Assignment 3: Deep Learning, Graphical Models

Due November 13 at 11:59pm
52 marks total

This assignment is to be done individually.

Important Note: The university policy on academic dishonesty (cheating) will be taken very seriously in this course. You may not provide or use any solution, in whole or in part, to or by another student.

You are encouraged to discuss the concepts involved in the questions with other students. If you are in doubt as to what constitutes acceptable discussion, please ask! Further, please take advantage of office hours offered by the instructor and the TA if you are having difficulties with this assignment.

DO NOT:

- Give/receive code or proofs to/from other students
- Use Google to find solutions for assignment

DO:

- Meet with other students to discuss assignment (it is best not to take any notes during such meetings, and to re-work assignment on your own)
- Use online resources (e.g. Wikipedia) to understand the concepts needed to solve the assignment

---

1 Graphical Models (22 marks)

Consider the problem of determining whether a local high school student will attend SFU or not. Define a boolean random variable A (true if the person will attend SFU), discrete random variables L (maximum of parents' education level: can take values o for non-university or u for university) and G (current provincial government: l for Liberal Party, d for NDP), and continuous valued random variables E (current provincial economy size) and T (SFU tuition level).

1. **4 marks**. Draw a simple Bayesian network for this domain.
2. **2 marks**. Write the factored representation for the joint distribution $p(A, L, G, E, T)$ that is described by your Bayesian network.
3. **8 marks**. Supply all necessary conditional distributions. Provide the type of distribution that should be used and give rough guidance / example values for parameters (do this by hand, educated guesses).
4. **8 marks**. Suppose we had a training set and wanted to **learn** the parameters of the distributions using maximum likelihood. Denote each of the N examples with its values for each random variable by $\boldsymbol{x}_n = (a_n, l_n, g_n, e_n, t_n)$. The training set is $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$.

   Which elements of the training data are needed to learn the parameters for $p(A|pa_A)$? Why?[1]

   Start by writing down the likelihood and argue from there.

[1] $pa_A =$ parents of A

2 Neural Networks (30 marks)

In this question you will implement backpropagation for training a neural network with one hidden layer[2].

Start by downloading the code and data from the course website.

- Take a look at the code, starting with `nntrain.m`. A data structure for representing a neural network and its weights is provided, as is code for feeding examples through the network to compute activations and node values.

- Fill in the code for computing `dW2`, the partial derivatives of the error on one sample with respect to the last layer. With the softmax in the output layer, for weights w_{kj} connected to an output node, the derivative is given by

  $$\frac{\partial}{\partial w_{kj}} E_n = \delta_k z_j = (y_k - t_k) z_j \tag{1}$$

  where y_k is the current value of node k, and t_k is the target (training ground truth) value for node k on the n^{th} example. z_j is the current value of hidden node j.

  Try running `nntrain.m` – note that the training process will still be successful even without updates for the hidden layer weights, essentially you are training a set of perceptrons.

- Fill in the code for computing `dW1`, the partial derivatives of the error on one sample with respect to the hidden layer. With the logistic activation functions, for weights w_{kj} connected to a hidden node, the derivative is given by

  $$\frac{\partial}{\partial w_{ji}} E_n = \delta_j x_i \tag{2}$$
  $$\delta_j = h(a_j)(1 - h(a_j)) \cdot \sum_k (w_{kj}\delta_k) \tag{3}$$

  where $h(a_j)$ is the output of the logistic, and δ_k is the error at output node k ($y_k - t_k$, as above). x_i is i^{th} component of the input for the training example.

We will now make some changes, to illustrate an issue involved in training neural networks.

1. Run `nntrain.m` with all the updates filled in. Put a copy of the training/testing error plot in your report. Take a look at `output.html` to see examples of your classifier's performance. Everything should look good – training accuracy should go to 1, and test accuracy should be around 0.8.

2. Uncomment line 40 of `nntrain.m` (the line `NN(1).weights = rand(D+1,H);`). In your report, state what you just changed about the training procedure.

[2] If you want a software package to use in a project, I suggest Theano `http://deeplearning.net/software/theano/` or Caffe `http://caffe.berkeleyvision.org/`

3. Run `nntrain.m` again, with this change. Figure out what went wrong, and explain in your report. Try running one iteration of stochastic gradient descent, and examining variables. You might find the MATLAB debugger helpful (see `help dbstop`, e.g. `dbstop in nntrain at 74`)

Submitting Your Assignment

The assignment must be submitted online at `https://courses.cs.sfu.ca`. You must submit two files:

1. An assignment report in **PDF format**, called `report.pdf`. This report must contain the solutions to question 1 as well as the figures / explanations requested for 2.
2. `nntrain.m`.