Computer Vision I
Bonus Assignment

Prof. Stefan Roth
Jannik Schmitt
Jan-Martin Steitz

TECHNISCHE UNIVERSITÄT DARMSTADT

08/02/2021

This assignment is due on February 21st, 2021 at 23:59.

Please refer to the previous assignments for general instructions and follow the handin process described there.

Problem 1: Neural Networks (20 Points)

In this problem, we will design a multilayer perceptron (MLP) to classify bike and plane images. Image and label data are already loaded in training and validation splits. The neural network is represented by the class `Network` which builds the network according to a list containing input, hidden and output dimensions.

- First, initialize the weights and biases of the neural network with `init_wb`. Biases should be initialized to zero and weights should be initialized randomly according to a normal distribution. The variance of the normal distribution from which the weights are drawn should be equal to $1/n_{i-1}$, where n_{i-1} is the input dimension of the i-th layer.

  (2 points)

The network will use ReLU activations and predict class probabilities for the binary problem using the sigmoid function.

- Implement the `relu` function and back propagate error derivatives in `relu_backward`.

  (1 + 2 points)

- Implement the `sigmoid` function and back propagate error derivatives in `sigmoid_backward`.

  (1 + 2 points)

Next, implement forward and backward propagation for the neural network:

- Implement the function `layer_forward` to perform forward propagation through a single layer. This function may make use of `activation_func` which selects and calls an activation function. `layer_forward` should update the state dictionaries for `z[i]`, the output of the i-th linear layer, and `x[i]`, the output of the i-th activation function, since these values are required to compute the backward propagation.

  (3 points)

- Implement the function `forward` which iteratively calls `layer_forward` in order to perform forward propagation and outputs the network's prediction.

  (2 points)

- Next, implement function `layer_backward` that propagates the derivatives backward through a layer. Here the values stored in the state dictionaries `z[i]` and `x[i-1]` have to be used. This function may make use of `activation_func_backward` which selects and calls the backward method of an activation function. The function `layer_backward` computes the partial derivatives of the error function with respect to the layer parameters `w[i]` and `b[i]` and updates the states of `dw[i]` and `db[i]` accordingly. Further, it returns the partial derivatives of the error function with respect to the layer's input.

  (3 points)

- We provide the function `back_propagation` which iteratively calls `layer_backward` to propagate derivatives through the network in order to update the states of `dw` and `db` for all layers.

We further require the following methods to train the network:

- Implement the function `update_wb` to update the states of the network's weights `w` and biases `b` by performing a SGD step given the learning rate and states of `dw` and `db`.

  (2 points)

- Finally, implement the function `shuffle_data` which permutes the training data so that we can extract random mini batches required for SGD from it.

  (2 points)

- The function `train` extracts mini batches from the shuffled arrays to train the neural network.

We already provide methods for evaluating the prediction accuracy of the neural network. You can try changing network and training parameters to improve the network's performance.

Submission: Please include only `problem1.py` in your submission.