COMPARISON OF SUPPORT VECTOR MACHINE AND NEURAL NETWORK FOR TEXT TEXTURE VERIFICATION

Datong Chen and Jean-Marc Odobez
IDIAP, Switzerland
chen, odobez@idiap.ch

Abstract.

In this paper we propose a method for classifying regions of images and videos frames into text and non-text regions using support vector machine (SVM). Different features are proposed to characterise the texture formed by text characters and background. SVM has an advantage that it is insensitive to the relative numbers of training examples in positive and negative classes. This advantage is are illustrated by comparing results with those obtained using a multiple layer perceptrons (MLP).

INTRODUCTION

Text texture verification aims at distinguishing image regions containing text characters from others that contain no text, so that text characters can be located and extracted in the development of advanced video and image annotation and retrieval systems. Text embedded in image and video usually provides brief and important information about the content, such as the name of a player or speaker, the title, location and date of an event, the category of the product etc. This kind of text is a powerful keyword resource in building image and video indexing and retrieval system. However, text characters contained in video are of low resolution, of any grayscale value and embedded in complex background. Experiments show that applying conventional OCR technology directly leads to poor recognition rate due to background noises. To improve recognition performance, an efficient algorithm for verifying text texture is a necessary pre-processing to fulfil the task of extracting text from images and videos.

One system for text extracting in covers of Journals or CDs is presented by Zhong [11], where text is located using spacial variance and connected component analysis. The approach assumed that text is in horizontal alignment and therefore had a high variance in horizontal direction. Text block was detected by finding regions with high horizontal variance and refined using the spatical information of text string in a connected component analysis process.

Smith et al. [6] developed a text detection algorithm by using vertical edge. In this method, vertical edges are first detected with a predefined template and then smoothed with a smoothing filter. The smoothing process removes isolated edge points yielded by the noise as well as connects vertical edges into text clusters. Another scheme of text extraction in images is proposed by Wu et al. [10], who describes a scheme for text location, segmentation and recognition based on texture segmentation. In this scheme, texture feature of each pixel is extracted in the derivatives of the image in Gaussian scale space and classified into three classes (text, non-text, other) using a K-means algorithm. In a more recent work, Garcia et al. [4] described a feature for detecting text pixel using the variance of edge orientation. These methods employ only texture information of single pixel, which can roughly detect some candidate regions but not able to gain a satisfied verification.

In 1999, Sobettka et al. [7] suggested that baseline detection could be used for text detection and location. In detail, if the top and bottom baselines can be detected in image or block of image, there is one text string between these two baselines. Otherwise, the image or block of image is regarded as background. Since baselines can also be detected in many background regions, plenty of false alarms are still produced using this method.

To exploit more texture information, Li et al. [5] presented a neural network based method to classify image block instead of single pixel. The method decomposes input image with Haar wavelet and employs a neural network to verify whether a block of pixels (fixed size) contains text or not. Because character scale is unknown, modelling text texture in wide varying scales requires very large training data. Another drawback is that the relative numbers of text regions and background regions are usually very different in real case and neural network tends to classify every thing into the class with majority number of training examples.

Addressing this problem, we have proposed a method for verifying text textures using SVM from roughly detected candidate text regions in [2]. In this paper, we compare SVM and a kind of typical neural network (MLP) for text texture verification in different feature spaces. We first extract different features for charactering texture of text of unknown grayscale values, which is highlighted in Section 2. The Section 3 describes the modelling of text texture using SVM. In the Section 4, we compare the performance of text verifiers of using SVM and MLP in different feature spaces.

FEATURE EXTRACTION

The candidate text regions are roughly detected using the method described in [2]. The method first clusters text regions using short vertical and horizontal edges and refines regions using baseline detection and connected component analysis. A yielded region is a rectangle whose size depends on the scale of the embedded text strings, if there is any. To avoid modelling text texture with different scales, candidate text regions are normalised into image regions of 16-pixel high, locking the height/width ratio (see examples in Figure 1).

Figure 1: Normalized candidate text regions.

Four kinds of features are extracted from the normalised candidate text regions will be extracted. They all share good properties with respect to the unknown value of the text grayscale. Features are extracted in a 16x16 window sliding from left to right inside each region.

Distance map

The first feature is the distance map, which is independent to the grayscale value of characters. The distance map [8] $DM(X)$ of a window X is defined by the set of all the associated distance values $v\,(x,y)$ in the window X with respect to a distance function (Equation 1).

$$\forall(x,y) \in X, DM(x) = \min_{(x_i,y_i)\in B} d[(x_i,y_i),(x,y)] \tag{1}$$

Here, B is a set of points, $B \in X$. We compute the point set B by extracting strong edges in window X. The distance function used here is Euclidean. The input feature vector x_i of SVM has 256 dimensions corresponding to a 16x16 slide window:

$$x_i = (f_1, f_2, \ldots, f_{256}) \tag{2}$$

Although distance map is independent to grayscale value of characters, the base set B still relies on the contrast between text and background and the threshold employed in edge detection.

Derivatives

In order to measure the contribute of the contrast of text to text verification, we introduce spatial derivatives of the image as the second feature. The feature is computed using Roberts 2x2 operator in two directions. Therefore, it has a feature vector of 512 dimensions in each sliding window instead of 256 dimensions.

Constant variance

In many text images, the contrast of text varies significantly in different backgrounds, which implies that contrast may be not a stable feature for text verification. We therefore compute the third feature, called constant variance (CV), to normalise the contrast by local image variance. The local image variance is a good measure of local contrast, where areas with low variance are essentially flat and have low contrast, while areas with high variance are often near edges and have high contrast. The local mean and variance are computed from the pixels in a neighbourhood of each point in the image. The contrast value of the point, subtracted by the local mean, is then normalised by dividing by the standard deviation. The size of the neighbourhood is 9 pixels in our system.

DCT coefficients

The last feature we used is coefficients of discrete cosine transform (DCT). This feature is widely used in JPEG and MPEG compression scheme and is a representative feature in frequency domain. We first reduce a image in 16x16 sliding window down to the 8x8 block, and compute DCT coefficients using a fast DCT algorithm presented by Feig [3] (generate a vector with 64 dimensions in each sliding window).

TEXT TEXTURE MODELLING USING SUPPORT VECTOR MACHINE

The SVM method

SVM is a technique motivated by statistical learning theory and has been successful applied to numerous classification tasks. The key idea is to separate two classes with a decision surface that has maximum margin. The extensive discussion of SVMs can be found in [9]. In this paper, we will only consider a binary classification task with m labelled training examples: $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, where $y_i = \pm 1$ indicating two different classes $i = 1, 2, \ldots, m$.

For linear separable case, we have some hyper-plane $w \cdot x + b = 0$ (decision surface) that separates all the training examples:

$$\begin{aligned} x_i \cdot w + b \geq +1 \; if \, y_i = +1 \\ x_i \cdot w + b \leq -1 \; if \, y_i = -1 \end{aligned} \tag{3}$$

or equivalently:

$$y_i (x_i \cdot w + b) \geq \pm 1 \; \forall i \tag{4}$$

where w is normal to the hyper-plane.

The margin of such hyper-plane $w \cdot x + b = 0$ is defined by the sum of the shortest distance from hyper-plane to the closest positive example and the shortest distance from hyper-plane to the closed negative example. Since this margin is simply $\frac{2}{\|w\|}$, where is the Euclidean norm of w, the maximum margin can be given by minimising $\|w\|^2$ subjecting to the constraints Eq. 4. This learning task can be reduced to maximisation of the Wolfe dual Lagrangian:

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{5}$$

with respect to the Lagrangian multipliers α_i subject to the constrains:

$$\begin{array}{rcl} \alpha_i & \geq & 0 \\ \sum_{i=1}^{m} \alpha_i y_i & = & 0 \end{array} \tag{6}$$

A training sample x_j is termed as support vector if the optimal $\alpha_j > 0$. This method can be easily generalised to non-linear case. Noticing that training of SVM only depends on examples through inner products, we can map the training examples $x_i \cdot x_j$ into an alternative space $\phi(x_i) \cdot \phi(x_j)$, so called feature space, by choosing kernel $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. The learning task therefore is the maximisation of the Lagrangian:

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i \cdot x_j) \tag{7}$$

subject to constraints Eq. 6. $W(\alpha)$ can be solved using quadratic programming techniques. Once we have found the optimal α_j, the classification of an unknown example is based on the sign of function:

$$G(z) = \sum_{j=1}^{m} \alpha_j y_j K(z, x_j) + b \tag{8}$$

For the non-separable case, the learning task involves the minimisation of

$$\|w\|^2 + C \sum_{i=1}^{m} \xi_i \tag{9}$$

subject to the constraints:

$$
\begin{array}{rcl}
y_i \left(x_i \cdot w + b \right) & \geq & 1 - \xi_i \; \forall i \\
\xi_i & \geq & 0 \; \forall i
\end{array} \tag{10}
$$

where C is the penalty to errors and ξ_i are positive slack variables that measure the amount of constraint violations.

Advantage of SVM with respect to other methods

One of advantages of SVMs is that the learning task is insensitive to the relative numbers of training examples in positive and negative classes. For example, in our case, the candidate text regions usually involve 15.4% false alarms (in terms of regions). The number of positive examples thereby is roughly 6 times as the negative examples. Most learning algorithm based on Empirical Risk Minimisation, for example MLP, will tend to classify only the positive class correctly to minimise the error over data set. Since SVM aims at minimising a bound on the generalisation error of a model in high dimensional space, so called Structural Risk Minimisation, rather than minimising error over data set, the training examples that far behind the hyper-planes will not change the support vectors. Therefore, SVM is used to verify text regions in the candidate text regions for achieving a lower false alarm rate.

Training of the SVM model

The aim of training a SVM is to find out a set of support vector to minimise Eq. 7. Here, we use typical Radial basis function (RBF) kernel:

$$
K(x, x_j) = e^{\frac{-||x - x_j||}{2\sigma^2}} \tag{11}
$$

where the kernel bandwidth σ is a parameter that close related with the distribution of sample vectors in original input space. During the training process, a wrong σ may lead to overfitting of the SVM, where the SVM gains a low error rate on the training data but a high error rate on testing data. In order to avoid overfitting, we perform K-fold cross-validation in training process. The K-fold cross-validation can be described like that:

1 Partition the training data set into K parts of equal size, refer to as "folds".

2 Assign each fold a possible value of σ.

3 For $i = 1$ to K do Train the SVM using all the folds except the ith as the training set and the ith σ as parameter. Evaluate the error of the output hypothesis using the ith fold as the test set.

4 Take the value of σ corresponding to the lowest error rate computed in (3) as the optimal parameter and train the SVM using all the training data to get a good support vector set.

The details of SVM training algorithm can be found in [9].

Text verification rating

During text verification, we extracted feature vectors by sliding 16x16 window every 4 pixels through feature images of normalised candidate text regions. Thus, for each candidate text region $\mathbf{r}$, we obtained a set of feature vectors $Z_r = (z_1^r, \ldots, z_l^r)$. Using support vector set S, we compute the magnitude of

$$G(z_i^r) = \sum_{x_j \in S} \alpha_j y_j K(z_i^r, x_j) + b \tag{12}$$

as the confidence of that vector z_i^r belongs to a text region. The confidence of the whole candidate text region r is defined as:

$$Conf(r) = \sum_{z_i^r \in Z_r} G(z_i^r) . \frac{1}{\sqrt{2\pi}\sigma_0} e^{\frac{d_x^2}{2\sigma_0^2}} \tag{13}$$

where, d_i is the distance from the centre of slide window i and the centre of the text region r. We experimentally let $\sigma_0 = 10$. A candidate text region r is verified as a text region if $Conf(r) \geq 0$.

EXPERIMENTS

Experiments were carried out on a database consisting 2,400 candidate text regions or 76,470 vectors for each of the four kinds of features, including both real text regions and false alarms, resulting from the detection step [2] in one hour video including advertisements, sports, interviews, news, movies and 50 compressed images including the covers of Journals, maps, fliers. Each video frame or image has 352x288 or 720x576 resolution in JPEG or MPEG format and been decompressed and converted into grayscale before applying text location and verification. Some video frames contain the same closed captions but with different backgrounds. The feature vectors are partitioned into two sets: training set and test set, with the same size. We assure that the training set of each feature contains the vectors extracted from the same images and locations.

In order to have an idea about the generalisation capability of SVM and to compare with an Empirical Risk Minimisation technique, we also train a multiple layer perceptron (MLP) for comparison. The MLP consists of an input layer, a hidden layer with 400 nodes and an output layer with two nodes indicating text and non-text. The activity function of neural is sigmoid.

The performance of verification listed in table 1 is measured in error rate of sample vectors. We can see that SVM shows better performances than MLP in any feature spaces due to its insensitive to the relative numbers of training examples in positive and negative classes. Using SVM, the constant variation feature illustrates a good performance in characterising texture of

Table 1: Error rate of SVM and MLP for text verification. DIS: distance mapping feature; DERI: derivative of image; CV: constant variation feature; DCT: DCT coefficients

Training Tools	DIS	DERI	CV	DCT
MLP	7.70%	6.00%	7.61%	5.77%
SVM	2.56%	3.99%	1.07%	2.92%

text of unknown grayscale by obtaining the best performance of 1.07% error rate, though it is not the best feature in MLP.

Using the confidence value computed by Eq. 13, we can remove all the false alarm regions from the 2,400 samples. Applying both text location and verification algorithms on our original database, Figure 2 shows some text verification results (text string has less than 2 characters is not regarded as a text string because it is difficult to be used as a keyword in indexing and retrieval task). The present method correctly located 98.7% text regions without producing any false alarms. The missing text regions are mostly yielded during the roughly detection step.

Figure 2: Verified text regions and false alarms in images or video frames.

CONCLUSION

In this paper, we proposed a SVM based method for verifying the texture of embedded text of any grayscale value in images and videos. Four kinds of feature: distance map, derivative of image, constant variation feature and DCT coefficients are presented and compared in this method. The experiments show that SVM is a better technique than MLP for addressing text texture verification problem and constant variation is the best feature in all the four proposed features in characterising text textures. As a pre-processing step, the presented verification method leads to a satisfied character and word recognition results in extracting text from images and videos. We apply text segmentation [1] and OCR on the verified text region and achieved a 92.5% character recognition rate and 91.7% word recognition rate.

ACKNOWLEDGMENT

The authors would like to thank Dr. Samy Bengio, Ronan Collobert, and Dr. Sebastien Marcel for their comments on this work.

REFERENCES

[1] D. Chen and J.-M. Odobez, "Text Recognition in Complex Background Based on Markov Random Field," in **ICPR**, 2002.

[2] D. Chen, K. Shearer and H. Bourlard, "Video OCR for Sport Video Annotation and Retrieval," in **Proc. of the 8th IEEE Int. Conf. on Mechatronics and Machine Vision in Practice**, Aug. 27 2001, pp. 57–62.

[3] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," **IEEE Trans. Signal Processing**, vol. 40, no. 28, pp. 2174–2193, Sept. 1992.

[4] C. Garcia and X. Apostolidis, "Text detection and segmentation in complex color images," in **ICASSP**, 2000, pp. 2326–2329.

[5] H. Li and D. Doermann, "Text enhancement in digital video using multiple frame integration," in **ACM Multimedia**, 1999, pp. 385–395.

[6] T. Sato, T. Kanade, E. K. Hughes and M. A. Smith, "Video OCR for digital news archives," in **IEEE Workshop on Content Based Access of Image and Video Databases**, Jan. 1998, Bombay.

[7] K. Sobottka, H. Bunke and H. Kronenberg, "Identification of text on colored book and journal covers," in **ICDAR**, 1999, pp. 57–63.

[8] J. Toriwaki and S. Yokoi, "Distance transformations and skeletons of digitized pictures with applications," **Pattern recognition**, pp. 187–264, 1981, L. N. Kanal and A. Rosenfeld editors.

[9] V. Vapnik, **Statistical Learning Theory**, John Wiley & Sons, 1998.

[10] V. Wu, R. Manmatha and E. M. Riseman, "Finding text in images," in **Proc. ACM Int. Conf. Digital Libraries**, 1997, pp. 23–26.

[11] Y. Zhong, K. Karu and A. K. Jain, "Locating text in complex color images," **Pattern Recognition**, vol. 10, no. 28, pp. 1523–1536, 1995.