582744 Advanced Course in Machine Learning

Exercise 7: Bonus exercise

Due Monday May 8, 23:55 AM

This set of exercises is voluntary, to be used for compensating missing submissions or points of earlier exercises. The points are added to the overall point tolta, but your total points cannot exceed 6x8=48 – if you already have full marks then there is no need to return these exercises.

Note that the deadline is earlier than in the regular schedule (and is now strict – the Moodle submission closest at the deadline), so that the model solutions can be released already on May 9th, two days before the exam.

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 Monday May 8, before 23:55 AM.

1 Playing with RNNs (3 points, programming)

In this exercise we will be using Jupyter Python notebooks again together with the Keras deep learning library. See Exercise 6 for more instructions on how to set it up.

If you already did the setup for Exercise 6, you just need to go to the same directory and type:

```
source env/bin/activate
jupyter notebook
```

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

The idea of this exercise is to create a simple LSTM model that learns to predict the next character based on the previous ones. Please try at least doing the alphabet, and a simple sentence (included in the code). If you have time you can play with some real texts, but this is not required. For the toy examples you should be able to reach more or less perfect accuracy.

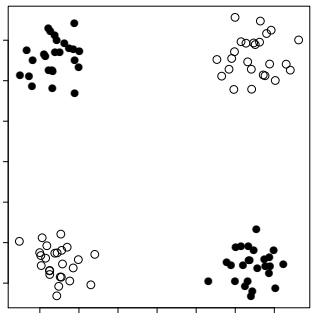
2 Sparsity (2 points)

Consider the regularized loss function $L(\theta_1, \theta_2) = (\theta_1 - 2)^2/4 + (\theta_2 - 2)^2 + \lambda|\boldsymbol{\theta}|$ (l_1-regularization), where $\boldsymbol{\theta} = [\theta_1, \theta_2]$.

Characterize the sparsity of the solution as a function of the regularization parameter λ. That is, find out for which values of λ the optimal solution has two non-zero elements, for which values it has one non-zero element, and for which values the result is the zero vector.

3 AdaBoost (3 points)

Consider the two-dimensional binary classification problem depicted below, corresponding to the XOR-problem. Linear classifiers ($\text{sign}(w_1 x_1 + w_2 x_2 + b)$) are here weak classifiers – they are clearly better than random but do not solve the whole classification problem.



Apply the AdaBoost algorithm (Algorithm 16.2 on page 561 of the course book) manually to construct the final classifier $f(\mathbf{x}) = \sum_{m=1}^{3} \beta_m f_m(\mathbf{x})$. That is, learn the first three weak classifiers to be used as parts of the final ensemble. Remember that the weights β_m are determined by the error rate of the classifier using $\beta_m = \frac{1}{2} \log \frac{1-\text{err}_m}{\text{err}_m}$ and the weights for the incorrectly classified training examples are multiplied by e^{β_m} for training the next classifier.

You should not use any learning algorithm to determine the decision boundaries – you can easily determine them manually once you know the weights for the samples (breaking ties arbitrarily). Besides drawing the decision boundaries, characterize how the weights of the samples change and provide the classifier weights β_m. What is the final classifier and how good is it? Would using more base learners improve it?

Hint: Each group has exactly 25 data points – no need to count. The axes are scaled so that the visual separation corresponds to the actual distance.