582744 Advanced Course in Machine Learning

Exercise 6 **Due May 2, 23:55 AM**

Rules:

1. Return your solutions in Moodle by the deadline.
2. The submission consists of two parts: (a) A single PDF file containing your answers to all questions. (b) A single file containing your code (either a single plain text source code or a compressed file). Do not include datasets, plots etc in this file, only the code.
3. If you feel comfortable, add an estimate of how many hours you worked on the problems in the beginning of your report.
4. Please typeset your work using appropriate software such as LaTeX. However, there is no need to typeset the pen and paper answers – you can also include a scanned hand-written version.
5. Pay attention to how you present the results. Be concise.
6. Spotted a mistake? Something is unclear? Ask for clarifications in Moodle.

This set of exercises is due on Tuesday May 2, before 23:55 AM.

1 Multilayer perceptron - structure and parameters (1 pt)

Consider a fully connected feedforward neural network with 3 input nodes, two hidden layers of 5 and 8 nodes, and 2 output nodes. The activation functions for the hidden layers are sigmoids and the activation function for the output layer is the identity function.

Answer the following questions:

- How many parameters does the model have?
- How many operations are needed to compute the output of the network for one sample? Count separately additions, multiplications, and other operations.
- How many operations are needed to compute the gradient of the network for one sample?

How would these answers change if the network had just one hidden layer of 13 nodes instead?

2 Multilayer perceptron - activation functions (2 pts)

Construct manually a multilayer perceptron that computes the product of two positive real values (with no error – the outputs should return exactly the correct answer). The network has two inputs and one output, and you can use as many hidden layers (with arbitrary number of nodes) as you need to solve the problem. You are also free to use any monotonic activation functions, and remember to specify also the weights of your network.

The most obvious solution for the problem does not work for negative inputs. If you came up with such a solution, explain how it could be extended to support negative inputs as well. If you are not able to specify the exact network, you can describe the rough idea.

3 Character classification with MLP (2 pts, programming)

In this exercise we will be using Jupyter Python notebooks, which is a nice way of showing a Python program together with documentation, all in an interactive way - you can change the code and rerun the changed parts. For the machine learning part we will be using the Keras deep learning library: https://keras.io/.

First, install Jupyter so that you can run a Python notebook. General instructions can be found here: http://jupyter.org/install.html, but since you also need some dependencies for the notebook, it is perhaps easiest to do it like this in Linux (first `cd` to the desired directory):

```
virtualenv --system-site-packages -p /usr/bin/python3 env
source env/bin/activate

pip install --upgrade pip
pip install --upgrade ipython
pip install tensorflow
pip install jupyter keras seaborn pydot3 graphviz
```

Next time you want to run it, you just need to enter the virtual environment:

```
source env/bin/activate
```

If the above command doesn't work, try replacing `pip` with `pip3.5`. Your system of course needs to have python and virtualenv installed beforehand.

Next, download the notebook for this exercise from here: https://www.cs.helsinki.fi/u/mvsjober/misc/aml-ex7-mlp.ipynb. Put the notebook file into the same directory and start Jupyter from there:

```
jupyter notebook
```

Now you can read through the instructions, and run each code block or "cell" by selecting it and clicking the "run cell" button, or pressing shift-Enter on the keyboard.

Typically each code cell will produce some output, also read this and see that it makes sense. Once you get to the part where we create the actual model, you will have to fill in your own code in the notebook. You can always go back, change the code and rerun the cell if you change things.

Try to reach at least 90% accuracy with the MLP on the test dataset. Note that we are using the notMNIST dataset here, which is a bit harder than MNIST.

4 Character classification with CNN (3 pts, programming)

This exercise continues on the previous exercise. The only difference is we will be using convolutional neural nets instead of MLPs.

The notebook for this exercise can be found here: https://www.cs.helsinki.fi/u/mvsjober/misc/aml-ex7-cnn.ipynb.

The setup code is almost the same as for the MLP case, except that the network now gets its input as a 2D matrix, instead of it being flattened to a vector as in the MLP case. Can you explain what advantage this might have for the character classification case?

You should be able to get a better result on the test data than with the MLP model.