582744 Advanced Course in Machine Learning

Exercise 7 **Due May 17, 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 7].
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 17th, before 10:00 AM.

1 Restricted Boltzmann machine (9 pts)

(a) The lecture slides state that the conditional density $p(\mathbf{h}|\mathbf{v}, W)$ of a restricted Boltzmann machine

$$p(\mathbf{v}, \mathbf{h}|\mathbf{W}) = \frac{1}{Z(\mathbf{W})} e^{\mathbf{v}^T \mathbf{W} \mathbf{h}}$$

for vectors of binary variables h_k and v_r is given by

$$p(\mathbf{h}|\mathbf{v}, \mathbf{W}) = \prod_k \text{Ber}(h_k|\text{sigm}(W_{:,k}^T \mathbf{v})).$$

Derive this result. The conditional density for $\mathbf{v}$ naturally follows directly since the formulation is symmetric.

(b) Any non-negative function $f(x|\theta)$ (here over discrete univariate x for simplicity) can be normalized to be a density using

$$p(x|\theta) = \frac{f(x|\theta)}{Z(\theta)} = \frac{f(x|\theta)}{\sum_x f(x|\theta)}.$$

We can compute the derivative of the log-partition function as

$$\frac{\delta \log Z(\theta)}{\delta \theta} = \sum_x p(x|\theta) \frac{\delta \log f(x|\theta)}{\delta \theta}.$$

In other words, if we know how to compute the gradient of the function $\log f(x|\theta)$ we get the gradient for the log-normalizer by computing its expectation over $p(x|\theta)$. Prove this.

(c) Verify that for the case of RBM the result of (b) gives the expression

$$\frac{\partial \log Z}{\partial W_{rk}} = \sum_{\hat{\mathbf{v}}} \sum_{\hat{\mathbf{h}}} p(\hat{\mathbf{v}}, \hat{\mathbf{h}}|\mathbf{W}) \hat{v}_r \hat{h}_k$$

shown on the lecture slides.

2 Dropout (6 pts, programming)

Dropout is a regularization technique that removes nodes from a neural network during training time, so that for each minibatch of stochastic gradient descent we randomly select which nodes to omit. We can use dropout as a regularization technique also for linear models, by interpreting the model as MLP with no hidden layers. The inputs are then part of the network as well and we can apply dropout for that layer.

In Exercise set 2 you already implemented stochastic gradient descent for linear regression (if you didn't, look at the model solution). Extend that solution by adding the dropout regularization – for each mini-batch you should randomly determine for each input dimension whether to use it for this batch or not, keeping each dimension with probability p.

Apply the model for the data set https://www.cs.helsinki.fi/aklami/teaching/AML/train_7_2.csv which consists of 100 examples represented by 200 dimensions; the 201th element on each line is the output variable y. Run the algorithm until convergence (just use high enough number of iterations and some reasonable step-size) and evaluate the validation error on the data https://www.cs.helsinki.fi/aklami/teaching/AML/valid_7_2.csv (which has 500 samples).

Plot the validation error as a function of the parameter p controlling the dropout rate. Compare the solution to that of standard ridge regression, plotting the validation error as a function of the regularization parameter λ. Does dropout help avoiding overfitting? Is it better or worse than ridge regression?

3 TensorFlow and convolutional neural networks (9 pts, programming)

Install TensorFlow according to the instructions at https://www.tensorflow.org/versions/r0.8/get_started/os_setup.html.

Go through the tutorial at https://www.tensorflow.org/versions/r0.8/tutorials/mnist/pros/index.html#deep-mnist-for-experts to build a convolutional neural network that solves the MNIST data classification problem with very high accuracy. Try the model out and plot the training and test errors as a function of the iteration (evaluated after every 100 iterations or so) – describe how the errors behave. Would early-stopping be useful here?

Now modify the network by changing and adding some layers. The main purpose of this exercise is to get familiar with the structure of layers and weights in a convolutional neural network, as well as the syntax for TensorFlow. Hence, you need not come up with a network that would solve the problem better. In fact, it is quite hard as the network already reaches 99% accuracy and the best solution are not much better – see http://yann.lecun.com/exdb/mnist/ and http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#4d4e495354.

Implement the following changes and report the accuracy of the resulting model:

- Reduce the depth of the second convolutional layer to just 16 filters.
- Add another convolutional layer right before that – it should be after the first pooling layer but before the second convolutional layer. Use 32 filters of size 3x3 and ReLU activation.
- Reduce the size of the fully connected layer to 256 units.
- Add another fully connected layer after that with 64 units.

Did training become slower of faster? Why?

In case you are interested on a more challenging data set, you can also check the CIFAR-10 tutorial https://www.tensorflow.org/versions/r0.8/tutorials/deep_cnn/index.html#cifar-10-model.