Лекция 7
Метод опорных векторов. Ядровой трюк.

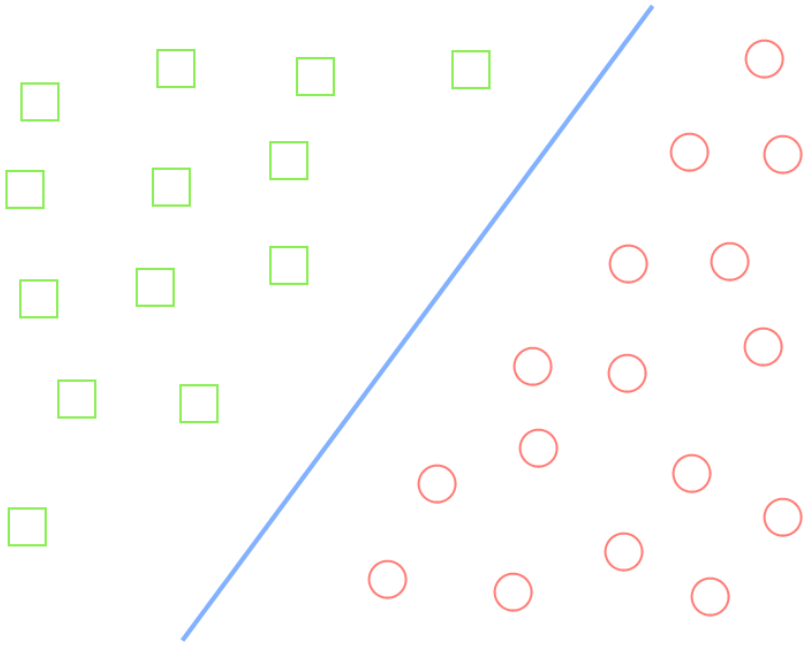
Елена Кантонистова

ВШЭ, 2023

МЕТОД ОПОРНЫХ ВЕКТОРОВ

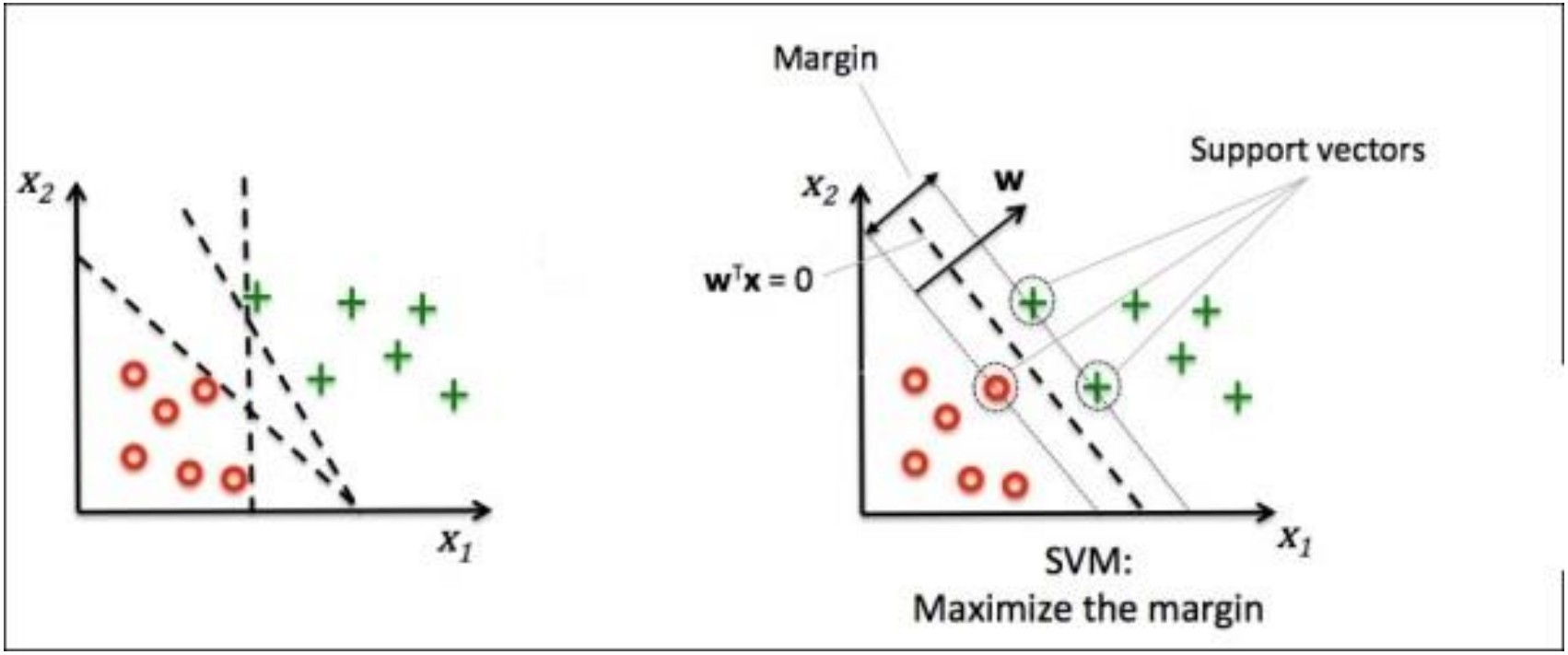
ЛИНЕЙНО РАЗДЕЛИМАЯ ВЫБОРКА

Выборка ***линейно разделима***, если существует такой вектор параметров w^*, что соответствующий классификатор $a(x)$ не допускает ошибок на этой выборке.



МЕТОД ОПОРНЫХ ВЕКТОРОВ: РАЗДЕЛИМЫЙ СЛУЧАЙ

Цель метода опорных векторов (Support Vector Machine) – максимизировать ширину разделяющей полосы.



МЕТОД ОПОРНЫХ ВЕКТОРОВ: РАЗДЕЛИМЫЙ СЛУЧАЙ

- $a(x) = sign\big((w, x) + w_0\big)$
- Нормируем параметры w и w_0 так, что

$$\min_{x \in X} |(w, x) + w_0| = 1$$

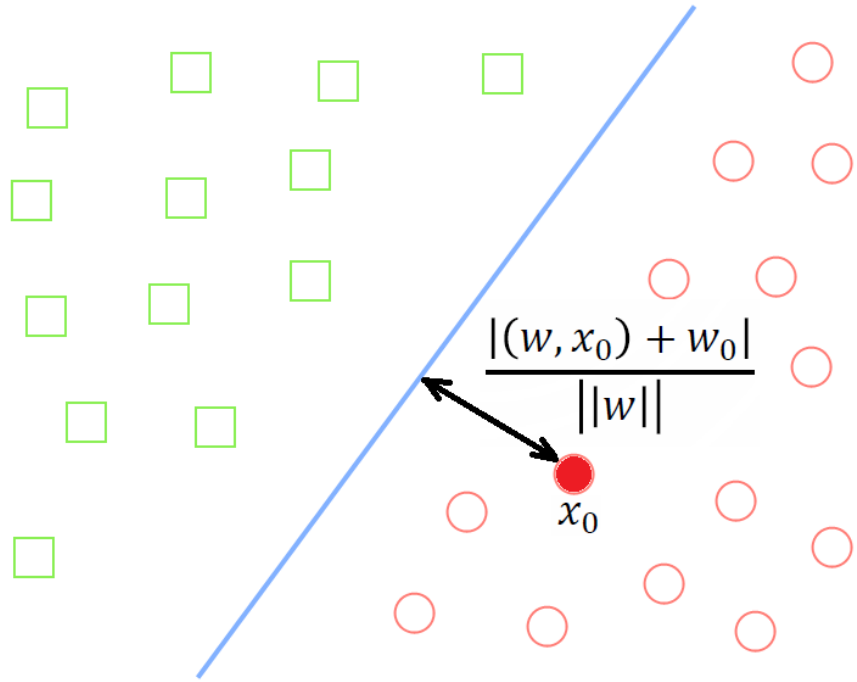
МЕТОД ОПОРНЫХ ВЕКТОРОВ: РАЗДЕЛИМЫЙ СЛУЧАЙ

- $a(x) = sign\big((w, x) + w_0\big)$
- Нормируем параметры w и w_0 так, что

$$\min_{x \in X} |(w, x) + w_0| = 1$$

Расстояние от точки x_0 до разделяющей гиперплоскости, задаваемой классификатором:

$$\rho(x_0, a) = \frac{|(w, x_0) + w_0|}{||w||}$$



МЕТОД ОПОРНЫХ ВЕКТОРОВ: РАЗДЕЛИМЫЙ СЛУЧАЙ

- Нормируем параметры w и w_0 так, что

$$\min_{x \in X} |(w, x) + w_0| = 1$$

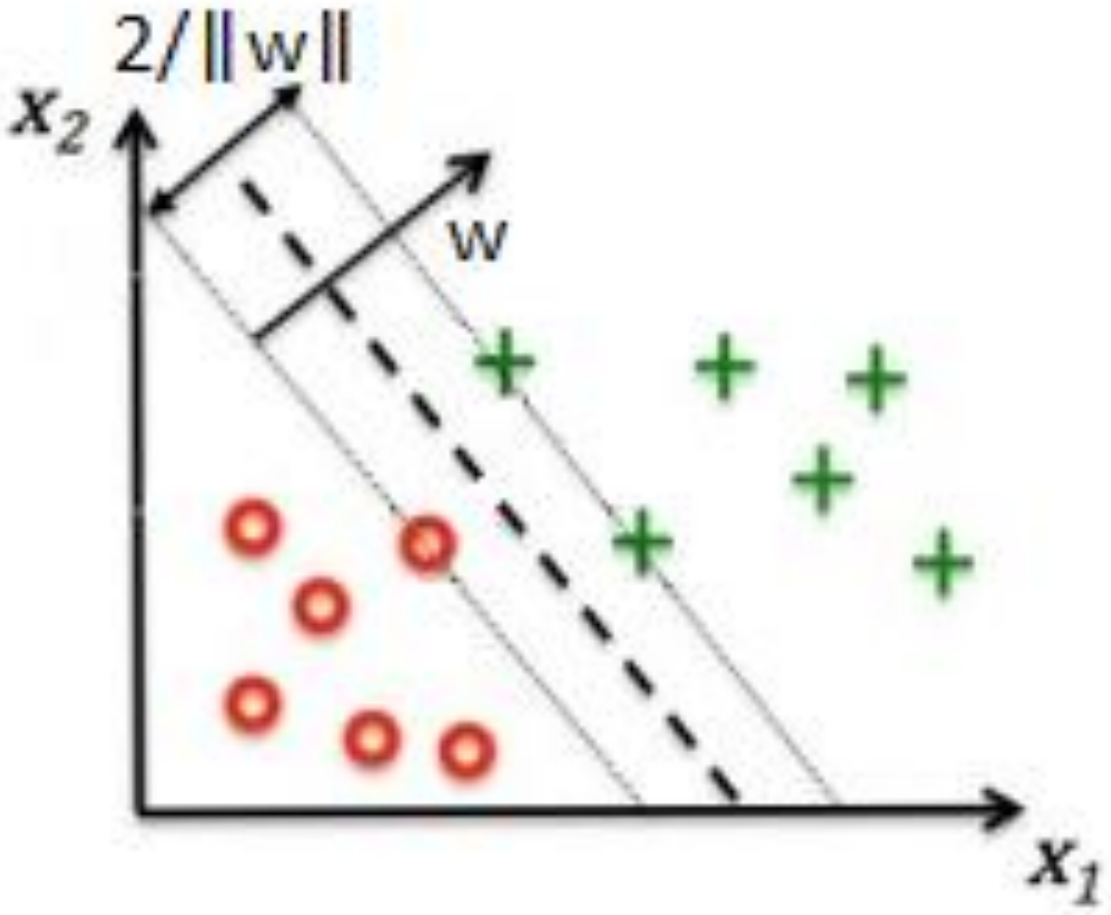
Тогда расстояние от точки x_0 до разделяющей гиперплоскости, задаваемой классификатором:

$$\rho(x_0, a) = \frac{|(w, x_0) + w_0|}{||w||}$$

- Расстояние до ближайшего объекта $x \in X$:

$$\min_{x \in X} \frac{|(w, x) + w_0|}{||w||} = \frac{1}{||w||} \min_{x \in X} |(w, x) + w_0| = \frac{\mathbf{1}}{\mathbf{||w||}}$$

РАЗДЕЛЯЮЩАЯ ПОЛОСА



ОПТИМИЗАЦИОННАЯ ЗАДАЧА SVM ДЛЯ РАЗДЕЛИМОЙ ВЫБОРКИ

$$\begin{cases} \frac{1}{2}\left|\left|w\right|\right|^2 \to \min\limits_{w} \\ y_i\left((w, x_i) + w_0\right) \geq 1, i = 1, \dots, l \end{cases}$$

Утверждение. Данная оптимизационная задача имеет единственное решение.

ЛИНЕЙНО НЕРАЗДЕЛИМАЯ ВЫБОРКА

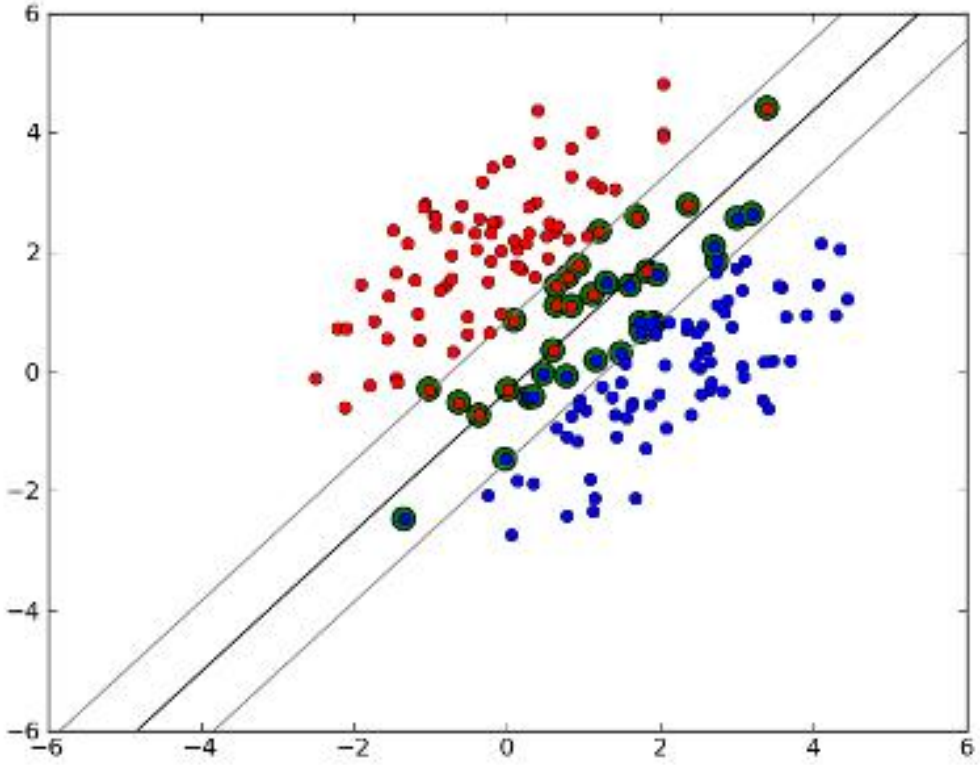
- Существует хотя бы один объект $x \in X$, что

$$y_i\big((w, x_i) + w_0\big) < 1$$

ЛИНЕЙНО НЕРАЗДЕЛИМАЯ ВЫБОРКА

- Существует хотя бы один объект $x \in X$, что

$$y_i\big((w, x_i) + w_0\big) < 1$$



ЛИНЕЙНО НЕРАЗДЕЛИМАЯ ВЫБОРКА

- Существует хотя бы один объект $x \in X$, что

$$y_i\big((w, x_i) + w_0\big) < 1$$

Смягчим ограничения, введя штрафы $\xi_i \geq 0$:

$$y_i\big((w, x_i) + w_0\big) \geq 1 - \xi_i, i = 1, \dots, l$$

МЕТОД ОПОРНЫХ ВЕКТОРОВ: НЕРАЗДЕЛИМЫЙ СЛУЧАЙ

Хотим:

- Минимизировать штрафы $\sum_{i=1}^{l} \xi_i$
- Максимизировать отступ $\frac{1}{||w||}$

МЕТОД ОПОРНЫХ ВЕКТОРОВ: НЕРАЗДЕЛИМЫЙ СЛУЧАЙ

Хотим:

- Минимизировать штрафы $\sum_{i=1}^{l} \xi_i$
- Максимизировать отступ $\frac{1}{||w||}$

Задача оптимизации:

$$\begin{cases} \frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{l} \xi_i \to \min\limits_{w, w_0, \xi_i} \\ y_i\left((w, x_i) + w_0\right) \geq 1 - \xi_i, i = 1, \dots, l \\ \xi_i \geq 0, i = 1, \dots, l \end{cases}$$

МЕТОД ОПОРНЫХ ВЕКТОРОВ: НЕРАЗДЕЛИМЫЙ СЛУЧАЙ

Утверждение. Задача

$$\begin{cases} \frac{1}{2}||w||^2 + C\sum_{i=1}^{l}\xi_i \to \min\limits_{w,w_0,\xi_i} \\ y_i\big((w,x_i)+w_0\big) \geq 1-\xi_i, i=1,\dots,l \\ \xi_i \geq 0, i=1,\dots,l \end{cases}$$

Является выпуклой и имеет единственное решение.

СВЕДЕНИЕ К БЕЗУСЛОВНОЙ ЗАДАЧЕ

$$\begin{cases} \frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{l}\xi_i \to \min\limits_{w,w_0,\xi_i} (1) \\ y_i\left((w,x_i)+w_0\right) \geq 1-\xi_i, i=1,\dots,l\ (2) \\ \xi_i \geq 0, i=1,\dots,l\ (3) \end{cases}$$

- Перепишем (2) и (3):

$$\begin{cases} \xi_i \geq 1 - y_i\left((w,x_i)+w_0\right) = 1 - M_i \\ \xi_i \geq 0 \end{cases}$$

СВЕДЕНИЕ К БЕЗУСЛОВНОЙ ЗАДАЧЕ

$$\begin{cases} \frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{l}\xi_i \to \min\limits_{w,w_0,\xi_i} (1) \\ y_i\big((w,x_i)+w_0\big) \geq 1-\xi_i, i=1,\dots,l\ (2) \\ \xi_i \geq 0, i=1,\dots,l\ (3) \end{cases}$$

- Перепишем (2) и (3):

$$\begin{cases} \xi_i \geq 1 - y_i\big((w,x_i)+w_0\big) \\ \xi_i \geq 0 \end{cases} \Rightarrow \xi_i = \max(0, 1 - y_i\big((w,x_i)+w_0\big))$$

СВЕДЕНИЕ К БЕЗУСЛОВНОЙ ЗАДАЧЕ

$$\begin{cases} \frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{l}\xi_i \to \min\limits_{w,w_0,\xi_i} (1) \\ y_i\big((w,x_i)+w_0\big) \geq 1-\xi_i, i=1,\dots,l\ (2) \\ \xi_i \geq 0, i=1,\dots,l\ (3) \end{cases}$$

- Перепишем (2) и (3):

$$\begin{cases} \xi_i \geq 1 - y_i\big((w,x_i)+w_0\big) \\ \xi_i \geq 0 \end{cases} \Rightarrow \xi_i = \max(0, 1 - y_i\big((w,x_i)+w_0\big))$$

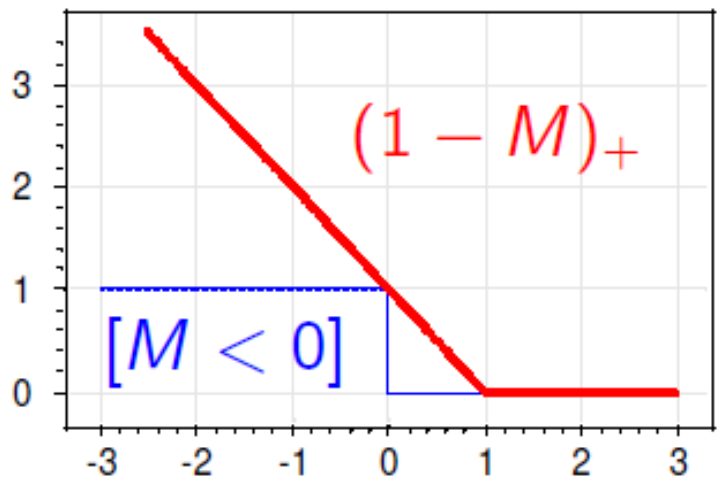
Получаем безусловную задачу оптимизации:

$$\boldsymbol{\frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{l}\max\Big(0, 1 - y_i\big((w,x_i)+w_0\big)\Big) \to \min\limits_{w,w_0}}$$

МЕТОД ОПОРНЫХ ВЕКТОРОВ: ЗАДАЧА ОПТИМИЗАЦИИ

- На задачу оптимизации SVM можно смотреть, как на *оптимизацию функции потерь* $L(M) = max(0,1 - M) = (1 - M)_+$ *с регуляризацией*:

$$Q(a, X) = \sum_{i=1}^{l} \left(1 - M_i(w, w_0)\right)_+ + \frac{1}{2C} ||w||^2 \to \min_{w, w_0}$$

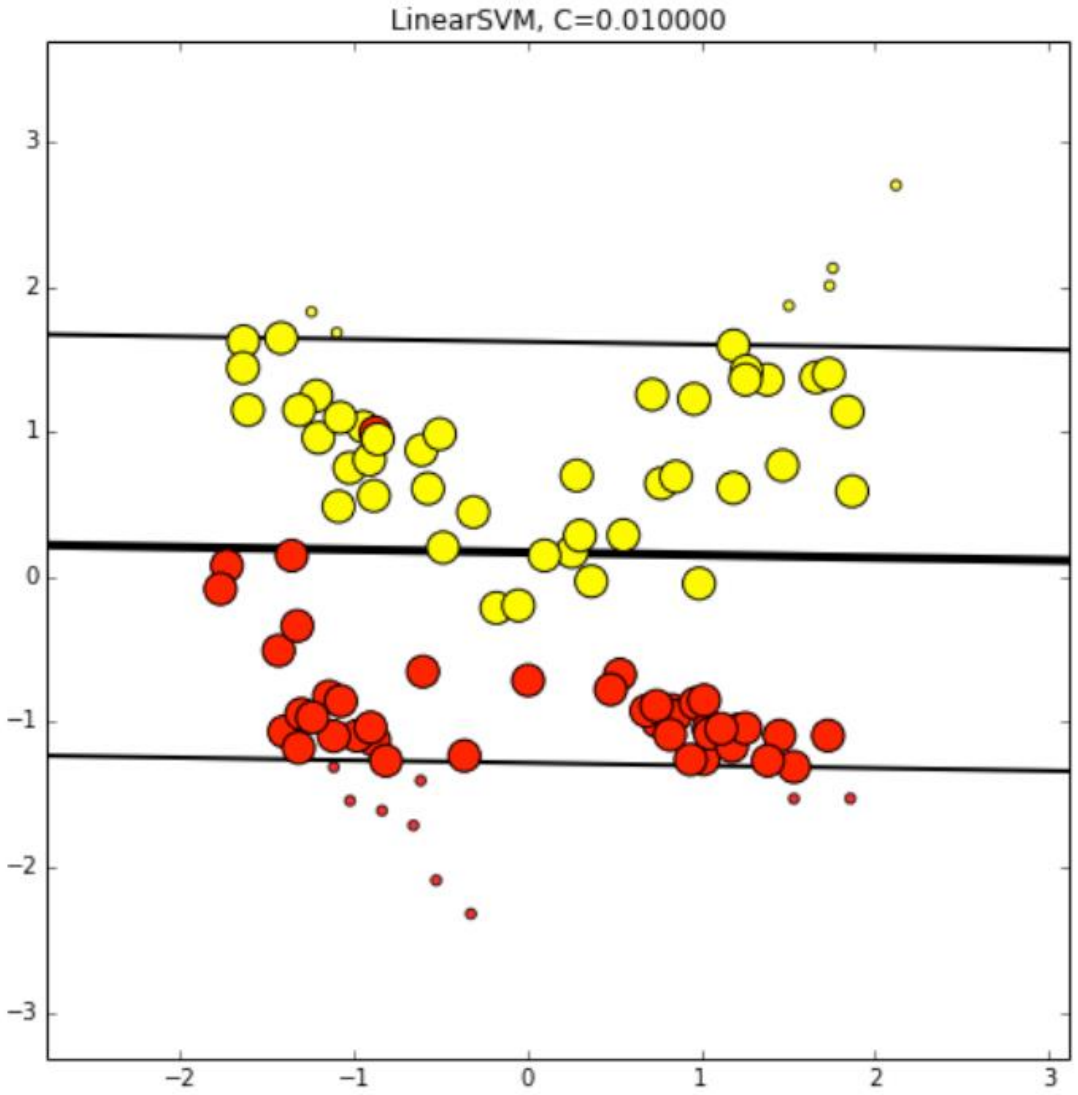


ЗНАЧЕНИЕ КОНСТАНТЫ C

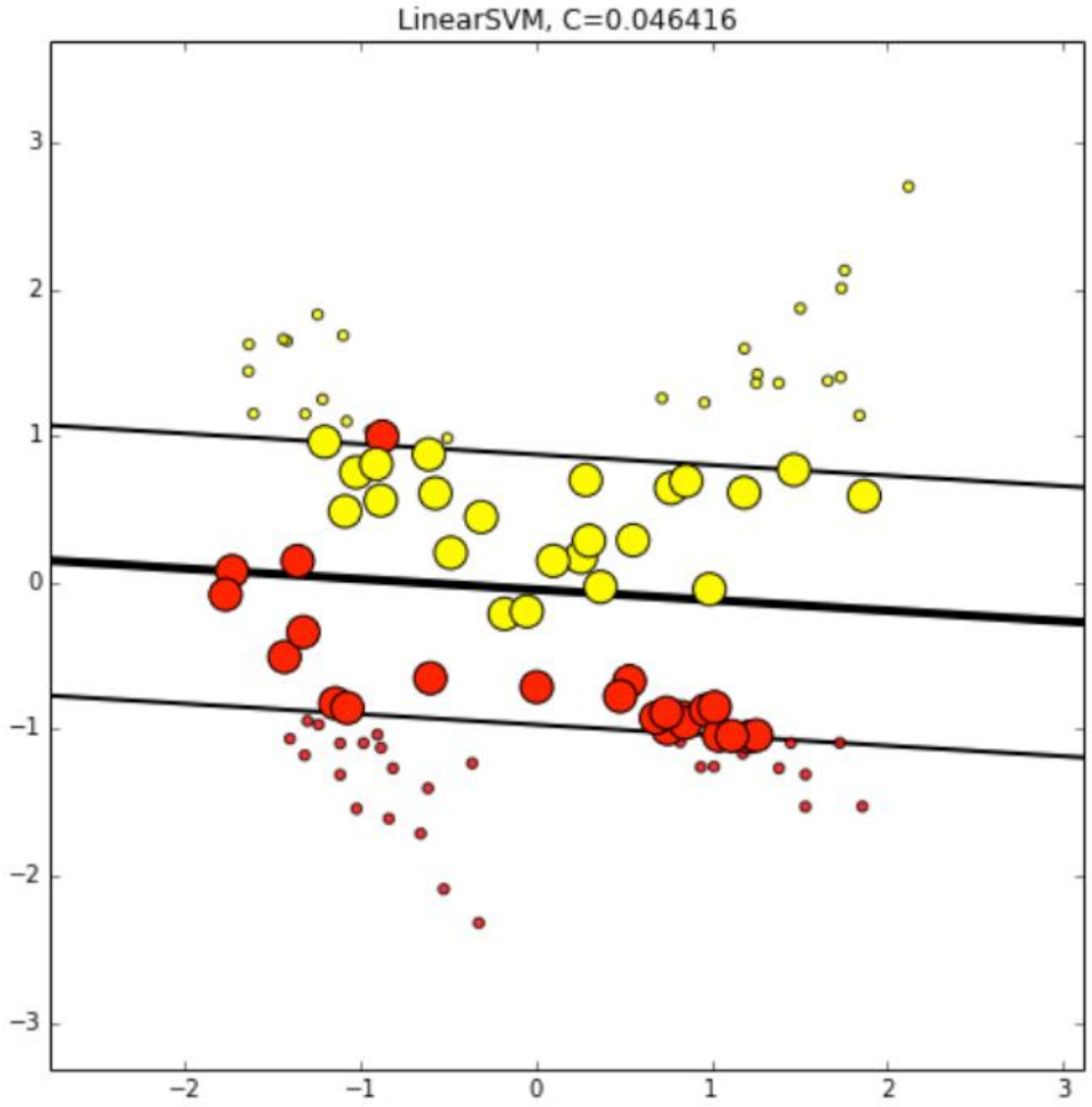
$$\begin{cases} \frac{1}{2}\left|\left|w\right|\right|^2 + \boldsymbol{C}\sum_{i=1}^{l}\xi_i \to \min\limits_{w,w_0,\xi_i} (1) \\ y_i\left((w, x_i) + w_0\right) \geq 1 - \xi_i, i = 1, \dots, l\ (2) \\ \xi_i \geq 0, i = 1, \dots, l\ (3) \end{cases}$$

Положительная константа C является управляющим параметром метода и позволяет находить компромисс между максимизацией разделяющей полосы и минимизацией суммарной ошибки.

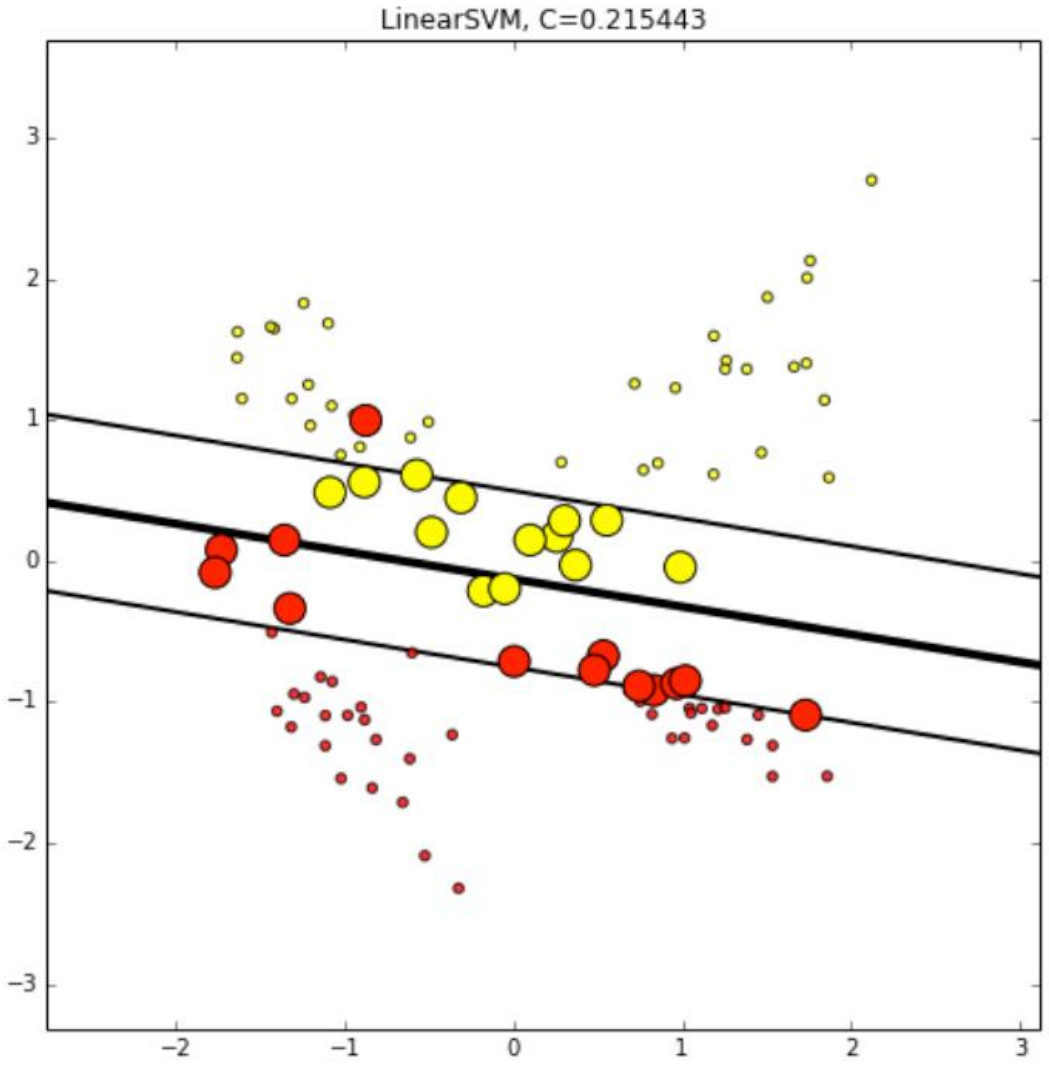
ЗНАЧЕНИЕ КОНСТАНТЫ C



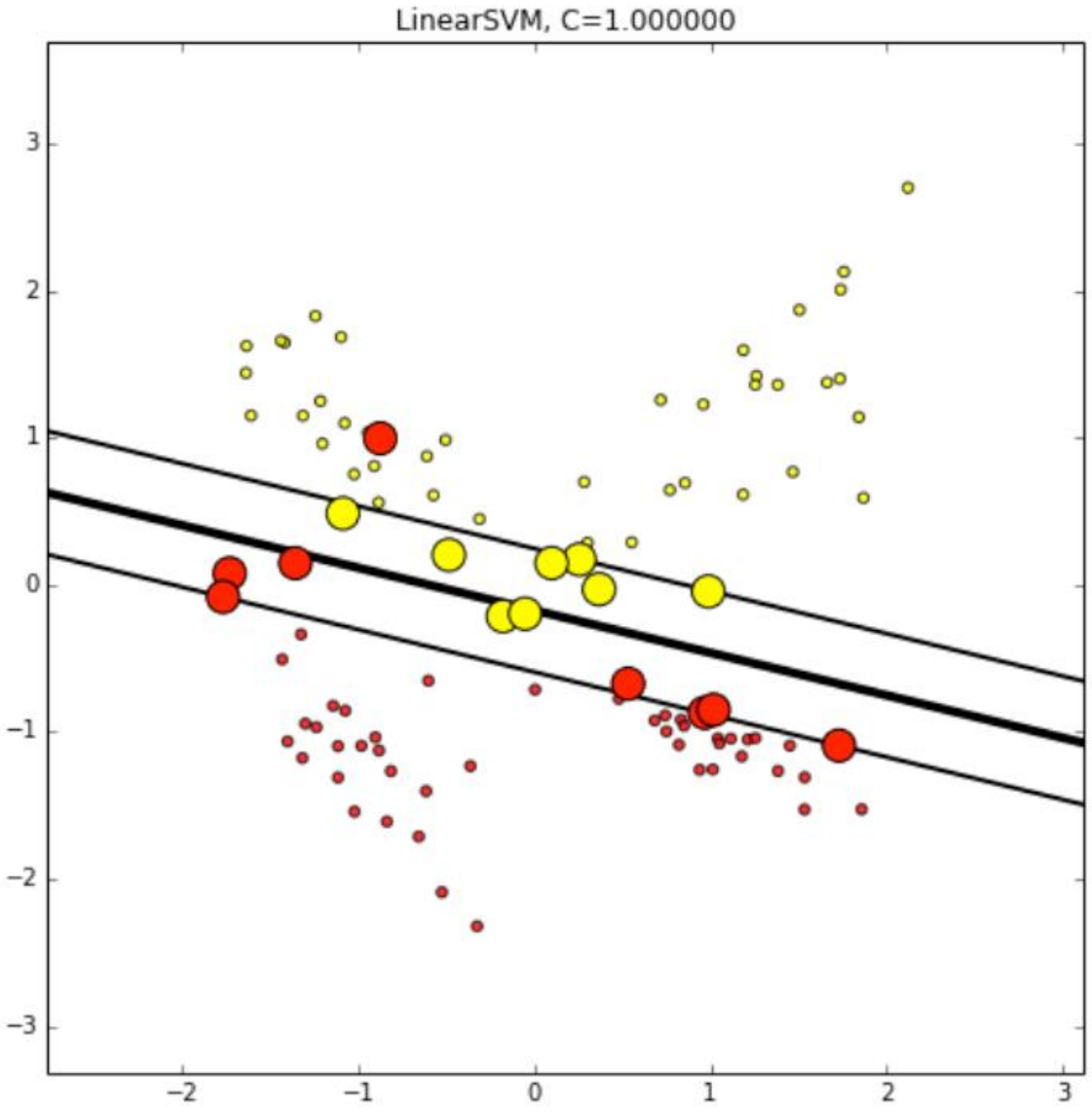
ЗНАЧЕНИЕ КОНСТАНТЫ C



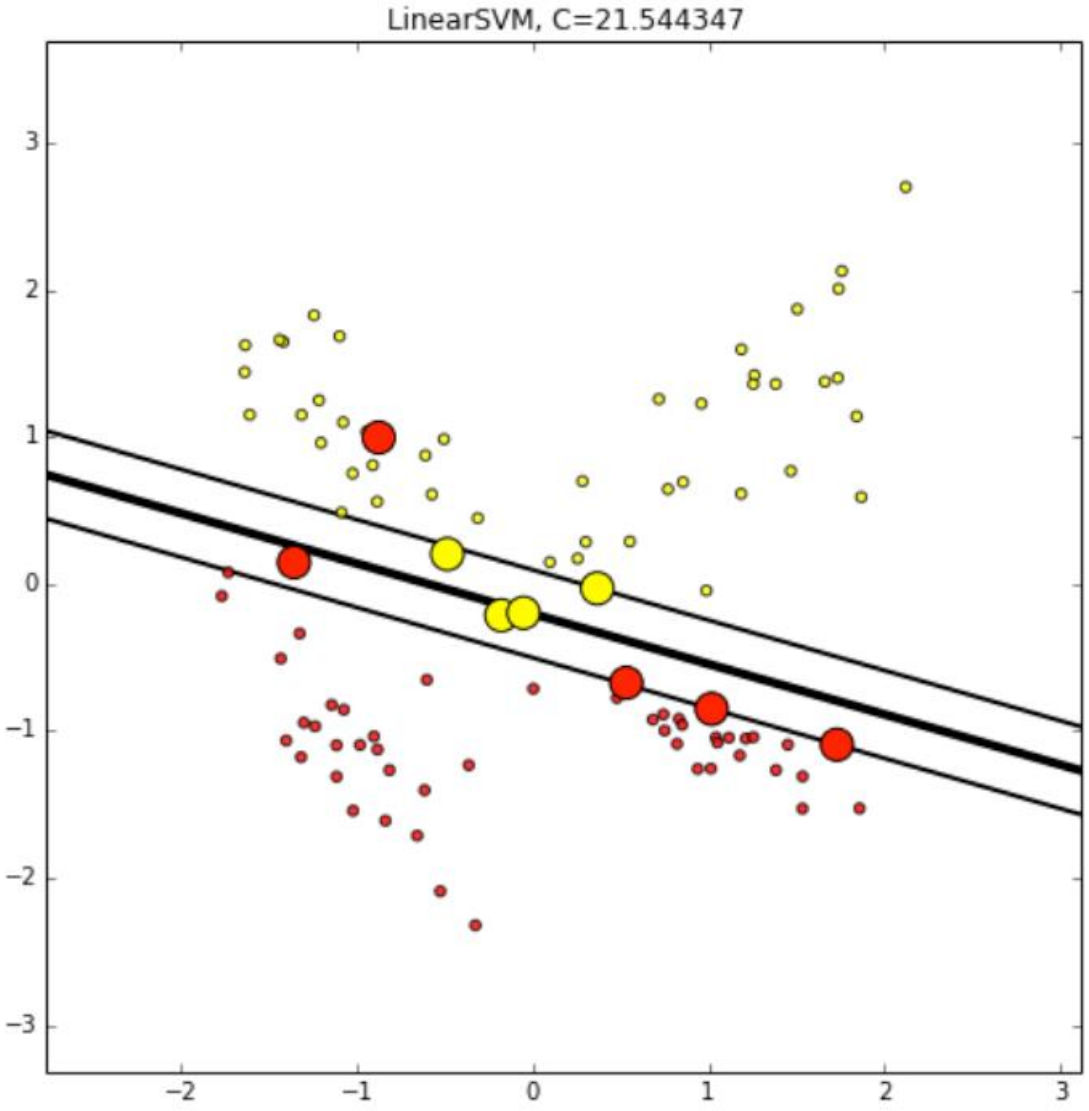
ЗНАЧЕНИЕ КОНСТАНТЫ C



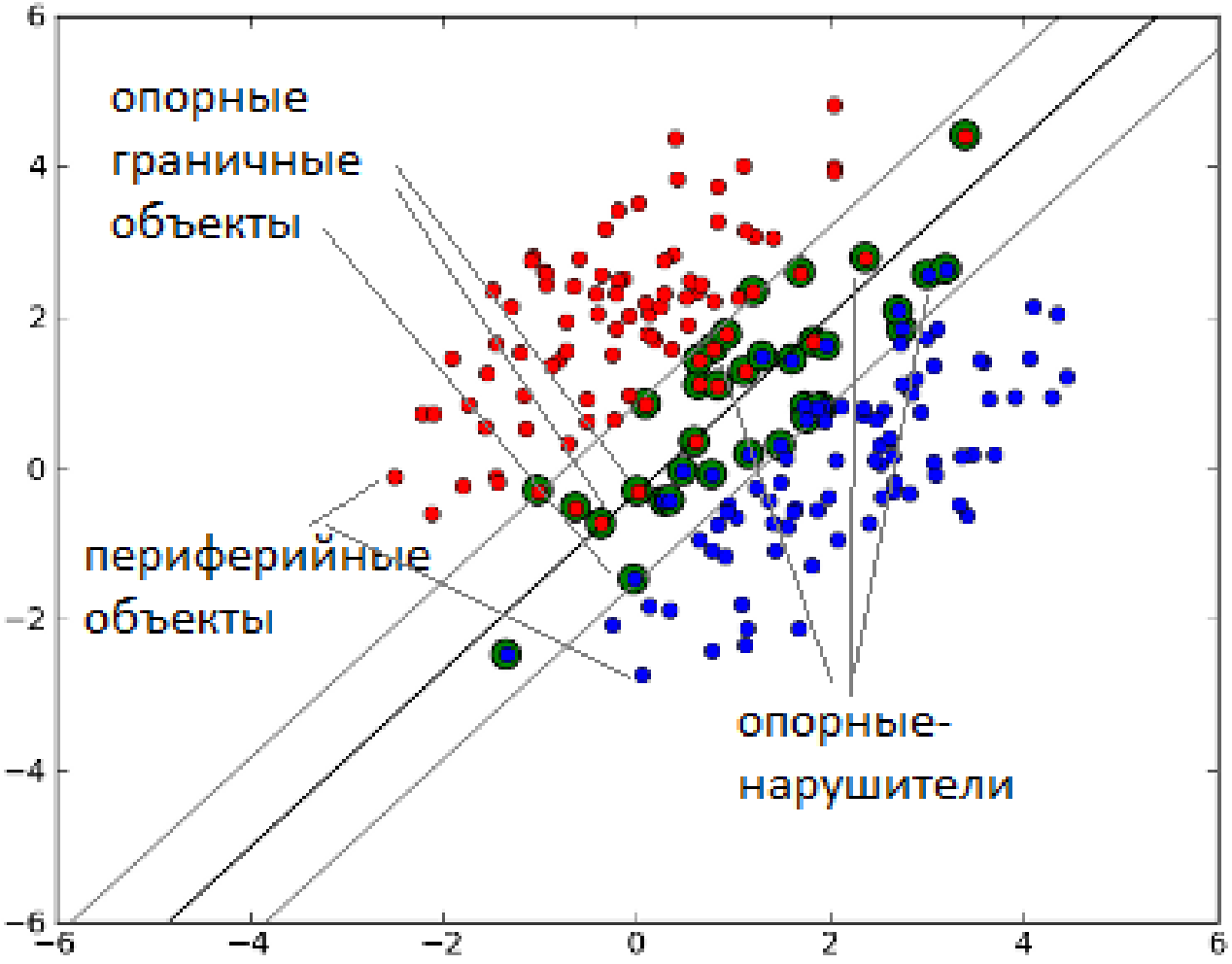
ЗНАЧЕНИЕ КОНСТАНТЫ C



ЗНАЧЕНИЕ КОНСТАНТЫ C

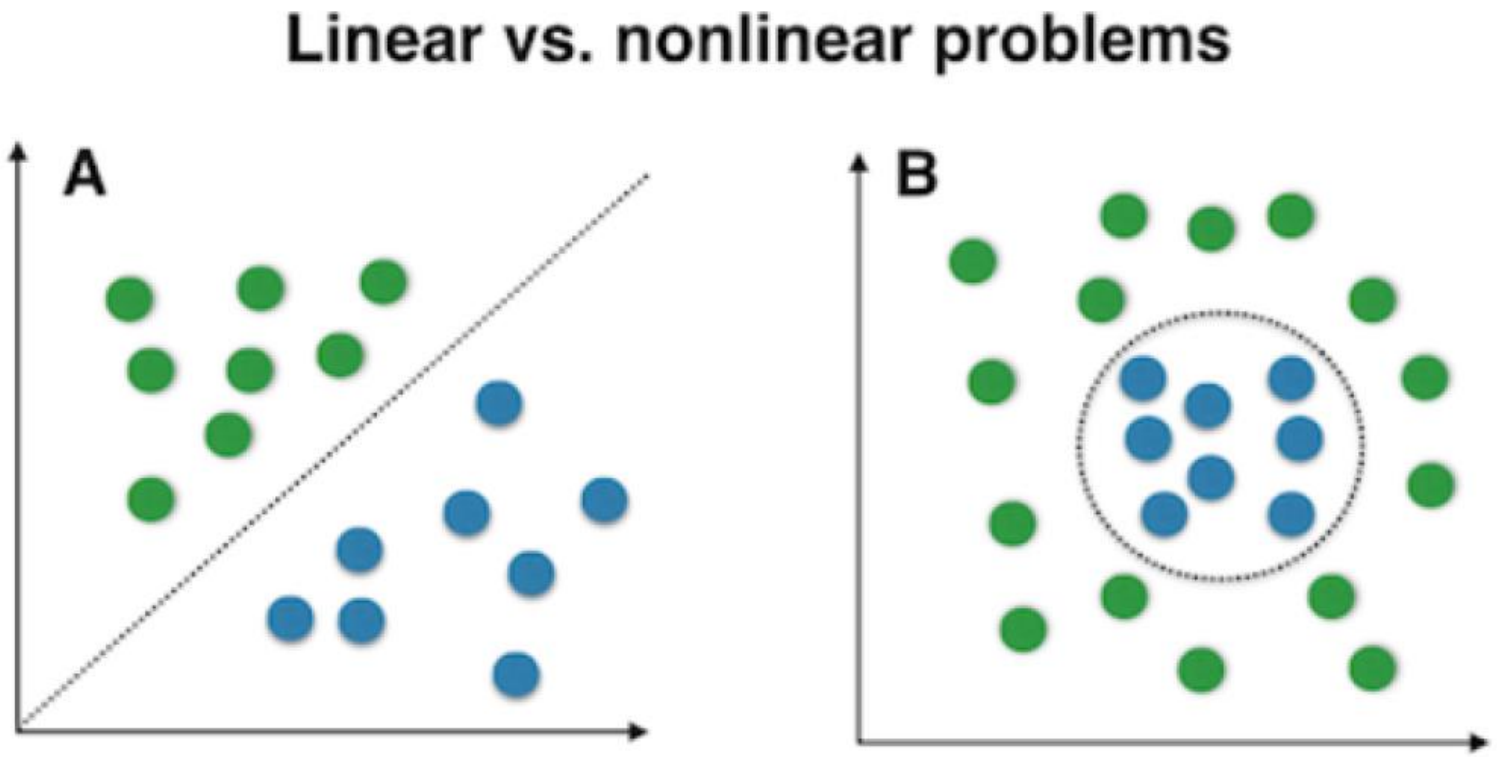


ТИПЫ ОБЪЕКТОВ В SVM

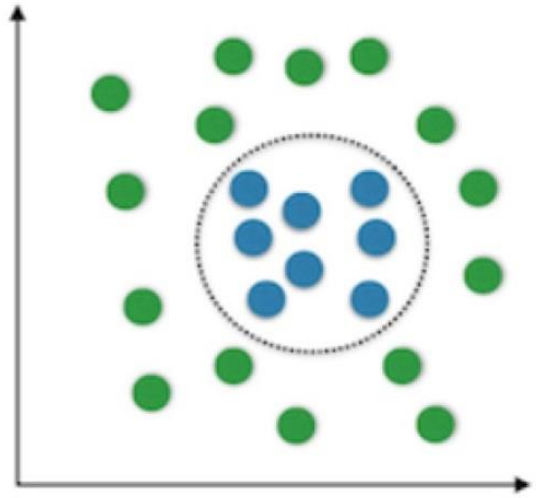


ЯДРОВЫЕ МЕТОДЫ

НЕЛИНЕЙНЫЕ ЗАДАЧИ КЛАССИФИКАЦИИ

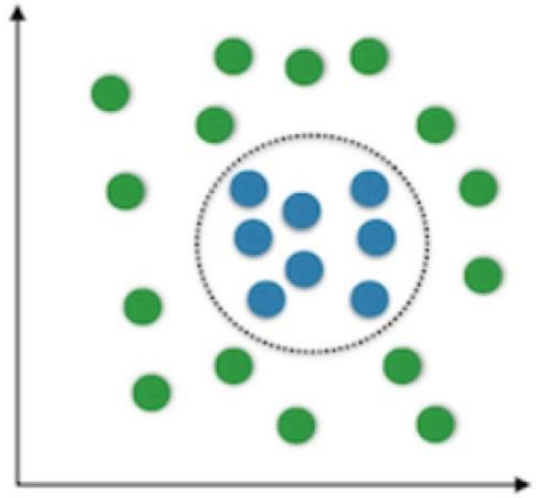


НЕЛИНЕЙНЫЕ ЗАДАЧИ КЛАССИФИКАЦИИ



$$(x_1, x_2) \rightarrow (x_1, x_2, x_3 = x_1^2 + x_2^2)$$

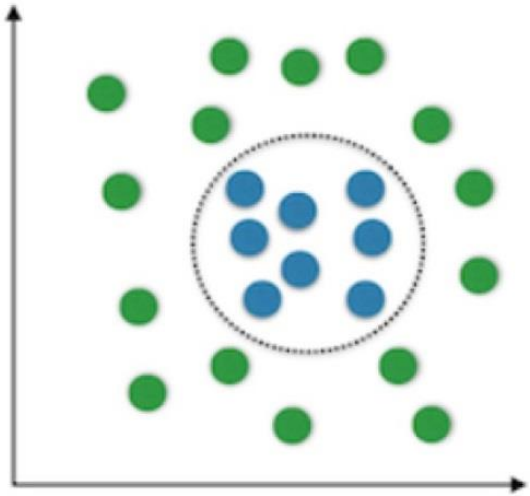
НЕЛИНЕЙНЫЕ ЗАДАЧИ КЛАССИФИКАЦИИ



$$(x_1, x_2) \rightarrow (x_1, x_2, x_3 = x_1^2 + x_2^2)$$

- В новом пространстве признаков выборка идеально разделяется гиперплоскостью.

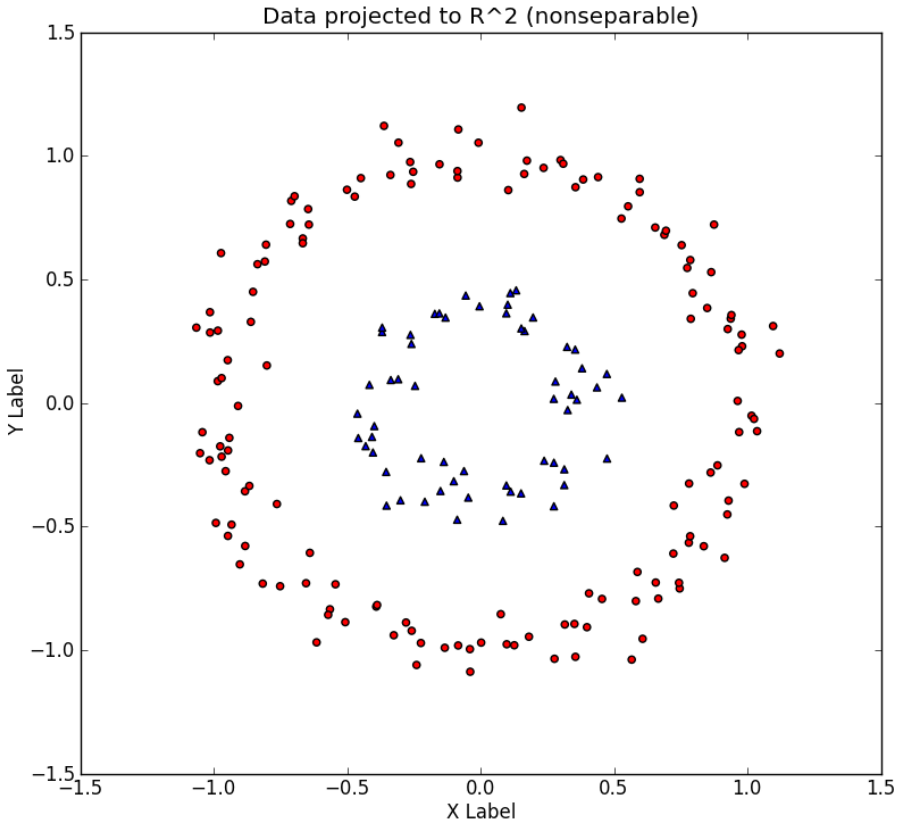
НЕЛИНЕЙНЫЕ ЗАДАЧИ КЛАССИФИКАЦИИ

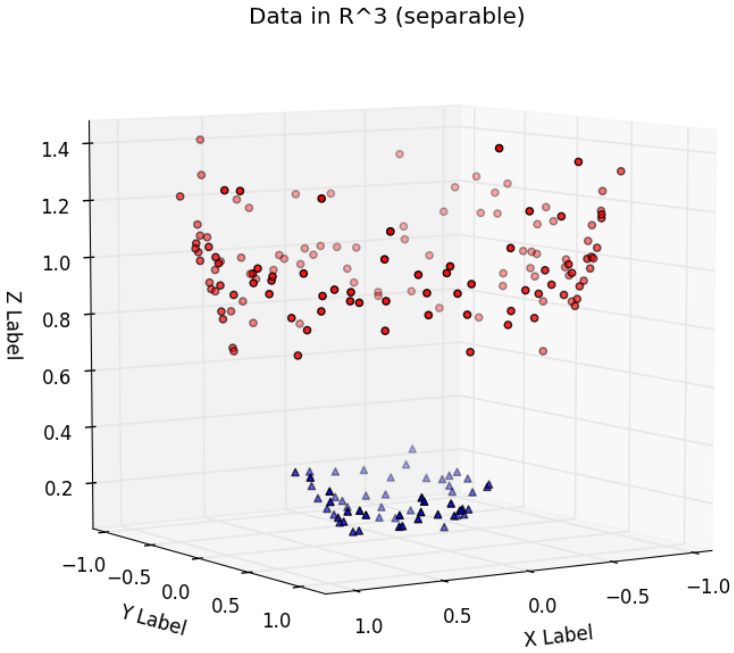


$$(x_1, x_2) \to (x_1, x_2, x_3 = x_1^2 + x_2^2)$$

- В новом пространстве признаков выборка идеально разделяется гиперплоскостью.
- ***В новом признаковом пространстве*** классификатор имеет вид $a(x) = sign(w, x) = sign(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3)$, то есть ***разделяющая поверхность*** $\boldsymbol{w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 = 0}$ ***– линейная.***

ПЕРЕХОД К НОВЫМ ПРИЗНАКАМ





ПЕРЕХОД К НОВЫМ ПРИЗНАКАМ

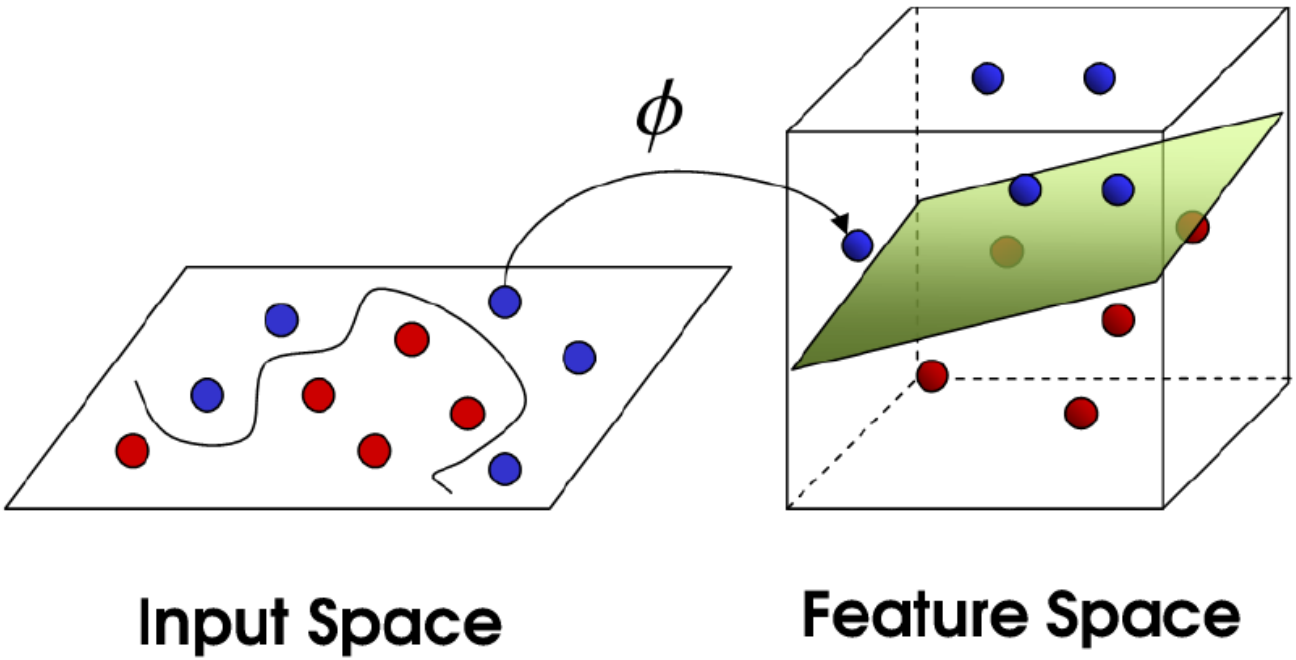
Переход к новым признакам

$$x = (x_1, x_2, \dots, x_d) \to \varphi_1(x), \varphi_2(x), \dots, \varphi_m(x)$$

Позволяет построить классификатор:

- *являющийся нелинейным в исходном пространстве признаков* $x_1, x_2, \dots, x_d$ (и строящий нелинейную разделяющую поверхность)
- *являющийся линейным в новом пространстве признаков* $\varphi_1(x), \varphi_2(x), \dots, \varphi_m(x)$ – *спрямляющем пространстве*

ПЕРЕХОД К НОВЫМ ПРИЗНАКАМ



ПЕРЕХОД К НОВЫМ ПРИЗНАКАМ

Линейный классификатор:

$$a(x) = sign((w, x) + w_0)$$

Классификатор в новом признаковом пространстве

$$a(x) = sign((w, \varphi(x)) + w_0)$$

ПРОБЛЕМА

При переходе к новым признакам

$$x = (x_1, \dots, x_n) \to \varphi_1(x), \varphi_2(x), \dots \varphi_N(x)$$

новых признаков может потребоваться довольно много, чтобы линейно разделить выборку в новом пространстве.

Поэтому сильно возрастает вычислительная сложность алгоритма, а также объем памяти, нужный для хранения всех данных.

ЯДРОВОЙ ТРЮК (KERNEL TRICK)

При переходе к новым признакам

$$x = (x_1, \dots, x_n) \to \varphi_1(x), \varphi_2(x), \dots \varphi_N(x)$$

новых признаков может потребоваться довольно много, чтобы линейно разделить выборку в новом пространстве.

Поэтому сильно возрастает вычислительная сложность алгоритма, а также объем памяти, нужный для хранения всех данных.

- Существует подход к решению этой проблемы под названием ***kernel trick (ядровой трюк)***. Ядровой трюк позволяет перейти в спрямляющее пространство без увеличения вычислительной сложности и требуемой памяти.

ЯДРО

Ядро – это функция $K(x, z)$, представимая в виде скалярного произведения $\boldsymbol{K(x, z) = \big(\varphi(x), \varphi(z)\big)}$, где $\varphi: X \to H$ – отображение из исходного признакового пространства X в некоторое спрямляющее пространство H.

ЯДРО

Ядро – это функция $K(x,z)$, представимая в виде скалярного произведения $K(x,z) = \big(\varphi(x), \varphi(z)\big)$, где $\varphi: X \to H$ – отображение из исходного признакового пространства X в некоторое спрямляющее пространство H.

Теорема (Мерсер). *Функция $K(x,z)$ является ядром тогда и только тогда, когда:*

1) $K(x,z) = K(z,x)$

2) Для любой конечной выборки $(x_1, \dots, x_l)$ *матрица* $K = \Big(K\big(x_i, x_j\big)\Big)_{i,j=1}^{l}$ *неотрицательно определена*

ЯДРО

Ядро – это функция $K(x,z)$, представимая в виде скалярного произведения $K(x,z) = \big(\varphi(x), \varphi(z)\big)$, где $\varphi: X \to H$ – отображение из исходного признакового пространства X в некоторое спрямляющее пространство H.

Теорема (Мерсер). *Функция $K(x,z)$ является ядром тогда и только тогда, когда:*

1) $K(x,z) = K(z,x)$

2) Для любой конечной выборки $(x_1, \dots, x_l)$ *матрица* $K = \Big(K\big(x_i, x_j\big)\Big)_{i,j=1}^{l}$ *неотрицательно определена*

Из теоремы Мерсера следует, что ядро $K(x,z)$ задаёт скалярное произведение объектов x и z.

KERNEL TRICK

- Идея ядрового трюка состоит в том, что некоторые модели машинного обучения (в частности, линейную регрессию и SVM) можно записать в таком виде, чтобы и модель, и функционал ошибки зависели ***только от скалярных произведений объектов (а не от самих объектов)***.

KERNEL TRICK

- Идея ядрового трюка состоит в том, что некоторые модели машинного обучения (в частности, линейную регрессию и SVM) можно записать в таком виде, чтобы и модель, и функционал ошибки зависели ***только от скалярных произведений объектов (а не от самих объектов)***.

Пример – SVM:

- Исходная модель: $a(x, w) = sign((w, x) + w_0)$
- Модель можно записать в виде (двойственная запись):

$$a(x, \lambda) = sign(\sum_{i=1}^{l} \lambda_i y_i (x_i, x) - w_0)$$

KERNEL TRICK

- Идея ядрового трюка состоит в том, что некоторые модели машинного обучения (в частности, линейную регрессию и SVM) можно записать в таком виде, чтобы и модель, и функционал ошибки зависели ***только от скалярных произведений объектов (а не от самих объектов)***.

Пример – SVM: *переходим к новым признакам* $\varphi(x)$

- Исходная модель: $a(x,w) = sign((w, \varphi(x)) + w_0)$
- Модель можно записать в виде (двойственная запись):

$$a(x,\lambda) = sign(\sum_{i=1}^{l} \lambda_i y_i K(x_i, x) - w_0)$$

KERNEL TRICK

- Идея ядрового трюка состоит в том, что некоторые модели машинного обучения (в частности, линейную регрессию и SVM) можно записать в таком виде, чтобы и модель, и функционал ошибки зависели ***только от скалярных произведений объектов (а не от самих объектов)***.

Пример – SVM: *переходим к новым признакам* $\varphi(x)$

- Исходная модель: $a(x,w) = sign((w, \varphi(x)) + w_0)$
- Модель можно записать в виде (двойственная запись):

$$a(x,\lambda) = sign(\sum_{i=1}^{l} \lambda_i y_i K(x_i, x) - w_0)$$

Также можно записать функционал ошибки только через скалярное произведение объектов

KERNEL TRICK

- Идея ядрового трюка состоит в том, что некоторые модели машинного обучения (в частности, линейную регрессию и SVM) можно записать в таком виде, чтобы и модель, и функционал ошибки зависели ***только от скалярных произведений объектов (а не от самих объектов)***.

Пример – SVM: *переходим к новым признакам* $\varphi(x)$

- Исходная модель: $a(x, w) = sign((w, \varphi(x)) + w_0)$
- Модель можно записать в виде (двойственная запись):

$$a(x, \lambda) = sign(\sum_{i=1}^{l} \lambda_i y_i K(x_i, x) - w_0)$$

➢ То есть для вычисления предсказания не нужно вычислять значения новых признаков $\varphi(x)$, а достаточно уметь вычислять ядро $K(x_i, x)$.

KERNEL TRICK

- Идея ядрового трюка состоит в том, что некоторые модели машинного обучения (в частности, линейную регрессию и SVM) можно записать в таком виде, чтобы и модель, и функционал ошибки зависели ***только от скалярных произведений объектов (а не от самих объектов)***.

Пример – SVM: *переходим к новым признакам* $\varphi(x)$

- Исходная модель: $a(x, w) = sign((w, \varphi(x)) + w_0)$
- Модель можно записать в виде (двойственная запись):

$$a(x, \lambda) = sign(\sum_{i=1}^{l} \lambda_i y_i K(x_i, x) - w_0)$$

➢ То есть для вычисления предсказания не нужно вычислять значения новых признаков $\varphi(x)$, а достаточно уметь вычислять ядро $K(x_i, x)$.

➢ *Таким образом, мы можем изначально задать только ядро и не задавать (даже не знать!) явный вид преобразования* $\varphi(x)$ *и тем самым решить проблему размерности.*

МЕТОДЫ ПОСТРОЕНИЯ ЯДЕР

Теорема 1. *Пусть* $K_1(x,z)$ *и* $K_2(x,z)$ *- ядра, заданные на множестве* X*. Тогда следующие функции являются ядрами:*

1) $K(x,z) = K_1(x,z) + K_2(x,z)$

2) $K(x,z) = \alpha K_1(x,z), \alpha > 0$

3) $K(x,z) = K_1(x,z)K_2(x,z)$

4) $K(x,z) = f(x)f(z)$, $f(x)$ – *вещественная функция на* X

5) $K(x,z) = K_3\big(\varphi(x), \varphi(z)\big)$, $\varphi: X \to \mathbb{R}^n$ $-$
векторная функция на X, K_3 $-$ ядро, заданное на $\mathbb{R}^n$.

МЕТОДЫ ПОСТРОЕНИЯ ЯДЕР

Теорема 2. Пусть $K_1(x,z), K_2(x,z)$, … - последовательность ядер, причем предел

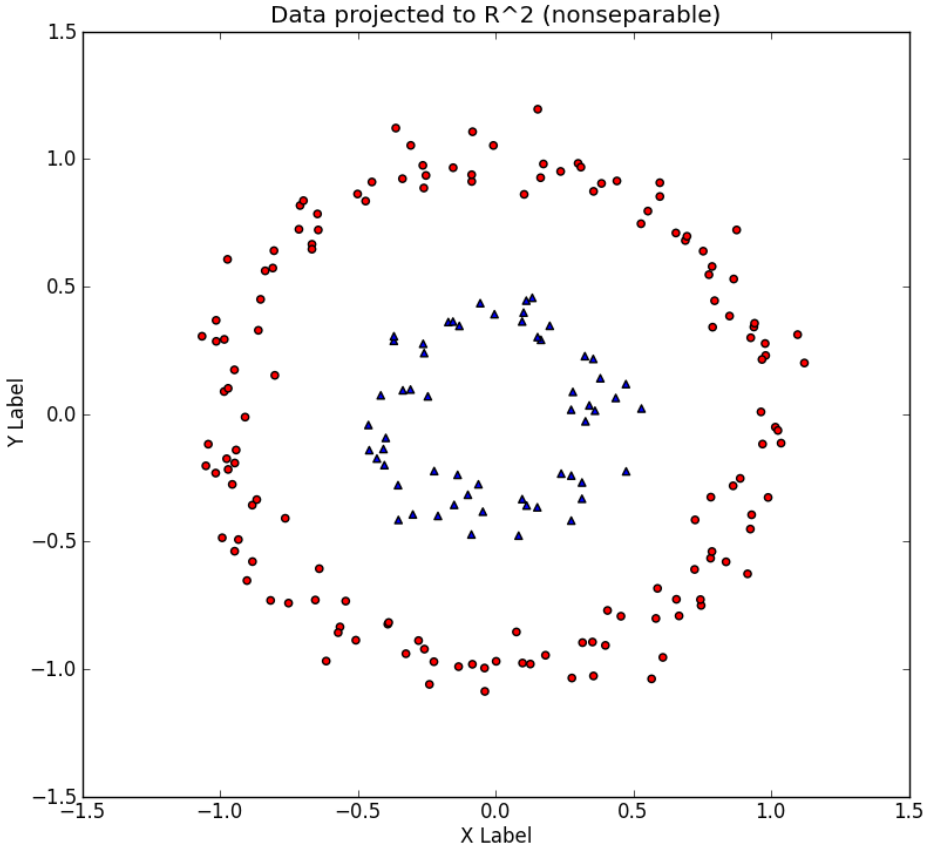
$$K(x,z) = \lim_{n\to\infty} K_n(x,z)$$

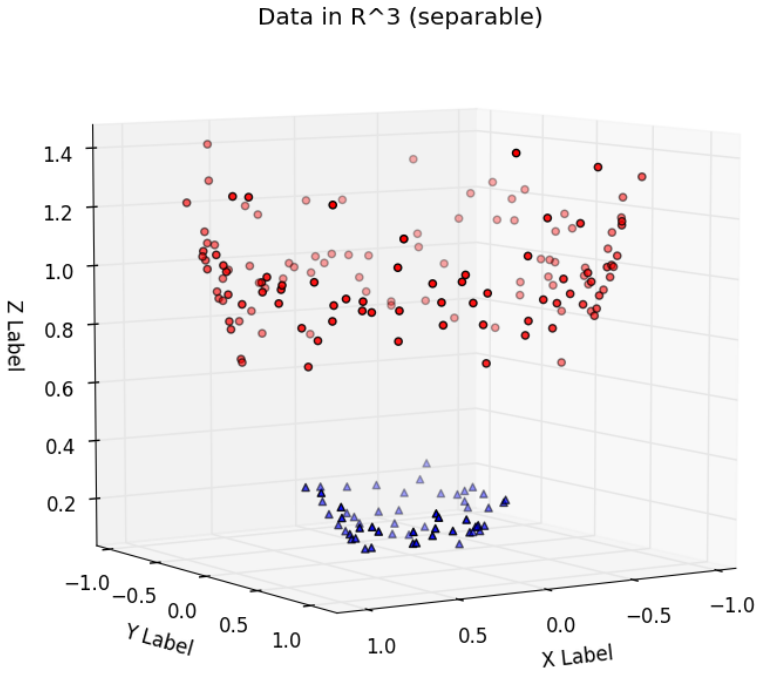
Существует для всех x и z. Тогда $K(x,z)$ ядро.

ПРИМЕРЫ ЯДЕР

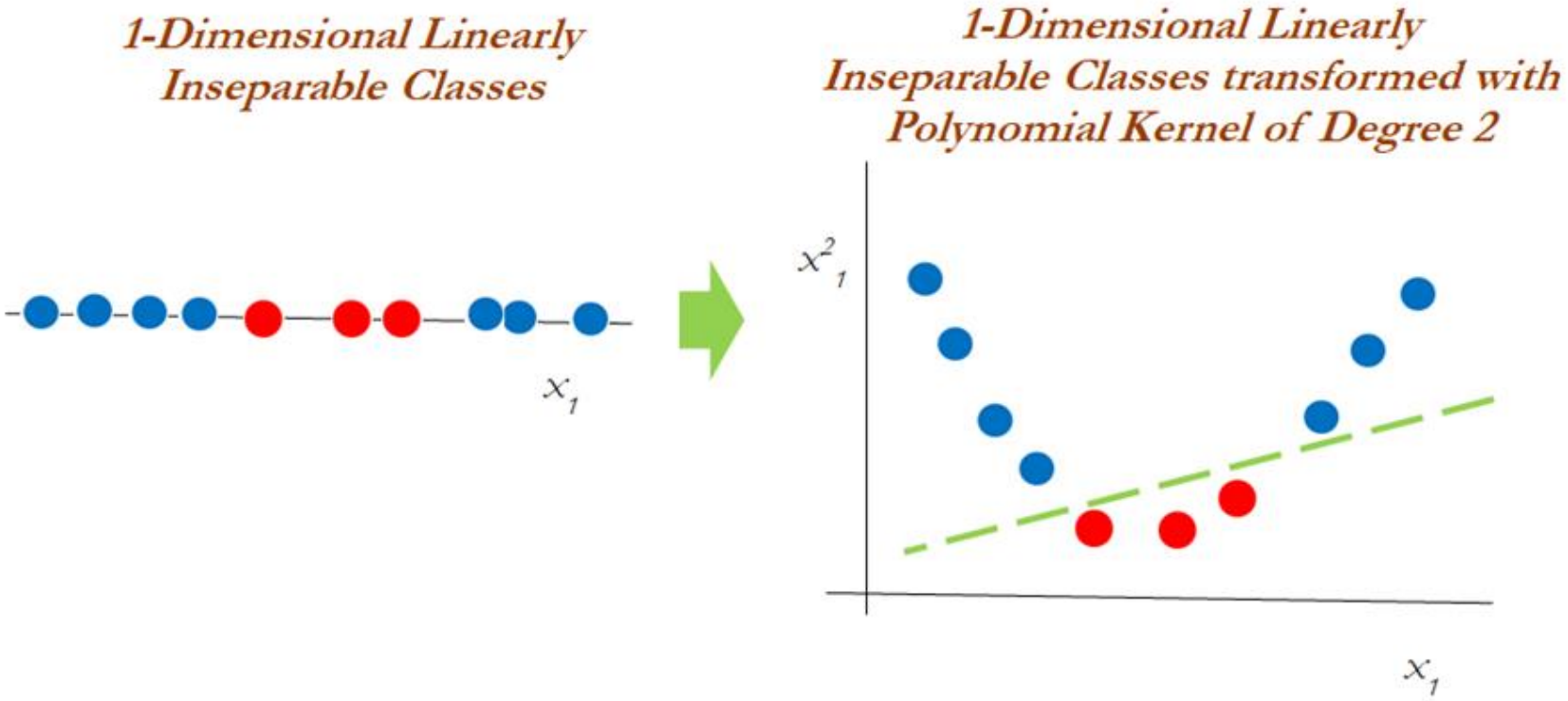
- $K(x,z) = 1$
- $K(x,z) = (x,z)$ – скалярное произведение
- $K(x,z) = (x,z)^2$, где $x = (x_1, x_2), z = (z_1, z_2)$
- $K(x,z) = \exp(-\gamma\left||x-y|\right|^2)$ – гауссовское или радиальное ядро (RBF-ядро).
- $K(x,z) = p\big((x,z)\big)$, где p – многочлен с положительными коэффициентами
- $K(x,z) = \big((x,z) + R\big)^d$, $\mathrm{R} > 0$ – полиномиальное ядро

РАДИАЛЬНОЕ ЯДРО



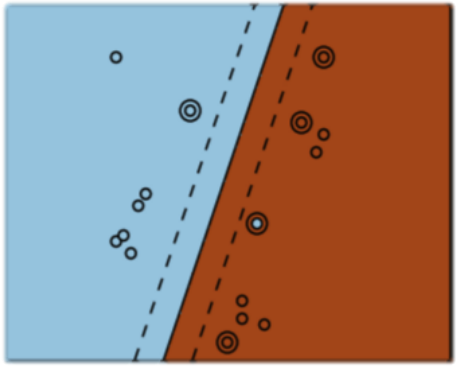


ПОЛИНОМИАЛЬНОЕ ЯДРО



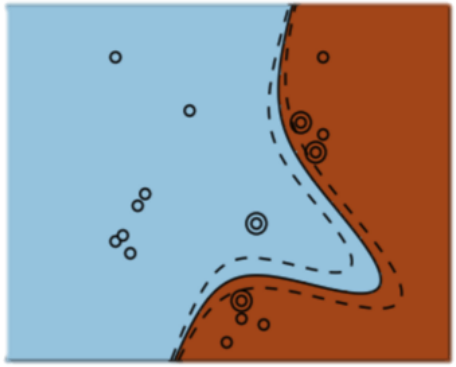
ПРИМЕР: SVM С РАЗЛИЧНЫМИ ЯДРАМИ (ДВА КЛАССА)

Linear Kernel



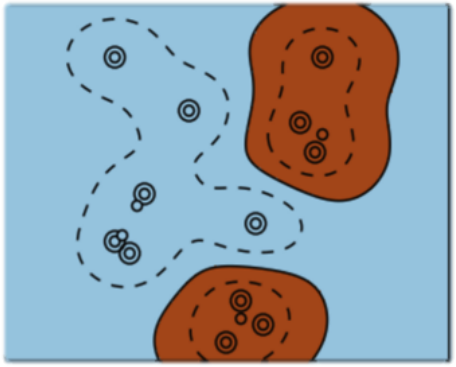
C hyperparameter

Polynomial Kernel



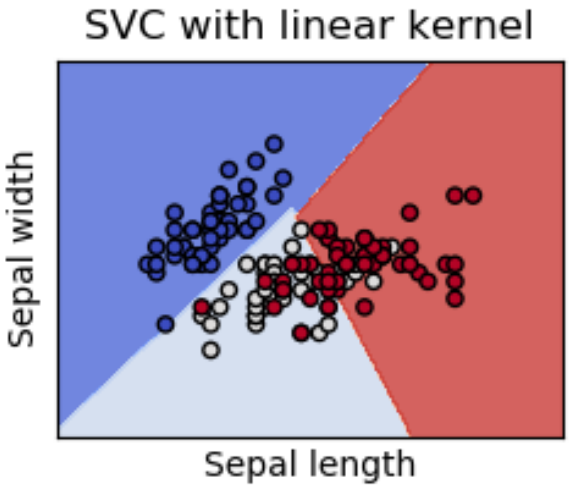
C plus gamma, degree and coefficient hyperparameters

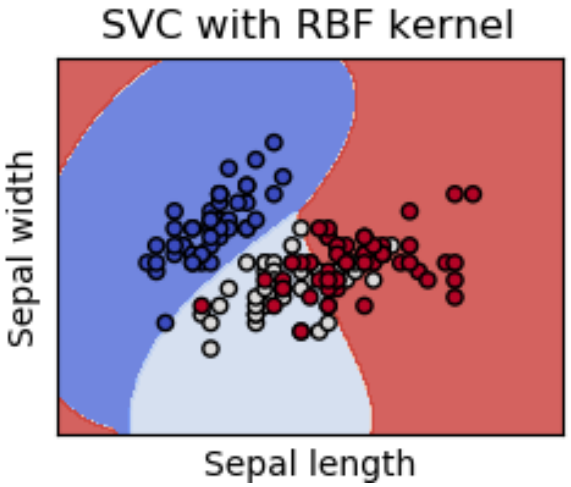
RBF Kernel

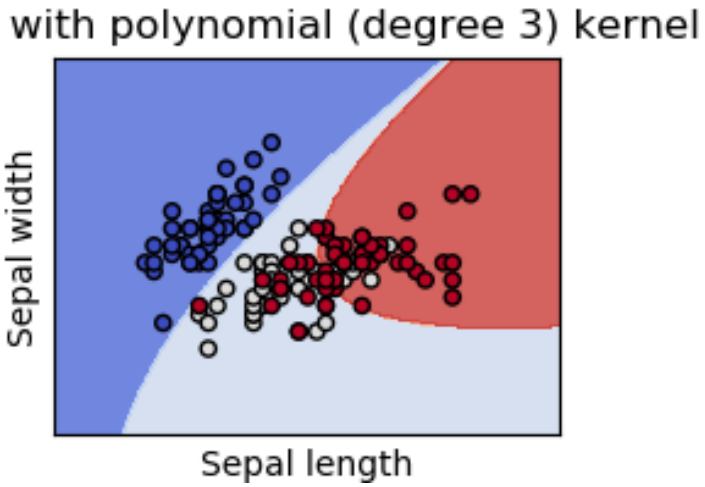


C plus gamma hyperparameter

ПРИМЕР: SVM С РАЗЛИЧНЫМИ ЯДРАМИ (ТРИ КЛАССА)



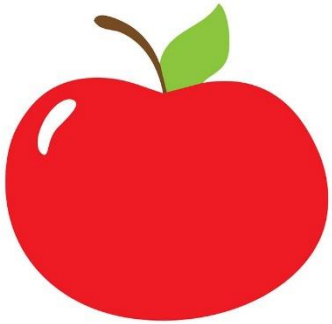




НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР

НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР

Наивный байесовский классификатор – это алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков.



Пример: фрукт может считаться яблоком, если:

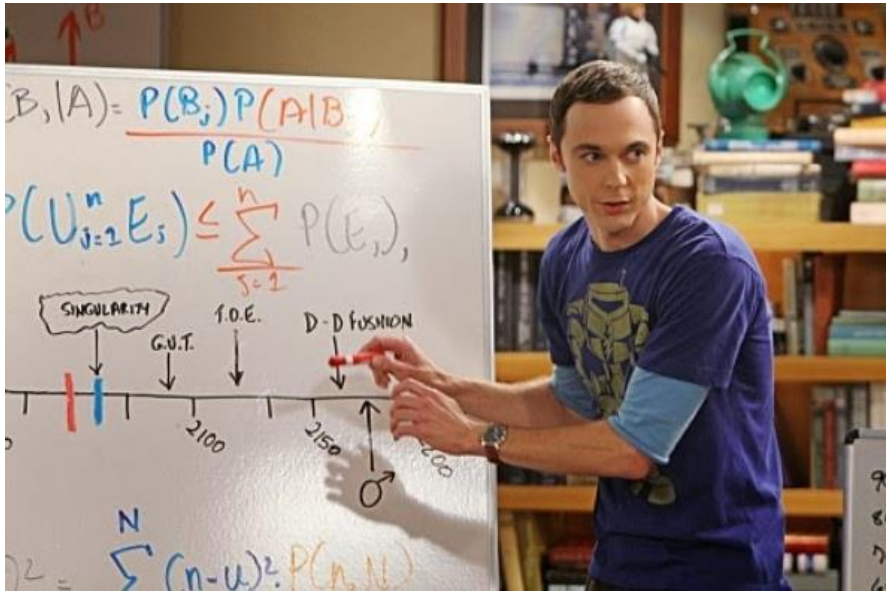
1) он красный
2) круглый
3) его диаметр составляет порядка 8 см

Предполагаем, что признаки вносят независимый вклад в вероятность того, что фрукт является яблоком.

ТЕОРЕМА БАЙЕСА

Теорема Байеса:

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$



- $P(c|x)$ - вероятность того, что объект со значением признака x принадлежит классу c.
- $P(c)$ – априорная вероятность класса c.
- $P(x|c)$ - вероятность того, что значение признака равно x при условии, что объект принадлежит классу c.
- $P(x)$ – априорная вероятность значения признака x.

ПРИМЕР РАБОТЫ БАЙЕСОВСКОГО АЛГОРИТМА

Пример: на основе данных о погодных условиях необходимо определить, состоится ли матч.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

- Преобразуем набор данных в следующую таблицу:

Weather	No	Yes
Overcast	0	4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

ПРИМЕР РАБОТЫ БАЙЕСОВСКОГО АЛГОРИТМА

Решим задачу с помощью теоремы Байеса:

$$P(Yes|Sunny) = P(Sunny|Yes) \cdot P(Yes)/P(Sunny)$$

Таблица частот				
Weather	**No**	**Yes**		
Overcast	0	4	**=4/14**	**0.29**
Rainy	3	2	**=5/14**	**0.36**
Sunny	2	3	**=5/14**	**0.36**
Grand Total	5	9		
	=5/14	**=9/14**		
	0.36	**0.64**		

- $P(Sunny|Yes) = \frac{3}{9}, P(Sunny) = \frac{5}{14}, P(Yes) = \frac{9}{14}.$
- $P(Yes|Sunny) = \frac{3}{9} \cdot \frac{9}{14} : \frac{5}{14} = \frac{3}{5} = 0{,}6 \Rightarrow 60\%.$

БАЙЕСОВСКИЙ АЛГОРИТМ ДЛЯ КЛАССИФИКАЦИИ

Аналогичным образом с помощью наивного байесовского алгоритма можно прогнозировать несколько различных классов на основе множества признаков.

+ классификация быстрая и простая

+ в случае, если выполняется предположение о независимости, классификатор показывает очень высокое качество

- если в тестовых данных присутствует категория, не встречавшаяся в данных для обучения, модель присвоит ей нулевую вероятность

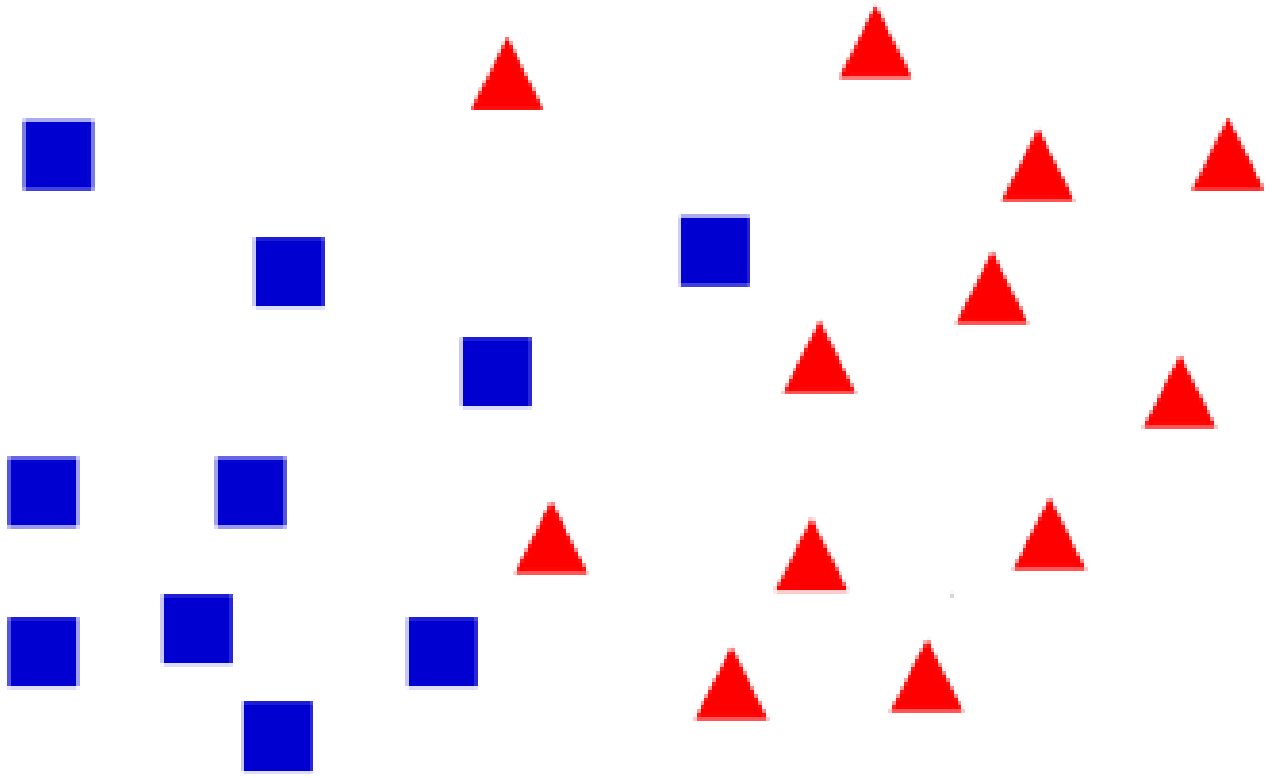
НАИВНЫЙ БАЙЕСОВСКИЙ АЛГОРИТМ

https://scikit-learn.org/stable/modules/naive_bayes.html

МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

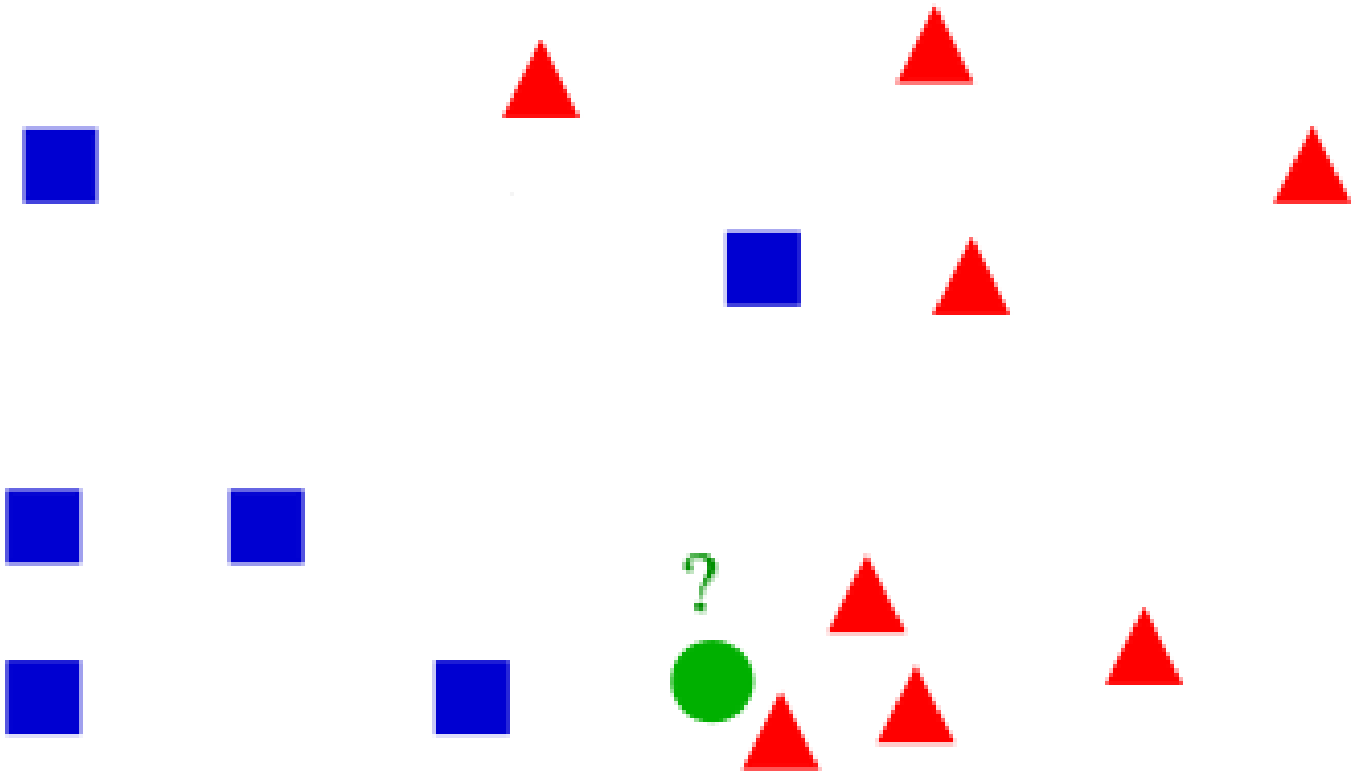
МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

Идея: схожие объекты находятся близко друг к другу в пространстве признаков.



МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

Как классифицировать новый объект?



МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

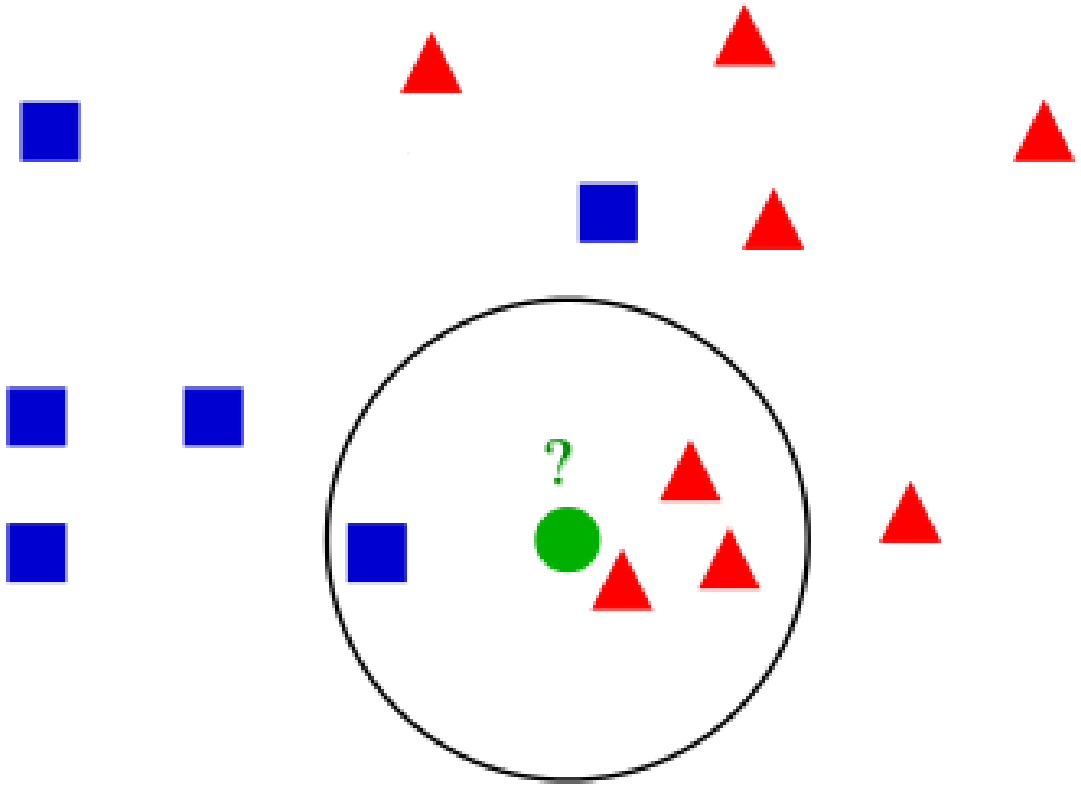
Чтобы классифицировать новый объект, нужно:

- Вычислить расстояние до каждого из объектов обучающей выборки.
- Выбрать k объектов обучающей выборки, расстояние до которых минимально.
- Класс классифицируемого объекта — это класс, наиболее часто встречающийся среди k ближайших соседей.

МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

Число ближайших соседей k – гиперпараметр метода.

Например, для $k = 4$ получим:



То есть объект будет отнесён к классу *треугольников*.

ФОРМАЛИЗАЦИЯ МЕТОДА

Пусть k – количество соседей. Для каждого объекта u возьмём k ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}.$$

Тогда класс объекта u определяется следующим образом:

$$\boldsymbol{a(u) = \operatorname*{argmax}_{y \in Y} \sum_{i=1}^{k} [y(x_{(i;u)}) = y]}.$$

ФОРМАЛИЗАЦИЯ МЕТОДА

Пусть k – количество соседей. Для каждого объекта u возьмём k ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}.$$

Тогда класс объекта u определяется следующим образом:

$$a(u) = \underset{y \in Y}{\operatorname{argmax}} \sum_{i=1}^{k} [y(x_{(i;u)}) = y].$$

Ближайшие объекты – это объекты, расстояние от которых до данного объекта наименьшее по некоторой метрике ρ.

ФОРМАЛИЗАЦИЯ МЕТОДА

Пусть k – количество соседей. Для каждого объекта u возьмём k ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}.$$

Тогда класс объекта u определяется следующим образом:

$$a(u) = \underset{y \in Y}{\operatorname{argmax}} \sum_{i=1}^{k} [y(x_{(i;u)}) = y].$$

Ближайшие объекты – это объекты, расстояние от которых до данного объекта наименьшее по некоторой метрике ρ.

- В качестве метрики ρ как правило используют евклидово расстояние, но можно использовать и другие метрики.
- Перед использованием метода необходимо масштабировать данные, иначе признаки с большими числовыми значениями будут доминировать при вычислении расстояний.

КАЛИБРОВКА ВЕРОЯТНОСТЕЙ

Калибровка вероятностей - приведение ответов алгоритма к значениям, близким к вероятностям объектов принадлежать конкретному классу.

Зачем это нужно?

- Вероятности гораздо проще интерпретировать
- Вероятности могут дать дополнительную информацию о результатах работы алгоритма

КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса, $Y = \{+1, -1\}$

Задача: для классификатора $a(x)$, предсказывающего значения из отрезка $[0, 1]$, либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями $p(y = +1|x)$.

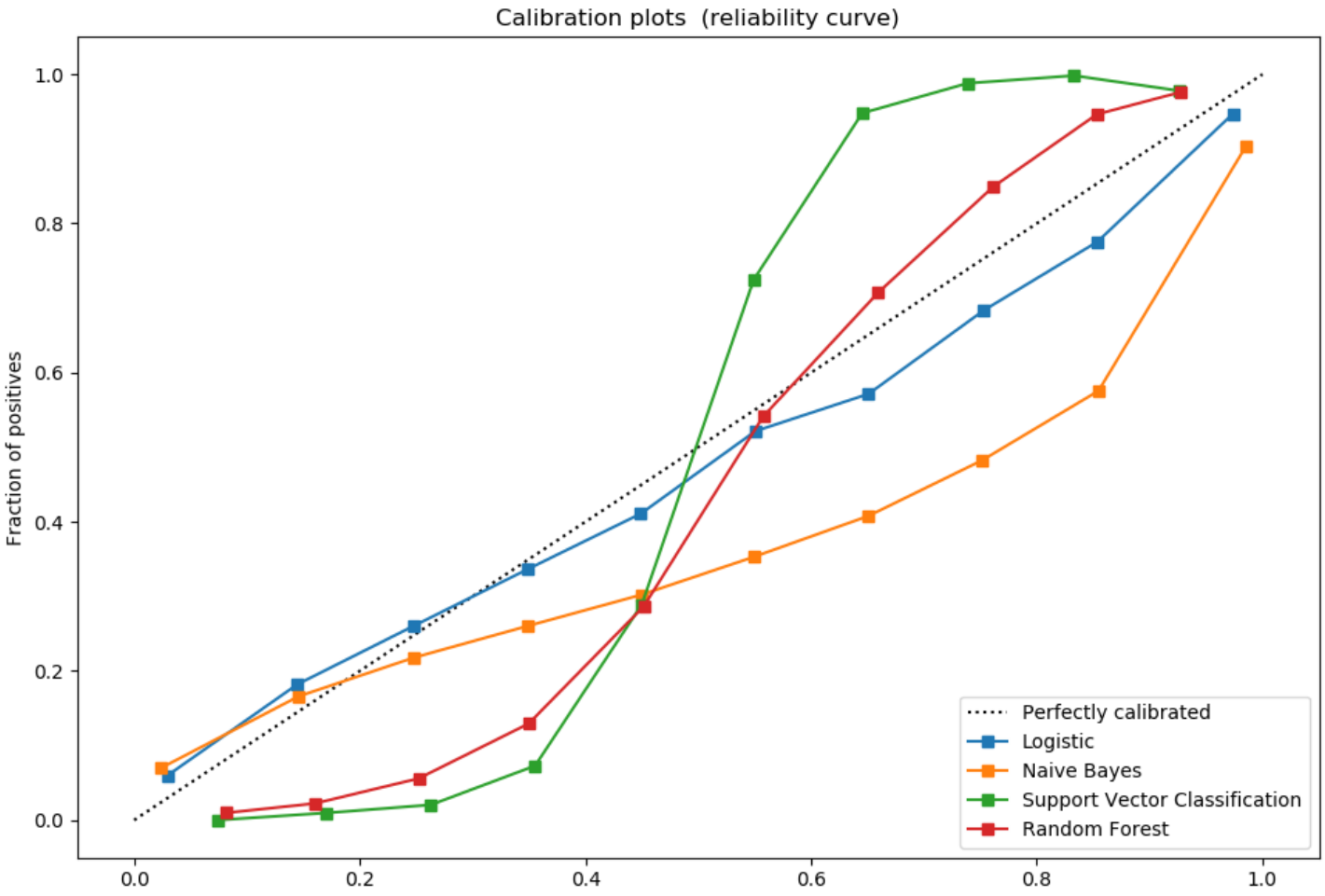
КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса, $Y = \{+1, -1\}$

Задача: для классификатора $a(x)$, предсказывающего значения из отрезка $[0, 1]$, либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями $p(y = +1|x)$.

Идея: обучаем логистическую регрессию на ответах классификатора $a(x)$.

ПРИМЕР ИЗ SKLEARN



КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса, $Y = \{+1, -1\}$

Задача: для классификатора $a(x)$, предсказывающего значения из отрезка $[0, 1]$, либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями $p(y = +1|x)$.

Идея: ***обучаем логистическую регрессию на ответах классификатора*** $\boldsymbol{a(x)}$.

КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса, $Y = \{+1, -1\}$

Задача: для классификатора $a(x)$, предсказывающего значения из отрезка $[0, 1]$, либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями $p(y = +1|x)$.

Идея: ***обучаем логистическую регрессию на ответах классификатора*** $\boldsymbol{a(x)}$.

- $\pi(x; \alpha; \beta) = \sigma(\alpha \cdot \boldsymbol{a(x)} + \beta) = \frac{1}{1+e^{-(\alpha \cdot a(x)+\beta)}}$

КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса, $Y = \{+1, -1\}$

Задача: для классификатора $a(x)$, предсказывающего значения из отрезка $[0, 1]$, либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями $p(y = +1|x)$.

Идея: ***обучаем логистическую регрессию на ответах классификатора*** $a(x)$.

- $\pi(x; \alpha; \beta) = \sigma(\alpha \cdot a(x) + \beta) = \frac{1}{1+e^{-(\alpha \cdot a(x)+\beta)}}$
- Находим α и β, минимизируя логистическую функцию потерь (*то есть обучаем логистическую регрессию*):

$$-\sum_{y_i=-1} \log\big(1 - \pi(x; \alpha; \beta)\big) - \sum_{y_i=+1} \log\big(\pi(x; \alpha; \beta)\big) \to \min_{\alpha,\beta}$$