Pattern Recognition

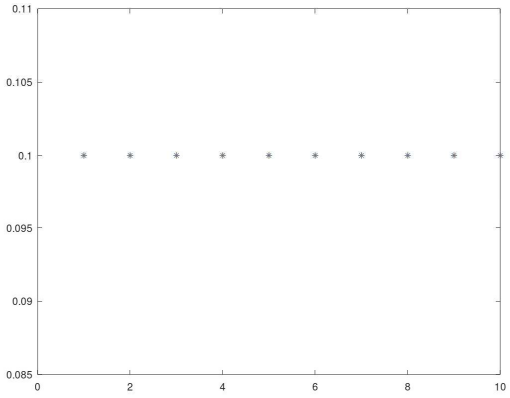
Lab 5-6: Classification with Neural Networks

Kevin Mato-K5529

January 2020

1. Reference Solution

Before the first implementation of the neural network I checked if the data in the training test and the test set had different distribution to understand since the beginning what is the reason of a possible overfitting in my net. The distribution of the classes inside training set and test set are of 10% for each class (T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot). Overfitting could be due to the fact that we use fully connected neural nets.



The reference solution had one hidden layer (100 neurons) trained incrementally with a constant learning rate of 0.001 for 50 epochs.

My first solution has no improvements yet, I chose as a first activation function to begin my experimentation a bipolar which has range of values (-1,1).

- Number of neurons= 100
- Learning rate = 0.001
- Number of epochs 50

The number of hidden layers in this first case is one, and the learning rate is constant.

Training SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	90.38%	9.62%	0.0%
	Experiment 1		
	OK	ERROR	REJECTION
Classification Coefficients	90.51%	9.48%	0.0%

Testing SET	
	Refence

	OK	ERROR	REJECTION
Classification Coefficients	87.42%	12.58%	0.0%
	Experiment 1		
	OK	ERROR	REJECTION
Classification Coefficients	87.76%	12.24%	0.0%

The first experiment has given better results with a very small margin, this is probably due to the initial weight assignment and these first results are not relevant. My implementation might be slow, in average it takes 82.2 seconds.

A **second variant** of my first experiment:

- Number of neurons= 130
- Learning rate = 0.001
- Number of epochs 50

Training SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	90.38%	9.62%	0.0%
	Experiment 2		
	OK	ERROR	REJECTION
Classification Coefficients	90.28%	9.71%	0.0%

Testing SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	87.42%	12.58%	0.0%
	Experiment 2		
	OK	ERROR	REJECTION
Classification Coefficients	87.73%	12.23%	0.0%

Average epoch takes 70 seconds, strangely faster than before. Just an imperceptible improvement in the test error.

Third Variant: I diminished the number of neurons to see how much making the model simpler would have lowered my training error, to have a reference, maybe if it would have made my model overfit more. The learning rate this time is 0.002, still reasonable for our number of epochs but I wanted to see how much faster it would have reached the final error rate.

- Number of neurons= 90
- Learning rate = 0.002
- Number of epochs 50

We see that in the last two iterations there's an evident overfitting. We would need to stop at iteration 48 to have a safe margin. I will take as reference iteration 48. Average epoch time is 80.26 seconds, slightly faster. The improvement compared to the reference function is still almost unnoticeable.

48.000000 78.589615 0.080850 0.120500

49.000000 79.507763 0.080300 0.119900

50.000000 79.114349 0.079733 0.120000

Training SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	90.38%	9.62%	0.0%
	Experiment 3		
	OK	ERROR	REJECTION
Classification Coefficients	91.91%	8.08%	0.0%

Testing SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	87.42%	12.58%	0.0%
	Experiment 3		
	OK	ERROR	REJECTION
Classification Coefficients	87.95%	12.05%	0.0%

Confusion matrixes:

Variation 1

852	0	18	38	8	2	69	0	13	0
4	960	2	25	4	0	4	0	1	0
20	1	804	13	102	0	57	0	3	0
24	11	12	900	33	0	17	0	3	0
0	0	89	38	823	1	42	0	7	0
0	0	0	1	0	935	0	42	2	20
140	0	97	40	81	0	623	0	19	0
0	0	0	0	0	22	0	951	0	27
1	1	5	6	5	2	6	4	970	0
0	0	0	0	0	5	0	36	1	958

Variation 2

845	1	20	39	4	1	75	0	15	0
4	960	2	25	4	0	4	0	1	0
16	0	803	14	108	0	52	0	7	0
25	11	11	897	33	0	18	0	5	0
0	0	88	41	822	0	44	0	5	0
0	0	0	2	0	936	0	43	2	17
150	1	85	38	86	0	623	0	17	0
0	0	0	0	0	19	0	953	0	28
1	1	2	4	4	3	6	4	975	0
0	0	0	0	0	7	0	36	1	956

Variation 3

849	1	21	31	3	2	80	0	13	0
4	966	3	21	3	0	2	0	1	0
17	0	809	15	102	0	54	0	3	0
23	6	12	898	35	0	23	0	3	0
0	2	86	42	815	1	49	0	5	0
0	0	0	2	0	934	0	37	2	25
144	1	83	33	81	0	639	0	19	0
0	0	0	0	0	15	0	962	0	23
3	1	3	5	5	1	6	5	971	0
0	0	0	0	0	3	0	39	1	957

Comparison:

The last column is omitted since the possibility rejection was ignored from the beginning.

It's evident in all the three cases a misclassification of the couples 1-7, 3-5, 4-5 (not too relevant but still present), 5-3,

7-1 is one of the biggest misclassifications. Class 7 seems to be misclassified for 3,4,5 quite often. The rest of misclassifications are not relevant. The best classifications are for class 2, 8, 9 and 10.

2. Improvements

The improvements can be found in the original files of ann_trainining and backprop, by commenting and de-commenting some parts of the code. For the two layer network there are 3 more files (with …2layer.m names). Where the numbers would have been useful I've put a plot.

2.1 Input Normalization (Bipolar a.f.):

Input normalization can give us a faster learning since, by combing input between -1, and 1 with weights in this range also we exploit the region of the sigmoid function where it is almost linearly incrementing. For this reason all the next experiments will be proceeded by the input normalization.

For comparison purposes I'm using:

- Number of neurons= 100
- Learning rate = 0.001
- Number of epochs 50

The epoch time in this case it actually decreased to 39.88 seconds.

Training SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	90.38%	9.62%	0.0%
	Experiment 2.1		
	OK	ERROR	REJECTION
Classification Coefficients	95.41%	4.58%	0.0%

Testing SET	
	Refence

	OK	ERROR	REJECTION
Classification Coefficients	87.42%	12.58%	0.0%
	Experiment 2.1		
	OK	ERROR	REJECTION
Classification Coefficients	88.81%	11.19%	0.0%

Confusion Matrix:

873	1	19	21	6	3	68	1	8	0
5	965	2	19	3	0	5	0	1	0
27	1	802	11	98	2	57	1	1	0
29	7	14	897	27	1	21	0	4	0
2	1	69	35	828	0	58	1	5	1
0	0	1	1	0	941	0	29	5	23
142	1	70	22	70	0	682	0	12	1
0	0	0	0	0	13	0	965	0	22
2	0	2	5	7	4	11	2	967	0
0	0	1	0	0	8	0	30	0	961

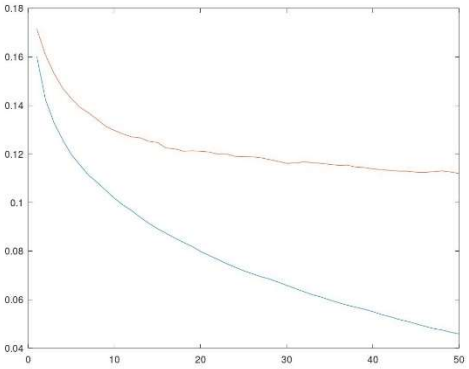
Comparison:

The results in terms of error ratios have improved compared to the first three experiments.

The misclassifications have improved but there are still couples 1-7, 3-5, 7-1 (plus if we want 7-3,7-5, 5-3), which keep misclassifying one for the other but in lower measure compared to before.

Thoughts:

There's a strong difference between the training error and the test error, it's an evidence of predictable overfitting but in 50 epochs the test error just decreased monotonically and the best value was found at the 50th.



EXP 2.1: INPUT NORMALIZATION

2.2 Input Normalization and Shuffling (Bipolar a.f.):

Besides the already tested input normalization I added in the backpropagation algorithm a first shuffling phase.

In a first check the samples seem to be in a random order but they might follow some pattern still, so I tried to add more randomness to the sets.

For comparison purposes I'm using:

- Number of neurons= 100
- Learning rate = 0.001
- Number of epochs 50

The results for this experiment aren't exceptional. There is a clear overfitting after epoch number 43.

37.000000 91.784906 0.054633 0.113200

38.000000 90.965967 0.052600 0.113400

39.000000 89.073392 0.052033 0.112800

40.000000 91.668049 0.051283 0.111900

41.000000 86.749740 0.050583 0.113700

42.000000 89.235110 0.051883 0.116100

43.000000 90.366582 0.049833 0.114000

44.000000 58.428497 0.049467 0.115800

45.000000 58.371749 0.047233 0.112600

46.000000 56.867730 0.047500 0.114200

47.000000 56.585483 0.044483 0.114100

48.000000 60.845803 0.045000 0.111700

49.000000 42.041867 0.042133 0.111900

50.000000 41.694862 0.042833 0.111400

Early stopping in this case is necessary to avoid greater overfitting, but I kept the values for the 50th epoch since it has the same value for the test error of the 43th epoch, and the smallest error in the training set. I kept the confusion matrix at this iteration also because there is a lot of oscillations during the training, due to the randomness of shuffling. So no better result is actually found with this method. The duration of an epoch hasn't improved, it's just slower, with 80.80 seconds.

Training SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	90.38%	9.62%	0.0%
	Experiment 2.2		
	OK	ERROR	REJECTION
Classification Coefficients	95.72%	4.28%	0.0%

Testing SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	87.42%	12.58%	0.0%
	Experiment 2.2		
	OK	ERROR	REJECTION
Classification Coefficients	88.86%	11.14%	0.0%

Confusion Matrix:

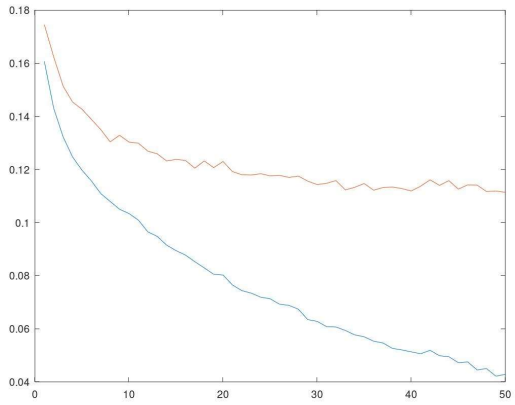
864	1	15	25	5	2	79	1	7	1
5	963	2	19	4	0	6	0	1	0
22	1	809	15	90	1	58	1	3	0
24	2	11	897	41	1	20	0	4	0
0	1	75	23	842	1	51	0	7	0
0	0	1	1	0	948	0	27	3	20
128	0	81	32	76	0	671	0	10	2
0	0	0	0	0	12	0	963	0	25
3	0	2	6	5	3	7	5	968	1
0	0	1	0	0	9	0	29	0	961

Comparison:

The results in terms of error ratios have improved again.

The misclassifications have improved but there are still couples 1-7, 3-5, 7-1, 7-3, which keep misclassifying one for the other but in lower measure compared to before. This time we add to the misclassifications couples 5-3 which has increased. This is the best result so far in terms of misclassification. Class 2, 8, 9, 10 remain the best.

Thoughts:



2.2 NORMALIZATION+SHUFFLING

2.3 Normalization and usage of Unipolar activation function:

I tried using a different activation function, to compare the performances of the bipolar to the unipolar, even if the bipolar has notoriously better results.

For comparison purposes I'm using:

- Number of neurons= 100
- Learning rate = 0.001
- Number of epochs 50

Training SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	90.38%	9.62%	0.0%
	Experiment 2.3		
	OK	ERROR	REJECTION
Classification Coefficients	95.72%	4.28%	0.0%

Testing SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	87.42%	12.58%	0.0%
	Experiment 2.3		
	OK	ERROR	REJECTION
Classification Coefficients	88.86%	11.14%	0.0%

The average epoch time is quite slow: 84.45 seconds, even if normalization is used.

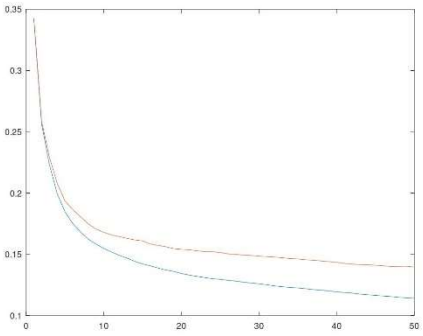
Confusion Matrix:

828	0	8	71	5	5	71	0	12	0
5	953	1	31	5	0	3	1	1	0
17	2	735	24	144	6	67	1	3	1
18	6	4	924	23	2	19	1	3	0
0	1	77	52	812	3	50	1	4	0
0	0	0	2	0	929	0	46	2	21
148	2	77	79	89	6	584	2	12	1
0	0	0	0	0	26	0	953	0	21
3	1	7	10	5	10	7	6	948	3
0	0	0	0	0	14	1	47	0	938

Comparison:

We have practically gone back to the initial situation where no improvements were applied, the unipolar activation function has decreased the classification power of the net. This result is important because it explains how fundamental is using the right function. All the classes compared to the previous experiments have shown quality deterioration. Even classes like class ten. Class 7 is the most affected.

The only good thing we could say about the unipolar function is that is has lower the difference between train error and test error, which could be symptom of less overfitting.



2.3 NORMALIZATION+UNIPOLAR

2.4 Bipolar a.f., input normalization, added momentum

The values from the training show that the learning finishes at the 28th epoch. I performed early stopping.

For comparison purposes I'm using:

- Number of neurons= 100
- Learning rate = 0.001
- Number of epochs 50

The average epoch time is 59 seconds.

27.000000	**59.935321**	**0.053067**	**0.112900**
28.000000	**55.137552**	**0.051917**	**0.112800**
29.000000	**57.153837**	**0.050833**	**0.113100**
30.000000	**57.525950**	**0.049683**	**0.113700**
31.000000	**55.224199**	**0.048633**	**0.113900**
32.000000	**58.379485**	**0.047850**	**0.114000**
33.000000	**60.454119**	**0.046833**	**0.113600**

Training SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	90.38%	9.62%	0.0%
	Experiment 2.4		
	OK	ERROR	REJECTION
Classification Coefficients	94.81%	5.19%	0.0%

Testing SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	87.42%	12.58%	0.0%
	Experiment 2.4		

	OK	ERROR	REJECTION
Classification Coefficients	88.72%	11.28%	0.0%

Confusion Matrix:

872	1	18	19	8	1	68	1	12	0
7	962	3	18	3	0	6	0	1	0
21	0	801	10	106	1	59	0	2	0
29	6	13	905	23	1	18	0	4	1
1	1	72	34	832	0	56	0	3	1
0	0	1	1	0	943	0	30	2	23
150	1	71	28	73	0	666	0	10	1
0	0	0	0	0	13	0	967	0	20
4	0	3	5	4	3	10	4	967	0
0	0	0	1	0	7	1	34	0	957

Comparison:

Momentum makes the neural net converge faster to the optimal solution. Class 7 is still affected by very bad results.

2.4 Bipolar a.f., input normalization, added momentum and decreasing learning rate

In this case I decreased my learning rate linearly of 0.0005. It's initial value is 0.003, found by increasing the value of 0.001 from the initial value of 0.001.

- Number of neurons= 100
- Learning rate = 0.003
- Number of epochs 50

Early stopping must be performed at epoch 19. Convergence to the best solution comes even earlier.

17.000000 62.877577 0.057333 0.118400

learningRate = 0.0021500

18.000000 60.778699 0.055400 0.117300

learningRate = 0.0021000

19.000000 55.614649 0.053683 0.116800

learningRate = 0.0020500

20.000000 57.279981 0.052817 0.117100

learningRate = 0.0020000

21.000000 54.774409 0.052267 0.117300

Confusion Matrix:

872	2	10	27	7	1	70	1	10	0
3	968	1	19	3	0	4	0	2	0
28	0	769	12	109	1	77	2	2	0

24	4	12	904	30	2	18	2	4	0
3	2	74	29	829	2	58	0	3	0
1	0	1	1	0	936	0	33	2	26
148	7	64	35	63	0	669	0	12	2
0	0	0	0	0	11	0	973	0	16
2	0	6	6	3	4	9	3	967	0
0	0	0	0	0	6	2	47	0	945

2.5 Two layers

This time I added a second layer of hidden neurons.

For comparison purposes I'm using:

- Number of neurons= 100
- Learning rate = 0.001
- Number of epochs 50

The number of neurons showed if for each hidden layer. The act. Fun. Is still bipolar and input normalization is performed.

46.00000 65.32185 0.14300 0.16150

47.00000 72.73408 0.14283 0.16130

48.00000 66.19879 0.14285 0.16120

49.00000 65.91356 0.14277 0.16120

50.00000 63.00151 0.14265 0.16140

Training SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	90.38%	9.62%	0.0%
	Experiment 2.5		
	OK	ERROR	REJECTION
Classification Coefficients	85.74%	14.26%	0.0%

Testing SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	87.42%	12.58%	0.0%
	Experiment 2.5		
	OK	ERROR	REJECTION
Classification Coefficients	83.86%	16.14%	0.0%

Confusion Matrix:

825	0	17	80	5	2	52	0	19	0
4	943	5	37	5	0	3	1	2	0
23	1	755	15	151	2	46	0	7	0

26	7	20	887	33	1	20	0	5	1
1	0	95	47	794	1	53	1	7	1
0	0	0	4	0	891	0	66	8	31
173	3	132	68	116	1	479	0	27	1
0	0	0	0	0	36	0	930	1	33
2	2	11	13	4	6	8	4	950	0
0	0	0	0	0	13	1	54	0	932

In this case the classes more affected were 1-4, 7-1, 3-5 and now it must be highlighted the misclassification for 7-3 and 7-5. The misclassifications even if higher now are more clear and there are very definite classes which misclassifify each other.

Two layers but with momentum:

30.00000 86.58081 0.13598 0.15570

31.00000 92.60944 0.13585 0.15580

32.00000 87.32944 0.13582 0.15580

33.00000 80.65899 0.13568 0.15540

34.00000 83.51344 0.13572 0.15510

35.00000 85.85716 0.13568 0.15480

36.00000 83.63889 0.13567 0.15510

37.00000 85.75222 0.13553 0.15480

38.00000 77.81586 0.13542 0.15470

39.00000 70.93562 0.13547 0.15480

Training SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	90.38%	9.62%	0.0%
	Experiment 2.5		
	OK	ERROR	REJECTION
Classification Coefficients	86.43%	13.57%	0.0%

Testing SET			
	Refence		
	OK	ERROR	REJECTION
Classification Coefficients	87.42%	12.58%	0.0%
	Experiment 2.5		
	OK	ERROR	REJECTION
Classification Coefficients	84.49%	15.51%	0.0%

Confusion Matrix: Practically identical to the previous one.

CONCLUSIONS

Some of the practical tricks from the paper of LeCun permitted better results from the standard solution.

Overfitting couldn’t be completely avoided. I tried the implementation of drop out with scarse success. The implementation of rejection of samples could be a solution.