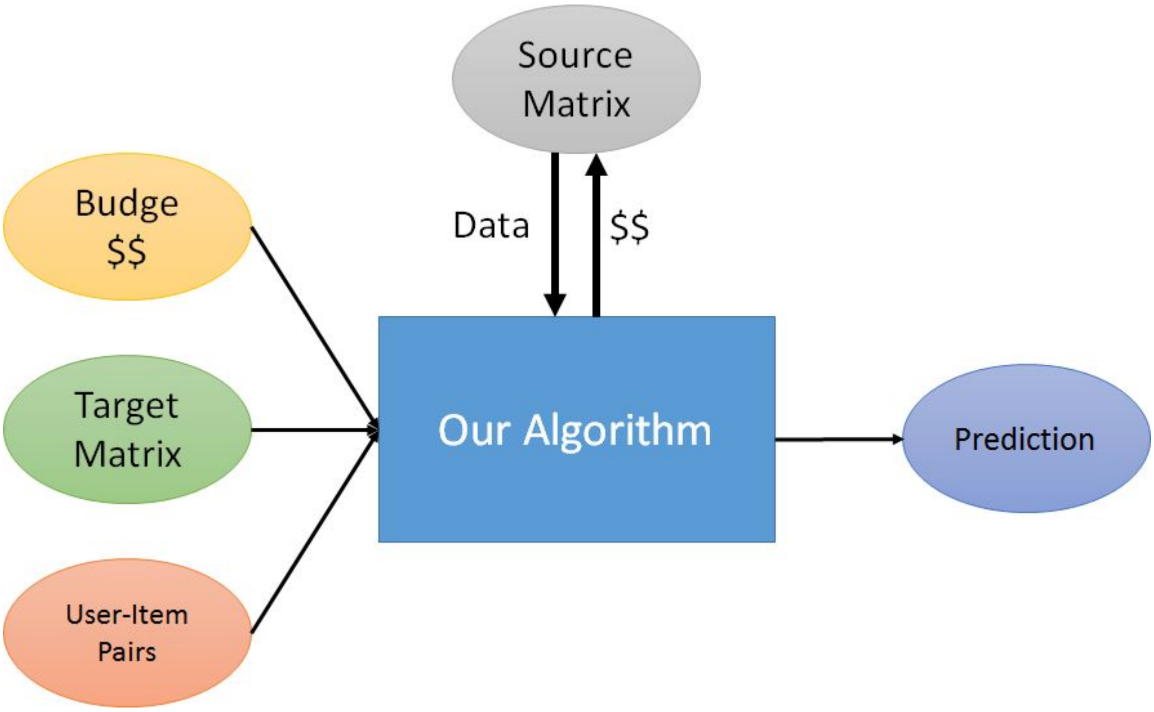
Machine Discovery (Fall 2016) - Final Project
No Free Source Matrix

隊名： No Free Source Matrix
R05922007 資工所碩一 林宗興
R05922037 資工所碩一 邱德旺
R05944018 網媒所碩一 賴至得

1. Problem Definition:

As the same setting in homework 3-1, we have the source matrix and target matrix. In homework 3-1, we can use all the rating data in source matrix without any cost. However, in this project, we assume that the source matrix rating data is not for free. We need the pay for it. Then, as a company, we have limited budget to get the rating data. Therefore, the goal of this project is to achieve the same or close performance as homework 3-1 with limited budget. The following figure is our model:



We have three inputs: budget , target matrix, and user-item pairs that need to be predicted. Then, we use the budget to query the data from source matrix. Finally, we use all this input to do the prediction.

2. Solution :

(1) Random Sample :

- Random sample the budget number of user/item rating data in source matrix.
- Use the sampled rating data as new source matrix to apply transfer learning method.

Following are transfer learning methods that we apply on the random sampled data:

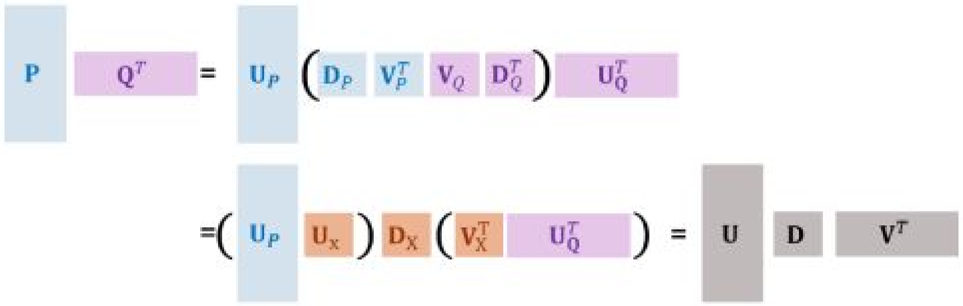
(a) Matching Matrix Method :

Use the idea of work by Li, Chung-Yi, and Shou-De Lin ACM 2014[1] to find the matching of users and items between source matrix and target matrix.

- Assumption :
  (i) The rating behavior of both domains are similar.
  (ii) There are some common users and common items across R1 and R2 , but we do not know the correspondence.
- Model :
  The goal is to find the matching of users and items between R1 and R2:

$$\hat{\mathbf{R}}_1 \approx \mathbf{G}_{\text{user}}\hat{\mathbf{R}}_2\mathbf{G}_{\text{item}}^T.$$

  (iii) Do matrix factorization on both R to obtain P and Q
  (iv) Do singular value decomposition on P and Q to obtain U, D, and V



  (v) Now the goal become:

$$\mathbf{U}_1\mathbf{D}_1\mathbf{V}_1^T \approx \mathbf{G}_{\text{user}}\mathbf{U}_2\mathbf{D}_2\mathbf{V}_2^T\mathbf{G}_{\text{item}}^T.$$

  (vi) Solve the following two objective functions:

$$\min_{\mathbf{G}_{\text{user}},\mathbf{S}_{\text{user}}} \|\mathbf{U}_1\mathbf{D}_1^{0.5} - \mathbf{G}_{\text{user}}\mathbf{U}_2\mathbf{D}_2^{0.5}\mathbf{S}_{\text{user}}\|_{\text{Fro}}^2$$

$$\min_{\mathbf{G}_{\text{item}},\mathbf{S}_{\text{item}}} \|\mathbf{V}_1\mathbf{D}_1^{0.5} - \mathbf{G}_{\text{item}}\mathbf{V}_2\mathbf{D}_2^{0.5}\mathbf{S}_{\text{item}}\|_{\text{Fro}}^2$$

  (vii) Pick the right G by minimize :

$$\|\mathbf{W}_1 \odot \left(\mathbf{R}_1 - \mathbf{G}_{\text{user}}\mathbf{U}_2\mathbf{D}_2\mathbf{V}_2\mathbf{G}_{\text{item}}^T\right)\|_{\text{Fro}}^2.$$

  (viii) Further refinement by pick top C nearest neighbors to satisfy the equation:

$$\min_{\mathbf{G}_{\text{user}},\mathbf{G}_{\text{item}}} \|\mathbf{W}_1 \odot \left(\mathbf{R}_1 - \mathbf{G}_{\text{user}}\mathbf{P}_2\mathbf{Q}_2^T\mathbf{G}_{\text{item}}^T\right)\|_{\text{Fro}}.$$

- Parameters :
  As we tuned in homework3, we pick the number of latent factors(K) as 200.

(b) Codebook Method :

- Assumption :

  (i) On the cluster level, the behavior of target domain and source domain is similar.

- Model :

  From the idea of codebook in [2], we use the orthogonal nonnegative tri-factorization algorithm in [3], that is,

$$\min_{F\geq 0, G\geq 0, S\geq 0} \|X - FSG^T\|^2, \;\; s.t.\, F^T F = I,\, G^T G = I.$$

  to tri-factorize the matrix in source domain.

  Then we use the matrix S (called codebook) to reconstruct the matrix in target domain. The goal is that using the cluster level codebook from one domain to help predict the behavior in the target domain that is more sparse. For random sampling on codebook, we could just random pick some users and only train the model base on those sampled users' score.

- Parameters :

  As we tuned in homework3, we pick the number of codebook dimension(k, l) as 20. And the iteration getting codebook is set to 3000.

(c) Transfer Probabilistic Collective Factorization Model (TPCF):

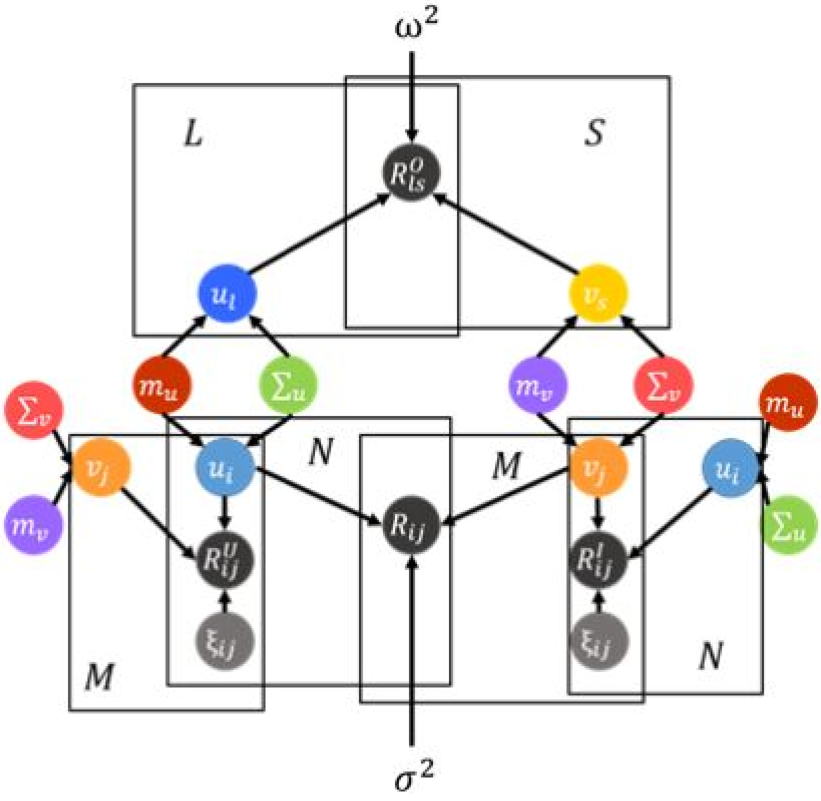
Following the idea from the work by How Jing, An-Chun Liang, Shou-De Lin, Yu Tsao in [6], where the authors strengthen the probabilistic matrix factorization by using information from auxiliary domains and transfer the latent representation of users and items from the one domain to the other. The following is the basic framework of the TPCF model.

- Data assumption:

  In the paper [4], the data is split into four user-item matrices, which are the target matrix R, three auxiliary matrices R^U, R^I, R^O. In R^U, R^I, elements are binary 0/1 standing for the implicit feedback, and in R, R^O, elements are discrete numbers corresponding to the rating values. In R^U and R, the user mapping is known. In R^I and R, the item mapping is known. In R^O, the users and items are different in R. In our tasks, we assign the target training data as R. Because we don't known the user and item mapping, R^U, R^I are empty. Also, we actively query source training data to R^O. The query size and the initialized size of R is 50.

- Model:

  TPCF model can be represented as the following probabilistic graphical model:



These colored nodes with the same colors are the same random variables. For rating matrices R, R^O, the distributions are modeled by Gaussian distributions. For implicit feedback matrices R^U, R^I, the distributions are modeled by Bernoulli distributions. The whole generative process is as follows:

For R, R^U, R^I:

(i) For each user i, generate
(ii) For each item j, generate
(iii) For each cell (i, j) in R, generate
(iv) For each cell (i, j) in R^U, generate
(v) For each cell (i, j) in R^I, generate

For R^O:

(i) For each user l, generate
(ii) For each item s, generate
(iii) For each cell (l, s) in R^O, generate

Where Bern() stands for the Bernoulli distribution,σstands for the sigmoid function. To train this model, variational expectation-maximization (VEM), an optimization approaches of PGM, is applied to search for the best parameters: m_u, m_v, $\sum_u$, $\sum_v$, ω, σ and the optimal values can be computed as the following forms:

$$m_u = \frac{1}{N+\beta L}\left(\sum_{i=1}^{N} \lambda_{ui} + \beta \sum_{l=1}^{L} \lambda_{ul}\right)$$

$$m_u = \frac{1}{M+\beta S}(\sum_{j=1}^{M} \lambda_{vj} + \beta \sum_{s=1}^{S} \lambda_{vs})$$

$$\Sigma_u = \frac{1}{N+\beta L}[\sum_{i=1}^{N}(\gamma_{ui} + (\lambda_{ui} - m_u)(\lambda_{ui} - m_u)^T) + \beta \sum_{l=1}^{L}(\gamma_{ul} + (\lambda_{ul} - m_u)(\lambda_{ul} - m_u)^T)]$$

$$\Sigma_v = \frac{1}{M+\beta S}[\sum_{j=1}^{M}(\gamma_{vj} + (\lambda_{vj} - m_v)(\lambda_{vj} - m_v)^T) + \beta \sum_{s=1}^{S}(\gamma_{ul} + (\lambda_{vs} - m_v)(\lambda_{vs} - m_v)^T)]$$

$$\sigma^2 = \frac{1}{A} \sum_{i=1}^{N} \sum_{j=1}^{M} Y_{ij}(T)[R_{ij}{}^2 + \lambda_{u_i}{}^T \gamma_{v_j} \lambda_{u_i} + \lambda_{v_i}{}^T \gamma_{u_i} \lambda_{v_i} - 2R_{ij} \lambda_{u_i}{}^T \gamma_{v_j} + (\lambda_{u_i}{}^T \gamma_{v_j})^2 + Tr(\gamma_{u_i} \gamma_{v_j})]$$

$$\omega^2 = \frac{1}{A} \sum_{l=1}^{L} \sum_{s=1}^{S} Y_{ls}(O)[R_{ls}{}^2 + \lambda_{u_l}{}^T \gamma_{v_s} \lambda_{u_l} + \lambda_{v_s}{}^T \gamma_{u_l} \lambda_{v_s} - 2R_{ls} \lambda_{u_l}{}^T \gamma_{v_s} + (\lambda_{u_l}{}^T \gamma_{v_s})^2 + Tr(\gamma_{u_l} \gamma_{v_s})]$$

For the details of the mathematics derivation, please refer to [4].

- Parameters:

  We use the source code provided in [4] and modify the parameters. We find that there are five parameters we can easily modify: α, β, d, max_iter, max_step, where α, β control the weight between target domain and auxiliary domains. d stands for the dimension of matrix factorization. max_iter is the maximal training iteration. max_step represents the number of E-steps in VEM. As we tuned in homework3, we pick α = 1.0, β = 0.05, d = 10, max_iter = 20, max_step = 5.

(2) Active Query on Codebook :

We have tried some active query algorithm on codebook. Instead of random sampling the users and spending all the quota we have in one query, maybe there are some more intelligent algorithm using those quota. We find that the performance of codebook transfer is highly related to the result of codebook construct. The idea is that more the data in the clusters, better result we will get because it will be more stable. We want to make each clusters more balance, so we need to reinforce the sparse one. It is trivial that we need to make some queries on items after some queries on users because only querying users is the same as random query. Now the question is how to select the set of items that we need to query next time. We propose three way trying to get a better codebook result then random sampling. They are

- Active query on low density user-item cluster
- Active query on low density item cluster
- Active query on small size item cluster

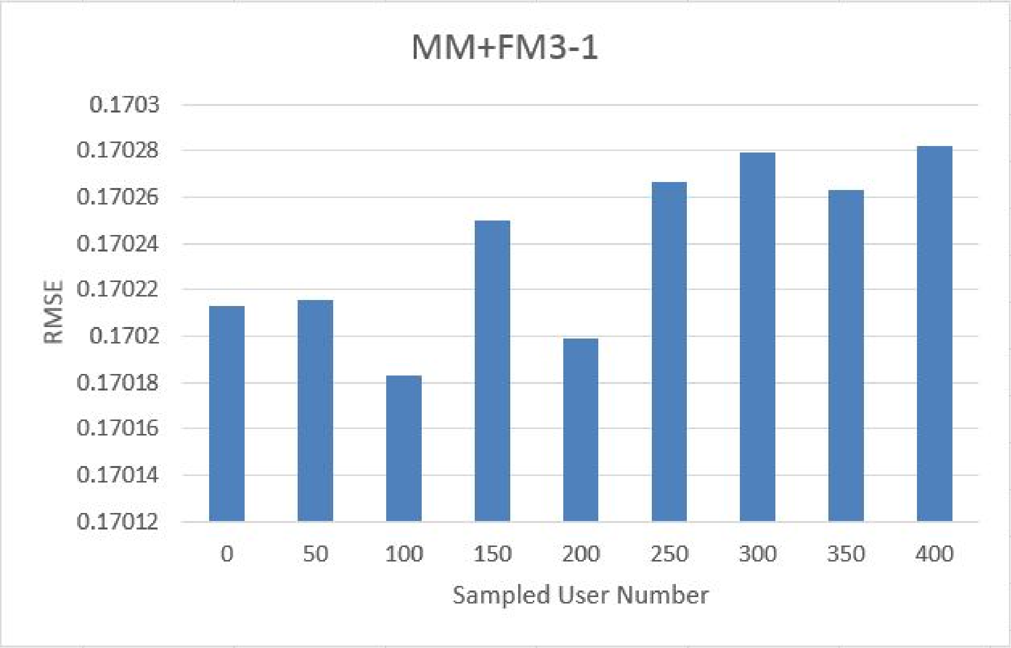
Two of them is based on density and one is based on size of the cluster. We try to answer that if these alternative query could improve the codebook performance under the same quota.

3. Experiment Result :

(1) Random Sample :

(a) Matching Matrix Method :

# of query	MM+FM3-1
0	0.170213054
50	0.170215777
100	0.170182763
150	0.170249556
200	0.170198856
250	0.170266685
300	0.170278736
350	0.170262987
400	0.170282195



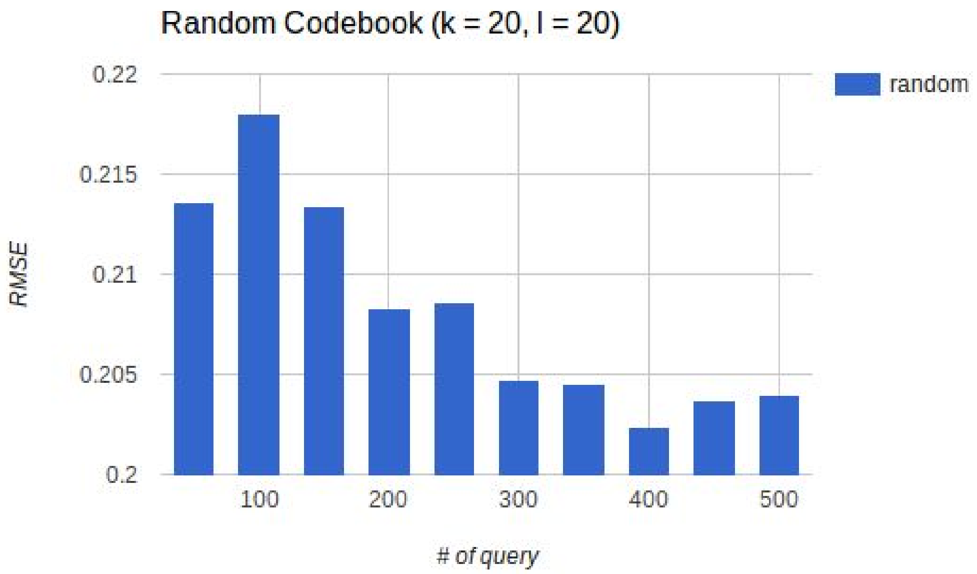
From the above results, we can see that the matching matrix method on this data perform not well. As sampled user number increasing, the RMSE is not decreasing subsequently.

(b) Codebook Method :
Below is the rmse performance of random sampling users on codebook.

# of query	random
50	0.2136
100	0.2180
150	0.2134
200	0.2083

250	0.2086
300	0.2047
350	0.2045
400	0.2024

And below is the chart format.

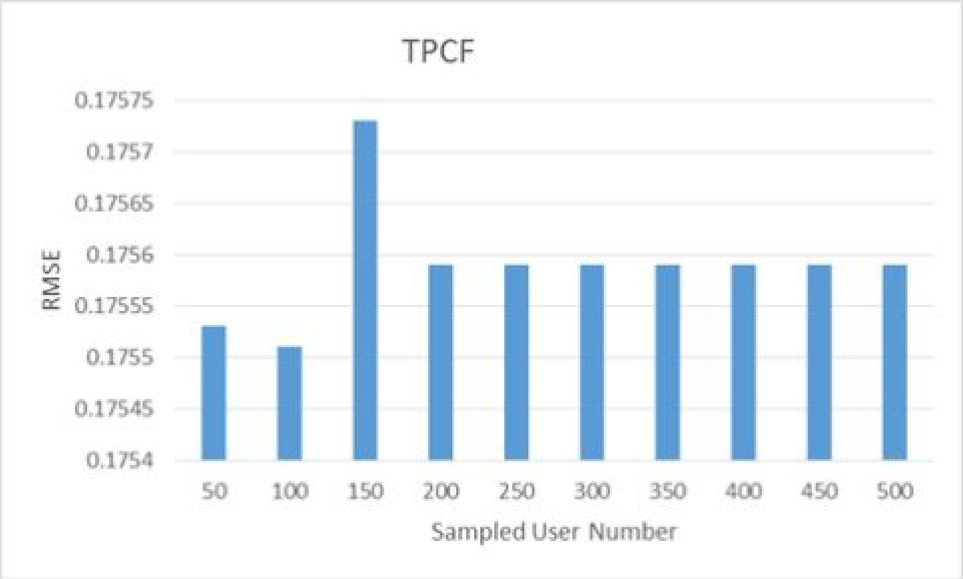


We could see that the RMSE does decrease when we increase our budget. So maybe the performance of active query is more significant in codebook transfer.

(c) Transfer Probabilistic Collective Model:
Below is the rmse performance of random sampling users on TPCF.

# of query	random
50	0.1755
100	0.1755
150	0.1757
200	0.1756
250	0.1756
300	0.1756
350	0.1756
400	0.1756

And below is the chart format.



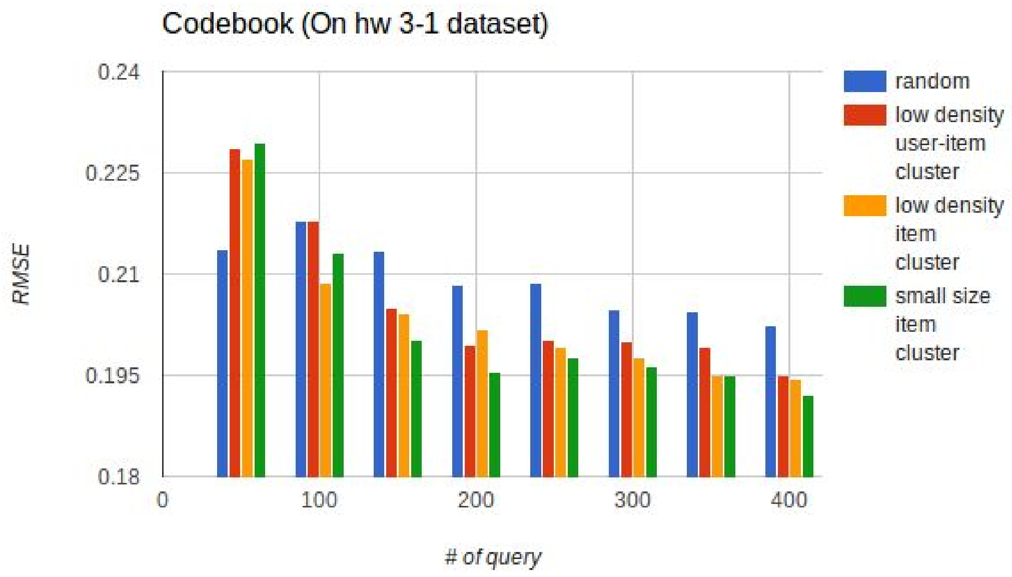
As the results, few information is brought from source domain to target by using random sample strategy on TPCF. After a few number of query, the model converges to some local optima and the transferred data does not affect the parameters of model..

(2) Active Query on Codebook :

For the experiment setting of codebook, we repeat codebook algorithm three times under each quota amount and each active algorithm we proposed, and then we report the average rmse score. Below is the rmse performance (rounded to the fourth digit after the decimal point) comparison between different sampling.

# of query	random	low density user-item cluster	low density item cluster	small size item cluster
50	0.2136	0.2287	0.2271	0.2293
100	0.2180	0.2178	0.2087	0.2132
150	0.2134	0.2049	0.2042	0.2002
200	0.2083	0.1995	0.2019	0.1954
250	0.2086	0.2001	0.1993	0.1976
300	0.2047	0.2000	0.1977	0.1962
350	0.2045	0.1992	0.1951	0.1949
400	0.2024	0.1950	0.1945	0.1920

And below is the chart format.



It is obvious that those three trick could really help improve the performance of codebook transfer. Under our setting the third active method could have near 0.01 RMSE score improvement on the quota amount 400.

On the other hand, it is interesting that what the performance is if we have more budget. We only see that the performance before 400 time queries, but we are not sure if we will have better result by these sampling algorithm comparing to using all scores in matrix.

4. What we have learned :

(1) Not all the data is fitted for the certain transfer learning methods.

(2) It seems that active query in each data or model setting is diverse. And it may be not easy to apply one setting to another model.

5. Future Work :

(1) Apply other transfer learning method on random sampled rating data.

(2) Apply other active learning method to achieve our goal.

6. Contribution of each member:

(1) R05922007 林宗興:

   (a) codebook algorithm on random sampling

   (b) active query method

   (c) write report

(2) R05922037 邱德旺:

   (a) MM(Matching Matrix) on random sampling

   (b) write report

(3) R05944018 賴至得:
    (a) TPCF (Transfer Probabilistic Collective Factorization) on random sampling
    (b) write report

7. Reference :

[1] Li, Chung-Yi, and Shou-De Lin. "Matching users and items across domains to improve the recommendation quality." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.

[2] Li Bin, Qiang Yang, and Xiangyang Xue. "Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction." IJCAI. Vol. 9. 2009.

[3] Ding, Chris, et al. "Orthogonal nonnegative matrix t-factorizations for clustering." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006.

[4] Liang, An-Chun, Shou-De Lin, and Yu Tsao. "A transfer probabilistic collective factorization model to handle sparse data in collaborative filtering." 2014 IEEE International Conference on Data Mining. IEEE, 2014.