1 Wine

2 MNIST

2.1 MLP

First of all, we added a holdout validation to the sklearn implementation. A factor of approximately one-third of dataset has been chosen as the evaluation dataset. This is a common value for a train/test split. We used the build-in function from sklearn to perform this task. The accuracy is measured as the categorical cross-entropy and evaluated using the test data set.
We trained the MLP multiple times with varying parameters. Some of them are shown in the following part.

2.1.1 Learning rate

As expected we observed a higher accuracy if we decreased the learning rate. This also increased training time significantly.
Here we compare the different learning rates.
The following params were used:

```
"hidden_layer_sizes": (20, 20),
"activation": 'relu',
"solver": 'adam',
"batch_size": "auto",
"learning_rate": 'constant',
"learning_rate_init": 0.001,
"max_iter": 200,
"shuffle": True,
"random_state": None,
"tol": 0.0001,
"verbose": 1,
"early_stopping": True,
"validation_fraction": 0.2,
"n_iter_no_change": 10,
```

This setting resulted in 30 Epochs until the early stopping kicked in. The max accuracy was 0.949221.
The same settings with a learning rate of 0.0001 resulted in a max accuracy 0.950130 but needed 122 Epochs before being stopped by the early stopping.
Setting the learning rate to 0.01 resulted in 17 epochs and an accuracy of 0.949394.
These results where expected.
Because we used the adam optimizer no learning rate decay had been applied.

2.1.2 Hidden Layers

Different hidden layer configs had been also evaluated.
We picked the following two:

- Two 20 neuron hidden layers as used in the config shown in 2.1.1. As already shown the maximum accuracy with our default learning rate of 0.001 was 0.949221.
- A 200 neuron layer followed by a 100 neuron layer. This significantly larger network took longer to train but resulted in an increased evaluation score of 0.977273 after 50 Epochs with a learning rate of 0.001. The other params are the same as described in 2.1.1.

Further numbers of layers and neurons had been evaluated but the results were not significant enough to discuss in detail.
A batch size of "auto" (sklearn sets it to the number of samples used for training) resulted in the best scores. We tried some over values like 512, 224, 64 with worse results.
The best overall score was 0.977273 with the larger mlp architecture.
The code can be found at GitHub.

2.2 CNN

We developed a simple convolutional neural network for the MNIST dataset using the keras framework build upon tensorflow.

The architecture:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
```

```
flatten_3 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               803072
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                2570
=================================================================
Total params: 861,386
Trainable params: 861,386
Non-trainable params: 0
```

The three convolutional layers are used for feature extraction. The two max-pooling layers reduce some spacial information. Dropout should help against the overfitting of the network. The last "logical" step consists of two dense layers. The last one uses softmax as an activation function to achieve a cleaner classification. All other layers use a normal relu activation.
The network achieves an accuracy of 0.9929 measured as categorical cross-entropy.
Code and params can be found at GitHub.