Lab 03

Source code for lab 03 is at src/mlcourse/digits.py.

Preparation

The digits dataset consists of 8x8 pixel images with a 4 bit grey scale palette. To prep the images, pixel values are transformed to the of range 0 to 1. The labels are converted to one-hot arrays of the length 10 - as there are 10 different digits in the set. After shuffeling, the data is split into a training and test portion using a ratio of 3:1.

Multi-layer perceptron classifier

The MLP classifier can be configured with a hidden layer topology. The first layer (input) needs to match the amount of features - 64 pixels in our case. The output layer however needs to match the number of labels: 10. In between, multiple other layers with varying node cound can be placed. I chose a single layer with 48 nodes, as further increasing the breadth or depth did not improve the accuracy (of currently 96%).

As an activation function, the ReLU (rectified linear unit) reaped the best results. It is basically an identity function, that drops negative input `f(x) = max(0, x)`.

The solver is used to optimize the cost function during training. Here, `adam` (a stochastic gradient-based optimizer) achieved the highest accuracy.

Trying different learning rate schedules, did not influence the outcome, so the default is used.

Using keras to train on the same dataset, a similar setup can be used: `adam` optimizer and `relu` hidden layer activation. Again, increasing the breadth or depth of the topology didn't benefit the accuracy of the network.

For the final layer of the keras network, the softmax activation had to be used: it scales all cells to the range from 0 to 1 and limits the row sum to 1. This improves comparability with our validation data.

Keras also lets us define a loss function, that will be minimized during training. Here, `mean_absolute_error` resulted in the best predictions.

Evaluation

To evaluate our networks, the common accuracy metrics is used. Additionally, a confusion matrix is constructed for both networks, to visualize which digits where *confused* with which in the prediction phase.