Edge Detection Based on Modified BP Algorithm of ANN

Lihong Zheng
Taiyuan University of Technology
Taiyuan, Shanxi, China
lh--zheng@sohu.com

Xiangjian He
University of Sydney,Technology
PO Box 123 Broadway NSW 2007 Australia
sean@it.uts.edu.au

Abstract

Accurately locating edge points is important for image measurement. This paper describes an edge detection method based on back-propagation algorithm of neural networks. A three-layer structure of artificial neural network and a learning algorithm are given in detail. Through training and amendment, the weight values are determined finally, and some factors that may have influenced on system are discussed.

Keywords: Edge detection, Back-Propagation, Neuron, Artificial Neural Network

1 Introduction

Edges are characterized by sharp transition in grey levels generally. They are important in computer image processing. Many edge detection methods are based on the idea to find the large difference area of grey level as discussed in [1][2]. Edge detection is to locate position where changes of image grey values are large. The methods include gradient method, template matching method, region fitting method and so forth. In industry, since high resolution and precision have led to the use of very costly Charge-Coupled-Device (CCD) sensor, a lot of edge detection techniques with sub-pixel resolution have become well known. From the industrial point of view, sub-pixel edge locating method can improve measurement accuracy while breaking the limit of physical resolving power and lowering cost. With the sub-pixel technique, the closer to the center of sampling window, the less error it has [3]. So, locating edge point location plays a key role in measure system. The accuracy of measure result mainly depends on the accurate location of edge point. There are many kinds of noises induced in images. Noise reduction can be achieved effectively with a filter whose basic function is to compute the average gray levels in the neighborhood in which the filter is located. That is the grey level of the point of edge being replaced by the mean of grey levels of all points in its neighborhood. This is a simple and efficient method for image smoothing. But this method also blurs the edges that are undesirable. The blurry degree is in the proportion of the radius of neighborhood. Image noise is caused by many factors such as electronics, temperature shift, unevenness of light background and so forth. Generally speaking, noise is stochastic and ruleless. It is hard to know exact noise in an image. Due to the distributed nature ANN can be shown that they are universal function approximators. This character does benefit to noise removing in image processing. In this paper, we demonstrate an edge detection method based on an artificial neural network (ANN). The architecture of ANN, the learning algorithm and improvement of ANN are discussed in detail as follows.

2 Edge Detection Based on ANN

2.1 Architecture of ANN

Generally, neural network has multiple layers. The number of inputs to the network is constrained by the problem, and the number of neurons in the output layer is constrained by the number of outputs required by the problem. There are always one or more hidden layers of neurons. It is found that any function with a finite number of discontinuities can be well approximated by a three-layer neural network with sufficient neurons in the hidden layer. The architecture of a three-layer neural network is shown in Figure 1 below. The three layers are input layer, hidden layer and output layer. It is common for different layers to have different numbers of neurons. The network used in this paper has 8 neurons (g_j , j=1,2, …, 8) in the first layer, q neurons in the second layer - hidden layer, and one neuron in output layer. The outputs of each layer are the inputs to the next layer. Each neuron is associated with a number for the neuron's activation. Similarly, each connection in the network is associated with a weight. The activation functions used for neuron outputs can be either linear or non-linear. In general, they are non-linear.

Here, each neuron input is a grey value at an image pixel. For the output of a neuron at the centre of a 3×3 cluster of image pixels, 8 neurons are selected as inputs. Each input neuron is a surrounding pixel of the central pixel. The eight neurons are the inputs of the first layer—input layer. In the learning process, called *forward propagation*, input signal is processed through input layer, hidden layer and output layer. The status of neurons in every layer affects status of neurons in the next layer only. If there is an error between the desired output and the actual output in original connect way, that is to say, the desired output can not be obtained, then the learning process will change to the process, called *backward propagation*, which tends to feedback the error and adjust the weight values for each layer.

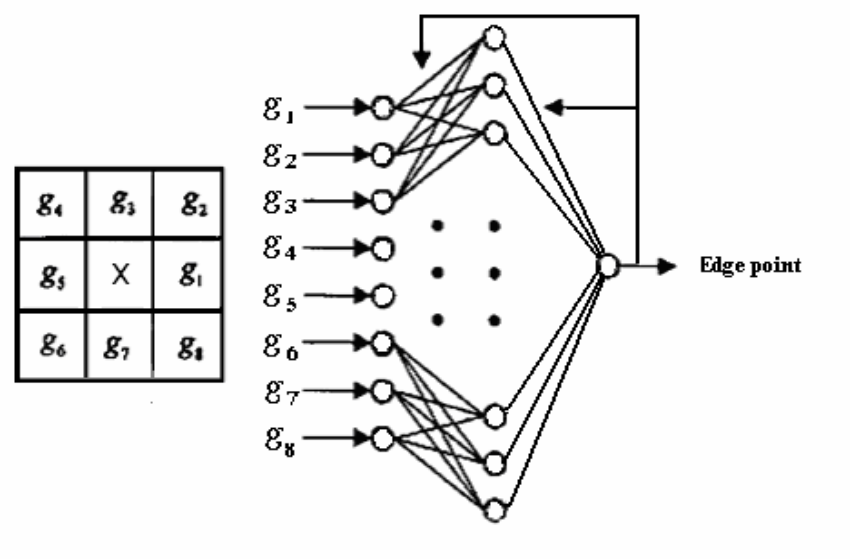

Fig. 1 Architecture of Neural Network

2.2 Back-propagation algorithm

The back-propagation algorithm [4][5] uses the gradient of the performance function to determine how to adjust the weights to minimize errors that affect performance. The simplest implementation of back-propagation updates the network weights and biases in the direction in which the performance function decreases most rapidly. This direction is opposite to the gradient direction.

In this paper, the activation function of each node uses a sigmoid function, i.e., $f_x = \frac{1}{1+e^{-x}}$,

which are characterized by the fact that its slope must approach zero as the input gets large. And therefore it is a continuous activation, mostly chosen from [0, 1] or [-1, +1].

In the following, w_{ij} represents the connection weight from node j of previous layer to node i of current layer, U^k_i represents the inputs sum of the i^{th} node in current layer k, and V^k_i is its output sum. Then, we have that

$$V_i^k = f(U_i^k),$$
$$U_i^k = \sum_j w_{ij} \cdot V_i^{k-1}.$$

If actual output of layer k is V^k and desired output is $\hat{y}$, then the error function can be represented as

$$E = \frac{1}{2}\sum_n (V_n^k - \hat{y}_n)^2.$$

The backpropagation algorithm adjusts the weights in the steepest descent direction (negative of the gradient). This is the direction in which the performance function is decreasing most rapidly. The update of wij can be obtained shown as follows.

$$\Delta w_{ij} \propto -\frac{\partial E}{\partial w_{ij}},$$

Let $e_i^k = \frac{\partial E}{\partial U_i^k}$.

So, $\Delta w_{ij}(t+1) = -\eta \cdot e_i^k \cdot V_j^{k-1} + \alpha \cdot \Delta w_{ij}(t)$,

$$w_{ij}(t+1) = w_{ij}(t) - \Delta w_{ij},$$

here α is a flat factor and belongs to (0, 1), η is the learning rate. Note that a large η can accelerate the learning speed.

2.3 Training

The training process requires a set of examples of proper network behavior that includes network inputs and target outputs. During the training, the weights and biases of the network are adjusted iteratively to optimize the network performance function. The performance function for the networks is mean square error (MSE) - the average squared error between the network output and the target output.

There are generally five steps in the training process:

Step 1. Preprocess data

To produce the most efficient training, it is helpful to preprocess the data before training. This means to scale the inputs and targets data to ensure they always fall within a specified range.

Step 2. Initialize the weight values.

Before training a network, the weights and biases must be initialized. In order to avoid the result located in flat area, random weight values are selected.

Step 3. For input sample, calculate outputs of arbitrary node in both hidden layer and output layer (in forward propagation).

Step 4. Calculate errors of hidden layer and output layer (in backward propagation).

Step 5. Adjust weight values according to error function until the minimal error is achieved.

After training the weights are fixed and can be used for practical edge detection.

2.4 Improvement of ANN

In order to improve the performance of ANN, four methods are applied. They are described as follows.

1 Fast processing

Considering an object of rectangular shape, in order to improve the performance of NN edge detector, four NN units for horizontal and vertical orientation are applied, that is to segment an image into four parts. By searching from the center point of CCD plane horizontally and vertically, four horizontal and vertical edge points can be found. The exact center point O can hence be obtained as a result. Based on point O, the image can now be segmented into four parts and the least groups of training data can be given to the neural network. The process speed can be fast by dealing with less discrepant region.

2 Selection of number of hidden neurons

Networks are very sensitive to the number of neurons in their hidden layers. Too few neurons can lead to underfitting. Too many neurons can contribute to overfitting. There is not a good method to ascertain how many hidden neurons are suitable. Insufficient hidden neurons may lead to bad error tolerance. In another hand, too many hidden neurons will prolong learning time. Up to now there has not an efficient method in number selecting of hidden neurons. The number of hidden neurons can be regarded as an input and is to be determined through the training process. An initial number of hidden neurons is selected based on some experimental formulas, such as

$$\text{number}_{\text{hidden}} = \log_2 \text{number}_{\text{input}}$$

Then, the actual number is increased or decreased during training.

3 Using adaptive learning rate

For standard method, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process.

The performance of the steepest descent algorithm can be improved if we allow the learning rate to change during the training process. An adaptive learning rate will attempt to keep the learning step size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface. An adaptive learning rate requires some changes in the training procedure. First, the initial network output and error are calculated. At each epoch new weights and biases are calculated using the current learning rate. New outputs and errors are then calculated. So the performance of the steepest descent algorithm can be improved when we allow the learning rate to change during the training process.

4 Adding a control loop

This causes a problem when using steepest descent to train a multilayer network with sigmoid functions, since the gradient can have a very small magnitude; and therefore, cause small changes in the weights and biases, even though the weights and biases are far from their optimal values. So, the back-propagation algorithm cannot be shown to converge. There are some reasonable criteria, each with its own practical merit, which may be used to terminate the weight adjustments. To formulate such a criterion, it is logical to think in the terms of the unique properties of a local or global. In order to avoid plunging into local minimum, a control loop in the outer loop can be added to make the decision.

An algorithm combined with Cauchy algorithm can also solve this problem.

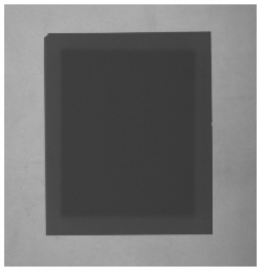

Fig.2 Workpiece in Visual Field

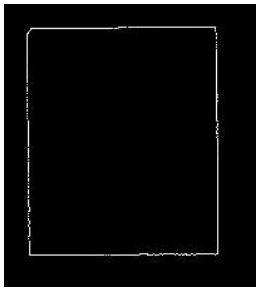

Fig.3 Detected Edge

3 Results and analysis

In our operation, the object is a rectangular piece in size of 50mm×60mm×0.4mm as shown in Fig. 2. Because of the manufacturing process, the sizes are distributed in width between 0.3mm to 0.4mm. There are urgent needs of

measurement and classification of product according to its sides, parallelism and rectangularity. Based on pixel edge point obtained using the neural network edge detection method, subdivision technology is used to improve higher resolution to meet the maximum permissible error at lower device cost. As a result of subdivision, the edge is located within a pixel, and the measure accuracy and error tolerance are improved by using trained NN to detect edge points (shown in Fig. 3).

By application in practical product line, up to 90 percentage of production can be classified correctly. After analyzing amounts of online data, we found that the errors are caused by some tiny particles attached to the edge of workpiece where is jagged. Therefore, algorithm based on BP network does benefit in on-line edge detection.

4 References

[1] Kenneth. R. Castleman, Digital Image Processing, PRENTICE HALL, 1998.

[2] Jiegu Li, Theory and Application of Computer vision, Shanghai Jiaotong University Press, 1991.

[3] Edward P. Lyvers, Owen Robert Mitchell, Mark L. Akey, Anthony P. Reeves, "Subpixel Measurements Using a Moment based Edge Operator", IEEE Transactions on Pattern. Analysis and Machine Intelligence, vol. 11 number 12, Dec. 1989, pp.1293-1309.

[4] Zhongzhi Shi, Neural Computation, Press of electronic Industry, 1993.

[5] Xueli Yu et al., Neural Network and Example Learning, Press of Railway of China, Beijing, 1996.