SVM и kernel methods

Сергей Николенко

Академический Университет, весенний семестр 2011

Outline

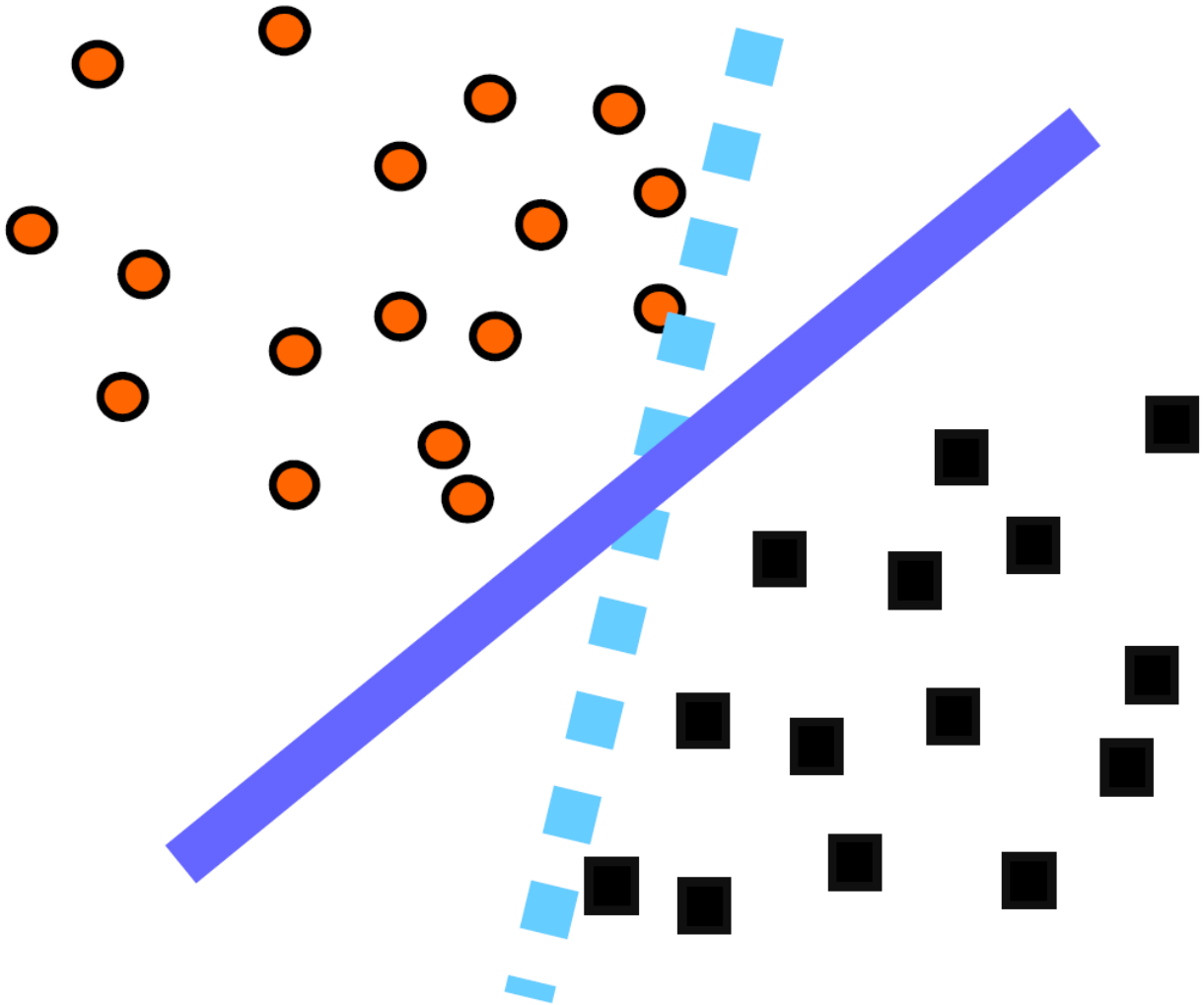
Постановка задачи

- 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.
- В частности, именно так мы обучали линейный перцептрон.
- 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

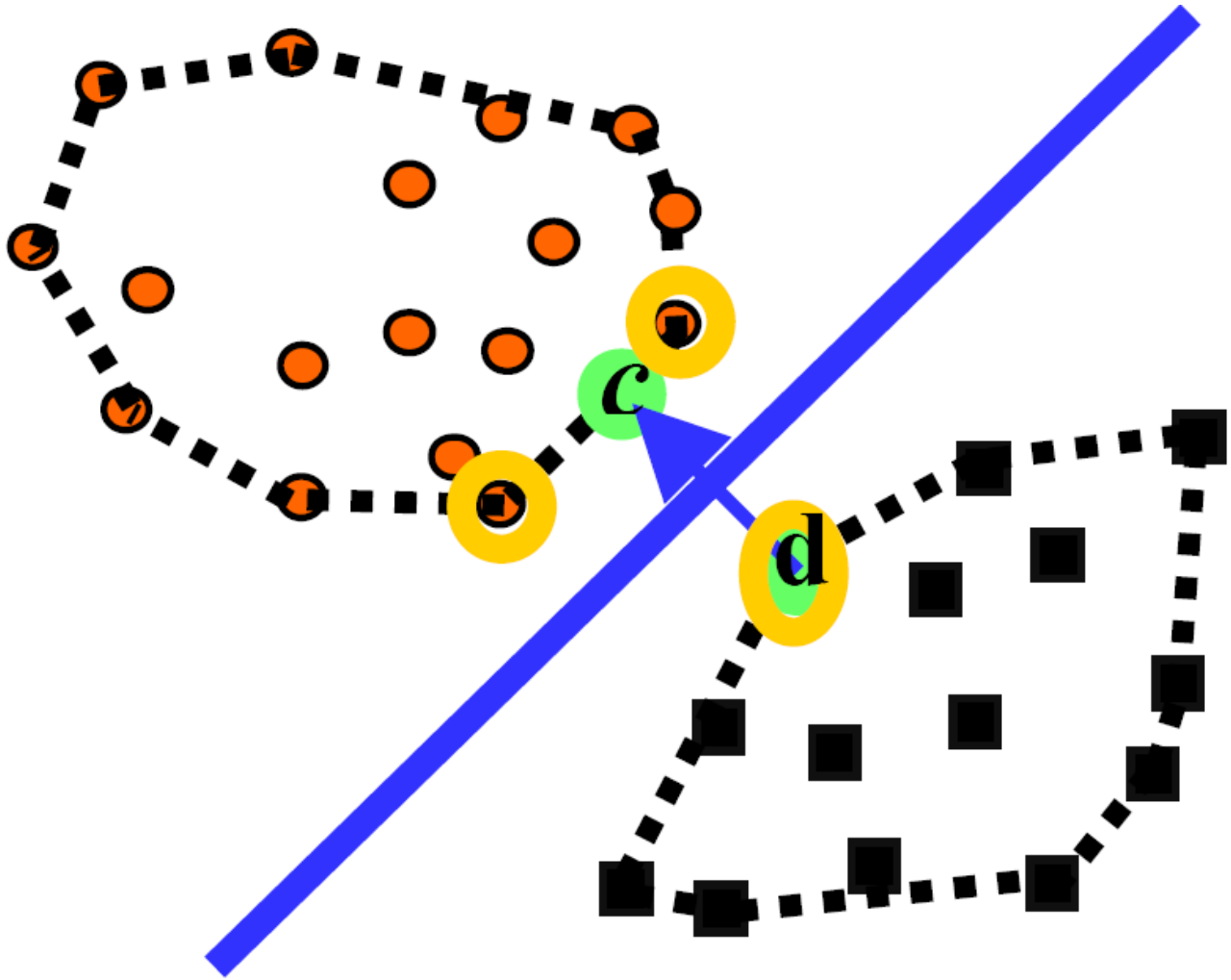
- Один подход: найти две ближайшие точки в выпуклых оболочках данных, а затем провести разделяющую гиперплоскость через середину отрезка.
- Формально это превращается в задачу квадратичной оптимизации:

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

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

- Эту задачу можно решать общими оптимизационными алгоритмами.

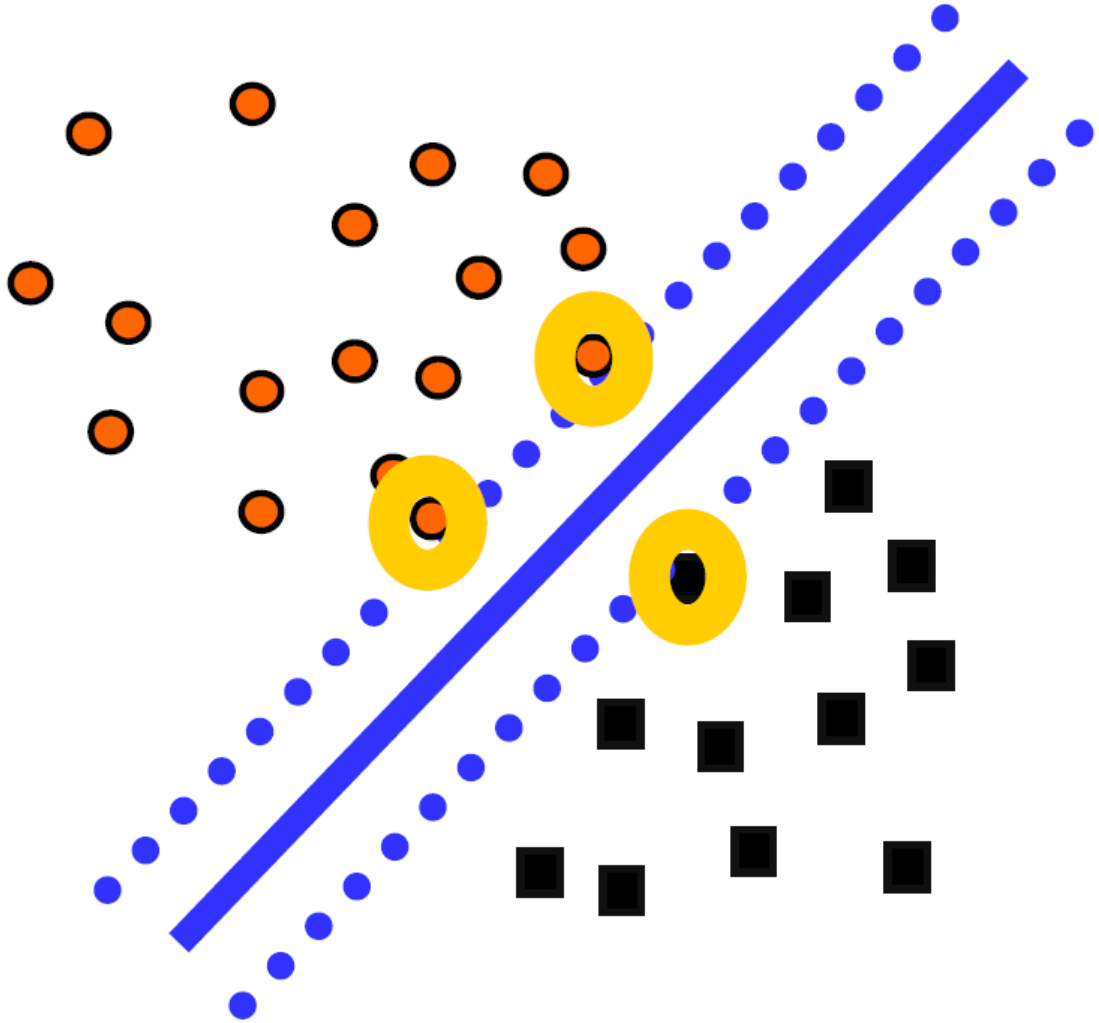
Пример



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: гиперплоскость $(\mathbf{w}, b)$ опорная, если $\mathbf{w} \cdot \mathbf{x}_i - b \geq 1$ для некоторых $\mathbf{w}$ и b и для всех $\mathbf{x}_i \in X$.
- Для другой гиперплоскости, с метками -1, будем требовать $\mathbf{w} \cdot \mathbf{x}_i - b \leq -1$.

Пример



Максимизация зазора

- Разница (зазор, margin) между этими гиперплоскостями равна $2/\|\mathbf{w}\|$ (проверьте!).
- Поэтому получается тоже задача квадратичного программирования:

$$\min_{\mathbf{w},b} \left\{ \|\mathbf{w}\|^2 \right\}$$

при условии $\mathbf{w} \cdot \mathbf{x}_i \geq b+1, y_i = 1; \quad \mathbf{w} \cdot \mathbf{x}_i \leq b-1, y_i = -1,$

или просто $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1.$

Сравнение двух методов

- В чём разница между двумя методами? Можете ли вы придумать пример, на котором эти два метода дают разные результаты?

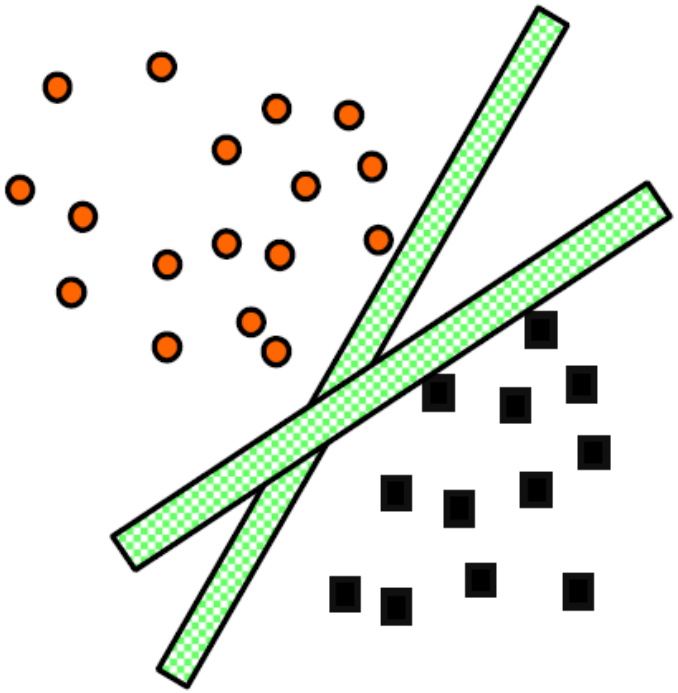
Сравнение двух методов

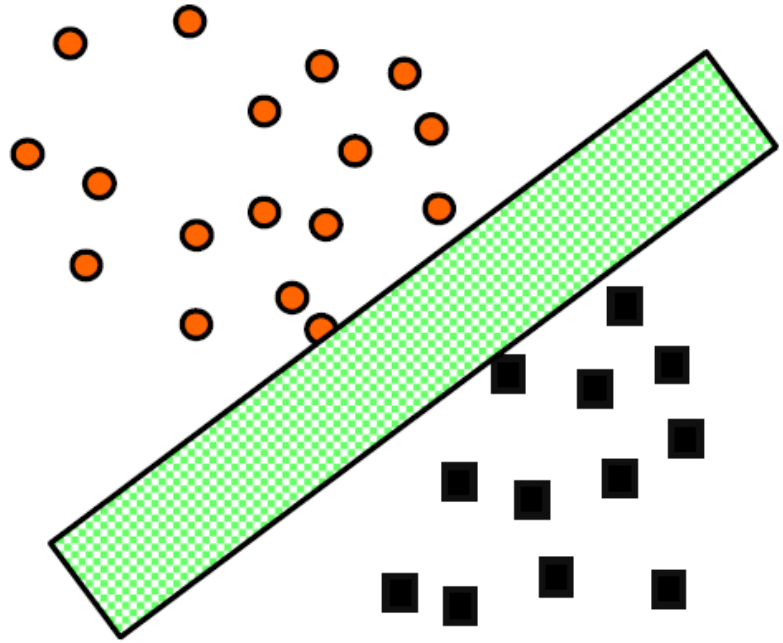
- В чём разница между двумя методами? Можете ли вы придумать пример, на котором эти два метода дают разные результаты?
- Это совершенно одно и то же. На самом деле эти задачи *дуальны* друг к другу (см. ниже).
- Решать можно как одну, так и другую, и в этом случае нет причины предпочитать метод опорных векторов (плоскостей). Но он может и больше...

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 статистически доказать (мы не будем).

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:

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

- 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), $d = -b - AQ^{-1}c$.

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?

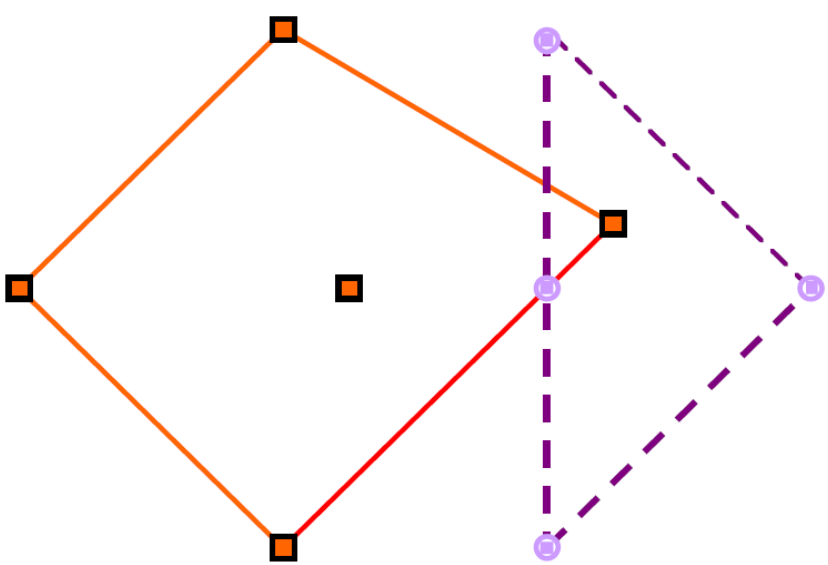
Редуцированные выпуклые оболочки

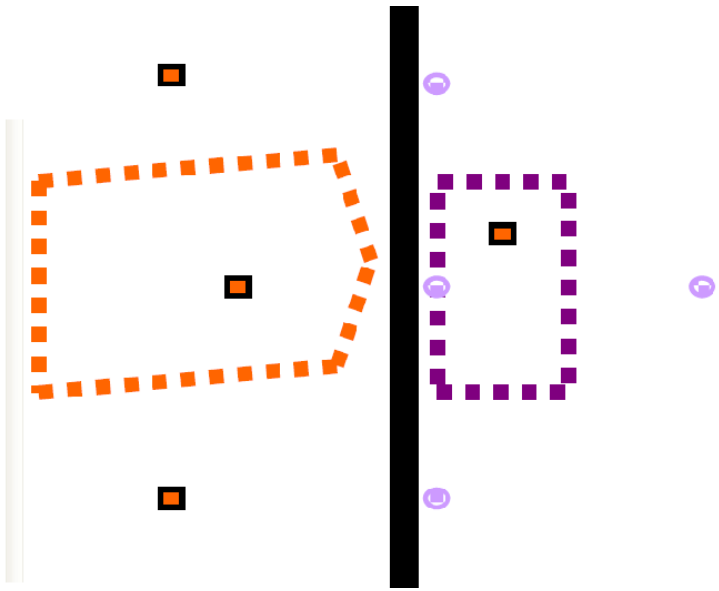
- Вместо обычных выпуклых оболочек можно рассматривать *редуцированные* (reduced), у которых коэффициенты ограничены не 1, а сильнее:

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

- 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 дуального метода опорных векторов тоже надо что-то изменить. Что?

Для метода опорных векторов

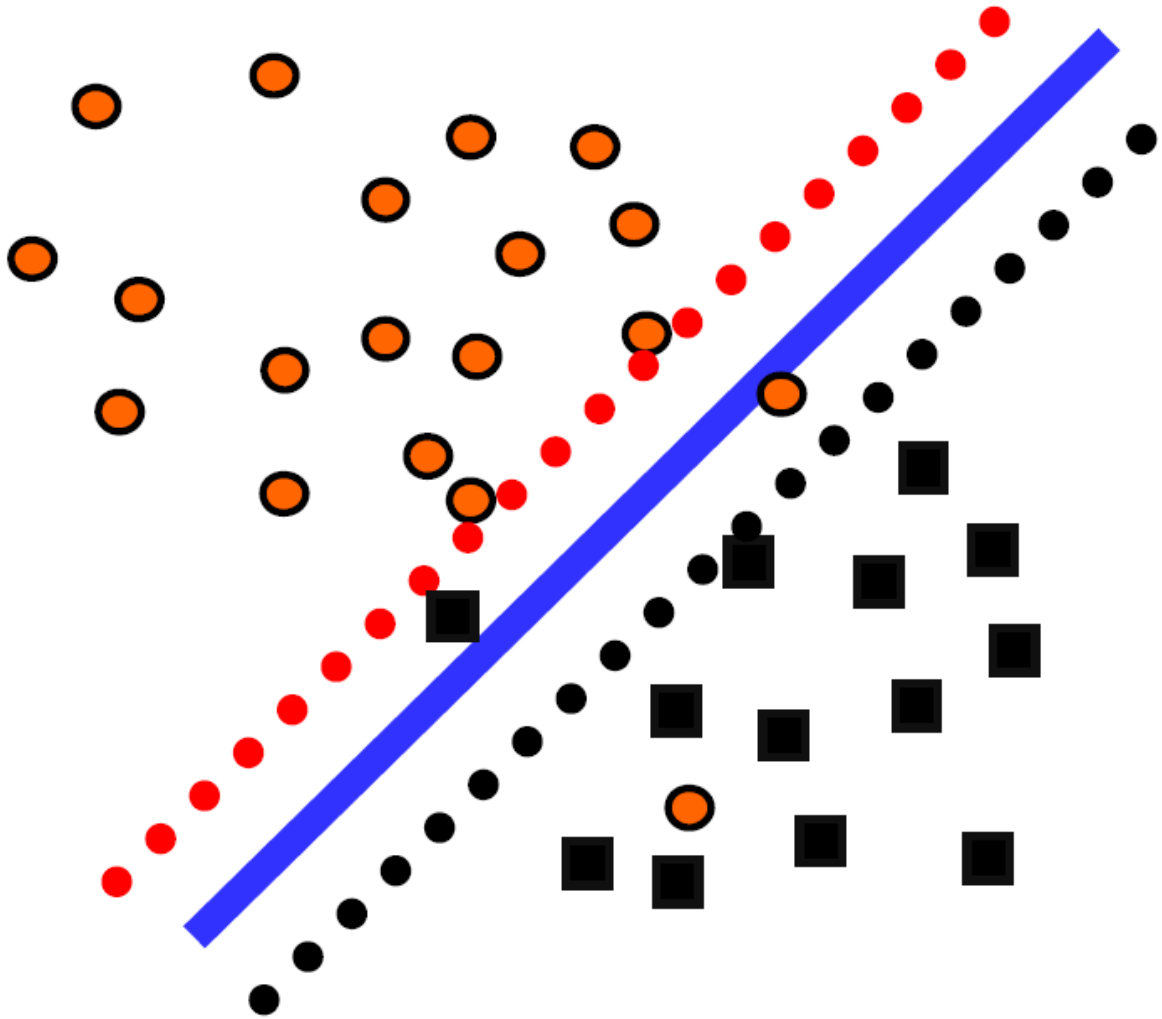
- Естественно, для дуального метода опорных векторов тоже надо что-то изменить. Что?
- Мы просто добавим в оптимизирующуюся функцию неотрицательную ошибку (slack):

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

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

- Это дуальная задача для поиска редуцированных выпуклых оболочек.

Пример



Дуальная переформулировка

- В дальнейшем нам скорее понадобится задача, дуальная к этой. Она выглядит так:

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

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

Итого

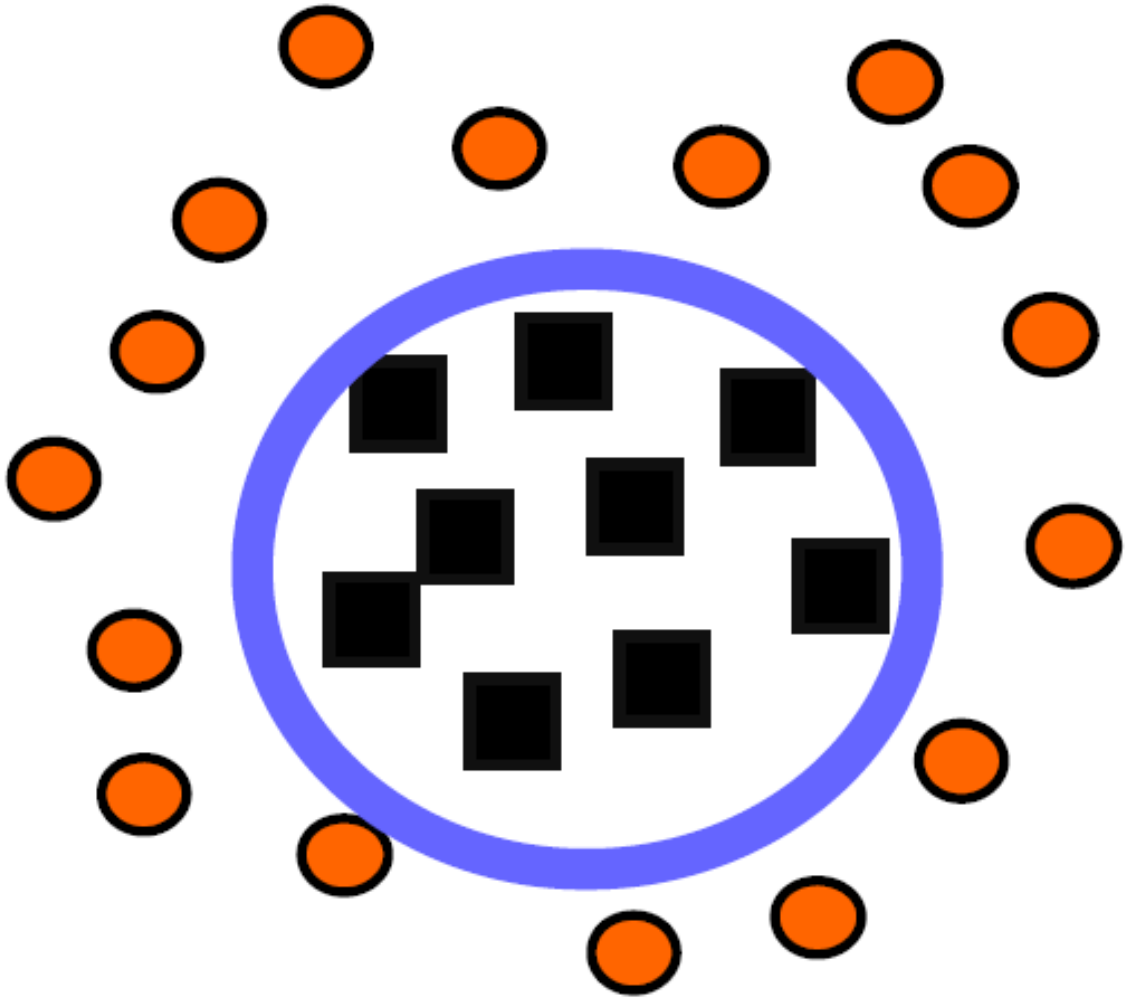
- Метод опорных векторов отлично подходит для линейной оптимизации.
- Решая задачу квадратичного программирования, мы получаем параметры оптимальной гиперплоскости.
- Точно так же, как и в дуальном случаей, если бы мы просто искали середину между выпуклыми оболочками.

Outline

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

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

- 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 $\mathbf{x}$ 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:

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

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 $K(\mathbf{u}, \mathbf{v}) \in L_2$ представляется в виде

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

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

$$\int K(\mathbf{u}, \mathbf{v}) \theta_i(\mathbf{u}) \mathbf{du} = \lambda_i \theta_i(\mathbf{v}).$$

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*)

$$\int\int K(\mathbf{u},\mathbf{v})g(\mathbf{u})g(\mathbf{v})\mathbf{du}\mathbf{dv} > 0$$

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

Теория Гильберта–Шмидта–Мерсера

- Итого задача наша выглядит так:

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

- 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:

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

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

Примеры ядер

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

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 b (как? подумайте!).
  5. Новые точки классифицировать как

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

Outline

Линейное программирование

- Квадратичное программирование всё-таки достаточно дорого. Может быть, линейного будет достаточно?
- Мы минимизировали архимедову норму вектора: $\min_{\mathbf{w},b} \left\{ \|\mathbf{w}\|^2 \right\}$.
- Но ведь можно попробовать и линейную норму минимизировать: $\min_{\mathbf{w},b} \{\|\mathbf{w}\|_1\}$, т.е. сумму модулей компонентов $\mathbf{w}$.

Линейное программирование

- Получится задача линейного программирования, которую можно решать ещё быстрее, и методы уже давно и хорошо реализованы (ILOG).
- А для нелинейной оптимизации можно так же переформулировать, но придётся делать это в feature space.
- Однако линейное программирование настолько быстрее квадратичного, и матрицы получатся настолько разреженными, что можно попробовать.

Упражнение. Выписать переформулировку на языке линейного программирования в feature space.

Суть задачи

- Часто интересно искать так называемые outliers — экстремальные точки, не лежащие внутри «границы» известных данных.
- Это нужно, чтобы фиксировать в данных нерегулярности, экстремальные случаи... яркий пример — медицинская диагностика.

Сфера

- Можно рассмотреть гиперсферу минимального радиуса R и минимизировать её объём и одновременно расстояние до аутлайеров:

$$\min_{R,z,a} \left\{ R^2 + \frac{1}{m\nu} \sum_{1}^{n} z_i \right\},$$

$$\text{где } (\mathbf{x}_i + a)^T (\mathbf{x}_i - a) \leq R^2 + z_i, \quad z_i \geq 0.$$

Сфера

- Теперь можно переформулировать то же самое для дуальной задачи:

$$\min_{\alpha} \left\{ -\sum_{1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}_i) + \sum_{1}^{n} \sum_{1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \right\},$$

$$\text{где } \sum_{1}^{n} \alpha_i = 1, \quad \frac{1}{h\nu} \geq \alpha_i \geq 0, i = 1..n.$$

Что с этим делать

- Если $m\nu > 1$, то *граничные* случаи — это случаи, в которых $\alpha_i = 1/m\nu$, и они соответствуют аутлайерам в тестовых данных.
- А после обучения, чтобы выяснить, аутлайер ли новая точка $\mathbf{v}$, нужно проверить

$$K(\mathbf{v}, \mathbf{v}) - 2\sum_{1}^{n} \alpha_i K(\mathbf{v}, x_i) + \sum_{1}^{n}\sum_{1}^{n} \alpha_i \alpha_j K(x_i, x_j) - R^2 \geq 0,$$

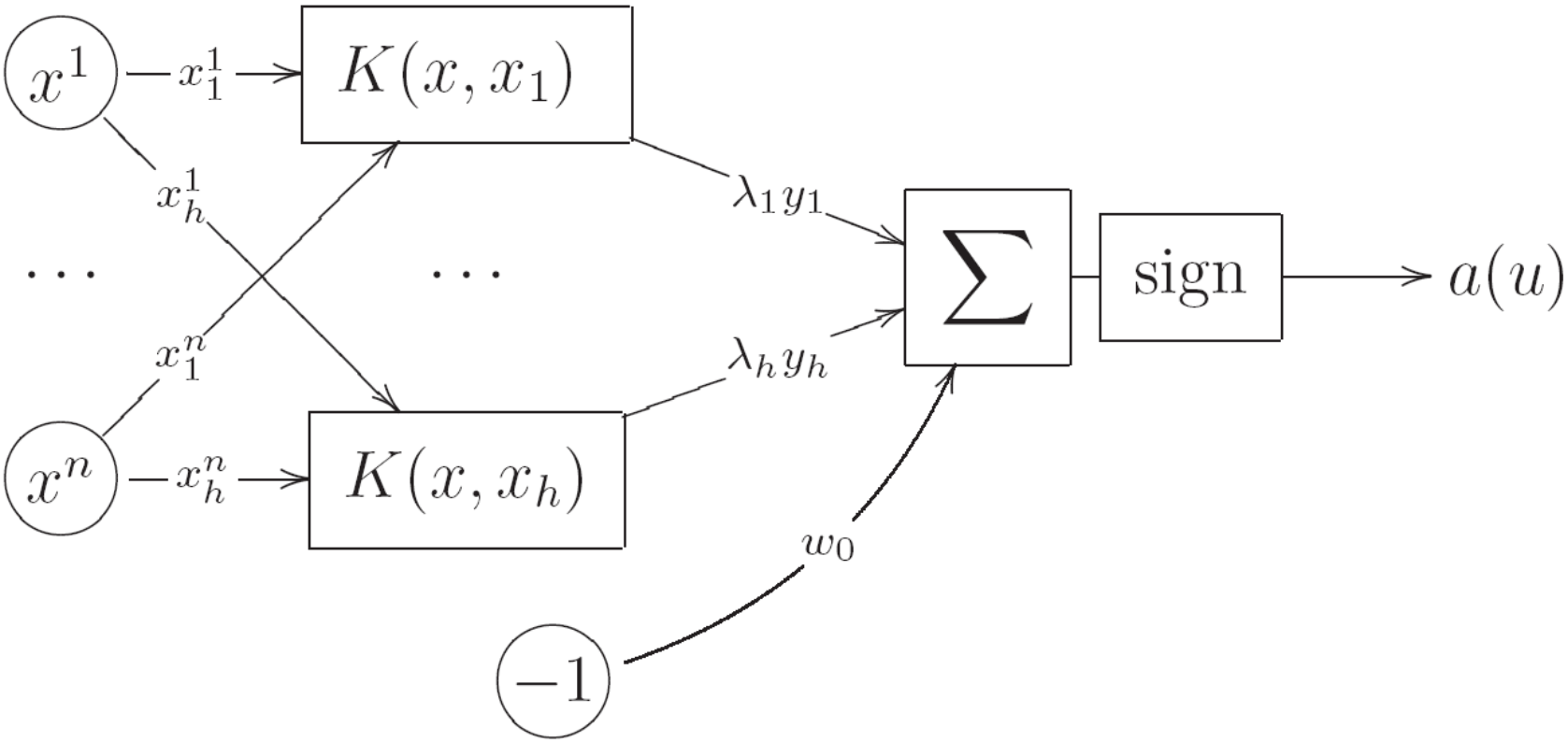
где R^2 нужно будет сначала вычислить, найдя *неграничный* пример и подобрав R так, чтобы это неравенство стало равенством.

Другие расширения

- Регрессия: SVM могут также вычислять регрессию, т.е. подбирать такую оптимальную прямую, чтобы как можно больше точек из данных лежали к ней как можно ближе (linear fitting).
- При помощи SVM можно выразить, например, нейронные сети, а также многие другие методы машинного обучения.
- И саму (уже обученную) SVM, кстати, можно условно выразить в виде нейронной сети. Вспомним, что классификация ведётся как

$$f(\mathbf{x}) = \mathrm{sign}(\sum_i y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) - b).$$

SVM как ANN


x^1
x_1^1
$K(x, x_1)$
x_h^1
$\lambda_1 y_1$
$\cdots$
$\cdots$
$\sum$
sign
$a(u)$
x_1^n
$\lambda_h y_h$
x^n
x_h^n
$K(x, x_h)$
w_0
-1

Преимущества

- Нет локальных минимумов! Вообще.
- Мало параметров надо выбирать (раньше во многих алгоритмах нужно было как-то из воздуха доставать правильные параметры). В частности, «число нейронов на скрытом уровне» выбирается автоматически (это число опорных векторов).
- Результаты стабильны и воспроизводимы, независимы от деталей конкретной реализации.
- Уже существуют хорошие алгоритмы и они постоянно улучшаются, потому что математически SVM — это просто классическая задача квадратичной оптимизации.
- Метод прост, уже существует множество применений.

Недостатки

- Метод крайне неустойчив к шуму в тестовых данных, будет происходить оверфиттинг (слишком много опорных векторов будет получаться). С этим борются relevance vector machines (RVM) – о них, может быть, позже.
- Надо всё-таки подбирать параметр C.
- Откуда брать ядро – непонятно, это пока что скорее искусство.

Thank you!

Спасибо за внимание!