CS 395T **Scalable Machine Learning** **Fall 2014**

Homework 1

Lecturer: Inderjit Dhillon Date Due: Sep 15, 2014

Keywords: *Netflix, Regression, Least Squares*

Note:

1. The datasets for this homework can be downloaded from the Assignments tab in Canvas (called `hw1-data.zip`).
2. Submit a hard copy of your answers and code.

Problem 1: Million Dollar Question

For this problem, you will work on the Netflix dataset, where one needs to predict missing (movie, user) ratings. The problem of predicting missing values for a recommender system is formally known as Collaborative Filtering. Read more about the Netflix challenge and the dataset at: `http://www.netflixprize.com`. The complete Netflix dataset has 480,189 users, 17,770 movies, and 100,480,507 ratings. For this assignment, we will work on three subsets of the full dataset: (1) `small`, consisting of about 0.1M ratings, (2) `medium`, consisting of about 1M ratings, and (3) `large`, consisting of about 10M ratings. You will find the three datasets as `.mat` files in `hw1-data.zip`.

We will take an approach based on Matrix Factorization which was also used by a leading team for Netflix challenge. Let us first introduce some notation. Let there be u users and m movies and let $R \in \mathbb{R}^{u \times m}$ be the ratings matrix where element R_{ij} is the rating given by user i for movie j. Note that the matrix R has many missing values - in the entire Netflix dataset, 98.82% of the entries are missing. Think of factoring the matrix $R \approx U^T M$, where $U \in \mathbb{R}^{k \times u}$ and $M \in \mathbb{R}^{k \times m}$ and $k \ll \min(u, m)$. Intuitively, you can think of column $\boldsymbol{u}_i$ of U as a low-dimensional feature vector for the i-th user, and the column $\boldsymbol{m}_j$ of M as a low-dimensional feature vector for the j-th movie.

Using the above notation, the rating R_{ij} for a movie j by user i is approximated as $R_{ij} \approx \boldsymbol{u}_i^T \boldsymbol{m}_j$. Thus, the Matrix Factorization approach involves finding U and M such that $U^T M$ is as close to R as possible. Formally the problem can be stated as:

$$\min_{U,M} \ \|R - U^T M\|_F^2 + \lambda(\|U\|_F^2 + \|M\|_F^2), \tag{1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and where we have added regularization terms (as in ridge regression) and the regularization parameter λ.

This problem is in general hard to solve as both U and M need to be inferred and no known method guarantees the optimal solution. Instead, we use a heuristic which works well in practice. We use the method of Alternating Minimization, where in each iteration we fix U to compute M, and subsequently fix M to compute U, and so on. Now we describe the method to compute M assuming U to be fixed.

Let U be fixed, then (1) reduces to:

$$\min_{M} \ \|R - U^T M\|_F^2 + \lambda \|M\|_F^2,$$

This problem can be decoupled in terms of columns of M, $\boldsymbol{m}_j$, i.e.,

$$\min_{\{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_m\}} \ \sum_j \|\boldsymbol{r}_j - U^T \boldsymbol{m}_j\|_2^2 + \lambda \sum_j \|\boldsymbol{m}_j\|_2^2,$$

where $\boldsymbol{r}_j$ denote the j-th column of R. Thus, we can solve a separate ridge regression problem for each j, i.e.,

$$\min_{\boldsymbol{m}_j} \ \|\boldsymbol{r}_j - U^T \boldsymbol{m}_j\|_2^2 + \lambda \|\boldsymbol{m}_j\|_2^2,$$

However, the vector $\boldsymbol{r}_j$, which represents ratings for movie j, may have many missing values. Let there be n_j known ratings for the movie j. Then,the ridge regression should be solved using only n_j ratings, i.e. let $\boldsymbol{r}_j^{(\mathcal{K}_|)} \in \mathbb{R}^{n_j}$ denote the known ratings for movie j, and $U^{(\mathcal{K}_j)} \in \mathbb{R}^{k \times n_j}$ denote the submatrix of U for the corresponding users who rated movie j. Thus the problem can be written as:

$$\min_{\boldsymbol{m}_j} \ \|\boldsymbol{r}_j^{(\mathcal{K}_|)} - \left(U^{(\mathcal{K}_j)}\right)^T \boldsymbol{m}_j\|_2^2 + \lambda \|\boldsymbol{m}_j\|_2^2,$$

An analogous procedure can be used for computing U.

So, for each iteration compute U and M using the procedure given above. Repeat this procedure for a fixed number of iterations until convergence.

(a) (5 pt) Implement the Alternating Least Squares algorithm outlined above in Matlab. Initialize U and M randomly. Set $\mathbf{u}_i$ or $\mathbf{m}_j$ to zero if there is no corresponding ratings observed in the training set. For the Ridge regression sub-problem, use the closed-form solution by invoking the backslash operator in Matlab.

(b) (1 pt) Select $\lambda = 0$ and compute U and M on, say, the `small` dataset. What problems do you face?

(c) (2 pt) Using a ten-fold cross validation on the `small` training data, arrive at the value of the regularization parameter λ (For example, you could try λ values $10^{-2}, 10^{-1}, 1$). Report the optimal λ. For the optimal λ, find out the RMSE on the test data.

(d) (2 pt) Repeat part (c) with `medium` and `large` datasets.

Note: You may have to appropriately choose the number of iterations (about 5 or 10) and the rank k (try $10, 20, 50$) — computations can become very expensive for `medium` or `large` datasets.

Problem 2: Coordinate Descent

In this problem, you are asked to implement coordinate descent to solve regularized regression problem in Matlab.

(a) (3 pt) Write a Matlab function `function w = cd_ridge(y, X, lambda)`, which solves the Ridge regression problem using Coordinate Descent:

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|X\mathbf{w} - \mathbf{y}\|_F^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2.$$

Report both the training and the test RMSE for $\lambda = \{10, 1, 0.1\}$ on `bodyfat` dataset in `hw1-data.zip`. This dataset contains training data `(y,X)` and test data `(yt, Xt)`.

(b) (5 pt) Write a Matlab function `function w = cd_lasso(y, X, lambda)`, which solves the Lasso problem using Coordinate Descent:

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|X\mathbf{w} - \mathbf{y}\|_F^2 + \lambda\|\mathbf{w}\|_1.$$

Report both the training and the test RMSE for $\lambda = \{10, 1, 0.1\}$ on `bodyfat` dataset.

Hint. To derive the closed form solution for the quadratic minimization problem $\min_x ax^2 + bx + c|x|$, you might have to consider three cases: $x^* > 0$, $x^* < 0$, and $x^* = 0$.

Below is a general guide to your coordinate descent implementation in parts (a) and (b).

- Initialize $\mathbf{w}$ to zero.
- Fix the number of iterations to 20.
- Use the fixed cyclic update sequence.

(c) (2 pt) Replace the "backslash" approach used to solve linear system in Problem 1 with `cd_ridge` and train the Alternating Least Squares code you developed in Problem 1 on `small`, `medium` and `large` movie rating datasets. You can use the optimal λ you found for the respective datasets in Problem 1 (no need to repeat the cross-validation). Report the running time and the test RMSE in each case, and compare with that of the results using backslash operator in Problem 1.

Problem 3. Parallel Alternating Least Squares (Extra Credit)

(5 pt) Speed up your code above for the Netflix problem. You can either use better software, or better (parallel) hardware. Note that this problem is optional.