Neural Network Project

By JJ

Final Project

- Team project of at most three people
- Create a C++ program which will allow user to create an artificial neural network
- Your program will ask the user to enter parameters to create a neural network

Final Project

- The parameters include the following
  - How hidden layers for your machine?
  - How many input nodes of input layer?
  - How many output nodes of output layer?
  - The learning rate of your machine to start with.
  - The maximum cycles for a training phase
  - The error tolerance
  - The data source file name

Final Project

- After creating the machine, your team shall use this machine to solve a financial application problem of your choice. For example, you may propose to use ANN to predict stock index.
- Your program shall allow user to specify whether to do training or testing

What next?

- Follow the Software Engineering guidelines to develop the system using C++ as implementation language.
- I.e., go through analysis, design, implementation, testing phases.

An example application

- The following is an example of using ANN to predict GDP based on the 10 leading economic indicators

Problem Description

Assuming that we want to develop correlations between Leading Economic Indicators (LEI) and Gross Domestic Product (GDP) using backpropagation neural network method. The goal is to develop correlations such that changes in LEI can be used to predict changes in GDP.

Gross Domestic Product is a measure of the total economic output of the United States.

Leading Economic Indicators

LEI	Description
BCI-01	Average weekly hours, manufacturing
BCI-05	Average weekly initial claims for unemployment insurance
BCI-08	Manufacturers new orders for consumer goods and materials from factory orders report
BCI-32	Vendor performance, slower deliveries diffusion index from Purchasing Managers' Index Report
BCI-27	Manufacturers' new orders for non-defense capital goods from the factory orders report
BCI-29	Building permits, new private housing units from the housing starts report
BCI-19	Stock prices of S&P 500
BCI-106	Inflation adjusted measure of the M2 money supply
BCI-129	Interest rate spread between 10 year Treasury note and Fed funds rate
BCI-83	Expectations portion of the University of Michigan's Consumer Sentiment Index

Data Normalization

- The raw economic data has been modified in order to get the LEI and the GDP change values with a minimum value of -1.0, and a maximum value of 1.0.
- In the following discussion, A represents the raw form of the economic data, B is the raw data transformed into a differential form, and C is the normalized version of B.

Data Normalization

- if component A_t is in percent change form or an interest rate, simple arithmetic differences are calculated as
- $$B_t = A_t - A_{t-1}$$
- otherwise, a symmetric alternative is used
- $$B_t = 200\left(\frac{A_t - A_{t-1}}{A_t + A_{t-1}}\right)$$

Data Normalization

In order to normalize this data so that the minimum value for the LEI, and GDP change is -1.0, and the maximum value is 1.0. This is

$$C_t = 2\left(\frac{B_t - B_{\min}}{B_{\max} - B_{\min}}\right) - 1$$

Normalization

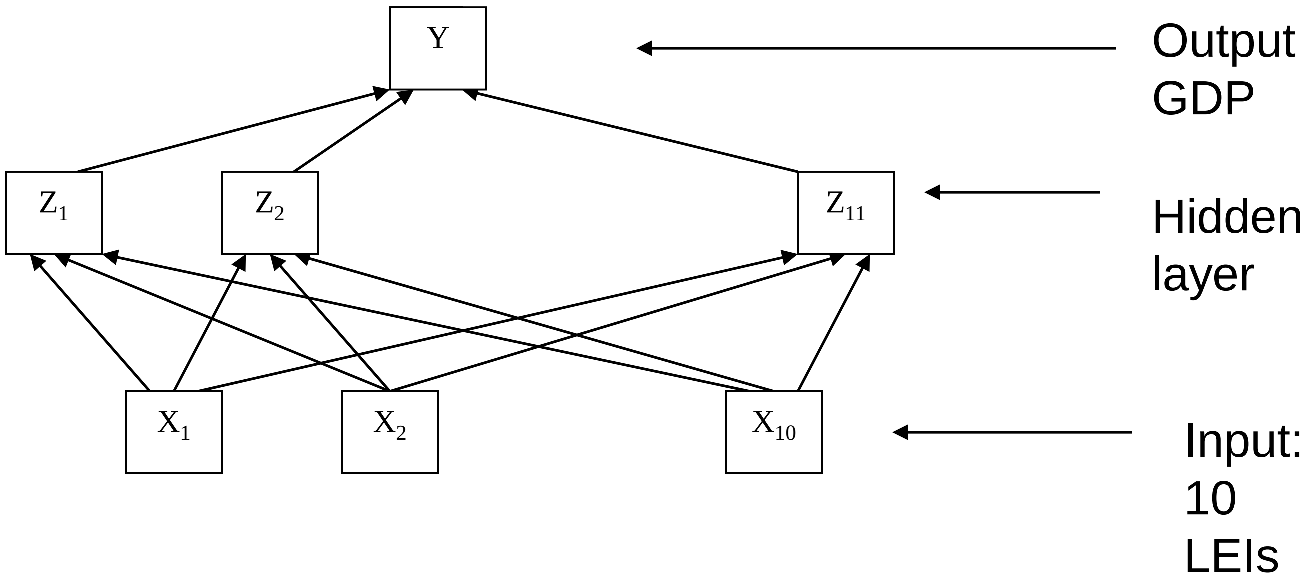
- Some use a formula such as the following to normalize
- New value = (old Value –mean)/(Standard deviation)

Backpropagation

The backpropagation network has an input layer with ten input nodes that correspond with the ten LEI variables. Let’s assume there is a hidden layer with eleven nodes

Note: The number of nodes on hidden layer can be different. Usually, one can get a good results by trials and errors.

Backpropagation



Backpropragation

- The link between nodes are called weights
- The weights between the layers contain the information for the pattern correlation.
- The initial weights may be determined using a random number generator.
- The training method is used to adjust the weights so that the GDP predicted using the LEI input data is as close as possible to the actual GDP.

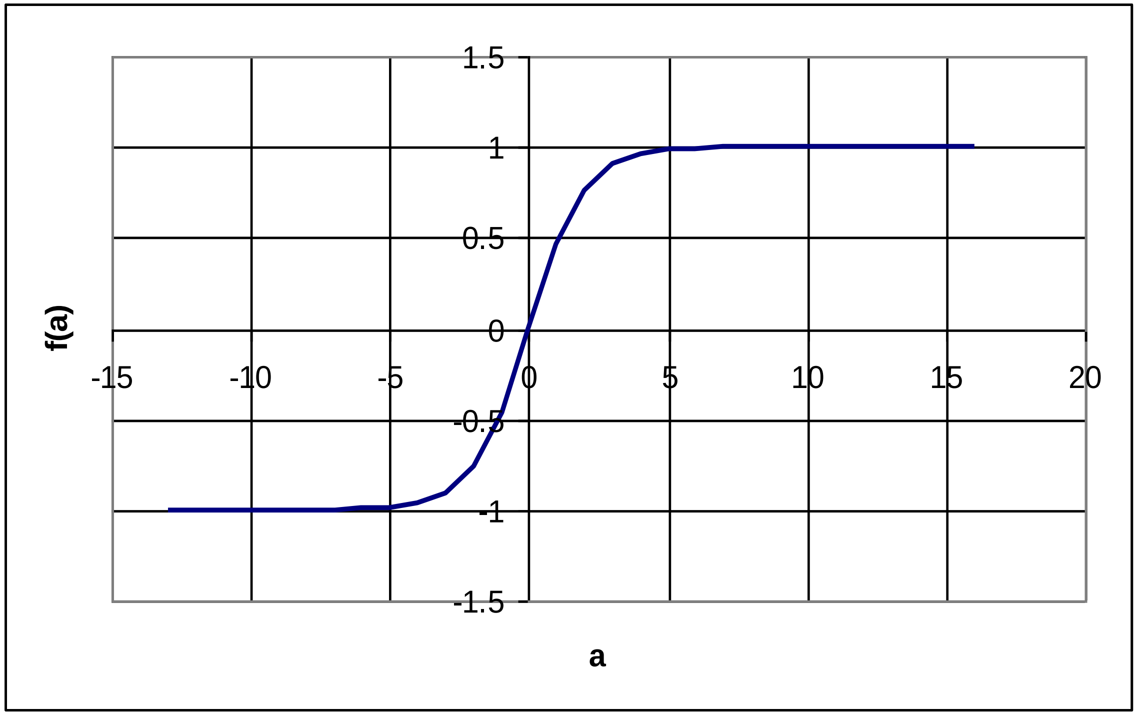
An Activation Function

The following activation function is a bipolar sigmoid function

$$f(a) = \frac{2}{1 + \exp(-a)} - 1$$

$$f'(a) = \frac{1}{2}[1 + f(a)][1 - f(a)]$$

Activation Function



Backpropagation Algorithm

- The weights between the layers are determined using a random number generator. Assume weights between X_i and Z_i are labeled as V_i, weights between Z_i and Y are labeled as W_i

Backpropagation Algorithm

- The input layer receives its signal and broadcasts its value to the hidden layer nodes.
- Each of the hidden layer nodes makes a summation of the input node signals

$$Z_{in,1} = V_{0,1} + X_1 V_{1,1} + \quad \ldots \quad + X_{10} V_{10,1}$$

Backpropagation Algorithm

- The hidden layer nodes use the activation function to determine their output signals

$$Z_1 = f\left(Z_{in,1}\right)$$

$$\vdots$$

$$Z_{11} = f\left(Z_{in,11}\right)$$

Backpropagation Algorithm

- The output layer makes a summation of the hidden layer signals, then use activation function to calculate its output

$$Y_{in} = W_0 + Z_1 W_1 + \quad \dots \quad + Z_{11} W_{11}$$

Backpropagation Algorithm

- The predicted GDP is calculated by applying the activation function on the summation shown above

$$Y = f\left(Y_{in}\right)$$

- The predicted GDP is compared against the actual GDP in order to correct the weights between the layers

$$\delta_Y = \left(Y_{actual} - Y\right) f'\left(Y_{in}\right)$$

$$\Delta W_1 = \alpha\ \delta_Y\ Z_1$$

$$\Delta W_0 = \alpha\ \delta_Y (NOTE : bias\_neuron\_always\_ fires\_1)$$

$$\delta_{Z,1} = \delta_Y\ W_1\ f\left(Z_{in,1}\right)$$

$$\Delta V_{1,1} = \alpha\ \delta_{Z,1}\ X_1$$

Errors

- Each hidden unit sums its delta inputs from the output node in the layer above, multiplies by the derivative of its activation function to calculate its error information term

$$\delta_{Z,1} = \delta_Y \, W_1 \, \frac{df\left(Z_{in,1}\right)}{dZ}$$

$$\vdots$$

$$\delta_{Z,11} = \delta_Y \, W_{11} \, \frac{df\left(Z_{in,11}\right)}{dZ}$$

Calculate delta V*i*

$$\Delta V_{1,1} = \alpha\ \delta_{Z,1}\ X_1$$

$$\vdots$$

$$\Delta V_{1,11} = \alpha\ \delta_{Z,11}\ X_1$$

$$\vdots$$

$$\Delta V_{10,1} = \alpha\ \delta_{Z,1}\ X_{10}$$

$$\vdots$$

$$\Delta V_{10,11} = \alpha\ \delta_{Z,11}\ X_{10}$$

Calculates the bias correction term

$$\Delta V_{0,1} = \alpha\, \delta_{Z,1}$$

$$\vdots$$

$$\Delta V_{0,11} = \alpha\, \delta_{Z,11}$$

Backpropagation Algorithm

- Each output unit and hidden unit updates its weights

$$W_1(new) = W_1(old) + \Delta W_1$$

$$V_{1,1}(new) = V_{1,1}(old) + \Delta V_{1,1}$$

Update weights

$$W_1(new) = W_1(old) + \Delta W_1$$
$$\vdots$$
$$W_{11}(new) = W_{11}(old) + \Delta W_{11}$$

Update weights

$$V_{1,1}(new) = V_{1,1}(old) + \Delta V_{1,1}$$
$$\vdots$$
$$V_{1,11}(new) = V_{1,11}(old) + \Delta V_{1,11}$$
$$\vdots$$
$$V_{10,1}(new) = V_{10,1}(old) + \Delta V_{10,1}$$
$$\vdots$$
$$V_{10,11}(new) = V_{10,11}(old) + \Delta V_{10,11}$$

List of variables

- $V_{0,1}$ through $V_{0,11}$ = bias to hidden layer nodes 1 through 11
- $\Delta V_{0,1}$ through $\Delta V_{0,11}$ = correction term of **bias** to hidden layer nodes 1 through 11

List of variables

- $V_{1,1}$ through $V_{1,11}$ = weights from inlet layer node 1 to hidden layer nodes 1 through 11
- $\Delta V_{1,1}$ through $\Delta V_{1,11}$ = correction term of weights between inlet and hidden layer nodes
- ……………..
- $V_{10,1}$ through $V_{10,11}$ = weights from inlet layer node 10 to hidden layer nodes 1 through 11
- $\Delta V_{10,1}$ through $\Delta V_{10,11}$ = correction term of weights between inlet and hidden layer nodes

List of variables

- W_0 = bias to the output layer node
- ΔW_0 = correction term of the bias to the output layer node
- W_1 through W_{11} = weights from the eleven hidden layer nodes to the output node
- ΔW_1 through ΔW_{11} = correction term of the weights between the hidden and output layer nodes

List of variables

- X_1 through X_{10} = input signals 1 through 10
- Y_{in} = the net input to the output layer node
- Y = the output signal from the output layer node. This is the predicted GDP change.
- Y_{target} = the actual value of GDP change

List of variables

- $Z_{in,1}$ through $Z_{in,11}$ = net input to hidden layer nodes 1 through 11
- Z_1 through Z_{11} = output signal from hidden layer nodes 1 through 11
- α = learning rate
- δ_Y = portion of error correction weight adjustment due to error at the output unit
- $\delta_{Z,j}$ = portion of error correction weight adjustment due to backpropagation of error

Parameters to the predictor program

- Training or Test mode
- Value of error tolerance
- Learning rate, you may have different learning rate on different set of weights (e.g., 0.2 for input to hidden and 0.3 for hidden to output layer, etc)
- Maximum cycles, or passes on training data
- Number of layers
- The size of each layer from input to output

When to stop training?

- When the average and total error is within error tolerance that user selected or the maximum number iterations reached
- Save the weights of your network onto a file. This set of weights will be used later when predicting is required (i.e., on the test mode)

How to predict (test mode)

- Load the final weight file to the network
- Give the input and receive the output

Suitable Final Project Examples

- You can write a program using LEI to predict GDP as outline above, or
- Write a program to predict S and P 500 index change (weekly for example) using reasonable inputs such as NASDAQ advancing/declining issues, total volumes, advancing/declining volumes; similar for NYSE, may include new highs and new lows, crude oil price(or gold price), S and P high and low, S and P closing price, interest rate and so on, or
- Trace specific stock or stock group of your choice using other economic data or other stocks as input
- Others

What to submit

- Final Project report which includes
- A cover page which has
  - Project name
  - Team name
  - Names of team members
  - Date submitted

What to submit

- Table of content
  - Analysis of your project
  - Design
  - Implementation
  - Testing
  - Results
    - The comparisons of using different parameters
  - Remarks and possible future improvement
  - References

About Testing

- At least test and compare results for using 3 different hidden layers (For example, when only use one hidden layer, use two hidden layers and use three hidden layers to create the machine)
- Test and check if different number of maximum cycles to train the machine make your results different.
- Test and try different parameters’ value and check what kinds of configuration will make your machine perform the best
- Can you draw some conclusions based on your experiments?

What to submit

- In addition to the final report
- Your submission also include the following file
  - Source code
  - User manual
  - Test data file
  - Training data file
  - Final weight file (from training phase)

What to submit

- C++ source Program files
  - All the files related to training and test of the network
  - Name your driver program yourlastnamefirstnameNN.cpp

What to submit

- User’s manual
  - Describe how to use your program
  - For example, how to prepare input file (format)
  - How to get output
  - Etc.

What to submit

- Training data set file, testing data set file (text files); prepare these files from the financial database
- Final weight file (contains final weights of the network)

Rubric

• –Problem Specification and Analysis of project	5%
• –Design	5%
• Description of your network	
• clearly define inputs and output(s)	
• –Implementation	
• correctness	45%
• Source code modualization	5%
• –Testing (screen shots)	10%
• –Results and remarks and possible future improvement	5%
• –References	5%

Rubric (continue)

- –User manual 10%
- Describe how to use your program
- how to prepare input file (format)
- How to get output
- Files and source file submitted 10%
- –Test data file
- –Training data file
- –Final weight file (from training phase)

Your implementation shall have a main program similar to the following

```
while (cycle < maxCycle  && averageError > toleranceEerr)
{
        for (int i= 0; i < totalRecords;  i++)
        {
                // calculate the error (desired value – actual value)
                forwardPropagation();
                backwardPropagation();
                accumulateError;
                updateWeights(learningRate);
        }
        compute averageErr;
}
```