Exercise 4 in Computer Vision: Deep Learning
Discussion on 11/18/2019 10.00 am – 11.00 am (NB 3/57)

We will demonstrate the vanishing gradient problem. To this end, we will train a number of multi-layer perceptrons with different numbers of hidden layers and look at the change of weights in the individual layers during gradient descent.

Train a multi-layer perceptron with one, two, three, and eight hidden layers with 20 hidden neurons each on the entire MNIST dataset (you can use `DLCVDatasets` from earlier exercises). That is, your models are:

$$\tilde{y} := W_2\sigma(W_1x + b_1) + b_2$$
$$W_2 \in \mathbb{R}^{10\times 20}, b_2 \in \mathbb{R}^{10}, W_1 \in \mathbb{R}^{20\times 784}, b_1 \in \mathbb{R}^{20}$$

$$\tilde{y} := W_3\sigma(W_2\sigma(W_1x + b_1) + b_2) + b_3$$
$$W_3 \in \mathbb{R}^{10\times 20}, b_3 \in \mathbb{R}^{10}, W_1 \in \mathbb{R}^{20\times 784}, b_1 \in \mathbb{R}^{20}, W_2 \in \mathbb{R}^{20\times 20}, b_2 \in \mathbb{R}^{20}$$

$$\tilde{y} := W_4\sigma(W_3\sigma(W_2\sigma(W_1x + b_1) + b_2) + b_3) + b_4$$
$$W_4 \in \mathbb{R}^{10\times 20}, b_4 \in \mathbb{R}^{10}, W_1 \in \mathbb{R}^{20\times 784}, b_1 \in \mathbb{R}^{20}, W_{2,3} \in \mathbb{R}^{20\times 20}, b_{2,3} \in \mathbb{R}^{20}$$

$$\tilde{y} := W_9\sigma(W_8\sigma(...\sigma(W_1x + b_1) + b_2)... + b_8) + b_9$$
$$W_9 \in \mathbb{R}^{10\times 20}, b_9 \in \mathbb{R}^{10}, W_1 \in \mathbb{R}^{20\times 784}, b_1 \in \mathbb{R}^{20}, W_{2,3,5,6,7,8} \in \mathbb{R}^{20\times 20}, b_{2,3,4,5,6,7,8} \in \mathbb{R}^{20}$$

For training choose a `GradientDescentOptimizer` with stepsize 0.1 and train every model for 20 epochs (that is choose a batch size and traverse the entire MNIST dataset 20 times). Compute the training and test accuracy and the change in each of the matrices $W_1, ..., W_9$ for each gradient descent step and print it in reasonable frequency (e.g. every 5 seconds of training).

Bonus: After that, train all the models with a ReLU activation function. What can you learn?

Although it has not yet been introduced in the lecture, please use the cross-entropy loss in combination with softmax to train the models. Both the loss and the softmax function are combined in the tensorflow method `tensorflow.compat.v1.nn.sparse_softmax_cross_entropy_with_logits`. That is, the loss function is given via

```
loss = tf.reduce_mean(
  tf.compat.v1.nn.sparse_softmax_cross_entropy_with_logits(logits=model_prediction_wo_softmax,
  labels=indexed_labels))
```