Machine Learning for Agricultural Applications

Assignment 4

Prof. Dr. Niels Landwehr
Dr. Julian Adolphs

Summer Term 2020

Release: 28.05.2020
Discussion: 04.06.2020

Task 1 – Keras [20 points]

In this exercise, we use Keras to train a multilayer perceptron. Load the fashion_mnist data set, which is included in Keras. Use the sequential API in Keras to create a multilayer perceptron, that is, a neural network consisting of multiple fully connected layers. You can freely choose the number of hidden layers and neurons per layer. Train the model on the training part of the fashion_mnist data set and evaluate it on the test part of the data set. Try to improve the model to reach a test accuracy around 90%. Use only dense layers and play with the number of hidden layers, nodes and number of training epochs.

If you worked on your computer locally, also try to run the program on **Colab** or a similar cloud platform that allows to run your code on GPUs.

Task 2 – AutoDiff in TensorFlow [15 points]

Write a python script that uses the `GradientTape` function of TensorFlow for calculating the gradient of the function

$$f(x_0, x_1, w_0, w_1, w_2) = \left[\tanh\left(\frac{1}{w_0x_0 + w_1x_1 + w_2}\right) - 1\right]^2, \qquad x_0, x_1, w_0, w_1, w_2 \in \mathbb{R}.$$

Also implement the forward pass as in Assignment 3 and use `GradientTape` to calculate all intermediate (downstream) gradients that you computed in the backward pass of Assignment 3.

Use the values $x_0 = 1.2$, $x_1 = 3.1$, $w_0 = 1.5$, $w_1 = -1.5$, $w_2 = 2.0$ and compare the outputs with the results from Assignment 3.

Task 3 – Regression with TensorFlow [15 points]

Use TensorFlow to perform a linear regression with squared loss on 3D random data, i.e. fit a plane

$$f(x, y) = z = w_1x + w_2y + b, \qquad \text{with} \quad x, y, w_1, w_2, b \in \mathbb{R}$$

through a cloud of points. Pick arbitrary "true" values for the parameters w_1, w_2, b and generate artificial data by drawing x and y values randomly from `tf.random.uniform`, computing z from your function and adding noise using `tf.random.normal` to the z-labels.

Use the `GradientTape` function of TensorFlow to implement a gradient descent algorithm for training (fitting) the model to your artificially generated data.

Plot the convergence of the parameters as well as the point cloud and the resulting plane.