

Tutorial on Support Vector Machine (SVM)

Vikramaditya Jakkula,
School of EECS,
Washington State University,
Pullman 99164.

Abstract: *In this tutorial we present a brief introduction to SVM, and we discuss about SVM from published papers, workshop materials & material collected from books and material available online on the World Wide Web. In the beginning we try to define SVM and try to talk as why SVM, with a brief overview of statistical learning theory. The mathematical formulation of SVM is presented, and theory for the implementation of SVM is briefly discussed. Finally some conclusions on SVM and application areas are included. Support Vector Machines (SVMs) are competing with Neural Networks as tools for solving pattern recognition problems. This tutorial assumes you are familiar with concepts of Linear Algebra, real analysis and also understand the working of neural networks and have some background in AI.*

Introduction

Machine Learning is considered as a subfield of Artificial Intelligence and it is concerned with the development of techniques and methods which enable the computer to learn. In simple terms development of algorithms which enable the machine to learn and perform tasks and activities. Machine learning overlaps with statistics in many ways. Over the period of time many techniques and methodologies were developed for machine learning tasks [1].

Support Vector Machine (SVM) was first heard in 1992, introduced by Boser, Guyon, and Vapnik in COLT-92. Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression [1]. They belong to a family of generalized linear classifiers. In another terms, Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. Support Vector machines can be defined as systems which use hypothesis space of a linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. Support vector machine was initially popular with the NIPS community and now is an active part of the machine learning research around the world. SVM becomes famous when, using pixel maps as input; it gives accuracy comparable to sophisticated neural networks with elaborated features in a handwriting recognition task [2]. It is also being used for many applications, such as hand writing analysis, face analysis and so forth, especially for pattern classification and regression based applications. The foundations of Support Vector Machines (SVM) have been developed by Vapnik [3] and gained popularity due to many promising features such as better empirical performance. The formulation uses the Structural Risk Minimization (SRM) principle, which has been shown to be superior, [4], to traditional Empirical Risk Minimization (ERM) principle, used by conventional neural networks. SRM minimizes an upper bound on the expected risk, where as ERM minimizes the error on the training data. It is this difference which equips SVM with a greater ability to generalize, which is the goal in statistical learning. SVMs were developed to solve the classification problem, but recently they have been extended to solve regression problems [5].

Statistical Learning Theory

The statistical learning theory provides a framework for studying the problem of gaining knowledge, making predictions, making decisions from a set of data. In simple terms, it enables the choosing of the hyper plane space such a way that it closely represents the underlying function in the target space [6].

In statistical learning theory the problem of supervised learning is formulated as follows. We are given a set of training data $\{(x_1, y_1) \dots (x_l, y_l)\}$ in $R^n \times R$ sampled according to unknown probability distribution $P(x, y)$, and a loss function $V(y, f(x))$ that measures the error, for a given x , $f(x)$ is "predicted" instead of the actual value y . The problem consists in finding a function f that minimizes the expectation of the error on new data that is, finding a function f that minimizes the expected error: $\int V(y, f(x)) P(x, y) dx dy$ [6] In statistical modeling we would choose a model from the hypothesis space, which is closest (with respect to some error measure) to the underlying function in the target space. More on statistical learning theory can be found on introduction to statistical learning theory [7].

Learning and Generalization

Early machine learning algorithms aimed to learn representations of simple functions. Hence, the goal of learning was to output a hypothesis that performed the correct classification of the training data and early learning algorithms were designed to find such an accurate fit to the data [8]. The ability of a hypothesis to correctly classify data not in the training set is known as its generalization. SVM performs better in term of not over generalization when the neural networks might end up over generalizing easily [11]. Another thing to observe is to find where to make the best trade-off in trading complexity with the number of epochs; the illustration brings to light more information about this. The below illustration is made from the class notes.

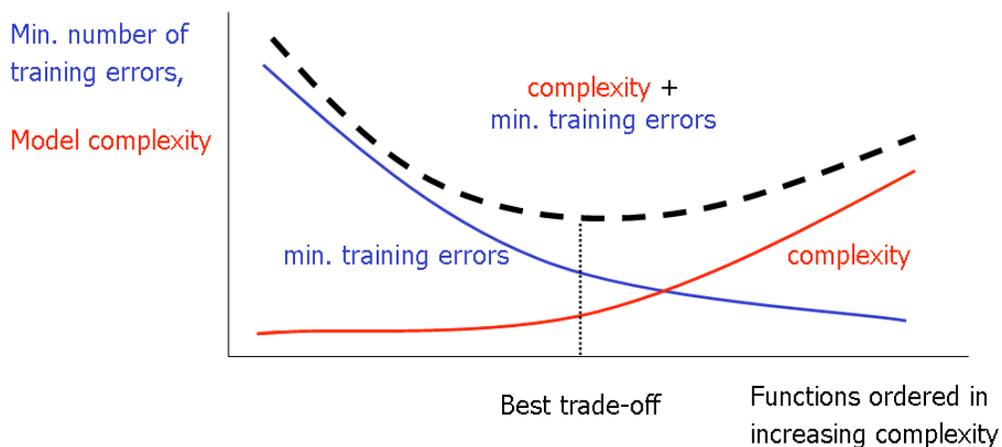


Figure 1: Number of Epochs Vs Complexity. [8][9][11]

Introduction to SVM: Why SVM?

Firstly working with neural networks for supervised and unsupervised learning showed good results while used for such learning applications. MLP's uses feed forward and recurrent networks. Multilayer perceptron (MLP) properties include universal approximation of continuous nonlinear functions and include learning with input-output patterns and also involve advanced network architectures with multiple inputs and outputs [10].

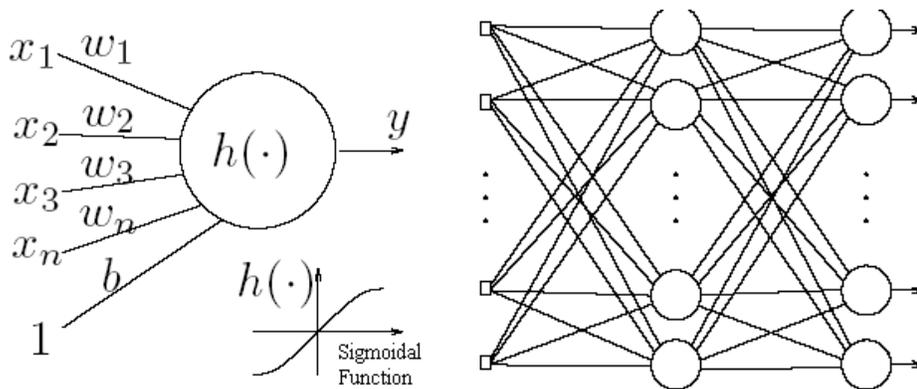


Figure 2: a) Simple Neural Network b) Multilayer Perceptron. [10][11]. These are simple visualizations just to have an overview as how neural network looks like.

There can be some issues noticed. Some of them are having many local minima and also finding how many neurons might be needed for a task is another issue which determines whether optimality of that NN is reached. Another thing to note is that even if the neural network solutions used tends to converge, this may not result in a unique solution [11]. Now let us look at another example where we plot the data and try to classify it and we see that there are many hyper planes which can classify it. But which one is better?

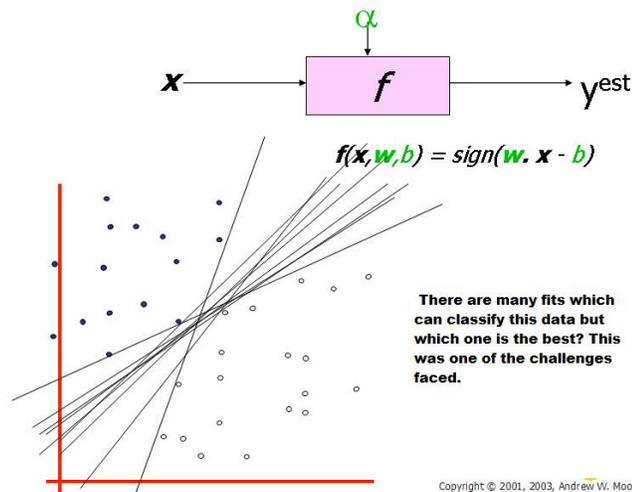


Figure 3: Here we see that there are many hyper planes which can be fit in to classify the data but which one is the best is the right or correct solution. The need for SVM arises. (Taken Andrew W. Moore 2003) [2]. Note the legend is not described as they are sample plotting to make understand the concepts involved.

From above illustration, there are many linear classifiers (hyper planes) that separate the data. However only one of these achieves maximum separation. The reason we need it is because if we use a hyper plane to classify, it might end up closer to one set of datasets compared to others and we do not want this to happen and thus we see that the concept of maximum margin classifier or hyper plane as an apparent solution. The next illustration gives the maximum margin classifier example which provides a solution to the above mentioned problem [8].

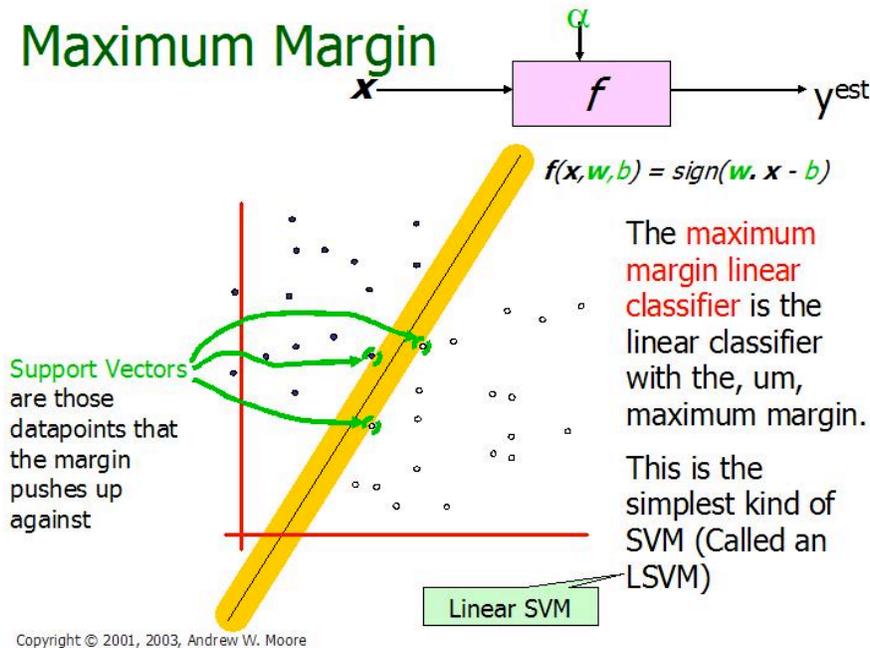


Figure 4: Illustration of Linear SVM. (Taken from Andrew W. Moore slides 2003) [2]. Note the legend is not described as they are sample plotting to make understand the concepts involved.

Expression for Maximum margin is given as [4][8] (for more information visit [4]):

$$\text{margin} = \arg \min_{x \in D} d(x) = \arg \min_{x \in D} \frac{|x \cdot w + b|}{\sqrt{\sum_{i=1}^d w_i^2}}$$

The above illustration is the maximum linear classifier with the maximum range. In this context it is an example of a simple linear SVM classifier. Another interesting question is why maximum margin? There are some good explanations which include better empirical performance. Another reason is that even if we've made a small error in the location of the boundary this gives us least chance of causing a misclassification. The other advantage would be avoiding local minima and better classification. Now we try to express the SVM mathematically and for this tutorial we try to present a linear SVM. The goals of SVM are separating the data with hyper plane and extend this to non-linear boundaries using kernel trick [8] [11]. For calculating the SVM we see that the goal is to correctly classify all the data. For mathematical calculations we have,

- [a] If $Y_i = +1$; $w x_i + b \geq 1$
- [b] If $Y_i = -1$; $w x_i + b \leq -1$
- [c] For all i ; $y_i (w x_i + b) \geq 1$

In this equation x is a vector point and w is weight and is also a vector. So to separate the data [a] should always be greater than zero. Among all possible hyper planes, SVM selects the one where the distance of hyper plane is as large as possible. If the training data is good and every test vector is located in radius r from training vector. Now if the chosen hyper plane is located at the farthest possible from the data [12]. This desired hyper plane which maximizes the margin also bisects the lines between closest points on convex hull of the two datasets. Thus we have [a], [b] & [c].

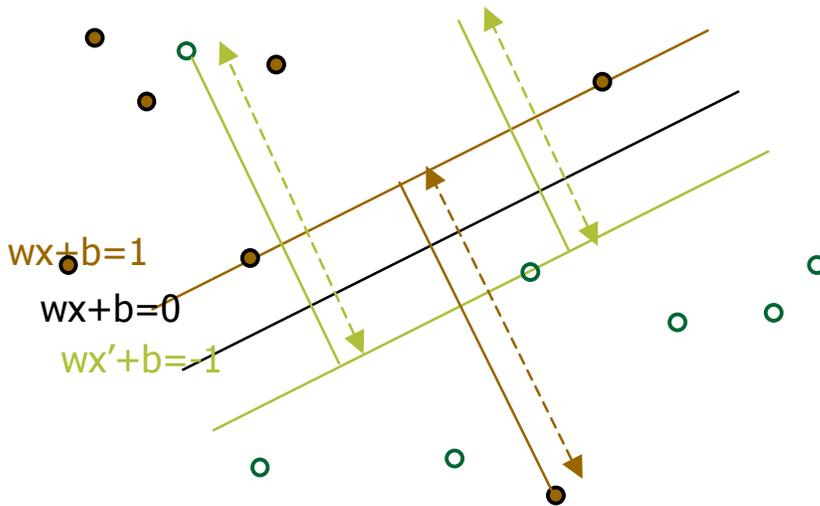


Figure 5: Representation of Hyper planes. [9]

Distance of closest point on hyperplane to origin can be found by maximizing the x as x is on the hyper plane. Similarly for the other side points we have a similar scenario. Thus solving and subtracting the two distances we get the summed distance from the separating hyperplane to nearest points. Maximum Margin = $M = 2 / \|w\|$

Now maximizing the margin is same as minimum [8]. Now we have a quadratic optimization problem and we need to solve for w and b . To solve this we need to optimize the quadratic function with linear constraints. The solution involves constructing a dual problem and where a Lagrange's multiplier α_i is associated. We need to find w and b such that $\Phi(w) = \frac{1}{2} \|w\|^2$ is minimized;

$$\text{And for all } \{(x_i, y_i)\}: y_i (w \cdot x_i + b) \geq 1.$$

Now solving: we get that $w = \sum \alpha_i \cdot x_i$; $b = y_k - w \cdot x_k$ for any x_k such that $\alpha_k \neq 0$

Now the classifying function will have the following form: $f(x) = \sum \alpha_i y_i x_i * x + b$

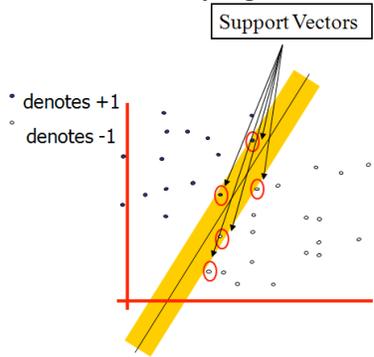


Figure 6: Representation of Support Vectors (Copyright © 2003, Andrew W. Moore)[2]

SVM Representation

In this we present the QP formulation for SVM classification [4][8][12][13]. This is a simple representation only.

SV classification:

$$\min_{f, \xi_i} \|f\|_K^2 + C \sum_{i=1}^l \xi_i \quad y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \text{ for all } i \quad \xi_i \geq 0$$

SVM classification, Dual formulation:

$$\min_{\alpha_i} \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad 0 \leq \alpha_i \leq C, \text{ for all } i; \quad \sum_{i=1}^l \alpha_i y_i = 0$$

Variables ξ_i are called slack variables and they measure the error made at point (\mathbf{x}_i, y_i) . Training SVM becomes quite challenging when the number of training points is large. A number of methods for fast SVM training have been proposed [4][8][13].

Soft Margin Classifier

In real world problem it is not likely to get an exactly separate line dividing the data within the space. And we might have a curved decision boundary. We might have a hyperplane which might exactly separate the data but this may not be desirable if the data has noise in it. It is better for the smooth boundary to ignore few data points than be curved or go in loops, around the outliers. This is handled in a different way; here we hear the term slack variables being introduced. Now we have, $y_i(w'x + b) \geq 1 - S_k$ [4][12]. This allows a point to be a small distance S_k on the wrong side of the hyper plane without violating the constraint. Now we might end up having huge slack variables which allow any line to separate the data, thus in such scenarios we have the Lagrangian variable introduced which penalizes the large slacks.

$$\min L = \frac{1}{2} w'w - \sum \lambda_k (y_k (w'x_k + b) + s_k - 1) + \alpha \sum s_k$$

Where reducing α allows more data to lie on the wrong side of hyper plane and would be treated as outliers which give smoother decision boundary [12].

Kernel Trick

Let's first look at few definitions as what is a kernel and what does feature space mean?

Kernel: If data is linear, a separating hyper plane may be used to divide the data. However it is often the case that the data is far from linear and the datasets are inseparable. To allow for this kernels are used to non-linearly map the input data to a high-dimensional space. The new mapping is then linearly separable [1]. A very simple illustration of this is shown below in figure 7 [9] [11] [20].

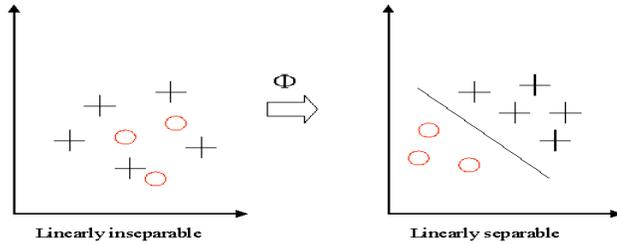


Figure 7: Why use Kernels? [11][9] [20]

This mapping is defined by the Kernel:

$$K(x, y) = \Phi(x) \cdot \Phi(y)$$

Feature Space: Transforming the data into feature space makes it possible to define a similarity measure on the basis of the dot product. If the feature space is chosen suitably, pattern recognition can be easy [1].

$$\langle x_1 \cdot x_2 \rangle \leftarrow K(x_1, x_2) = \langle \Phi(x_1) \cdot \Phi(x_2) \rangle$$

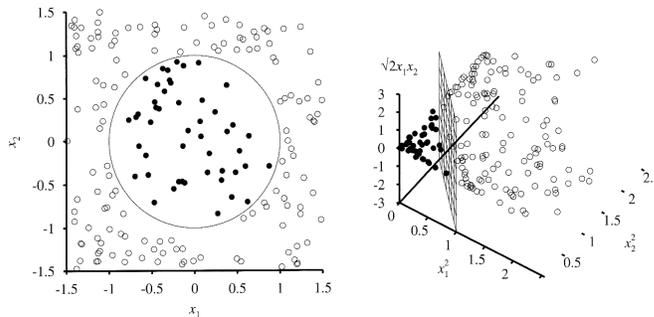


Figure 8: Feature Space Representation [11][9].

Note the legend is not described as they are sample plotting to make understand the concepts involved.

Now getting back to the kernel trick, we see that when w, b is obtained the problem is solved for a simple linear scenario in which data is separated by a hyper plane. The Kernel trick allows SVM's to form nonlinear boundaries. Steps involved in kernel trick are given below [12] [24].

[a] The algorithm is expressed using only the inner products of data sets. This is also called as dual problem.

[b] Original data are passed through non linear maps to form new data with respect to new dimensions by adding a pair wise product of some of the original data dimension to each data vector.

[c] Rather than an inner product on these new, larger vectors, and store in tables and later do a table lookup, we can represent a dot product of the data after doing non linear

mapping on them. This function is the kernel function. More on kernel functions is given below.

Kernel Trick: Dual Problem

First we convert the problem with optimization to the dual form in which we try to eliminate w , and a Lagrangian now is only a function of λ_i . There is a mathematical solution for it but this can be avoided here as this tutorial has instructions to minimize the mathematical equations, I would describe it instead. To solve the problem we should maximize the L_D with respect to λ_i . The dual form simplifies the optimization and we see that the major achievement is the dot product obtained from this [4][8][12].

Kernel Trick: Inner Product summarization

Here we see that we need to represent the dot product of the data vectors used. The dot product of nonlinearly mapped data can be expensive. The kernel trick just picks a suitable function that corresponds to dot product of some nonlinear mapping instead [4][8][12]. Some of the most commonly chosen kernel functions are given below in later part of this tutorial. A particular kernel is only chosen by trial and error on the test set, choosing the right kernel based on the problem or application would enhance SVM's performance.

Kernel Functions

The idea of the kernel function is to enable operations to be performed in the input space rather than the potentially high dimensional feature space. Hence the inner product does not need to be evaluated in the feature space. We want the function to perform mapping of the attributes of the input space to the feature space. The kernel function plays a critical role in SVM and its performance. It is based upon reproducing Kernel Hilbert Spaces [8] [14] [15] [18].

$$K(x, x') = \langle \phi(x), \phi(x') \rangle,$$

If K is a symmetric positive definite function, which satisfies Mercer's Conditions,

$$K(x, x') = \sum_m^{\infty} a_m \phi_m(x) \phi_m(x'), \quad a_m \geq 0,$$

$$\iint K(x, x') g(x) g(x') dx dx' > 0, \quad g \in L_2$$

Then the kernel represents a legitimate inner product in feature space. The training set is not linearly separable in an input space. The training set is linearly separable in the feature space. This is called the "Kernel trick" [8] [12].

The different kernel functions are listed below [8]: More explanation on kernel functions can be found in the book [8]. The below mentioned ones are extracted from there and just for mentioning purposes are listed below.

1] *Polynomial*: A polynomial mapping is a popular method for non-linear modeling. The second kernel is usually preferable as it avoids problems with the hessian becoming Zero.

$$K(x, x') = \langle x, x' \rangle^d.$$

$$K(x, x') = (\langle x, x' \rangle + 1)^d.$$

2] *Gaussian Radial Basis Function*: Radial basis functions most commonly with a Gaussian form

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

3] *Exponential Radial Basis Function*: A radial basis function produces a piecewise linear solution which can be attractive when discontinuities are acceptable.

$$K(x, x') = \exp\left(-\frac{\|x - x'\|}{2\sigma^2}\right)$$

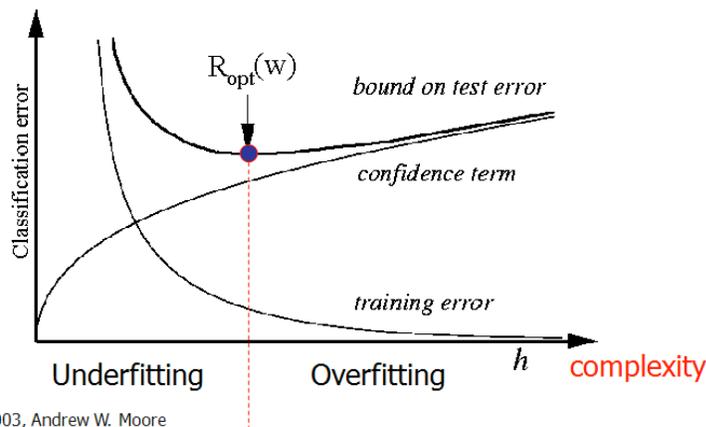
4] *Multi-Layer Perceptron*: The long established MLP, with a single hidden layer, also has a valid kernel representation.

$$K(x, x') = \tanh(\rho\langle x, x' \rangle + \varrho)$$

There are many more including Fourier, splines, B-splines, additive kernels and tensor products [8]. If you want to read more on kernel functions you could read the book [8].

Controlling Complexity in SVM: Trade-offs

SVM is powerful to approximate any training data and generalizes better on given datasets. The complexity in terms of kernel affects the performance on new datasets [8]. SVM supports parameters for controlling the complexity and above all SVM does not tell us how to set these parameters and we should be able to determine these Parameters by Cross-Validation on the given datasets [2] [11]. The diagram given below gives a better illustration.



Copyright © 2001, 2003, Andrew W. Moore

Figure 9: How to control complexity [2] [9]. Note the legend is not described as they are sample plotting to make understand the concepts involved.

SVM for Classification

SVM is a useful technique for data classification. Even though it's considered that Neural Networks are easier to use than this, however, sometimes unsatisfactory results are obtained. A classification task usually involves with training and testing data which consist of some data instances [21]. Each instance in the training set contains one target

values and several attributes. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes [8].

Classification in SVM is an example of Supervised Learning. Known labels help indicate whether the system is performing in a right way or not. This information points to a desired response, validating the accuracy of the system, or be used to help the system learn to act correctly. A step in SVM classification involves identification as which are intimately connected to the known classes. This is called feature selection or feature extraction. Feature selection and SVM classification together have a use even when prediction of unknown samples is not necessary. They can be used to identify key sets which are involved in whatever processes distinguish the classes [8].

SVM for Regression

SVMs can also be applied to regression problems by the introduction of an alternative loss function [8] [17]. The loss function must be modified to include a distance measure. The regression can be linear and non linear. Linear models mainly consist of the following loss functions, e-intensive loss functions, quadratic and Huber loss function. Similarly to classification problems, a non-linear model is usually required to adequately model data. In the same manner as the non-linear SVC approach, a non-linear mapping can be used to map the data into a high dimensional feature space where linear regression is performed. The kernel approach is again employed to address the curse of dimensionality. In the regression method there are considerations based on prior knowledge of the problem and the distribution of the noise. In the absence of such information Huber's robust loss function, has been shown to be a good alternative [8] [16].

Applications of SVM

SVM has been found to be successful when used for pattern classification problems. Applying the Support Vector approach to a particular practical problem involves resolving a number of questions based on the problem definition and the design involved with it. One of the major challenges is that of choosing an appropriate kernel for the given application [4]. There are standard choices such as a Gaussian or polynomial kernel that are the default options, but if these prove ineffective or if the inputs are discrete structures more elaborate kernels will be needed. By implicitly defining a feature space, the kernel provides the description language used by the machine for viewing the data. Once the choice of kernel and optimization criterion has been made the key components of the system are in place [8]. Let's look at some examples.

The task of text categorization is the classification of natural text documents into a fixed number of predefined categories based on their content. Since a document can be assigned to more than one category this is not a multi-class classification problem, but can be viewed as a series of binary classification problems, one for each category. One of the standard representations of text for the purposes of information retrieval provides an ideal feature mapping for constructing a Mercer kernel [25]. Indeed, the kernels somehow incorporate a similarity measure between instances, and it is reasonable to assume that

experts working in the specific application domain have already identified valid similarity measures, particularly in areas such as information retrieval and generative models [25] [27].

Traditional classification approaches perform poorly when working directly because of the high dimensionality of the data, but Support Vector Machines can avoid the pitfalls of very high dimensional representations [12]. A very similar approach to the techniques described for text categorization can also be used for the task of image classification, and as in that case linear hard margin machines are frequently able to generalize well [8]. The first real-world task on which Support Vector Machines were tested was the problem of hand-written character recognition. Furthermore, multi-class SVMs have been tested on these data. It is interesting not only to compare SVMs with other classifiers, but also to compare different SVMs amongst themselves [23]. They turn out to have approximately the same performance, and furthermore to share most of their support vectors, independently of the chosen kernel. The fact that SVM can perform as well as these systems without including any detailed prior knowledge is certainly remarkable [25].

Strength and Weakness of SVM:

The major strengths of SVM are the training is relatively easy. No local optimal, unlike in neural networks. It scales relatively well to high dimensional data and the trade-off between classifier complexity and error can be controlled explicitly. The weakness includes the need for a good kernel function [2] [4] [8] [12] [24].

Conclusion

The tutorial presents an overview on SVM in parallel with a summary of the papers collected from the World Wide Web. Some of the important conclusions of this tutorial are summarized as follows. SVM are based on statistical learning theory. They can be used for learning to predict future data [25]. SVM are trained by solving a constrained quadratic optimization problem. SVM, implements mapping of inputs onto a high dimensional space using a set of nonlinear basis functions. SVM can be used to learn a variety of representations, such as neural nets, splines, polynomial estimators, etc, but there is a unique optimal solution for each choice of the SVM parameters [4]. This is different in other learning machines, such as standard Neural Networks trained using back propagation [26]. In short the development of SVM is an entirely different from normal algorithms used for learning and SVM provides a new insight into this learning. The four most major features of SVM are duality, kernels, convexity and sparseness [24].

Support Vector Machines acts as one of the best approach to data modeling. They combine generalization control as a technique to control dimensionality. The kernel mapping provides a common base for most of the commonly employed model architectures, enabling comparisons to be performed [8]. In classification problems generalization control is obtained by maximizing the margin, which corresponds to minimization of the weight vector in a canonical framework. The solution is obtained as a

set of support vectors that can be sparse. The minimization of the weight vector can be used as a criterion in regression problems, with a modified loss function. Future directions include: A technique for choosing the kernel function and additional capacity control; Development of kernels with invariance. Finally, new directions are mentioned in new SVM-related learning formulations recently proposed by Vapnik [19].

References:

- [1] Wikipedia Online. [Http://en.wikipedia.org/wiki](http://en.wikipedia.org/wiki)
- [2] Tutorial slides by Andrew Moore. [Http://www.cs.cmu.edu/~awm](http://www.cs.cmu.edu/~awm)
- [3] V. Vapnik. The Nature of Statistical Learning Theory. Springer, N.Y., 1995. ISBN 0-387-94559-8.
- [4] Burges C., "A tutorial on support vector machines for pattern recognition", In "Data Mining and Knowledge Discovery". Kluwer Academic Publishers, Boston, 1998, (Volume 2).
- [5] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, M. Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems 9, pages 281– 287, Cambridge, MA, 1997. MIT Press.
- [6] Theodoros Evgeniou and Massimiliano Pontil, Statistical Learning Theory: a Primer 1998.
- [7] Olivier Bousquet, Stephane Boucheron, and Gabor Lugosi, "Introduction to Statistical Learning Theory".
- [8] Nello Cristianini and John Shawe-Taylor, "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods", Cambridge University Press, 2000.
- [9] Image found on the web search for learning and generalization in svm following links given in the book above.
- [10] David M Skapura, Building Neural Networks, ACM press, 1996.
- [11] Tom Mitchell, Machine Learning, McGraw-Hill Computer science series, 1997.
- [12] J.P.Lewis, Tutorial on SVM, CGIT Lab, USC, 2004.
- [13] Vapnik V., "Statistical Learning Theory", Wiley, New York, 1998.
- [14] M. A. Aizerman, E. M. Braverman, and L. I. Rozono'er. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control, 25:821–837, 1964.
- [15] N. Aronszajn. Theory of reproducing kernels. Trans. Amer. Math. Soc., 686:337–404, 1950.
- [16] C. Cortes and V. Vapnik. Support vector networks. Machine Learning, 20:273 – 297, 1995
- [17] A. J. Smola. Regression estimation with support vector learning machines. Master's thesis, Technische Universit'at M'unchen, 1996.
- [18] N. Heckman. The theory and application of penalized least squares methods or reproducing kernel hilbert spaces made easy, 1997.
- [19] Vapnik, V., Estimation of Dependencies Based on Empirical Data. Empirical Inference Science: Afterword of 2006, Springer, 2006
- [20] http://www.enm.bris.ac.uk/teaching/projects/2004_05/dm1654/kernel.htm
- [21] Duda R. and Hart P., "Pattern Classification and Scene Analysis", Wiley, New York 1973.

- [22] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Giles, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, pages 276 – 285, New York, 1997. IEEE.
- [23] M. O. Stitson and J. A. E. Weston. Implementational issues of support vector machines. Technical Report CSD-TR-96-18, Computational Intelligence Group, Royal Holloway, University of London, 1996.
- [24] Burges B.~Scholkopf, editor, “Advances in Kernel Methods--Support Vector Learning”. MIT press, 1998.
- [25] Osuna E., Freund R., and Girosi F., “Support Vector Machines: Training and Applications”, A.I. Memo No. 1602, Artificial Intelligence Laboratory, MIT, 1997.
- [26] Trafalis T., "Primal-dual optimization methods in neural networks and support vector machines training", ACAI99.
- [27] Veropoulos K., Cristianini N., and Campbell C., "The Application of Support Vector Machines to Medical Decision Support: A Case Study", ACAI99