A New Hardware Friendly Vector Distance Evaluation Function for Vector Classifiers

Hiroomi Hikawa and Kaori Kugimiya

Oita University, Oita, 870-1192 Japan

Abstract. This paper proposes a new vector distance evaluation function for vector classifications. The proposed distance evaluation function is the weighted sum of the differences between vector elements. The weight values are determined according to whether the input vector element is in the neighborhood of the prototype vector element or not. If the element is not within the neighborhood, then the weight is selected so that the distance measure is less significant The proposed distance measure is applied to a hardware vector classifier system and its feasibility is verified by simulations and circuit size evaluation. These simulations and evaluations reveal that the performance of the classifier with the proposed method is better than that of the Manhattan distance classifier and slightly inferior to Gaussian classifier. While providing respectable performance on the classification, the evaluation function can be easily implemented in hardware.

1 Introduction

Pattern classification covers very wide applications, such as face recognition, character recognition, voice recognitions, etc. In the above mentioned applications, given patterns or data are treated as vectors. The vectors could be a sequence of sampled voice data, feature vectors generated from the given images. Then a vector classification is carried out to identify the class to which the given pattern belongs. The vector classification is a mapping process of a D-dimensional space vectors into a finite set of clusters, each of which represents a particular class. Each cluster is associated with a reference prototype $\mathbf{v_i}$ that is center of the cluster, and a set of the prototypes is called as a codebook $\nu = \{\mathbf{v^{(1)}}, \mathbf{v^{(2)}}, \cdots, \mathbf{v^{(C)}}\}$. A vector classification algorithm encodes an input vector with the closest prototype that minimizes the distance to the input vector $\mathbf{x}$.

$$\mathbf{s}^{(*)} = \arg \min_{\mathbf{v_j} \in \nu} d(\mathbf{x}, \mathbf{v_j}) \tag{1}$$

where, $d(\mathbf{x}, \mathbf{v_j})$ is the distance between $\mathbf{x}$ and $\mathbf{v_j}$. $\mathbf{x}$ and $\mathbf{v}$ are D-dimensional vectors, $\mathbf{x} = \{x_1, x_2, \cdots, x_D\}$, $\mathbf{v_s} = \{m_1^{(c)}, m_2^{(c)}, \cdots, m_D^{(c)}\}$. Not only in the pattern classification, but also the distance measure plays an important roles in various field such as, data mining including self organizing maps, vector quantization, etc.

M. Ishikawa et al. (Eds.): ICONIP 2007, Part II, LNCS 4985, pp. 137–146, 2008.

Many applications use Euclidian metrics to measure the distance between two vectors.

$$d_E(\mathbf{x}, \mathbf{v_c}) = \sqrt{(x_1 - m_1^{(c)})^2 + (x_2 - m_2^{(c)})^2 + \cdots (x_D - m_D^{(c)})^2} \tag{2}$$

In hardware point of view, Manhattan distance is more desirable as it does not require square root function.

$$d_M(\mathbf{x}, \mathbf{v_c}) = \sum_{i=1}^{D} \mid x_i - m_i^{(c)} \mid \tag{3}$$

Gaussian classifiers with the following function is widely used in the pattern recognitions and radial basis function (RBF) networks.

$$d_G(\mathbf{x}, \mathbf{v_c}) = \exp(-\frac{\sum_{i=1}^{D}(x_i - m_i^{(c)})^2}{2\sigma^2}) \tag{4}$$

The vector distance is evaluated by using the nonlinear function. $d_G(\mathbf{x}, \mathbf{v})$ reaches its largest value if the input vector is at the center of the cluster.

As equations (2) $\sim$ (4) show, the conventional distance measure treat all vector elements with an identical weight. However, the relative importance of each vector element varies and improvement on the classification performance can be achieved by taking into account the relative importance of the vector elements. In [1], a new weighted distance measure has been proposed, in which the variances and mean values of vector elements of sample vectors are utilized to determine the weight factors.

On the other hand, in spite of its formal simplicity, the computational cost involved by (1) to associate a given input pattern with the best-matching prototype, can be remarkable at run time, especially in high-dimensional domains or when the code book is very large. The time required by an exhaustive-search process may be impractical for most real-world problems. Many research tackled this drawback by direct hardware implementations of the quantization math [2]-[5].

This paper proposes a new vector distance evaluation function that can be implemented in hardware with low hardware cost. The function is an weighted sum of the element distance, which is a modified version of the Manhattan distance measure. The weight value is selected according whether the input vector element is within the neighbor of the prototype vector element or not. If the input vector is not within the neighbor, the distance value is made less significant. As the proposed method requires no multipliers, or complicated function, it is suitable for the hardware implementation.

The proposed distance measure is applied to a hardware vector classifier to evaluate the performance improvement on the pattern classification, and the additional hardware cost. This paper is organized as follows: Section 2 describes the new distance measure function. In section 3, the hardware vector classifier with the proposed method is discussed. The feasibility of the method is verified by simulations. Results of the simulations are presented in section 4. Then the classifiers are designed by using VHDL, and their hardware costs are evaluated in section 5 followed by conclusions in Section 6.

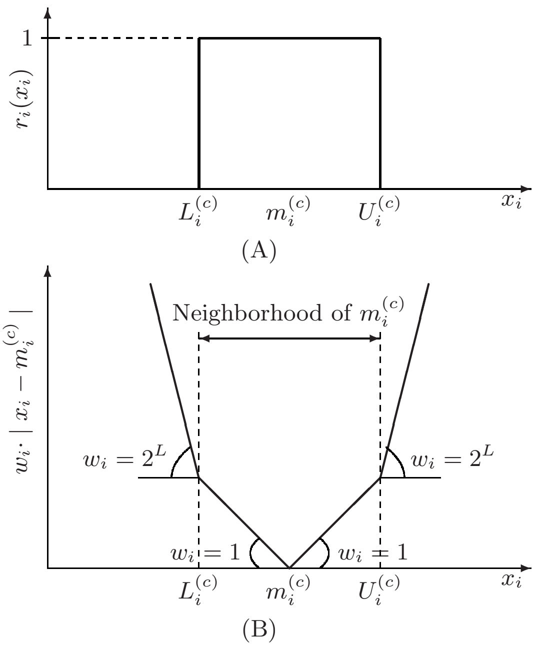

Fig. 1. Evaluation functions, (A) range check, (B) proposed method

2 New Vector Distance Measure Function

The Manhattan distance measure in (3) is modified by introducing the weighting on each $\mid x_i - m_i^{(c)} \mid$ calculation. The proposed vector measure function is,

$$d_N(\mathbf{x}, \mathbf{m}) = \sum_{i=1}^{D} w_i \mid x_i - m_i^{(c)} \mid \tag{5}$$

where w_i is the weight, and its magnitude is selected from two values adaptively according to whether the input vector element is within the neighbor of the prototype vector element $m_i^{(c)}$ or not.

$$w_i = \begin{cases} 1 & \text{if } x_i \text{ is within the neighbor of } m_i^{(c)} \\ 2^L & \text{otherwise} \end{cases} \tag{6}$$

where, L is an integer that determines the magnitude of the weight, which is a power of two value, so that no actual multiplier is necessary.

The prototype vectors and their neighborhoods are defined from the training vectors. First, the data processed by the proposed system, including the training vectors are normalized as follows,

$$x_i = \hat{x}_i / X_i \tag{7}$$

where, $\hat{x}_i$ is a raw sample data and X_i is the largest value among all i-th vector element, $X_i = \max_c \ x_i^{(c)}$. From the training data set, X_i is obtained in the training phase.

Here, the training vector is expressed as,

$$\boldsymbol{T}^{(\mathbf{c})} = \{\xi_1^{(c)}, \xi_2^{(c)}, \cdots, \xi_D^{(c)}\} \in \Re^D \tag{8}$$

where, $\xi_i^{(c)}$ is i-th training vector element belonging to class c. Class c prototype vector is defined as

$$\mathbf{v}^{(\mathbf{c})} = \{m_1^{(c)}, m_2^{(c)}, \cdots, m_D^{(c)}\} \in \Re^D \tag{9}$$

$m_i^{(c)}$ is the mean value of the samples,

$$m_i^{(c)} = \frac{\sum_{i=1}^{M^{(c)}} \xi_i^{(c)}}{M^{(c)}} \tag{10}$$

where, $M^{(c)}$ is the number of the training vectors.

Then the neighborhood of the prototype vector elements are defined by $U_i^{(c)}$, $L_i^{(c)}$, which are the upper and lower limit of the neighborhood of the cluster c vector element i, respectively.

$$U_i^{(c)} = \mu_i^{(c)} + \alpha \cdot \sigma_i^{(c)}, \tag{11}$$

$$L_i^{(c)} = \mu_i^{(c)} - \alpha \cdot \sigma_i^{(c)} \tag{12}$$

$\sigma_i^{(c)}$ is the standard deviation of the vector elements and α is a coefficient to adjust the range. To test if the input vector element x_i is within the neighborhood or not, following range check function is employed.

$$r_i^{(c)}(\ x_i\ ) = \begin{cases} 1 & \text{if} \quad U_i^{(c)} > x_i > L_i^{(c)} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Fig. 1(A) shows the function of the range check. As the figure shows, the function is a crisp function, which can be considered as the binary quantized Gaussian function. In [8], the classifier using the range check method has been proposed. Using eq.(13), the equation (6) is rewritten as,

$$w_i = \begin{cases} 1 & \text{if} \quad r_i^{(c)}(\ x_i\ ) = 1 \\ 2^L & \text{otherwise} \end{cases} \tag{14}$$

The evaluation function realized by the eq. (14) is depicted in Fig. 1(B).

If the input vector element is not in the neighborhood, the larger weight value 2^L is assigned to that element difference, resulting that the distance is made larger than the actual distance. As eq. (1) shows, in the classifying process, the smaller the distance, the more the possibility of the input vector belonging to

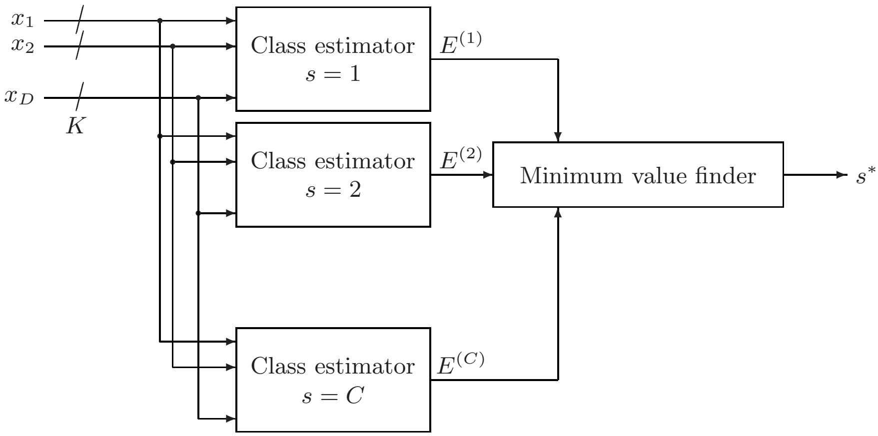

Fig. 2. Vector classification system

that cluster increases. Thus the assignment of the large weight decreases the possibility of the vector having the smallest distance to the prototype vector.

3 Vector Classifiers Based on the Proposed Distance Measure

The proposed vector distance measure described in Section 2 is applied to the hardware vector classifier. The block diagram of the classifier is shown in Fig. 2. The system consists of class estimators and a minimum value finder circuit.

3.1 Class Estimator

The class estimator output $E^{(c)}$ is given by calculating the weighted sum of the element distance as shown in Fig. 3.

$$E^{(c)} = d_N(\mathbf{x}, \mathbf{m^{(c)}}) \tag{15}$$

The class estimator consists of D subtractors, absolute circuits, range check circuits, 2:1 multiplexers, and an adder. While the absolute values of the difference between the input vector and prototype vector elements, $\mid x_i$ - $m_i \mid$ are calculated, the range check circuit checks if the input x_i is in the neighborhood by comparing it with the upper and the lower limit values. If the input is in the neighborhood, then the absolute value $\mid x_i$ - $m_i \mid$ is selected and fed to the adder, otherwise $2^L \times \mid x_i$ - $m_i \mid$ is sent to the adder. In this way eq. (14) is realized. It should be noted that the multiplication with 2^L requires no hardware as it can

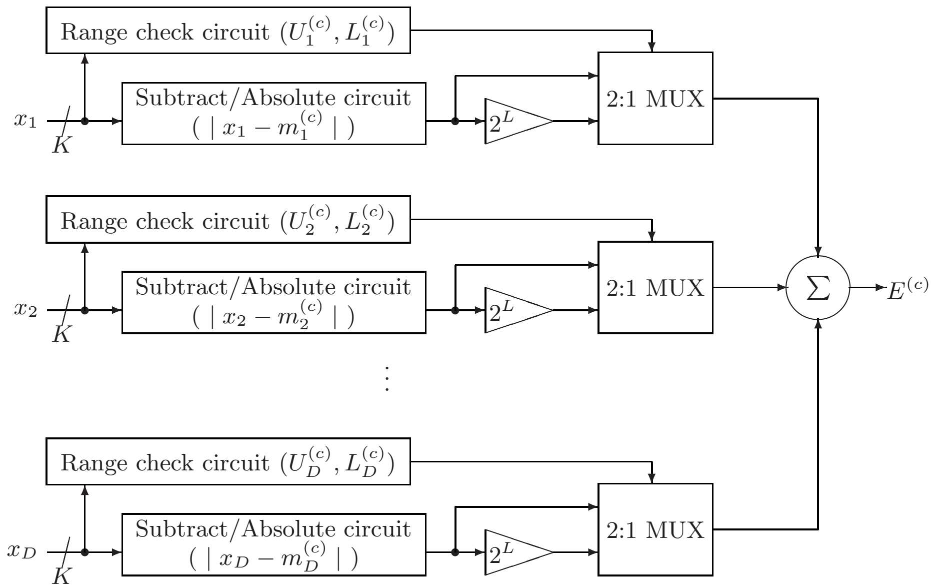

Fig. 3. Class estimator with the proposed vector distance measure

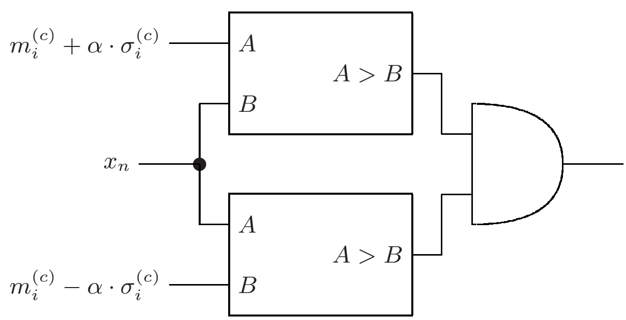

Fig. 4. Range check circuit

be implemented by the bit-shift wiring. The output $E^{(c)}$ is given as the sum of the multiplexer outputs.

3.2 Range Check Circuit

The range check circuit shown in Fig. 4 performs the range test given by equation (13). Comparator becomes active and yields '1' if the input element is between the upper and lower limit.

3.3 Class Identification

As described by the previous section, $E^{(c)}$ becomes smaller as the input vector is closer to the prototype vector of class c. Winner-takes-all competition by the minimum finder circuit is employed for the final classification. The minimum finder circuit searches for the minimum output from the class estimators, which is the winner and the class assigned to that estimator is given as the recognition results.

Each class uses a single estimator in the classifier shown in Fig. 2 as it is assumed that each class can be associated with a single clusters. However, in the case where classes are made of multiple clusters, then each class must have multiple estimators.

4 Simulations

The classifier system is described by C and the classification performance is examined.

4.1 Data Set

This section presents performance of the proposed algorithm on three data sets, i.e., IRIS [7], THYROID [6] and WINE [6] data set. They are different in terms of the data structure and the dimensionality of the feature vectors.

The IRIS data set [7] is frequently used as an example of the problem of pattern recognition. The data set consists of four features belonging to three physical classes. The features are; sepal length, sepal width, petal length and petal width. The four dimensional vector is classified into three classes, i.e., Iris Setosa, Iris Versicolour, and Iris Virginica. This data set contains 50 samples per class, totaling 150 samples.

The THYROID data set consists of five features belonging to three physical classes. This data set was obtained by recording the results of five laboratory tests conducted to determine if a patient has hypothyroidism, hyperthyroidism, or normal thyroid function.

The WINE data set consists of 13 features belonging to three physical classes. This data set was obtained by chemical analysis of wine produced by three different cultivators from the same region of Italy. This data set contains 178 feature vectors with 59 in class 1, 71 in class 2 and 48 in class 3.

All vectors in the data sets are normalized beforehand according to eq. (7).

4.2 Simulation Procedure

Following procedure is repeated 100 times and the average classification rate is used for the evaluation so that classification performance can be accurately checked.

Table 1. Recognition rate

Neuron type	IRIS	THYROID	WINE	Average
Gaussian	94.0 %	96.4 %	94.9 %	95.1 %
Manhattan	91.5 %	94.3 %	95.2 %	93.7 %
[1]	94.9 %	94.3 %	93.4 %	94.2 %
[8] ($M = 1$)	93.5 % $\alpha = 2.4$	94.6 % $\alpha = 2.4$	92.8 % $\alpha = 2.0$	93.6 %
Proposed ($L = 2$)	93.6 % $\alpha = 2.5$	95.4 % $\alpha = 3.0$	95.7 % $\alpha = 2.0$	94.9 %

Table 2. Circuit size and speed of the hardware classifier for IRIS data set

Classifier type	Gate count	Maximum delay
Manhattan	6,088	5.857 ns
[1]	14,233	5.857 ns
Proposed	8,582	5.857 ns

1. For each class, the quarter of the sample data set is randomly selected, and used as "learning data". The remaining data is used as "evaluation data".
2. Using the learning data, the prototype vectors $\mathbf{v^{(c)}}$, the upper and lower limits $U_i^{(c)}$, $L_i^{(c)}$ are defined. Then, classification rate is obtained by classification test using the evaluation data.

After the trials, the average recognition rate is used for the evaluation.

4.3 Simulation Results

The simulation results of the classifier with the proposed method is shown in Table 1. The recognition rates of the Gaussian classifier, classifier using the vector distance measure proposed in [1] and classifier with the range check circuit proposed in [8], are also obtained by the simulations and shown in the same table.

The table shows that the recognition rate of the proposed method is slightly worse than the Gaussian classifier but better than other types of classifiers.

5 Circuit Size Evaluation

The vector classifiers with the proposed method, Manhattan distance, and the measure proposed in [1], are described by VHDL, and the circuit size and speed evaluations are carried out. The correctness of the VHDL design is verified by confirming that VHDL simulation results and the C simulation results are identical. The circuit size and speed of the system are estimated by XILINX tool,

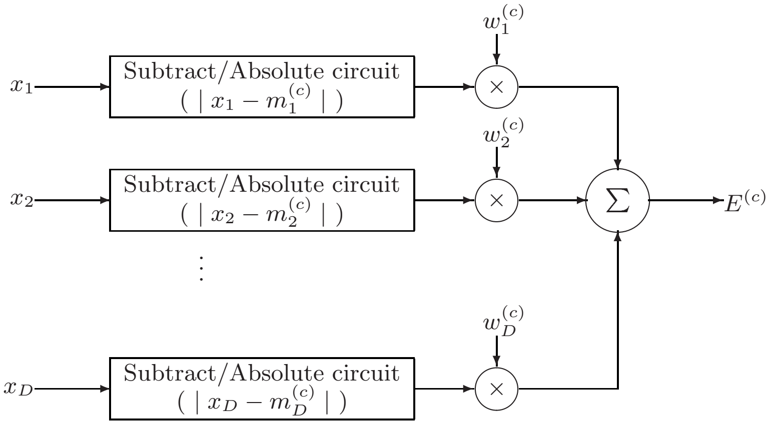

Fig. 5. Configuration of the class estimator using the vector distance measure in [1]

assuming that the design is implemented on XILINX Virtex-E device, XCV400-FG676-8. Circuit size and maximum delay of the proposed system targeting the IRIS data are summarized in Table. 2. As the classifier is realized as a combinatorial digital circuits, the maximum delay is used for the speed evaluation.

The circuit size of the proposed classifier is slightly larger than that of the Manhattan classifier and its size is less than half of the classifier using the distance measure proposed in [1]. As shown in Fig. 5, the class estimator using the distance measure in [1] uses numerical multipliers. The use of multipliers increases the total hardware cost of the system. Due to the complex function required by the Gaussian function, it is easily expected that the hardware cost of the Gaussian classifier is much higher than the classifiers listed in the table.

With regard to the speed, all systems can process all three data within 6 ns with the above mentioned FPGA.

6 Conclusions

This paper has proposed a new vector distance measure function, that is suitable for hardware implementation. The proposed method employs weighting on the vector element difference. The weight values are determined so that the element evaluation is made less significant if the element is outside the neighborhood of the prototype vector element. The proposed distance measure function is applied to the hardware vector classifier system.

The algorithm and its hardware configuration have been described followed by computer simulations to evaluate its performance. It has been revealed that the performance of the classifier with the proposed method is better than Manhattan distance and close to that of the Gaussian classifier. Even though the classification performance of the proposed method is slightly inferior to the Gaussian classifier, the smaller hardware cost of the proposed method is the great advantage over the Gaussian classifier.

Acknowledgements. This work was supported by KAKENHI, Grant-in-Aid for Scientific Research (C) (19500153) from Japan Society for the Promotion of Science (JSPS).

References

1. Kawahara, K., Shibata, T.: A New Distance Measure Employing Element-Significance Factor for Robust Image Classification. In: Proc. EUSIPCO 2005 (September 2005)
2. Lipman, A., Ynag, W.: VLSI hardware for example-based learning. IEEE Trans. VLSI Syst. 5, 320–328 (1997)
3. Rovetta, S., Zunino, R.: Efficient training of neural gas vector quantizers with analog circuit implementation. IEEE Trans. Circuits Syst. II 46, 688–698 (1999)
4. Bracco, M., Ridella, S., Zunino, R.: Digital Implementation of Hierarchical Vector Quantization. IEEE Trans. Neural Networks 14(5), 1072–1083 (2003)
5. Moritake, Y., Hikawa, H.: Category Recognition System Using Two Ultrasonic Sensors and Combinational Logic Circuit (Japanese). IEICE Transactions on Fundamentals J87-A(7), 890–898 (2004)
6. University of California at Irvine web site, `http://wwww.ics.uci.edu/~mlearn/MLRepository.html`
7. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of Eugenics 7(2), 170–188 (1936)
8. Matsubara, S., Hikawa, H.: Hardware Friendly Vector Quantization Algorithm. In: Proc. IEEE ISCAS 2005, pp. 3623–3626 (2005)