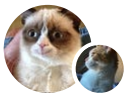
Support Vector Machine — Explained (Soft Margin/Kernel Tricks)



Lujing Chen · Follow

Published in Bite-sized Machine Learning

6 min read · Dec 17, 2018

In this blog — support vector machine Part 2, we will go further into solving the non-linearly separable problem by introducing two concepts:

1. **Soft Margin**
2. **Kernel Tricks**

Hopefully, you get the idea of what support vector machine is and how it works in the linear separable cases during my previous blog.

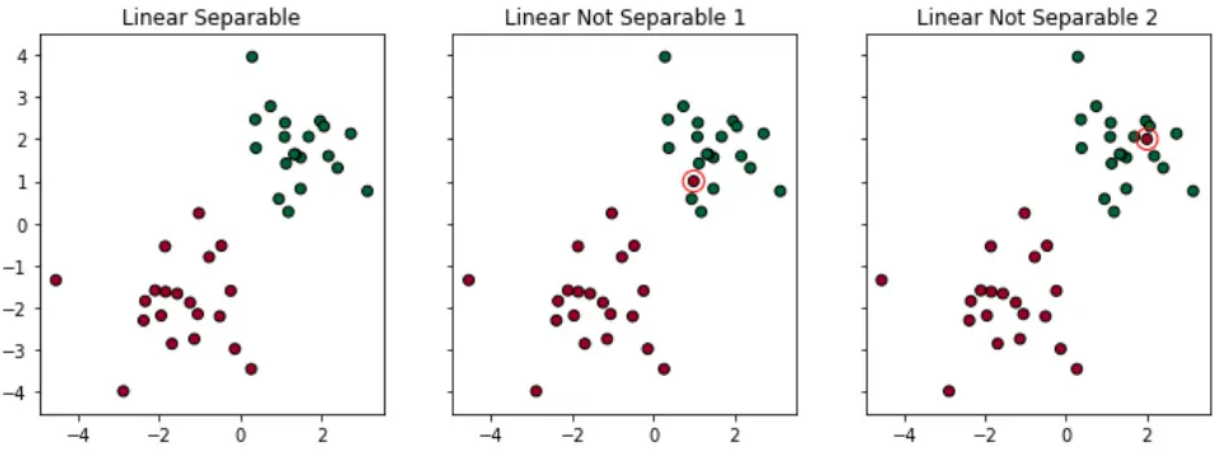
Photo Source: https://theculturetrip.com/north-america/usa/florida/articles/best-shopping-streets-miami/

In the linearly separable case, Support Vector Machine is trying to find the line that maximizes the margin (think of a street), which is the distance between those closest dots to the line. SVM stretches this 'street' to the max and the decision boundary lays right in the middle, with the condition that both classes are classified correctly, in other words, the dataset is linearly separable, but in real life, we rarely find the dataset which is linearly separable.

For example, in scatter plot shown on the left below, if I add one red dot in the green cluster, the dataset becomes non-linearly separable anymore. Two solutions to this problem:

1. we still can try to find a line to separate red and green dots, but we tolerate one or few misclassified dots (e.g. the dots circled in red). This is called the Soft Margin.

2. Or we can try to find a non-linear decision boundary to separate red and green dots. This is called the Kernel Trick.



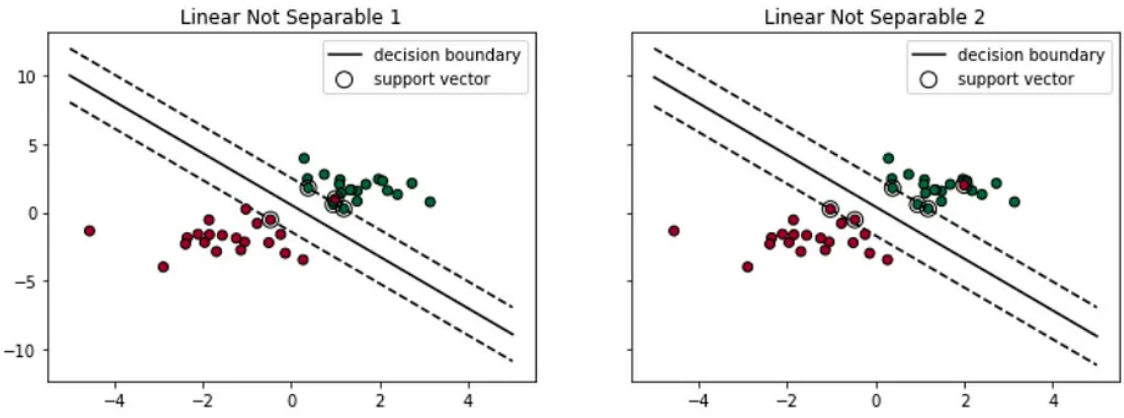
Soft Margin

What Soft Margin does is

1. it tolerates a few dots to get misclassified
2. it tries to balance the trade-off between finding a line that maximizes the margin and minimizes the misclassification.

Two types of misclassifications can happen:

1. The dot is on the wrong side of the decision boundary but on the correct side/ on the margin (shown in left)
2. The dot is on the wrong side of the decision boundary and on the wrong side of the margin (shown in right)

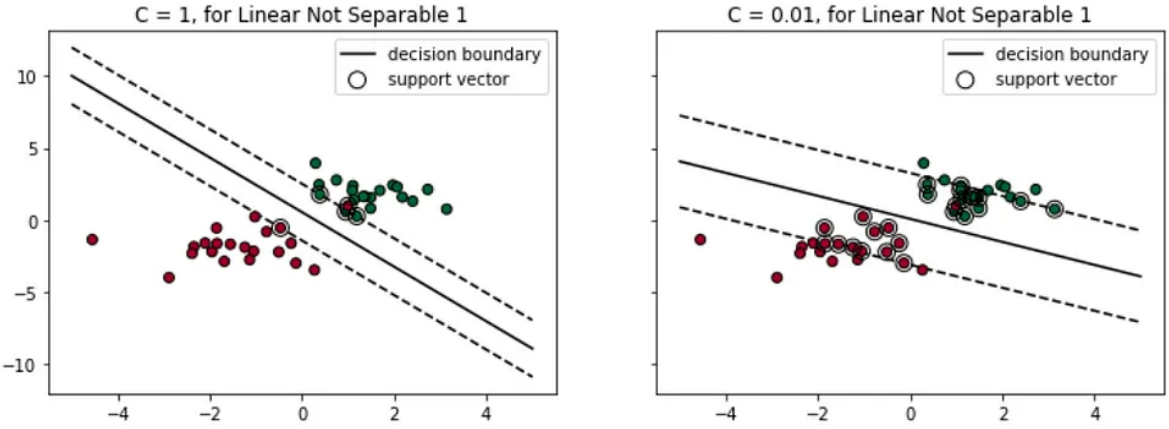


Either case, the support vector machine tolerates those dots to be misclassified when it tries to find the linear decision boundary.

Degree of tolerance

How much tolerance we want to set when finding the decision boundary is an important hyper-parameter for the SVM (both linear and nonlinear solutions). In Sklearn, it is represented as the penalty term — 'C'. The bigger the C, the more penalty SVM gets when it makes misclassification. Therefore, the narrower the margin is and fewer support vectors the decision boundary will depend on.

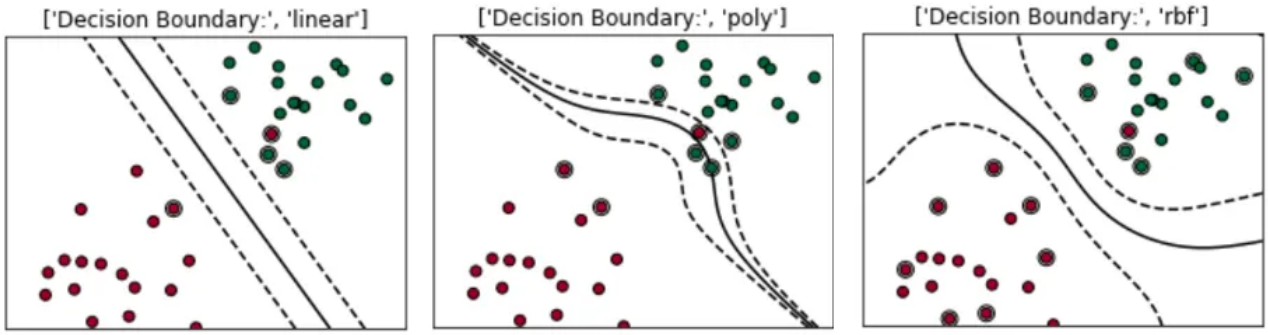
```
# Default Penalty/Default Tolerance
clf = svm.SVC(kernel='linear', C=1)
# Less Penalty/More Tolearance
clf2 = svm.SVC(kernel='linear', C=0.01)
```



Kernel Trick

What Kernel Trick does is it utilizes existing features, applies some transformations, and create new features. Those new features are the key for SVM to find the nonlinear decision boundary.

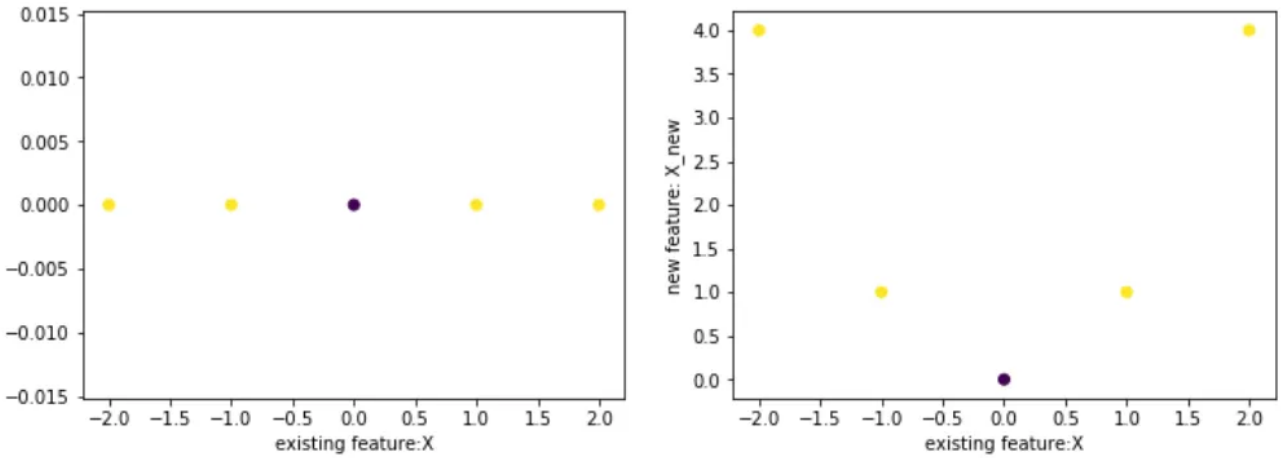
In Sklearn — svm.SVC(), we can choose 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable as our kernel/transformation. I will give examples of the two most popular kernels — polynomial and Radial Basis Function(RBF).



Polynomial Kernel

Think of the polynomial kernel as a transformer/processor to generate new features by applying the polynomial combination of all the existing features.

To illustrate the benefit of applying a polynomial transformer, let's use a simple example:



Existing Feature: X = np.array([-2,-1,0, 1,2])
Label: Y = np.array([1,1,0,1,1])
it's impossible for us to find a line to separate the yellow (1)and purple (0) dots (shown below on the left).

But, if we apply transformation X^2 to get:
New Feature: X = np.array([4,1,0, 1,4])
By combing the existing and new feature, we can certainly draw a line to separate the yellow purple dots (shown on the right).

Support vector machine with a polynomial kernel can generate a non-linear decision boundary using those polynomial features.

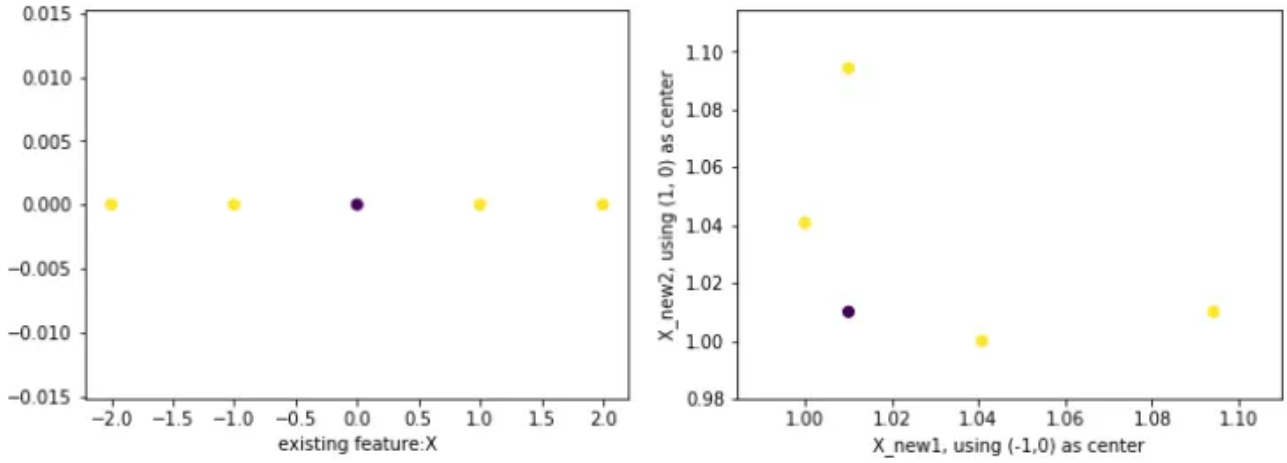
Radial Basis Function (RBF) kernel

Think of the Radial Basis Function kernel as a transformer/processor to generate new features by measuring the distance between all other dots to a specific dot/dots — centers. The most popular/basic RBF kernel is the Gaussian Radial Basis Function:

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

gamma (γ) controls the influence of new features — **Φ(x, center)** on the decision boundary. The higher the gamma, the more influence of the features will have on the decision boundary, more wiggling the boundary will be.

To illustrate the benefit of applying a Gaussian rbf (gamma = 0.1), let's use the same example:



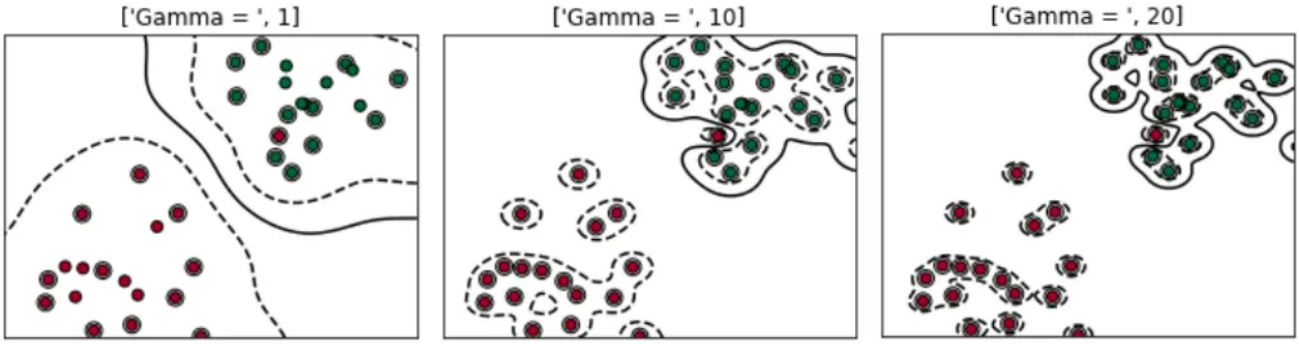
Existing Feature: X = np.array([-2,-1,0, 1,2])
Label: Y = np.array([1,1,0,1,1])
Again, it's impossible for us to find a line to separate the yellow (1) and purple (1) dots (on left hand).

But, if we apply Gaussian RBF transformation using two centers (-1,0) and (2,0) to get new features, we will then be able to draw a line to separate the yellow purple dots (on the right):
New Feature 1: X_new1 = array([1.01, 1.00, 1.01, 1.04, 1.09])
New Feature 2: X_new2 = array([1.09, 1.04, 1.01, 1.00, 1.01])

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

Similar to the penalty term — C in the soft margin, Gamma is a hyperparameter that we can tune for when we use SVM.

```
# 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)
```



To sum up:

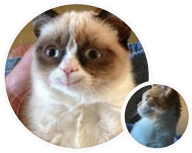
- By combining the **soft margin** (tolerance of misclassification) and **kernel trick** together, Support Vector Machine is able to structure the decision boundary for linearly non-separable cases.
- Hyper-parameters like C or Gamma control how wiggling the SVM decision boundary could be.

1. the higher the C, the more penalty SVM was given when it misclassified, and therefore the less wiggling the decision boundary will be
2. the higher the gamma, the more influence the feature data points will have on the decision boundary, thereby the more wiggling the boundary will be

Please feel free to leave any comment, question or suggestion if you have any thought related to this post.

Please clap if you feel the content is useful! :) Thanks you!

Machine Learning | Svm | Support Vector Machine | Kernel | Python



Follow

Written by Lujing Chen

603 Followers · Editor for Bite-sized Machine Learning

MLE at Apple (Ex-Amazon)/ a lifelong learner

More from Lujing Chen and Bite-sized Machine Learning