Lab exercise 04

Exercise 1.

We make comparison with SGD classifier, train-test partition is used with percentage (0.99, 0.01).

	SGD		MLP: 5 neurons		MLP: 10 neurons		MLP: 15 neurons		MLP: 20 neurons	
	Train accuracy	Test accuracy	Train accuracy	Test accuracy	Train accuracy	Test accuracy	Train accuracy	Test accuracy	Train accuracy	Test accuracy
Before normalizing	0.987071	0.96	0.100606	0.04	0.100707	0.030000	0.100606	0.04	0.100606	0.05
After normalizing (130 iterations)	0.974545	0.93	0.920505	0.840000	0.984343	0.919999	0.987070	**0.95**	0.988080	0.909999
Time	1.491018 s		1.210842 s		1.359977 s		1.314998 s		2.123635 s	
After normalizing (250 iterations)	0.975656	0.85	0.944040	0.809999	0.995858	**0.96**	0.997777	0.95	0.998787	0.940000
Time	2.992220 s		4.014692 s		4.055813 s		2.691710 s		2.549641 s	
After normalizing (400 iterations)	0.974949	0.879999	0.962525	0.85	0.999191	**0.93**	0.999494	0.920000	1.0	0.909999
Time	2.182561 s		3.970130 s		4.027547 s		4.324236 s		7.001980 s	

Conclusions

As we can see from the table, it is necessary to normalize the data while working with MLP classifier. We can also notice that increasing the number of neurons makes the result better, but only until certain moment. In our case rather good results can be obtained with 10-15 neurons in the layer. Concerning time, it can be noted that sometimes we need more time to work with less number of neurons. The reason lies in the fact that for certain number of neurons process does not converge, therefore it takes more time. We can also see, that it is not necessary to take too many iterations, because it can lead to overtraining (as in the case with 20 neurons and 400 iterations).

Exercise 2.

130 iterations:

	Layer 2: 5 neurons	Layer 2: 10 neurons	Layer 2: 15 neurons	Layer 2: 20 neurons
Layer 1: 5 neurons	MLP train accuracy: 0.7868686868686868 MLP test accuracy: 0.72 MLP average time: 1.0526331424713136	MLP train accuracy: 0.9933333333333334 MLP test accuracy: 0.9 MLP average time: 1.393911647796631	MLP train accuracy: 0.9994949494949494 MLP test accuracy: **0.9400000000000001** MLP average time: 2.6541562557220457	MLP train accuracy: 0.9994949494949494 MLP test accuracy: 0.9299999999999999 MLP average time: 2.7596498012542723
Layer 1: 10 neurons	MLP train accuracy: 0.9372727272727273 MLP test accuracy: 0.8800000000000001 MLP average time: 1.2697892665863038	MLP train accuracy: 0.9981818181818183 MLP test accuracy: **0.93** MLP average time: 1.5316901206970215	MLP train accuracy: 0.9994949494949494 MLP test accuracy: **0.95** MLP average time: 2.7261413097381593	MLP train accuracy: 0.9994949494949494 MLP test accuracy: **0.9400000000000001** MLP average time: 2.719267463684082
Layer 1: 15 neurons	MLP train accuracy: 0.9434343434343434 MLP test accuracy: 0.8099999999999999 MLP average time: 1.2670989036560059	MLP train accuracy: 0.9973737373737374 MLP test accuracy: 0.9099999999999999 MLP average time: 1.5906025886535644	MLP train accuracy: 0.9994949494949494 MLP test accuracy: 0.9099999999999999 MLP average time: 2.554352140426636	MLP train accuracy: 0.9994949494949494 MLP test accuracy: 0.93 MLP average time: 2.7849754810333254
Layer 1: 20 neurons	MLP train accuracy: 0.8938383838383837 MLP test accuracy: 0.85	MLP train accuracy: 0.9974747474747474 MLP test accuracy: 0.93 MLP average time: 2.373381757736206	MLP train accuracy: 0.9994949494949494 MLP test accuracy: 0.9199999999999999	MLP train accuracy: 0.9994949494949494 MLP test accuracy: **0.9400000000000001**

	MLP average time: 1.246848726272583		MLP average time: 2.69682297706604	MLP average time: 2.7055124759674074

250 iterations:

	Layer 2: 5 neurons	Layer 2: 10 neurons	Layer 2: 15 neurons	Layer 2: 20 neurons
Layer 1: 5 neurons	MLP train accuracy: 0.9053535353535354 MLP test accuracy: 0.8400000000000001 MLP average time: 2.979155397415161	MLP train accuracy: 0.9991919191919191 MLP test accuracy: **0.93** MLP average time: 2.644731569290161	MLP train accuracy: 1.0 MLP test accuracy: **0.93** MLP average time: 4.039958477020264	MLP train accuracy: 1.0 MLP test accuracy: 0.9200000000000002 MLP average time: 4.388377666473389
Layer 1: 10 neurons	MLP train accuracy: 0.4735353535353536 MLP test accuracy: 0.36 MLP average time: 2.1620946884155274	MLP train accuracy: 0.8856565656565657 MLP test accuracy: 0.82 MLP average time: 2.1534940242767333	MLP train accuracy: 1.0 MLP test accuracy: 0.9199999999999999 MLP average time: 4.182448101043701	MLP train accuracy: 1.0 MLP test accuracy: **0.96** MLP average time: 4.0920525074005125
Layer 1: 15 neurons	MLP train accuracy: 0.6157575757575757 MLP test accuracy: 0.49000000000000005 MLP average time: 1.8117749214172363	MLP train accuracy: 0.9703030303030303 MLP test accuracy: 0.8699999999999999 MLP average time: 2.390315532684326	MLP train accuracy: 1.0 MLP test accuracy: 0.8899999999999999 MLP average time: 3.5060703277587892	MLP train accuracy: 1.0 MLP test accuracy: 0.9 MLP average time: 3.584311056137085
Layer 1: 20 neurons	Number of neurons: 5, 20 MLP train accuracy: 0.3561616161616162 MLP test accuracy: 0.33 MLP average time: 0.6531939029693603	MLP train accuracy: 0.9996969696969698 MLP test accuracy: 0.89 MLP average time: 3.804465961456299	MLP train accuracy: 1.0 MLP test accuracy: 0.9199999999999999 MLP average time: 4.703778457641602	MLP train accuracy: 1.0 MLP test accuracy: **0.9299999999999999** MLP average time: 4.407886791229248

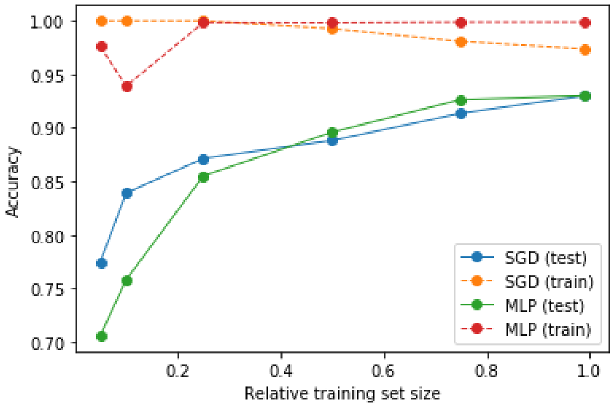
Conclusions

In this case we use two layers. We can notice that increasing the number of iterations did not lead us to much better results and many of them are overtrained, and the time has increased. For our dataset 5 neurons are not enough, so a good solution will be to use, for example, (10, 20).

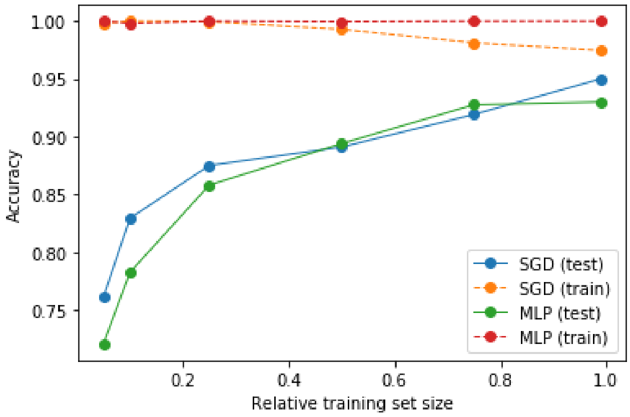
Exercise 3.

We will test the MLP classifier with 10 neurons on the first layer and 20 on the second.

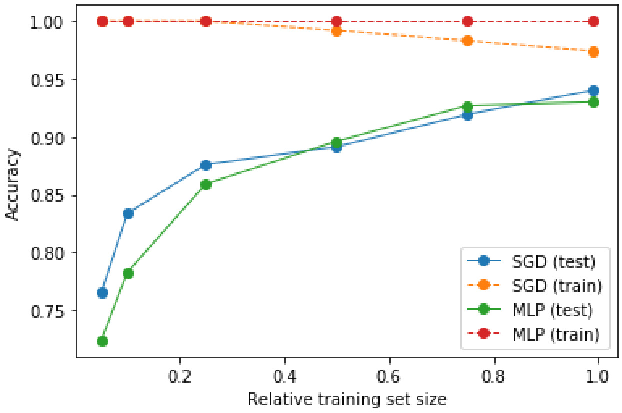
max_iter = 130



max_iter = 200



max_iter = 250

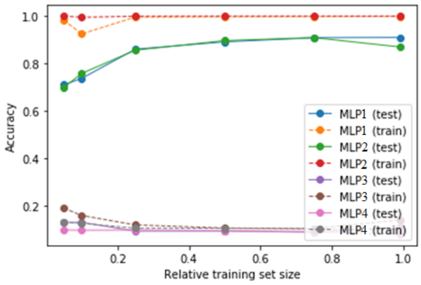


Conclusions

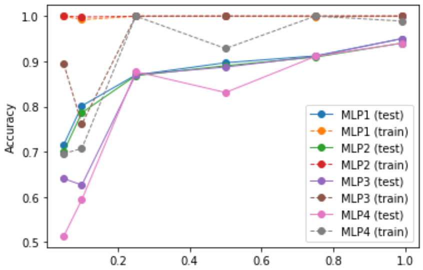
For all numbers of iterations we see more or less the same picture: MLP classifier is more prone to adapting to training data, therefore it gives perfect score for training data, but not so good for testing. I think it is more inclined to overtrain (at least, in this case). Also we can notice that

Exercise 4.

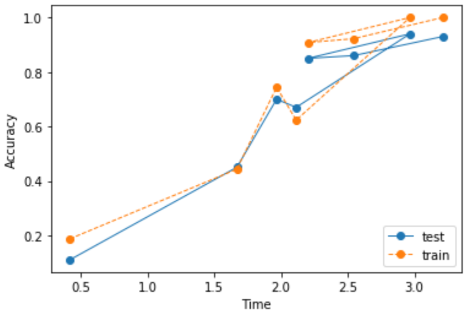
In the first picture you see the classifiers with learning_init_rate = [0.01, 0.1, 1, 10]. Last two are very bad because of too big learning rate. They just jump over the desired point. For the first one I set big number of iterations – 1000, for the second – 200. First one shows the average time of learning 19.1800655, the second – 2.688358.



In the second picture you see the classifiers with learning_init_rate = [0.01, 0.1, 0.15, 0.2]. As I mentioned earlier, first one is veeery slow, but not much more precise.



Next you see the plot of accuracy, depending on time of learning. I used here classifiers with learning_init_rate = [0.8, 0.4, 0.3, 0.25, 0.23, 0.21, 0.2, 0.1]



Conclusions

Too small learning rate is infeasible, because it takes very big amount of time to learn, but the outcome is not much better than for lower learning rates. On the other hand, too big learning rate makes the process faster, but from some point leads to the significant loss of accuracy because of jumping over the point. So, for this case I would define a range of rather efficient rates as something like [0.1, 0.2].

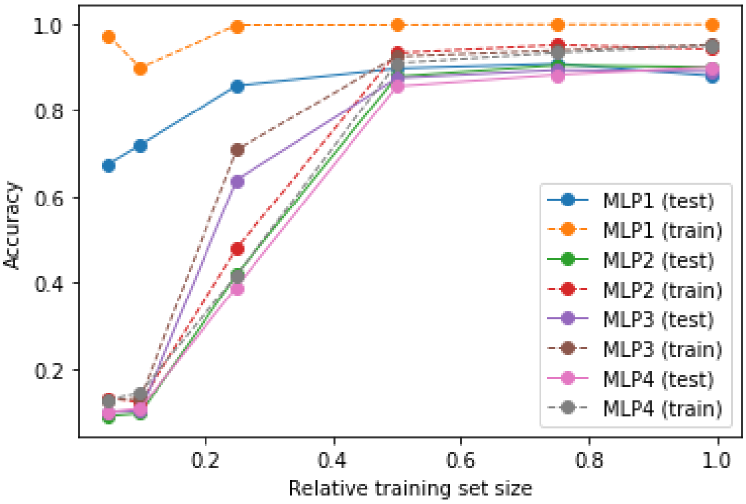
Exercise 5.

MLP1: without early_stopping

MLP2: validation_fraction = 0.1

MLP3: validation_fraction = 0.2

MLP4: validation_fraction = 0.4



As I understood, validation fraction is a part of the training set that is used to check whether it is time to stop learning (if validation score is not improving), so it lies inside training data. But when we use train_test_split, we leave a part of dataset to check the accuracy after learning, so this part doesn't affect the result.
At the picture we can see that with this option accuracy becomes a bit higher. About test and training scores:

- ✓ Starting from the partition (0.5, 0.5) accuracies of classifiers with early stopping almost does not increase, but before this point the curve is rather steep.
- ✓ Classifiers with early stopping do not reach such high scores in training, thus they are not going to overtrain.