# 10-701 Machine Learning, Spring 2011: Homework 3

Due: Feb 17th, at the beginning of class

**Instructions** There are 2 questions on this assignment. The last question involves coding. Please submit your writeup as 2 **separate** sets of pages according to TAs, with your name and userid on each set.

## 1 Bayes Net [Carl, 40 points]

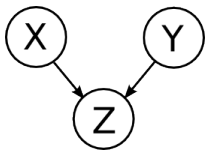


Figure 1: Graphical model for question 1.1

1. In this problem, you will study the behavior of the simple graphical model shown in figure 1, often called a 'collider'. Here, the probability of $Z$ depends on two variables $X$ and $Y$.

   (a) Write the factored distribution represented by this graph. That is, write the distribution as a product of factors each of which could be replaced by entries in the conditional probability tables (CPTs) in order to obtain the probability of an event.

   ★ **SOLUTION:** $P(X,Y,Z) = P(X)P(X)P(Z|X,Y)$

   (b) Assume that $X$,$Y$, and $Z$ are binary. Write the CPTs for a distribution such that $P(X=1) < P(X=1|Y=1,Z=1) < P(X=1|Z=1)$. Compute the values $P(X=1)$,$P(X=1|Y=1,Z=1)$, and $P(X=1|Z=1)$.

   ★ **SOLUTION:** The trick here was to think about examples from class. The example of $X = car\_battery\_dead$, $Y = no\_gas$, $Z = engine\_starts$ almost behaves the way we want: knowing that the car doesn't start makes it more likely that the battery is dead ($P(X=1) < P(X=1|Z=1)$), and knowing that there is no gas makes it less likely that the battery is dead ($P(X=1|Y=1,Z=1) < P(X=1|Z=1)$). To enforce that $P(X=1) < P(X=1|Y=1,Z=1)$, we need to make sure that $Y$ doesn't completely explain away the evidence $Z=1$: i.e., $P(Z=1|X=1,Y=1) > P(Z=1|X=0,Y=1)$. It turns out that the choices of $P(X)$ and $P(Y)$ don't make any difference unless they're 0 or 1. The following distribution works:
   $P(X=1) = .5; P(Y=1) = .5$
   $P(Z=1|X=1,Y=1) = 1$
   $P(Z=1|X=0,Y=1) = .5$
   $P(Z=1|X=1,Y=0) = .5$
   $P(Z=1|X=0,Y=0) = 0$
   Thus, we have $P(X) = .5$, $P(X|Y,Z) = P(X,Y,Z)/P(Y,Z) = .25/(.25+.125) = .66$, $P(X|Z) = P(X,Z)/P(Z) = (.25+.125)/(.25+.125+.125+0) = .75$,

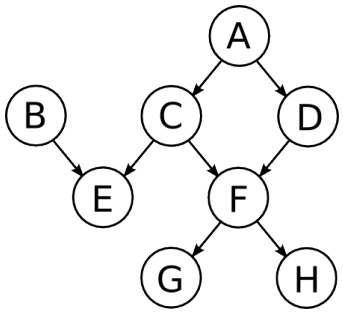


Figure 2: Graphical model for question 1.2

2. Let $P(A, B, C, D, E, F, G, H)$ factorize according to the graph given in figure 2. Prove or disprove whether the following independence properties hold.

   (a) $(A \perp H)$

   ★ **SOLUTION:** **false**: the path $A - D - F - H$ is not blocked.

   (b) $(B \perp G)$

   ★ **SOLUTION:** **true**: all paths from $B$ to $G$ pass through $E$, which is not known and therefore blocks.

   (c) $(B \perp G | E)$

   ★ **SOLUTION:** **false**: the path $B - E - C - F - G$ is not blocked.

   (d) $(C \perp D | G, A)$

   ★ **SOLUTION:** **false**: The path $C - F - D$ is not blocked (note that since we condition on $G$, and $G$ is a descendant of $F$, $F$ does not block.

   (e) $(C \perp D)$

   ★ **SOLUTION:** **false**: The path $C - A - D$ is not blocked since we do not condition on $A$.

   (f) $(B \perp D | C)$

   ★ **SOLUTION:** **true**: for the same reason that $B \perp G$.

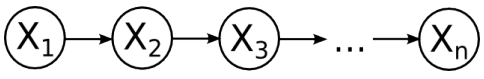


Figure 3: Graphical model for question 1.3

3. Let $\mathbf{X} = \{X_1, ..., X_n\}$ have a distribution $P(\mathbf{X})$ that factorizes according to the graph given in figure 3, and assume each $X_i$ is binary. Design a polynomial-time algorithm which will compute the following

quantities. Use pseudocode for your answer. In your pseudocode, assume that the variables `px1` and `px` already exist, where `px1` is the prior probability $P(X_1 = 1)$, and `px` is an n-1 by 2 dimensional array, where `px[i][j]` is $P(X_{i+1} = 1 | X_{\mathtt{i}} = \mathtt{j} - 1)$, for $\mathtt{j} \in 1, 2$. For this problem, you may assume that the numbers have arbitrary precision, so you do not need to worry about underflow.

(a) Compute $P(X_n = 1 | X_1 = \mathtt{x1})$. Assume that `x1` has already been initialized with the value of $X_1 \in \{0.1\}$.

★ **SOLUTION:** For the following, I will use $P(X_k)$ to denote $P(X_k = 1)$. First, notice that:

$$\begin{aligned} P(X_n|X_1) &= \frac{P(X_n, X_1)}{P(X_1)} \\ &= \frac{\sum_{j\in\{0,1\}} P(X_n, X_{n-1} = j, X_1)}{P(X_1)} \\ &= \frac{\sum_{j\in\{0,1\}} P(X_n|X_{n-1} = j, X_1)P(X_{n-1} = j, X_1)}{P(X_1)} \end{aligned}$$

Using the conditional independence assumptions of our graph and rearranging, this may be rewritten:

$$\sum_{j\in\{0,1\}} P(X_n|X_{n-1} = j)P(X_{n-1} = j, X_1)$$

Thus, assuming we have a linear time algorithm to compute $P(X_{n-1}, X_1)$, then computing $P(X_n, X_1)$ remains linear. The actual code looks something like this:

```
function compute_pxn_giv_x1(px,px1,x1)
  P_xi_giv_x1=x1          //initialize with P(X_1=1|X_1=x1),
                          //which is trivially 0 or 1.
  for i=1 to n
    P_xi_giv_x1=(1-P_xi_giv_x1)*px[i][1]+P_xi_giv_x1*px[i][2]
  end
  return P_xi_giv_x1
end
```

(b) Compute $P(X_1 = 1 | X_n = \mathtt{xn})$. Assume that `xn` has already been initialized with the value of $X_n \in \{0.1\}$.

★ **SOLUTION:** The easiest solution is to use Bayes rule:

$$P(X_1|X_n) = \frac{P(X_n|X_1)P(X_1)}{P(X_n)} = \frac{P(X_n|X_1)P(X_1)}{P(X_n|X_1)P(X_1) + P(X_n|\neg X_1)P(\neg X_1)}$$

We already have expressions for all of these terms, so the final solution is:

```
function compute_px1_giv_xn(px,px1,xn)
  P_xn_giv_x1=compute_pxn_giv_x1(px,px1,1)
  P_xn_giv_not_x1=compute_pxn_giv_x1(px,px1,0)
  P_x1_giv_xn=P_xn_giv_x1*px1/(P_xn_giv_x1*px1 + P_xn_giv_not_x1*(1-px1))
  return P_x1_giv_xn
end
```

## 2 Multi-class Logistic Regression with Applications to Handwritten Digit Recognition [Xi, 60 points]

Recall the multi-class logistic regression model on page 6 of the lecture on Jan 27th. Assume that we have $K$ different classes and each input $\mathbf{x}$ is a $d$ dimensional vector. The posterior probability is given by:

$$\begin{aligned} P(Y=k|X=\mathbf{x}) &= \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{1+\sum_{l=1}^{K-1}\exp(\mathbf{w}_l^T\mathbf{x})} \quad \text{for} \quad k=1,\ldots,K-1 \\ P(Y=K|X=\mathbf{x}) &= \frac{1}{1+\sum_{l=1}^{K-1}\exp(\mathbf{w}_l^T\mathbf{x})}, \end{aligned}$$

where $\mathbf{w}_k^T$ denotes the transpose of the $\mathbf{w}_k$ vector and $\mathbf{w}_k^T\mathbf{x}$ is the inner product of $\mathbf{w}_k$ and $\mathbf{x}$ (i.e., $\mathbf{w}_k^T\mathbf{x} = \sum_i w_{ki}x_i$). To simplify, we *ignore* the constant term $w_{k0}$ in the logistic regression model.

For ease of notation, we can replace the two above expressions for $P(Y|X)$ by a single expression. We do this by introducing a fixed, pseudo parameter vector: $\mathbf{w}_K = \mathbf{0}$. Now for any class label $k$, we simply write

$$P(Y=k|X=\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{1+\sum_{l=1}^{K-1}\exp(\mathbf{w}_l^T\mathbf{x})} \quad \text{for} \quad k=1,\ldots,K$$

1. How many parameters do we need to estimate? What are these parameters? [5pt]
2. Given $n$ training samples: $\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\ldots,(\mathbf{x}_n,y_n)\}$, please write down explicitly the log likelihood function and simplify it as much as you can:

   $$L(\mathbf{w}_1,\ldots,\mathbf{w}_{K-1}) = \sum_{i=1}^{n} \ln P(Y=y_i|X=\mathbf{x}_i).$$

   [5pt]
3. Compute the gradient of $L$ with respect to each $\mathbf{w}_k$ and simply it. [5pt]
4. Now we add the regularization term $\frac{\lambda}{2}\sum_{l=1}^{K-1}\|\mathbf{w}_l\|_2^2$ and define a new objective function:

   $$f(\mathbf{w}_1,\ldots,\mathbf{w}_{K-1}) = L(\mathbf{w}_1,\ldots,\mathbf{w}_{K-1}) - \frac{\lambda}{2}\sum_{l=1}^{K-1}\|\mathbf{w}_l\|_2^2.$$

   Compute the gradient of $f$ with respect to each $\mathbf{w}_k$. [5pt]
5. In this part, you are going to play with the "handwritten digit recognition" problem on the USPS digit dataset. Each hand-written digital image is 16 by 16 pixels. If we treat the value of each pixel as a boolean feature (either 0 for black or 1 for white), then each example has $16\times 16 = 256$ $\{0,1\}$-valued features, and hence $\mathbf{x}$ has 256 dimension. Each digit (i.e. 1,2,3,4,5,6,7,8,9,0) corresponds to a class label $y$ ($y = 1...K, K = 10$). For each digit, we have 600 training samples and 500 testing samples. You can view these images at http://www.cs.nyu.edu/~roweis/data/usps_0.jpg, ..., http://www.cs.nyu.edu/~roweis/data/usps_9.jpg.

   Please download the data from the website. Load the **usps_digital.mat** file in **usps_digital.zip** into MATLAB. You will have four matrices:

   - tr_X: training input matrix with the dimension $6000\times 256$.
   - tr_y: training label of the length 6000, each element is from 1 to 10.
   - te_X: testing input matrix with the dimension $5000\times 256$.
   - te_y: testing label of the length 5000, each element is from 1 to 10.

For those who do NOT want to use MATLAB, we also provide the text file for these four matrices in **usps_digital.zip**.

Note that if you want to view the image of a particular training/testing example in MATLAB, say the 1000th training example, you may use the MATLAB command: **imshow(reshape(tr_X(1000,:),16,16))**

(a) [15pt] Use the *gradient ascent algorithm* to train a multi-class logistic regression classifier. Plot (1) the objective value (log-likelihood), (2) the training accuracy, and (3) the testing accuracy versus the number of iterations. Report your final testing accuracy, i.e. the fraction of test images that are correctly classified.

Note that you must choose a suitable learning rate (i.e. stepsize) of the gradient ascent algorithm. A hint is that your learning rate cannot be too large otherwise your objective will increase only for the first few iterations.

In addition, you need to choose a suitable stopping criterion. You might use the number of iterations, the decrease of the objective value, or the maximum of the L2 norms of the gradient with respect to each $\mathbf{w}_k$. Or you might watch the increase of the testing accuracy and stop the optimization when the accuracy is stable.

(b) [20pt] Now we add the regularization term $\frac{\lambda}{2}\sum_{l=1}^{K-1}\|\mathbf{w}_l\|_2^2$. For $\lambda = 1, 10, 100, 1000$, report the final testing accuracies.

(c) [5pt] What can you conclude from the above experiment? (hint: the relationship between the regularization weight and the prediction performance).

## ★ SOLUTION:

1. We need to estimate $\{\mathbf{w}_1, \ldots, \mathbf{w}_{K-1}\}$, where each one is a $d$-dimensional vector. Therefore, we need to estimate in total $(K-1)*d$ parameters.

2.
$$\begin{aligned} L(\mathbf{w}_1, \ldots, \mathbf{w}_{K-1}) &= \sum_{i=1}^{n} \ln P(Y = y_i | X = \mathbf{x}_i) \\ &= \sum_{i=1}^{n} \ln \frac{\exp(\mathbf{w}_{y_i}^T \mathbf{x}_i)}{1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l^T \mathbf{x}_i)} \\ &= \sum_{i=1}^{n} \left[ \mathbf{w}_{y_i}^T \mathbf{x}_i - \ln \left( 1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l^T \mathbf{x}_i) \right) \right] \end{aligned}$$

3.
$$\begin{aligned} \nabla L(\mathbf{w}_k) &= \sum_{i=1}^{n} \left[ I(y_i = k)\mathbf{x}_i - \frac{\exp(\mathbf{w}_k^T \mathbf{x}_i)}{1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l^T \mathbf{x}_i)} \mathbf{x}_i \right] \\ &= \sum_{i=1}^{n} \left[ I(y_i = k) - \frac{\exp(\mathbf{w}_k^T \mathbf{x}_i)}{1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l^T \mathbf{x}_i)} \right] \mathbf{x}_i \\ &= \sum_{i=1}^{n} \left[ I(y_i = k) - P(y = k | X = \mathbf{x}_i) \right] \mathbf{x}_i \end{aligned}$$

4.
$$\nabla f(\mathbf{w}_k) = \nabla L(\mathbf{w}_k) - \lambda w_k.$$

5. (a) I use the stepsize $\eta = 0.0001$ and run the gradient ascent method for 5000 iterations. The objective value vs. the number of iterations; training error vs. the number of iterations; testing error vs. the number of iterations are presented in Figure 4.

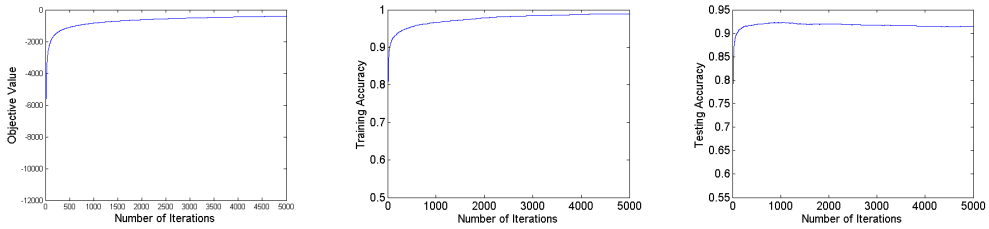


Figure 4: Performance of multi-class logistic regression on USPS digital dataset.

| $\lambda$ | 0 | 1 | 10 | 100 | 1000 |
|---|---|---|---|---|---|
| Testing Accuracy | 91.44 % | 91.58 % | 91.92 % | 89.74 % | 79.78 % |

Table 1: Comparison of testing accuracy for different regularization parameters.

(b) For $\lambda = 0, 1, 10, 100, 1000$, the comparison of the testing accuracy is presented in Table 1.

(c) From the above result, we can see that adding the regularization could avoid overfitting and lead to better generalization performance (e.g. $\lambda = 1, 10$). However, the regularization cannot be too large. Although a larger regularization can decrease the variance, it introduces additional bias and may lead to worse generalization performance.