2005

A neural network with localized receptive fields for visual pattern classification

Son Lam Phung
University of Wollongong, phung@uow.edu.au

Abdesselam Bouzerdoum
University of Wollongong, bouzer@uow.edu.au

A neural network with localized receptive fields for visual pattern classification

Abstract

We introduce a new neural network for 2D pattern classification. The new neural network, termed as localized receptive field neural network (RFNet), consists of a receptive field layer for 2D feature extraction, followed by one or more 1D feedforward layers for feature classification. All synaptic weights and biases in the network are automatically determined through supervised training. In this paper, we derive five different training methods for the RFNet, namely gradient descent, gradient descent with momentum, resilient backpropagation, Polak-Ribiere conjugate gradient, and Levenberg-Marquadrt algorithm. We apply the RFNet to classify face and nonface patterns, and study the performances of the training algorithms and the RFNet classifier in this context.

Disciplines

Physical Sciences and Mathematics

Publication Details

Phung, S. & Bouzerdoum, A. (2005). A neural network with localized receptive fields for visual pattern classification. In Suvisoft (Eds.), The Eight International Symposium on Signal Processing and Its Applications (pp. 94-97). USA: IEEE.

A NEURAL NETWORK WITH LOCALIZED RECEPTIVE FIELDS FOR VISUAL PATTERN CLASSIFICATION

Son Lam Phung and Abdesselam Bouzerdoum

School of Electrical, Computer and Telecommunications Engineering
University of Wollongong
Northfields Av, Wollongong, NSW 2522, Australia
E-mails: phung@uow.edu.au, salim@uow.edu.au

ABSTRACT

We introduce a new neural network for 2D pattern classification. The new neural network, termed as *localized receptive field neural network* (RFNet), consists of a receptive field layer for 2D feature extraction, followed by one or more 1D feedforward layers for feature classification. All synaptic weights and biases in the network are automatically determined through supervised training. In this paper, we derive five different training methods for the RFNet, namely gradient descent, gradient descent with momentum, resilient backpropagation, Polak-Ribiere conjugate gradient, and Levenberg-Marquadrt algorithm. We apply the RFNet to classify face and nonface patterns, and study the performances of the training algorithms and the RFNet classifier in this context.

1. INTRODUCTION

Artificial neural networks have proven to be a powerful computational tool for many tasks: pattern classification, data clustering, function approximation, data compression, to name a few. The growing computation powers have supported new and complex network architectures for solving difficult cognitive tasks. In this paper, we develop a new neural network architecture, known as the receptive field neural net (RFNet) that is specifically designed for the classification of visual patterns. The new neural network integrates a 2D feature extraction layer and feature classification layers. Also in this paper, we devise efficient training methods for the RFNet and apply the network to detect human faces in digital photos.

This paper is organized as follows. In section 2, we describe the network model of the new neural network. In section 3, we derive algorithms for supervised training of the RFNet. In section 4, we analyze the performances of RFNet training algorithms and RFNet classifier in the context of face and nonface classification, and section 5 is the conclusion.

2. RFNET NETWORK MODEL

The network model of the RFNet is illustrated in Fig. 1. An RFNet contains a 2D feature extraction layer followed by one or more 1D layers. The 2D feature extraction layer

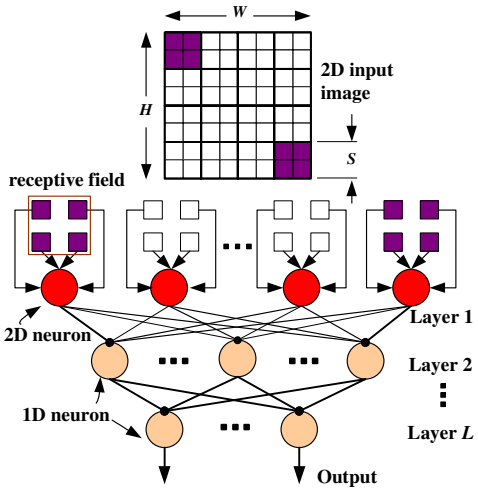

Fig. 1. Receptive field neural network model.

consists of an array of neurons, each handles a square region of the input image. Each 2D neuron is connected to pixels in an image region through synaptic weights. The neuron is localized to the image region and acts as a 2D feature extractor by applying a nonlinear activation function on the weighted sum of its input pixels. Each remaining layer is a feedforward layer that consists of several neurons arranged in 1D. An 1D neuron applies an activation on the weighted sum of its inputs, and the neuron's output is sent to neurons in the next 1D layer. The outputs of the last layer are taken as the network outputs.

Suppose we need to process an image $\mathbf{X}$ of size $H \times W$ pixels. The input image is divided into N_1 non-overlapping square regions, where $N_1 = \frac{H \times W}{S^2}$ and S is the region width. Let $\mathbf{x}_i$ be the pixels in image region i, and $\mathbf{w}_i^1$ be the synaptic weights to the region. Let f_l be the activation function of layer l. The output of 2D neuron i in layer 1 is computed as

$$y_i^1 = f_1(\mathbf{w}_i^1 \, \mathbf{x}_i + b_i^1) \tag{1}$$

where b_i^1 is the bias of the 2D neuron.

For the 1D feedforward layers, let $\mathbf{w}_i^l$ and b_i^l be the respective synaptic weights to and the bias of neuron i in layer l, where $l > 1$. Let N_l be the number of neurons in

layer l. The output neuron i in layer l is computed as :

$$y_i^l = f_l(\mathbf{w}_i^l\ \mathbf{y}^{l-1} + b_i^l) : \tag{2}$$

The network outputs are the outputs $\mathbf{y}^L$ of the last layer, where L is the number of layers.

Compared to convolutional neural network [1], the new RFNet is different in that each image region is assigned to a different receptive field. This allows image features with strong spatial dependency to be extracted. Compared to the neural network used for face detection by Rowley et al. [5], our RFNet has non-overlapping receptive fields that are fully connected to the next 1D feedforward layer. Therefore, the RFNet's connection scheme is more systematic, which allows training algorithms to be devised and the network to be applied in diverse image analysis tasks.

3. TRAINING ALGORITHMS FOR RFNET

The RFNet can be designed to approximate a given input-output mapping when a training set of input samples and corresponding target outputs is available. Network training is formulated as an optimization problem in which the objective is to minimize the mean-square-error (MSE) between the actual and desired outputs, through systematic modification of network weights and biases. There are numerous optimization algorithms that can be used for neural network training. In this paper, we focus on five algorithms that are commonly used.

Let $\{\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^K\}$ be the input samples in the training set, and $\{\mathbf{d}^1, \mathbf{d}^2, ..., \mathbf{d}^K\}$ be the desired outputs. The error function is defined as

$$E = \frac{1}{K \times N_L} \sum_{k=1}^{K} ||\mathbf{e}^k||^2 \tag{3}$$

where $\mathbf{e}^k = \mathbf{y}^{L,k} - \mathbf{d}^k$ is the error for training sample k. The mean-square-error is treated as a function $E(\mathbf{w})$ of all network weights and biases.

3.1. Gradient Descent Algorithm (GD)

At each epoch, the network parameters are updated along the direction of the steepest descent:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t), \tag{4}$$

where $\Delta\mathbf{w}(t) = -\alpha\nabla\mathbf{E}(t)$, $\nabla\mathbf{E}$ is the error gradient, t is the epoch number, and α is a scalar learning rate.

We now derive the error gradient $\nabla\mathbf{E}$ for the RFNet. Let $\delta_i^{l,k}$ be the error sensitivity of neuron i in layer l for input sample $\mathbf{x}^k$. It is defined as

$$\delta_i^{l,k} = \frac{\partial E}{\partial y_i^{l,k}} \tag{5}$$

The error sensitivities can be computed using the error back-propagation technique:

- For layer L and $i = 1, 2, ..., N_L$:

$$\delta_i^{L,k} = \frac{2}{K \times N_L}\ e_i^k \tag{6}$$

- For layer l, where $1 \le l < L$, and $i = 1, 2, ..., N_l$:

$$\delta_i^{l,k} = \sum_{j=1}^{N_{l+1}} \delta_j^{l+1,k}\ w_{j,i}^{l+1}\ f_{l+1}^{'}(\mathbf{w}_j^{l+1}\ \mathbf{y}^{l,k} + b_j^{l+1}) \tag{7}$$

Equation (7) shows that the error sensitivity of a neuron is the weighted sum of the error sensitivities that are propagated backwards from the subsequent layer.

Once the error sensitivities are computed, the gradient can be obtained through the chain rule of differentiation:

- For 1D layer l, where $2 \le l \le L$, $i = 1, 2, ..., N_l$ and $j = 1, 2, ..., N_{l-1}$:

$$\frac{\partial E}{\partial w_{i,j}^l} = \sum_{k=1}^{K} \delta_i^{l,k}\ f_l^{'}(\mathbf{w}_i^l\ \mathbf{y}^{l-1,k} + b_i^l)\ y_j^{l-1,k} \tag{8}$$

$$\frac{\partial E}{\partial b_i^l} = \sum_{k=1}^{K} \delta_i^{l,k}\ f_l^{'}(\mathbf{w}_i^l\ \mathbf{y}^{l-1,k} + b_i^l) \tag{9}$$

- For the 2D layer (layer 1), $i = 1, 2, ..., N_1$, and $j = 1, 2, ..., S^2$:

$$\frac{\partial E}{\partial w_{i,j}^1} = \sum_{k=1}^{K} \delta_i^{1,k}\ f_1^{'}(\mathbf{w}_i^1\ \mathbf{x}_i^k + b_i^1)\ x_{i,j}^k \tag{10}$$

$$\frac{\partial E}{\partial b_i^1} = \sum_{k=1}^{K} \delta_i^{1,k}\ f_1^{'}(\mathbf{w}_i^1\ \mathbf{x}_i^k + b_i^1) \tag{11}$$

3.2. Gradient Descent with Momentum Algorithm (GDMV)

The GDMV algorithm specifies the weight update as

$$\Delta\mathbf{w}(t) = \lambda\ \Delta\mathbf{w}(t-1) - (1-\lambda)\ \alpha(t)\ \nabla\mathbf{E}(t) \tag{12}$$

where λ is the momentum parameter, $0 < \lambda < 1$. The role of the momentum term in (12) is to maintain the general trend of weight update. At each epoch, if the error E increases by some factor, the learning rate $\alpha(t)$ is reduced. Vice versa, if E decreases by some factor, the learning rate is increased. This variable learning rate strategy leads to faster training in "flat" regions of the error surface.

3.3. Resilient Back-propagation Algorithm (RPROP)

Proposed by Riedmiller and Braun [4], the RPROP algorithm uses only the sign of the gradient to determine the direction of weight update; the gradient magnitude is discarded. The RPROP algorithm is designed to address the slow speed of GD training when the gradient magnitude becomes small. The weight update of the RPROP algorithm is given by

$$\Delta w_i(t) = \begin{cases} -\Delta_i(t), & \text{if } \frac{\partial E}{\partial w_i}(t) > 0 \\ +\Delta_i(t), & \text{if } \frac{\partial E}{\partial w_i}(t) < 0 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where $\Delta_i(t)$ is the adaptive learning rate for network parameter w_i,

$$\Delta_i(t) = \begin{cases} \eta_{\text{inc}}\ \Delta_i(t-1), & \text{if } \frac{\partial E}{\partial w_i}(t)\ \frac{\partial E}{\partial w_i}(t-1) > 0 \\ \eta_{\text{dec}}\ \Delta_i(t-1), & \text{if } \frac{\partial E}{\partial w_i}(t)\ \frac{\partial E}{\partial w_i}(t-1) < 0 \\ \Delta_i(t-1), & \text{otherwise} \end{cases} \tag{14}$$

In (14), $\eta_{\text{inc}} > 1$ and $0 < \eta_{\text{dec}} < 1$ are two scalar terms. All adaptive learning rates are initialized to the same value at training start.

3.4. Conjugate Gradient Algorithm (CG)

Conjugate gradient algorithms update weights along a search direction $\mathbf{s}(t)$, which is a combination of the previous direction and the negative gradient:

$$\mathbf{s}(t) = \begin{cases} -\nabla\mathbf{E}(1) & \text{for } t = 1 \\ -\nabla\mathbf{E}(t) + \beta(t)\ \mathbf{s}(t-1) & \text{for } t > 1 \end{cases} \tag{15}$$

where $\beta(t)$ is a scalar. In our work, the Polak-Ribiere update for $\beta(t)$ is used [2]:

$$\beta(t) = \frac{[\nabla\mathbf{E}(t) - \nabla\mathbf{E}(t-1)]^T\ \nabla\mathbf{E}(t)}{\|\nabla\mathbf{E}(t-1)\|^2} \tag{16}$$

The search direction is reset to the negative gradient after every P epochs, where P is the number of weights. The CG weight update is given as

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \alpha(t)\ \mathbf{s}(t) \tag{17}$$

where $\alpha(t)$ is found through a line search. We use the Charalambous line search method [2], which is based on cubic interpolation and bracketing of the minimum.

3.5. Levenberg-Marquardt Algorithm (LM)

The Levenberg-Marquardt algorithm is based on a second-order approximation of the error function. It has been applied to train the multilayer perceptron by Hagan and Menhaj [3]. The weight update rule of the Levenberg-Marquardt algorithm can be written as

$$\Delta\mathbf{w}(t) = -[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\ \mathbf{J}^T\mathbf{e} \tag{18}$$

where

- $\mathbf{J}$ is the Jacobian matrix that is made up from derivatives of the error e_i^k ($i = 1, 2, ..., N_L$ and $k = 1, 2, ..., K$) w.r.t. each network parameter.
- $\mathbf{e}$ is a column vector consisting of all errors e_i^k, $i = 1, 2, ..., N_L$ and $k = 1, 2, ..., K$.
- μ is a regularization parameter. When μ is large, the weight update rule is similar to the gradient descent method. When μ is small, the weight update is similar to the Newton method.

Computation of the Jacobian matrix for the RFNet is similar to computation of the gradient $\nabla\mathbf{E}$ in (5)-(11). The only major modification needed is the definition of the error sensitivities in (5).

4. EXPERIMENTS AND ANALYSIS

We analyze the RFNet and its training algorithms in the context of face and nonface classification. Face and nonface classification is used in detecting the presence and the location of human faces in digital images [1, 5]. Usually, windows of the input image are scanned exhaustively to determine if each window is a face or nonface pattern. Obviously, there are two main aspects in this face detection approach: (i) the classifier for differentiating face and nonface patterns, and (ii) the post-processing strategy for combining the results of window-scanning. Since heuristics used in post-processing can influence the outcome of face detection, in this paper we will focus only on the face-nonface classification aspect of the RFNet. The neural network classifier will be trained and evaluated on a large dataset of face and nonface patterns.

4.1. RFNet face-nonface classifier design

The RFNet is designed to process an input window of size 16×16 pixels in gray scale. The input window is divided by 256 to bring pixel values to the range of [0,1). We design a RFNet that have three layers:

- a 2D layer with receptive fields of size 2×2 pixels,
- a hidden 1D layer with 2 neurons
- an output 1D layer with one neuron.

We select the configuration $(16, 2)$ after analyzing several possibilities, including $(18, 3)$, $(20, 2)$,$(20, 4)$, $(21, 3)$, $(24, 4)$, $(24, 6)$, $(27, 3)$,$(28, 4)$,$(30, 5)$, and $(32, 4)$. The total number of network weights and biases is 453. The activation function of the output layer is the *hyperbolic tangent*. The activation functions of other layers are the *logistic sigmoid*. The RFNet is trained to produce outputs of 0.9 and 0.1 for face and nonface pattern, respectively.

Our experiments used a dataset of 20,000 patterns, of which 10,000 face patterns were manually cropped from web images and 10,000 nonface patterns were randomly extracted from 2,000 photos that contain no human face. The five-fold cross validation technique was used to compare training algorithm performances and estimate network generalization ability. The entire dataset was divided into five subsets. For each run, ten RFNets were trained on a combination of four subsets using five training algorithms, and then tested on the remaining fifth subset. Performance measures were averaged over the five validation runs.

4.2. Comparison of RFNet training algorithms

The time evolution of the training MSE, produced by the five training algorithms, is shown in Fig. 2. To achieve a machine-independent comparison, we measure time in terms of $gdeu$, which is defined as the average time taken to complete one GD epoch. The $gdeu$ time unit remains stable for a fixed-size network and a fixed-size training set.

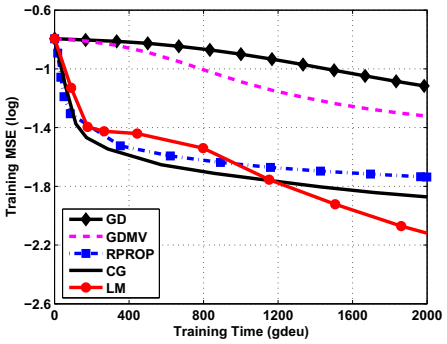

Fig. 2. Time evolution of the training mean-square-error.

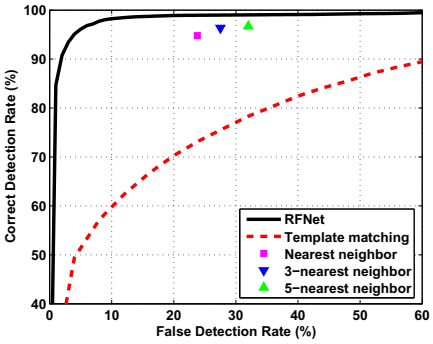

Fig. 3. Face-nonface classification ROC curves of the RFNet, template matching, and k-NN classifiers.

Table 1. Memory usage of five RFNet training algorithms

Algorithm	Memory Usage
GD	$K \times P$
GDMV	$K \times P$
RPROP	$(K+2) \times P$
CG	$(K+3) \times P$
LM	$K \times P \times N_L + P^2$

Figure 2 shows that all four algorithms GDMV, RPROP, CG and LM were faster than the standard GD algorithm. We found that the GDMV algorithm achieved a significant speed advantage over the GD algorithm only if the momentum parameter λ was within the range of $(0.6, 0.9)$. Furthermore, imposing an upper limit on the adaptive learning rate $\alpha(t)$ made GDMV training more stable. The CG algorithm converged slightly faster than the RPROP algorithm; both algorithms had faster speeds than the GDMV algorithm. The LM algorithm became faster than the CG algorithm only after 1200 *gdeu*. As shown in Table 1, among the five algorithms, the LM algorithm requires the largest storage due to the computation of the Jacobian and Hessian matrices. The RPROP and CG algorithms require slightly more storage than the GD algorithm, but their convergence speeds on large training sets are comparable to that of the LM algorithm.

4.3. Analysis of RFNet face-nonface classifier

The face-nonface classification rates of the RFNet are shown in Fig. 3. The RFNets were trained using the CG algorithm for a maximum of $1,500$ epochs and a target MSE of 0.01. Estimated over the five validation runs, the RFNet achieved CDRs of 97.0% and 98.6% at FDRs of 5.0% and 10.0%, respectively. It had a maximum CR of 96.0%.

For comparison purposes, we implemented and tested two other face-nonface classifiers using the same data as for the RFNet: correlation-based template matching (TM) classifier, and k-nearest neighbor (k-NN) classifier. The RFNet clearly outperformed the TM classifier, which had CDRs of 51.9% and 60.1% at FDRs of 5.0% and 10.0%, respectively. The maximum CR of the TM classifier was only 75.3%. Compared to the TM classifier, the RFNet consists of several trainable 2D templates (ie. receptive fields) that are compared to localized image regions. The RFNet classifier is also more accurate than the nonlinear k-NN classifier. The overall classification rates of the 1-NN, 3-NN and 5-NN classifiers were 85.5%, 84.4%, and 82.4%, respectively. The k-NN classifier in our implementation required about 390 times more storage than the RFNet classifer.

5. CONCLUSION

We have introduced a new neural network in which neurons with receptive fields localized to image regions are trained as 2D feature extractors. The network also contains 1D feedforward layers for classification of the extracted 2D features. We derive five different network training algorithms, which have been shown capable of handling large training sets. Results on the face-nonface classification problem have shown that the new network is a promising tool for image classification.

6. REFERENCES

[1] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Trans. PAMI*, vol. 26, no. 11, pp. 1408–1423, 2004.

[2] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*. Boston, MA: PWS Publishing, 1996.

[3] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Trans. Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[4] M. Riedmiller and H. Braun, "A direct adaptive method of faster backpropagation learning: The rprop algorithm," in *IEEE Int. Conf. on Neural Networks*, San Francisco, 1993, pp. 586–591.

[5] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. PAMI*, vol. 20, no. 1, pp. 23–38, 1998.