The Catholic University of America

School of Engineering – Department of Electrical Engineering and Computer Sciences

Predict individual income using an artificial neural network

Project 2

CSC530 – Introduction to Data Analysis

By: Duong Le and Vy Bui

Advisor: Dr. Le He

Fall 2014

I. Topic Selection

In this project, we will implement an artificial neural network for the classification task. The objective of this project is to predict whether an individual makes over $50,000 a year, based on a set of characteristics.

The remainder of this report organized as follows. Section II gives the details about how we obtained the public data. Section III introduces the background review about the state-of-the-art techniques for this problem or similar problems. Section IV presents the implementation approach. Section V provides results analysis. Section VI concludes about the lessons and the future work of this study.

II. Data Collection

Our project uses demographic data from the U.S. Census Bureau, available at http://archive.ics.uci.edu/ml/machine-learning-databases/adult/

The values of the nominal attributes are as follows:

Age: continuous.

Workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

Fnlwgt: continuous.

Education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

Education-num: continuous.

Marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

Occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

Relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

Race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

Sex: Female, Male.

Capital-gain: continuous.

Capital-loss: continuous.

Hours-per-week: continuous.

Native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Income class: >50K, <=50K.

III. Background Review

In machine learning and related fields, artificial neural networks (ANNs) are computational models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which can compute values from inputs, and are capable of machine learning as well as pattern recognition thanks to their adaptive nature.

1. Backpropagation algorithm

Backpropagation, an abbreviation for "backward propagation of errors", is a common method of training artificial neural networks used in conjunction with an optimization method such as gradient descent. The method calculates the gradient of a loss function with respects to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

The backpropagation learning algorithm can be divided into two phases: propagation and weight update.

a. Phase 1: Propagation

Each propagation involves the following steps:

Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.

Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas of all output and hidden neurons.

b. Phase 2: Weight update

For each weight-synapse follow the following steps:

Multiply its output delta and input activation to get the gradient of the weight.

Subtract a ratio (percentage) of the gradient from the weight.

This ratio (percentage) influences the speed and quality of learning; it is called the learning rate. The greater the ratio, the faster the neuron trains; the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates where the error is increasing, this is why the weight must be updated in the opposite direction.

Repeat phase 1 and 2 until the performance of the network is satisfactory.

2. Feed-forward algorithm

A feedforward neural network is a biologically inspired classification algorithm. It consist of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit in a layer is connected with all the units in the previous layer. These connections are not all equal, each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes.

Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feedforward neural networks.

The operation of this network can be divided into two phases:

a. The learning phase

During the learning phase the weights in the feedforward neural network will be modified. All weights are modified in such a way that when a pattern is presented, the output unit with the correct category, hopefully, will have the largest output value.

The feedforward neural network uses a supervised learning algorithm: besides the input pattern, the neural net also needs to know to what category the pattern belongs. Learning proceeds as follows: a pattern is presented at the inputs. The pattern will be transformed in its passage through the layers of the network until it reaches the output layer. The units in the output layer all belong to a different category. The outputs of the network as they are now are compared with the outputs as they ideally would have been if this pattern were correctly classified: in the latter case the unit with the correct category would have had the largest output value and the output values of the other output units would have been very small. On the basis of this comparison all the connection weights are modified a little bit to guarantee that, the next time this same pattern is presented at the inputs, the value of the output unit that corresponds with the correct category is a little bit higher than it is now and that, at the same time, the output values of all the other incorrect outputs are a little bit lower than they are now. (The differences between the actual outputs and the idealized outputs are propagated back from the top layer to lower layers to be used at these layers to modify connection weights. This is why the term backpropagation network is also often used to describe this type of neural network.

If you perform the procedure above once for every pattern and category pair in your data set you have have performed 1 epoch of learning.

The hope is that eventually, probably after many epochs, the neural net will remember these pattern-category pairs. You even hope that the neural net when the learning phase has terminated, will be able to generalize and has learned to classify correctly any unknown pattern presented to it.

Because real-life data many times contains noise as well as partly contradictory information these hopes can only be partly fulfilled.

For learning you need to select 3 different objects together: a feedforward neural network (the classifier), a Pattern (the inputs) and a Categories (the correct outputs).

b. The classification phase

In the classification phase the weights of the network are fixed.

A pattern, presented at the inputs, will be transformed from layer to layer until it reaches the output layer. Now classification can occur by selecting the category associated with the output unit that has the largest output value. For classification we only need to select an feedforward neural network and a Pattern together. In contrast to the learning phase classification is very fast.

IV. Implementation approach & Results analysis

1. Data preprocessing

ANNs learn faster and give better performance if the input variables are pre-processed before being used to train the network. There are two type of data in our dataset: continuous data and categories data.

a. Data preprocessing for continuous data

The continuous data have to be scaled before using in ANN, the reasons for scaling the data is to equalise the importance of variables

To scale data, we find the maximum value and the minimum value for that type of data and then calculate the scaled value of that data

$$ScaleData = \frac{data - min}{\max - min}$$

All the continuous data will be in range of [0,1] after scaling

b. Data preprocessing for categories data

We convert categories data into binary form, for example, for workclass attribute, we have 8 categories, we choose to represent each workclass category by using 3-bit binary

Workclass	Binary represent
Private	000
Self-emp-not-inc	001
Self-emp-inc	010
Federal-gov	011
Local-gov	100
State-gov	101
Without-pay	110
Never-worked	111

Similar process for other categories data

2. Training algorithms & Results

In our project, we divide our dataset into 3 sets: training set, validation set and testing set. The training set is used to update the weights iteratively, the validation set is used to stop the training algorithm, and the testing set is used to estimate how well our ANN performs.

The training algorithm:

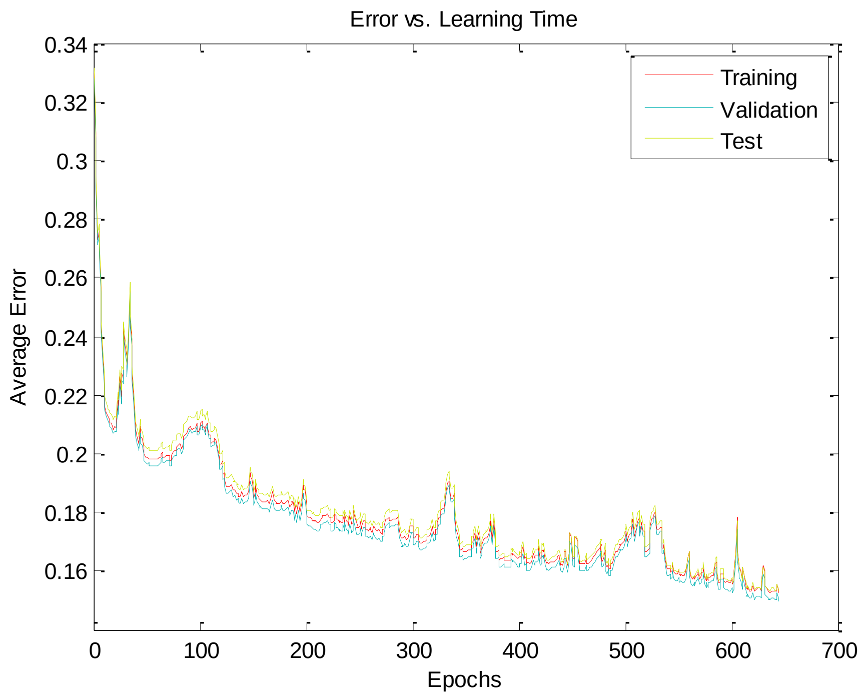
- Loading training samples, validation samples, and test samples
- Initialize a weight matrix using initialize weights
- Construct three bias vectors bias_training, bias_validate, and bias_test. Each must contain only 1s, with as many rows as there are inputs in the training, validation and test sets respectively.
- Implement a while loop that stops after a number of epoch (or a condition)
- Inside the loop, call the backpropagation algorithm. Use the training set inputs, the weights, (for now) a fixed learning rate of 0.1, and bias vector bias_train. Assign the result to weights.
- Still inside the loop, call the network error function three times: one time for each of the training, validation, and test sets. Use the weight matrix, and the appropriate bias vector.
- Store the errors in arrays with the current epoch number as index.
- After the loop, plot the error as a function of epoch number

Note:

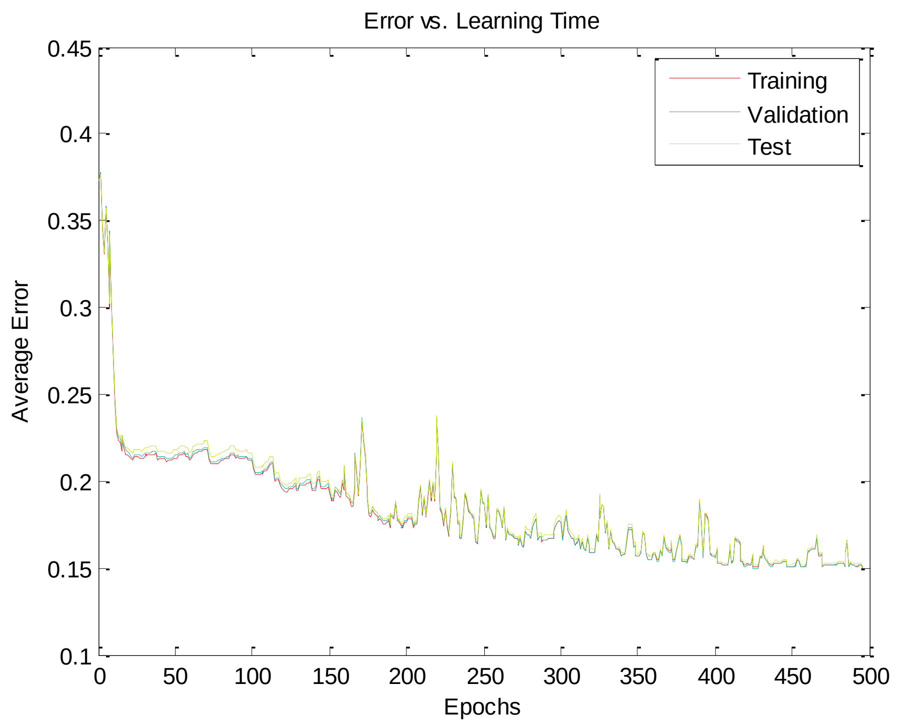
- Many different things affect the number of epochs we will need to train the neural net: the problem complexity, how we scale our updates, and how the weight matrix has been initialized
- We can also control the speed with a parameter called the learning rate. In general, higher learning rate means faster learning (although, when it is set too high, the network might become unstable and not learn at all).
- The while loop to stop when the validation error drops below a threshold or we we reach the number of epochs.

For each learning rate, we train the algorithm 30 times and then report the mean, standard deviation and maximum of the training time and the regression error (max_epoch = 2000, max_weight = 1/2, validation_stop_threshold = 0.15).

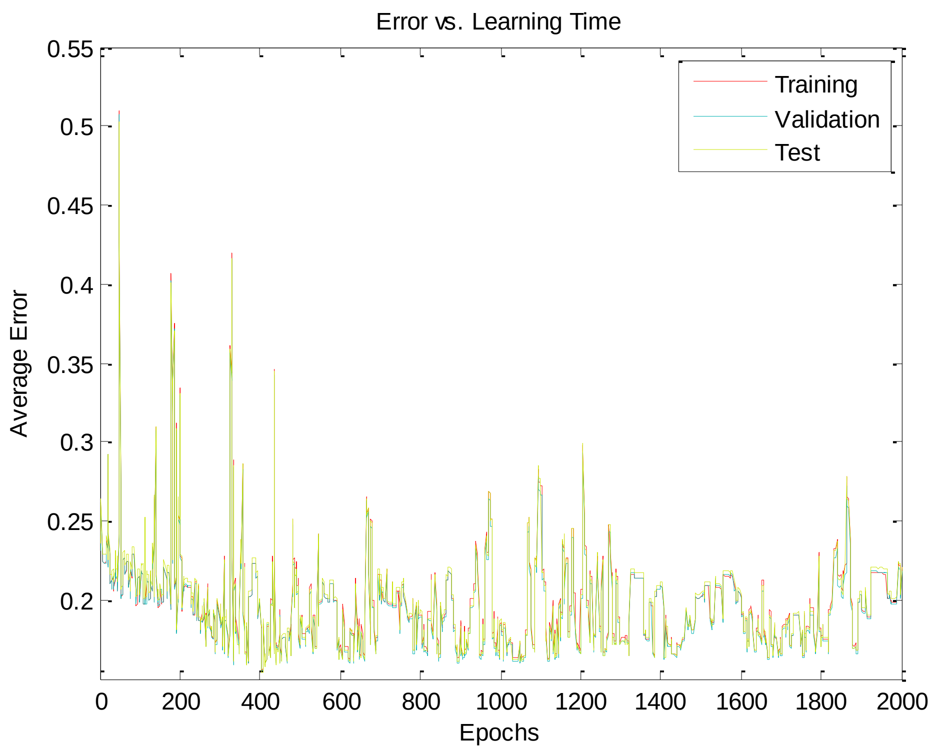
	Mean	Standard Deviation	Maximum
Eta = 0.05			
Training time	798.2	315.978	1598
Regression error	0.151679	0.000527208	0.152252
Eta = 0.10			
Training time	416	107.516	629
Regression error	0.151896	0.00140273	0.154145
Eta = 0.50			
Training time	2000	0	2000
Regression error	0.184011	0.0161992	0.216942



ETA = 0.05


Error vs. Learning Time
Average Error
Epochs
Training
Validation
Test
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0
50
100
150
200
250
300
350
400
450
500

ETA = 0.1


Error vs. Learning Time
Average Error
Epochs
Training
Validation
Test
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0
200
400
600
800
1000
1200
1400
1600
1800
2000

ETA = 0.5

V. Conclusion

In this project, we will implement an artificial neural network for the classification task. The objective of this project is to predict whether an individual makes over $50,000 a year, based on a set of characteristics. Our project uses demographic data from the U.S. Census Bureau. We ran several experiments to show the algorithm’s performance and validate with cross validation, the classification error is acceptable. The source code is included.