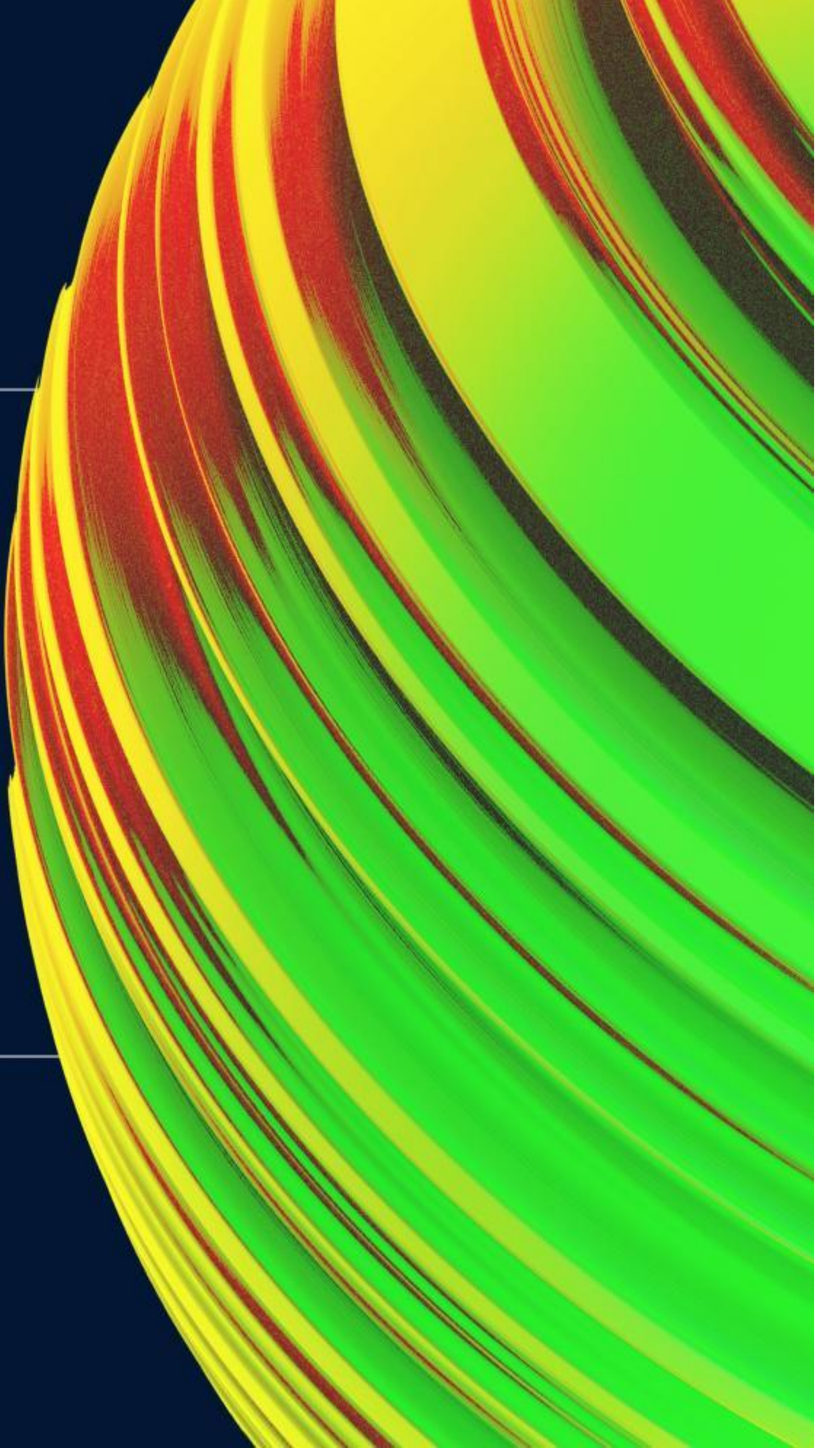
УНИВЕРСИТЕТ ИННОПОЛИС

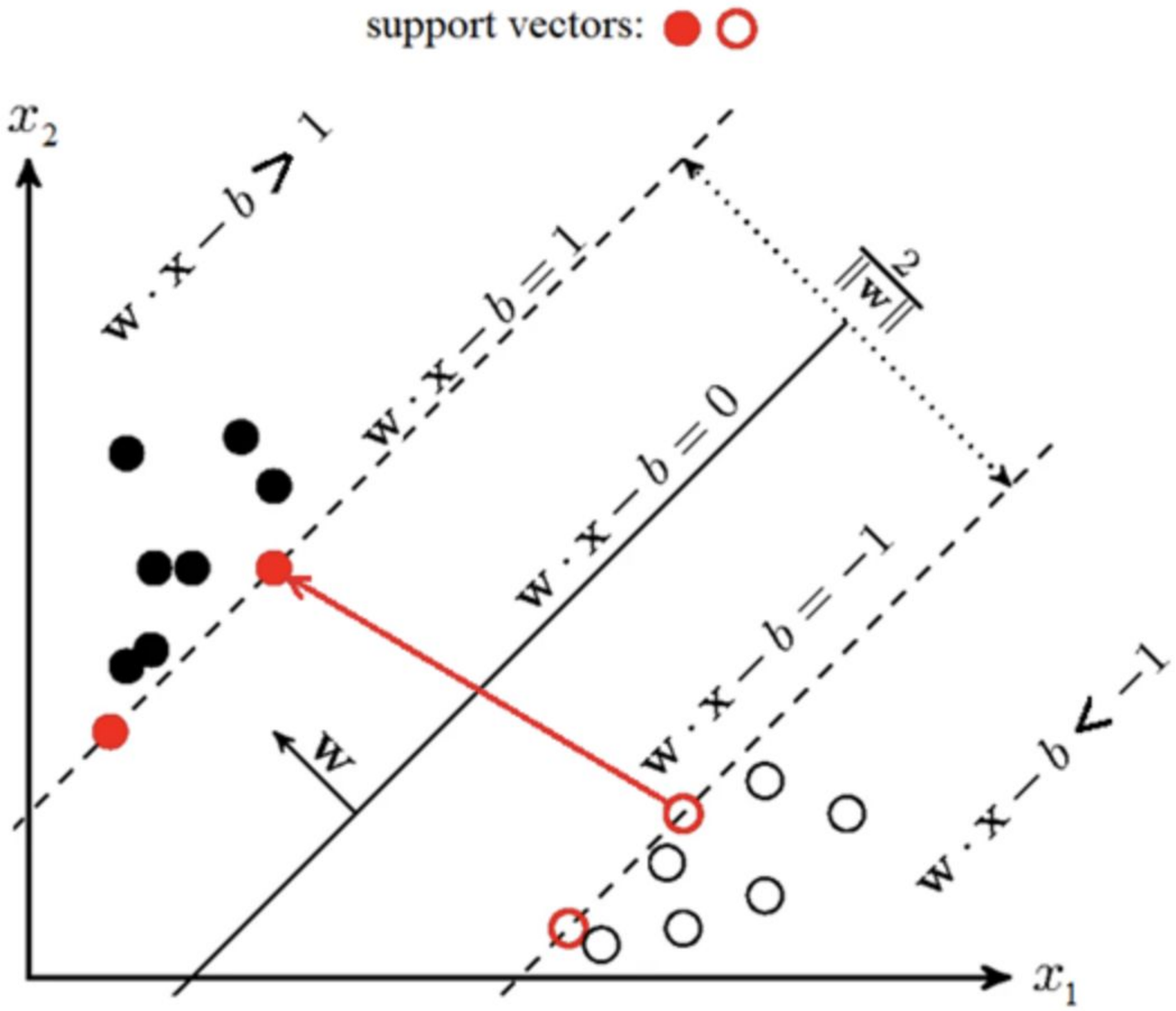
SVM. Часть 2

Воробьёва Мария

- maria.vorobyova.ser@gmail.com
- @SparrowMaria



Алгоритм SVM



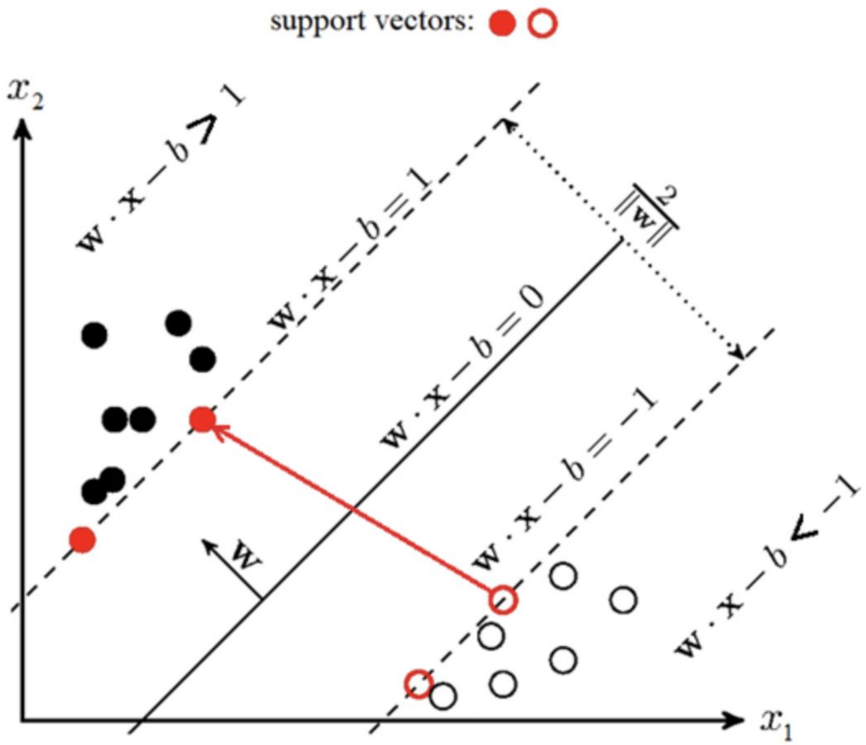
Задача SVM с жестким зазором

(hard-margin SVM)

$$\begin{cases} (w^T w)/2 \to min \\ y(w^T x - b) \geqslant 1 \end{cases}$$

Решается аналитически через теорему Куна-Таккера. Получаемая задача эквивалентна двойственной задаче поиска седловой точки функции Лагранжа.

Алгоритм SVM

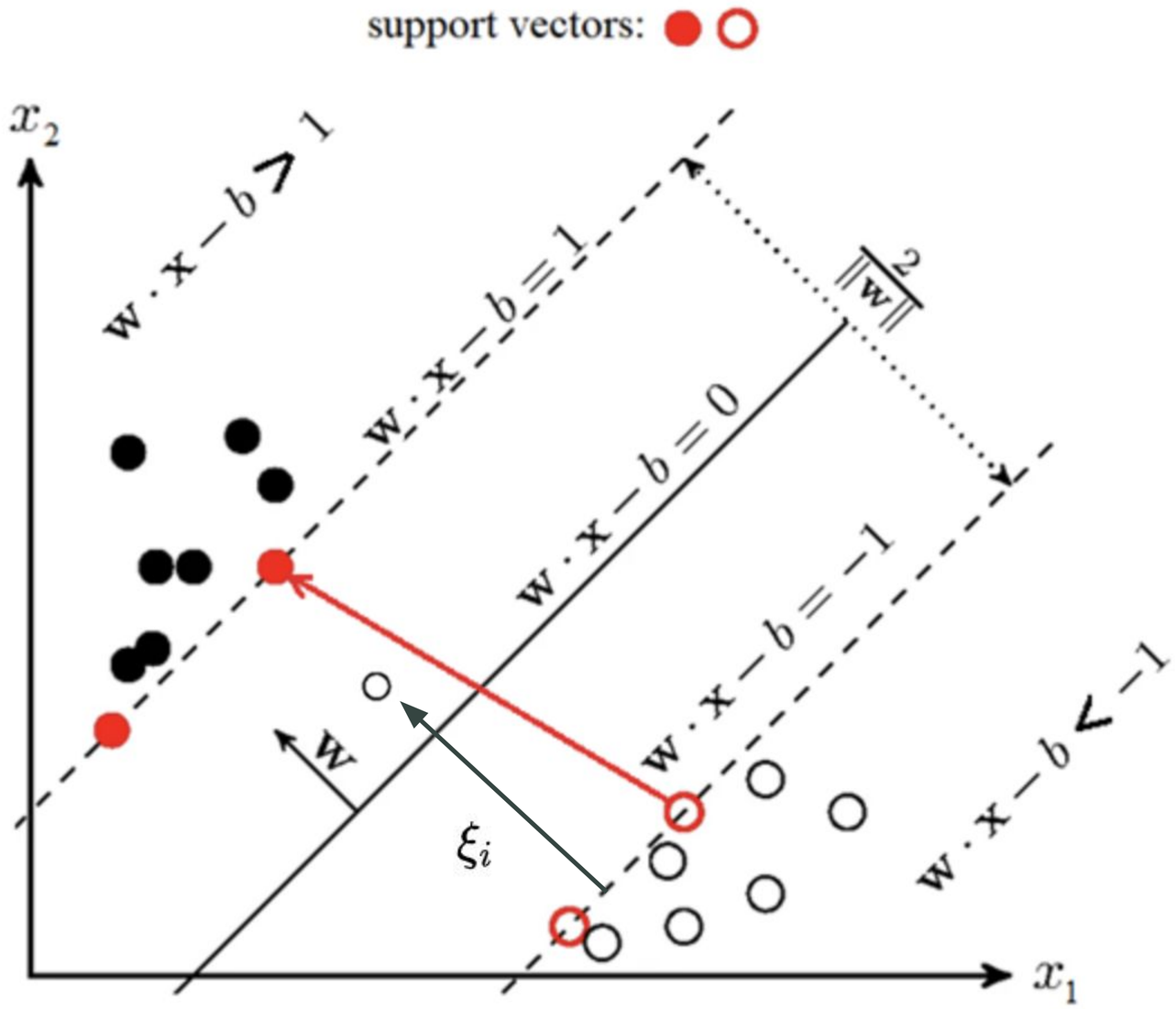


В итоге мы получаем дефолтную настройку SVM с жестким зазором (hard-margin SVM)

$$\begin{cases} (w^T w)/2 \to min \\ y(w^T x - b) \geqslant 1 \end{cases}$$

Почему именно ≥ 1? Это связано с целью максимизации зазора. В задачах SVM с жестким зазором образцы, которые находятся непосредственно на границе, удовлетворяют уравнению $y(w^T x - b) = 1$. Образцы, которые лежат дальше от границы, удовлетворяют неравенству $y(w^T x - b) > 1$.

Алгоритм SVM



Пусть алгоритм допускает ошибки на обучающих объектах, но при этом постараемся, чтобы ошибок было поменьше.

$\xi_i > 0$ - величина ошибки на каждом объекте x_i

$$\begin{cases} (w^T w)/2 + \alpha \sum \xi_i \to min \\ y(w^T x_i - b) \geqslant 1 - \xi_i \\ \xi_i \geqslant 0 \end{cases}$$

Алгоритм SVM

INNOPOLIS UNIVERSITY

$$\begin{cases} (w^T w)/2 + \alpha \sum \xi_i \to min \\ y(w^T x_i - b) \geqslant 1 - \xi_i \\ \xi_i \geqslant 0 \end{cases}$$

- w – это вектор весов,
- α – гиперпараметр, контролирующий баланс между шириной зазора и штрафом за ошибки классификации,
- ξ_i – величина мягкого зазора для каждого образца.

$$y_i(w^T x_i - b) \geq 1 - \xi_i$$

- y_i – метка класса для i-го образца,
- x_i – вектор признаков i-го образца,
- b – смещение (bias).

Это ограничение означает, что каждый образец должен быть правильно классифицирован с учетом мягкого зазора ξ_i. Если образец правильно классифицирован, то ξ_i равно нулю. Если образец классифицирован неправильно, то ξ_i будет положительным и отражает степень ошибки.

Таким образом, задача сводится к нахождению такого вектора весов w и смещения b, которые максимизирует $\frac{2}{w \cdot w^T}$ ширины зазора и минимизирует штрафы за ошибки классификации, одновременно удовлетворяя ограничениям на правильную классификацию образцов с учетом мягкого зазора.

Алгоритм SVM

innopolis university

$$\begin{cases} (w^T w)/2 + \alpha \sum \xi_i \to min \\ y(w^T x_i - b) \geqslant 1 - \xi_i \\ \xi_i \geqslant 0 \end{cases}$$

Формулирование Лагранжевой функции
Мы можем использовать метод множителей Лагранжа для преобразования этой задачи в оптимизацию без ограничений. Лагранжева функция для этой задачи:

$$L(w, b, \xi, \alpha, \lambda, \mu) = \frac{1}{2}(w^T w) + \alpha \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \lambda_i [y_i(w^T x_i - b) - 1 + \xi_i] - \sum_{i=1}^{n} \mu_i \xi_i$$

где $\lambda_i \geq 0$ и $\mu_i \geq 0$ — множители Лагранжа.

Алгоритм SVM

Двойственная задача
подробнее http://machinelearning.ru/wiki/images/2/25/SMAIS11_SVM.pdf

В оригинальной (**прямой**) задаче SVM, если **пространство признаков имеет высокую размерность**, оптимизация может быть вычислительно затратной.

Двойственная задача выражается через переменные Лагранжа λ, число которых равно числу обучающих примеров. Это может быть более эффективно для больших наборов данных, особенно если **число признаков значительно больше числа примеров**.

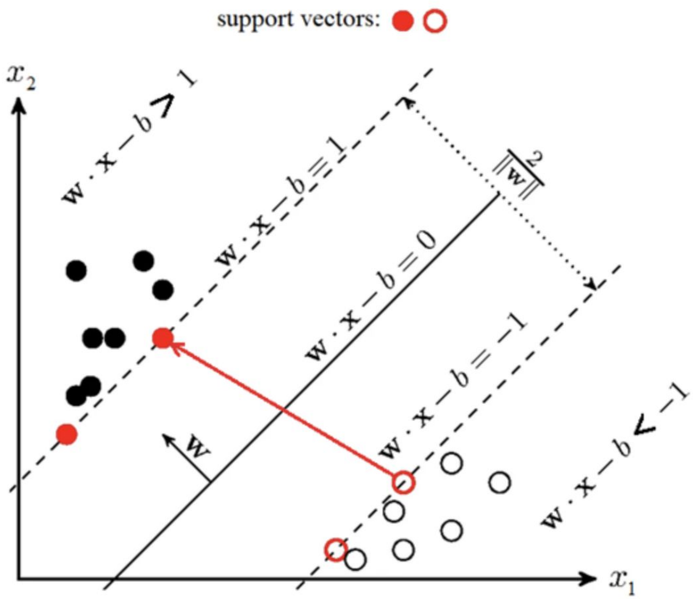
$$\begin{cases} \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j (x_i^T x_j) \to \max \\ \sum_{i=1}^{n} \lambda_i y_i = 0 \\ 0 \leq \lambda_i \leq \alpha \end{cases}$$

Алгоритм SVM

Функция потерь hinge определяется как:

$$L(y_i, f(x_i)) = \max(0, 1 - y_i \cdot f(x_i))$$

где $f(x_i) = w^T x_i - b$.



Переменные ξ_i связаны с функцией потерь hinge следующим образом:

$$\xi_i = \max(0, 1 - y_i(w^T x_i - b))$$

Это выражение говорит нам, что ξ_i представляет собой положительную часть отклонения от идеального случая, когда $y_i(w^T x_i - b) \geq 1$.

Алгоритм SVM

Было:

$$\begin{cases} (w^T w)/2 + \alpha \sum \xi_i \to min \\ y(w^T x_i - b) \geqslant 1 - \xi_i \\ \xi_i \geqslant 0 \end{cases}$$

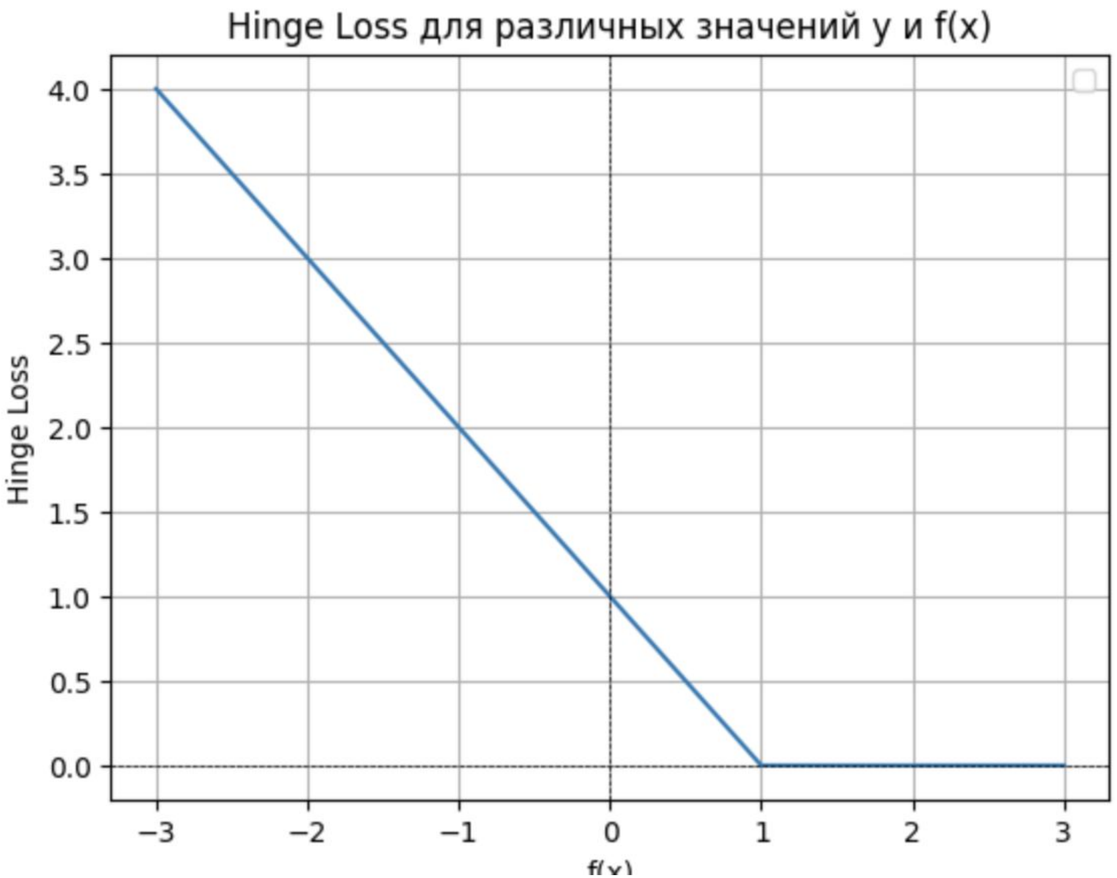
Стало:

$$\min_{w,b} \left(\frac{1}{2} \|w\|^2 + \alpha \sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i - b)) \right)$$

Пояснение исчезновения условий

1. **Условие** $y_i(w^T x_i - b) \geq 1 - \xi_i$:
   - Это условие учитывается в функции потерь hinge. Величина $\max(0, 1 - y_i(w^T x_i - b))$ непосредственно соответствует этому условию. Если $y_i(w^T x_i - b) \geq 1$, то $1 - y_i(w^T x_i - b) \leq 0$, и $\max(0, 1 - y_i(w^T x_i - b)) = 0$. Если $y_i(w^T x_i - b) < 1$, то $\max(0, 1 - y_i(w^T x_i - b))$ будет положительным и отразит отклонение от этого условия.
2. **Условие** $\xi_i \geq 0$:
   - Величина $\max(0, 1 - y_i(w^T x_i - b))$ по определению всегда неотрицательна. Это означает, что ξ_i всегда будет больше или равно нулю, и это условие выполняется автоматически.

Алгоритм SVM. Hinge Loss



$$Q(X,w) = \alpha \sum_{i=1}^{l} \underbrace{\max\{0, 1 - y_i(\langle x_i, w\rangle)\}} + \|w\|^2 \to \min_w$$

Hinge loss для одного объекта

Hinge loss штрафует неправильные предсказания, пропорционально их отступу от правильного класса.

Большое значение α :

- hinge loss -> меньше
- точность на обучающей выборке -> больше

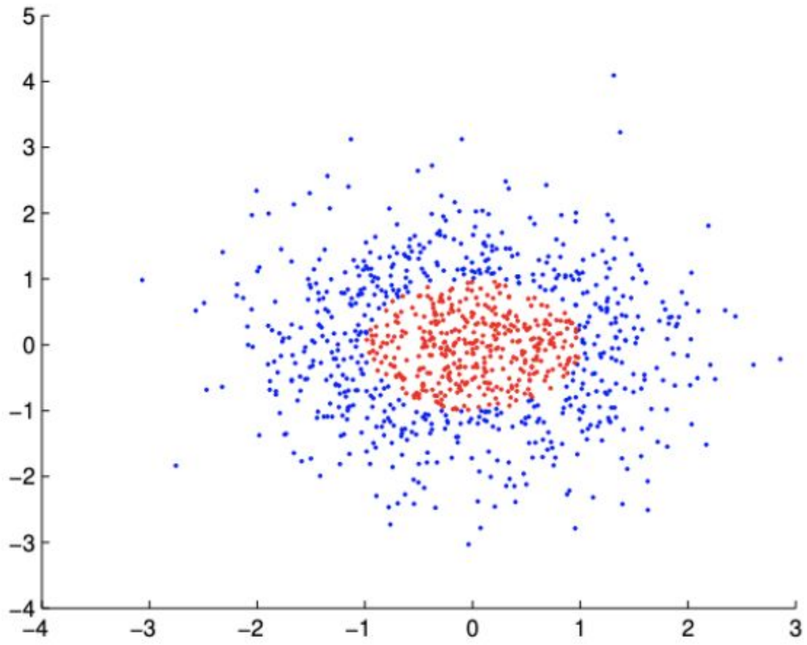
  ->риск переобучения

Маленькое значение α

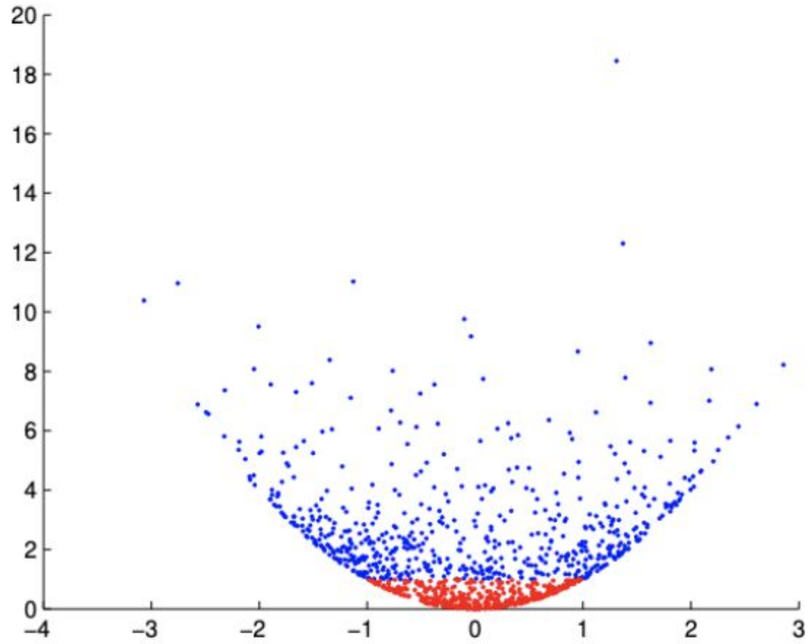
- hinge loss -> больше
- точность на обучающей выборке -> меньше

  ->риск недообучения

Алгоритм SVM. Нелинейный случай



Добавили третий признак - квадрат признака и выборка стала линейно разделимой



https://github.com/esokolov/ml-course-hse/blob/master/2021-spring/lecture-notes/lecture13-kernels.pdf

Алгоритм SVM. Ядерные функции. Kernel

INNOPOLIS UNIVERSITY

Ядро - любая симметричная, положительно полуопределенная матрица K, которая составлена из скалярных произведений пар векторов x_i и x_j:

$K(x_i, x_j) = \left\langle \phi(x_i) \,|\, \phi(x_j) \right\rangle$, характеризующих меру их близости.

Здесь ϕ - произвольная преобразующая функция, формирующая ядро

https://ru.wikipedia.org/wiki/Положительно_определённая_матрица

Алгоритм SVM. Ядерные функции. Kernel

Предположим, что у нас есть отображение ϕ, которое преобразует исходные признаки x в более высокоразмерное пространство:

$$\phi : \mathbb{R}^n \to \mathbb{R}^m$$

где $m > n$.

Например, пусть x — это двумерный вектор $[x_1, x_2]$. Тогда $\phi(x)$ может быть трехмерным вектором, например, $[x_1, x_2, x_1^2 + x_2^2]$.

В новом пространстве задача оптимизации SVM выглядит следующим образом:

$$\min_{w,b} \left(\frac{1}{2}\|w\|^2 + \alpha \sum_{i=1}^{n} \max(0, 1 - y_i(w^T \phi(x_i) - b)) \right)$$

Алгоритм SVM. Ядерные функции. Kernel

То есть можно применить явные преобразования для x_i и x_j, но это долго, можно применить Kernel Trick.

Вместо того, чтобы вычислять новые координаты в этом пространстве, Kernel Trick позволяет нам вычислять скалярные произведения в этом высокоразмерном пространстве напрямую с помощью ядровых функций.

Предположим, что x — это двумерный вектор $[x_1, x_2]$. Тогда преобразование ϕ может быть, например:

$$\phi(x) = [x_1, x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]$$

$$\phi(x_i) \cdot \phi(x_j) = x_{i1}x_{j1} + x_{i2}x_{j2} + x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2}$$

Теперь возьмем полиномиальное ядро $K(x_i, x_j) = (x_i \cdot x_j + 1)^2$

$$(x_i \cdot x_j + 1)^2 = (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)^2 = x_{i1}^2x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_i$$

Алгоритм SVM. Ядерные функции. Kernel

Когда данные не могут быть линейно разделены в исходном пространстве признаков, мы можем преобразовать их в более высокоразмерное пространство, где они могут стать линейно разделимыми. Вместо того, чтобы вычислять новые координаты в этом пространстве, kernel trick позволяет нам вычислять скалярные произведения в этом высокоразмерном пространстве напрямую с помощью ядровых функций.

$$\begin{cases} \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j K(x_i, x_j) \to \max \\ \sum_{i=1}^{n} \lambda_i y_i = 0 \\ 0 \leq \lambda_i \leq \alpha \end{cases}$$

Алгоритм SVM. Ядерные функции. Kernel

INNOPOLIS UNIVERSITY

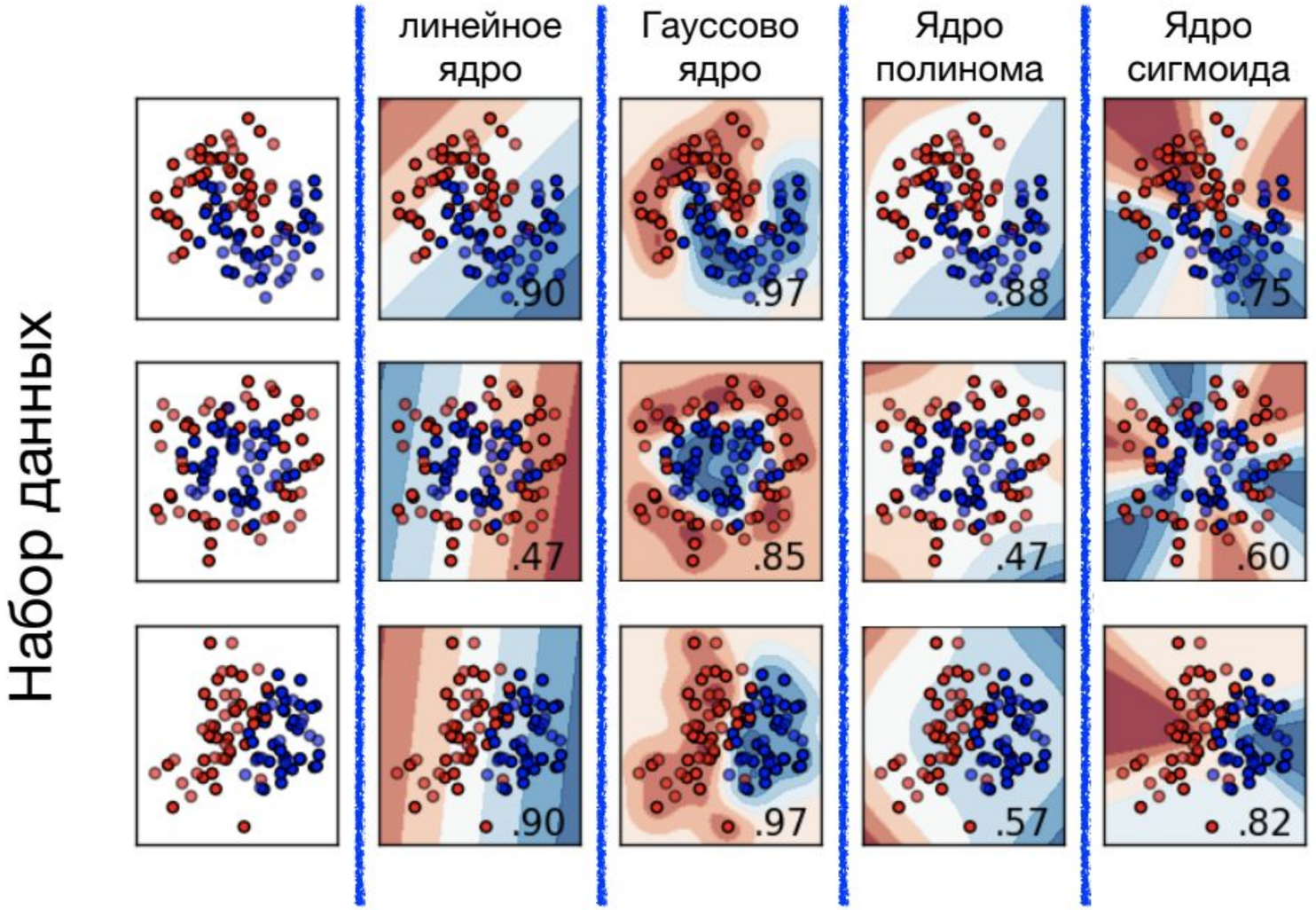
линейное ядро $K(x_i, x_j) = x_i^T x_j$

полиномиальное ядро со степенью p, $K(x_i, x_j) = (1 + x_i^T x_j)^p$

гауссово ядро с радиальной базовой функцией (RBF), $K(x_i, x_j) = e^{\gamma ||x_i - x_j||^2}$

сигмоидное ядро, $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + \beta_0)$

Алгоритм SVM. Ядерные функции. Kernel



УНИВЕРСИТЕТ
ИННОПОЛИС

ВОПРОСЫ И ОТВЕТЫ