Using an Intermediate Network to Optimize Parameters in Backpropagation Neural Networks

Gil Goldshlager
Anirudh Tadanki
Claus Zheng

Arising only fairly recently, the field of neural networks unites the disciplines of many different sciences, once thought to be independent areas of research. With incredible recent advances in technology, scientists were able to take inspiration from the biological neural network to create the artificial neural networks. By combining the speed of modern day computational technologies with the power of the biological neural network to establish connections and refine its learning with new information, computer scientists created an incredibly powerful method to mathematically establish knowledge and recognize patterns.

The inspiration to pursue neural networks came from a math and science summer institute we attended, where we took a class on automata theory. For our required original research project, we designed a new type of automata, the nodal automata. This type of automata contained input nodes that sent information to computational nodes whose final result was sent to the output nodes; the similarities to the neural networks are fairly evident. Continuing interest in our newly created machine led to further research, and we came across neural networks, whose applications go far beyond the capacities of normal computers. For example, they are prevalent in facial recognition, credit score prediction, and artificial intelligence. The potential of what neural networks can do encouraged us to do further research.

Programmers of neural networks write training algorithms to interpret gathered data and find patterns within this data, just as humans, for example, learn to associate bright red objects with hot objects. Furthermore, as new information is provided to the system, the neural network is able to refine what it knows to accommodate new findings, similar to how discovery of white

and blue stars led humans to conclude that these objects are even hotter than bright red objects. Neural networks can recognize and infer these complicated patterns because they are nonlinear; every node in the input layer sends information to every node in the hidden layer, after which every node in the hidden layer sends information to every node in the output layer, which synthesizes this information into meaningful outputs. As new information is fed into the hidden layers, the nodes and connections between these nodes are constantly changed to become more consistent in associating the given data with the correct output (Neural Networks and APA, n.d.).

Today, neural networks are used for prediction, classification, and processing of data. For prediction, neural networks model current data and estimate what future data might be, such as taking oil consumption data for the past fifteen years and predicting what oil consumption will be for the next five years. For classification, neural networks find patterns in data, such as in recognition of faces. For data processing, neural networks store data in an organized manner for easy retrieval, such as in search indexing. Once the training algorithms for neural networks are established, they can constantly incorporate new data into their databases, reducing the need for constant reprogramming. This makes neural networks not only more accurate but more cost-effective than traditional computing methods (Neural Networks and APA, n.d.).

One specific area in which the authors see potential for neural networks is in disease diagnosis based on symptoms. The network could train itself on the huge amounts of medical data that hospitals have about past patients' symptoms and afflictions and, once trained, would be a quick and easy way for doctors to get an idea of what diseases a patient might have. This method could be a quicker, more efficient, more accurate, and more cost-effective way to diagnose diseases. The major difficulty in the creation of such a network is the training process: training a neural network can be a long and arduous endeavor that requires much time and

resources. For this reason, much research has been done to develop faster ways to train neural networks, and specifically backpropagation neural networks.

Jing-Ru Zhang, Jun Zhang, Tat-Ming Lok, and Michael R. Liu (2007) all agree that the efficiency of backpropagation neural networks can be improved through the improved selection of training parameters with artificial intelligence. Currently, the constants are often chosen arbitrarily, but Zhang et al. suggest that using a neural network or other optimization algorithm to find the constants for another neural network would improve efficiency and overall time it takes one to find the best way to train a neural network. In this paper, we explore the idea of developing an intermediate backpropagation neural network that can determine based on a small representative sample of the training data the best parameters and structure for the backpropagation network that will recognize the data.

In this experiment, we work with two levels of neural networks. The "lower level" neural networks addressed in this research experiment take the original and mostly arbitrary data gathered by the user and calculate a mapping between the input data and the output data. For example, one of the neural networks is capable of taking GDP per capita, unemployment rate, and other pieces of information to calculate whether a country is considered to be a first or third world country. The "higher level" neural network is designed to calculate optimal constants used to train the "lower level" neural network so that the lower level network can be trained more efficiently. The way that the higher level network operates is essentially the same as the way the lower networks operate. However, a significantly larger effort is needed to generate the training data for this higher level network: essentially, for each lower level network, many different parameter values are tested to find the optimal set of parameters. The training data for the high level network then consists of representative samples of the eight low level networks'

data; each sample is mapped to the optimal parameter for its corresponding network. With this data in hand, the higher level network is trained in the same manner as the lower level networks- the inputs are fed repeatedly into the network, the outputs generated are compared with the desired outputs, and the network's weights are adjusted to bring the calculated outputs closer to the desired outputs. In this way, it was hoped that the higher level network would recognize patterns that link the type of data that a neural network is trying to recognize to the best parameters to train that network.

The successful development of an all-purpose network that could determine optimal training parameters based on a network's training data would have applications anywhere that backpropagation networks are used.

THE GOAL

The fundamental goal of this project is to create a backpropagation neural network that can determine the optimal training parameters to train another neural network with a different goal based on only characteristics of the training data. Such a neural network has the capability of improving the accuracy and speed of the training process for backpropagation networks used for any application.

THE PROCEDURES

This research was based on improving the optimization of the learning process of any backpropagation network. Thus, a variety of different networks were created and, for each one, optimal training parameters were empirically determined. A total of eight various sets of data were created and tested with varying parameters in the neural network. For the sake of uniformity, all of the sets of data map a group of sixteen numbers to a single value.

For the first group, chains of sixteen zeros and ones, representing Boolean false values

and Boolean true values, undergo a fixed, preset chain of Boolean operations. One hundred fifty such chains that are mapped to their actual outputs constituted the training data. After training on this set of data, fifty random new chains of sixteen numbers are inputted to test whether the neural network can predict the final result when these chains of false and true values undergo the same Boolean operation.

For the second group, data for various countries, such as GDP, life expectancy, and infant mortality, are proportionally scaled down to a zero-to-one interval. These twelve pieces of data are put into a chain of twelve numbers, which is then followed by four zeros to create a chain of sixteen numbers. For the ninety countries that constituted the training data, a one is associated with a first-world country, and a zero is associated with a third-world country. After the network is taught to recognize these ninety, data for forty-six other countries is given to test whether the neural network can associate these countries with their first-world or third-world status.

For the third group, three four-by-four bitmap "templates" of zeros and ones are created along with thirty three other bitmaps that are each clearly most similar to one of the three templates. These thirty six bitmaps are listed as chains of sixteen zeros and ones and together constitute the training data. The neural network first learns to tell apart the three templates based on these thirty six bitmaps and then tests itself on a verification data set of eight other bitmaps. If the network correctly identifies at least 7 of the 8, it is deemed to have successfully recognized the pattern. As far as outputs, a negative one represents the first template, a zero represents the second, and a positive one represents the first.

For the fourth group, forty-four random, sixteen-digit binary sequences, twenty-one human-generated and twenty-three computer generated, serve as the training data. Each human-generated sequence is mapped to a zero, and each computer-generated sequence is mapped to a

one. Then, thirty-six sequences, some human-generated and some computer-generated, are inputted to find if the neural network can correctly classify each sequence. Though this network is not as accurate in its predictions as some of the others, it is still correct about 75% of the time, which is deemed fairly consistent considering the minute differences that it is trying to notice.

For the fifth group, the neural network is provided with one hundred fifty pieces of training data that associate chains of sixteen zeros and ones with the proportion of ones in each sequence. Then, fifty sixteen-number sequences are inputted to determine whether the neural network can predict the proportion of ones in the sequences.

For the sixth group, fifty three-letter syllables are randomly generated. Each letter is encoded as a five-digit binary chain. The three binary representations of the letters are put together and a single zero placeholder is added to obtain a sixteen-number chain. Then, for the training data, each pronounceable syllable is given a value of one, and each unpronounceable syllable is given a value of zero. Finally, four hundred and fifty pieces of input data are given to test if the neural network can determine if the syllables are pronounceable.

For the seventh group, the daily high temperature for sixteen consecutive days is associated with the temperature of the next day. Thirty is subtracted from each of the seventeen temperatures in Fahrenheit, and each of these temperatures is then divided by hundred. For the learning data, one hundred seventy-five sets of sixteen numbers are provided with their correct output. Then, sixty-eight new sequences of temperature data are provided to determine if it can accurately predict the temperatures of the seventeenth day.

For the eighth group, twenty-five four-by-four bitmaps are created to serve as the learning data. Each bitmap in which there exists a pathway consisting of only zeros from the top left corner to the bottom right corner is assigned an output of one. Otherwise, the bitmap is given an

output value of zero. Then, six more maps are tested to find if the neural network can accurately classify the additional bitmaps.

For each of the 8 networks, a table giving a sample of the training data for the network is shown below. The first column gives the desired output associated with each input, which is specified by the following 16 entries in the table.

Group 1:

0.2	1.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.8	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0
0.2	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
0.2	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0
0.2	1.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
0.8	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0

Group 2:

0.8	0.1548	0.7582	1.79	0.12	0.757	0.081	0.005	0.01558	0.1205	0.1205	0.1215	0.1205
0.2	0.1748	0.04508	0.052	0.235	0.7619	0.128	0.069	0.059	0.2302	0.2302	0.2285	0.2302
0.8	0.085	0.2182	0.621	0.14	0.8214	0.082	0.02	0.006712	0.0232	0.0232	0.2475	0.0232
0.2	0.089	0.9207	0.357	0.05	0.7832	0.019	0.052	0.08295	0.0518	0.0518	0.35	0.0518
0.2	0.2097	2.38	0.035	0.6	0.668	0.134	0.108	6.64082	0.4757	0.4757	0.374	0.4757
0.2	0.1229	4.284	0.076	0.16	0.7468	0.049	0.043	7.7216	0.1606	0.1606	0.3515	0.1606

Group 3: Note that here, the strings are meant to represent 4x4 arrays. The three templates were

1 0 1 0	1 1 1 1	0 1 0 0
0 1 0 1	1 0 0 1	0 1 1 1
1 0 1 0	1 0 0 1	1 1 1 0
0 1 0 1	1 1 1 1	0 0 1 0

And here are 6 other examples of strings input:

-1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0
0.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0
0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0
0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0
1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0
-1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0

Group 4:

0.8	0.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0
0.8	0.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0
0.2	1.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0
0.2	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0

0.2	1.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
0.2	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

Group 5:

0.5	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
0.25	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
0.5625	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0
0.375	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0
0.4375	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0
0.625	1.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0

Group 6:

0.2	1.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0
0.8	0.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0
0.2	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0
0.2	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0
0.2	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0
0.2	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0

Group 7:

.19	0.42	0.34	0.29	0.31	0.24	0.27	0.33	0.26	0.13	0.08	0.1	0.12	0.13	0.23	0.26	0.3
.35	0.34	0.29	0.31	0.24	0.27	0.33	0.26	0.13	0.08	0.1	0.12	0.13	0.23	0.26	0.3	0.19
.29	0.29	0.31	0.24	0.27	0.33	0.26	0.13	0.08	0.1	0.12	0.13	0.23	0.26	0.3	0.19	0.35
.29	0.31	0.24	0.27	0.33	0.26	0.13	0.08	0.1	0.12	0.13	0.23	0.26	0.3	0.19	0.35	0.29
.22	0.24	0.27	0.33	0.26	0.13	0.08	0.1	0.12	0.13	0.23	0.26	0.3	0.19	0.35	0.29	0.29
.19	0.27	0.33	0.26	0.13	0.08	0.1	0.12	0.13	0.23	0.26	0.3	0.19	0.35	0.29	0.29	0.22

.

Group 8: Note that these again are meant to represent 4x4 arrays. For example, the first is
0 1 1 1
0 1 0 1
1 0 1 1
1 1 0 0, which yields 0 because there is a path of 0s from the top left to bottom right.

0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0
0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
1.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0

After optimizing the training parameters of the neural networks that recognize each of the previously described data sets, another neural network is created with the goal of determining the

optimal parameters for a backpropagation neural network in general. Specifically, this network takes in as input a representative sample of 10 input-output pairs from one of the eight listed networks as well as the total amount of data that the network was trying to recognize, and gives as output the best training parameter and the number of neurons as a proportion of one thousand necessary in the hidden layer of a network that recognizes this data.

We originally tried to develop the framework for the backpropagation neural network with Encog, which is a professional artificial intelligence framework for neural networks. However, the structure of Encog is very complex, which makes it difficult to make the low level changes that are necessary for a project such as this one. This led us to make our own framework, which is far simpler and contains only the needed classes so that it is easier to edit. The general algorithm is the same as the original Encog core, but it is simpler and just as quick. Both Encog and our own neural network framework use the Java programming language. In our framework, we created classes to represent an entire network, a layer of a network, and a single neuron of a network, as well as classes to represent various activation functions – we used linear functions for input and output layers and sigmoidal functions for the hidden layer- and a class to implement the backpropagation training algorithm.

We worked well as a team and divided and conquered what had to be done. We split the project into programming and data collection. One person did the programming, for this person was the most adept at it, and the other two people did the data collection, the data interpretation, and the data manipulation. All three of us worked on the research paper. Our mentor was there for general assistance and moral support. Our mentor helped us when we were stuck on a particularly difficult part of the programming or when we needed someone to discuss general concepts with, but the vast majority of the work was done by us.

THE RESULTS

For each of the eight networks, many different values for the learning rate parameter are tested and the resulting number of training epochs is recorded to determine the optimal constant with which to train the network. Since the work is essentially the same for all of the eight networks, we will only display the full information for two of the eight networks. For the rest, we will simply state the determined ideal value.

Now, the following is a table that shows the number of training epochs to train the image recognition neural network based on the value of the learning rate. It should be noted that the results gathered directly from trials were originally too sporadic; in order to determine the best value both speed at the value and speed at surrounding values are thus considered. Specifically, the five numbers centered on each value are averaged, and it is these averages that are displayed below. The value with the lowest such average is considered to be best, and is listed below as well as bolded in the table.

Learning Rate	Epochs	Learning Rate	Epochs	Learning Rate	Epochs
0.0163	317.4	0.0352	225.6	0.0548	99.8
0.017	277.4	0.0359	261.4	0.0555	94.6
0.0177	283.2	0.0366	226.4	0.0562	96.8
0.0184	269.4	0.0373	291.4	**0.0569**	**79.6**
0.0191	233.8	0.038	260.4	0.0573	82.2
0.0198	241.6	0.0387	254.8	0.0583	92.4
0.0205	265.4	0.0394	217.4	0.059	148.8
0.0212	253.2	0.0401	216.2	0.0597	154.0
0.0219	250.4	0.0408	149.6	0.0604	148.6
0.0226	236.8	0.0415	114.8	0.0611	163.4
0.0233	293.6	0.0422	130.8	0.0618	197.8
0.024	296.4	0.0429	121.2	0.0625	139.6
0.0247	266.2	0.0436	214.0	0.0632	208.8
0.0254	242.2	0.0443	209.2	0.0639	211.8
0.0261	241.4	0.0457	240.2	0.0646	208.0
0.0268	189.6	0.0464	231.4	0.0653	173.4
0.0275	168.2	0.0471	180.0	0.066	203.0
0.0282	182.6	0.0478	215.6	0.0667	142.0

0.0289	221.8	0.0485	190.8	0.0674	192.0
0.0296	212.4	0.0492	189.2	0.0681	189.4
0.0303	216.2	0.0499	196.6	0.0688	178.6
0.031	204.6	0.0506	175.8	0.0695	159.6
0.0317	190.6	0.0513	140.2	0.0702	151.0
0.0324	163.4	0.052	141.2	0.0709	133.8
0.0331	193.4	0.0527	113.2	0.0716	142.0
0.0338	184.2	0.0534	126.8	0.0723	142.2
0.0345	228.2	0.0541	108.6		

Optimal learning rate: .0569

In this particular instance, the network is trained to have a sum squared error of at most .1. For this network, .1 error is low enough to ensure successful classification of never before seen images. In the following table we display the ideal and calculated values of the outputs for 8 never before seen images; clearly, the results are close enough to avoid any ambiguity.

Ideal value	Calculated value
0.0	0.047445494302309224
1.0	1.0026982494240626
1.0	1.028626025158099
-1.0	-1.0761292589947873
-1.0	-1.02133091383444
1.0	1.0120860307251314
-1.0	-1.0025694466263535
1.0	1.0026982494240626

Now, as a second example, we present a table of learning rate values for the pathfinder network along with the number of epochs to train the network with each value. Because this network takes significantly longer to train than the image recognition network, restrictions must be placed on how long the network is allowed to train itself before simply being cut off and dismissed as a non-optimal constant. Specifically, the network is first trained with a human selected constant known to lead to a reasonable number of training epochs. Then, as the training process proceeds, the minimum number of epochs taken is recorded and any training process that takes more than twice as many epochs as the minimum attained value is terminated and recorded as a -1.

Learning Rate	Epochs	Learning Rate	Epochs	Learning Rate	Epochs
5.0E-4	-1.0	0.01832	2648.0	0.03614	-1.0
9.95E-4	-1.0	0.018815	3797.0	0.036635	-1.0
0.00149	-1.0	0.01931	-1.0	0.03713	-1.0
0.001985	-1.0	0.019805	-1.0	0.037625	845.0
0.00248	-1.0	0.0203	-1.0	0.03812	-1.0
0.002975	-1.0	0.020795	-1.0	0.03861	-1.0
0.003467	-1.0	0.02129	-1.0	0.03911	-1.0
0.003965	-1.0	0.021785	-1.0	0.039605	1116.0
0.00446	-1.0	0.02228	4111.0	0.0401	-1.0
0.004955	-1.0	0.022775	2285.0	0.040595	-1.0
0.00545	-1.0	0.02327	1218.0	0.04109	-1.0
0.005945	-1.0	0.023765	1922.0	0.041585	-1.0
0.00644	3124.0	0.02426	846.0	0.04208	-1.0
0.006935	-1.0	0.024755	-1.0	0.042575	-1.0
0.00743	3625.0	0.02525	-1.0	0.04307	-1.0
0.007925	2258.0	0.025745	-1.0	0.043565	-1.0
0.00842	-1.0	0.02624	1134.0	0.04406	1431.0
0.008915	3858.0	0.026735,	1687.0	0.044555	-1.0
0.00941	3105.0	0.02723	-1.0	0.04505	-1.0
0.009905	-1.0	0.027725	1353.0	0.04554	-1.0
0.0104	-1.0	0.02822	-1.0	0.04604	-1.0
0.010895	2719.0	0.028715	-1.0	0.046535	-1.0
0.01139	3576.0	0.02921	-1.0	0.04703	-1.0
0.011885	2904.0	0.029705	-1.0	0.047525	-1.0
0.01238	3748.0	0.0302	-1.0	0.04802	-1.0
0.012875	3722.0	0.030695	1451.0	0.048515	-1.0
0.01337	-1.0	0.03119	1651.0	0.04901	1511.0
0.013865	3158.0	**0.031684**	**1671.0**	0.049505	-1.0
0.01436	-1.0	0.03218	998.0		
0.014855	2582.0	0.032675	1403.0		
0.01535	4018.0	0.03317	-1.0		
0.015845	2541.0	0.033665	-1.0		
0.01634	-1.0	0.03416	-1.0		
0.016835	3933.0	0.034655	1055.0		
0.01733	-1.0	0.03515	1330.0		
0.017825	3156.0	0.035645	-1.0		

Optimal learning rate: 0.031685

In this case, the optimal rate was again chosen based on the number of epochs yielded both at each specific value and at the values surrounding each one. The network was trained to have a sum-squared error of at most .01. Like last time, .01 error is small enough to guarantee

that the network will not make ambiguous predictions. The table comparing its predicted output and ideal output for six never before seen inputs is shown below:

Ideal	Calculated
1.0,	0.9377693360420392
1.0,	0.9444113635037008
1.0,	0.9697974505787205
1.0,	0.9321931919729862
1.0,	0.9664791146876943
0.0,	-0.01634569876847005

Now, a summary of the optimal values for all eight networks:

Number	Network Function	Optimal Learning Rate
1	Country classifier	.001
2	Boolean Function	.0374
3	Image recognition	.0569
4	Human vs. computer generated	.0013
5	Proportion of 1's	.004412
6	Syllable recognizer	.003805
7	Temperature	.00078
8	Path finder	.031685

With this data in hand, we can proceed to the results of the higher level network. The reader will recall that in this network, a representative sample of 10 input-output pairs for a neural network is mapped to the best learning rate to train that network. Before entering the network, all inputs are normalized so that the network can recognize patterns rather than magnitude. For example, for the image network, 10 images with their corresponding outputs of -1, 0, or 1, are scaled down by a large factor and used as input, and .0569 is used as output. For this network, the primary result is that it has thus far been impossible to lower its sum-squared error to the point that it can actually recognize patterns in the data. To illustrate this, we first present the following chart showing the sum-squared error of the network after successive training epochs. The chart reads from the top left, down the first column, down the second column, and then down the third column.

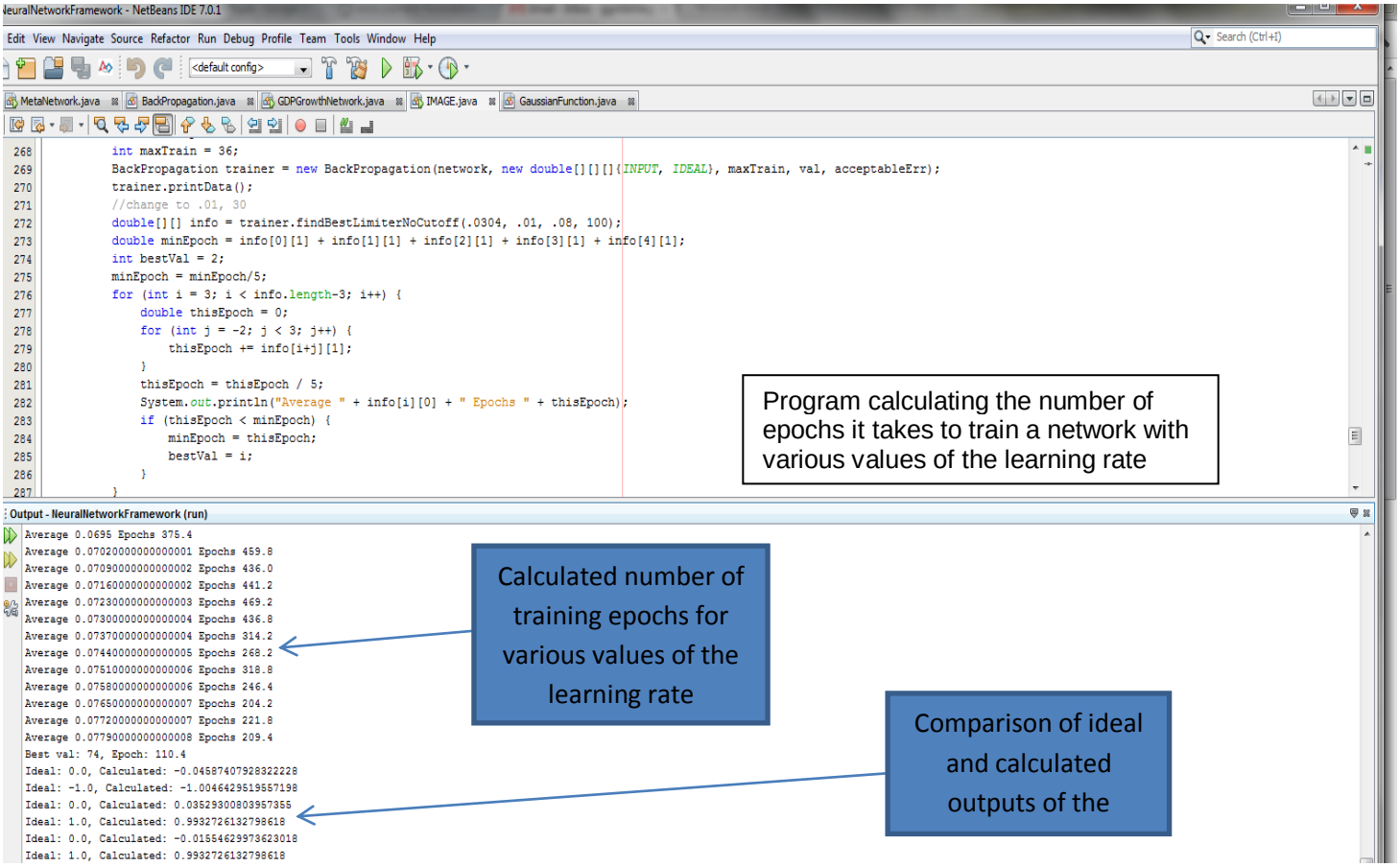
0.015881223509064003	0.015814089657039454
0.01587709304906262	0.015811968391593767
0.015873089345430587	0.015809908706717366
0.015869208139913462	0.015807908693675446
0.015865445334066915	0.01580596650871996
0.01586179698256015	0.015804080370609045
0.015858259286786906	0.015802248558240486
0.015854828588773295	0.01580046940838301
0.0158515013653547	0.015798741313500144
0.015848274222625414	0.01579706271966953
0.01584514389063089	0.015795432124586065
0.01584210721830655	0.01579384807564781
0.01583916116863535	0.01579230916812587
0.015836302814033844	0.015790814043402254
0.0158335293319312	0.015789361387288484
0.015830838000555587	0.015787949928403086
0.015828226194901762	0.01578657843662636
0.01582569138287654	0.015785245721609897
0.0158232311216162	0.015783950631347597
0.01582084305396613	0.015782692050804865
0.015818524905111542	0.015781468900606794
0.015816274479355224	0.015780280135769594

Here is a chart of the calculated and desired outputs for the verification data set after the training shown above was completed.

Ideal	Calculated
7.353333333333334E-6,	-7.125471856433974E-5
-0.1590404182398875,	-0.16067814738365432
7.353333333333334E-6,	-7.125471856433974E-5
-0.1590404182398875,	-0.16067814738365432
7.353333333333334E-6,	-7.125471856433974E-5
-0.1590404182398875,	-0.16067814738365432
1.3E-6,	-7.125471856433974E-5
-0.20068666377598746,	-0.16067814738365432
1.3E-6,	-7.125471856433974E-5
-0.20068666377598746,	-0.16067814738365432

ILLUSTRATIONS

1.



JUSTIFICATION OF RESULTS

The results of the eight lower level neural networks are largely straightforward, but still merit some discussion. One of the most shocking results is that even miniscule changes in the learning rate can lead to immense differences in the number of epochs necessary to train the network. If the learning rate is too far off, the network may simply stagnate at some relatively large level of error and never fully converge to the correct outputs; additionally, with a constant that is too large, a network can spiral out of control and cause its error to tend towards infinity rather than zero. These rather unexpected findings just go to show how necessary it is to have an accurate way of selecting effective training constants rather than selecting them randomly.

Another point that is worthy of some discussion is that all eight of the tested networks were successful in detecting the patterns presented to them; while this is not surprising for most of them, it is interesting to note that network 4 was able to recognize with a fairly high accuracy the difference between computer and human generated random sequences. This could have applications in detecting fraud by discovering when supposedly random data was not actually generated randomly.

Now, as for the higher level network, it can be seen that the error shrinks to a very low level- around .0157- but at that point begins to stagnate. In fact, the network appears to reach an asymptote here, never falling below .0157 after any number of epochs. While this may seem good enough, the verification data shows that the network has simply trained itself to output a consistent value of the same magnitude as the desired outputs in order to give overall low error. Many hidden layer sizes, from 50 to 10000, have been used to attempt to remedy this problem, and additionally everywhere from one to four hidden layers have been used. Also, different activation functions have been tested, including the Gaussian function, a function resembling a normal distribution, and a linear function. It does turn out to be more effective to train this network by adding up all the errors over all of the training data before updating connection weights, but the improvement is with speed and not accuracy.

Of course, it should also be kept in mind that the task presented to the higher level network was an enormous one. The lower backpropagation networks map inputs to outputs that the authors either knew or had significant reason to believe could be related to the inputs relatively simply. The higher level network, on the other hand, creates a mapping between an input and an output data related in an incredibly complex way. It is entirely possible that no

backpropagation network of a reasonable size can successfully solve the problem that this project set out to solve.

What makes this experiment unique is that it is the first project to use an intermediate neural network to optimize learning rates and efficiency. While many have researched ways to optimize learning rates for backpropagation networks, none have used another backpropagation neural network.

THE CONCLUSION AND FUTURE WORK

The final high level backpropagation neural network holds great promise for the future if it can made a little more accurate in order to have predictive power. As far as neural networks in general, we believe that there are still areas in which neural networks could be very successful but to which neural networks have not been applied to significantly; one example of this is in diagnosis of diseases. As far as this project and its specific goals, while the project was not a complete success, we believe that it is certainly a good first step for a potentially very fruitful task.

The final result for the intermediate network is just one that we made. Though the final network is not yet accurate enough to be used to make predictions, that may be because of an insufficient amount of the data. In general, the idea of using an intermediate network in the training process shows great promise. Additionally, the research done in this project strongly agrees with other projects of similar nature that a good way to select neural network training parameters is greatly needed.

Now, there are several ways in which this project might be improved, some of which have been mentioned earlier. First of all, it may be that we simply did not have the time and resources to produce enough example networks to successfully train the higher level network.

Additionally, although backpropagation neural networks are the most common and most practical networks used to recognize most data sets, there are many other kinds of neural networks that could be used, some of which may give better results. Examples include the counterbackpropagation neural network, the simulated annealing neural network, and the networks described by adaptive resonance theory. These neural networks approach the data differently than backpropagation networks, and therefore have the potential to offer different insights.

If there was more time to work on this project, we would find larger pool of data to feed to the higher level network, explore other kinds of networks, and/or look for different kinds of data to feed to the lower level network. The most promising change to our research may be to experiment with different types of networks for the higher level network, for the backpropagation network may not be the most effective type for this application. If the project was started today, then we would also investigate dynamically determining the learning rate of a network during the training process, so that any initial error in the choice of parameters could be corrected as the network is trained. Additionally, we would further investigate similarities between an artificial neural network and a biological neural network and see if this could lead us to any new ideas or insight into the field of neural networks in general.

After this project, we still do not know whether it is possible to develop on a large scale the kind of intermediate network that we have explored, and we do not know whether we might have had more success with different types of networks. However, we believe that this could be a promising area of research for future projects by ourselves or others.

References

Neural Networks. (n.d.). *Data Mining, Statistical Analysis Software, Predictive Analytics, Credit Scoring*. Retrieved August 2, 2011, from http://www.statsoft.com/textbook/neural-networks/

Deng, W., Chen, W., & Pei, W. (2006, December 22). ScienceDirect - Expert Systems with Applications : Back-propagation neural network based importance “performance analysis for determining critical service attributes. *ScienceDirect - Home*. Retrieved August 2, 2011, from http://www.sciencedirect.com/science/article/pii/S0957417406003988

Skapura, D. (1995). Foundations & Paradigms. *Building Neural Networks* (pp. 6-41). New York: ACM Press.

Zhang, J., Zhang, J., Lok, T., & Liu, M. (2007, February 15). ScienceDirect - Applied Mathematics and Computation : A hybrid particle swarm optimization back-propagation algorithm for feedforward neural network training. *ScienceDirect - Home*. Retrieved August 2, 2011, from http://www.sciencedirect.com/science/article/pii/S0096300306008277

Abstract

Over the last several decades, neural networks have become all but indispensable in such fields as facial recognition, fraud detection, and data mining, and by far the most commonly used type of network is the backpropagation network. An essential part of any project that utilizes a backpropagation network is the correct selection of the learning rate- a parameter that can determine whether the network efficiently learns to recognize patterns or spirals out of control. Nevertheless, learning rates are often chosen through trial and error or even at random, causing a severe risk of inefficiency. In this paper, we propose and explore the development of an intermediate neural network to determine the optimal parameters to train a backpropagation neural network. This secondary network has the potential to take the guesswork out of the process of selecting the learning rate, and thus optimize the speed at which neural networks can be trained for any application. In this paper, a network is explored that does this for a certain restricted class of networks, but the results have the potential to be generalized to apply to networks of any type.

Executive summary

Despite the incredible computational power and speed of computers in the modern world, there are some areas in which humans are simply more effective than computers. For example, recognizing a person's face is for a computer a monstrously difficult task, but is one that comes naturally to people. In order to attempt to mimic the capability of humans to quickly and accurately recognize extremely complex patterns, several computational models that mimic the brain have been created. One of the most common and most effective model is the neural network, which consists of a set of neurons that send impulses- really, numbers- to each other, much like the human brain. The most common type of neural network by far is the backpropagation network. In this type of network, inputs are entered into an input layer of neurons, processed by a number of intermediate layers, and then transformed into outputs by an output layer. The calculated output is then compared to known data specifying what the output should be, and the error between the two values is used to adjust the weights of the network in order to come closer to the ideal value.

In the development of any backpropagation network, there are certain parameters that must be somehow selected by the user. An incorrectly chosen parameter can literally cause a thousand-fold increase in the amount of time necessary to train a network, and yet often such parameters are chosen by trial and error or randomly. In our project, we aim to provide a better way to select one of these parameters- namely, the learning rate of the network. Our goal is to use another, secondary neural network to determine the parameters based on a small representative sample of the training data. With this method, backpropagation neural networks in all applications could be used more efficiently.