Machine Learning 10-701 November 18, 2013
http://alex.smola.org/teaching/cmu2013-10-701x/ due November 27, 2013
Carnegie Mellon University

Homework 4

START HERE: Instructions

- The homework is due at 11:59 pm on Wednesday November 27, 2013. Anything that is received after that time will not be considered.
- Answers to everything but question 3 will be submitted electronically through the submission website: http://alex.smola.org/teaching/cmu2013-10-701x/submission.html. Let us know if you have any problems.
- **Read this before handwriting or LaTeXing your solutions:** Previously, some students reported difficulty with submitting large image files to the handin server. So, we recommend that you should not handwrite or LaTeXyour solutions; instead use "plain text" or "markup text" mode and type or paste your solutions into the compose box. We will make our best effort to provide support for image-based handins, but until further notice they should be considered an experimental feature.
- Please follow the instructions for code submission in problem 3 correctly. The code handout is at http://www.cs.cmu.edu/~dsutherl/assignment_4_handout.tar; note that it's 750 MB.
- Collaboration on solving the homework is allowed (after you have thought about the problems on your own). When you do collaborate, you should list your collaborators! You might also have gotten some inspiration from resources (books or online etc...). This might be OK only after you have tried to solve the problem, and couldn't. In such a case, you should cite your resources.
- If you collaborate with someone or use a book or website, you are expected to write up your solution independently. That is, close the book and all of your notes before starting to write up your solution.

1 Graphical Models [Jing; 30 pts]

In this problem, you will get some practice with directed and undirected graphical models.

1.1 Directed Graphical Models (Bayesian Networks) [22 pts]

1. For the following problem, you will use D-separation to figure out which independence relations hold for the graphical model specified in Figure 4. You will receive 1 pt for the correct answer and 1 pt for the explanation. There are many names for the types of paths observed in the graphical model. Please follow the naming convention below when referring to the paths in your explanation.

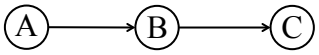

Figure 1: Serial Connection

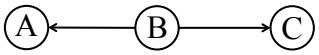

Figure 2: Diverging Connection

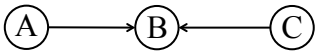

Figure 3: Converging Connection

Homework 4

For this graphical model in Figure 4, determine if the independence relation is correct or incorrect, and provide a short explanation.

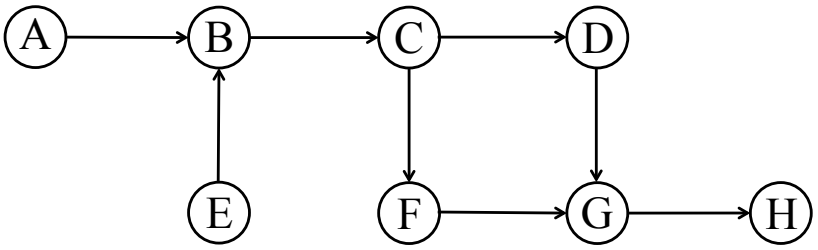

Figure 4: Bayesian Network

1. a) $C \perp E \mid B$
1. b) $A \perp E$
1. c) $A \perp C \mid E$
1. d) $D \perp F$
1. e) $F \perp H \mid G, D$

2. Write out the factorization of the joint probability distribution of the random variables that guarantees the same independence relations as in the directed graphical model given above (Figure 4).

3. Next we will perform variable elimination. Consider the following graphical model which models how likely it is for a graduate student to go on holidays at the end of semester (Figure 5). When a variable with multiple neighbors is eliminated, new intermediate factors are introduced. The variables are described below.

L	Indicator that the last homework for 10-701 was completed.
C	Indicator that a supply of coffee is constantly available.
F	Indicator that the final project for 10-701 was finished.
R	Indicator that the semester research goals were reached.
H	Indicator that the student is going on holiday in December.

Homework 4

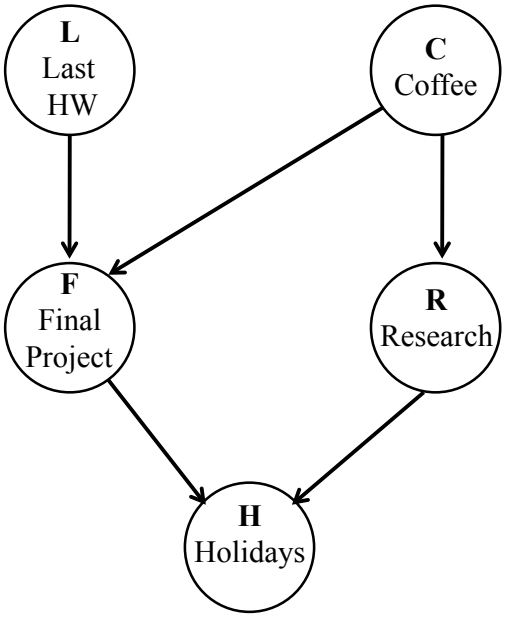

Figure 5: Bayesian Network modeling the probability of winter holidays.

3. a) Give an elimination ordering that yields an intermediate factor with 4 variables.

3. b) Give a perfect elimination ordering, one that never increases the size of a factor.

3. c) Using the perfect ordering and the following probabilities, find the probability of going on holidays given that the last homework was completed and that there was a boundless supply of coffee.

$$
\begin{aligned}
P(L = T) &= 0.8 \\
P(C = T) &= 0.9 \\
P(R|C = T) &= 0.6 \\
P(F|C = T, L = T) &= 0.9 \\
P(F|C = T, L = F) &= 0.8 \\
P(F|C = F, L = T) &= 0.7 \\
P(F|C = F, L = F) &= 0.4 \\
P(H|F = T, R = T) &= 0.9 \\
P(H|F = T, R = F) &= 0.7 \\
P(H|F = F, R = T) &= 0.6 \\
P(H|F = F, R = F) &= 0.2
\end{aligned}
$$

1.2 Undirected Graphical Models (Markov Networks) [8 pts]

In this question, we will investigate another class of graphical models called undirected graphical models or Markov networks. These models are useful representations when the setting does not naturally require directionality amongst the variables. In addition, certain probability distributions can be represented as a Markov network, but not as a Bayesian network.

1. Consider the following Markov network in Figure 6 with five variables: x_1, x_2, x_3, x_4, and x_5, where each variable can take on the value of 0 or 1. Write down the joint distribution of this undirected graphical model.

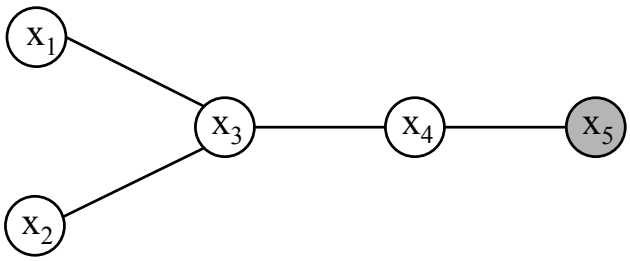

Figure 6: Markov Network.

2. Suppose we observe that $x_5 = 1$. Given the edge potentials $\psi_{13}(x_1, x_3)$, $\psi_{23}(x_2, x_3)$, $\psi_{34}(x_3, x_4)$, $\psi_{45}(x_4, x_5)$, use belief propagation (message passing) to find all the marginals. That is, give $P(x_i = 0 \mid x_5 = 1)$ and $P(x_i = 1 \mid x_5 = 1)$ for $i = 1, 2, 3, 4$.

   Hint: The potentials given are not normalized. You may find it helpful to normalize each message after it's been computed. However, this is not required and it's also fine to normalize the final marginal probability.

$$\psi_{13}(x_1, x_3) = \begin{array}{c} \\ x_3 = 0 \\ x_3 = 1 \end{array} \begin{array}{c} \begin{array}{cc} x_1 = 0 & x_1 = 1 \end{array} \\ \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \end{array}$$

$$\psi_{23}(x_2, x_3) = \begin{pmatrix} 0.5 & 1 \\ 1 & 0.5 \end{pmatrix}$$

$$\psi_{34}(x_3, x_4) = \begin{pmatrix} 0.2 & 1 \\ 1 & 0.2 \end{pmatrix}$$

$$\psi_{45}(x_4, x_5) = \begin{pmatrix} 0.1 & 1 \\ 1 & 0.1 \end{pmatrix}$$

2 Bootstrap [Ahmed; 20 pts]

[Note: This question involves simulation. You are not required to submit the code; you can use any language or toolkit you choose.]

2.1 Two modes, one variance [4+0+7+3+2=16 pts]

Let $X_1, \ldots, X_n$ be an i.i.d sample from a mixture of two Gaussians with means $\mu_1 = -1$ and $\mu_2 = +1$ and standard deviation $\sigma_1 = \sigma_2 = \sigma = 0.1$. That is, X_i is generated from $\mathcal{N}(\mu_1, \sigma_1)$ with probability $p = 0.05$, and from $\mathcal{N}(\mu_2, \sigma_2)$ with probability $1 - p$.

Let $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ denote the sample mean. We want to use bootstrap to estimate its variance so that we can report our confidence in $\bar{X}$.

1. Derive the variance of $\bar{X}$ in terms of σ, μ, p and n. Hint: recall that the law of total variance says $\mathrm{Var}[X] = \mathbb{E}_Y[\mathrm{Var}[X|Y]] + \mathrm{Var}_Y[\mathbb{E}[X|Y]]$.

2. Implement the following variance estimation procedure: Given $X_1, \ldots, X_n$, draw 100 bootstrap resamples each of size n. Estimate the sample mean of each bootstrap sample and then compute the variance across bootstrap resamples. [You are not required to submit your answer to this part. You may find the MATLAB function `bootstrp` function helpful.].

3. To analyze the general behavior of the estimator, sample n data points from the bimodal distribution and compute the variance using the aforementioned bootstrap procedure. Repeat the process on 50 different samples of size n and compute the median of the variance estimates obtained from these samples. Do this for $n \in \{2, 4, 6, 8, 10, \ldots, 30\}$ and plot a graph showing the median variance estimate vs. n. In the same graph, show the true variance as a function of n (using the derivation in 1 or estimating from a large sample).

4. From the graph, how well does the bootstrap estimate the variance for small n? How does it change with n ? Can you explain the reason behind this behavior?

5. Repeat 3 using $p = 0.5$. How does that affect the performance of the bootstrap estimate for small n? Can you explain why?

2.2 Bootstrap Cross-Validation [4 pts]

Consider the following method to estimate the generalization error of a classifier using limited data: Take a bootstrap sample from the data and perform 10 fold cross-validation on that sample to estimate the error. Repeat the process on B bootstrap resamples and report the average.

What do you think can go wrong with such a method and why?

3 Solving Systems of Linear Equations [Carlton; 40 pts + 5 Bonus]

A system of linear equations is an equation of the form $Ax = b$ where $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ are known, and $x \in \mathbb{R}^n$ is unknown.

Solving a system of linear equations is the problem of finding an x which satisfies this equation. It is one of the most fundamental operations in machine learning. Many of the most important algorithms we will teach you reduce to solving such systems. Because of this it is vitally important that we can find solutions quickly.

Homework 4

You might think that this is a simple problem with a single "correct" solution. Nothing could be further from the truth. In reality there are hundreds of different algorithms for solving this problem, each with its own advantages and disadvantages. This means that we need to carefully consider the stucture of A before deciding which algorithm we should use.

In this question you will implement several common linear system solvers, and explore how different values of A affect their execution time.

For each subquestion you will be given a single function signature, and asked to write a single octave function which satisfies the signature.

This problem is autograded using the CMU Autolab system. The code which you write will be executed remotely against a suite of tests, and the results used to automatically assign you a grade. In order for your code to execute correctly on our servers you should avoid using libraries beyond the *basic* octave libraries.

- **Submission Instructions** We have provided you with a single folder containing each of the functions you need to complete. *Do not modify the structure of this directory or rename these files*. Complete each of these functions, then compress this directory as a tar file and submit to autolab online. You may submit as many times as you like.
- **SUBMISSION CHECKLIST**
  - MUST execute on our machines in less than 20 minutes.
  - MUST be smaller than 100K.
  - MUST be a .tar file.
  - MUST return matrices of the exact dimension specified.
- **You can assume for all questions that A is full rank**.
- $A_{small}, x_{small}, b_{small}$ is a small system of linear equations you can use to test your code for parts 3.1-3.4. A_i, x_i, b_i are the systems of linear equations for the challenge question 3.5.

3.1 Matrix Inverse Solver

The most naive algorithm is to explicitly calculate A^{-1}, then use A^{-1} to calculate $x = A^{-1}b$. While seemingly reasonable, in practice this is a terrible idea: it's both slow and numerically unstable. Nevertheless this gives us a reasonable place to start from.

- Complete the function f_inv() which solves the system of linear equations $Ax = b$ by calculating $A^{-1}b$. Feel free to use the octave `inv` function here.
- f_inv() takes as input A and b and outputs x.

3.2 Gaussian Elimination Solver

Gaussian elimination is another common approach to solving systems of linear equations. This was probably the first algorithm you were taught in your Linear Algebra 101 course. Gaussian elimination has the advantage that it works on any system of linear equations. As a reminder, in its simplest form Gaussian elimination consists of two steps: *Row Reduction*, which results in an upper triangular matrix in *row echelon* form, followed by *Back Substitution*, which results in a diagonal matrix.

- Complete the function f_gaussian() which solves the system of linear equations $Ax = b$ using Gaussian elimination. Implement the algorithm yourself rather than using any Octave builtins that do this.
- f_gaussian() takes as input A and b and outputs x and $Atri$. Here $Atri \in \mathbb{R}^{n \times n}$ is the upper triangular matrix in row echelon form which is obtained after row reduction, but before back substitution.
- We require $Atri$ for autograding purposes.

3.3 Steepest Descent Solver

If A is positive semi-definite (PSD) then our problem is convex, and we can use algorithms based on the idea of gradient descent.

Consider the function $f(x) = \frac{1}{2}\|Ax - b\|^2$. This function will be minimized when $Ax = b$. Therefore solving the equation $Ax = b$ is equivalent to minimizing $f(x) = \frac{1}{2}\|Ax - b\|^2$. We can minimize $f(x)$ by calculating the gradient $\nabla f(x) = Ax - b$ and using gradient descent:

$$x^{(i+1)} = x^{(i)} + \alpha_i(Ax^{(i)} - b)$$

where $x^{(i)}$ is the ith step and $\alpha_i \in \mathbb{R}$ is the ith step size. Here $\alpha_i = -\dfrac{\nabla f(x^{(i)})^T \nabla f(x^{(i)})}{\nabla f(x^{(i)})^T A \nabla f(x^{(i)})}$.

Gradient descent with this (optimal) step size is called *steepest descent*.

- Complete the function f_steepest() which solves the system of linear equations $Ax = b$ using the steepest descent algorithm with the α_i values above.
- f_steepest() takes as input A, b, $x0$, and $nIter$, and outputs x. $x0$ is the starting point, and $nIter$ is the number of iterations. Note that one iteration corresponds to a single gradient descent step.

3.4 Jacobi Solver

If A is PSD and diagonally dominant then we can use the Jacobi method. A matrix A is diagonally dominant iff for each row i, $|A_{ii}| > \sum_{j \neq i} |A_{ij}|$.

Suppose we split the matrix A into two parts: D, whose diagonal elements are identical to those of A, and whose off-diagonal elements are zero; and E, whose diagonal elements are zero, and whose off-diagonal elements are identical to those of A. Thus $A = D + E$. Then:

$$\begin{aligned} Ax &= b \\ Dx &= -Ex + b \\ x &= -D^{-1}Ex + D^{-1}b. \end{aligned}$$

Because D is diagonal, it is easy to invert. This identity can be converted to an iterative algorithm by forming the following recurrence:

$$x^{(i+1)} = -D^{-1}Ex^{(i)} + D^{-1}b.$$

This iterative algorithm is known as the Jacobi solver.

- Complete the function f_jacobi() which solves the system of linear equations $Ax = b$ using the Jacobi solver.
- f_jacobi() takes as input A, b, $x0$, and $nIter$, where $x0$ is the starting point, and $nIter$ is the number of iterations. Note that one iteration corresponds to a single update step.

3.5 CHALLENGE: The fastest solver

Welcome to the third and final 10-701 Challenge Question! Once again this is a competition (view the class leaderboard on the autolab website) and 5 bonus points will be awarded to students who have top 10 classification accuracy on the class leaderboard.

- You are given 6 systems of linear equations $A_i x_i = b_i$ for $i \in 1, 2, ..., 6$. **Your task is to write 6 algorithms, one for each system of equations, such that your algorithms solve these systems as quickly as possible.**
- Each of these systems will be of a different form, and can be solved best with specific algorithms.
- **Your algorithms must produce a solution $\hat{x}$ such that if x is the true solution, then $\|\hat{x} - x\|_2 < 10^{-5}$.** If the solution produced by your algorithm breaks this bound, then your code will receive a grade of zero.
- Your code will be tested serverside using systems of linear equations which are of the same form as those given to you, but which contain different values.
- Your grade will be inversely logarithmically proportional to the time it takes your algorithm to correctly solve the equations. In other words faster code $\rightarrow$ better grade.
- You are welcome to use builtin Octave libraries *for this challenge*, or to write your own code. I don't care what you use so long as your solvers run quickly.

HINT: The following is a list of techniques you might want to consider trying (in addition to the ones you already implemented above). Many of these have built in octave implementations:

- Richardson Iteration
- Gauss-Seidel
- Successive Over-Relaxation
- Conjugate Gradient
- Preconditioning
- Cholesky Decomposition
- Chebychev Iteration
- Generalized Minimal Residual
- Biconjugate Gradient
- Pivoted Gaussian Elimination