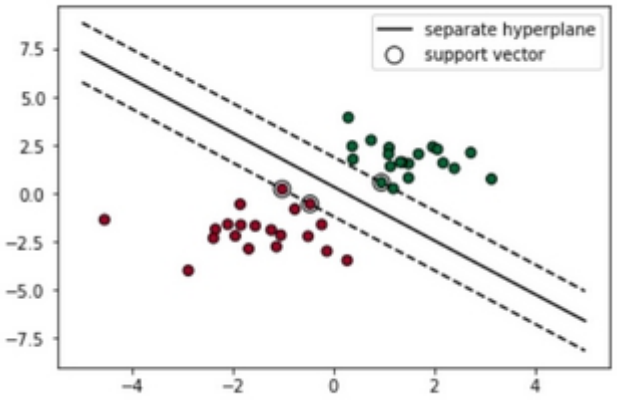
Машина опорных векторов отвечает за поиск границы решения для разделения различных классов и максимизации маржи.

Поля — это (перпендикулярные) расстояния между линией и ближайшими к ней точками.

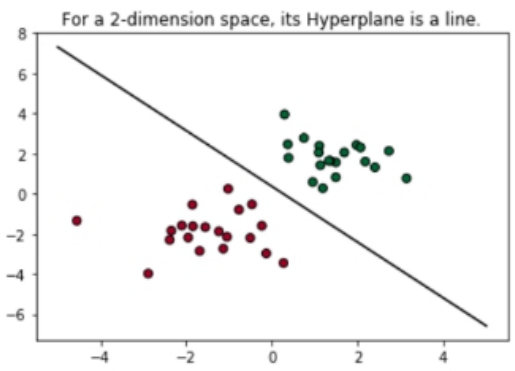


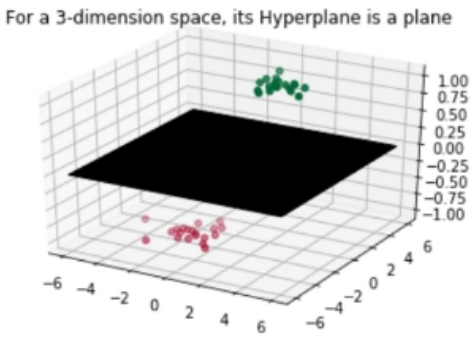
SVM в линейно разделимых случаях

Очевидно, что для разделения красных и зеленых точек в приведенном выше примере существуют бесконечные линии. SVM должен найти оптимальную строку с ограничением правильной классификации любого класса:

1. Соблюдайте ограничение: смотрите только на отдельные гиперплоскости (например, отдельные строки), гиперплоскости, которые правильно классифицируют классы.
2. Проведите оптимизацию: подберите ту, которая максимизирует маржу

Так что же такое гиперплоскость?





Гиперплоскость — это (n минус 1)-мерное подпространство для n-мерного пространства. Для двумерного пространства его гиперплоскость будет одномерной, то есть просто линией. Для трехмерного пространства его гиперплоскость будет двумерной, то есть плоскостью, разрезающей куб. Хорошо, вы поняли идею.

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \ldots + \beta_n * x_n = 0$$

Any Hyperplane can be written mathematically as above

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 = 0$$

For a 2-dimensional space, the Hyperplane, which is the line.

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 > 0$$

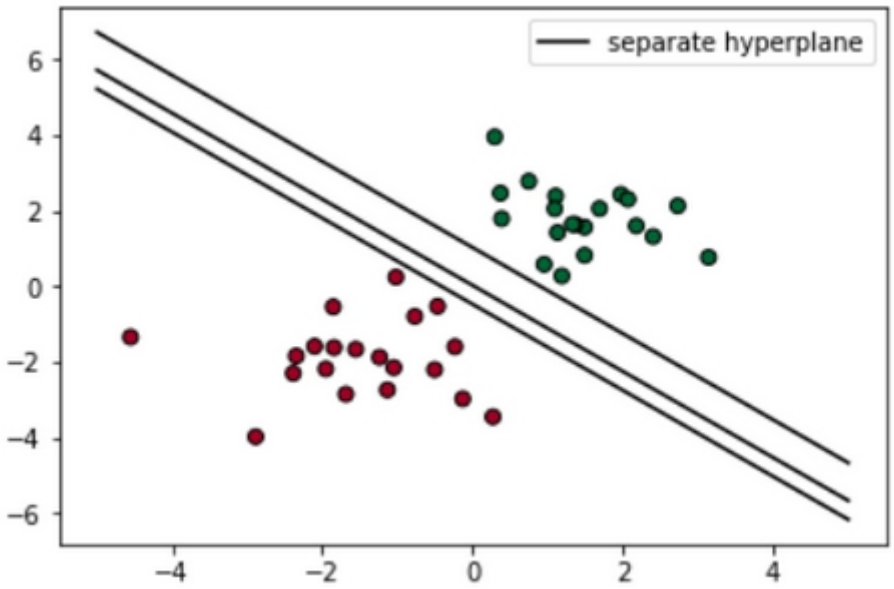
The dots above this line, are those x1, x2 satisfy the formula above

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 < 0$$

The dots below this line, similar logic.

Что такое разделяющая гиперплоскость?

Предполагая, что метка y равна 1 (для зеленого) или -1 (для красного), все эти три линии ниже являются разделяющими гиперплоскостями. Потому что все они имеют одно и то же свойство — над чертой выделено зеленым цветом; ниже линии, красный.



Это свойство можно снова записать в математике следующим образом:

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 > 0 \; if \; y = 1 \; theGreen$$

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 < 0 \; if \; y = -1 \; theRed$$

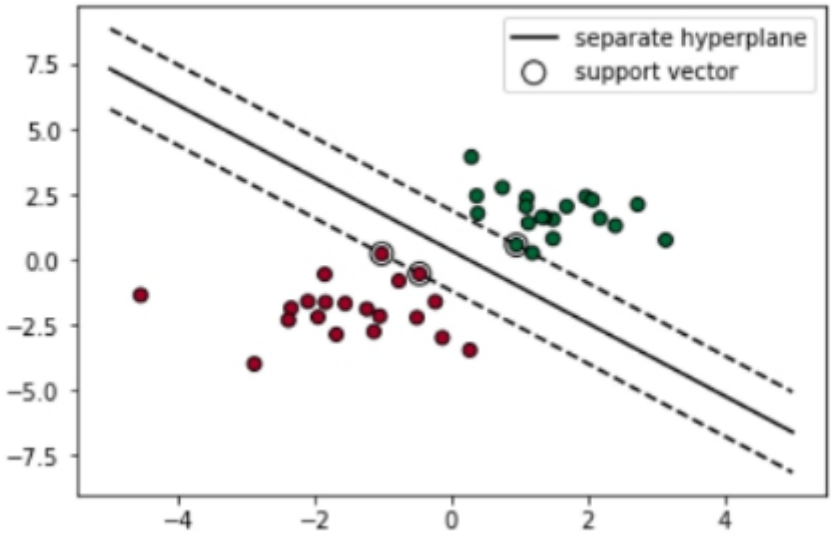
Если мы далее обобщим эти два в одно, то получится:

$$y * (\beta_0 + \beta_1 * x_1 + \beta_2 * x_2) > 0$$

Отдельное ограничение гиперплоскости математически записано выше. В идеальном сценарии — сценарии с линейной разделимостью это ограничение может быть удовлетворено с помощью SVM. Но в неразделимом сценарии нам нужно будет его ослабить.

Так что же такое маржа?

1. Допустим, у нас есть гиперплоскость — линия X
2. рассчитайте перпендикулярное расстояние от всех этих 40 точек до линии X, это будет 40 различных расстояний
3. Из 40, наименьшее расстояние, это наш запас!



Расстояние между обеими сторонами пунктирной линии и сплошной линией является полем. Мы можем думать об этой оптимальной линии как о средней линии самого широкого растяжения, которое мы можем иметь между красными и зелеными точками.

Подводя итог, SVM в линейных отделимых случаях:

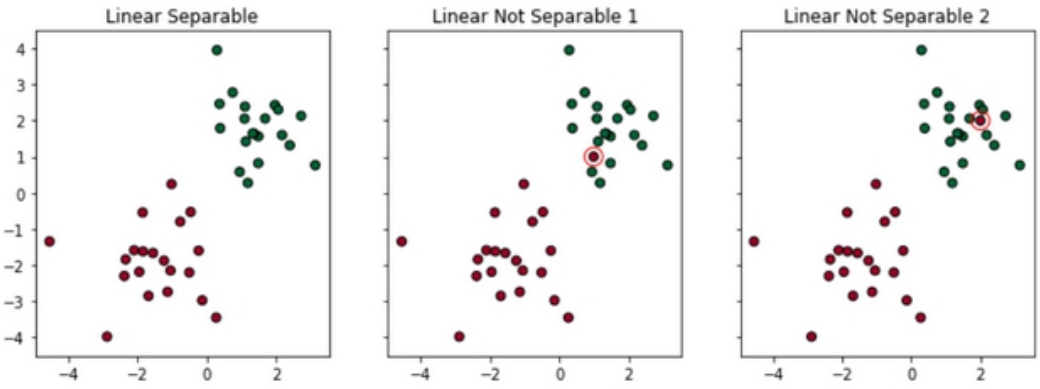
1. Ограничьте/убедитесь, что каждое наблюдение находится на правильной стороне гиперплоскости
2. Подберите оптимальную линию так, чтобы расстояние от ближайших точек до Гиперплоскости, так называемое поле, было максимальным.

SVM в линейных несепарабельных случаях

В линейно разделимом случае SVM пытается найти гиперплоскость, которая максимизирует запас, при условии, что оба класса классифицированы правильно. Но на самом деле наборы данных, вероятно, никогда не бывают линейно разделимыми, поэтому условие 100% правильной классификации гиперплоскостью никогда не будет выполнено.

SVM решает нелинейно разделимые случаи, вводя две концепции: Soft Margin и Kernel Tricks.

Давайте рассмотрим пример. Если я добавлю одну красную точку в зеленый кластер, набор данных больше не станет линейным, неразделимым. Два решения этой проблемы:

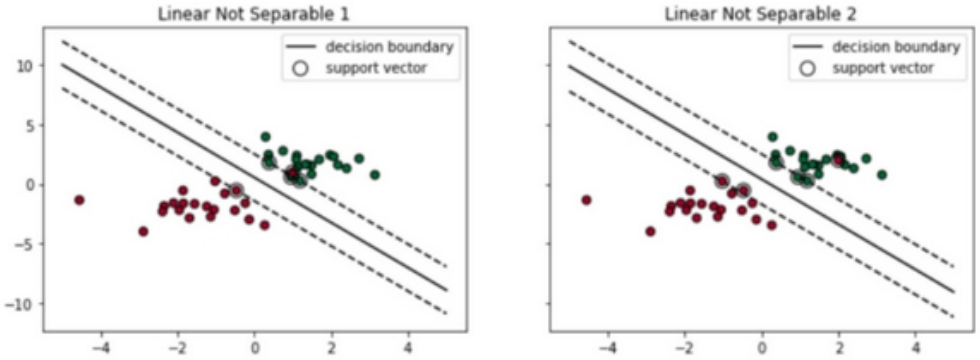


1. Soft Margin: попробуйте найти линию для разделения, но допустите одну или несколько неправильно классифицированных точек (например, точки, обведенные красным).
2. Kernal Trick: попытайтесь найти нелинейную границу решения.

Soft Margin

SVM допускает два типа ошибочных классификаций в рамках мягкой маржи:

1. Точка находится не на той стороне границы решения, но на правильной стороне/на поле (показано слева)
2. Точка находится на неправильной стороне границы решения и на неправильной стороне поля (показано справа)

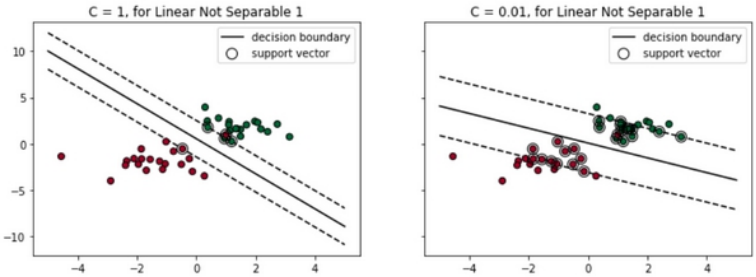


Применяя Soft Margin, SVM допускает неправильную классификацию нескольких точек и пытается сбалансировать компромисс между поиском линии, которая максимизирует маржу и минимизирует ошибочную классификацию.

Степень толерантности

Насколько допуск (мягкий) мы хотим дать при нахождении границы решения, является важным гиперпараметром для SVM (как линейных, так и нелинейных решений). В Sklearn он представлен как штрафной термин — «C». Чем больше C, тем больше штрафа получает SVM, когда допускает неправильную классификацию. Следовательно, чем уже граница и от меньшего количества опорных векторов будет зависеть граница решения.

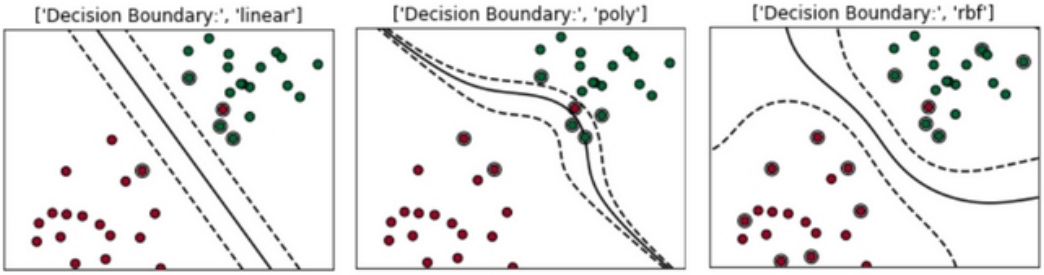
```
# Default Penalty/Default Tolerance
clf = svm.SVC(kernel='linear', C=1)
# Less Penalty/More Tolearance
clf2 = svm.SVC(kernel='linear', C=0.01)
```



Kernel Trick

Что делает Kernel Trick, так это использует существующие функции, применяет некоторые преобразования и создает новые функции. Эти новые функции являются ключом к SVM для нахождения нелинейной границы решения.

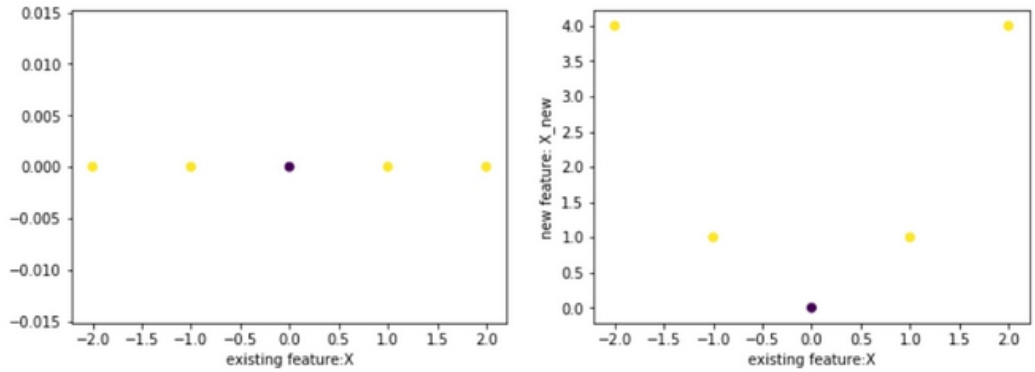
В Sklearn — svm.SVC() мы можем выбрать «linear», «poly», «rbf», «sigmoid», «precomputed» или callable в качестве нашего ядра/преобразования. Я приведу примеры двух самых популярных ядер — полиномиального и радиальной базисной функции (РБФ).



Polinomial Kernel

Думайте о полиномиальном ядре как о преобразователе/процессоре для создания новых функций путем применения полиномиальной комбинации всех существующих функций.

Чтобы проиллюстрировать преимущества применения полиномиального преобразователя, давайте рассмотрим простой пример:



Существующая функция: X = np.array([-2,-1,0, 1,2])
Метка: Y = np.array([1,1,0,1,1])
нам невозможно найти линию, разделяющую желтую (1) и фиолетовую (0) точки (показаны слева).

Но если мы применим преобразование X^2, чтобы получить:
Новая функция: X = np.array([4,1,0, 1,4])
Сочетая существующую и новую функцию, мы, безусловно, можем провести линию, отделяющую желто-фиолетовые точки (показаны справа).

Машина опорных векторов с полиномиальным ядром может генерировать нелинейную границу решения, используя эти полиномиальные функции.

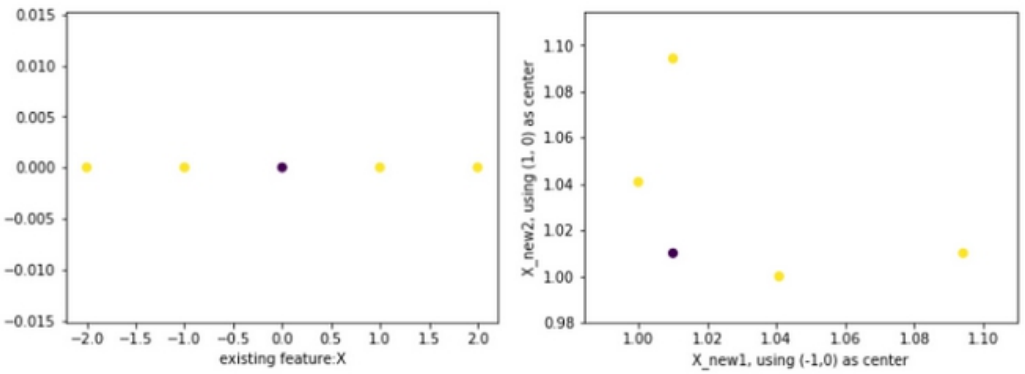
Ядро радиальной базисной функции (RBF)

Думайте о ядре Radial Basis Function как о преобразователе/процессоре для создания новых функций путем измерения расстояния между всеми остальными точками до определенной точки/точек — центров. Самым популярным/базовым ядром RBF является гауссова радиальная базисная функция:

$$\phi(x, center) = exp(-\gamma \|x - center\|^2)$$

гамма (γ) контролирует влияние новых признаков — Φ(x, center) на границу решения. Чем выше гамма, тем большее влияние признаки будут иметь на границу решения, тем более подвижной будет граница.

Чтобы проиллюстрировать преимущество применения гауссовой rbf (gamma = 0,1), давайте используем тот же пример:



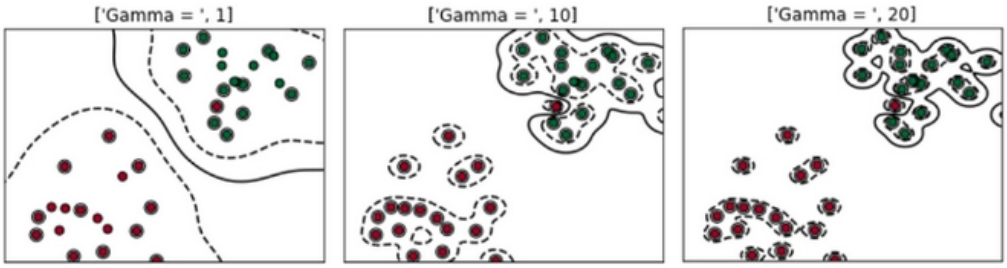
Существующая функция: X = np.array([-2,-1,0, 1,2])
Метка: Y = np.array([1,1,0,1,1])
Опять же, мы не можем найти линию, разделяющую точки (слева).

Но если мы применим преобразование Гаусса RBF с использованием двух центров (-1,0) и (2,0), чтобы получить новые функции, мы сможем провести линию, чтобы разделить желто-фиолетовые точки (справа):
Новая функция 1: X_new1 = массив ([1.01, 1.00, 1.01, 1.04, 1.09])
Новая функция 2: X_new2 = массив ([1.09, 1.04, 1.01, 1.00, 1.01])

```
# Note for how we get New Feature 1, e.g. 1.01:
Φ(x1,center1) = np.exp(np.power(-(gamma*(x1-center1)),2)) = 1.01
# gamma = 0.1
# center1 = [-1,0]
# x1 = [-2,0]
```

Подобно штрафному термину — C в Soft Margin, Gamma — это гиперпараметр, который мы можем настроить, когда используем SVM с ядром.

```
# Gamma is small, influence is small
clf = svm.SVC(kernel='rbf', Gamma=1)
# Gamma gets bigger, influence increase, the decision boundary get
wiggled
clf2 = svm.SVC(kernel='rbf', Gamma=10)
# Gamma gets too big, influence too much, the decision boundary get
too wiggled
clf3 = svm.SVC(kernel='rbf', Gamma=20)
```



Подводя итог, SVM в линейных несепарабельных случаях:

1. Объединив вместе Soft Margin (допустимость ошибочных классификаций) и Kernel Trick, машина опорных векторов может структурировать границу решения для линейных неразделимых случаев.

2. Гиперпараметры, такие как C или Gamma, контролируют, насколько подвижной может быть граница решения SVM.

3. чем выше C, тем больше штрафа за SVM за неправильную классификацию, и, следовательно, тем меньше будет покачивание границы решения.
4. чем выше гамма, тем большее влияние точки данных объекта будут иметь на границу решения, тем больше будет покачивание границы.