2016 International Conference on Computational Science and Computational Intelligence

A Parallel Bi-perceptron Approach and its Application to Data Classification

Daw-Ran Liou*
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 10617
*Correspondent;
dawran6@gmail.com

Yang-En Chen
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 10617

Cheng-Yuan Liou
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 10617

***Abstract*—Since the kernel function in support vector machine is arbitrary, it carries no physical meaning in practical applications. This work presents a bi-perceptron network that works in real physical space. All network parameters can be obtained in a constructive way without training. It is a divide-and-conquer way with perfect performance. We show how to operate this network to classify records.**

Keywords—perceptron; classification; machine learning; neural network; text categorization

I. INTRODUCTION

Much effort has been expended in attempts to classify various datasets. Neural networks have been developed with varying degrees of success. Some of this effort has been motivated by active research and development of training algorithms for the network to improve its performance gradually. There is no effective training algorithm that gives perfect performance [1]. This work presents a design for the network, named bi-perceptron network. This network can give perfect performance on any datasets without training. We will illustrate the design in this section and present its operation in the next section.

The bi-perceptron network is a fixed feed-forward network with three hidden layers, the output OR hidden layer; the AND hidden layer; the perceptron hidden layer; and the input layer, Fig.1. The single neuron in the output layer has a fixed OR function. Each neuron in the AND layer has a fixed AND function and receives two inputs from its two perceptrons. The number of neurons in the AND layer is flexible and depends on the difficulty of the dataset. This number is proportional to the total number of data in the set and is inverse proportional to the number of dimensions of the data roughly. The weights of the two perceptrons can be obtained from the data set.

Let the set of all patterns be $X = \{\mathbf{x}^p, p = 1, \dots, P\}$. Each pattern $\mathbf{x}^p$ is a D-dimensional column vector. The label function, $C: R^D \to N$, maps each pattern, $\mathbf{x}^p$, to its class identity number, c_P. This network has 3 hidden layers, $\{m = 1, 2, \dots, L\}$, $L = 3$, Fig. 1(c). Let n_m be the total number of neurons in the m^{th} layer.

This work has been supported by Ministry of Science and Technology (MOST), Republic of China, project no.: MOST104-2221-E002-110-MY2.

We will show that a general-position two class classification problem can be solved perfectly with these three hidden layers. This design is very different from all BP algorithms that solve the perceptron complexity, $\sum_{k=0}^{n_{m-1}} \binom{n_m}{k}$ in hidden layers [2]. Fig. 1(a) illustrates the network for a two-class problem, $c_{\mathrm{p}} \in \{'1', '2'\}$, in a two dimensional space, $D = n_0 = 2$.

In this D space, a center line of a strip, $\overline{\mathbf{x}^p\mathbf{x}^q}$, is allocated for two different patterns, $\mathbf{x}^p$ and $\mathbf{x}^q$, that are in the same class $'1'$, $\mathbf{x}^p \in '1'$ and $\mathbf{x}^q \in '1'$. Note that in a $D = 2$ space. a perceptron is represented as a line. We assume that class $'1'$ contains fewer number of patterns than that of class $'2'$. Then, this center line is split into two parallel lines, line $\bar{a}$ and line $\bar{b}$. They are in the two opposite sides of the center line and parallel to the center line, $\bar{a} \parallel \overline{\mathbf{x}^p\mathbf{x}^q} \parallel \bar{b}$.

For $\bar{a}$, pick a pattern $\mathbf{x}^r$, $\mathbf{x}^r \in '2'$, where $\mathbf{x}^r$ is the closest pattern to $\overline{\mathbf{x}^p\mathbf{x}^q}$. $\mathbf{x}^r$ and $\bar{a}$ are on the same side of $\overline{\mathbf{x}^p\mathbf{x}^q}$. Plot a parallel line $\bar{a}^r$, $\bar{a}^r \parallel \overline{\mathbf{x}^p\mathbf{x}^q}$, that passes the pattern $\mathbf{x}^r$. Pick a pattern $\mathbf{x}^s$, $\mathbf{x}^s \in '1'$, that is in between the two lines, $\bar{a}^r$ and $\overline{\mathbf{x}^p\mathbf{x}^q}$, and is the closest pattern to the line $\bar{a}^r$. Plot a parallel line, $\bar{a}^s$, $\bar{a}^s \parallel \overline{\mathbf{x}^p\mathbf{x}^q}$, that passes the pattern $\mathbf{x}^s$. Plot a decision border line, $\bar{a}^{rs}$, right in between the two parallel lines, $\bar{a}^r$ and $\bar{a}^s$. $\bar{a}^{rs}$ is the discriminate perceptron that has wide margin between the different class pair $(\mathbf{x}^r, \mathbf{x}^s)$. This $\bar{a}^{rs}$ perceptron is one of the borders of its strip. The two patterns, $\mathbf{x}^r$ and $\mathbf{x}^s$, serve as the margin-limiting stops of the region in between the two lines, $\bar{a}^r$ and $\bar{a}^s$.

The decision border line $\bar{b}^{uv}$ for $\bar{b}$ can be accomplished in a similar way on the other side of $\overline{\mathbf{x}^p\mathbf{x}^q}$. These two decision perceptrons, $\bar{a}^{rs}$ and $\bar{b}^{uv}$, are two neurons in the first hidden layer, $m = 1$, that enclose certain $'1'$ patterns in their strip. All patterns enclosed in between the two perceptrons $\bar{a}^{rs}$ and $\bar{b}^{uv}$ belong to the same class $'1'$. These enclosed patterns will be subtracted from the pattern set $'1'$ and will not be used for the determination of all other strips in the rest iterations. We will show a divide-and-conquer scheme to construct strip after strip.

Note that the strip width between the two decision perceptrons, $\bar{a}^{rs}$ and $\bar{b}^{uv}$, is useful in the determination of the

978-1-5090-5510-4/16 $31.00 © 2016 IEEE
DOI 10.1109/CSCI.2016.217

significance of its strip. Those strips with large width will be preserved with high priority. Small width strips will be omitted.

The patterns enclosed in between the two perceptrons, $\bar{a}^{rs}$ and $\bar{b}^{uv}$, are well isolated from the patterns in the other class $'2'$. Not that the stops $\mathbf{x}^{\mathrm{r}}$ and $\mathbf{x}^{\mathrm{s}}$ are different from the support vectors in SVM. The space in between the two parallel lines, $\bar{a}^{rs}$ and $\bar{b}^{uv}$, is a sector of the D space. An example of typical strips is illustrated in the Fig. 1(b). The four strips for the class $'1'$ compose of four bar-like strips. There exists physiological evidences on receptive fields, $D = 2$, for the bar-like strips [3] [4]. Note that the shape of a strip resembles that of the elongated Gaussian distribution that is used in many statistical methods. Also note that there are many other techniques to pick the center patterns $\mathbf{x}^p$ and $\mathbf{x}^q$ to build a strip.

As for the general-position two-class classification problem, each neuron in the second hidden layer has the same 'AND' function that represents the patterns in each individual strip, Fig. 1(c). The output of this neuron is "+1" for its strip patterns. The output neuron in the third hidden layer has a fixed global 'OR' function. The output of this global output neuron is "+1" for all patterns in all strips. *To our knowledge, this is the simplest MLP architecture in many aspects.*

In Figure 1(c), there are $n_1 = 2n_2$ neurons (perceptrons) in the first hidden layer, and each strip is enclosed by two parallel perceptrons.

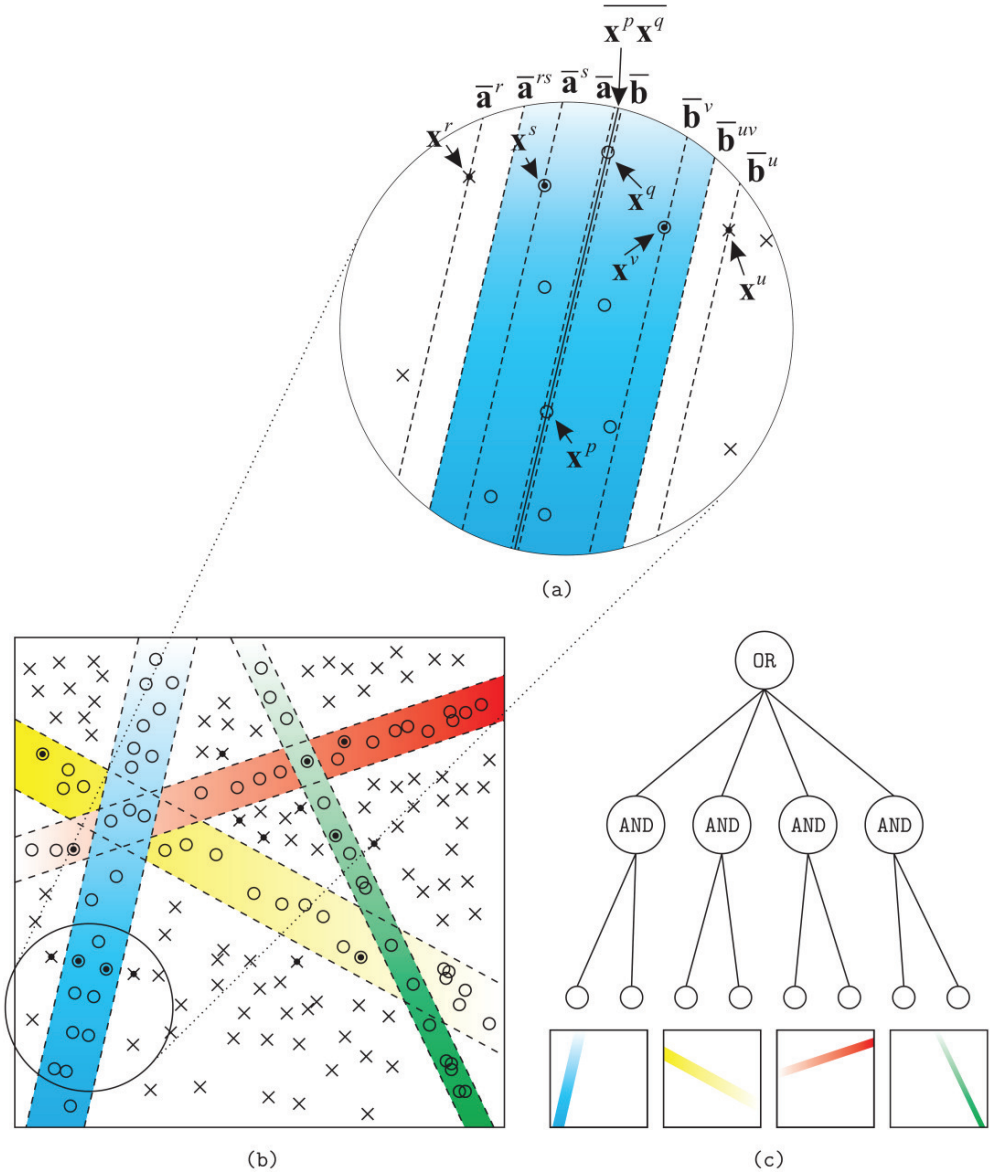

Fig. 1. The Concept of Bi-Perception Network [9].

II. METHOD

The following strategy is an example of ways to construct those strips to discriminate one selected class from the rest classes. It is expected that the strips enclose the cluster areas of the selected class and the network can be used for prediction. Therefore, both the number of patterns enclosed in each strip and its width are used as references for the level of significance. We will generate a lot of random center hyperplanes and build the strips from them. Then, pick few most significant strips as the neurons in the first layer. Note that keeping few significant strips can reduce a lot of overlapped similar strips

In each iteration, we randomly construct as many strips for a selected class, 1000 strips in all following cases, and enlarge them as possible in a similar way as in Fig. 1(a). The three most significant strips are kept for the neurons in the first hidden layer. In all our cases we keep three most significant strips and omit the rest strips. In the next iteration, we construct the significant strips for those patterns that are not enclosed in the strips constructed in all former iterations. These iterations will be continued until all patterns in the selected class are exhausted and enclosed in the strips. The last iteration may get one, two or three strips. In the last iteration, when we do not have enough D patterns in the last strip, we randomly assign a center hyperplane that passes these patterns and enlarge the strip from this center hyperplane.

To generate a strip, we start from its center hyperplane. We randomly pick D patterns in the selected class to form a hyperplane. These D patterns should not co-hyperplane. Then, we calculate the $D-1$ vectors that represent the relative position vectors of the $D-1$ patterns to the reference pattern. There is only one unit vector orthogonal to all of these vectors, which can be calculated by solving a linear system. This unit vector will be the normal vector of the center hyperplane. When there are less than D patterns to pick from, we randomly generate enough $D-1$ vectors so the normal vector of the hyperplane can be decided. Since the normal vector of the hyperplane and the point it passes through are decided, the equation of the hyperplane can be decided. When the center hyperplane is decided, move the hyperplane to both its opposite sides to get two parallel decision hyper-planes, which form a "hyper-band or strip", in a similar way as that in Fig. 1(a).

The accomplished strips are used as the neurons in the first hidden layer of the bi-perceptron network. The coefficients of each hyper-plane are exactly the weights of its neuron in the first hidden layer. All patterns in each strip will have the same outputs, "+1", from its "AND" neuron representing this strip in the second hidden layer. The functions of all neurons in the second layer and third layer are fixed logic gates. There is no need to train their weights. That is, the bi-perceptron network can still function after replacing these two layers with logic gates.

III. RESULTS AND DISCUSSION

We took six datasets to test the performance of this network. Cross Dataset and Circle Dataset were manually generated in a 2D space. Sonar, Wine, Ionosphere, and Promoters are real datasets from UCI Machine Learning Repository [5]. Their pattern dimensions, D, are 60, 13, 34, and 57 respectively. The first and the second attributes of Ionosphere dataset were ignored. This is because they contain less information but causing collinearity. So, the number of dimensions of the patterns in Ionosphere dataset was reduced to 32. The Promoters dataset is composed of DNA sequences, and each of them contains 57 nucleobases. We transformed each sequence into 57 integer attributes by encoding "A", "T", "C", and "G" as 1, 2, 3, and 4 respectively. We construct 50 networks for each dataset. We randomly picked 1/10 data out from each dataset as the testing set and then constructed the network with the rest 9/10 data. For each dataset, 50 different networks were constructed from different 9/10 datasets. Then, each network is tested by its 1/10 datasets. Table I shows the number of patterns in each class. Table II shows the average accuracies of the 50 networks that predict their test datasets for different classes. Table III compares the accuracies of various methods. The results show that the network works well in different datasets.

TABLE I. NUMBER OF PATTERNS IN SIX TESTING DATASETS.

(#)	Class 1	Class 2	Class 3
Cross Dataset	138	165	-
Circle Dataset	140	97	-
Sonar	97	111	-
Wine	59	71	48
Ionosphere	225	126	
Promoters	53	53	

TABLE II. AVERAGE ACCURACICES OF BI-PERCEPTION FOR DIFFERENT CLASSES.

	Class 1	Class 2	Class 3
Cross Dataset	94.6%	73.9%	-
Circle Dataset	96.9%	69.4%	-
Sonar	53.4%	51.8%	-
Wine	87.3%	92.2%	94.2%
Ionosphere	79.4%	71.4%	-
Promoters	53.3%	50.9%	-

TABLE III. ACCURACIES OF DIFFERENT METHODS

	Bi-perceptron	Gaussian SVM	Cubic SVM
Cross Dataset	94.6%	95.0%	77.9%
Circle Dataset	96.9%	97.0%	97.9%
Sonar	53.4%	59.6%	88.9%
Wine	94.2%	42.7%	98.3%
Ionosphere	79.4%	93.7% [a]	90.3%
Promoters	53.3%	89.6% [b]	88.7%

[a.] Kernel scale = 5.7

[b.] Kernel scale = 7.5

The bi-perceptron network predicts as good as SVM do. It does not predict well for Sonar dataset. The Sonar dataset is not linearly separable, and the clusters of its two classes are quite close to each other.

We also simulated the examples in [6] and compared the results. Five machine learning techniques, k-NN (k-nearest neighbors algorithm), SOM perceptron [6], MLP, SVM and Bi-perceptron, are compared using the 5-fold cross-validation. Table IV and Table V listed all their parameters. The dataset is randomly split into five partitions, four of them are used in the training process and the rest one is used in the testing process. The results are the average of the 5-fold cross-validation. The kept ratios in each iteration of building the bi-perceptrons are listed in Table IV. Parameter k indicates the number of neighboring cells in the k-NN algorithm. The parameters of SVM are the cost C for the error tolerance and the gamma γ in the Gaussian kernel. The values of C, γ, and k, are optimized using an inner 5-fold cross-validation procedure. The settings that produce the lowest errors are used to learn the models with the whole set of training data. The MLP has two hidden layers. Table VI and Table VII recorded the training accuracies and the testing accuracies. The bi-perceptron network can always get perfect performance 100% in training accuracy, and its testing accuracies are also comparable.

TABLE IV. PARAMETERS IN K-NN, SOM PERCEPTRON, AND BI-PERCEPTRON [6]

	k-NN	SOMP	Bi-perceptron
		(n_m, n_1^c, n_2^c)	*kept ratio*
Sonar	(3,1,1,3,3) (3,1,1,1,1)	(35,5,1)	3 of 1000
Wine	(3,15,13,11,19) (15,19,13,15,11)	(10,5,3)	3 of 1000
Ionosphere	(1,3,1,1,1) (1,11,1,1,1)	(10,5,1)	3 of 1000
Promoters	(3,5,1,3,1) (3,3,3,3,3)	(100,40,1)	3 of 1000

TABLE V. PARAMETERS IN SVM AND MLP [6]

	SVM		MLP	
	C	*gamma*	n_1^{MLP}	n_2^{MLP}
Sonar	$(2^5, 2^1, 2^5, 2^3, 2^7)$ $(2^7, 2^7, 2^3, 2^3, 2^5)$	$(2^{-3},2^{-1},2^{-3},2^{-3},2^{-5})$ $(2^{-5},2^{-5},2^{-1},2^{-3},2^{-3})$	30	10
Wine	$(2^{-1}, 2^1,2^{-1}, 2^3, 2^1)$ $(2^5,2^{-1}, 2^1,2^{-1}, 2^1)$	$(2^{-1},2^{-1},2^{-1},2^{-9},2^{-3})$ $(2^{-5},2^{-1},2^{-1},2^{-1},2^{-3})$	20	5
Ionosphere	$(2^3, 2^3, 2^3, 2^3, 2^1)$ $(2^5,2^{-1}, 2^1,2^{-1}, 2^1)$	$(2^{-1},2^{-1},2^{-1},2^{-1},2^{-5})$ $(2^{-3},2^{-5},2^{-1},2^{-3},2^{-1})$	20	5
Promoters	$(2^1, 2^1, 2^1, 2^3, 2^1)$ $(2^5, 2^1, 2^3, 2^1, 2^5)$	$(2^{-9},2^{-11},2^{-7},2^{-13},2^{-7})$ $(2^{-15},2^{-9},2^{-11},2^{-9},2^{-11})$	20	5

TABLE VI. TRAINING ACCURACICES ON REAL DATASETS [6]

	Training Accuracy				
	k-NN	*SOMP*	*MLP*	*SVM*	*Bi-perceptron*
Sonar	95.46%	100.00%	98.24%	100.00%	100.0%
Wine	97.88%	100.00%	100.00%	99.56%	100.0%
Ionosphere	97.69%	100.00%	99.46%	99.05%	100.0%
Promoters	93.72%	100.00%	100.00%	100.00%	100.0%

TABLE VII. TESTING ACCURACICES FOR REAL DATASETS [6]

	Testing Accuracy				
	k-NN	*SOMP*	*MLP*	*SVM*	*Bi-perceptron*
Sonar	82.74%	86.60%	84.14%	88.00%	53.4%
Wine	97.78%	98.33%	97.78%	98.30%	87.3%
Ionosphere	85.17%	90.60%	88.32%	94.87%	79.4%
Promoters	72.64%	86.55%	85.73%	89.36%	53.3%

Fig. 2 and Fig. 3 plotted the results of two networks constructed for the Cross dataset and the Circle dataset. Fig. 2(b) and Fig. 3(b) recorded all strips in the networks. We constructed these networks through several iterations, and the three significant strips generated in each iteration are kept and all rest insignificant strips are omitted. The locations and directions of the three most significant strips in an iteration are very similar. This is reasonable. This suggests that there is no need to keep more than three strips in each iteration.

From these six simulations, we observed that it is difficult to predict the class data when the class patterns scattered over isolated regions. A region is isolated when it is somehow surrounded by patterns in other classes. This difficulty happens because we use strips, instead of polyhedrons, as building blocks to solve isolated regions. Using polyhedrons will increase the flexibility of the network, but it also increases the complexity drastically [2]. The network needs a large number of discriminate patterns to construct reliable

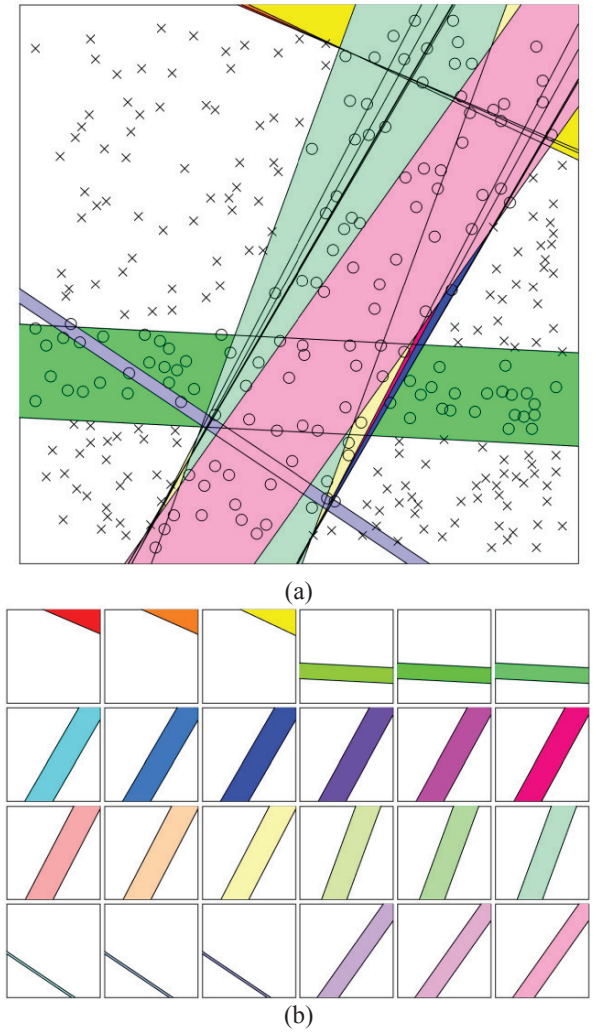

Fig. 2. The bi-perceptron trained for the Cross dataset.

strip complex when the patterns are randomly scattered over isolated regions.

IV. Summary

This work presents a divide-and-conquer construction for the perceptrons in the network without any training process. Both the number of neurons and the number of layers are much less than those obtained by the tiling algorithm [7]. This network gives perfect performance for any training datasets. The strips discriminates different classes in physical attributes directly. This will make comprehension easier on classification results than that obtained from SVM. To our knowledge, the performance of this network is comparable to that of SVM [8]. It is easy to extend the design for discriminating multiple-class problems, class after class.

Instead of strips, one may construct isolated polyhedral regions to improve the accuracies recorded in Table I, II, III, and VII. This is because the strips may not suitable for certain data distributions and not good for their predictions. Note that the shape of a strip resembles that of an elongated Gaussian distribution. The method for the construction of strips can be directly extended to the construction of isolated polyhedral regions. Each region, that contains the same class patterns, is enclosed by $(D + 1)$, or more, hyperplanes (perceptrons). Several isolated polyhedral shapes are illustrated in Fig. 4.

As for training, one can fix the functions of all neurons in the second hidden layer and the third hidden layer, with 'AND' and 'OR' functions respectively. Then apply the BP algorithm [1] to train all weights of the $n_1 = 2n_2$ perceptrons in the first hidden layer only. During training, the sigmoid function is used in all network neurons. Each weight of the 'OR' neuron is fixed and set to 0.2. Each weight of the 'AND' neuron is also fixed and set to +1 during the training process. Note that the method in [9] provides a training algorithm for the whole network weights.

When we extend the strip region to the polyhedral region, we may set $D + 1$, or more, neurons to enclose an isolated polyhedral region and connect these $D + 1$ neurons to their "AND" neuron in the second hidden layer. Then, we apply the BP algorithm [1] to train all $n_1 = n_2(D + 1)$ neurons in the first hidden layer only. During training, the sigmoid function is used in all network neurons. Each weight of the 'OR' neuron is fixed and set to 0.2. Each weight of the 'AND' neuron is also fixed and set to +1 during the training process. After training, all patterns in a single polyhedral region must

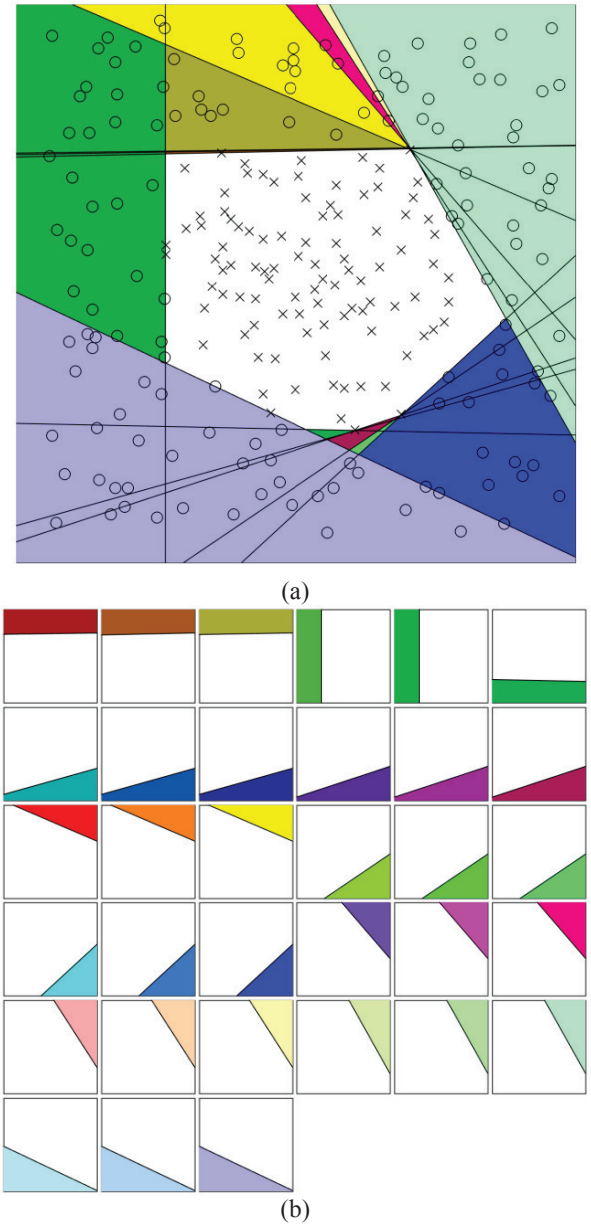

Fig. 3. The bi-perceptron trained for the Circle dataset.

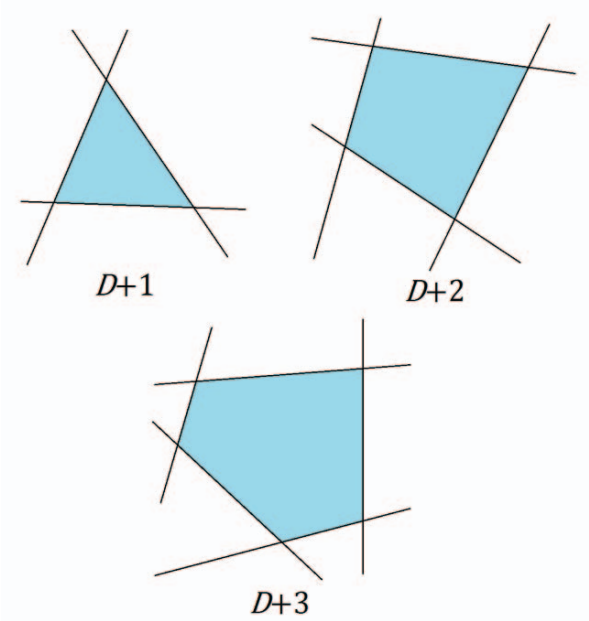

Fig. 4. Polyhedral Shapes Enclosed by Hyperplanes (Perceptrons).

belong to the same class [8]. To our knowledge, this training can improve the testing accuracies drastically and outperform all others.

REFERENCES

[1] Sejnowski, T. J. and Rosenberg, C. R. 1986. NETtalk: a parallel network that learns to read aloud. Johns Hopkins University Department of Electrical Engineering and Computer Science Technical Report 86/01.

[2] Mirchandini, G. and Cao, W. 1989. On hidden nodes in neural nets. IEEE Transaction Circuits and Systems 36:661-664.

[3] Daugman, J. G. 1980. Two-dimensional spectral analysis of cortical receptive field profiles. Vision Research 20:847–856.

[4] Dobbins, A., Zucker, S. W., and Cynader, M. S. 1987. Endstopped neurons in the visual cortex as a substrate for calculating curvature. Nature 329:438-441.

[5] Lichman, M. 2013. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[6] Liou, C.-Y. and Cheng,W.-C. 2011. Forced Accretion and Assimilation Based on Self-organizing Neural Network. In: Self Organizing Maps - Applications and Novel Algorithm Design, Chapter 35 in book edited by: Josphat Igadwa Mwasiagi, page 683~702, ISBN: 978-953-307-546-4, Publisher: InTech, Publishing date: January 2011.

[7] Mezard, M. and Nadal, J. P. 1989. Learning in feedforward layered networks: the tiling algorithm. Journal of Physics A22:2191-2203.

[8] Boser, B. E., Guyon, I. M., and Vapnik, V. N. 1992. A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 144-152. Pittsburgh, PA, USA: ACM.

[9] Liou, C.-Y. and Yu, W.-J. 1994. Initializing the weights in multilayer network with quadratic sigmoid function. In Proceedings of the International Conference on Neural Information Processing, 1387-1392. Seoul, Korea: APNNA.