University of Applied Sciences RheinMain
Prof. Dr. Adrian Ulges

17.11.2016

Machine Learning

Course Work 4

to complete by: 24.11.2016

Please execute this course work (as well as any following) in teams of two.

This exercise will get you started with neural networks. We will use Google's **Tensorflow** library for that. Tensorflow allows you to construct neural network structures flexibly, like MLPs or deep CNNs (which we will cover next week). You will code in Python, and construct a network by building a so-called **flow graph** with neurons as nodes.

Remark: Tensorflow can run on GPUs out-of-the-box, which can give you a 10-20 × speed-up. We have two servers with graphics cards ready for you: Simply log into **cccompute2** *or* **cuda2** *(.local.cs.hs-rm.de) with your computer science LDAP account:*

```
> import tensorflow
...
> session = tensorflow.Session()
I tensorflow/core/common_runtime/gpu/gpu_device.cc:951] Found device 0
with properties: name: Tesla K40c
...
```

Exercise 4.1 (Tensorflow Tutorial)
Go to `www.tensorflow.org` and "get started". Take the blue pill. Work through the tutorial "MNIST for ML Beginners" thoroughly, i.e. implement your own tensorflow-based *softmax regression* classifier for digit classification. Evaluate your classifier on the MNIST dataset (as in the tutorial) and record your results.

As an extra, write out the ten *most badly misclassified* images from the test set. These are the images for which the cross-entropy cost function (as defined in the tutorial) is the highest. Find them, reshape the features to 28x28, re-scale the pixels from $[0, 1]$ to $[0, 255]$, and store the ten resulting images with `cv2.imwrite()`.

Exercise 4.2 (Regression Weights)
In Exercise 02, logistic regression weighted the importance of single terms for a news category. Your digit classifier does just the same for the pixels in your image (i.e., the weights indicate how important a pixel x is for choosing class, say, 0).

Extract the learned weights from your softmax regression model and visualize them as grayscale images. A white/gray/black intensity should indicate that a pixel is positive/neutral/negative for a certain character class.

Exercise 4.3 (MLPs)
Replace softmax regression with an MLP classifier. Use one hidden layer, fully connected layers, and sigmoid activation functions for all neurons. As a cost function, use the squared error introduced in the lecture:

$$E = \sum_i (z_i - t_i)^2$$

(where $\mathbf{z} = (z_1, ..., z_n)$) is your network's output and $\mathbf{t} = (t_1, ..., t_{10})$ the desired output (the one-hot encoding of the true character class). Train and evaluate your classifier with 10, 100 and 300 hidden units. Does it work better than softmax regression?

Remarks: (1) There is no need to code the propagation and activation functions yourself $\rightarrow$ *Use* `tf.matmul()` *and* `tf.Sigmoid()` *to model your layers! (2) Preprocessing helps: Shift your input data from range* $[0, 1]$ *to* $[-0.5, 0.5]$. *(3) Careful initialization helps: Initialize your weights with random values (check out* `tf.random_normal()`*) instead of zeros.*

Exercise 4.4 (Text Classifier)
Implement a new version of your news category classifier (from Exercise 02) with bag-of-words features. As a classifier, use your MLP from above. As a cost function, use cross-entropy. Train and test your classifier on the NYTimes data. Use randomly sampled minibatches of size 200 and test a varying number of hidden units. How well does your classifier work?

Exercise 4.5 (Report)
Put together a presentation of 2-3 slides, showing your quantitative results, visualizations of worst misclassifications and regression weights (Exercise 4.1 and 4.2), and any open questions.