MACHINE LEARNING

SHEET 4

13.01.2022

10:15 A.M. UNTIL 11:45 A.M. VIA ZOOM

Please prepare the exercises in order to present them in the meeting. Fill-in the exercises you solved in the questionnaire in StudIP until 12.01.2022 4:00 p.m.
Join the meeting via this Zoom link:
https://uni-trier.zoom.us/j/81742172869?pwd=UjYvODM0TytqcjBnckxvaXE0cVJHQT09

TASK 1: FEED FORWARD NEURAL NETWORK FROM SCRATCH

In this exercise, we will build a Feed Forward Neural Network (FFNN) from scratch using Python. Assume that each layer in this toy example has the same number of neurons.

a) Build a class *Network* that contains several layers of neurons and has a method for feeding in data to get the result. It should be possible to set an arbitrary number of layers.
b) A layer consists of several neurons. For the sake of simplicity, we will not model each neuron as a separate unit. Instead keep (n-dimensional) vectors that track the state of the layer's weights and biases. Initialize these vectors with suitable values.
c) Furthermore, apply an activation on the results of the layer. The activation function can be defined by the user. It can be one of the following three:
    - Identity
    - Rectified Linear Unit (ReLU)
    - Logistic Function (Sigmoid)
    - Softmax

    Take the following link as a reference to the definition of the functions (https://en.wikipedia.org/wiki/Activation_function). What are strengths and weaknesses of these activations?
d) Test your implementation by defining a network of 5 layers with 10 neurons each. What effect do different activation functions have? How did you initialize your weights and biases?

TASK 2: FEED FORWARD NEURAL NETWORK WITH KERAS

In this exercise, we will build a FFNN by using the Deep Learning library Keras integrated in Tensorflow (https://www.tensorflow.org/guide). Use it to build your model in the following tasks.

a) Rebuild your neural network from Exercise 1 with the configuration in d). Use a ReLU activation for all inner layers. Initialize all weights with uniformly distributed values and all biases with ones. Feed it some sample data and look at the results. Print a summary of your model with its layers and the number of trainable parameters. Explain how the number of trainable parameters is computed in this particular case.

DEPARTMENT OF BUSINESS INFORMATION SYSTEMS II
PROF. DR. RALPH BERGMANN
UNIVERSITY OF TRIER

b) In order to train our model, put together a synthetic training data set. The examples have ten numerical features. Each feature of the examples is drawn randomly from a uniform distribution with a minimum value of -2 and a maximum value of 4. The dataset contains 15000 examples for training and 1000 for testing.
c) Label the training data according to the following rule: The label for each example is the third lowest feature value unless this value is smaller than 0.5. In this case the label is the third lowest value divided by two.
d) Setup a training loop by
    - Preparing the created model to output data that fits the shape of the label data
    - Using the Adam optimizer
    - Using a suitable training loss (Why did you choose it?)
    - Using a suitable training metric (Why did you choose it?)
e) Train your model with a batch size of 32 for 10 epochs and visualize the aggregated training statistics for each epoch in multiple plots, i.e., a plot of your training loss and one of your training metric.
f) Validate your trained model on the testing data. How can you determine if model training should be stopped early?

TASK 3: OPTIMIZING NEURAL NETWORK TRAINING

In this exercise, we will explore the effects of several optimization techniques for the training and structure of neural networks. We will explore those techniques on the MNIST dataset (https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist/load_data). Make sure to set the global seed of TensorFlow to a fixed value to ensure comparability across all implementations (see https://www.tensorflow.org/api_docs/python/tf/random/set_seed).

a) Load the MNIST dataset via the respective built-in function of Keras. Scale the data to the range [0,1]. Visualize a few of the digits with its label.
b) Setup a model with an input layer that transforms the 28 by 28 input to a flattened vector of 784 features, a ReLU-activated dense layer with 32 hidden neurons, and a softmax-activated output layer with 10 neurons (1 for each class). Train this model on the training data for five epochs with an Adam optimizer, categorical cross-entropy as loss function, and accuracy as a metric. What is the training and testing accuracy of this setup?
c) Try to improve the model by replacing the hidden layer with three ReLU-activated layers of 256, 128, and 64 neurons. What is the training and testing accuracy of this setup? What are advantages and disadvantages of a deeper model?
d) Visualize the training curves of accuracy and cross-entropy of both models from before in a single diagram. Compare the curves. What do you notice? Are the training metrics suitable for representing the testing metrics?

TASK 4: SOLVING TRAINING PROBLEMS

In this exercise, we will look at possible problems that might occur during training and how to solve them

a) Setup a training environment similar to the one that was used in Exercise 3c):
   - Fetch the MNIST dataset and scale the data.
   - Build the same deep model variant.
   - Setup the same training loop.

b) Try to improve the model by adding a dropout layer after the last hidden layer. Set its dropout rate to 0.25. What is the training and testing accuracy of the modified setup? How does dropout work and why is it effective?

c) Try to improve the model by adding batch normalization to each hidden layer. What is the training and testing accuracy of the modified setup? How does batch normalization work and why is it effective?

d) Try to improve the model by adding an *l1 regularization* on the weights of each hidden layer. What is the training and testing accuracy of the modified setup? How does regularization work and why is it effective?

e) Visualize the training curves of accuracy and cross-entropy of all models from before in a single diagram. Compare the curves.