Assignment 3: Neural Networks and Graphical Models

Due November 7 at 11:59pm
84 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

---

Question 1 (10 marks)

Consider the sigmoidal activation function for a node in a neural network:

$$g(a) = \frac{1}{1 + \exp(-a)}$$

Show that the derivative of this activation function takes the simple form:

$$\frac{d}{da} g(a) = g(a)(1 - g(a))$$

This fact will be used in Question 5, below.

Question 2 (14 marks)

Consider the car insurance Bayesian network on slide 33 of `chapter8a_slides.pdf`. Answer the following questions.

1. (2 marks) Are *Age* and *SocioEcon* independent?
2. (3 marks) *Age* (driver's age) and *Mileage* (mileage on vehicle) are independent. Do you think this is a reasonable assumption? Why or why not?
3. (4 marks) The node for *ExtraCar* (whether the person owns another car), has no children, and seems to hang off the rest of the network. What purpose does it serve? For instance, if we are trying to determine *MedicalCost*, is *ExtraCar* useful? How would it affect this determination?
4. (5 marks) Total insurance costs could be modeled as the sum of *MedicalCost*, *LiabilityCost*, and *PropertyCost*. Suppose you are only able to ask a prospective client his/her age and vehicle year (*VehicleYear*).

   Write an expression for the expected total insurance cost of the prospective client. You do not need to simplify this, just write down the expression for the **expectation** that should be computed.

Question 3 (10 marks)

Consider learning the parameters for a Bayesian network with a training set $(\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N)$, where each $\boldsymbol{x}^i = (x_1^i, \ldots, x_D^i)$ is a vector containing values for all random variables in the network.

Let $\boldsymbol{\theta}_k$ be the parameters for the distribution $p(x_k | pa_k; \boldsymbol{\theta}_k)$. E.g. $\boldsymbol{\theta}_k = (\mu_k, \sigma_k)$ if this is a Gaussian distribution.

Show that learning the parameters for the network according to the maximum likelihood criterion can be done by separately learning the parameters $\boldsymbol{\theta}_k$ of each conditional distribution using the relevant components of the training data.

Question 4 (50 marks)

In this question you will implement backpropagation for training a neural network with one hidden layer[1].

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 outputs.
- 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 output layer, for weights w_{kj} connected to an output, 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 output node k, and t_k is the target (training ground truth) value for output 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

$$\frac{\partial}{\partial w_{ji}} E_n = \delta_j x_i \tag{2}$$

$$\delta_j = g'(a_j) \cdot \sum_k (w_{kj} \delta_k) \tag{3}$$

  where $g'(a_j)$ is the derivative of the output of the j^{th} node (with respect to (a_j), answer in Question 1), 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.

[1] 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/`

2. Comment out line 39 of `nntrain.m` (the line `NN(1).weights = eye(D+1,H);`). **In your report, state what you just changed about the training procedure.**

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. You must create an assignment report in **PDF format**, called `report.pdf`. This report must contain the solutions to questions 1-3 as well as the figures / explanations requested for 4.

2. Your modified `nntrain.m` with the filled-in backpropagation code.