Franklin Hu, Sunil Pedapudi
CS 194-10 Machine Learning
Fall 2011
Assignment 4

1. Linear neural networks

   (a) Suppose we have a three layer neural network with one input layer x, one hidden layer h, and one output layer y. Each layer can be expressed as a vector of the values of the nodes in that layer. For example,

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

   Assume that each neural node has its own set of weights $\mathbf{w_i}$ where i is the node index. We can express the value of a particular output in terms of the hidden layer

$$y_k = g(\mathbf{h})$$

   Since we are only considering linear activation functions, we can write this equation in terms of a constant multiplied by the weighted sum of inputs

$$y_k = c_{y_k} \cdot \mathbf{w_{y_k}} \cdot \mathbf{h}$$

   where c_{y_k} is the constant multiplier of y_k, w_{y_k} is the set of weights for y_k, and h is the vector of hidden nodes.
   Similarly, we can express the value of each node in the hidden layer in terms of the inputs.

$$h_j = c_{h_j} \cdot \mathbf{w_{h_j}} \cdot \mathbf{x}$$

   Now, we can see that the output layer nodes can simply be written in terms of the inputs without the hidden layer. For a particular output node:

$$\begin{aligned} y_k &= c_{y_k} \cdot \mathbf{w_{y_k}} \cdot \mathbf{h} \\ &= c_{y_k} \cdot \mathbf{w_{y_k}} \cdot \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{pmatrix} \\ &= c_{y_k} \cdot \mathbf{w_{y_k}} \cdot \begin{pmatrix} c_{h1} \cdot \mathbf{w_{h_1}} \cdot \mathbf{x} \\ c_{h2} \cdot \mathbf{w_{h_2}} \cdot \mathbf{x} \\ \vdots \\ c_{hn} \cdot \mathbf{w_{h_n}} \cdot \mathbf{x} \end{pmatrix} \\ &= c_{y_k} \cdot (\mathbf{w_{y_k}} \cdot \mathbf{c_h} \cdot \mathbf{I} \cdot \mathbf{w_h}) \cdot \mathbf{x} \end{aligned}$$

   where c_h is a vector of the constant weight for each hidden node, I is the identity matrix, and w_h is a matrix of the weight vectors of the hidden nodes. Thus we can define a new weight vector $\mathbf{u_{y_k}}$ for the output node y_k

$$\mathbf{u_{y_k}} = \mathbf{w_{y_k}} \cdot \mathbf{c_h} \cdot \mathbf{I} \cdot \mathbf{w_h}$$

   We can thus simply compute the value of y_k in terms of x.

(b) For an arbitrary number of hidden nodes, the same computation can be done. Suppose we have h hidden layers with $\mathbf{h_1}$ having $\mathbf{x}$ as inputs and $\mathbf{y}$ having $\mathbf{h_n}$ as inputs. In this case, consider the sub-network of $\mathbf{h_2}$, $\mathbf{h_1}$, and $\mathbf{x}$. Using the technique in part (a), we can write the expression for $\mathbf{h_2}$ in terms only of $\mathbf{x}$. The resulting expression

$$\mathbf{h_{2i}} = c_{h_2} \cdot \mathbf{u_{h_2}} \cdot \mathbf{x}$$

still has a linear activation function and therefore this technique generalizes to the remaining hidden nodes. We repeat this process for all the hidden nodes and can thus write the expression for any output node y_k

$$y_k = c_{y_k} \cdot \mathbf{w_{y_k}} \cdot \left(\prod_{\mathbf{i=1}}^{\mathbf{h}} \mathbf{c_{h_i}} \cdot \mathbf{I} \cdot \mathbf{w_{h_i}} \right) \cdot \mathbf{x}$$

(c) For the case when $h \ll n$, a neural net with the hidden layer will do $O(hn)$ computations to find the linear combination of the weighted sum of inputs whereas without the hidden layer, as shown in (a), the output is only dependent on x. This computations is $O(n)$, so we save those $h-1$ other computations over the inputs.

2. ML estimation of exponential model
Knowing

$$P(x) = \frac{1}{b} e^{-\frac{x}{b}}$$

(a) We write the likelihood function given x_i as

$$\begin{aligned}
\mathcal{L}(b) &= \prod_{i=1}^{N} \frac{1}{b} e^{-\frac{x_i}{b}} \\
&= \left(\frac{1}{b}\right)^N \prod_{i=1}^{N} e^{-\frac{x_i}{b}} \\
&= \left(\frac{1}{b}\right)^N e^{\sum_{i=0}^{N} -\frac{x_i}{b}} \\
&= \left(\frac{1}{b}\right)^N exp(-\frac{1}{b}\sum_{i=0}^{N} x_i) \\
\text{Let } \bar{x} &= \frac{1}{N}\sum_{i=1}^{N} x_i, \\
\mathcal{L}(b) &= \left(\frac{1}{b}\right)^N exp(-\frac{1}{b} N\bar{x})
\end{aligned}$$

(b) We first find

$$\begin{aligned}
log(\mathcal{L}) &= log\left(\left(\frac{1}{b}\right)^N exp(-\frac{1}{b}N\bar{x})\right) \\
&= log\left(\frac{1}{b}\right)^N + log\left(e^{-\frac{1}{b}N\bar{x}}\right) \\
&= N(log(1) - log(b)) - \frac{1}{b}N\bar{x} \\
&= -Nlog(b) - \frac{1}{b}N\bar{x}
\end{aligned}$$

Then, let $\theta = \frac{1}{b}$, the parameter variable

$$\begin{aligned}\frac{\partial log(\mathcal{L}(\theta))}{\partial \theta} &= \frac{\partial N log(\theta)}{\partial \theta} - \frac{\partial \theta N\bar{x}}{\partial \theta} \\ &= \frac{N}{\theta} - \frac{\partial \theta N\bar{x}}{\partial \theta} \\ &= \frac{N}{\theta} - N\bar{x} \\ &= Nb - N\bar{x}\end{aligned}$$

(c) We aim to maximize $\mathcal{L}$ so,

$$\frac{\partial \mathcal{L}}{\partial \theta} = Nb - N\bar{x} = 0$$

We can solve this to find

$$b = \bar{x} = \frac{1}{N}\sum_{i=0}^{N} x_i$$

3. ML estimation of noisy-OR model

4. Naive Bayes models for spam filtering

(a)

(b) We modified the skeleton code in NBmodel.py. We were not sure what the point was to have each NaiveBayesModel have a pointer to another model and just made this base class the model used. Most of the training and classification code was common between the Boolean and NTF subclasses with the exception of the classification code to calculate the log probabilities.

(c) If classifying ham as spam is c times worse than classifying spam and ham and the model returns a probability p that the message is spam, we can simply calculate the expected value given that probability.
If the cost ratio is c, we assign the following values:

$$\begin{aligned}\text{Value}(Y = \text{SPAM}|\hat{Y} = \text{SPAM}) &= +1 \\ \text{Value}(Y = \text{HAM}|\hat{Y} = \text{HAM}) &= +1 \\ \text{Value}(Y = \text{SPAM}|\hat{Y} = \text{HAM}) &= -c \\ \text{Value}(Y = \text{HAM}|\hat{Y} = \text{SPAM}) &= -1\end{aligned}$$

For correctly classifying an example, we give it value $+1$. For incorrectly classifying spam as ham, assign value -1, and for incorrectly classifying ham as spam, we give it the cost ratio value $-c$. For example, if classifying ham as spam is twice as bad as classifying spam as ham, $c = 2$.
When the model returns a probability p that the example is spam, we do the following:

- If $p > 0.5$, we would normally classify this example as spam.

$$\begin{aligned}\mathbb{E}[\text{Value}] &= \text{Value}(Y = \text{SPAM}|\hat{Y} = \text{SPAM}) \cdot p + \text{Value}(Y = \text{HAM}|\hat{Y} = \text{SPAM}) \cdot (1 - p) \\ &= 1 \cdot p - c \cdot (1 - p) \\ &= p - c + cp\end{aligned}$$

If the expected value is positive, then we go ahead and classify the item as SPAM, and otherwise HAM.

- If $p < 0.5$, we would normally classify this example as ham.

$$\begin{aligned}\mathbb{E}[\text{Value}] &= \text{Value}(Y = \text{HAM}|\hat{Y} = \text{HAM}) \cdot (1-p) + \text{Value}(Y = \text{SPAM}|\hat{Y} = \text{HAM}) \cdot p \\ &= 1 \cdot (1-p) - 1 \cdot p \\ &= 1 - p - p \\ &= 1 - 2p\end{aligned}$$

  Since $p < 0.5$, this expected value is always positive, so we always classify HAM if the model says to.
- If $p = 0.5$, then we just look at the cost ratio. If the cost ratio is more than 1, we classify it as SPAM; if less than 1, HAM; and if equal to 1, we flip a fair coin.

(d) In general, we calcuate the log probability ratio:

$$\frac{P(Y=1|X)}{P(Y=0|X)} \rightarrow \log\frac{P(Y=1|X)}{P(Y=0|X)}$$

If the original ratio is greater than 1, then we classify the example as spam. If it is less than 1, we classify it as ham. If it is equal to one, we flip a fair coin.
When we take the log, we simply adjust the value we compare it to since $log1 = 0$.
For Boolean features, our expression for the likelihood is

$$\begin{aligned}P(Y=1|X) &= P(Y=1) \cdot P(X|Y=1) \\ &= P(Y=1) \cdot \prod_{i=1}^{n} P(x_i|Y=1) \\ &= \alpha\theta_1 \prod_{i=1}^{n} {\theta_{1i}}^{x_i}(1-\theta_{1i})^{1-x_i}\end{aligned}$$

Since we assume that the attributes are independent, we can turn $P(X|Y=1)$ into a product of the individual probabilities.
When we take the log ratio, we get

$$\begin{aligned}\frac{P(Y=1|X)}{P(Y=0|X)} &= \frac{\alpha\theta_1 \prod_{i=1}^{n} {\theta_{1i}}^{x_i}(1-\theta_{1i})^{1-x_i}}{\alpha\theta_0 \prod_{i=1}^{n} {\theta_{0i}}^{x_i}(1-\theta_{0i})^{1-x_i}} \\ &= \frac{\theta_1 \prod_{i=1}^{n} {\theta_{1i}}^{x_i}(1-\theta_{1i})^{1-x_i}}{\theta_0 \prod_{i=1}^{n} {\theta_{0i}}^{x_i}(1-\theta_{0i})^{1-x_i}} \\ \log\frac{P(Y=1|X)}{P(Y=0|X)} &= \log\left(\theta_1 \prod_{i=1}^{n} {\theta_{1i}}^{x_i}(1-\theta_{1i})^{1-x_i}\right) - \log\left(\theta_0 \prod_{i=1}^{n} {\theta_{0i}}^{x_i}(1-\theta_{0i})^{1-x_i}\right) \\ &= \log\theta_1 - \log\theta_0 + \sum_{i=1}^{n} x_i \cdot \log\theta_{1i} + (1-x_i) \cdot \log(1-\theta_{1i}) - \sum_{i=1}^{n} x_i \cdot \log\theta_{0i} - (1-x_i) \cdot \log(1-\theta_{0i})\end{aligned}$$

For NTF, we can make the same independence assumption and derive a log probability ratio

$$\begin{aligned}
\frac{P(Y=1|X)}{P(Y=0|X)} &= \frac{P(Y=1)\cdot P(X|Y=1)}{P(Y=0)\cdot P(X|Y=0)} \\
&= \frac{P(Y=1)\cdot \prod_{i=1}^{n} P(x_i|Y=1}{P(Y=0)\cdot \prod_{i=1}^{n} P(x_i|Y=0} \\
&= \frac{\alpha\theta_1 \prod_{i=1}^{n} \frac{1}{b_i} e^{-\frac{x_i}{b_i}}}{\alpha\theta_0 \prod_{j=1}^{n} \frac{1}{b_j} e^{-\frac{x_j}{b_j}}} \\
&= \frac{\theta_1 \prod_{i=1}^{n} \frac{1}{b_i} e^{-\frac{x_i}{b_i}}}{\theta_0 \prod_{j=1}^{n} \frac{1}{b_j} e^{-\frac{x_i}{b_j}}} \\
\log\frac{P(Y=1|X)}{P(Y=0|X)} &= \log\frac{\theta_1 \prod_{i=1}^{n} \frac{1}{b_i} e^{-\frac{x_i}{b_i}}}{\theta_0 \prod_{j=1}^{n} \frac{1}{b_j} e^{-\frac{x_i}{b_j}}} \\
&= \log\left(\theta_1 \prod_{i=1}^{n} \frac{1}{b_i} e^{-\frac{x_i}{b_i}}\right) - \log\left(\theta_0 \prod_{j=1}^{n} \frac{1}{b_j} e^{-\frac{x_i}{b_j}}\right) \\
&= \log\theta_1 - \log\theta_0 + \sum_{i=1}^{n} \log\left(\frac{1}{b_i} e^{-\frac{x_i}{b_i}}\right) - \sum_{j=1}^{n} \log\left(\frac{1}{b_j} e^{-\frac{x_i}{b_j}}\right) \\
&= \log\theta_1 - \log\theta_0 + \left(\sum_{i=1}^{n} -\log b_i - \frac{x_i}{b_i}\right) - \left(\sum_{j=1}^{n} -\log b_j - \frac{x_j}{b_j}\right)
\end{aligned}$$

where b_i and b_j are the appropriate constants calucated with the closed form solution in question 2 for each feature.