RESEARCH ARTICLE
doi: 10.2306/scienceasia1513-1874.2016.42.046

ScienceAsia 42 (2016): **46–51**

Character classification framework based on support vector machine and *k*-nearest neighbour schemes

Teera Siriteerakul[a,*], **Veera Boonjing**[b], **Rutchanee Gullayanon**[a]

[a] Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520 Thailand
[b] International College, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520 Thailand

*Corresponding author, e-mail: kfteera@kmitl.ac.th

Received 9 Jan 2014
Accepted 7 Feb 2016

ABSTRACT: The problem of Thai character classification can be difficult because of the large number of characters and the similarity in the shape of many characters. While previous work combined different fonts to build their classifier, this paper proposes a framework based on support vector machine (SVM) and *k*-NN schemes to exploit characteristics of each font separately. In this framework, each font is used to train an SVM separately. With the trained SVMs, a vector of predicted values can be produced for any input image. Then a class label of the input image can be found by a *k*-NN based scheme. The proposed framework performs well with familiar fonts while providing an acceptable performance on unfamiliar fonts.

KEYWORDS: stacked generalization, optical character recognition, Thai characters

INTRODUCTION

Character classification is one of the vital parts of any optical character recognition (OCR) system. Apart from the obvious image to text application, an OCR system can be used to recognize license plates[1,2], detect aircraft tail numbers[3], or, coupled with a translation system, translate documents[4–6]. Thus since the early work of Tauschek[7], English character OCR systems are now well developed.

Thai character OCR, on the other hand, still needs considerable improvements. This is due to the large number of characters (Fig. 1) and the fact that some of the characters are very similar to each other. Also, in Thai words are usually not separated by space, and the vowels and tone marks can be placed either on top, below, or on either side of consonants. However, this work will deal only with the classification of printed Thai characters individually.

There have been several attempts at Thai character classification. Classification frameworks were reported based on artificial neural networks[8–10], fuzzy rough sets[11], as well as a hybrid approach based on fuzzy membership function and neural networks[12]. However, these studies train their classification tool(s) by grouping all fonts together which can cause a reduction in the separability of characters.

In our proposed framework, the characteristics of each font are captured individually by a separate support vector machine (SVM). Then, all the trained SVMs are used to classify all the character images in the training set, resulting in a vector of predicted values. To predict an unknown input, its image is classified by all the trained SVMs in order to obtain a vector of predicted values. The final prediction is the class of a character in the training set with the most similar vector of predicted values.

There is a notable reported work making use of multiple SVMs by Lin and Hauptmann[13]. In their work, the vectors of predicted values were used as input to another meta-classifier (another SVM). While their results were promising, it is not applicable to this work since the vectors of predicted values are a class label, and are not ordered. This observation is confirmed by preliminary experiments with less than 3% accuracy.

The key contribution of this paper is a proposed framework that exploits the nature of problem specific data to reduce the input dimension (and in turn computational requirements). This can be vital to computer vision research/application since the input data in this field can be large. Thus rather than grouping all fonts together to create a training set, this paper offers a framework to exploit each font individually to maintain their separability. Furthermore, this framework can be extended to another

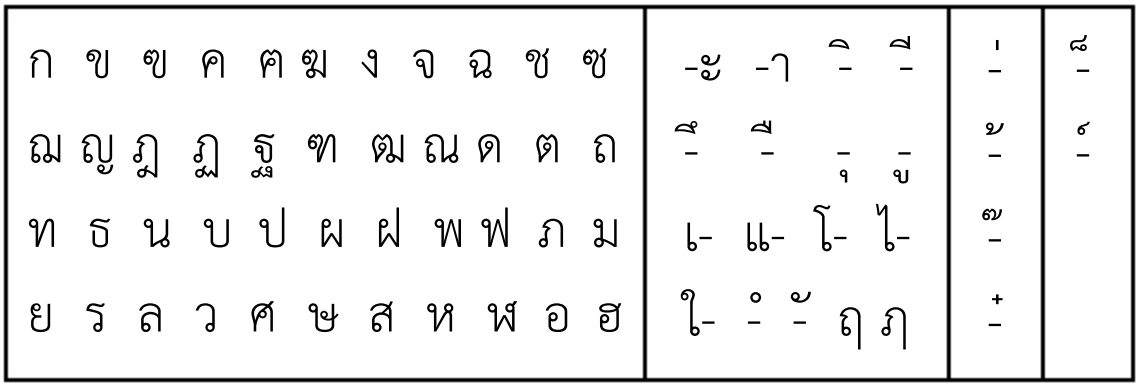

Fig. 1 Thai characters with 44 consonants (left box), 17 vowels (second box), 4 tones (third box), and 2 punctuation symbols (right box). Note that a hyphen (-) is used as place holder for a consonant.

classification task of a similar nature. It is important to note that, unlike stacked generalization[14] which applies multiple classifiers on the whole input, our framework explores the nature of the input data to achieve higher accuracy.

BACKGROUND INFORMATION

Thai characters

Like any other language, there are many Thai fonts. This poses a problem since one character from one font may look more similar to another character from a different font than the same character from that different font. Fig. 2 illustrates this situation. Thus grouping the same character from different fonts together would make it difficult for any kind of classification tool.

Support vector machines

A support vector machine[15] is a classification tool widely used in computer vision and pattern recognition. Originally, SVM was developed for separating two classes by finding an optimal hyperplane which maximizes the distance/margin between the two classes. In a simple case where the two classes are linearly separable, the hyperplane chosen by SVM is the one that maximizes the distance between the closest points (the support vectors) from both classes to the hyperplane. Thus given M training data points, $\{(xi, yi) \mid i \in \{1, \ldots, M\}, yi \in \{-1, 1\}\}$,

Fig. 2 Groups of similar Thai characters.

the linear SVM classifier can be defined as

$$y_i(\omega^{\mathrm{T}} x_i + \omega_0) \geqslant 1$$

where $\{x_i\}$ is the set of support vectors and the pair (ω, ω_0) defines the hyperplane of equation $\omega^{\mathrm{T}} x + \omega_0$ which can be determined by

$$\min \frac{\|\omega\|^2}{2}$$

subject to $y_i(\omega^{\mathrm{T}} x_i + \omega_0) \geqslant 1$.

However, in the general case where two classes are not linearly separable, a new constraint, $\epsilon_{\mathrm{j}} > 0$ is added. The objective function becomes

$$\min_{\omega, \omega_0, \xi_i} \frac{\|\omega\|^2}{2} + C \sum_i \xi_i$$

subject to $y_i(\omega^{\mathrm{T}} x_i + \omega_0) \geqslant 1 - \xi_i$ and $\xi_i \geqslant 0$ where C is a parameter to be set during training.

In order to use SVM for multi-class classification, we can take the 'one-against-one' approach[16]. Thus if k is the number of classes, then $k(k-1)/2$ classifiers are constructed by training with data from two classes. Then the final prediction is the class label with the maximum number of votes.

k-nearest neighbour method

The k-nearest neighbour (k-NN) method is a non-parametric method which classifies objects by examining the closest k training samples in a feature space. Then the class label of the input object is determined by the voting of the labels of the k nearest neighbours. Two choices of parameters for the k-NN method are k and the distance metric which are typically determined from the training samples.

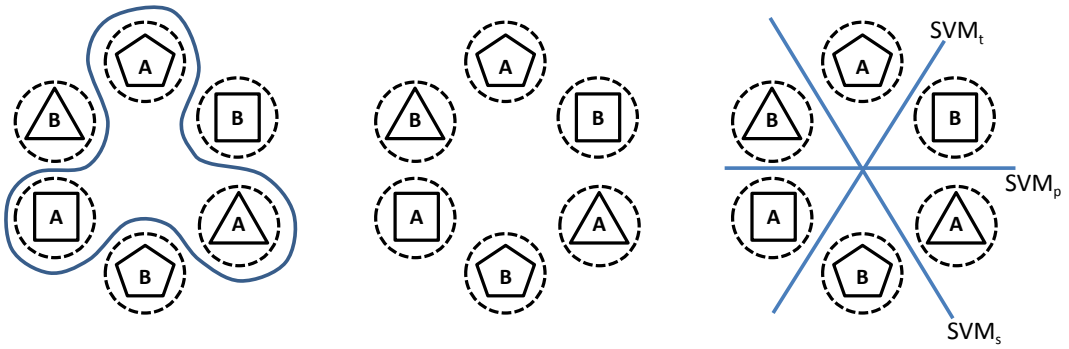

Fig. 3 The A and B are class labels (analogous to character labels) while triangle, square, and pentagon represent types of data (analogous to fonts). The circle around a class label of a data type represents a boundary in which that class label of that data type typically lies.

MULTI-MODAL CLASSIFIER

This section describes the proposed classification framework in detail. First, a motivating example is given to illustrate the reason for employing this framework. Then the section goes on to describe the framework in detail.

Motivation

Let us explore a simple example to motivate the idea behind the proposed system. Fig. 3-middle illustrates data consisting of 2 classes of 3 types (analogous to 2 types of character with 3 fonts). Obviously, if all the data are grouped together according to their class labels, the data would not be linearly separable. In this case, a nonlinear boundary must be manually selected via a kernel trick. (See Fig. 3-left for an example of a complex boundary.) However, the selected nonlinear boundary usually results in increases in computational time and the accuracy can suffer if the wrong kernel were picked. On the other hand, if 3 SVMs (one for each type of data) are employed, then 3 linear boundaries will be formed to separate the two classes of each type. Let us define them as SVM$_t$ for the triangle type, SVM$_s$ for the square type, and SVM$_p$ for the pentagon type (Fig. 3). These 3 SVMs can be used to predict a class label for any input. Thus Table 1 provides predictions for any input lying within the boundary of a certain class of a certain type.

As illustrated in Table 1, if a new input is classified and produces a predicted vector (A, B, A) from the 3 SVMs, one can infer from the lookup table that the new input should be of class B. Furthermore, one can even infer the type of square from the predicted vector. However, real data are typically far more complex and may not be as clear-cut as in this example. Thus rather than doing a table look up, the final predicted class will be determined by the nearest-neighbour process.

Table 1 Data prediction of the motivating example.

input class and type	predicted class label by		
	SVM$_t$	SVM$_s$	SVM$_p$
A Triangle	A	B	B
B Triangle	B	A	A
A Square	B	A	B
B Square	A	B	A
A Pentagon	B	B	A
B Pentagon	A	A	B

Note that in the general case, the three lines may not intersect at the same point. If so, the prediction of the extra area can be either (A, A, A) or (B, B, B) which provides a trivial prediction.

Classifier system

The overall system is separated into two phases: training and predicting. There are two processes within each phase: one that involves SVM and the other that involves the k-NN scheme.

For the training phase, let us assume that there are n individual character images which can be separated into m fonts. First, the training process starts by grouping individual character images from the training set according to their fonts. Then, images from each font are used to train an SVM, resulting in m trained SVMs: SVM$_1, \ldots,$ SVM$_m$. Then all the m SVMs will be used to classify all n images in the training set. Thus there will be n vectors of predicted values of size m, one for each character image. The final n vectors of size m will be used as a training set for the k-NN process.

Algorithm 1 Training process. Input: training dataset of n individual character images of m fonts. Output: (i) trained SVM for each font $(\text{SVM}_1, \ldots, \text{SVM}_m)$; (ii) vectors of predicted values $p^c = (p_1^c, \ldots, p_m^c)$ where $c = 1 \ldots, n$ (one for each character image c).

Step 1: for each font f train SVM_f using character images from the font f.

Step 2: for each character image c: for trained SVM_f set p_i^c = predicted result of SVM_f of character image c.

For the predicting phase, an input image of an unknown class has its class label predicted by all m SVMs to form a vector of predicted value of size m. The resulting vector of the predicted values will be used in the nearest neighbour search using

$$\text{similarity} = \sum_{i=0}^{m} \Phi(x_i, y_i),$$

where x_i and y_i are components of the vector of predicted values and

$$\Phi(x_i, y_i) = \begin{cases} 1, & \text{if } x_i = y_i, \\ 0, & \text{otherwise.} \end{cases}$$

In other words, the similarity of the two vectors of predicted values is computed by counting how many components they have in common. Note that this similarity is an inverse of distance. Thus in the nearest neighbour search, rather than finding a neighbour with the minimum distance, the algorithm will be searching for a neighbour with the maximum similarity. A summary of the predicting phase can be found in Algorithm 2.

In the case where there are multiple vectors with the same value of similarity, the algorithm will select the vector with the highest number of occurrences (maximum voting). If the highest number of occurrences is still the same, the algorithm will search for the vectors with the next highest similarity to break the tie.

Algorithm 2 Predicting process. Inputs: (i) a set of SVM models $(\text{SVM}_1, \ldots, \text{SVM}_m)$; (ii) vectors of predicted values p^c; (iii) input image i. Output: a predicted class L of the input image.

Step 1: for each SVM_f set v_f = predicted result of SVM_f of the input image i.

Step 2: select p^c which is the most similar to $(v_1, \ldots, v_m)$.

Step 3: look up the class label L of the training image c.

Step 4: return L as the predicted class label for the input image i.

SYSTEM SETUP AND RESULTS

Dataset

This paper uses a Thai character subset of datasets made available by NECTEC (www.nectec.or.th/corpus). There are two different datasets based on different ways of image acquisition. The first dataset is generated by printing and scanning characters using different fonts and sizes. This can be considered as a clean dataset. The first dataset is then further distinctly separated into a training set of 146 640 images and a validating set of 117 312 images. The second dataset, which will be used as a testing set, contains 62 724 images acquired from books, journals, magazines, and newspapers. See Fig. 4 for samples of a Thai character from both datasets.

For the experiment, intensity features are extracted from the input images. First, the images are padded (to preserve their aspect ratio) and resized to 16x16 pixels. Then grey-scale values of the resized image are rearranged to make one column vector by concatenating all rows together and then transposing. Thus the resulting input vector is a 256-dimensional column vector.

Since there are 12 fonts with 4 styles, there will be 48 SVMs, one for each font-style combination. Thus the vector of prediction is 48-dimensional.

System configuration

The following tests were done using a Windows 64 bits based PC with a 2.3 GHz Intel Core i7-3615QM CPU and 4 GB of RAM. The algorithms are implemented in C++ using the OpenCV library[17] with LibSVM[18].

Results

The first dataset can be considered as synthetic data since we can reproduce them with similar quality. The images from the second dataset can be considered a real world dataset since we have no control over them at all. In the experiment, the training subset of the first dataset was used for training. Then, the training set, the validating set

Fig. 4 Samples of a Thai character from the first dataset (left) and the second dataset (right).

Table 2 Comparison with SVM.

dataset	approach used for prediction		
	proposed	combined	voting
training	99.78	93.58	85.89
validating	97.13	92.24	85.13
testing	89.09	87.54	78.63

Table 3 Comparison with k-NN scheme.

dataset	approach used for prediction			
	proposed	1-NN	3-NN	5-NN
training	99.78	99.99	99.48	99.31
validating	97.13	97.73	97.58	97.51
testing	89.09	89.78	90.02	90.32

(also a subset of the first dataset), and the testing set (the second dataset) were used for evaluating the performance of the method.

To provide an empirical comparison, the proposed framework was evaluated against two other approaches: combined SVM and voting SVMs. In the combined SVM approach, image data from all fonts are combined according to their classes in order to be used to train and evaluate a single linear SVM. For the voting SVMs approach, a set of SVMs were trained in the same way as the proposed method except the final class label is determined by voting from all of the SVMs predictions.

The accuracies (Table 2) demonstrate that the proposed framework outperformed the other two approaches. For the training and the validating set, the input images were very similar to an image of the same character from the same font. This contributed to the high accuracy of the proposed framework. One interesting fact is that the accuracy of the training set was not 100% even though the nearest neighbour search should find the input image itself within the set of vectors of predicted values. This suggests that some other character from a different font can produce the same vector of predicted values. Thus there should be two characters from different fonts which are very similar to each other which may contribute to the lower accuracy of the combined approach. On the other hand, all approaches yielded relatively low accuracy when performed on the testing set. This can be the result of some unfamiliar fonts within the second dataset.

The low accuracy of the voting scheme can be explained by the 'closeness' of the majority fonts. These majority fonts will outvote the outliner fonts even on the characters from those outliner ones which result in incorrect prediction. On the other hand, the combined scheme constructs a bigger boundary for each character when grouping the same character from the different fonts together. When two different characters from two different fonts are close (in feature space), these bigger boundaries would overlap. This overlapping can cause poor separability of SVM.

We also performed another set of tests to compare our method with the k-NN method (with $k =$ 1, 3, and 5). As shown in Table 3, the accuracy of our framework was comparable to that of k-NN. This is not surprising since our method approaches the k-NN method as the number of SVMs is raised. However, the prediction time of k-NN is $O(kd\log(n))$[19], where d is the number of dimensions (which is the number of features in this case), and for our proposed method it is $O(d)$ for the SVM prediction and $O(t\log(n))$ for the k-NN part of our framework (where t is the number of SVMs used). Furthermore, the memory required for the k-NN and our method is $O(\mathrm{d}n)$ and $O(\mathrm{d}t + tn)$, respectively. Thus our method requires lower CPU time and memory than k-NN. This can help in computer vision applications since the input data in this field can be very large. For example, to keep all 146 640 images of 16x16 pixels in the training set would require 38 MB while keeping only the prediction vectors as in our proposed method would require only 3.4 MB.

CONCLUSIONS

This paper provides a framework for a character classification task in a multiple fonts environment. Rather than employing several classifiers onto all the inputs, the proposed framework exploits the nature of the input data to improve the classification accuracy. In this frameworks, a set of SVMs are used to capture the variety of fonts while a k-NN based process is employed to approximate the location of the input image in the feature space. The proposed framework performed well on familiar fonts while maintaining acceptable performance on a real world dataset. Furthermore, our proposed method can help reduce memory and CPU time requirements.

Although this paper demonstrated the capability of the framework on Thai character classification, the framework would also be applicable to any classification work where each class is composed of several known subclasses.

REFERENCES

1. Kulkarni P, Khandebharad A, Khope D, Chavan P (2012) License plate recognition: a review. In: *IEEE 4th International Conference on Advanced Computing*, Chenni, pp 1–8.
2. Du S, Ibrahim M, Shehata M, Badawy W (2013) Automatic license plate recognition (ALPR): a state-of-the-art review. *IEEE Trans Circ Syst Video Tech* **23**, 311–25.
3. Berlanga A, Besada J, Herrero J, Molina J, Portillo J, Casar J (2004) Optimizing statistical character recognition using evolutionary strategies to recognize aircraft tail numbers. *EURASIP J Appl Signal Process* **8**, 1125–34.
4. Du J, Hou Q, Sun L, Sun J (2011) Snap and translate using Windows phone. In: *International Conference on Document Analysis and Recognition*, pp 809–13.
5. Fragoso V, Gauglitz S, Zamora S, Kleban J, Turk M (2011) TranslatAR: A mobile augmented reality translator. In: *IEEE Workshop on Applications of Computer Vision* pp 497–502.
6. Nakajima H, Matsuo Y, Nagata M, Saito K (2005) Portable translator capable of recognizing characters on signboard and menu captured by its built-in camera. In: *Proceedings of the Annual Meeting of the Association of Computer Linguistics*, pp 61–4.
7. Tauschek G (1933) *Statistical Machine*. US patent 1 915 993.
8. Tanprasert C, Koanantakool T (1996) Thai OCR: a neural network application. In: *IEEE TENCON Digital Signal Processing Applications* vol 1, pp 90–5.
9. Kijsirikul B, Sinthupinyo S, Supanwansa A (1998) Thai printed character recognition by combining inductive logic programming with backpropagation neural network. In: *IEEE Asia-Pacific Conference on Circuits and Systems*, pp 539–42.
10. Thammano A, Duangphasuk P (2005) Printed Thai character recognition using the hierarchical cross-correlation artmap. In: *IEEE International Conference on Tools with Artificial Intelligence*, pp 695–8.
11. Kasemsiri W, Kimpan C (2001) Printed Thai character recognition using fuzzy-rough sets. In: *IEEE Region 10 International Conference on Electrical and Electronic Technology*, vol 1, 326–30.
12. Thammano A, Ruxpakawong P (2002) Printed Thai character recognition using the hybrid approach. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* vol E85-A, pp 1236–41.
13. Lin W, Hauptmann A (2002) News video classification using SVM-based multimodal classifiers and combination strategies. In: *Proceedings of the 10th ACM International Conference on Multimedia*, pp 323–6.
14. Wolpert D (1992) Stacked generalization. *Neural Network* **5**, 241–59.
15. Cortes C, Vapnik V (1995) Support-vector networks. *Mach Learn* **20**, 273–97.
16. Knerr S, Personnaz L, Dreyfus G (1990) Single-layer learning revisited: a stepwise procedure for building and training a neural network. In: Fogelman J (ed) *Neurocomputing: Algorithms, Architectures and Applications, Neurocomputing*, pp 41–50.
17. Bradski G (2000) *The OpenCV Library: Dr. Dobb's Journal of Software Tools*.
18. Chang C, Lin C (2011) LIBSVM: A library for support vector machines. *ACM Trans Intell Syst Tech* **2**, 27.
19. Arya S, Mount D, Netanyahu N, Silverman R, Wu A (1998) An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J ACM* **45**, 891–923.