

Efficient Training of RBF Networks Via the BYY Automated Model Selection Learning Algorithms

JIANWU MA, YI LI, and JIANWU AN*

Department of Mathematics, School of Mathematical Sciences, Peking University
Beijing 100871, P. R. China
jwma@math.pku.edu.cn

Abstract.

Efficient training of RBF networks is a challenging task. In this paper, we propose a novel algorithm for training RBF networks, which is based on the BYY automated model selection learning algorithm. The proposed algorithm is able to automatically select the optimal number of hidden nodes and the optimal kernel function for the RBF network. The experimental results show that the proposed algorithm is more efficient and accurate than the traditional algorithms. The proposed algorithm is able to handle large-scale data sets and can be applied to various applications.

1 Introduction

RBF networks have been widely used in many applications, such as function approximation, pattern recognition, and signal processing. However, the training of RBF networks is a non-linear and non-convex optimization problem, which is often difficult to solve. In this paper, we propose a novel algorithm for training RBF networks, which is based on the BYY automated model selection learning algorithm. The proposed algorithm is able to automatically select the optimal number of hidden nodes and the optimal kernel function for the RBF network. The experimental results show that the proposed algorithm is more efficient and accurate than the traditional algorithms. The proposed algorithm is able to handle large-scale data sets and can be applied to various applications.

*Corresponding author.

1.3.1.2.1.1. Adaptive Gradient Learning

In this section, we consider the following scenario: a function $f: X \rightarrow Y$ is given, where $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}^m$. The input x is a vector in \mathbb{R}^d , and the output y is a vector in \mathbb{R}^m . The function f is assumed to be smooth and invertible. The goal is to learn the function f from a set of training samples $\{(x_i, y_i)\}_{i=1}^N$, where $x_i \in X$ and $y_i \in Y$. The learning is performed using an adaptive gradient method. The training samples are generated by a process that involves a set of weights w and a set of parameters k . The weights w are used to weight the training samples, and the parameters k are used to control the learning process. The adaptive gradient method is designed to efficiently learn the function f by adapting the learning rate to the local curvature of the function. The method is based on the following principle: the learning rate should be proportional to the inverse of the local curvature of the function. This ensures that the learning process is stable and convergent. The adaptive gradient method is implemented using the following algorithm:

2 BYY-AMS Adaptive Gradient Learning Algorithm

In this section, we describe the BYY-AMS Adaptive Gradient Learning Algorithm. The algorithm is based on the following principle: the learning process is performed by minimizing the Kullback-Leibler (KL) divergence between the distribution of the training samples and the distribution of the function output. The KL divergence is a measure of the difference between two probability distributions. In this context, the KL divergence is used to measure the difference between the distribution of the training samples and the distribution of the function output. The KL divergence is defined as follows:

$$D_x = \sum_{i=1}^N x_i |_{t=1}^N$$

The KL divergence is used to define the learning objective function. The learning process is performed by minimizing the KL divergence between the distribution of the training samples and the distribution of the function output. The KL divergence is a measure of the difference between two probability distributions. In this context, the KL divergence is used to measure the difference between the distribution of the training samples and the distribution of the function output. The KL divergence is defined as follows:

$p, y, x' \pm p, x' \pm q, x, y' \pm q, y'$...

$$H(p, q) = \int p(y, x) p(x) \dots q(x, y) q(y) dx dy - z_q$$

z_q ...

$p(y, x) = q(x, y) \dots$

$$\alpha_j \geq \sum_{j=1}^K \alpha_j = z_q$$

$$p(x) = \frac{1}{N} \sum_{t=1}^N \delta(x - x_t)$$

$$p(y = j, x) = p(j, x) = \frac{\alpha_j q(x, \theta_j)}{q(x, \Theta_K)}$$

$$U_j, x_t = \alpha_j q, x, m_j \Sigma_j^{-1} \quad j = 1, \dots, K \quad J, \Theta_{K'} \quad \text{W}$$

$$J, \Theta_{K'} = \frac{1}{N} \sum_{t=1}^N J_{t'} \Theta_{K'} + J_{t'} \Theta_{K'} = \sum_{j=1}^K \frac{U_j, x_{t'}}{\sum_{i=1}^K U_i, x_{t'}} \cdot U_j, x_{t'} \quad \text{W}$$

$$J, \Theta_{K'} \quad \text{W} \quad \beta_j + m_j \quad B_j \quad \text{W}$$

$$\frac{\partial J_{t'} \Theta_{K'}}{\partial \beta_j} = \frac{1}{q, x_{t'}, \Theta_{K'}} \sum_{i=1}^K \lambda_{i, t'} \delta_{ij} - \alpha_{j'} U_i, x_{t'} \quad \text{W}$$

$$\frac{\partial J_{t'} \Theta_{K'}}{\partial m_j} = p, j, x_{t'} \lambda_{j, t'} \Sigma_j^{-1} x_t - m_{j'} \quad \text{W}$$

$$\text{vec} \frac{\partial J_{t'} \Theta_{K'}}{\partial B_j} = \frac{\partial B_j B_j^T}{\partial B_j} \text{vec} \frac{\partial J_{t'} \Theta_{K'}}{\partial \Sigma_j} \quad \text{W}$$

$$\delta_{ij} \quad \text{vec} A \quad \text{W}$$

$$\lambda_{i, t'} = - \sum_{l=1}^K p, l, x_{t'} - \delta_{il} - \alpha_l q, x_t, m_l \Sigma_l^{-1} \quad \text{W}$$

$$\frac{\partial J_{t'} \Theta_{K'}}{\partial \Sigma_j} = - p, j, x_{t'} \lambda_{j, t'} \Sigma_j^{-1} x_t - m_{j'} x_t - m_{j'}^T \Sigma_j^{-1} - \Sigma_j^{-1} \quad \text{W}$$

$$\frac{\partial BB^T}{\partial B} = I_{d \times d} \otimes B^T + E_{d \times d} \cdot B^T \otimes I_{d \times d} \quad \text{W}$$

$$\otimes \quad \text{W}$$

$$E_{d \times d} = \frac{\partial B^T}{\partial B} = \Gamma_{ij'}_{d \times d} = \begin{pmatrix} \Gamma_{11} & \dots & \Gamma_{1d} \\ \vdots & \ddots & \vdots \\ \Gamma_{d1} & \dots & \Gamma_{dd} \end{pmatrix}_{d \times d} \quad \text{W}$$

$$\Gamma_{ij} \quad \text{W} \quad j \text{th} \quad \text{W}$$

$$\frac{\partial BB^T}{\partial B} \quad \text{W}$$

$$\text{vec} \frac{\partial J_c \Theta_{k'}}{\partial B_j} = -p_j \cdot x_{t'} \lambda_j \cdot t' \cdot I_{d \times d} \otimes B_{d \times d}^T + E_{d \times d} \cdot B_{d \times d}^T \otimes I_{d \times d} \\ \times \text{vec} \Sigma_j^-, x_{t'} - m_{j'} \cdot x_{t'} - m_{j'}^T \Sigma_j^- - \Sigma_j^- \cdot x_{t'} - m_{j'}$$

$$\Delta \beta_j = \frac{\eta}{q \cdot x_{t'} \cdot \Theta_{k'}} \sum_{i=1}^K \lambda_i \cdot t' \cdot \delta_{ij} - \alpha_{j'} U_{i'} \cdot x_{t'}$$

$$\Delta m_j = \eta p_j \cdot x_{t'} \lambda_j \cdot t' \Sigma_j^- \cdot x_{t'} - m_{j'}$$

$$\Delta \text{vec} B_j = \frac{\eta}{q} p_j \cdot x_{t'} \lambda_j \cdot t' \cdot I_{d \times d} \otimes B_{d \times d}^T + E_{d \times d} \cdot B_{d \times d}^T \otimes I_{d \times d} \\ \times \text{vec} \Sigma_j^-, x_{t'} - m_{j'} \cdot x_{t'} - m_{j'}^T \Sigma_j^- - \Sigma_j^- \cdot x_{t'} - m_{j'}$$

Learning rate η is a scalar value that controls the step size of the updates. It is typically chosen to be small, e.g., $\eta = 0.01$ or $\eta = 0.001$.

$$\lim_{n \rightarrow \infty} \eta \cdot n = \text{const} \quad \sum_{n=1}^{\infty} \eta \cdot n = \infty$$

The learning rate η is a scalar value that controls the step size of the updates. It is typically chosen to be small, e.g., $\eta = 0.01$ or $\eta = 0.001$.

The learning rate η is a scalar value that controls the step size of the updates. It is typically chosen to be small, e.g., $\eta = 0.01$ or $\eta = 0.001$.

3 Training of the RBF Network

The training of the RBF network involves finding the optimal weights w_{ij} and biases b_j for the hidden units. This is done by minimizing the error function J with respect to the weights and biases.

$$y_j \cdot x_t = \sum_{i=1}^n w_{ij} \phi_i \cdot x_t$$

2.2.2. The Gaussian Mixture Model

Let $\mathbf{W} = \{w_{ij}\}$ be a matrix of weights, where w_{ij} is the weight of the j^{th} component in the i^{th} class. Let $\phi_j(x)$ be the Gaussian density function for the j^{th} component, and let λ_j be the weight of the j^{th} component in the mixture.

$$\phi_j(x) = \frac{1}{\sigma_j} \exp\left(-\frac{x - m_j}{\sigma_j}\right)$$

Let m_j and σ_j be the mean and standard deviation of the j^{th} component, respectively. Let λ_j be the weight of the j^{th} component in the mixture. Let $y(x)$ be the output of the mixture model, and let $f(x)$ be the target function.

$$y(x) = \sum_{j=1}^n \lambda_j \phi_j(x) = \sum_{j=1}^n \lambda_j \frac{1}{\sigma_j} \exp\left(-\frac{x - m_j}{\sigma_j}\right)$$

Let λ_j be the weight of the j^{th} component in the mixture. Let m_j and σ_j be the mean and standard deviation of the j^{th} component, respectively. Let λ_j be the weight of the j^{th} component in the mixture. Let $y(x)$ be the output of the mixture model, and let $f(x)$ be the target function.

$$D_{x,y} = \sum_{i=1}^N |x_i - y_i| \quad i = 1, \dots, N$$

$$\begin{aligned} E &= - \sum_{i=1}^N y_i - f(x_i) = - \sum_{i=1}^N y_i - \sum_{j=1}^n \lambda_j \phi_j(x_i) \\ &= - \sum_{i=1}^N y_i - \sum_{j=1}^n \lambda_j \frac{1}{\sigma_j} \exp\left(-\frac{|x_i - m_j|}{\sigma_j}\right) \end{aligned}$$

Let η_λ , η_m , and η_σ be the learning rates for the weights, means, and standard deviations, respectively. Let $\Delta \lambda_j$, Δm_j , and $\Delta \sigma_j$ be the updates to the weights, means, and standard deviations, respectively.

$$\begin{cases} \Delta \lambda_j = \eta_\lambda \sum_{i=1}^N y_i - \sum_{l=1}^n \lambda_l \phi_l(x_i) \phi_j(x_i) \\ \Delta m_j = \eta_m \sum_{i=1}^N y_i - \sum_{l=1}^n \lambda_l \phi_l(x_i) \phi_j(x_i) (x_i - m_j) \lambda_j \sigma_j \\ \Delta \sigma_j = \eta_\sigma \sum_{i=1}^N y_i - \sum_{l=1}^n \lambda_l \phi_l(x_i) \phi_j(x_i) (x_i - m_j)^T (x_i - m_j) \lambda_j \sigma_j \end{cases}$$

Let η_λ , η_m , and η_σ be the learning rates for the weights, means, and standard deviations, respectively. Let λ_j be the weight of the j^{th} component in the mixture. Let m_j and σ_j be the mean and standard deviation of the j^{th} component, respectively.

σ_j^w is the variance of $x_t - m_j$ for $t \in C_j$. The variance of $x_t - m_j$ is given by $\sigma_j^w = \frac{1}{N_j} \sum_{t \in C_j} (x_t - m_j)^2$. The variance of $x_t - m_j$ is given by $\sigma_j^w = \frac{1}{N_j} \sum_{t \in C_j} (x_t - m_j)^2$. The variance of $x_t - m_j$ is given by $\sigma_j^w = \frac{1}{N_j} \sum_{t \in C_j} (x_t - m_j)^2$.

$$\sigma_j = \frac{1}{N_j} \sum_{t \in C_j} (x_t - m_j)^2$$

σ_j^w is the variance of $x_t - m_j$ for $t \in C_j$. The variance of $x_t - m_j$ is given by $\sigma_j^w = \frac{1}{N_j} \sum_{t \in C_j} (x_t - m_j)^2$. The variance of $x_t - m_j$ is given by $\sigma_j^w = \frac{1}{N_j} \sum_{t \in C_j} (x_t - m_j)^2$. The variance of $x_t - m_j$ is given by $\sigma_j^w = \frac{1}{N_j} \sum_{t \in C_j} (x_t - m_j)^2$.

4 Experiment Results

The results of the experiment are shown in Figure 1. The results of the experiment are shown in Figure 1. The results of the experiment are shown in Figure 1.

4.1 On the Noisy XOR Problem

The results of the experiment are shown in Figure 2. The results of the experiment are shown in Figure 2. The results of the experiment are shown in Figure 2.

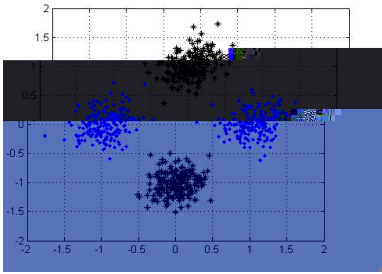


Fig. 1. Scatter plot of data points X



Fig. 2. Time series plot of y^w

The data points are clustered into two main regions: a blue region at the bottom and a black region at the top. There are also some black points scattered in the middle region.

4.2 On the Mackey-Glass Time Series Prediction

The Mackey-Glass time series is defined by the following differential equation:

$$\dot{x}(t) = -bx(t) + \frac{ax(t - \tau)}{1 + x(t - \tau)},$$

where a , b , and τ are parameters. The time series is highly oscillatory and noisy. The prediction error is defined as $y_i = f(x_i) - x_i$.

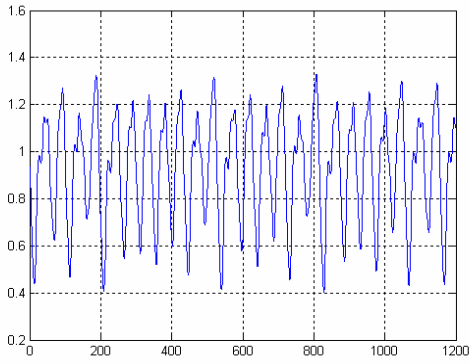


Fig. 3. \sqrt{W} ...

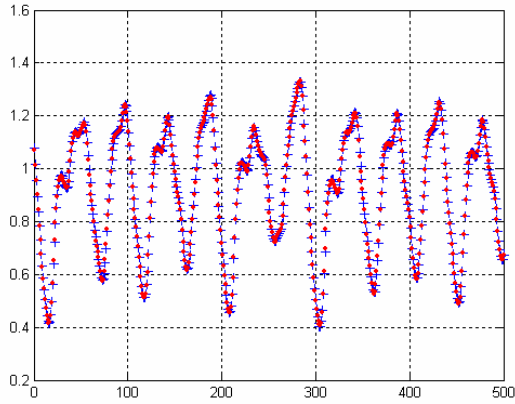


Fig. 4. \sqrt{W} ...

5 Conclusions

The analysis of \sqrt{W} ...

References

1. S. H. W. ... 2 ... //

1. ... //

W. S. ... 247 ... //

3

E. ... S ... S ... W ... W ... 13

S ... S ... S ... W ... 42

W. Y.

X ... E ... W ... 4

...

... EEE ... S ... 36,

E ... EEE ... W ... //

W ... EEE ... 7 ... //

S ... E ... W ... 34 ... //

X ... EEE ... 16 ... //

X ... Y ... Y ... S ... W ... //

... (2 ... //

X ... S ... Y ... Y ... //

X ... S ... S ... //

X ... S ... 11 ... //

W ... S ... Y ... 56 ... //

Y ... Y ... W ... S ... 19

Y ... 24 ... //

S ... S ... 22 ... //