[redacted]

[redacted]

Mr. Ruppel

Internship

15 May 2018

Graphical Representations for Neural Network Training Analysis

Introduction:

For the unacquainted, the world of artificial intelligence may seem mysterious and vague. However, the processes behind decision making in a computational setting are well backed by years of mathematical proof. Many of the leading ideas in the field were developed decades before computers were able to use them effectively. Despite this, it was proven that these methods would be effective using a statistical basis.

One of the main goals was to create general purpose AI, capable of adapting to a wide variety of tasks. The current methodologies were not able to do this, as they were based on specific rules and heuristics. In order to move in the direction of generalization, the entire basis of algorithm construction had to be redone. Researchers noticed that small biological organisms with limited brain size were still able to carry out a wide variety of complex tasks. An example of this is certain worms and ants with neuron counts in the thousands instead of billions (Wehner, 2003).

On an entirely different plane of research, statistics had the tools for generalized prediction. A popular one that many will be familiar with is linear regression. Linear regression is widely used for drawing correlations between sets of data. If we consider one set to be an input, and the other an output, we have created a very simple predictor for linear data. An

addition to this is logistic regression which brings with it the ability to correlate in a non-linear fashion (Shalizi, 2012). This small change, while mostly ignored in the field of statistics, became instrumental in AI design.

These two disparate fields came together to finally solve this dilemma. A series of linear regressions followed by transfer functions is a reflection of its biological counterpart. As with brains, connections can be strengthened between neurons to produce different functionality. Each part of the brain is near identical on the micro scale, but the strength of connections adjusts the way in which it works. This is paralleled by the adjustment of weights in each individual transformation matrix. This potent combination of statistical methodology with computational power for tuning leads to a versatile artificial intelligence system.

Even though neural networks have been around for several decades, they are still used with suspicion. Use cases for which they are well suited, like weather forecasting, still heavily rely on traditional methods. This is testament to the general distrust of biologically inspired methods in practical AI usage. While there is mathematical proof that neural networks do work, they are also unpredictable and occasionally cannot be trained for tasks effectively.

In order to improve this perception, the inner workings of neural networks must be better understood. It is difficult for humans to look at a list of numbers and gain meaning. However, by visualizing this data, it could be easier to use nets effectively. The question we work to solve is, what insights into the operation of neural networks can be gained by visualizing the data produced?

Background:

Neural networks are a recent addition to the world of artificial intelligence, but are based on a long history of statistical methodology and biological sciences. The original inspiration for neural networks was modeling of organic nervous tissue structure, and its incredible ability to recognize patterns and adapt to a wide range of scenarios. The field of AI changed from a focus on long lists of rules and heuristics to make decisions, to a training based model. These were more unpredictable, but significantly more accurate and useful than their non-neuron based counterparts (Anderson, 1995).

Neural networks, as the name would imply, are composed of individual computational units referred to as “neurons”. These units sum all the inputs based on their respective weights, run the results through a transfer function, and then produce an output. These neurons are organized in layers, with each neuron in the previous layer connected to all the neurons in the next layer. Adjusting the weights of these neurons can modify the behavior of the network to produce nearly any computation. The mathematical description of this phenomenon is that each layer represents a transformation of the input vector by the weight matrix followed by a component-wise transfer function. This series of pseudo-linear transformations can relocate all input points into a linearly separable space for classification by a hyperplane. Even problems that are not separable in their input dimension can be processed in the higher dimensions of other layers. A classic example of this is the XOR problem, which is unsolvable by linear methods in two dimensions (Cancelliere). However, a neural network with three neurons in the second layer can solve this. Intuitively, this makes sense because we can define a linear transformation that moves [1, 0] and [0, 1] higher in the third dimension, and then define a plane than separates it from the other points.

For problems such as XOR, the weights can be computed by hand and used. However, this is not feasible for more complex computations. Therefore, the goal of research in the past couple decades has been to find ways of "training" these networks to accomplish a specific task. Many different methods have come forth, but they are all based on the idea of gradient descent (Pendharkar, 2007). The gradient of a function is similar to the derivative, but with more dimensions. A neural network can be represented by a function where each weight is an input, and the prediction error on the test set is the output. By finding the gradient of this function, we can "descend" the slope to areas of lower error. Each researcher has their own perspective on the correct stopping criterion, learning rate, and error functions, but the fundamental method remains the same. When used for neural networks with hidden layers, this is known as backpropagation (Wehner, 1995).

A consideration taken by all data analysts using biologically inspired prediction methods is the structure of their net. At first, neural networks were simply logistic regression machines. This meant they had only a single layer followed by a hyperplane classification. However, it was soon realized that many problems required more than one transformation to separate adequately. The idea of deep neural networks, with a series of hidden layers, became popular. This was very well suited to complex classification problems with many inputs. Later, it was realized that circular neural nets, with outputs leading to previous layers, also had a use. This sort of construction is known as a recursive neural network, and simulates a primitive type of short-term memory. This became a very popular method in sentence comprehension algorithms (Lecun, Yann, et al. 2015). Lastly, the size of a network must be optimal. If a network is too small, it cannot be trained to learn a task, since the dimensions required to process the data are not present

in the structure. If it is too large, the network can learn an overly complicated decision process, and be unable to generalize to new data. These are all parameters that must be decided in the creation of a neural network.

One of the main issues with neural network based prediction algorithms is that the inner workings are not always clear. We can see that after a training period certain neurons do in fact represent specific features. Carnegie Mellon researchers state, “By limiting the size of the hidden layers, the network is forced to develop appropriate feature detectors to efficiently classify large sets of input parameters (Touretzky, 1989).” It is a common consensus that hidden layers train to become feature detectors, and the output layer weights those features and their importance. However, we still do not know which neurons are detecting which features, or what features are being found. In this way, neural networks are somewhat of a “black box” in the field of AI. We know that they work, we can prove why they work, but we cannot look at a specific network and understand how it reaches a conclusion. This has been a goal of NN research for many years.

Methods:

In order to solve this quandary, several steps were taken. The first of these steps was to determine a suitable problem for plotting an error function. The problem was required to be simple enough to represent graphically, but complex enough to show the practical use of neural networks. Additionally, this problem could only have 2 or 3 dimensions. Any more, and it becomes impractical to visualize. The chosen problem was prediction of gas prices across the United States based on coordinates. Training data was collected and compiled for this purpose into a format usable by software. This included the mean prices for each state, along with the mean coordinates of the area of that state.

It was decided that Theano would be used to create this neural network program, due to the flexibility of operation. This flexibility allowed for specific methods to export gradient data. Each value in the starting net was assigned to a random value in [-1, 1]. For gradient descent, the mean squared error function was used. There were some issues with the large values in the input paired with a sigmoid transfer function. To remedy this, all values (price, north coordinate, and south coordinate) were normalized before use in the neural network. They were rescaled to their true value afterwards.

The final program to estimate this simulated a neural network with two inputs, two hidden neurons, and a single result neuron. There was a training time of 600,000 cycles, and a descent rate of 0.1. At the conclusion of the training period, the current estimates were exported and the weights for each neuron were recorded. These estimates and the percentage error from the true value is listed in [Appendix-1]. The program was then modified to remove the training period, and have the starting values assigned to the trained version instead of randomly. The error calculation was run over a set of viable weights for the last layer, and the result exported. After some empirical examination, a range of [-20, 20] was found to be optimal for both dimensions. The format for data was a simple text file with values laid out in a grid.

In order to display the data collected as such, a graphing program known as *Gnuplot* was used. The surface mesh found from this rendering can be found in [Appendix-2].

Analysis:

Graphing the error plot reveals interesting properties about the operation of neural networks. The first feature to note is that there are many local minima that are not the absolute minimum of the error function. This matches expectations of the operation of neural network

training. Often, the network will find an approximation based on the wrong features that is "good enough", and any incremental change will only reduce accuracy. In order to reach the correct weight matrix, a complete modification of the parameters is necessary. On the graph, this would be equivalent to making a large jump in order to escape the local minimum. There have been several methods proposed in AI literature about how to avoid this problem with gradient descent (Guely, F., and P. Siarry, 1993).

Another feature of this graph is that the fluctuations become flatter and less noticeable as the distance from the origin increases. At first glance, this may seem strange, given that very incorrect weights should lead to very high spikes in the error plot. However, this result makes perfect sense when looking at the sigmoid transfer function present after each layer. This function maps all real values to a range of [-1, 1], and therefore lessens all extreme results to a reasonable range (Yonaba, 2010). Yonaba discusses the importance of transfer functions for prediction purposes. This is why the graph contains a flat look near the edges, since even large changes in the input lead to small changes near the output when far from the origin.

One last note on the error plot is that only a two dimensional cross section of a six dimensional function is shown. We cannot gain any meaning from higher dimensional graphing, so taking cross sections can be an incredibly useful tool in analyzing neural network operation. This is very similar to viewing a variety of correlation graphs from a population in statistics. There are many dimensions which we care to analyze, but we can view them in a variety of cross sections to gain meaning from the data.

In [Appendix-1] the results of the network prediction can be seen. The first column is the state name, followed by the predicted gasoline price and the percentage error from the real price

in that area. There are some interesting intuitions that can be drawn from this data. For example, we see that the highest error is for California, at 18 percent. This makes sense for two reasons. Firstly, the neural network is a continuous transformation of points from the two dimensional input (north and south coordinates) to the one dimensional output (price). This nature means that abrupt changes, such as those between Nevada and California, are difficult to represent. Secondly, the reasons behind higher gasoline prices in California are not strictly based on geographical location. For this reason, the neural network has difficulty developing transformations that can describe it based only on coordinates.

Neural networks can be extremely difficult to study and understand, but using visual aids to peer into the inner workings is a great tool in any researcher's toolbox. It can be used to find optimal transfer functions, learning rates, and possibly even suitable network structures. Additionally, by reviewing the data closely, the underlying pitfalls and structure of the network can be examined.

Conclusion:

As the field of computer science grows rapidly, so too does artificial intelligence. We are finding new uses and applications of AI in all sectors of life, business, and science. Likewise, our methods and algorithms are gaining in complexity and breadth. In order to meet this demand, many students must be educated on the basics of neural networks each year. It can be exceedingly difficult for these students to develop an intuition for the function of neural nets. Most courses start with high-level mathematics to justify the validity of the proofs, without first providing a very simple overview of the transformations and the process of gradient descent.

One of the best ways to explain mathematical concepts is through geometric allegories. A prime example of this is the pythagorean theorem, which can seem obtuse until the area-based proof is understood. This carries to neural networks as well. A series of summations and exponential functions seems foreign and unusual, but moving around points within a space seem natural and intuitive. We know that each layer can be represented by a matrix and a transfer function, but we often forget that the whole net can be seen as a collection of transformations of the input vector through spaces. This explanation is far more friendly to fledgling students and experienced researchers alike.

What this means for the research taking place here is that it is a small step in the larger goal of understanding the function of individual networks. There is plenty more that could have been investigated, but was outside the scope of this timeframe. For example, the only type of network investigated was a multilayer perceptron, arguably the simplest form. A look at other types, such as recurrent neural networks, could have proved interesting. Visualizations of the state memory (a primitive form of short term memory) plotted across time would be an excellent extension.

Another extension that was not investigated was the best way to determine what data to look at. For the gas prediction program, the net was small, simple, and uncomplicated. The error plot cross section was an obvious choice, containing the last layer in its entirety. However, larger networks have hundred of options for the same type of graph. How can we determine which correlations are important to function, and which are not? There is likely a procedural way to determine this, but such a method was not found in the course of the research.

Despite all the shortcomings of the limited research that took place, there is still valuable progress. This demonstrates that even a small graph can reveal a multitude of poignant details about the function and flaws of multilayer perceptrons. Careful analysis brings to light the effect of the sigmoid transfer function and limits of simple gradient descent. Looking even further at the predictions made, California highlights the inability of neural networks to overcome specific estimation errors, even after 600,000 training epochs. All of this can be derived from the data shown in the appendix. Therefore, it is reasonable to assume that further probing into more complicated neural network structures, such as CNNs and RNNs would yield even more useful insights.

Appendix:

Hawaii [3.05132626] [14.4328025]%

California [2.91022093] [18.09116436]%

Florida [2.52253501] [5.84042524]%

Alaska [3.11348927] [3.0367713]%

Alabama [2.61963157] [-5.03735238]%

Texas [2.73191533] [-9.4517359]%

Massachusetts [2.5590423] [3.90378146]%

Pennsylvania [2.60277749] [11.01615421]%

Minnesota [2.77930587] [-10.42136943]%

New_Jersey [2.56860591] [5.66999946]%

North_Carolina [2.57601072] [2.0528242]%

Arizona [2.83563885] [-3.52825307]%

Colorado [2.82696396] [-7.98181662]%

Michigan [2.69551392] [2.37182457]%

Tennessee [2.64341651] [-4.68976289]%

Illinois [2.7043418] [2.7914522]%

Washington [2.95865456] [8.31563195]%

Wisconsin [2.71837193] [-5.36325292]%

Virginia [2.58670553] [-2.07993406]%

Missouri [2.70705615] [-10.49208767]%

Oregon [2.94595142] [6.12009509]%

South_Carolina [2.57935249] [-3.79688105]%

Maryland [2.58117127] [4.82406818]%

Ohio [2.64763513] [-2.7011301]%

Kentucky [2.64080753] [-0.64053087]%

Oklahoma [2.73059772] [-12.04750582]%

Connecticut [2.56853698] [7.9377427]%

Montana [2.89049492] [-9.57145255]%

Louisiana [2.65438948] [-6.77351089]%

Rhode_Island [2.55352045] [4.43411489]%

Georgia [2.58068509] [2.54210378]%

Utah [2.86444251] [3.35888974]%

New_Mexico [2.79837311] [-4.76874245]%

Kansas [2.76801167] [-11.16512742]%

Mississippi [2.64284802] [-7.17145242]%

Wyoming [2.85479464] [-11.25466237]%

New_Hampshire [2.56669902] [1.92208554]%

West_Virginia [2.6124409] [3.09937298]%

Indiana [2.67444188] [1.99919816]%

Arkansas [2.68724735] [-9.90786704]%

Nevada [2.89920584] [7.63918956]%

Maine [2.56238664] [5.55154278]%

Vermont [2.58557532] [4.48558114]%

Iowa [2.74203724] [-6.36296513]%

Delaware [2.56977963] [1.72926829]%

Nebraska [2.79296909] [-7.38058777]%

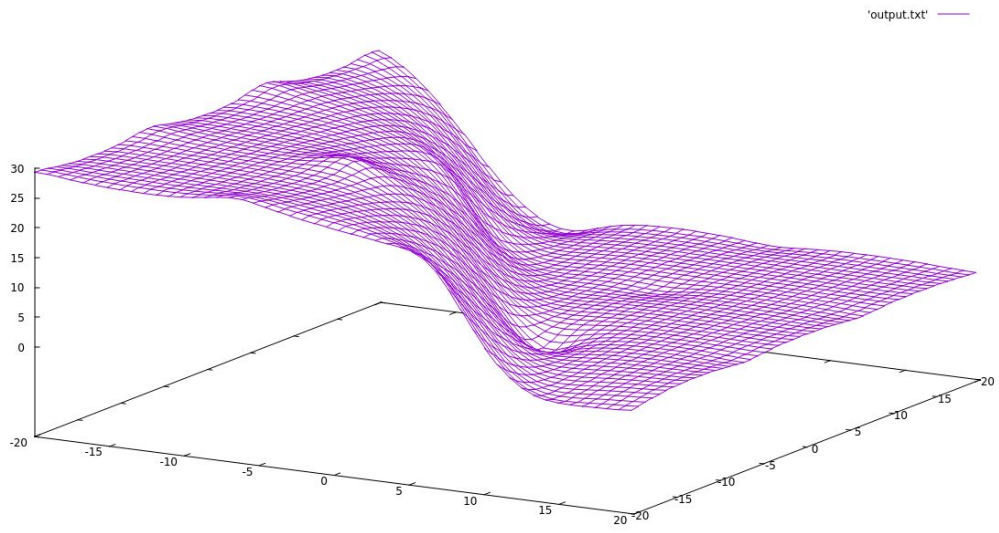
Idaho [2.90953965] [3.01534509]%

North_Dakota [2.82879711] [-7.47709384]%

South_Dakota [2.80537328] [-6.78999914]%

New_York [2.57333495] [8.74698752]%

The data above is from the neural network predictions of gasoline prices for each state. The percentage error from the actual price is also listed. This is after 60,000 training cycles with a 5 neuron network.



The above graph is a 2D cross section of the 6D error plot for the following network. This is the approximate gradient that the network descends over training.

Works Cited

Anderson, James A. *An Introduction to Neural Networks*. MIT Press, 1995.

Cancelliere, Rossella. "FNN." NN. www.di.unito.it/~cancelli/retineu11_12/FNN.pdf.

Guely, F., and P. Siarry. "Gradient Descent Method for Optimizing Various Fuzzy Rule Bases." *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*, 1993, doi:10.1109/fuzzy.1993.327570.

Lecun, Yann, et al. "Deep Learning." *Nature*, vol. 521, no. 7553, 2015, pp. 436–444., doi:10.1038/nature14539.

Pendharkar, Parag C. "A Comparison of Gradient Ascent, Gradient Descent and Genetic-Algorithm-Based Artificial Neural Networks for the Binary Classification Problem." *Expert Systems*, vol. 24, no. 2, 2007, pp. 65–86., doi:10.1111/j.1468-0394.2007.00421.x.

Shalizi, Cosma. "Logistic Regression." Undergraduate Advanced Data Analysis. www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf.

Touretzky, David S, and Dean A Pomerleau. "What's Hidden in the Hidden Layers?" *Byte*, 1 Aug. 1989, pp. 227–233.

Wehner, R. “Desert Ant Navigation: How Miniature Brains Solve Complex Tasks.” *Journal of Comparative Physiology A: Sensory, Neural, and Behavioral Physiology*, vol. 189, no. 8, Jan. 2003, pp. 579–588., doi:10.1007/s00359-003-0431-1.

Yonaba, H., et al. “Comparing Sigmoid Transfer Functions for Neural Network Multistep Ahead Streamflow Forecasting.” *Journal of Hydrologic Engineering*, vol. 15, no. 4, 2010, pp. 275–283., doi:10.1061/(asce)he.1943-5584.0000188.