Neural Network Topology and Performance, an empirical analysis

———Big Data and Machine Learning Project Set 2 Question 3

Weichuan Wu ,Li Ding,Yangyang Li,Yuefan Zhu

December 19, 2016

1 Introduction

It is intuitive that neural network can do a better job by adding more layers or neurons. But in a fully connected network, number of parameters, or weights and biases grow exponentially with number of layers and neurons. Then given size of data set and limitation of computational power, it seems impossible to improve the power of neuron networks. On the other hand, overfitting becomes a major problem, if we fail to reduce number of parameters. The resulting neural network will suffer huge test error, even if the training error is low.

Facing this tradeoff, we want to come up ways to reduce number of parameters, given a fixed number of neurons and layers, while maintaining the power of neural network. By 'power', we refer to level of test error, given certain amount of time and computational power. An alternative definition would be the time it would take to achieve certain level of test error.

The line of thinking naturally raise the need to perform the following empirical research: by randomly turning off edges in different layers, we want to know the performance of network. More specifically, we want to understand the relationship between various parameters describing the graph of connections between neurons and the learning rate and quality.

The natural selection of indicators would be some topological features of the network, such as number of vertex-disjoint/edge-disjoint paths between neurons from the input layer and from the output layer, the connectivity of the graph, the sparsity of the graph, etc.

For each neural network under review, we record its average performance and its topological features. We first present some typical cases to give some intuitive result. Then we use the whole sample to come up with strict empirical results.

2 Data

2.1 MNIST

The MNIST database (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. It has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size images. Each sample image is 28 x 28 and linearized as a vector of size 1 x 784. In this project, we randomly divided the database into train, valid and test subsets and then created mini-batches of size 100 from the train and valid sets for training and examined all the empirical results based on the test subset.

2.2 Variable Summary

Vertex disjoint path (disjoint in terms of hidden layer): By definition, two paths are vertex disjoint if they do not have any internal vertex in common, yet they can share same start and end vertex. In our project, our focus is to find the maximum disjoint paths. Since there is no edges within each layer, all the paths are connected horizontally among input layer, hidden layers and output.

Edge disjoint path: By definition, two paths are edge-disjoint if they have no edge in common. Our goal is to find the maximum edge disjoint paths in our network. Similar to vertex disjoint path maximization, since there is no edges vertically within each layer, all the paths are horizontally connected.

k-Connectivity: By definition, a network is said to be k-connected if there does not exist a set of k-1 vertices whose removal disconnects the network. Our goal is to find the maximum k such that the network is k-connected.

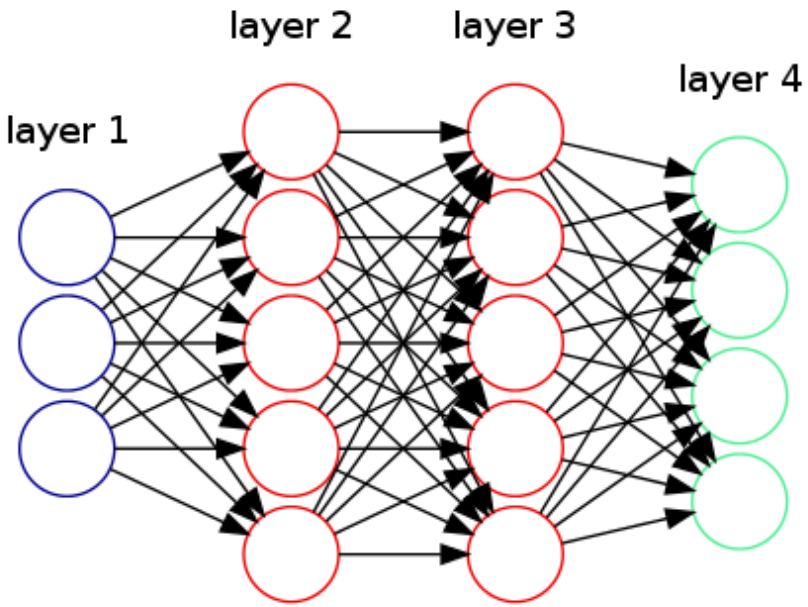

Figure 1: Full connected network with two hidden layers.

Sparsity of the graph – Vertex Average Degree: By definition, the degree of a graph vertex v of a graph G is the number of graph edges which touch v. In our project, we used the vertex average degree to evaluate the sparsity of the graph.

3 Model Review

3.1 Setup

Figure1 is a sample structure of the network. The model input is a 784-dim vector. The output is a 10-dim vector, with one and only one entry non-zero. This fixed the form of input and output layer.

We use two hidden layer, to first allow more flexibility in adjusting network topology. It make little sense to turn off edges for one-hidden-layer network. Also, the training time is reasonable, compared with three-hidden-layer case.

Number of neurons in the hidden layer are set to be 100, also an intuitive choice, between 784 and 10. It is chosen such that our independent variables are well spread, which is good for empirical research.

Table 1 show other technical details.

3.2 Training

We use backpropagation to train a feedforward network. More specifically, we formulating the problem as a series of matrix multiplies. This allow us to manipulate the weighting matrix between layers, W_1, W_2,W_3, which determines the structure of the network.

More specifically, the weighting matrix W can be formulated as the pointwise product between two matrices C and M, such that $W = C \cdot M$ where $C_{ij} = 1$ if i^{th} neuron in previous layer and j^{th} in the next layer is connected,else $C_{ij} = 0$, and M is the matrix that contains the weights and is to be updated.

We get our samples by manipulating matrix C in different ways: Randomly[5] turn off edges between

1. input and hidden layer I.
2. first and hidden layer II.
3. hidden layer II and the output layer.

[5]Each entry in C is made binomial

Table 1: Network technical details

Training Algorithm	Backpropagation
Activation Function	Sigmoid Function
Batch Size[1]	60000
Mini-Batch Size[2]	100
Epoch Size[3]	10
Learning Rate[4]	0.05

[1] We unpacks the MNIST data of size 60000 and chunks it into mini-batches randomly, and divide them into train/validate/test set.
[2] Mini-Batch Size means the size of training data. This choice avoid overfitting, and manage the training time in a reasonable range.
[3] Epoch means number of training iterations.
[4] The learning rate does not change during training

4. all three matrices all together.

The reason for doing this is to compare relative importance of connections between different layers. Given the structure of the network, the training algorithm is standard, with the following steps:

1. Forward propagation:
   - Propagate the input to the first layer
   - For each of the hidden layers compute input and store the activation derivatives for the back-propagation step
   - Calculate output for the layer, for the sigmoid activation function
2. Backpropagation
3. Weights update

3.3 Variables Calculation

3.3.1 *Edge Disjoint Path*

- Formulate an artificial source S and terminal T. Connect S with every neuron in first input layer, T with output layer, set those edges' capacity big enough.
- Set the capacity of other edges to be 1
- Solve for number of paths as maximum flow problem[6] between S and T

3.3.2 *Vertex Disjoint Path*

- Formulate an artificial source $'S'$ and terminal $'T'$. Connect $'S'$ with every neuron in first input layer, $'T'$ with output layer, set those edges' capacity big enough.
- For hidden layer I and II, create an artificial layer I' and II' corresponding . Connect neurons between each pair of layer one by one, by an edge with capacity 1.
- Solve for number of paths as maximum flow problem between S and T

[6] We use 'networkx' package for this purpose

3.3.3 *K-connectivity*

The edge connectivity is equal to the minimum number of edges that must be removed to disconnect G or render it trivial. If source and target nodes are provided, this function returns the local edge connectivity: the minimum number of edges that must be removed to break all paths from source to target in G.

3.3.4 *Vertex Average Degree (Sparsity)*

It is derived by taking average degrees in the whole network.

3.3.5 *Input Layer Dropout number*

It is the number of neurons in input layer that is isolated with the rest of network.

3.3.6 *Output Layer Dropout number*

It is the number of neurons in output layer that is isolated with the rest of network.

4 Experiment Result

We examine four cases, according to different ways of manipulating weighting matrices.

4.1 Randomly turn off all three sets of connections among four layers

As it show in figure2, when we randomly turn off all three sets of connections among four layers, we have the observation below:

- The relationships between test errors against three connectivity variables (edge disjoint path, vertex disjoint path, K connected) are similar. It indicates that the sensitivity of test error with connectivity of graph is high when the graph is sparse and then decreases drastically when connectivity goes higher. It is supported by the relationship between test error and number of input/output layer nodes not in use.

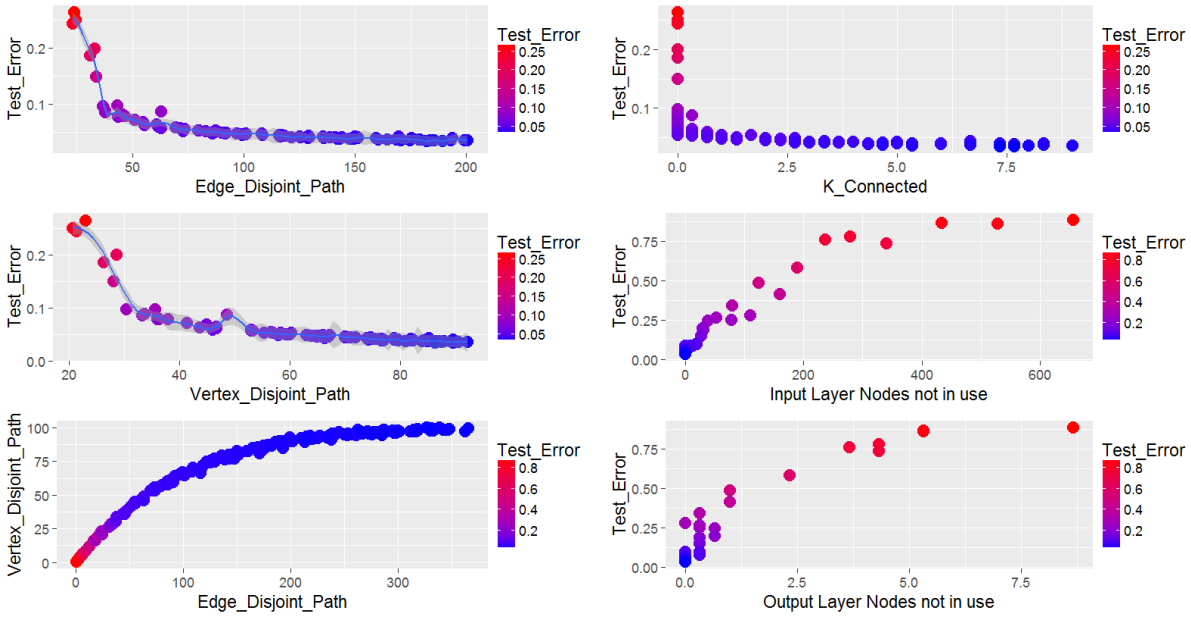

Figure 2: The result of randomly turn off all three sets of connections among four layers

4.2 Randomly turn off connection between input layer and first hidden layer

As it show in figure3, when we randomly turn off connection between input layer and first hidden layer, we have the observation below:

- In the upper left picture, we plot the relationship between running time and average vertex degree. Average vertex degree measures the sparsity of the graph and its positive related to the number of edges. We draw the conclusion that running time is positively correlated to the sparsity of the network.
- We plot other topological features against test error. As average vertex degree increase from 5 to 20, the test error decrease dramatically from 15% to 4% and converge after that.
- Even if half of the input layer neurons are not in use, the error rate is still reasonable! This is particularly important in practical.
- Networks with 20 vertex average degree performs similar with 60 vertex average degree, indicating that we can reduce most of the edges.

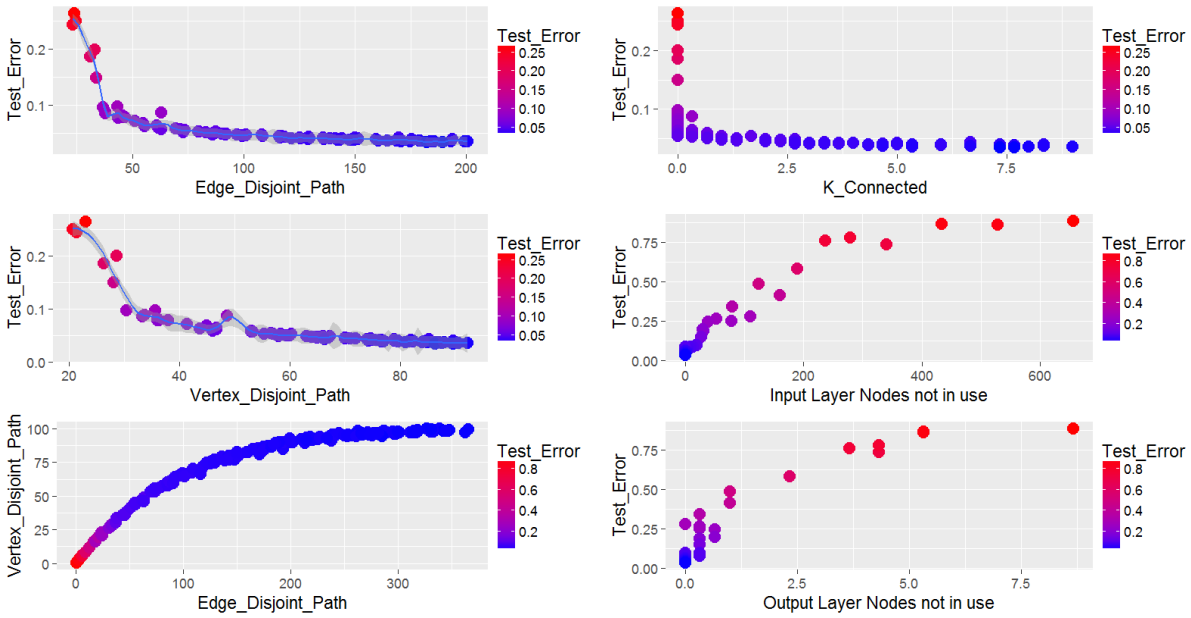

Figure 3: The result of randomly turn off connection between input layer and first hidden layer

4.3 Randomly turn off connection between first and second hidden layer

As it show in figure4, when we randomly turn off connection between first and second hidden layer, we have the observation below:

- In the first graph, the test error decreases rapidly as the variable, namely vertex average degree, gradually increases from a small value and the test error starts to drift upward when the variable reaches a critical point (around 160). Similarly, in the second graph, the test error reaches its minimum when K-Connectivity is approximately 10.
- As a result, we can conclude that the test error does not necessarily decrease as the two variables increase. A large vertex average degree or k connectivity imply more neuron connections of the network. While intuitively speaking, more connections of the network would result in less test error, the two graphs indicate that test error in fact increases as the two variables reach a critical point.

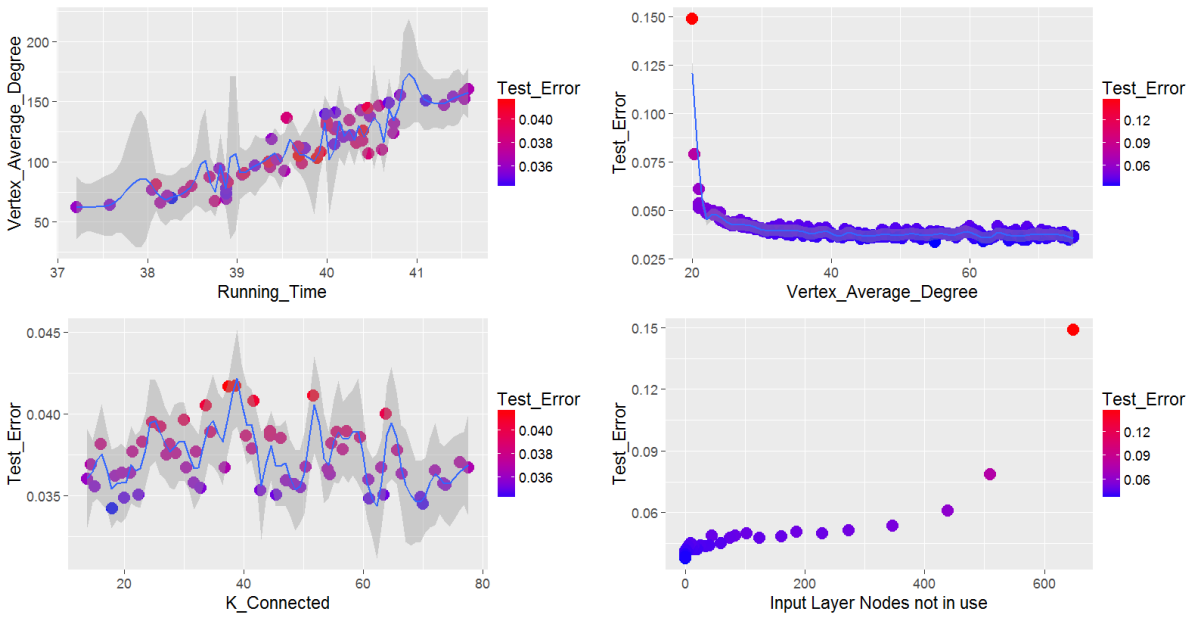

Figure 4: The result of randomly turn off connection between first and second hidden layer

4.4 Randomly Turn off Edges between Hidden Layer 2 and Output Layer

As it show in figure5, when we randomly turn off connection between first and second hidden layer, we have the observation below:

- Test error is negatively correlated with number of Vertex/Edge Disjoint Path; yet the relationships are nonlinear. Networks with 30 vertex disjoint paths functions similar with those with 100 paths.
- K-connectivity does not directly relate to the quality of the network. Yet networks with 1-connectivity or 2-connectivity suffers from huge test error, while networks with a little higher connectivity performs extremely well.
- Running time is barely related to number of paths, suggesting sparse networks spend similar training time as dense networks do.

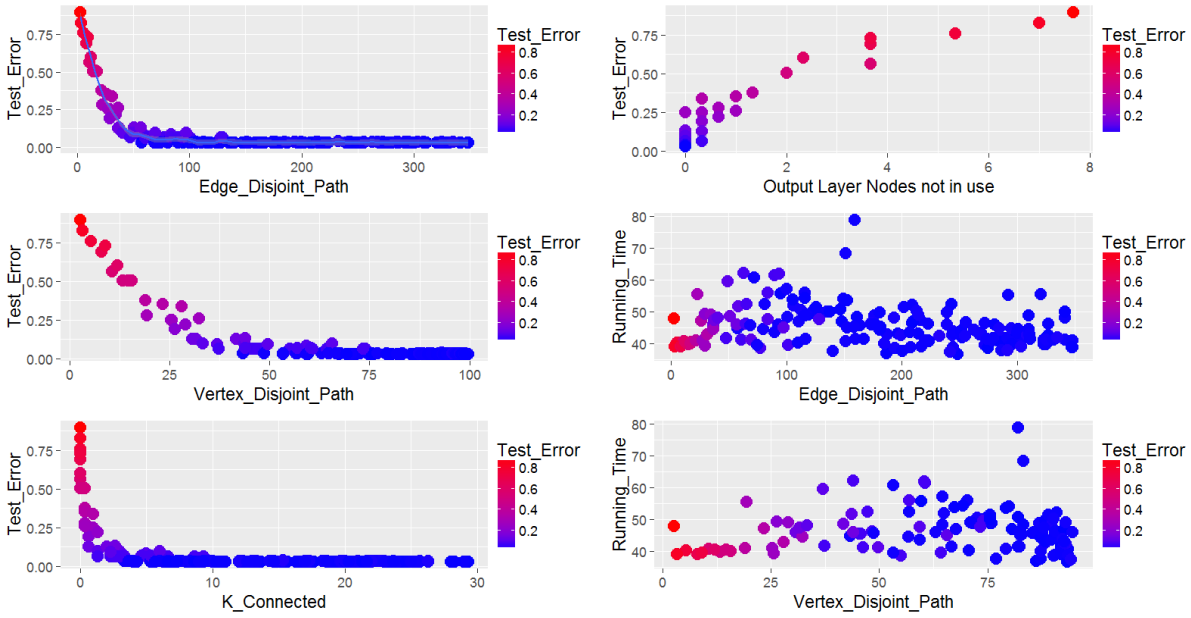

Figure 5: The result of randomly turn off connection between hidden layer II and output Layer

Conclusion: An idea network in this case would be with 50 vertex disjoint path, and very low connectivity.

5 Empirical Result

5.1 Regression

It is time to compare the impact of different topological features to neural network performance. We try different specification, as there are some multi-collinearity between numbers of vertex/edge disjoint path.

We choose 'test error' to be the dependent variables. It is also viable to use 'running time', yet it may carry some noise from computer operations.

Here is the regression function:

$$\begin{aligned} TestError = \beta_0 &+ \beta_1 Path_{edge} + \beta_2 Dropout_{inputneuron} \\ &+ \beta_3 Dropout_{outputneuron} + \beta_4 Path_{vertex} \\ &+ \beta_5 Degree_{avg} + \beta_6 K_{connectivity} \end{aligned} \tag{1}$$

We present the regression result in figure6.

	Dependent variable:		
	test_error		
	(1)	(2)	(3)
edge_disjoint_path	-0.0002	-0.001**	
	(0.0005)	(0.0005)	
Input_Layer_ptg_dropout	0.001***	0.001***	0.001***
	(0.0002)	(0.0002)	(0.0002)
Output_Layer_ptg_dropout	0.038***	0.035**	0.039***
	(0.014)	(0.015)	(0.014)
vertex_disjoint_path	-0.003***		-0.003***
	(0.001)		(0.001)
vertex_average_degree	0.003	0.022***	0.003
	(0.005)	(0.003)	(0.004)
K_connected	0.002	-0.004	0.001
	(0.003)	(0.003)	(0.002)
Constant	0.223***	0.187***	0.224***
	(0.015)	(0.014)	(0.015)
Observations	175	175	175
R2	0.942	0.934	0.942
Adjusted R2	0.940	0.932	0.940
Residual Std. Error	0.038 (df = 168)	0.041 (df = 169)	0.038 (df = 169)
F Statistic	454.382*** (df = 6; 168)	475.840*** (df = 5; 169)	547.792*** (df = 5; 169)
Note:			*p<0.1; **p<0.05; ***p<0.01

Figure 6: The result of regression

5.2 Observation

- **Result 1:** Number of Vertex-Disjoint path reduce the test error significantly, given other factors constant.
- **Result 2:** Number of Edge-Disjoint path reduce the test error significantly, given other factors constant; the impact is slightly smaller, compared to vertex-disjoint one.
- **Result 3:** Number of isolated input layer neurons is positively significant related to the test error, given other factors constant.

- **Result 4:** Number of isolated output layer neurons is positively significant related to the test error, given other factors constant.
- **Result 5:** The k-connectivity of the network is not significantly related to test error, after controlling other factors, suggesting the relative weakness of this topological feature.

6 Conclusion

The goal of this project is to understand the relation between various parameters describing the graph of connections between neurons and the learning rate and quality. The motivation is to potentially improve the efficiency of neural network algorithms while maintaining very low predicting error.

In this empirical research, we see a significant nonlinear relationship between topological features and performance measure. This has important practical implication, especially when we want to use a neural network with multiple layers and many neurons.

We can keep half of the neurons in input layer and still keep the error rate reasonable(at least in the MINST data set). On the other hand, to reduce training time, we can remove most of the edges between input layer and first hidden layer. In fact, from our data, we can see that 95% of these edges can be removed.

References

[1] https://github.com/bdol "Artificial Neural Networks: Matrix Form"

[2] Wang, Zhenni *,A procedure for determining the topology of multilayer feedforward neural networks* , Neural networks, 1994.