

# МЕТОД ОПОРНЫХ ВЕКТОРОВ

---

Сергей Николенко

СПбГУ — Санкт-Петербург

27 октября 2018 г.

---

*Random facts:*

- 27 октября 1553 г. в Женеве сожгли Мигеля Сервета
- 27 октября 1959 г. «Правда» впервые в мире опубликовала изображение обратной стороны Луны, полученное аппаратом «Луна-3»
- 27 октября 1922 г. в Италии начался марш чернорубашечников на Рим; через два дня Виктор-Эммануил утвердил Бенито Муссолини на пост премьер-министра
- 27 октября 1982 г. Китай объявил, что на свет появился миллиардный китаец
- 27 октября 2017 г. Каталония провозгласила независимость от Испании

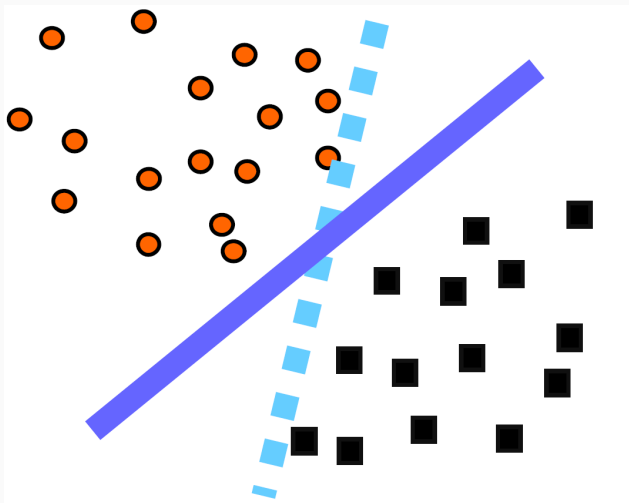
# SVM И ЗАДАЧА ЛИНЕЙНОЙ КЛАССИФИКАЦИИ

---

# 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?

- 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.

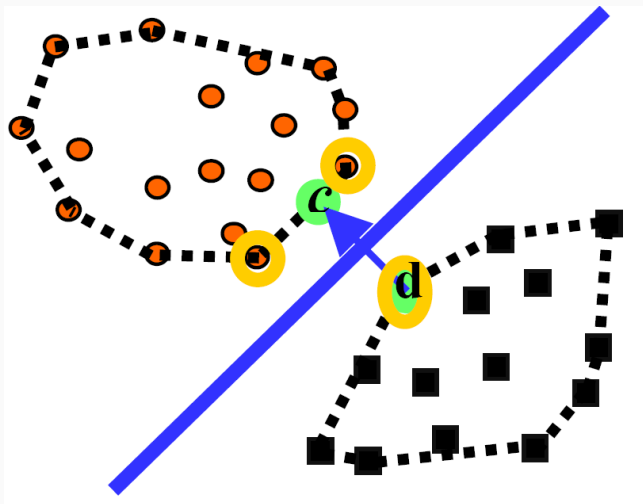


- Один подход: найти две ближайшие точки в выпуклых оболочках данных, а затем провести разделяющую гиперплоскость через середину отрезка.
- Формально это превращается в задачу квадратичной оптимизации:

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

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.



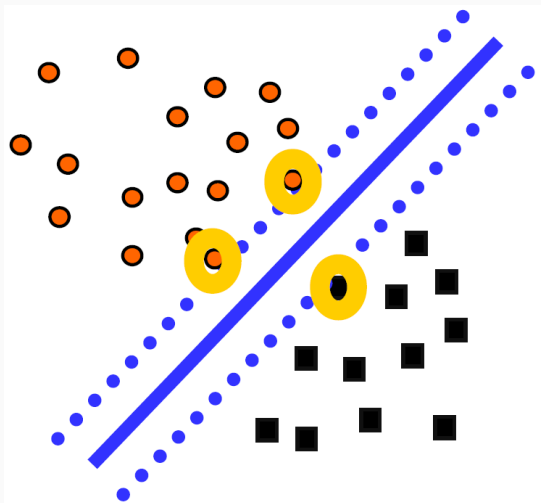
- Другой подход: максимизировать зазор (margin) между двумя параллельными опорными плоскостями, затем провести им параллельную на равных расстояниях от них.
- Гиперплоскость называется *опорной* для множества точек  $X$ , если все точки из  $X$  лежат под одну сторону от этой гиперплоскости.
- Формально: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



- 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

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

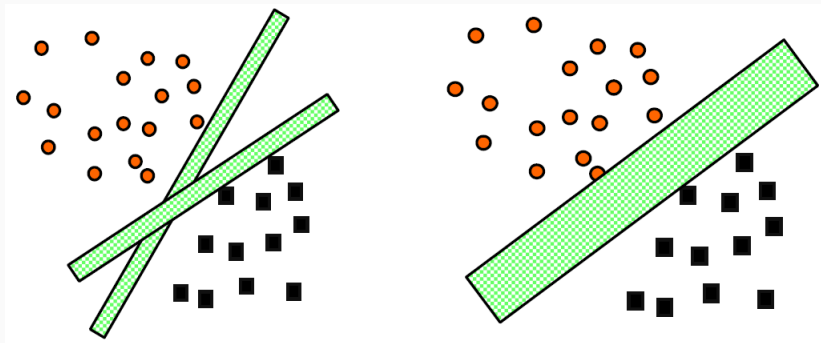
- $\arg \max_{w, w_0} \left\{ \frac{1}{\|w\|} \min_n [t_n(w^\top x_n + w_0)] \right\}$ . Сложно.
- Но если перенормировать  $w$ , гиперплоскость не изменится.
- Давайте перенормируем так, чтобы  $\min_n [t_n(w^\top x_n + w_0)] = 1$ .



- 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 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).



- 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  $\lambda$ , соответствующие равенствам, и  $\mu$ , соответствующие неравенствам.

- Прямая задача оптимизации:

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

- Дуальная задача оптимизации:

$$\min \{\phi(\lambda, \mu)\} \text{ при условии } \mu \geq 0,$$

$$\text{где } \phi(\lambda, \mu) = \inf_{x \in X} \{f(x) + \lambda^\top h(x) + \mu^\top g(x)\}.$$



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

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

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

- 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:

$$\begin{aligned}\phi(\lambda) &= \inf_{x \geq 0} \{c^\top x + \lambda^\top (b - Ax)\} = \\ &= \lambda^\top b + \inf_{x \geq 0} \{(c^\top - \lambda^\top A)x\} = \\ &= \begin{cases} \lambda^\top b, & \text{если } c^\top - \lambda^\top A \geq 0, \\ -\infty & \text{в противном случае.} \end{cases}\end{aligned}$$

- Для линейного программирования прямая задача:

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

- Дуальная задача:

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

- Для квадратичного программирования прямая задача:

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

где  $Q$  – положительно полуопределённая матрица (т.е.  $x^\top Q x \geq 0$  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  $D = -A Q^{-1} A^\top$  (отрицательно определённая матрица),  
 $d = -b - A Q^{-1} c$ .

- В случае SVM надо ввести множители Лагранжа:

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

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

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

$$0 = \sum_n \alpha_n t_n.$$

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

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

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

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

- 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  $x_n$ ...

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

$$\alpha_n \geq 0,$$

$$t_n y(x_n) - 1 \geq 0,$$

$$\alpha_n (t_n y(x_n) - 1) = 0.$$

- 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).

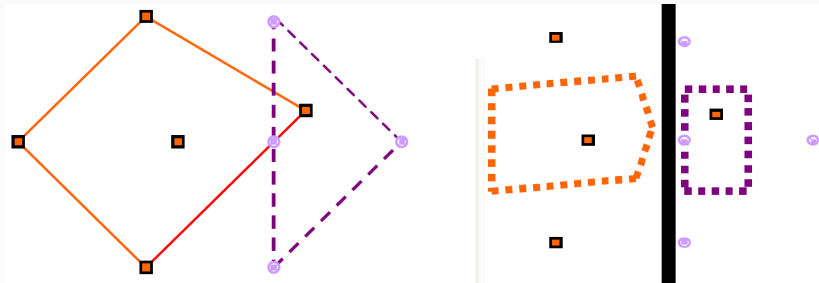


- Все эти методы работают, когда данные действительно линейно делимы.
- А что делать, когда их всё-таки немножко не получается разделить?
- Первый вопрос: что делать для первого метода, метода выпуклых оболочек?

- 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.



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

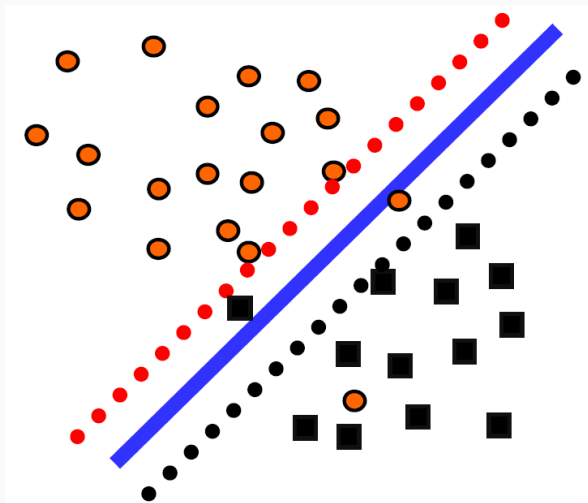
## Для метода опорных векторов

- Естественно, для метода опорных векторов тоже надо что-то изменить. Что?
- Мы просто добавим в оптимизируемую 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...



- ...а вот дуальная:

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

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

- 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.



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

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

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

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

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

- ...поэтому заменим на оценку сверху:

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

- А потом ещё добавим регуляризатор для стабильности:

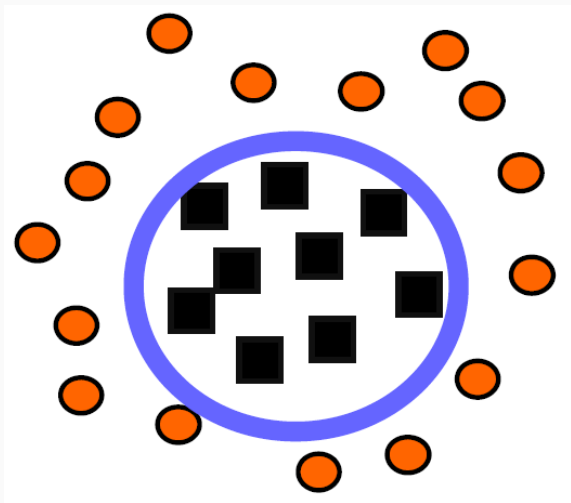
$$\sum_n [M_i < 0] \leq \sum_n (1 - M_i) + \frac{1}{2C} \|w\|^2 \rightarrow \min_w .$$

- И это снова получилась задача SVM!

# SVM и РАЗДЕЛЕНИЕ НЕЛИНЕЙНЫМИ ФУНКЦИЯМИ

---

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



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

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

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

- 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.



- Тривиальная схема алгоритма классификации такова:
  - входной вектор  $\vec{x}$  трансформируется 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.

- 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. :)

- Напомним, что наша задача поставлена следующим образом:

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

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

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

- 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:

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

- 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  $\theta_i$  — собственные векторы интегрального оператора с ядром  $k$ , т.е.

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

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

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

для всех  $g$  таких, что  $\int g^2(\vec{u}) d\vec{u} < \infty$ .

- Вот, собственно и всё. Теперь мы можем вместо подсчёта  $\theta(\vec{u}) \cdot \theta(\vec{v})$  в задаче квадратичного программирования просто использовать подходящее ядро  $k(\vec{u}, \vec{v})$ .

- 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.



- Рассмотрим ядро

$$k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v})^2.$$

- Какое пространство ему соответствует?

- После выкладок получается:

$$\begin{aligned} k(\vec{u}, \vec{v}) &= (\vec{u} \cdot \vec{v})^2 = \\ &= (u_1^2, u_2^2, \sqrt{2}u_1u_2) \cdot (v_1^2, v_2^2, \sqrt{2}v_1v_2). \end{aligned}$$

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

- 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?

- Поверхность, описываемая 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.

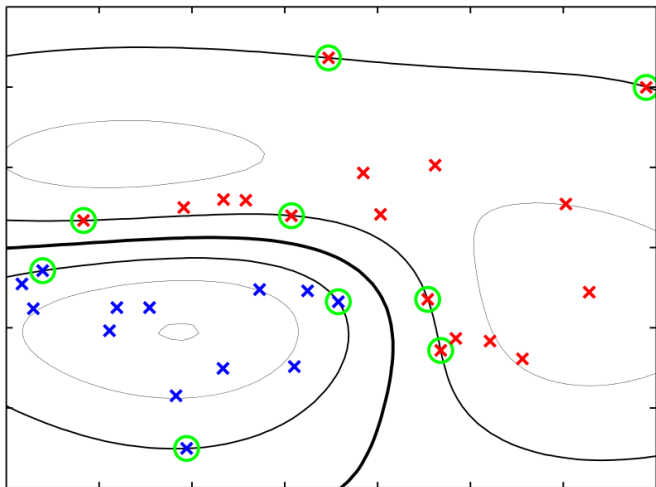
- Нормальное распределение (radial basis function):

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

- Двухуровневая нейронная сеть:

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

где  $o$  — СИГМОИД.



- 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

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

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

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

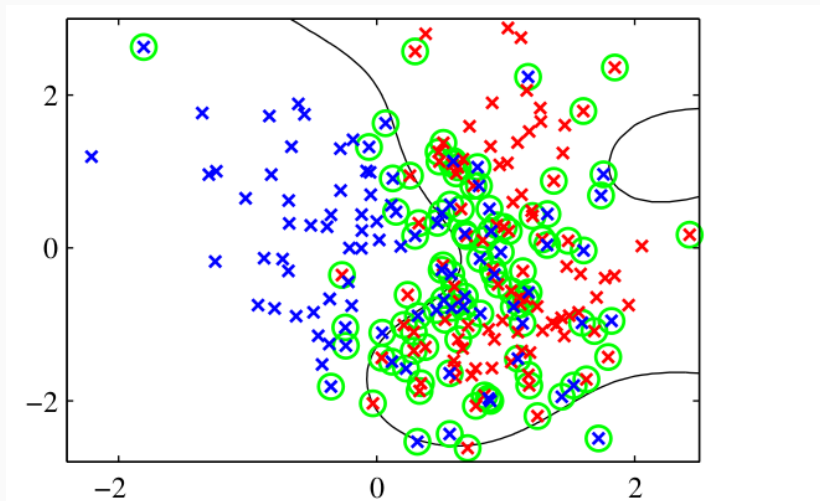
с ограничениями

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

- Параметр  $\nu$  можно интерпретировать как верхнюю границу на долю ошибок.



## SVM для КЛАССИФИКАЦИИ



- 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

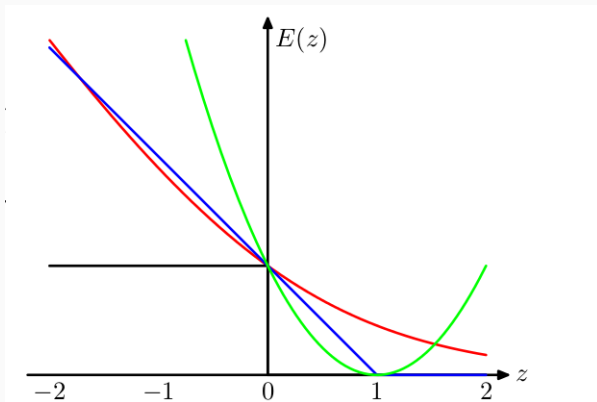
$$\sum_{n=1}^N E_{SV}(y_n t_n) + \lambda \|w\|^2.$$

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

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



- 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 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  $\nu$ .

- 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  $\epsilon$ -окрестность предсказания, то ошибки, будем считать, совсем нет.



- $\epsilon$ -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

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

- 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

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

- 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

$$0 \leq a_n \leq C,$$

$$0 \leq \hat{a}_n \leq C,$$

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

- 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

$$\begin{aligned} b &= t_n - \epsilon - w^\top \phi(x_n) = \\ &= t_n - \epsilon - \sum_{m=1}^N (a_m - \hat{a}_m) k(x_n, x_m). \end{aligned}$$

- А условия ККТ превращаются в

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

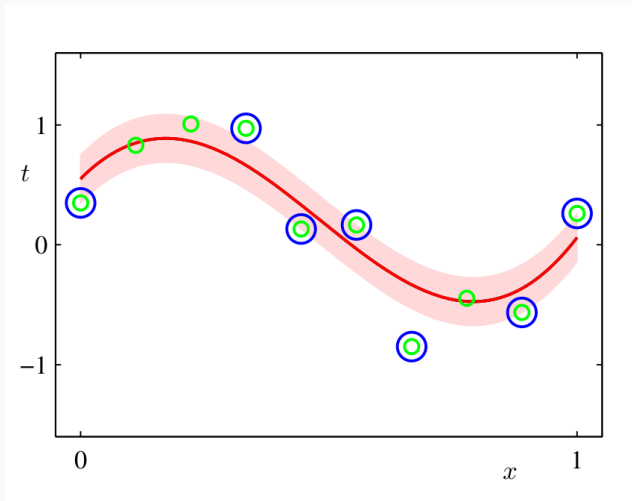
- Отсюда очевидно, что либо  $a_n$ , либо  $\hat{a}_n$  всегда равны 0, и хотя бы один из них не равен, только если точка лежит на или за границей «трубки».
- Опять получили решение, зависящее только от «опорных векторов».

- Но снова можно переформулировать в виде  $\nu$ -SVM, в котором параметр более интуитивно ясен: вместо ширины трубки  $\epsilon$  рассмотрим  $\nu$  – 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

$$\begin{aligned} 0 \leq a_n \leq \frac{C}{N}, & \quad \sum_{n=1}^N (a_n - \hat{a}_n) = 0, \\ 0 \leq \hat{a}_n \leq \frac{C}{N}, & \quad \sum_{n=1}^N (a_n + \hat{a}_n) \leq \nu C. \end{aligned}$$



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



Спасибо за внимание!