582744 Advanced Course in Machine Learning

Exercise 6 **Due May 3, 10:00 AM**

Rules:

1. Your submission is composed of two parts: (a) A single PDF file containing your answers to all questions, including both the pen and paper questions and the written answers and various plots for the programming questions. (b) a single compressed file (zip or tar.gz) containing your code (and nothing else). If your code is in a single file, it can be sent also a plain source code.
2. The submission should be sent directly to BOTH Arto (`arto.klami@cs.helsinki.fi`) and Aditya (`aditya.jitta@cs.helsinki.fi`).
3. All material must be renamed to your student ID. Always mention your name and student ID also in the written report.
4. The subject field of your email should be [AML][your student ID][exercise 6].
5. 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.
6. The pen and paper exercises can alternatively be returned in paper form during the Tuesday lecture.

This set of exercises is due on Tuesday May 3rd, before 10:00 AM.

1 Decision trees and forests (12 pts, programming)

In this exercise we implement a simplified decision tree and then build a random forest out of those. The tree we consider is a decision stump, a tree of exactly one decision node. Implement the decision stump so that:

- It picks a random input feature to be used for the decision
- It computes the information gain (of the class labels) for all possible threshold for that input feature
- It picks the decision threshold with the highest information gain
- When used as a classifier, it outputs the probabilities for each class

The information gain is defined as the reduction in class entropy after making the decision. If the initial set of samples is denoted by D (with $|D|$ samples) and the decision rule splits it into two parts L and R, then the information gain for that rule is

$$H(Y \in D) - \frac{|L|}{|D|} H(Y \in L) - \frac{|R|}{|D|} H(Y \in R).$$

Here the notation $H(Y \in D)$ refers to estimating the entropy of the class variable Y based on the subset of samples that belong to the set D.

Now construct a random forest out of the decision stumps by simply learning M such stumps, using randomly selected subset of n samples for training each of them. Average the predictions over the stumps.

Finally, apply the classifier on classifying the MNIST digits studied in Exercise 4. Train the model on the first 5000 samples and test on the 5000 samples following those. Note that some input features in MNIST are identically zero – it is a good idea to drop those before running the algorithm.

Plot the training and validation losses as a function of M (you should only train $M_{\max}$ models once and then compute only the errors for smaller emsembles), using the classification error as the loss (error of one if the prediction is incorrect). Is the forest you created a good classifier? If not, can you think of how to improve it? You can here use $n = 100$ samples for training each stump, unless you can think of a better choice.

Hints:

- It is a good idea to represent the class labels as ten-dimensional binary vectors where 1 indicates the class label.
- You probably want to sort the inputs values for finding the decision threshold. Then you can compute the entropies for all possible thresholds in one loop by using cumulative counts of the class indicator vectors.
- Remember to have some safeguards for empty leaves – perhaps output a uniform distribution over the classes
- You do not need any actual data structures for trees here, since we only consider individual decisions

2 Multilayer perceptron - structure and parameters (4 pts)

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?

3 Multilayer perceptron - activation functions (4 pts)

Construct manually a multilayer perceptron that computes the product of two positive real values (with no error – the outpus 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.

4 Multilayer perceptron - the whole model (4 pts)

Use the playground tool playground.tensorflow.org to create a MLP for analysing the fourth data set (the two spirals) without adding noise.

Create a network that uses only the raw inputs x_1 and x_2 and reaches test loss of 0.2. Explain the parameters you used in the final network: The network structure, the activation function, and the parameters for the optimization algorithm. Give also the rough number of iterations needed for reaching the solution. Describe briefly the process you used to find the parameters.

How small error you are able to reach? Would using other inputs help?