Support Vector Machines Illuminated

David R. Musicant
Carleton College, USA

INTRODUCTION

In recent years, massive quantities of business and research data have been collected and stored, partly due to the plummeting cost of data storage. Much interest has therefore arisen in how to mine this data to provide useful information. Data mining as a discipline shares much in common with machine learning and statistics, as all of these endeavors aim to make predictions about data as well as to better understand the patterns that can be found in a particular dataset. The support vector machine (SVM) is a current machine learning technique that performs quite well in solving common data mining problems.

BACKGROUND

The most common use of SVMs is in solving the classification problem, which we focus on in the following example.

The dataset in Table 1 contains demographic information for four people. These people were surveyed to determine whether or not they purchased software on a regular basis. The dataset in Table 2 contains demographic information for two more people who may or may not be good targets for software advertising. We wish to determine which of the people in Table 2 purchase software on a regular basis.

This classification problem is considered an example of *supervised learning*. In supervised learning, we start off with a *training set* (Table 1) of examples. We use this training set to find a rule to be used in making predictions on future data. The quality of a rule is typically determined through the use of a *test set* (Table 2). The test set is another set of data with the same attributes as the training set, but which is *held out* from the training process. The values of the *output attributes*, which are indicated by question marks in Table 2, are hidden and "pretended" not to be known. After training is complete, the rule is used to predict values for the output attribute for each point in the test set. These output predictions are then compared with the (now revealed) known values for these attributes, and the difference between them is measured. Success is typically measured by the fraction of points which are classified correctly. This measurement provides an estimate as to how well the algorithm will perform on data where the value of the output attribute is truly unknown.

One might ask: why bother with a test set? Why not measure the success of a training algorithm by comparing predicted output with actual output on the training set alone? The use of a test set is particularly important because it helps to estimate the true error of a classifier. Specifically, a test set determines if *overfitting* has occurred. A learning algorithm may learn the training data "too well," i.e. it may perform very well on the training data, but very poorly on unseen testing data. For example, consider the following classification rule as a solution to the above posed classification problem:

- **Overfitted Solution:** If our test data is actually present in Table 1, look it up in the table to find the class for which the point belongs. If the point is not in Table 1, classify it in the "No" category.

This solution is clearly problematic – it will yield 100% accuracy on the training set, but should do poorly on the test set since it assumes that all other points are automatically "No". An important aspect of developing supervised learning algorithms is ensuring that overfitting does not occur.

In practice, training and test sets may be available a priori. Most of the time, however, only a single set of data is available. A random subset of the data is therefore held out from the training process in order to be used as a test

Table 1. Classification example training set

Age	Income	Years of Education	Software Purchaser?
30	$56,000 / yr	16	Yes
50	$60,000 / yr	12	No
16	$2,000 / yr	11	Yes
35	$30,000 / yr	12	No

Table 2. Classification example test set

Age	Income	Years of Education	Software Purchaser?
40	$48,000 / yr	17	?
29	$60,000 / yr	18	?

set. This can introduce widely varying success rates, depending on which data points are chosen. This is traditionally dealt with by using *cross-validation*. The available data is randomly broken up into k disjoint groups of approximately equal size. The training process is run k times, each time holding out a different group to use as a test set and using all remaining points as the training set. The success of the algorithm is then measured by an average of the success over all k test sets. Usually we take k=10, yielding the process referred to as tenfold cross-validation.

A plethora of methodologies can be found for solving classification and regression problems. These include the backpropagation algorithm for artificial neural networks, decision tree construction algorithms, spline methods for classification, probabilistic graphical dependency models, least squares techniques for general linear regression, and algorithms for robust regression. SVMs are another approach, rooted in mathematical programming.

Euler observed that "Nothing happens in the universe that does not have...a maximum or a minimum". Optimization techniques are used in a wide variety of fields, typically in order to maximize profits or minimize expenses under certain constraints. For example, airlines use optimization techniques in finding the best ways to route airplanes. The use of mathematical models in solving optimization problems is referred to as *mathematical programming*. The use of the word "programming" is now somewhat antiquated, and is used to mean "scheduling." Mathematical programming techniques have become attractive for solving machine learning problems, as they perform well while also providing a sound theoretical basis that some other popular techniques do not. Additionally, they offer some novel approaches in addressing the problem of overfitting (Herbrich, 2002; Vapnik, 1998).

It is not surprising that ideas from mathematical programming should find application in machine learning. After all, one can summarize classification and regression problems as "Find a rule that *minimizes* the errors made in predicting an output." One formalization of these ideas as an optimization problem is referred to as the SVM (Herbrich, 2002; Vapnik, 1998). It should be noted that the word "machine" is used here figuratively, referring to an algorithm that solves classification problems.

MAIN THRUST

Suppose that we wish to classify points that can be in one of two classes, which we will label as 1 and -1. In the previous examples, class 1 could correspond to those people that are software purchasers, and class -1 could correspond to those that are not. We will assume that the training data consists of m examples, each of which has n features. In Table 1, for example, $m = 4$ and $n = 3$. The "software purchaser" column is considered to be the classification for each row, and not one of the features. Each point can therefore be represented as a vector $\mathbf{x}_i$ of size n, where i ranges from 1 to m. To indicate class membership, we denote by y_i the class for point i, where $y_i = +1$ for points in class 1 and $y_i = -1$ for points in class -1. For example, if we consider class "Yes" as class 1 and "No" as class -1 we would represent the training set in our classification example as:

$$\begin{aligned} \mathbf{x}_1 &= [30 \quad 56000 \quad 16], & d_1 &= 1 \\ \mathbf{x}_2 &= [50 \quad 60000 \quad 12], & d_2 &= -1 \\ \mathbf{x}_3 &= [16 \quad 2000 \quad 11], & d_3 &= 1 \\ \mathbf{x}_4 &= [35 \quad 30000 \quad 12], & d_4 &= -1 \end{aligned} \tag{1}$$

The goal is to find a *hyperplane* that will best separate the points into the two classes.

To solve this problem, let us visualize a simple example in two dimensions which is completely linearly separable, i.e. a straight line can perfectly separate the two classes.

Figure 1 shows a simple linearly separable classification problem, where the *separating hyperplane*, or *separating surface*

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2}$$

separates the points in class 1 from the points in class- 1. The goal then becomes one of finding a vector $\mathbf{w}$ and scalar b such that the points in each class are correctly classified. In other words, we want to find $\mathbf{w}$ and b such that the following inequalities are satisfied:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x} + b &> 0, \quad \text{for all i such that } y_i = 1 \\ \mathbf{w} \cdot \mathbf{x} + b &< 0, \quad \text{for all i such that } y_i = -1 \end{aligned} \tag{3}$$

In practice, however, finding a solution to this problem is considerably easier if we express these as non-strict inequalities. To do this, we define $\delta > 0$ as:

Figure 1. Linearly separable classification problem