Your Name: Leonardo Neves

Your Andrew ID: lneves

Homework 6

0. Statement of Assurance

You must certify that all of the material that you submit is original work that was done only by you. If your report does not have this statement, it will not be graded.

All the material submitted is my original work.

1. Training Set Construction (5 pts)

Construct the training set for LETOR as instructed and report the following statistics.

Statistics	
the total number of observed ratings in R	820367
the total number of training examples in T	3520432
the ratio of positive examples to negative examples in T	1.0
the number of training examples in T for user ID **1234**	196
the number of training examples in T for user ID **4321**	64

2. NDCG of Memory-based methods (10 pts)

2.1 User-user similarity (Do the same imputation first as in HW4)

Rating Method	**Similarity Metric**	**K**	**NDCG@20**	**Runtime(sec)***
Mean	Dot product	10	0.937979	104
Mean	Dot product	100	0.943163	177
Mean	Dot product	500	0.942097	516
Mean	Cosine	10	0.937979	95
Mean	Cosine	100	0.943163	172
Mean	Cosine	500	0.942097	499

*runtime should be reported in seconds.

2.2 Movie-movie similarity (Do the same imputation first as in HW4)

Rating Method	Similarity Metric	K	NDCG@20	Runtime(sec)
Mean	Dot product	10	0.934675	705
Mean	Dot product	100	0.941917	701
Mean	Dot product	500	0.940577	853
Mean	Cosine	10	0.934675	553
Mean	Cosine	100	0.941917	641
Mean	Cosine	500	0.940577	951

3. NDCG of PMF (6 pts)

Report the best results you can get on dev set from training PMF with different dimensions of latent factors.

Num of Latent Factors	NDCG@10	NDCG@20	NDCG@30	Runtime(sec)*
10	0.942888	0.947098	0.947512	400.05
20	0.944110	0.948245	0.948677	555.21
50	0.944876	0.948951	0.949376	670.06
100	0.944727	0.948764	0.949168	805.01

4. RankSVM (12 pts)

a. After the optimum w is learned by SVM on the training set T, given a testing user, how to generate the ranked list of unrated items and why? (Please be concise and clear; use formulas to express your idea when possible) (2 pts)

It would have been possible to use the predictions of the model to improve the pmf ratings. An experiment I did showed that adding the class label (-1 or 1) to the pmf prediction ratings improved the NDCG a lot, since we were penalizing the ratings that were classified as the negative class. To use the w vector, however, the best way was to take the dot product of the weights and the feature vector created based on the PMF factors. From this, we would have a score for each item for each user, making it easy to sort them.

b. Report the best results you can get on dev set from training RankSVM with features from PMF (10 pts)

Num of Latent Factors from PMF	NDCG@10	NDCG@20	NDCG@30	Runtime(sec)*
10	0.942990	0.947235	0.947687	40.831

20	0.941060	0.945422	0.945875	61
50	0.943286	0.947386	0.947816	93
100	0.942459	0.946574	0.946993	202

*report the time of training SVM after PMF outputs the features. For each case, you might want to tune the regularization parameters of both PMF and SVM.

5. LR-LETOR (12 pts)

a. After the optimum w is learned by LR-LETOR on the training set T, given a testing user, how to generate the ranked list of unrated items and why? (Please be concise and clear; use formulas to express your idea when possible) (2 pts)

   For LR, the same idea of SVM can be used. We could use the dot product of the feature vector and the w vector. This would be equivalent to making a prediction using the sigmoid output because, although the values would be different, the ranking would be the same.

b. Report the best results you can get on dev set from training LR-LETOR with features from PMF (10 pts)

Num of Latent Factors from PMF	**NDCG@10**	**NDCG@20**	**NDCG@30**	**Runtime(sec)***
10	0.943009	0.947281	0.947719	82.4
20	0.943651	0.947905	0.948320	111
50	0.944645	0.948652	0.949059	134
100	0.944702	0.948897	0.949310	279

*report the time of training LR-LETOR after PMF outputs the features. For each case, you might want to tune the regularization parameters of both PMF and LR.

6. Analysis of results (20 pts)

Discuss the complete set of experimental results, comparing the algorithms to each other. Discuss your observations about the various algorithms, i.e., differences in how they performed, different parameters, what worked well and didn't, patterns/trends you observed across the set of experiments, etc. Try to explain why certain algorithms or approaches behaved the way they did.

PMF was a better algorithm for 20 and 50 factors. From the previous homework, it was expected that, at least for those values, PMF would have a good result since we saw the optimum value for this problem is between 20 and 50. On the other hand, both SVM and LR were better with a smaller number of factors and LR was better with a 100 features, showing that PMF is only good around its optimum number of factors, while the other models were able to learn with several features. I will select LR with 100 features as my

test prediction model, since it apparently is able to perform as well as pmf on most tasks with a more flexibility with the number of features. Also, since we are being judged on the basis of NDCG@20, this would be the best result I had.

SVM was never the best model. I tried to train it with various parameters but the improvements are not worth the time it takes to train on some parameters, which took much longer than the reported time for the default parameters.

7. The software implementation (15 pts)

Add detailed descriptions about software implementation & data preprocessing, including:

1. A description of what you did to preprocess the dataset to make your implementations easier or more efficient.

To preprocess the data, I have created an inverted list of users to the respective movie#ratings.

This way, each line of my new file would be easily turned into a matrix of shape #users x

#movies with the rating as the elements.

2. A description of major data structures (if any); any programming tools or libraries that you used;

I have used scipy sparse matrix for the rating matrix. I have also used sklearn for the cosine similarity on the first experiments. From numpy, I have used numpy arrays for the dense matrices, the linear algebra norm for the pmf.

To run svm, I called liblinear from python using subprocess, so it would call the command line version instead of the python wrapper.

3. Strengths and weaknesses of your design, and any problems that your system encountered;

Because I have called the command line version of liblinear instead of the python wrapper, it takes some time to compute svms. On the other hand, I have avoided the problem of python using to much memory.

I have also reimplemented my pmf since my results for hw4 were not good. It is now getting 0.9 rmse and using stochastic gradient descent to train.