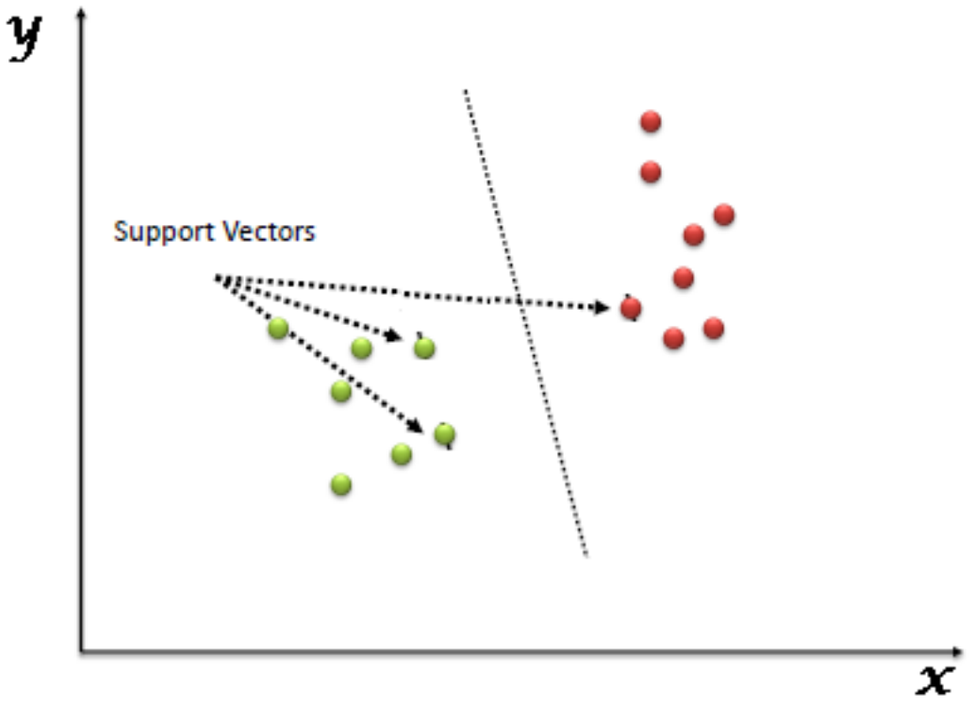
# Support Vector Machine

# Outline

- What is Support Vector Machine?
- How does it work?
- Kernels
- Pros and Cons associated with SVM

# SVM

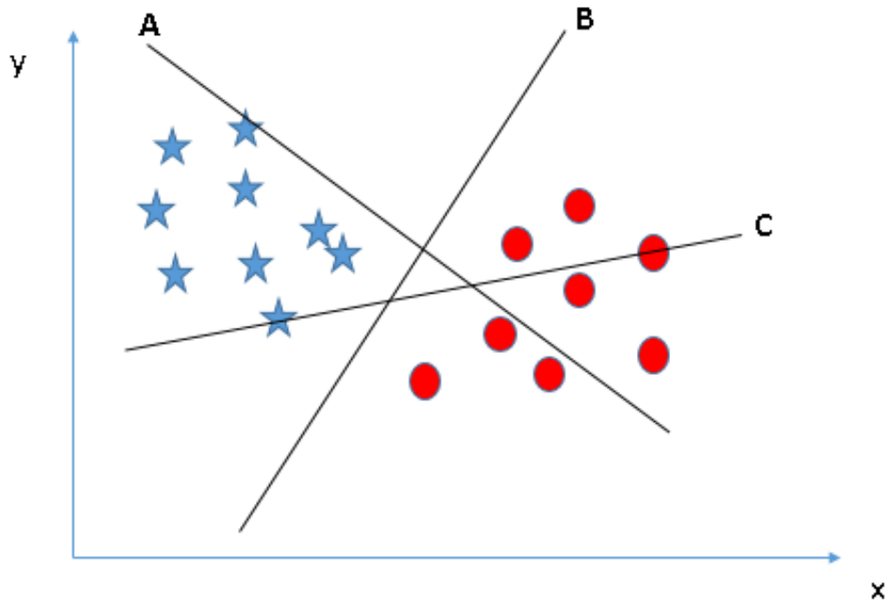
- a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems.
- we plot each data item as a point in p-dimensional space (where p is number of features you have) with the value of each feature being the value of a particular coordinate.
- we perform classification by finding the hyper-plane that differentiate the two classes very well .

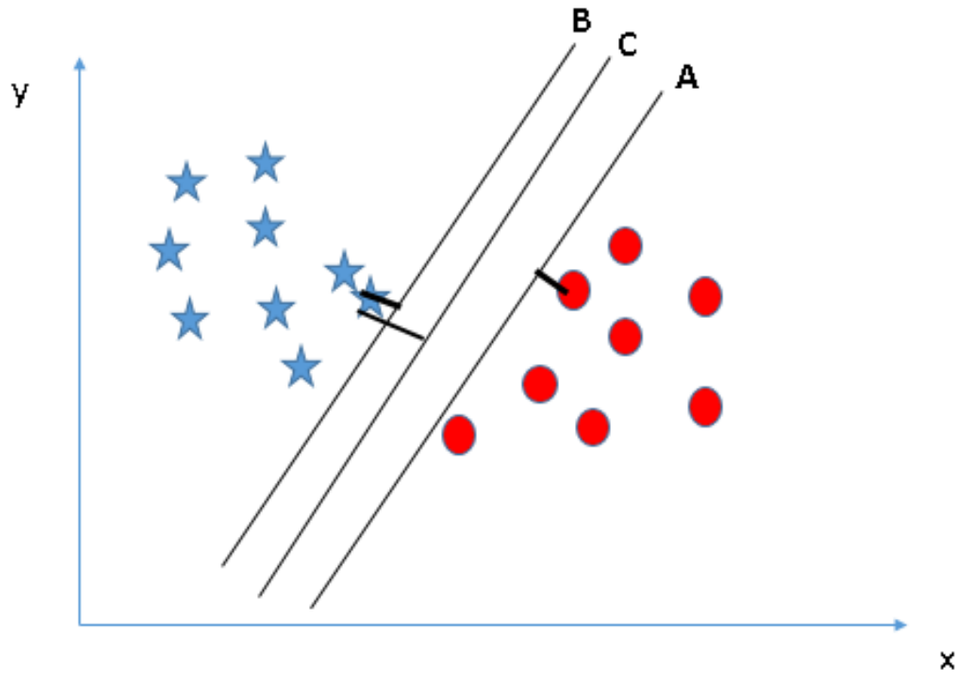



Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

# SVM

- Identify the right hyper-plane
  - “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.



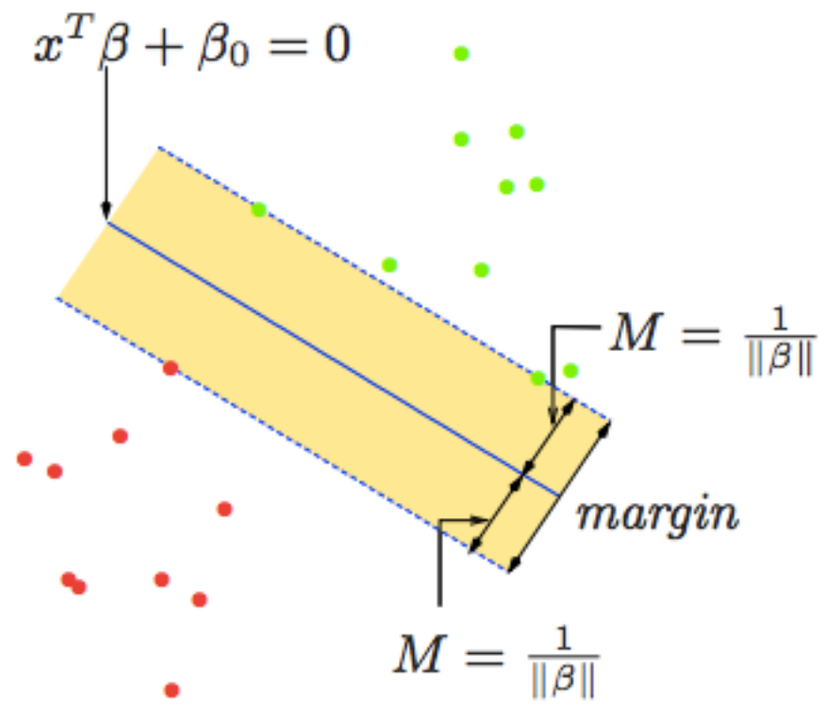

- Identify the right hyper-plane
  - maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane.

# Maximal-Margin Classifier

- The numeric input variables (x) in your data (the columns) form an p-dimensional space.
- A hyperplane is a line that splits the input variable space.
- In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1.
- Two-dimensional data:

$$B0 + (B1 * X1) + (B2 * X2) = 0$$

- Where the coefficients (B1 and B2) that determine the slope of the line and the intercept (B0) are found by the learning algorithm, and X1 and X2 are the two input variables.




- Above the line, the equation returns a value greater than 0 and the point belongs to the first class (class 1).
- Below the line, the equation returns a value less than 0 and the point belongs to the second class (class 0).
- A value close to the line returns a value close to zero and the point may be difficult to classify.
- If the magnitude of the value is large, the model may have more confidence in the prediction.
- The distance between the line and the closest data points is referred to as the margin.

# Maximal-Margin Classifier

- The best line that can separate the two classes is the line that as the largest margin. This is called the Maximal-Margin hyperplane.

- The margin is calculated as the perpendicular distance from the line to only the closest points. Only these points are relevant in defining the line and in the construction of the classifier. These points are called the support vectors. They support or define the hyperplane.

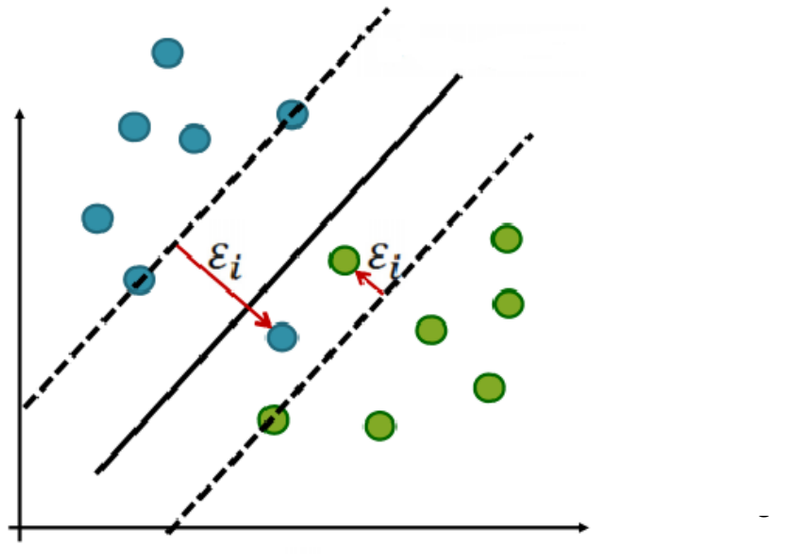
- The hyperplane is learned from training data using an optimization procedure that maximizes the margin

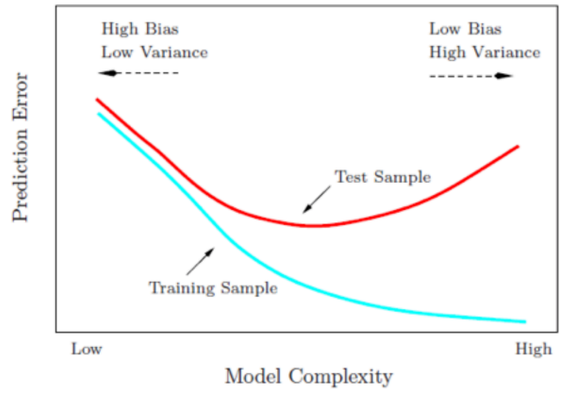
# Soft Margin Classifier

- The constraint of maximizing the margin of the line that separates the classes must be relaxed. This is often called the soft margin classifier. This change allows some points in the training data to violate the separating line.

- An additional set of coefficients are introduced that give the margin wiggle room in each dimension. These coefficients are sometimes called slack variables. This increases the complexity of the model as there are more parameters for the model to fit to the data to provide this complexity.

- A tuning parameter is introduced called simply C that defines the magnitude of the wiggle allowed across all dimensions. The C parameters defines the amount of violation of the margin allowed. C=0 is no violation and we are back to the inflexible Maximal-Margin Classifier described above. The larger the value of C the more violations of the hyperplane are permitted.




- During the learning of the hyperplane from data, all training samples that lie within the distance of the margin will affect the placement of the hyperplane and are referred to as support vectors. And as C affects the number of samples that are allowed to fall within the margin, C influences the number of support vectors used by the model.
- The smaller the value of C, the less model complexity (higher bias and lower variance).
- The larger the value of C, the more model complexity (lower bias and higher variance).

# Kernel Funciton

- A kernel function measures the similarity between two data points. So, for instance, if your task is object recognition, then a good kernel will assign a high score to a pair of images that contain the same objects, and a low score to a pair of images with different objects. Note that such a kernel captures much more abstract notion of similarity, compared to a similarity function that just compares two images pixel-by-pixel.

- Mathematical definition: K(x, y) = <f(x), f(y)>. Here K is the kernel function, x, y are n dimensional inputs. f is a map from n-dimension to m-dimension space. <x,y> denotes the dot product. usually m is much larger than n.

- Intuition: normally calculating <f(x), f(y)> requires us to calculate f(x), f(y) first, and then do the dot product. These two computation steps can be quite expensive as they involve manipulations in m dimensional space, where m can be a large number. But after all the trouble of going to the high dimensional space, the result of the dot product is really a scalar: we come back to one-dimensional space again! Now, the question we have is: do we really need to go through all the trouble to get this one number? do we really have to go to the m-dimensional space? The answer is no, if you find a clever kernel.

- Simple Example: x = (x1, x2, x3); y = (y1, y2, y3). Then for the function f(x) = (x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3), the kernel is K(x, y ) = (<x, y>)^2.
- Let's plug in some numbers to make this more intuitive: suppose x = (1, 2, 3); y = (4, 5, 6). Then:
  - f(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)
  - f(y) = (16, 20, 24, 20, 25, 30, 24, 30, 36)
  - <f(x), f(y)> = 16 + 40 + 72 + 40 + 100+ 180 + 72 + 180 + 324 = 1024
- A lot of algebra. Mainly because f is a mapping from 3-dimensional to 9 dimensional space.
- use the kernel instead:
  - K(x, y) = (4 + 10 + 18 ) ^2 = 32^2 = 1024
  - Same result, but this calculation is so much easier.

# Why Use Kernels in SVM?

- Additional pros of Kernel: kernels allow us to do stuff in infinite dimensions! Sometimes going to higher dimension is not just computationally expensive, but also impossible. f(x) can be a mapping from n dimension to infinite dimension which we may have little idea of how to deal with. Then kernel gives us a wonderful shortcut.

How is related to SVM?

The idea of SVM is that y = w phi(x) +b, where w is the weight, phi is the feature vector, and b is the bias. if y> 0, then we classify to class 1, else to class 0. We want to find a set of weight and bias such that the margin is maximized. The feature vector phi(x) makes the data linearly separable. Kernel is to make the calculation process faster and easier, especially when the feature vector phi is of very high dimension (for example, x1, x2, x3, ..., x_D^n, x1^2, x2^2, ...., x_D^2).

if we put the definition of kernel above, <f(x1), f(x2)>, in the context of SVM and feature vectors, it becomes <phi(x1), phi(x2)>. The inner product means the projection of phi(x1) onto phi(x2), how much overlap do x1 and x2 have in their feature space. In other words, how similar they are.

# Support Vector Machines (Kernels)

- The linear SVM can use the inner product of any two given observations, rather than the observations themselves. The inner product between two vectors is the sum of the multiplication of each pair of input values.

- For example, the inner product of the vectors [2, 3] and [5, 6] is 2*5 + 3*6 or 28.

- The equation for making a prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows: f(x) = B0 + sum(ai * (x,xi))

- This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

- Computationally, it turns out that the parameters ai are only nonzero if the corresponding observation is itself a support vector!  If a training observation is not a support vector, then its corresponding is necessarily 0

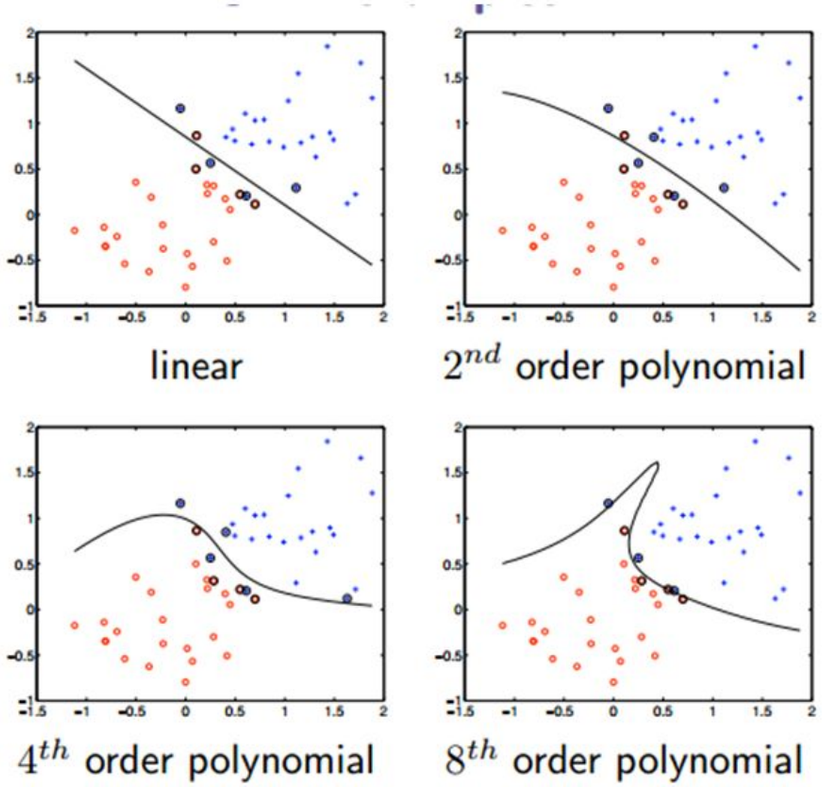
- Using kernels is much more computationally efficient because we only need to compute the kernel for distinct pairs of observations in our dataset.

# Linear Kernel SVM

- The dot-product is called the kernel and can be re-written as:
- K(x, xi) = sum(x * xi)
- The kernel defines the similarity or a distance measure between new data and the support vectors. The dot product is the similarity measure used for linear SVM or a linear kernel because the distance is a linear combination of the inputs.
- Other kernels can be used that transform the input space into higher dimensions such as a Polynomial Kernel and a Radial Kernel. This is called the Kernel Trick.
- It is desirable to use more complex kernels as it allows lines to separate the classes that are curved or even more complex. This in turn can lead to more accurate classifiers.

# Polynomial Kernel SVM

## Polynomial Kernel SVM Example

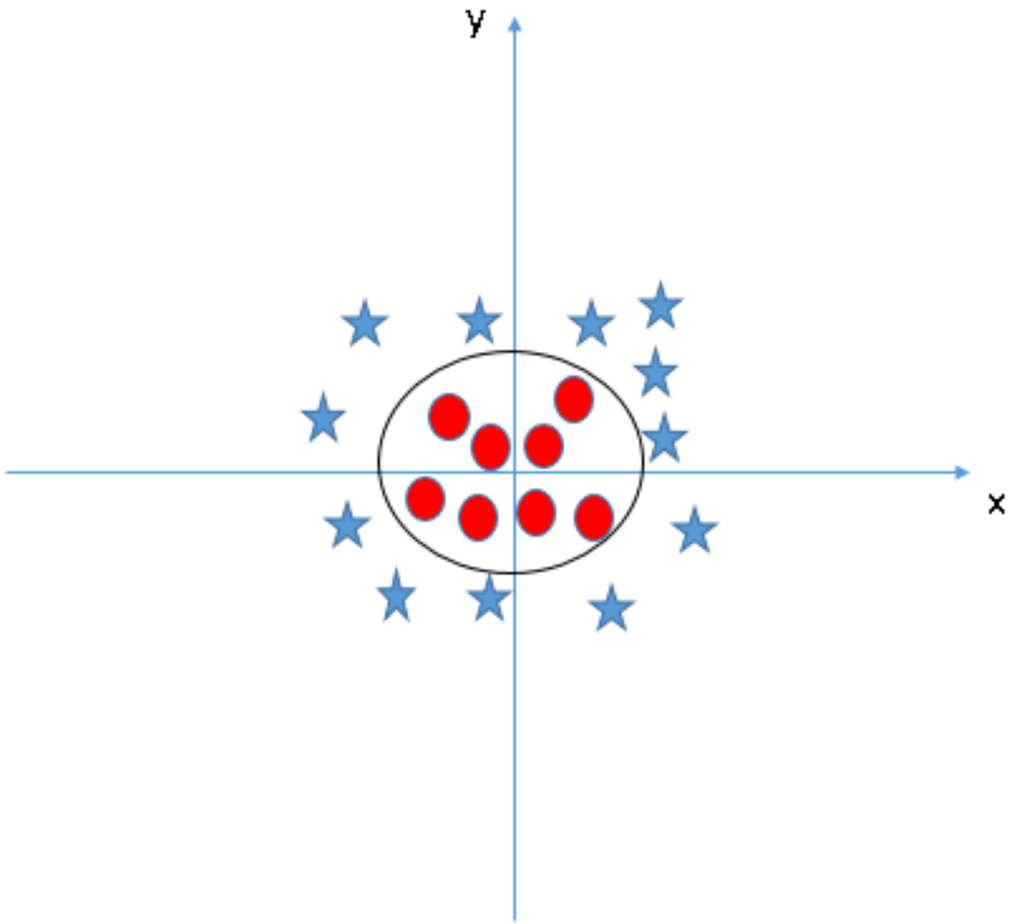



Slide from Tommi S. Jaakkola, MIT

61

- Instead of the dot-product, we can use a polynomial kernel, for example:
- K(x,xi) = 1 + sum(x * xi)^d
- Where the degree of the polynomial must be specified by hand to the learning algorithm. When d=1 this is the same as the linear kernel. The polynomial kernel allows for curved lines in the input space.

# Radial Kernel SVM




- Finally, we can also have a more complex radial kernel. For example:
- K(x,xi) = exp(-gamma * sum((x – xi^2))
- Where gamma is a parameter that must be specified to the learning algorithm. A good default value for gamma is 0.1, where gamma is often 0 < gamma < 1. The radial kernel is very local and can create complex regions within the feature space, like closed polygons in two-dimensional space.

# Pros and Cons of SVM

Pros:

- It works really well with clear margin of separation.
- It is effective in high dimensional spaces.
- It is effective in cases where number of dimensions is greater than the number of samples.
- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Cons:

- It doesn’t perform well, when we have large data set because the required training time is higher.
- It also doesn’t perform very well, when the data set has more noise i.e. target classes are overlapping.
- SVM doesn’t directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.