A SEMI-SUPERVISED SUPPORT VECTOR MACHINE FOR TEXTURE SEGMENTATION

Saeid Sanei

Deptartment of Electronic Engineering,
King's College London,
Strand, London, WC2R 2LS, UK
saeid.sanei@kcl.ac.uk

Tracey K. M. Lee

Department of Electrical Engineering
National University of Singapore,
Singapore
tlee@sp.edu.sg

ABSTRACT

A novel semi-supervised support vector machines (S^3VM) algorithm is developed for segmentation of texture images. The method classifies the uniform-texture regions from the regions of boundaries. The various-order statistics of the textures within a sliding two-dimensional window are measured. K-mean algorithm is used to approximately label the two clusters in the overall image. However, only those clusters, which are very close to the class centres, are labelled. Therefore at this stage we have both the training and the working sets. A non-linear S^3VM is then developed to exploit both sets to classify the regions. It is demonstrated that the algorithm is robust and the misclassification error is negligible. However, there may be a minor misplacement of the edges.

1. INTRODUCTION

Support vector machines (SVMs) would appear to be good candidates for texture segmentation because of their ability to generalize in high-dimensional spaces spanned by texture patterns [1]. SVMs are convex learning algorithms, which approximate implementation of the structural risk minimization (SRM) method [2]. For many pattern classification applications SVMs have shown a better performance in comparison with other classifiers such as neural networks and Bayesian classifiers [3][4]. For traditional nonlinear SVMs [5][6] an optimal separating hyperplane (OSH) is identified that maximizes the margin of the nearest training samples. The OSH (generally nonlinear,) is then computed as a decision surface of the form

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{L_s} \mathbf{y}_i \alpha_i K(\mathbf{x}_i^s, \mathbf{x}) + b\right) \qquad (1)$$

In this formula $\text{sgn}(\cdot) \in \{\pm 1\}$, $\mathbf{x}_i^s$ is a subset of the training data set called support vectors (SVs), L_s is the number of SVs and $K(\mathbf{x}_i^s, \mathbf{x})$ is the nonlinear kernel function. SVs are the points from the dataset that fall closest to the separating hyperplane. Constructing an optimal hyperplane is equivalent to finding all non-zero α_i. The coefficients α_i (Lagrange multipliers) and *b*, the bias, are determined by solving the large-scale quadratic programming problem [7] with linear constraints for non-separable sets. This leads to maximization of

$$L(\alpha) = \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{j=1}^{L} \sum_{i=1}^{L} \alpha_i \alpha_j \mathbf{y}_i \mathbf{y}_j K(\mathbf{x}_i^s, \mathbf{x})$$

$$\text{s.t.} \sum_{i=1}^{L} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C \text{ for } i = 1, \ldots, L \qquad (2)$$

with respect to α_i using conjugate gradient method. Here, *L* is the number of training vectors and $y_i \in \{\pm 1\}$ are the output targets. The parameter *C* is a fixed soft margin, also called (misclassification) penalty term, and can be considered as the regularization parameter is selected by the user. A larger *C* is equivalent to assigning a higher penalty to the training errors. *C* is usually set to a high value to avoid any training error.

Any vector $\mathbf{x}_i$ that corresponds to a non-zero α_i is a support vector (SV) of the optimal hyperplane. It is desirable to have the number of SVs small to have more compact classifier. A popular and suitable kernel is radial based function (RBF) as $K(\mathbf{x}_i^s, \mathbf{x}) = \exp(-\gamma|\mathbf{x} - \mathbf{x}_i|^2)$. Therefore the operation of an SVM in traditional texture segmentation is two-fold; (1) the nonlinear mapping of a texture space into a possibly high-dimensional feature space, and (2) the construction of an OSH in the feature space. The major problem with the traditional SVM classifiers is that they are supervised i.e. the targets have to be known by the learning algorithm. This is not suitable when a wide range of texture images with features of various statistical properties, or in places where the data to be classified is much larger than the training data. A semi-supervised SVM overcomes this problem.

0-7803-8554-3/04/$20.00 ©2004 IEEE.

2. SEMI-SUPERVISED SVM

The objective is to assign class labels to the working set such that the best SVM is constructed. To formulate the S^3VM we add two constraints for each point in the working set. One constraint calculates the misclassification error as if the point were in class 1 and the other constraint calculates the misclassification error as if the point were in class -1. The objective function calculates the minimum of the two possible misclassification errors. The final class of the points corresponds to the one that results in the smallest error. By amending equation (2) we have

$$\begin{aligned} \min_{w,b,\gamma,\eta,\xi} \left(\frac{1}{2}||\mathbf{w}||^2 + C \left[\sum_{i=1}^{l} \gamma_i + \sum_{j=l+1}^{l+k} \min(\xi_j, \beta_j) \right] \right) \\ \text{s.t.} \quad y_i(\mathbf{w}.\mathbf{x}_i + b) + \gamma_i \geq 1 \quad \gamma_i \geq 0 \;\; i = 1, \ldots, l \\ and \quad (\mathbf{w}.\mathbf{x}_j + b) + \xi_j \geq 1 \quad \xi_j \geq 0 \;\; j = l+1, \ldots, l+k \\ and \quad -(\mathbf{w}.\mathbf{x}_j + b) + \beta_j \geq 1 \quad \beta_j \geq 0 \end{aligned} \tag{3}$$

where $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$ determines the orientation of a discriminant plane and *l* is the number of training vectors and *k* is the size of working set. γ, ξ and β are slack parameters. The optimum solution for above equation, in the form of $\mathbf{w} = \sum_{i=1}^{L_s} \alpha_i \mathbf{y}_i \mathbf{x}_i$, requires joint minimisation of an objective function and a number of constraints. L_s is the number of support vectors. Practically integer programming is used as an alternative for solving the problem. The basic idea is to add a decision variable $d_j = \{0, 1\}$ for each point $\mathbf{x}_j$ in the working set. This variable indicates the class of the point; if $d_j = 1$ then the point belongs to class 1 and if $d_j = 0$ the point is in class -1. Therefore (3) changes to

$$\begin{aligned} \min_{w,b,\gamma,\eta,\xi,d} \left(\frac{1}{2}||\mathbf{w}||^2 + C \left[\sum_{i=1}^{l} \gamma_i + \sum_{j=l+1}^{l+k} \min(\xi_j, \beta_j) \right] \right) \\ \text{s.t.} \quad y_i(\mathbf{w}.\mathbf{x}_i + b) + \gamma_i \geq 1 \quad \gamma_i \geq 0 \;\; i = 1, \ldots, l \\ and \quad (\mathbf{w}.\mathbf{x}_j + b) + \xi_j + \chi(1 - d_j) \geq 1 \\ \xi_j \geq 0 \;\; j = l+1, \ldots, l+k \\ and \quad -(\mathbf{w}.\mathbf{x}_j + b) + \beta_j + \chi d_j \geq 1 \quad \beta_j \geq 0 \quad d_j = \{0, 1\} \end{aligned} \tag{4}$$

The constant $M > 0$ is chosen sufficiently large such that if $d_j = 0$ then $\xi_j = 0$ is feasible for any optimal **W** and *b*. Likewise if $d_j = 1$ then $\beta_j = 0$. A globally optimal solution to this problem is achieved by integer programming [8]. A similar algorithm developed in MATLAB separates the edges from non-edge regions. This scheme is different from other classification algorithms by which distinct textures are classified based on a number of features extracted from those textures. Therefore, the initial values for *M* and d_j for above equations are identified within the sliding window. Finally an S^3VM is developed to exploit the working set within the sliding frame to classify the parameters and detect the regions of the boundaries. Minor post-processing will be then performed to provide a one-pixel-width edge map from these regions.

3. TEXTURE SEGMENTATION

We deem edges as those areas where change occurs. Changes often take the form of a step, which might occur in various order features. Between these changes the regions are assumed stationary or semi-stationary.

3.1. Feature Detection

We extend the one dimensional non-parametric detection methodology proposed in [9] and [10] into a two-dimensional space to detect a change in a discrete two-dimensional random process $I(x, y)$, as an image, at point $s_0 = (x_0, y_0)$. It is assumed that $I(x, y)$ is stationary for $s < s_0$ and it is also stationary for $s > s_0$ but with some new set of characteristics.

Let $\eta(z, \tau) = \max(\eta_x(z, \tau_x), \eta_y(z, \tau_y))$ as the corresponding HOS, where τ values are space variables $\eta_x(z, \tau) = E[I(x, y)I(x + \tau_1, y)I(x + \tau_2, y)]$ and $\eta_y(z, \tau) = E[I(x, y)I(x, y + \tau_3)I(x, y + \tau_4)]$ denote the space varying third order feature function along x and y directions respectively. $z = (x, y)$, $\tau_1, \tau_2, \tau_3, \tau_4 > 0, \tau_x = \max(\tau_1, \tau_2)$, $\tau_y = \max(\tau_3, \tau_4)$ and $\tau = \arg_{\tau_x, \tau_y} \max(\eta(z, \tau))$. Then we can state the boundary detection problem as finding the pixels around which a function of $\eta(z, \tau)$ has a significant change. Assuming there is a stepwise jump in statistics of the signal and $s = s_0 + 1$; we consider

$$J(s) = E\left\{ [\eta(z, \tau) - \bar{\eta}(\tau) - \theta_s(z, \tau)]^2 \right\} \tag{5}$$

where

$$\bar{\eta}(\tau) = \frac{1}{(N - \tau)^2} \sum_{y=0}^{N-1-\tau} \sum_{x=0}^{N-1-\tau} \eta(z, \tau) \tag{6}$$

$\theta_s(z, \tau)$ is matched to the behaviour of $\eta(z, \tau)$. Expression (5) can be simplified by dropping terms that do not depend on *s*, and scaling $\theta_s(z, \tau)$ to be zero mean and to have unit energy for all values of *s* resulting in,

$$J_N^{(3)}(s) = \frac{1}{(N - \tau)^2} \sum_{y=0}^{N-1-\tau} \sum_{x=0}^{N-1-\tau} \eta(x, y, \tau)\theta_s(x, y, \tau) \tag{7}$$

where

$$\theta_s(z, \tau) = \begin{cases} -\left(\frac{N-s-\tau}{s}\right)^{1/2} & z = 0, \ldots, s-1 \\ \left(\frac{s}{N-s-\tau}\right)^{1/2} & z = s, \ldots, N-\tau \end{cases} \tag{8}$$

The same procedure is used to find the second order statistics using the same $\theta_s(z,\tau)$, called $J_N^{(2)}(s)$. It is important to note that in (8) z can be x or y depending on the direction of the maximum change in $J_N^{(2)}(s)$ and $J_N^{(3)}(s)$. We may easily see that around the edges $J_N^{(2)}(s)$ and $J_N^{(3)}(s)$ have lower values. In practice $N = N_w$ which is the size of the sliding window. The use of $J_N^{(2)}(s)$ and $J_N^{(3)}(s)$ relies on the estimate of $\eta(z;\tau)$. These estimates are of high variance typically, so it is necessary to smooth them. Adaptive smoothing has been used successfully to reduce image noise while preserving edge information. We adopt the smoothing technique proposed in [10]. Under this method, we first estimate the variance in two windows of either side of the point of interest and taking the mean over the window that exhibits low variance.

3.2. Edge Detection Using SVM

A two-class SVM with input vectors $J_N(s) = [J_N^{(2)}(s)$ $J_N^{(2)}(s)]$ is used to detect the changes in the texture image. We consider that there is no a priori information about the textures. So, prior to the application of SVM first a *K*-mean algorithm [11] is used to find the class centres and allocate a target value based on the $J_N(s)$ values in the entire image. The points far from the class centres, by a threshold value, are discarded so they will not be considered for training. Then, we have

$$y_c(s) = \begin{cases} 1 & \max(J_N(s)) > T_u \\ -1 & \max(J_N(s)) < T_l \end{cases} \tag{9}$$

where $y_c(s)$ are the initial target values. T_u and T_l are upper and lower threshold levels respectively. SVM is used here to classify those $J_N(s)$ for which $y_c(s)$ has a (target) value. Next the support vectors J_i^s are calculated for all non-zero α_i. To restore the edges, single points are removed by means of median filters and the boundary regions are thinned and linked. Unfortunately, there is usually a minor misplacement of the boundaries. Application of an intelligent post-processing technique may avoid the problem.

4. EXPERIMENTAL RESULTS

Some texture combinations from Brodatz collection [12] have been used. The size of the images is 256×256. Two of the images are given in Figure 1. Figure 2 illustrates the actual boundaries. An RBF kernel, with a variance equal to the variance of $J_N(s)$ is used. The best value for *C* may change with the size of the training and working set, $l + k$. We consider $1 \leq C = 1000/(l+k) < 4$. T_u and T_l are considered 60% and 20% of maximum of $J_N(s)$ respectively. Using these values false detection of edges is minimised. In Figure 2 the results of implementation of S^3VM are given. In this experiment the windows overlap each other by half a frame. Figure 3 shows the result of running the S^3VM algorithm over the images. The final edge-maps are illustrated in Figure 4. Misplacement (*MP*) of the boundaries is quantified using

$$MP = \frac{1}{N_w^2} \sum_i \min_j \left(|z_i^a - z_j^e| \right) \tag{10}$$

where $z_i^a = (x_i^a, y_i^a)$ and $z_i^a = (x_i^e, y_i^e)$ are the i^{th} points on the actual and estimated boundaries respectively. Table 1 shows the values for *MP* for the textures in Figure 1 without (Brian method [10]) and after using the S^3VM.

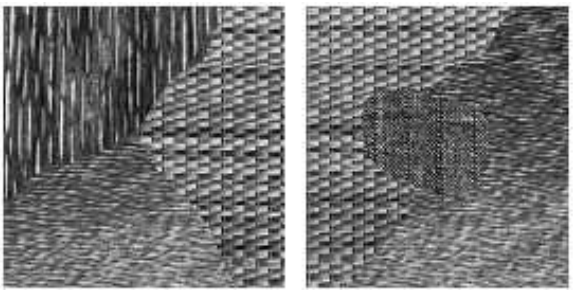

Fig. 1. Images of original four combinations of 256×256 textures.

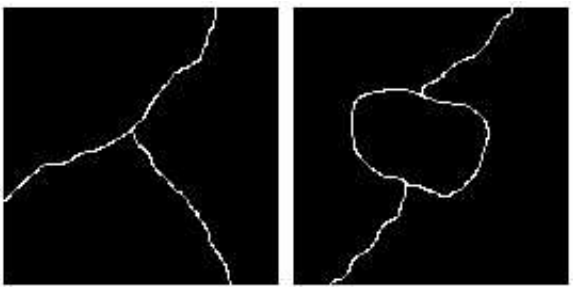

Fig. 2. The actual boundaries of the textures in Figure 1.

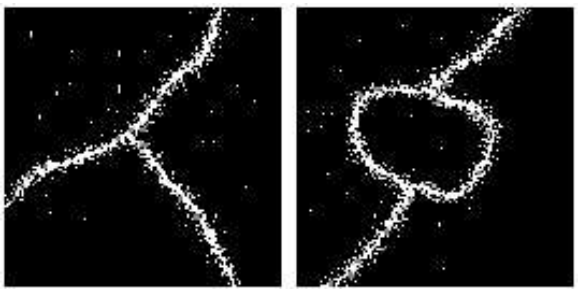

Fig. 3. The detected boundaries using S^3VM.

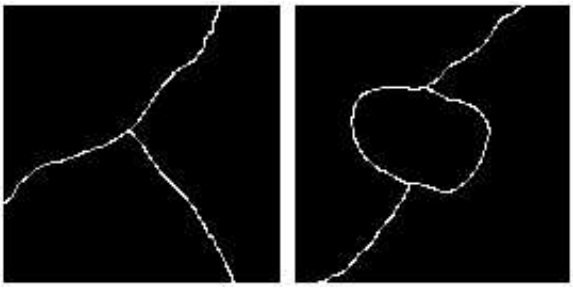

Fig. 4. The final edge map after post-processing.

Table 1. The boundary's misplacement error for the images in Figure 1.

MP	16×16 window size		32×32 window size	
	Without S^3VM	With S^3VM	Without S^3VM	With S^3VM
Image 1	2.4%	1.3%	2.3%	1.2%
Image 2	2.8%	1.4%	3.0%	1.7%

5. SUMMARY AND CONCLUSIONS

A novel algorithm for texture segmentation has been presented by developing a semi-supervised support vector machine. The points in the images are initially labelled as texture or boundary using *K*-mean algorithm. Then for a sliding window, the developed S^3VM is used to classify the regions of textures and boundaries. The features are the second and third-order statistics. The method is robust since it always allocates class labels to both the uniform and boundary regions. Moreover, it does not require any reference (target) vector during training and classification. In comparison with [10], this method performs better although still there may be slight displacement error.

6. REFERENCES

[1] Kwang In Kim, Keechul Jung, Se Hyun Park, and Hang Joon Kim, Support vector machines for texture classification, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, No. 11, pp. 1542-1550, 2002.

[2] Vapnik V., *The nature of statistical learning theory*, (Springer-Verlag, 1995).

[3] Scholkopf B. et al, Comparing support vector machines with Gaussian Kernels to radial basis function classifiers, *IEEE Trans. Signal Processing*, vol. 45, No. 11, pp. 2758-2765, 1997.

[4] Haykin S., *Neural network- A comprehensive fundation*, 2nd ed. (Prentice Hall, 1999).

[5] DeCoste D. and Scholkopf B., *Training invariant support vector machines, Machine Learning*, (Kluwer Press, 2001).

[6] Burges C.J.C., A tutorial on support vector machines for pattern recognition, *Data mining and knowledge discovery*, vol.2 No. 2, pp. 1-47, 1998.

[7] Fletcher, R., *Practical methods of optimisation*, 2nd ed., (John Wiley, 1987).

[8] Benneth K.P., Demirez A., Semi-supervised support vector machines, *Proc. Neural Information Processing*, Denver, USA, 1998.

[9] Tsatsanis M. K. and Giannakis G. B., a non-parametric approach for detecing changes in the autocorrelation structure, *Proc. 6th European Signal Processing Conference (EUSIPCO-92)*, II, 843-846, Belgium,1992.

[10] Sadler, B.M. texture segmentation by change detection in second and higher order statistics, *Conference Record of The 27th Asilomar Conference on Signals, Systems, and Computers*, 1, 436 -440, 1993.

[11] Jain A.K. and Karu K., Learning texture discrimination masks, *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 18, No. 2, pp. 195-205, 1996

[12] Brodatz P., *Texture: A photographic album for artists and designers*, (Dover, New York 1966).