Testing the viability of bidirectional multi-layer perceptrons and low population sine-distribution evolutionary algorithms in web snippet classification

Edan Landow

Research School of Computer Science, Australian National University

Email: u6378020@anu.edu.au

1 Abstract

A previous experiment determined which lengths of the snippet provided in web search results were most effective in minimising the time to click and maximising the accuracy of web searches[2]. During this experiment, other variables were recorded such as the fixation time of the searchers on elements of the result page. This provides a categorical dataset with variables that are affected by the length, but much more by undocumented noise such as the state of the searcher before starting the search. Noise is a restricting factor in many classification problems and many studies have been conducted on minimising its effect. Finding a model which can effectively model extremely noisy data would be useful, and so there is benefit from testing various models on the noisy data to determine their performance. One approach the does not seem to have been tested as a classifier for extremely noisy data is evolutionary algorithms, using sine waves with periods selected from a gaussian distribution as the 'genes.' This model will be abbreviated to the SDE model. Another potential approach is the Bidirectional Multilayer Perceptron model, or BDNN. This has been shown in another study to be able to make predictions on moderately noisy data by classifying student final grades with the final marks. The SDE and BDNN both failed to develop any useful model of the noisy data, however other benefits of each model were discovered during the study.

2 Introduction

Many studies have already attempted to compensate for noise in data[4] and perform analysis on data with noise. However, none are completely effective, so new models that can handle noise are desirable.

The bidirectional multi-layer perceptron network (BDNN) is a network of perceptrons that is designed to be invertible: given the output it should ideally be able to predict the input.

The multilayer perceptron (MLP) model is already in common usage to solve classification problems. However, not much analysis has gone in to the BDNN model. As such, we don't know how it performs against extremely noisy data.

The evolutionary algorithm is already a popular solution to many problems[5] including classification with less noise than our data. Conventionally, the functions used by the algorithm are fixed, and only scalar weights are updated with each generation. This creates a bound on how good the model can be, as the true relation to the data usually is not spanned by a set of initially chosen functions. For the SDE model, I introduced a chance for the functional basis to change each generation in addition to scalar weights. A useful property of the sine wave function is the property that sin(a*x) is orthogonal to sin(b*x) for a/=b, which makes it a great choice to encompass as many potential solution functions as practical and prevent function redundancy. Resultingly, it may have the potential to model functions to much higher precision than a conventional evolutionary model with fixed functions. This creates a risk of overfitting, however a regularisation term in the measured error is included to minimise this risk.

Both models were trained and tested on the extremely noisy web snippet dataset to determine if they are suitable for modelling such data. The models were also tested using the Iris flower classification data as a control. This data is already known to be able to be modelled, and as such is a safety feature to ensure the implementation of the models is not critically broken, and that loss of performance is due to the noise of the web snippet dataset.

3 Method

For the SDE model, the individuals were defined by a set of functions and weights for each output vector. The functions were defined as $x_0 + x_1*\sin(i*x_2) + x_3*\cos(i*x_4)$, where x_i is a random constant from the gaussian distribution of 0 mean and standard deviation equal to the scale factor and i is defined as $v_0 \vartriangle v_1 \vartriangle v_2 \vartriangle v_3$. Each vi is one of the input parameters, and the $(a \vartriangle b)$ operator is selected as one of the following operations with equal probability:

a+c*b, where c is a random constant from the 0 mean, scale factor variance gaussian distribution.

a*b^c, where c is defined as above.

a/b^c, where c is defined as above, and returns 0 if b is 0.

Individuals were initialised with 8 of these functions generated per output vector, with associated weights for each taken from the 0 mean, 1 variance gaussian distribution.

This structure is visualised in figure 1.

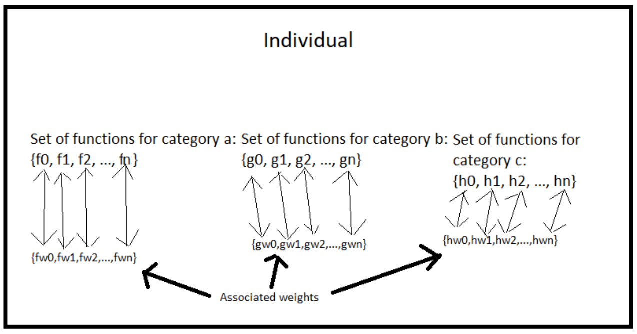

Figure 1

Each generation, 3 individuals were randomly selected from the population and competed to have the lowest error. This error was defined by taking the squared difference between the boolean value of whether the data corresponded to a category, and the sigmoid of the sum of each function associated with that category times the associated weight of the function. An additional term of the square of lambda * the sum of all weights was included to limit overfitting, where lambda is a constant. This was repeated for each of a random selection of 12 of the training points and summed. After error calculation, the individual with the highest error was removed and replaced with a child of the two other individuals. This was done by giving the child functions from both parents and the average associated weights between them, where the associated weight was 0 for a parent that didn't have the function. Each function also had a constant chance of mutation. Mutated functions did not actually change, however a constant from the gaussian distribution of 0 mean and variance equal to the constant mutation intensity was added to its associated weight. If this caused the sign of the associated weight to change, the function was removed instead. There was also a constant chance of another function being generated which was not in either parent. This function was generated in the same was as the initialisation functions, and the associated weight was taken from the 0 mean gaussian distribution with variance equal to the insert mutation intensity. When the desired generation was reached, each member of the population was

The following constants were determined to be effective and were used for recorded data.

Scale: 1

Lambda: 0.00001

Mutation frequency: 0.08

Mutation intensity: 0.04

Insert mutation frequency: 0.3

Insert mutation intensity: 0.3

Population: 20

Generations: 1000

The model was tested on the Iris flower control dataset, as well as the real data. The data was split into 80% training data and 20% testing data. It was ensured that the training data had an even quantity of all categories to prevent the model from becoming biased towards the most observed category. The categories were converted to integer representation; however, no further pre-processing was done on the data. The confusion matrix of the trained and untrained model was taken by predicting the category of each element of the testing data. To make a prediction with the model, each of the function sets associated with a category were applied to the input vector, multiplied by their corresponding weight and added. The category who's associated function set returned the largest value from this method was returned as the predicted category.

For testing of the BDNN model, I started by constructing a single layer network modelling the network used in the BDNN study. Each weight was randomly initialised to a value in [0, 1 / n], where n is the number of connected input or output nodes. All input, hidden and output nodes used the sigmoid activation function. The output category was transformed into a length 3 vector, where each element corresponded to a category, and its value was the binary value of 'is the original output equal to this category.'

The number of epochs and hidden nodes were originally varied, however after several observations it was determined that the epoch number and nodes didn't affect the result, excluding extremely low cases which reduced performance. In data collection, 6 nodes and 70*L epochs were used, where L is the length of the training dataset, at a learning rate of 0.05. The model was also tested by using different learning rates for the first and second connection layers, which yielded no improvement in results.

Since the network needed to be operable both ways, the input data was bounded to the range (0,1) by applying the sigmoid function to it. 2 datasets were then imported. One was the main web snippet dataset, and the other was the public Iris classification dataset [3]. The Iris dataset was used as a control to compare the MLP and BDNN on

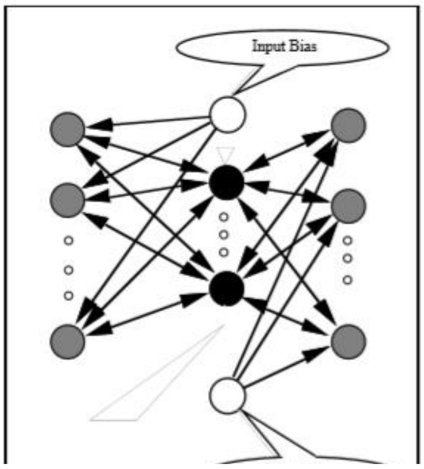

Figure 2

easier data, as well as to provide a safety catch in case a mistake was made during model implementation. It was also a convenient choice as it had the same number of inputs and outputs as the snippet dataset, which means it could be tested without making changes to the code.

During training of the BDNN model, the entire training set was iterated over sequentially in the standard, forward direction. The backpropagation algorithm was used to update the weights after each iteration, using a loss function equal to 2*n times the mean squared loss, where n is the number of nodes in the final layer. In the next cycle, the training set was iterated over again by feeding the output class vector to the output nodes and passing it through the BDNN in reverse direction. It then used the same loss function and weights as the forward connection and updated the weights from its perspective through gradient descent. This was repeated until 70 cycles had passed. Interestingly, the dual weight updates were able to converge after enough cycles.

Topology of the single layer BDNN, sourced from Reference [1]

Note that due to the direction of propagation, the input biases were not used when training in the forward direction, and the output biases were not used during training in the backward direction.

The MLP model did not need to be implemented separately, and instead was simulated by using the BDNN model without any reverse passes. As in the SDE model, the training data was ensured to have even quantities of all categories to prevent bias to the most frequent category.

4 Results and discussion

The following are confusion matrices taken from basic testing of the SDE model.

	Iris Dataset	Snippet Dataset
Untrained	[[0, 0,13] [0, 0,11] [0, 1, 5]]	[[1,19,0] [1,18,1] [1,19,0]]
Trained	[[13,0,0] [0,11,0] [0,0,6]]	[[10,3,7] [8,6,6] [7,9,4]]

Table 1

	Iris Dataset	Snippet Dataset
Untrained	[[0,13,0] [0,11,0] [0,6,0]]	[[3,12,5] [6,10,4] [5,5,10]]
Trained	[[13,0,0] [0,10,1] [0,1,5]]	[[5,8,7] [5,8,7] [2,11,7]]

Table 2

The following are confusion matrices from decaying the intensity of mutations as the generation increases

	Iris Dataset	Snippet Dataset
Untrained	[[0,0,13] [1,0,10] [0,0,6]]	[[16,0,4] [14,0,6] [14,0,6]]
Trained	[[13,0,0] [0,6,5] [0,0,6]]	[[5,7,8] [9,6,5] [9,6,5]]

Table 3

In all cases, the model performed well on the control data, and basic testing achieved accuracies of 100% and 93%. Decaying the mutation intensity over time worsened the performance on the control dataset to 80% accuracy. In all cases, the model failed to learn on the snippet dataset which shows it is not effective in modelling data as noisy as the snippet length dataset, and achieved accuracies of 33%, 33% and 26%. However, the high performance on the iris dataset demonstrates the SDE to be a useful model for less noisy data.

Training the SDE model was slow, which was a factor that restricted the model from producing enough tests to make accurate statistical inferences about its performance. The cubic scaling of time complexity with respect to population was also a potentially restrictive factor, since it made using a very small population necessary to train the model in reasonable time.

The MLP and BDNN models were trained and tested 100 times on the control and snippet datasets, and the performance averaged. Untrained models were also included for comparison.

	Accuracy
Iris Dataset, MLP Untrained	0.319
Iris Dataset, MLP Trained	0.724
Snippet Dataset, MLP Untrained	0.338
Snippet Dataset, MLP Trained	0.359
Iris Dataset, BDNN Untrained	0.338
Iris Dataset, BDNN Trained	0.746
Snippet Dataset, BDNN Untrained	0.341
Snippet Dataset, BDNN Trained	0.353

Table 4

In all untrained instances, the accuracy was approximately 33%, which is expected for an untrained decision between 3 categories. The regular MLP achieved an accuracy or 72% for the Iris data, and the BDNN achieved 74% for the same dataset. This matches well with the original BDNN paper, which achieved 74% accuracy on their student marks dataset.

Both the regular MLP and BDNN were not successful in classifying the snippet dataset.

5 Conclusion and future work

The problem of classifying data with as much noise as the snippet length dataset is difficult, and 3 models have been demonstrated to be unsuccessful in solving the problem. As such, the problem of creating a useful classification model for such data remains open, and more models could be tested on this data.

The problem may also be impossible; although we can be confident from the results of the original experiment that a relation exists there may be no model that can remove the noise to an extent where it becomes useful. For this case, it may be useful to identify reasonable assumptions to make on the dataset that allow us to mathematically or logically prove it is unsolvable, as has been done for other problems[6].

The training of the SDE model was also very limited due to time and computational power restrictions. With access to a more powerful computer, the experiment could be repeated with a larger population and number of generations. The number of training samples included in each generation was also not experimented with, and it is possible that a different number is more optimal.

The model also uses a lot of space and could be optimised by storing encodings of functions instead of storing the set of functions for every individual. The current version of the model is not exhaustive of the 4-dimensional function space. Although it theoretically can reproduce any function of i (recall $i = v_0 \vartriangle v_1 \vartriangle v_2 \vartriangle v_3$), the conversion between the 4 dimensional input variables and one dimensional intermediate i value is not exhaustive and limited to a basic set of functions. As such, this version of the SDE model is not truly able to find any function from the input to output. In future, it could be updated to allow the conversion from the inputs to the intermediate value to encompass all functions as well.

However, the results of the SDE model on the control dataset are promising. Using the fixed intensity approach, it was shown to have very good accuracy on the iris classification problem and performed better than the multi-layer perceptron network which is already commonly used in classification problems, even with a population of only 20. It also has an underlying benefit of being resistant to bad assumptions on what underlying functions are appropriate for the problem, since the set of underlying functions are elements of a basis of all functions. This makes it a potentially great model for data with a low to moderate amount of noise where the testers have low understanding or intuition of the data and could be explored further to determine how useful it is on more datasets.

The BDNN model also showed reasonable classification of the control dataset and showed equivalent performance to the multilayer perceptron model. However, the bidirectional property makes it desirable in cases where you want to reverse a mapping, but also value space and wish not to store two separate models. More testing could be done on this model as well to determine its limitations and where it is practical to use this to conserve space.

6 References

[1] BiDirectional Neural Networks and Class Prototypes, A.F. Nejad and T.D. Gedeon

[2] Kim, J., Thomas, P., Sankaranarayana, R., Gedeon, T., & Yoon, H. J. (2017, March). What snippet size is needed in mobile web search?

[3] https://archive.ics.uci.edu/ml/datasets/Iris

[4] An Adaptive Noise-Filtering Algorithm for AVIRIS Data With Implications for Classification Accuracy
Rhonda D. Phillips, Student Member, IEEE, Christine E. Blinn, Member, IEEE,
Layne T. Watson, Fellow, IEEE, and Randolph H. Wynne, Member, IEEE

[5] Comparison of evolutionary algorithms in gene regulatory network model inference
Alina Sîrbu, Heather J Ruskin, Martin Crane
[6] Incomputability C. A. R. HOARE AND D. C. S. ALUSON