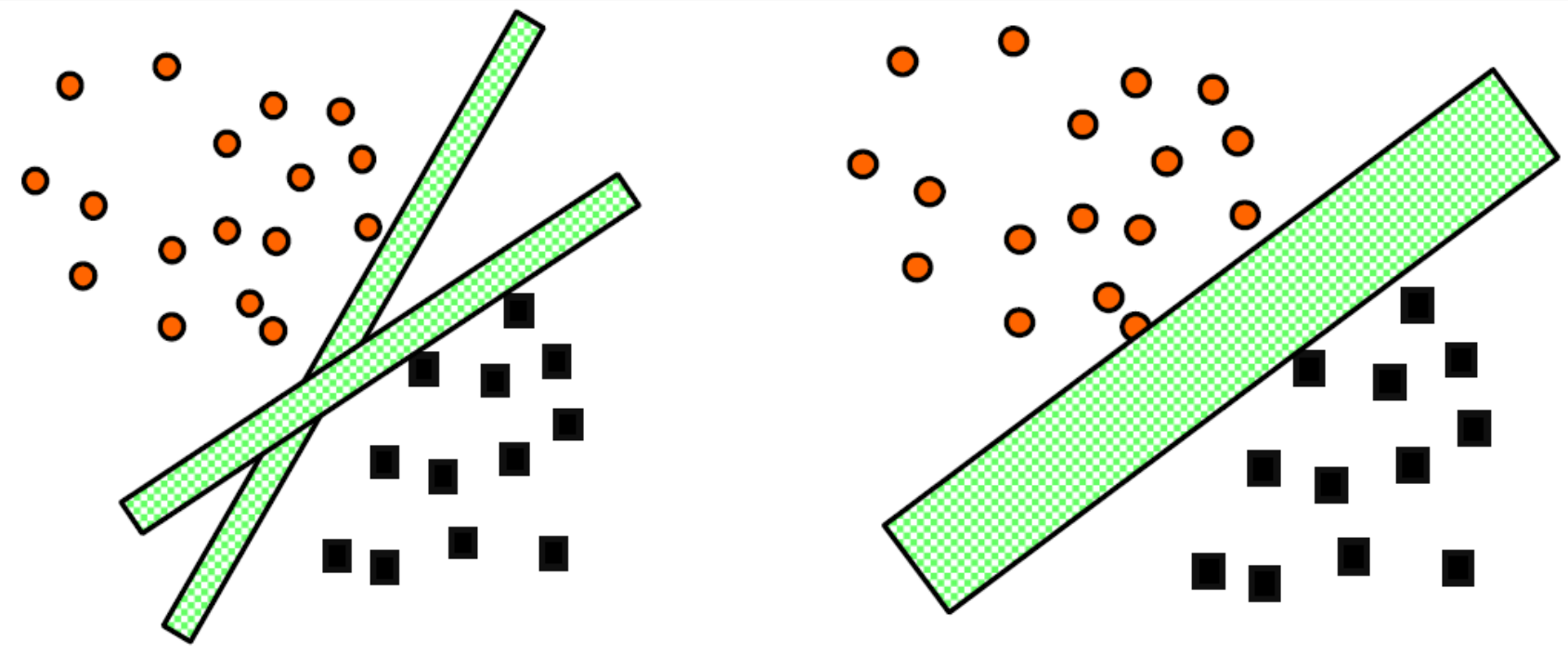
- Получается тоже задача квадратичного программирования:

$$\min_{\vec{w},b}\left\{\frac{1}{2}||w||^2\right\} \text{ при условии } t_n(w^\top x_n + w_0) \geq 1.$$

Результаты

- Результаты получаются хорошие. Такой подход позволяет находить *устойчивые* решения, что во многом решает проблемы с оверфиттингом и позволяет лучше предсказывать дальнейшую классификацию.
- В каком-то смысле в решениях с «толстыми» гиперплоскостями между данными содержится больше информации, чем в «тонких», потому что «толстых» меньше.
- Это всё можно сформулировать и доказать (позже).

Пример



Дуальные задачи

- Напомним, что такое дуальные задачи.
- Прямая задача оптимизации:

$$\min\{f(x)\} \text{ при условии } h(x) = 0,\ g(x) \leq 0,\ x \in X.$$

- Для дуальной задачи вводим параметры λ, соответствующие равенствам, и μ, соответствующие неравенствам.

Дуальные задачи

- Прямая задача оптимизации:

$$\min\{f(x)\} \text{ при условии } h(x) = 0,\ g(x) \leq 0,\ x \in X.$$

- Дуальная задача оптимизации:

$$\min\{\phi(\lambda, \mu)\} \text{ при условии } \mu \geq 0,$$
$$\text{где } \phi(\lambda, \mu) = \inf_{x \in X} \left\{ f(x) + \lambda^\top h(x) + \mu^\top g(x) \right\}.$$

Дуальные задачи

- Тогда, если $(\bar{\lambda}, \bar{\mu})$ – допустимое решение дуальной задачи, а $\bar{x}$ – допустимое решение прямой, то

$$\phi(\bar{\lambda}, \bar{\mu}) = \inf_{x \in X} \left\{ f(x) + \bar{\lambda}^\top h(x) + \bar{\mu}^\top g(x) \right\} \leq$$
$$\leq f(\bar{x}) + \bar{\lambda}^\top h(\bar{x}) + \bar{\mu}^\top g(\bar{x}) \leq f(\bar{x}).$$

- Это называется *слабой дуальностью* (только $\leq$), но во многих случаях достигается и равенство.

Дуальные задачи

- Для линейного программирования прямая задача:

$$\min c^\top x \text{ при условии } Ax = b,\ x \in X = \{x \leq 0\}.$$

- Тогда дуальная задача получается так:

$$\phi(\lambda) = \inf_{x \geq 0} \left\{ c^\top x + \lambda^\top (b - Ax) \right\} =$$

$$= \lambda^\top b + \inf_{x \geq 0} \left\{ (c^\top - \lambda^\top A) x \right\} =$$

$$= \begin{cases} \lambda^\top b, & \text{если } c^\top - \lambda^\top A \geq 0, \\ -\infty & \text{в противном случае.} \end{cases}$$

Дуальные задачи

- Для линейного программирования прямая задача:

$$\min\left\{c^\top x\right\} \text{ при условии } Ax = b,\ x \in X = \{x \leq 0\}.$$

- Дуальная задача:

$$\max\left\{b^\top \lambda\right\} \text{ при условии } A^\top \lambda \leq c,\ \lambda \text{ не ограничены.}$$

Дуальные задачи

- Для квадратичного программирования прямая задача:

$$\min\left\{\frac{1}{2}x^\top Qx + c^\top x\right\} \text{ при условии } Ax \leq b,$$

где Q – положительно полуопределённая матрица (т.е. $x^\top Qx \geq 0$ всегда).

- Дуальная задача (проверьте):

$$\max\left\{\frac{1}{2}\mu^\top D\mu + \mu^\top d - \frac{1}{2}c^\top Q^{-1}c\right\} \text{ при условии } c \geq 0,$$

где $D = -AQ^{-1}A^\top$ (отрицательно определённая матрица), $d = -b - AQ^{-1}c$.

Дуальная задача к SVM

- В случае SVM надо ввести множители Лагранжа:

$$L(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 - \sum_n \alpha_n \left[t_n(w^\top x_n + w_0) - 1\right], \quad \alpha_n \geq 0.$$

- Берём производные по w и w_0, приравниваем нулю, получаем

$$w = \sum_n \alpha_n t_n x_n,$$
$$0 = \sum_n \alpha_n t_n.$$

Дуальная задача к SVM

- Подставляя в $L(w, w_0, \alpha)$, получим

$$L(\alpha) = \sum_n \alpha_n - \frac{1}{2} \sum_n \sum_m \alpha_n \alpha_m t_n t_m \left(x_n^\top x_m\right)$$

при условии $\alpha_n \geq 0, \sum_n \alpha_n t_n = 0.$

- Это дуальная задача, которая обычно в SVM и используется.

Предсказание и ККТ

- А для предсказания потом надо посмотреть на знак $y(x)$:

$$y(x) = \sum_{n=1}^{N} \alpha_n t_n x^\top x_n + w_0.$$

- Получилось, что предсказания зависят от всех точек x_n...

Предсказание и ККТ

- ...но нет. :) Условия ККТ (Karush–Kuhn–Tucker):

$$\begin{aligned} \alpha_n &\geq 0, \\ t_n y(x_n) - 1 &\geq 0, \\ \alpha_n \left(t_n y(x_n) - 1 \right) &= 0. \end{aligned}$$

- Т.е. реально предсказание зависит от небольшого числа *опорных* векторов, для которых $t_n y(x_n) = 1$ (они находятся собственно на границе разделяющей поверхности).

Постановка задачи

- Все эти методы работают, когда данные действительно линейно разделимы.
- А что делать, когда их всё-таки немножко не получается разделить?
- Первый вопрос: что делать для первого метода, метода выпуклых оболочек?

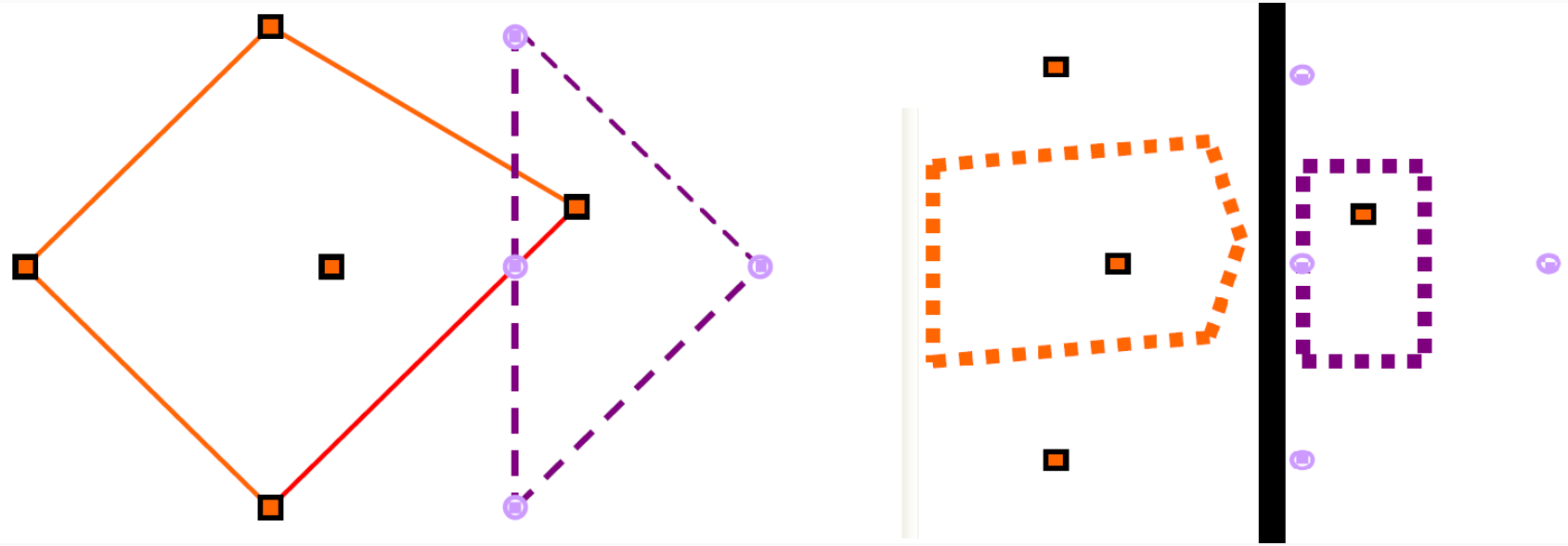
Редуцированные выпуклые оболочки

- Вместо обычных выпуклых оболочек можно рассматривать *редуцированные* (reduced), у которых коэффициенты ограничены не 1, а сильнее:

$$c = \sum_{y_i=1} \alpha_i x_i, \quad 0 \leq \alpha_i \leq D.$$

- Тогда для достаточно малых D редуцированные выпуклые оболочки не будут пересекаться.
- И мы будем искать оптимальную гиперплоскость между редуцированными выпуклыми оболочками.

Пример



Для метода опорных векторов

- Естественно, для метода опорных векторов тоже надо что-то изменить. Что?

Для метода опорных векторов

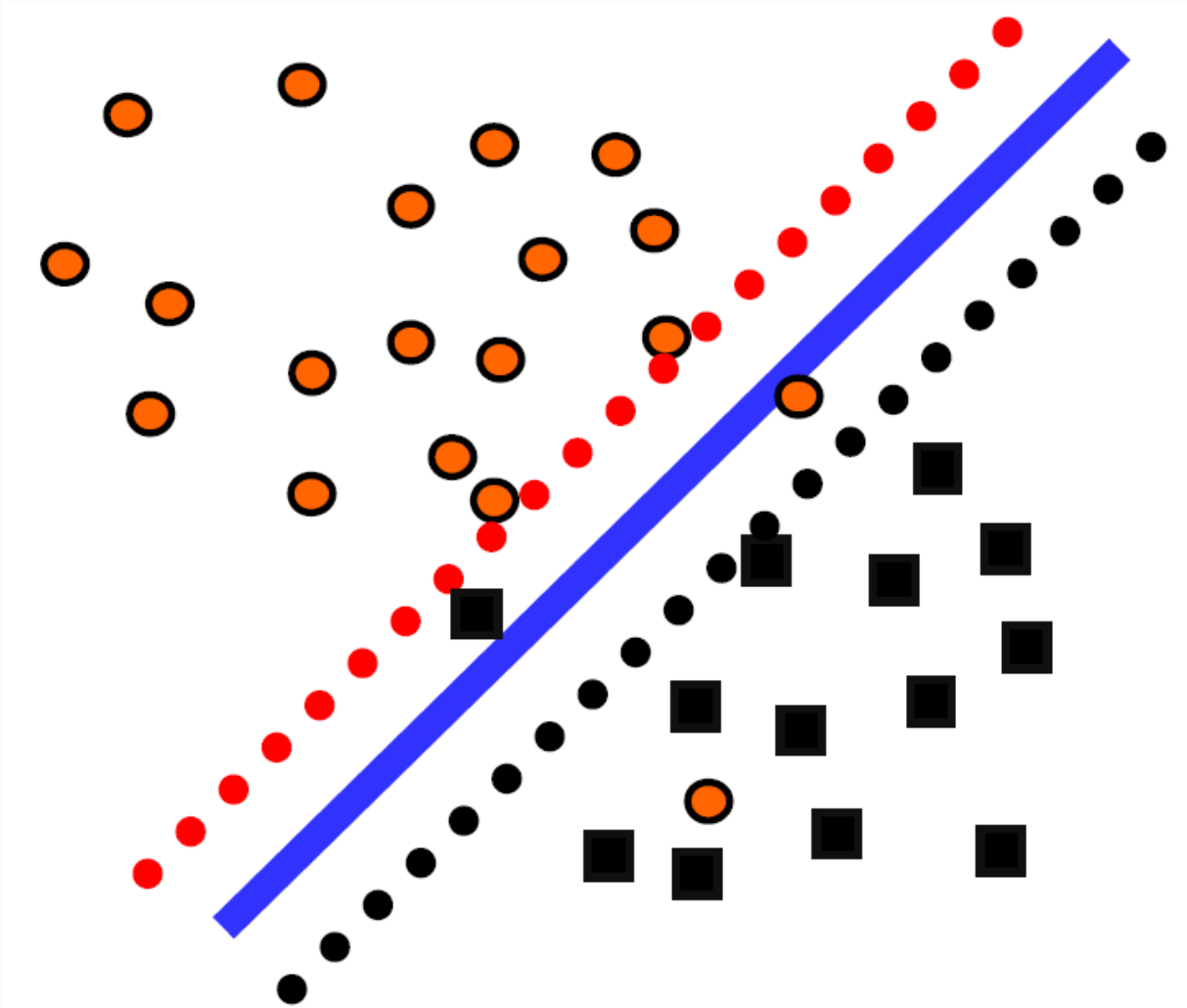
- Естественно, для метода опорных векторов тоже надо что-то изменить. Что?
- Мы просто добавим в оптимизирующуюся функцию неотрицательную ошибку (slack):

$$\min_{\vec{w},w_0}\left\{||\vec{w}||^2 + C\sum_{i=1}^{m} z_i\right\}$$

при условии $t_i(\vec{w}\cdot\vec{x}_i - w_0) + z_i \geq 1.$

- Это прямая задача...

Пример



Дуальная переформулировка

- ...а вот дуальная:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} t_i t_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j \right) - \sum_{i=1}^{m} \alpha_i, \right.$$
$$\left. \text{где } \sum_{i=1}^{m} t_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$$

- Эта формулировка чаще всего используется в теории SVM.
- Единственное отличие от линейно разделимого случая – верхняя граница C на α_j, т.е. на влияние каждой точки.

Итого

- Метод опорных векторов отлично подходит для линейной классификации.
- Решая задачу квадратичного программирования, мы получаем параметры оптимальной гиперплоскости.
- Точно так же, как и в дуальном случае, если бы мы просто искали середину между выпуклыми оболочками.

SVM и эмпирический риск

- Ещё один взгляд на SVM — какая вообще задача у любой классификации?
- Мы хотим минимизировать эмпирический риск, то есть число неправильных ответов:

$$\sum_n [y_i \neq t_i] \to \min_w .$$

- И если функция линейная с параметрами w, w_0, то это эквивалентно

$$\sum_n [t_i \left(x_n^\top w - w_0\right) < 0] \to \min_w .$$

- Величину $M_i = x_n^\top w - w_0$ назовём *отступом* (margin).
- Оптимизировать напрямую сложно...

SVM и эмпирический риск

- ...поэтому заменим на оценку сверху:

$$\sum_n [M_i < 0] \leq \sum_n (1 - M_i) \to \min_w .$$

- А потом ещё добавим регуляризатор для стабильности:

$$\sum_n [M_i < 0] \leq \sum_n (1 - M_i) + \frac{1}{2C}\|w\|^2 \to \min_w .$$

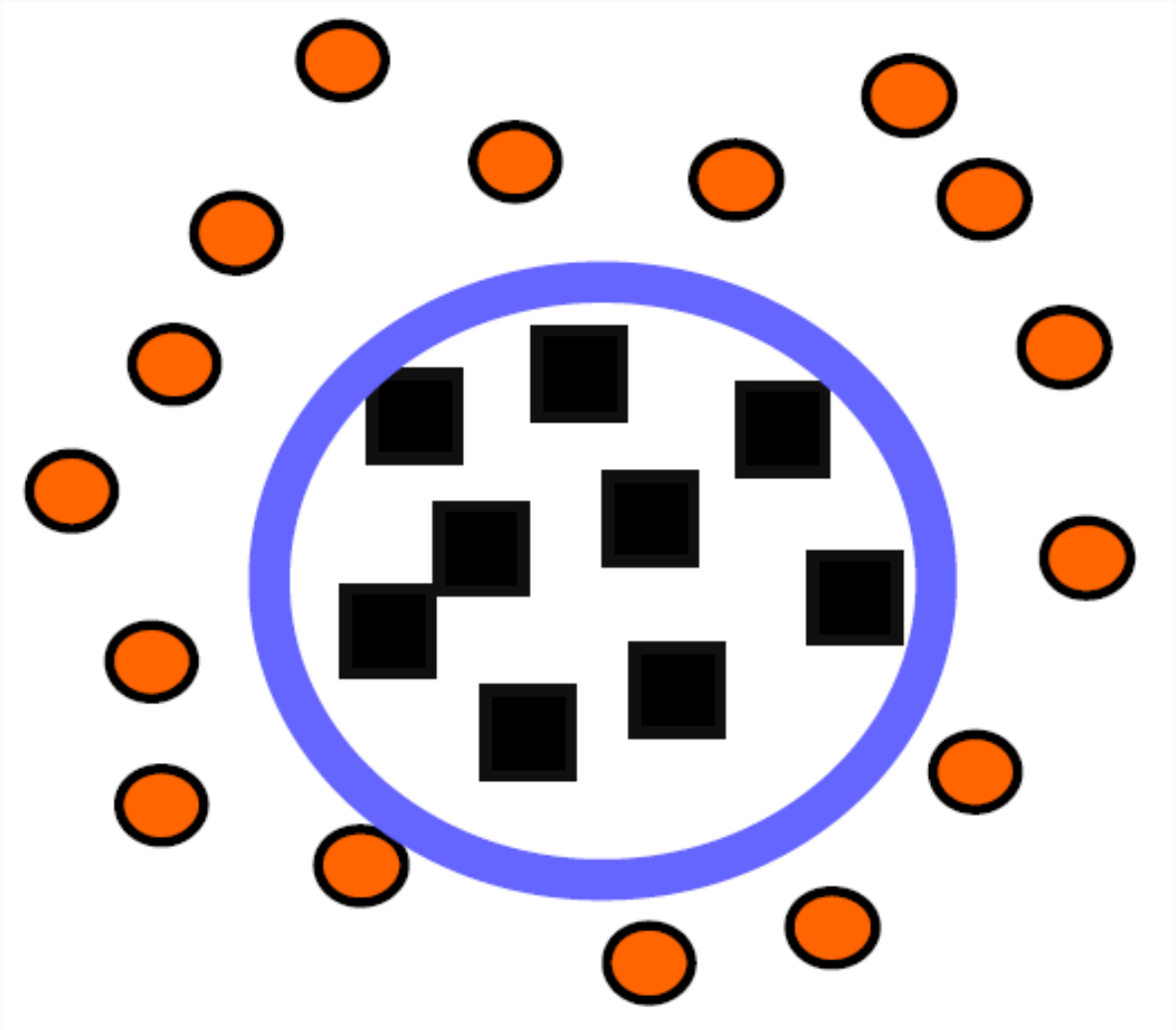
- И это снова получилась задача SVM!

SVM и разделение нелинейными функциями

Введение

- Часто бывает нужно разделять данные не только линейными функциями.
- Что делать в таком случае?

Пример



Введение

- Часто бывает нужно разделять данные не только линейными функциями.
- Классический метод: развернуть нелинейную классификацию в пространство большей размерности (feature space), а там запустить линейный классификатор.
- Для этого просто нужно для каждого монома нужной степени ввести новую переменную.

Пример

- Чтобы в двумерном пространстве $[r, s]$ решить задачу классификации квадратичной функцией, надо перейти в пятимерное пространство:

$$[r, s] \longrightarrow [r, s, rs, r^2, s^2].$$

- Или формальнее; определим $\theta : \mathbb{R}^2 \to \mathbb{R}^5$: $\theta(r, s) = (r, s, rs, r^2, s^2)$. Вектор в $\mathbb{R}^5$ теперь соответствует квадратичной кривой общего положения в $\mathbb{R}^2$, а функция классификации выглядит как

$$f(\vec{x}) = \operatorname{sign}(\theta(\vec{w}) \cdot \theta(\vec{x}) - b).$$

- Если решить задачу линейного разделения в этом новом пространстве, тем самым решится задача квадратичного разделения в исходном.

Проблемы с классическим подходом

- Во-первых, количество переменных растёт экспоненциально.
- Во-вторых, по большому счёту теряются преимущества того, что гиперплоскость именно оптимальная; например, оверфиттинг опять становится проблемой.
- Важное замечание: *концептуально* мы задачу уже решили. Остались *технические* сложности: как обращаться с гигантской размерностью. Но в них-то всё и дело.

Основная идея и схема работы SVM

- Тривиальная схема алгоритма классификации такова:
  - входной вектор $\vec{x}$ трасформируется во входной вектор в feature space (большой размерности);
  - в этом большом пространстве мы вычисляем опорные векторы, решаем задачу разделения;
  - потом по этой задаче классифицируем входной вектор.
- Это нереально, потому что пространство слишком большой размерности.

Основная идея и схема работы SVM

- Оказывается, кое-какие шаги здесь можно переставить. Вот так:
  - опорные векторы вычисляются в исходном пространстве малой размерности;
  - там же они перемножаются (сейчас увидим, что это значит);
  - и только потом мы делаем нелинейную трансформацию того, что получится;
  - потом по этой задаче классифицируем входной вектор.
- Осталось теперь объяснить, что всё это значит. :)

Постановка задачи

- Напомним, что наша задача поставлена следующим образом:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j \right) - \sum_{i=1}^{m} \alpha_i, \right.$$
$$\left. \text{где } \sum_{i=1}^{m} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$$

Постановка задачи

- Мы теперь хотим ввести некое отображение $\theta : \mathbb{R}^n \to \mathbb{R}^N$, $N > n$. Получится:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \left(\theta(\vec{x}_i) \cdot \theta(\vec{x}_j) \right) - \sum_{i=1}^{m} \alpha_i, \right.$$
$$\left. \text{где } \sum_{i=1}^{m} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$$

Теория Гильберта--Шмидта--Мерсера

- Придётся немножко вспомнить (или изучить) функциональный анализ.
- Мы хотим обобщить понятие *скалярного произведения*; давайте введём новую функцию, которая (минуя трансформацию) будет сразу вычислять скалярное произведение векторов в feature space:

$$k(\vec{u}, \vec{v}) := \theta(\vec{u}) \cdot \theta(\vec{v}).$$

Теория Гильберта--Шмидта--Мерсера

- Первый результат: любая симметрическая функция $k(\vec{u}, \vec{v}) \in L_2$ представляется в виде

$$k(\vec{u}, \vec{v}) = \sum_{i=1}^{\infty} \lambda_i \theta_i(\vec{u}) \cdot \theta_i(\vec{v}),$$

где $\lambda_i \in \mathbb{R}$ — собственные числа, а θ_i — собственные векторы интегрального оператора с ядром k, т.е.

$$\int k(\vec{u}, \vec{v}) \theta_i(\vec{u}) \mathrm{d}\vec{u} = \lambda_i \theta_i(\vec{v}).$$

Теория Гильберта--Шмидта--Мерсера

- Чтобы k задавало скалярное произведение, достаточно, чтобы все собственные числа были положительными. А собственные числа положительны тогда и только тогда, когда (*теорема Мерсера*)

$$\int\int k(\vec{u},\vec{v})g(\vec{u})g(\vec{v})\mathrm{d}\vec{u}\mathrm{d}\vec{v} > 0$$

  для всех g таких, что $\int g^2(\vec{u})\mathrm{d}\vec{u} < \infty$.
- Вот, собственно и всё. Теперь мы можем вместо подсчёта $\theta(\vec{u}) \cdot \theta(\vec{v})$ в задаче квадратичного программирования просто использовать подходящее *ядро* $k(\vec{u},\vec{v})$.

Теория Гильберта--Шмидта--Мерсера

- Итого задача наша выглядит так:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j k(\vec{x}_i, \vec{x}_j) - \sum_{i=1}^{m} \alpha_i, \right.$$
$$\left. \text{где } \sum_{i=1}^{m} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$$

- Просто меняя ядро k, мы можем вычислять самые разнообразные разделяющие поверхности.
- Условия на то, чтобы k была подходящим ядром, задаются теоремой Мерсера.

Примеры ядер

- Рассмотрим ядро
$$k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v})^2.$$
- Какое пространство ему соответствует?

Примеры ядер

- После выкладок получается:

$$k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v})^2 = \\ = \left(u_1^2, u_2^2, \sqrt{2}u_1 u_2\right) \cdot \left(v_1^2, v_2^2, \sqrt{2}v_1 v_2\right).$$

- Иначе говоря, линейная поверхность в новом пространстве соответствует квадратичной поверхности в исходном (эллипс, например).

Примеры ядер

- Естественное обобщение: ядро $k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v})^d$ задаёт пространство, оси которого соответствуют всем *однородным* мономам степени d.
- А как сделать пространство, соответствующее произвольной полиномиальной поверхности, не обязательно однородной?

Примеры ядер

- Поверхность, описывающаяся полиномом степени d:

$$k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v} + 1)^d.$$

- Тогда линейная разделимость в feature space в точности соответствует полиномиальной разделимости в базовом пространстве.

Примеры ядер

- Нормальное распределение (radial basis function):

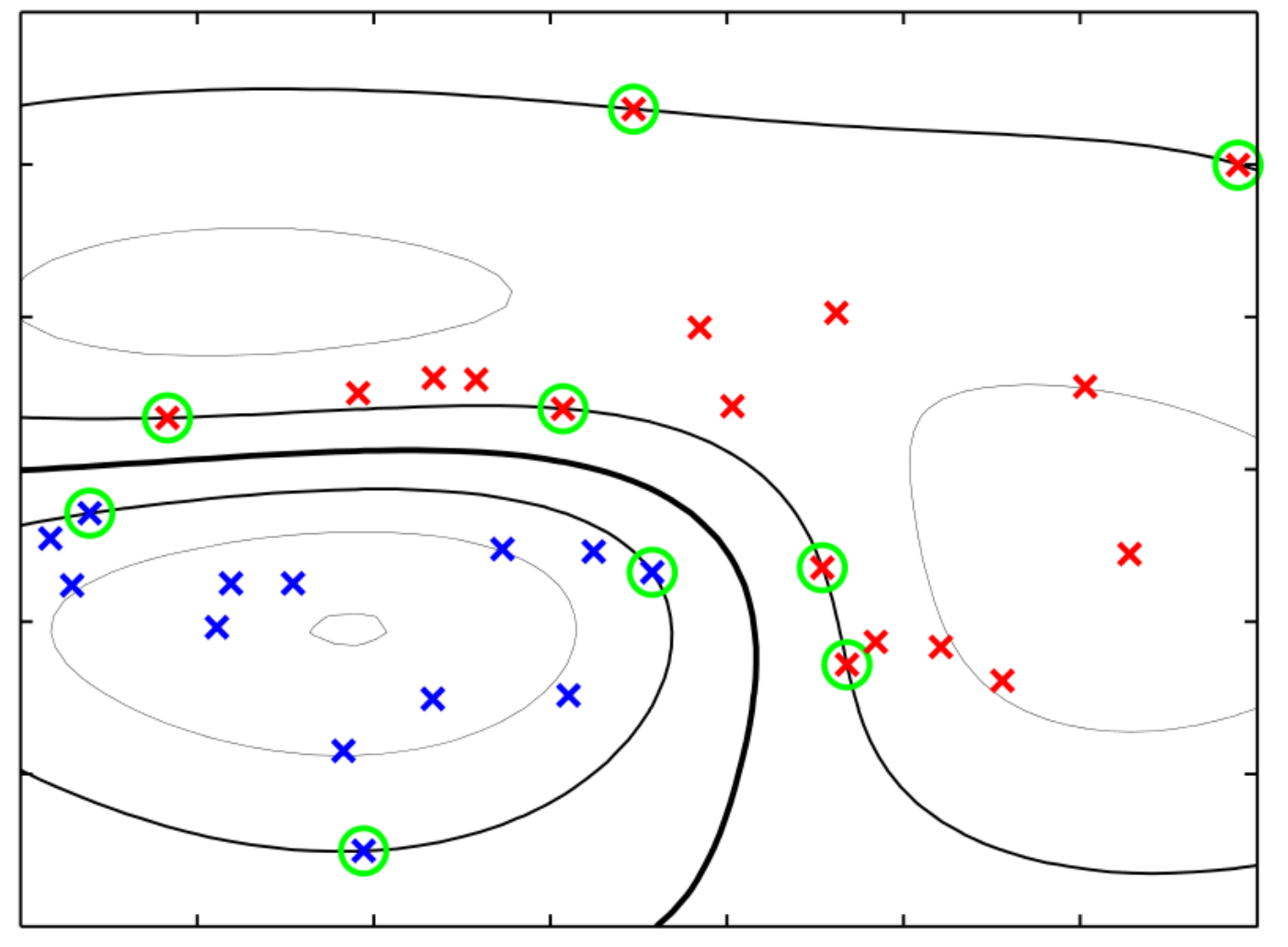
$$k(\vec{u}, \vec{v}) = e^{-\frac{||\vec{u}-\vec{v}||^2}{2\sigma}}.$$

- Двухуровневая нейронная сеть:

$$k(\vec{u}, \vec{v}) = o(\eta\vec{u} \cdot \vec{v} + c),$$

где o — сигмоид.

Пример



Резюме

- Вот какой получается в итоге алгоритм.
  1. Выбрать параметр C, от которого зависит акцент на минимизации ошибки или на максимизации зазора.
  2. Выбрать ядро и параметры ядра, которые у него, возможно, есть.
  3. Решить задачу квадратичного программирования.
  4. По полученным значениям опорных векторов определить w_0 (как именно?).
  5. Новые точки классифицировать как

$$f(\vec{x}) = \text{sign}(\sum_i y_i \alpha_i k(\vec{x}, \vec{x}_i) - w_0).$$

ν-SVM

- Другой вариант для неразделимых данных – ν-SVM [Schölkopf et al., 2000].
- Максимизируем

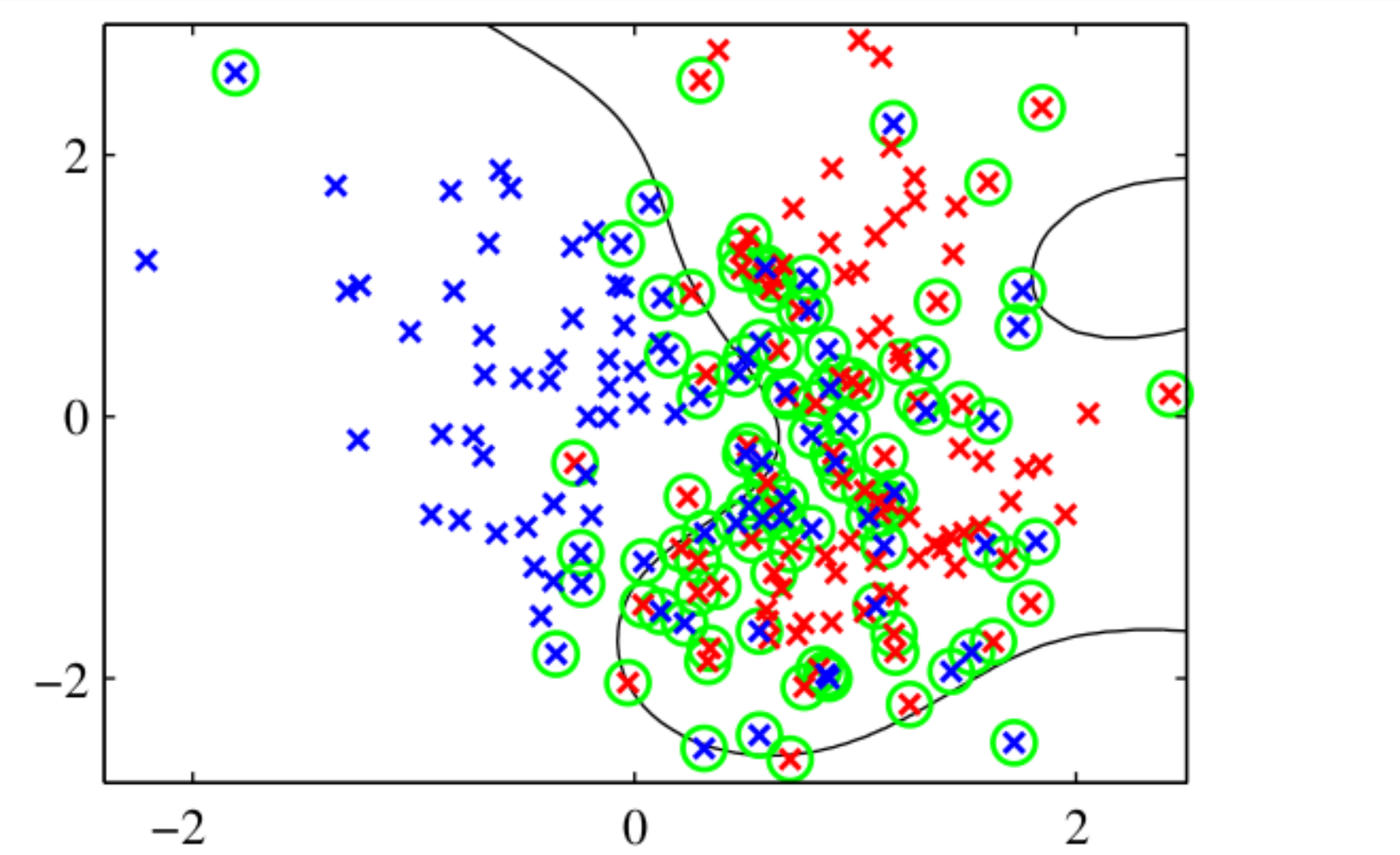
$$L(a) = -\frac{1}{2}\sum_n\sum_m a_n a_m t_n t_m k(x_n, x_m)$$

с ограничениями

$$0 \leq a_n \leq \frac{1}{N},\ \sum_n a_n t_n = 0,\ \sum_n a_n \geq \nu.$$

- Параметр ν можно интерпретировать как верхнюю границу на долю ошибок.

SVM для классификации



Связь с логистической регрессией

- В случае SVM с возможными ошибками мы минимизируем

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|w\|^2.$$

- Для точек с правильной стороны $\xi_n = 0$, с неправильной – $\xi_n = 1 - y_n t_n$.
- Так что можно записать *hinge error function* $E_{SV}(y_n t_n) = [1 - y_n t_n]_+$ и переписать как задачу с регуляризацией

$$\sum_{n=1}^{N} E_{SV}(y_n t_n) + \lambda\|w\|^2.$$

Связь с логистической регрессией

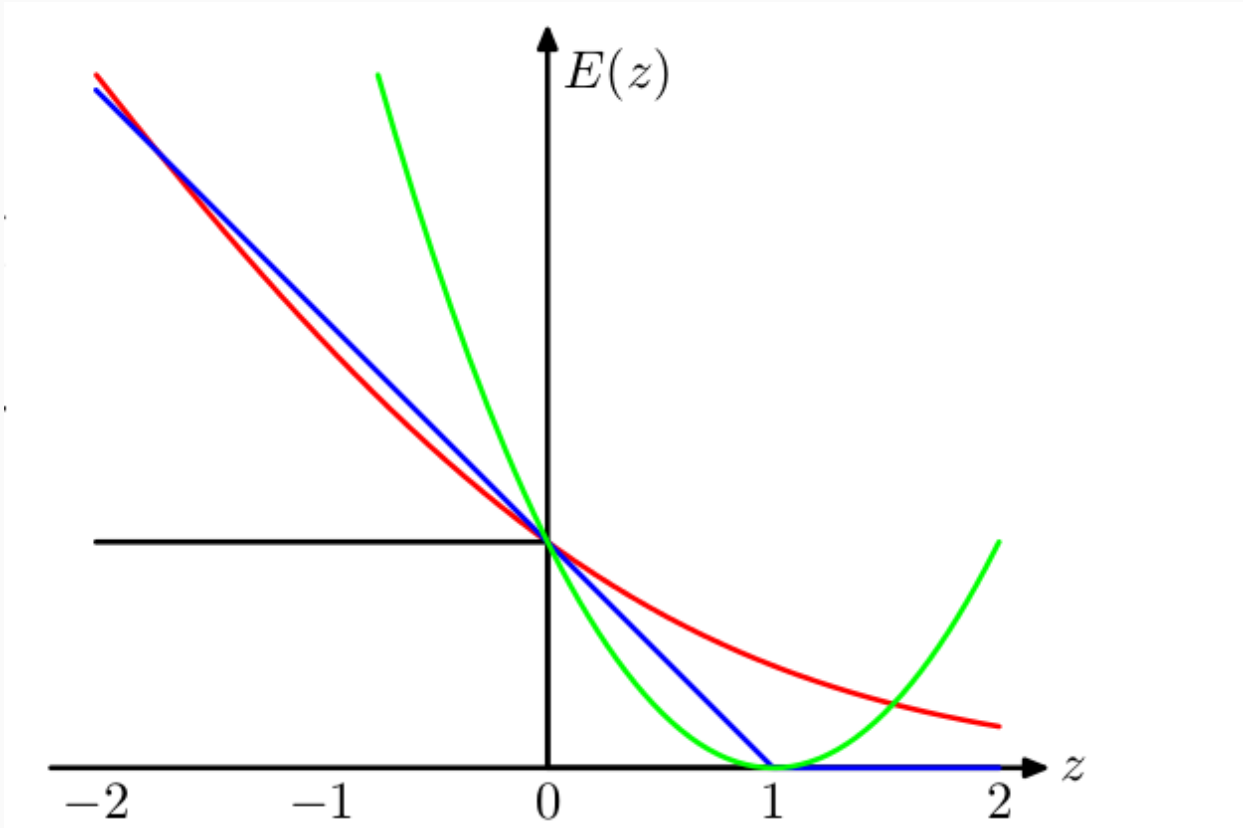
- Вспомним логистическую регрессию и переформулируем её для целевой переменной $t \in \{-1, 1\}$: $p(t = 1 \mid y) = \sigma(y)$, значит, $p(t = -1 \mid y) = 1 - \sigma(y) = \sigma(-y)$, и $p(t \mid y) = \sigma(yt)$.
- И логистическая регрессия – это минимизация

$$\sum_{n=1}^{N} E_{LR}(y_n t_n) + \lambda \|w\|^2,$$

где $E_{LR}(y_n t_n) = \ln\left(1 + e^{-yt}\right)$.

Связь с логистической регрессией

- График hinge error function, вместе с функцией ошибки для логистической регрессии:



SVM с несколькими классами

- Как обобщить SVM на несколько классов? Варианты (без подробностей):
  1. обучить одну против всех и классифицировать $y(x) = \max_k y_k(x)$ (нехорошо, потому что задача становится несбалансированной, и $y_k(x)$ на самом деле несравнимы);
  2. можно сформулировать единую функцию для всех K SVM одновременно, но обучение становится гораздо медленнее;
  3. можно обучить попарно $K(K-1)/2$ классификаторов, а потом считать голоса – кто победит;
  4. DAGSVM: организуем попарные классификаторы в граф и будем идти по графу, для классификации выбирая очередной вопрос;
  5. есть даже методы, основанные на кодах, исправляющих ошибки.

SVM с одним классом

- SVM также можно использовать с *одним* классом.
- Как и зачем?

SVM с одним классом

- SVM также можно использовать с *одним* классом.
- Как и зачем?
- Можно при помощи SVM очертить границу области высокой плотности.
- Тем самым найдём выбросы данных (outliers).
- Задача будет такая: найти наименьшую поверхность (сферу, например), которая содержит все точки, кроме доли ν.

SVM для регрессии

- SVM можно использовать для регрессии, сохраняя свойство разреженности (т.е. то, что SVM зависит только от опорных векторов).
- В обычной линейной регрессии мы минимизировали

$$\frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 + \frac{\lambda}{2}\|w\|^2.$$

- В SVM мы сделаем так: если мы попадаем в ϵ-окрестность предсказания, то ошибки, будем считать, совсем нет.

SVM для регрессии

- ϵ-insensitive error function:

$$E_\epsilon(y(x) - t) = \begin{cases} 0, & |y(x) - t| < \epsilon, \\ |y(x) - t| - \epsilon & \text{иначе.} \end{cases}$$

- И задача теперь выглядит как минимизация

$$C \sum_{n=1}^{N} E_\epsilon \left(y(x_n) - t_n \right) + \frac{\lambda}{2} \|w\|^2.$$

SVM для регрессии

- Чтобы переформулировать, нужны по две slack переменные, для обеих сторон «трубки»:

$$y(x_n) - \epsilon \leq t_n \leq y(x_n) + \epsilon$$

превращается в

$$t_n \leq y(x_n) + \epsilon + \xi_n,$$
$$t_n \geq y(x_n) - \epsilon - \hat{\xi}_n,$$

и мы оптимизируем

$$C \sum_{n=1}^{N} E_\epsilon \left(\xi_n + \hat{\xi}_n \right) + \frac{\lambda}{2} \|w\|^2.$$

SVM для регрессии

- Если же теперь пересчитать дуальную задачу, то получится

$$L(a, \hat{a}) = -\frac{1}{2}\sum_n\sum_m (a_n - \hat{a}_n)(a_m - \hat{a}_m)\, k(x_n, x_m) - \\ - \epsilon\sum_{n=1}^{n}(a_n + \hat{a}_n) + \sum_{n=1}^{N}(a_n - \hat{a}_n)\, t_n,$$

и мы её минимизируем по $a_n, \hat{a}_n$ с условиями

$$0 \leq a_n \leq C,$$
$$0 \leq \hat{a}_n \leq C,$$
$$\sum_{n=1}^{N}(a_n - \hat{a}_n) = 0.$$

SVM для регрессии

- Когда решим эту задачу, сможем предсказывать новые значения как

$$y(x) = \sum_{n=1}^{N} (a_n - \hat{a}_n)\, k(x, x_n) + b,$$

где b можно найти как

$$b = t_n - \epsilon - w^\top \phi(x_n) =$$

$$= t_n - \epsilon - \sum_{m=1}^{N} (a_m - \hat{a}_m)\, k(x_n, x_m).$$

SVM для регрессии

- А условия ККТ превращаются в

$$
\begin{aligned}
a_n \left(\epsilon + \xi_n + y(x_n) - t_n\right) &= 0, \\
\hat{a}_n \left(\epsilon + \hat{\xi}_n - y(x_n) + t_n\right) &= 0, \\
(C - a_n)\xi_n &= 0, \\
(C - \hat{a}_n)\hat{\xi}_n &= 0.
\end{aligned}
$$

- Отсюда очевидно, что либо a_n, либо $\hat{a}_n$ всегда равны 0, и хотя бы один из них не равен, только если точка лежит на или за границей «трубки».
- Опять получили решение, зависящее только от «опорных векторов».

SVM для регрессии

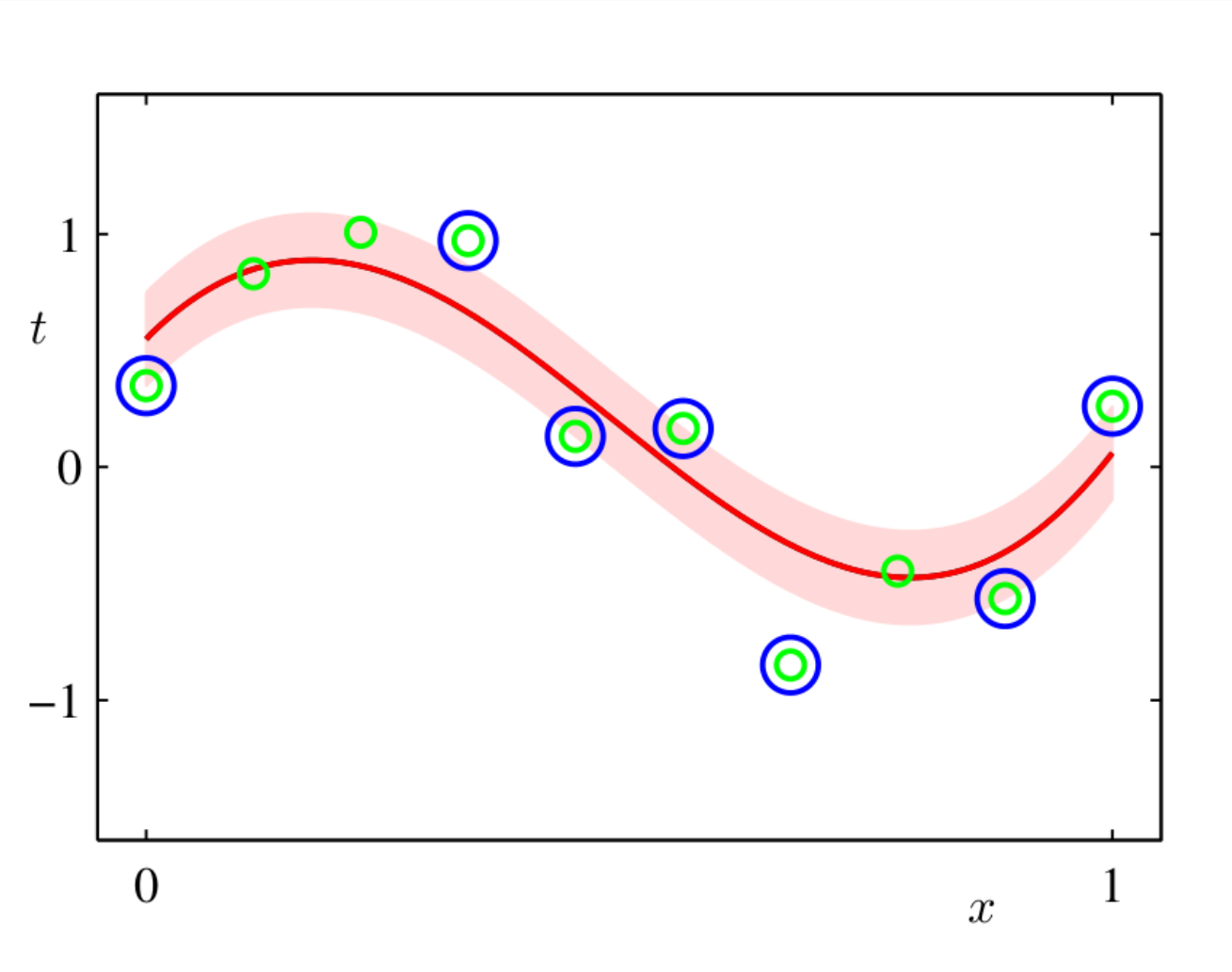
- Но снова можно переформулировать в виде ν-SVM, в котором параметр более интуитивно ясен: вместо ширины трубки ϵ рассмотрим ν – долю точек, лежащих вне трубки; тогда минимизировать надо

$$L(a) = -\frac{1}{2}\sum_n\sum_m (a_n - \hat{a}_n)\,(a_m - \hat{a}_m)\,k\,(x_n, x_m) + \sum_{n=1}^{N}(a_n - \hat{a}_n)\,t_n$$

при условиях

$$\begin{aligned} 0 \le a_n \le \tfrac{C}{N}, \qquad & \textstyle\sum_{n=1}^{N}(a_n - \hat{a}_n) = 0, \\ 0 \le \hat{a}_n \le \tfrac{C}{N}, \qquad & \textstyle\sum_{n=1}^{N}(a_n + \hat{a}_n) \le \nu C. \end{aligned}$$

SVM для регрессии



Практика

- На практике:
  - маленький C – гладкая разделяющая поверхность, мало опорных векторов;
  - большой C – сложная разделяющая поверхность, много опорных векторов.
- Для RBF ядра:
  - маленькое γ – опорные векторы влияют далеко, модель более простая;
  - большое γ – опорные векторы влияют только непосредственно рядом, модель более сложная.

Спасибо!

Спасибо за внимание!