Assignment: A.2 Classifier NNs Name: Alisher Siddikov Class: MSDS 458
Date: 02/16/2020

Introduction

This assignment is similar to the previous assignment. It provides further explanations of hidden nodes. We are asked to explore how the hidden layers in a simple single-hidden layer network learn to represent features within the input data. Input data is an array form of 26 capital letters of the English alphabet. Each letter has 9x9 pixels and this is transformed into a vector of length 81 giving one input node for each pixel in the letter. You can see 9x9 pixeled capital letters in Figure 1. Therefore, by default, we have 81 input nodes, 6 hidden nodes, and 8 output nodes. During the analysis, we will see that hidden nodes cluster letters by identifying learned features such as finding mutual patterns: diagonal, vertical, horizontal, rounded, and straight lines. So, our NN needs to learn that two (or more) different kinds of patterns will lead to the same classification; some elements will be in common (longer horizontal/vertical lines), but there will be elements that are different (square corners vs diagonal/rounded corners). We will identify which hidden nodes responded to which classes – and identify which hidden nodes responded to classes that had similar letter shapes. We will train our model to put our 26 pixelated letters into the correct class assignments. For example, letters 'D', 'O' and 'Q' can be grouped to 'O' like shape because they all seem to share similar pixels.

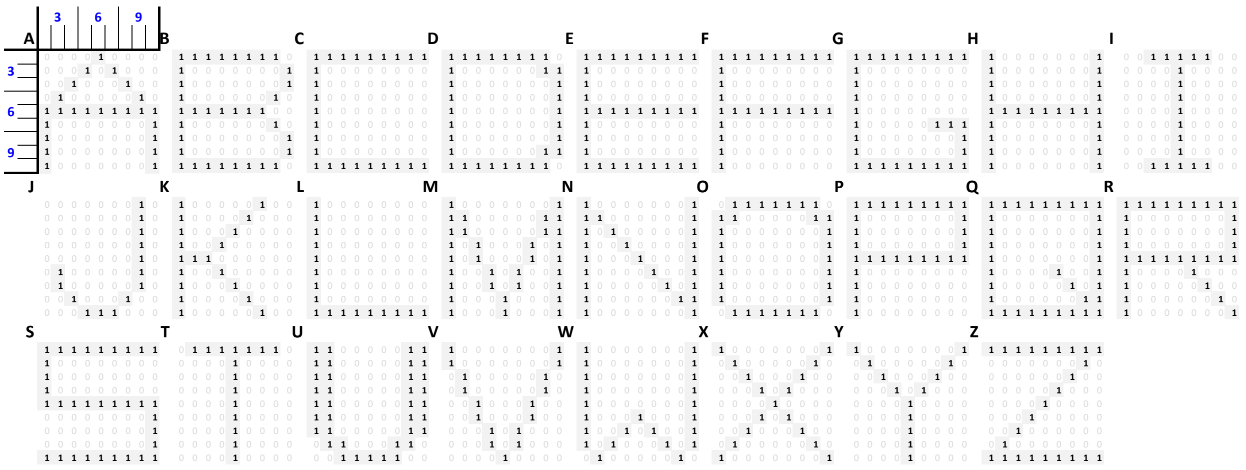


Figure 1

We changed the following parameters during our analysis:

- numHiddenNodes: I started with a small number of hidden nodes, and then increased them. The change in the number of nodes had an impact on the activations.

- **numOutputNodes**: During our experiment, I started with 8 classes (default) and I was able to change the class assignments to improve our results.
- **noise:** introduced noise to reduce overfitting and being able to better generalize and faster learning. 10% (0.1) of our pixels were randomly flipped for each character retrieved during the training step. For example, according to figure 1, we can see that pixelated O and Q do not have the same walls. So, noise should take care of this problem.
- **eta:** the learning rate helped us to determine how big steps we had to take at each iteration while moving toward a minimum of a loss function.

Analysis with Default Parameters

If we do not use backpropagation to minimize the cost function, we will have 19.23% accuracy rate for the "before training." However, in our analysis, we are interested in backpropagation such as "after training" results and interpretations. When I ran the model with default parameters (numHiddenNodes = 6, numOutputNodes = 8, trainingNoise = 0.00, and eta or learning rate = 0.5, SSE threshold epsilon = 0.01), we were able to classify about 85% of the 26 capital letters to the desired 8 classes. It took 2,290 iterations (out of max iteration 5,000 iterations), which has slower convergence. We can achieve lower iterations by modifying the default learning rate (eta = 1.0) in our future analysis. Let's analyze results; we can see that the following capital letter A, C, H, and J misclassified. I reordered those letters by their class.

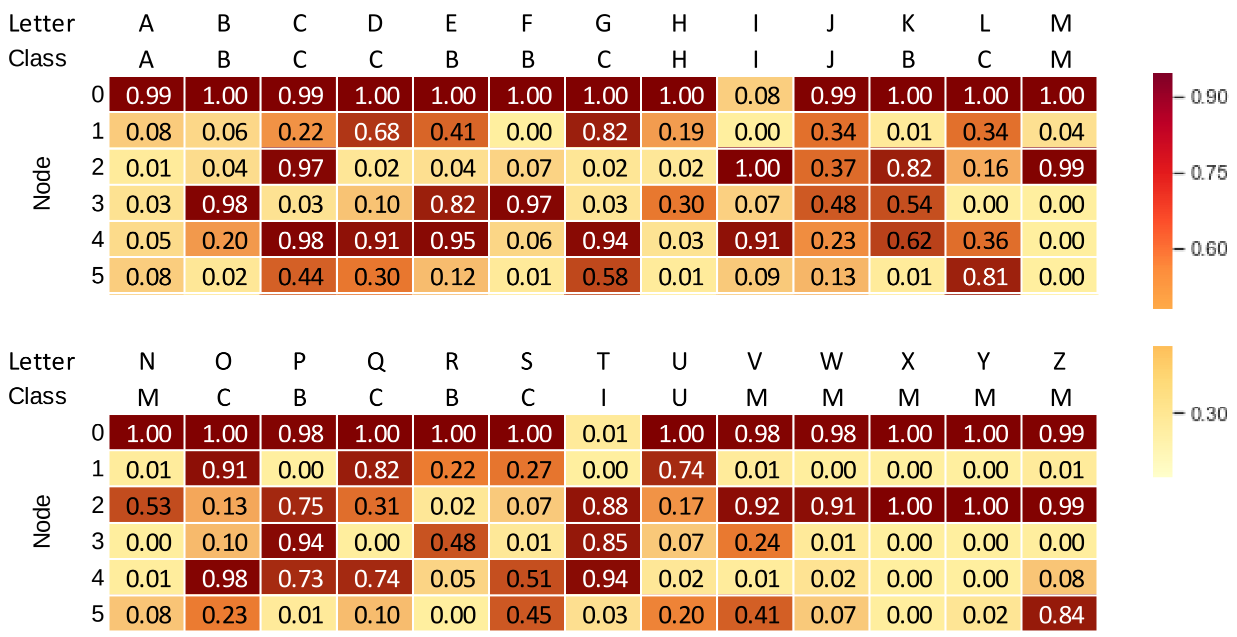
Letter	A	B	E	F	K	P	R	C	D	G	L	O	Q	S	H	I	T	J	M	N	V	W	X	Y	Z	U
Desired	0	1	1	1	1	1	1	2	2	2	2	2	2	2	3	4	4	5	6	6	6	6	6	6	6	7
Selected	6	1	1	1	1	1	1	6	2	2	2	2	2	2	1	4	4	1	6	6	6	6	6	6	6	7

Percent Accuracy 84.62

Figure 2

We can see that the desired class should be rearranged and reclassified. For example, according to figure 1, the pixelated letter A is closer to the pixelated letter H, and H is close to F. The pixelated letter J is closer to the pixelated letter U. We can reclassify them next time when we run the model. Now we can focus on interpreting the hidden nodes of default parameters. We will expand on hidden node interpretation when we re-run the model with the right parameters.

Assignment: A.2 Classifier NNs Name: Alisher Siddikov Class: MSDS 458
Date: 02/16/2020



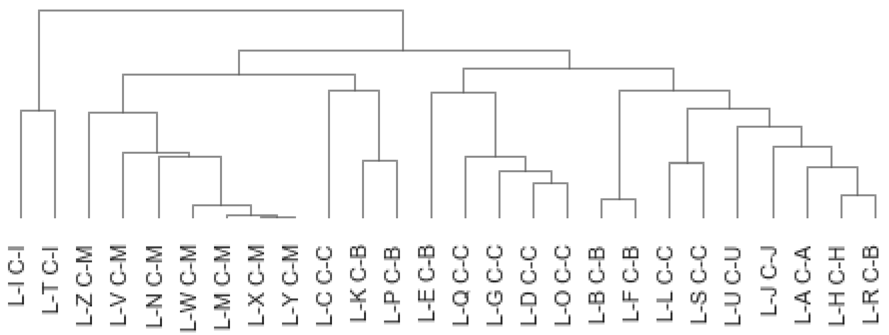
Letter	A	B	C	D	E	F	G	H	I	J	K	L	M
Class	A	B	C	C	B	B	C	H	I	J	B	C	M
0	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	0.08	0.99	1.00	1.00	1.00
1	0.08	0.06	0.22	0.68	0.41	0.00	0.82	0.19	0.00	0.34	0.01	0.34	0.04
2	0.01	0.04	0.97	0.02	0.04	0.07	0.02	0.02	1.00	0.37	0.82	0.16	0.99
3	0.03	0.98	0.03	0.10	0.82	0.97	0.03	0.30	0.07	0.48	0.54	0.00	0.00
4	0.05	0.20	0.98	0.91	0.95	0.06	0.94	0.03	0.91	0.23	0.62	0.36	0.00
5	0.08	0.02	0.44	0.30	0.12	0.01	0.58	0.01	0.09	0.13	0.01	0.81	0.00

Letter	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Class	M	C	B	C	B	C	I	U	M	M	M	M	M
0	1.00	1.00	0.98	1.00	1.00	1.00	0.01	1.00	0.98	0.98	1.00	1.00	0.99
1	0.01	0.91	0.00	0.82	0.22	0.27	0.00	0.74	0.01	0.00	0.00	0.00	0.01
2	0.53	0.13	0.75	0.31	0.02	0.07	0.88	0.17	0.92	0.91	1.00	1.00	0.99
3	0.00	0.10	0.94	0.00	0.48	0.01	0.85	0.07	0.24	0.01	0.00	0.00	0.00
4	0.01	0.98	0.73	0.74	0.05	0.51	0.94	0.02	0.01	0.02	0.00	0.00	0.08
5	0.08	0.23	0.01	0.10	0.00	0.45	0.03	0.20	0.41	0.07	0.00	0.02	0.84

Figure 3

- **H0:** it reacts to all letters except I and T (negative weight) which fall to I class
- **H1:** it reacts to mostly rounded and squared letters which are part of C class such as O, Q, G, U, and D
- **H2:** it reacts to white space of Y, I, X, Z, M, C, V, W, T, K, and P – unclear features
- **H3:** it reacts to mostly B class which has middle line such as B, F, P, E, and K
- **H4:** it reacts to square and vertical line of C, O, E, G, T, D, I, Q, and P – somewhat unclear features
- **H5:** it reacts to letters Z, L, G or mostly C class – somewhat unclear features

We can see this visually in the dendrogram.



Please note that too few nodes and the model will not be able to pick up enough features to distinguish the letters. Too many, and the nodes will be tied to specific features, and the model will not be able to generalize.

I reordered the letters by their similarities and reclassified them.

Letters	**A**	**H**	**F**	**E**	**S**	**Z**	**B**	**P**	**R**	**K**	**Q**	**O**	**D**	**L**	**C**	**G**	**J**	**U**	**V**	**M**	**N**	**W**	**X**	**Y**	**T**	**I**
Classes	A	A	B	B	B	B	B	B	B	B	C	C	C	C	C	C	U	U	M	M	M	M	M	M	I	I
Desired	0	**0**	1	1	**1**	**1**	1	1	1	1	2	2	2	2	2	2	**3**	**3**	4	4	4	4	4	4	5	5

Figure 4

I reduced output nodes from 8 to 6 and reclassified some 9x9 pixelized capital letters such as ‘A’ and ‘H’ as A-class, ‘S’ and ‘Z’ as B-class, and ‘J’ and ‘U’ as U-class. However, I hold the rest of the parameters at the same default level. After reclassifying the letters, the accuracy of the model went up from 84.62% to 96.15% (or 25/26; ‘H’ letter misclassified) and the iterations reduced from 2,290 to 1,463.

Analysis After Tweaking Parameters

When I introduced the 10% noise during training the model, the accuracy score went up to 100%, and iterations slightly increased to 1,616. Therefore, 10% of our pixels were randomly flipped for each character retrieved during the training and the hidden nodes had less tendency to assign weights to the same pixels; now ‘H’ and ‘A’ letters classified as A class. Here is the hidden node visual after reclassifying letters and introducing noise.

```
                0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Letter          A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
Desired         0  1  2  2  1  1  2  0  5  3  1  2  4  4  2  1  2  1  1  5  3  4  4  4  4  1
Selected        0  1  2  2  1  1  2  0  5  3  1  2  4  4  2  1  2  1  1  5  3  4  4  4  4  1
Percent Accuracy 100.00
```

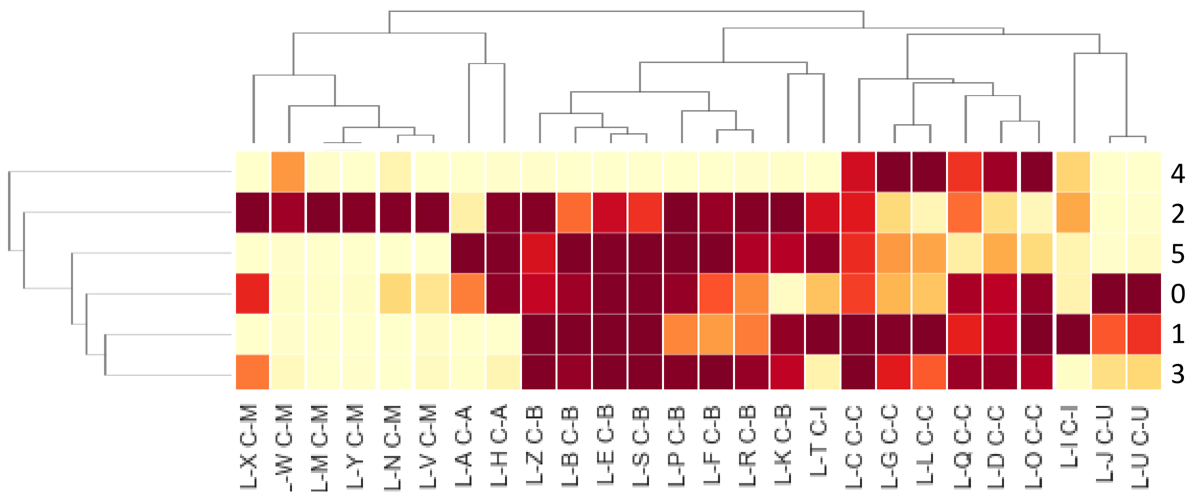


Figure 5

- **H4:** it reacts to C class only. The hidden node activation is strongest if letter has vertical line at LHS and horizontal line at the bottom such as L and G. The activation is stronger if letter has rounded corners and covered with borders (no open space) such as O and D. The activation is somewhat stronger if the letter does not have rounded corners and has vertical line at LHS and horizontal line at the top and bottom such as C and Q.

Letter	G	L	O	D	C	Q
Class_L	C	C	C	C	C	C
Hidden Node 4	1.00	1.00	0.99	0.93	0.81	0.69

- **H2:** it reacts to mostly M and B classes. The hidden node activation is stronger if the letter has vertical line at LHS and horizontal line at the middle such as P, K, R, H, F, and E. The activation is also stronger if letter has diagonal lines such as M, K, V, X, N, Y, and W.

Letter	M	P	K	V	X	N	Y	R	Z	H	F	W	E
Class_L	M	B	B	M	M	M	M	B	B	A	B	M	B
Hidden Node 2	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.95	0.94	0.83

- **H5:** it reacts B and A classes primarily. This hidden node groups the letters in a nice way. The hidden node activation is strongest if the letter is B or A classes, somewhat stronger if it is C class, somewhat weaker if it is U class, and weakest if it is M class. If we take B or A (the strongest activated) classes, we can see the hidden node activations are around semi-vertical line in both edges and horizontal line in a middle such as S, A, and P letters; or vertical line at LHS and horizontal line at the middle such as F, B, H, and E. Please see figure 1 for the construction of letters.

Letter	F	S	A	P	B	H	E	T	R	K	Z	C	G
Class_L	B	B	A	B	B	A	B	I	B	B	B	C	C
Hidden node 5	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.96	0.90	0.89	0.79	0.71	0.46

Letter	L	D	O	Q	I	U	J	W	X	V	N	Y	M
Class_L	C	C	C	C	I	U	U	M	M	M	M	M	M
Hidden node 5	0.41	0.40	0.23	0.11	0.08	0.03	0.01	0.01	0.00	0.00	0.00	0.00	0.00

- **H0:** it reacts oval, square, and B class; it does not react M and I classes. The hidden node activation is strongest if letter has rounded corners at the bottom such as U and J, or somewhat stronger if it has rounded or square corners such as O, Q, and D. It is also stronger if it has middle horizontal line such as E, S, H, P, and B.

Letter	J	U	E	S	H	P	O	B	Q	D
Class_L	U	U	B	B	A	B	C	B	C	C
Hidden node 0	1.00	1.00	0.99	0.99	0.96	0.96	0.95	0.94	0.91	0.87

- **H1:** it reacts B, I, and C classes. The hidden node activation is strongest if letter has one vertical line and often has horizontal lines at the top, bottom, or in the middle such as T, I, L, C, G, E, and B, or it has horizontal lines at both top and bottom such as S, Z, and O.

Letter	E	O	T	I	B	S	Z	C	L	G
Class_L	B	C	I	I	B	B	B	C	C	C
Hidden node 1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99

- **H3:** it is somewhat similar to the previous node, H1.

Letter	E	C	F	S	Z	P	B	R	D	Q	O
Class_L	B	C	B	B	B	B	B	B	C	C	C
Hidden node 3	1.00	1.00	1.00	0.99	0.99	0.99	0.96	0.95	0.95	0.94	0.90

By looking at figure 5, hidden nodes are somehow duplicating each other. There might be too many hidden nodes. Here is the table on how the accuracy and iteration behave when we reduce or increase the hidden nodes. We can see that with four hidden nodes and an increased learning rate, we can achieve 100% accuracy and less iteration. If we had time, we could have rerun our model and re-interpreted the hidden nodes.

Changes	**Learning Rate**	**Accuracy**	**Iteration**
default parameters (output = 8)	0.5	84.62	2290
reclassifying letters (output = 6)	0.5	96.15	1463
training noise change from 0.0 to 0.1	0.5	100	1616
Hidden nodes = 2	0.5	53.85	5000
Hidden nodes = 3	0.5	80.77	3889
Hidden nodes = 4	0.5	100	2531
Hidden nodes = 4	1.0	100	955
Hidden nodes = 5	0.5	100	2322
Hidden nodes = 6	0.5	100	1616
Hidden nodes = 7	0.5	88.46	1191
Hidden nodes = 8	0.5	100	1100
Hidden nodes = 9	0.5	88.46	1200

Figure 6

Conclusion

Figure 6 helps us to summarize the results. I started with a small number of hidden nodes, and then increased them and saw what happened with the node activations. I started analysis with two hidden nodes along with six output classes and 10% noise, my model did not converge (reaches to max iteration 5,000), and accuracy dropped to 53.85%. Thus, not enough hidden nodes were not able to pick up enough features to cluster the letters. We did not have enough hidden nodes. With four hidden nodes, we reached the highest accuracy; so, it was able to cluster 9x9 pixelated capital letters correctly at a learning rate 0.5. Furthermore, when I increased the learning rate from 0.5 to 1.0, it took 955 iterations only. So, it took big steps at each iteration while moving toward a minimum of a loss function. This is our ideal hidden node numbers and parameters. After this, as I increased the hidden nodes, my model overfits and training accuracy started dropping. By looking at figure 6, with too few hidden nodes, the model was not able to pick up enough features to distinguish the pixilated capital letters. Too many, and the hidden nodes were tied to specific features, and the model was not able to generalize.