

Title: Support Vector Machine (SVM) Algorithm

School: Computer Science and Engineering

Major: Computer Science and Technology

Author Name: Mamo Tadiyos Hailemichael

Teachers Name: Associate Professor Xiaoyu Li

Course Title: Data Mining

27 May, 2020

Table of Contents

| Description | Page No. |
|--|----------|
| 01 Background of data mining and AI ----- | 3 |
| 02 Introduction of SVM----- | 6 |
| 03 Main principle of SVM algorithm ----- | 7 |
| 04 Mathematics Behind SVM ----- | 8 |
| 4.1 Linear Discriminants and Margin ----- | 8 |
| 4.2 Separating Hyper-plane----- | 8 |
| 4.3 Margin----- | 9 |
| 05 Advanced Classification Problem and Its Solution Using SVM----- | 9 |
| 06 Outlier Detection and Sensitivity problem of SVM----- | 11 |
| 07 Tuning Parameter for SVM Algorithm (Variations)----- | 12 |
| 7.1 Kernel ----- | 12 |
| 7.2 Regularization----- | 12 |
| 7.3 Gamma ----- | 14 |
| 7.4 Margin----- | 14 |
| 08 Advantages and Disadvantages of SVM----- | 15 |
| 8.1 Advantages of SVM----- | 15 |
| 8.2 Disadvantages of SVM----- | 16 |
| 09 Application of SVM----- | 17 |
| 10 Accomplishment by python ----- | 18 |
| Conclusion ----- | 23 |
| Acknowledgements----- | 24 |
| References----- | 24 |

1. Background of data mining and AI

In the **1990s**, the term "Data Mining" was introduced, but data mining is the evolution of a sector with an extensive history. Early techniques of identifying patterns in data include Bayes theorem (**1700s**), and the evolution of regression(**1800s**). The generation and growing power of computer science have boosted data collection, storage, and manipulation as data sets have broad in size and complexity level. Explicit hands-on data investigation has progressively been improved with indirect, automatic data processing, and other computer science discoveries such as neural networks, clustering, genetic algorithms (**1950s**), decision trees(**1960s**), and supporting vector machines (**1990s**).

Data mining origins are traced back to three family lines: Classical statistics, Artificial intelligence, and Machine learning.

Classical statistics: Are the basis of most technology on which data mining is built, such as regression analysis, standard deviation, standard distribution, standard variance, discriminatory analysis, cluster analysis, and confidence intervals. All of these are used to analyze data and data connection.

Artificial Intelligence (AI): Is based on heuristics as opposed to statistics. It tries to apply human-thought like processing to statistical problems. A specific AI concept was adopted by some high-end commercial products, such as query optimization modules for Relational Database Management System (RDBMS).

Machine Learning: Is a combination of statistics and AI. It might be considered as an evolution of AI because it mixes AI heuristics with complex statistical analysis. Machine learning tries to enable computer programs to know about the data they are studying so that programs make a distinct decision based on the characteristics of the data examined. It uses statistics for basic concepts and adding more AI heuristics and algorithms to accomplish its target.

The major reason that data mining has attracted attention is due to the wide availability of vast amounts of data, and the need for turning such data into useful information and knowledge. The knowledge gained can be used for applications ranging from risk monitoring, business management, production control, market analysis, engineering, and science exploration. In general, three types of data mining techniques are used: association, regression, and classification.

Association analysis: Is the discovery of association rules showing attribute-value conditions that occur frequently together in a given set of data. Association analysis is widely used to identify the correlation of individual products within shopping carts.

Regression analysis: Creates models that explain dependent variables through the analysis of independent variables. As an example, the prediction for a product's sales performance can be created by correlating the product price and the average customer income level.

Classification and prediction: Are the process of designing a set of models to predict the class of objects whose class label is unknown. The derived model may be represented in various forms, such as if-then rules, decision trees, or mathematical formulas. Classification can be used for predicting the class label of data objects. Prediction encompasses the identification of distribution trends based on the available data.

The data mining process consists of an iterative sequence of the following steps:

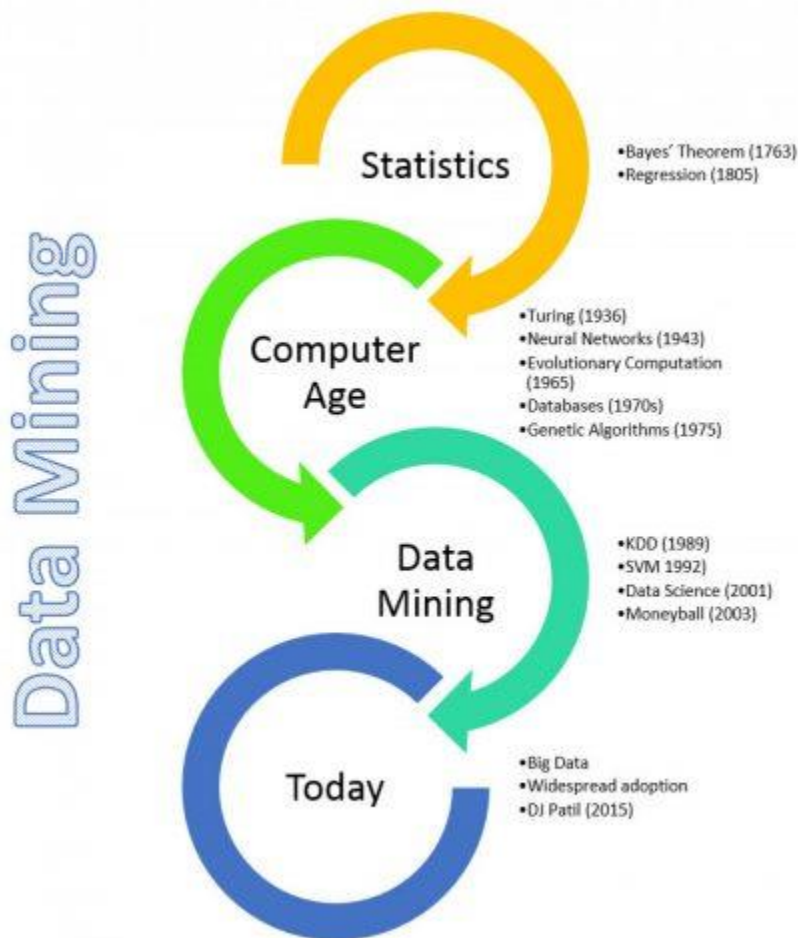
1. Data coherence and cleaning to remove noise and inconsistent data
2. Data integration such that multiple data sources may be combined
3. Data selection where data relevant to the analysis are retrieved
4. Data transformation where data are consolidated into forms appropriate for mining
5. Pattern recognition and statistical techniques are applied to extract patterns
6. Pattern evaluation to identify interesting patterns representing knowledge
7. Visualization techniques are used to present mined knowledge to users

Increasing power of technology and complexity of data sets has led Data Mining to evolve from static data delivery to more dynamic and proactive information deliveries; from tapes and disks to advanced algorithms and massive databases.

Artificial intelligence in data mining and big data techniques are wide utilized in several domain to resolve classification, planning, prediction, optimization problems, diagnosis, computation, collecting and analyzing customer information, gleaning insights into what customers want and need, and acting on those insights.

Data mining is the process of analyzing large data sets (Big Data) from different perspectives and uncovering correlations and patterns to summarize them into useful information. Nowadays it is blended with many techniques such as artificial intelligence, statistics, data science, database theory and machine learning.

We can briefly see the major milestones of data mining history on this chronological table below:



The popularity of data mining escalated notably in the 1990's, with the help of dedicated conferences, in addition to the fast increase in technology, data storage capabilities and computers' processing speeds. It was also possible for organizations to keep data in computer readable form and processing of large volumes of data using desk top machines were not far from reality.

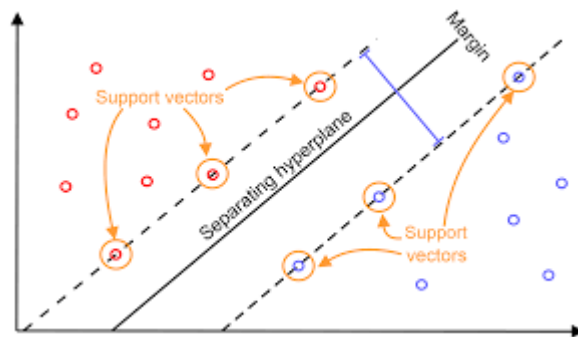
The main focus of data mining was tabular data; however, with the evolving technology and different needs new sources were formed to be mined!

1. **Text Mining:** Still a popular data mining activity, it categorizes or clusters large document collections such as news articles or web pages. Another application is opinion mining where the techniques are applied to obtain useful information from the questionnaire style data.
2. **Image Mining:** In image mining, mining techniques are applied to images (2D and 3D)
3. **Graph Mining:** It is formed from frequent pattern mining, which is focused on frequently occurring sub-graphs. A popular extension of graph mining is social network mining.

Data mining has become very popular over the last two decades as a discipline in its own. Data mining applications are used in every field of business, government, and science just to name a few. Starting from text mining, it has evolved a lot and it will be very interesting to watch with the usage of different data (e.g. spatial data, different sources of multimedia data) in the future.

2. Introduction of SVM

A Support Vector Machine (SVM) is supervised Machine Learning Algorithm which can be used for classification challenges. It uses a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side. A Support Vector Machine 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.



Hyper-planes are a decision boundary that help classify the data points. We can also say that the distance between the points and the line should be far as possible. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyper-plane depends upon the number of features. If the number of input features is 2, then the hyper-plane is just a line. If the number of input features is 3, then the hyper-plane becomes a two-dimensional plane.

| 1D Data | 2D Data | 3D Data | 4D Data or more |
|---|--|--|---|
| The support vector classifier is a point. | The support vector classifier is a line. | Three dimensional data, the support vector is a plane. | The support vector classifier is a hyper-plane. |

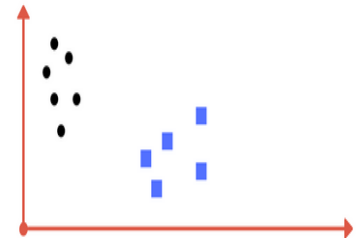
Generally, A hyperplane in an n -dimensional Euclidean space is a flat, $n-1$ dimensional subset of that space that divides the space into two disconnected parts. For example, let's assume a line to be our one-dimensional Euclidean space (i.e. let's say our datasets lie on a line). Now pick a point on the line, this point divides the line into two parts. The line has 1 dimension, while the point has 0 dimensions. So, a point is a hyperplane of the line. For two dimensions the separating line is the hyperplane. Similarly, for three dimensions a plane with two dimensions divides the 3d space into two parts and thus act as a hyperplane. Thus, for a space of n dimensions we have a hyperplane of $n-1$ dimensions separating it into two parts.

3. Main principle of SVM algorithm

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. Suppose we are given plot of two label classes on graph as shown in image (A). Can we decide a separating line for the classes?

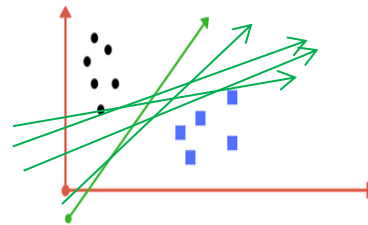
Image A:

- Draw a line that separates black circles and blue squares.



To separate the two classes of data points (black circle and blue square), there are many possible hyper-planes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes.

- ❖ Which one is the best possible line?



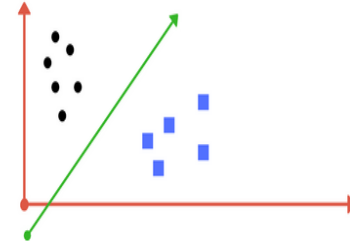
To select best line let's find maximum space:

Support vectors –closest vector or data point to line is important to select best line. Support vectors are data points that are closer to the hyper-plane and influence the position and orientation of the hyper-plane. Using these support vectors, we maximize the margin of the classifier. A key to this classifier's success is that for the fit, only the position of the support vectors matter; any points further from the margin which are on the correct side do not modify the fit! Technically, this is because these points do not contribute to the loss function used to fit the model, so their position and number do not matter so long as they do not cross the margin. Deleting these support vectors will change the position of the hyper-plane. Because these are the points that help us build our SVM. The line which has maximum space will be selected while the other line doesn't have the maximum space that separates the two classes. Shortest distance is important to sort the new data in SVM algorithm. Only support vectors are important. Finally, new data belongs to the class which has shortest distance support vector.

We might have come up with something similar to following image (image B). It fairly separates the two classes. Any point that is left of line falls into black circle class and on right falls into blue square class. **Separation of classes that's what SVM does.** It finds out a line/ hyper-plane (in multidimensional space that separate outs classes).

Image B:

- Sample cut to divide into two classes.



How do we know this one is best optimized line?

To answer this question let's see how the mathematics performing behind this svm algorithm:

4. Mathematics Behind SVM

4.1 Linear Discriminants

Let $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be a classification dataset, with n points in d -dimensional space.

Further, let us assume there are only two class labels, i.e., $y_i \in \{+1, -1\}$, denoting the positive and negative classes and the hyper-plane defined as follows:

$$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

$$= w_1 x_1 + w_2 x_2 + \dots + w_d x_d + b$$

Here, " \mathbf{w} " is a d -dimensional *weight vector*, and b is a scalar, called the *bias*. For points that lie on the hyper-plane, we have: $w_i x_i = -b$ or $x_i = \frac{-b}{w_i}$

The weight vector " \mathbf{w} " specifies the direction that is orthogonal or normal to the hyper-plane whereas the bias b fixes the offset of the hyper-plane in the d -dimensional space: $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = 0$

4.2 Separating Hyper-plane

If the input dataset is linearly separable, then we can find a *separating* hyper-plane $h(\mathbf{x}) = 0$, such that for all points labeled $y_i = -1$, we have $h(\mathbf{x}_i) < 0$, and for all points labeled $y_i = +1$, we have $h(\mathbf{x}_i) > 0$. The hyper-plane function $h(\mathbf{x})$ thus serves as a linear classifier or a linear discriminant, which predicts the class y for any given point \mathbf{x} , according to the decision rule:

$$y = \begin{cases} +1 & \text{if } h(\mathbf{x}) > 0 \\ -1 & \text{if } h(\mathbf{x}) < 0 \end{cases}$$

4.3 Margin

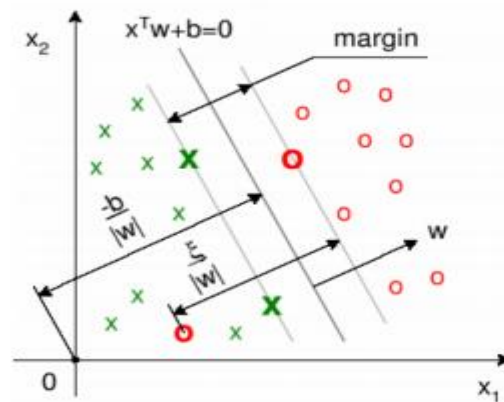
The **margin** is the minimum distance of a point from the separating hyper-plane: $\delta^* = \min_{x_i} \left\{ \frac{y_i(\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|} \right\}$

All the points (or vectors) that achieve the minimum distance are called support vectors for the hyper-plane. They satisfy the condition:

$$\delta^* = \frac{y^*(\mathbf{w}^T \mathbf{x}^* + b)}{\|\mathbf{w}\|}$$

where y^* is the class label for \mathbf{x}^* . The numerator $y^*(\mathbf{w}^T \mathbf{x}^* + b)$ gives the absolute distance of the support vector to the hyper-plane, whereas the denominator $\|\mathbf{w}\|$ makes it a relative distance in terms of \mathbf{w} .

Figure 1. Geometrical Representation of the SVM Margin

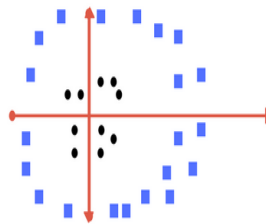


5. Advanced classification problem & Its solution using SVM

So far, we learned the first SVM type called Hard Margin Classifier (Linear Support Vector Machine). Which chooses the best hyper-plane to classify perfectly separable dataset by maximizing the margin. But not always the datasets are perfectly separable. Let's look at some data that is not linearly separable:

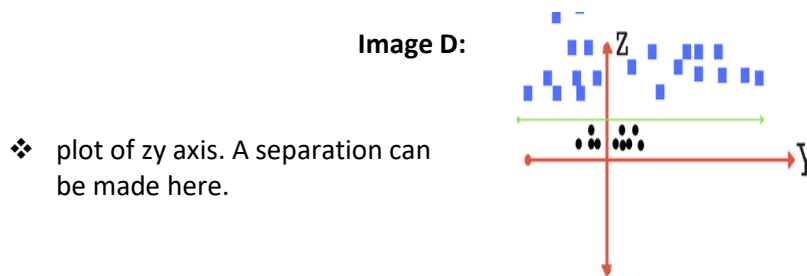
Image C:

- ❖ Can you draw a separating line in this plane?



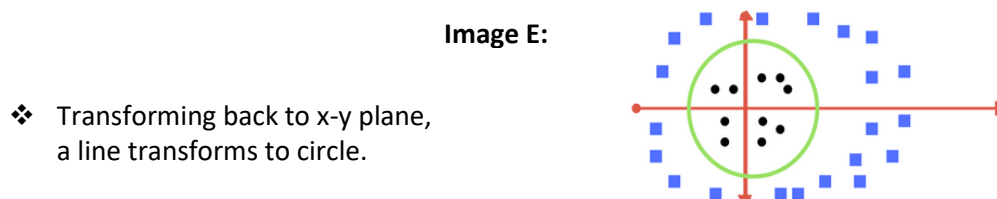
In this section we will look into issues of hard margin SVM and methods to solve these issues. Now consider what if we had data as shown in image above? Clearly, there is no line that can separate the two classes in this x-y plane. So, what do we do? Think about how we might project the data into a higher dimension such that a linear separator *would* be sufficient. For example, one simple projection we could use would be to compute a *radial basis function* centered on the middle clump:

We apply transformation and add one more dimension as we call it **z-axis**. Let's assume value of points on z plane, $w = x^2 + y^2$. In this case we can manipulate it as distance of point from z-origin. Now if we plot in z-axis, a clear separation is visible and a line can be drawn as follow:



Now the data is clearly linearly separable. Let the green line separating the data in higher dimension be $z=k$, where k is a constant. Since, $z=x^2+y^2$ we get $x^2 + y^2 = k$; which is an equation of a circle. Thus, we can classify data by adding an extra dimension to it so that it becomes linearly separable and then projecting the decision boundary back to original dimensions using mathematical transformation. But finding the correct transformation for any given dataset isn't that easy. Thankfully, we can use kernels in sklearn's SVM implementation to do this job.

Here we had to choose and carefully tune our projection: if we had not centered our radial basis function in the right location, we would not have seen such clean, linearly separable results. In general, the need to make such a choice is a problem: we would like to somehow automatically find the best basis functions to use. One strategy to this end is to compute a basis function centered at every point in the dataset, and let the SVM algorithm sift through the results. This type of basis function transformation is known as a kernel transformation, as it is based on a similarity relationship (or kernel) between each pair of points. A potential problem with this strategy—projecting N points into N dimensions is that it might become very computationally intensive as N grows large. However, because of a neat little procedure known as the **kernel trick**, a fit on kernel-transformed data can be done implicitly—that is, without ever building the full N -dimensional representation of the kernel projection! This kernel trick is built into the SVM, and is one of the reasons the method is so powerful. When we transform back this line to original plane, it maps to circular boundary as shown in image E. These transformations are called **kernels**.

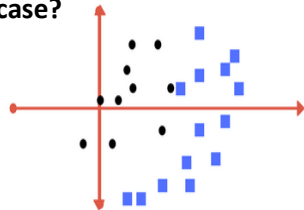


Using this kernelized support vector machine, we learn a suitable nonlinear decision boundary. This kernel transformation strategy is used often in machine learning to turn fast linear methods into fast nonlinear methods, especially for models in which the kernel trick can be used. Thankfully, we don't have to guess/ derive the transformation every time for your data set. The sklearn library's SVM implementation provides it inbuilt.

6. Outlier Detection & Sensitivity Problem of SVM

Our discussion thus far has centered around very clean datasets, in which a perfect decision boundary exists. But what if your data has some amount of overlap? Or, what in case some of the black points are inside the blue ones? For example, you may have data like this:

What in this case?



To handle this case, the SVM implementation has a bit of a fudge-factor which "softens" the margin: that is, it allows some of the points to creep into the margin if that allows a better fit. The hardness of the margin is controlled by a tuning parameter, most often known as **C**.

Which line among 1 or 2? should we draw?

Image 1:

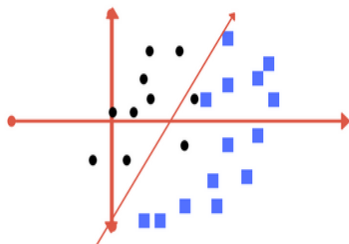
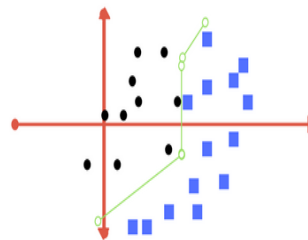


Image 2:



Which one do you think? Well, both the answers are correct. The first one tolerates some outlier points, which uses a **Soft margin** that would allow for some misclassification of data for better margin and decision boundary. The second one is trying to achieve 0 tolerance with perfect partition, which uses **Hard margin** SVM that have a strict rule that all dataset must be classified properly. **But there is trade off**. In real world application, finding perfect class for millions of training data set takes lot of time. This is called **regularization parameter**. In the section below, we define two terms **regularization parameter** and **gamma**. These are tuning parameters in SVM classifier. Varying those we can achieve considerable nonlinear classification line with more accuracy in reasonable amount of time. One more parameter is **kernel**. It defines whether we want a linear or nonlinear separation. This is also discussed in next section.

7. Tuning Parameters for SVM Algorithm (Variations)

7.1 Kernel

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

For **linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (x_i) is calculated as: $f(x) = B(0) + \sum (a_i * (x, x_i))$

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 $B(0)$ and a_i (for each input) must be estimated from the training data by the learning algorithm.

The **polynomial kernel** can be written as $K(x, x_i) = 1 + \sum (x * x_i)^d$. It is popular in image processing. Here, our kernel induces space of polynomial combinations of our features, up to certain degree. Consequently, we can work with slightly "bended" decision boundaries, such as parabolas with degree=2.

Gaussian Kernel (Exponential Kernel) can be written as $K(x, x_i) = \exp(-\gamma * \sum ((x - x_i)^2))$. It is a general-purpose kernel; used when there is no prior knowledge about the data. Polynomial and exponential kernels calculate separation line in higher dimension. This is called **kernel trick**

Linear kernels $k(x, x') = x^T x'$

Polynomial kernels $k(x, x') = (1 + x^T x')^d$ for any $d > 0$

– Contains all polynomials terms up to degree d

Gaussian kernels $k(x, x') = \exp(-||x - x'||^2 / 2\sigma^2)$ for $\sigma > 0$

– Infinite dimensional feature space

7.2 Regularization

The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example by softening the margin. Here the scalar C is a regularization constant that controls the trade-off between maximizing the margin or minimizing the loss.

How do we know the soft Margin?

The answer is simple we need to determine how many misclassification and observation to allow inside of the soft margin to get the best classification.

To achieve this, they introduced a variable called slack variable (denoted by ξ_i) into Objective Function:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

where $\xi_i \geq 0$ is the slack variable for point x_i , which indicates how much the point violates the separability condition.

The slack values indicate three types of points:

- If $\xi_i = 0$, then the corresponding point x_i is at least $1/|\mathbf{w}|$ away from the hyper-plane.
- If $0 < \xi_i < 1$, then the point is within the margin and still correctly classified, that is, it is on the correct side of the hyper-plane.
- However, if $\xi_i \geq 1$ then the point is misclassified and appears on the wrong side of the hyper-plane.

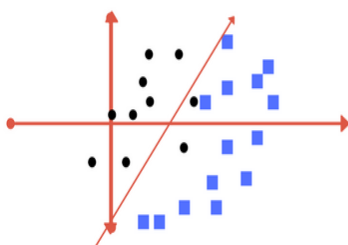
For large values of C ($C \rightarrow \infty$), the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Then the margin ceases to have much effect, and the objective function tries to minimize the loss.

Conversely, a very small value of C ($C \rightarrow 0$) will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. Then the loss component essentially disappears, and the objective defaults to maximizing the margin.

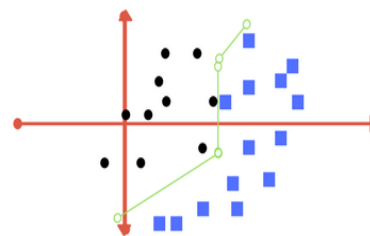
The complete minimization problem for soft margin SVM becomes:

$$\begin{aligned} \text{Objective Function: } \min_{\mathbf{w}, b, \xi_i} & \left\{ \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^n (\xi_i)^k \right\} \\ \text{Linear Constraints: } & y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall \mathbf{x}_i \in \mathbf{D} \\ & \xi_i \geq 0 \quad \forall \mathbf{x}_i \in \mathbf{D} \end{aligned}$$

The images below (same as image 1 and image 2 in above section 3) are example of two different regularization parameter. Left one has some misclassification due to lower regularization value. Higher value leads to results like right one.



Low regularization value

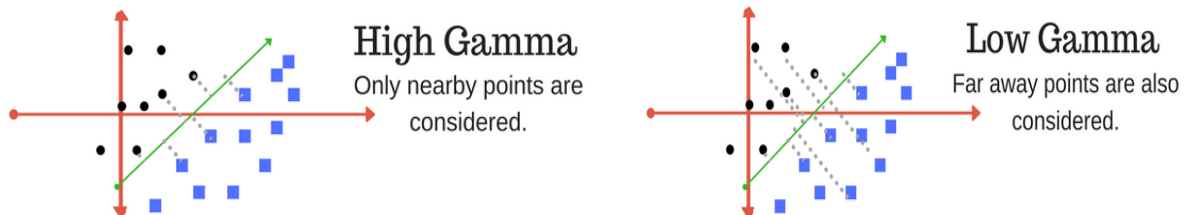


High regularization value

The optimal value of the C parameter will depend on your dataset, and should be tuned using cross-validation or a similar procedure.

7.3 Gamma

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.



7.4 Margin

And finally, last but very important characteristic of SVM classifier. SVM to core tries to achieve a good margin. A margin is a separation of line to the closest class points. A good margin is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.



8. Advantages and Disadvantages of SVM

8.1 Advantages of SVM:

- **Regularization capabilities:** SVM has L2 Regularization feature. So, it has good generalization capabilities which prevent it from over-fitting.
- **Handles non-linear data efficiently:** SVM can efficiently handle non-linear data using Kernel trick and it works relatively well when there is a clear margin of separation between classes.
- **Solves both Classification and Regression problems:** SVM can be used to solve both classification and regression problems. SVM is used for classification problems while SVR (Support Vector Regression) is used for regression problems.
- **Stability:** A small change to the data does not greatly affect the hyperplane and hence the SVM. So, the SVM model is stable.
- **Abundance of Implementations:** We can access it from Python or Matlab.
- **More effective in high dimensional space:** Because they are affected only by points near the margin, they work well with high-dimensional data—even data with more dimensions than samples, which is a challenging regime for other algorithms. It scales relatively well to high dimensional data than other algorithms.
- **Kernel trick is the real strength of SVM:** with appropriate kernel function we solve any complex problem and very good when we have no idea on the data. Works well with even unstructured and semi structured data like text, Images and trees. Their integration with kernel methods makes them very versatile, able to adapt to many types of data.
- **It takes a little memory space:** their dependence on relatively few support vectors means that they are very compact models, and take up very little memory and once the model is trained, the prediction phase is very fast.

8.2 Disadvantages of SVM:

- **Choosing an appropriate Kernel function is difficult:** Choosing an appropriate Kernel function (to handle the non-linear data) is not an easy task. It could be tricky and complex. In case of using a high dimension Kernel, you might generate too many support vectors which reduce the training speed drastically. So, generally choosing a “good” kernel function is not easy.
- **Extensive memory requirement:** This one is also mentioned in the advantage of svm part but due to the algorithmic complexity the memory requirements of SVM are very high. You need a lot of memory since you have to store all the support vectors in the memory and this number grows abruptly with the training dataset size.
- **Requires Feature Scaling:** One must do feature scaling of variables before applying SVM. Then in cases where number of features for each data point exceeds the number of training data sample, the SVM will underperform.
- **Long training time:** SVM takes a long training time on large datasets. So, SVM algorithm is not suitable for large data sets specially, when the data set has more noise i.e. target classes are overlapping. The scaling with the number of samples N is $O[N^3]$ at worst, or $O[N^2]$ for efficient implementations. For large numbers of training samples, this computational cost can be prohibitive.
- **Difficult to interpret:** It is difficult to understand and interpret the interpret the final model, variable weights and individual impact compared to Decision tree as SVM is more complex. Since the final model is not so easy to see, we cannot do small calibrations to the model hence it's tough to incorporate our business logic.
- **Difficult to tune the parameters:** The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact. The results are strongly dependent on a suitable choice for the softening parameter C. This must be carefully chosen via cross-validation, which can be expensive as datasets grow in size.
- **It's not a probabilistic algorithm:** As the support vector classifier works by putting data points, above and below the classifying hyper plane there is no probabilistic explanation for the classification. When classes in the data points are not well separated, which means overlapping classes are there, SVM does not perform well.

Applications of SVM

Face Detection:

- It classifies the parts of the image as face and non-face and create a square boundary around the face.
- It contains training data of $n \times n$ pixels with a two-class face (+1) and non-face (-1). Then it extracts features from each pixel as face or non-face. Creates a square boundary around faces on the basis of pixel brightness and classifies each image by using the same process.

Text & Hypertext categorization:

- SVM allows text and hypertext categorization for both types of models; inductive and transductive models.
- It Uses training data to classify documents into different categories such as news, articles, e-mails, and web pages.
- It categorizes on the basis of the score generated and then compares with the threshold value.

Classification of images:

- Use of SVMs provides better search accuracy for image classification.
- It provides better accuracy in comparison to the traditional query-based searching techniques.
- For e.g. Given an input image the classification task is to decide whether an image is a cat or a dog.

Bioinformatics:

- We use SVM for identifying the classification of genes, patients on the basis of genes and other biological problems.
- It includes protein classification and cancer classification

Protein fold and remote homology detection:

- Apply SVM algorithms for protein remote homology detection.
- We use SVM on protein structure prediction

Hand-Writing Recognition:

- We use SVM to recognize hand-written characters that used widely specially for data entry and validating signatures on documents.

Generalized Predictive Control (GPC):

- We use SVM-based GPC to control chaotic dynamics with useful parameters.
- It provides excellent performance in controlling the systems. The system follows chaotic dynamics with respect to the local stabilization of the target.

Accomplishment by python

Now let's tweak and play tuning parameters and implement a mini project for SVM classifier (also known as SVC) using python's sklearn library. Using SVM algorithm that classifies the email into spam and non-spam. In this exercise we shall train the model with set of emails labelled as either from Spam or Not Spam. There are 702 emails equally divided into spam and non-spam category. Next, we shall test the model on 260 emails. We shall ask model to predict the category of this emails and compare the accuracy with correct classification that we already know.

Prerequisite:

- Install Python.
- Install pip.
- Install sklearn for python: `pip install scikit-learn`
- Install numpy: `pip install numpy`
- Install SciPy: `pip install scipy`

Before we can apply the sklearn classifiers, we must clean the data. Cleaning involves removal of stop words, extracting most common words from text etc. In the code example concerned we perform following steps:

1. Build dictionary of words from email documents from training set.
2. Consider the most common 3000 words.
3. For each document in training set, create a frequency matrix for these words in dictionary and corresponding labels. [spam email file names start with prefix "spmsg".

1. Cleaning and Preparing the data

We have two folders **test-mails** and **train-mails**. We will use train-mails to train the model. The sample email data set looks like:

```
Subject: re: 2. 882 s - > np np
```

```
> deat: sun,15 dec 91 2:25:2 est >:michael <mmorse@vm1.yorku.ca>>
subject: re: 2 . 864 queries > > wlodek zadrozny ask " anything
interest " > construction " s > np np " . . . second, > much
reduplication? > logical sense " john mcnamara name " tautologous
thus, > level, indistinguishable ", here? "." john mcnamara name
attribute value. fact value name-attribute relevant entity ' chaim
shmendrik ', ' john mcnamara name ' false. Tautology,reduplication)
```

First line is subject and the rest are the content of the email message.

The very first step in text data mining task is to clean and prepare the data for a model.

In cleaning we remove the non-required words, expressions and symbols from text.

Consider a text: *“Hi, this is Alice. Hope you are doing well and enjoying your vacation.”*

Here the words like is, this, are, and etc. don't really contribute to the analysis. Such words are also called **stop words**. Hence in this implementation, we consider only most frequent 3000 words of dictionary from email. After cleaning what we need is every email document, we should be some matrix representation of the word frequency. For example, if document contains the text: “Hi, this is Alice. Happy Birthday Alice” after cleaning, we want something in line

| | | | | | | | |
|-----------|---|----|------|----|-------|-------|----------|
| word | : | Hi | this | is | Alice | Happy | Birthday |
| frequency | : | 1 | 1 | 1 | 2 | 1 | 1 |

And we need this for every document, the **extract_features** (section 2) function below does this and then removes fewer common words for every document.

```
def make_Dictionary(root_dir):
    all_words = []
    emails = [os.path.join(root_dir,f) for f in s.listdir(root_dir)]
    for mail in emails:
        with open(mail) as m:
            for line in m:
                words = line.split()
                all_words += words
    dictionary = Counter(all_words)
    # if you have python version 3.x use commented version.
    # list_to_remove = list(dictionary)
    list_to_remove = dictionary.keys()
    for item in list_to_remove:
        # remove if numerical.
        if item.isalpha() == False:
            del dictionary[item]
        elif len(item) == 1:
            del dictionary[item]
    # consider only most 3000 common words in dictionary.
    dictionary = dictionary.most_common(3000)
    return dictionary
```

This **make_Dictionary** function task is to reads the email files from a folder, constructs a dictionary for all words. Next, we delete words of length 1 and that are not purely alphabetical. At last we only extract out 3000 most common words.

2. Extracting features and corresponding label matrix.

After splitting the data set, the next steps include feature engineering. We will convert our text documents to a matrix of token counts (CountVectorizer), then transform a count matrix to a normalized representation. After that, we train several classifiers from Scikit-Learn library. Now with the help of dictionary, we generate a label and word frequency matrix:

```
word      :  Hi this is Alice Happy Birthday
frequency :  1  1  1  2      1      1

word      :  Hi this is Alice Happy Birthday
frequency :  1  1  1  2      1      1
```

```
def extract_features(mail_dir):
    files = [os.path.join(mail_dir,fi) for fi in s.listdir(mail_dir)]
    features_matrix = np.zeros((len(files),3000))
    train_labels = np.zeros(len(files))
    count = 0;
    docID = 0;
    for fil in files:
        with open(fil) as fi:
            for i,line in enumerate(fi):
                if i == 2:
                    words = line.split()
                    for word in words:
                        wordID = 0
                        for i,d in enumerate(dictionary):
                            if d[0] == word:
                                wordID = i
                                features_matrix[docID,wordID] = words.count(word)
    train_labels[docID] = 0;
    filepathTokens = fil.split('/')
    lastToken = filepathTokens[len(filepathTokens) - 1]
    if lastToken.startswith("spmsg"):
        train_labels[docID] = 1;
        count = count + 1
    docID = docID + 1
    return features_matrix, train_labels
```

After we have our features, we can train a classifier to try to predict the email. We will start with a support vector classifier(SVC) built-in module in scikit-learn includes several variants of this classifier on the next step.

3. Entering into world of SVC

We first import the svc from library. Next, we extract training features and labels. Lastly, we ask model to predict the labels for test set. The basic code block snippet looks like below:

```
from sklearn import svm
from sklearn.metrics import accuracy_score

TRAIN_DIR = "../train-mails"
TEST_DIR = "../test-mails"

dictionary = make_Dictionary(TRAIN_DIR)

print "reading and processing emails from file."
features_matrix, labels = extract_features(TRAIN_DIR)
test_feature_matrix, test_labels = extract_features(TEST_DIR)

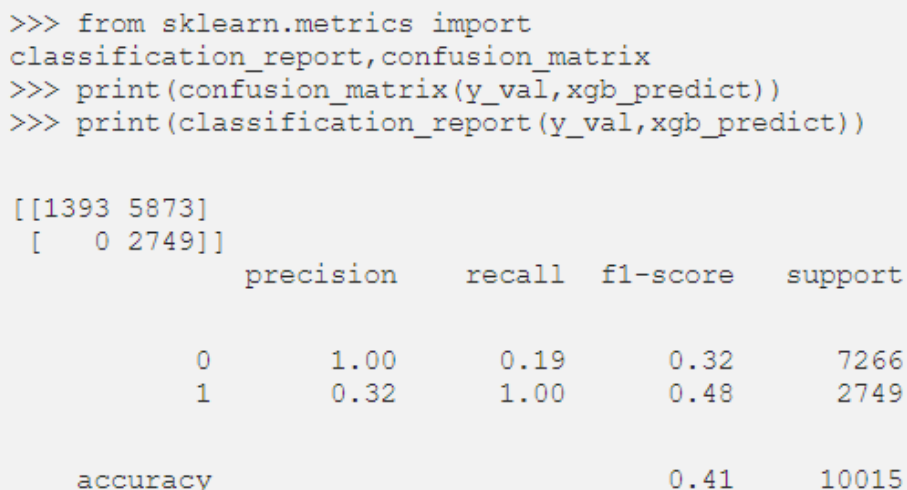
model = svm.SVC()

print "Training model."
#train model
model.fit(features_matrix, labels)

predicted_labels = model.predict(test_feature_matrix)

print "FINISHED classifying. accuracy score : "
print accuracy_score(test_labels, predicted_labels)
```

Output:



```
>>> from sklearn.metrics import
classification_report, confusion_matrix
>>> print(confusion_matrix(y_val, xgb_predict))
>>> print(classification_report(y_val, xgb_predict))
```

| | | | | |
|--------------|-----------|--------|----------|---------|
| [[1393 5873] | | | | |
| [0 2749]] | | | | |
| | precision | recall | f1-score | support |
| 0 | 1.00 | 0.19 | 0.32 | 7266 |
| 1 | 0.32 | 1.00 | 0.48 | 2749 |
| accuracy | | | 0.41 | 10015 |

This is very basic implementation. It assumes default values of tuning parameters (kernel = linear, C = 1 and gamma = 1). How do we decrease the training time? One way is to reduce the training set size. We shall try to reduce the training time by reducing the training data set size by 10% of original.

We then vary tuning parameters to increase the accuracy. We shall see how varying kernel, C and gamma changes accuracy and timings. Here we have 702 emails for training. 1/10 would mean 70 emails for training that is very less. By adding the following lines before training our model. (It reduces the feature_matrix and labels by 1/10th).

```
features_matrix = features_matrix[:len(features_matrix)/10]
labels = labels[:len(labels)/10]
```

Output:

```
[[12650 33578]
 [ 1972 31130]]
      precision    recall  f1-score   support

     0       0.87       0.27       0.42       46228
     1       0.48       0.94       0.64       33102

 accuracy                   0.55       79330
```

Parameter tuning: Now the training time and accuracy here we received around 56%. That's too low.

Now keeping the training set as 1/10th, let's try to tune three parameters: kernel, C and gamma:

1. Kernel: Change **kernel** to rbf. i.e. in model = SVC() add kernel parameter

```
model = svm.SVC(kernel="rbf", C = 1)
```

Output:

```
[[36926  9302]
 [12484 20618]]
      precision    recall  f1-score   support

     0       0.75       0.80       0.77       46228
     1       0.69       0.62       0.65       33102

 accuracy                   0.73       79330
```

2. C: Next vary **C (regularization parameter)** as **10, 100, 1000, 10000**. Determine whether accuracy increases or decreases?

```
model = svm.SVC(kernel="rbf", C = 100)
```

Here will notice that at C = 100, the accuracy score increases to 85.38% and remains almost same beyond that.

3. Gamma: At last, let's play with gamma. Try by adding one more parameter gamma = 1.0

```
model = svm.SVC(kernel="rbf", C=100, gamma=1)
```

A greater accuracy score can be achieved even after reducing the training data set to 1/10th.

But, why do we need to reduce training set?

Because there are applications where we need the prediction faster compared to accuracy:

- Think of credit card transaction. It is far more important to respond quickly for fraud flag of transaction than 99% accuracy. 90% accuracy can be tolerable here.
- On the other hand, only labelling emails into spam or not-spam may tolerate delays and we can strive for greater accuracy.

Do we need to tune parameters always? Not really. There are inbuilt functions in sklearn tool kit that does for us.

Conclusion

We have seen here a brief intuitive introduction with the details of mathematics and implementation example of principals behind support vector machines. Thus, we conclude that the SVMs can not only make the reliable prediction but also can reduce redundant information. The SVMs also obtained results comparable with those obtained by other approaches and these methods are a powerful classification method for a number of reasons that discussed in the advantages of svm part and those mentioned in disadvantages of svm can be slightly balanced by the fact, that SVM tend to perform better in the form of the unrestricted model. Support Vector Machines provides very simple method for linear classification. But, performance, in case of nonlinearly separable data, largely depends on the choice of kernel. So, support vector machines remain an important concept from the educational and theoretical point of view.

References

1. B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens and J. Vanthienen, 2003, Benchmarking State-of-the-art Classification Algorithms for Credit Scoring, Journal of the Operational Research Society (2003), 0, 1-9.
2. Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, A Practical Guide to Support Vector Classification, <http://www.csie.ntu.edu.tw>.
3. N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Repr. 2006, Cambridge University Press, 2000.
4. T. Van Gestel, B. Baesens, J. Garcia, P. Van Dijcke, A Support Vector Machine Approach to Credit Scoring, http://www.defaultrisk.com/pp_score_25.htm.
5. W. K. Härdle, R. A. Moro., D. Schäfer, Rating Companies with Support Vector Machines, DIW Discussion Paper No. 416, Berlin, 2004.
6. W. K. Härdle, R. A. Moro., D. Schäfer, Support Vector Machines – Eine neue Methode zum Rating von Unternehmen, DIW Wochenbericht No. 49/04, Berlin, 2004.