Exercise 2b - MLP

Christina, Pascal, Silas

Task

With this exercise you should use your framework for applying an MLP approach to the MNIST dataset.

The goal of this exercise is to train an MLP with one hidden layer. And experiment with different parameters.

learning rate c: 0.1, 0.05, 0.011, 0.01
(training) epochs: 100
size of hidden layer: 20, 40, 60, 80

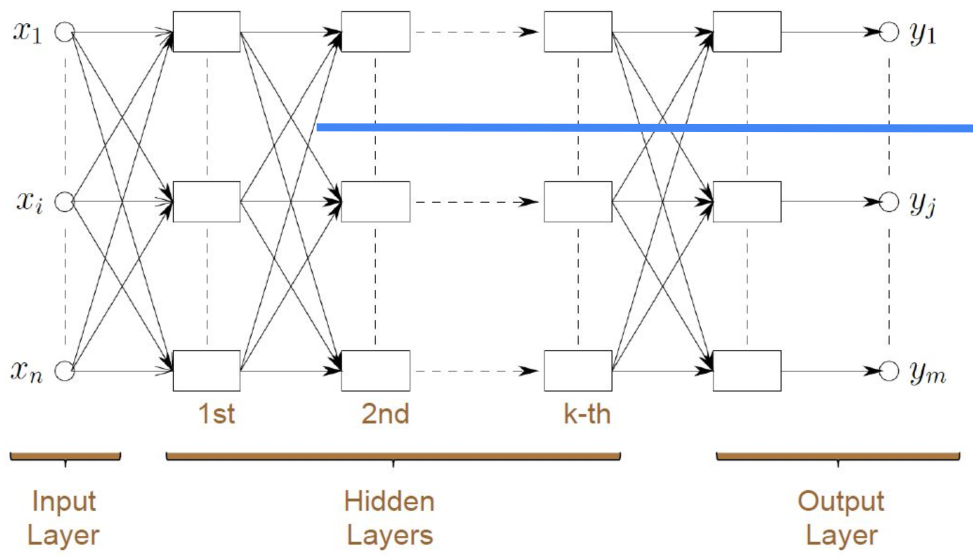
Test accuracy with the best parameters found during cross-validation.

4-fold-cross-validation

source of picture: ILIAS

Theoretical Background: MLP

- Note that the bias of each individual neuron is not shown explicitly.



$$y = g(w'x) = g\left(\sum_{i=1}^{n} w_i x_i\right)$$

Softmax function:

$$\sigma : \mathbb{R}^K \to (0,1)^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \dots, K.$$

Learning rate:

for all predecessors j of neuron i **do**

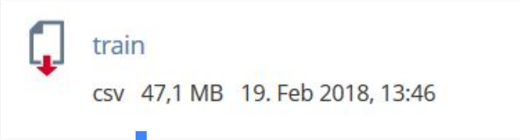
$$\hat{w}_{ji} = w_{ji} + c \cdot out(j) \cdot \delta_i$$

end for

source of pictures: lecture slides, https://en.wikipedia.org/wiki/Softmax_function, 21.04.2018

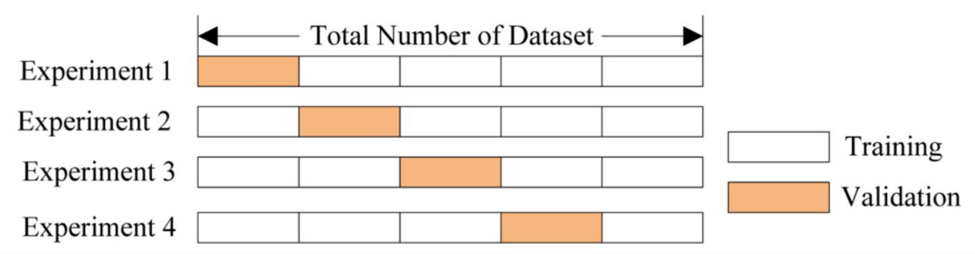
Theoretical Background: 4-fold-cross-validation

1

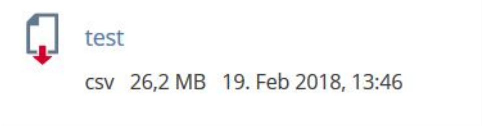


split train.csv randomly in four parts

do 4-fold-cross-validation



2



final accuracy calculation

source of picture: ILIAS,
https://www.kaggle.com/dansbecker/cross-validation, 21.04.2018

Used library: TensorFlow

- library for numerical computation using data flow graphs
- built to scale
- currently one of the most widely used deep learning libraries in research and industry

- every operation (matrix multiplication, softmax function, placeholder, variables etc) is represented as a node in a graph.
- placeholder and variables always n-dimensional
- graphs are lazy-evaluated

source of picture and text: https://www.tensorflow.org/, 21.04.2018, https://en.wikipedia.org/wiki/TensorFlow, 21.04.2018

Code Samples

```
W_1 = tf.Variable(tf.random_uniform([size_input_layer, self.size_hidden_layer]))  # weights input to hidden
b_1 = tf.Variable(tf.random_uniform([self.size_hidden_layer]))  # biases hidden

W_2 = tf.Variable(tf.random_uniform([self.size_hidden_layer, size_output_layer]))  # weights hidden to output
b_2 = tf.Variable(tf.random_uniform([size_output_layer]))  # biases output

# Matrix with dimension [batch_size x size_hidden_layer], each row represents the hidden layer for one input row.
# Use softmax as activation function.
hidden_layer = tf.nn.softmax(tf.matmul(x, W_1) + b_1)

# Matrix with dimension [batch_size x size_output_layer], each row represents the output layer for one input row.
# Use softmax as activation function.
output_layer = tf.nn.softmax(tf.matmul(hidden_layer, W_2) + b_2)
```

MLP

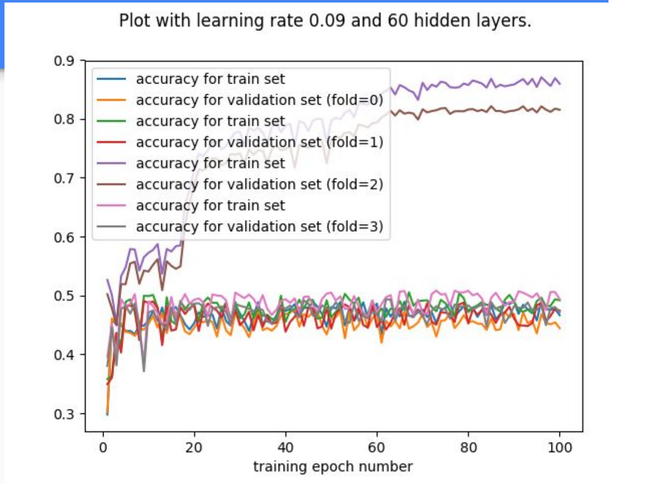
```
        # Optimize model with the current batch
        self.sess.run(optimizer, feed_dict={x: batch_xs, y_labels: batch_ys})
```

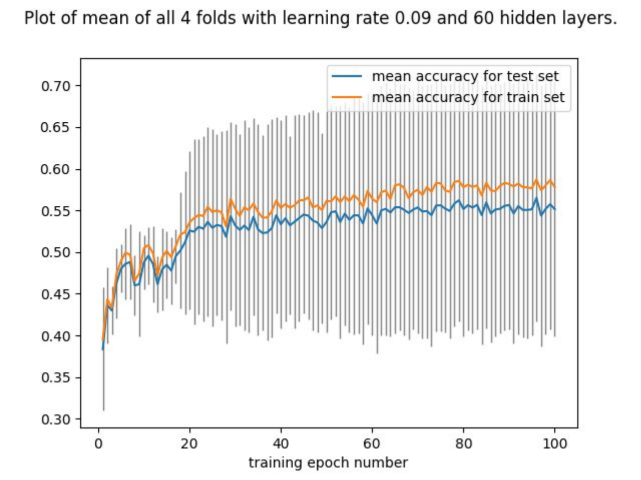
training = optimization

```
    # print the accuracy on training and validation set after each training epoch
    # and store it in an array
    self.accurancy_train_set.append(self.sess.run(accuracy, feed_dict={x: train_x, y_labels: train_y}))
    self.accurancy_valid_set.append(self.sess.run(accuracy, feed_dict={x: test_x, y_labels: test_y}))
```

cross-validation

Results:
plots of cross-validation.



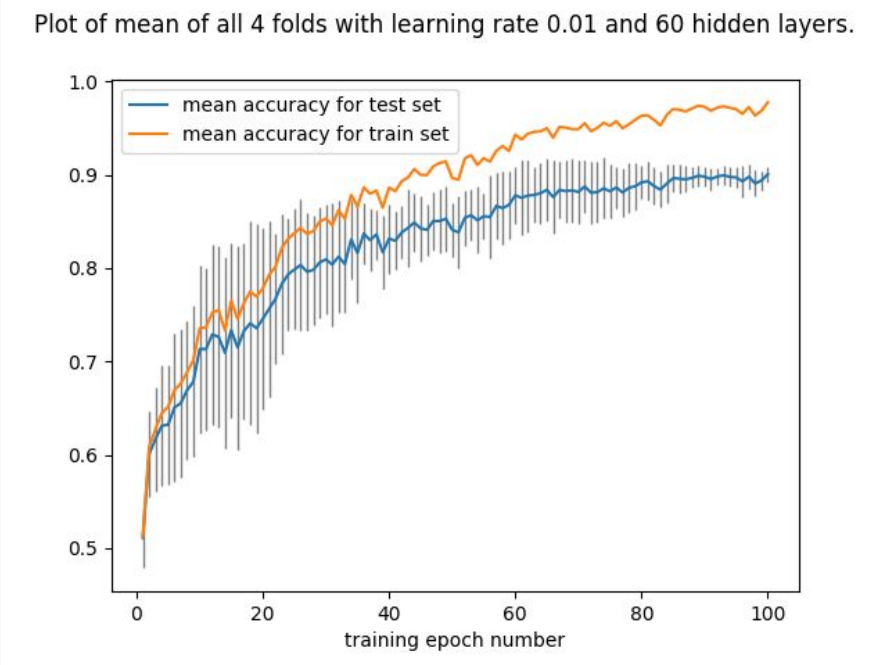


learning rates c:
1, 0.5, 0.1, 0.05, 0.09, 0.011, 0.01, 0.009, 0.005

(training) epochs: 100
size of hidden layer: 20, 40, 60, 80

Learning rates of 1 and 0.5 did not produce useful results

Results:
best parameter found in cross-validation



learning rate c: 0.01
size of hidden layer: 60

results in an accuracy of 0.90
for the cross-validation

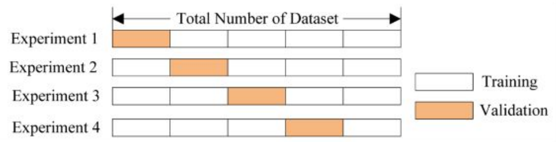
Results:
accuracy for test set

1 train

csv 47,1 MB 19. Feb 2018, 13:46

split train.csv randomly in four parts

do 4-fold-cross-validation



2 test

csv 26,2 MB 19. Feb 2018, 13:46

final accuracy calculation

source of picture: ILIAS, https://www.kaggle.com/dansbecker/cross-validation, 21.04.2018

final accuracy: 0.92

Comparison

0.91 (20 hidden layers, learning rate 0.02)

https://mmlind.github.io/Simple_3-Layer_Neural_Network_for_MNIST_Handwriting_Recognition/ [21.4.2018]

0.953 (300 hidden layers)

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998, p.13 https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=726791 [21.4.2018]

Possible Improvements

Testing with more layers

Preprocess data (slant correction, cropping, resizing)

Use other threshold functions instead of soft-max

Use dynamic learning rate (start with greater learning rate and then decrease)