control of the uncertain process with time delays. The analytical formula of the fuzzy controller was used to adaptively regulate fuzzy rules, and the neural network model for self-learning dynamics of the uncertain process or plants. The neural network predictor for the uncertain process with time delays has been used in the closed-loop systems for compensating the adverse effects of the time delays in the process. The self-learning fuzzy neural control scheme has been used for control of the pulse TIG welding process, and the experiment results show the control scheme available.

Acknowledgment

The authors would like to thank the referees for their constructive comments and suggestions.

References

[1] O. H. Smith, "Closed control of loops with dead times," *Chem. Eng. Prog.,* vol. 53, no. 6, pp. 217–219, 1957.
[2] N. Shanmugathasan and R. D. Johnston, "Exploitation of time delays for improved process control," *Int. J. Control,* vol. 48, no. 3, pp. 1137–1162, 1988.
[3] C. Batur and V. Kasparian, "Adaptive expert control," *Int. J. Control,* vol. 54, no. 4, pp. 867–881, 1991.
[4] J. A. Bernard, "Use of rule-based system for process control," *IEEE Contr. Syst. Mag.,* vol. 8, pp. 3–13, 1988.
[5] P. J. King and E. H. Mamdani, "The application of fuzzy control system to industrial processes," *Automatica*, vol. 13, pp. 235–242, 1977.
[6] W. J. M. Kickert and E. H. Mamdani, "Analysis of a fuzzy logic controller," *Fuzzy Sets Syst.,* vol. 1, pp. 29–44, 1978.
[7] W. J. M. Kickert and H. R. Van Lautc Lemke, "Application of a fuzzy controller in warm water plant," *Automatica,* vol. 12, pp. 301–308, 1976.
[8] N. J. Mandic, E. M. Scharf, and E. H. Mamdani, "Practical application of a realistic fuzzy rule-based controller to the dynamic control of a robot arm," *Proc. IEEE,* vol. 132, part D, no. 4, 1985.
[9] T. J. Procyk and E. H. Mamdani, "A linguistic self-organizing process controller," *Automatica,* vol. 15, pp. 15–30, 1979.
[10] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Networks,* vol. 3, no. 5, pp. 724–740, 1992.
[11] J. R. Jang, "Self-learning fuzzy controllers based on temporal back propagation," *IEEE Trans. Neural Networks,* vol. 3, no. 5, pp. 714–723, 1992.
[12] C. C. Lee, "A self-learning rule-based controller employing approximate reasoning and neural net concepts," *Int. J. Intell. Syst.,* vol. 5, no. 3, pp. 71–93, 1991.
[13] K. J. Hunt *et al*., "Neural networks for control systems—A survey," *Automatica,* vol. 28, no. 6, pp. 1083–1112, 1992.
[14] M. J. Willis *et al*., "Artificial neural networks in process estimation and control," *Automatica,* vol. 28, no. 6, pp. 1181–1187, 1992.
[15] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Contr. Syst. Mag.,* pp. 18–23, 1990.
[16] G. A. Rovithakis and M. A. Christodoulou, "Adaptive control of unknown plants using dynamical neural networks," *IEEE Trans. Syst., Man, Cybern.,* vol. 24, no. 3, pp. 400–411, 1994.
[17] B. Kosko, *Neural Networks and Fuzzy Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1992.
[18] S. Z. Long and P. Z. Wang, "Self-regulation problem of fuzzy control rules," *Fuzzy Math.,* vol. 3, pp. 105–111, 1982.
[19] S. Y. Li and H. Z. Hu, "Design and study of a class three-dimensional fuzzy controller," in *Proc. Int. Symp. Fuzzy Systems and Knowledge Engineering,* 1987, vol. 2, pp. 447–448.
[20] G. M. Ahdelnour *et al*., "Design of a fuzzy controller using input and output mapping factors," *IEEE Trans. Syst., Man, Cybern.,* vol. 21, pp. 952–960, 1991.
[21] A. E. Bentley and S. J. Marburger, "Arc welding penetration control using quantitative feedback theory," *Welding J.,* vol. 71, no. 11, pp. 397s–405s, 1992.
[22] N. M. Carlson, J. A. Johnson, and D. C. Kunerth, "Control of GMAW: Detection of discontinuities in the weld pool," *Welding J.,* vol. 69, no. 7, pp. 266s–269s, 1990.
[23] W. Chen and B. A. Chen, "Monitoring joint penetration using infrared sensing techniques," *Welding J.,* vol. 69, no. 9, pp. 181s–185s, 1990.
[24] S. Nagarajan *et al*., "Infrared sensing for adaptive arc welding," *Welding J.,* vol. 68, no. 11, pp. 462s–466s, 1989.

On-Line Signature Verification Using LPC Cepstrum and Neural Networks

Quen-Zong Wu, I-Chang Jou, and Suh-Yin Lee

***Abstract*—In this paper, an on-line signature verification scheme based on Linear Prediction Coding (LPC) cepstrum and neural networks is proposed. Cepstral coefficients derived from linear predictor coefficients of the writing trajectories are calculated as the features of the signatures. These coefficients are used as inputs to the neural networks. A number of single-output multilayer perceptrons (MLP's), as many as the number of words in the signature, are equipped for each registered person to verify the input signature. If the summation of output values of all MLP's is larger than verification threshold, the input signature is regarded as a genuine signature; otherwise, the input signature is a forgery. Simulations show that this scheme can detect the genuineness of the input signatures from our test database with an error rate as low as 4%.**

I. Introduction

As the computer industry is on its way to change the life styles of human beings, processes and schedules for many works are somehow affected or even completely reorganized. Office automation is one of the examples. Due to the wide use of computers and peripherals, such as printers, scanners, and digital tablets, we are in the era of intelligent input/output and electronic documentation.

Signature verification is a way to determine the validity and authority of documents. In addition, it could also be applied to security systems to prevent critical data or information from being modified or stolen. Therefore, signature verification is of great importance in electronic documentation. Its applications may include banking, credit card authorization, and personal identification. Fig. 1 demonstrates examples of genuine signatures as well as forgeries in Chinese.

Visual examination is the most popular approach for signature verification, yet there are occasions for the examiners to make mistakes or lower their thresholds of acceptance. For instance, in a commercial transaction, comparing the signature with a previously written signature, such as the signature on the back of a credit card, is more or less a challenge to the clerks. As a result, large financial losses may occur. It is clear that signature verification through visual examination can only judge the authentication roughly. Human eyes can hardly analyze the detailed writing features.

Manuscript received April 26, 1995; revised November 1, 1995.

Q.-Z. Wu is with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan, R.O.C. He is also with Telecommunication Laboratories, Chung-Li, Taiwan, R.O.C.

I. C. Jou is with Telecommunication Laboratories, Chung-Li, Taiwan, R.O.C.

S. Y. Lee is with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan, R.O.C.

Publisher Item Identifier S 1083-4419(97)00031-9.

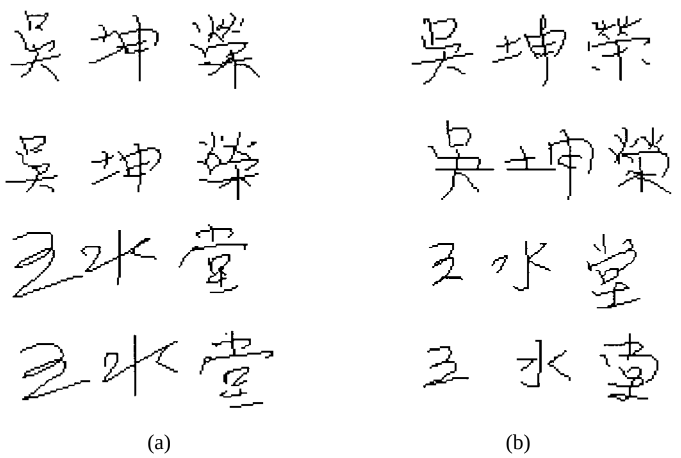

Fig. 1. The example of (a) genuine signatures and (b) forgeries.

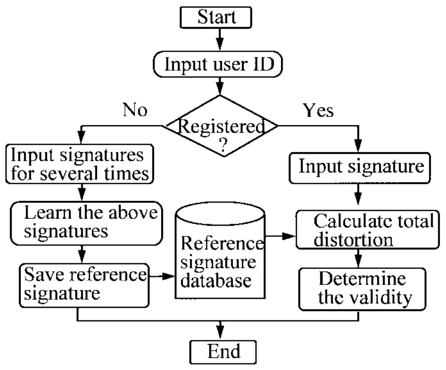
Start
Input user ID
Registered ?
No
Yes
Input signatures for several times
Learn the above signatures
Save reference signature
Reference signature database
Input signature
Calculate total distortion
Determine the validity
End

Fig. 2. Common modules for a signature verification system.

Recently, many methods [1] have been developed for computer-based signature verification, especially for on-line signature verification. Fig. 2 shows the common modules for on-line signature verification. Static features, such as coordinates, as well as dynamic features, including writing velocity and acceleration, are useful characteristics for on-line signature verification. Other issues to be considered in a verification system are distortion measurement scheme, learning scheme on the signatures, and the criteria for forgery determination [1].

Numerous pattern recognition methods have been applied to on-line signature verification [1]. Among the methods that have been proposed for pattern recognition, two broad categories can be identified: memory-based techniques in which incoming patterns are matched to a (usually large) dictionary of templates, and parameter-based methods in which preprocessed patterns are sent to a trainable classifier such as a neural network [2], [3]. Memory-based recognition methods require a large memory space to store the templates, while a neural network is a parameter-based approach which just requires a small amount of memory space to store the linking weights among neurons. For the memory-based methods, it is a difficult problem to generate a reference template (or templates) when there is a large number of training examples. In this case, the number of reference templates is usually raised in order to obtain a reasonable error rate, which requires more storage space. As for neural networks, although convergence is an issue for learning from examples, convergence can be achieved by careful tuning of the initial linking weights or adding more hidden nodes. Correlation function analysis and Euclidean distance measurement [1] are currently viewed as practical memory-based methods for signature verification. However, these methods are

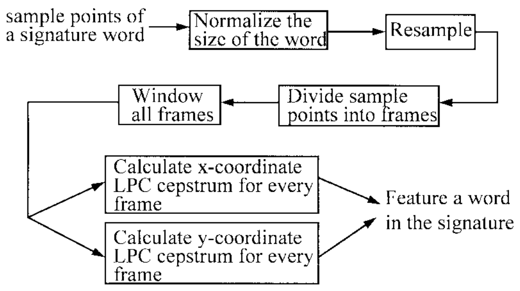
sample points of a signature word
Normalize the size of the word
Resample
Divide sample points into frames
Window all frames
Calculate x-coordinate LPC cepstrum for every frame
Calculate y-coordinate LPC cepstrum for every frame
Feature a word in the signature

Fig. 3. The procedures for feature extraction.

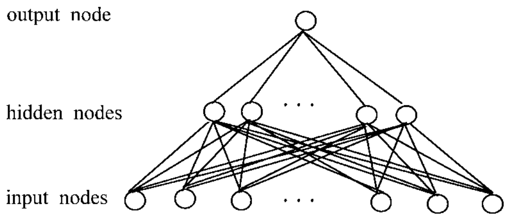

Fig. 4. A single-output multilayer perceptron.

either inherently complex or require large amount of data storage for comparison purpose.

In this paper, we propose an on-line signature verification scheme based on multilayer perceptron (MLP). For each registered customer, a number of single-output MLP's (as many as the number of words in the signature) are used to measure the similarity between input signature and the system-learned signatures. By comparing the summation of output values of all MLP's with a threshold value, we can figure out whether the input signature is a forgery or not.

Most signature verification researches select trajectory-based (or time domain) features for measuring the similarity of the signatures [4]–[10]. Trajectory-based features basically use the characteristics of all (or part of) the sample points, such as coordinates or velocities, for similarity measurement. Therefore, the system should either maintain a large amount of storage or prune away some sample data by preprocessing skills which may inherently distort the input signature. Parameters obtained from spectral domains or through some transform processes, such as Fourier transform, on the raw trajectories have been used to feature a signature. However, the amount of data is not compressed through Fourier transform. The system still faces the problem of large storage requirement. In [11], the largest 15 harmonics derived from fast Fourier transform (FFT) are used to extract the features for the signatures. In this case, the amount of data is compressed under the risk of distorting the spectrum of the signature.

As linear prediction [12] is popularly employed in a wide range of applications, we adopt cepstral coefficients derived from linear predictor coefficients (LPC) as features. A lower error rate can be achieved by using LPC cepstrum instead of LPC as the features of the signatures, which is verified in the performance comparison shown in Fig. 6. Therefore, LPC cepstrum is used to model the signatures. In this way, the amount of data is compressed, while the global characteristics of the signatures are still retained.

It is quite intuitive to link LPC cepstrum with hidden Markov model (HMM) [14]. However, the selection of the verification threshold is more difficult for HMM. As for MLP, it is intuitive and practical to adopt 0.5 as the boundary for separating positive and negative stimuli. Thus, the verification threshold could be easily selected if MLP is used for signature verification. Hence, in this paper, we propose a new scheme for signature verification which uses LPC

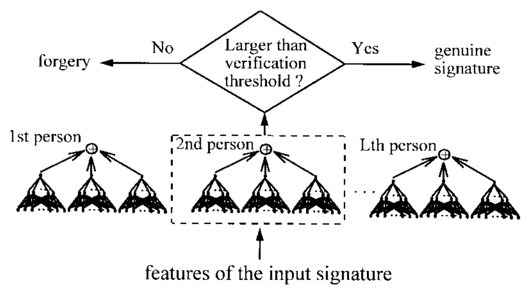
Larger than verification threshold ?
No
Yes
forgery
genuine signature
1st person
2nd person
Lth person
features of the input signature

Fig. 5. The decision of whether the input signature is a genuine one.

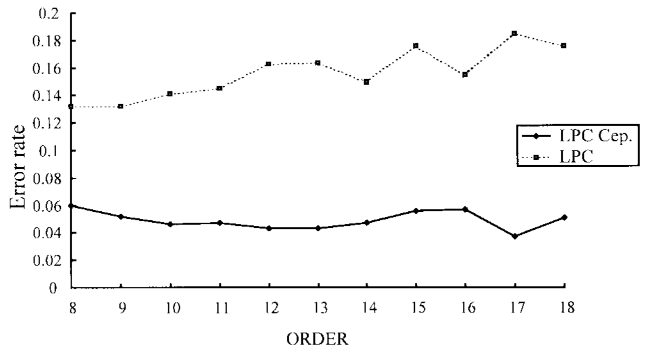

Fig. 6. Total error rates of using LPC and LPC cepstrum as input features, respectively.

cepstrum as input features and single-output MLP classifiers for mismatch measurement.

The organization of this paper is as follows. In Section II, feature extraction based on LPC cepstral analysis is discussed. Section III discusses the neural network architectures used for the classifiers. Simulations and performance evaluations of the proposed techniques are given in Section IV. Finally, Section V provides the conclusion.

II. Feature Extraction

Cepstral analysis [13]–[15] is widely used in the area of speech processing due to its capability to well represent speech waveforms and characteristics with a small number of parameters. For a time series of samples $\{s(n)\}$, an Nth order linear predictor of the sample $s(n)$, say $s^*(n)$, is a linear combination of the N previous samples $\{s(n-i),\ 1 \le i \le N\}$, denoted by

$$s^*(n) = \sum_{i=1}^{N} a_i \cdot s(n-i). \tag{1}$$

The spectrum based on the linear predictive coding (LPC) is defined as

$$S(e^{jw}) = \frac{\sigma^2}{1 - \sum_{i=1}^{N} a_i e^{-jwi}}. \tag{2}$$

LPC cepstrum is defined as the Fourier representation of the *logarithmic amplitude spectrum* based on LPC modeling. The LPC cepstral coefficients $\{c_i\}$ are defined as

$$\log|S(e^{jw})| = \sum_{i=-\infty}^{\infty} c_i e^{-jwi}. \tag{3}$$

Better performance could be achieved by selecting LPC cepstrum $\{c_i\}$ instead of LPC $\{a_i\}$ for feature extraction. Therefore, we use LPC cepstral coefficients as the features of the signatures. PARtial autoCORrelation (PARCOR) algorithm and line spectrum pair (LSP) algorithm [13]–[15] are two approaches to compute $\{a_i\}$. The LPC cepstral coefficients $\{c_i\}$ can be obtained from $\{a_i\}$ through the following equations:

$$c_1 = a_1, \tag{4}$$

$$c_i = a_i + \sum_{m=1}^{i-1} \frac{m}{i} \cdot a_m \cdot c_{i-m}, \quad \text{where } 1 < i \le N. \tag{5}$$

In speech processing, LPC coefficients as well as LPC cepstral coefficients are calculated from sample points of a speech waveform the x-axis of which is time scale and the y-axis of which is amplitude. As the speech waveform can be featured by LPC cepstral coefficients, similarly the projective waveforms $\{(x,t)\}$ and $\{(y,t)\}$ of x-coordinates and y-coordinates from a writing trajectory $\{(x,y,t)\}$ can also be characterized by LPC cepstral coefficients. Therefore, the skill of using LPC cepstrum for feature extraction can also be applied to signature verification. In addition, due to fast amplitude variation in a speech waveform, the whole set of speech samples is divided into small frames of samples by windowing techniques, such as Hamming window. This helps to preserve "short time stationary" for LPC analysis. The LPC cepstral coefficients are then calculated for each speech frame. In the case of signature, the variation of coordinates in signature waveform is not so wild as that in speech waveform. Therefore, we just divide the sample points in a word of the signature into smaller number of frames.

The procedures of feature extraction are shown in Fig. 3 and the detailed steps for feature extraction are listed as follows:

Step 1): *This step normalizes the size of the word.* At the beginning of extracting features from a word in the signature, all x-coordinates and y-coordinates of the sample points in the word are linearly normalized to be between 0 and 127 such that

$$x_{\text{new}} = 127 \cdot (x_{\text{old}} - x_{\min})/(x_{\max} - x_{\min})$$

and

$$y_{\text{new}} = 127 \cdot (y_{\text{old}} - y_{\min})/(y_{\max} - y_{\min})$$

where $x_{\max},\ x_{\min},\ y_{\max}$ and $y_{\min}$ are the maximal and minimal values of x coordinates and y coordinates, respectively.

Step 2): *This step resamples the signature word and divides the sample points into frames.* We resample every word in the signature into fixed number of sample points by interpolation skills because the sample rates of the writing tablets may be different. Assume that there are FR frames in a signature word with the size of each frame being FRAME_SIZE and the number of sample points shift between two adjacent frames being FRAME_SHIFT. The total number of sample points after resampling by interpolation skills is FRAME_SIZE + (FR − 1) $*$ FRAME_SHIFT. In addition, there are (FRAME_SIZE − FRAME_SHIFT) overlapped sample points between two adjacent frames. If the time duration for writing the signature word is T, then the signature word is resampled every $T/$[FRAME_SIZE + $(\text{FR} - 1) *$ FRAME_SHIFT] time unit.

Step 3): *This step windows each individual frame so as to minimize the sample point discontinuities at the beginning and end of each frame.* Using a Hamming window function $w(n)$ on a frame of sample points $\{(x_f(n), y_f(n))\}$, then the result of windowing, say $\{(x_f^*(n), y_f^*(n))\}$, is

$$x_f^*(n) = x_f(n)w(n) \quad \text{and} \quad y_f^*(n) = y_f(n)w(n) \tag{6}$$

where

$$w(n) = 0.54 - 0.46\cos\left(\frac{2n\pi}{\text{FRAME\_SIZE} - 1}\right). \quad (7)$$

Step 4): *This step calculates x-coordinate LPC cepstrum and y-coordinate LPC cepstrum for every frame.* Applying LPC analysis to $\{x_f^*(n)\}$ and $\{y_f^*(n)\}$, we obtain two sets of linear predictor coefficients, say $\{a_i^x\}$ and $\{a_i^y\}$, respectively. Using (4) and (5), we get LPC cepstral coefficients $\{c_i^x\}$ and $\{c_i^y\}$, which characterize a frame of a word in the signature. Therefore, there are $2 \cdot N \cdot \text{FR}$ coefficients for a word in the signature, where N is the order of LPC cepstrum and FR is the number of frames of a word. Usually, $2 \cdot N \cdot \text{FR}$ is much smaller than the number of sample points of a signature word.

A signature could be made up of several words, depending on the skills to segment the signature. The easiest way is to regard the whole signature as a big word no matter how individual words are separated. Another convenient approach is to ask users to write each signature word in a different grid on the tablet to avoid segmentation problem. In the case that the signature contains M separate words, there will be $2 \cdot N \cdot \text{FR} \cdot M$ coefficients to feature the whole signature. Typically, M is 3 for Chinese signatures.

III. Neural Network Architecture

Over the last few years, different kinds of neural network models have been proposed or derived; nevertheless the multilayer perceptrons (MLP's) are still the most widely used ones. An MLP with multiple output nodes is usually used in the applications of pattern classification while a single-output MLP is suitable for the applications such as function approximation [17], [18]. An interested reader can find more information about neural network in [2] and [3]. Fig. 4 shows the architecture of a single-output MLP. In this paper, "*back-propagation with selective updates*" [16] is used to update the linking weights of the MLP. For a presented training pattern, the linking weights are not updated when the actual output value of the MLP is very close to its desired output value based on this input pattern.

In this work, each registered person is equipped with a number of single-output MLP's (as many as the number of words in his/her signature) to determine whether the signature is genuine or not. There are $2 \cdot N \cdot \text{FR}$ nodes in the input layer of every MLP, where N is the order of LPC cepstrum and FR is the number of sample frames of a word in the signature. For a registered person with M words in his/her signature, these M MLP's are trained based on the features of the corresponding words in the signatures independently. In other words, features extracted from the first word of the signature are used to train the first MLP, and so forth.

Every MLP learns from corresponding word of both genuine and forgery signatures of the registered customer that this MLP is dedicated to. The features of genuine signature word are fed into the corresponding MLP with desired output value being 1, while the features of forgery signature word are fed into MLP with desired output value being 0.

Trained MLP's are used for verifying the signatures. The MLP's corresponding to the person that the customer declares to be are picked up and equipped with trained linking weights. Features of the input signature words are fed into the corresponding MLP's for recalling. If the summation of the output values of MLP's is larger than verification threshold, this signature is regarded as a genuine one; otherwise, it is a forgery. The diagram of deciding whether the input signature is a genuine one of the Lth registered person is shown in Fig. 5.

The forward phase of training and recalling is bounded within (0, 1). Therefore, 0.5 is suitable to be the threshold of a word in the signature and $0.5M$ is chosen to be the verification threshold of the whole signature. In typical Chinese signatures, M is 3 and the verification threshold is set to be 1.5. Certainly, verification threshold can be adjusted in order to meet different security requirements. For instance, larger verification threshold, say 1.7, sets higher barrier for forgeries to be misverified as genuine signatures but increases the possibility for genuine signatures to be rejected. In the contrast, lower verification threshold, say 1.3, sets lower barrier for forgeries to be misverified as genuine signatures yet decreases the possibility for genuine signatures to be rejected. Therefore, in highly confidential systems, the verification threshold could be elevated. When confidential rank is not so high, lower verification threshold could prevent users from unpleasant retries.

IV. Performance Evaluation

Real Chinese signatures are collected for evaluating the performance of the verification scheme proposed in this paper. Twenty-seven people are invited to register to the simulation system. Each of them is asked to write his (or her) signatures. Another four people imitate the signatures of all registered people. Based on the signatures of registered people, every imitator forges every registered person's signature.

Type I error rate is defined to be the rate that the genuine signatures are classified to be forgeries. Type II error rate is the rate that the forgeries are classified to be genuine ones. The total error rate is defined to be the ratio of the number of misverified genuine and forgery signatures over the total number of genuine and forgery signatures. We use 321 genuine signatures from 27 registered people and 321 forgeries from the first two imitators to train the single-output MLP's. The remaining 489 genuine signatures and 317 forgeries from another two imitators are used for obtaining the error rates. Therefore, if Type I and Type II error rates are $A/489$ and $B/317$, then the total error rate should be $(A + B)/(489 + 317)$.

In the simulation, FRAME_SIZE, the size of a frame, is set to be 50 sample points, while FRAME_SHIFT, the number of sample points shift between two adjacent frames, is set to be 30. Number of frames of a word in the signature (FR), order of the LPC cepstrum (ORDER), verification threshold (THRLD) and the number of hidden nodes in every MLP (H_NODE) are control parameters.

The first simulation is to validate our assumption that LPC cepstrum are better features for signature verification than LPC. Fig. 6 demonstrates the total error rates by using different orders of LPC and LPC cepstrum as input features respectively with $\text{FR} = 6$, $\text{H\_NODE} = 6$ and $\text{THRLD} = 1.5$. It shows that we can get lower error rates by using LPC cepstrum rather than LPC as input features. Fig. 7 depicts the performance that varies according to different orders of LPC cepstrum with $\text{FR} = 6$, $\text{THRLD} = 1.5$ and $\text{H\_NODE} = 6$. As the number of sample points in a frame is 50, therefore, the data compression ratio is 50/ORDER, where ORDER varies from 8 to 18 in Fig. 7. Under data storage consideration, higher data compression ratio is more economic.

Fig. 8 depicts the error rates according to different verification thresholds with $\text{FR} = 6$, $\text{ORDER} = 12$ and $\text{H\_NODE} = 6$. Lower verification threshold results in lower Type I error rates and higher Type II error rates, while higher Type I error rates and lower Type II error rates are obtained for higher verification threshold. As the total error rate is defined to be the ratio of the number of misverified genuine and forgery signatures over the total number of genuine and forgery signatures, it may fluctuate and not be increasing when verification threshold increases, although the total error rate in Fig. 8 increases when verification threshold increases.

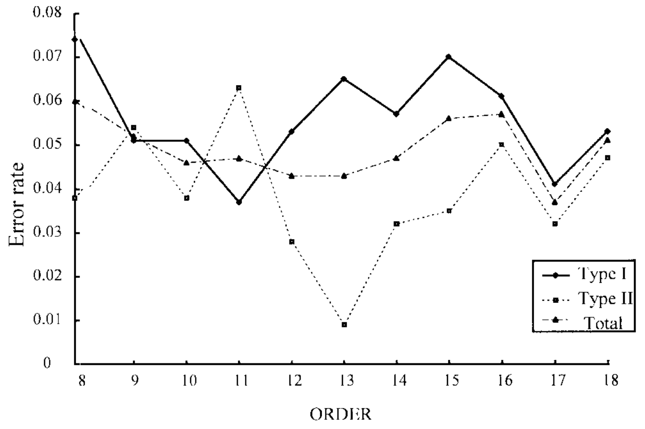

Fig. 7. Error rates according to different orders of LPC cepstrum.

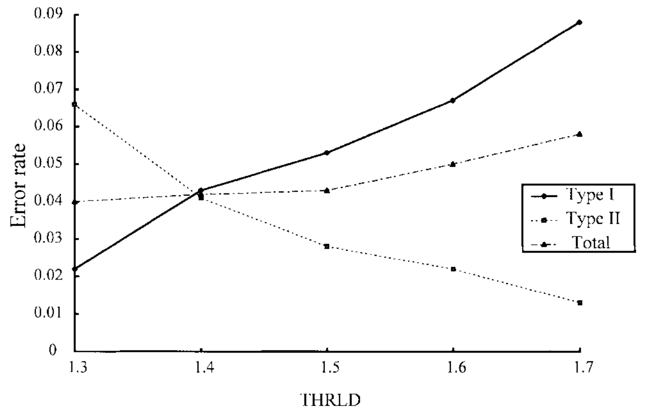

Fig. 8. Error rates according to different verification thresholds.

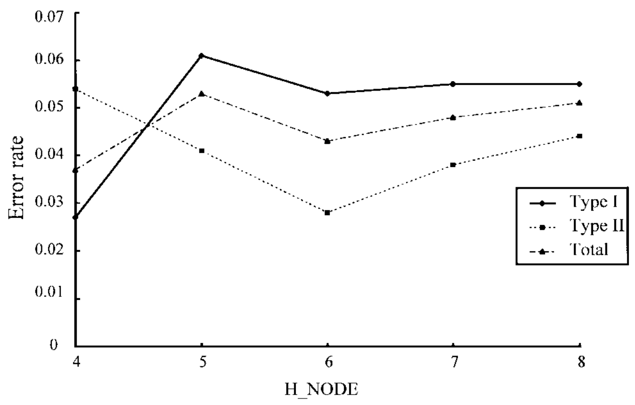

Fig. 9. Error rates according to different numbers of hidden nodes.

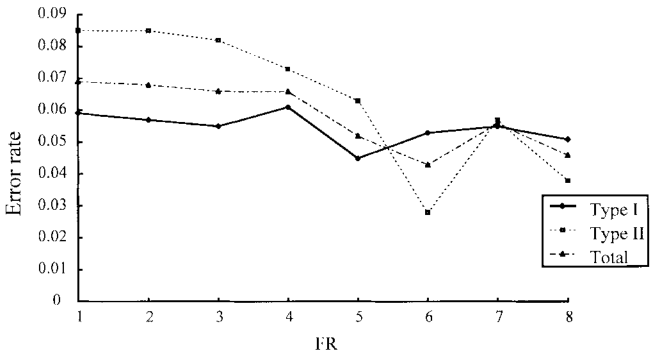

Fig. 10. Error rates according to different numbers of frames.

Fig. 9 depicts the error rates of different numbers of hidden nodes assigned to the MLP's with $\mathrm{FR} = 6$, $\mathrm{THRLD} = 1.5$ and $\mathrm{ORDER} = 12$. It is obvious that, in this case, assigning more hidden nodes to the MLP's does not always lead to lower error rates. But the number of linking weights and computations of the MLP's increases when $\mathrm{H\_NODE}$ increases.

Fig. 10 depicts the error rates by setting different numbers of frames to a signature word with $\mathrm{H\_NODE} = 6$, $\mathrm{THRLD} = 1.5$ and $\mathrm{ORDER} = 12$. It should be noticed that there are totally 50, 80, 110, 140, 170, 200, 230, or 260 sample points of a signature word after resampling as FR varies from 1 to 8.

In the illustrations of these figures, there is small variation in error rates with different settings of control parameters. Although this scheme provides a novel framework for verifying signatures, careful tuning of control parameters is necessary in order to obtain better performance. Simulations show that by careful tuning of the control parameters this scheme can determine the genuineness of the input signatures from our test database with an error rate as low as 4%.

Of course, it takes some efforts to achieve the optimal settings of control parameters. There is no theoretical warranty showing that the performance of the neural network could be upgraded by adding some hidden nodes into the MLP although it is more likely. This also holds for the settings of the order of LPC cepstrum. The complexities and degrees of variation of the signatures are different from person to person. Hence, it would be a good idea to assign different values of the control parameters to different registered people although the parameters are uniformly set in our simulation.

Most of the errors occur because signing on the tablet is not as smooth as signing on the paper. Furthermore, even genuine signatures are not time-invariant. Therefore, some variation of the signatures are "innocent" and may affect the error rates to some extent. The signatures can be embedded with innocent noises, which may train the MLP's in the wrong ways. As a result, some genuine signatures may look like forgeries and some forgeries may look like genuine signatures to the MLP.

Unlike the problems of writer identification, signature verification (or writer verification) is a process of determining the acceptance/rejection on the input writings. The MLP's dedicated to a specific registered customer are trained using only the genuine and forgery signatures of that customer. These MLP's are irrelevant to the signatures of other customers. In verification, only the MLP's which are dedicated to the claimed customer is activated for recalling. Therefore, it is clear that the performance of this verification scheme is independent of the number of customers in the system.

Although the simulations use Chinese signatures to validate the proposed signature verification scheme, it is clear that this scheme can also be applied in the verification of other kinds of signatures, such as signatures written in English, Japanese, etc.

V. Conclusion

In this paper, a novel combination of MLP classifier and cepstrum-based feature extraction is proposed for on-line signature verification. LPC cepstrum provides useful features for characterizing signatures. A number of single-output MLP's (as many as the number of words in the signature) are used as a classifier to determine whether the input signature is a genuine one or not. The result of verification is "yes" when the summation of output values of MLP's is larger than verification threshold, while it is "no" when the summation is not larger than verification threshold.

From performance tests on 27 registered people, it is clear that most genuine signatures can be successfully verified and most forgeries can be pointed out. With suitable setting of the order of LPC cepstrum, verification threshold, the number of hidden nodes in MLP and the number of frames of a word in the signature, this verification scheme performs very well. In addition, because we logically equip each registered person with a number of single-output MLP's, the verification system can be expanded by simply equipping MLP's for each new customer and training these MLP's independently. This verification scheme thus possesses the merits of flexibility, scalability and system expansion.

Although the term "signature" is generally known in western countries to refer to a handwritten name written in an alphabetic script, there is no doubt that this term could also refer to handwritten names in character form, such as Chinese characters. Section IV uses Chinese signatures, which are often written in character-by-character form, for simulation. But, in fact, this work can easily be adapted to other types of signatures.

References

[1] R. Plamondon and G. Lorette, "Automatic signature verification and writer identification—The state of the art," *Pattern Recognit.*, vol. 22, no. 2, pp. 107–131, 1989.

[2] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, pp. 4–22, Apr. 1987.

[3] D. E. Rumelhart, J. L. McClelland, and the PDP research group, *Parallel Distributed Processing, Vol. 1.* Cambridge, MA: MIT Press, 1989.

[4] Q.-Z. Wu *et al.*, "On the distortion measurement of on-line signature verification," in *Proc. 4th Int. Workshop on Frontiers in Handwriting Recognition*, Dec. 1994, pp. 347–353.

[5] Y.-T. Chen and Z. Chen, "Automatic authentication of on-line signature data," in *Proc. National Computer Symp.*, 1991, pp. 446–451.

[6] M. Yasuhara and M. Oka, "Signature verification experiment based on nonlinear time alignment: A feasibility study," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 212–216, Mar. 1977.

[7] J. S. Lew, "An improved regional correlation algorithm for signature verification which permits small speed changes between handwriting segments," *IBM J. Res. Develop.*, vol. 27, no. 2, pp. 181–185, Mar. 1983.

[8] N. M. Herbst and C. N. Liu, "Automatic signature verification based on accelerometry," *IBM J. Res. Develop.*, vol. 21, pp. 245–253, May 1977.

[9] G. Congedo, G. Dimauro, S. Impedovo, and G. Pirlo, "A new methodology for the measurement of local stability in dynamic signatures," in *Proc. 4th Int. Workshop Frontiers in Handwriting Recognition*, Dec. 1994, pp. 135–144.

[10] H.-M. Suen, J.-F. Wang, and H.-D. Chang, "On-line Chinese signature verification," in *Proc. 1993 IPPR Conf. Computer Vision, Graphics and Image Processing*, Aug. 23–25, 1993, pp. 29–36.

[11] C. F. Lam and D. Kamins, "Signature recognition through spectral analysis," *Pattern Recognit.*, vol. 22, pp. 39–44, 1989.

[12] N. Wiener, *Extrapolation Interpolation and Smoothing of Stationary Time Series.* Cambridge, MA: MIT Press, 1966.

[13] J. D. Markel and A. H. Gray Jr., *Linear Prediction of Speech.* Berlin, Germany: Springer-Verlag, 1980.

[14] S. Furui, *Digital Speech Processing, Synthesis, and Recognition.* New York: Marcel Dekker, 1989.

[15] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition.* Englewood Cliffs, NJ: Prentice-Hall, 1993

[16] S. C. Huang and Y. F. Huang, "Learning algorithms for perceptrons using back-propagation with selective updates," *IEEE Control Syst. Mag.*, pp. 56–61, Apr. 1990.

[17] T. M. Nabhan and A. Y. Zomaya, "Toward generating neural network structures for function approximation, *New Neural Network*, vol. 7, no. 1, pp. 89–99, 1994.

[18] T. P. and F. Girosi, "New networks for approximation and learning," in *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sept. 1990.

Similarity Measures Between Vague Sets and Between Elements

Shyi-Ming Chen

***Abstract*—This paper presents some similarity measures between vague sets and between elements. An example is also presented to illustrate the application of the proposed similarity measures in handling behavior analysis problems. The proposed method can provide a useful way in handling the behavior analysis problems.**

I. Introduction

Since the theory of fuzzy sets [15] was proposed in 1965, many measures of similarity between fuzzy sets have been developed in the literature [2], [4], [6], [12], [13], [17]. In [2], we presented a similarity measure for weighted reasoning for medical diagnosis. In [4], we presented a method for calculating the degree of similarity between fuzzy sets for handling fuzzy decision-making problems. In [12], Leekwang *et al.* presented two similarity measures for behavior analysis. In [13], Pappis *et al.* made a comparative assessment of measures of similarity of fuzzy values. In [6], we extended the work of [13] to present and compare the properties of some similarity measures of fuzzy values. In [17], Zwick reviewed 19 similarity measures of fuzzy sets and compared their performance in an experiment.

Roughly speaking, a fuzzy set is a class with fuzzy boundaries. A fuzzy set A of the universe of discourse U, $U = \{u_1, u_2, \ldots, u_n\}$, can be represented by

$$A = \mu_A(u_1)/u_1 + \mu_A(u_2)/u_2 + \cdots + \mu_A(u_n)/u_n \tag{1}$$

where μ_A is the membership function of the fuzzy set A, $\mu_A: U \rightarrow [0, 1]$, and $\mu_A(u_i)$ indicates the grade of membership of u_i in the fuzzy set A. When the universe of discourse U is an infinite set, then a fuzzy set A is often written in the form

$$A = \int_U \mu_A(u_i)/u_i, \quad \forall u_i \in U. \tag{2}$$

It is obvious that $\forall u_i \in U$, the membership value $\mu_A(u_i)$ is a single value between zero and one. In [7], Gau *et al.* pointed out that this single value combined the evidence for $u_i \in U$ and the evidence against $u_i \in U$, without indicating how much there is of each. Therefore, in [7], Gau *et al.* presented the concepts of vague sets, where the notion of vague set is similar to that of intuitionistic fuzzy sets [1]. They used a truth-membership function t_A and a false-membership function f_A to characterize the lower bound on μ_A. The lower bounds are used to create a subinterval on $[0, 1]$, namely $[t_A(u_i), 1 - f_A(u_i)]$, to generalize the $\mu_A(u_i)$ of fuzzy sets, where $t_A(u_i) \leq \mu_A(u_i) \leq 1 - f_A(u_i)$. For example, if $[t_A(u_i), 1 - f_A(u_i)] = [0.5, 0.7]$, then we can see that $t_A(u_i) = 0.5$, $1 - f_A(u_i) = 0.7$, and $f_A(u_i) = 0.3$. It can be interpreted as "the degree that object u_i belongs to the vague set A is 0.5, the degree that object u_i does not belong to the vague set A is 0.3." As another example, in a voting model, the vague value $[0.5, 0.7]$ can be interpreted as "the vote for resolution is 5 in favor,

Manuscript received May 15, 1995; revised October 5, 1995. This work was supported in part by the National Science Council, Republic of China, under Grant NSC 84-2213-E-009-100.

The author is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C.

Publisher Item Identifier S 1083-4419(97)00032-0.