Frankfurt University of Applied Sciences

Machine Learning

Advanced Seminar Paper - Computation Intelligence

Department of Engineering FRA-UAS

Winter Semester 2015

Supervised by:

Prof. Dr. A. Pech

Prepared by:

Muhammad Ehsan-Ul-Haq

Matriculation: 1098587

Date: 7/12/2015

Signature: _____

Table of Contents

Table of figures

Part 1: Overview

Over the past several years, machine learning has evolved into main subareas of information technology and has considerable importance in solving real life issues. With the advances in information technology, more and more data is available at our hands and there is a good reason to believe that smart data analysis will become a more necessary ingredient of technological progress.

The main purpose of this paper is to cover an overview of variety of applications which machine learning can help solve and bring an order to the chaos. Later, I will discuss different types of problems and most common machine learning algorithms. Finally, I will outline a fairly basic, yet effective, algorithm to solve an important problem. Use of the sophisticated tool e.g. Matlab, discussion of the general problem, and a detailed analysis will follow at the end.

1.1 Introduction

Machine learning is evolved from artificial intelligence, which aims to give human abilities to the machines. The most important question in the field of machine learning is, "How to make machines able to learn?" Learning in this context is thought of as how to complete a task, or to make accurate predictions, or to behave intelligently. The learning that is being done is always based on the observation of some sort of data, direct experience or common observation. Therefore, machine learning is actually learning to do better in the future based on the observation of the past.

The emphasis of machine learning is on automatic methods. In other words, the goal is to devise an algorithm to complete learning on its own without human intervention. Machine learning can be viewed as "programming by example". Instead of programming the computer to solve a specific problem, we allow it to program itself based on the examples we provide.

Machine learning is considered a core subarea of artificial intelligence and it is highly unlikely that we build any kind of intelligent subsystem capable of facilities associated with intelligence, such as vision or language without using 'learning' to get there. Furthermore, we would not consider a system to be truly intelligent if it doesn't have ability to learn, since learning is the core of intelligence.

1.2 Supervised Learning

In supervised learning, the algorithm analyses the training data and produces an inferred function which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. Most common supervised learning problems are classification and regression.

1.2.1 Classification

The most common problem machine learning can be used to solve is 'classification'. Which is also referred to as pattern recognition. Pattern is defined as the inverse of Chaos. Examples of patterns are text documents, human faces, handwritten letters or digits, DNA sequences, ECG signals,

online transactions, human speech and weather behavior. More generally, the goal of classification task is to find functional mapping between input data *X* to a class label *Y*, such that *Y = f(X)* which is called *training data* given to classification algorithm [1].

Pattern classification tasks is usually divided into subtasks as following:

1. Data collection and representation
2. Feature selection or feature reduction
3. Classification

Data collection and representation is very much problem specific and hard to give a general statement about. Feature selection and feature reduction tries to reduce the dimensionality (number of features) for the next step. Finally, the classification phase finds actual mapping between patterns and labels.

1.2.2 Regression

Regression problems commonly involve prediction and forecasting. Regression analysis involves understanding how the typical value of dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed. Regression model involves the following variables

a. The unknown parameter *ß*
b. The independent variable *X*
c. The dependent variable *Y*

A regression model relates *Y* to a function of *X* and *ß* and approximated as $\boldsymbol{Y} \approx f(\boldsymbol{X}, \boldsymbol{\beta})$. The approximation is usually formalized as $E(Y \mid \mathbf{X}) = f(\boldsymbol{X}, \boldsymbol{\beta})$

1.2.3 Common Classification and Regression algorithms

Though machine learning is a relatively new field but still there exit tens of algorithms. Therefore, I have chosen few most common algorithms used for data analysis tasks e.g. classification or regression.

K-Nearest Neighbor (KNN) is arguably the simplest algorithm [2]. Here, the k points of training data closest to the test points are found and a label is given to the test point by a majority vote between the k points. This method is highly intuitive and attains (given its simplicity) remarkably low classification errors. The main disadvantage is that it is computationally expensive and requires large memory for training data.

Linear Discriminant Analysis (LDA*)* seeks to reduce the dimensionality by preserving as much of the class discriminatory information as possible. Actually, it computes a hyper plane in the input space which that minimizes the within class variance and maximizes the within class distance [3]. In linear cases it can be efficiently computed even with the large data sets. However, often linear separation is not enough. So, no linear extensions by using Kernels exist but make it hard to apply ti the problems with large data sets.

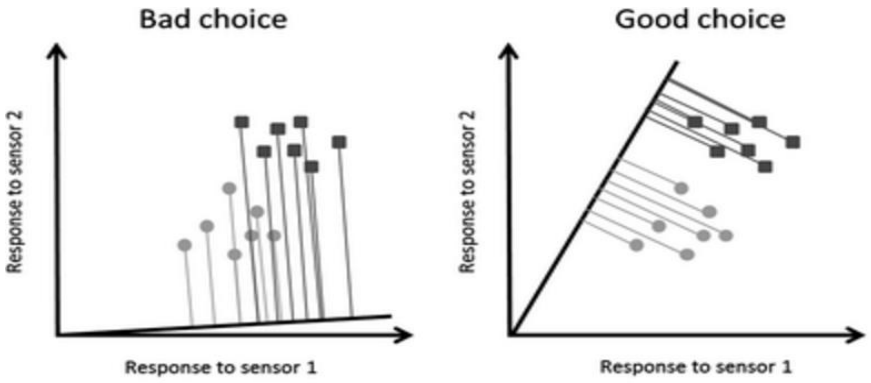

Figure 1: LDA Linar Discriminant Analysis using two different projections

Decision Trees (DTs) are another important classification algorithm which seeks to build the model in form of a tree structure. It breaks down data sets into smaller and smaller subsets, while at the same time an associated decision tree is incrementally developed. Final result is a tree with decision nodes and leaf nodes. Decision trees are simple yet effective classification schemes for small datasets. The computational complexity scales unfavorably with the number of dimensions of data. Mostly, large datasets tend to result in complicated trees which as a result require large memory of storage. Most common decision tree implementations are C4.5, C5.0 and Chi^2 [4].

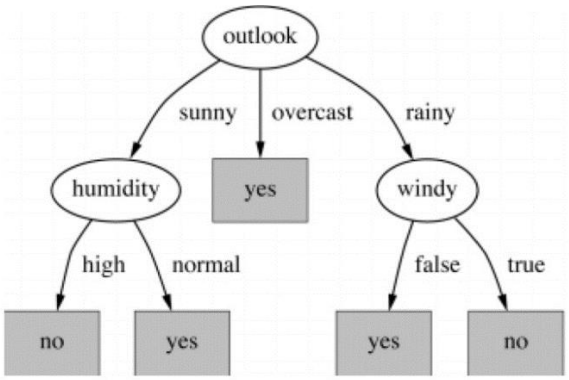

Figure 2: Decision tree example for playing golf w.r.t weather conditions

Neural Networks (NNs*)* are most commonly used approach for classification. Actually, this algorithm is inspired by the connectivity of neurons in our nervous system. Many tasks human tend to perform fast, such as recognition of face, proves to be very complicated task for computer when conventional programming methods are used. By applying neural network technique a program can learn by examples and create a structure of rules to classify different inputs e.g. recognizing images [5]. As illustrated in Fig 3. Each circle represents a computational element referred to as neuron, which computes weighted sum of its inputs and possibly performs nonlinear function on this sum. Neural networks can approximate any function provided that enough neurons exist in the network and enough training examples are provided.

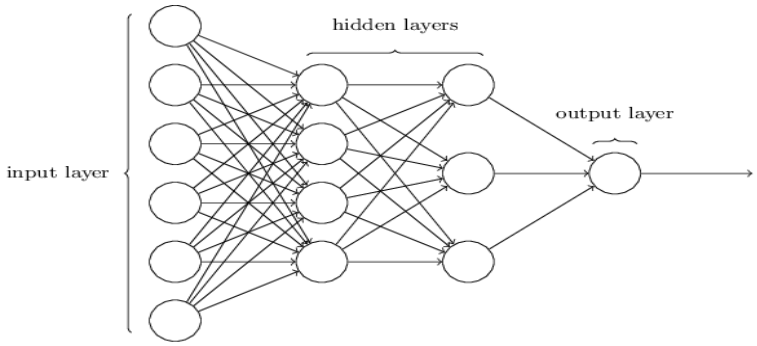

Figure 3: A schematic diagram of a neural network

Support Vector Machine works by mapping training data into the feature space by using so called Kernel functions and then separating data using a large margin hyper planes. Most commonly used Kernel functions are RBF and Polynomial Kernels. SVM are very effective in high dimensional spaces and still effective when dimensions are greater than number of samples. And they are also memory effective as compared to neural networks. But in cases where number of features are greater than number of samples, SVM give poor result. Despite all this, SVM gives the best results and is considered as one of the best classification algorithm.

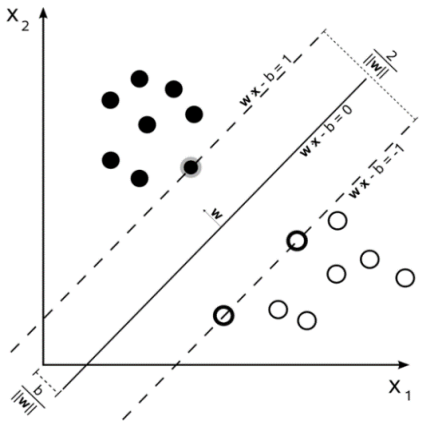

Figure 4: Maximum margin hyper planes and margins for SVM trained with samples for two classes

Random Forests is another efficient machine learning algorithm which is trade mark for an ensemble of decision trees. Unlike decision trees which suffer from high variance or bias, Random forests use averaging to find natural balance between two extremes. Moreover, random forest has few parameters to tune and can be simply used with default parameter settings. One of the main advantages of RF is that it provides effective methods for missing data and also offers an experimental method for detecting variable interactions. One of the disadvantage of this algorithm is that it seems to overfit for noisy datasets.

1.3 Unsupervised Learning

In machine learning unsupervised learning is trying to find hidden structure in unlabeled data. Since, the examples given to algorithm as input are unlabeled, so there is no error or reward signal to evaluate potential solution. The most common unsupervised learning method is clustering and the most famous algorithm which serves the purpose is K-Means.

K-Means Clustering is an unsupervised clustering method in which data is partitioned into k clusters based on their features. Each cluster is represented by it centroid and defined as center of points in the cluster and each point is assigned to the cluster whose center is the nearest. The main goal of algorithm is to minimize intra-cluster variance or the sum of the squares of the distances between data and the corresponding cluster centroid. The main advantage of this algorithm is that it is very simple and fast and the disadvantage is that it does not yield to same result with each run. Moreover you need to choose priori the number of clusters.

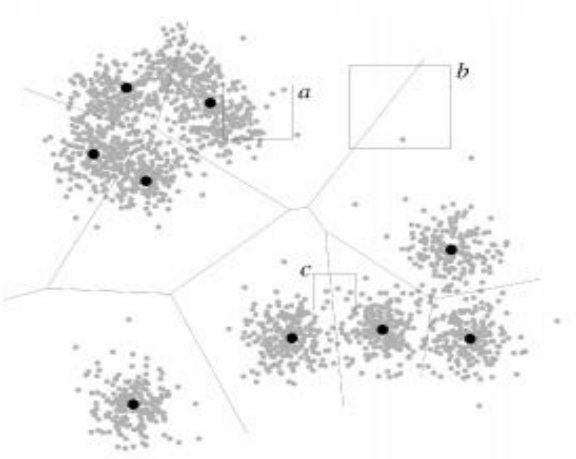

Figure 5: cluster formation with the help of K-Means algorithm

1.4 Research in Machine Learning

The main goal of research in machine learning is to develop efficient algorithms of practical value. The algorithms should not only be fast enough but also require less amounts of data. Each learning algorithm should be as general purpose as possible, to be able to applied broad class of learning problems. Another thing of primary importance is that we want the result of learning to be a prediction rule which is as accurate as possible in the prediction it makes. Sometimes we are also interested in interpretability of prediction rules produced by learning e.g. in medical diagnose we want the computer to find the prediction rules that is easily understandable by human experts.

Part 2: Selected Machine Learning Strategy

2.1 Support Vector Machines

Support Vector Machines (SVM) learning algorithms are considered among the best off shelf supervised learning algorithms. To understand about SVM, we need to first grasp the idea of maximum margins (to separate the data with maximum gap). Next, we will talk about optimal margin classifier which will lead us into Lagrange duality. Then we will discuss the idea of Kernels and how they effectively help SVMs to apply in infinitesimal dimensions. In the end, we will analyze the effectiveness of algorithm with some experimental results.

2.2 Introduction

The Support Vector Machines (SVMs) algorithm was first introduced in 1992 by Boser, Guyon, and Vapnik [6]. Due to its high accuracy and ability to deal with high dimensional data it has many p applications in identification, e-learning, text classification, image clustering, speech and handwriting recognition and in bioinformatics e.g. gene expression.

The efficiency of SVMs largely depends on numerous of factors and a deep understanding of how they work. When training an SVM the practitioner is required to make many decisions: How to process data, what kernel to use and finally setting the parameter of SVMs and Kernel.

2.3 Intuition

The main intuition comes from the idea of separation of two classes of data with maximum margin. Consider the following figure in which x's represent positive training examples and o's represent negative training examples, a decision boundary (a separating **hyperplane)** is also shown and three points A, B and C have also been labeled. Let's define the variable y = +1 for positive samples and -1 for negative samples.

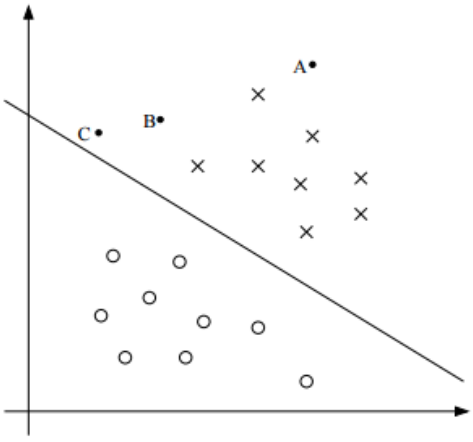

Figure 6: Intuitive example for data separation by a hyper plane

So, we can notice that point A is very far from separating boundary. If we are asked to make a prediction about separating boundary for the value of y at A. It seems that we should be quite confident that y = 1 there. Conversely, the point C is very close to decision boundary, and while it's on the side of boundary where we will predict y = 1 but it seems likely that a small change to decision boundary could have caused sample point to go from 1 to -1. Moreover point B lies in between and is less likely to be affected as compared to C. Broadly speaking, if a point is far from the separating hyperplane, then we may be significantly more confident in our predictions.

Let's consider the following figure, in which hyperplane is surrounded by two margin lines (also called gutter lines of the street) as represented in dotted form. Here $\vec{w}$ is a vector perpendicular to the margin line and $\vec{u}$ is some unknown point in the street.

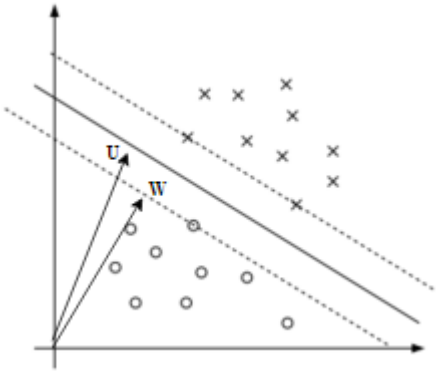

Figure 7: Positive and negative data points are separated by a hyperplane and two margin lines

So, in order to know whether it is on the positive side of street or negative side of street we can take dot product of two vectors as following

$$\vec{w}.\vec{u} \geq c \tag{2.1}$$

Here we can define certain value of c for positive and negative threshold, or in more generalized form we can write this equation in following form

$$\vec{w}.\vec{u} + b \geq 0 \tag{2.2}$$

Now, the equation 2.2 can be marked as **Decision Rule** for certain values of constant b and perpendicular vector $\vec{w}$**.** But we don't know yet what value of b and $\vec{w}$ to choose.

Instead of unknown vector $\vec{u}$, let's now take two vector $\overrightarrow{x_+}$ *and* $\overrightarrow{x_-}$ at positive and negative margin lines respectively and redefine our decision rule as following

$$\vec{w}.\overrightarrow{x_+} + b \geq 1 \tag{2.3}$$

$$\vec{w}.\overrightarrow{x_-} + b \geq -1 \tag{2.4}$$

According to above two equations we have set a threshold value for positive and negative margin lines with the values 1 and -1 respectively.

Now, let's generalize above two equations for all the sample points. In order to look our equations more mathematically convenient, let's take another variable y and define it such that

$$y_i = +1 \qquad \text{for all + sample points}$$

$$y_i = -1 \qquad \text{for all - sample points}$$

And, after multiplying this variable with equation 2.3 and 2.4 we get as following

$$y_i\ (\vec{w}.\vec{x_i} + b) \geq\ 1 \tag{2.5}$$

In more general form, we can rewrite above equation as following

$$y_i\ (\vec{w}.\vec{x_i} + b) - 1 \geq\ 0 \tag{2.6}$$

On margin lines :

$$\boxed{y_i\ (\vec{w}.\vec{x_i} + b) - 1 =\ 0} \tag{2.7}$$

Our main goal here is to maximize the separation between positive and negative margin lines. So, first we have to find the width of the separation. Let's consider the following figure.

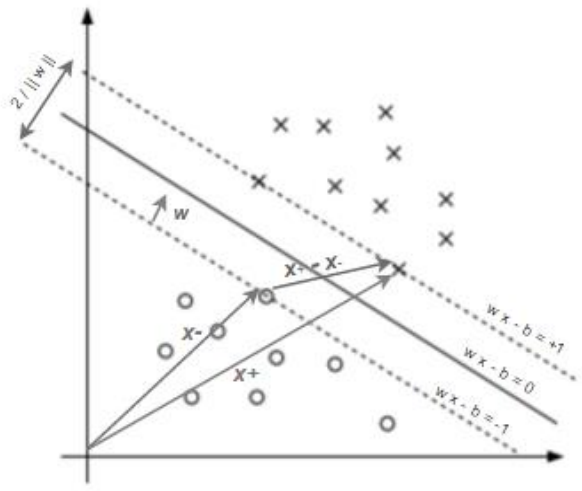

Figure 8: two vectors at margin lines, a difference vector and width of the margin

Let's consider two vectors x_+ and x_- at decision boundaries (margin lines) and their difference vector $x_+ - x_-$ as shown in the figure. We can get the width of street if we multiply difference vector with the unit normal vector $\vec{w}$ as following

$$\text{Width L} = (\overrightarrow{x_+} - \overrightarrow{x_-}).\ \frac{\vec{w}}{||w||} \tag{2.8}$$

By making use of equation 2.7, we can find that the dot product of difference vector and unit normal vector is as following

$$\text{Width L} = \frac{2}{||w||} \tag{2.9}$$

So, in order to maximize the width of the street we need to maximize = $\frac{2}{||w||}$ and noting that maximizing $\frac{1}{||w||}$ is the same thing as minimizing $\|w\|^2$, we have following relation

$$\boxed{\begin{aligned} &\min \quad \frac{1}{2}\|w\|^2 \\ s.t. \quad & y_i\,(\vec{w}.\overrightarrow{x_i} + b) - 1 \geq 0 \end{aligned}} \tag{2.10}$$

2.4 Lagrange Multipliers

As we know, Lagrange Multipliers help us to solve optimization problems where we want to maximize or minimize a function with constraints. First consider our problem, as we see in the figure 7, the points with the smallest margin are the ones closest to the decision boundary, these three points (one negative and two positive) lie on the dashed line parallel to the decision boundary thus only three of α_i's – namely, will be non-zero at the optimal solution of our optimization problem. These three points are called **Support Vectors.** The fact that the number of support vectors are much smaller than the size of training set will be very useful later.

Let's construct the Lagrangian for our optimization problem.

$$L(w,b) = \frac{1}{2}\,||\vec{w}||^2 - \sum_{i=1}^{m} \alpha_i[y_i(\vec{w}.\overrightarrow{x_i} + b) - 1] \tag{2.11}$$

Here, α_i's are called Lagrange multipliers, we would then find set of *L's* partial derivatives to zero:

$$\frac{\partial L}{\partial \vec{w}} = 0\ ;\ \frac{\partial L}{\partial b} = 0$$

And solving for *w* and *b* gives following.

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \tag{2.12}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.13}$$

Now, plugging the results from equations 2.12 and 2.13 back into equation 2.11, we get following

$$L = \frac{1}{2}\left(\sum_{i=1}^{m} \alpha_i y_i \vec{x}_i\right)\left(\sum_{j=1}^{m} \alpha_j y_j \vec{x}_j\right) - \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \,.\,(\sum_{j=1}^{m} \alpha_j y_j \vec{x}_j) - \sum_{i=1}^{m} \alpha_i y_i b + \sum_{i=1}^{m} \alpha_i$$

After simplifying we get following important result:

$$\boxed{L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j \, y_i y_j \, \vec{\boldsymbol{x}}_{\boldsymbol{i}} . \vec{\boldsymbol{x}}_{\boldsymbol{j}}} \tag{2.14}$$

So, from the result of 2.14 we can see that optimization depends only on dot product of pair of sample vectors. Moreover if we put value of $\vec{w}$ from equation 2.12 into our decision rule 2.2, we can observe that

For all + samples

$$\sum_i \alpha_i y_i \vec{\boldsymbol{x}}_{\boldsymbol{i}} . \vec{\boldsymbol{u}} + b \geq 0 \tag{2.15}$$

Only depends on dot product of two vectors. So, we can write entire algorithm in term of inner products between input feature vectors.

2.5 Kernels

Up until now, we have seen the cases where data is linearly separable. Let's now consider the case where data is non-linearly separable [7, 8]. In that case, we need to project all data points into another space where they are linearly separable. When that is mapped to new set of quantities that are then passed to the learning algorithm, calculate hyperplane and margin lines and then remap it back again to original data set.

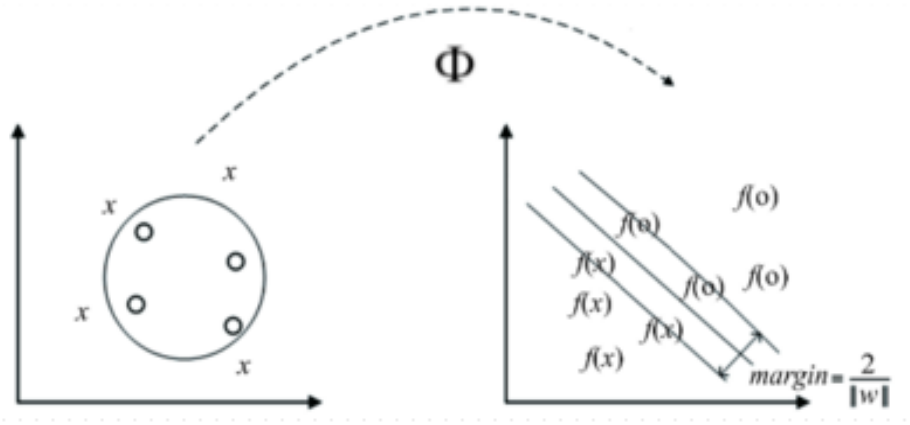

Figure 9: Non-Liner mapping in Support Vector machines

So, rather than applying SVM using original input attributes, we may want to learn using some features $\emptyset(x)$. To do so, we simply need to go over our previous algorithm and replace everywhere in it with $\emptyset(x)$.

Since, algorithm can be entirely written in terms of the inner products of $(\vec{x}_i, \vec{x}_j)$, this means that we would replace all those inner products with $(\emptyset(\vec{x}_i), \emptyset(\vec{x}_j))$. Specifically, given feature mapping $\emptyset$, we define the corresponding **Kernel** to be

$$K(\vec{x}_i, \vec{x}_j) = \emptyset(\vec{x}_i).\emptyset(\vec{x}_j) \tag{2.16}$$

So, everywhere we previously had $(\vec{x}_i, \vec{x}_j)$ in our algorithm, we could simply replace it with $K(\vec{x}_i, \vec{x}_j)$ and our algorithm would now be learning using features $\emptyset$. The benefit of using Kernel method is that you just don't need to know the transformation, all we need to know is the Kernel function which provides us the dot products into other space.

One can choose many types of kernels, ranging from simple polynomial basis function to sigmoid functions. One doesn't have to do the transformation beforehand, but simply has to select the appropriate Kernel function and the software will take care of transforming data, classifying it and retransforming it back to original space.

But unfortunately, with large number of attributes in data sets, sometimes it is difficult to know which Kernel would give best results. The most commonly used are Polynomial and Radial Basis. So, from a practical standpoint it is good idea to start with Polynomial or a Gaussian and work your way up in some more exotic Kernel function until we reach desired accuracy level [10].

A Kernel function can be interpreted as a type of similarity measure between input objects and therefore, it should be somehow possible to combine different similarity measures to create new Kernels. Following are some properties of Kernel functions.

1. $K(\vec{x}_i, \vec{x}_j) = c.K_1(\vec{x}_i, \vec{x}_j)$ (2.17)
2. $K(\vec{x}_i, \vec{x}_j) = c + K_1(\vec{x}_i, \vec{x}_j)$ (2.18)
3. $K(\vec{x}_i, \vec{x}_j) = K_1(\vec{x}_i, \vec{x}_j) + K_2(\vec{x}_i, \vec{x}_j)$ (2.19)
4. $K(\vec{x}_i, \vec{x}_j) = K_1(\vec{x}_i, \vec{x}_j).K_2(\vec{x}_i, \vec{x}_j)$ (2.20)
5. $K(\vec{x}_i, \vec{x}_j) = f(\vec{x}_i).f(\vec{x}_j)$ (2.21)

Although some Kernels are domain specific, there is, in general, no best choice. Since each Kernel has some degree of variability in practice there is nothing else for it but to experiment with some Kernels and adjust their parameters via model search to minimize the error on test set. Some of the most commonly used Kernels are shown in following table.

Type of Kernel	Inner product kernel $K(\vec{x}, \vec{x}_i),\ i = 1, 2, \ldots, N$	Comments
Polynomial Kernel	$K(\vec{x}, \vec{x}_i) = \left(\vec{x}^T \vec{x}_i + \theta\right)^d$	Power p and threshold θ is specified a priori by the user
Gaussian Kernel	$K(\vec{x}, \vec{x}_i) = e^{-\frac{1}{2\sigma^2}\|\vec{x}-\vec{x}_i\|^2}$	Width σ^2 is specified a priori by the user
Sigmoid Kernel	$K(\vec{x}, \vec{x}_i) = tanh(\eta\, \vec{x}\, \vec{x}_i + \theta)$	Mercer's Theorem is satisfied only for some values of η and θ
Kernels for Sets	$K(\chi, \chi') = \sum_{i=1}^{N_\chi} \sum_{j=1}^{N_{\chi'}} k(x_i, x'_j)$	Where $k(x_i, x'_j)$ is a kernel on elements in the sets χ, χ'
Spectrum Kernel for strings	count number of substrings in common	It is a kernel, since it is a dot product between vectors of indicators of all the substrings.

Table 1: Summary of inner product Kernels [9]

2.6 Soft Margin SVM

A standard SVM seeks to find a margin that separates all positive and negative examples. However, this can lead to poorly fit models if any examples are mislabeled or extremely unusual. In order to

account this, in 1995, Vapnik and Cortes proposed the idea of "soft margin" SVMs that allows some example sets to be ignored or placed on the wrong side of margin. This innovation leads to a better overall fit. C is the parameter of soft margin cost function which controls the influence of each individual support vector. A large C gives you low bias and high variance because you penalized the cost of misclassification a lot.

In order to control the sensitivity of SVM to possible outliers, we introduced the slack variables $\boldsymbol{\xi}$'s, and now we have slightly different form of maximum margin problem.

$$\begin{aligned} \min \quad & \frac{1}{2}\,||\mathrm{w}||^2 \;+\; C\sum \xi_i \\ s.t. \quad & y_i\,(\vec{w}.\vec{x_i} + b) - 1 \geq\; \xi_i \\ & \xi_i \geq 0 \; for\; all\; i \end{aligned} \tag{2.22}$$

Before this margin was defined by hard constraint, but now we allow some amount of slackness in constraint which is represented by ξ's. Also we can observe that if $0 < \xi \leq 1$, it means that data point lie somewhere in between the margin and correct side of hyperplane and if $\xi \leq 1$ then data point is misclassified.

We are also given a free parameter C which controls the relative importance of minimizing the norm of w, which is equivalent to maximizing the margin and satisfying the margin constraint for each data point.

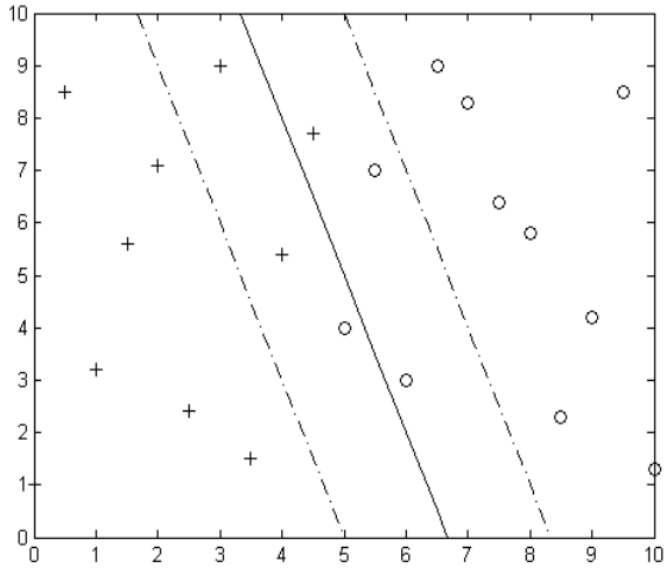

Figure 10: Soft Margin Support Vector Machines SVM with data sets crossing margin lines

If C is close to zero then we don't pay much for the points violating margin constraint. We can minimize cost function by setting W to be small vector. If C is close to infinity then we pay out a lot for the points that violate margin constraint.

We take Lagrangian as usual manner:

$$\max \; L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j \, y_i y_j \, \vec{\boldsymbol{x}}_{\boldsymbol{i}}.\vec{\boldsymbol{x}}_{\boldsymbol{j}}$$

$$\boldsymbol{s.t} \sum_{\boldsymbol{i}} \alpha_i y_i = \mathbf{0} \qquad (2.23)$$

$$0 \leq \alpha_i \leq C \;\; for\ all\ i$$

The soft margin SVM has quadratic objective that is equivalent to hard margin case. The only difference is that there is now a box constraint on α_i's. The influence of any point x_i on weight vector W is controlled by C. While in the hard margin case it was intact.

$\alpha_i = 0$	⇨	$y_i\,(\vec{w}.\vec{x_i} + b) \geq 1$	"easy"
$\alpha_i = C$	⇨	$y_i\,(\vec{w}.\vec{x_i} + b) \leq 1$	"hard"
$0 < \alpha_i < C$	⇨	$y_i\,(\vec{w}.\vec{x_i} + b) = 1$	"marginally hard"

Here again, α_i is non zero for support vectors. But support vectors now include not only data points on the margin but also data points within the margin and those on the wrong side of the boundary.

Part 3: Advantages and Limitations of SVMs

Mostly, all classification techniques have some advantages and disadvantages, which are more or less relevant to the data being analyzed, and therefore have relative relevance. Generally, choosing an algorithm depends on factors like, how large is the data set, training time and how much accuracy is required, or in other words how much error you can tolerate, or the tradeoff between any of these factors. Following are given some advantages and limitations of SVMs algorithm [11].

3.1 Advantages

Kernel trick:

Data sets with large dimensions can take a lot of computation power. That's where Kernels come in handy, all we need is just to choose appropriate function where transformed space is of high dimension, yet it is easy to compute similarity score in original space. So actually, we don't need to talk about transformation anymore.

No local minima:

SVMs are formulated as quadratic programming problem and there is global optimal solution. Therefore it never gets stuck into the local minima. But on the other hand, algorithms like multilayer neural networks are known to have many local minima.

Generalization:

SVMs provide a good out of sample generalization, if parameters r (in case of Gaussian Kernel) and C are chosen appropriately. It means by choosing an appropriate generalization grade, SVMs can be robust, even when training samples have some bias.

Robustness to outliers:

In case of SVMs the margin parameter C controls the misclassification error. If a large value is set to C, misclassification can be suppressed and if small value of C is set, training data that are away from gathered data are allowed to be misclassified. Thus properly setting the value of C one can suppress outliers.

3.2 Limitations

Extension to multiclass problem:

Support Vector Machines use direct decision functions, thus extension to multiclass problem is not straightforward unlike neural networks and involves several formulations.

Tricky Selection of parameters:

When training the data set, we need to choose an appropriate Kernel and its parameters and then we need to set the margin value of parameter C. To select the optimal parameters to a given problem is called model selection and it is done by estimating the generalization ability through repeatedly training support vector machines. Therefore, it is time consuming

Long training times for small feature dataset:

In case of data sets where number of features are smaller than training examples, SVMs take larger training time and give poor results. As shown in the following figure, the processing time is dramatically high where the number of features are small.

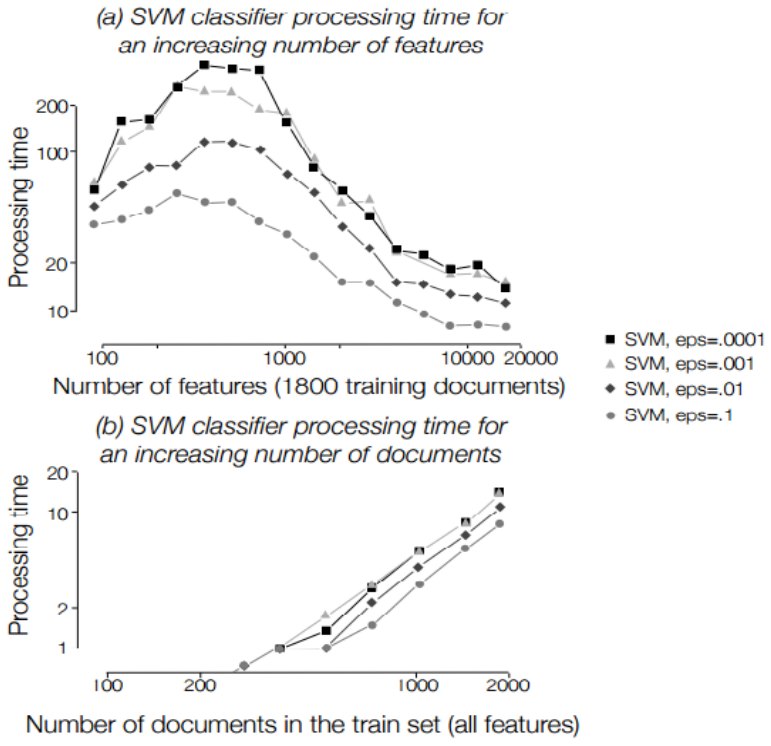

Figure 11: SVM processing time for increasing number of features (a) and increasing number of documents (b) for different values of ξ. Experiments were performed on the 20newsgroups dataset.

Part 4: Usage: List of scientific papers

In the early nineties, SVMs appeared as optimal margin classifier in the context of Vapnik's statistical learning theory. Since then, it has been successfully applied to real world data analysis problems and often provide much better results than traditional techniques. Following is the list of SVM based applications with reference to IEEE research papers.

1. Eye and Face Detection using SVM

In the real-time automatic face recognition system, accurate face detection is essential and very important part, because it has the effect on face recognition performance. The current paper describes to use color, edge, and binary information to detect regions of candidate eye pair from input image, then extract the face region with the detected eye pair. Then verify both eye pair regions and face region using Support Vector Machines (SVM). It is possible to perform fast and reliable face detection because we can protect false detection through this verification process. Experimental results demonstrate that the proposed approach shows very excellent face detection performance over 99.2%.

Source: Hyungkeun Jee, Kyunghee Lee and Sungbum Pan, "Eye and Face Detection using SVM," in Proceedings of IEEE International Professional Communication Conference, 2004, pp. 577–580.

2. Machine Vision

Classification of objects has been a significant area of concern in machine vision applications. In recent years, Support Vector Machines (SVM) is gaining popularity as an efficient data classification algorithm and is being widely used in many machine vision applications due to its good data generalization performance. The present paper describes the development of multi-class SVM classifier employing one-versus-one max-wins voting method and using Radial Basis Function (RBF) and Linear kernels. The developed classifiers have been applied for color-based classification of apple fruits into three pre-defined classes and their performance is compared with conventional K-Nearest Neighbor (KNN) and Naïve Bayes classifiers. The multi-class SVM classifier with RBF kernel has shown superior classification performance.

Source: J. Suriya Prakash, K. Annamalai Vignesh, C. Ashok and R. Adithyan, "Multi Class Support Vector Machines Classifier for Machine Vision Application," in Proceedings of IEEE International Professional Communication Conference, 2012, pp. 197–199.

3. Network Security Situation Awareness

Network Security Situation Awareness (NSSA) is an emerging technique in the field of network security and helps administrators to monitor the actual security situation of their networks. This

paper mainly focuses on NSSA based on heterogeneous multisensor data fusion. The technique used a model which adopted Snort and NetFlow as sensors to gather data from real network traffic. Then employed Support Vector Machines as the fusion engine of our model and used efficient feature reduction approach to fuse the gathered data from heterogeneous sensors. Furthermore, also discussed the alert aggregation and security awareness generation techniques in detail.

Source: Xiaowu Liu, Huiqiang Wang, Jibo Lai and Ying Liang, "Multiclass Support Vector Machines Theory and Its Data Fusion Application in Network Security Situation Awareness," in Proceedings of IEEE International Professional Communication Conference, 2007, pp. 6349–6352.

4 Stock Market Forecasting

Stock market forecasting has attracted a lot of research interests in previous literature, and recent studies have shown that artificial neural networks (ANN) method achieved better performance than traditional statistical ones. ANN approaches have, however, suffered from difficulties with generalization, producing models that can overfit the data. This paper employs a relatively new machine learning technique, support vector machines (SVM), to the stock market forecasting problem in an attempt to provide a model with better explanatory power. To evaluate the prediction accuracy of SVM, the technique is compared with three-layer fully connected back propagation neural networks (BNN). The experiment results show that SVM outperforms the BNN.

Source: Wang Zeng-min and Wu Chong, "Application of Support Vector Regression Method in Stock Market Forecasting," in Proceedings of IEEE International Professional Communication Conference, 2010, pp. 1-4.

5. Hyperspectral Remote Sensing

The principle of Support Vector Machine based on spot is to choose an appropriate scale to split the image into a series of segmentation, according to certain strategy using spectral information. And this principle ensures the spectral features of the majority of patch pixel similar. This method gathers statistics of each pixel value in the spot and obtains the mean value of each band to replace the original value of all pixels in the spot. The purpose of this classification is that the pixel having the noise brought by various causes is assimilated by the surrounding pixels to merge into a single spot. In other words, under the information of its surrounding pixels recovering the value of the pixel having noise is to not appear the fault is oblation in the classification map and to avoid the salt and pepper phenomenon. The results show that this method is feasible and the classification accuracy and speed is better than traditional support vector machine.

Source: Han Ling, Wu Jing and Zhang Ruolan, "The Classification Research of Support Vector Machine Based on Spot for Hyperspectral Remote Sensing Application," in Proceedings of IEEE International Professional Communication Conference, 2010, pp. 1009-1012.

Part 5: Code of the Matlab Example

Support Vector Machines can be used to detect objects in images. I have used Matlab to implement SVM to detect human faces in greyscale images. In order to train SVM to learn about facial features, I used 69 face and 59 non-face example images and to classify faces vs non-faces linear kernel is used. The program works as following:

1. First of all it creates gabor filter and feature image for facial edge detections. It contains 5 x 8 cells and each cell contains 32 x 32 matrix.

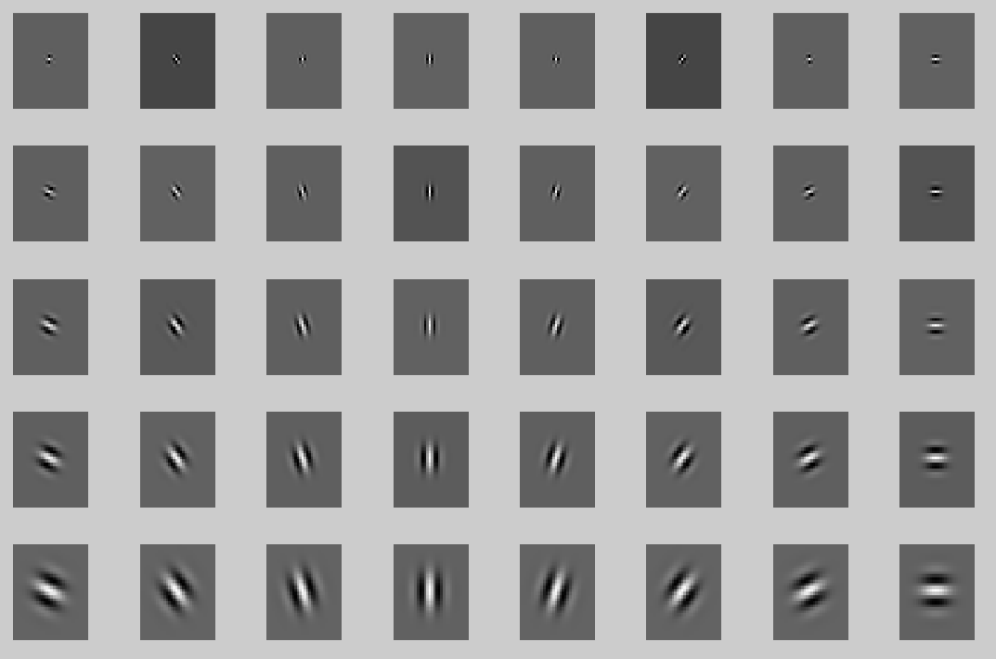

Figure 12: Feature image extracted using gabor filter

2. Then it loads example images in IMGDB database. Database contains three rows. 1st row contains images name string, 2nd row contains assigned value (1 for faces & 0 for non-faces) and 3rd row contains image data.

3. After that, SVM is trained with loaded example images using function *svmtrain()* and *svmclassify()* functions.

4. Then, test image file is uploaded and scanned for faces using *imscan()* function and detected images are identified with squares around the faces. [12][13][14][15]

Matlab code main file:

```
clear all;
close all;
clc;

if ~exist('gabor.mat','file')
    fprintf ('Creating Gabor Filters ...');
    create_gabor;
end
if exist('imgdb.mat','file')
    load imgdb;
else
    IMGDB = loadimages;
end
if exist('net.mat','file')
    load net;
end
while (1==1)
    choice=menu('Face Detection',...
                'Generate Database',...
                'Train SVM',...
                'Test on Photos',...
                'Exit');
    if (choice == 1)
        IMGDB = loadimages;
    end
    if (choice == 2)
        net = trainnet(IMGDB);
    end
    if (choice == 3)
        pause(0.01);
        [file_name, file_path] = uigetfile ('*.jpg');
        if file_path ~= 0
            im = imread ([file_path,file_name]);
            tic
            im_out = imscan (net,im);
            toc
            figure;
            imshow(im_out,'InitialMagnification','fit');
        end
    end
    if (choice == 4)
        clear all;
        clc;
        close all;
        return;
    end
end
```

SVM training code:

```
function NET = trainnet(IMGDB)

options = optimset('maxiter',100000);

fprintf('Creating & training the machine ->\n');
fprintf('Please wait for a while ...\n');

T = cell2mat(IMGDB(2,:));
P = cell2mat(IMGDB(3,:));

net = svmtrain(P',T','Kernel_Function','linear','quadprog_opts',options);
fprintf('Number of Support Vectors: %d\n',size(net.SupportVectors,1));

classes = svmclassify(net,P');
fprintf('done. %d \n',sum(abs(classes-T')));
save net net
NET = net;
```

Results of face detection:

Figure 13: Face detection using Matlab, SVM with Liner Kernel.

References

[1] Watanabe. Pattern recognition: Human and mechanical. Wiley, 1985.

[2] T.M. Cover and P.E. Hart. Nearest neighbor pattern classifications. IEEE transaction on information theory, 21—27, 1967.

[3] Fisher. The use of multiple measurements in taxonomic problems. Annals of Eugenics, 179–188, 1936.

[4] J.R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1992.

[5] S. Haykin. Neural Networks: A comprehensive foundation, 2nd Ed. Prentice-Hall, 1999.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144.

[7] T. Hastie, R. Tibshirani, and J.H. Friedman. The Elements of Statistical Learning. Springer, 2001

[8] C.M. Bishop. Pattern Recognition and Machine Learning. Springer, 2007

[9] Simon Haykin. Neural Networks: A Comprehensive Foundation (2nd Edition). Prentice Hall, 1998.

[10] U. M. Fayyad and R. Uthurusamy, editors. Extracting support data for a given task. AAAI Press, 1995.

[11] C. Cortes and V. Vapnik. Support vector networks. Machine Learning, 273-297, 1995.

[12] http://www.svms.org/

[13] http://www.kernel-machines.org/

[14] http://research.microsoft.com/pubs/67119/svmtutorial.pdf

[15] https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-choice/

[16] http://de.mathworks.com/help/images/index.html