Classification of Peanut Leaves Disease using Back Propagation Neural Network

Naw Aye Aye Maw, Sandar Htay

University of Computer Studies, Hpa-an, Myanmar
nawayeayemaw2010@gmail.com

Abstract

A number of classification systems have been developed depending on the intended purpose of the system. This system tries to classify peanut leaves diseases.. The images of the peanut leaf are acquired by means of an digital imaging device, such as a scanner. The acquired color image is prepared with Image processing steps in order to get the digital image to be suitable with Neural Network. Back-Propagation Algorithm is used to train the Network in order to classify the peanut leaves diseases. In this system, there are many different types of peanut leaves diseases namely Early and Late Leaf spot, Rust, Web blotch, Leaf scorch, Alternaria Leaf spot, Phyllosticta Leaf spot, Peanut Mottle virus, Tomato Spotted wilt virus, Iron Chlorosis,and Ultraviolet radiation.

Keywords : Peanut Leaves Disease, Back-Propagation Algorithm, Artificial Neural Network.

1. Introduction

The main goal of this thesis is to develop a software model, to suggest remedial measures for disease management in agricultural crops. This system presents the classification of infected diseases in agricultural crops. In this system, Neural networks are used for pattern recognition and classification. Classification of peanut leaves disease is one of the most important attributes to agriculture and irrigated field. This system can provide the farmers to know what kinds of the diseases infected to peanut plants. When diseases affect the Peanut leaf, there will be a tremendous decrease in production. In order to increase the crop productivity. the farmers need to approach to experts to get their advices regarding the treatments of incidence of diseases. The system aims to recognize the images of leaves by using Neural Network.

2. Background Theory

2. 1 Image Processing

In electrical engineering and computer science, image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or, a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. Image processing usually refers to digital image processing, but optical and analog image processing also are possible. This article is about general techniques that apply to all of them. The *acquisition* of images (producing the input image in the first place) is referred to as imaging[1].

2.2 Artificial Neural Network

Artificial neural network (ANN) is a technique for creating artificial intelligence in the machine. This is an attempt of the modeling of the human brain in the serial machine for the various pattern recognition tasks The network processes the data, and a controlling algorithm adjusts each weight to arrive at the correct or final answer(s) for the data. These algorithms or procedures are called learning algorithms. Their adaptability and learning capabilities make them excellent choices for tasks requiring comparison of data sets or extracting subtle patterns from complex data.

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras. Many important advances have been boosted by the use of inexpensive computer emulations. Following an initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, important advances were made by relatively few researchers. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding[3]. An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system.

It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process[4]. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. Artificial neural networks(ANNs) provides a general pratical method for real-valued, discrete-valued, and vector-valued functions from examples. The back-propagation algorithm which is widely used in ANNs, uses gradient descent to tune network parameters to best fit a training set of input-output pairs[5].

2.3 Training and Testing Neural Networks

The best training procedure is to compile a wide range of examples (for more complex problems, more examples are required), which exhibit all the different characteristics of the problem. To create a robust and reliable network, in some cases, some noise or other randomness is added to the training data to get the network familiarized with noise and natural variability in real data. Poor training data inevitably leads to an unreliable and unpredictable network. Usually, the network is trained for a prefixed number of epochs or when the output error decreases below a particular error threshold.

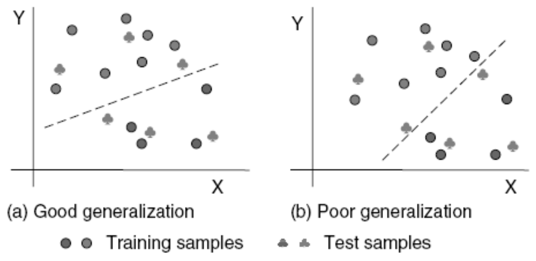


Figure 1. Illustration of Generalization Performance

Special care is to be taken not to overtrain the network. By overtraining, the network may become too adapted in learning the samples from the training set, and thus may be unable to accurately classify samples outside of the training set. Figure 1 illustrates the classification results of an overtrained network. The task is to correctly classify two patterns X and Y.The test patterns were not shown during the training phase. As shown in Figure 1 (left side), each class of test data has been classified correctly, even though they were not seen during training. The trained network is said to have good generalization performance. . Figure 1 (right side) illustrates some misclassification of the test data. The network initially learns to detect the global features of the input and, as a consequence, generalizes very well. But after prolonged training, the network starts to recognize individual input/output pairs rather than settling for weights that generally describe the mapping for the whole training set[7].

3. Backpropagation Algorithm

One of the most commonly used supervised ANN model is back-propagation network that uses back-propagation learning algorithm. Back-propagation algorithm is one of the well-known algorithms in neural networks. The back-propagation neural network is essentially a network of simple processing elements working together to produce a complex output. These elements or nodes are arranged into different layers: input, middle and output. The output from a back-propagation neural network is computed using a procedure known as the forward pass:

* The input layer propagates a particular input vector's components to each node in the middle layer.
* Middle layer nodes compute output values, which become inputs to the nodes of the output layer.
* The output layer nodes compute the network output for the particular input vector. The forward pass produces an output vector for a given input vector based on the current state of the network weights. Since the network weights are initialized to random values, it is unlikely that reasonable outputs will result before training. The weights are adjusted to reduce the error by propagating the output error backward through the network. This process is where the back-propagation neural network gets its name and is known as the backward pass:
* Compute error values for each node in the output layer. This can be computed because the desired output for each node is known.
* Compute the error for the middle layer nodes. This is done by attributing a portion of the error at each output layer node to the middle layer node, whichfeed that output node. The amount of error due to each middle layer node depends on the size of the weight assigned to the connection between the two nodes.
* Adjust the weight values to improve network performance using the Delta rule.

* Compute the overall error to test network performance. The training set is repeatedly presented to the network and the weight values are adjusted until the overall error is below a predetermined tolerance. Since the Delta rule follows the path of greatest decent along the error surface, local minima can impede training. The momentum term compensates for this problem to some degree[6].

3.1 Peanut Leaves disease

Most widespread peanut diseases of today were noted and described during the early days of commercial peanut production. As the peanut became more economically important in world agriculture, emphasis in production shifted from minimal input to intense input, which resulted in high yields and improved seed quality. As the peanut went from a low-value to a high-value crop and production intensity increased, both old and new diseases prevailed. Many different type of peanut leave diseases are

- Early and Late leaf spot
- Rust
- Web blotch
- Leaf scorch
- Alternaria leaf spot
- Phyllosticta leaf spot
- Peanut Mottle virus
- Tomato spotted wilt virus
- Iron chlorosis
- Ultraviolet radiation[2]

3.2 Step by Step Calculation of the System

Step 1. Design the structure of neural network and input parameters of the network .
Step 2. Get initial weights W and initial θ values from randomizing.
Step 3. Input training data matrix X and output matrix T.
Step 4. Compute the output vector of each neural units.
(a) Compute the output vector H of the hidden layer

$$net_k = \sum W_{ik} X_i - \theta_k$$
$$H_k = f(net_k)$$

(b) Compute the output vector Y of the output layer

$$net_j = \sum W_{kj} H_i - \theta_j$$
$$Y_j = f(net_j)$$

Step 5. Compute the distances d
(a) Compute the distances d of the output layer

$$\delta_j = (T_j - Y_j) \bullet f'(net_j)$$

(b) Compute the distances d of the hidden layer

$$\delta_k = (\sum_j \delta_j W_{kj}) \bullet f'(net_k)$$

Step 6. Compute the modification of W and θ (η is the learning rate)
(a) Compute the modification of W and θ of the output layer

$$\Delta W_{kj} = \eta \delta_j H_k$$
$$\Delta \theta_j = -\eta \delta_j$$

(b) Compute the modification of W and θ of the hidden layer

$$\Delta W_{ik} = \eta \delta_k X_i$$
$$\Delta \theta_k = -\eta \delta_k$$

Step 7. Renew W and θ
(a) Renew W and θ of the output layer

$$\mathrm{Wkj} = \mathrm{Wkj} + \Delta \mathrm{Wkj}$$
$$\theta_j = \theta_j + \Delta \theta_j$$

(b) Renew W and θ of the hidden layer

$$\mathrm{Wik} = \mathrm{Wik} + \Delta \mathrm{Wik}$$
$$\theta_k = \theta_k + \Delta \theta_k$$

Step 8. Repeat step 3 to step7 until convergence.

4. System Design

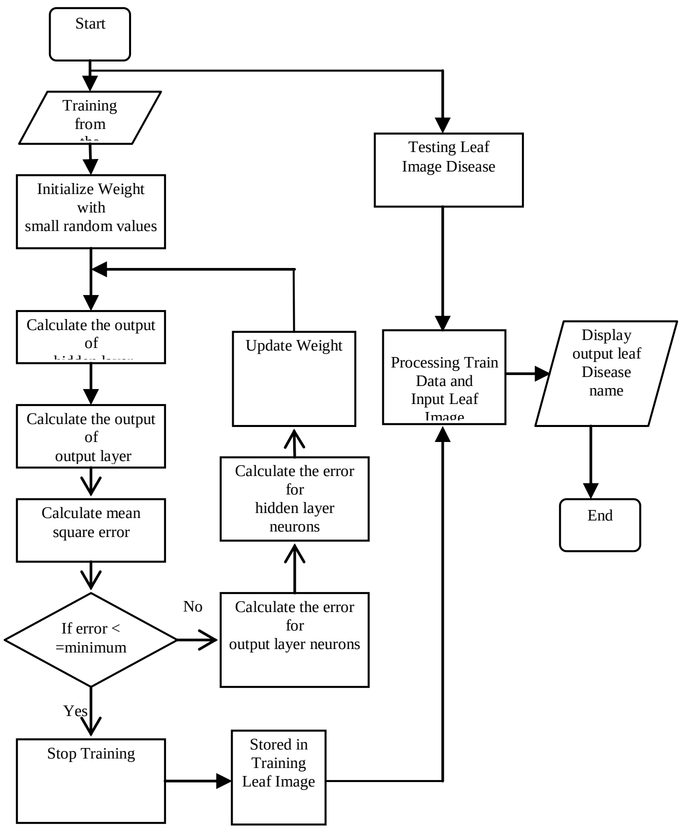

Start
Training from
Testing Leaf Image Disease
Initialize Weight with small random values
Calculate the output of
Update Weight
Processing Train Data and Input Leaf
Display output leaf Disease name
Calculate the output of output layer
Calculate the error for hidden layer neurons
End
Calculate mean square error
No
If error < =minimum
Calculate the error for output layer neurons
Yes
Stop Training
Stored in Training Leaf Image

Figure 2. System flow diagram

In this system, inputs are extracted from the leaf image. There are six steps of this component to generate feature vectors. The first step is pre-processing. Leaf images are filtered, grayscale, resized and edge detection for feature enhancement. Then, input image is recognized. In this system, there are 256 nodes of input layer and ten nodes of output layer. These input enough for classification.

The system initialize the weight with small random values, calculate the output of hidden layer neurons and calculate the output of output layer neurons. Then calculate mean square error value. If error is greater than minimum value, calculate error for output and hidden layer and update the weight. If error less than or equal with minimum value, stop training and kept the train image in the train database. If user want to tests the leaf disease image, the system recognizes the input test leaf image and then checks with train data leaf disease image. If the test data and train data is matched, the system displays the result leaf disease. If it is not , the system returns "Undefined leaf disease" will shown.

A neural network for recognition of leaves is implemented in this paper. The training set contains minimum five spices for each type of leaf in each data file. Using more number of species in training set and number of output nodes can enhance the recognition ability. Using quick training algorithms without losing recognition performance can enhance the scope of this paper.

4.1 Implementation

In this system, in the first step, one input leaf is loaded. It is shown in figure 3.

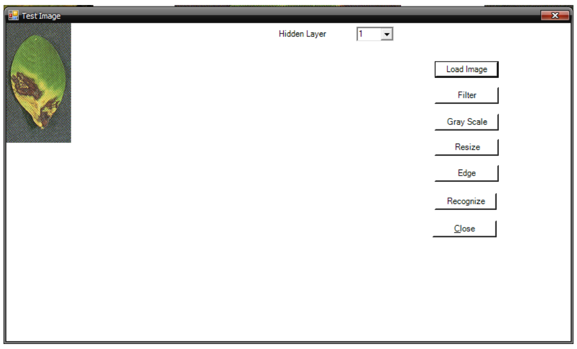

Hidden Layer 1

Figure 3. Load Leaf Image

In the second step, this input peanut leaf is filtered. It is shown in figure 4.

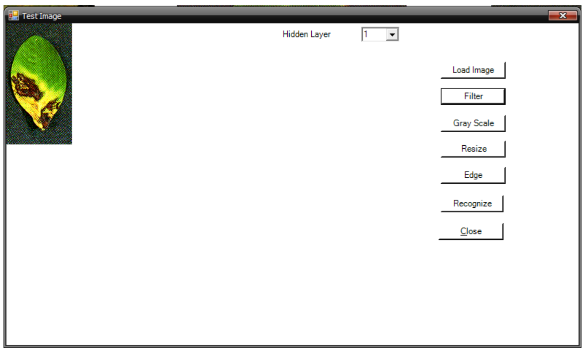

Hidden Layer 1

Figure 4. Filter Leaf Image

In the third step , the filtering result is changed to Grayscale shown in figure 5.

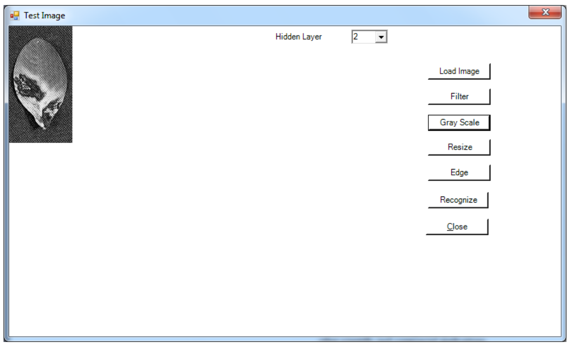

Hidden Layer 2

Figure 5 . Grayscale Leaf Image

In the 4th step, the Grayscale leaf is Resized shown in figure 6.

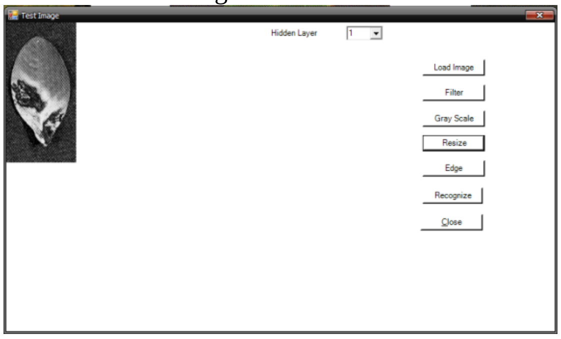

Hidden Layer 1

Figure 6 . Resize Leaf Image

In the 5th step, the Resized leaf is edged shown in figure 7.

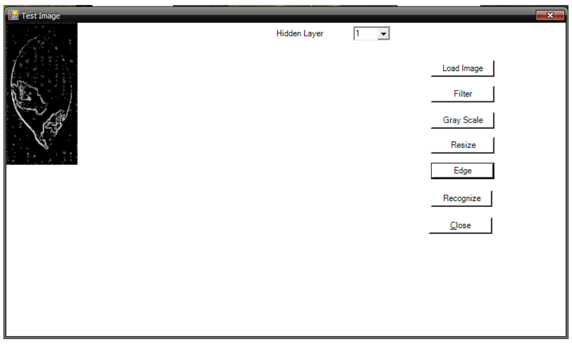
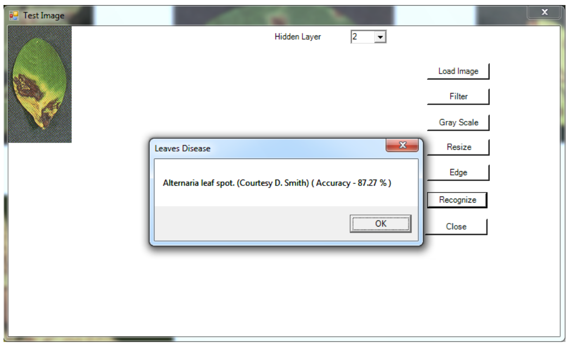

Hidden Layer 1

Figure 7 . Edge Leaf Image

As the final step, edge leaf is recognized with Neural Network . And then the system can show leaf diseases name and accuracy for hidden layer one as the output result shown in figure 8.



Alternaria leaf spot. (Courtesy D. Smith) (Accuracy - 87.27 %)

Figure 8 . Recognize Leaf Image

4.2. Analyzing Result

No.	Diseases	Accuracy
1	Early and late	88.22%
2	Rust pustules	86.65%
3	Web blotch	89.31%
4	Leaf scorch	87.32%
5	Alternaria leaf spot	91.1%
6	Phyllosticta leaf spot	87.45%
7	Mottle virus	89.22%
8	Ringspot symptom	86.52%
9	Iron chlorosis	88.22%
10	Leaf injury	90.09%

In this analyzing result, the accuracies is not same in different leaves eg. Early and late 88.22% and Rust pustules 86.65%.

5. Conclusion

Image recognition can be done both in normal computers and neural networks. Computers use conventional arithmetic algorithms to detect whether the given pattern matches an existing one. It is a straightforward method. It will say either yes or no. It does not tolerate noisy images. On the other hand, neural networks can tolerate noise and, if trained properly, will respond correctly for unknown patterns. Neural networks may not perform miracles, but if constructed with the proper architecture and trained correctly with good data, they will give amazing results, not only in image recognition but also in other scientific and commercial applications.

5.1 Limitation and Further Extension

This system is aimed to develop to test peanut leaves diseases using image processing and artificial neural network. This system uses ten types of peanut leaves diseases.

Further enhancement of this work involves more experimentation's with large training sets to recognize various leaves with damaged leaves due to insects or diseases and develop an expert system. The diseased leaf images are to be collected in large number and are to be sent to experts in agricultural research.

REFERENCES

[1]. Andrew B. Graham, "Image Processing Algorithm for Detection of Two-Dimensional Visual Code Makers", Stanford University, Stanford, USA, 2006.

[2]. D. Morris Porter, Donald H. Smith, and R. Rodriguez-Kabana, "Compendium of Peanut Diseases", U.S,1984

[3] Fausett, L. (1994), "*Fundamentals of Neural Networks"*, Prentice Hall, USA.

[4]. Mandic, D. and Chambers, J. (2001) *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*, John Wiley & Sons, New York.

[5]. McCulloch, W.S. and Pitts, W.H. (1943) A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*

[6]. "Meta-Learning Evolutionary Artificial Neural Networks"Abraham, A. (2004), *Neurocomputing Journal*, Vol. 56c, Elsevier Science,Netherlands, (1–38).

[7]. *"Neural Networks for Pattern Recognition*," Bishop, C.M. (1995) Oxford University Press, Oxford, UK.