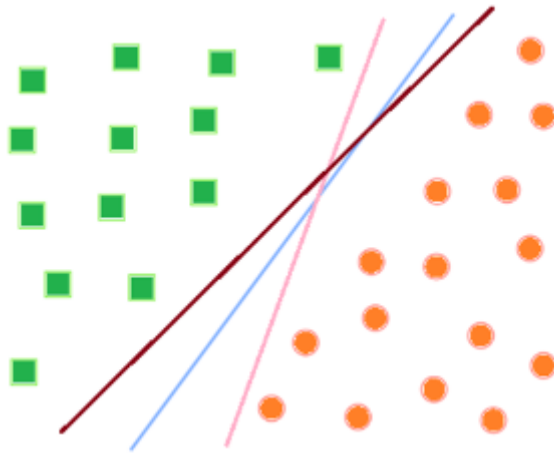


Метод опорных векторов (SVM)

SVM для линейно-разделимой выборки

Линейно разделимая выборка в задаче бинарной классификации - это набор объектов двух классов, которые можно безошибочно классифицировать, разбив пространство на две части с помощью линейной поверхности (гиперплоскости)



Какой из классификаторов лучше?

(самый лучший - самый уверенный в своих предсказаниях классификатор)

Введём понятие отступа (margin) на объекте. Отступ M_i на объекте x_i вычисляется по формуле

$$M_i = y_i * (\omega, x_i)$$

где

y_i - правильный ответ (класс +1 или -1)

(ω, x_i) - скалярное произведение вектора весов модели и вектора признаков

1. Знак отступа говорит о корректности классификации объекта

Если объект классифицирован верно, то предсказание модели $a(x_i) = \text{sign}(w, x_i)$ совпадает с правильным классом y_i . Это происходит в двух случаях:

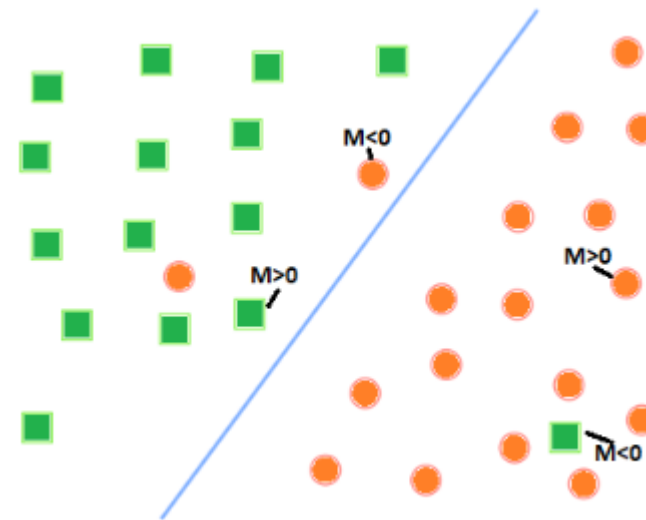
- $\text{sign}(w, x_i) = +1$ и $y_i = +1$, при этом $(w, x_i) > 0$,
- $\text{sign}(w, x_i) = -1$ и $y_i = -1$, при этом $(w, x_i) < 0$.

В обоих случаях $M_i = y_i \cdot (w, x_i) > 0$.

Если объект классифицирован моделью неверно, то предсказание $a(x_i) = \text{sign}(w, x_i)$ не совпадает с правильным классом y_i . Это происходит в двух случаях:

- $\text{sign}(w, x_i) = +1$ и $y_i = -1$, при этом $(w, x_i) > 0$,
- $\text{sign}(w, x_i) = -1$ и $y_i = +1$, при этом $(w, x_i) < 0$.

В обоих случаях $M_i = y_i \cdot (w, x_i) < 0$.



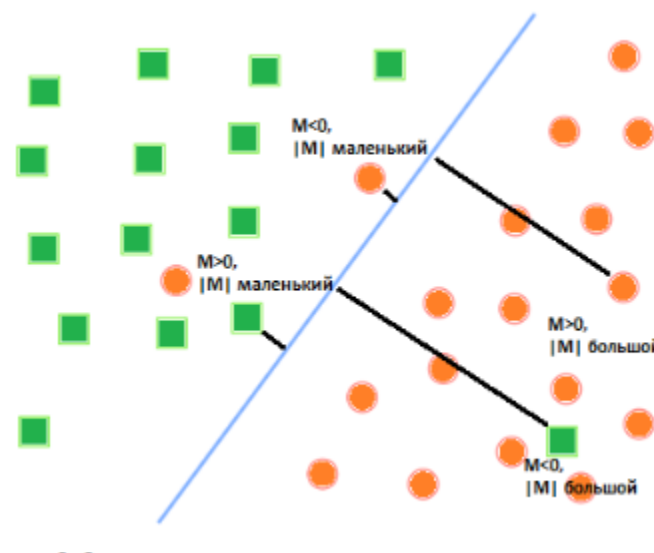
2. Абсолютная величина отступа говорит об уверенности модели в предсказании на объекте

Геометрическое значение отступа обозначает расстояние от объекта до разделяющей полосы. Модуль отступа пропорционален этому расстоянию.

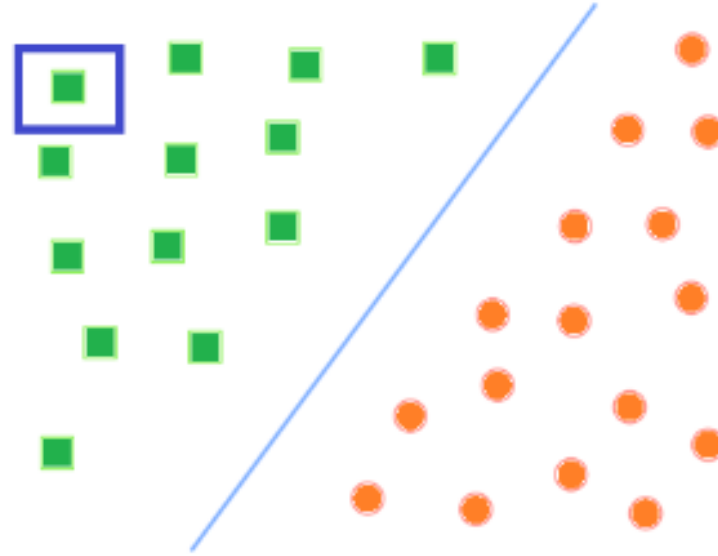
- Если модуль отступа велик - объект находится далеко от разделяющей полосы, что означает, что классификатор более уверен в своём предсказании для данного объекта.
- Если модуль отступа маленький - объект находится близко к разделяющей полосе, что означает малую уверенность модели в предсказании для этого объекта.

Таким образом, по знаку и абсолютной величине отступа можно судить:

- 1) о корректности классификации на объекте,
- 2) об уверенности модели в предсказании на объекте.

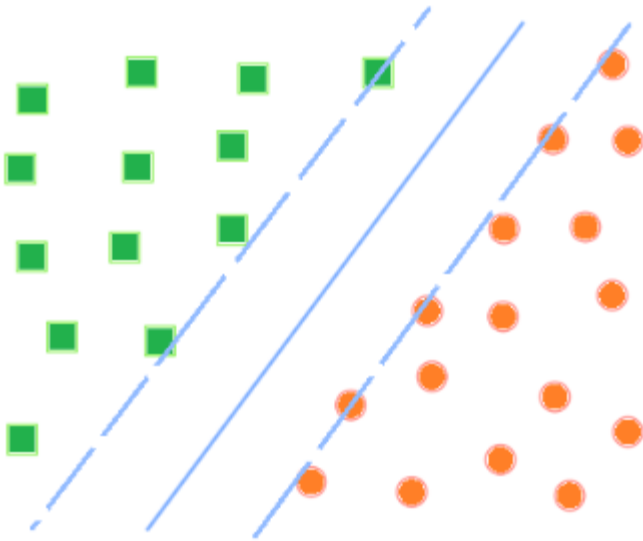


Охарактеризуйте отступ на выделенном объекте



- Большой положительный
- Маленький положительный
- Большой отрицательный
- Маленький отрицательный

Вокруг разделяющей прямой любого классификатора есть пространство, в которое не попадает ни один объект выборки. Другими словами, можно провести полосу *максимальной ширины* вокруг разделяющей прямой, которая не содержит в себе ни одного объекта (на границе этой полосы будут какие-то объекты). Эта полоса называется **разделяющей полосой**.



В терминах разделяющей полосы можно сказать, что **метод опорных векторов (SVM) - это классификатор, имеющий наиболее широкую разделяющую полосу среди всех возможных линейных классификаторов!**

Сформулируем задачу метода SVM с **математической точки зрения**:

SVM делает предсказания классов по формуле $a(x) = \text{sign}(w, x)$

Задача обучения SVM в линейно разделимом случае имеет вид:

$$\frac{1}{2} ||w||^2 \rightarrow \min_w$$

при условии, что для всех объектов выборки выполняется неравенство:

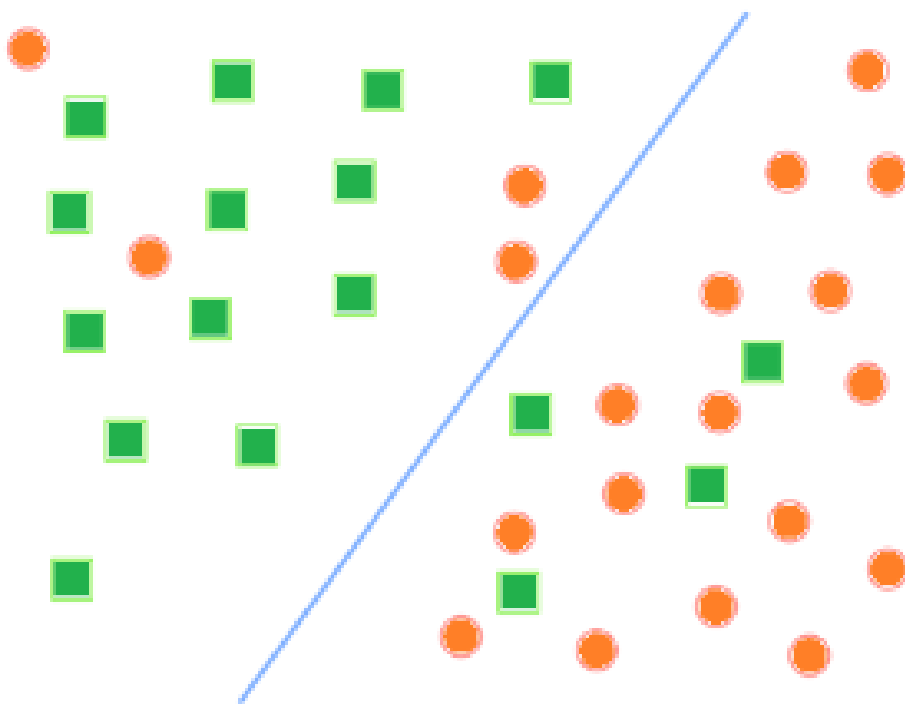
$$y_i \cdot (w, x_i) \geq 1 \quad \text{для всех } i$$

Здесь y_i — это истинные метки классов,

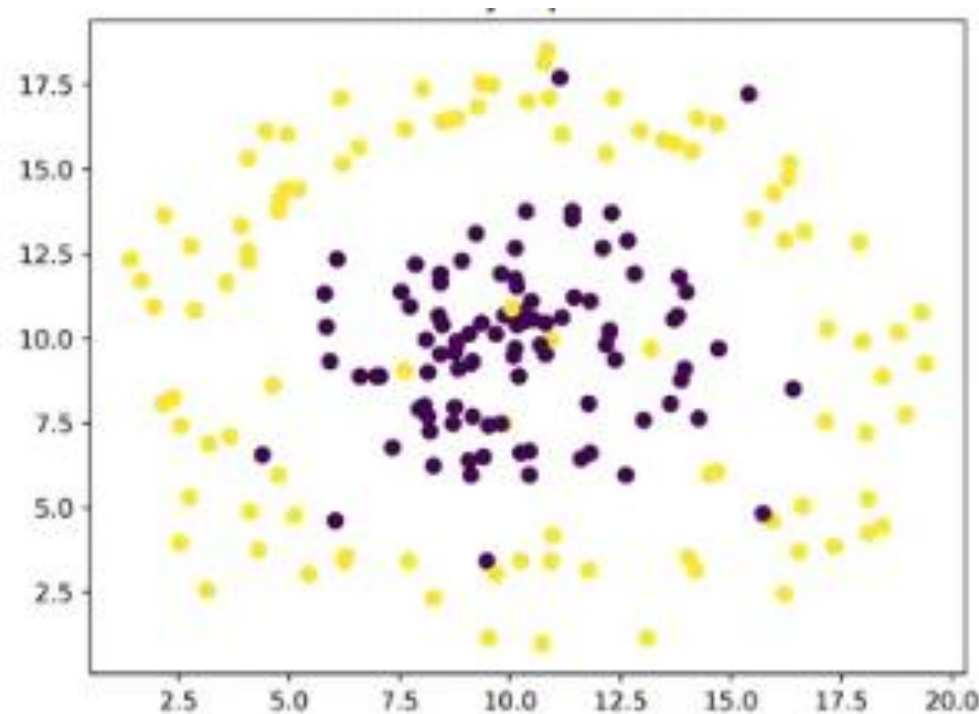
x_i — это объекты выборки.

2. Линейно неразделимая выборка

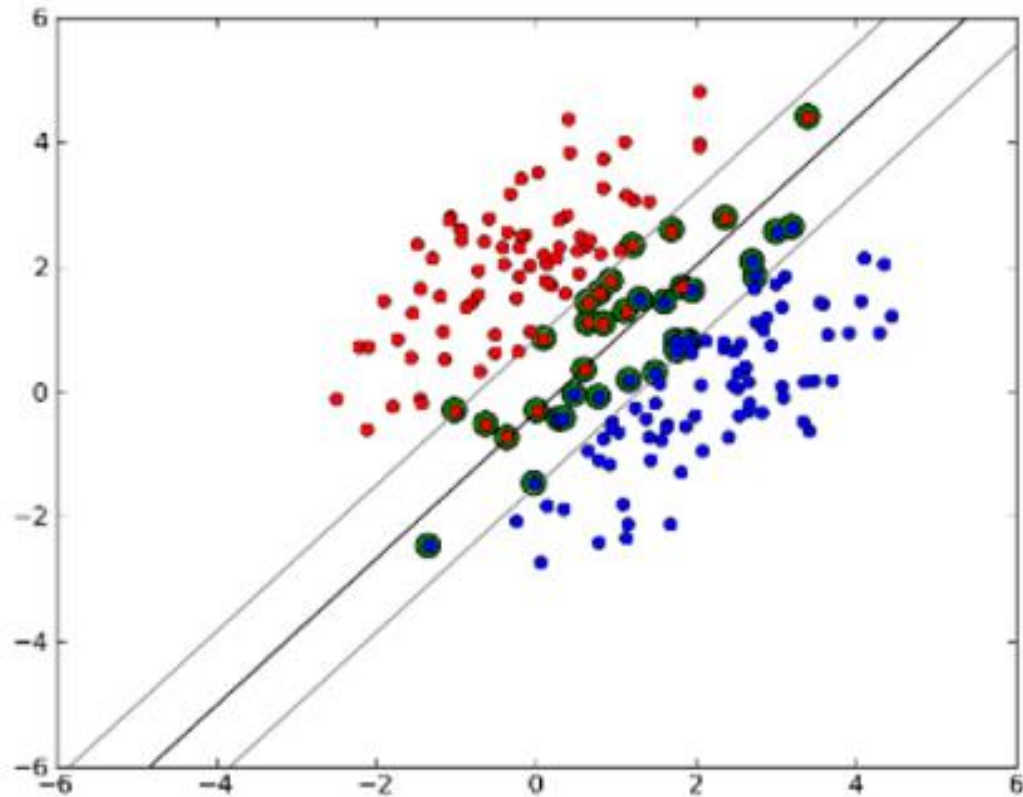
Есть линейная зависимость в данных: классы можно разделить прямой линией, но из-за шума в данных строгая линейная делимость отсутствует



Нет линейной зависимости в данных: классы не могут быть разделены прямой линией



Рассмотрим ситуацию, когда в данных есть линейная зависимость, но линейной разделимости нет



Построим классификатор с наиболее широкой разделяющей полосой

В разделяющую полосу всегда будут попадать какие-то объекты

Будем **штрафовать** объекты, попадающие внутрь разделяющей полосы (так как это объекты, в которых модель не уверена).

Штрафовать означает добавлять в функцию потерь штрафы, то есть некоторые положительные числа:

- чем дальше объект от своего класса внутри разделяющей полосы (или даже снаружи, но не в своем классе) - тем больше штраф ξ_i на этом объекте
- если объект попал в свой класс снаружи разделяющей полосы, то он не штрафуются: $\xi_i=0$

При обучении SVM в линейно неразделимом случае решаются две задачи:

- максимизируется ширина разделяющей полосы;
- минимизируется сумма штрафов на объектах выборки.

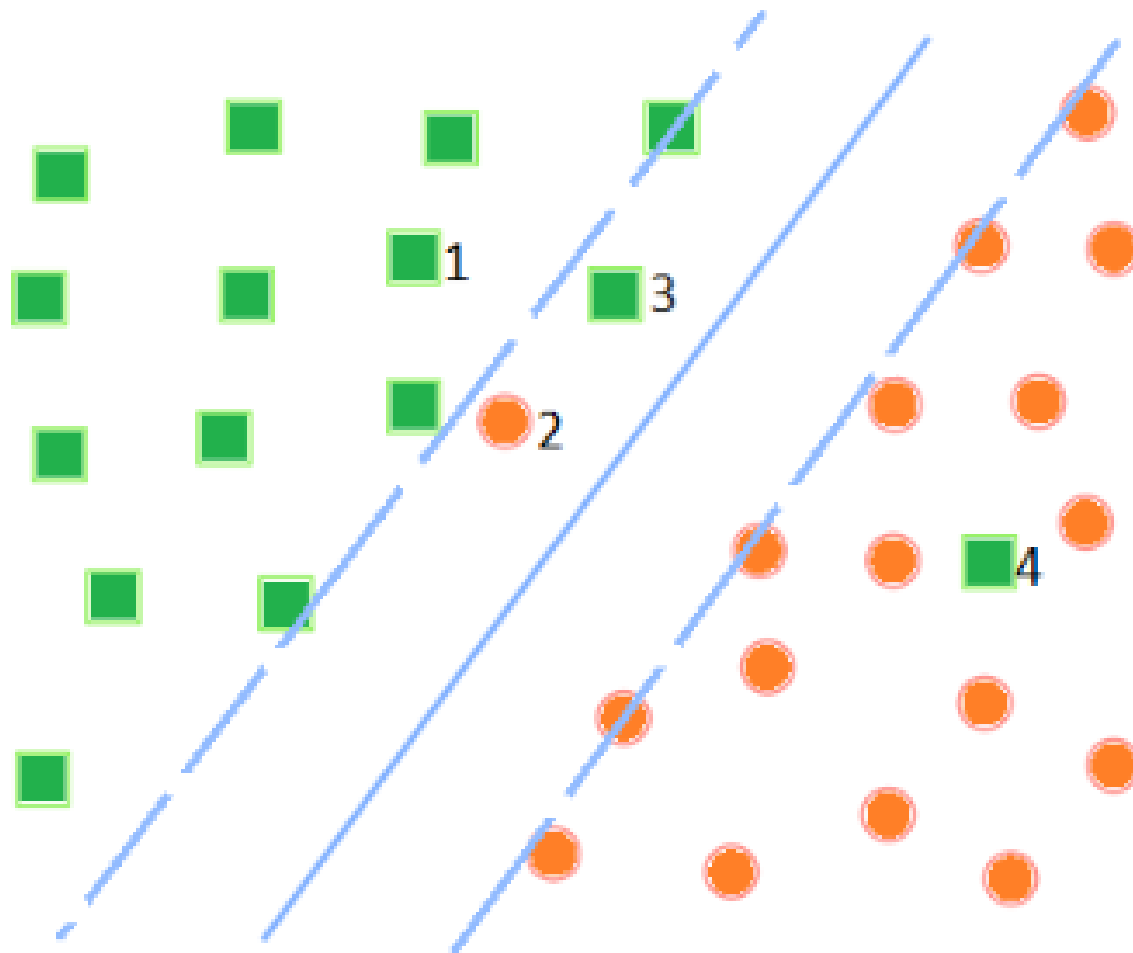
С точки зрения математики функция потерь SVM в этом случае имеет вид

$$\frac{1}{2}||w||^2 + C \sum_{i=1}^l \xi_i \rightarrow \min_w$$

где C - гиперпараметр, регулирующий силу штрафов,
 ξ_i - штраф на i -м объекте выборки.

В результате обучения получаем наиболее уверенный в предсказаниях линейный классификатор.

Упорядочите штрафы на объектах 1, 2, 3, 4 в порядке возрастания
(от самого маленького до самого большого)



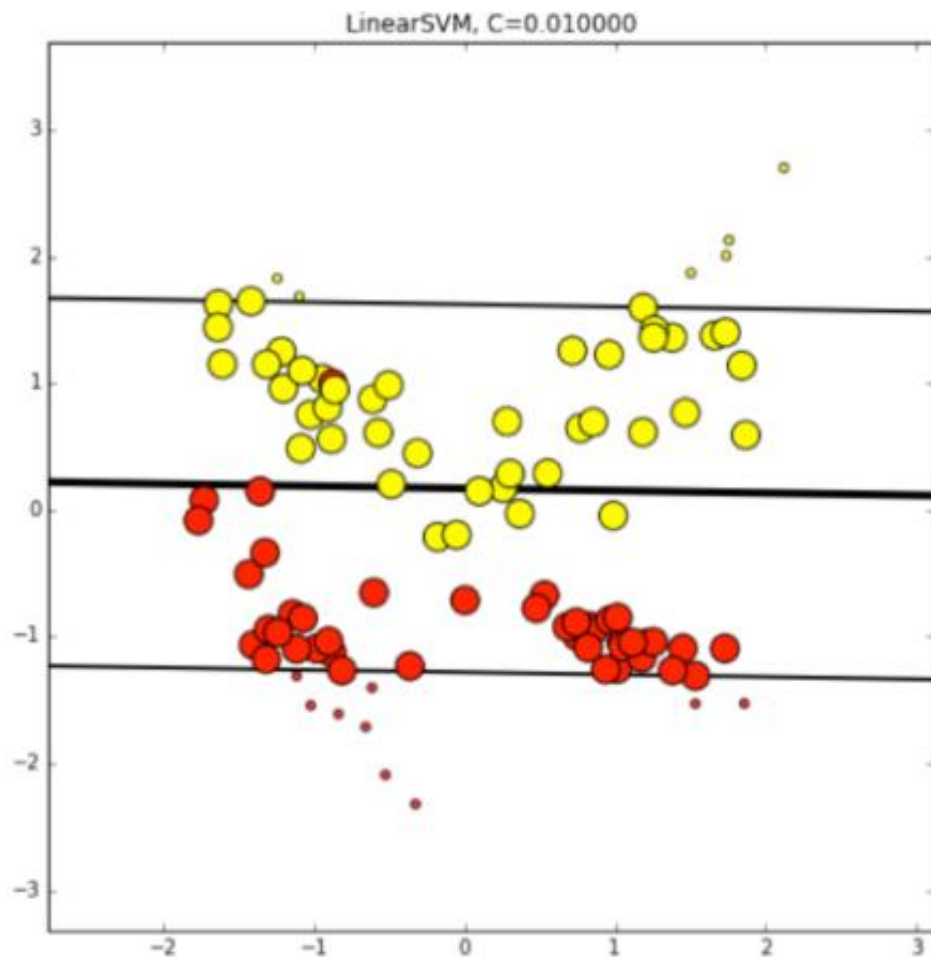
Гиперпараметр C

Метод позволяет регулировать то, насколько сильно мы хотим максимизировать ширину полосы или же нам важнее получить меньше штрафуемых объектов. Напомним, что функция потерь метода SVM выглядит так:

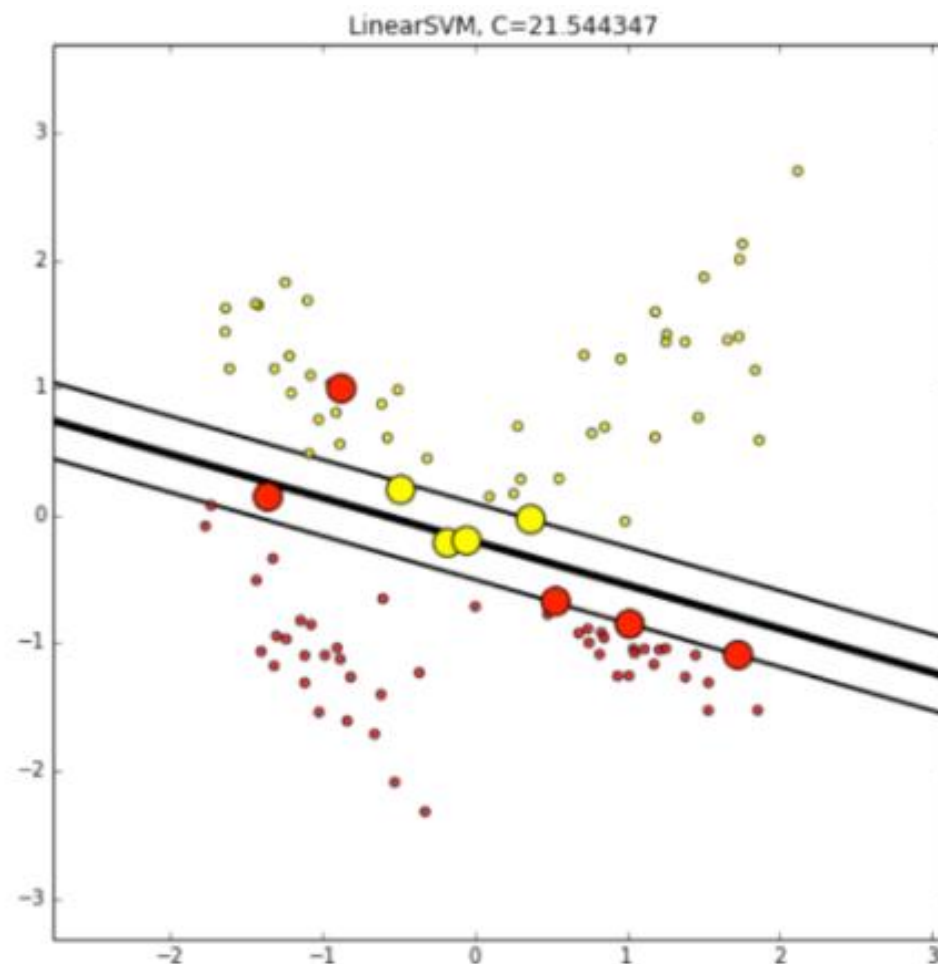
$$\frac{1}{2} ||w||^2 + C \sum_{i=1}^l \xi_i \rightarrow \min_w$$

Тогда в зависимости от значения C происходят следующие изменения:

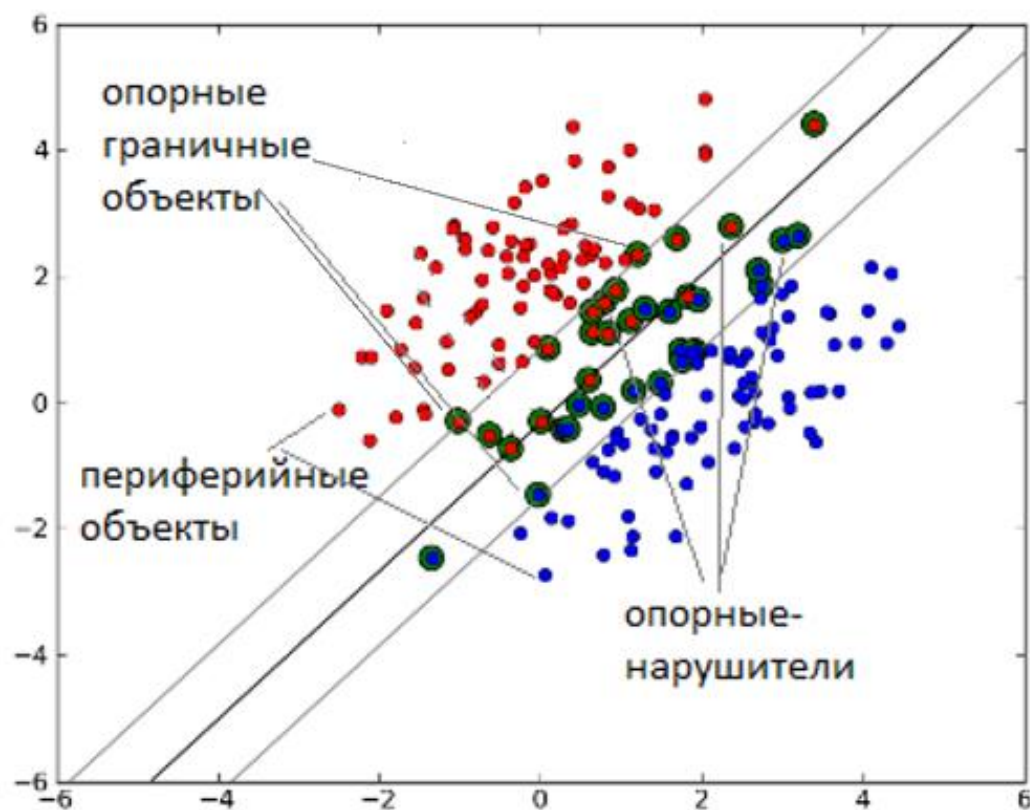
При маленьких C нам не важно, как ведут себя штрафы, и метод фокусируется на построении алгоритма с наиболее широкой разделяющей полосой



При больших C нам важно минимизировать штрафы, поэтому и разделяющая полоса будет узкой.

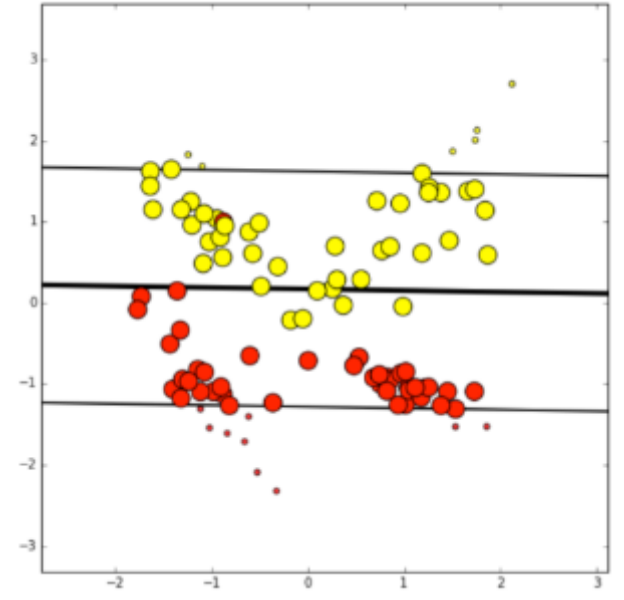
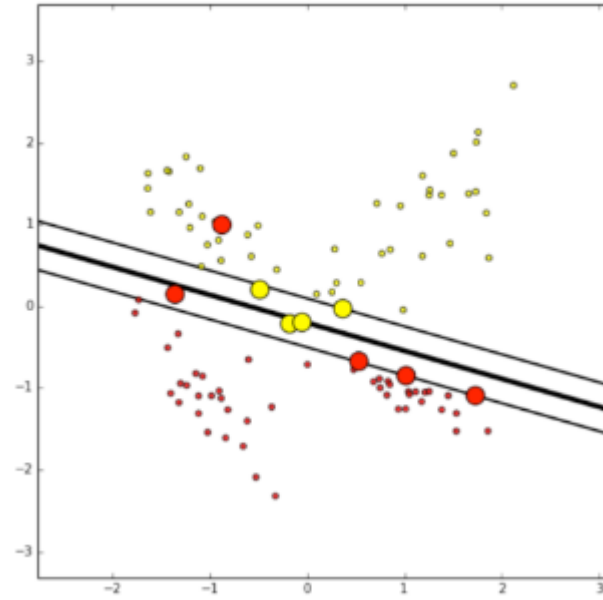
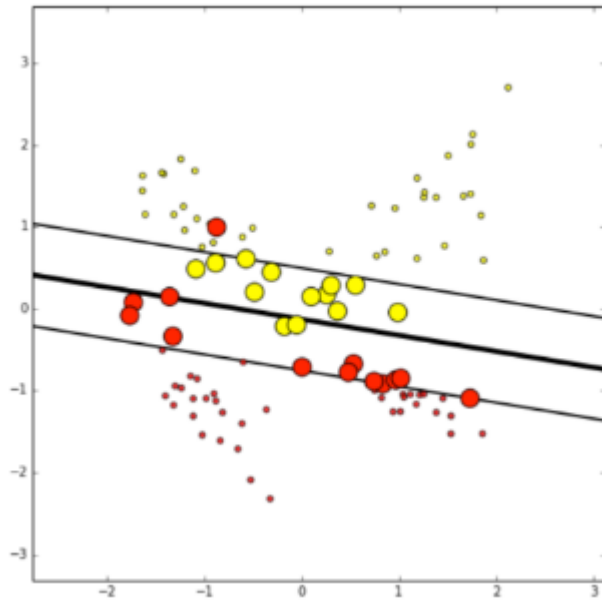


Объекты, попадающие на границу или внутрь разделяющей полосы, а также объекты, попадающие не в свой класс, называются **опорными векторами**.



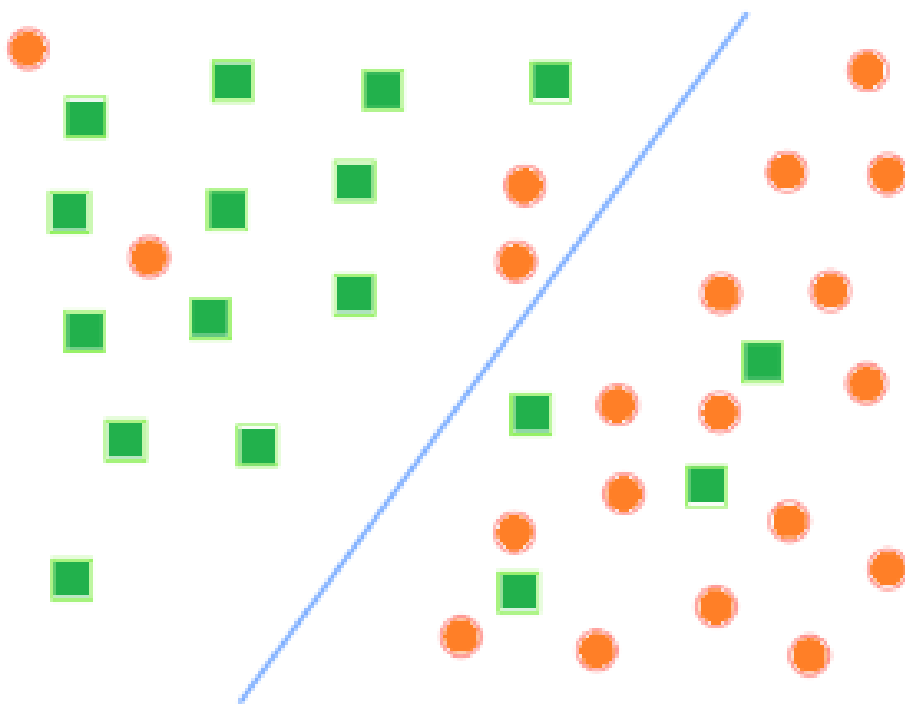
SVM старается минимизировать количество таких объектов, то есть при обучении ориентируется именно на них. Поэтому он и называется **метод опорных векторов**

Соотнесите картинки со значениями C в методе опорных векторов

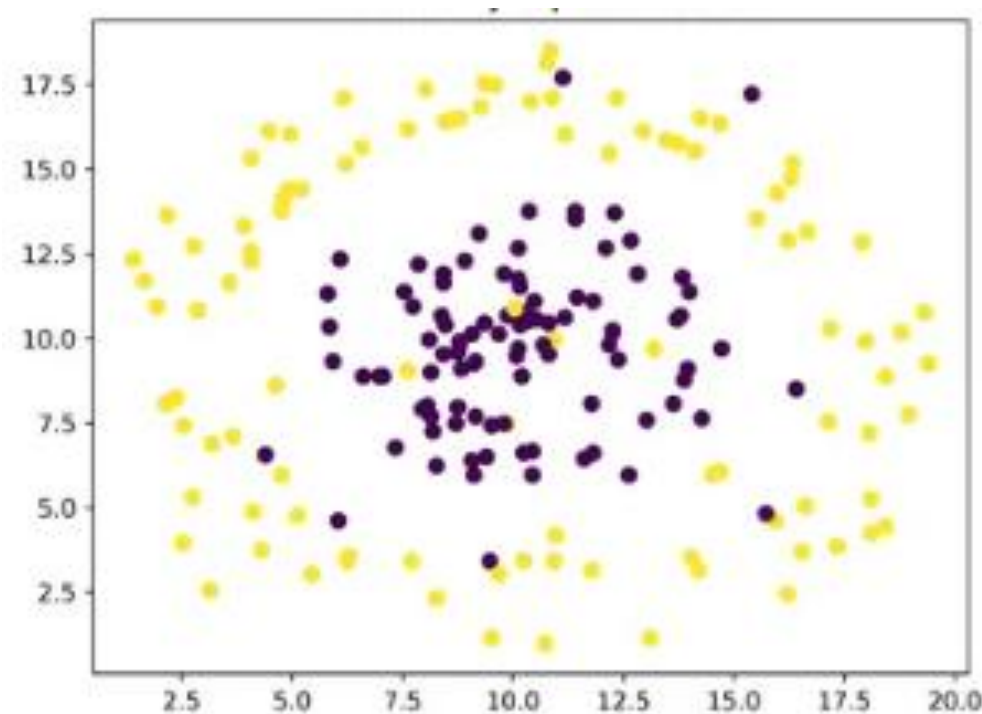


2. Линейно неразделимая выборка

Есть линейная зависимость в данных: классы можно разделить прямой линией, но из-за шума в данных строгая линейная разделимость отсутствует

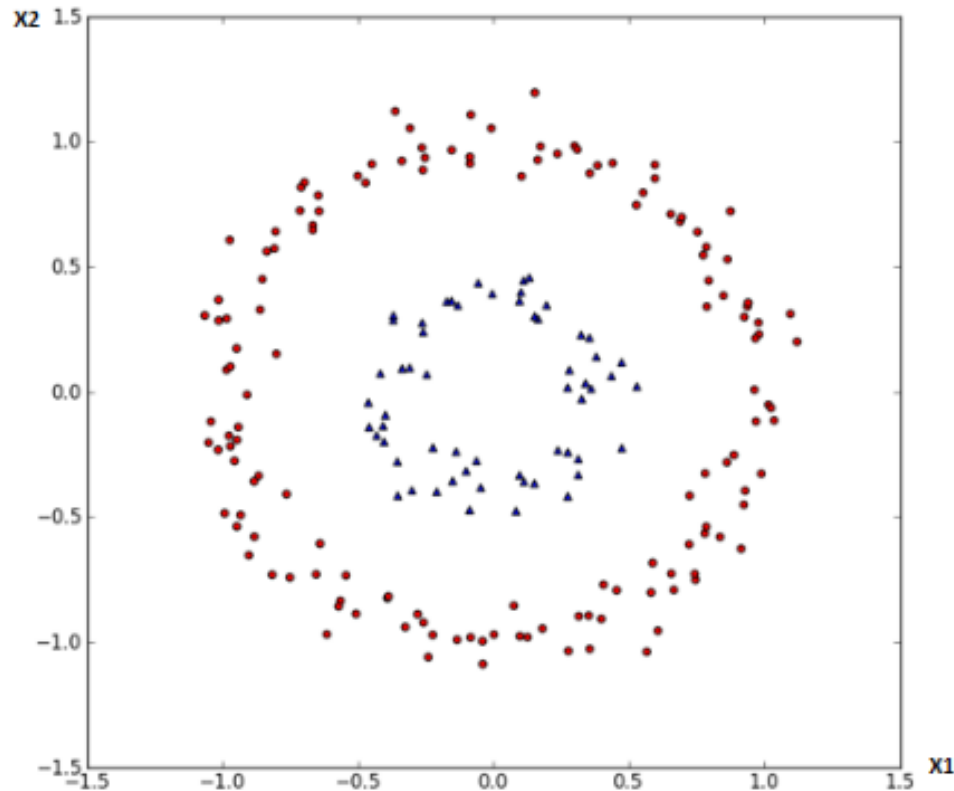


Нет линейной зависимости в данных: классы не могут быть разделены прямой линией



3. Решение задач классификации, где в данных нет линейной зависимости

Модификация SVM называется **ядровым методом опорных векторов**.



Существует зависимость между признаками и целевой переменной: синие точки находятся на окружности маленького радиуса, а красные - на окружности большего радиуса. Однако эта зависимость нелинейная, поэтому линейные модели не могут её восстановить

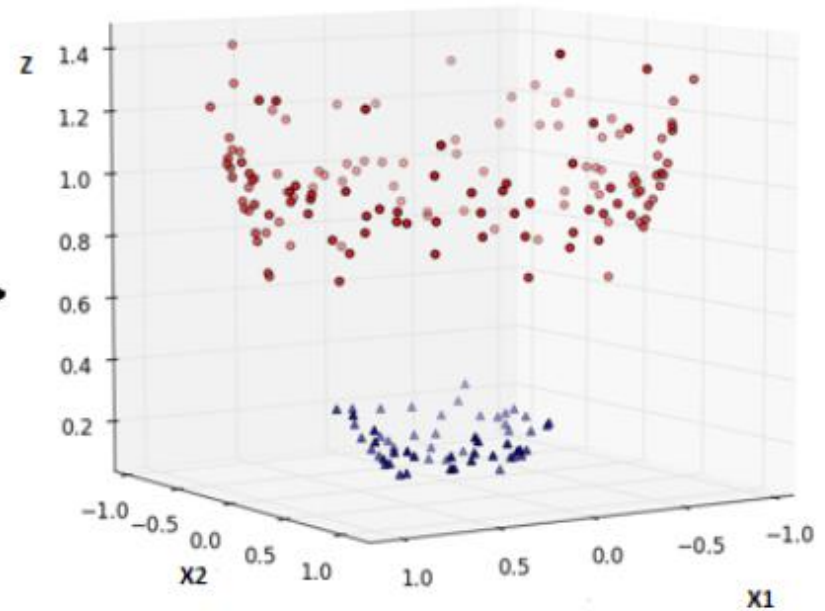
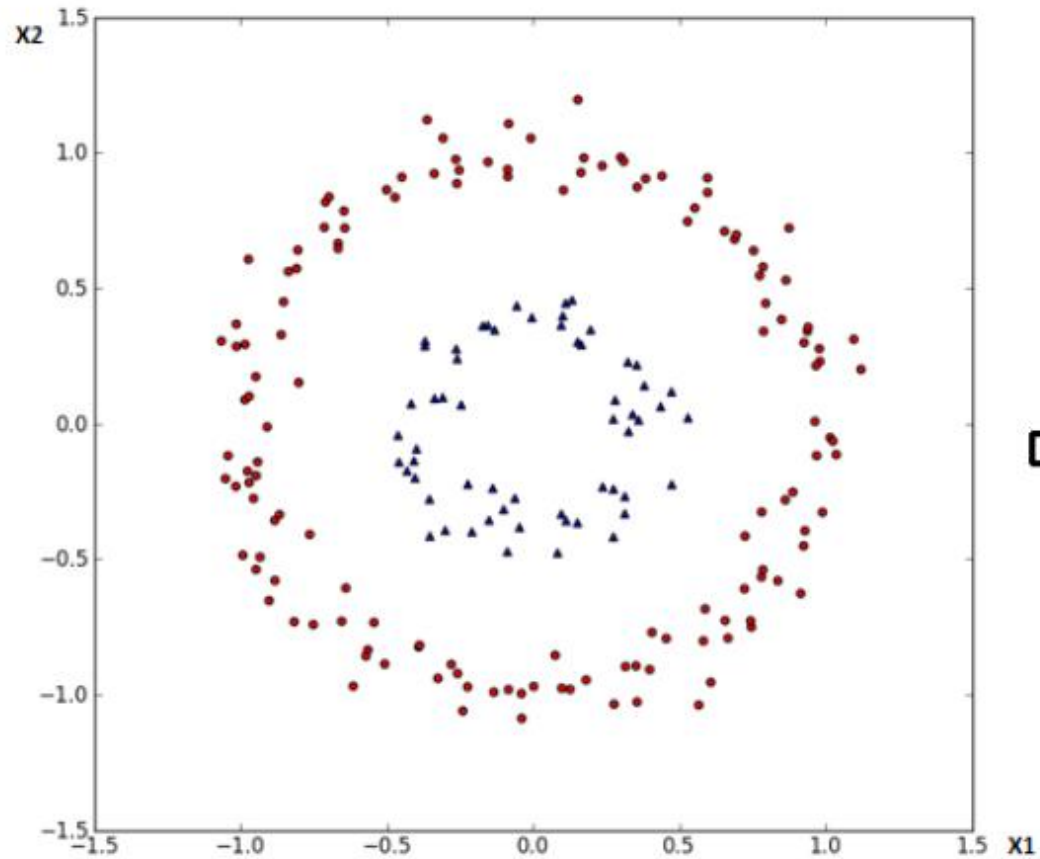
Создадим новый набор признаков, в котором точки становятся линейно разделимыми. В новых признаках количество признаков может быть как больше, так и меньше, чем в исходных данных.

В этом примере можно к исходным признакам добавить новый признак - расстояние от начала координат до точки $z = \sqrt{x_1^2 + x_2^2}$.

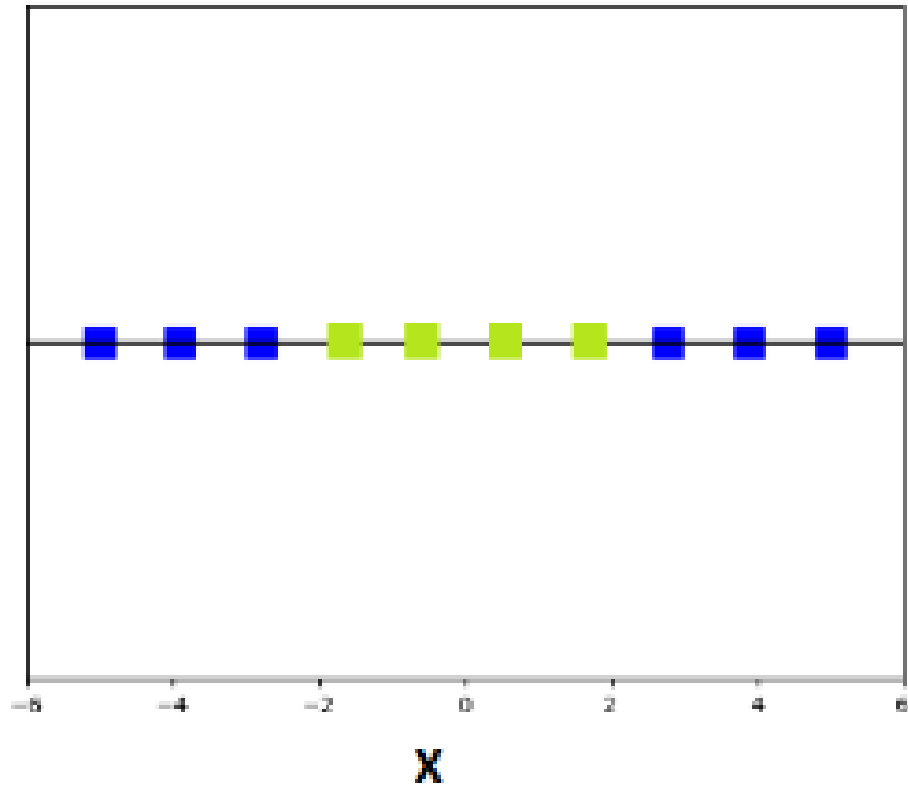
Тогда мы переходим из исходного двумерного пространства признаков к трехмерному

$$(x_1, x_2) \rightarrow (x_1, x_2, z = \sqrt{x_1^2 + x_2^2})$$

В новом пространстве признаков задача становится легко решаемой линейным классификатором

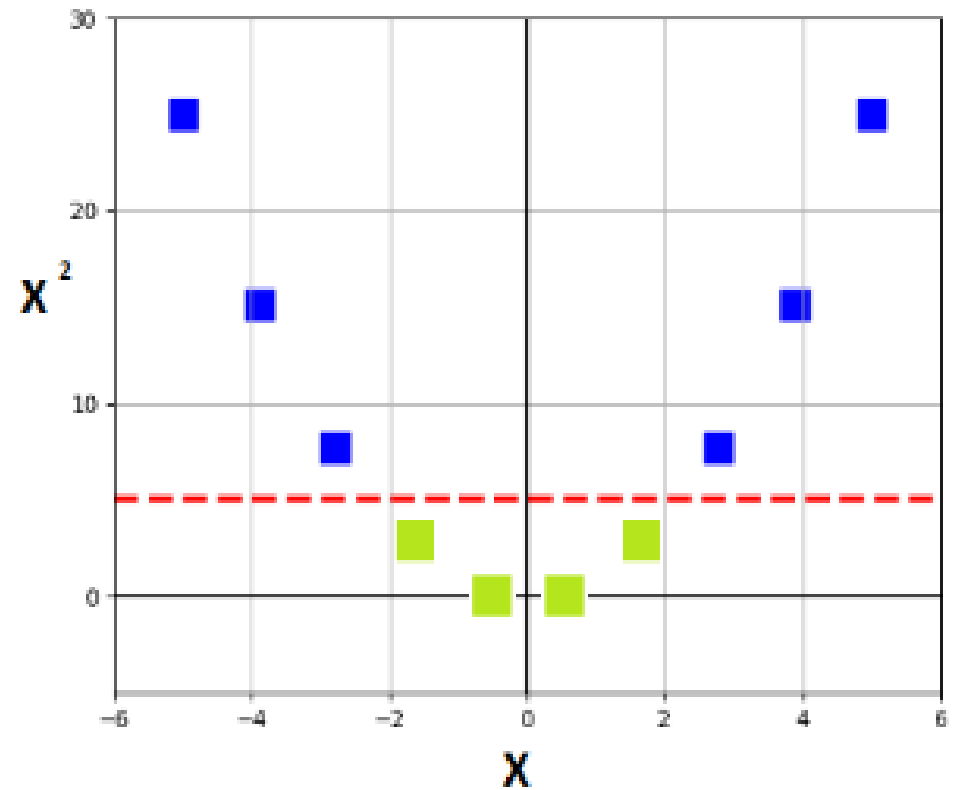


Дана выборка для задачи бинарной классификации. Объекты имеют один признак - x .



В признаках x , x^2 задача хорошо решается линейным классификатором.

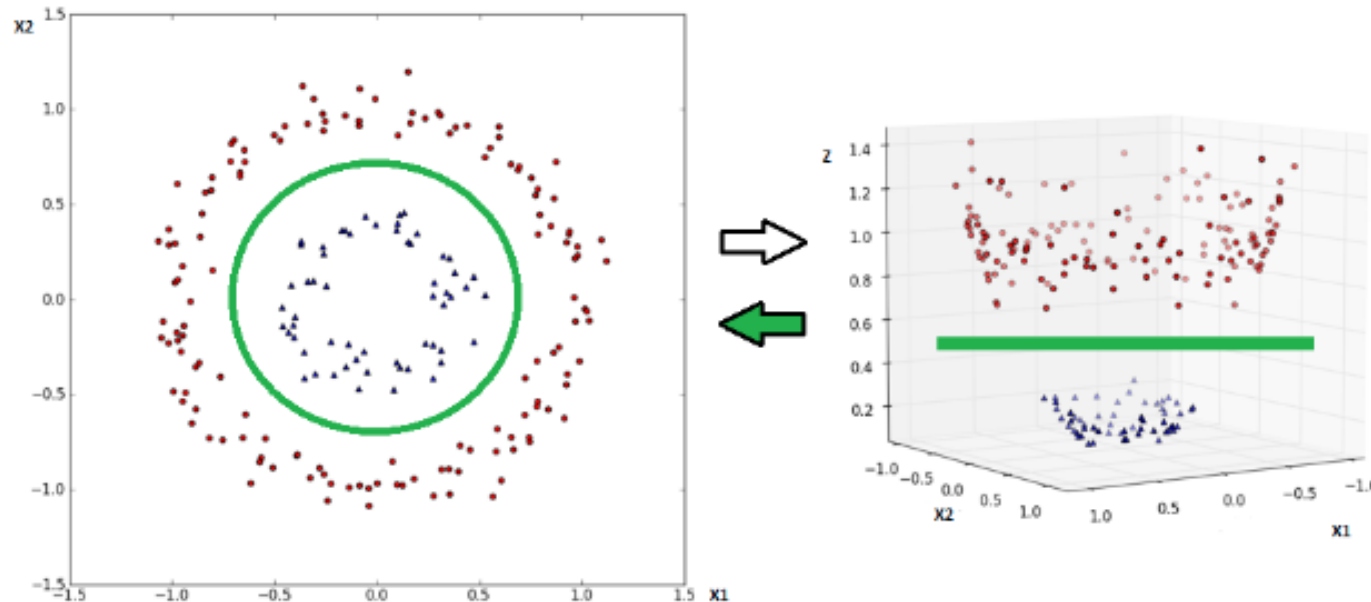
Если возвести признак x в квадрат, то данные будут выглядеть так



Сформулируем идею в виде алгоритма

Чтобы решить задачу бинарной классификации с нелинейно разделимыми данными, нужно:

1. подобрать нелинейное преобразование признаков: $x \rightarrow \varphi(x)$ - такое, что в новом признаковом пространстве $\varphi(x)$ данные станут линейно разделимыми;
2. обучить линейный классификатор на новых признаках $\varphi(x)$.



Разделяющая поверхность в исходном пространстве

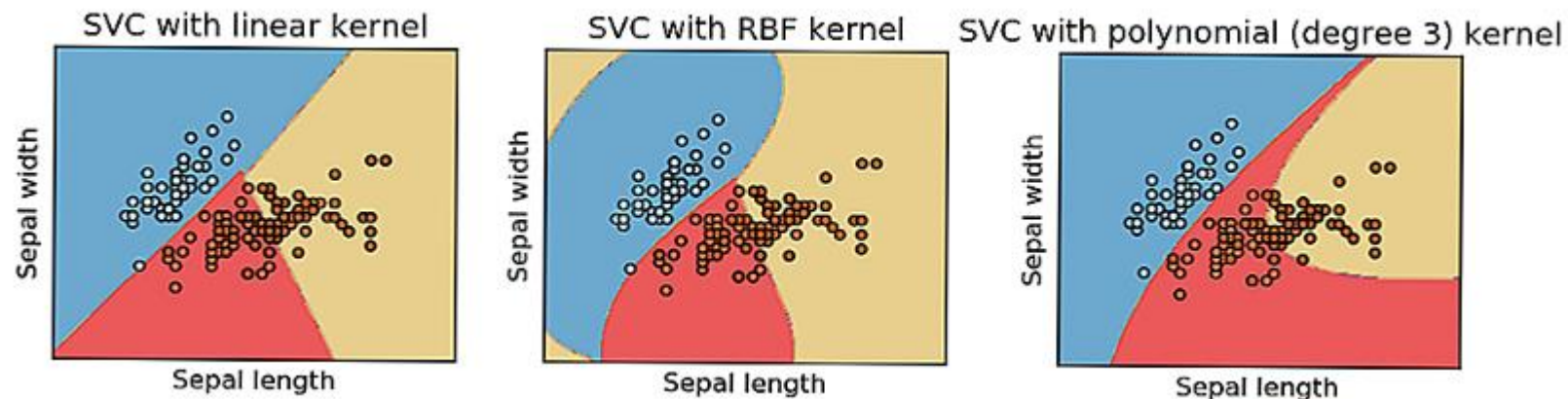
Линейный классификатор в новых признаках

Ядровой метод опорных векторов - это алгоритм, в котором:

- пользователь выбирает тип преобразования признаков (он задается функцией под названием *ядро*)
- линейный метод опорных векторов применяется к точкам в новом признаковом пространстве.

В итоге решается нелинейно разделимая задача :)

Примеры, как выглядят разделяющие поверхности ядрового SVM с разными ядрами:



С ядровым SVM тесно связано понятие ядра.

Пусть мы применили к признакам x некоторое преобразование и получили признаки $\varphi(x)$.

Тогда ядро - это функция от двух аргументов $K(a,b)=(\varphi(a),\varphi(b))$,

то есть значение ядра на объектах a и b (где каждый объект описывается исходными признаками) - это скалярное произведение векторов $\varphi(a)$ и $\varphi(b)$ в новом признаковом пространстве.

Зачем нам понадобилась некоторая новая функция $K(a,b)$?

Когда мы переходим в новое признаков пространство $x \rightarrow \varphi(x)$, то формулы для предсказания модели и функции потерь также записываются в новых признаках. Однако новых признаков может быть очень много (например, из 10 признаков может получиться 10 000), что делает обучение и предсказание в новых признаках значительно более ресурсоёмким.

Ядровой трюк позволяет решить эту проблему. Для некоторых моделей, в частности для SVM, предсказание модели и функцию потерь можно выразить через ядро $K(x)$. А это функция от исходных признаков, и, следовательно, обучение и предсказание будут занимать столько же времени, сколько и раньше, но при этом можно решать задачи с нелинейной разделимостью.

Пусть в исходных признаках $x = \{x_1, x_2, \dots, x_n\}$ предсказание модели SVM имеет вид

$$a(x) = \text{sign}(w, x) = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n. \quad (1)$$

Тогда после перехода к новым признакам $\varphi(x) = \{x'_1, x'_2, \dots, x'_N\}$ формула примет вид

$$a(x) = \text{sign}(w, \varphi(x)) = w_0 + w_1x'_1 + w_2x'_2 + \dots + w_Nx'_N. \quad (2)$$

При проведении некоторых преобразований задачу (2) можно переписать в виде

$$a(x) = \text{sign} \sum_{i=1}^n w_i y_i K(x_i, x). \quad (3)$$

В каждой из трех формул вектор весов w свои.

Для SVM моделей можем переписать задачу (2) в виде (3) - то есть записать предсказание модели после перехода в новые координаты в исходных признаках, с использованием функции ядра K !

По известному преобразованию признаков φ можно вычислить ядро K по формуле выше.

- Однако, зная ядро K , по этой же формуле можно найти и само преобразование φ .
- Поскольку для обучения метода SVM нам необходимо уметь вычислять только скалярное произведение, то удобнее задавать методу не преобразование φ , а сразу ядро K . Именно поэтому метод называется ядровым методом опорных векторов (kernel SVM).

Именно ядра задают нам то, как будет выглядеть результат метода опорных векторов!

Но не любая функция может быть ядром. По определению $K(a,b)=(\varphi(a),\varphi(b))$ ядро - это скалярное произведение.

Это значит, что функция является ядром (или задает преобразование признакового пространства) только в том случае, если она соответствует скалярному произведению.

Свойства ядер (= свойства скалярного произведения):

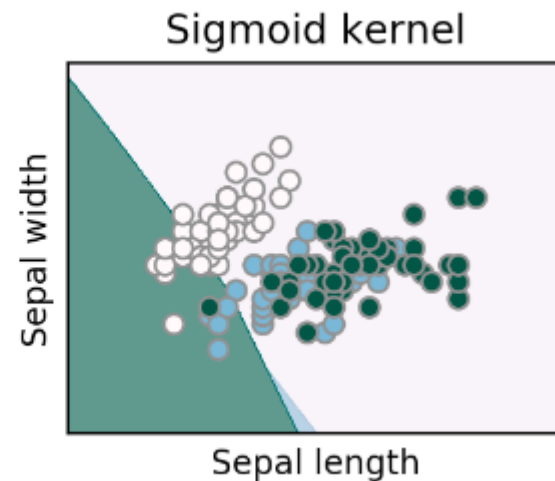
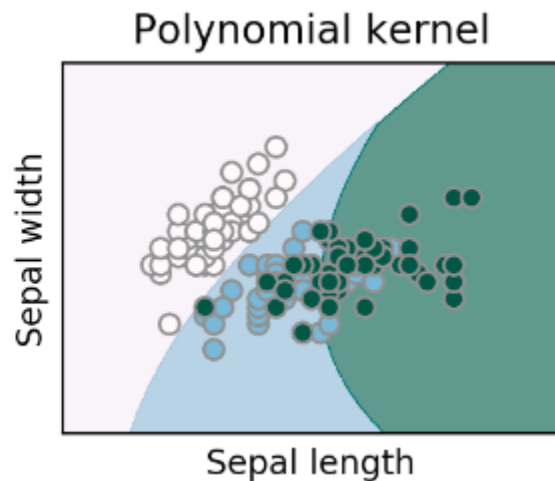
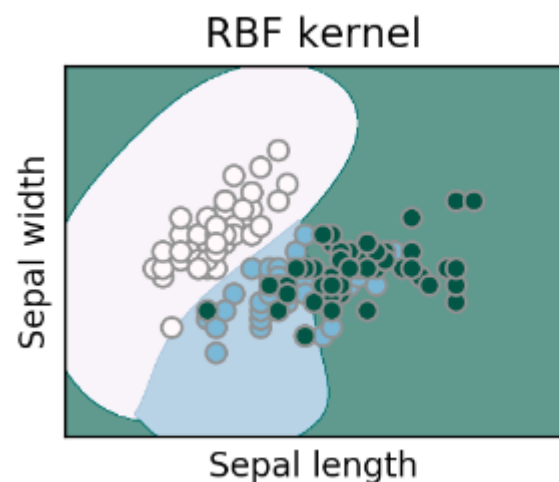
- $K(a,b)=K(b,a)$ - симметричность;
- матрица ядра KK положительно определена.

У ядер есть полезное и удобное свойство: линейная комбинация ядер, а также любая положительная степень ядра - это тоже ядро. Поэтому, имея несколько ядер, путем их линейной комбинации или возведения в степени можно конструировать новые ядра.

Популярные ядра:

- Радиальное: $K(a, b) = \exp(-\gamma \cdot ||a - b||^2)$.
- Полиномиальное: $K(a, b) = (\gamma \cdot (a, b) + r)^d$.
- Сигмоидальное: $K(a, b) = \tanh(\gamma \cdot (a, b) + r)$.

Здесь γ , r и d - гиперпараметры.



Интегральные методы классификации

accuracy, precision, recall и f1-score зависят от порога, на основании которого предсказанные моделью вероятности переводятся в классы.

Хотелось бы, чтобы были метрики, которые отражают *только качество модели* и никак не связаны с порогом. С помощью этих метрик мы могли бы знать, насколько хорошо делает предсказания та или иная модель, не привязываясь к конкретному порогу.

Метрики, которые учитывают все возможные пороги и не зависят от конкретного значения порога называются *интегральными*.

Самая известная интегральная метрика - это ***ROC-AUC***.


Рассмотрим пример:

На первом столбце представлена вероятность положительного класса, предсказанная моделью.

На втором столбце указаны правильные ответы (классы).

Отсортируем таблицу по убыванию предсказанных вероятностей:


р	класс
0.5	0
0.1	0
0.25	0
0.6	1
0.2	1
0.3	1
0.0	0



р	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0

Как выглядит отсортированная таблица для идеального алгоритма?
(то есть для алгоритма, который может идеально решить задачу классификации)

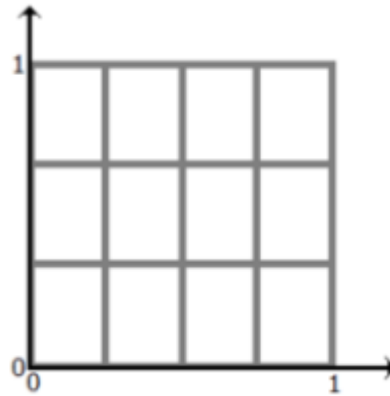
р	класс
0.5	0
0.1	0
0.25	0
0.6	1
0.2	1
0.3	1
0.0	0



р	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0

Нам нужен численный способ оценить описанное свойство модели правильно упорядочивать объекты по вероятностям. Для этого для начала нарисуем квадрат размера 1 на 1, и:

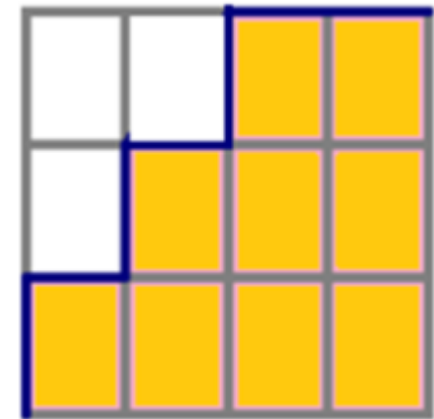
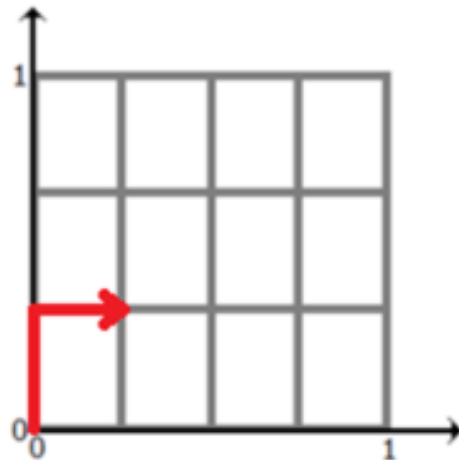
- разобьем его горизонтальную сторону на равные отрезки, количество которых равно числу 0 в выборке;
- разобьем его вертикальную сторону на равные отрезки, количество которых равно числу 1 в выборке.



Алгоритм построения ROC-кривой, с помощью которой мы оценим качество модели, очень прост. Кривую начинаем строить из точки (0,0):

- идем сверху вниз по отсортированной таблице;
- если встречаем 1, то делаем шаг на 1 вверх;
- если встречаем 0, то делаем шаг на 1 вправо.

р	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0



В результате получаем следующую кривую
- это и есть ROC-кривая

Площадь под построенной ROC-кривой - и есть метрика ROC-AUC (AUC - Area Under Curve).

В разобранном примере $\text{ROC-AUC} = 9/12 = 0.75$.

Свойства ROC-AUC:

- ROC-AUC принимает значение от 0 до 1.
- ROC-AUC идеальной модели равен 1 (так как в таблице после сортировки стоят сначала все 1, а потом все 0, и мы идем по квадрату до конца вверх, а потом до конца вправо - получаем, что ROC-кривая идет по сторонам квадрата).
- Чем больше ROC-AUC, тем лучше алгоритм сортирует объекты по вероятностям - то есть тем лучше алгоритм решает задачу классификации.

Вычислите ROC-AUC алгоритма, который предсказывает следующие вероятности
(в таблице в правом столбце стоят правильные ответы):

Предсказанная вероятность	Правильный ответ
0.6	-1
0.8	1
0.3	-1
0.55	-1
0.1	-1
0.96	1
0.33	1
0.2	-1
0.14	-1
0.88	1

Есть общепринятое определение и связанный с ним алгоритм построения ROC-кривой.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

По матрице ошибок можно посчитать следующие величины:

False Positive Rate (FPR) - доля неверно принятых объектов отрицательного класса: $FPR = \frac{FP}{FP + TN}$

True Positive Rate (TPR) - доля верно принятых объектов положительного класса: $TPR = \frac{TP}{TP + FN}$

Алгоритм бинарной классификации предсказывает следующие вероятности (в таблице в правом столбце стоят правильные ответы):

Предсказанная вероятность	Правильный ответ
0.6	-1
0.8	1
0.3	-1
0.55	-1
0.1	-1
0.96	1
0.33	1
0.2	-1
0.14	-1
0.88	1

Переведите вероятности в классы по порогу 0.4 и вычислите TP .

Алгоритм бинарной классификации предсказывает следующие вероятности (в таблице в правом столбце стоят правильные ответы):

Предсказанная вероятность	Правильный ответ
0.6	-1
0.8	1
0.3	-1
0.55	-1
0.1	-1
0.96	1
0.33	1
0.2	-1
0.14	-1
0.88	1

Переведите вероятности в классы по порогу 0.4 и вычислите TPR .

Алгоритм построения ROC-кривой следующий:

- Рисуем плоскость, ось x называем FPR, ось y - TPR.
- Для каждого возможного порога перевода вероятностей в классы вычисляем TPR и FPR. Отмечаем точку с координатами (FPR, TPR) на плоскости.
- Соединяем полученные точки кривой - получаем ROC-кривую.

