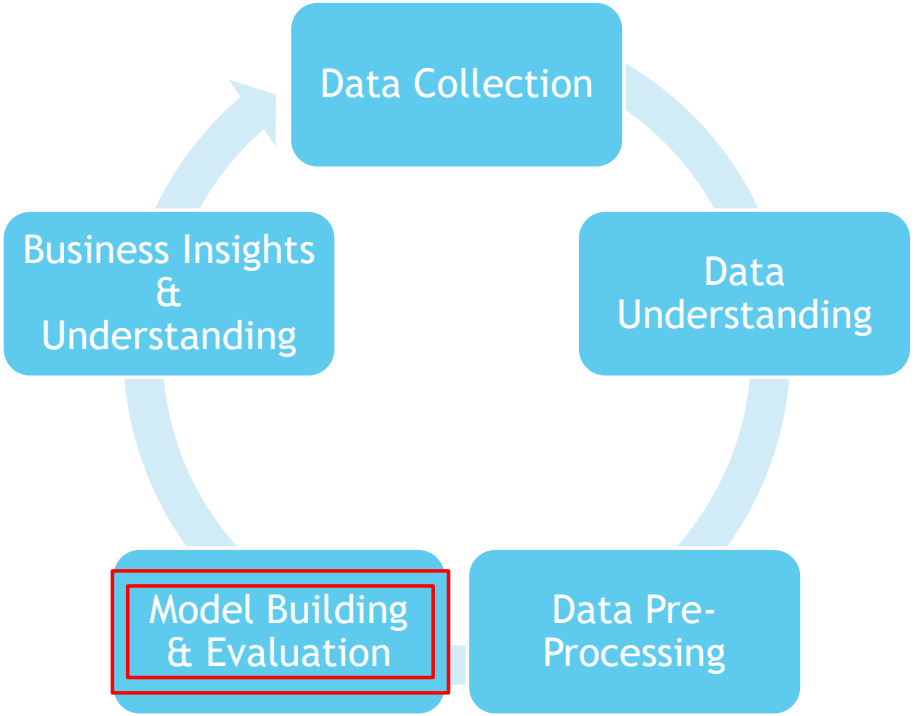
SUPPORT VECTOR MACHINES

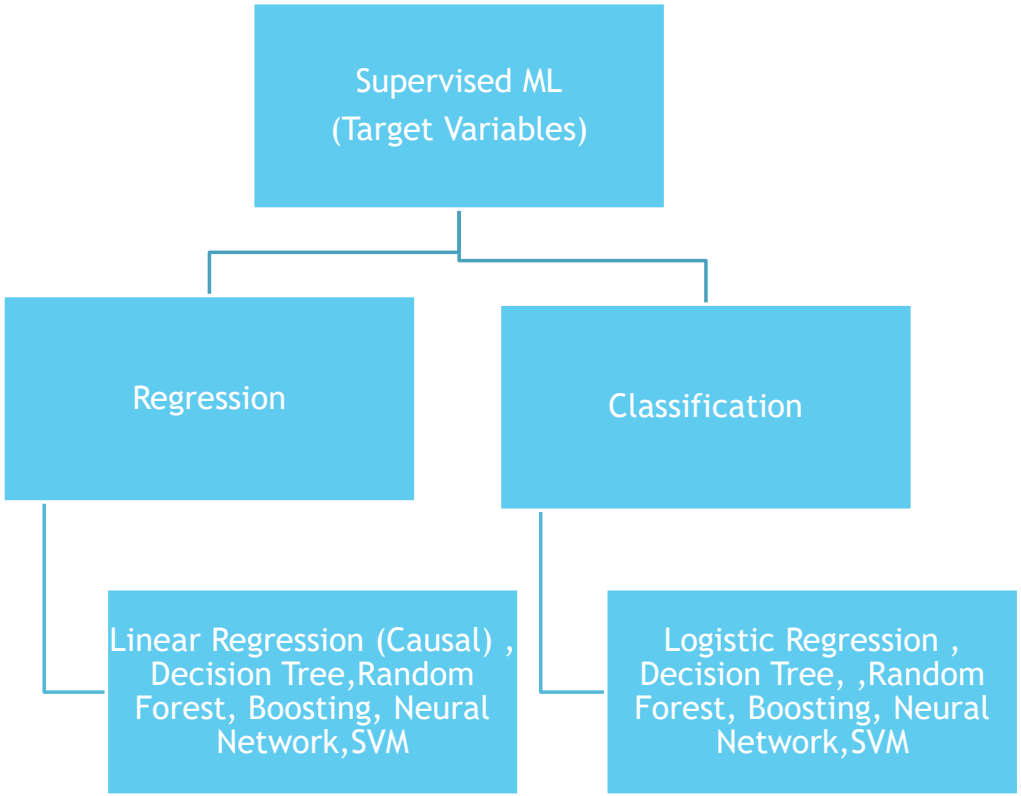
DataDNA

Typical Data Science Cycle



DataDNA

Machine Learning



DataDNA

Introduction

- **SVM** (support vector machine) is a supervised ML algorithm.
- SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.
- Support-vector machine constructs a line or hyperplane in a high or infinite-dimensional space.
- SVMs can efficiently perform a non-linear classification using what is called the kernel trick.
- It is applicable for classification and regression.

How SVM Works

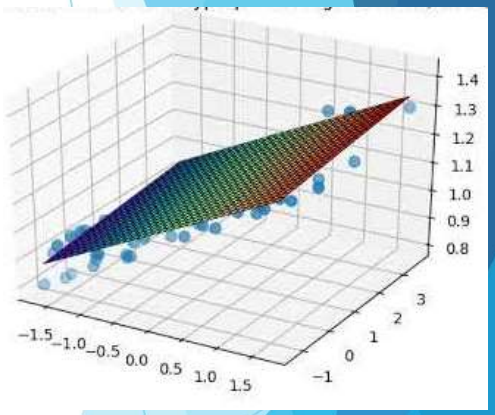
1. Identify target variable and independent variables.

2. SVM comes up with an imaginary Hyper-Plane that splits the data into target outputs.

If we have p-dimensional space, a hyperplane is a flat subspace with dimension p-1.

e.g. For three-dimensional space, a hyperplane is a two-dimensional subspace.

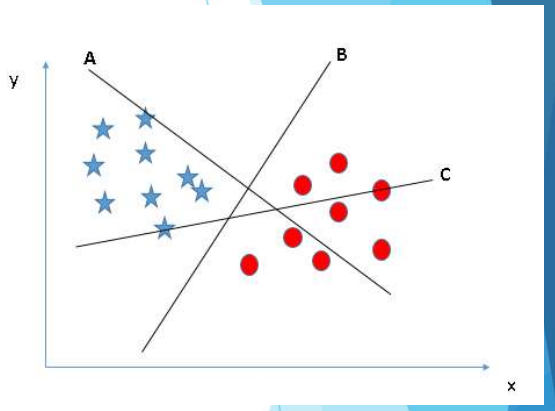
e.g. For 2 dimensional data a line will be hyperplane.



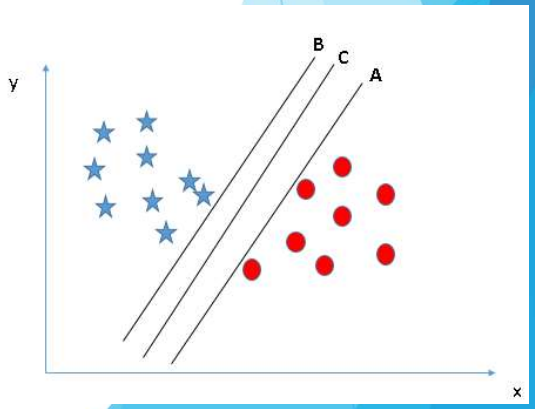
How SVM Works

3. SVM creates multiple hyper plane. Each plane will try to classify observations based upon its equation.

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 < 0$ then y = -1
$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 > 0$ then y = 1

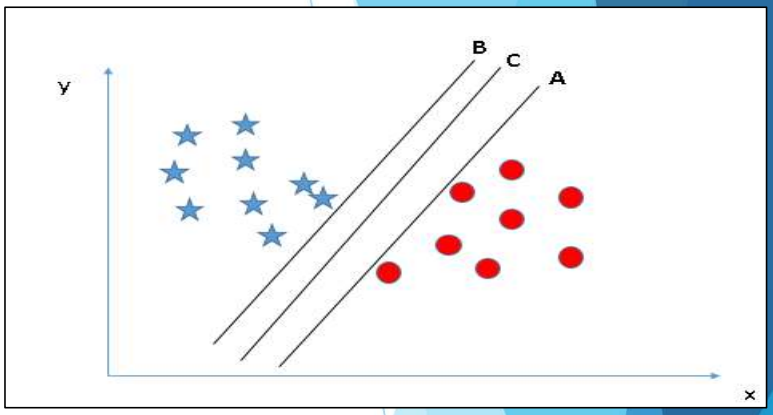


4. In multidimension , many hyperplanes can create classification around training data but to choose best hyperplane we need to define degree of Separation which is called **Margin**

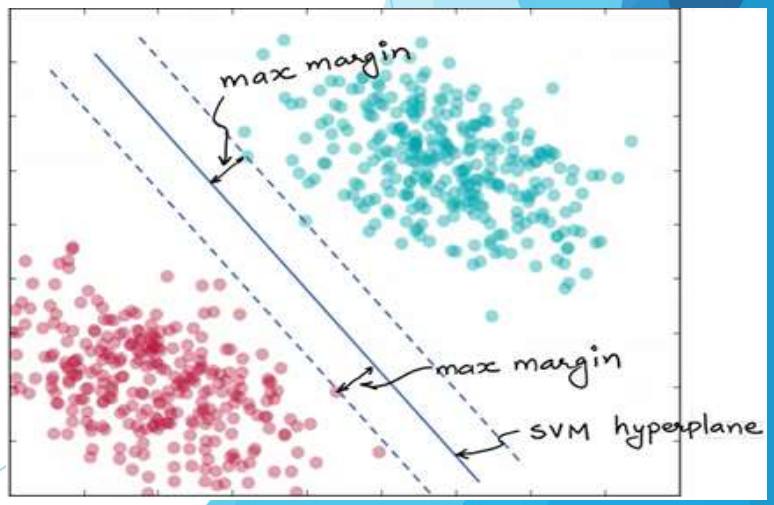


How SVM Works

5. Distance of closest points of separate classes around individual hyperplane is calculated. This is called margin.

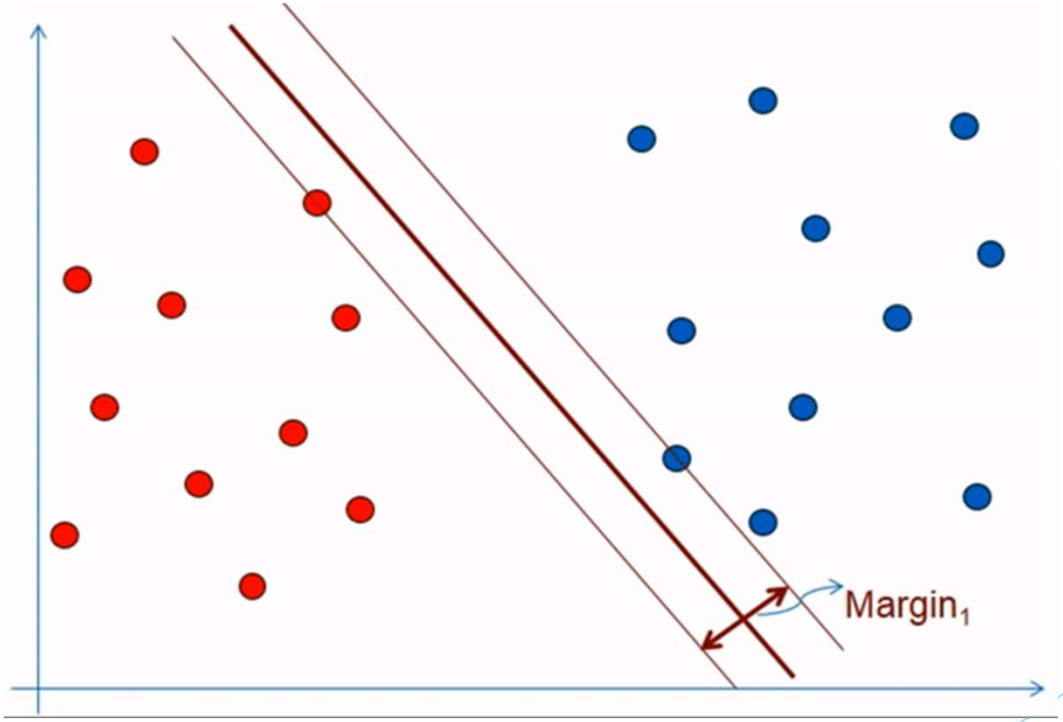


6. Best hyper plane will have **maximum margin** around nearest training data points**.** This Plane is also called **maximal margin classifier .** It contains two lines called margin lines**,** each on one and other side of that plane**.**



DataDNA

How SVM Works



How SVM Works

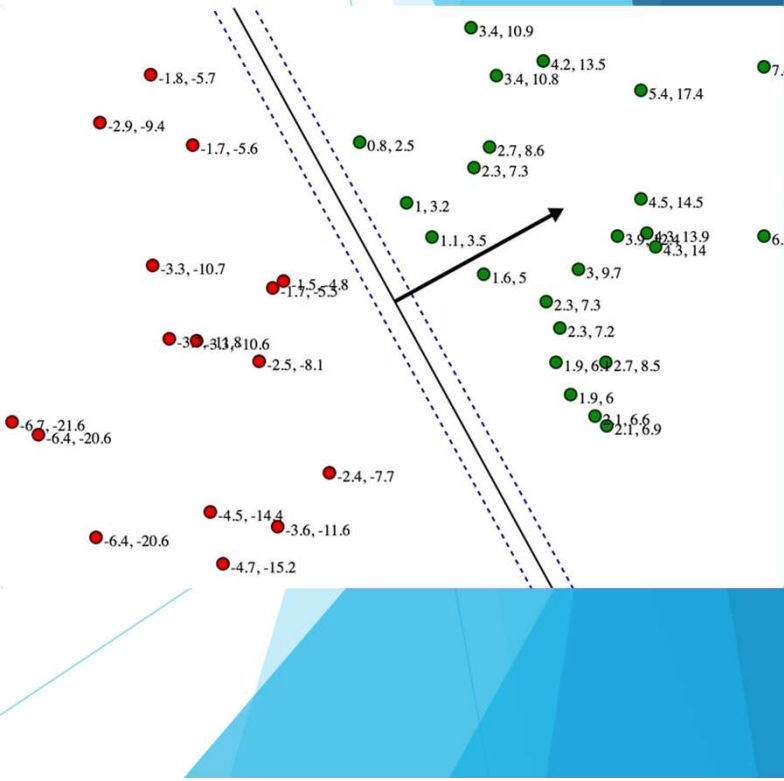
7. Maximum Margin (M) for hyperplane is the distance of the closest point in the data set with hyperplane.

objective function $\underline{\textbf{maximize}} \ \frac{2}{\|W\|}$

where $\frac{1}{\|W\|}$ is distance of support vectors from hyperplane

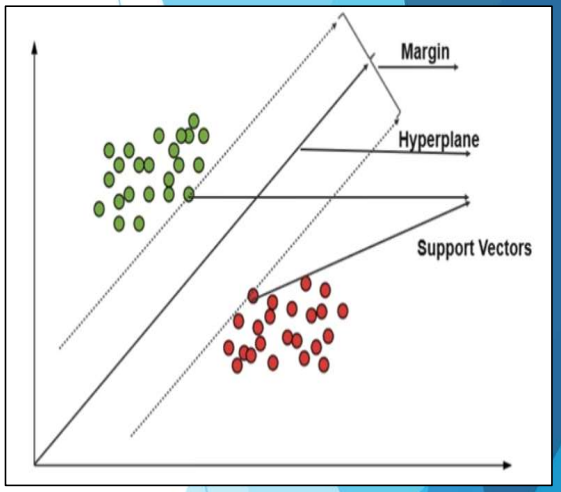
While calculating margins (M) it is made sure that each observation is on the correct side (+1 class and -1 class) of the hyperplane and at least a distance M from the hyperplane. (**constraint**)

$$(\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i}) * Y_i >= M$$

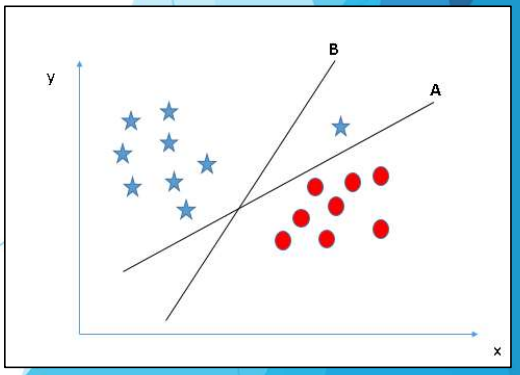


How SVM Works

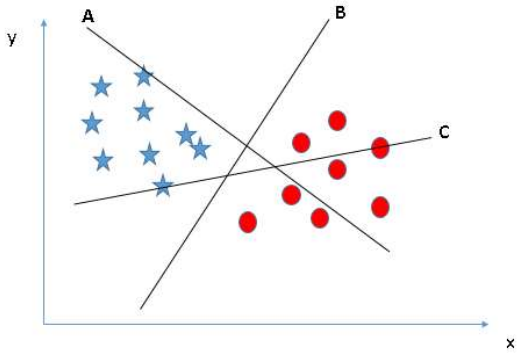
8. Support Vectors are simply the coordinates of closest observation (lying on margin lines). They define the length of maximal margin. These affect the position and orientation of the hyper-plane. We have to select a hyperplane, for which the margin, i.e the distance between support vectors and hyper-plane is maximum. Even a little interference in the position of these support vectors can change the hyper-plane.



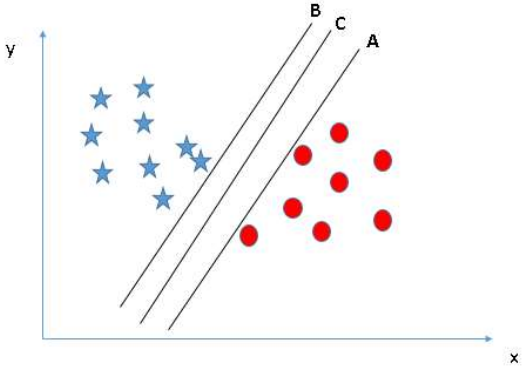
9. Apart from selecting maximum margin, we need to consider minimum misclassification error. Although hyperplane B has higher margin but right Hyperplane is A because it has classified all samples correctly. SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin.



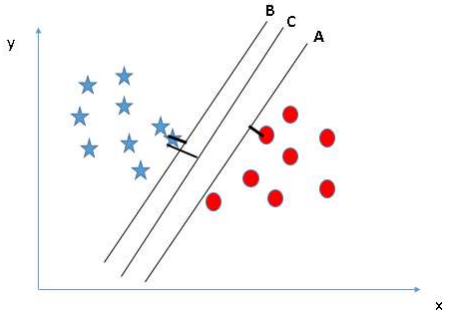
How SVM Works



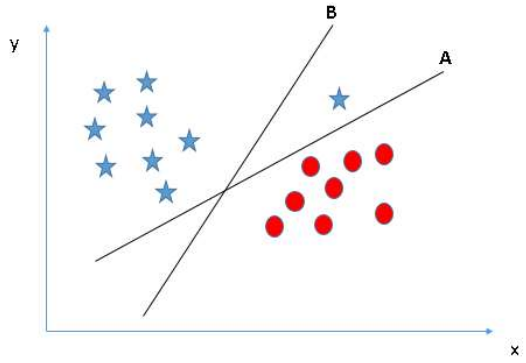
1. MULTIPLE HYPERPLANES



2. HYPERPLANES CLASSIFYING POINTS SUCCESSFULLY



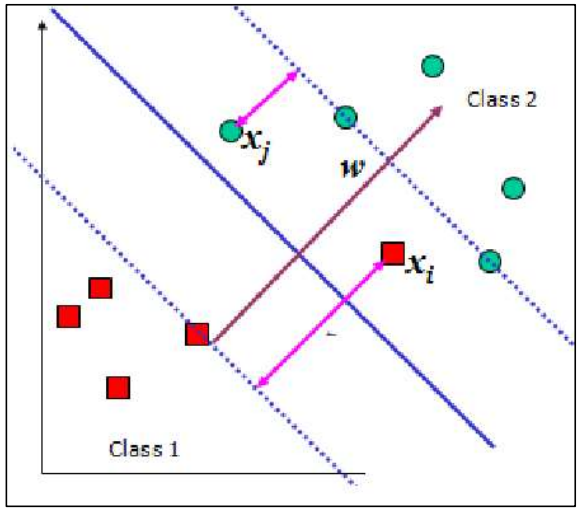
3. SELECTED HYPERPLANE based on margin

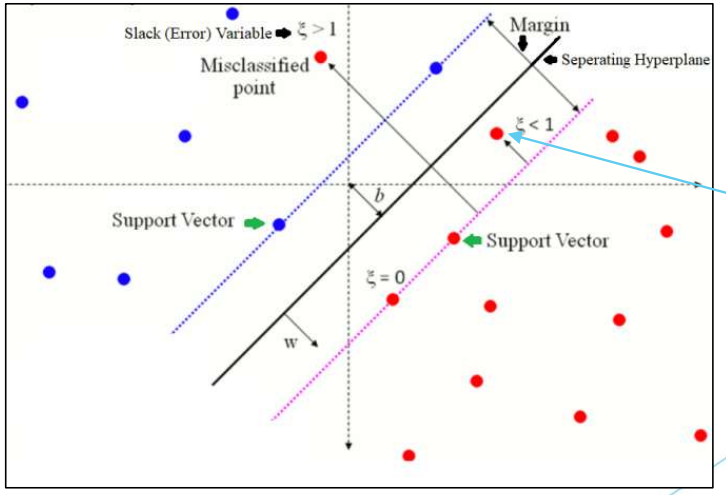


4. MISCLASSIFICATION HYPERPLANE

Soft margin SVM

If data is overlapping and hard to separate linearly then allow SVM to make a certain number of mistakes (i.e. margin violation) keeping margin as wide as possible so that other points can still be classified correctly. This is called soft margin.

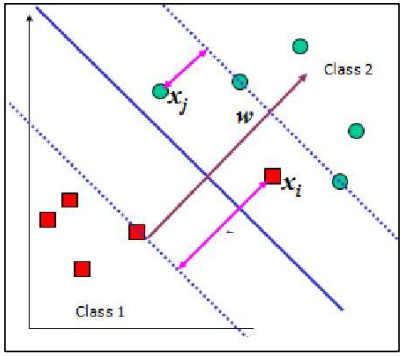


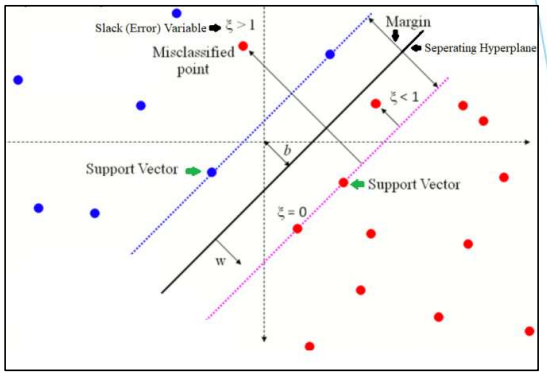


Sample violating margins

This can be done simply by modifying the constraint of objective function of SVM.

Soft margin SVM





$$(\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i})^* Y_i >= M (1-\xi_i)$$

ξ_i = Slack variable for ith variable

$\xi_i >= 0$

$\sum \xi_i <= C$

C = C is a hyperparameter that decides the trade-off between maximizing the margin and minimizing the mistakes

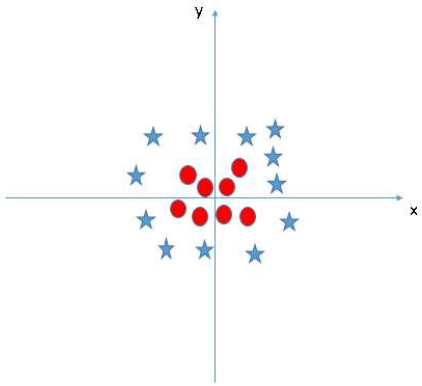
When C is small, classification mistakes are given less importance and focus is more on maximizing the margin, whereas when C is large, the focus is more on avoiding misclassification at the expense of keeping the margin small.

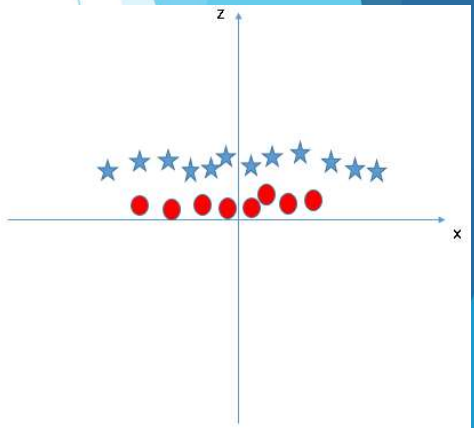
We also use ν as regularization parameter that limits (0,1) and ν = (A+B)/C

Kernels

WHAT IF POINTS ARE NOT LINEAR

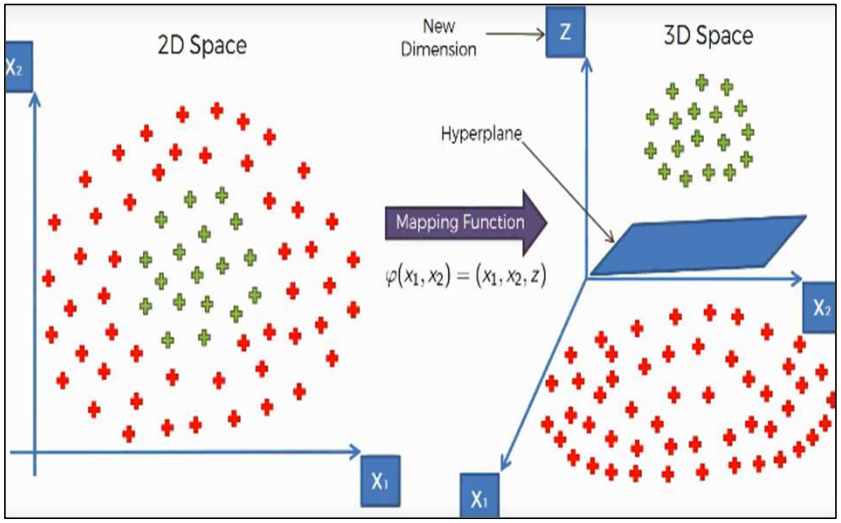
A liner hyperplane cannot separate training observations belonging to two different classes.

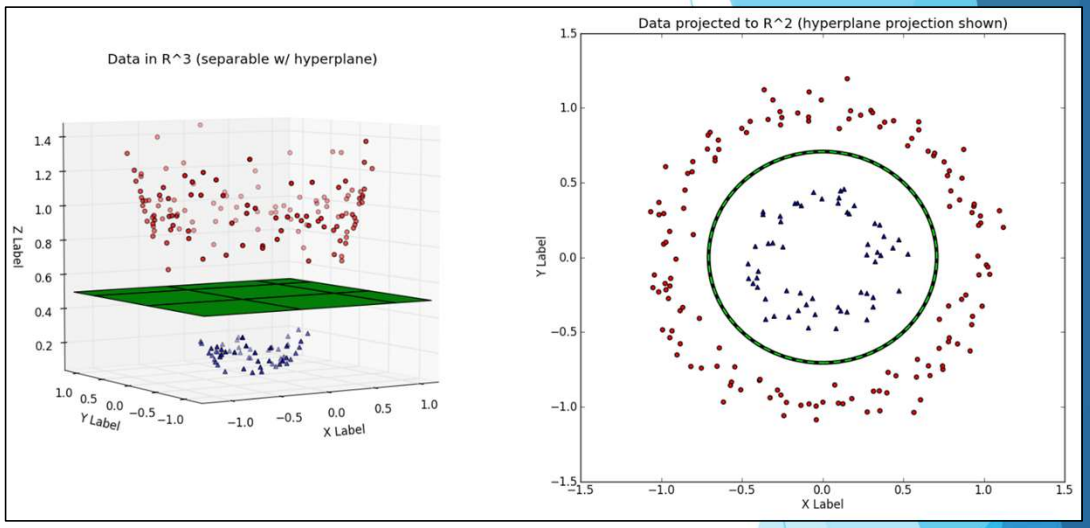




We can consider enlarging the feature space using functions of the predictors such as quadratic or cubic terms in order to address the non-linearity. E.g. we can convert 2 D data (x1,y1) , (x2,y2) into 3D data (X1,Y1,Z1) and (X2,Y2,Z2).

Kernels



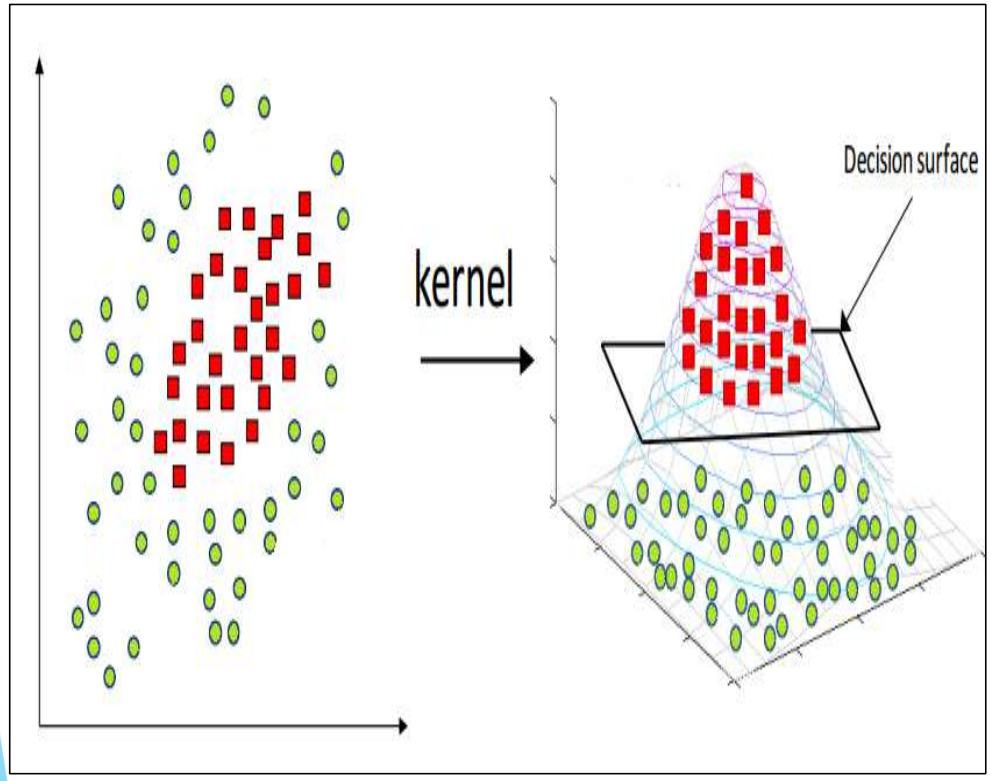


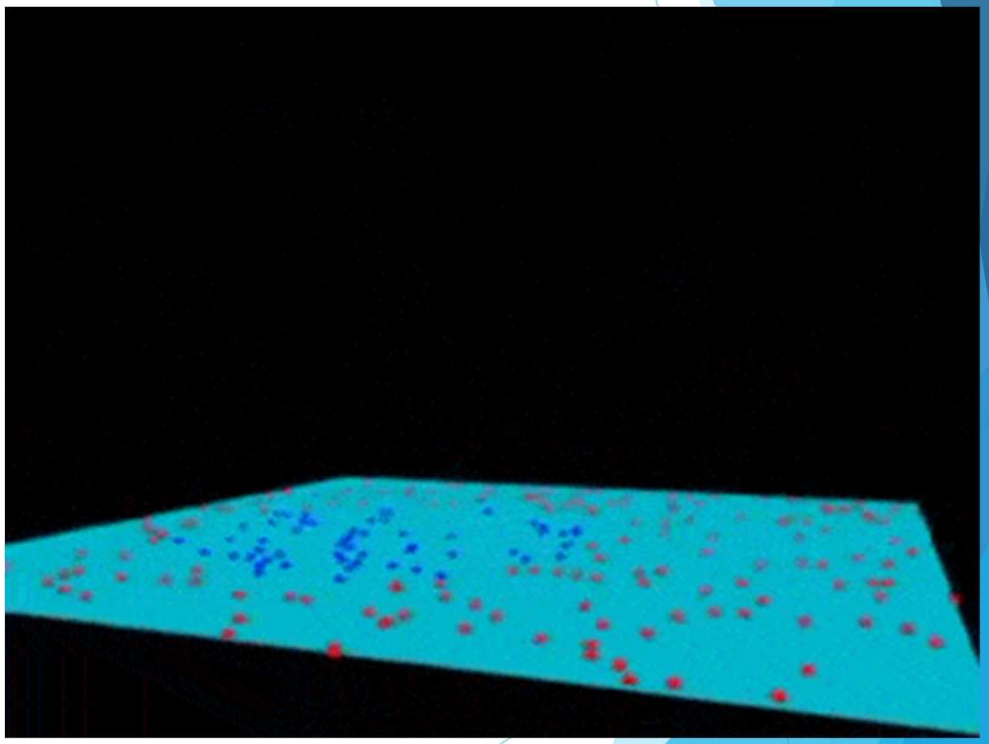
When we can easily separate data with hyperplane by drawing a straight line is **Linear SVM**. When we cannot separate data with a straight line we use **Non – Linear SVM**.

The change in feature space can be anything as long as it is successfully converting the space into higher dimensional space such that two classes can be separated using linear plane ($\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i}$).

DataDNA

Kernels

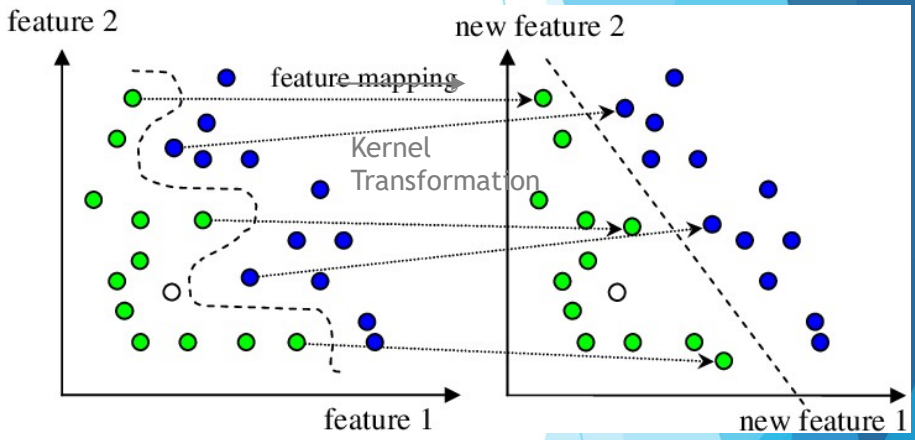




Kernels

In SVM, The kernel trick is an effective approach for enlarging the feature space.

X1, X2 → Kernel Transformation → X1,X2, X1`, X2`,X3`,X4`



The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space

How SVM Works

Most of the common Kernels available are:

- ✓ **Linear Kernel :** The decision boundary is a straight line. Unfortunately, most of the real-world data is not linearly separable, this is the reason the linear kernel is not widely used in SVM. Mostly preferred for text classification.

$$f(x) = sum(xi,xj))$$

- ✓ **Polynomial Kernel :**

$$K(xi,xj) = (1 + (xi * xj))^d$$

Sum of dot product between input observation (xi,xj) in training data.

This shows high dimensional relationship for xi and xj using Polynomial kernel and also this decides the decision boundary to separate the given classes.

DataDNA

How SVM Works

✓ **Gaussian Radial basis function kernel (RBF)**

$$K(xi,xj) = \exp(-\text{gamma} * ((xi - xj)^2)$$

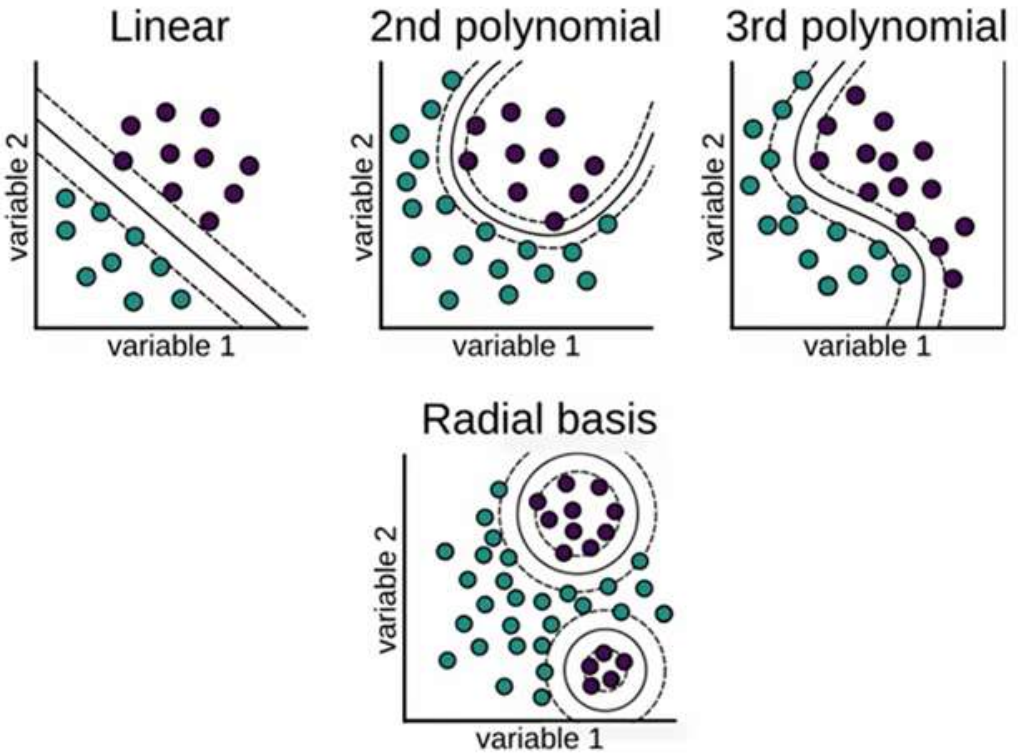
Gamma determines the influence of training observations to classify new observation. The value of gamma varies from 0 to 1

- Time of SVM learning: linear < poly < rbf
- Ability to fit any data: linear < poly < rbf
- Risk of overfitting: linear < poly < rbf
- Risk of underfitting: rbf < poly < linear
- we should start with linear kernel and try other kernels if performance is not good.

	Large Gamma	Small Gamma	Large C	Small C
Variance	Low	High	High	Low
Bias	High	Low	Low	High

How SVM Works

- Time of SVM learning: linear < poly < rbf
- Ability to fit any data: linear < poly < rbf
- Risk of overfitting: linear < poly < rbf
- Risk of underfitting: rbf < poly < linear
- we should start with linear kernel and try other kernels if performance is not good.



Demo

https://jgreitemann.github.io/svm-demo

Hyper parameter Tuning

C float, default=1.0
Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used.

Degree int, default=3
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma{'scale', 'auto'} or float, default='scale'

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if gamma='scale' (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.

Performance Validation

Performance validation of SVM

Customer	City	NoOfGamesPlayed	Channel	FavoriteGame	Actual	
1059	1	36	Favorite	Uniform	1	Training Data
1060	1	298	Favorite	Uniform	0	
1061	1	27	Uniform	Uniform	1	
1062	1	156	Favorite	Uniform	0	
1063	1	217	Favorite	Uniform	0	
1064	1	51	Favorite	Uniform	0	
1065	1	30	Favorite	Uniform	1	
1066	2	74	Favorite	Uniform	0	
1067	2	37	Favorite	Uniform	0	
1068	2	110	Favorite	Uniform	0	
1069	2	140	Favorite	Uniform	0	
1070	2	60	Favorite	Uniform	1	
1071	2	75	Favorite	Uniform	0	Test Data
1072	2	116	Favorite	Uniform	0	
1073	2	171	Favorite	Uniform	1	
1074	2	67	Favorite	Uniform	0	
1075	2	16	Favorite	Uniform	1	
1076	2	121	Uniform	Uniform	0	
1077	2	78	Favorite	Uniform	1	

Modelling on Training Data

ML Model

Apply on Test Data

Prediction

Customer	City	NoOfGamesPlayed	Channel	FavoriteGame	Actual	Prediction	
1071	2	75	Favorite	Uniform	0	1	Test Data
1072	2	116	Favorite	Uniform	0	1	
1073	2	171	Favorite	Uniform	1	0	
1074	2	67	Favorite	Uniform	0	1	
1075	2	16	Favorite	Uniform	1	1	
1076	2	121	Uniform	Uniform	0	0	
1077	2	78	Favorite	Uniform	1	1	