SVM и kernel methods

Сергей Николенко

Казанский Федеральный Университет, 2014

Outline

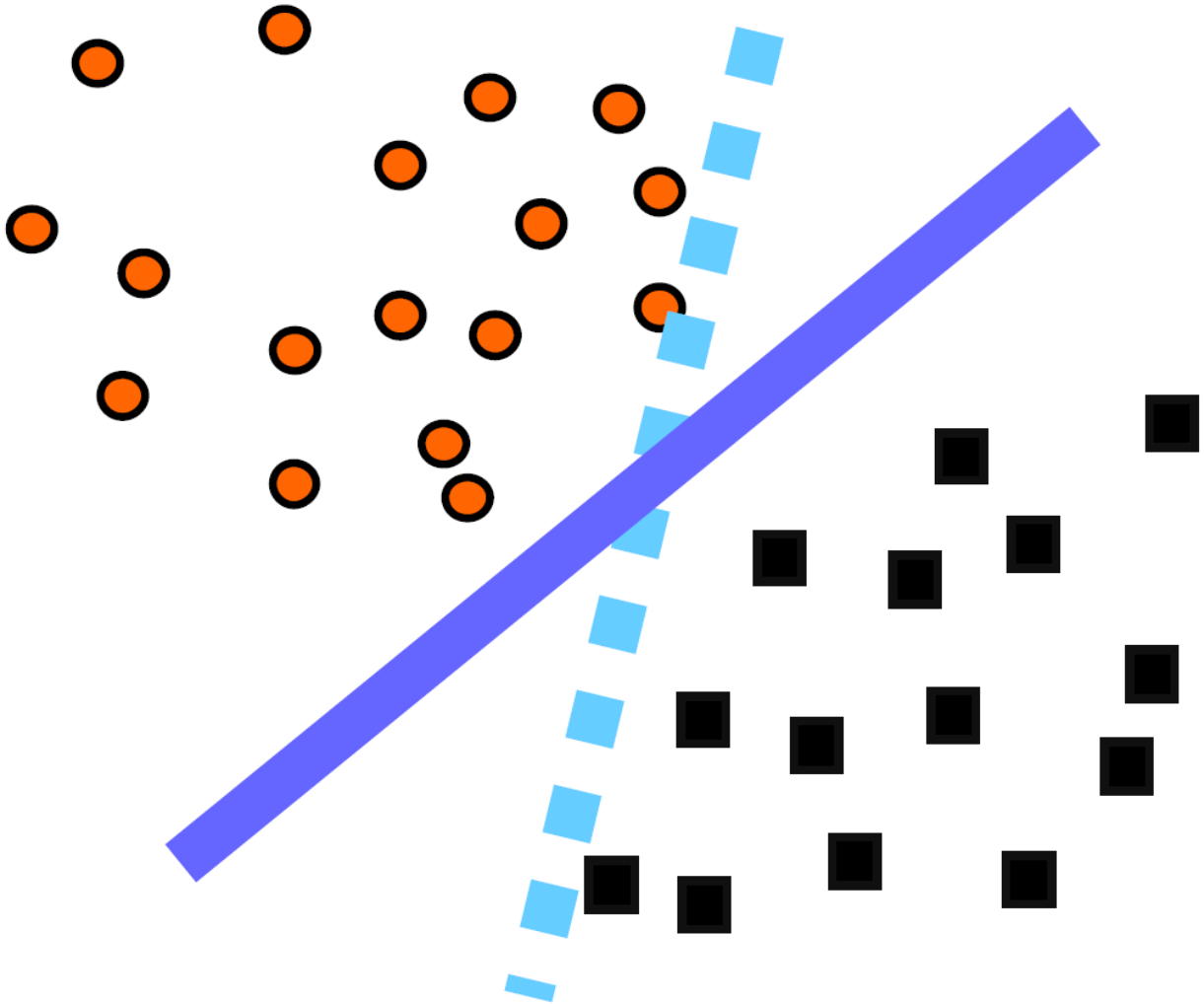
Постановка задачи

- 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

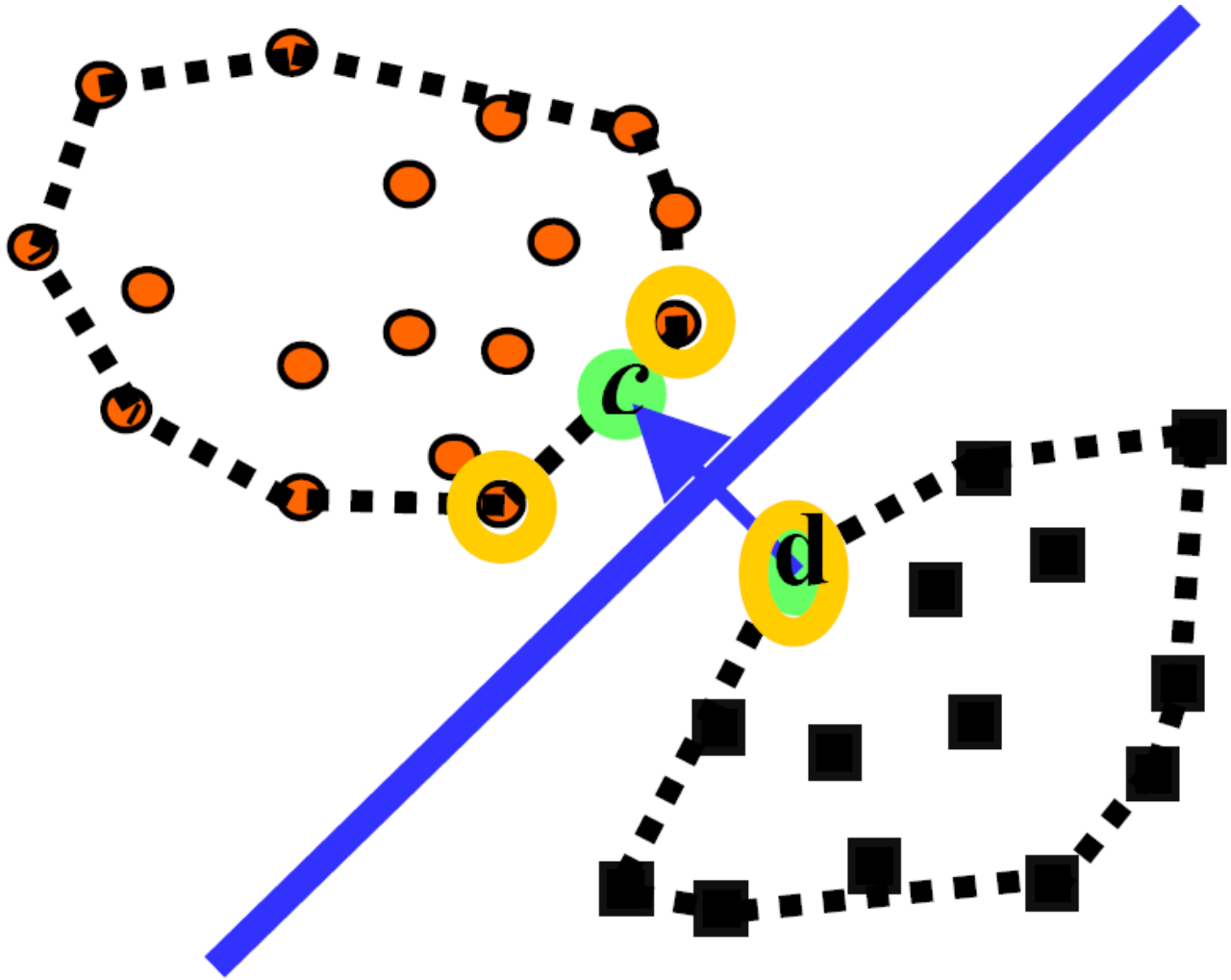
- Один подход: найти две ближайшие точки в выпуклых оболочках данных, а затем провести разделяющую гиперплоскость через середину отрезка.
- Формально это превращается в задачу квадратичной оптимизации:

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

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

- 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: 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

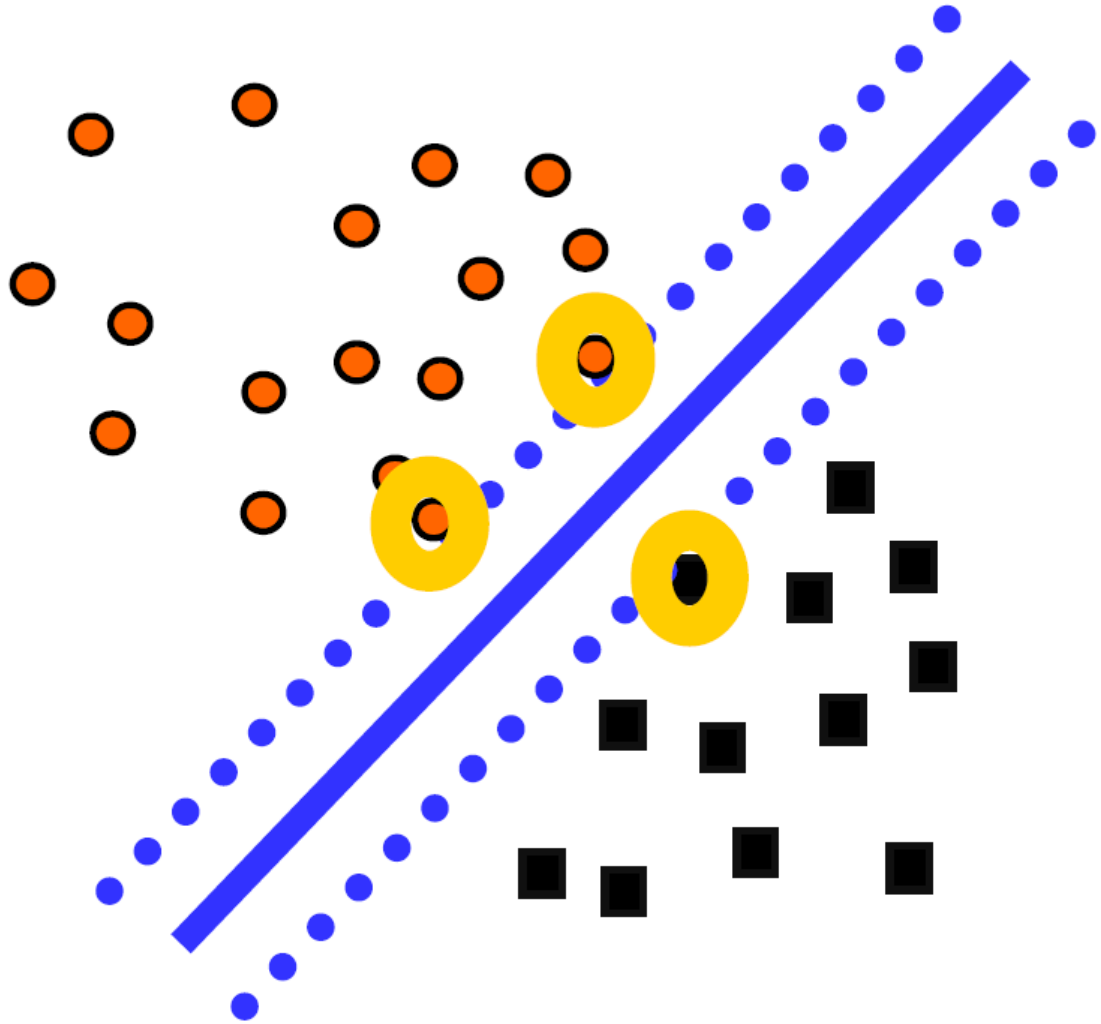
$$\arg\max_{\mathbf{w}, w_0} \min_n \frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} =$$

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

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

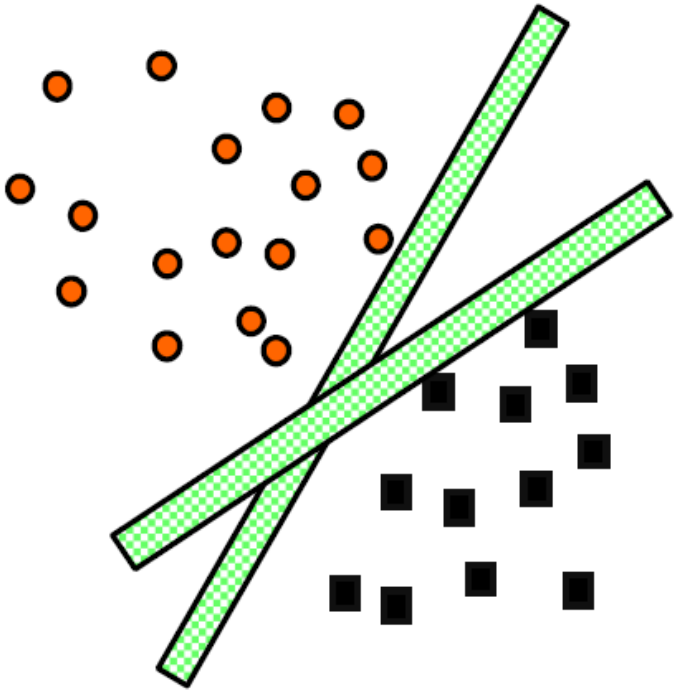
- Получается тоже задача квадратичного программирования:

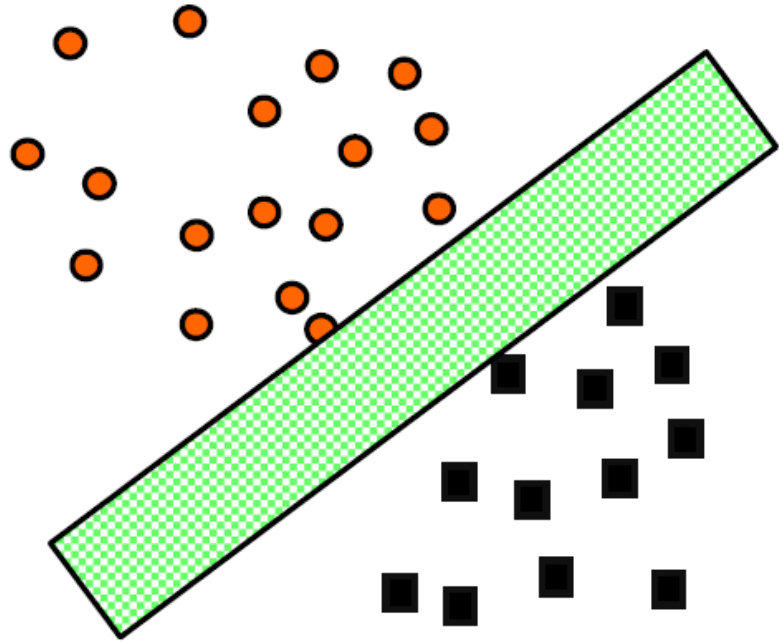
$$\min_{\mathbf{w},b}\left\{\frac{1}{2}\|\mathbf{w}\|^2\right\} \text{ при условии } t_n(\mathbf{w}^\top \mathbf{x}_n + w_0) \geq 1.$$

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:

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

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:

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

- 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

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

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 KKT

- ...но нет. :) Условия KKT (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?

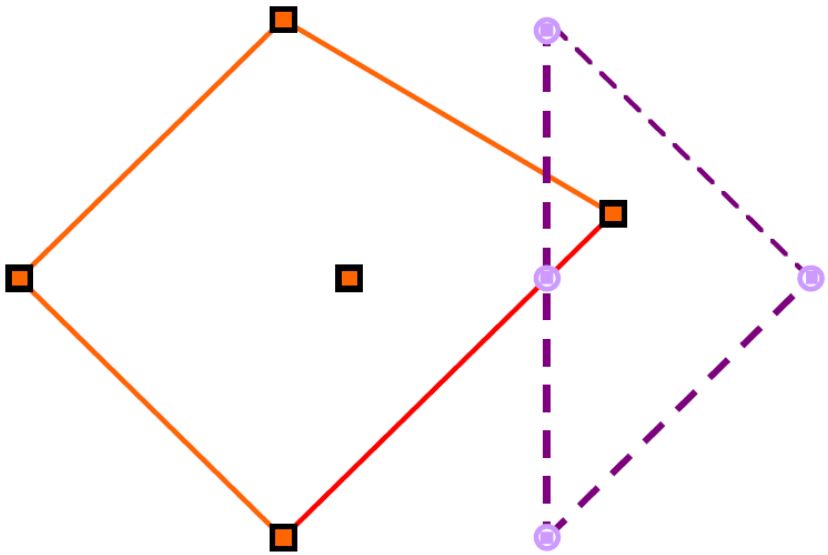
Редуцированные выпуклые оболочки

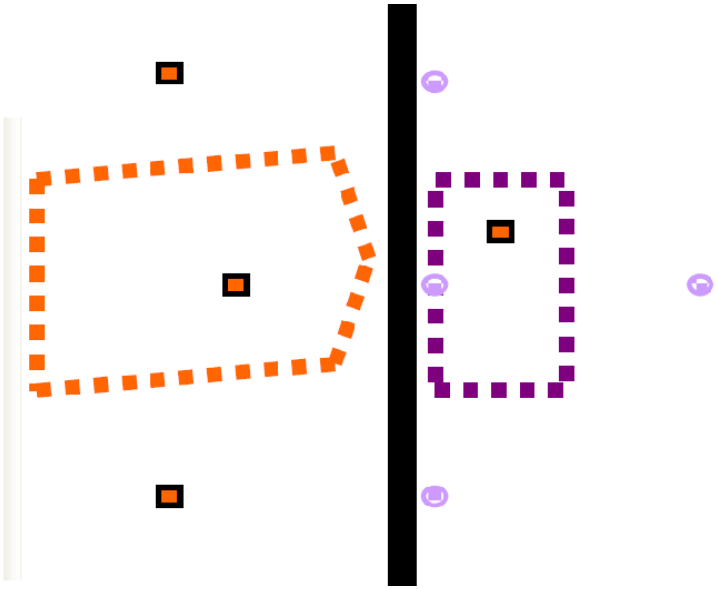
- Вместо обычных выпуклых оболочек можно рассматривать *редуцированные* (reduced), у которых коэффициенты ограничены не 1, а сильнее:

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

- 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

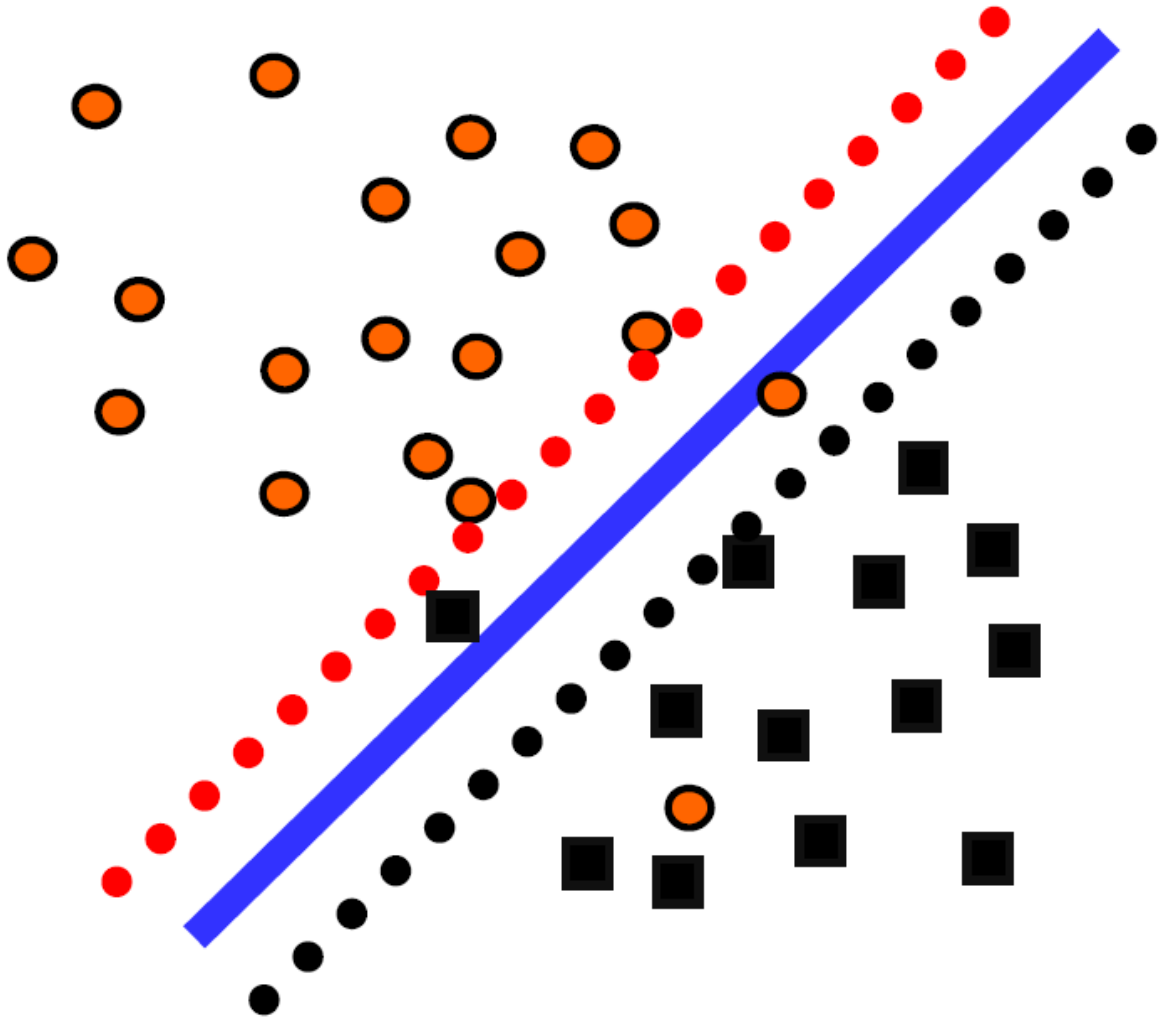
- Естественно, для метода опорных векторов тоже надо что-то изменить. Что?
- Мы просто добавим в оптимизирующуюся функцию неотрицательную ошибку (slack):

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

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

- Это прямая задача...

Пример



Дуальная переформулировка

- ...а вот дуальная:

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

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

Итого

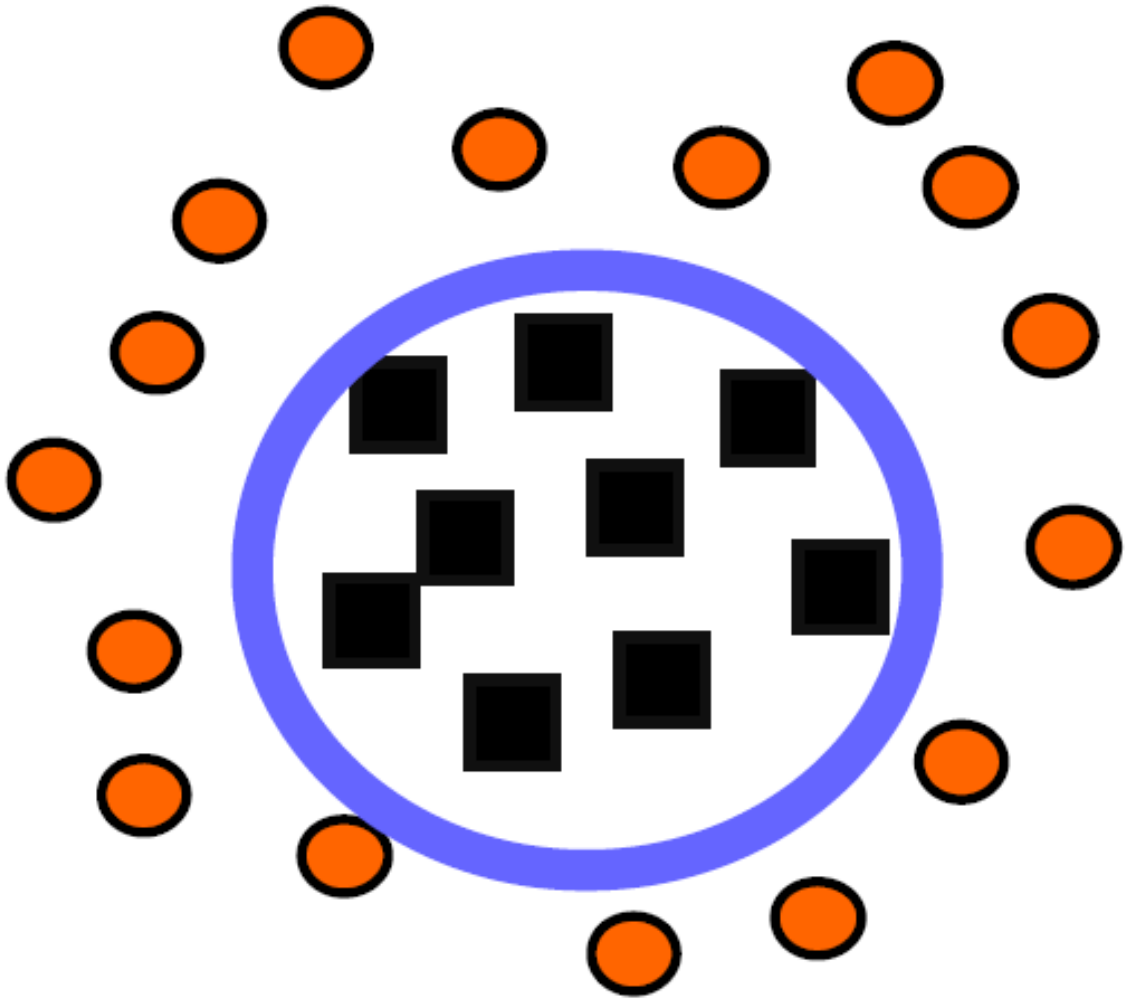
- Метод опорных векторов отлично подходит для линейной классификации.
- Решая задачу квадратичного программирования, мы получаем параметры оптимальной гиперплоскости.
- Точно так же, как и в дуальном случае, если бы мы просто искали середину между выпуклыми оболочками.

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

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

- 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 $\mathbf{x}$ 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:

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

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 $k(\mathbf{u}, \mathbf{v}) \in L_2$ 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.

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

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:

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

- 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:

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

- 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):

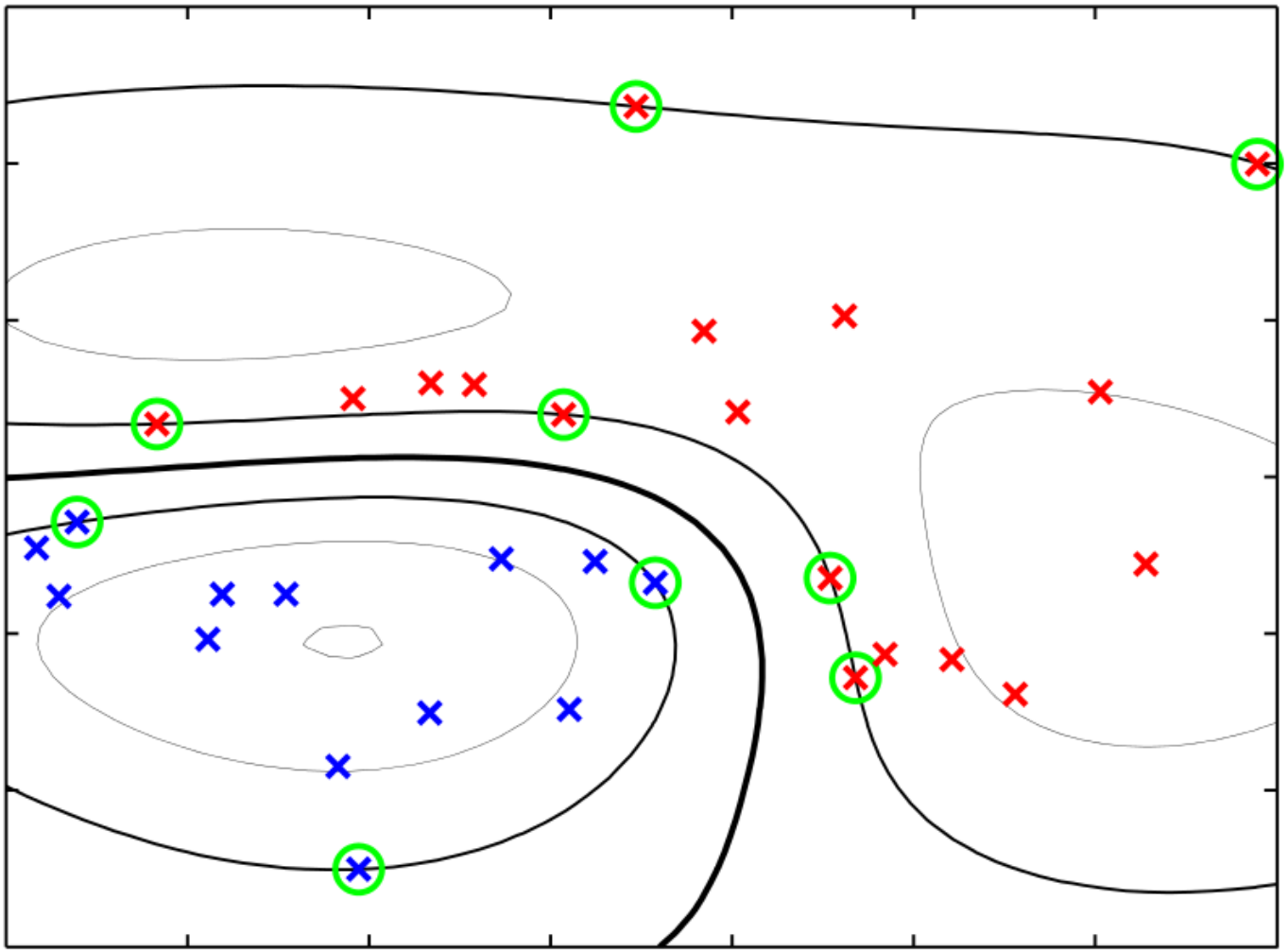
$$k(\mathbf{u}, \mathbf{v}) = e^{-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma}}.$$

- Двухуровневая нейронная сеть:

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

где o — сигмоид.

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

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

Thank you!

Спасибо за внимание!