Support Vector Machines for Object Recognition

B. Smith, R. Gosine
C-CORE

1. Abstract

Support Vector Machines [1-4] is a learning technique that can be viewed as a new method for training polynomial, neural network, or Radial Basis Functions Classifiers. The decision surfaces are found by solving a linearly constrained quadratic programming problem. This paper evaluates the use of Support Vector Machines for object recognition. The dataset used is made up of different poses of 3-D objects. The system does not require feature extraction or pose estimation. The recognition rates achieved in the experiments were very good, in comparison with neural networks.

2. Introduction

In general, an Object Recognition System (ORS) has several parts. These include image sensors, image preprocessing, object detection, object segmentation, feature extraction and object classification. The problem that is addressed in this paper is object classification. Specifically, algorithms are considered that can take, as input, a digital image, and classify it according to some criterion. There are many different approaches to object classification, including image analysis, pattern recognition, model-based vision, artificial neural net classification, and knowledge-based reasoning. Here a new technique, Support Vector Machines (SVMs), is used.

3 Classifiers

Optimal Separating Hyperplane

To illustrate the concept of the Optimal Separating Hyperplane (OSH), we will use an example. Given examples with weight, height and sex of a person, we develop a hypothesis which enables us to determine a persons sex from their weight and height. This can be done by plotting the weights and heights in a 2-D coordinate system and drawing a dividing line or separating hyperplane to separate the weight/height points into male and female regions. A typical plot is shown in Figure 1.

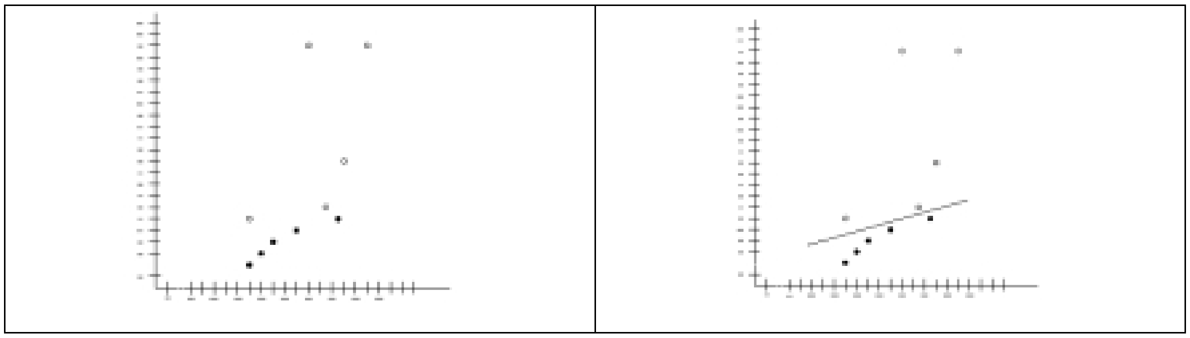
Figure 1. (a) Weight vs Height of People. The solid dots are female, circles male. (b) OSH separating males and females.

There are a number of possible lines that can separate the data, but there is only one that maximizes the margin (maximizes the distance between it and the nearest data point of each class). This line gives the ‘best’ results, where ‘best’ means that it gives the highest classification rate when new data is used. Intuitively, we would expect this line to generalize well as opposed to other possible ones. This line is called the Optimal Separating Hyperplane. Figure 1 (b) shows the data redrawn with the OSH included.

The OSH algorithm is based on finding a pair of parallel hyperplanes, which separate the data and which has the largest perpendicular distance between them. It is conjectured that this provides a good approximation to the ‘best’ separating hyperplane. Thus if we can find an OSH from the training data, as a mathematical function, then only basic geometry is used to determine which side of the line any given point lies and make a classification of an unseen point. SVMs use geometric properties to exactly calculate the OSH directly from the training data.

Given the following training data:

$$(x_1,y_1),\dots,(x_m,y_m),\ x = \text{real},\ y = \{+1,-1\} \qquad (1)$$

where each data point is described by a feature vector x_i and a truth value y_i, the latter of which can take the values of +1 and –1, depending on the class. The two hyperplanes are required to pass through at least one point of each class and there can be no points between them. The boundary between the classes is then defined to be a third parallel hyperplane that is halfway between the other two. The data points that the outer hyperplanes pass through, are called *Support Vectors*. The two outer hyperplanes are described by the following expressions,

$$\begin{aligned}(w \bullet x) + b &= +1, \qquad (2)\\ (w \bullet x) + b &= -1, \qquad .\end{aligned}$$

with the first going through a point of class y = +1 and the second going through a point of class y = -1. The constants w and b define the hyperplanes, with w being normal to the hyperplanes and $-b/\|w\|$ being the perpendicular distance from the origin to the middle hyperplane. The RHS of Equation (2) will be greater than or equal to +1 for all points of class y = +1 and will be less than or equal to –1 for all points of class y = -1. These can be combined into the following constraint on all the data points,

$$y_i[(w \bullet x_i) + b] - 1 \geq 0,\ i = 1,\dots m \qquad (3)$$

The perpendicular distance between the two outer hyperplanes (margin) is equal to $2/\|w\|$. Therefore, finding the hyperplanes with the largest margin reduces to finding values for w and b that maximize $2/\|w\|$ or equivalently minimize $\frac{1}{2}\|w\|^2 = \frac{1}{2}(w \bullet w)$, subject to the constraint in Equation (3).

In other words,
Minimize: $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$
Subject to: $y_i[(w \bullet x_i) + b] \geq 1$, $i = 1, \dots m$

A standard method for handling optimization problems with constraints is through the minimization of the Lagrangian. The constraints are taken into account by adding terms involving Lagrange multipliers to the objective function. In this case, this results in the following primal Lagrangian,

$$L_P = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m}\alpha_i y_i (w \cdot x_i + b) + \sum_{i=1}^{m}\alpha_i \qquad (4)$$

where α_i are the Lagrange multipliers associated with each of the constraints in Equation (3). The Lagrangian has to be minimized with respect to the primal variables w and b, and maximized with respect to the dual variables α_i (ie a saddle point exists). At the saddle point, the derivatives of L_p with respect to the primal variables must be equal to zero. Doing this results in the following expressions,

$$w = \sum_{i=1}^{m}\alpha_i y_i x_i \qquad (5)$$

$$\sum_{i=1}^{m}\alpha_i y_i = 0 \qquad (6)$$

while from the definition of the Lagrange multipliers, we get,

$$\alpha_i \bullet \left(y_i\left(w \bullet x_i + b\right) - 1\right) = 0 \qquad (7)$$

Inserting Equations (5) and (6) into (4), removes the primal variables and results in the Wolfe dual Lagrangian where we just have to find the α_i which maximize:

$$L_D = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum \alpha_i \alpha_j y_i y_j (x_i \bullet x_j) \qquad (8)$$

subject to $\alpha_i > 0$, i = 1 … m, and Equation (6)

This is attractive because the problem is reduced to finding the Lagrange multipliers (the dual variables) that maximize Equation (8) and satisfy both the non-negative constraints and the constraints of Equation (6). Equation (7) means that only those data points which lie on the outer hyperplanes (and hence are active constraints) will have non-zero Lagrange multipliers. These data points are called the support vectors and they are the points that determine the position of the hyperplanes. One can move the other points around the feature space or remove them entirely and the solution will not change, provided one does not move a point across one of the outer hyperplanes.

Equation (8) can be solved using any quadratic programming solver. Once the Lagrange multipliers are known, the solution for w is given by Equation (5), where the sum is over the support vectors, since they are the only ones with non-zero α. One can find b from Equation (7), using any of the support vectors, although one generally averages over all the support vectors for better accuracy. Once these constants are known, the classification of an unknown vector, v, is given by the sign of,

$$b + \sum \alpha_i y_i x_i \bullet v \qquad (9)$$

where the sum is over the support vectors. This determines on which side of the boundary (or middle) hyperplane that the data point falls.

Extending OSH to SVM

SVMs form an extension to the OSH method. They map the input space (ie the 32x32 pixel image) into a high-dimensional feature space through some non-linear mapping function and then construct the OSH in the feature space. This makes it possible to construct linear decision surfaces in feature space which correspond to non-linear surfaces in input space. This is for the case where a linear boundary is unable to separate the data in input space. For this case, the SVM can map the input vector, **x**, into a high dimensional feature space, **z,** through a nonlinear transformation Φ. By choosing the non-linear mapping before training, the SVM constructs an OSH in this higher dimensional space. The most common mappings are polynomials, radial basis functions and various sigmoid functions.

Adding another point to the set of points in the previous example, we see that an OSH can't be used to separate the data (Figure 2 (a)), but if we use a non-linear separating hyperplane, which is equivalent to mapping into a high dimensional space, a separation is possible. See Figure 2 (b).

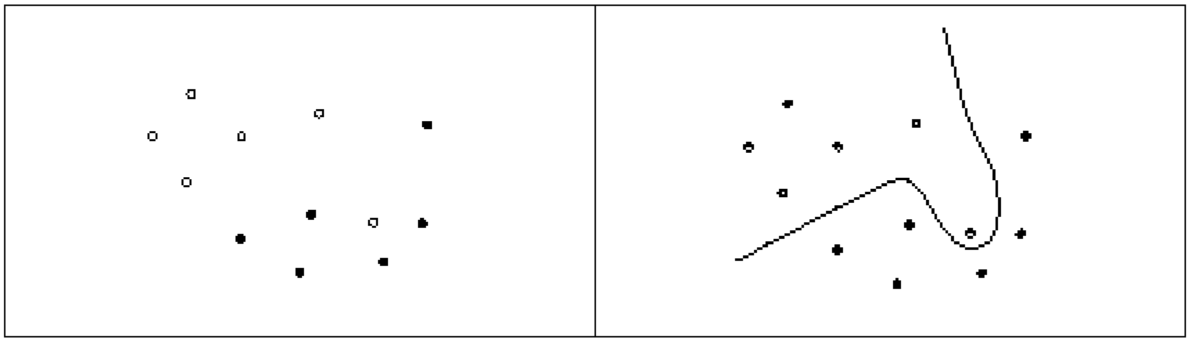

Figure 2. (a) A linear separating hyperplane can't be used to separate the data. (b) A non-linear separating hyperplane separates the data.

This results in the Lagrangian in Equation (8) being transformed to,

$$L_D = \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j \Phi(x_i) \bullet \Phi(x_j) \qquad (10)$$

and the classification relation in Equation (9) becomes,

$$b + \sum \alpha_i y_i \Phi(x_i) \bullet \Phi(v) \qquad (11)$$

Since Equations (10) and (11) depend only on the dot product between the two transformed feature vectors, one can employ a kernel function,

$$K(x, y) = \Phi(x) \bullet \Phi(y) \qquad (12)$$

and never need to compute the transformation Φ explicitly. Equation (11) then becomes,

$$b + \sum \alpha_i y_i K(x_i, v) \qquad (13)$$

with the test feature vector now inside the summation over the support vectors. In general, the mapping Φ will be to a higher dimensional space. Since one is still solving the linear problem, just in a different space, the computational overhead is essentially the same. The solution and parameters for the hyperplane are in the higher dimensional space and when one transforms back to the original space the boundary become nonlinear. However, in general, there is no way to analytically invert the solutions for w and b. Hence, one must use Equation (13) to classify test feature vectors.

Artificial Neural Networks – Backpropagation

In order to obtain a better idea of the performance of SVMs, we run the same experiments using Backpropagation [5] Neural Networks (BP-Net). The BP-Net accepts the same input as SVMs, a 32x32 = 1024 vector. It has 10 hidden layer neurons and 2 output neurons, each with a log sigmoid transfer function. The BP-Net uses as the training function, a function that updates weight and bias values according to gradient descent momentum and an adaptive learning rate. The BP-Net uses a Sum-Squared Error with a goal of 0.1 for its training performance function.

4. Dataset Used

The COIL (Columbia Object Image Library) database consists of 1,440 images of 20 objects (72 views for each of the 20 objects). The COIL images are 8-bit grayscale images of 32x32 pixels. The images were obtained from www.cs.columbia.edu. The objects are positioned in the center of a turntable and observed from a fixed viewpoint. The turntable is rotated 5 degrees for each image, giving a total of 72 images per object. A sample of the objects are shown in Figure 3 and Figure 4. Note that the object region seems to have been re-sampled so that the larger of the two object dimensions fits the image area. This means that the apparent size of an object may change a lot from image to image.

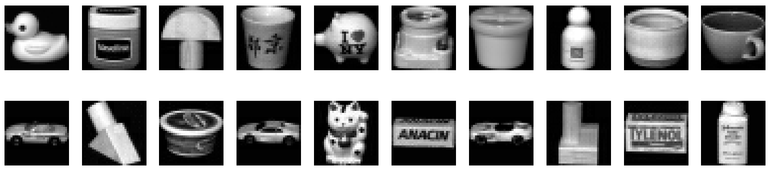

Figure 3. Some COIL Images

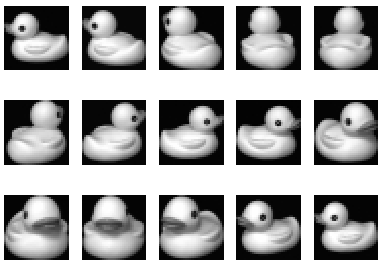

Figure 4. Various views of one COIL object

5. Experimental Results

The general strategy behind each experiment is essentially the same. Of the 72-images of an object, 36 are randomly chosen to be used as training data, the remaining 36 are used as test data. This is done 100 times and the average hit rates and correct rejection rates are recorded. To get a better feel for the performance of the classifier under a more realistic approach, zero mean noise with a guassian distribution was added to the images. The amount of noise, determined by the noise variance, was varied to see the effect of increased noise on the classifiers. The variances include 0.01, 0.02, 0.05, 0.10, 0.20, 0.50 and 0.80.

The D-Prime statistic is used to compare the ability of the classifiers to classify the objects. To give some perspective on the numbers, a perfect classifier (ie 100% hit rate and 100% correct rejection rate) would have a D-Prime of 7.8, whereas a very bad classifier (ie 0% hit rate and 0% correct rejection rate) would have a D-Prime of minus 6.16.

Experiment 1: Training with Noiseless Data, Testing with Noisy Data

The general experiment algorithm is used with varying amounts of noise added to the test data only. The experiment was run for some of the harder to classify objects, with typical results being shown in Table 1 for an Anacin Pill Box vs a Tylenol Pill Box. The SVM algorithm performs much better at all levels of noise than the BP-Net. Even at 0.20 noise variance, the SVM algorithm is performing about the same as the BP-Net without noise added.

Table 1. D-Primes for Anacin Box vs Tylenol Box for (a) noise added to Test Set only and (b) noise added to Training and Test Sets

(a)			(b)		
Noise	**SVM**	**BP-Net**	**Noise**	**SVM**	**BP-Net**
0.00	5.95	4.36	0.00	5.95	4.36
0.01	5.95	2.03	0.01	5.37	4.91
0.02	5.64	1.54	0.02	5.22	4.59
0.05	5.36	0.97	0.05	4.62	3.92
0.10	5.13	0.56	0.10	3.74	3.04
0.20	4.22	0.41	0.20	2.67	2.04
0.50	2.88	0.16	0.50	1.44	0.91
0.80	2.32	0.14	0.80	1.05	0.66

Experiment 2: Training with Noisy Data, Testing with Noisy Data

The general experiment algorithm is used with varying amounts of noise added to both the training data and the test data. The experiment was run for some of the harder to classify objects, with typical results being shown in Table 1 for an Anacin Pill Box vs a Tylenol Pill Box. The SVM algorithm performs much better at all levels of noise than the BP-Net. Adding noise to the training set improved the classification of the BP-Net, as compared with training without noise, yet decreased the ability of the SVM. This is because as noise was added to the training set, more support vectors are needed for good classification, and since training was performed on only a small number of samples (36 for each object), the classifier quickly ran out of SVs. More SVs are required as the noise level increases.

6. Conclusions

SVMs appear to offer superior performance for two-object classification, both in terms of their ability to classify and in the amount of processing time required for classification. Future work will include integrating a SVM classifier with a full object recognition system for application in robot vision.

7. References

1. Scholkoph, B., Burges, C., Smola, A., “Advances in Kernel Methods: Support Vector Learning”, MIT Press, 1999
2. Pontil, M., Verri, A., “Support Vector Machines for 3-D Object Recognition”, IEEE Trans. PAMI, Vol 20, pp 637-646, 1998.
3. C. Papageorgiou, M. Oren, and T. Poggio. “A general framework for object detection”, International Conference on Computer Vision ICCV'98, 1998.
4. E. Osuna, R. Freund, and F. Girosi., “Support Vector Machines: Training and Applications”, Technical Report AIM-1602, MIT A.I. Lab., 1996.
5. Muller. B., Reinhardt. J., “Neural Networks: An Introduction”, Springer-Verlag, 1990