

A VERY FAST NEURAL LEARNING FOR CLASSIFICATION USING ONLY NEW INCOMING DATUM AND HYPER-ELLIPSOIDAL FUNCTION

Saichon Jaiyen, Chidchanok Lursinsap, and Suphakant Phimoltares

Advanced Virtual and Intelligent Computing Center (AVIC)
Department of Mathematics,
Faculty of Science,
Chulalongkorn University

Supported by Thailand Research Fund

Agenda

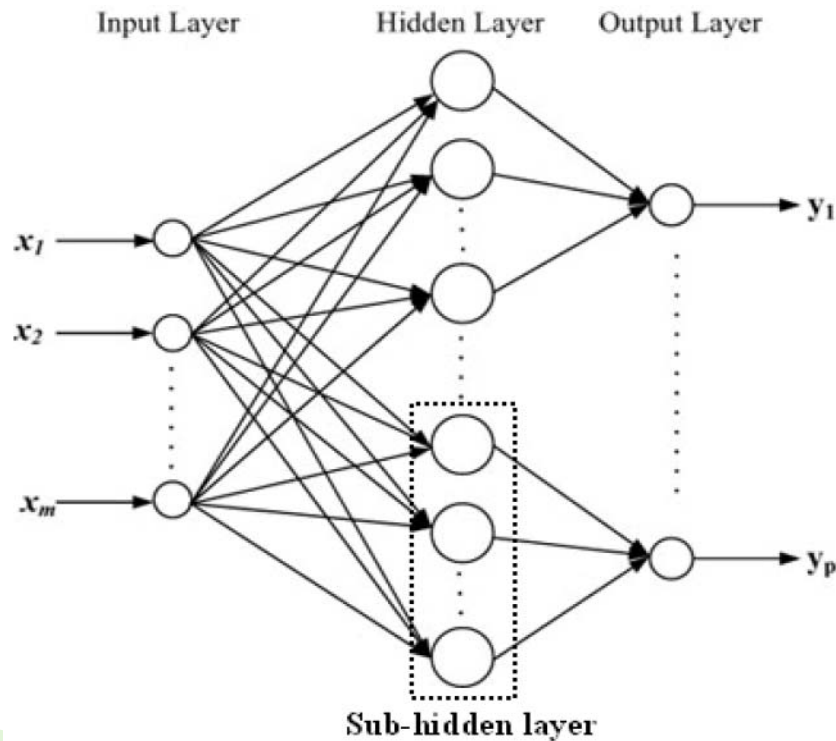
- ▶ Introduction
- ▶ Methodology
- ▶ Experimental Results
- ▶ Conclusion

Introduction

- ▶ All traditional learning algorithms still cannot learn a new data set without mixing with the previously learned data set.
- ▶ All traditional neural networks cannot learn a new training data set if the old training data set is discarded.
- ▶ Many learning algorithms use so many epochs in learning process because of optimization techniques.

VEBF neural network

- ▶ We proposed a new neural network with the incremental learning capability called *versatile elliptic basis function neural network* (VEBF neural network).



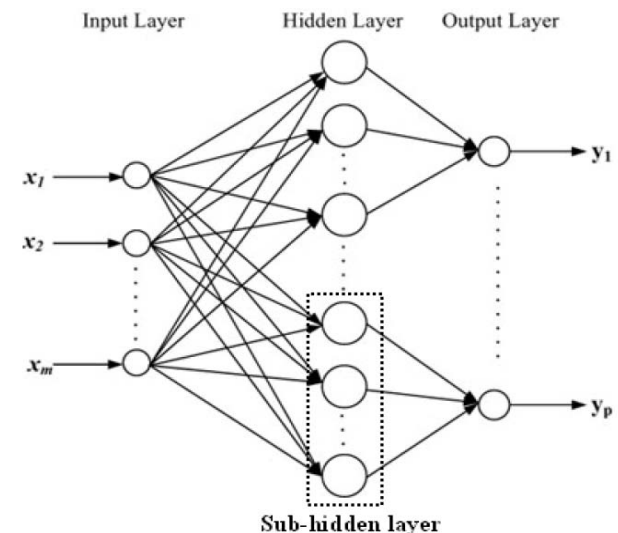
VEBF neural network

- ▶ The output of the k^{th} hidden neuron can be calculated by

$$\varphi_k(\mathbf{x}) = \sum_{i=1}^n \frac{((\mathbf{x}-\mathbf{c}_k)^T \mathbf{u}_i)^2}{a_i^2} - 1$$

- ▶ The output of the p^{th} output neuron can be calculated by

$$y_p(\mathbf{x}) = \min_j(\varphi_j(\mathbf{x}))$$



Decision Function

- ▶ We define the *decision function*, $D(\mathbf{x})$, for predicting the class label of the new input vector \mathbf{x} as follows.

$$D(\mathbf{x}) = \begin{cases} k, & \text{if } k = \arg \min(y_i(\mathbf{x})) \text{ and } y_k(\mathbf{x}) \leq 0 \\ \textit{unknown}, & \text{otherwise} \end{cases}$$

Recursive Mean Computation

- ▶ **Theorem 1.** Let $X = \{x_1, x_2, \dots, x_N\}$ be a set of N data vectors and μ_{old} be the mean vector of the data set X . If x_{N+1} is the new data vector added into the data set X then

$$\mu_{new} = \alpha \mu_{old} + \beta$$

where $\alpha = \frac{N}{N+1}$ and $\beta = \frac{x_{N+1}}{N+1}$

Recursive Covariance Computation

- ▶ **Theorem 2.** Let $X = \{x_1, x_2, \dots, x_N\}$ be a set of N data vectors, μ_{old} the mean vector of the data set X , and S_{old} the covariance matrix of the data set X . If a new data vector x_{N+1} is added into the data set X then

$$S_{new} = \alpha S_{old} + K$$

Recursive Covariance Computation

where

$$\alpha = \frac{N}{N + 1}$$

$$K = K_1 + K_2$$

$$K_1 = \frac{\mathbf{x}_{N+1} \mathbf{x}_{N+1}^T}{N + 1} - \mu_{new} \mu_{new}^T + \mu_{old} \mu_{old}^T$$

$$K_2 = -\frac{\mu_{old} \mu_{old}^T}{N + 1}$$

Orthonormal Basis Computation

- ▶ Calculate the mean vector.
- ▶ Calculate the covariance matrix.
- ▶ Calculate the eigenvalues of covariance matrix:

$$\lambda_1 > \lambda_2 > \dots > \lambda_n$$

- ▶ Calculate the eigenvectors of covariance matrix:

$$u_1, u_2, \dots, u_n$$

- ▶ $\{u_1, u_2, \dots, u_n\}$ is the orthonormal basis.

The Proposed Learning Algorithm

▶ Geometrical Growth Criterion

- If there is the nearest neuron, then the temporary parameters are calculated as follows.

$$C_{temp} = \alpha C_K + \beta$$

$$S_{temp} = \alpha S_K + \kappa$$

- The temporary output of the closest hidden neuron is computed using the new temporary parameters.

$$\varphi_{temp}(\mathbf{x}) = \sum_{j=1}^n \frac{((\mathbf{x} - C_{temp})^T \mathbf{u}_j)^2}{a_j^2} - 1$$

The Proposed Learning Algorithm

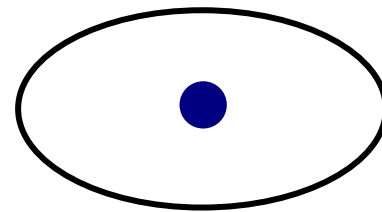
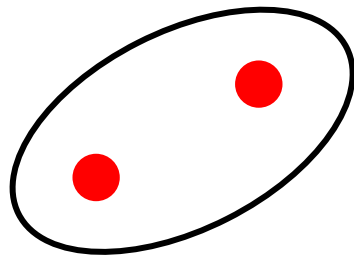
- If $\varphi_{CS}(\mathbf{x}) > 0$ or there is no nearest neuron, then a new hidden neuron is allocated and added into the network.

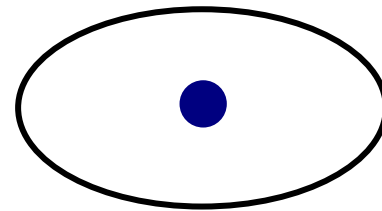
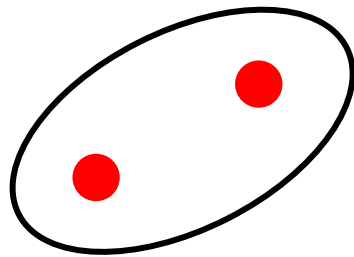
$$C_{K+1} = \mathbf{x}$$

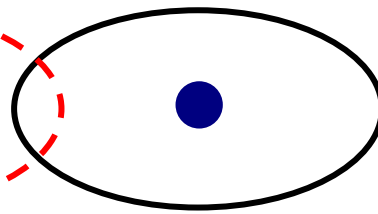
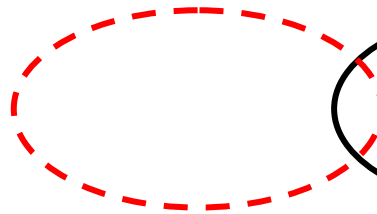
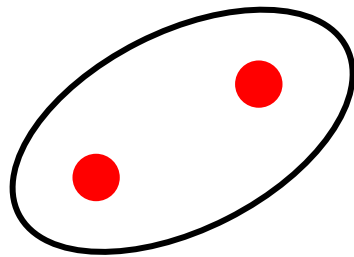
$$S_{K+1} = 0$$

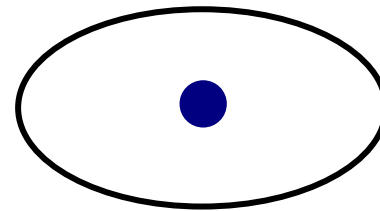
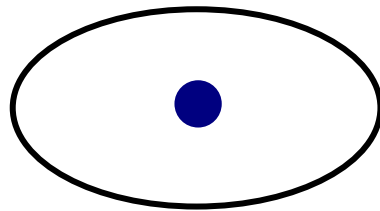
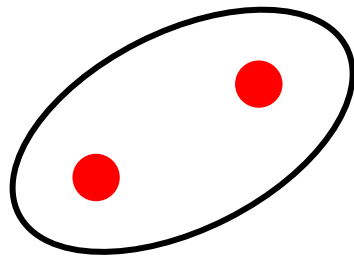
$$a_j^{K+1} = a_j^0$$

$$N_{K+1} = 1$$









The Proposed Learning Algorithm

▶ Parameter Adjustment

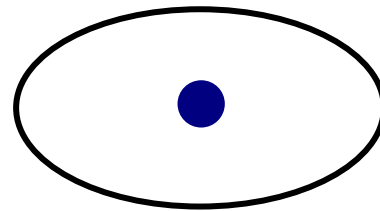
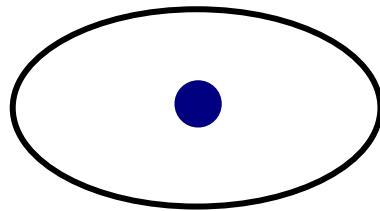
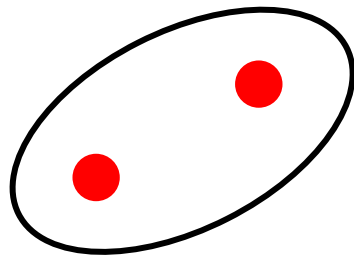
- If $\varphi_{CS}(\mathbf{x}) \leq 0$, then the parameters of the nearest neuron are adjusted as follows.

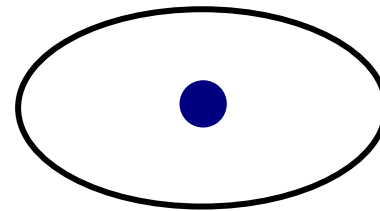
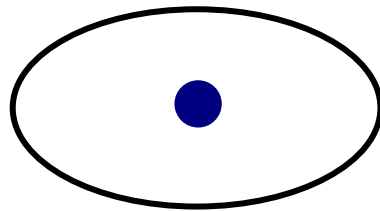
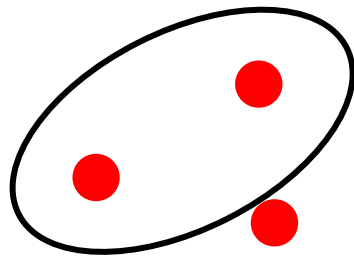
$$C_K^{new} = C_{temp}$$

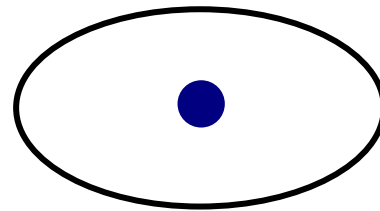
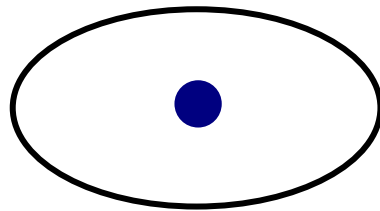
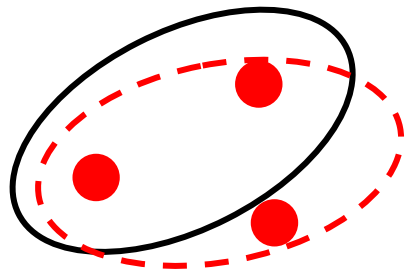
$$S_K^{new} = S_{temp}$$

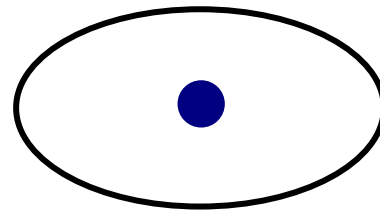
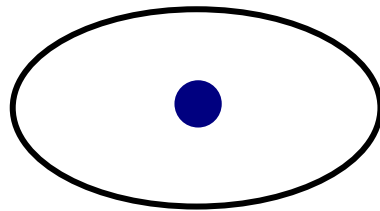
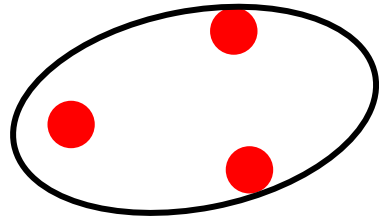
$$a_j^{new} = a_j^{old} + | (C_K^{new} - C_K^{old})^T \mathbf{u}_j |$$

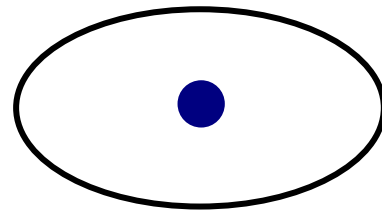
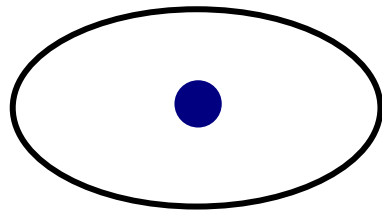
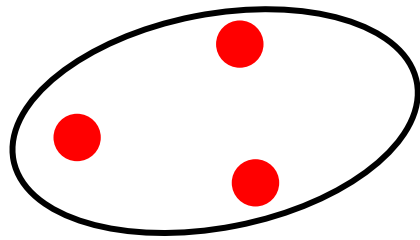
$$N_K^{new} = N_K^{old} + 1$$











The Proposed Learning Algorithm

► Merging Strategy

- we define the *merging function* as follows.

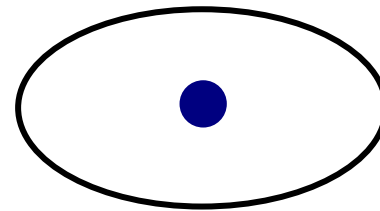
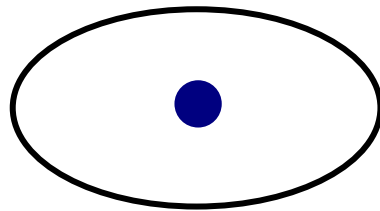
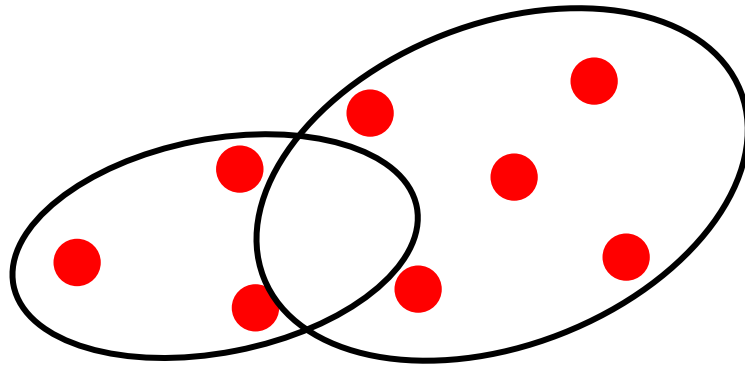
$$\phi(C_x, C_y) = \sum_{i=1}^n \frac{((C_x - C_y)^T \mathbf{u}_i^{(y)})^2}{a_i^{(y)2}} - 1$$

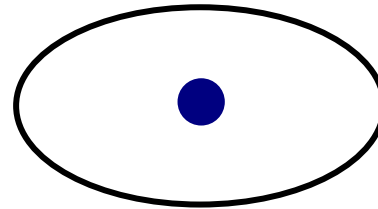
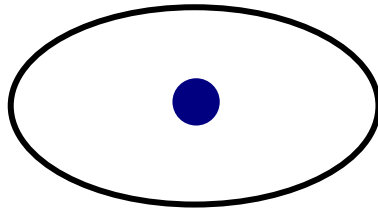
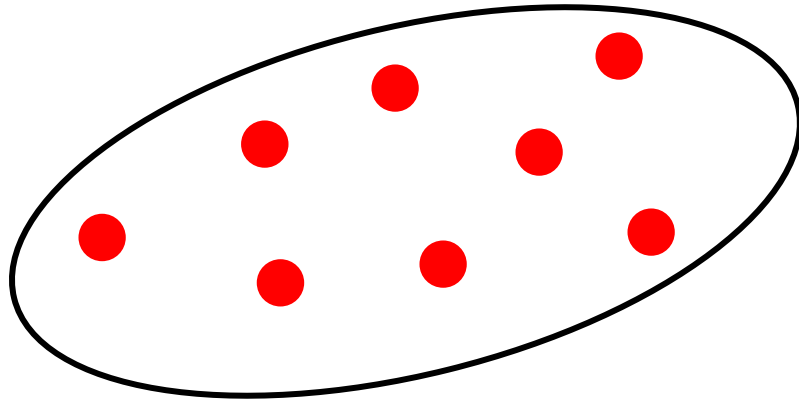
- If merging criterion, $\phi(C_x, C_y) \leq \theta$ then these two hidden neurons are merged into one new hidden neuron.

The Proposed Learning Algorithm

- ▶ The new parameters of this new hidden neuron can be computed as follows.

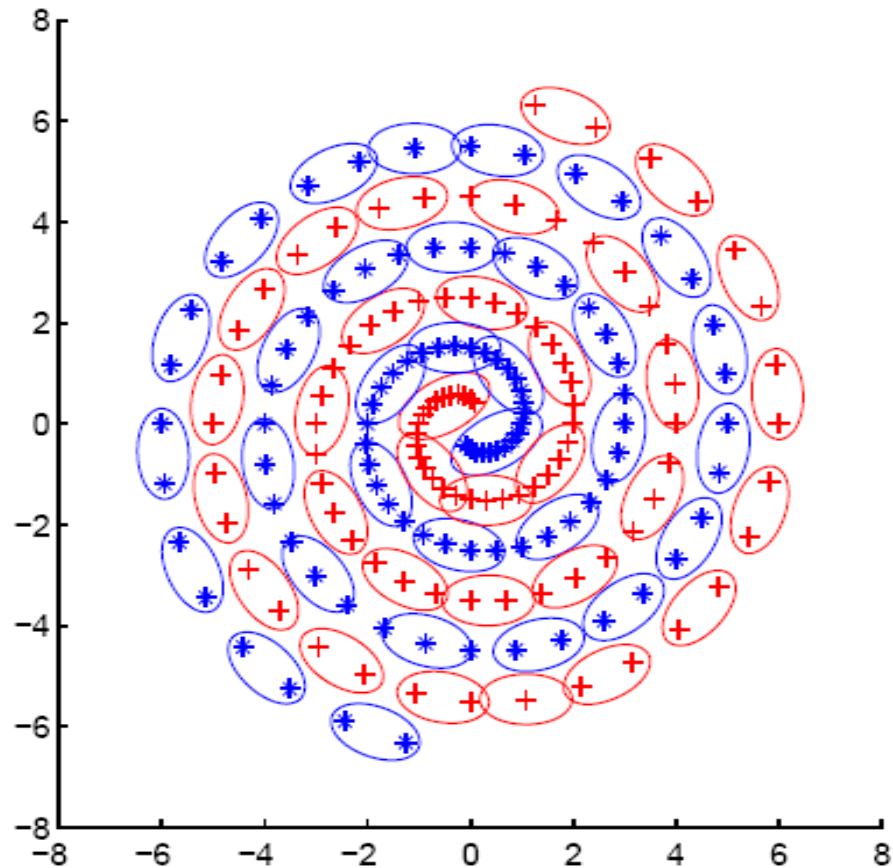
$$\begin{aligned}C^{new} &= \frac{1}{N_x + N_y} (N_x C^{(x)} + N_y C^{(y)}) \\S^{new} &= \frac{N_x}{N_x + N_y} S^{(x)} + \frac{N_y}{N_x + N_y} S^{(y)} \\&\quad + \frac{N_x N_y}{(N_x + N_y)^2} (C^{(x)} - C^{(y)})(C^{(x)} - C^{(y)})^T \\a_j^{new} &= \sqrt{2\pi |\lambda_j|}, \quad j = 1, \dots, n \\N^{new} &= N_x + N_y\end{aligned}$$





Experimental Results

- ▶ The two spiral data set trained by our proposed algorithm.



Experimental Results (Multi-Class Classification)

► Iris Data Set

Testing fold	VEBF			RBF			MLP		
	Time (second)	No. of neuron	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)
1	0.08	3	96.67	1.11	3	96.67	1.18	3	96.67
2	0.03	3	100.00	0.13	3	96.67	0.43	3	100.00
3	0.03	3	96.67	0.10	3	100.00	0.43	3	100.00
4	0.03	3	100.00	0.10	3	96.67	0.44	3	96.67
5	0.03	3	96.67	0.10	3	96.67	0.43	3	96.67
Average	0.04	3	98.00	0.31	3	97.33	0.58	3	98.00

Experimental Results (Multi-Class Classification)

► Ecoli Data Set

Testing fold	VEBF			RBF			MLP		
	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)
1	0.12	8	88.41	1.13	8	68.12	1.24	8	66.67
2	0.09	8	87.88	0.16	8	65.15	0.49	8	72.73
3	0.09	8	92.54	0.15	8	64.18	0.57	8	62.69
4	0.08	8	79.71	0.14	8	56.52	0.52	8	62.32
5	0.09	8	84.62	0.15	8	67.69	0.43	8	72.31
Average	0.09	8	86.63	0.35	8	64.33	0.65	8	67.34

Experimental Results (Multi-Class Classification)

► Yeast Data Set

Testing fold	VEBF			RBF			MLP		
	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)	time (second)	No. of neurons	Accuracy (%)
1	0.58	17	58.59	3.04	17	41.08	1.86	17	42.09
2	0.48	17	55.70	2.08	17	39.26	2.26	17	42.62
3	0.50	18	56.76	2.16	18	43.24	0.75	18	38.18
4	0.59	17	55.41	2.13	17	35.14	1.44	17	43.58
5	0.53	17	54.88	2.08	17	37.71	1.02	17	36.36
Average	0.54	17.2	56.27	2.30	17.2	39.29	1.47	17.2	40.57

Experimental Results (Multi-Class Classification)

► Image Segmentation Data Set

Testing fold	VEBF			RBF			MLP		
	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)
1	1.69	13	75.76	5.63	13	39.61	7.70	13	89.18
2	1.38	12	80.74	4.61	12	45.02	3.20	12	87.45
3	1.16	10	76.62	4.19	10	40.26	3.13	10	89.18
4	1.42	12	80.74	4.59	12	45.45	1.71	12	86.36
5	1.38	12	79.00	4.54	12	44.59	3.20	12	81.17
Average	1.41	11.8	78.57	4.71	11.8	42.99	3.79	11.8	86.67

Experimental Results (Multi-Class Classification)

► Waveform Data Set

Testing fold	VEBF			RBF			MLP		
	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)
1	2.65	3	83.82	15.90	3	58.14	2.32	3	74.13
2	3.45	3	85.60	14.71	3	60.10	2.08	3	79.80
3	2.45	3	83.58	14.75	3	60.86	1.83	3	77.08
4	2.46	3	84.50	14.68	3	64.80	7.24	3	78.20
5	2.45	3	85.30	14.78	3	60.50	2.65	3	75.30
Average	2.69	3	84.56	14.96	3	60.88	3.22	3	76.90

Experimental Results (Multi-Class Classification)

► Balance Scale Data Set

Testing fold	VEBF			RBF			MLP		
	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)
1	0.24	4	91.27	1.16	4	84.13	1.58	4	84.92
2	0.15	3	91.94	0.38	3	78.23	0.66	3	88.71
3	0.16	5	86.51	0.41	5	82.54	0.65	5	84.92
4	0.15	3	95.12	0.38	3	79.67	0.83	3	91.06
5	0.16	5	89.68	0.41	5	84.92	0.88	5	89.68
Average	0.17	4	90.90	0.55	4	81.90	0.92	4	87.86

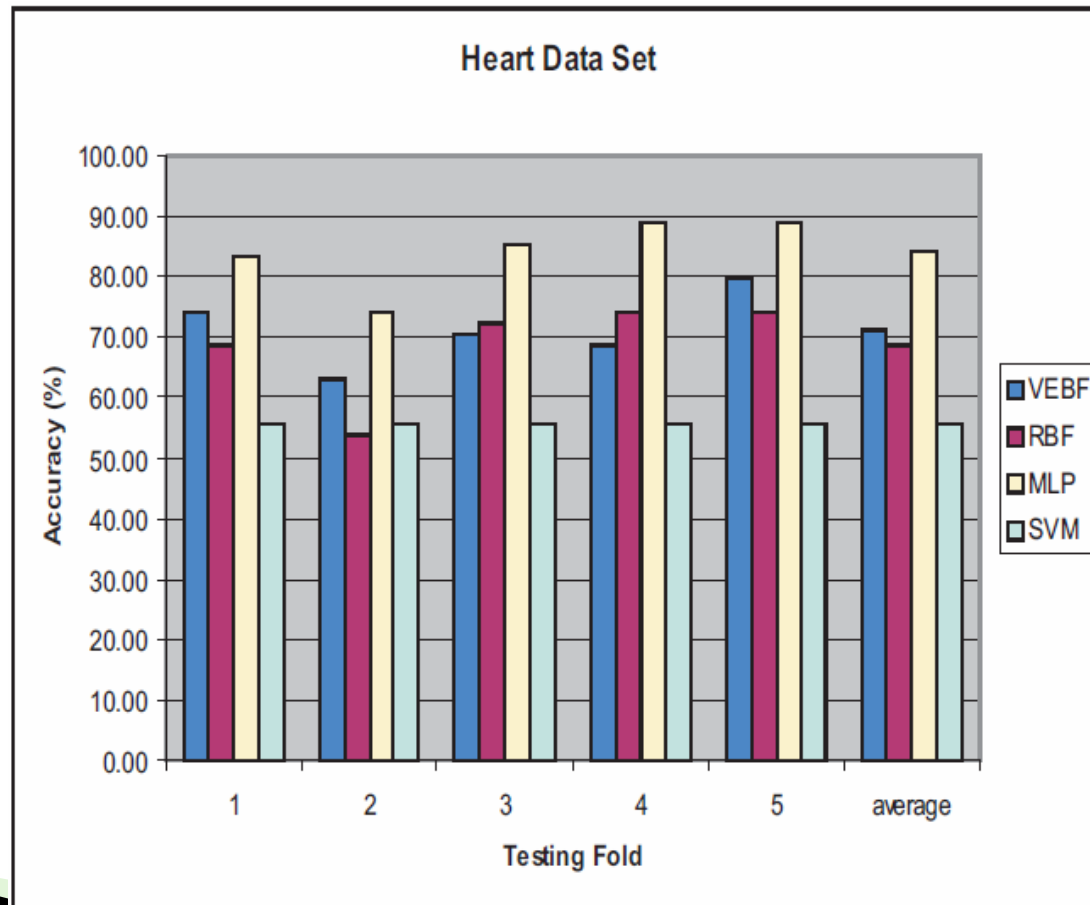
Experimental Results (Multi-Class Classification)

► Zoo Data Set

Testing fold	VEBF			RBF			MLP		
	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)	Time (second)	No. of neurons	Accuracy (%)
1	0.10	11	100.00	0.82	11	78.95	1.29	11	94.74
2	0.05	11	100.00	0.13	11	75.00	0.48	11	60.00
3	0.04	11	95.24	0.09	11	80.95	0.51	11	85.71
4	0.04	10	100.00	0.19	10	75.00	0.41	10	80.00
5	0.05	11	95.24	0.21	11	90.48	0.46	11	95.24
Average	0.05	10.8	98.10	0.29	10.8	80.08	0.63	10.8	83.14

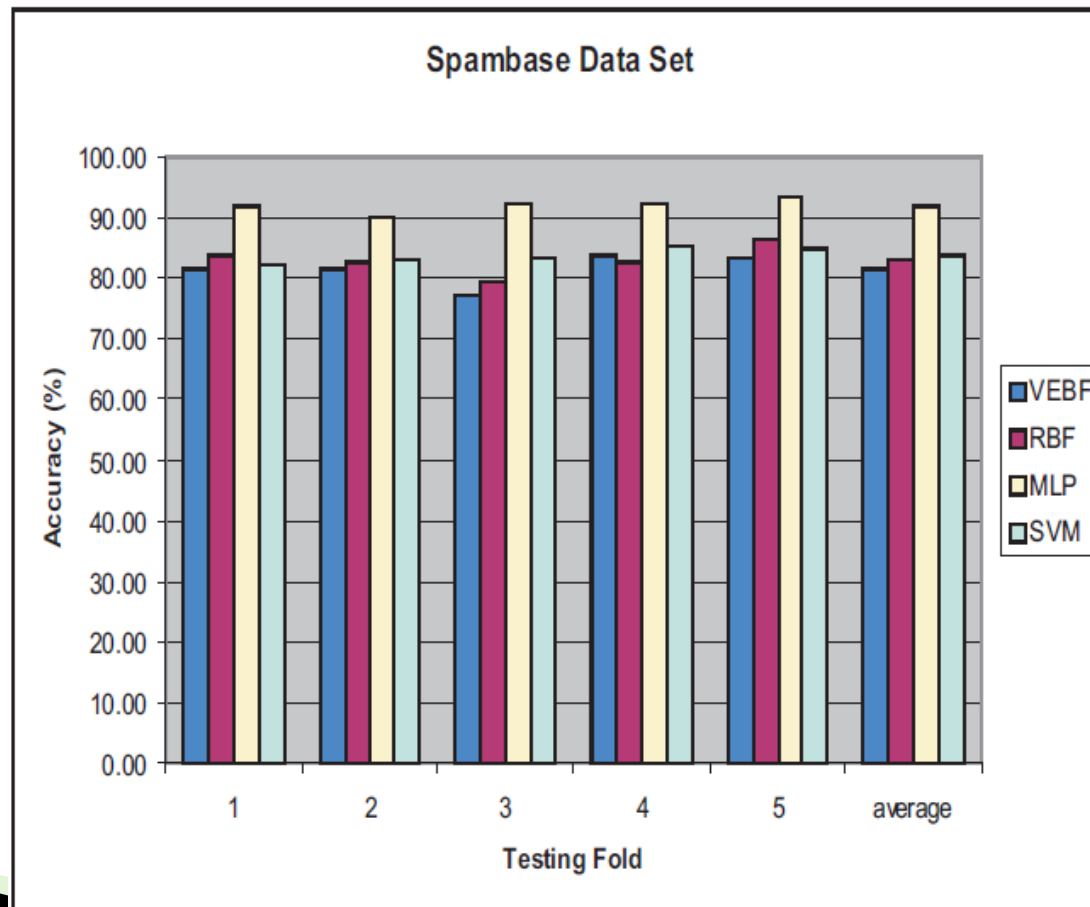
Experimental Results (Two-Class Classification)

▶ Heart Data Set



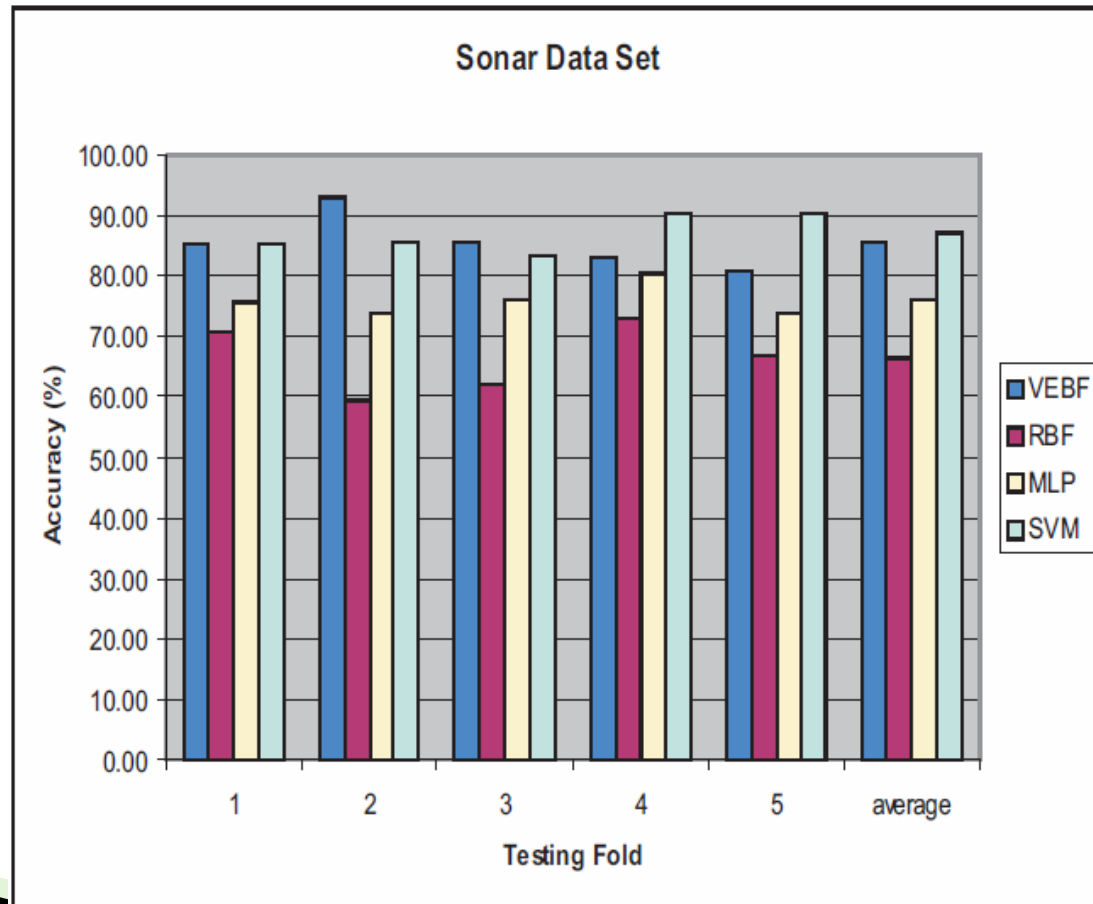
Experimental Results (Two-Class Classification)

► Spambase Data Set



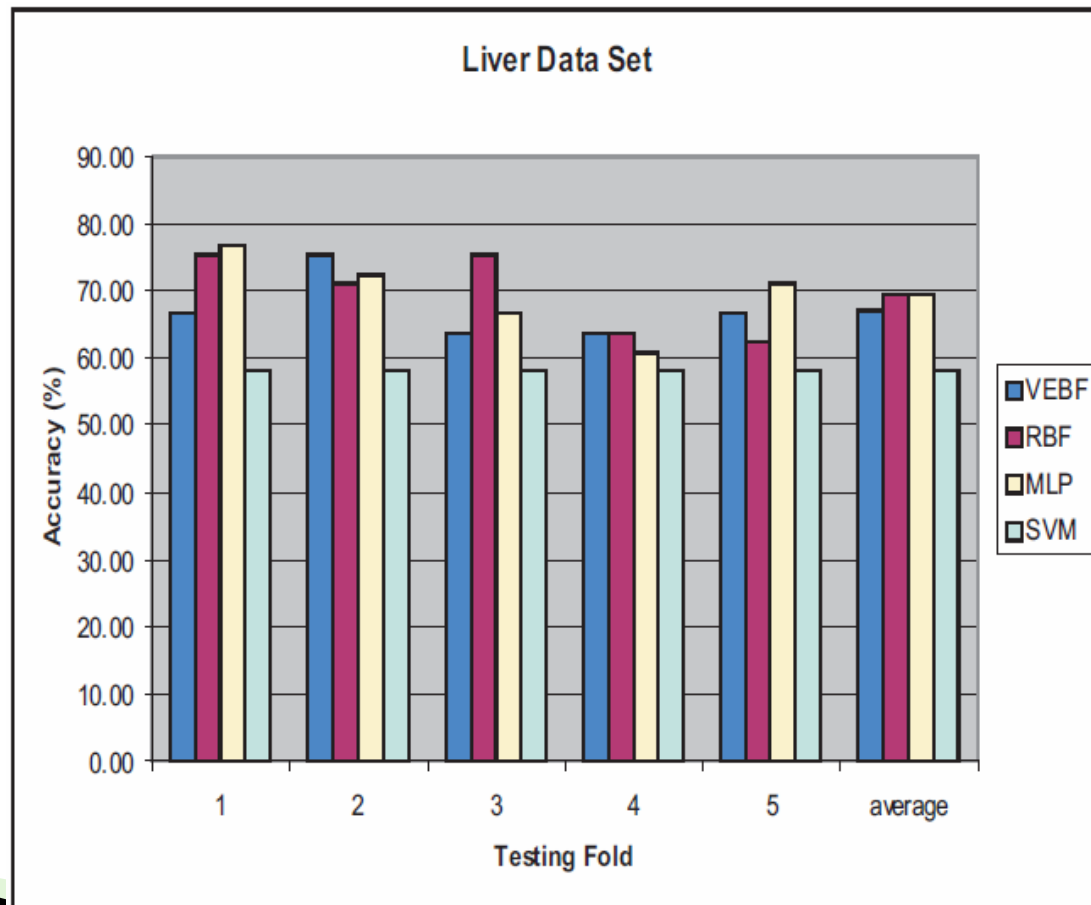
Experimental Results (Two-Class Classification)

► Sonar Data Set



Experimental Results (Two-Class Classification)

► Liver Data Set



Conclusion

- ▶ In this paper, a versatile hyper-ellipsoidal basis function for function approximation in high dimensional space is proposed.
- ▶ The basis function can be translated and rotated to cover the input data in high dimensional space depend upon the distribution of the data set in the high dimensional data space.

Conclusion

- ▶ The performance of proposed learning algorithm has been compared with other sequential well known learning algorithms such as RBF, MRAN, and EMRAN on three real world problems in the function approximation area.
- ▶ The results indicate that the average accuracy of the testing set of the proposed model is better than the other models.

Conclusion

- ▶ A very fast one-pass-throw-away learning algorithm based on hyper-ellipsoid function and VEBF neural network is proposed.
- ▶ This neural network trained by the proposed algorithm uses only one pass to learn a data set.
- ▶ the hidden neurons in the network can be merged into the new neuron whenever the condition is reached.
- ▶ From the preliminary results, we can see that the performance of the proposed learning is higher than RBF and MLP.



Thank You