Support Vector Machine

Yilin Wu

DigiPen Institute of Technology

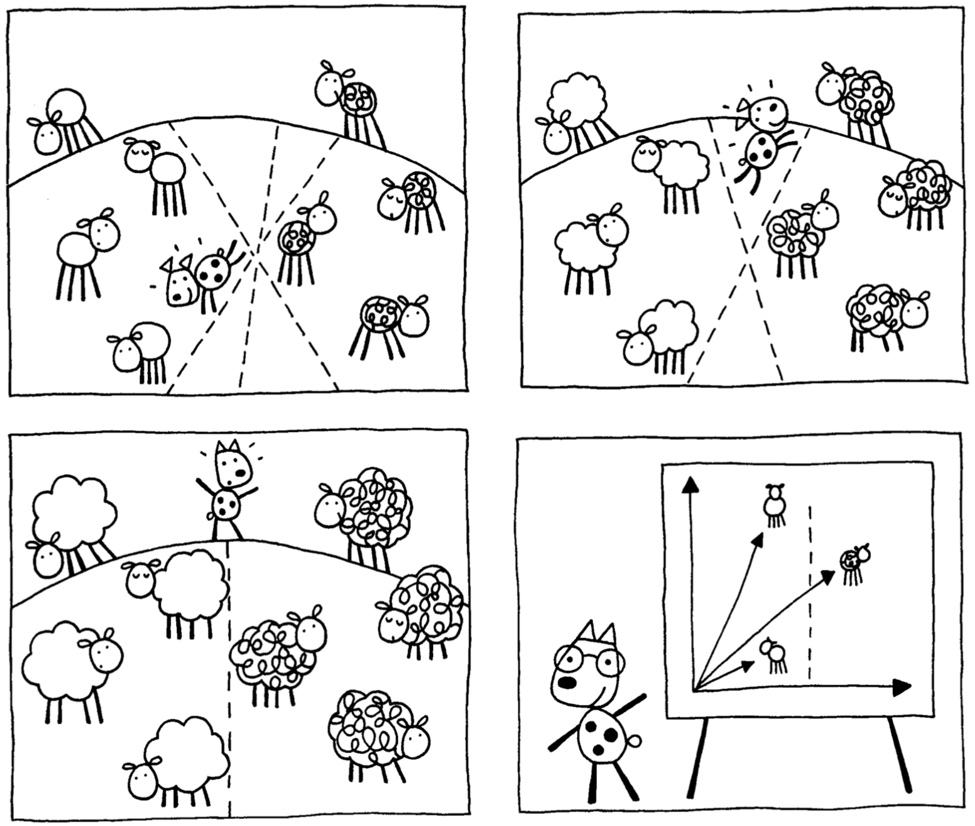
Advanced Machine Learning

+ Unit 1 – Support Vector Machine
+ Unit 2 – Clustering-1
+ Unit 3 – Clustering-2
+ Unit 4 – Dimensionality Reduction

Outline

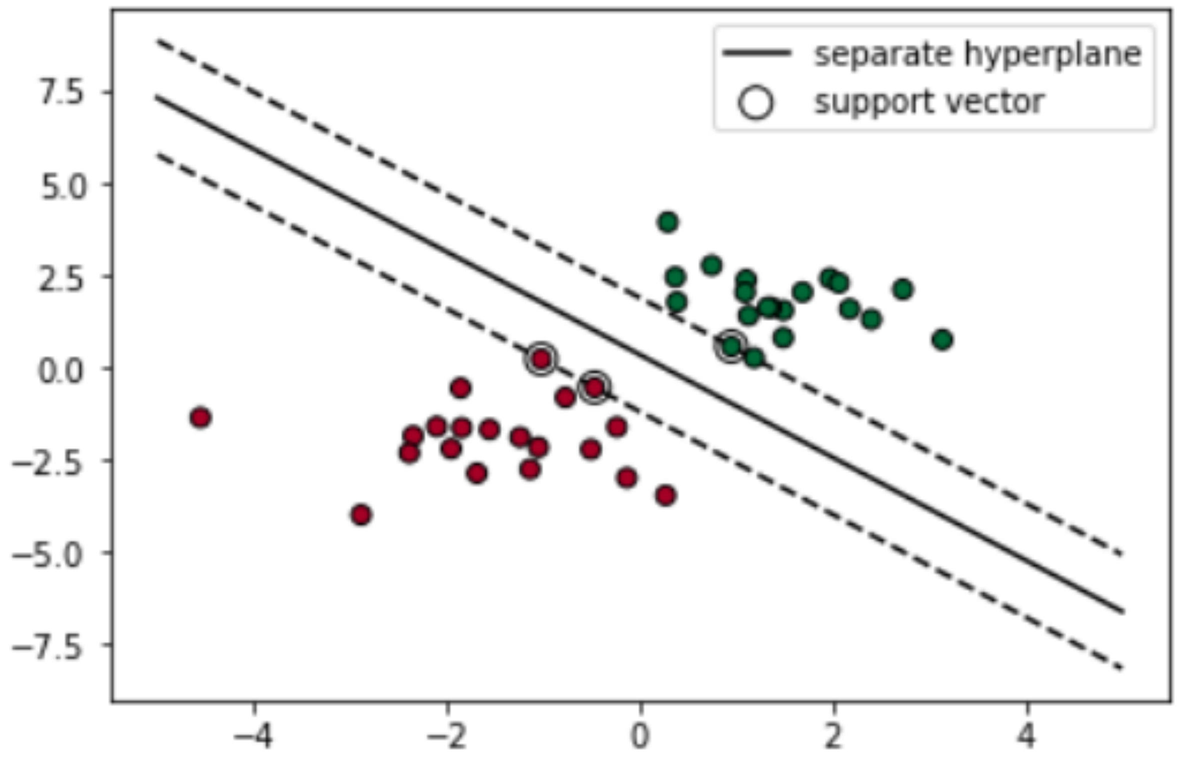
- What is the support vector machine
- How does it work in linear separable scenarios
- How does it work in linear non-separable scenarios

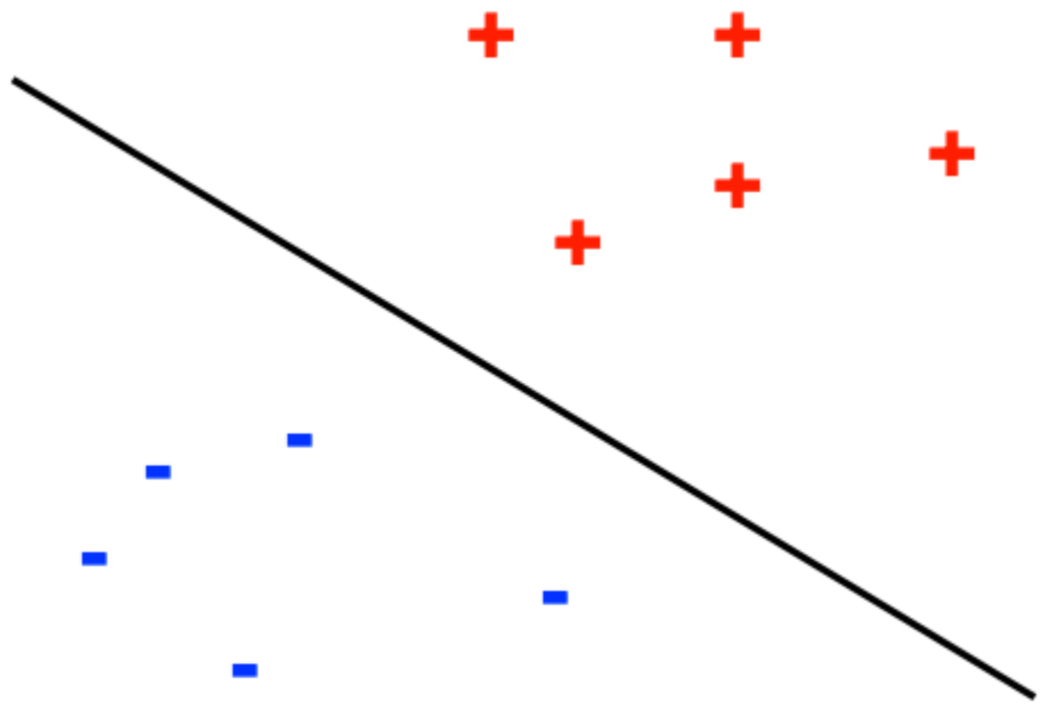
Support Vector Machine



. . . the story of the sheep dog who was herding his sheep, and serendipitously invented both large margin classification and Sheep Vectors. . .

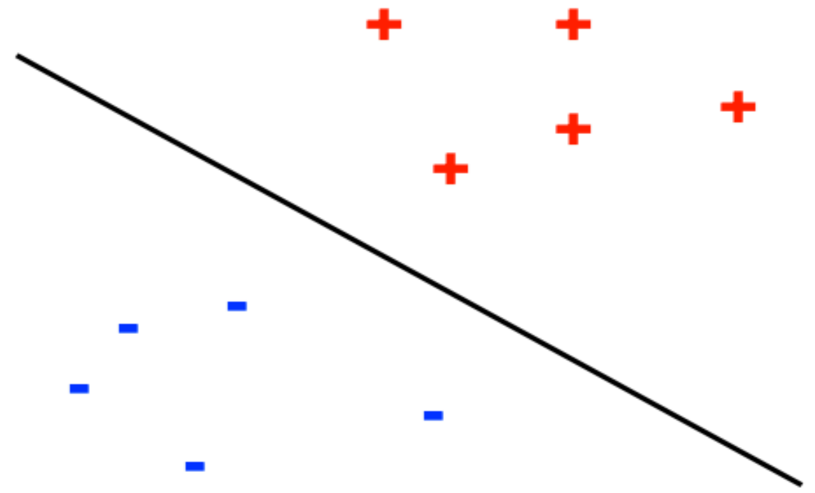
What is Support Vector Machine (Classifier)



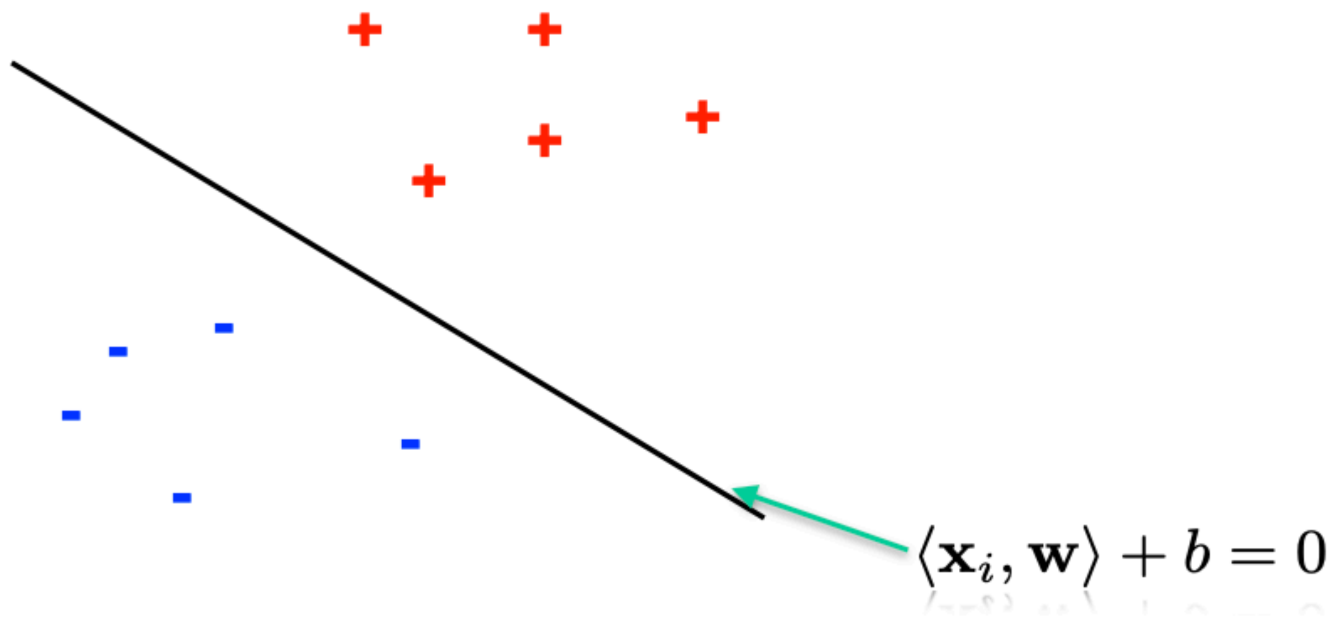


- Support Vector Machine (the "road machine") is responsible for finding the decision boundary to separate different classes and maximize the margin.
- Margins are the (perpendicular) distances between the line and those dots closest to the line.

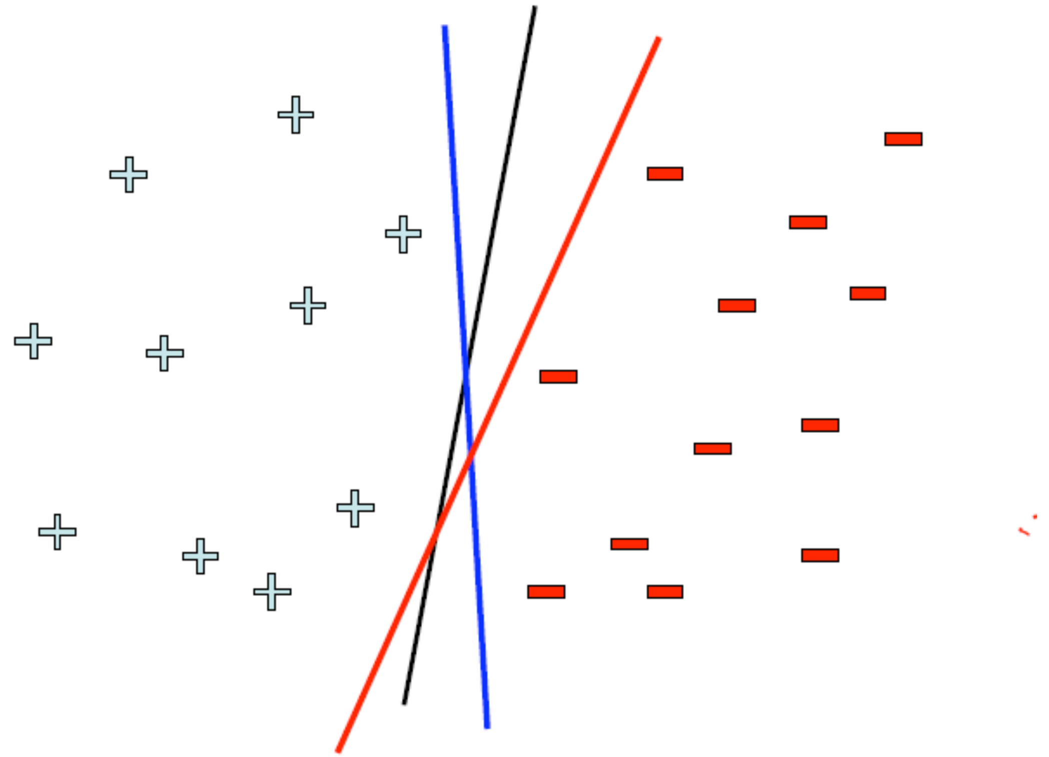
Find a linear separator with the largest margin



Find a linear separator with the largest margin



Linear Classifier – which line is better?



Pick the one with the largest margin!

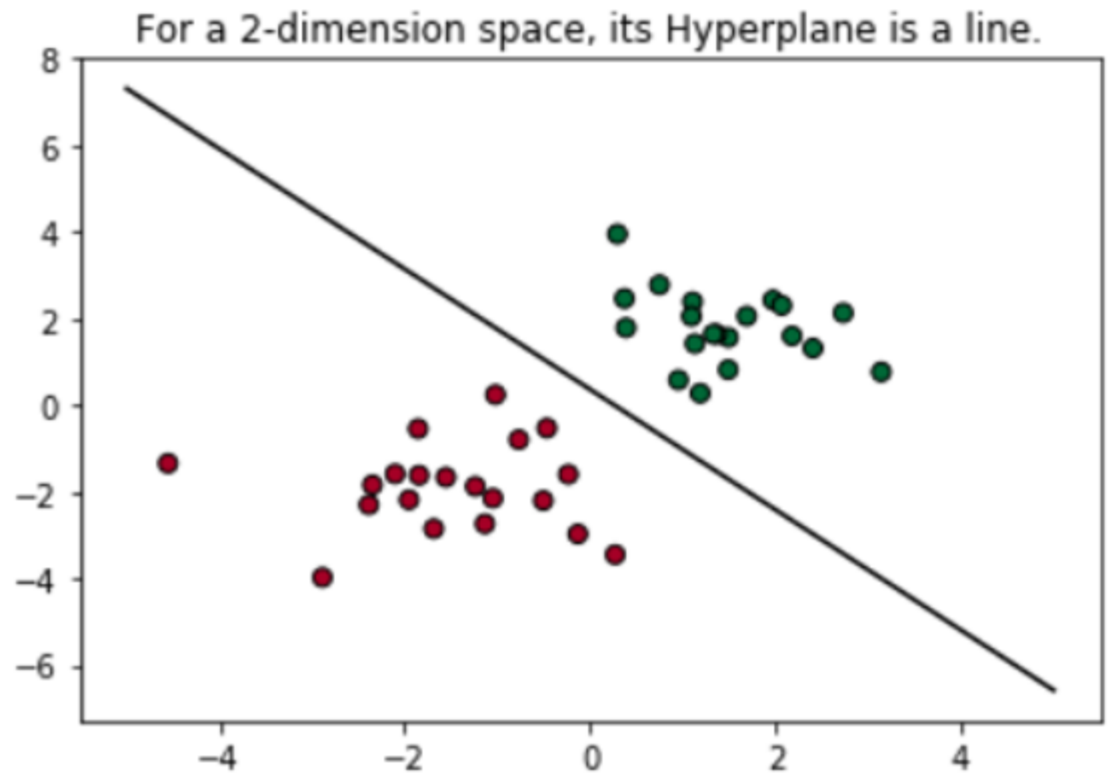
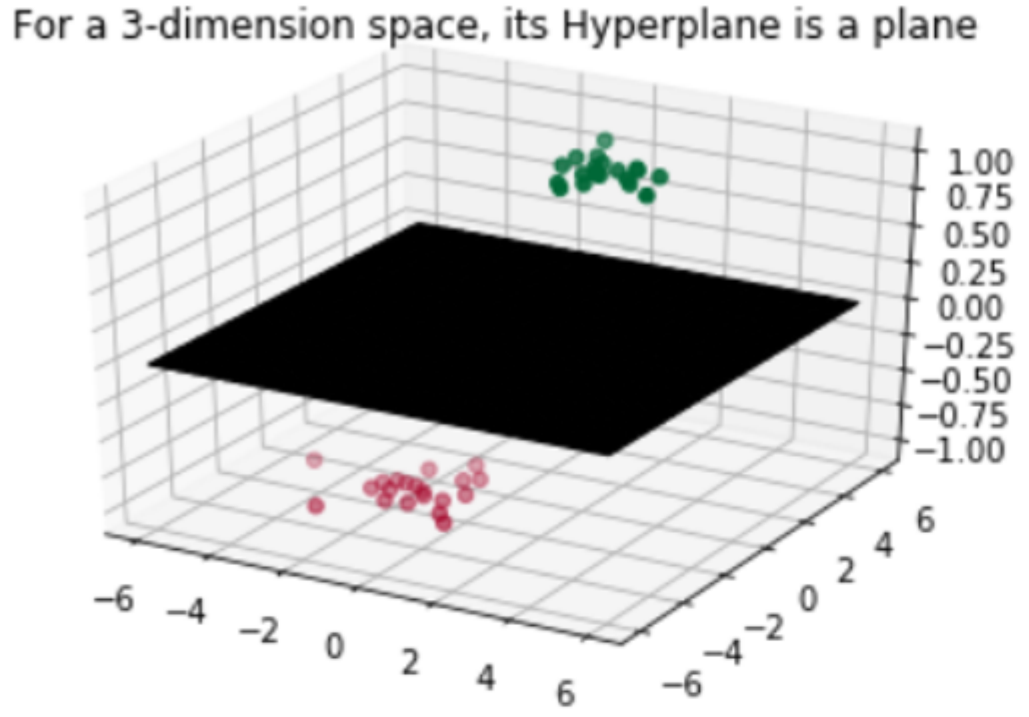
SVM needs to find the optimal line with the constraint of correctly classifying either class:

1.Follow the constraint: only look into the **separate hyperplanes** (e.g. separate lines), hyperplanes that classify classes correctly

2.Conduct optimization: pick up the one that maximizes the **margin**

Hyperplane

+ Hyperplane is an (n minus 1)-dimensional subspace for an n-dimensional space.
+ For a 2-dimension space, its hyperplane will be 1-dimension, which is just a line.
+ For a 3-dimension space, its hyperplane will be 2-dimension, which is a plane that slice the cube.


For a 2-dimension space, its Hyperplane is a line.
8
6
4
2
0
−2
−4
−6
−4
−2
0
2
4

For a 3-dimension space, its Hyperplane is a plane
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−6
−4
−2
0
2
4
6
−6
−4
−2
0
2
4
6

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

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.

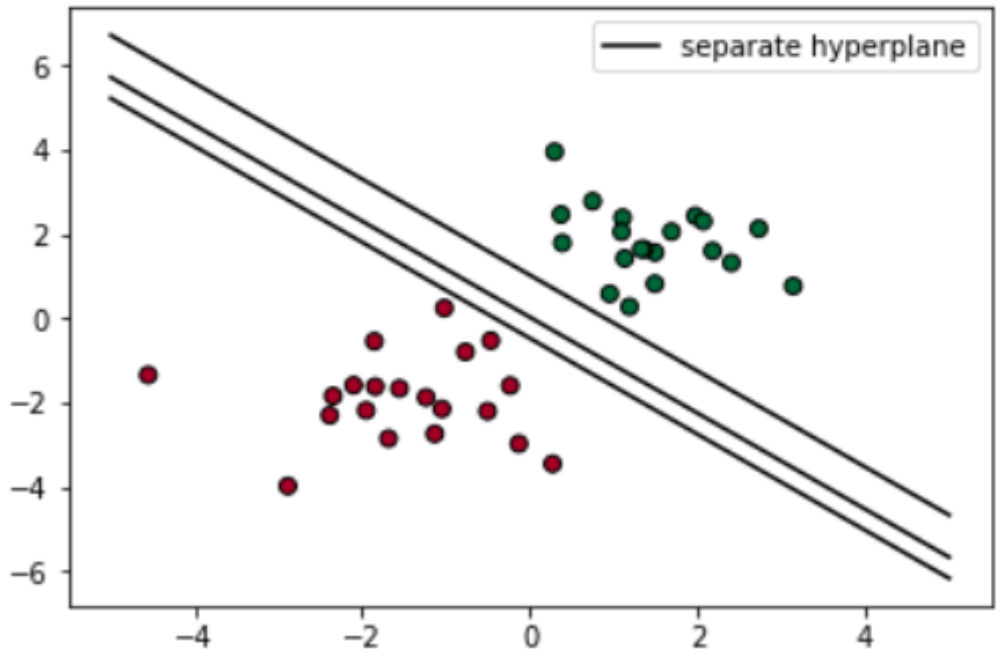
$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 > 0$$

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.

Assuming the label y is either 1 (for green) or -1 (for red), all those three lines below are separating hyperplanes. Because they all share the same property — above the line, is green; below the line, is red.



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

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

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

what is **margin?**

- Let’s say we have a hyperplane — line X
- calculate the perpendicular distance from all those 40 dots to line X, it will be 40 different distances
- Out of the 40, the smallest distance, that’s our margin!

+ Margin = Distance of closest examples from the decision line/hyperplane

SVM in the linear separable cases:

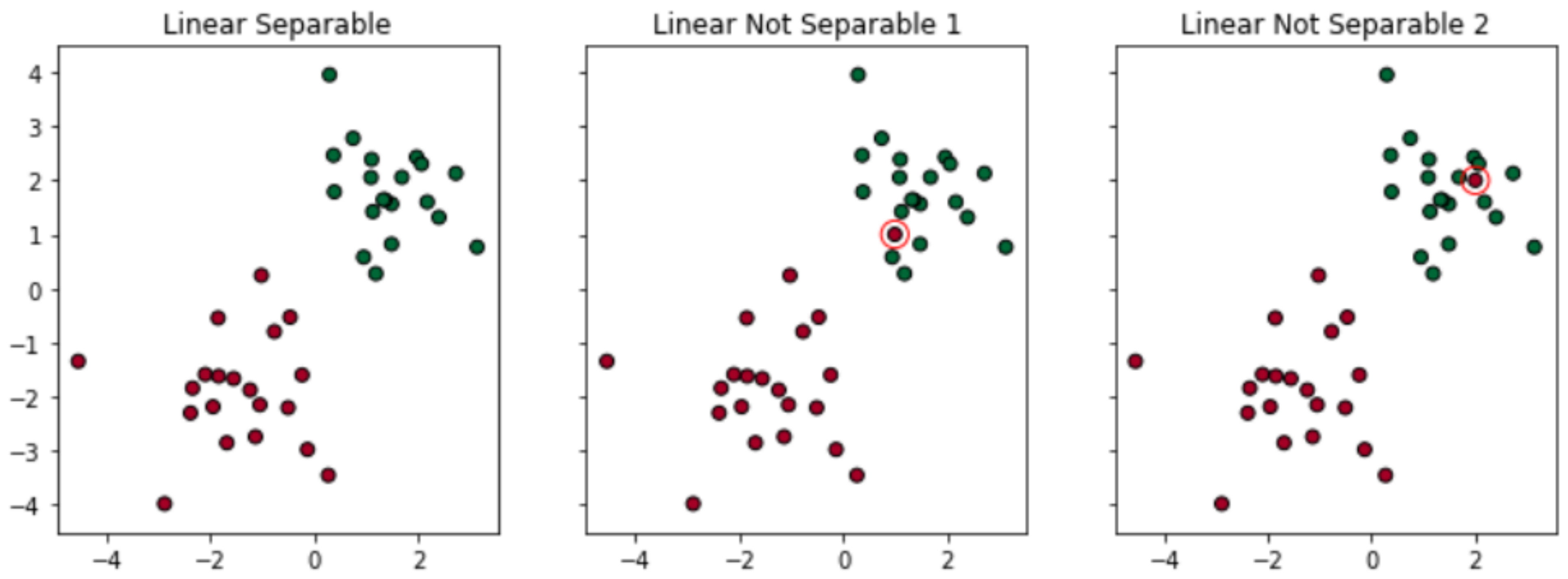
- Constrain/ensure that each observation is on the correct side of the Hyperplane
- Pick up the optimal line so that the distance from those closest dots to the Hyperplane, so-called margin, is maximized

SVM in linear non-separable cases

In the linearly separable case, SVM is trying to find the hyperplane that maximizes the margin, with the condition that both classes are classified correctly.

But in reality, datasets are probably never linearly separable, so the condition of 100% correctly classified by a hyperplane will never be met.

Use: **Soft Margin or Kernel Tricks**



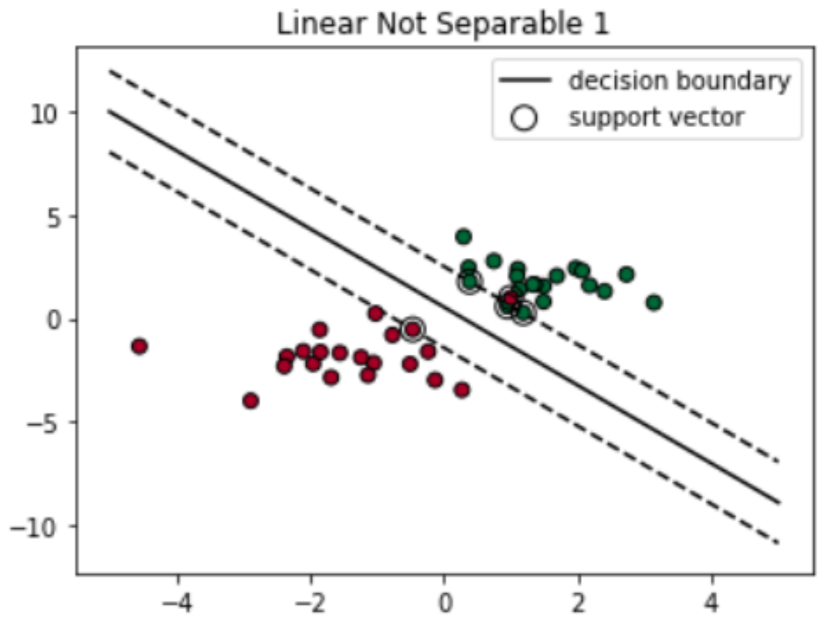
1. **Soft Margin:** try to find a line to separate, but tolerate one or few misclassified dots (e.g. the dots circled in red)
2. **Kernel Trick:** try to find a non-linear decision boundary

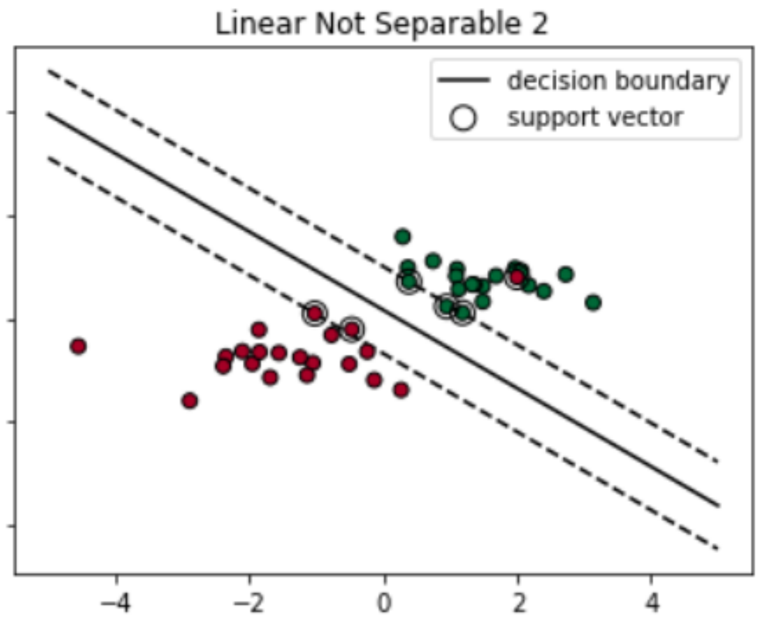
Soft Margin

Two types of misclassifications are tolerated by SVM under soft margin:

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)

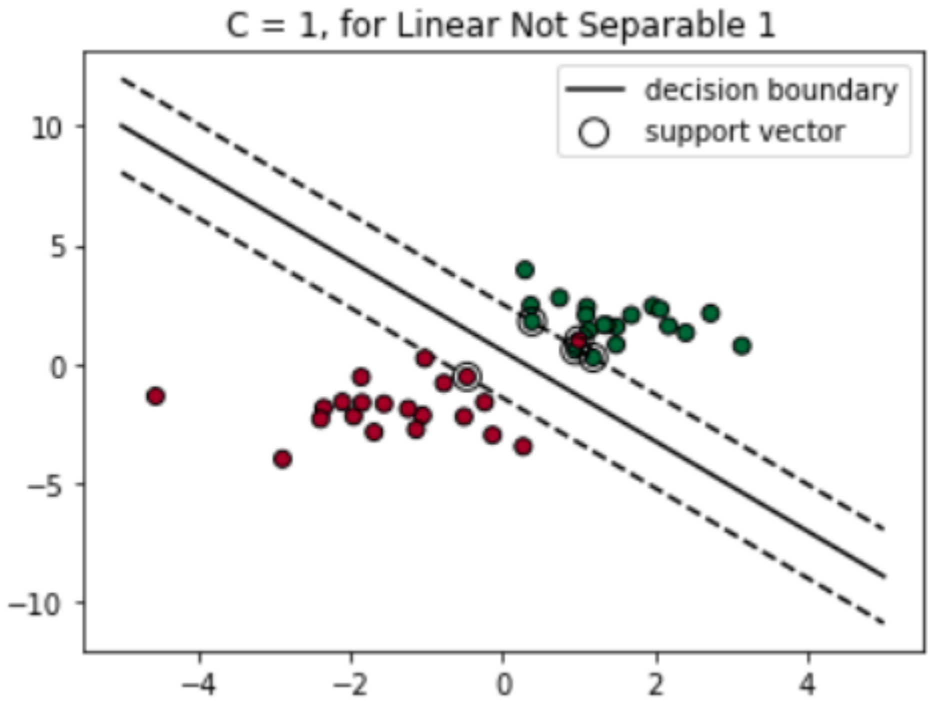


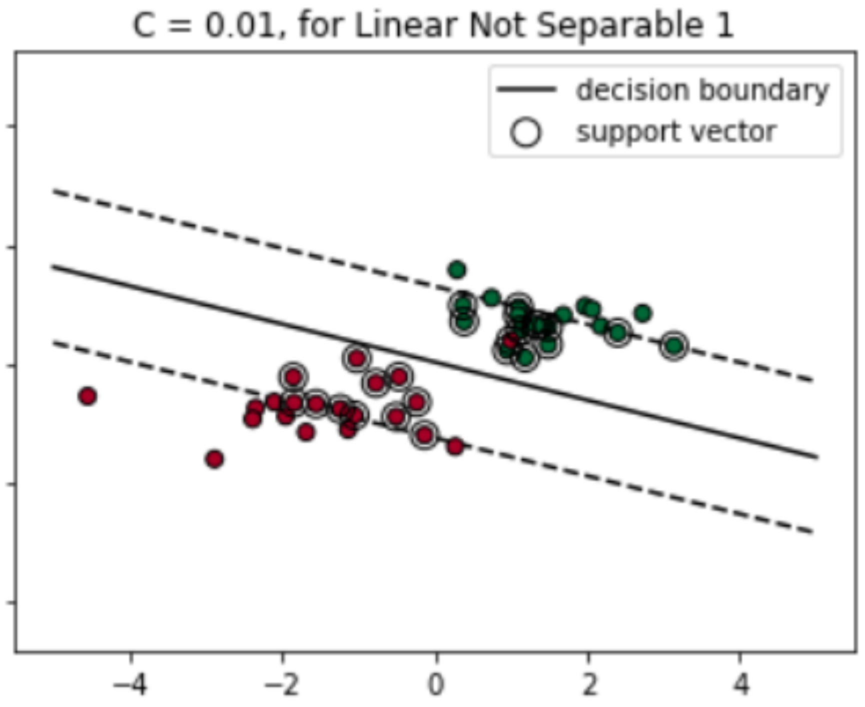


Applying Soft Margin, SVM tolerates a few dots to get misclassified and tries to balance the trade-off between finding a line that maximizes the margin and minimizes the misclassification.

Degree of tolerance
How much tolerance(soft) we want to give 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.

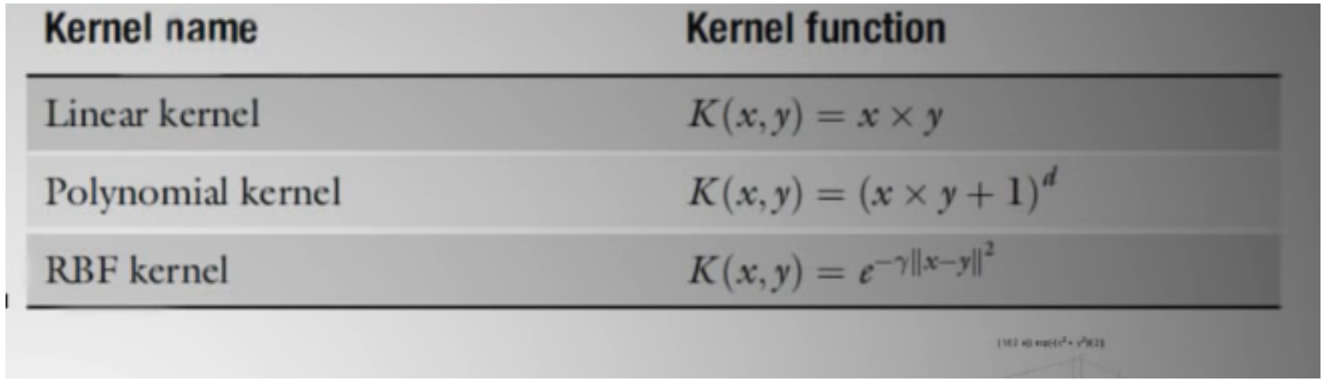
```
# Default Penalty/Default Tolerance
clf = svm.SVC(kernel='linear', C=1)
# Less Penalty/More Tolearance
clf2 = svm.SVC(kernel='linear', C=0.01)
```



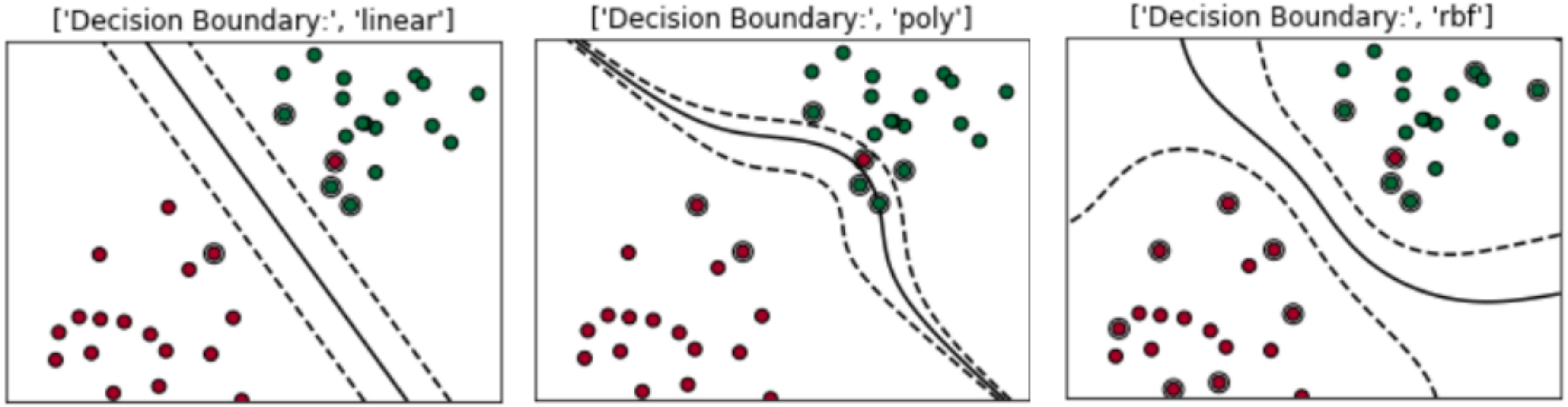


Kernel Trick

What Kernel Trick does is it utilizes existing features, applies some transformations, and creates 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.

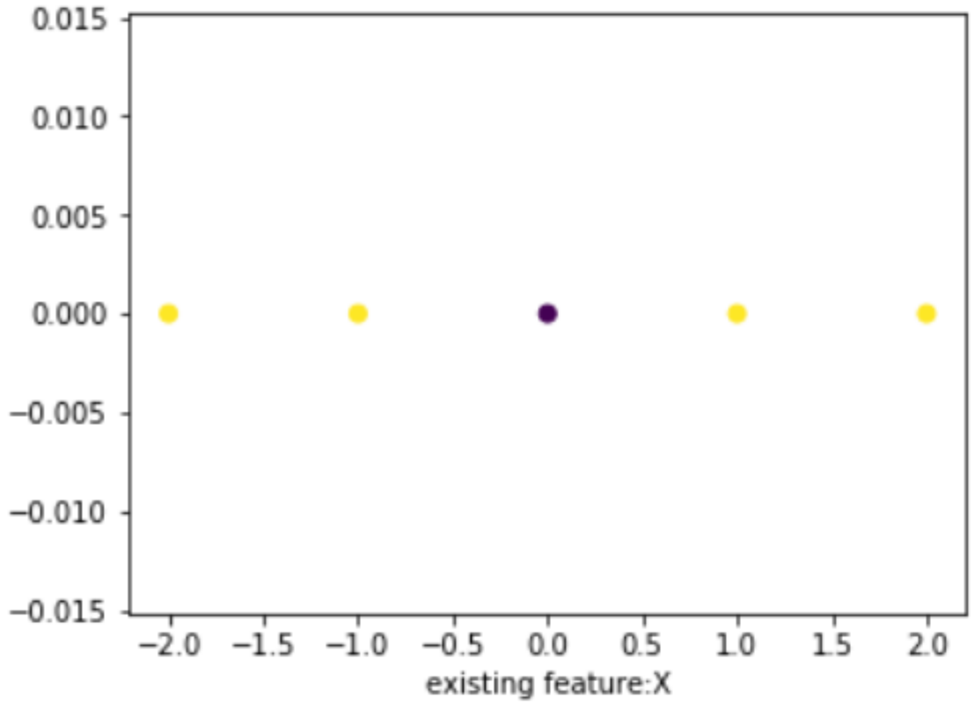


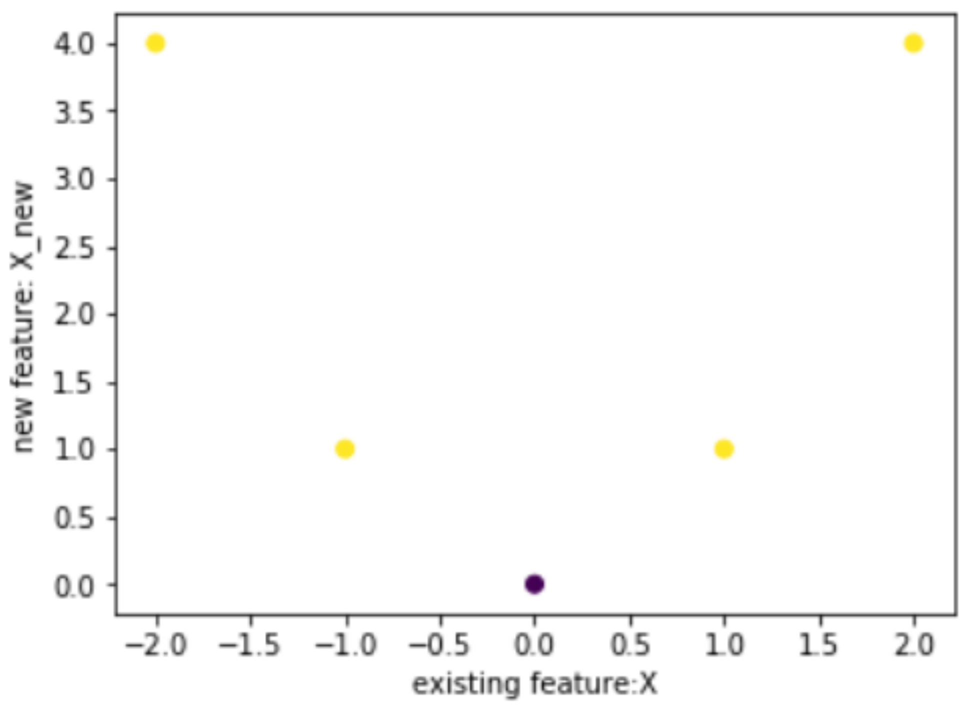
Kernel name	Kernel function
Linear kernel	$K(x,y) = x \times y$
Polynomial kernel	$K(x,y) = (x \times y + 1)^d$
RBF kernel	$K(x,y) = e^{-\gamma\|x-y\|^2}$



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





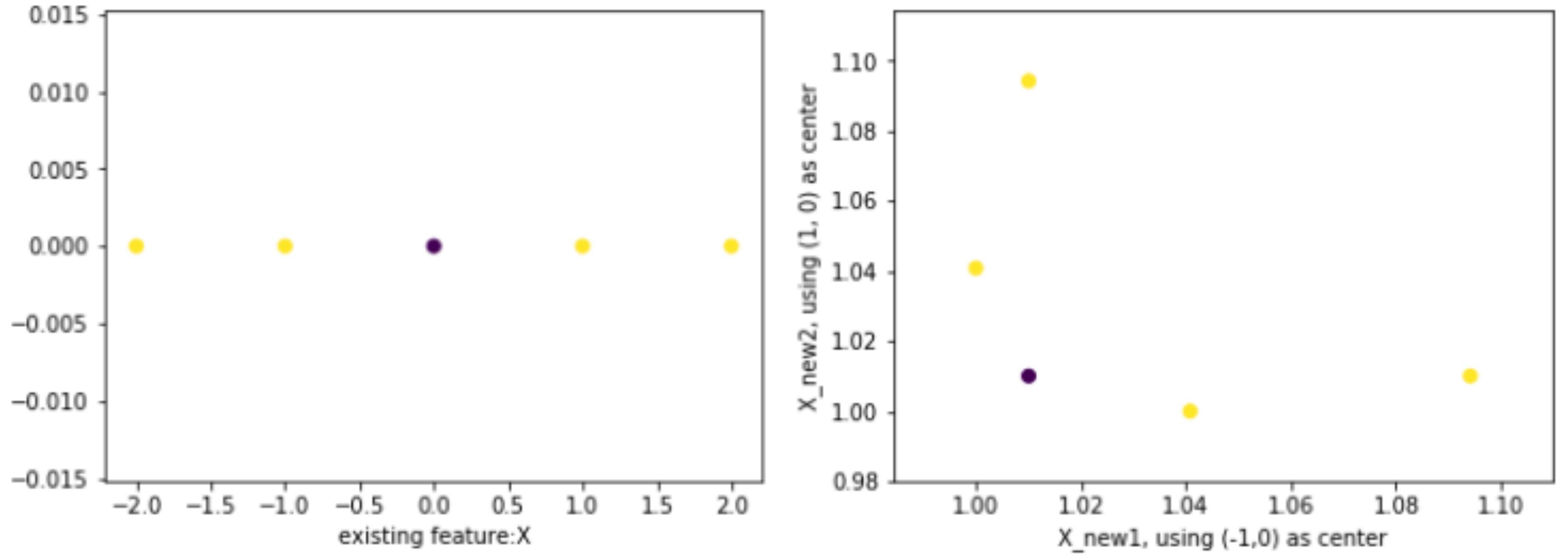
Radial Basis Function (RBF) kernel

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

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

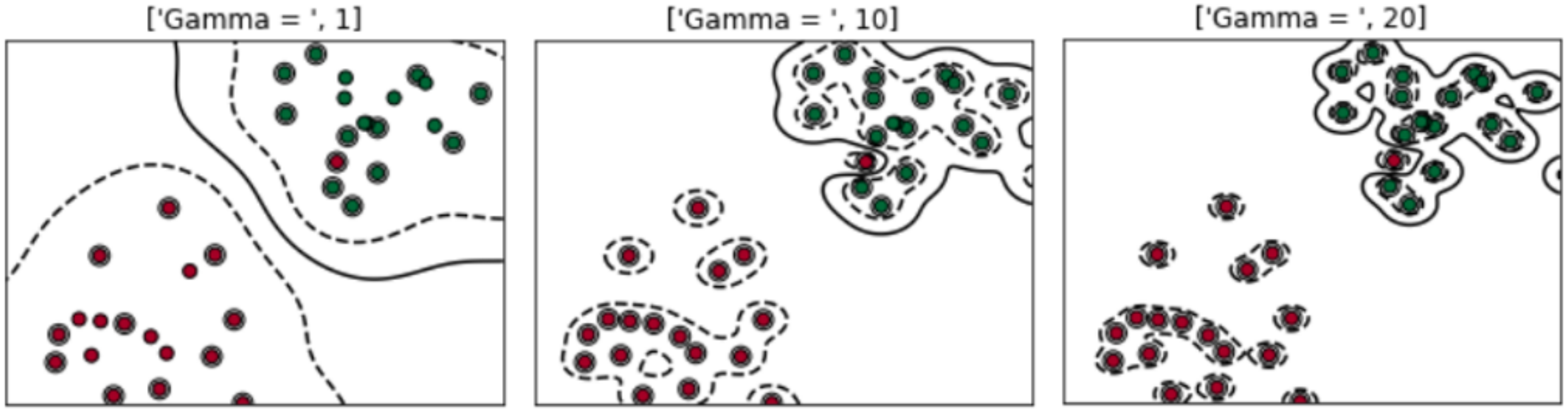
gamma (γ) controls the influence of new features — **Φ(x, center)** on the decision boundary. The higher the gamma, the more influence of the features will have on the decision boundary, more wiggling the boundary will be. Gamma can be determined by **Cross Validation**

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



https://www.youtube.com/watch?v=OdlNM96sHio

```
# 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)
```



Advantages:

1.SVM works relatively well when there is a clear margin of separation between classes.
2.SVM is more effective in high dimensional spaces.
3.SVM is effective in cases where the number of dimensions is greater than the number of samples.
4.SVM is relatively memory efficient

Disadvantages:

1.SVM algorithm is not suitable for large data sets.
2.SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
3.In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
4.As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.