Customer Purchase Behavior in the Airline Industry

White Paper

April, 2012
www.mu-sigma.com

CLASSIFICATION OF CUSTOMER PURCHASE BEHAVIOR IN THE AIRLINE INDUSTRY USING SUPPORT VECTOR MACHINES

Pravin V,

Innovation and Development Team,

Mu Sigma Business Solutions Pvt. Ltd, Bangalore.

April 2012

SYNOPSIS

The objective of this paper is to give an understanding of the various aspects of *Support Vector Machines (SVM)*, which is used widely for classification and regression problems. SVM classification process follows a supervised learning model, in which the available data is used to train the model based on similarity measures. With the trained model, the unseen or new data is then classified into classes. The non-linear input space is handled using various type of *kernel functions*, which is chosen based on the application.

1. INTRODUCTION

This section begins with familiarizing the tools and technologies used. It gives a brief overview of the requirements and the system configuration.

1.1 TECHNOLOGIES USED

1.1.1 R

R is a language and environment for statistical computing and graphics. It is both software and a language considered a dialect of the S language created by the AT&T Bell Laboratories. R is freely distributed under the terms of the General Public License (GNU); its development and distribution are carried out by several statisticians known as the R Development Core Team. The files needed to install R, either from the sources or from the pre-compiled binaries, are distributed through the Internet site of the Comprehensive R Archive Network (CRAN) where the instructions for the installation are also available. R has many functions for statistical analyses and graphics. The results from a statistical analysis are displayed on the screen; some intermediate results (P-values, regression coefficients, residuals, etc) can be saved, written in a file, or used in subsequent analyses. Some of the packages of R used in this project are *lpSolveAPI*, *datasets*, *stats*, *utils*, and *graphics*.

1.1.2 RStudio

RStudio is an integrated development environment (IDE) for R that works with the standard version of R available from CRAN. Like R, RStudio is available under a free software license. The goal of RStudio developers was to develop a powerful tool that supports the practices and techniques required for creating trustworthy, high quality analysis. At the same time, RStudio is as straightforward and intuitive as possible to provide a user friendly environment for R users.

1.2 REQUIREMENTS SPECIFICATION

The requirement given was to classify the customer purchase behavior in the airline industry dataset using a Support Vector Machine (SVM). An in-depth study and understanding of SVMs is detailed and following that the classification task is done for a supplied airline industry dataset. The basic ideas of Support Vector Classification and a detailed study about the Support Vector Machines, Kernel Methods, underlying concepts and the related topics are discussed in the following sections. Following that, the implementation and the related details are described.

2. SUPPORT VECTOR MACHINES

This section describes the background, the objective, and a brief explanation about Support Vector Machines (SVM) and its role in the project.

2.1 INTRODUCTION

The SVM is a concept in statistics and computer science for a set of related supervised learning methods that analyze data and recognize patterns. This is used for both classification and regression analysis. The standard SVM takes a set of input data and predicts, for each given new input, which of the two possible classes forms the input, making the SVM a non-probabilistic binary linear classifier. An SVM 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. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

2.2 BACKGROUND

With their introduction in 1995, SVM marked the beginning of a new era in the learning from examples paradigm. Rooted in the Statistical Learning Theory developed by Vladimir Vapnik at AT&T, SVMs quickly gained attention from the pattern recognition community due to a number of theoretical and computational merits.

2.3 SVMs FOR CLASSIFICATION

Classifying data is a common task in machine learning. The SVM is a supervised learning method that generates input-output mapping functions from a set of labeled training data. In the case of SVMs, a data point is viewed as a *p*-dimensional vector (a list of *p* numbers), and we need to know whether we can separate such points with a (*p - 1*) - dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So a hyperplane is chosen such that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum margin classifier; or equivalently, the perceptron of optimal stability. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Commonly in machine learning, a generalized model must be selected from a finite data set, with the consequent problem of overfitting – the model being tailored to the particularities of the training set and underfitting - generalizing poorly to new data. The S*tructural risk minimization* (SRM) principle addresses this problem by balancing the model's complexity against its success at fitting the training data. It describes a general model of capacity control

and provides a trade-off between hypothesis space complexity (bound on test error) and the quality of fitting the training data (empirical error).

2.4 LINEAR SVM

One of the fundamental problems of learning theory is, assuming there are two classes of objects and then upon facing a new object it has to be assigned to one of the two assumed classes. The training data, *D* a set of *n* records, will be of the form,

$$D = \{(x_i, y_i) | \, x_i \in \mathbb{R}^p, \, y_i \in \{-1, 1\}\}_{i=1}^{n} \quad (1)$$

Here, each x_i is a *p*- dimensional real vector. There are only two classes of data. They are labeled as +1 and -1 respectively. The new data has to be classified only among these two classes. This type of classification is a particularly simple situation, known as the binary classification. An example is illustrated in figure 2.1.

2.4.1 Similarity Measure

The choice of similarity measure for the inputs is a deep question that lies at the core of the field of machine learning. Consider a similarity measure of the form $k: D \times D \rightarrow \mathbb{R}$. That is,$(x, x') \rightarrow k(x, x')$, a function that, given two data points, returns a real number characterizing their similarity. Unless stated otherwise it is assumed that k is symmetric, that is, $k(x, x') = k(x', x), \forall x \in D$. A simple type of similarity measure that is of particular mathematical appeal is a *dot product*.

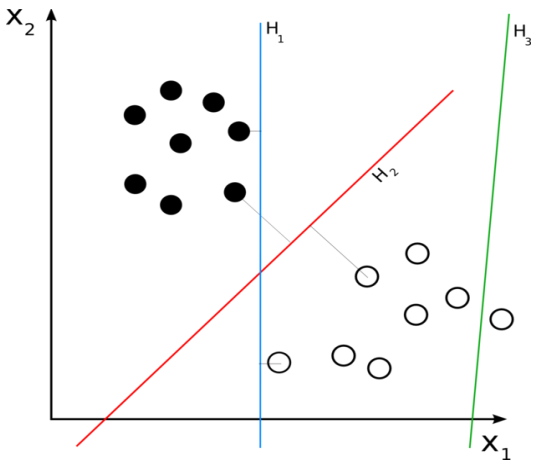

Figure 2.1 – Example of binary classification with hyperplanes

Characterizing the similarity of the outputs (y_i) as $\{\pm 1\}$ is easy: in binary classification, only two situations can occur: two labels can either be identical or different. The objective is to find a hyperplane that divides the points having the class label (y_i) as +1 from those having the class label (y_i) as -1. Now, it is assumed that a dot product space $\mathcal{H}$, and a set of data vectors, $x_1, \ldots, x_n \in \mathcal{H}$ which is linearly separable, are given. Any hyperplane in the space $\mathcal{H}$ can be written as a set of points x satisfying, $< w \cdot x > -b = 0$ where " $\cdot$ " denotes the dot product, $w \in \mathcal{H}$ the normal vector to the hyperplane and $b \in \mathbb{R}$. The parameter $\frac{b}{\|w\|}$, denotes the offset of the hyperplane from the origin along the normal vector w. The decision function for this classification will be,

$$f(w, b) = sign\{(w \cdot x) - b\} \quad (2)$$

2.4.2 Linear SVM Formulation

In linearly separable data sets, there are many decision boundaries. In order to have a good generalization for a new decision, a good decision boundary is needed. The decision boundaries of two sets should be as far away from two classes as possible in order to make an effective generalization. The shortest distance between two boundaries is called a margin. The optimal hyperplane would be the one that represents the largest separation, or margin, between the two classes. So such a hyperplane is chosen so that the distance from it to the nearest data point on each side is maximized, hence the name *maximum-margin hyperplane*. Now the optimization problem to be solved is derived for computing the optimal hyperplane. So, w and b are chosen so as to maximize the margin or the distance between the parallel hyperplanes that are as far apart as possible while still separating the data.

By using geometry, the distance between these two hyperplanes is found to be, $\frac{2}{\|w\|}$

so, the variable to minimize is $\| w \|$. Also, to prevent data points from falling into the margin, the following constraints are added, for each x_i of the first class, $(w \cdot x_i - b) \geq 1$ and for each x_i of the second class, $(w \cdot x_i - b) \leq -1$. An example is illustrated in figure 2.2.

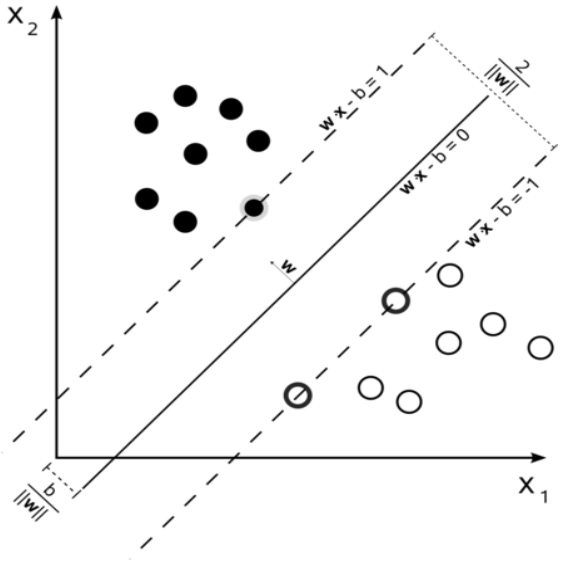

Figure 2.2 – Example of Linear SVM with the Optimal Hyperplane

The addition of the class label to the above constraints allows distinguishing two orientations of a hyperplane. So the above two constraints can be combined and stated as follows:

For each x_i, $\quad y_i(w \cdot x_i - b) \geq 1, for\ all\ 1 \leq i \leq n$ (3)

Now, that the objective and the constraints of the problem are clear, the optimization problem can be put together as follows:

$$f(x_i) = \min \text{ (in } w \text{ and } b), \ \| w \|$$

Subject to, $\quad y_i(w \cdot x_i - b) \geq 1, \ \forall\ 1 \leq i \leq n$ (4)

2.4.3 Primal Form

The optimization problem presented in equation (4) is difficult to solve because it depends on $\| w \|$, the norm factor, which involves a square root. Fortunately it is possible to

alter the equation by substituting $\| w \|$ with, $\frac{1}{2} \| w \|^2$ (the factor of 1/2 being used for mathematical convenience) without changing the solution (the minimum of the original and the modified equation have the same w and b). Now, this is a quadratic programming optimization problem which is stated as follows:

$$f(x_i) = \min \text{ (in } w \text{ and } b), \frac{1}{2} \| w \|^2$$

$$\text{Subject to,} \quad y_i(w \cdot x_i - b) \geq 1, \ \forall \ 1 \leq i \leq n \tag{5}$$

2.4.4 Lagrangian Multipliers

The above described optimization problem is called the constrained optimization problem. Generally problems of this kind are dealt with by introducing Lagrangian multipliers $\alpha_i \geq 0$. By introducing those Lagrangian multipliers, the above described constrained problem can be expressed as:

$$L(w, b, \alpha) = \underset{w,b}{Min} \ \underset{\alpha \geq 0}{Max} \left\{ \frac{1}{2} \| w \|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w \cdot x_i - b) - 1] \right\} \tag{6}$$

Now, the Lagrangian *L* must be maximized with respect to α_i, and minimized with respect to w and b. If a constraint (5) is violated, then, $y_i(w \cdot x_i - b) < 0$ in which case *L* can be increased by increasing the corresponding α_i. At the same time, w and b will have to change such that *L* decreases. In order to prevent $\alpha_i [y_i(w \cdot x_i - b) - 1]$ from becoming an arbitrarily large negative number, the change in w and b will ensure that, the constraint will eventually be satisfied, provided the problem is separable. Similarly, one can understand that for all constraints that are not precisely met as equalities (that is, for which, $y_i(w \cdot x_i - b) > 0$), the corresponding α_i must be 0: this is the value of α_i that maximizes *L*. The latter is the statement of the Karush-Kuhn-Tucker (KKT) conditions of optimization theory. The statement that at the saddle point, the derivatives of *L* with respect to the primal variables must vanish,

$\frac{\partial}{\partial b} L(w, b, \alpha) = 0$, $\frac{\partial}{\partial w} L(w, b, \alpha) = 0$, this leads to,

$$\sum_{i=1}^{n} y_i \alpha_i = 0, \text{ and,} \quad (9)$$

$$w = \sum_{i=1}^{n} y_i \alpha_i x_i. \quad (10)$$

The solution vector thus has an expansion in terms of training examples. Note that although the solution w is unique, the coefficients α_i need not be. According to the KKT theorem, only those Lagrange multipliers α_i that are non-zero at the saddle point correspond to constraint (5) which is precisely met. So finally it simplifies to $\alpha_i[\, y_i(\, w \cdot\, x_i - \, b\,) - 1] = 0.$

2.4.5 Support Vectors

The data points for which $\alpha_i > 0$ are called *Support vectors.* According to the above description, they lie only on the margin. All remaining examples in the training set are irrelevant: Their constraints (5) are satisfied automatically, and they do not appear in the expansion (8), since their multipliers satisfy $\alpha_i = 0.$ This leads directly to an upper bound on the generalization ability of optimal margin hyperplanes.

2.4.6 Dual Form

Now the focus is on the optimization problem to be solved. Substituting the conditions for the extreme values, (7) and (8), into the Lagrangian (6), the dual form of the optimization problem is arrived at:

$$W(\alpha) = \underset{\alpha}{\text{Max,}} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i \cdot x_j \rangle \right\}$$

$$\text{Subject to,} \quad \sum_{i=1}^{n} y_i \alpha_i = 0 \;\; and \;\; \alpha_i \geq 0, \; \forall\, 1 \leq i \leq n \quad (9)$$

Now, the final decision function for this problem is obtained on substituting the expansion (8) in (9), we obtain an expression which can be evaluated in terms of dot products, taken between the data points to be classified and the Support Vectors,

$$f(x) = sign\left\{\sum_{i=1}^{n} y_i \alpha_i \langle x \cdot x_j \rangle - b\right\} \quad (10)$$

Thus, expressing the classification rule in its unconstrained dual form reveals that the maximum margin hyperplane and therefore the classification task is only a function of the *support vectors*, the training data that lie on the margin.

2.5 NON-LINEAR SVM

Thus far, it is shown why it is that a large margin hyperplane is good from a statistical point of view, and it has been demonstrated how to compute it using optimization techniques. Although these two points have worked out nicely, there is still a major drawback to the approach: Everything that has been done so far is linear in the data. To allow for much more general decision surfaces, kernels are used to non-linearly transform the input data $x_1, \ldots, x_m \in \mathcal{X}$ into a high-dimensional feature space; using a map Φ: $x_i \rightarrow \mathbb{x}_i$; then doing a linear separation there. To justify this procedure, Cover's Theorem is sometimes alluded to.

2.6 COVER'S THEOREM

Cover's Theorem is a statement in computational learning theory and is one of the primary theoretical motivations for the use of non-linear kernel methods in machine learning applications. The theorem states that given a set of training data that is not linearly separable, one can with high probability transform it into a training set that is linearly separable by projecting it into a higher dimensional space via some non-linear transformation. This theorem characterizes the number of possible linear separations of *m* points in general position in an *N*-dimensional space. If $m \leq N + 1$, then $2m$ separations are possible but if $m > N + 1$, then Cover's Theorem states that the number of linear separations equals,

$$2 \sum_{i=0}^{N} \binom{m-1}{i} \quad (11)$$

The more *N* is increased the more terms there are in the sum, and thus the larger is the resulting number. This theorem formalizes the intuition that the number of separations increases with the dimensionality. It requires, however, that the points are in general position — therefore, it does not strictly make a statement about the separability of a given dataset in a given feature space. For example, the feature map might be such that all points lie on a rather restrictive lower-dimensional manifold, which prevents finding points in general position.

2.7 KERNEL METHODS

Kernel methods (KMs) are a class of algorithms for pattern analysis, whose best known element is the SVM. The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components, correlations, classifications) in general types of data (such as sequences, text documents, sets of points, vectors, images, etc.). KMs approach the problem by mapping the data into a high dimensional feature space, where each coordinate corresponds to one feature of the data items, transforming the data into a set of points in a Euclidean space. In this space, a variety of methods can be used to find relations in the data. Since the mapping can be quite general (not necessarily linear, for example), the relations found in this way are accordingly very general. This approach is called the *kernel trick*. Using a kernel typically amounts to using a larger function class, thus increasing the capacity of the learning machine, and rendering problems separable that are not linearly separable to start with.

KMs owe their name to the use of kernel functions, which enable them to operate in the feature space without ever computing the coordinates of data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates. An example is illustrated in figure 2.3. In Figure 2.3, by mapping the input data nonlinearly (via Φ) into a higher-dimensional feature space (here: $\mathbb{R}^3$), and constructing a

separating hyperplane there, an SVM corresponds to a nonlinear decision surface in input space (here: $\mathbb{R}^2$). Here, x_1, x_2 is used to denote the entries of the input vectors, and w_1, w_2, w_3 denote entries of the hyperplane normal vector in $\mathbb{R}^3$.

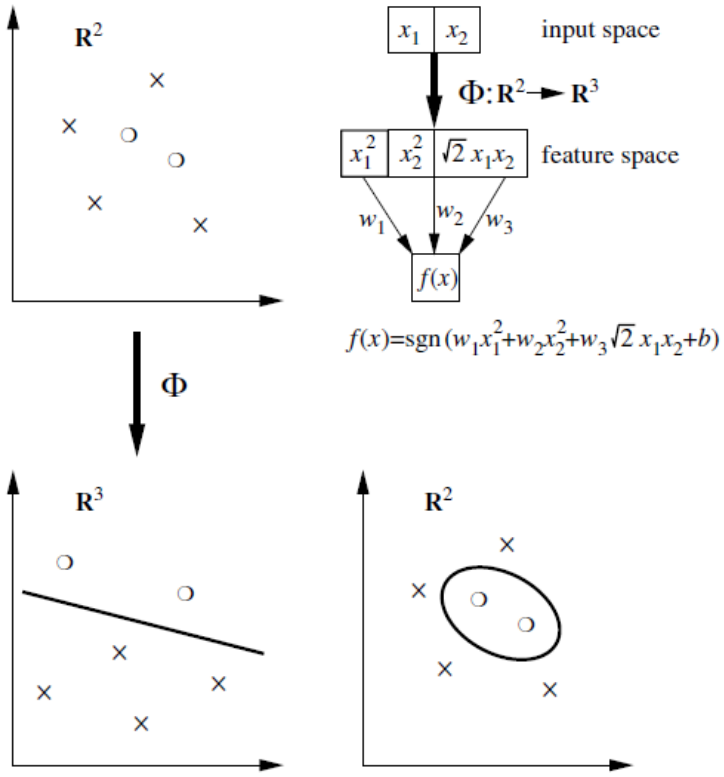

Figure 2.3 – Kernel Method for Non Linear Data

2.8 KERNELIZING THE OPTIMAL HYPERPLANE

On the practical level, the modifications necessary to perform the algorithm in a high-dimensional feature space are minor. In the above sections, no assumptions were made on the dimensionality of $\mathcal{H}$, the space in which we assumed our patterns belong to. It is required that only $\mathcal{H}$ is being equipped with a dot product. The patterns $\mathbb{x}_i$ that have been talked about previously thus need not coincide with the input patterns. They can equally be the results of mapping the original input patterns x_i into a high-dimensional feature space.

Consequently, an instance is taken such that wherever $\mathbb{x}_i$ is written, it actually means $\Phi(x)$. Maximizing the target function (9) and evaluating the decision function (10),

then requires the computation of dot products $\langle \Phi(x), \Phi(x_i) \rangle$ in a high-dimensional space. These expensive calculations are reduced significantly by using a positive definite kernel k, such that,

$$\langle \Phi(x), \Phi(x_i) \rangle = k(x, x_i), \text{ leading to decision functions of the form,} \quad (12)$$

$$f(x) = sign \{ \sum_{i=1}^{n} y_i \alpha_i \, k(x, x_i) - b \} \quad (13)$$

At this point, a small aside regarding terminology is in order. As explained before the input domain need not be a vector space. Therefore, the Support Vectors in (13) (i.e., those x_i with $\alpha_i > 0$) are not necessarily vectors. One could choose to be on the safe side, and only refer to the corresponding $\Phi(x_i)$ as Support Vectors. Consequently, everything that has been said about the linear case also applies to non-linear cases, which are obtained using a suitable kernel *k*, instead of the Euclidean dot product (Figure 2.3).

In certain cases, there exists a way of computing dot products in these high-dimensional feature spaces without explicitly mapping into the spaces, by means of kernels nonlinear in the input space *N* . Thus, if the subsequent processing can be carried out using dot products exclusively, we are able to deal with the high dimension. In order to compute dot products of the form $\langle \Phi(x), \Phi(x') \rangle$, kernel representations are employed of the form,

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \quad (14)$$

which allows to compute the value of the dot product in $\mathcal{H}$ without having to explicitly compute the map Φ. An example is illustrated in figure 2.4.

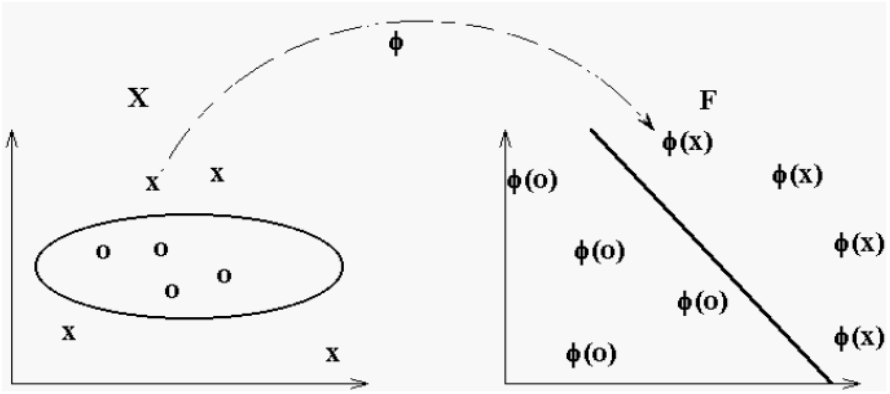

Figure 2.4 – Kernel Trick

2.9 QUADRATIC PROGRAM

The resulting algorithm is formally similar to that of the linear SVM, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space. The quadratic program that is to be solved is described as,

$$W(\alpha) = \underset{\alpha}{\text{Max,}} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \, k(x_i, x_j) \alpha_i \alpha_j \right\}$$

$$subject\ to, \quad \sum_{i=1}^{n} y_i \alpha_i = 0,$$

$$0 \leq \alpha_i \leq C\,;\ \forall i = 1, \ldots, n. \tag{15}$$

If k is positive definite, $Q_{ij} = (y_i y_j \, k(x_i, x_j))_{ij}$ is a positive definite matrix, which provides us with a convex problem that can be solved efficiently. A larger class of kernels can be used without destroying the convexity of the quadratic program. This is due to the fact that the constraint (15) excludes certain parts of the space of multipliers α_i. As a result, the kernel is required to be positive definite on the remaining points. This is precisely guaranteed if k is required to be as conditionally positive definite. In this case it is,

$\alpha^T Q\ \alpha\ \geq 0$ for all coefficient vectors α, satisfying (15). To compute the threshold b, the KKT conditions are taken into account and so $\alpha_i > 0$, implies,

$$\sum_{i=1}^{n} y_i\ k(x_i, x_j)\alpha_i + b = y_j \text{ , and thus,}$$

$$b = y_j - \sum_{i=1}^{n} y_i\ k(x_i, x_j)\alpha_i; \text{ For all the support vectors.} \qquad (16)$$

The Support Vectors are the patterns closest to the decision boundary — not only in the feature space- (where by construction, the Support Vectors are the patterns closest to the separating hyperplane), but also in the input space. This feature differentiates SVMs from other types of classifiers.

2.10 SVM TRAINING

Training a SVM requires the solution of a very large quadratic programming (QP) optimization problem. Due to its immense size, the QP problem that arises from SVMs cannot be easily solved via standard QP techniques. The quadratic form involves a matrix that has a number of elements equal to the square of the number of training examples. The main advantage in developing algorithms specific for SVM is that SVM solutions are sparse; they do not have many support vectors (where $\alpha_i \neq 0$).

Sequential Minimal Optimization (SMO) is a simple algorithm that can quickly solve the SVM QP problem without any extra matrix storage and without using numerical QP optimization steps. SMO decomposes the overall QP problem into QP sub-problems. Unlike other methods, SMO chooses to solve the smallest possible optimization problem at every step. For the standard SVM QP problem, the smallest possible optimization problem involves two Lagrange multipliers, because the Lagrange multipliers must obey a linear equality constraint. At every step, SMO chooses two Lagrange multipliers to jointly optimize, finds the optimal values for these multipliers, and updates the SVM to reflect the new optimal values. The advantage of SMO lies in the fact that solving for two Lagrange multipliers can be done analytically. Thus, numerical QP optimization is avoided entirely.

Even though more optimization sub-problems are solved in the course of the algorithm, each sub-problem is so fast that the overall QP problem is solved quickly. In addition, SMO requires no extra matrix storage at all. Thus, very large SVM training problems can fit inside of the memory of an ordinary personal computer or workstation. There are two components to SMO: an analytic method for solving for the two Lagrange multipliers, and a heuristic for choosing which multipliers to optimize. Because of the linear equality constraint involving the Lagrange multipliers α_i, the smallest possible problem involves two such multipliers. Then, for any two multipliers α_1 and α_2, the constraints are reduced to the following and then can be solved analytically.

$0 \leq \alpha_1$ and $\alpha_2 \leq C$, where C is the regularization parameter,

$$y_1\alpha_1 + y_2\alpha_2 = k \tag{17}$$

The brief algorithm of the SMO proceeds as follows:

1. Find a Lagrange multiplier α_1 that violates the KKT conditions for the optimization problem.
2. Pick a second multiplier α_2 and optimize the pair (α_1, α_2).
3. Repeat steps 1 and 2 until convergence.

When all the Lagrange multipliers satisfy the KKT conditions (within a user-defined tolerance), the problem has been solved. Although this algorithm is guaranteed to converge, heuristics are used to choose the pair of multipliers so as to accelerate the rate of convergence. SMO is an improved training algorithm for SVMs. Like other SVM training algorithms, SMO breaks down a large QP problem into a series of smaller QP problems. Unlike other algorithms, SMO utilizes the smallest possible QP problems, which are solved quickly and analytically, generally improving its scaling and computation time significantly.

3. IMPLEMENTATION

This section will focus on the implementation of the classification using the techniques discussed in the previous sections.

3.1 INTRODUCTION

SVMs have very strong mathematical foundation and have proven to be useful empirically. There are various mathematical formulations of the SVM optimization problem, some of which have a quadratic objective function while other a linear objective function. There are various advantages in using a linear programming formulation; including speed and sparse nature. There are various ready-to-use implementations of SVM; like LIBSVM, LIBLINEAR, SVMlight and many more. Moreover it is possible to implement an SVM with the help of an optimization package like CVX, a MATLAB based optimization software that allows writing the optimization problem in a modeling language nicely and cleanly.

There are also freely available optimization solvers. There are a couple of solvers available in the open source domain like the GNU linear Programming Kit (GLPK) and LpSolve. Apart from being free, these solvers have bindings to programming languages like C, Python and R. As these are linear solvers, the objective function has to be linear, allowing only some special cases of SVM (say LPSVM) to be solved. The trick here is to take the equations from an LPSVM formulation and express it in a way that is familiar to the linear solver. Being general purpose solvers both GLPK and LpSolve (and most of the other solvers) understand the linear program in its standard form.

3.2 LINEAR SVM

The dataset to be classified using linear SVM is a purchase dataset of an airline industry. The objective is to classify the purchase behavior of the customers from the historical purchase data which is provided for the same. As SVM is one of the supervised learning methods, a model has to be created from a sample of the original data called as the

training data. Then the unseen or new data that is to be classified, known as the testing set is given as input to the model created. The output obtained from this testing phase is then validated with the known values. Accuracy measure of the model is calculated. Some details about the dataset that is used for classification is given in table 3.1.

Field Value	**Size**
Airline Dataset	1479
Features	18 (5 – Categorical, 13 – Numerical)
Classes	2
Training Set	1331 (90% of actual)
Testing Set	148 (Remaining 10% of actual)

Table 3.1 – Linear SVM - Airline Industry Dataset

The classification of the dataset might not be perfect as the dataset is not completely linearly separable. That is, the dataset provided contains too many abnormal records (noise). But the dataset should be linearly separable in order for the linear SVM to classify perfectly. Here, to train the SVM, the method of linear programming is sought to implement this. The SVM problem is a QP problem as mentioned in the earlier chapters, which is converted into a linear programming model and solved using the lp_solve package. The given dataset is modeled into a constrained linear programming problem and the optimized outputs are obtained accordingly. The QP can be converted into a standard linear program understandable to a solver.

Given a training set of size l with examples of dimensionality d, $x_i \in \mathbb{R}^d$ and their corresponding class labels $y_i \in \{-1, +1\}$, where $i = 1, \ldots, l$. $|P|$ is the number of positive examples and $|N|$ is the number of negative examples. The variables in the LP1 are $a, b \in \mathbb{R}, x \in \mathbb{R}^d$ which adds up to $d + 2$ number of variables. Thus there are $d +$

2 columns. The objective coefficient vector (which is also of length $d + 2$) has an element for each of the variables and it is required to set the element corresponding to the variable a to 1, which will indicate the solver to minimize a. The number of constraints is equal to $|P| + |N| + d + d$, which is same as $l + 2d$. All the constraints combined can be viewed as a matrix with $d + 2$ columns and $l + 2d$ rows, and there is a vector of "dir" and "rhs" entries for each constraint. There is still one mismatch to be resolved, which is about the right hand side (rhs) entries. A solver normally requires that the rhs of a constraint is a constant and not a variable. So, the last two constraints has to be rewritten as follows;

$$a - w_j \geq 0 \; \forall \; j \in 1, \ldots, d \tag{18}$$

$$a + w_j \geq 0 \; \forall \; j \in 1, \ldots, d \tag{19}$$

Now it is required to set the corresponding variable coefficients in the constraints to the correct value and feed the problem to a solver as well as get the solution. The linear programming formulation is illustrated in figure 3.1.

The linear program solver has following components:

1. Columns: These are the variables in the linear program.
2. Objective coefficients: These describe how to linearly combine the variables for the objective function.
3. Variable bounds: These describe what values can be assumed by a variable.
4. Direction: This is the direction of optimization (either minimize or maximize).
5. Rows: Rows correspond to the constraints.
6. Constraint coefficients: These describe how to linearly combine the variables for a constraint.
7. Direction (dir): The direction of the constraint (one of the >=, <= and =).
8. Right hand side (rhs): This is the right hand side of the constraints.

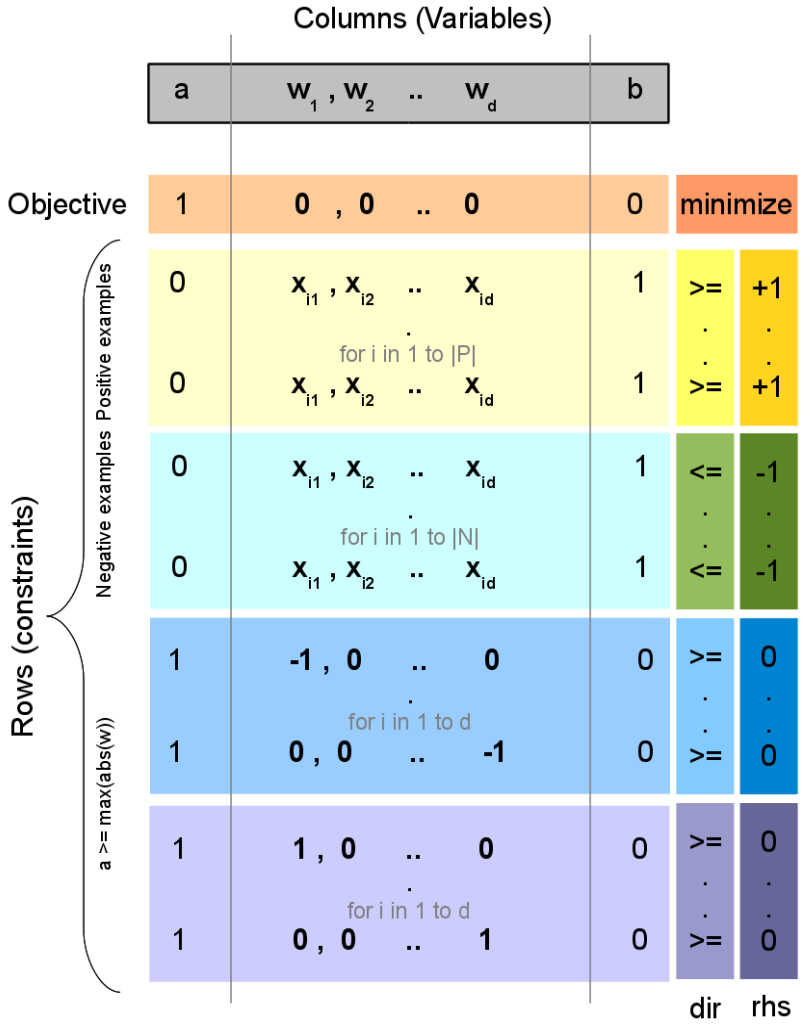

Figure 3.1 – Linear SVM – LP Formulation

This formulation can be implemented to work with a linear solver in R by training with the training set. A reasonable accuracy is obtained on testing with the testing set. The output of the optimizer has the values of the weight vectors, the bias and the value of the objective function. A sample output of this optimizer is tabulated in table 3.2. The corresponding accuracy details are mentioned below. The confusion matrix for this classification, which has the maximum accuracy (78%), is tabulated below in table 3.3. Some more accuracy measure values are shown in table 3.4. The accuracy plot for various levels of hold-out testing is shown in figure 3.2.

Variable	Value

svm.objval	1.0
svm.bias	3.0716
svm.weights1	0.01584720
svm.weights2	0.00922340
svm.weights3	0.10132301
svm.weights4	0.12211707
svm.weights5	0.04579520
svm.weights6	0.06298352
svm.weights7	0.26225212
svm.weights8	0.01718045
svm.weights9	0.04314350
svm.weights10	0.02192901
svm.weights11	0.03976413
svm.weights12	0.11711836
svm.weights13	0.14132301

Table 3.2 – Linear SVM – LP Solver Sample Output

True Positive = 60	**False Positive = 23**
False Negative = 10	**True Negative = 55**

Table 3.3 – Contingency Table for Linear SVM Classification

Sensitivity: (TP + FN)	**70%**

Specifity: (FP + TN)	**78%**
Precision: TP / (TP+ FP)	**72%**
Recall: TP / (TP+ FN)	**85%**
F-Measure	**77.9%**

Table 3.4 – Different measures of Linear SVM Classification.

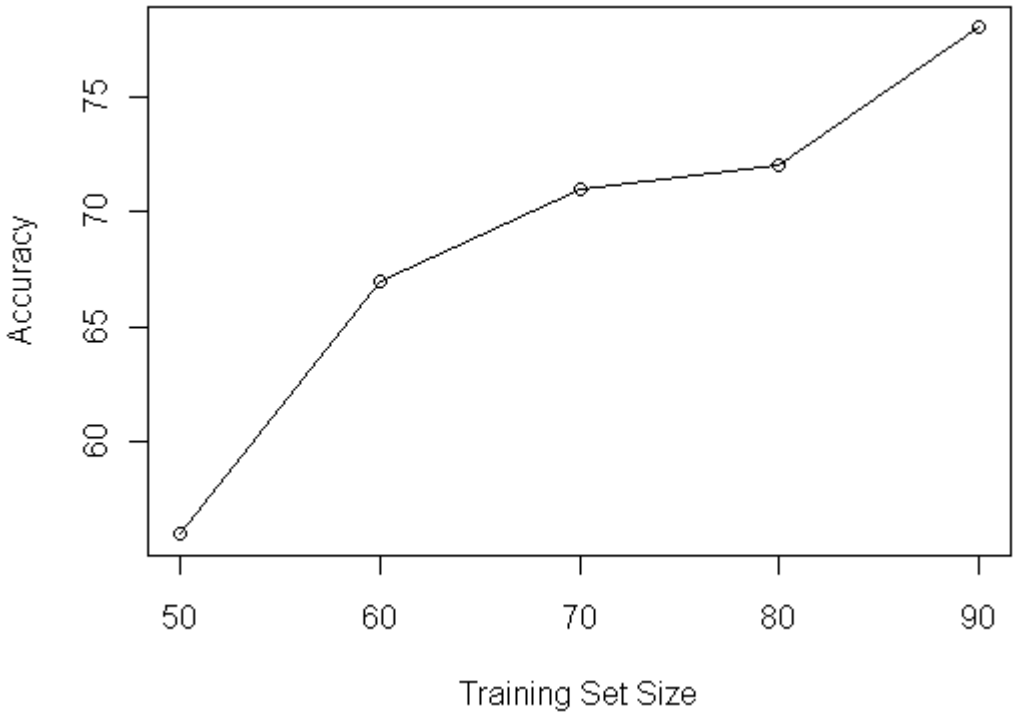

Figure 3.2 – Accuracy Plot of Linear SVM Classification

3.3 NON-LINEAR SVM

The airline industry dataset can be best classified using Non-linear SVM as it has too many noisy data. So, to implement the Non-linear SVM, one of the training algorithms has to be used. Here, the Sequential Minimal Optimization (SMO) method is adopted. The reason for choosing this algorithm for the SVM training process is clearly stated in the previous chapter. SVM classification problem is derived as a QP problem. SMO is a simple algorithm that can quickly solve the SVM classification problem which is QP problem

without any extra matrix storage and without using numerical QP optimization steps.

4. CONCLUSION

The Classification of Customer Purchase Behavior in the Airline Industry using Support Vector Machines is a work that is aimed to start with an in-depth study and understanding of the various aspects of *Support Vector Machines (SVM),* which is used widely for classification and regression purposes. The study and understanding of the SVM technique and its role in classification tasks are done. The classification process follows supervised learning model, in which the available or known data is used to train the model based on similarity measures. With the trained model, the unseen or new data is classified into either of the classes. The non-linear input space is handled using various type of *kernel functions,* which are chosen based on the application. This technique is then implemented in the *R* programming language to perform data classification task for the business data set. The implementation of the Linear SVM for classification is done for the airline industry dataset and the results are clearly tabulated. The implementation of the Non-linear SVM using the SMO algorithm is in process and following that the classification of the airline dataset will be done in the near future.

5. BIBLIOGRAPHY

REFERENCES

[1] Bernhard Schlkopf, Alexander J. Smola, “Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)”, MIT Press, 2002

[2] Christopher Burges, “A Tutorial on Support Vector Machines for Pattern Recognition”, Data Mining and Knowledge Discovery, Kluwer Publishers, 1998

[3] Emmanuel Paradis, “R for Beginners”, 2005

[4] M.A.Hearst, B.Scholkopf, S.Dumais, E.Osuna, J.Platt, "Trends and controversies — Support vector machines", IEEE Intelligent Systems, 13:18–28, 1998

[5] John C. Platt, “Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines”, Microsoft Research, 1998

[6] John Verzani, “Simple-R”, Lecture Notes (www.math.csi.cuny.edu/R), 2002

[7] Thomas Lumley, “R Fundamentals and Programming Techniques”, Lecture Notes (faculty.washington.edu/tlumley/Rcourse/R-fundamentals.pdf), 2005

WEBSITES

1. http://www.svms.org/

2. http://www.kernel-machines.org/

3. http://www.support-vector.net/

4. http://www.cran.r-project.org/

5. http://www.stat.cmu.edu/~cshalizi/uADA/12/

6. http://www.wikipedia.org/

7. http://www.stat.ethz.ch/R-manual/

8. http://www.r-bloggers.com/

Mu Sigma is a leading provider of decision sciences and analytics solutions, helping companies institutionalize data-driven decision making. We work with market-leading companies across multiple verticals, solving high impact business problems in the areas of Marketing, Supply Chain and Risk analytics. For these clients we have built an integrated decision support ecosystem of people, processes, methodologies & proprietary IP and technology assets that serve as a platform for cross-pollination and innovation. Mu Sigma has driven disruptive innovation in the analytics industry by integrating the disciplines of business, math, and technology in a sustainable model. With over 75 Fortune 500 clients and over 2000 decision science professionals we are one of the largest pure-play decision sciences and analytics companies.

Learn more at *http://www.mu-sigma.com/contact.html* us for further information:

Mu Sigma Inc., 3400 Dundee Rd, Suite 160, Northbrook, IL – 60062 www.mu-sigma.com