CPSC-540 **Machine Learning** **2013**

Homework # 2

NAME:____

Signature:______________________________________

STD. NUM: _____________________________________

General guidelines for homeworks:

You are encouraged to discuss the problems with others in the class, but all write-ups are to be done on your own.
Homework grades will be based not only on getting the "correct answer," but also on good writing style and clear presentation of your solution. It is your responsibility to make sure that the graders can easily follow your line of reasoning.
Try every problem. Even if you can't solve the problem, you will receive partial credit for explaining why you got stuck on a promising line of attack. More importantly, you will get valuable feedback that will help you learn the material.
Please acknowledge the people with whom you discussed the problems and what sources you used to help you solve the problem (e.g. books from the library). This won't affect your grade but is important as academic honesty.
When dealing with python exercises, please attach a printout with all your code and show your results clearly.

1 Bias Variance Trade-off

Prove the decomposition of the mean squared error in terms of bias and variance that was presented on the lecture on *maximum likelihood and linear prediction*. (**Recommended reading**: The section titled **Model Selection and the Bias-Variance Tradeoff** in the book of Hastie, Tibshirani and Friedman, which is freely-available on the course website).

$$MSE = E[(\hat{\theta}-\theta_0)^T(\hat{\theta}-\theta_0)]$$

We want write in the form of $\bar{\theta}-\theta_0$ (bias) and $\hat{\theta}-\bar{\theta}$ (for variance)

where $\bar{\theta} = E[\hat{\theta})$, θ_0 is the true value.

$$= E[(\bar{\theta}-\theta_0+\hat{\theta}-\bar{\theta})^T(\bar{\theta}-\theta_0+\hat{\theta}-\bar{\theta})]$$

$$= E[(\bar{\theta}-\theta_0)^2+(\hat{\theta}-\bar{\theta})^2+2(\bar{\theta}-\theta_0)^T(\hat{\theta}-\bar{\theta})]$$

$$= (\bar{\theta}-\theta_0)^2+E(\hat{\theta}-\bar{\theta})^2+2E[\bar{\theta}^T\hat{\theta}-\theta_0^T\hat{\theta}-\bar{\theta}^T\bar{\theta}+\theta_0^T\bar{\theta})$$

$$= (\bar{\theta}-\theta_0)^2+E(\hat{\theta}-\bar{\theta})^2+2(\bar{\theta}^T\bar{\theta}-\theta_0^T\bar{\theta}-\bar{\theta}^T\bar{\theta}+\theta_0^T\bar{\theta})$$

$$= \underbrace{(\bar{\theta}-\theta_0)^2}_{\text{Square of bias}}+\underbrace{E(\hat{\theta}-\bar{\theta})^2}_{\text{Variance.}}$$

2 Entropy of a Gaussian Distribution

Consider the multivariate Gaussian distribution for $\mathbf{x} \in \mathbb{R}^N$:

$\mu \in R^N$

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = |2\pi\boldsymbol{\Sigma}|^{-1/2} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}.$$

$\Sigma \in R^{N\times N}$

Show that the *differential entropy* (entropy of a continuous variable) of this distribution is given by:

$$h(p) = -\int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} = \frac{1}{2} \log\left\{(2\pi e)^N |\boldsymbol{\Sigma}|\right\}.$$

we know $\int p(x)\,dx = 1$, $|2\pi\Sigma| = 2\pi^N|\Sigma|$.

$$h(p) = -\int p(x)\left[-\frac{1}{2}\log(2\pi)^N|\Sigma| - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]dx$$

$$= \frac{1}{2}\log(2\pi)^N|\Sigma| - \int p(x)\cdot\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]dx.$$

$$= \frac{1}{2}\log(2\pi)^N|\Sigma| + \frac{1}{2}E\left[(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

$E[(x-\mu)^T\Sigma^{-1}(x-\mu)] = E[tr((x-\mu)^T\Sigma^{-1}(x-\mu))]$ since $(x-\mu)^T\Sigma^{-1}(x-\mu) \in R$.

$= E[tr(\Sigma^{-1}(x-\mu)(x-\mu)^T)]$ since $tr(AB) = tr(BA)$

$= tr(\Sigma^{-1}E[(x-\mu)(x-\mu)^T])$ By linearity of $tr(\cdot)$ and $E(\cdot)$

$= tr(\Sigma^{-1}\Sigma)$

$= N$

Therefore, we have

$$h(p) = \frac{1}{2}\log(2\pi)^N|\Sigma| + \frac{1}{2}N = \frac{1}{2}\log\{(2\pi)^N|\Sigma|\cdot e^N\} = \frac{1}{2}\log\{(2\pi e)^N|\Sigma|\}$$

3 Collaborative Filtering for Recommendation

Assume that user u has indicated preference for item i via the variable

$$P_{ui} = \begin{cases} +1 & \text{if user } u \text{ liked (thumbed up) item } i \\ 0 & \text{if user } u \text{ did not rate item } i \\ -1 & \text{if user } u \text{ disliked (thumbed down) item } i. \end{cases}$$

We have m users and n items so that $u = 1, \ldots, m$ and $i = 1, \ldots, n$.

Given n movies, our first objective will be to learn a matrix of *factors* $\mathbf{Y} \in \mathbb{R}^{f \times n}$ for these movies. That is, each movie will be described by a column vector $\mathbf{y}_i \in \mathbb{R}^{f \times 1}$ of f factors (features). Our second objective will be to learn a matrix of factors for the m users of the social network $\mathbf{X} \in \mathbb{R}^{m \times f}$. Each user will be described by a row vector of factors $\mathbf{x}_u \in \mathbb{R}^{1 \times f}$.

Since we have two unknowns, the method of solution will be alternating ridge regression. That is, we first fix $\mathbf{Y}$ and solve for $\mathbf{X}$, use this estimate of $\mathbf{X}$ and solve for $\mathbf{Y}$, use the latest estimate of $\mathbf{Y}$ and solve for $\mathbf{X}$ again, etc. In effect, we are estimating an approximation of $\mathbf{P}$ as follows: $\widehat{\mathbf{P}} = \widehat{\mathbf{X}}\widehat{\mathbf{Y}}$.

In addition, we will introduce a matrix of weights c_{ui}, defined as follows:

$$c_{ui} = |p_{ui}|.$$

To compute the factors for each user, we assume the $\mathbf{y}_i$ are given and proceed to minimize the following quadratic cost function:

$$J(\mathbf{x}_u) = \sum_{i=1}^{n} c_{ui}(p_{ui} - \mathbf{x}_u \mathbf{y}_i)^2 + \lambda \|\mathbf{x}_u\|_2^2 \qquad \text{for each user } u.$$

This is known as a weighted ridge regression problem. Note that if all the c_{ui} were equal to 1, then we would be simply going back to the standard ridge estimates. Importantly, the addition of c_{ui} will enable us to focus only on minimizing the error for the movies that the user actually rated!

For each user, assume we construct the diagonal matrix $\mathbf{C}_u \in \mathbb{R}^{n \times n}$ with diagonal entries c_{ui}. Let $\mathbf{p}_u \in \mathbb{R}^{1 \times n}$ denote the vector of preferences for user u. As shown in your last homework, the weighted ridge regression objectives can be re-written in matrix format as follows:

$$\begin{aligned} J(\mathbf{x}_u) &= (\mathbf{p}_u - \mathbf{x}_u \mathbf{Y})\mathbf{C}_u(\mathbf{p}_u - \mathbf{x}_u \mathbf{Y})^T + \lambda \mathbf{x}_u \mathbf{x}_u^T \\ J(\mathbf{y}_i) &= (\mathbf{p}_i - \mathbf{X}\mathbf{y}_i)^T \mathbf{C}_i (\mathbf{p}_i - \mathbf{X}\mathbf{y}_i) + \lambda \mathbf{y}_i^T \mathbf{y}_i, \end{aligned}$$

with solutions:

$$\begin{aligned} \mathbf{x}_u &= \left(\mathbf{Y}\mathbf{C}_u\mathbf{Y}^T + \lambda \mathbf{I}\right)^{-1} \mathbf{Y}\mathbf{C}_u\mathbf{p}_u \\ \mathbf{y}_i &= \left(\mathbf{X}^T\mathbf{C}_i\mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^T\mathbf{C}_i\mathbf{p}_i. \end{aligned}$$

In the above, $\mathbf{P} \in \mathbb{R}^{m \times n}$, $\mathbf{X} \in \mathbb{R}^{m \times f}$, $\mathbf{Y} \in \mathbb{R}^{f \times n}$, $\mathbf{C}_u \in \mathbb{R}^{n \times n}$, $\mathbf{C}_i \in \mathbb{R}^{m \times m}$, $\mathbf{x}_u$ and $\mathbf{p}_u$ are the uth rows of $\mathbf{X}$ and $\mathbf{P}$, respectively, and finally, $\mathbf{y}_i$ and $\mathbf{p}_i$ are the ith column of $\mathbf{Y}$ and $\mathbf{P}$, respectively.

1. Finish implementing the alternating weighted ridge regression estimation in the code below. Note that the code already implements the alternating ridge regression code.

```
from __future__ import division
import numpy as np
import pdb

# MOVIES: Legally Blond; Matrix; Bourne Identity; You’ve Got Mail;
# The Devil Wears Prada; The Dark Knight; The Lord of the Rings.
P = [[0,0,-1,0,-1,1,1],   # User 1
     [-1,1,1,-1,0,1,1],   # User 2
     [0,1,1,0,0,-1,1],    # User 3
     [-1,1,1,0,0,1,1],    # User 4
     [0,1,1,0,0,1,1],     # User 5
     [1,-1,1,1,1,-1,0],   # User 6
     [-1,1,-1,0,-1,0,1],  # User 7
     [0,-1,0,1,1,-1,-1],  # User 8
     [0,0,-1,1,1,0,-1]]   # User 9
P = np.array(P)
print ’Raw Preference Matrix:’
print P
print ’\n’

# Parameters
reg = 0.1       # regularization parameter
f = 2           # number of factors
m,n = P.shape

# Random Initialization
# X is (m x f)
# Y is (f x n)
X = 1 - 2*np.random.rand(m,f)
Y = 1 - 2*np.random.rand(f,n)
X *= 0.1
Y *= 0.1

# Alternating Ridge Regression
for _ in xrange(100):
    # Least-squares keeping Y fixed
    X = np.linalg.solve(
            np.dot(Y, Y.T) + reg * np.eye(f),
            np.dot(Y, P.T)
            ).T
    # Least-squares keeping X fixed
    Y = np.linalg.solve(
            np.dot(X.T, X) + reg * np.eye(f),
            np.dot(X.T, P)
            )
print ’Alternating Ridge Regression:’
print np.dot(X,Y)
print ’\n’
```

```
# Re-initialize
X = 1 - 2*np.random.rand(m,f)
Y = 1 - 2*np.random.rand(f,n)
X *= 0.1
Y *= 0.1

# Alternating Weighted Ridge Regression
C = np.abs(P)       # Will be 0 only when P[i,j] == 0.
for _ in xrange(100):
    # Each user u has a different set of weights Cu
    for u,Cu in enumerate(C):
        X[u] = ???
    for i,Ci in enumerate(C.T):
        Y[:,i] = ???
print 'Alternating Weighted Ridge Regression:'
print np.dot(X,Y)
```

2. Which top movie would you recommend for each user?