Segmentation of On-Line Handwritten Japanese Text Using SVM for Improving Text Recognition

Bilan Zhu, Junko Tokuno, and Masaki Nakagawa

Tokyo University of Agriculture and Technology, Naka-cho 2-24-16,
Koganei, Tokyo 184-8588, Japan
{zhubilan, j-tokuno}@hands.ei.tuat.ac.jp,
nakagawa@cc.tuat.ac.jp
http://www.tuat.ac.jp/~nakagawa/

Abstract. This paper describes a method of producing segmentation point candidates for on-line handwritten Japanese text by a support vector machine (SVM) to improve text recognition. This method extracts multi-dimensional features from on-line strokes of handwritten text and applies the SVM to the extracted features to produces segmentation point candidates. We incorporate the method into the segmentation by recognition scheme based on a stochastic model which evaluates the likelihood composed of character pattern structure, character segmentation, character recognition and context to finally determine segmentation points and recognize handwritten Japanese text. This paper also shows the details of generating segmentation point candidates in order to achieve high discrimination rate by finding the combination of the segmentation threshold and the concatenation threshold. We compare the method for segmentation by the SVM with that by a neural network using the database *HANDS-Kondate_t_bf-2001-11* and show the result that the method by the SVM bring about a better segmentation rate and character recognition rate.

1 Introduction

On-line recognition was first employed in real products in 1980s for Japanese input with hard constraints such as character writing boxes. Due to the development of pen-based systems such tablet PC, electronic whiteboard, PDA, Anoto pen, e-pen and so on and the expansion of writing surfaces, handwritten text recognition rather than character recognition is being sought with less constraints since larger writing surfaces allow people write more freely.

The model and system for separating freely written text into text line and estimating the line direction and character orientation was reported in [1]. If the initial segmentation is not good, however, it determines the upper limit of text recognition performance.

Aizawa et al. reported real-time segmentation for on-line handwritten Japanese text by applying features preceding a segmentation point candidate to a neural network in [2]. Okamoto et al. showed that several physical features are effective to segment

H. Bunke and A.L. Spitz (Eds.): DAS 2006, LNCS 3872, pp. 208 – 219, 2006.

on-line handwritten Japanese text deterministically [3]. We previously proposed a segmentation method for on-line handwritten Japanese text by a neural network [4].

The SVM method [5], [6] for pattern recognition is recently being paid more and more attentions. It is a technique motivated by statistical learning theory and has been developed to construct a function for nonlinear discrimination by the kernel method. SVMs have been successful applied to numerous classification tasks. It is thought that SVMs are the learning models that provides with the best recognition performance in a lot of techniques known now. The key idea of SVMs is to learn the parameters of the hyperplane to classify two classes based on maximum margin from training patterns.

In this paper, we employ a SVM to determine segmentation point candidates for on-line handwritten Japanese text of horizontal writing from left to right. We compare the method for segmentation by the SVM with that by a neural network. We incorporate the method into the segmentation by recognition scheme. We follow the stochastic model proposed in [7] to evaluate the likelihood composed of character pattern structure, character segmentation, character recognition and context and finally determine segmentation points and recognize text.

In this paper, section 2 presents the flow of processing. Section 3 describes text segmentation and a method for generating character segmentation point candidates. Section 4 presents evaluation. Section 5 concludes this paper.

2 Flow of Processing

A stroke denotes a sequence of pen-tip coordinates from pen-down to pen-up while an off-stroke denotes a vector from the pen-up to the next pen-down. Japanese text is composed of several text lines separated by a large off-stroke from a previous line to a new line. Its detection is not difficult. We don't go into this matter in this paper.

We process each text line as follows:

Step1: Generation of segmentation point candidates

Each off-stroke is classified into segmentation point, non-segmentation point and undecided point according to the features such as distance and overlap between adjacent strokes detailed later. A segmentation point should be between two characters while a non-segmentation point is within a character pattern. An undecided point is a point where segmentation or non-segmentation judgment cannot be made. A segmentation unit bounded by two adjacent segmentation points is assumed as a character pattern. An undecided point is treated as two ways of a segmentation point or a non-segmentation point. When it is treated as a segmentation point, it is used to extract a segmentation unit.

Step2: Modification of segmentation point candidates

For text written aslant rather than horizontally or vertically, segmentation point candidates made by the step 1 are modified using the skew space feature defined in [4].

Step3: Segmentation and recognition

A candidate lattice is constructed where each arc denotes segmentation point and each node denotes a character recognition candidate produced by character recognition for each segmentation unit as shown in *Fig. 1*. Scores are associated to each arc or node following the stochastic model evaluating the likelihood composed of

character pattern structure, character segmentation, character recognition and context. The Viterbi search is made into the candidate lattice for a handwritten text line and the best segmentation and recognition is determined.

This paper will describe the details of the step 1. For the step 2 and step 3, refer to the literature [4], [7].

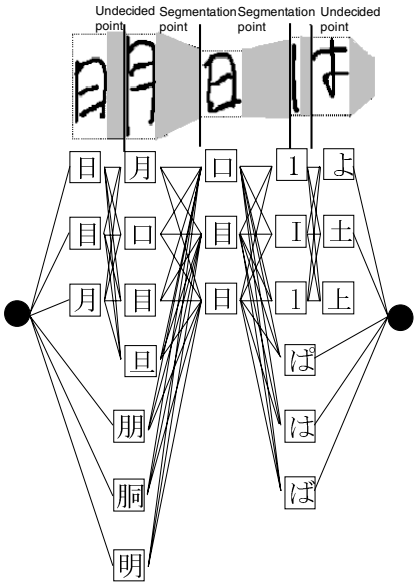


Fig. 1. Candidate lattice

3 Segmentation

First, we extract multi-dimensional features from off-strokes within a text line. Then, each off-stroke is classified into segmentation point, non-segmentation point and undecided point by applying a SVM or a neural network for the extracted features.

3.1 Selection of Off-Stroke Features

First, we define the following terminology:

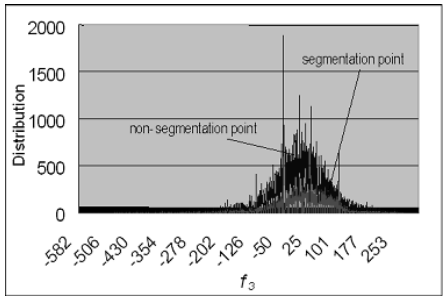
Bb_{p1}: Bounding box of the immediately preceding stroke
Bb_{s1}: Bounding box of the immediately succeeding stroke
Bb_{p_all}: Bounding box of preceding all the strokes
Bb_{s_all}: Bounding box of succeeding all the strokes
acs: Average character size
D_{Bx}: Distance between Bb_{p_all} and Bb_{s_all} to x-axis
  D_{Bx} = X coordinate of left position of Bb_{s_all} - X coordinate of right position of Bb_{p_all}
D_{By}: Distance between Bb_{p_all} and Bb_{s_all} to y-axis
  D_{By} = Y coordinate of top position of Bb_{s_all} - Y coordinate of bottom position of Bb_{p_all}
D_{bx}: Distance between Bb_{p1} and Bb_{s1} to x-axis
  D_{bx} = X coordinate of left position of Bb_{s1} - X coordinate of right position of Bb_{p1}
D_{by}: Distance between Bb_{p1} and Bb_{s1} to y-axis
  D_{by} = Y coordinate of top position of Bb_{s1} - Y coordinate of bottom position of Bb_{p1}

O_b: Overlap area between Bb_{p1} and Bb_{s1}
D_{bsx}: Distance between centers of Bb_{p1} and Bb_{s1} to x-axis
D_{bsx} = X coordinate of center of Bb_{s1} – X coordinate of center of Bb_{p1}
D_{bsy}: Distance between centers of Bb_{p1} and Bb_{s1} to y-axis
D_{bsy} = Y coordinate of center of Bb_{s1} – Y coordinate of center of Bb_{p1}
D_{bs}: Absolute distance of centers of Bb_{p1} and Bb_{s1}
Df_b: Difference between Bb_{p_all} and Bb_{s1}
Df_b = abs(Y coordinate of top position of Bb_{p_all} - Y coordinate of top position of Bb_{s1})

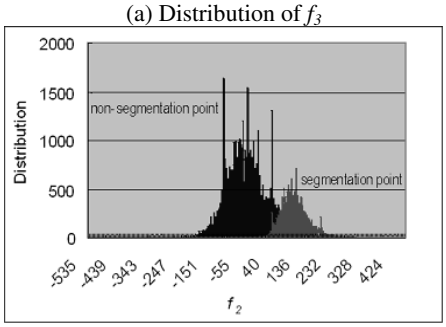
The average character size *acs* is estimated by measuring the length of the longer side of the bounding box for each stroke, sorting the lengths from all the strokes and taking the average of the larger 1/3 of them.

Then, the following 21 features of off-strokes are extracted for segmentation:

f_1: Passing time for the off stroke
f_2: D_{Bx} / *acs*
f_3: D_{By} / *acs*
f_4: Overlap area between Bb_{p_all} and Bb_{s_all} / $(acs)^2$
f_5: D_{bx} / width of Bb_{p1}
f_6: D_{bx} / width of Bb_{s1}
f_7: D_{bx} / *acs*
f_8: D_{by} / height of Bb_{p1}
f_9: D_{by} / height of Bb_{s1}
f_{10}: D_{by} / *acs*
f_{11}: O_b / (width x height of Bb_{p1})
f_{12}: O_b / (width x height of Bb_{s1})
f_{13}: O_b / $(acs)^2$
f_{14}: D_{bsx} / *acs*
f_{15}: D_{bsy} / *acs*
f_{16}: D_{bs} / *acs*
f_{17}: Df_b / *acs*



(a) Distribution of f_3



(b) Distribution of f_2

Fig. 2. Distributions of f_2 and f_3 features for training patterns

f_{18}: Length of off-stroke / *acs*
f_{19}: Sine value of off-stroke
f_{20}: Cosine value of off-stroke
f_{21}: f_2 / the maxinum f_2 in text

We examined the distributions of these features using training patterns, and deleted the features such as f_3 shown in *fig. 2(a)* that two classes of segmentation points and non-segmentation points are not clearly divided while retained those such as f_2 shown in *fig. 2(b)* that the two classes are divided to some extent. Moreover, some features have very similar effect. Employment of them at the same time doesn't affect the discrimination rate although it takes processing time. Therefore, we examined the correlation coefficient for each pair of features and selected either one from the pair that has 0.90 or more correlation coefficient. The finally selected features are shown in *table 1*.

Table 1. Selected features

Selected features	Number
f_1, f_2, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{12}, f_{13}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19}, f_{20} f_{21}	18

3.2 Neural Network

A three-layers neural network can be used for distinguishing two classes of segmentation points and non-segmentation points [8]. We constructed a neural network that has an input layer composed of a feature vector $\boldsymbol{v}$ from an off-stroke plus one additional input, a middle layer of n_{mu} units and the single output. The output O is calculated as follows:

$$O = \sum_{\alpha=1}^{n_{mu}} c_\alpha \, \sigma(\mathbf{w}_\alpha \cdot \mathbf{v} + b_\alpha).$$
$$\sigma(u) = \frac{1}{1+\exp(-u)}. \tag{1}$$

We set the target value of segmentation points as 1 and that of non-segmentation points as 0, and obtain the network coefficients of $\mathbf{w}_\alpha$, b_α, c_α by training the neural network using backpropagation for training patterns collected. The network coefficients are initialized with random values, and then they are changed to the direction that will reduce the learning error as follows:

$$\Delta\boldsymbol{\theta} = -\eta \frac{\partial \mathrm{J}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{2}$$

where $\boldsymbol{\theta}$ represents all the network coefficients, η is the learning rate, $\mathrm{J}(\boldsymbol{\theta})$ is the learning error, and $\Delta\boldsymbol{\theta}$ indicates the relative size of change in the network coefficients. $\boldsymbol{\theta}$ is updated at iteration t as:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \Delta\boldsymbol{\theta}(t). \tag{3}$$

Moreover, we use learning with momentum for speedup as follows:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + (1-\beta)\Delta\boldsymbol{\theta}(t) + \beta\Delta\boldsymbol{\theta}(t-1). \tag{4}$$

where β is set as 0.9.

For the learning rate η, we initialize it as a large value, and update it at each iteration t as follows:

$$\begin{aligned} &\text{if(J(t) - J(t -1) >= 0 \& \& It occurs } n_1 \text{times continuous ly)}\ \eta = \eta - \gamma_1 \eta \ . \\ &\text{if(J(t) - J(t -1) < 0 \& \& It occurs } n_2 \text{ times continuous ly)}\ \eta = \eta + \gamma_2 \eta \ . \end{aligned} \tag{5}$$

where n_1 is set as 3, n_2 is set as 2, γ_1 is set as 0.5, γ_2 is set as 0.1. The learning speed can be remarkably improved by the above method.

For the number of units for the middle layer n_{mu}, we will test several numbers and select the number that makes the smallest learning error.

3.3 Support Vector Machine

The key idea of SVMs is to separate two classes with the hyperplane that has maximum margin. Finding this hyperplane $\boldsymbol{\omega}.\mathbf{x}_i + b = 0$ can be translated into the following optimization problem:

$$\begin{cases} \text{minimize}: \dfrac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{l}\xi_i \ . \\ \text{subject to}: \xi_i \geq 0, y_i(\boldsymbol{\omega}.\mathbf{x}_i + b) \geq 1 - \xi_i \ . \end{cases} \tag{6}$$

where $\dfrac{1}{2}\|\boldsymbol{\omega}\|^2$ is for the maximum margin, ξ_i is the learning error of a training pattern i, C is the trade-off between learning error and margin, $\mathbf{x}_i$ is the feature vector of a training pattern i, y_i is the target value of a training pattern i , l is the number of training patterns, respectively.

Then, the feature vectors are mapped into an alternative space by choosing kernel $\mathrm{K}(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i).\varphi(\mathbf{x}_j)$ for nonlinear discrimination. Consequently, it leads to the following quadratic optimization problem:

$$\begin{cases} \text{minimize}: \mathrm{W}(\boldsymbol{\alpha}) = \sum_{i=1}^{l}\alpha_i + \dfrac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \ \mathrm{K}(\mathbf{x}_i \mathbf{x}_j) \ . \\ \text{subject to}: \sum_{i=1}^{l} y_i \alpha_i = 0, \forall i : 0 \leq \alpha_i \leq C \ . \end{cases} \tag{7}$$

where, $\boldsymbol{\alpha}$ is a vector of l variables and each component α_i corresponds to a training pattern ($\boldsymbol{x}_i$, y_i). The solution of the optimization problem is the vector $\boldsymbol{\alpha}^*$ for which $\mathrm{W}(\boldsymbol{\alpha})$ is minimized and the constraints of the *eq. (7)* are fulfilled. The classification of an unknown pattern $\mathbf{z}$ is made based on the sign of the function:

$$\mathrm{G}(\mathbf{z}) = \sum_{i:SV} \alpha_i y_i \ \mathrm{K}(\mathbf{x}_i, \mathbf{z}) + b \ . \tag{8}$$

We set the target value of segmentation points as 1 and that of non-segmentation points as -1. We obtain the separating hyperplane by solving this optimization

problem shown in the *eq. (7)* for training patterns using SVM^{light} [9] that can efficiently handle problems with many thousand support vectors, converges fast with minimal memory requirements.

3.4 Generation of Segmentation Point Candidates

Now, we must consider how to judge segmentation, non-segmentation and undecided points for generating segmentation point candidates.

We could set 0.5 as the threshold *th* because the target value of segmentation points is 1 and that of non-segmentation points is 0, then judge the values of the outputs based on the *eq. (1)* larger than *th* as segmentation points and the others as non-segmentation points for the classification by the neural network. For the classification by the SVM, we could set 0 as the threshold *th* because the target value of segmentation points is 1 and that of non-segmentation points is -1, then judge the values of the outputs based on the *eq. (8)* larger than *th* as segmentation points and the others as non-segmentation points. We could do so if it were only a classification of two classes for segmentation points and non-segmentation points. However, this does not allow the later processing to apply likelihood factors such as character recognition or context to better segment handwritten text.

Fig. 3 shows the distribution of the outputs of the neural network trained for the training patterns. We can set the concatenation threshold th_c and the segmentation threshold th_s for the both sides of *th* and judge values smaller than th_c as concatenation (non-segmentation) points, values larger than th_s as segmentation points, and the others as undecided points to obtain the higher segmentation rate for the step 3 in *Section 2*. The widths $th - th_c$ and $th_s - th$ are not certainly equal, because the distribution of the outputs for two classes of non-segmentation points and segmentation points are unbalanced as shown in *Fig. 3*. Therefore, we take the segmentation measure (the *f* measure according to the *eq. (9)* where r is recall, p is precision) after applying the step 3 for all the combinations of th_c and th_s using the

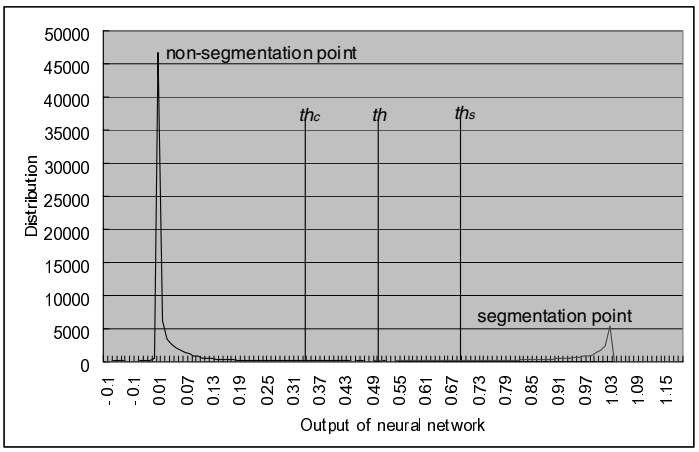


Fig. 3. Distribution of the outputs of the neural network trained for training patterns

training patterns and take the combination of th_c and th_s producing the best segmentation measure f. We employ th_c and th_s determined from the training patterns for the testing patterns, because the distribution of the testing patterns is approximated by that of the training patterns.

$$f = \frac{2}{1/r + 1/p}.$$
$$r = \frac{number\ of\ correctly\ detected\ segmentation\ points}{number\ of\ true\ segmentation\ points}. \tag{9}$$
$$p = \frac{number\ of\ correctly\ detected\ segmentation\ points}{number\ of\ detected\ segmentation\ points\ (including\ false)}.$$

4 Experiments

We extracted text lines of horizontal writing from left to right from the database of character-orientation and line-direction free on-line handwritten Japanese text: *HANDS-Kondate_t_bf-2001-11* collected from 100 people. We took 20 people's patterns as training patterns while 5 people's patterns as test patterns. We used a part of the database since it takes longer time for learning as patterns increase. Their details are shown in *table 2*, where N_{sp}, N_{nsp}, N_{ac} and N_{al} denote the number of true segmentation points, the number of true non-segmentation points, the average number of characters in a text line, the average number of characters written by one people, respectively. We use them to compare the methods for segmentation by the SVM with that by the neural network.

Table 2. Sample patterns

Number / Patterns	Text lines	N_{sp}	N_{nsp}	N_{ac}	N_{al}	English letters	Numbers	Karas	Chinese characters	Other characters
Training	2772	27062	79301	11	1193	1231	4647	10715	9949	3292
Testing	695	6887	20196	11	1516	307	1139	2733	2613	790

4.1 Setting Parameters

We tested several neural networks which have the number of units for the middle layer n_{mu} as 2, 4, 6, 8 and 10, and trained the parameters for these neural networks using the training patterns until getting the smallest learning error. We selected the neural network with n_{mu} 4 because it made the smallest learning error.

For the SVM, we used the following radial basis function kernel:

$$\mathrm{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right). \tag{10}$$

We set σ as 0.4 and C shown in the *eq. (7)* as 100 by testing several values in experiments using the training patterns. Then, we obtained the parameters of the separating hyperplane for the SVM using the training patterns again.

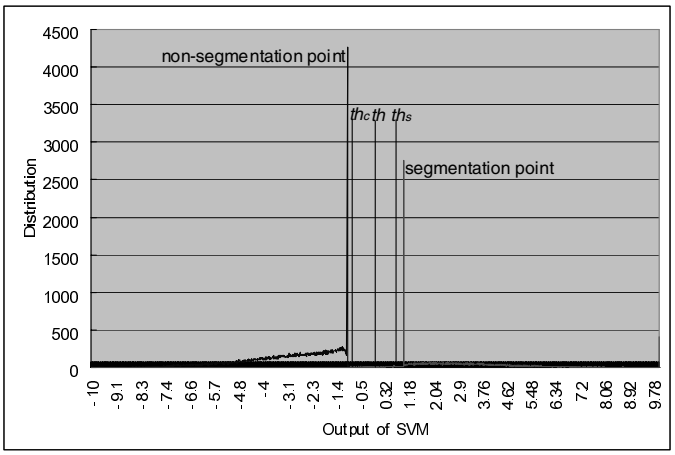


Fig. 4. Distribution of the outputs of the SVM for training patterns

Moreover, we took the distributions of the outputs of the neural network and the SVM for the training patterns. The results are shown in *Fig. 3* and *Fig. 4*. We can see the distribution of the outputs of the SVM is small from –1 to 1, because the training patterns having the outputs from –1 to 1 are regarded as having training errors and the SVM has been trained to have the smallest sum of the training errors.

Then, we measured the f measures according to the *eq. (9)* after applying the step 3 using the training patterns for all the combinations of th_c and th_s within every 0.01 step from 0.0 to 0.5 for th_c and from 0.5 to 1.05 for th_s for the neural network, and within every 0.02 step from –1.1 to 0 for th_c and from 0 to 1.1 for th_s for the SVM, respectively. We took the combination of th_c and th_c producing the best segmentation measure f. According to the result on the training patterns, we set the parameters th_c and th_s as 0.08 and 1.0 for the neural network, -0.98 and 0.98 for the SVM, respectively. The details of the result for the training patterns according to these parameters are shown in *table 3*.

Table 3. The result of segmentation for the training patterns

Off-strokes \ Method		Neural Network	SVM
True non-segmentation points	Classified into non-segmentation points	83.24%	91.99%
	Classified into undecided points	16.69%	7.74%
	Classified into segmentation points	0.07%	0.26%
True segmentation points	Classified into non-segmentation points	0.76%	0.58%
	Classified into undecided points	64.18%	6.75%
	Classified into segmentation points	35.05%	92.67%

Table 4. How often off-strokes are classified into thress classes

Off-strokes \ Method	Neural Network	SVM
Classified into non-segmentation points	62.25%	68.74%
Classified into undecided points	28.78%	7.49%
Classified into segmentation points	8.97%	23.78%

Table 4 summarises the result from the different viewpoint. It shows how often off-strokes are classified into non-segmentation points, undecided points and segmentation points, because undecided points incur processing time significantly.

4.2 Comparison of Neural Network and SVM

We compare the performance by the SVM and that by the neural network for the training patterns and the testing patterns on a Pentium (R) 4 3.40 GHz CPU with 0.99 GB memory. *Table 5* shows the result, where f, Cr, T_{train}, T_{ac}, T_{ar} denote the f measure after applying the step 3, the character recognition rate after applying the step 3, the time for training the parameters for the neural network or the SVM using the training patterns, the average time for classifying an off-stroke into the three classes, the average time for processing a text line by the three steps mentioned in *Section 2*, respectively.

Table 5. Comparison of the two methods

Performance \ Method		Neural Network	SVM
Training patterns	f	0.9600	0.9859
	Cr	72.22%	77.60%
Testing patterns	f	0.9413	0.9578
	Cr	69.76%	72.92%
T_{train}		About 1.5 hours	About 10 hours
T_{ac}		0.009 (ms)	5.845 (ms)
T_{ar}		92.15 (ms)	279.07 (ms)

We also measured the segmentation measure f for classifying only two classes of segmentation points and non-segmentation points by the neural network and the SVM. The result is that the f measure is 0.9045 for the training patterns and 0.8886 for the testing patterns by the neural network, 0.9733 for the training patterns and 0.9268 for the testing patterns by the SVM, respectively.

The *eq. (11)* shows a formula of the average time for processing a text line from components. The terms N_{as}, N_{udp} denote the average number of off-strokes in a text line, the average number of undecided points in a text line, respectively. The terms T_{Se}, T_{Cr}, T_{Lcs}, T_{Las}, are the average time for extracting the features from an off-stroke, the average time of character recognition for a text line, the average time for constructing the candidate lattice for a text line, the average time to search into the candidate lattice for a text line, respectively. The latter three terms depend on how many consecutive undecided points appears, but approximately they have the order of two to the power of N_{udp}.

$$\begin{aligned} T_{ar} &= N_{as}T_{Se} + N_{as}T_{ac} + T_{Cr} + T_{Lac} + T_{Las} \\ T_{Cr} &= \mathrm{O}(2^{N_{udp}}) \\ T_{Lac} &= \mathrm{O}(2^{N_{udp}}) \\ T_{Las} &= \mathrm{O}(2^{N_{udp}}) \end{aligned} \tag{11}$$

From *table 4*, *table 5* and *eq. (11)*, we consider as follows:

(1) The result of the segmentation measure and the character recognition rate by the SVM are better than that by the neural network.
(2) The best neural network has three layers with the middle layer of 4 units. The larger the number of units for the middle layer n_{mu} is, the smaller the learning error should be, but it is practically difficult to find the global minimum for the learning error.
(3) The distribution of the outputs is very small form –1 to 1 for the SVM as shown in *Fig. 4,* which provides reliable margin to discriminate segmentation points and non-segmentation points.
(4) Although the classification time T_{ac} by the SVM is about 649 times longer than that by the neural network because the SVM must count the sum of the support vectors according to the *eq. (8)*, the average time T_{ar} for processing a text line by the SVM is only about 3 times longer than that by the neural network. This is because the segmentation by the neural network has a more number of undecided points, which incurs longer time for character recognition, constructing the candidate lattice and searching into the candidate lattice as shown in *table 4* and the *eq. (11)*. We consider that the average time T_{ar} for processing a text line by the SVM is acceptable because it is not so long.
(5) The training time T_{train} by the neural network is much shorter than that of the SVM.

5 Conclusion

This paper described a segmentation method of on-line handwritten Japanese text. We extracted multi-dimensional features from off-strokes in on-line handwritten text and applied a neural network and a SVM to produce segmentation point candidates. The SVM brought about better segmentation performance and character recognition rate, although its processing time is behind the neural network. By employing the full set of the database, we will report more accurate and reliable evaluation.

Acknowledgement

This research is being supported by Grant-in-Aid for Scientific Research under the contract number (B)17300031.

References

1. M. Nakagawa and M. Onuma, “On-line Handwritten Japanese Text Recognition Free from Constrains on Line Direction and Character Orientation,” *Proc. 7th ICDAR,* Edinburgh, 2003, pp.519-523
2. H. Aizawa, T. Wakahara and K. Odaka, “Real-Time Handwritten Character String Segmentation Using Multiple Stroke Features (in Japanese)”, *IEICE Transactions in Japan,* Vol.J80-D- II, No.5, 1997, pp.1178-1185

3. M. Okamoto, H. Yamamoto, T. Yosikawa and H. Horii, "Online Character Segmentation Method by Means of Physical Features (in Japanese)", *Technical Report of IEICE in Japan,* PRU, Vol.95, No.43, 1995, pp.93-100
4. Bilan. Zhu and M. Nakagawa, "Segmentation of On-line Handwritten Japanese Text of Arbitrary Line Direction by a Neural Network for Improving Text Recognition," *Proc. 8th ICDAR,* Seoul, Korea, 2005, pp.157-161
5. V.N.Vapnik, *Statistical Learning Theory*, J.Wiley, 1998
6. N.Cristianini and J.Shawe-Talor, *An Introduction to Support Vector Machines*, Cambridge University Press, 2000
7. M. Nakagawa, B. Zhu and M. Onuma, "A Formalization of On-line Handwritten Japanese Text Recognition free from Line Direction Constraint," *Proc. 17th International Conference on Pattern Recognition (ICPR),* Cambridge, England, 2P. Tu-i, 2004
8. R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification, Second Edition,* J. Wiley & sons, 2001
9. T. Joachims, "Making large-scale SVM learning practical," in B. Schölkopf, C. J. C. Burges, and A. J. Smola, edits, *Advances in Kernel Methods — Support Vector Learning*, Cambridge, MIT Press, 1999, pp. 169-184