**SVM**

Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems

‘’’’’’’’’’’’’’’key points which kernel to use’’’’’’’’

* Use linear kernel when number of features is larger than number of observations.
* Use gaussian kernel when number of observations is larger than number of features.
* If number of observations is larger than 50,000 speed could be an issue when using gaussian kernel; hence, one might want to use linear kernel.
* Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set. ... Training a SVM with a Linear Kernel is Faster than with any other Kernel

5. Support Vector Machines [25 marks] You have been asked to tackle three independent problems using Support Vector Machines:

1- a regression-style problem

2- a multi-class classification problem: recognition of images of handwritten numbers: 0-9; and

3- a binary classification problem on a wide, but sparse data set

1. Suggest an appropriate kernel to use with each problem and state why it is suitable [12 Marks]

1- a regression-style problem

IN THIS WE ARE USING LINEAR KERNAL

**Linear Kernel** is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common **kernels** to be used. It is mostly used when there are a Large number of Features in a particular Data Set. ... Training a **SVM** with a **Linear Kernel** is Faster than with any other **Kernel**

Now applying this to a **regression problem**, linear **regression** could be described as an attempt to draw a line (or similarly plane or hyperplane in higher dimensions) that minimizes the error(or the loss function).

2- a multi-class classification problem: recognition of images of handwritten numbers: 0-9; and

Some [classification algorithms](https://www.sciencedirect.com/topics/computer-science/classification-algorithm), such as [support vector machines](https://www.sciencedirect.com/topics/computer-science/support-vector-machines), are designed for [binary classification](https://www.sciencedirect.com/topics/computer-science/binary-classification). How can we extend these algorithms to allow for [multiclass classification](https://www.sciencedirect.com/topics/computer-science/multiclass-classification) (i.e., classification involving more than two classes)?

A simple approach is **one-versus-all** (OVA). Given m classes, we train m binary classifiers, one for each class. Classifier j is trained using tuples of class j as the positive class, and the remaining tuples as the negative class. It learns to return a positive value for class j and a negative value for the rest. To classify an unknown tuple, ***X***, the set of classifiers vote as an ensemble. For example, if classifier j predicts the positive class for ***X***, then class j gets one vote. If it predicts the negative class for ***X***, then each of the classes except j gets one vote. The class with the most votes is assigned to ***X***.

**All-versus-all** (AVA) is an alternative approach that learns a classifier for each pair of classes. Given m classes, we construct m(m−1)2binary classifiers. A classifier is trained using tuples of the two classes it should discriminate. To classify an unknown tuple, each classifier votes. The tuple is assigned the class with the maximum number of votes. All-versus-all tends to be superior to one-versus-all.

A problem with the previous schemes is that binary classifiers are sensitive to errors.

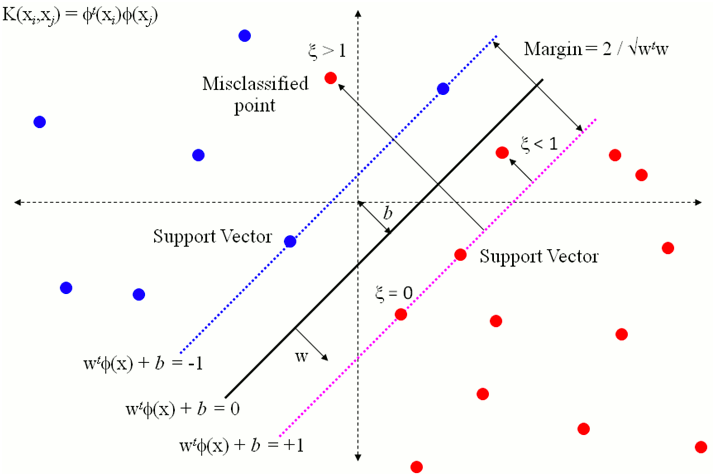
3. a binary classification problem on a wide, but sparse data set

**Algorithm**

First understand, in the case of binary classification WE USE linear SVM

:

1. Identify the correct hyperplane which segregates the two classes better.
2. Look for the maximum distance between nearest data point (of either any class) and hyperplane, the distance is measured as margin. So look for hyperplane with maximum margin both sides equally. Hyperplane with higher margin is more robust, whereas low margin has changed for misclassification.
3. SVM selects the classifier accurately to maximized margin.
4. SVM is robust to the classifier and have a feature to ignore outliers and try to look for a hyperplane with maximum margin.



1. Support Vector Machines can be impacted by the curse of dimensionality. In which of the 3 problems above is this most likely to be an issue. Discuss. [5 Marks]

In a binary classification problem on a wide, but sparse data set the curse of dimensionality will be high as it contains sparse dataset

The common theme of these problems is that when the **dimensionality increases**, the volume of the space **increases** so fast that the available **data** become **sparse**. This sparsity is problematic for any method that requires statistical significance

4 c) Describe, highlighting any assumptions you make and discussing their implications, how you would handle the curse of dimensionality for one of the problems you noted in part b as suffering from the curse of dimensionality. [8 Marks] 6. Text Mining [25

3.methods for dimensionality reduction like (linear/nonlinear) principal component analysis or manifold learning are used for transformation in a lower dimensional space

## Mini-batch learning with feature hashing

The following code demonstrates practical implementation of iterative training with feature hashing. The function train(model, lines) updates the model over 1 epoch using mini-batch training. The training dataset is stored in lines as a list of strings with each string being a record. It is a generic function that is application to any model class within sklearn that supports partial\_fit() method. Popular classification model choices that support partial\_fit() are logistic regression, support vector machines, and artificial neural networks. This function supports minibatch training; each batch of size 1000 samples. In each iteration, the function constructs a pandas DataFrame out of 1000 samples from lines, computes the feature hashing according to a pre-defined feature dimensions, and updates the model.

The second function hash\_representation(df) is used by train() function. It computes feature hashing on the input strings which are stored as a column in the input df. The computed feature representation is concatenated with the DataFrame df and is returned.

Curse of dimensionality

The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces (often with hundreds or thousands of dimensions) that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience.

In that case, SVM with linear kernel seems to be fine since the decision boundary is linear and everything can be run through relatively quickly.

However, for SVM with non-linear kernel such as RBF, since the decision boundary is non-linear and can take any arbitrary shape, solving kernelized SVM can take a huge amount of time and is subject to over-fitting. You can find a lot of references in this area.