MACHINE LEARNING USING SUPPORT VECTOR MACHINES

M. Palaniswami*, A. Shilton*, D. Ralph** and B.D. Owen*

*Department of Electrical and Electronic Engineering
The University of Melbourne, Victoria-3101, Australia.

**The Judge Institute of Management Studies
University of Cambridge
Trumpington St, Cambridge CB2 1AG, UK.

Abstract
Machine learning invokes the imagination of many scientific minds due to its potential to solve complex and difficult real world problems. This paper gives methods of constructing machine learning tools using Support Vector Machines (SVMs). We first give a simple example to illustrate the basic concept and then demonstrate further with a practical problem. The practical problem is concerned with electronic monitoring of fishways for automatic counting of different fish species for the purpose of environmental management in Australian rivers. The results illustrate the power of the SVM approaches on the sample problem and their computational attractiveness for practical implementations.

1. INTRODUCTION
Machine learning is an attractive field in the domain of Artificial Intelligence (AI) with the scope to learn from presented experimental data for the purpose of intelligent interpretation when the system is confronted with unseen situations. In the field of artificial neural networks, several neural networks architectures have been presented with the view to generating generalised mappings from input to output in a robust manner. In this paper, we give a technique that is increasing in popularity under the name Support Vector Machines [1, 4], which is also a universal feedforward approximator much like the layered feedforward networks and Radial Basis Function Networks. We give the basic concept behind this emerging paradigm and illustrate it with a practical example.

2. SUPPORT VECTOR MACHINES

2.1 Background
A common problem that can be observed in many AI engineering applications is pattern recognition [4]. The problem is as follows – given a “training set” of vectors, each belonging to some known category, the machine must learn, based on the information implicitly contained in this set, how to classify vectors of unknown type into one of the specified categories.

Support vector machines (SVMs) provide one means of tackling this problem. In order to provide a basis for classification, SVMs implicitly map the training data into a high-dimensional feature space. A hyperplane is then constructed in this feature space which maximises the margin of separation between the plane and those points lying nearest to it (called the support vectors).

The plane so constructed can then be used as a basis for classifying vectors of uncertain type.

2.2 Linearly Separable Data
For simplicity, we consider the problem of two category classification. Consider the training pair: \((x_i, d_i)\) where \(x_i = \) training vector, and \(d_i = \) category (±1). Here \(i\) runs from 1 to \(N\), the number of training points.

Assume that a hyperplane which divides the training data can be found, without mapping to feature space.

The decision surface hyperplane will be defined by a discriminant function,

\[ g(x) = w^T x + b = 0 \]

where the vector \(w\), of dimension equal to that of \(x\), and scalar \(b\) are chosen such that

\[ w^T x_i + b > 0 \Rightarrow d_i = +1 \]
\[ w^T x_i + b < 0 \Rightarrow d_i = -1 \]

(Note we assume the classification is strict: no data point lies on the decision surface \(g(x) = 0\).)

Classification of an unknown vector \(x\) into class membership \(d\) (±1) is done using the discriminant function:
\[ d = \text{sign}(g(x)) \]

The support vectors are those training vectors \( \mathbf{x} \) which lie closest to this plane. The notation \( (\mathbf{x}^{(s)}, d^{(s)}) \) refers to a training pair \((\mathbf{x}, d)\) such that \( \mathbf{x} \) is a support vector.

Consider any vector \( \mathbf{x} \). This can be expressed as:

\[ \mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{||\mathbf{w}||}, \text{where: } r = \frac{g(\mathbf{x})}{||\mathbf{w}||} \]

By scaling \( \mathbf{w} \) and \( b \), we can ensure that:

\[ d_i g(\mathbf{x}_i) \geq 1 \]

where equality implies that the point is a support vector. Hence, for the support vectors:

\[ r = \frac{g(\mathbf{x}^{(s)})}{||\mathbf{w}||} = \frac{d^{(s)}}{||\mathbf{w}||} \]

Now, based on this, we define the margin of separation between the two classes as:

\[ \rho = 2r = \frac{2}{||\mathbf{w}||} \]

So, in order to maximise the margin of separation, we must minimise \( ||\mathbf{w}|| \), or \( ||\mathbf{w}||^2 \) for convenience, subject to the constraints given previously. To summarise the classification problem, it is a quadratic program [3] in the variables \( \mathbf{w} \) and \( b \):

\[
\text{Minimise } \Psi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \\
\text{subject to } d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1
\]

### 2.3 Nonlinear Decision Surface

For complex problems, we must first map our data into feature space prior to constructing our decision surface. Suppose that this is done via some arbitrary mapping into an arbitrary dimension \( m \):

\[ \varphi(x) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_m(\mathbf{x})]^T \]

The classification problem in feature space is now:

\[
\text{Minimise: } \Psi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \\
\text{Subject to: } d_i (\mathbf{w}^T \varphi(\mathbf{x}_i) + b) \geq 1
\]

where the variable vector \( \mathbf{w} \) is now an \( m \)-dimensional vector and the variable \( b \) is a scalar as before. Given the optimal \( \mathbf{w} \) and \( b \), the discriminant function coming out of this support vector machine is

\[ g(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) + b \]

#### 2.3.1 Non-separable data

If the training data is not separable in the chosen feature space, then the above approach will fail, as the conditions can never be met (causing \( \mathbf{w} \) to diverge to \( \infty \)). To deal with this, a soft-margin technique can be used. Non-negative slack variables \( \xi \) are introduced which allow the constraints to be weakened when they cannot be met.

The re-formulated quadratic program has variables \( \mathbf{w} \), \( b \) and \( \xi \) and uses a constant \( C \):

\[
\text{Minimise: } \Psi(\mathbf{w}, \xi) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \xi^T \xi \\
\text{Subject to: } d_i (\mathbf{w}^T \varphi(\mathbf{x}_i) + b) \geq 1 - \xi_i
\]

\( C \) is an arbitrary constant that may be used to control the trade off between machine complexity (correctly classifying outliers, possibly at the expense of misclassifying some good data) and robustness (limiting the ability of outliers to distort the separating plane). See Vapnik [1] for an elegant theoretical underpinning of this idea.
2.3.2 Kernel functions and the Wolfe dual

In feature space, the dot product is:

\[
(x, y) \rightarrow \phi(x)^T \phi(y)
\]

\[
= \phi_1(x)\phi_1(y) + \ldots + \phi_m(x)\phi_m(y)
\]

The inner product kernel is defined as:

\[
K(x, y) = \phi(x)^T \phi(y)
\]

This can be treated as a generalised form of the dot product for a curved input space. Note that the feature space may have infinite dimension so long as we never have to refer to the mapping to feature space explicitly. For the linearly separable case, we can simply use the standard dot product \(K(x, y) = x^T y\).

Any arbitrary function can be used as a kernel, but good theoretical properties rely on it being related to an inner product on the feature space. This will be true if the kernel satisfies Mercer’s condition, which is:

\[
\text{For all } \psi(\bullet) \text{ for which } \int_{\mathbb{R}^d} \psi^2(x) d\lambda \text{ is defined,}
\]

\[
\int_{\mathbb{R}^d} \int_{\mathbb{R}^d} K(x, y) \psi(x)\psi(y) d\lambda x d\lambda y \geq 0
\]

and the kernel is symmetric.

The primal problem appears to be impractical when feature spaces of very high dimension are used (and impossible if the feature space has infinite dimension). To overcome this we solve the Wolfe dual quadratic program instead of the primal quadratic program. Derivation of the following Wolfe dual can be found elsewhere, in Burges, for example, and more generally in [3]. The Wolfe dual can be written:

\[
\text{Minimise: } Q(\alpha) = \frac{1}{2} \alpha^T G \alpha - \alpha^T \psi
\]

\[
\text{Subject to: } d^T \alpha = 0
\]

\[
0 \leq \alpha \leq C
\]

where the \(G\) matrix has \(N\) rows and columns:

\[
G_{ij} = d_i d_j K(x_i, x_j)
\]

Having solved the dual problem to obtain an optimal dual vector \(\alpha\), we can easily recover the discriminant function \(g\) as we describe next. What is interesting, and very important from the point of view of implementation, is that this is possible without needing to know what the feature mapping \(\phi\) is, but only that it exists and satisfies the kernel equation (1).

It can be shown that the optimal \(w\) and \(b\) are given by

\[
w = \sum_{i=1}^{N} \alpha_i d_i \phi(x_i)
\]

\[
b = d^{(S)} - \sum_{i=1}^{N} \alpha_i d_i K(x_i, \xi^{(S)})
\]

where \((\xi^{(S)}, d^{(S)}) = (x_i, d_i)\) is any training pair such that \(0 < \alpha_i < C\); the fact that \(\alpha\) is positive can be shown to imply that \(\xi\) is a support vector. Thus

\[
g(x) = w^T \phi(x) + b
\]

\[
= \sum_{i=1}^{N} \alpha_i d_i \phi(x_i)^T \phi(x) + b
\]

\[
= \sum_{i=1}^{N} \alpha_i d_i K(x_i, x) + b
\]

which is expressed without reference to \(\phi\). As usual, a new data vector \(x\) can be classified by \(d = \text{sign } g(x)\).

This overcomes the problems inherent when dealing with high dimensional feature spaces that are intrinsic to a kernel mapping.

2.4 Example – simulating an XOR gate

We have adapted an example from [4].

Training set:

<table>
<thead>
<tr>
<th>(x)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-1,-1)</td>
<td>-1</td>
</tr>
<tr>
<td>(-1,1)</td>
<td>1</td>
</tr>
<tr>
<td>(1,-1)</td>
<td>1</td>
</tr>
<tr>
<td>(1,1)</td>
<td>-1</td>
</tr>
</tbody>
</table>

Choose the quadratic kernel

\[
K(x, y) = (1 + x^T y)^2
\]

We can compute

\[
G = \begin{bmatrix}
9 & -1 & -1 & 1 \\
-1 & 9 & 1 & -1 \\
-1 & 1 & 9 & -1 \\
1 & -1 & -1 & 9
\end{bmatrix}
\]
Solving the Wolfe-dual gives

Solution: \[ \alpha = \begin{bmatrix} 0.125 \\ 0.125 \\ 0.125 \\ 0.125 \end{bmatrix} \]

\[ b = 0 \]

The feature mapping associated with the above kernel can be constructed explicitly [4], leading to the following compact representation of the decision surface: \( x_1x_2 = 0 \)

This is sensible, as all it does is to divide the quadrants of the two-dimensional input space. Explicitly, the decision surface represented here is:

3. FISH CLASSIFICATION USING SUPPORT VECTOR MACHINES

Fishways are constructed in rivers to help migratory fish get over obstacles (dams, weirs, etc.). In order to study their success, it is important to monitor both the number and species of fish using them. It is very time consuming to do this manually (by viewing video footage and counting by hand). A better alternative would be to train a machine to recognize the different species and use this machine to do the count in real time.

The following results were obtained to assess the feasibility of using support vector machines for this purpose.

3.1 Training data and experimental details

Results were obtained based on fish in a tank. The species involved were silver perch, giant danio, danio, tiger barb and rainbow fish; species 1-5 respectively. Multiple images of each species were taken, and feature extraction was used to reduce each image to a 110 dimensional vector.

We have coded a set of 110 feature extractors from various sources. 13 shape features [5], 11 length features [6], 2 area ratios [7], 15 binary moments (up to 4\textsuperscript{th} order) and the corresponding 15 shade moments, 19 moment invariants [8] and the remaining features are some of the previous features normalized such that the isolated fish has unit area. A linear transform is found such that the final training features are normalized such that they lie between 0 and 1. The same linear transform is then applied to all the testing features.

1000 training and testing vectors were used. Of these, 800 were chosen at random to train the support vector machine and the remainder used to evaluate the results. For consistency, the process was repeated 10 times.
Multiclass (5 species) classification was achieved using a simple winner-takes-all method. This corresponds to a baseline accuracy of 20% (i.e. 20% accuracy would be achieved by randomly assigning a species to each test vector).

3.2 Results using support vector machine

The first set of results were generated using a simple polynomial kernel, namely:

$$K(x, y) = (1 + \frac{r}{y} \cdot x)^p$$

Results are shown in the following graph:

A radial basis function kernel was also tried:

$$K(x, y) = e^{-\frac{1}{2\sigma^2}\|x - y\|^2}$$

4. BATCH VERSUS INCREMENTAL SOLUTION

The traditional approach to solving the Wolfe dual has been to use a batch solving method. The main disadvantage to this approach is that it is not possible to incorporate new training data as it comes to hand (as we must solve again from scratch).

An alternative approach is that of incremental learning. This makes the assumption that the new data only contains a small amount of information. Hence, our current solution is very close to the optimal solution, and can be made optimal through relatively minor modifications.

Active set methods for quadratic programs [3] present a natural framework in which to investigate the effect of perturbations of the problem data on the optimal value and optimal solution – known as sensitivity analysis – and then to derive incremental methods. The active set technique can be traced back to pivotal methods like the Simplex method [3] for linear programming.

For this problem, we compared the computational cost of repeated batch solving and incremental solving using a quadratic kernel. The method was as follows:

- The SVM was initially trained using N-20 training pairs
- 20 more training pairs were added, and the resulting problem solved using both incremental and batch methods.

Both the batch and incremental methods used were active set methods. In the incremental method, the alphas corresponding to each of the new data points was set to 0 or C depending on whether the new training pair was correctly classified by the old SVM or not. Batch methods simply solved from scratch.

The following graph shows the difference between the flop counts for the incremental and batch solves.

5. CONCLUSION

This paper has given methods of constructing Support Vector Machines for the purpose of machine learning. The classification performance is illustrated with an on-going practical application problem in computer vision and environment management. In many practical situations, new information is added and this information is to be incorporated efficiently without
significantly affecting the old information. For this purpose, results are provided to demonstrate the usefulness of incremental algorithms that deal with incremental data.

6. REFERENCES