Journal of Machine Learning Research 1 (2020) 1-6 Submitted 11/13; Published 11/15

Backpropagation on Neural Networks

Brian Loughran BLOUGHRAN618@GMAIL.COM
Statistical Research Department
Loughran Institute
Breckenridge, CO 80424, USA

Editor: Brian Loughran

Abstract

This document describes a feed-forward neural network as well as a method of training the network, backpropagation. The feed-forward neural network described can be used to predict real world classification and regression data sets. Included in this paper is a problem statement summarizing assumptions made for the algorithm and projections on how the algorithm is expected to perform against a variety of different data sets. Also discussed is a description of the experimental approach and any processing that is done on input data to fit the feed-forward neural network algorithm. Results for each the classification and regression data sets are presented, and finally conclusions are drawn on algorithm accuracy.

1 Problem statement

Feed-forward neural networks describe a variety of different network architectures. Feed-forward networks are supervised learning algorithms that can be used to solve classification and regression problems. A variety of different feed-forward neural networks are discussed in this report. For example, feed-forward neural networks adjusted for classification problems will have a number of output nodes corresponding to the number of possible result classes, while feed-forward neural networks adjusted for regression problems will have a single output node corresponding to the result of the regression. Feed-forward neural networks can also have a varying number of hidden layers. This report discusses feed-forward neural networks with either 0, 1 or 2 hidden layers. Further, the number of nodes can be varied for each of the hidden layers.

6 data sets are used to evaluate the performance of the feed-forward neural networks proposed; those data sets are abalone, breast-cancer-wisconsin, forestfires, glass, machine and soybean-small. While abalone, forestfires and machine are regression sets, breast-cancer-wisconsin, glass and soybean-small are classification sets. Predictions are made based on trained weights. Classification loss is considered for the classification sets while mean squared error is considered for the regression sets.

For evaluating algorithm performance we use a 5-fold cross validation. 10% of the data is taken out before the 5-fold cross validation for optimizing hyperparameters. Applicable hyperparameters for feed-forward neural networks are the number of iterations the learning algorithm should follow, also known as the number of epochs, the learning rate, and the number of nodes in each hidden layer. The remainder of the data is used for the 5-fold cross validation, and performance over each of the folds is considered to get the performance over the entire set.

Based on the success of similar algorithms on previous classification data sets, it is expected that the feed-forward neural networks will be able to predict with greater than 90% accuracy on the classification sets. This is not including the glass data set, which has confounded all previous learning algorithms. The regression predictions are expected to follow very closely (within 5% on average) of each of the abalone and machine data sets, while the predictions are

expected to have more variation (within 15% on average) for the forestfires data set just because of how much more difficult the forestfires data set is than other data sets.

2 Experimental Approach

Feed-forward neural networks and the back-propagation training method is a well known algorithm, thus the algorithm will not be discussed in detail in this section. Some assumptions are made to streamline the algorithm and those assumptions are discussed in this section, including reading in data sets and assumptions made in the computations as well as implementation details.

Feed-forward neural networks expect real-valued numbers as inputs to the training process. It is also helpful to normalize the attributes in some way around 0 to ensure that different inputs do not have outsized effect on the result class. For this reason, the attributes were normalized to be in the range of -1 to +1. Some of the data sets included data that was non-numeric.

Any attribute with less than 2 real values was dropped from the training set since this did not contribute any information for the algorithm to learn. This was the case with several of the soybean attributes, such as stem, fruit spots, seed, etc.

In some cases special care must be taken to handle the input data sets to ensure that the values fed to the network are usable by the algorithm. Feed-forward neural networks expect a real-valued number. One data set that does not have exclusively real-valued numbers is machine. One method to handling non-numeric data is to one-hot encode the values, which is what is used in this case. Another method which can be used to handle non-numeric data is to encode the values of the non-numeric data to numeric values. For example, in the abalone data set, we can encode the gender of male to 1, female to -1, and infant to 0. Another example is the forestfires data set, where we can encode the days of the week to values 1 through 7, and the months of the year to the values of 1 through 12 and then follow the normalization process laid out earlier on those values. This method is deemed appropriate for this learning model because of the ability for feed-forward neural networks to learn non-linear problems.

For feed-forward neural networks, a decision needs to be made if there is any missing data in the data set. Some of the data sets do not have any missing data. Abalone, glass, forestfires, machine and soybean-small all have 0 missing attributes, thus no determinations need to be made for these data sets. Breast-cancer-wisconsin has 16 missing attribute values. This is a relatively small amount of missing data in a data set with 699 instances, thus instances with missing attributes are ignored for the breast-cancer-wisconsin data set.

For each feed-forward neural networks, a set of hyperparameters must be tuned. The hyperparameters tuned as discussed above are the number of iterations the learning algorithm should follow, also known as the number of epochs, the learning rate, and the number of nodes in each hidden layer. The way that these hyperparameters are tuned was to extract 10% of the dataset for tuning purposes. Once the tuning dataset was extracted, a training set was determined for each of the 5 validation folds. Hyperparameters were selected using a grid search, and the model with the highest classification accuracy on the validation set, or the lowest mean squared error on the regression set was selected as the hyperparameters to use for testing. The purpose of doing this is to ensure that the learning rate, number of iterations, and number of nodes in the hidden layers selected for each algorithm is appropriate for the specified data set.

An important consideration in selecting tuning and training sets is ensuring each of the sets generated (tuning set and 5 validation groups) has a representative sample of the result values. The way that this was done was to start by ordering the data set as a whole based on the result. The tuning set then took every 10^{th} data point from the set, thus resulting in a representative sample for the tuning set. Each of the validation groups took every 5^{th} data point from the remaining set resulting in each of the validation sets also having a representative sample.

Splitting the data sets in this way ensured that the tuning set and each validation fold had representative data to train and test on.

There are a few different ways to handle classification on non-binary classification sets. Methods used previously include training multiple networks to predict each of the classes and utilize a one vs. all strategy, or to train multiple networks and utilize a one vs. one strategy. Neither of those strategies were used for the feed-forward neural networks in this case. For this implementation, a multi-net was trained. This means that a single network was trained with multiple outputs, one for each outcome that needs to be predicted. This applies only to classification problems with more than 2 result classes, since regression has a single output.

Feed-forward neural networks with backpropagation can utilize a variety of activation functions. Some activation functions that can be used include the logistic activation function, the hyperbolic tangent activation function, ReLU, or linear activation functions. It is noted that linear activation functions are not preferred for networks with more than one layer since then the entire network could be mathematically compressed to a single layer. Since the logistic activation function had been used previously, the logistic activation function was re-used for the feed-forward neural network implementation.

Another experimental choice made for the regression data sets was to normalize the results of the regression between 0 and 1. This was done as a way to keep the weights in check for the different layers as well as to get the base prediction closer to the actual result for early learning iterations. It was found that learning happened quicker by doing this normalization, and had negligible effect on the accuracy of the learning long-term. Calculation of mean squared error was done on the de-normalized values to ensure that no information was lost using this normalization.

Some different options were considered in terms of running the feed-forward network with different numbers of hidden layers. Some advanced options were considered in terms of running each data set with increasing number of hidden layers and re-using those learned layers as the initial values for larger networks, similar to some of the deep-learning principals that were introduced in module 11. However, that approach was scrapped due to the complexity and the fact that running each of the experiments with different numbers of hidden layers separately had more readability in terms of organizing the outputs of the algorithm.

3 Results

The feed-forward neural network algorithm discussed above is run on all 6 of the data sets abalone breast-cancer-wisconsin, forestfires, glass, machine and soybean-small. Each of the data sets is run with 0 hidden layers, 1 hidden layer and 2 hidden layers. This results in a total of 18 runs for the feed-forward network. Each of the runs produces an output which is located in the output folder of the run directory. The format of the output file is of the form <set>_<number of hidden layers>_hiddenlayers.output.txt. Thus, running abalone with 1 hidden layer would produce output of the name abaline_1_hiddenlayers.output.txt in the output folder of the run directory.

There is a wealth of information included in the output files, and referencing those files should give greater insight into low-level items not included in this report. There are two levels of reporting available in the output files. There is the standard level, which is the level that each of the output files included in the submitted module conform to. The standard level is a more readable version of the output data, and includes the following:

- An echo of the options.yaml file specifying preprocessing options for the datafile as well as any special considerations for the preprocessing such as attributes to drop
- The preprocessed data, as well as the splits of the data into the validation set and the 5 cross-fold validation sets

- For each of the tuning attempts, the learning rate, the number of iterations and the number of hidden nodes per layer
- For every 100 iterations, the current epoch and the current error value (this is a nice way to watch the model learn)
- The trained network, including the following:
    - The number of hidden layers
    - The number of input nodes, corresponding to the number of inputs
    - The number of nodes in each of the hidden layers
    - The number of outputs, corresponding to the number of result classes for classification or 1 for regression
    - For each layer the array of nodes
    - For each node the array of input weights, corresponding to the number of nodes in the previous layer
- For the trained network the predicted and actual value (for classification the predicted class and for regression the predicted value)
- After tuning the hyperparameters, the hyperparameters that best meet the validation fold
- The resulting classification accuracy or mean squared error for each fold
- The set name
- The average classification accuracy over each of the 5 cross-validation folds for classification or the total mean squared error for regression

There is also a verbose level of reporting available. It was found that some of the items requested in the project document made the output tremendously difficult to read and expanded the size of the output files beyond what was acceptable. Flipping the verbose flag from False to True in the config.yaml file in the project directory will activate verbose reporting, and will include the following information not included in the standard level of output:

- For every iteration, the current epoch and the current error value (this is an even nicer way to watch the model learn)
- Forward propagation, including the following:
    - The inputs for each layer
    - The node activation level and corresponding sigmoid value for that activation level
- Gradient calculation, including the following:
    - For each node in each layer the sigmoid activation for that node
    - For each node in each layer the corresponding gradient calculated with the derivative of the sigmoid activation
- Weight update including the following:
    - For each node in each layer the current weight value and the calculated gradient

- The new weight value

Closely analyzing the results files are a great way to look at the lower level implementation components for the feed-forward neural networks. Some useful things that can be done with the information contained includes basic debugging, following along with the algorithm logic, and further analysis into incorrect classifications. The feed-forward neural network outputs relative confidence values for each prediction. A further look into the confidence values, such as evaluating whether the correct result class had a similar confidence value as the predicted class can shed light on whether the algorithm was simply unsure of a particular prediction, if the instance is noise, or if perhaps the algorithm has not fully trained its weights. Tracing poor regression predictions is more difficult than classification.

We can summarize the results of the classification loss against the classification sets breast-cancer-wisconsin, glass and soybean-small. This is done using an averaged classification loss on each of the 5 cross validation folds and averaging the accuracy as a percentage. The results of this are included at the base of each of the *.output.txt files and is tabulated in Table 1. Results are shown for 0 hidden layers, 1 hidden layer, and 2 hidden layers.

	Classification Accuracy		
Set	0 Hidden Layers	1 Hidden Layer	2 Hidden Layers
breast-cancer-wisonsin	96.1%	96.4%	95.8%
glass	62.0%	70.4%	64.6%
soybean-small	100%	100%	100%

Table 1: Summary of classification accuracy for feed-forward neural network

The accuracy of the feed-forward neural network is impressive with regard to classification loss. The exception to the rule is the glass set, and reasons for this are discussed in the conclusions section.

Similarly, we can summarize the results of the mean squared error against the regression sets abalone, forestfires and machine. Shown in the table are the total mean squared error over all 5 cross validation folds. The results of this are included at the base of each of the *.output.txt files and is tabulated in Table 1. Results are shown for 0 hidden layers, 1 hidden layer, and 2 hidden layers.

	Regression Accuracy		
Set	0 Hidden Layers	1 Hidden Layer	2 Hidden Layers
abalone	18851	16736	16172
forestfires	2744264	2125827	2242312
machine	133835	89680	173527

Table 1: Summary of classification accuracy for logistic regression algorithm

4 Algorithm Behavior

Feed-forward neural network is an algorithm which aims to reduce the loss of the models against the loss function through incremental updates. Interesting to consider is the behavior of the result of the loss function as the algorithms increment. There are cases where the loss may increase for a given increment, this is usually in an attempt by the network to get out a local maximum. For appropriate learning rates, the loss will likely never be worse than the first iteration, with the loss improving rapidly in the early iterations. As the loss approaches an asymptotic minimum, diminishing returns are observed for each subsequent iteration over the training data. Choosing an appropriate number of iterations for each of logistic regression and adaline is key to reducing compute costs for both of the algorithms.

This is the behavior of the result of the loss function when appropriate learning rates are selected. However, if the learning rate is too high, there can be an oscillating effect on the result of the loss function, with the loss function behaving unpredictably as the model over-compensates in trying to update the weight vectors. Of course, this is not desired behavior. On the opposite end of the spectrum, if the learning rate is set too low for the number of iterations specified, the number of iterations may terminate before the algorithm is able to near an asymptotic minimum for the loss function. Tuning the values of the learning rate and the number of iterations for each feed-forward neural networks are thus imperative to getting an accurate prediction. Looking closely at the tuning sections of the *.output.txt files produced for each algorithm show instances where the learning rate and number of iterations are not appropriate for the given data set. This is as expected considering each data set will be optimized by a different combination of learning rate and number of iterations.

Another behavior that is observed for the classification and regression data sets is the relative performance of the algorithm with differing numbers of hidden layers. One interesting observation is that the networks with just one hidden layer typically perform better than the networks with 0 or 2 hidden layers. While the reason that networks with 1 hidden layer perform better than those with 2 hidden layers should be trivial, the reason that networks with 2 hidden layers perform worse than with 1 hidden layer may be due to the vanishing gradient problem, or perhaps because the number iterations was too small for the models. Because of this vanishing gradient, it would be predicted that networks with 3 hidden layers would perform even worse that those with 2 hidden layers. Some steps that can be taken to help remediate this vanishing gradient problem include more training iterations, a higher learning rate, or using deep learning principals.

5 Conclusion

The problem statement predicted that each of the classification sets would have a classification accuracy of greater than 90% on breast-cancer-wisconsin and soybean-small. Classification was perfect against soybean-small, which was relatively unsurprising due to the separable nature of the data set. Performance on breast-cancer-wisconsin was also excellent. The glass data set continues to confound learning algorithms with at best 70% accuracy with 1 hidden layer, indicating that it may be a challenging set to learn.

On the regression side, the performance of the sets abalone, forestfires and machine are a bit harder to parse. This is due to the fact that mean squared error is dependent on the number of data points in the set and the magnitude of the data (larger data ranges produce higher mean squared error values). That being said, a closer look at the results in the output files can give some more insight into how well the regression performs. Predicted was the regression predictions are expected to follow very closely (within 5% on average) of each of the abalone and machine data sets, while the predictions are expected to have more variation (within 15% on average) for the forestfires data set. A closer look at the output data reveals that predictions for abalone were within on average of 10% of the actual value, predictions for machine were within on average 8% and predictions for forestfires were within on average within 20 acres. This was slightly higher than projected, however these predictions are still pretty good for regression problems.