SMLDM HT 2014 - Part C Problem Sheet 6

1. Recall the definition of a 1 hidden layer neural network for binary classification in the lectures. The objective function is:

$$J = -\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) + \frac{1}{2} \sum_{jk} C|W_{jk}^h|^2 + \frac{1}{2} \sum_{k} C|W_k^o|^2$$

and the network definition is:

$$\hat{y}_i = s\left(b^o + \sum_{k=1}^{m} W_k^o h_{ik}\right) \qquad h_{ik} = s\left(b_k^h + \sum_{j=1}^{p} W_{jk}^h x_{ij}\right)$$

   (a) Verify that the derivatives needed for gradient descent are:

$$\frac{dJ}{dW_k^o} = CW_k^o + \sum_{i=1}^{n} (\hat{y}_i - y_i) h_{ik}$$
$$\frac{dJ}{dW_{jk}^h} = CW_{jk}^h + \sum_{i=1}^{n} (\hat{y}_i - y_i) W_k^o h_{ik}(1 - h_{ik}) x_{ij}$$

   (b) Suppose instead that you have an L layer neural network for binary classification, with each hidden layer having m neurons with logistic nonlinearity. Define carefully the network, giving the parameterization of each layer, and derive the backpropagation algorithm to compute the derivatives of the objective with respect to the parameters. You may ignore bias terms for simplicity.

2. A **mixture of experts** is a type model in which a number of experts "compete" to predict a label.

   Consider a regression problem with dataset $(x_i, y_i)_{i=1}^n$ and $y_i \in \mathbb{R}$. We have E experts, each being a parametrized function $f_j(x; \theta_j)$, for $j = 1, \ldots, E$. For example each expert could be a neural network. Each expert $f_j(x; \theta_j)$ tries to predict the response y corresponding to data vector x.

   (a) A simple mixture of experts model uses as it's objective function

$$J(\pi, \sigma^2, (\theta_j)_{j=1}^E) = \sum_{i=1}^{n} \log \sum_{j=1}^{E} \pi_j e^{-\frac{1}{2\sigma^2} \|f_j(x_i;\theta_j) - y_i)\|^2}$$

   where $\pi = (\pi_1, \ldots, \pi_E)$ are mixing proportions and σ^2 is a parameter.

   Relate the objective function to the log likelihood of a mixture model where each component is a conditional distribution of Y given $X = x$.

   (b) Differentiate the objective function with respect to θ_j, interpreting the computation of the derivative as a generalized EM algorithm, where in the E step the posterior distribution is computed, and in the M step gradient descent is used to update the expert parameters θ_j.

   (c) A mixture of experts allows each expert to specialize in predicting the response in a certain part of the data space, with the overall model having better predictions than any one of the experts.

However to encourage this specialization, it is useful also for the mixing proportions to depend on the data vectors x, i.e. to model $\pi_j(x;\phi)$ as a function of x with parameters ϕ. The idea is that this **gating network** controls where each expert specializes. To ensure $\sum_{j=1}^{E}\pi_j(x;\phi)=1$, we can use the softmax nonlinearity:

$$\pi_j(x;\phi)=\frac{\exp(g_j(x;\phi_j))}{\sum_{\ell=1}^{E}\exp(g_\ell(x;\phi_\ell))}$$

where $g_j(x;\phi_j)$ are parameterized functions for the gating network.

The previous generalized EM algorithm extends to this scenario easily. Describe what changes Derive a gradient descent learning update for ϕ_j.

3. In this question you will investigate issues of fitting neural networks using the `nnet` library in R.

   We will use the USPS dataset of handwritten digits, which you can obtain from the course webpage. Each handwritten digit is 16×16 in size, so that data vectors are $p=256$ dimensional. There are 2000 digits (200 digits of each class) in each of the training set and test set.

   On the course webpage is also an R script which trains a 1 hidden layer neural network with $S=10$ hidden units for $T=10$ iterations, reports the training and test errors, runs it for another 10 iterations, and collects the training and test errors. You will find the documentation for the `nnet` library useful: `http://cran.r-project.org/web/packages/nnet/nnet.pdf`.

   To make computations quicker, the script down-samples the training set to 200 cases, by using only one out of every 10 training cases.

   (a) Edit the script to report the training and test error after every iteration of training the network. Use networks of size $S=10$ and up to $T=100$ iterations. Plot these as a function of the number of iterations. Comment on any salient features of the learning curve, and elaborate on whether the learning curve agrees with what you learnt in class.

   (b) Edit the script to vary the size of the network, reporting the training and test errors for network sizes $S=1,2,\ldots,10,20,30,40,50$. Use $T=100$. Plot these as a function of the network size. Comment on any salient features of the learning curve, and elaborate on whether the learning curve agrees with what you learnt in class.