Course: Machine learning

By: Pavlo Mozharovskyi

Tutorial to Lecture 5 (part `Python`): Perceptron, neural network, and the back-propagation algorithm

Task 1: Multilayer perceptron.

(a) Set up the `tensorflow` environment and load the MNIST data set from `tensorflow.keras.datasets`. Re-scale the input to $[0, 1]$. Visualize first 10 training digits.

(b) Consider first training 200 digits containing only '0's and '1's for training and in the same way 2000 testing digits for testing. Flatten the data to obtain matrices of dimensions 200×784 for training and 2000×784 for testing.

(c) For this input (and output) construct a simple neural network with two hidden layers each containing 10 neurons and 1 neuron in the output layer; use '`sigmoid`' as activation function for all neurons. (You can use `tensorflow.keras.models.Sequential` for the neural network and `tensorflow.keras.layers.Dense`.) Observe the output of non-fitted neural network on the first training sample.

Setup the optimizer (*e.g.*, '`adam`'), the loss (*e.g.*, `tensorflow.keras.losses.MSE`), and the metric(s) (*e.g.*, '`accuracy`') to be evaluated during training for the neural network. (You can use function `compile` of the neural network object.) Fit the neural network (*e.g.*, during 15 epochs) using the training data.

Use the testing data set from sub-task (b) to evaluate the accuracy of the fitted neural network.

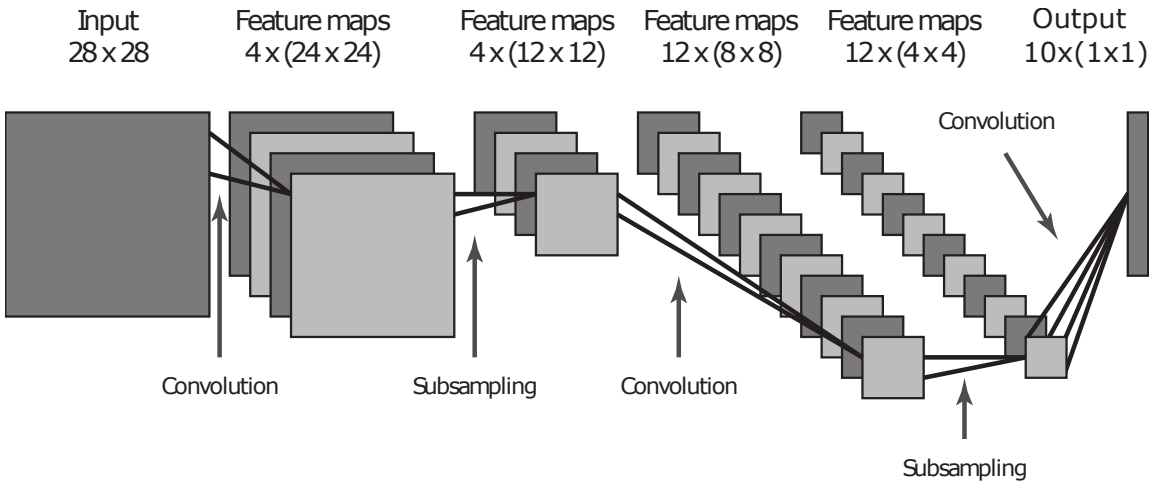
(d) Repeat the same workflow for the pair of digits '4' and '9', with the following differences: use 2 sigmoid neurons in the output layer; use original non-flatted input and flatten it at the "beginning" of the neural network (you can use `tensorflow.keras.layers.Flatten`); try not to modify the training labels but create your own loss function instead; train the neural network longer, *e.g.* during 100 epochs. Evaluate the accuracy of the fitted neural network on the test data set.

(e) Perform the training and testing from sub-tasks (c) and (d) on the linear discriminant analysis (you can use the class `sklearn.discriminant_analysis.LinearDiscriminantAnalysis`). Report the accuracy on the test data, compare it with neural networks' outputs, comment.

Task 2: Convolutional neural network.

(a) From the same MNIST data set from task 1, consider first 1000 training samples for fitting the neural network and this time the entire test set for testing its accuracy. Prepare the data.

(b) Create a convolution neural network using the following architecture:



During each convolution use a 5×5 kernel and 2×2 kernel during each pooling. Use the rectified linear unit ('`relu`') as the activation function of the feature maps and `tensorflow.keras.losses.SparseCategoricalCrossentropy` as the loss.

Train the neural network, *e.g.* during 10 epochs.

(c) Check the accuracy of the fitted neural network on the entire test data set.