THE UNIVERSITY OF TEXAS
AT AUSTIN

CS363D STATISTICAL LEARNING AND DATA MINING

Homework 03

Edited by LaTeX

Department of Computer Science

STUDENT
Jimmy Lin
xl5224

INSTRUCTOR
Pradeep Ravikumar

TASSISTANT
Adarsh Prasad

RELEASE DATE
March. 25 2014

DUE DATE
April. 27 2014

TIME SPENT
7 hours

March 28, 2014

Contents

List of Figures

1 MF Implementation

Figure 1: Errors With Regard to Parameter λ

For specific details, see the source code in Appendix A. For debugging, see the Execution logs in Appendix B.

2 Report optimal λ

3 Problems when $\lambda = 0$

The problems encountered when $\lambda = 0$ are

4 RMSE of Test Set under optimal λ

The RMSE of test set under optimal $\lambda =$ is ...

A Source Code

```
% Solution for the data mining homework 03
% Author: Jimmy Lin (xl5224)

function solution()
%%% load the dataset
load('./dataset/hw3_netflix.mat');
%%% Setting about data
nCVFolds = size(cvSet, 1);
FOLDRANGE = 1:nCVFolds;
sRatings = size(Ratings);
nUsers = sRatings(1);
nMovies = sRatings(2);

%%
% PRE-SETTING
LAMBDAS = 0:0.05:1;
NITERATIONS = 30;
K = 10;
nLambdas = size(LAMBDAS, 2);

%%
% CROSS VALIDATION
avgError = zeros(1, nLambdas);
for l = 1:nLambdas,
    lambda = LAMBDAS(l);
    foldError = zeros(1,nCVFolds);
    for f = FOLDRANGE,
        %% prepare elements for training
        nItems = length(cvSet(f,:));
        cvTrainR = trR;
        cvTrainR(cvSet(f,:)) = 0;
        cvTestR = trR(cvSet(f,:));
        %% apply Alternating Minimization for training
        [U, M] = trainMF (cvTrainR, lambda, NITERATIONS, K);
        %% make prediction rating matrix
        PredictedRatings = U * M';
        %% generate prediction array for error computation
        cvPrediction = PredictedRatings(cvSet(f,:));
        %% compute root mean square error
        foldError(f) = computeRMSE (cvPrediction, cvTestR, nItems);
        fprintf('(Lambda, Fold, Error) = (%0.2f, %d, %f)\n', ...
                lambda, f, foldError(f))
    end
    fprintf('Errors when lambda=%0.2f: ', lambda)
    disp(foldError)
    %% take the mean of fold errors as error of lambda
    avgError(l) = mean(foldError);
end
    plot(LAMBDAS, avgError, 'x-')
    hold on
    %% pick up the optimal lambda
    optIdx = find(avgError <= min(avgError) + 1e-3);
    optLambda = LAMBDAS(optIdx)
    assert(all(optLambda <= avgError) == 1)
    plot([optLambda], [avgError(optIdx)], 'dr', 'MarkerSize', 10)
    %% training by using optimal lambda
    [U, M] = trainMF (trR, optLambda, NITERATIONS, K);
    optPredictedRatings = U * M';
    %% compute optimal
    optRMSE = computeRMSE(optPredictedRatings(testIdx), ...
        Ratings(testIdx), length(testIdx))
end
```

```
%%% subfunction:
%% functionality: apply matrix factorization on training data
function [U, M] = trainMF (trainData, lambda, iterations, K)
nUsers = size(trainData, 1);
nMovies = size(trainData, 2);
U = rand(nUsers, K);
M = rand(nMovies, K);

for iter = 1:iterations,
    for j = 1:nMovies,
        M(j,:) = inv(U' * U + lambda * eye(K)) * U' * trainData(:,j);
    end
    for i = 1:nUsers,
        U(i,:) = inv(M' * M + lambda * eye(K)) * M' * trainData(i,:)';
    end
    PredictedRatings = U * M';
end
end

function err = computeRMSE (Prediction, GroundTruth, nItems)
err = sqrt(sum(sum((Prediction-GroundTruth).^2)) / nItems);
end
```

B Execution Logs