SB2b HT 2017 - Problem Sheet 2

Due date: Friday, 10 February at 5pm in the pigeon hole in the stats department

1. Consider using logistic regression model
$$p(Y = y|X = x) = s(y(a + b^\top x))$$
for the conditional distribution of binary labels $Y \in \{+1, -1\}$ given data vectors X.

   (a) Suppose that the data is linearly separable, i.e., there is a hyperplane separating the two classes. Show that the maximum likelihood estimator is ill-defined.

   (b) Suppose the first entry $X^{(1)}$ in X is binary, i.e. it takes on only values 0 or 1. Suppose that in the dataset, whenever $y_i = -1$, the corresponding entry $x_i^{(1)} = 0$, but there are data cases with $y_i = +1$, and $x_i^{(1)}$ taking on both values. Show that the maximum likelihood estimator of b_1 is ∞, but that the dataset need not be linearly separable.

2. Consider a mixture model with K components and mixing proportions $\Pi = (\pi_1, \ldots, \pi_K)$, where $\sum_{k=1}^K \pi_k = 1$. A data sample drawn from this model is denoted as $(Z_i, X_i)_{i=1}^N$, where Z_i is the *hidden* component to which sample i belongs, and X_i is the *observable* variable, which follows the following conditional distribution
$$\mathbb{P}_\Theta(X_i = x|Z_i = k) = 2\theta_k\, x\, e^{-\theta_k\, x^2},\ x \geq 0,$$
where $\Theta = (\theta_1, \ldots, \theta_K)$.

   (a) Put the distribution $\mathbb{P}_\Theta(X_i = x|Z_i = k)$ in the exponential family form highlighting the *sufficient statistic* and the *log-partition* function.

   (b) Given the dataset $\mathcal{D} = \{X_i\}_{i=1}^N$, write down the log-likelihood explicitly as a function of the parameters Π and Θ.

   (c) We want to estimate the model's parameters by maximizing the log-likelihood using the EM algorithm. Derive explicitly the EM update equations.

3. (a) Consider a dataset $\mathcal{D} = \{X_i\}_{i=1}^N$ whose data points are drawn independently from the following distribution
$$\mathbb{P}(X = x \,|\, \theta, k) = \theta\, k^\theta\, x^{-\theta-1},\ x \geq k, \theta > 1, k > 2.$$
Find the Maximum Likelihood Estimate (MLE) of θ and k.

   (b) The *method of moments* is another estimation technique that operates by equating the sample moments with the distribution moments and solving the following system of equations for the distribution's parameters:
$$\mathbb{E}[X] = \frac{1}{N}\sum_{i=1}^N X_i$$
$$\mathbb{E}[X^2] = \frac{1}{N}\sum_{i=1}^N X_i^2$$
$$\vdots$$
$$\mathbb{E}[X^m] = \frac{1}{N}\sum_{i=1}^N X_i^m.$$

Now consider a dataset $\mathcal{D} = \{X_i\}_{i=1}^N$ whose data points are drawn independently from a uniform distribution on $[0, \theta]$. Find both the MLE and the method of moments estimates for θ.

4. Recall the definition of a one-hidden layer neural network for binary classification in the lectures. The objective function is L_2-regularized log loss:

$$J = -\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i) + \frac{\lambda}{2}\left(\sum_{jl}(w_{jl}^h)^2 + \sum_{l}(w_l^o)^2\right)$$

and the network definition is:

$$\hat{y}_i = s\left(b^o + \sum_{l=1}^{m} w_l^o h_{il}\right), \qquad h_{il} = s\left(b_l^h + \sum_{j=1}^{p} w_{jl}^h x_{ij}\right),$$

with transfer function $s(a) = \frac{1}{1+e^{-a}}$.

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

$$\frac{\partial J}{\partial w_l^o} = \lambda w_l^o + \sum_{i=1}^{n}(\hat{y}_i - y_i)h_{il},$$
$$\frac{\partial J}{\partial w_{jl}^h} = \lambda w_{jl}^h + \sum_{i=1}^{n}(\hat{y}_i - y_i)w_l^o h_{il}(1 - h_{il})x_{ij}.$$

   (b) Suppose instead that you have a neural network for binary classification with L hidden layers, each hidden layer having m neurons with logistic transfer function. Give the parameterization for each layer, and derive the backpropagation algorithm to compute the derivatives of the objective with respect to the parameters. For simplicity, you can ignore bias terms.

5. In this question you will investigate fitting neural networks using the `nnet` library in R. We will train a neural network to classify handwritten digits 0-9. Download files `usps_trainx.data`, `usps_trainy.data`, `usps_testx.data`, `usps_testy.data` from `http://www.stats.ox.ac.uk/~flaxman/sml17/data/`.

   First, you'll need to make sure that the nnet library is installed. Use `install.packages("nnet")` and select a package repository (CRAN mirror) nearby.

   You can load the data files using `read.table(...)`. If you need more help getting started, consult slide 59 of the notes available at `http://www.stats.ox.ac.uk/~evans/statprog/prog_slid.pdf` (linked from the course website).

   Each handwritten digit is 16×16 in size, so that data vectors are $p = 256$ dimensional and each entry (pixel) takes integer values 0-255. There are 2000 digits (200 digits of each class) in each of the training set and test set.

   You can view the digits with:

```
plot_digit = function(x,y) {
  # input x should be a numeric vector of length 256
  image(matrix(as.matrix(x), 16,16), col=grey(seq(0,1,length=256)),
        main=sprintf("Label: %d",y))
}
plot_digit(trainx[500,],trainy[500,])
```

Download the R script `nnetusps.R` from the course webpage. The script 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 reports the new training and test errors. To make computations quicker, the script down-samples the training set to 200 cases, by using only one out of every 10 training cases. You will find the documentation for the `nnet` library useful: `http://cran.r-project.org/web/packages/nnet/nnet.pdf`.

(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 the training and test errors as functions of the number of iterations. Discuss the results and the figure.

(b) Edit the script to vary the size of the network, reporting the training and test errors for network sizes $S = 1, 2, 3, 4, 5, 10, 20, 40$. Use $T = 25$ iterations. Plot these as a function of the network size. Discuss the results and the figure.

(c) **Optional.** Can you get a significant performance increase by adding multiple layers? To investigate, you'll need to switch to the package `neuralnet`. Here is code to run `neuralnet` with one hidden layer, check the documentation (in R type `?neuralnet`) to see how to add more layers.

```
library(neuralnet)
data = cbind(class.ind(trainy),trainx)
names(data)[1:10] = paste0("label",names(data[1:10]))
fml <- as.formula(paste(paste(paste0('label',0:9),collapse=" + "),
  ' ~ ' ,paste(paste0("V",1:256),collapse='+')))

net <- neuralnet(fml, data, err.fct="ce", linear.output=FALSE)
```