Exercise Sheet 2

1 Collaborative Filtering for Movie Recommendation

This exercise is based on the Netflix Competition. The goal is to recommend movies to users based on their votes. Here we will derive a key technique by the people who won the million dollar prize. Note that although we are focusing on movies, we could use this technique for recommending any items in a social network. The actual reference is:


We assume that user $u$ has indicated preference for item $i$ via the variable $p_{ui} \in \{0, 1\}$, where $u = 1, \ldots, m$ and $i = 1, \ldots, n$. For example, a 1 could mean that the user gave the item a thumbs up. These votes are part of a matrix $P \in \mathbb{R}^{m \times n}$ that has many missing entries. (The machine learning method here will take care of filling in the missing entries!).

Yifan Hu and collaborators applied this technique to movies, where users provide ratings $r_{ui}$ in the form of stars. They convert these ratings to our preference variables as follows:

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases},$$

and $r_{ui}$ is the vote of user $u$ for item $i$.

We also assume that we have an additional confidence signal $c_{ui}$ that measures the extent of the interaction (attraction, etc.) of user $u$ to item $i$. For example if user $u$ gives a movie $i$ a thumbs up and finishes watching it; or if the user is an expert in the genre of the movie, we would use a large value of $c_{ui}$. Yifan Hu use $c_{ui}$ as the “confidence” of user $u$ on item $i$:

$$c_{ui} = 1 + \alpha r_{ui},$$

where $\alpha$ is a parameter, which is set to 40 in the original paper.

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

Since we have two unknowns, the method of solution will be alternating ridge regression. That is, we first fix $Y$ and solve for $X$, use this estimate of $X$ and solve for $Y$, use the latest estimate of $Y$ and solve for $X$ again, etc.

To compute the factors for each user, we assume the $y_i$ are given, and proceed to minimize the following quadratic cost function:

$$J(x_u) = \sum_{i=1}^{n} c_{ui}(p_{ui} - x_u y_i)^2 + \lambda \|x_u\|_2^2$$

for each user $u$. 


This is known as a weighted ridge regression problem.

1. For each user, assume we construct the diagonal matrix $C_u \in \mathbb{R}^{n \times n}$ with diagonal entries $c_{ui}$. Let $p_u \in \mathbb{R}^{1 \times n}$ denote the vector of preferences for user $u$. Show that the above weighted ridge regression objective can be re-written in matrix format as follows:

$$J(x_u) = (p_u - x_u Y) C_u (p_u - x_u Y)^T + \lambda x_u x_u^T$$

2. Derive the solution $\hat{x}_u$ to the above objective using the matrix differentiation rules discussed in the lectures.

2 Variant of collaborative filtering to deal with missing votes

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

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

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 $y_i$ are given and proceed to minimize the following quadratic cost function:

$$J(x_u) = \sum_{i=1}^{n} c_{ui} (p_{ui} - x_u y_i)^2 + \lambda \|x_u\|^2_2 \quad \text{for each user } u.$$ 

This is known as a weighted ridge regression problem.

For each user, assume we construct the diagonal matrix $C_u \in \mathbb{R}^{n \times n}$ with diagonal entries $c_{ui}$. Let $p_u \in \mathbb{R}^{1 \times n}$ 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:

$$J(x_u) = (p_u - x_u Y) C_u (p_u - x_u Y)^T + \lambda x_u x_u^T$$

$$J(y_i) = (p_i - X y_i)^T C_i (p_i - X y_i) + \lambda y_i y_i^T$$,

for each user $u$.
with solutions:

\[
x_u^T = (Y C_u Y^T + \lambda I)^{-1} Y C_u p_u^T
\]
\[
y_i = (X^T C_i X + \lambda I)^{-1} X^T C_i p_i.
\]

In the above, \( P \in \mathbb{R}^{m \times n} \), \( X \in \mathbb{R}^{m \times f} \), \( Y \in \mathbb{R}^{f \times n} \), \( C_u \in \mathbb{R}^{n \times n} \), \( C_i \in \mathbb{R}^{m \times m} \), \( x_u \) and \( p_u \) are the \( u \)th rows of \( X \) and \( P \), respectively, and finally, \( y_i \) and \( p_i \) are the \( i \)th column of \( Y \) and \( P \), respectively.

1. Explain why the above choice of \( c_{ui} \) makes sense.

2. **Advanced/optional:** Implement the alternating weighted ridge regression method in Torch. To help you, I’ve provided the code below, but it is written in Python.

```python
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,-1],  # 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)

# 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 Weighted Ridge Regression

C = np.abs(P)  # Will be 0 only when P[i,j] == 0.
for _ in xrange(100):
    # Solve for X keeping Y fixed
    # Each user u has a different set of weights Cu
    for u,Cu in enumerate(C):
        X[u] = np.linalg.solve(
            np.dot(Y * Cu, Y.T) + reg * np.eye(f),
            np.dot(Y * Cu, P[u])
        )

    # Solve for X keeping Y fixed
    for i,Ci in enumerate(C.T):
        Y[:,i] = np.linalg.solve(
            np.dot(X.T * Ci, X) + reg * np.eye(f),
            np.dot(X.T * Ci, P[:,i].T)
        )

print 'Alternating Weighted Ridge Regression:'
print np.dot(X,Y)

3. Which movie would you recommend for each user?