Метод опорных векторов

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург

21 марта 2020 г.

Random facts:

- 21 марта — день весеннего равноденствия
- 21 марта в ЮНЕСКО — Всемирный день поэзии, а в ООН — Всемирный день людей с синдромом Дауна
- 21 марта 630 г. Ираклий I отбил у персов крест Иисуса Христа и вернул его в Иерусалим
- 21 марта 1924 г. Вольфганг Паули опубликовал статью со своим знаменитым принципом
- 21 марта 1926 г. в Теннесси принят закон, запрещающий преподавание эволюционной теории Дарвина во всех учебных заведениях
- 21 марта 1981 г. в городе Мобил, Алабама, зарегистрирован последний случай линчевания, жертвой которого стал Майкл Дональд
- 21 марта 2014 г. Совет Федерации ратифицировал договор о принятии Крыма в состав России, а Владимир Путин подписал соответствующие указы

SVM и задача линейной классификации

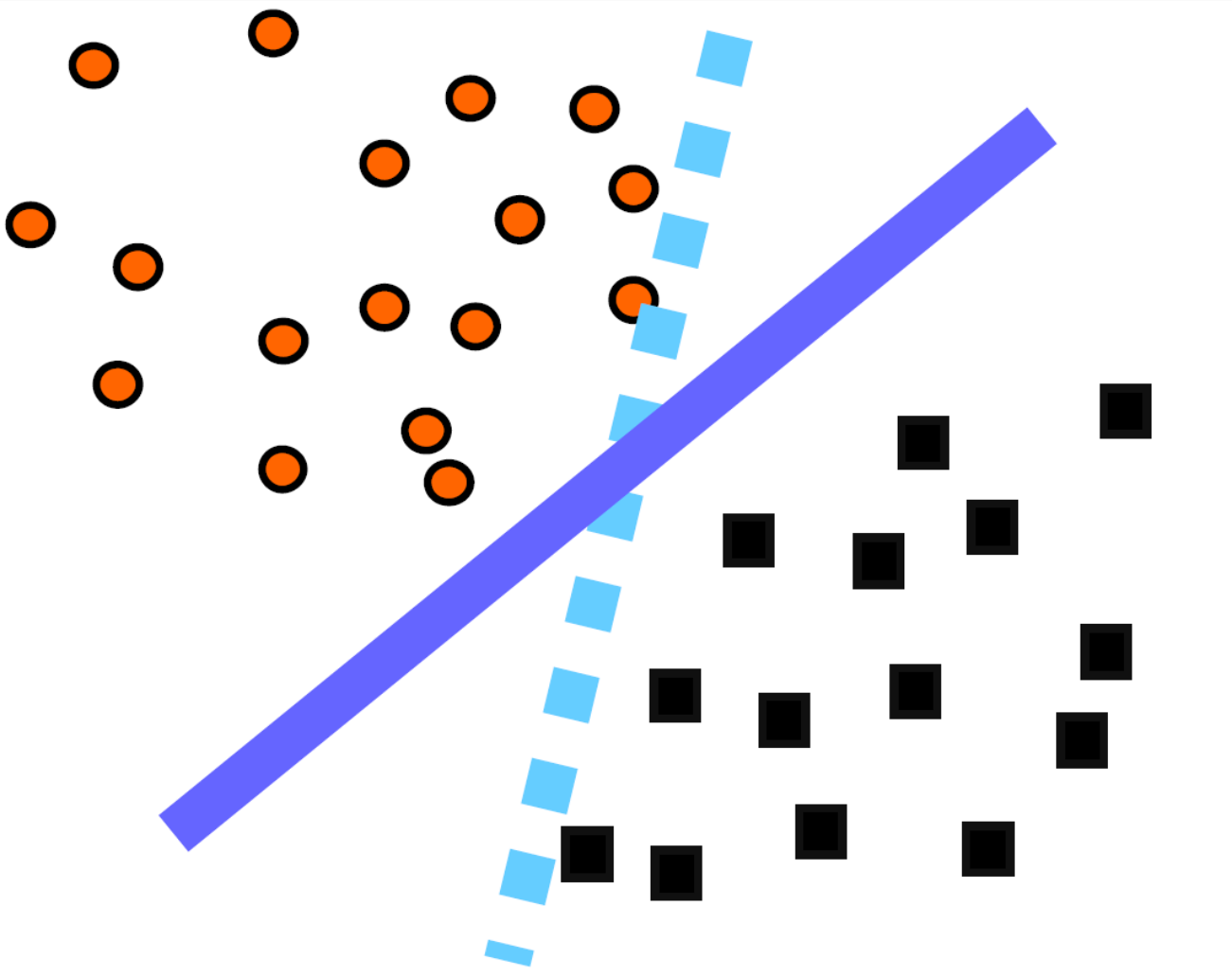
Постановка задачи

- 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



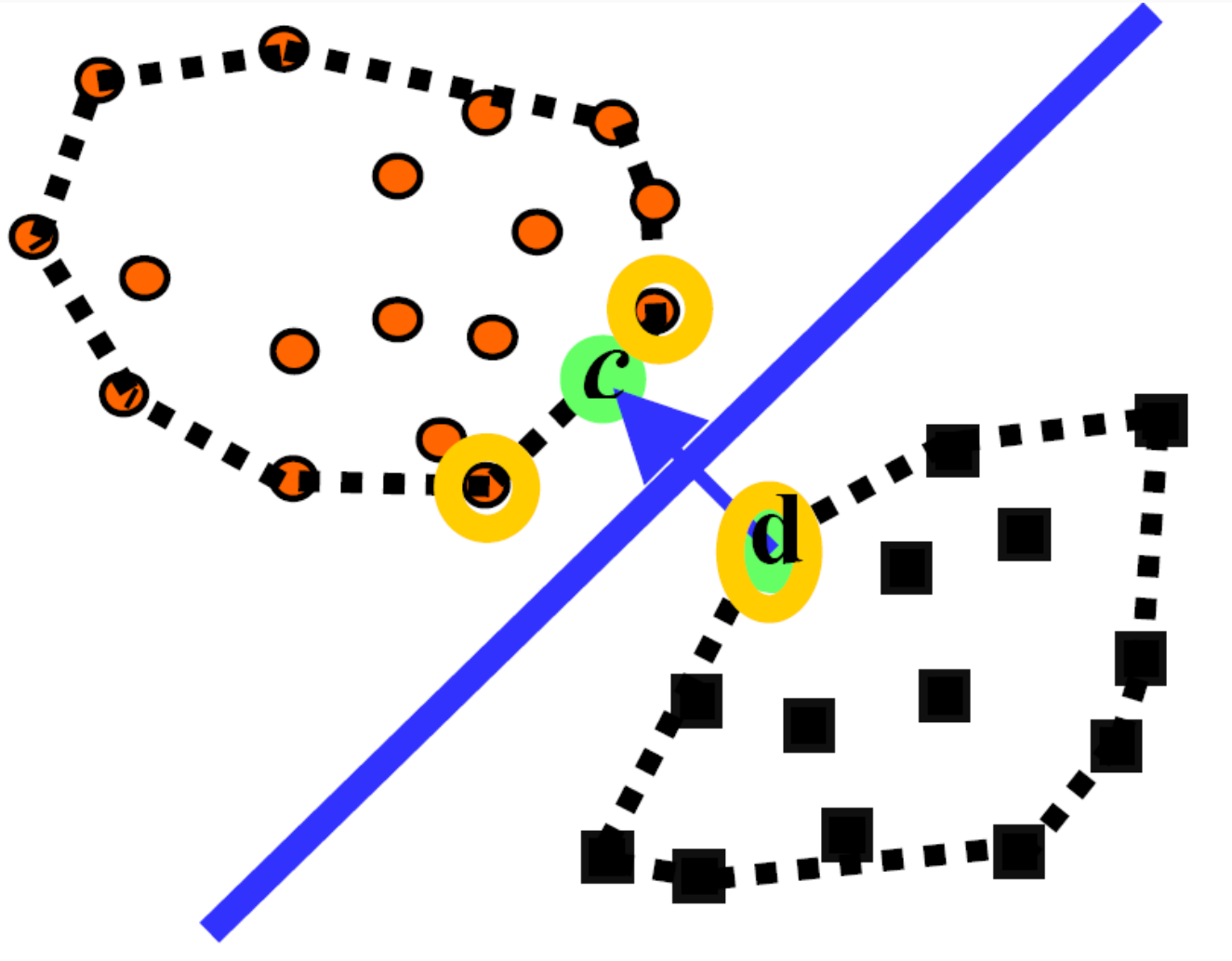
Выпуклые оболочки

- 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 $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0 = 0$ равно $\frac{|y(\mathbf{x})|}{\|\mathbf{w}\|}$.

Максимизация зазора

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

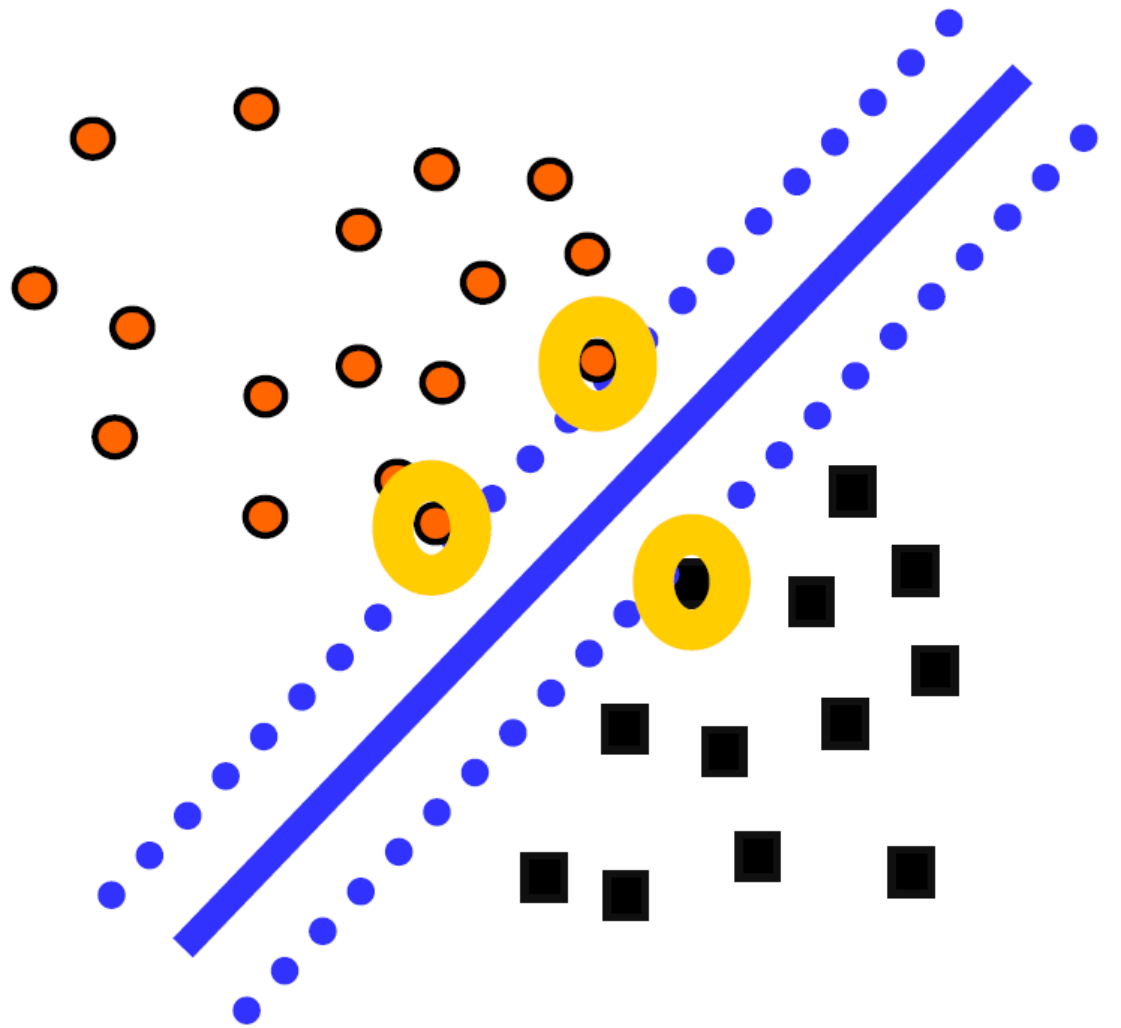
$$\arg\max_{\mathbf{w}, w_0} \min_n \frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} =$$

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

Максимизация зазора

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

Пример



Максимизация зазора

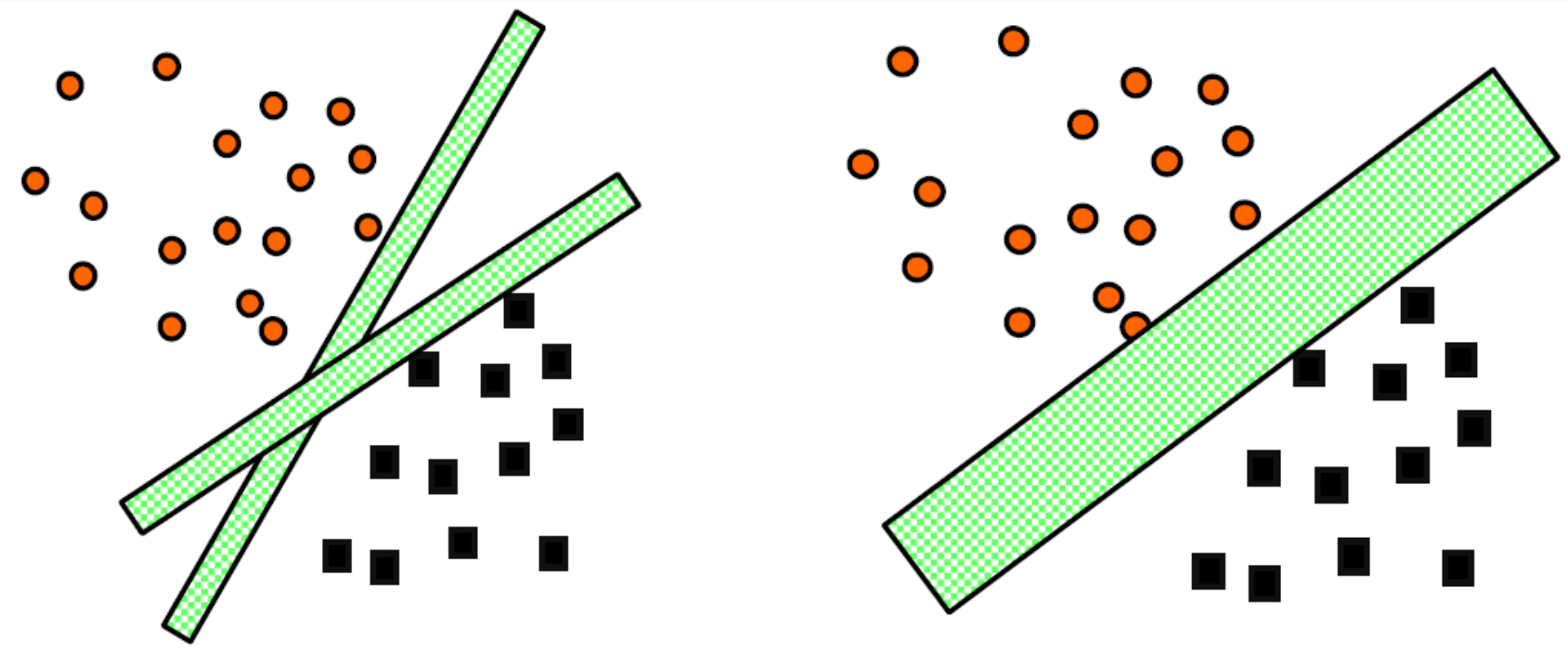
- Получается тоже задача квадратичного программирования:

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

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:

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

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:

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

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

Дуальная задача к SVM

- В случае SVM надо ввести множители Лагранжа:

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

- Берём производные по $\mathbf{w}$ и w_0, приравниваем нулю, получаем

$$\mathbf{w} = \sum_n \alpha_n t_n \mathbf{x}_n,$$
$$0 = \sum_n \alpha_n t_n.$$

Дуальная задача к SVM

- Подставляя в $L(\mathbf{w}, w_0, \boldsymbol{\alpha})$, получим

$$L(\boldsymbol{\alpha}) = \sum_n \alpha_n - \frac{1}{2} \sum_n \sum_m \alpha_n \alpha_m t_n t_m \left(\mathbf{x}_n^\top \mathbf{x}_m\right)$$

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

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

Предсказание и ККТ

- А для предсказания потом надо посмотреть на знак $y(\mathbf{x})$:

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

- Получилось, что предсказания зависят от всех точек $\mathbf{x}_n$...

Предсказание и ККТ

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

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

- Т.е. реально предсказание зависит от небольшого числа *опорных* векторов, для которых $t_n y(\mathbf{x}_n) = 1$ (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?

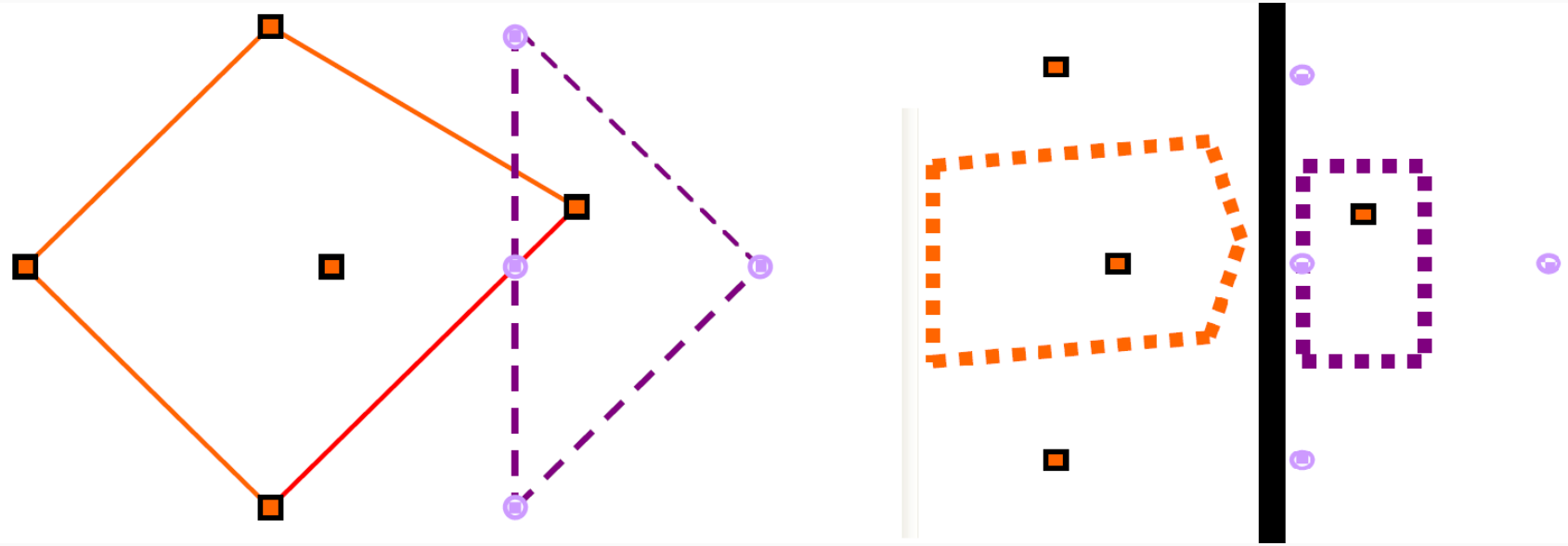
Редуцированные выпуклые оболочки

- 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

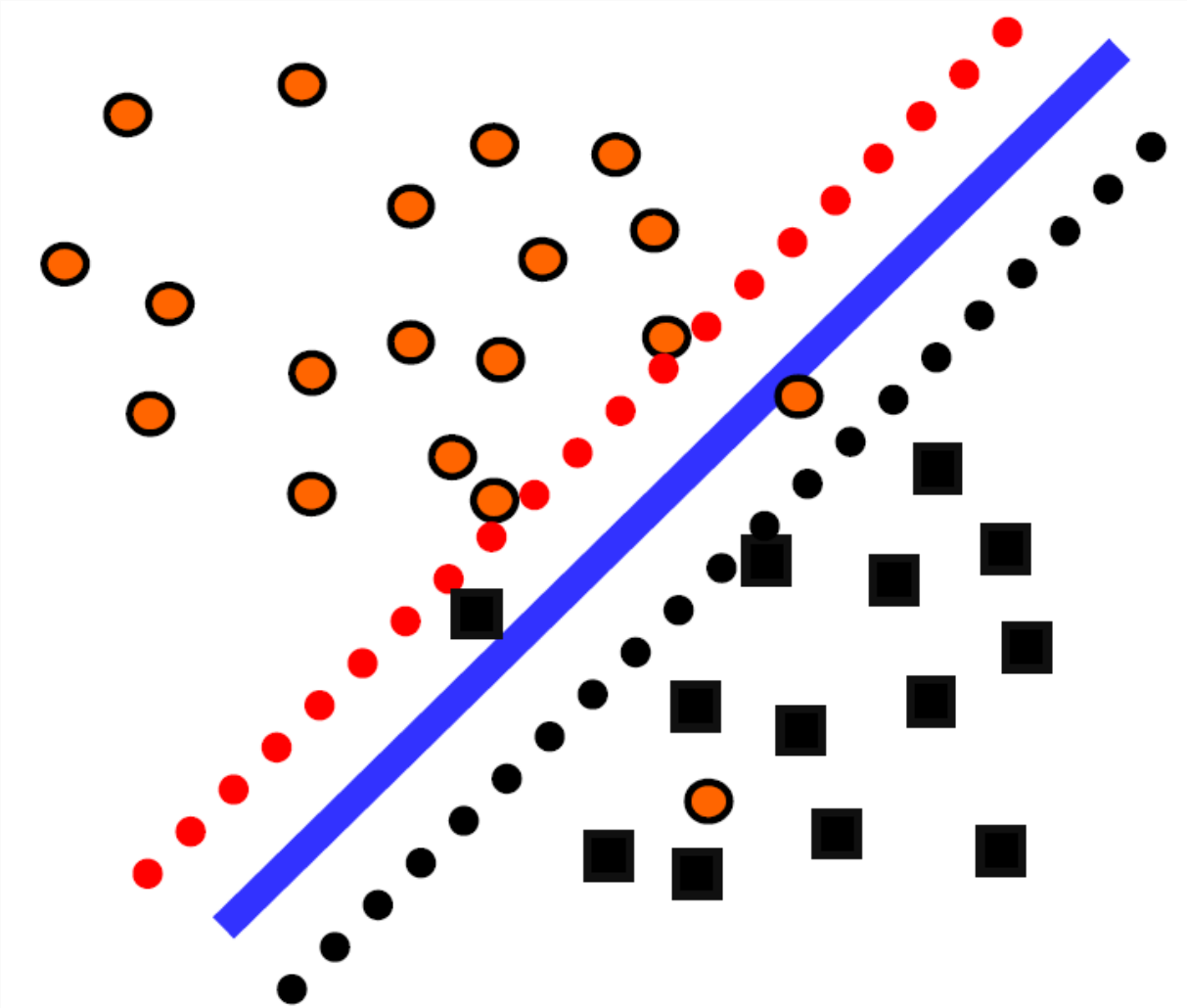
- Естественно, для метода опорных векторов тоже надо что-то изменить. Что?
- Мы просто добавим в оптимизирующуюся функцию неотрицательную ошибку (slack):

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

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

- Это прямая задача...

Пример



Дуальная переформулировка

- ...а вот дуальная:

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

- 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:

$$\sum_n [y_i \neq t_i] \to \min_{\mathbf{w}}.$$

- И если функция линейная с параметрами $\mathbf{w}$, w_0, то это эквивалентно

$$\sum_n \left[t_i \left(\mathbf{x}_n^\top \mathbf{w} - w_0\right) < 0\right] \to \min_{\mathbf{w}}.$$

- Величину $M_i = \mathbf{x}_n^\top \mathbf{w} - w_0$ назовём *отступом* (margin).
- Оптимизировать напрямую сложно...

SVM и эмпирический риск

- ...поэтому заменим на оценку сверху:

$$\sum_n [M_i < 0] \leq \sum_n (1 - M_i) \to \min_{\mathbf{w}}.$$

- А потом ещё добавим регуляризатор для стабильности:

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

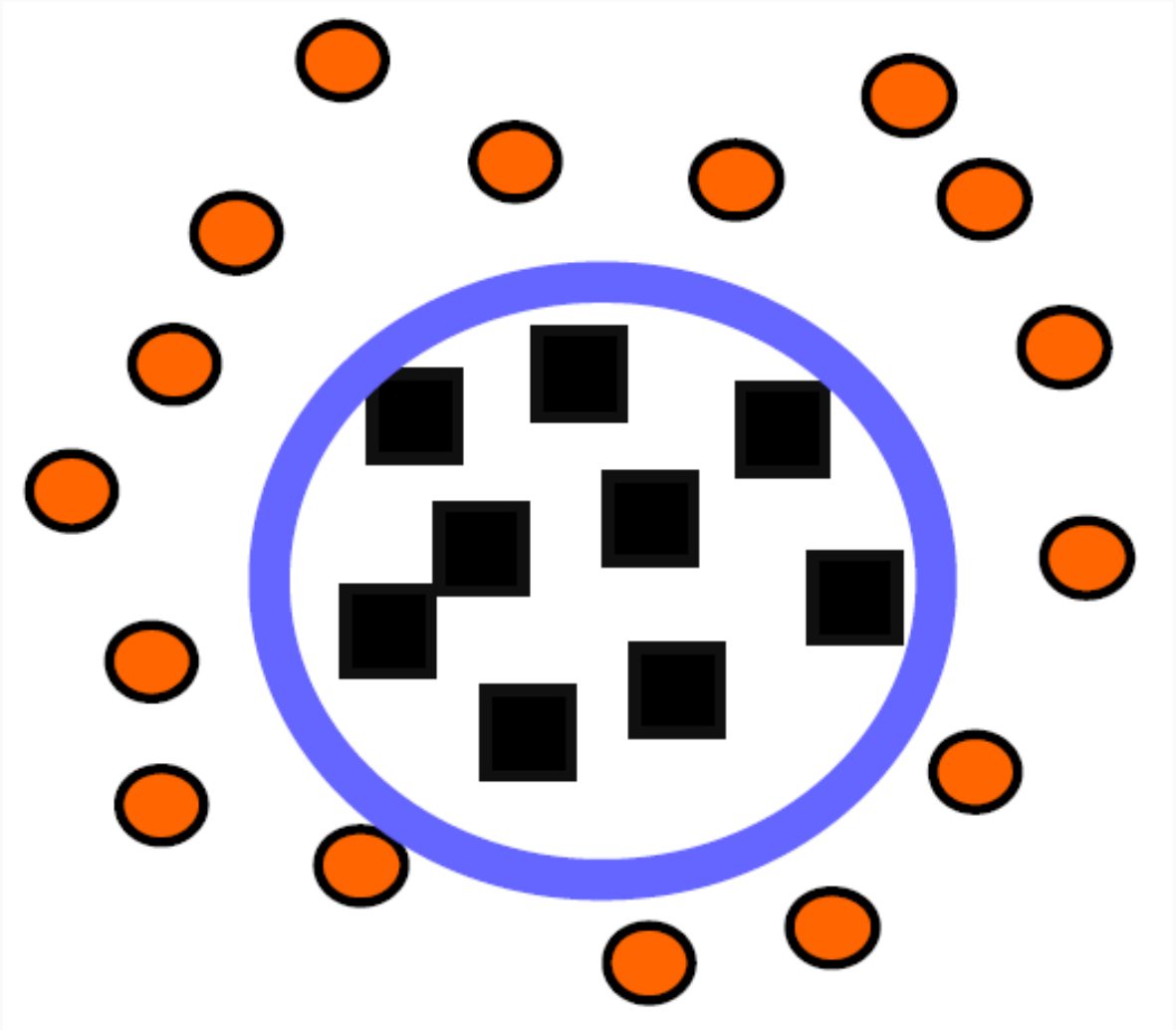
- И это снова получилась задача SVM!

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

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

- 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 **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:

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

Постановка задачи

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

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

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:

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

Теория Гильберта–Шмидта–Мерсера

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

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

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

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

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*)

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

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

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

Теория Гильберта–Шмидта–Мерсера

- Итого задача наша выглядит так:

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

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

Примеры ядер

- После выкладок получается:

$$k(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^2 = \left(u_1^2, u_2^2, \sqrt{2}u_1u_2\right) \cdot \left(v_1^2, v_2^2, \sqrt{2}v_1v_2\right).$$

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

Примеры ядер

- Естественное обобщение: ядро $k(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$ 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:

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

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

Примеры ядер

- Нормальное распределение (radial basis function):

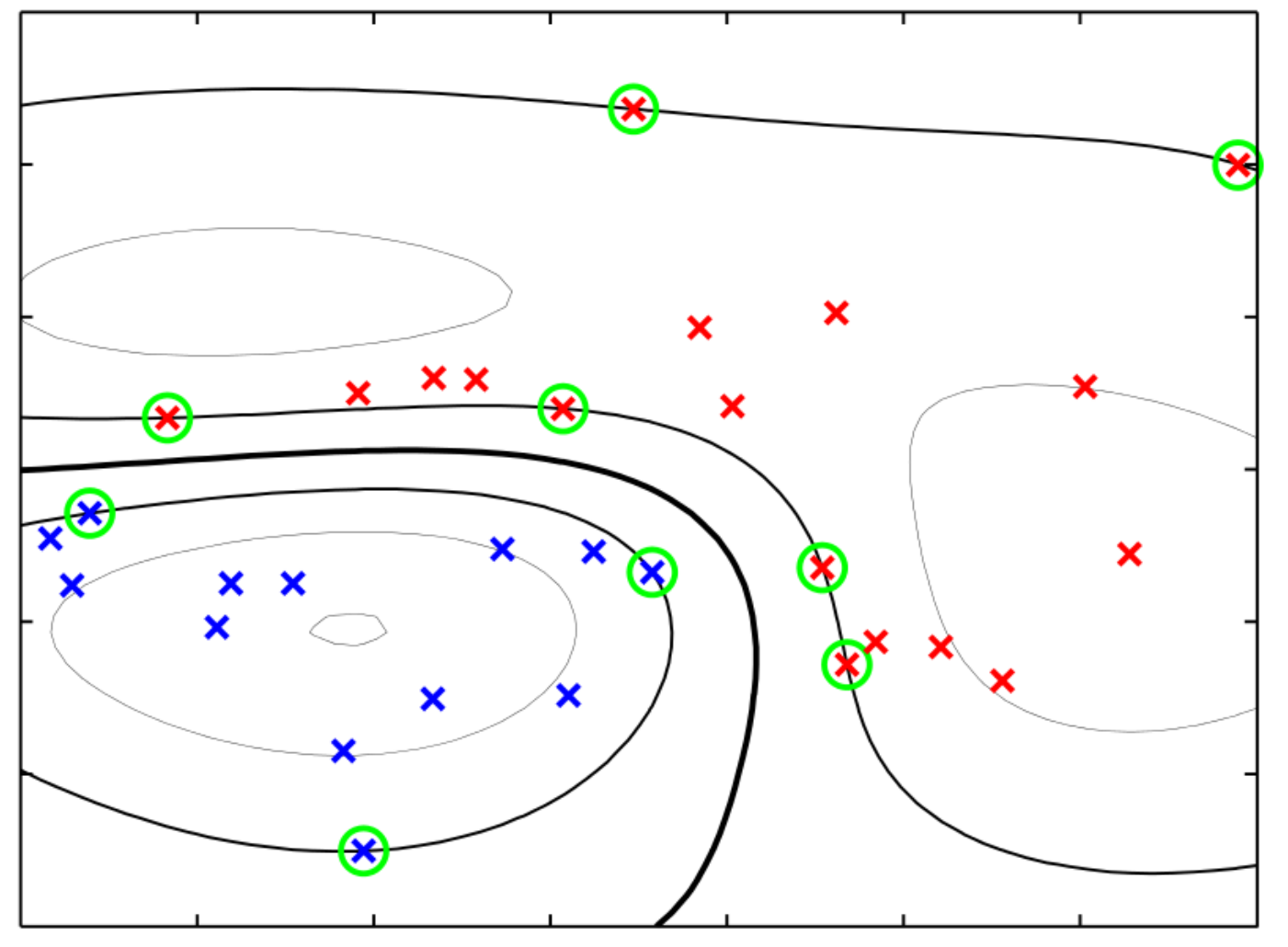
$$k(\mathbf{u}, \mathbf{v}) = e^{-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma}}.$$

- Двухуровневая нейронная сеть:

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

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

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

ν-SVM

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

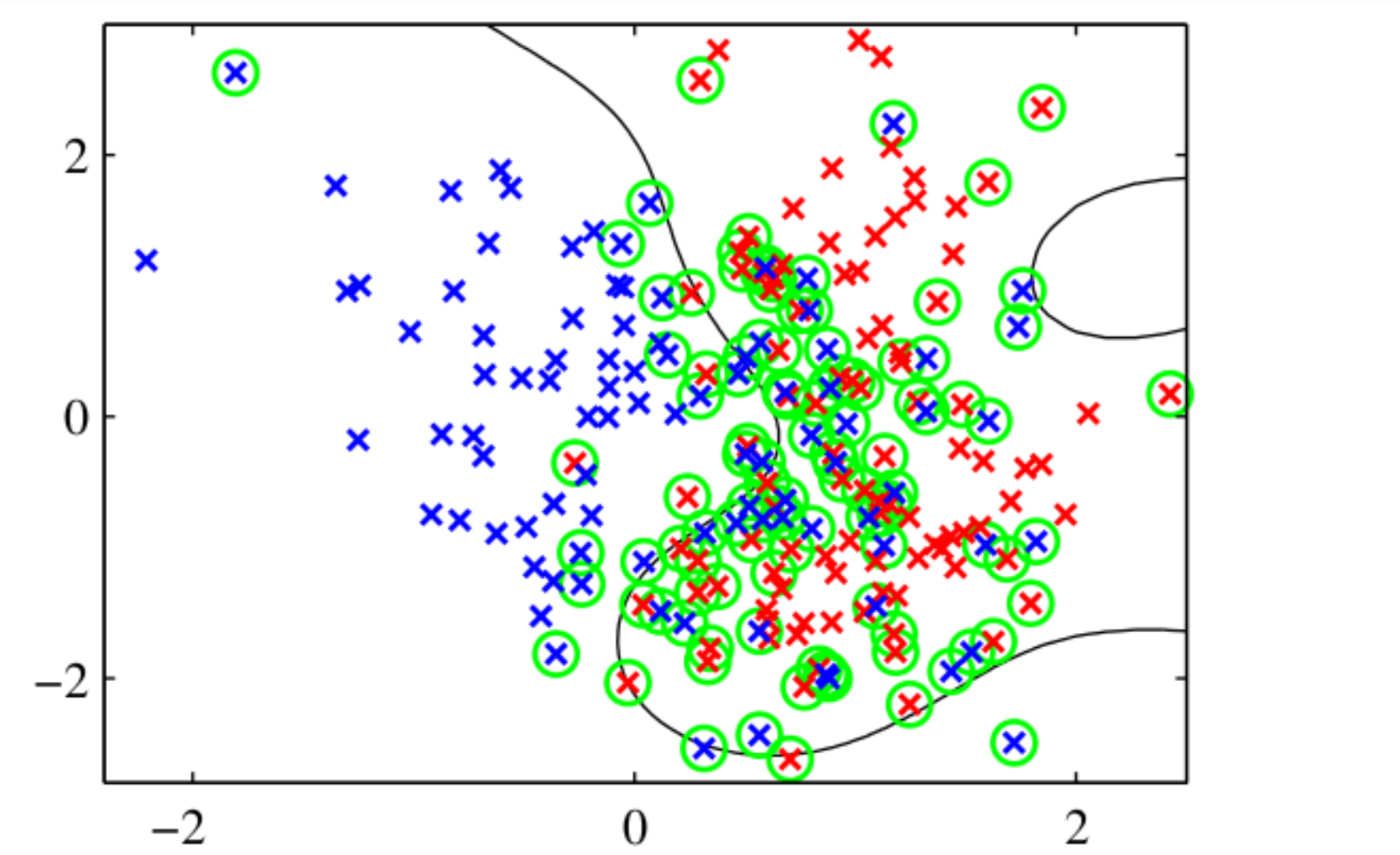
$$L(\mathbf{a}) = -\frac{1}{2}\sum_n\sum_m a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

  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

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

- 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

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

Связь с логистической регрессией

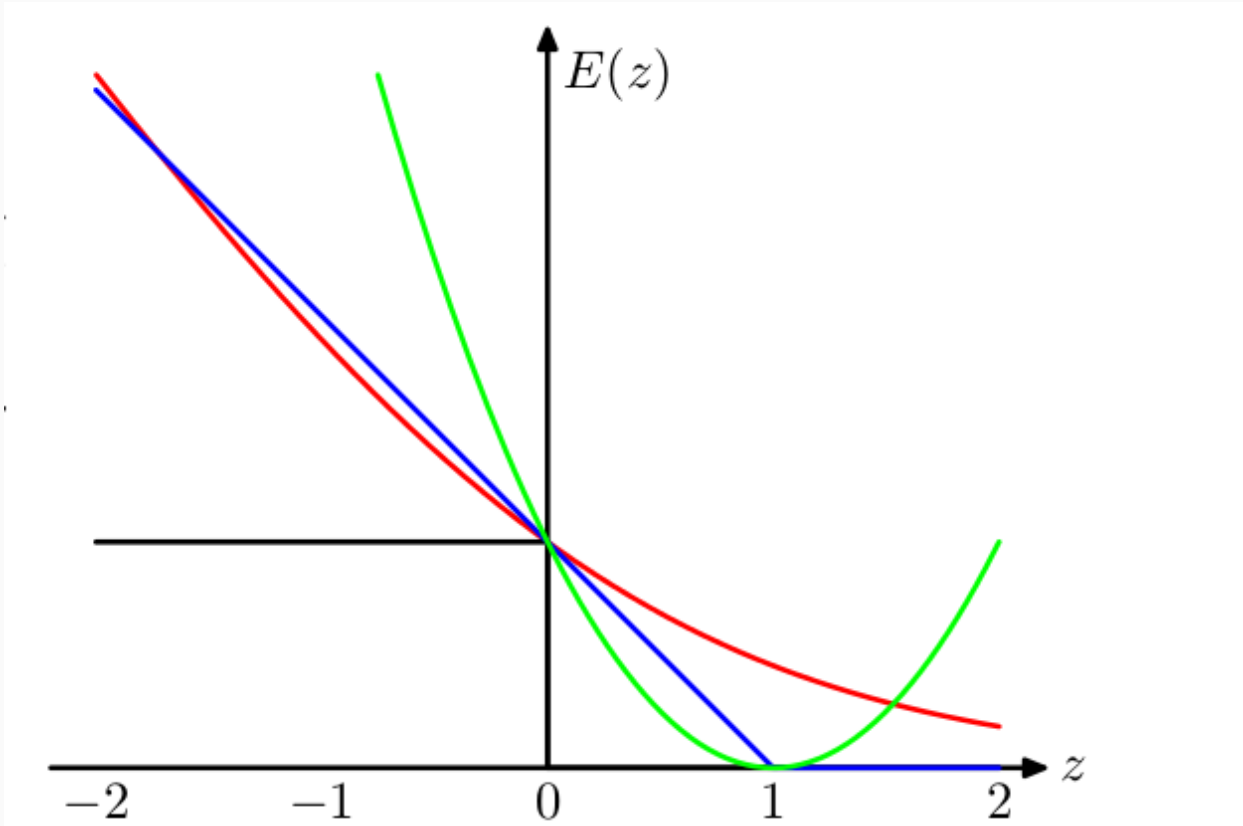
- Вспомним логистическую регрессию и переформулируем её для целевой переменной $t \in \{-1, 1\}$: $p(t = 1 \mid y) = \sigma(y)$, значит, $p(t = -1 \mid y) = 1 - \sigma(y) = \sigma(-y)$, и $p(t \mid y) = \sigma(yt)$.
- И логистическая регрессия – это минимизация

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

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 $y(\mathbf{x}) = \max_k y_k(\mathbf{x})$ (нехорошо, потому что задача становится несбалансированной, и $y_k(\mathbf{x})$ 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

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

- 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:

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

- И задача теперь выглядит как минимизация

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

SVM для регрессии

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

$$y(\mathbf{x}_n) - \epsilon \leq t_n \leq y(\mathbf{x}_n) + \epsilon$$

превращается в

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

и мы оптимизируем

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

SVM для регрессии

- Если же теперь пересчитать дуальную задачу, то получится

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

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

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

где b можно найти как

$$b = t_n - \epsilon - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) =$$

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

SVM для регрессии

- А условия ККТ превращаются в

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

- 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

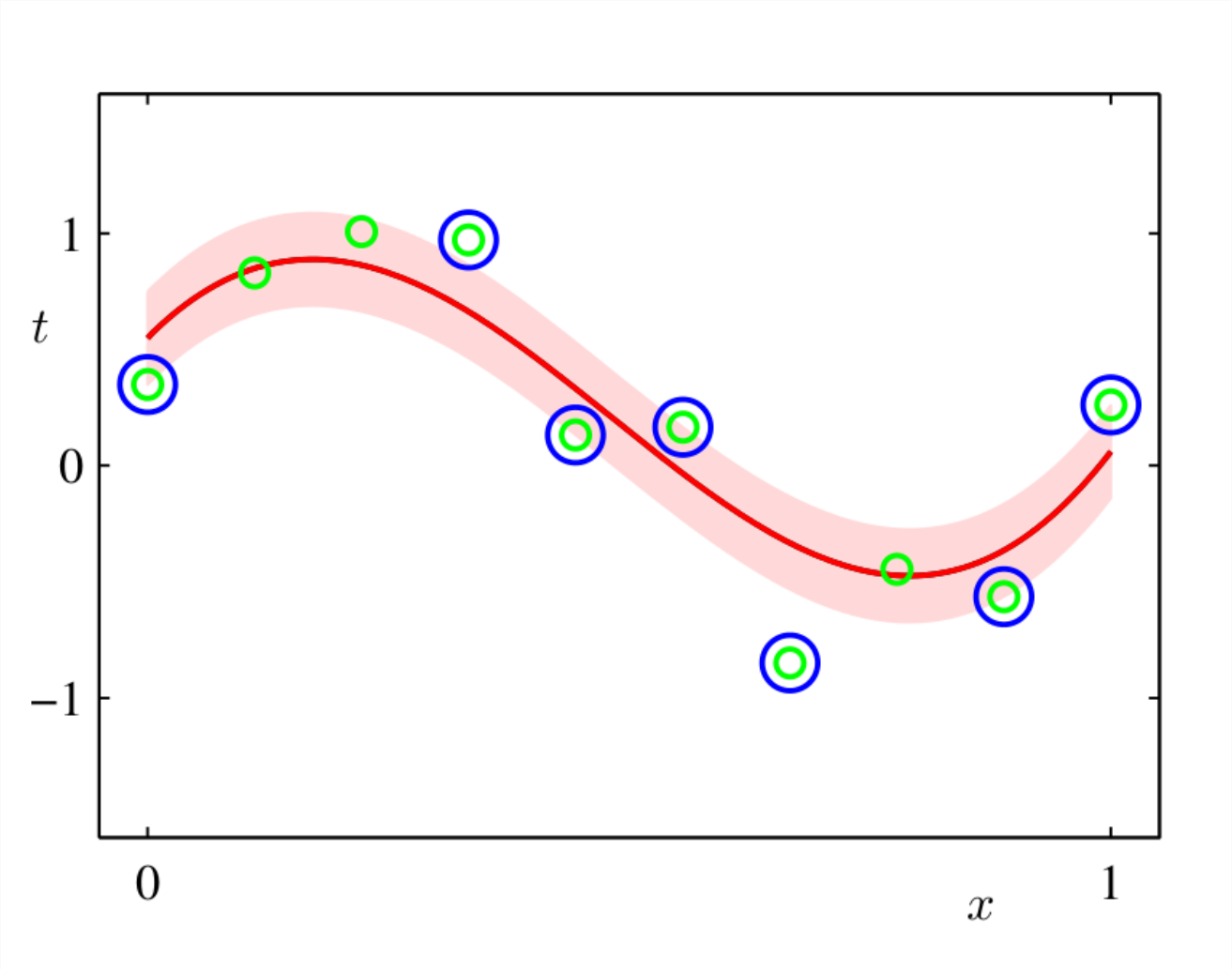
- Но снова можно переформулировать в виде ν-SVM, в котором параметр более интуитивно ясен: вместо ширины трубки ϵ рассмотрим ν – долю точек, лежащих вне трубки; тогда минимизировать надо

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

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!