Journal of Computer Science 2024
Catalysing Research Institute https://www.catalysingresearch.org/

Investigation Of Dataset Optimization Techniques for Neural Network Classification Algorithms

Hadi Farah

E-mail: hadifarah2008@gmail.com

Abstract

Computer science encompasses a wide array of functions. A more dataset-focused subfield of computer science, Artificial Intelligence (AI), uses algorithms that require training datasets to improve their accuracy without needing human intervention. These algorithms are a part of an idea known as Machine Learning (ML). Neural networks are one such machine learning technique for predicting and modeling dataset behavior. If the datasets are large or complex relationships exist between dataset variables, such techniques can lead to unacceptably long training times. One approach to dealing with this issue is to apply the strategic removal and optimization of certain dataset parts. There are already attempts to approach this problem, such as principal component analysis (PCA) and linear discriminant analysis (LDA). This work, however, will propose an alternate approach to data optimization: removing unneeded points based on their z-scores. Using this technique, we demonstrated a ninefold increase in execution speed.

Keywords: Robotics and Intelligent Machines; Machine Learning; Artificial Neural Networks; Z-scores; Principal Component Analysis

1. Introduction

Artificial neural networks (ANNs) have become widely used in many sectors and companies. Companies like Metamind and Clarifai use AI for purposes ranging from text interpretation to image recognition [1]. Most applications use feed-forward ANNs with backpropagation (BP) to help the algorithm learn more efficiently and improve its performance. Neural networks have multiple versions. Rigid ANN architectures are the foundation for these methods, consisting of node and property transfer functions that only train weights. Although ANNs are useful, creating an ideal design for their application is unlikely. This is a serious issue, as there is substantial evidence suggesting that an ANN's knowledge-processing capacities are determined by how it's designed [2].

We set out to create a new dataset reduction method that works by only considering data points with the most extreme z-scores aka. the farthest points from the mean. In concept, we theorized that data points closer to the mean have less overall effect on an algorithm's results. Thus, if we can preserve the dataset's attributes while removing the bulk of the data points, the algorithm speeds up with little to no compromise on accuracy. However, most data reduction methods try removing outliers from data sets rather than including them. Take Aman Preet Gulati's 2022 paper on the subject. In the paper, the researchers attempted to create an algorithm that effectively removed outliers from a dataset, as they theorized that outliers were more of a nuisance than an asset. They acknowledged that their method of outlier omission had limitations as it couldn't perform very well on datasets that weren't normally distributed [3]. Using our approach, on the other hand, could eliminate this problem since we are removing points based on their z-scores, which are not dependent on data distribution.

2. Background

2.1 Long Machine Learning Training Times

An example of this can be seen once ANNs start processing larger and larger datasets. It becomes evident that these networks take a long time to train. Thus, this paper will elaborate on some ways to speed up training times and explore a new technique of data optimization: compiling selective traits into a smaller, more compact dataset. In theory, this greatly speeds up training times at the expense of a small delay in data preprocessing. In parallel, numerous recent studies on different optimization methods can improve training times, but there is still room for improvement.

2.2 Principal Component Analysis & Linear Discriminant Analysis

There are many ways to reduce dataset size. One way to do that while minimizing information loss is principal component analysis (PCA). PCA works by gradually maximizing variance by creating new, uncorrelated variables. It simplifies the data into smaller datasets with the same general patterns, allowing it to maintain accuracy while speeding up the training process. Also, PCA does not require any assumptions on the distribution of data, making it versatile for use in many datasets [4]. On the other hand, it is relatively bad at yielding results when used on datasets with a large number of outliers, since outliers increase variance and interfere with the program's ability to identify trends, decreasing its accuracy [5].

Another common method is LDA (aka. Linear Discriminant Analysis). LDA was first developed as a method for figuring out which combinations of variables could best divide datasets. These linear combinations allowed researchers to identify variables that would most influence group classification. LDA also offers a data point's most likely classification when its group is unclear. LDA operates under the assumption that the provided data is normally distributed. Due to this, LDA becomes unreliable when faced with datasets possessing many features. In addition, it assumes that data is linearly separable (classes of different points can be separated by a straight line), which is a disadvantage when working with datasets having scattered data [6].

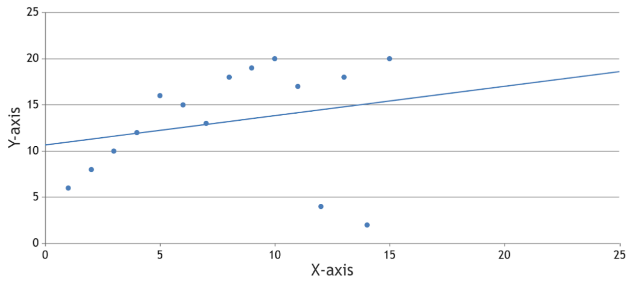
This method solves both problems mentioned above: PCA's issue of being unable to deal with outliers effectively (since this model utilizes rather than antagonizes outliers) and LDA's relative weakness when dealing with datasets using a large number of features (due to the algorithm being scalable across many features). We hypothesize that using optimization techniques may speed up training times by up to 60% at the expense of a small increase in preprocessing times. The reason for this, which will be clarified shortly, is that our method reduced dataset size by more than half.

2.4 Z-score Computation

A z-score, expressed in standard deviation units, indicates where a raw score falls relative to the mean. If the value is above the mean, the z-score is positive; if it is below the mean, it is negative [7]. The equation to calculate a z-score is as follows:

$$Z = \frac{x - \mu}{\sigma}$$

In this equation, x is the data point in question, μ is the mean of the data, and σ is the standard deviation. As mentioned above, the z-score measures how many standard deviations away a point is from the mean. This will be used to identify outliers. The diagram below shows an example that will demonstrate how z-scores will help us remove unused data.



The mean y value in Figure 1 can be calculated as 198/15, or 13.2. When applying the equation above, n will be found as 6. Thus, the six highest and six lowest z-scores can be taken, with the mean being relatively the same, in this case, 13.5. This may also apply to features on a dataset, with even more accuracy due to the larger number of data points in most training sets. They could be reduced this way without changing the overall mean of the features. This means that similar results can be achieved with as little as ten percent of the original dataset, provided that the dispersion of the data and mean are relatively unchanged. It is also important to note that the function will become more accurate the more data points there are, since small datasets such as the above may have radial distributions. As the scale of the data increases, the percentage of points needed for an accurate prediction decreases, which is where this method starts to save time, performing relatively better on larger datasets.

2.3 Optimization Functions

Moving onto optimization functions, there are many that could be used for purposes like this. The first is RMSProp. RMSProp is an optimization function invented (unofficially) in 2012 by Geoffrey Hinton. RMSProp determines an adaptive learning rate based on the gradients of each parameter. RMSProp converges more quickly than conventional gradient descent techniques and can accommodate a variety of gradient scales by varying the learning rate for each parameter. This optimization function was the predecessor to Adam [8].

Adam is a gradient-based stochastic function optimization function. Thanks to its ability to handle large datasets and parameter sets, it's simple to use for any model. It requires fewer memory resources in terms of hardware and is very computationally efficient. Additionally, it is effective at addressing non-stationary targets, spare slopes, and noise. The Adam optimization method often involves minor adjustments; however, any effective model must be adjusted. The Adaptive Moment Estimation (Adam) technique maintains adaptive learning rates regardless of the parameters, and a single learning rate is maintained for all weight updates that stay constant during the training phase. Adam was created as a sort of improvement upon RMSProp [9].

3. Methods

3.1 The Dataset

Our research mainly focused on a medium-sized gender classification dataset with 5001 facial data entries and 8 features. The dataset was found on Kaggle [10]. The study's main focus was on the forehead width and height, which were the features chosen to test the algorithm since they were the only features that were continuous data. Before testing, the gender column of the dataset was converted from string values (Male and Female) to numerical values (0 and 1) to allow the NN to process it, and the dataset was split into a training and testing set at a ratio of 8:2.

3.2 Data preprocessing

3.2.1 PCA

Upon testing the optimal efficiency of the PCA algorithm, we settled on using two principal components for data reduction (n_components = 2). To reach this conclusion, we varied n_components from 1-10, and two components were found to be the most accurate for its time. The PCA module was imported from the sklearn.decomposition library in Python. All data reduction methods were carried out with optimal parameters. PCA, for instance, used three layers to get its best result: an input layer of 128 neurons, another layer with 64 neurons, and a final layer with 3 neurons. All layers used sigmoid activation with Adam as the optimization function. In the end, the algorithm achieved a 96.6% accuracy in approximately 5 seconds, after 10 epochs.

3.2.2 LDA

In addition to PCA, we also compared our results with LDA. When testing the optimal LDA setup, we found that it yielded the best results with 1 component (n_components = 1). Just like PCA, we experimented with various values of n_components, and 1 was found to be the most efficient. The modules for this test were found in the Python library sklearn.discriminant_analysis. The algorithm used three layers; the first had 64 neurons, the second had 12, and the last was the output layer of 1 neuron. In addition, sigmoid was also found to be the best activation function in this case, along with adam as the optimization. In terms of loss minimization, all three algorithms worked best with binary cross-entropy. In the end, the final model hit a peak accuracy of 96.2% in 5.3 seconds.

3.2.3 Z-scores

Now comes the proposed method. To preprocess the data, we first calculated the z-scores of data points. The next step was to keep n values from both extremities of z-scores (omitting and discarding values that are closer to the mean), with n being the result of the equation:

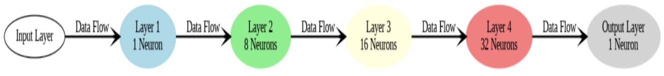
$$n = \frac{\#\ of\ datapoints^{0.92}}{2}$$

$$y = \frac{Ceil(x^{0.92})}{2}$$

The value of the exponent was found through trial and error, and 0.92 was found to yield reliable accurate results. We had planned on the exponent of the function being less than one, to increase the percentage of data points taken the larger the dataset was. We set a benchmark where datasets with 100,000 entries were to be shrunk to around a fifth of their size, while datasets with one million points were reduced to almost 15%. The result of this calculation would be divided by two to take a total amount of n/2 data points, assuming a normal distribution.

3.3 Structure of the NN model

As for the structure of our model, we used a Tensorflow neural network with 5 layers. The output, first, second, third, and fourth layers had 1, 8, 16, and 32 neurons respectively. They were used alongside simple hyperparameter optimization to test for optimal parameters, but this was removed in the final algorithm. In doing so, we found the initial training times and preprocessing times, in addition to receiving and improving upon prediction accuracy (the last layer was varied, and its results depended on the function).



3.4 Training of the model

The training process was carried out using hyperparameter optimization. This function went over several different combinations of neurons, activation functions, and optimizers. Let's cover everything, beginning with the activation functions. Activation functions allow neural networks to convert complex, non-linearly separable input data into potentially linearly separable representations [11]. The first function tested was the sigmoid function. It's been one of the most commonly used activation functions in the past, along with tanh, or hyperbolic tangent. However, sigmoid is relatively unreliable in some situations as it saturates gradients, making it less reliable in certain situations [12].

ReLU (rectified linear unit) was created to solve this problem. ReLU's biggest advantage was that it solved the sigmoid function's vanishing gradient issue [13]. This issue occurs because of the way sigmoid functions work; which is squeezing inputs from a full range of numbers to a value between zero and one. As the function is centered around deriving inputs to return the gradient, these derivatives can get quite small, meaning the gradient would slowly diminish. Despite these problems, the sigmoid function was experimentally proved to be the best fit for our dataset. Keep in mind that dataset specifications could make the vanishing gradient problem impertinent.

3.5 Parameter Comparisons

After repeated testing of all the algorithms that were to be used, utilizing the hyperparameter optimization function spoken about earlier, the parameters that yielded the best results were the following:

No data reduction: sigmoid activation, 256 neurons, adam optimization

Z-score method: sigmoid activation, 256 neurons, adam optimization

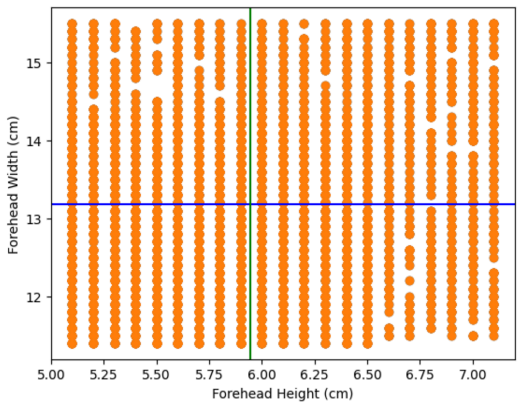
PCA: sigmoid activation, 128 neurons, adam optimization

LDA: sigmoid activation, 64 neurons, adam optimization

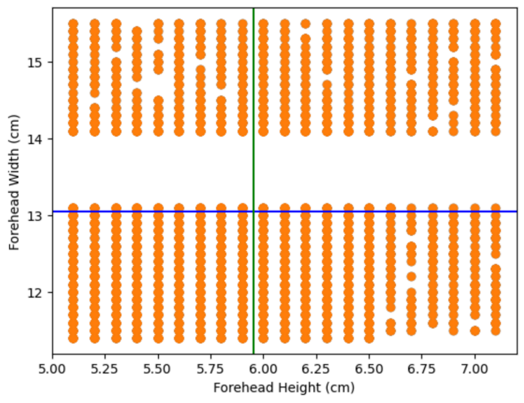
Binary cross-entropy was used as the loss function for all cases, as it was found to perform better than MSE (Mean Squared Error) and MAE (Mean Absolute Error)

3.5 Post-result Findings

To prove the algorithm's validity, we confirmed that the data's mean and variance remained unchanged upon reduction, to ensure that the results were due to the algorithm's competency and not lucky classification of data. Here is a comparison of the data before and after reduction:



Before reduction



After reduction

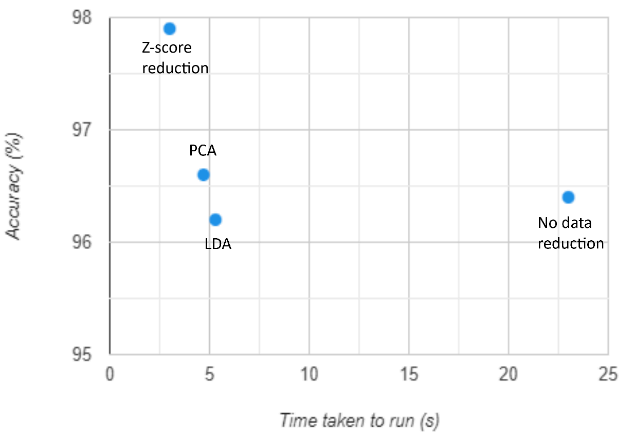
Both figures come from a scatterplot of the data made using the Python matplotlib library.

As demonstrated by the figures, the more central data points were successfully removed, with a marginal change in forehead width mean. Upon further research, however, we noticed that the initial method of selecting an equal number of extremities was unfinished since it relied on a normal data distribution, just like LDA. Thus, we tweaked the code to identify the data distribution and take several points proportional to the original.

In essence, the algorithm accommodated skewed distributions by varying the number of outliers taken from each side of z-scores. For example, if data points were positively skewed, and three-quarters were less than the mean, 75% of n points were taken from the lower extreme z-scores and 25% from the upper extremities.

After running tests with the improved algorithm and rerunning the tests, the mean forehead width had been adjusted slightly and was now more accurate than before (from 13.08 to 13.14). After reduction, the new training dataset was fed into the NN, with even more positive results; the revised algorithm acquired a 97.9% accuracy in 3 seconds (using the same parameters as the previous algorithms). In this way, the algorithm could now process negatively and positively skewed datasets.

4. Results and Discussion



The above figure displays a measure of the accuracy vs. the time taken to run the algorithm.

In the end, all results were measured after the appropriate hyperparameter optimization. In terms of hardware specifications, all algorithms were run on Google Colab with 12.7 GB of RAM, 107.7 available GB of disk space, and an Intel(R) Xeon(R) CPU @ 2.20GHz processor. Before any reduction, the model reached a peak accuracy of 96.4% in 23 seconds. The results were pleasantly surprising after the z-score filtering. In only 3 seconds of training, the model achieved a peak accuracy of 97.9%, exceeding initial expectations. Rather than the expected timesaving of around 60%, there was almost an 87% reduction in training time. Comparing the Z-score method to PCA and LDA, PCA yielded a 96.6% accuracy in 4.7 seconds, while LDA ended with 96.2% accuracy in 5.3 seconds.

During testing, we faced many hurdles in preventing overfitting and underfitting. Eventually, we found a balance by using a simple, 3-hidden layer ANN with sigmoid activation for the z-score method. Tests were repeated using larger datasets with similar structures and results could be seen in both cases. After testing, we noticed a potential downfall of our function, since it was failing to classify datasets with skewed distributions. To fix that problem, we set out (in the Post-results Findings section) to resolve this issue and ultimately came up with a function to calculate the number of points above and below the mean. With the new changes, the algorithm now took a specific percentage of outliers from each extremity in proportion to the function's result. For example, if the function returned 45% of points above the mean, the algorithm would take 45% of its outliers from the upper extremities and 55% from the lower extremities.

The most promising aspect of the algorithm is that it proved capable of being both faster and more accurate than current solutions in the field, rather than the tradeoff between speed and accuracy that is expected by similar algorithms. With more extensive research, the method proposed in this paper could potentially become even faster and more accurate and could be changed to accommodate diverse types and distributions of datasets, or expanded to handle more than two features at a time.

5. Conclusion

To conclude, we set out to find whether or not we could develop a dataset reduction that could rival conventional ones by using z scores. We were ultimately successful in producing a simple model that could accurately and quickly filter datasets, accurately filtering our dataset(s) in less than 5 seconds. After comparing our model with the LDA and PCA models, we can conclude that our data reduction method is more accurate and slightly faster than the two above, with our algorithm being 1.7 seconds faster than PCA, the fastest method, and 1.3% more accurate than PCA, the most accurate method. We experimented with hyperparameter optimization, and the results remained consistent.

This algorithm holds promise as it (so far) proves to be a faster, more accurate way to process data. With all said and done, this means that companies and firms that require large amounts of data processed quickly (Amazon's product filters, for example) can use this method to process datasets where other algorithms fall short. All in all, future research can still be done, such as testing and improving the function's ability to accurately classify data from smaller datasets, and comparing it to even more data reduction algorithms commonly used to see where it stands.

Acknowledgements

Hadi Farah is a 16-year-old Lebanese-American student studying at the International School of Choueifat, Doha, aspiring to become a Computer Science Engineering Student in the US. He is especially passionate about AI and Robotics and aspires to improve the consumer technology field through his work.

Special thanks to Dr. Eric Sakk (Morgan State University) and Timothy Adamson (Yale University) for their support of this paper.

References

[1] Çoban, Enis Berk. “Neural Networks and Their Applications.” ResearchGate, Feb. 2016, https://doi.org/10.13140/RG.2.1.5176.0404.

[2] Qamar, Roheen, and Baqar Ali Zardari. “Artificial Neural Networks: An Overview.” ResearchGate, Aug. 2023, pp. 130–39. https://doi.org/10.58496/mjcsc/2023/015.

[3] Gulati, Aman Preet. “Dealing With Outliers Using the Z-Score Method.” Analytics Vidhya, 2 Sept. 2022, www.analyticsvidhya.com/blog/2022/08/dealing-with-outliers-using-the-z-score-method.

[4] Jolliffe, Ian T., and Jorge Cadima. “Principal Component Analysis: A Review and Recent Developments.” Philosophical Transactions - Royal Society. Mathematical, Physical and Engineering Sciences/Philosophical Transactions - Royal Society. Mathematical, Physical and Engineering Sciences, vol. 374, no. 2065, Apr. 2016, p. 20150202. https://doi.org/10.1098/rsta.2015.0202.

[5] Chen, Xiaoying, et al. “Robust Principal Component Analysis for Accurate Outlier Sample Detection in RNA-Seq Data.” BMC Bioinformatics, vol. 21, no. 1, June 2020, https://doi.org/10.1186/s12859-020-03608-0.

[6] Desai, Chitra G. “Understanding Linear Discriminant Analysis for Classification and Dimensionality Reduction.” ResearchGate, Oct. 2022, www.researchgate.net/publication/364322214_Understanding_Linear_Discriminant_Analysis_for_Classification_and_Dimensionality_Reduction.

[7] Mcleod, Saul, PhD. “Z-Score: Definition, Formula, Calculation &Amp; Interpretation.” Simply Psychology, Oct. 2023, www.simplypsychology.org/z-score.html.

[8] Elshamy, Reham, et al. “Improving the Efficiency of RMSProp Optimizer by Utilizing Nestrove in Deep Learning.” Scientific Reports, vol. 13, no. 1, May 2023, https://doi.org/10.1038/s41598-023-35663-x.

[9] Alom, Mohammed. “Adam Optimization Algorithm.” ResearchGate, June 2021, www.researchgate.net/publication/352497171_Adam_Optimization_Algorithm.

[10] Jifry Issadeen https://www.kaggle.com/datasets/elakiricoder/gender-classification-dataset

[11] Dubey, Shiv Ram, et al. “Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark.” arXiv.org, 29 Sept. 2021, arxiv.org/abs/2109.14545.

[12] Boedeker, Peter, and Nathan T. Kearns. “Linear Discriminant Analysis for Prediction of Group Membership: A User-Friendly Primer.” Advances in Methods and Practices in Psychological Science, vol. 2, no. 3, July 2019, pp. 250–63. https://doi.org/10.1177/2515245919849378.

[13] “Z. Hu, J. Zhang, and Y. Ge, "Handling Vanishing Gradient Problem Using Artificial Derivative," in IEEE Access, vol. 9, pp. 22371-22377, 2021, doi: 10.1109/ACCESS.2021.3054915.