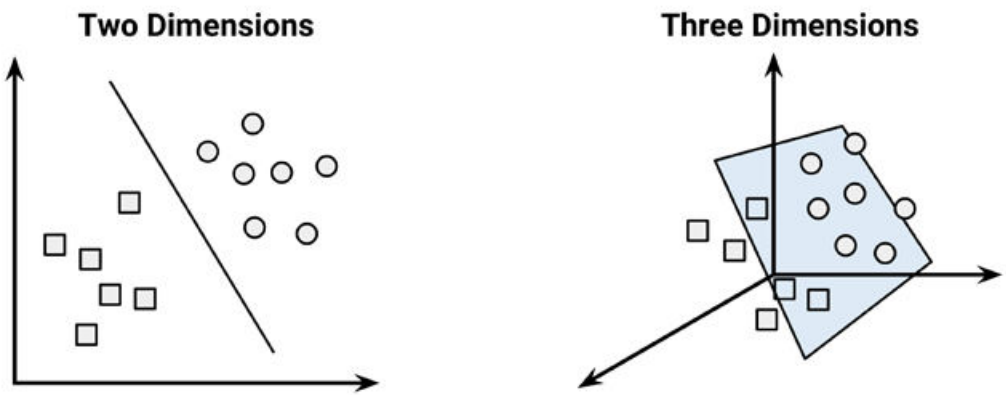
SVM

Introduction

- Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges.
- it is mostly used in classification problems.
- A **Support Vector Machine** (**SVM**) can be imagined as a surface that creates a boundary between points of data plotted in multidimensional that represent examples and their feature values.
- The goal of a SVM is to create a flat boundary called a **hyperplane**, which divides the space to create fairly homogeneous partitions on either side.

.

- Hyperplanes that separate groups of circles and squares in two and three dimensions is shown below. Because the circles and squares can be separated perfectly by the straight line or flat surface, they are said to be **linearly separable**
- **SVM combines KNN and linear regression methods**

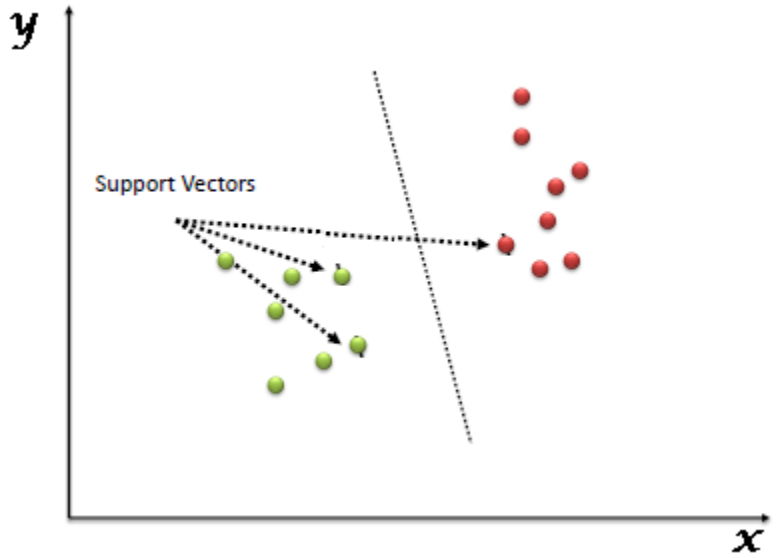

Two Dimensions
Three Dimensions

Notable applications include:

- Classification of microarray gene expression data in the field of bioinformatics to identify cancer or other genetic diseases
- Text categorization such as identification of the language used in a document or the classification of documents by subject matter
- The detection of rare yet important events like combustion engine failure, security breaches, or earthquakes

Support Vectors

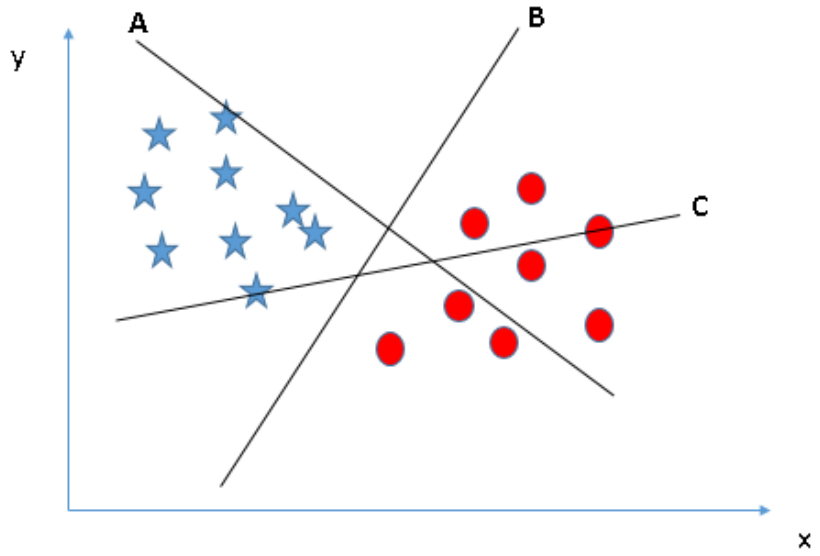
- The **support vectors** (indicated by arrows in the figure that follows) are the points from each class that are the closest to the MMH
- Each class must have at least one support vector, but it is possible to have more than one.
- Using the support vectors alone, it is possible to define the MMH. This is a key feature of SVMs
- The support vectors provide a very compact way to store a classification model, even if the number of features is extremely large.


y
Support Vectors
x

How we can identify the right hyper plane

(Scenario-1):

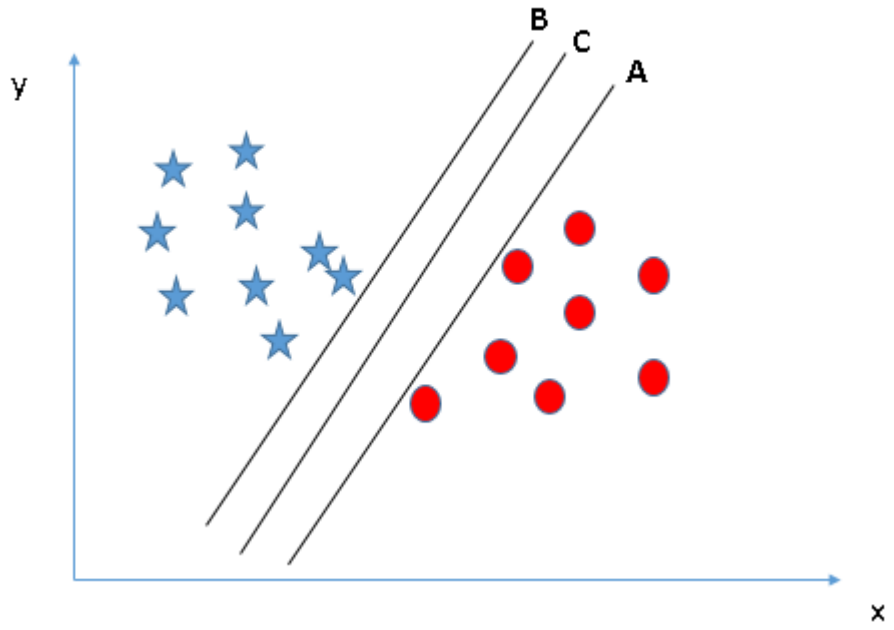
- Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.



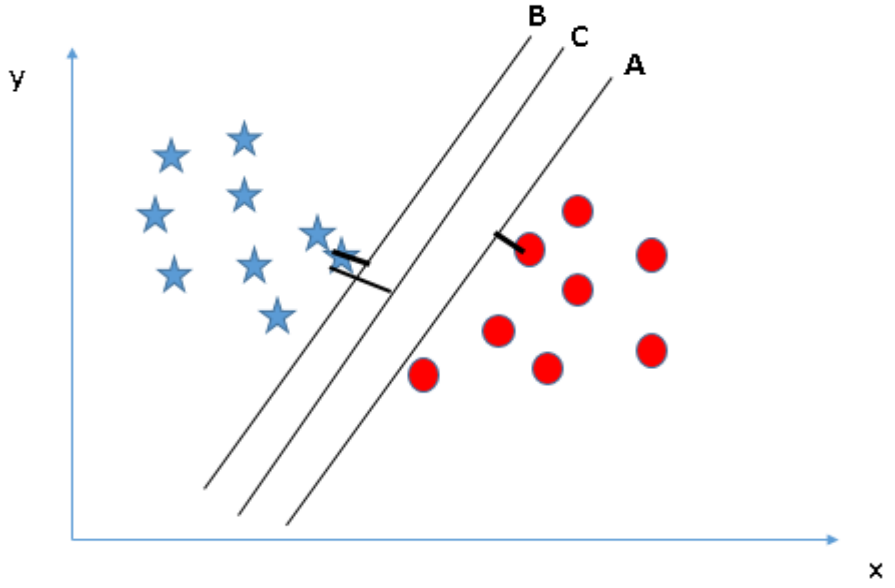
- Select the hyper-plane which segregates the two classes better
- Hyper-plane “B” has excellently performed this job

Identify the right hyper-plane (Scenario-2):

- Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

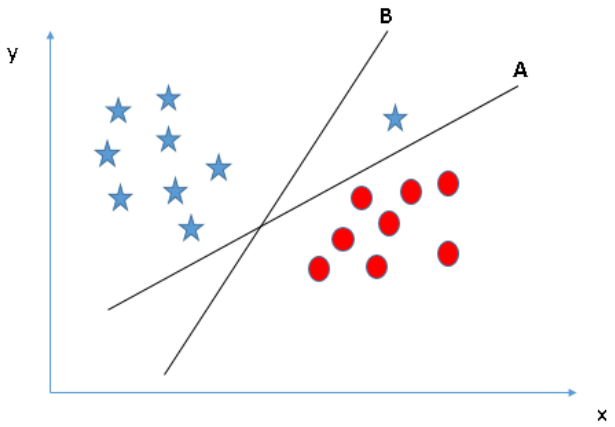


- Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.



- The margin for hyper-plane C is high as compared to both A and B (**Maximum margin Hyperplane**). Hence, So the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

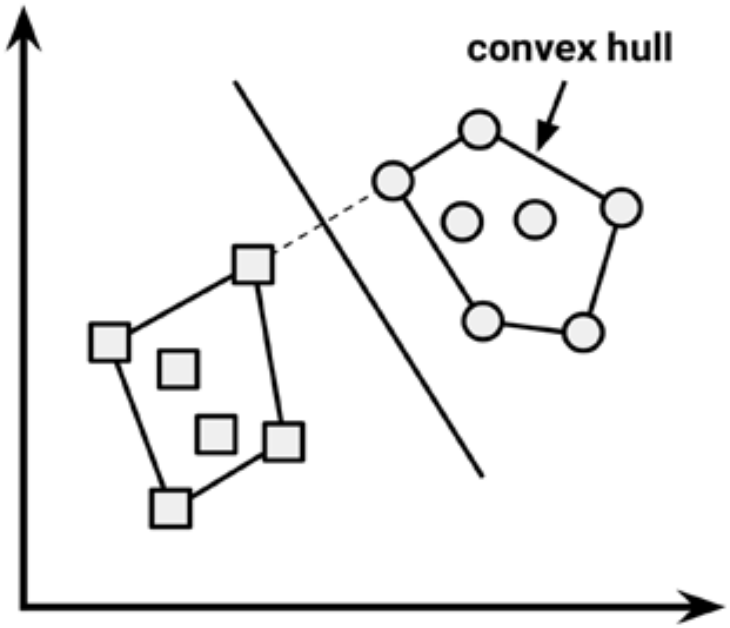
Identify the right hyper-plane (Scenario-3)



- Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

Convex Hull

- In the case of linearly separable data, the MMH is as far away as possible from the outer boundaries of the two groups of data points. These outer boundaries are known as the **convex hull**. The MMH is then the perpendicular bisector of the shortest line between the two convex hulls. Sophisticated computer algorithms that use a technique known as **quadratic optimization** are capable of finding the maximum margin in this way.


convex hull

Hyperplane in n dimensional space

- $\overrightarrow{\omega}.\ \vec{x} + b = 0$
- *w* is a vector of *n* weights, that is, *{w1, w2, …, wn}*, and *b* is a single number known as the **bias**. The bias is conceptually equivalent to the intercept term in the slope-intercept form discussed in *Regression Methods*.

- The goal of the process is to find a set of weights that specify two hyperplanes

$$\overrightarrow{\omega.}\,\vec{x} + b \geq +1$$
$$\overrightarrow{\omega.}\,\vec{x} + b \leq -1$$

- These hyperplanes are specified such that all the points of one class fall above the first hyperplane and all the points of the other class fall beneath the second hyperplane.
- This is possible so long as the data are linearly separable.

- These hyperplanes are specified such that all the points of one class fall above the first hyperplane and all the points of the other class fall beneath the second hyperplane. This is possible so long as the data are linearly separable.

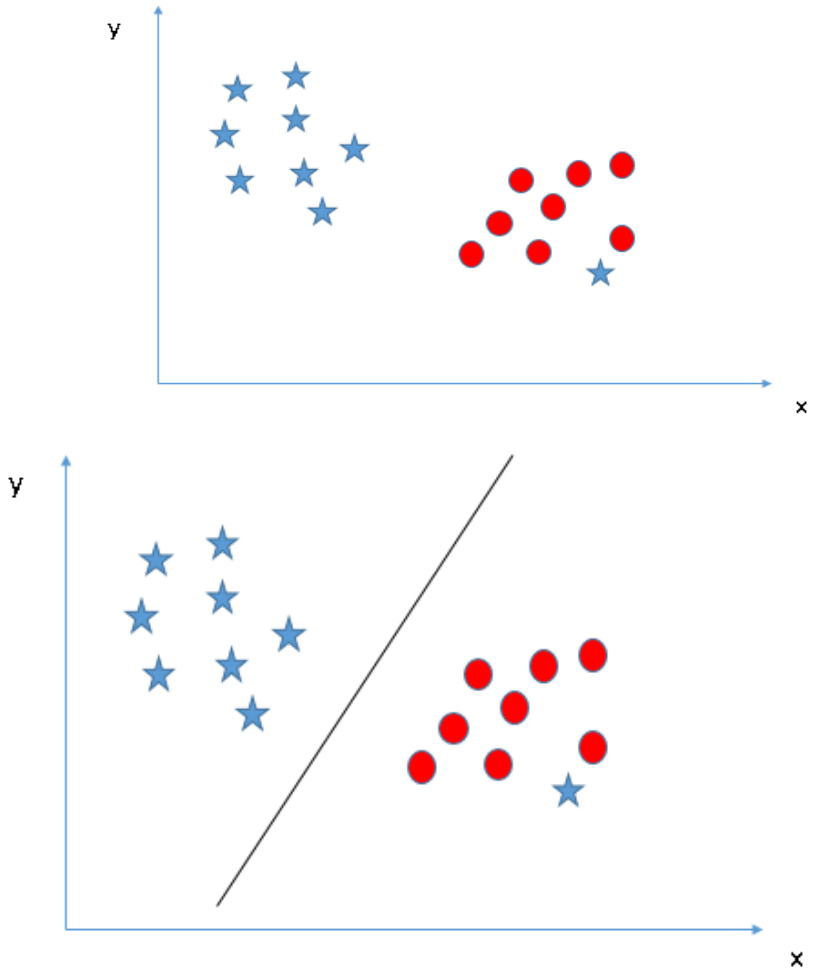
- Vector geometry defines the distance between these two planes as:

$$\frac{2}{||\vec{w}||}$$

- Here, //*w*// indicates the **Euclidean norm** (the distance from the origin to vector *w*). Because //*w*// is in the denominator, to maximize distance, we need to minimize //*w*//. The task is typically reexpressed as a set of constraints, as follows:

- $$\min \frac{1}{2} \|\vec{w}\|^2$$
$$s.t.\ \ y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \forall \vec{x}$$

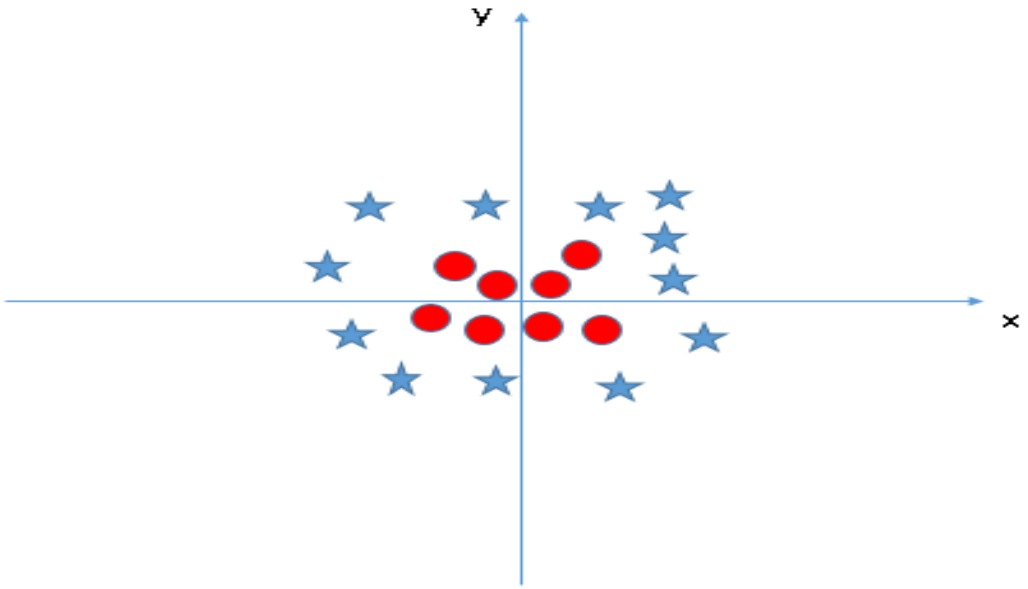
Scenario 4



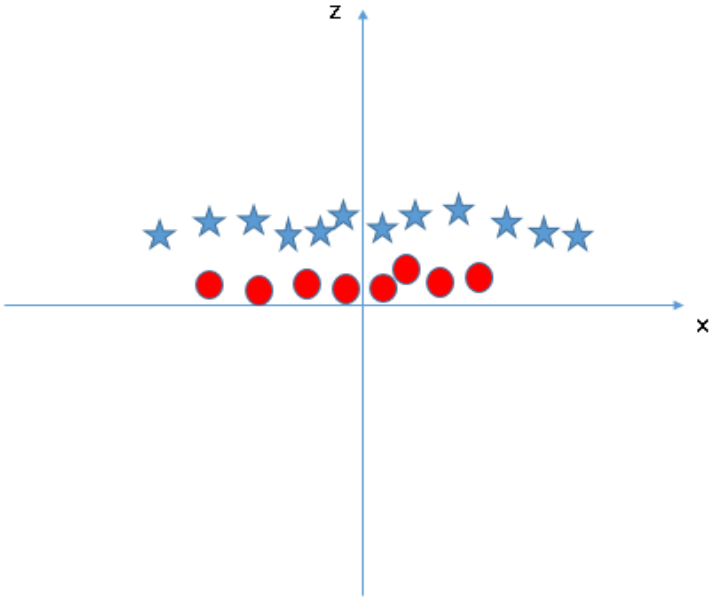
As I have already mentioned, one star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.

Find the hyper-plane to segregate to classes (Scenario-5): Non linear spaces

- In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



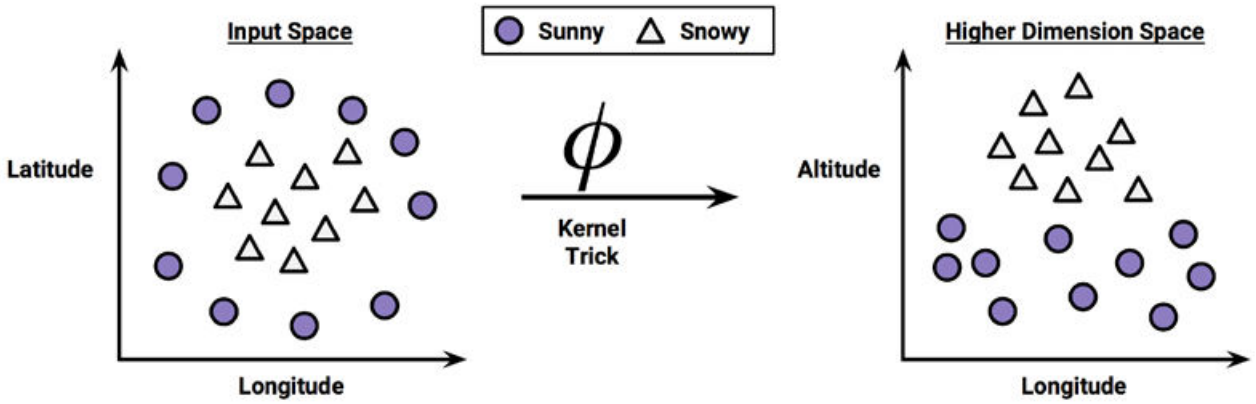
- SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:



- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

- In the SVM classifier, it is easy to have a linear hyper-plane between these two classes.
- SVM algorithm has a technique called the **kernel trick**. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then non liner relationship appear to be linear

- In the following figure, the scatterplot on the left depicts a nonlinear relationship between a weather class (sunny or snowy) and two features: latitude and longitude. The points at the center of the plot are members of the snowy class, while the points at the margins are all sunny.


Input Space
Sunny
Snowy
Higher Dimension Space
Latitude
Kernel
Trick
Altitude
Longitude
Longitude

- On the right side of the figure, after the kernel trick has been applied, new dimension: altitude is evolved.
- With the addition of this feature, the classes are now perfectly linearly separable.
- snowy weather is found at higher altitudes.
- SVMs with nonlinear kernels add additional dimensions to the data in order to create separation in this way. Essentially, the kernel trick involves a process of constructing new features that express mathematical relationships between measured characteristics. For instance, the altitude feature can be expressed mathematically as an interaction between latitude and longitude—the closer the point is to the center of each of these scales, the greater the altitude. This allows SVM to learn concepts that were not explicitly measured in the original data.

SVMs with nonlinear kernels are extremely powerful classifiers

Strengths	Weaknesses
• Can be used for classification or numeric prediction problems • Not overly influenced by noisy data and not very prone to overfitting • May be easier to use than neural networks, particularly due to the existence of several well-supported SVM algorithms • Gaining popularity due to its high accuracy and high-profile wins in data mining competitions	• Finding the best model requires testing of various combinations of kernels and model parameters • Can be slow to train, particularly if the input dataset has a large number of features or examples • Results in a complex black box model that is difficult, if not impossible, to interpret

- Kernel functions, in general, are of the following form.
- The function denoted by the Greek letter phi, that is, φ(x), is a mapping of the data into another space. Therefore, the general kernel function applies some transformation to the feature vectors *xi* and *xj* and combines them using the **dot product**, which takes two vectors and returns a single number.

$$\mathrm{K}(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$$

- The **linear kernel** does not transform the data at all. Therefore, it can be expressed simply as the dot product of the features:

$$\mathrm{K}(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$$

- The **polynomial kernel** of degree *d* adds a simple nonlinear transformation of the data:\

$$\mathrm{K}(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^d$$

- The **sigmoid kernel** results in a SVM model somewhat analogous to a neural network using a sigmoid activation function. The Greek letters kappa and delta are used as kernel parameters:

$$K(\vec{x_i}, \vec{x_j}) = \tanh(\kappa \vec{x_i} \cdot \vec{x_j} - \delta)$$

- The **Gaussian RBF kernel** is similar to a RBF neural network. The RBF kernel performs well on many types of data and is thought to be a reasonable starting point for many learning tasks:

$$\mathrm{K}(\vec{x_i}, \vec{x_j}) = e^{\frac{-\|\vec{x_i} - \vec{x_j}\|^2}{2\sigma^2}}$$

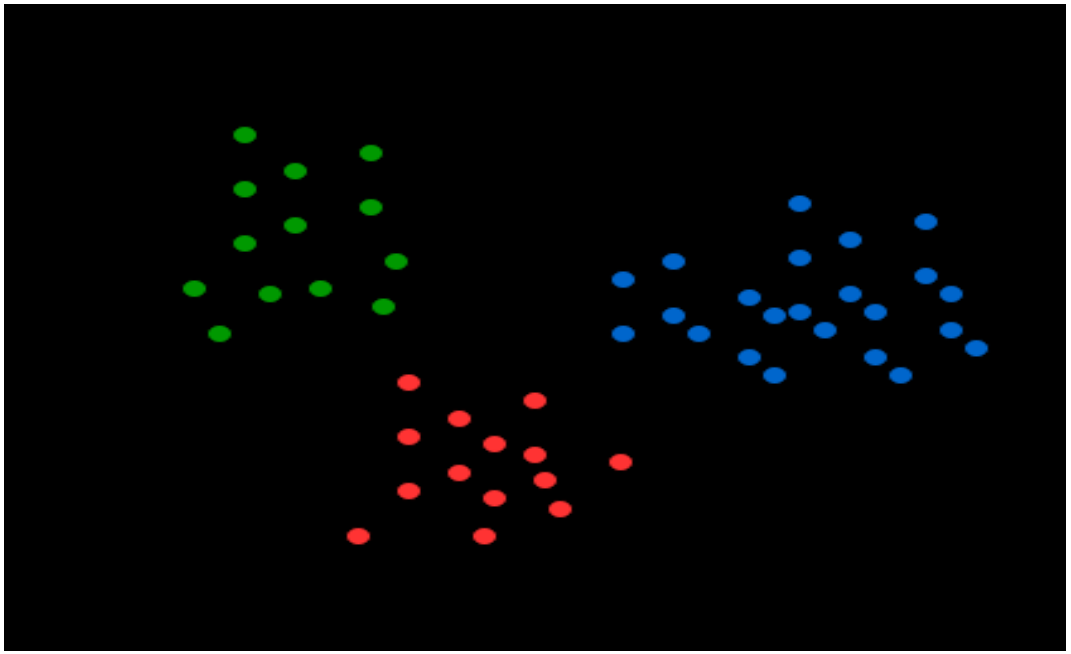
- The fit depends heavily on the concept to be learned as well as the amount of training data and the relationships among the features.
- A bit of trial and error is required by training and evaluating several SVMs on a validation dataset. This said, in many cases, the choice of kernel is arbitrary, as the performance may vary slightly.

Multiclass SVM

- In this type, the machine should classify an instance as only one of three classes or more.
- Classifying a text as positive, negative, or neutral
- Determining the dog breed in an image
- Categorizing a news article to sports, politics, economics, or social

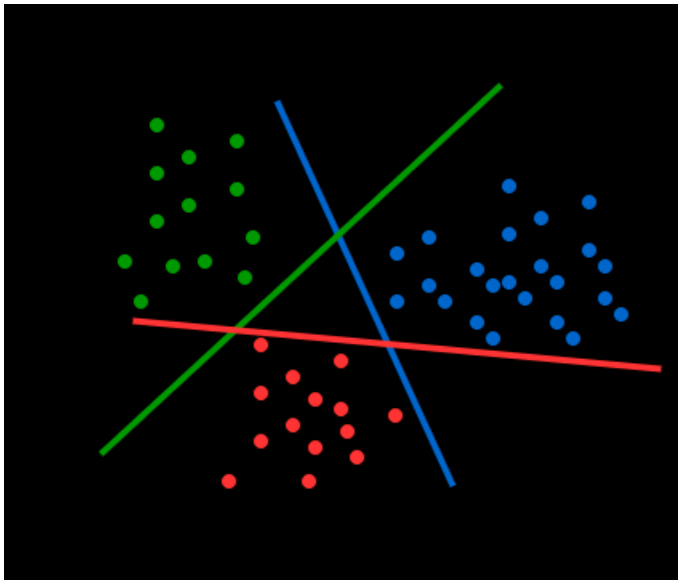
- In its most simple type, SVM doesn't support multiclass classification natively. It supports binary classification and separating data points into two classes. For multiclass classification, the same principle is utilized after breaking down the multiclassification problem into multiple binary classification problems.
- The idea is to map data points to high dimensional space to gain mutual linear separation between every two classes. **This is called a *One-to-One* approach, which breaks down the multiclass problem into multiple binary classification problems. A binary classifier per each pair of classes.**
- **Another approach one can use is *One-to-Rest*. In that approach, the breakdown is set to a binary classifier per each class.**
- A single SVM does binary classification and can differentiate between two classes. So that, according to the two breakdown approaches, to classify data points from m classes data set:
  - In the *One-to-Rest* approach, the classifier can use m SVMs. Each SVM would predict membership in one of the m classes.
  - In the *One-to-One* approach, the classifier can use m(m-1)/2 SVMs.

- Let’s take an example of 3 classes classification problem; green, red, and blue, as the following image:
-

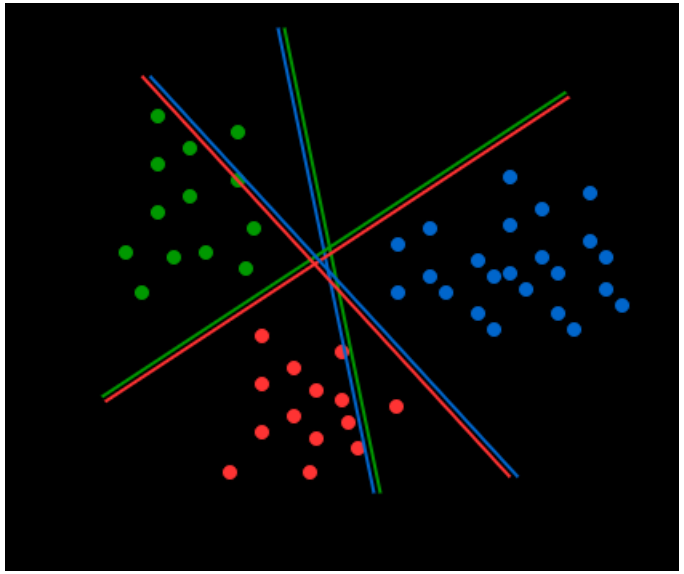


- In the *One-to-Rest* approach, we need a hyperplane to separate between a class and all others at once. This means the separation takes all points into account, dividing them into two groups; a group for the class points and a group for all other points. For example, the green line tries to maximize the separation between green points and all other points at once:
-

https://www.baeldung.com/cs/svm-multiclass-classification



- In the *One-to-One* approach, we need a hyperplane to separate between every two classes, neglecting the points of the third class. This means the separation takes into account only the points of the two classes in the current split. For example, the red-blue line tries to maximize the separation only between blue and red points. It has nothing to do with green points:
-



```
from sklearn import svm, datasets
import sklearn.model_selection as model_selection
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, train_size=0.80, test_size=0.20, random_state=101)
rbf = svm.SVC(kernel='rbf', gamma=0.5, C=0.1).fit(X_train, y_train)
poly = svm.SVC(kernel='poly', degree=3, C=1).fit(X_train, y_train)
poly_pred = poly.predict(X_test)
rbf_pred = rbf.predict(X_test)
poly_accuracy = accuracy_score(y_test, poly_pred)
poly_f1 = f1_score(y_test, poly_pred, average='weighted')
print('Accuracy (Polynomial Kernel): ', "%.2f" % (poly_accuracy*100))
print('F1 (Polynomial Kernel): ', "%.2f" % (poly_f1*100))
rbf_accuracy = accuracy_score(y_test, rbf_pred)
rbf_f1 = f1_score(y_test, rbf_pred, average='weighted')
print('Accuracy (RBF Kernel): ', "%.2f" % (rbf_accuracy*100))
print('F1 (RBF Kernel): ', "%.2f" % (rbf_f1*100))
```