Project 4: Neural Networks

Dongjun Cho DCHO13@JH.EDU
Programming Project 4
605.649
13 April 2022

Abstract

This project is intended to implement the neural network for classification and regression. This algorithm was applied to discrete and continuous-valued data using six data sets obtained from the UCI Machine Learning Repository.

1. Introduction

The neural network is one of the algorithms for machine learning, and it is also called by Artificial neural network (ANN). ANN is an artificially constructed network that mimics the signaling process of nerve cells in the brain. The reason why it's meaningful to artificially configure these networks on computers is that they allow computers to learn and recognize human high capabilities.

The purpose of the paper is to research and implement a simple neural network, Feedforward neural network with backpropagation, and Autoencoder. In this paper, we will implement two different simple neural network algorithms: logistic regression for classification and linear regression for regression. It will implement the Feedforward that is trained with backpropagation. Feedforward neural network is a supervised learning algorithm to estimate the non-linear relationship between the set of inputs, and the output. Also, it uses a hidden layer to identify the feature that it has to learn about the data. The main concern of the neural network is the curse of dimensionality. The curse of dimensionality means that the error increases with the increase in the number of features. To help reduce the curse of dimensionality Alpaydın (2020), Autoencoder will be implemented in this paper. Autoencoder reduces the feature spaces that are projected into.

In this paper, Feedforward neural network with backpropagation has 4 layers: Input layer, First Hidden layer, Second Hidden layer, and Output layer. Since Feedforward neural network has a more complex layer the network, Feedforward neural network will outperform a simple network. Since Autoencoder can reduce the curse of dimensionality of the Feedforward neural network, Autoencoder will outperform than Feedforward neural network with backpropagation.
Hypothesis 1: We hypothesize that Autoencoder will outperform feedforward neural networks with back-propagation, and feedforward neural networks with back-propagation will outperform simple networks. To test this hypothesis, we classification accuracy score for classification and mean squared error for regression tasks on several datasets from the UCI repository.

In Section 2, we describe how each algorithms works, in section 3, we describe the experimental approach to the algorithm, in section 4, we present the result of our experiment. We discussed the behavior of the algorithm in section 5, and then we conclude the discussion with the result and possible future works in section 6.

2. Description of Algorithms

2.1 Simple Networks

2.1.1 Logistic Regression

Binary Classes Logistic Regression is the machine learning algorithm for classification tasks that predicts the result using regression analysis and sigmoid function. For binary classes, it uses the sigmoid function to compute the probability of each class.

$$S(x) = \frac{1}{1+e^{-x}}$$

At first, it initializes the weight vectors with random values ranging from -0.01 to +0.01. Then, it repeats the updating rule procedure until it reaches to maximum iteration. It uses gradient descent to find the optimal learning parameters for the model, and it tries to minimize the loss function. It uses cross-entropy for the classification of the loss function.

$$E(w,\ w_o|X) = -\sum_t r^t log y^t + (1-r^t) log(1-y^t)$$

It uses gradient descent to minimize cross-entropy, equivalent to maximizing the likelihood or the log likelihood Alpaydın (2020). The gradient of the loss function and the following weight update rule is shown below;

$$\triangle w_j = -\eta \frac{dE}{\mathrm{dw}_x} = \eta \sum_t (r^t - y^t)$$

Multiple Classes Handling multiple classes for logistic regression is very similar to handling binary classes. However, logistic regression for multiples classes uses the softmax function to compute the probability of each class. For cross entropy for multi-class classification is

$$E(w_i,\ w_{i0_i}|X) = -\sum_t \sum_t r_i^t log y_i^t$$

Just like binary class logistic regression, it uses gradient descent for the weight update rule. It continues to update the weights until it reaches the maximum iteration.

$$\triangle w_{jo} = \eta \sum_t (r_j^t - y_j^t)$$

Prediction After the training, it has a final weight. For logistic regression for binary classes, it computes the output of each discriminant using the sigmoid function with the final weight. When it computes for each discriminant, if the value is greater than 0.5 which is the decision boundary, then it chooses class 1, and class 2 otherwise. This process is to minimize the number of misclassification. For logistic regression for multiple classes, it computes the output of each discriminant using the softmax function. After computing each discriminant using either sigmoid or softmax, the class that has the largest value are best candidate class for that distance.

2.1.2 Simple Linear Network

For simple linear network, it uses linear regression for regression tasks to predict the variable based on the other variable. Linear regression is a linear approach to model linear relationship between feature variables. It computes the weight sum for every feature dataset. Then, it uses gradient descent to minimize the cost function/MSE (Mean Squared Error) for the linear regression. It keeps updating the weight based on the gradient weight rule until it reaches the number of maximum iterations.

2.2 Feedforward Network

Feedforward neural network is a supervised learning algorithm to estimate the non-linear relationship between the set of inputs, and the output. It is used to solve classification and regression tasks. For this project, it has 4 layers for the feedforward network with backpropagation: input layer, first hidden layer, second hidden layer, and output layer. The hidden layer in Feedforward Network acts as a detector to identify what it has to learn about the non-linear feature data. For nonlinear regression, it computes the output using

$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_o$$

In the nonlinear regression formula, z_h is sigmoid function. Based on this formula, the error function over the every feature in the regression is

$$E(W, v|X) = \frac{1}{2}\sum_t \left(r^t - y^t\right)^2$$

For hidden nodes, it consists of a perceptron with hidden units as inputs units. So, it uses least-squares rules to update the weights.

$$\triangle v_h = \eta \sum_t (r^t - y^t) z_h^t$$

For output nodes, it consists of a perceptron with the hidden unit as output units. It updates the first layer weight in the output node. It needs to apply the chain rule differentiation to get desired output.

$$\triangle w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}}$$

$$\triangle w_{hj} = -\eta \sum_t \frac{\partial E^t}{\partial y^t} \frac{\partial y^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}}$$

$$\frac{\partial E^t}{\partial y^t} = -(r^t - y^t)$$

$$\frac{\partial y^t}{\partial z_h^t} = v_h$$

$$\frac{\partial z_h^t}{\partial w_{hj}} = z_h^t (1 - z_h^t) x_j^t$$

$$\triangle w_{hj} = \eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

In the formula above, $\left(r^t - y^t\right) v_h$ acts likes error for hidden unit h. This error is backpropagated from the error to the hidden unit Alpaydın (2020).

For Feedforward Network, it first initializes weights $(w_h j,\ v_h)$ for the entire network with the value ranging from -0.01 to +0.01. For each feature in the dataset, it computes for the $\triangle v_h$ and $\triangle w_{hj}$. Then, it updates the weight for the first layer and second layer. Since the Feedforward network with backpropagation in this project uses 2 hidden layers, it repeats the same procedures. It keeps updating the weights until it reaches the maximum number of iterations.

2.3 Autoencoder

Autoencoder is very similar to the Feedforward network. The purpose of Autoencoder is to help reduce the curse of dimensionality from a normal Feedforward network. In the Feedforward network, it has 4 layers: the input layer, the first hidden layer, the second hidden layer, and the output layer. The Autoencoder also has 4 layers: input layer, encoding layer, hidden layer, and output layer. For Autoencoder, it uses the same method of updating weight. In the encoding layer, it tries to reduce the feature spaces that are projecting into by setting the number of smaller nodes than the input/output node. For the hidden layer, it updates the weight for feature space and hidden nodes. After the autoencoder-based network is constructed, it trains those layers using a feedforward neural network with backpropagation.

3. Experimental Approach

3.1 Tuning

For this project, there are several parameter factors that need to consider. For the tuning parameters, there is the number of the hidden layer, number of encoding layers, learning rate, and maximum iteration. The number of layers is the number of nodes that constructs the networks. For learning rate, it uses for updating weights for stochastic gradient descent. The maximum iteration determines the maximum number of weight updates for the network.

For logistic regression, there are two parameters that needs to consider: learning rate, and maximum iteration. The number of learning rate sets to $\{0.01,\ 0.05,\ 0.1\}$. The number of maximum iteration sets to $\{10,\ 20,\ 50\}$. For Linear regression, learning rate is the parameter that needs to consider. The number of learning rate sets to $\{0.01,\ 0.05,\ 0.1,\ 0.5\}$.

For Feedforward neural network with backpropagation, there are three parameters that need to consider: number of the hidden node, learning rate, and maximum iteration. For this report, it has two hidden layers: the first hidden layer, and the second hidden layer. For the implementation, the number of hidden layers sets to the same for both the first hidden node and second hidden node. The number of hidden node sets to $\{5,\ 6,\ 7\}$. The number of learning rate sets to $\{0.001,\ 0.005,\ 0.01,\ 0.01,\ 0.05,\ 0.1\}$. The number of maximum iteration sets to $\{10,\ 20\}$.

For Autoencoder, there are four parameters: number of encoding node, number of hidden node, learning rate, and maximum iteration. The number of hidden node sets to $\{2,\ 3\}$. The number of hidden node sets to $\{5,\ 6,\ 7\}$. The number of learning rate sets to $\{0.001,\ 0.005,\ 0.01,\ 0.01,\ 0.05,\ 0.1\}$. The number of maximum iteration sets to $\{10,\ 20\}$.

To tune these parameters, the total dataset is split into training (80%) and tuning (20%). Using 20% of the dataset, it processes the tuning to find the best parameter combination. Once it finds the best parameter, it is applied to the training set. With the best parameter values, the training set is applied to a 5-fold cross-validation.

4. Results of Experiments

For classification, the table showed the accuracy score for the unpruned and pruned decision tree. For regression, it shows the best parameter and the mean squared error for unpruned and pruned decision trees.

4.1 Breast Cancer Dataset

The Breast Cancer set represents a classification problem used to distinguish whether it is benign or malignant based on 10 feature attributes. Table 1 is demonstrated the accuracy score for logistic regression, Feedforward with Backpropagation, and Autoencoder.

Table 1: Accuracy Score of Logistic Regression, FeedForward, Autoencoder

	H1/E1	H2	Max Iteration	η	1	2	3	4	5	Avg
Logistic Regression	-	-	50	0.1	77.68	82.88	81.08	61.26	84.68	77.52
FeedForward	6	6	20	0.1	83.04	98.20	94.59	97.30	97.30	94.09
Autoencoder	3	5	10	0.1	85.71	98.2	94.59	96.4	98.2	94.62

4.2 Car Evaluation

The Car Evaluation set represents a classification problem used to distinguish car acceptability based on 6 feature attributes. Table 2 is demonstrated the accuracy score for logistic regression, Feedforward with Backpropagation, and Autoencoder.

Table 2: Accuracy Score of Logistic Regression, FeedForward, Autoencoder

	H1/E1	H2	Max Iteration	η	1	2	3	4	5	Avg
Logistic Regression	-	-	10	0.01	94.22	79.35	66.67	65.58	60.14	73.19
FeedForward	7	7	10	0.1	86.28	86.59	83.70	86.96	66.67	82.04
Autoencoder	2	5	10	0.01	94.22	79.35	66.67	65.58	60.14	73.19

4.3 Congressional Vote

The Congressional Vote set represents a classification problem used to distinguish house party (republican/democrat) based on 16 features of attributes of legislation votes. Table 3 is demonstrated the accuracy score for logistic regression, Feedforward with Backpropagation, and Autoencoder.

Table 3: Accuracy Score of Logistic Regression, FeedForward, Autoencoder

	H1/E1	H2	Max Iteration	η	1	2	3	4	5	Avg
Logistic Regression	-	-	20	0.1	98.57	92.86	95.71	94.29	97.06	95.69
FeedForward	6	6	20	0.1	98.57	94.29	95.71	94.29	98.53	96.28
Autoencoder	2	7	10	0.1	97.14	94.29	94.29	94.29	98.53	95.71

4.4 Abalone

The Abalone set represents a regression problem used to distinguish the associate age of abalone based on 8 features attributes. Table 4 is demonstrated the Mean Squared Error for linear regression, Feedforward with Backpropagation, and Autoencoder.

Table 4: MSE of Linear Regression, FeedForward, Autoencoder

	H1/E1	H2	Max Iteration	η	1	2	3	4	5	Avg
Linear Regression	-	-	-	0.05	19.01	4.77	10.1	15.74	100.10	29.94
FeedForward	5	5	10	0.001	128.02	82.15	62.02	40.08	21.48	66.75
Autoencoder	2	5	20	0.001	127.95	82.15	62.02	40.08	21.48	66.74

4.5 Computer Hardware

The Computer hardware set represents a regression problem used to distinguish the CPU performance based on 10 features attributes. Table 5 is demonstrated the Mean Squared Error for linear regression, Feedforward with Backpropagation, and Autoencoder.

Table 5: MSE of Linear Regression, FeedForward, Autoencoder

	H1/E1	H2	Max Iteration	η	1	2	3	4	5	Avg
Linear Regression	-	-	-	0.5	0.12	0.47	0.49	0.87	10.96	2.58
FeedForward	6	6	20	0.001	2.78	0.47	0.47	0.3	8.6	2.52
Autoencoder	2	5	20	0.001	2.72	0.43	0.43	0.28	8.56	2.48

4.6 Forest Fires

The Forest Fires set represents a regression problem used to predict the burned area of forest fires by using 12 attributes features of meteorological data. Table 6 is demonstrated the Mean Squared Error for linear regression, Feedforward with Backpropagation, and Autoencoder.

Table 6: MSE of Linear Regression, FeedForward, Autoencoder

	H1/E1	H2	Max Iteration	η	1	2	3	4	5	Avg
Linear Regression	-	-	-	0.05	0.19	0.26	0.14	1.74	10.5	2.56
FeedForward	5	5	10	0.001	3.05	3.06	1.88	0.32	7.48	3.16
Autoencoder	2	5	10	0.001	2.98	3.03	1.83	0.33	7.48	3.13

5. Behavior of Algorithms

The performance of the autoencoder and feedforward with backpropagation was not expected. Hypothesis 1 stated that autoencoder would outperform feedforward with backpropagation, and feedforward would outperform simple linear networks. Throughout the 6 datasets (3 classification, 3 regression), 2 datasets (Breast Cancer, Computer) match the hypothesis. In breast cancer, it is very interesting that the accuracy score of Feedforward and Autoencoder made huge improvements compared to simple logistic regression. For the other two classification datasets (Car, Congressional Vote), it shows feedforward performs best. For regression tasks, autoencoder outperforms feedforward with backpropagation for every dataset. But, it is very interesting that simple linear regression performs best compared to other networks for Abalone and Forest datasets.

In terms of execution time of the algorithms, the simple network executed fast. Unlike simple network, the execution time for both Feedforward with backpropagation and autoencoder was very slow, execution time of autoencoder was the slowest. For the large dataset, the abalone dataset, it took 5 hours to finish the tuning process.

The purpose of the autoencoder is to reduce the curse of dimensionality. It tries to find that reduces the feature space that it's projecting into. Even though it tries to reduce feature space to solve the curse of dimensionality, it is very interesting that it doesn't guarantee a better performance than larger feature space.

6. Conclusion

This paper implemented the various types of the neural network: logistic regression, linear regression, Feedforward with backpropagation, and Autoencoder. Throughout this paper, I was able to learn about the learning process of the neural network. I learned that autoencoder can reduce the curse of dimensionality, but I found that it doesn't guarantee better performance. For Future analysis and implementation, it should implement to speed up the feedforward neural network with backpropagation and autoencoder.

References

Ethem Alpaydın. *Introduction to Machine Learning, fourth edition.* The MIT Press, 2020.