10701/15781 Machine Learning, Fall 2009: Homework 5

Due: Monday, December 7, beginning of the class, by noon

Instructions
There are 4 questions on this assignment. Problem 2 involves coding. Do not attach your code to the writeup. Instead, copy your implementation to

/afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW5

To write in this directory, you need a kerberos instance for andrew, or you can log into, for example, unix.andrew.cmu.edu.

Please submit each problem separately with your name and Andrew ID on each problem. Refer to the webpage for policies regarding collaboration, due dates, and extensions.

IMPORTANT: We will not accept homework turned in after noon on Saturday December 12th (even for partial credit). We want to be able to post the solutions for this homework as early as possible.

1 [35 points] Expectation Maximization [Shay]

You are running a Naive Bayes classifier for a classification problem with one (unobserved) binary class variable $Y$ (e.g. whether it’s too hot for your dog in here) and 3 binary feature variables $X_1, X_2, X_3$. The class value is never directly seen but approximately observed using a sensor (e.g. you see your dog panting). Let $Z$ be the binary variable representing the sensor values. One morning (your dog is out to play and) you realize the sensor value is missing in some of the examples. From the sensor specifications (that come with your dog), you learn that the probability of missing values is four times higher when $Y = 1$ than when $Y = 0$. More specifically, the exact values from the sensor specifications are:

$P(Z\text{ missing} | Y = 1) = .08$, $P(Z = 1 | Y = 1) = .92$
$P(Z\text{ missing} | Y = 0) = .02$, $P(Z = 0 | Y = 0) = .98$

1. Draw a Bayes net that represents this problem with a node $Y$ that is the unobserved label, a node $Z$ that is either a copy of $Y$ or has the value “missing”, and the three features $X_1, X_2, X_3$.

2. What is the probability of the unobserved class label being 1 given no other information, i.e., $P(Y = 1 | Z = \text{“missing”})$? Derive the quantity using the Bayes rule and write your final answer in terms of $\theta_{Y=1}$, our estimate of $P(Y = 1)$.

3. We would like to learn the best choice of parameters for $P(Y), P(X_1 | Y), P(X_2 | Y)$, and $P(X_3 | Y)$. Assume $Y, X_1 | Y, X_2 | Y, X_3 | Y$ are all Bernoulli variables and let us denote the parameters as

$\theta_{Y=y} = P(Y = y)$, $\theta_{X_1=x_1 | Y=y} = P(X_1 = x_1 | Y = y)$,
$\theta_{X_2=x_2 | Y=y} = P(X_2 = x_2 | Y = y)$, $\theta_{X_3=x_3 | Y=y} = P(X_3 = x_3 | Y = y)$.

$^1$We only need $\theta_Y = P(Y = 1), \theta_{X_1 | Y=y} = P(X_1 = 1 | Y = y), ...$ since $\theta_{Y=0} = 1 - \theta_{Y=1}, ...$, but the set of $\theta$s defined here should help you notationally.
Write the log-probability of \( X, Y \) and \( Z \) given \( \theta \), in terms of \( \theta \), and \( P(Z|Y) \), first for a single example \((X_1 = x_1, X_2 = x_2, X_3 = x_3, Z = z, Y = y)\), then for \( n \) i.i.d. examples \((X_1^i = x_1^i, X_2^i = x_2^i, X_3^i = x_3^i, Z^i = z^i, Y^i = y^i)\) for \( i = 1, ..., n \).

4. Provide the E-step and M-step for performing expectation maximization of \( \theta \) for this problem.

In the E-step, compute the distribution \( Q_{t+1}(Y|Z,X) \) using
\[
Q_{t+1}(Y = 1|Z,X) = E[Y|Z,X]_1 = \sum_y Q(Y^i = y|Z^i, X^i_1, X^i_2, X^i_3, \theta_t)
\]

using your Bayes net from part 1 and conditional probability from part 2 for the unobserved class label \( Y \) of a single example.

In the M-step, compute
\[
\theta_{t+1} = \arg\max_{\theta} \sum_{i=1}^{n} \sum_y Q(Y^i = y|Z^i, X^i_1, X^i_2, X^i_3, Y^i, Z^i|\theta)
\]

using all of the examples \((X^i_1, X^i_2, X^i_3, Y^i, Z^i), ..., (X^n_1, X^n_2, X^n_3, Y^n, Z^n)\). Note: it is OK to leave your answers in terms of \( Q(Y|Z,X) \).

2 Randomized Singular Value Decomposition [Babis 50 points]

Background

Singular Value Decomposition is a powerful matrix decomposition with many applications: HITS algorithm by J. Kleinberg, Latent Semantic Indexing (Deerwester et al.; Papadimitriou et al), linear dimensionality reduction are some prominent applications of SVD. It is also widely called as “the Swiss Army knife of matrix computation” or even the “sledgehammer of linear algebra”.

In a real world scenario one is interested in a low rank approximation of the matrix. Specifically, consider a matrix \( A \in \mathbb{R}^{n \times m} \). According to the SVD theorem, \( A \) can be expressed as:
\[
A = \sum_{i=1}^{r} \sigma_i u_i v_i^T
\]

where \( \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \) are the singular values, \( u_i \in \mathbb{R}^n \) for \( i = 1, \ldots, r \) are the left singular vectors and \( v_i \in \mathbb{R}^m \) for \( i = 1, \ldots, r \) are the right singular vectors and \( r \) is the rank of the matrix.

Let \( A_k \) be the \( k \) rank approximation of \( A \):
\[
A = \sum_{i=1}^{k} \sigma_i u_i v_i^T
\]

Typically, \( k < r \). Matrix \( A_k \), the \( k \)-rank approximation of \( A \) is optimal with respect to the 2 norm and the Frobenius norm. For example: \( ||A - A_k||_F \leq ||A - C||_F \) for any matrix \( C \) of rank at most \( k \) (including \( k \)).

In this exercise you will study some fundamental properties of the SVD and implement a practical algorithm which comes with guarantees on its quality.

2.1 Programming Part

In many cases we are willing to compute a sub-optimal \( k \) rank approximation, \( \hat{A}_k \) if we have important computational savings. You will implement an algorithm that finds \( \hat{A}_k \) instead of \( A_k \) and comes with guarantees on the quality of the approximation as already mentioned.

The function you will write should have the following input and output arguments:
• Input
  1. Matrix $A \in \mathbb{R}^{n \times m}$
  2. Integer $s \leq n$.
  3. Integer $k \leq s$.

• Output
  1. Matrix $H \in \mathbb{R}^{m \times k}$
  2. $\lambda_1, \ldots, \lambda_k \in \mathbb{R}^+$.

Matrix $H$ will contain the approximation to the top-$k$ right singular vectors and $\lambda_1, \lambda_k$ is the approximation to the singular values.

1 [25 points] Implement the following algorithm:

1. For $i = 1$ to $n$ compute $p_i = ||A(i,:)||^2/||A||^2_F$.
2. For $i = 1$ to $s$
   • Pick an integer $j$ from 1 to $n$ with probability $Pr(\text{pick } j) = p_j$.
   • Include $A(j,:)$ as a row of S.
3. Compute $SS^T$ and its singular value decomposition, i.e., $SS^T = \sum_{t=1}^s \lambda_t^2 w_t w_t^T$.
4. Compute $h_t = \frac{S^T w_t}{||S^T w_t||}$ for $t = 1 \ldots k$.
5. Return matrix $H$ whose columns are the vectors $h_1, \ldots, h_k$ and $\lambda_1 \geq \ldots \geq \lambda_k$.

You should hand in your code printed since it should not be more than one page.

2 [10 points] Apply your implementation of the baboon image and compare to the optimal 60-rank approximation. In other words your $k$ equals 60.

Report the following:

1. [5 points] Plot the two reconstructed images. The following commands may prove useful: $\text{image}(A)$, $\text{colorbar}(\text{gray}(256))$. To read the input use the following command:

   \[ \text{A} = \text{imread(} \text{‘baboon.tif’}) \text{; A} = \text{double(A);} \]

   \textbf{Hint:} Think of how you should use matrix $H$.

2. [5 points] Report the error in terms of the Frobenius norm for both the optimal 60 rank produced from the SVD and for the 60 rank approximation produced by your implementation.

3 [2 points] Explain how you can adapt your algorithm to approximate the left singular vectors instead of the right singular vectors.

2.2 Eigenvalues and SVD [5 points]

Create a random graph on 100 nodes with probability of 0.5 of having an edge between nodes $i$ and $j$, for all possible $i, j$. The following piece of code can do this for you:

\[ \text{A} = \text{rand(100)} \text{; A} = \text{rand}(100) \leq 0.5; \]

\[ \text{A} = \text{triu(A,1);} \]

\[ \text{A} = \text{A} + \text{A}' ; \]

Compute the eigendecomposition using command $\text{eig}$ and its singular value decomposition using command $\text{svd}$. What do you observe on the values of the eigenvalues and the singular values? How can you recover the signs of the eigenvalues through the SVD?
2.3 Projections (again) [4+4 points]

Suppose you are given a set of points \( \{x_1, x_2, \ldots, x_n\} \) where \( x_i \in \mathbb{R}^m \). You create a matrix \( A \in \mathbb{R}^{n \times m} \), where the \( i \)-th row of matrix \( A \) corresponds to point \( x_i \).

You recompute the \( k \) rank approximation of \( A \), i.e., the left singular vectors \( u_1, \ldots, u_k \), the right singular vectors \( v_1, \ldots, v_k \) and the singular values \( \sigma_1, \ldots, \sigma_k \). Explain how you will use the above to do the following tasks when you are given a new point \( y \).

1. Project \( y \) on the space spanned by the left singular vectors \( u_1, \ldots, u_k \), \( k < r \), where \( r \) is the rank of the matrix \( A \).
2. Project \( y \) on the space spanned by the left singular vectors \( u_{k+1}, \ldots, u_r \), \( k < r \), where \( r \) is the rank of the matrix \( A \). Observe that your solution however must use only the vectors \( u_1, \ldots, u_k \). In other words, even if you want to project \( y \) on the space spanned by \( u_{k+1}, \ldots, u_r \) you can only use \( u_1, \ldots, u_k \).

*Hint:* Use the orthogonality properties of the singular vectors.

2.4 A favorite inequality for Extra Credit [7 points]

Let \( A, B \) be \( n \times n \) symmetric matrices. Let \( \lambda_i, \mu_i \) be the \( i \)-the eigenvalue of \( A, B \) respectively, both ordered in increasing order, \( i = 1, \ldots, n \). Prove the following inequality:

\[
\sum_{i=1}^{n} (\lambda_i - \mu_i)^2 \leq ||A - B||_F^2
\]  

(3)

where the Frobenius norm of a matrix is given by the following equation: \( ||A||_F^2 = \sum_i \sum_j a_{ij}^2 \)

*Remark:* Only complete answers will receive credit.

3 [15 points] Reinforcement Learning [Kate/Yucheng]

Consider the following Markov Decision Process:

We have states \( S_1, S_2, S_3, S_4, \) and \( S_5 \). We have actions Left and Right, and the chosen action happens with probability 1. In \( S_1 \) the only option is to go back to \( S_2 \), and similarly in \( S_5 \) we can only go back to \( S_4 \). The reward for taking any action is \( r = 1 \), except for taking action Right from state \( S_4 \), which has a reward \( r = 10 \). For all parts of this problem, assume that \( \gamma = 0.8 \).

1. What is the optimal policy for this MDP?
2. What is \( V^*(S_5) \)? It is acceptable to state it in terms of \( \gamma \), but not in terms of state values.
3. Consider executing Q-learning on this MDP. Assume that the Q values for all \( \text{(state, action)} \) pairs are initialized to 0, that \( \alpha = 0.5 \), and that Q-learning uses a greedy exploration policy, meaning that it always chooses the action with maximum Q value. The algorithm breaks ties by choosing Left. What are the first 10 \( \text{(state, action)} \) pairs if our robot learns using Q-learning and starts in state \( S_3 \) (e.g. \( (S_3, \text{Left}), (S_2, \text{Right}), (S_3, \text{Right}), \ldots) \)?
4. Now consider executing $R_{\text{max}}$ on this MDP. Assume that we trust an observed $P(x'|x,a)$ transition probability after a single observation, that the value of $R_{\text{max}} = 100$, and that we update our policy each time we observe a transition. Also, assume that $R_{\text{max}}$ breaks ties by choosing a policy of Left. What are the first 10 (state, action) pairs if our robot learns using $R_{\text{max}}$ and starts in state $S_3$ (e.g., $(S_3, \text{Left}), (S_2, \text{Right}), (S_3, \text{Right}), \ldots$)?