605.649 Programming Project 5: Feedforward Neural Networks

Christopher El-Khouri

1. Introduction

Our journey into the world of machine learning leads us to Artificial Neural Networks (ANN), models that were inspired by the way that the neural networks in the brain are structured. Feedforward Neural Networks are a type of ANN that generally utilize perceptrons, which are supervised linear learning algorithms that can be used for classification or regression.

Figure 1 shows a simple perceptron:

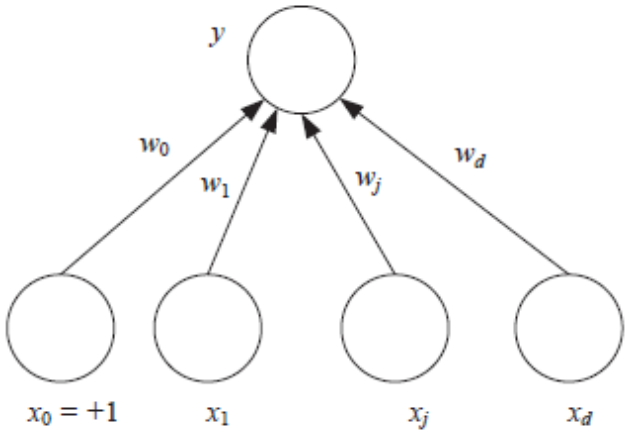

Figure 1: Simple Perceptron

The output y has the form of a typical linear model:

$$y = w_0 + \sum_{i=1}^{d} w_i x_i \quad (1)$$

Multilayer Perceptrons (MLP) are perceptrons with 1 or more hidden layers, where the neurons in one layer are connected to the neurons in the other in a feedforward type structure. Figure 2 shows a 1-layer feedforward neural network with multiple outputs, $z_{1...h}$ are the activation functions that typically take sigmoid, tanh, or gaussian transformations.

Our main focus for this project will be on Feedforward Neural Networks that are trained with backpropagation. We will be analyzing the performance of MLP Feedforward Networks with 0,1, and 2 hidden layers for both classification and regression.

Training MLPs utilizes the typical gradient descent algorithm used in the previous projects, however, due to the multiple layers, the error propagates from the output y back to the inputs.

Taking the MLP shown in Figure 2, the gradient calculation would be as follows:

$$\frac{\delta E}{\delta w_{hj}} = \frac{\delta E}{\delta y_i}\frac{\delta y_i}{\delta z_h}\frac{\delta z_h}{\delta w_{hj}}$$

This is known as Backpropagation, credited to D. Rumelhart et al. (1986).

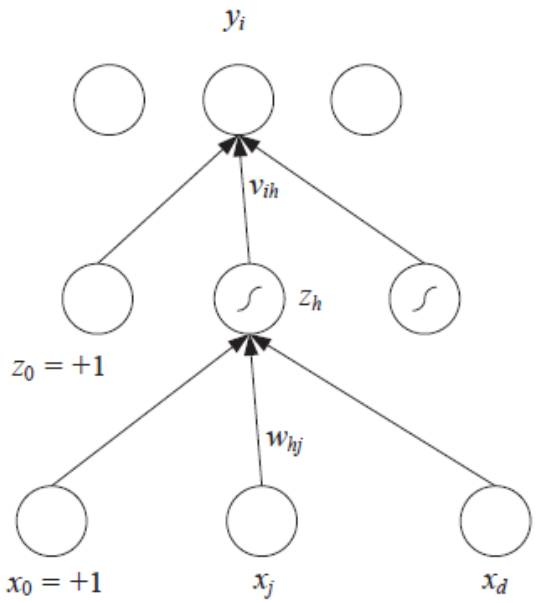

Figure 2: 1-layered MLP

We will be comparing the performance of feedforward neural networks with 0,1 and 2 hidden layers on 6 different datasets. We hypothesize that the accuracy of the outputs produced will increase with the number of layers used. The accuracy of our classification problems will be calculated using the classification error, which is the percentage of incorrectly classified points. The accuracy of our regression problems will be calculated using the Mean Squared Error (MSE).

In the next section, we will describe the technical and theoretical basis of our algorithms, in section 3 we will list the datasets involved in this experiment, section 4 will present and compare the results, and in section 5 we will discuss the significance of our experiment and compare the performance with the other algorithms previously encountered in the course with the same datasets.

2. Algorithms and Experimental Methods

The Backpropagation algorithm used in this project is based on the following pseudocode shown in Figure 3. This pseudocode is for the implementation of Backpropagation for regression on a Feedforward Neural Network with 1 hidden layer, H hidden dimensions, and K outputs. Gradient descent is being implemented on an incremental basis as the dataset is being traversed row by row.

Initialize all v_{ih} and w_{hj} to rand$(-0.01, 0.01)$
Repeat
  For all $(\boldsymbol{x}^t, r^t) \in \mathcal{X}$ in random order
    For $h = 1, \ldots, H$
      $z_h \leftarrow \text{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x}^t)$
    For $i = 1, \ldots, K$
      $y_i = \boldsymbol{v}_i^T \boldsymbol{z}$
    For $i = 1, \ldots, K$
      $\Delta \boldsymbol{v}_i = \eta(r_i^t - y_i^t)\boldsymbol{z}$
    For $h = 1, \ldots, H$
      $\Delta \boldsymbol{w}_h = \eta(\sum_i (r_i^t - y_i^t) v_{ih}) z_h (1 - z_h) \boldsymbol{x}^t$
    For $i = 1, \ldots, K$
      $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \Delta \boldsymbol{v}_i$
    For $h = 1, \ldots, H$
      $\boldsymbol{w}_h \leftarrow \boldsymbol{w}_h + \Delta \boldsymbol{w}_h$
Until convergence

Figure 3: Backpropagation pseudocode

This algorithm has been modified for our project by defining the convergence as the maximum error difference between the previous values of the coefficients and the new values to be less than 10^{-1}. Furthermore, the implementation of gradient descent on the evaluation of the coefficients has been modified from incremental updating to batch updating by taking the average of the gradient values of a complete pass over the dataset when the number of hidden layers is less than 2. If we are training for a network with 2 hidden layers, the dataset provided is then split into ND datasets each having 100 rows each and the gradient values are updated after an incremental pass on each of the ND datasets, this approach has been taken in an effort to reduce the probability of the vanish gradient problem. Batch updating has been implemented to enhance processing time when training the model.

Backpropagation$(\mathbf{n}, \mathbf{nhl}, \mathbf{nhd}, \mathbf{df}(\mathbf{X}, \mathbf{r}))$:
Where:
n: The learning rate upon which the coefficients w are updated
nhl: The number of hidden layers
nhd: The number of hidden dimensions H if nhl $= 1$, [L, H] if nhl $= 2$
df: A data frame containing d features X, and target variable r with K outputs
Initialize w, v1, v2 to rand $(-0.01, 0.01)$
If len(df) > 100 then split df into ND dataframes of 100 rows each **dfs**
If classification:
While err $> 10^{-1}$:
oldw = w
If nhl $= 0$:
For i $= 1, \ldots, K$
$y_i = \mathbf{softmax}(\mathbf{X}, ws, i, \mathbf{d})$
For i $= 1, \ldots, K$
$\Delta w_i = \mathbf{n} \times \text{mean}\left(\sum_i (\mathbf{r_i} - y_i)\right)\mathbf{X}$
$w_i = w_i + \Delta w_i$

$$\text{err} = \max\left(\left|\frac{\text{w} - \text{oldw}}{\text{oldw}}\right|\right)$$

Else If nhl $= 1$:

oldv1 $=$ v1

For h $= 1, \ldots, \text{H}$

$\text{z}_\text{h} = \textbf{sigmoid}(\mathbf{X}, \text{ws}, \text{h}, \mathbf{d})$

For i $= 1, \ldots, \text{K}$

$\text{y}_\text{i} = \textbf{softmax}(\text{z}, \text{v1}, \text{i}, \text{H})$

$\Delta\text{v1}_\text{i} = \mathbf{n} \times \text{mean}(\mathbf{r_i} - \text{y}_\text{i})\text{z}$

For h $=\ 1, \ldots, \text{H}$

$$\Delta\text{w}_\text{h} = \mathbf{n} \times \text{mean}\left(\sum_\text{i}(\mathbf{r_i} - \text{y}_\text{i})\,\text{v1}_\text{ih}\right)\text{z}_\text{h}(1 - \text{z}_\text{h})\mathbf{X}$$

For i $= 1, \ldots, \text{K}$

$\text{v1}_\text{i} = \text{v1}_\text{i} + \Delta\text{v1}_\text{i}$

For h $= 1, \ldots, \text{H}$

$\text{w}_\text{h} = \text{w}_\text{h} + \Delta\text{w}_\text{h}$

$$\text{err} = \max\left(\max\left(\left|\frac{\text{w} - \text{oldw}}{\text{oldw}}\right|\right), \max\left(\left|\frac{\text{v1} - \text{oldv1}}{\text{oldv1}}\right|\right)\right)$$

Else :

oldv1 $=$ v1

oldv2 $=$ v2

For N $= 0, \ldots, \text{ND}$

$\Delta\text{v1}, \Delta\text{v2}, \Delta\text{w} = 0$

For all $(\mathbf{X}^\text{t}, \text{r}^\text{t})$ in dfs[N]:

For h $= 1, \ldots, \text{H}$

$\text{z1}_\text{h} = \textbf{sigmoid}(\mathbf{X}^\text{t}, \text{ws}, \text{h}, \mathbf{d}_\text{N})$

For l $= 1, \ldots, \text{L}$

$\text{z2}_\text{l} = \textbf{sigmoid}(\text{z1}, \text{v1}, \text{l}, \text{H})$

For i $= 1, \ldots, \text{K}$

$\text{y}_\text{i} = \textbf{softmax}(\text{z2}, \text{v2}, \text{i}, \text{L})$

$\Delta\text{v2}_\text{i} \mathrel{+}= \mathbf{n} \times (\mathbf{r_i} - \text{y}_\text{i})\text{z2}$

For l $=\ 1, \ldots, \text{L}$

$$\Delta\text{v1}_\text{l} \mathrel{+}= \mathbf{n} \times \left(\sum_\text{i}(\mathbf{r_i} - \text{y}_\mathbf{i})\,\text{v2}_\text{i}\right)\text{z2}_\text{l}(1 - \text{z2}_\text{l})\text{z1}$$

For h $=\ 1, \ldots, \text{H}$

$$\Delta\text{w}_\text{h} \mathrel{+}= \mathbf{n} \times \left(\sum_\text{l}\left(\sum_\text{i}(\mathbf{r_i} - \text{y}_\text{i})\,\text{v2}_\text{i}\right)\text{z2}_\text{l}(1 - \text{z2}_\text{l})\right)\text{v1}_\text{h}\text{z1}_\text{h}(1 - \text{z1}_\text{h})\mathbf{X}$$

For i $= 1, \ldots, \text{K}$

$\text{v2}_\text{i} = \text{v2}_\text{i} + \Delta\text{v2}_\text{i}$

For l $= 1, \ldots, \text{L}$

$\text{v1}_\text{l} = \text{v1}_\text{l} + \Delta\text{v1}_\text{l}$

For h $= 1, \ldots, \text{H}$

$\text{w}_\text{h} = \text{w}_\text{h} + \Delta\text{w}_\text{h}$

$$\text{err} = \max\left(\max\left(\left|\frac{\text{w} - \text{oldw}}{\text{oldw}}\right|\right), \max\left(\left|\frac{\text{v1} - \text{oldv1}}{\text{oldv1}}\right|\right), \max\left(\left|\frac{\text{v2} - \text{oldv2}}{\text{oldv2}}\right|\right)\right)$$

If regression:

While $\text{err} > 10^{-1}$:

$\text{oldw} = \text{w}$

If $\text{nhl} = 0$:

$\text{y} = \text{w}^{\text{T}}\mathbf{X}$

$\Delta\text{w} = \mathbf{n} \times \text{mean}\big((\mathbf{r_i} - \mathbf{y_i})\mathbf{X}\big)$

$\text{w}_\text{i} = \text{w}_\text{i} + \Delta\text{w}_\text{i}$

$\text{err} = \max\left(\left|\frac{\text{w} - \text{oldw}}{\text{oldw}}\right|\right)$

Else If $\text{nhl} = 1$:

$\text{oldv1} = \text{v1}$

For $\text{h} = 1, \dots, \text{H}$

$\text{z}_\text{h} = \textbf{sigmoid}(\mathbf{X}, \text{ws}, \text{h}, \mathbf{d})$

$\text{y} = \text{v1}^{\text{T}}\text{z}$

$\Delta\text{v1} = \mathbf{n} \times \text{mean}\big((\mathbf{r} - \text{y})\text{z}\big)$

For $\text{h} = 1, \dots, \text{H}$

$\Delta\text{w}_\text{h} = \mathbf{n} \times \text{mean}(((\mathbf{r} - \text{y})\text{v1}_\text{h})\text{z}_\text{h}(1 - \text{z}_\text{h})\mathbf{X})$

$\text{v1} = \text{v1} + \Delta\text{v1}$

For $\text{h} = 1, \dots, \text{H}$

$\text{w}_\text{h} = \text{w}_\text{h} + \Delta\text{w}_\text{h}$

$\text{err} = \max\left(\max\left(\left|\frac{\text{w} - \text{oldw}}{\text{oldw}}\right|\right), \max\left(\left|\frac{\text{v1} - \text{oldv1}}{\text{oldv1}}\right|\right)\right)$

Else :

$\text{oldv1} = \text{v1}$

$\text{oldv2} = \text{v2}$

For $\text{N} = 0, \dots, \text{ND}$

$\Delta\text{v1}, \Delta\text{v2}, \Delta\text{w} = 0$

For all $(\mathbf{X}^{\text{t}}, \text{r}^{\text{t}})$ in dfs[N]:

For $\text{h} = 1, \dots, \text{H}$

$\text{z1}_\text{h} = \textbf{sigmoid}(\mathbf{X}^{\text{t}}, \text{ws}, \text{h}, \mathbf{d}_\text{N})$

For $\text{l} = 1, \dots, \text{L}$

$\text{z2}_\text{l} = \textbf{sigmoid}(\text{z1}, \text{v1}, \text{l}, \text{H})$

$\text{y} = \text{v2}^{\text{T}}\text{z2}$

$\Delta\text{v2} \mathrel{+}= \mathbf{n} \times (\mathbf{r} - \text{y})\text{z2}$

For $\text{l} = 1, \dots, \text{L}$

$\Delta\text{v1}_\text{l} \mathrel{+}= \mathbf{n} \times \big((\mathbf{r} - \text{y})\text{v2}\big)\text{z2}_\text{l}(1 - \text{z2}_\text{l})\text{z1}$

For $\text{h} = 1, \dots, \text{H}$

$\Delta\text{w}_\text{h} \mathrel{+}= \mathbf{n} \times \sum_\text{l}\Big(\big((\mathbf{r} - \text{y})\text{v2}\big)\text{z2}_\text{l}(1 - \text{z2}_\text{l})\Big)\text{v1}_\text{h}\text{z1}_\text{h}(1 - \text{z1}_\text{h})\mathbf{X}$

$\text{v2} = \text{v2} + \Delta\text{v2}$

For $\text{l} = 1, \dots, \text{L}$

$\text{v1}_\text{l} = \text{v1}_\text{l} + \Delta\text{v1}_\text{l}$

For $\text{h} = 1, \dots, \text{H}$

$\text{w}_\text{h} = \text{w}_\text{h} + \Delta\text{w}_\text{h}$

$$\text{err} = \max\left(\max\left(\left|\frac{\text{w} - \text{oldw}}{\text{oldw}}\right|\right), \max\left(\left|\frac{\text{v1} - \text{oldv1}}{\text{oldv1}}\right|\right), \max\left(\left|\frac{\text{v2} - \text{oldv2}}{\text{oldv2}}\right|\right)\right)$$

In the pseudocode above, the ***sigmoid*** function is the logistic function as shown below:

$$\text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x} + w_0) = \frac{1}{1 + \exp\left[-(\boldsymbol{w}^T\boldsymbol{x} + w_0)\right]}$$

The ***softmax*** function (Bridle,1990):

$$\text{softmax}(\text{x}, \text{w}, \text{i}, \text{K}) = \frac{\exp(\boldsymbol{w}_i^T\boldsymbol{x} + w_{i0})}{\sum_{j=1}^{K}\exp(\boldsymbol{w}_j^T\boldsymbol{x} + w_{j0})}, i = 1, \dots, K$$

Tuning and 5-fold cross-validation

Primarily, 10% of our dataset is extracted as a validation set for tuning purposes, the remaining 90% of the data underwent 5-fold cross-validation. The data is split into five parts (or folds) and the model is trained on each of the folds accordingly. In each run, one of the five folds is used as a test set and the remaining four folds are concatenated and used as the training set.

The model is tuned by performing backward propagation on the training set by varying the number of hidden dimensions on each hidden layer from 1 to the number of inputs of that respective layer. For each iteration of the hidden dimensions, the learning rate is tuned by varying the values of n from 1 to 0.1. The largest n value that results in the convergence of the coefficients in the least amount of epochs is chosen. Furthermore, the number of hidden dimensions that result in the lowest error when testing our model on the validation set is chosen.

Using the converged coefficients, the model proceeds to predict the values of the test sets as follows:

Classification:

No Hidden Layers: $y_i = w_i^T x + w_{i0}, i = 1, \dots, K$

1 Hidden Layer: $y_i = softmax(z, \text{v1}, i, K), i = 1, \dots, K$

2 Hidden Layers: $y_i = softmax(z2, \text{v2}, i, K), i = 1, \dots, K$

The class *i* that results in the highest value of *y* is selected. The performance of the model is determined by calculating the classification errors of each of the folds by dividing the number of incorrect predictions over the total size of the folds, then calculating the mean of the errors.

Regression:

No Hidden Layers: $y = w^T x + w_0$

1 Hidden Layer: $y = \text{v1}^T z + \text{v1}_0$

2 Hidden Layers: $y = \text{v2}^T z2 + \text{v2}_0$

The performance of the model is determined by calculating the mean squared error of each of the folds, then calculating the mean of the fold errors.

3. Datasets

The following datasets were used in this project:

1. Breast Cancer
2. Glass
3. Soybean (Small)
4. Abalone
5. Computer Hardware
6. Forest Fires

Dataset 1: Breast Cancer
This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The dataset contains 699 points and 2 classes: Benign (denoted by '2') and Malignant (denoted by '4'). Including the class variable 'Class', the data contains a total of 11 features. 16 missing values need pre-processing and are all under the feature 'Bare Nuclei'. The missing values were filled by separating the data into the 2 different classes and producing sampled data based on the median of the 'Bare Nuclei' feature of the 2 classes. The feature attributes were scaled to have their values between -1 and 1.

Dataset 2: Glass
This study of the classification of types of glass was motivated by a criminological investigation. The dataset contains 214 instances, 6 classes, and 11 columns. The feature attribute values were scaled to have their values between -1 and 1.

Dataset 3: Soybean (Small)
The dataset consists of a small subset of the original soybean database. The dataset contains 47 instances, 4 classes, and 35 columns. Columns 10, 12-18, and 28-33 contained only 1 unique value and were dropped before modeling. The feature attribute values were scaled to have their values between -1 and 1.

Dataset 4: Abalone (Regression)
Predicting the age of abalone from physical measurements. The dataset contains 4177 instances and 9 columns. The values were normalized before entering them into the regression algorithm.

Dataset 5: Computer Hardware (Regression)
This dataset consists of relative CPU Performance Data. The estimated relative performance values were estimated by the authors using a linear regression method. The results are indicated in the data set as "ERP." The dataset contains 209 instances and 10 columns. The values were normalized before entering them into the regression algorithm.

Dataset 6: Forest Fires (Regression)
This is a difficult regression task, where the aim is to predict the burned area of forest fires, in the northeast region of Portugal, by using meteorological and other data. The dataset contains 517 instances and 13 columns. The values were normalized before entering them into the regression algorithms. The columns "month" and "day" were dropped during pre-processing as it was assumed that the meteorological data would be more productive.

4. Results

Dataset 1: Breast Cancer (Classification)
Modeling the Breast Cancer dataset with our algorithms we get the following:

Table 1: Dataset 1 Results

Dataset 1: Breast Cancer				
N Layers	Dimensions	n	Epochs	Error
0	0	1	5	8.73
1	4	1	149	5.56
2	3,2	0.01	197	5.56
Average		**0.67**	**117**	**6.62**

Dataset 2: Glass
Modeling the Glass dataset with our algorithms we get the following:

Table 2: Dataset 2 Results

Dataset 2: Glass				
N Layers	Dimensions	n	Epochs	Error
0	0	1	13	23.08
1	2	1	198	17.95
2	3,2	0.1	139	20.51
Average		**0.7**	**117**	**20.51**

Dataset 3: Soybean (Small)
Modeling the Soybean (Small) dataset with our algorithms we get the following:

Table 3: Dataset 3 Results

Dataset 3: Soybean (Small)				
N Layers	Dimensions	n	Epochs	Error
0	0	1	27	25
1	4	1	144	0
2	1,1	0.1	35	75.00
Average		**0.7**	**69**	**33.33**

Dataset 4: Abalone
Modeling the Abalone dataset with our algorithms we get the following:

Table 4: Dataset 4 Results

Dataset 4: Abalone				
N Layers	Dimensions	n	Epochs	Error
0	0	0.1	37	0.5
1	4	1	952	0.45
2	3,2	0.01	83	0.89
Average		**0.37**	**357**	**0.61**

Dataset 5: Computer Hardware
Modeling the Computer Hardware dataset with our algorithms we get the following:

Table 5: Dataset 5 Results

Dataset 5: Computer Hardware				
N Layers	Dimensions	n	Epochs	Error
0	0	0.1	6	0.06
1	3	1	84	0.36
2	1,1	0.01	2	0.44
Average		**0.37**	**31**	**0.29**

Dataset 6: Forest Fires
Modeling the Forest Fires dataset with our algorithms we get the following:

Table 6: Dataset 6 Results

Dataset 6: Forest Fires				
N Layers	Dimensions	n	Epochs	Error
0	0	0.1	27	0.12
1	4	1	756	1.48
2	3,1	0.01	89	0.64
Average		**0.37**	**291**	**0.75**

5. Discussion

Errors tended to decrease as the number of layers increased from 0 to 1 in Datasets 1 to 4. Dataset 5 saw an increase in error, and Dataset 6 once again failed to be modeled. Neither of the datasets saw a decrease in error as the number of layers increased from 1 to 2, most actually saw an increase.

Referring to Table 3, the Soybean Dataset saw an increase in error from 0 to 75 as the number of layers increased from 1 to 2, this is due to a phenomenon in backpropagation known as the vanishing gradient problem. This is when, due to training multiple layers and propagating the error to the previous layers, the gradient becomes vanishingly small preventing the coefficients from changing and converging.

The vanishing gradient problem occurs more frequently as the number of layers increases, however, research has shown that implementing a rectifier activation function (ReLU) instead of a sigmoid activation function may address this issue.

6. Conclusion

In conclusion, our hypothesis did not stand with our model proving to be unreliable with more hidden layers. As for model accuracy, our hypothesis stood for all datasets except for Dataset 5.

Comparing the results of our previous models with our current results:

Table 7: Comparison with Other Classification Models

Dataset	Classification Error		
	Naïve Bayes	Logistic Regression	Adaline
Breast Cancer	3.2	3.81	4.15
Glass	36.8	15.53	31.76
Soybean	0	0	4.22
Average	**13.33**	**6.45**	**13.38**

Based on Table 7, Logistic regression is the most effective classification model for our datasets.

Table 8: Comparison with Other Regression Models

Dataset	MSE	
	K-NN	CART
Abalone	0.67	0.7
Computer Hardware	0.42	0.7
Forest Fires	1.11	1.28
Average	**0.73**	**0.89**

Based on Table 8, our Feedforward Neural Network model is more effective for the Abalone and Computer Hardware datasets. The Forest Fires dataset failed to be modeled with any of our algorithms so far.

7. References

David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams.
Learning representations by back-propagating errors
Nature, 323: 533-536, 1986.

John S. Bridle
Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition.
In *Neurocomputing, NATO ASI Series (Series F: Computer and Systems Sciences)*, vol 68, pages 227-236, Springer, Berlin, Heidelberg.