Transfer Learning from User's Mobile App Profile to Content Profile

CHEN Yuxiang
ycheneu@connect.ust.hk
ycheneu - 20448351

LIN He
hlinam@connect.ust.hk
hlinam - 20476097

XU Mengdi
md.xu@connect.ust.hk
mxuas - 20471255

ZHENG Dingyi
dzhengaf@connect.ust.hk
dzhengaf - 20470081

Hong Kong University of Science and Technology

ABSTRACT

A major assumption in machine learning tasks is that there are sufficient amount of data for the algorithms to learn from. However, collecting data is usually costly and time consuming. It would be very helpful if we could make good use of the existing data. In such cases, transfer learning between tasks and domains would be ideal.

Keywords

Transfer learning, deep learning, machine learning, recommender systems.

1. INTRODUCTION

A major assumption in machine learning tasks is that there are sufficient amount of data for the algorithms to learn from. However, collecting data is usually costly and time consuming. It would be very helpful if we could make good use of the existing data. In such cases, transfer learning between tasks and domains would be ideal.

The dataset in this project is provided by Cheetah Mobile, containing sufficient amount of users' preferences data on mobile applications, which we call it "source domain"; and very limited information on users reading preferences, which we call it "target domain". The sparsity of the rating matrix between users and articles leads to a poor performance in providing a list of recommended articles to each user. However, transferring knowledges from users' preferences on mobile applications to their interests in readings may help in addressing such difficulties. In this project, we will apply the ideas of transfer learning (TL) in building a recommender system (RS).

2. RELATED WORKS

Traditional machine learning requires consistency on source and target learning settings. Transfer learning allows the diversity of the domains, tasks, and distributions used in training and testing. Depending on various learning settings, transfer learning can be categorized into *inductive transfer learning* (i.e. same domains, different but related tasks), *unsupervised transfer learning* (i.e. different but related domains and tasks), and *transductive transfer learning* (i.e. different but related domains, same tasks). [1]

In this project, we use APP profile data as source and reading preferences as target. The target task is to recommend a list of articles of interests by analyzing their preferences on mobile APPs. The type of project has fallen into inductive transfer learning, as neither domains nor tasks are the same for source and target. We would like to focus more on the techniques used in inductive transfer learning.

2.1 Approaches of Inductive Transfer Learning

Based on different settings, there are four approaches for inductive transfer learning – instances, feature representations, parameters, and relational-knowledge transfers. As the name suggests, relational-knowledge transfer approach deals with relational domains. Since the data we will be working on is assumed to be independent and identically distributed, the method is not applicable.

Instance-transfer is under the assumption that the source domain, or at least part of them can be used in the target domain. A classical algorithm for Instance-transfer is called *TrAdaBoost* proposed by Dai et al. [2]

Transferring feature representations can be applied to both supervised and unsupervised learning problems. The idea is to find some "good common features" for both domains to minimize domains divergence. [3]

Finally, parameter transferring is very similar to parameter based Multitask Learning (MTL). The only difference is that Parameter-transfer method in transfer learning attaches little significance towards the behavior in source domain, while the MTL improves performance on both tasks simultaneously.

2.2 Transfer Learning in Recommender System

After introducing some basics of transfer learning, we are ready to explore more on applying transfer learning to recommender systems.

2.2.1 Major Recommendation Algorithms

Collaborative filtering (CF) is the most used in digging patterns of users and clustering similar items in user-item rating matrix. The advantage of CF is that the excellence of the item can be estimated through user ratings and it has no special restrictions on items' property. However, the problems caused by cold start, data sparsity is also obvious and are main challenges projects of this kind. Moreover, stability and scalability issues also need to be aware of.

2.2.2 Relationship Between Mainstream Approaches of RS

According to paper [5], people usually build a recommender system with the help of hybrid filtering system, content-based filtering, collaborative filtering, and demographic filtering. Due to the sparsity of rating matrix, applying dimensionality reduction techniques such as principal component decomposition and LSI first can be very helpful.

Subsequently, the sparsity problem can be further improved by collective matrix fatorization(CMF). Paper [6] makes use of tag information to deal with this problem in the user-item matrix. Firstly, a dense user-tag matrix and a sparse user-item matrix have been constructed to represents users' preferences about tags, and then the user-item matrix which is sparse can by converted into a dense one through learning the user-tag matrix in the process of CMF.

2.3 Deep Learning in Recommender System

Especially with deep neural networks (DNN), it is accessible to obtain the corresponding latent features, based on which we can make prediction by matrix product computation.

The first neural network based model is the Neural Collaborative Filtering (NCF)[7]. It uses a simple neutral network structure to measure the two-way interaction between users' preferences and items features.

Then there are two representative papers [8] and [9], which are about Solely MLP approach and have the state-of-arts result.

For Deep Factorization Machine (DeepFM)[10] which alleviate the feature engineering cost of wide & deep learning, is an end-to-end model integrates factorization machine and MLP by the inspiration of wide & deep model . It can model the high-order feature interactions via deep neural network and low-order interactions via factorization machine.

Deep Learning has been applied to deal with large scale dataset in paper [11]. Two stages are designed with the purpose that the data scale being reduced in the first stage and then the output being generated from the second stage.

2.3.2 Cross Domain Recommendation(related)

The works of this paper [13], Multi-View Learning recommendation system, based on deep structured semantic model (DSSM), is highly related with our projects objective.

To achieve this target, it is important to map the k items and user feature to the same semantic space, then measure the cosine distance between mapped user vector and mapped item vectors, decide whether user is interested in this item or not. Finally, we pick the (N-1) * top-k item which have smallest distance to pivot view feature(user) as our recommendation for users.

3. SOLUTION OVERVIEW

3.1 Data Preprocessing

There are near 8000 users in the training set, each user has a list of mobile applications that they have downloaded. Over 6000 users have read at least one article and there are 140,000 pairs of user-article reading record in total, each labeled with a reading time (dwelling time).

We split original dataset into 3 specific datasets: user profile, app profile, article info.

User profile: for each user, it includes 9 personal features: gender, age group, mobile brand, mobile model, country minded, state minded, etc.

App profile: for each user, it includes the all package names of applications he downloaded, and the frequency of each application category.

Article info: for each reading record, we collect the information including user id, article id, article l1 category, article l2 category, article dwelling time, article title, etc.

There are some missing values or feature, which should be handled delicately.

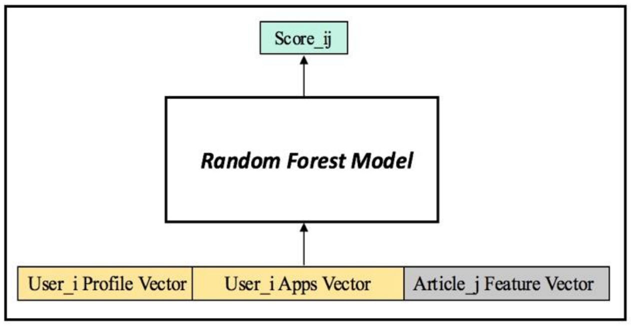
3.2 Supervised Learning Model

We chose random forest algorithm, a simple supervised learning model, as our baseline model because it can take in both numerical data and categorical data. We assumed that the longer a user spends on an article, the more they enjoy the article. Based on this assumption, user's reading time on each article in the test set. Five articles with the longest predicted reading time are to be presented to the corresponding users.

While training, we fed the model with user profiles data (e.g. age, gender, location, etc.), article features, and app-installation information. The training target was the reading time of each user-article pair. For each numerical feature, values below the 5th percentile or over top 95% are treated as outliers and has been set to 0 or 1, respectively. The rest of the data (i.e. inliers) is normalized to be between 0 and 1.

Since there are 8,000 users and around 40,000 articles in training set, it is infeasible to feed the model with every user-article pairs. Therefore, we only selected articles that had been read by more than two users to be paired up with the users.

The model precision is 0.167% and recall is 0.058% for top 5 recommendations.



3.3 Pattern Transfer Learning

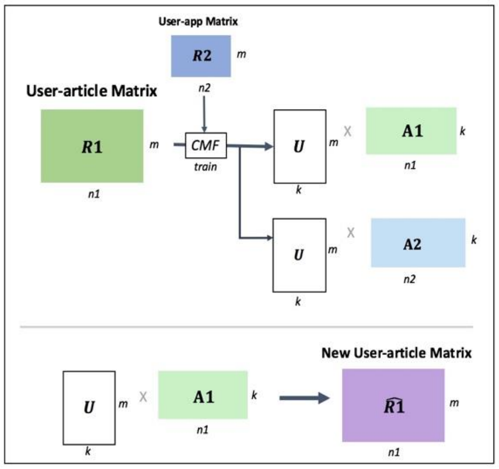
Since the features in our dataset is way too sparse, standard collaborative filtering methods is not applicable. Therefore, we proposed a new method using collective matrix factorization which can be applied on our specific given datasets. We decompose simultaneously two rating matrices: user-app matrix and user-article matrix. The major idea of the model is originally from the fact that the user ids of two rating matrices are identical. By this method, we can transfer some app rating patterns of users to predict their rating scores on new articles. Non-sparse app matrix contained huge amount of information, which can be transferred as latent factors of each users.

In this method, we applied collective matrix factorization on user-app rating matrix R2 and user-article rating matrix R1 simultaneously. By minimizing the regularized square error on the set of known ratings, it generates 3 feature matrices characterizing apps, articles and users by k-dimensional vectors of factors inferred from item rating patterns. In other words, the algorithm implements mappings of these 3 different entities into the same feature space. Latent feature matrices contain integrated information from the other domain. Thus, we calculate dot product to obtain a denser predicted user-article rating matrix. The final product represents users' preferences on each article.

We define the loss function as follows:

$$\text{loss} = \frac{1}{2}\left\|J \circ \left(R_1 - UA_1^T\right)\right\|^2 + \frac{\alpha}{2}\left\|R_2 - UA_2^T\right\|^2 + \frac{\beta}{2}(\|U\|^2 + \|A_1\|^2 + \|A_2\|^2)$$

we are focusing on decreasing the difference between prediction matrices and original matrices. At the same time, we add some regularization penalty to keep its generalization ability.



We use Tensorflow's Optimizer to train the 3 feature matrices. Tensorflow can speed up the algebra computation and transformation by some inner optimization based on CPU or GPU multi-threading programming. The dot product between big matrices is faster than functions for computation in Numpy and Scipy library. firstly

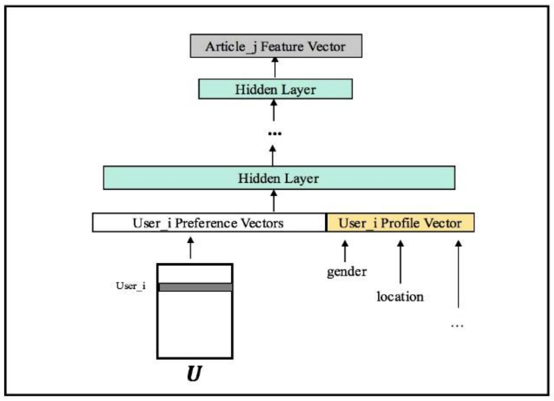
We initialized 3 feature matrices with random normal-distribution numbers range from -0.5 to 0.5, and used SGD optimizer at beginning of training phase to speed up. Afterwards Adagrad optimizer was applied to decrease loss steadily and to avoid converging at a suboptimal result.

The model has 0.374% in precision and 0.129% in recall, which is better than the baseline.

We did not use any information other than the user-applications and user-articles relationships. However, we have more information regarding to users and articles. Therefore, we proposed a third method, which includes more user information than solution two.

3.4 Domain Transfer Learning

The idea of recommender system is to map users and items (i.e. articles) into the same space, by finding the most similar user-article pairs we can thus recommend the article to the corresponding user.



Based on this idea, we constructed our third method in a way that is totally different from the two methods described before. Since the output matrices in method two had encoded users' preferences on mobile applications and how the articles are being liked. The output matrices can be treated as the embedded preference vectors of each user, or the feature vector for each article.

The input for model 3 is the users' information, which consists of two parts – preference vector and profile vector. It outputs the embedded feature vector of the user's favorite article.

It is controversial when defining what is a user's favorite article. In our experiment, we measured how much a user likes an article by the time a user spent on an article in the past period. We first set a time threshold to filter out the reading histories that is too far away from current time. Then, the article with the longest dwelling time from the surviving article set was chosen to be the user's favorite.

In the inference, the model outputs an embedded vector which represents what the user's favorite article should look like. We then selected the top K (we chose K to be 3 in our model) articles with embedded vectors the most like the predicted output. (Here we implemented a modified K-Nearest-Neighbor (KNN) algorithm – a weighted-KNN, which will be descripted in detail later.) We can find the articles that are mostly likely to be the users favorite article from the training set. Then we calculated a weighted sum for each attribute in the selected articles to get an estimated profile for the user's favorite article. By comparing the estimated favorite profile with the profiles information of articles from testing set, we can finally find the user's favorite article. We present five most similar articles to each user, correspondingly.

The weighted-KNN is a modified KNN algorithm, which can take in both numerical and categorical data with a weight. Since embedded vector of categorical data such as users' states of origins can have length 20, but it only represents one feature. It may dilute the importance of categorical data such as age, language, mobile's brand, which are all of 1 dimensional. Therefore, we need to set a weight for each feature depending on the length of representing vectors.

The performance of this model on training set is no better than the second model. It is as expected because there are too many things can be tuned to get a better performance. The method for selecting a user's favorite, the embedding methods, and the parameters as well as the activation functions can all be modified. We would fine tune all of them if we could have more time to work on the project, but for now we just leave it as is. Thus, we did not bother to test the performance of it on test set.

This model included more information as training and predicting features than the previous models. However, it did not use the title information for articles. Thus, we would like to include title information in our next model, which will be described below.

3.5 Enhancing with Deep Learning on Scaling Issues

To include title information in the model, each user must be paired up with all the articles. Considering there are 8000 users and 40,000 articles on records, it is not feasible to use the fully paired-up samples. In fact, only selecting the articles that has been read by two users still creates a large article candidate set. To address this problem, we proposed a two-staged deep recommender system, where the first stage is a deep candidate selection procedure and the second stage is a deep ranking model.

The first stage takes in users' profile data and their embedded preference vectors as input. It outputs a category-probability vector for everyone, representing that categories of articles the user might be interested in. The model emphasizes on sampling articles in the categories with a higher probability. The selected articles will be paired up with the corresponding user and be put into the second deep ranking model.

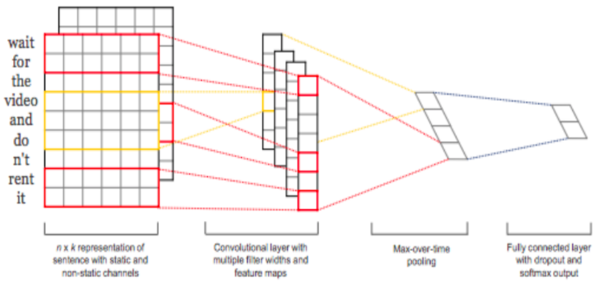
The second stage inputs is the filtering result of the stage one algorithm, that help us to shrink the computation and memory loads, in second stage networks we take users profile data ,user preference information and each articles information as input to predict the each user's preferabilty about each articles, which we modeled by the reading time of each article for that user.

To implement stage 2 network we used an 8 hidden layers fully connected neutral networks using RELU activation function.

Article titles feature extraction:

We firstly tokenized the title, cleaned text data, check the word spelling and done the named entity tagging use NLTK and Stanford NER library.

Then We used CBOW model trained in a large corpus to embed each word to a fixed length vector (300 dim), finally we got matrixes for each titles, we tried to use convolutional neutral network[14] to model the article title information


wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

So finally we can have an fixed length feature vector for each titles, and concatenate our CNN output with other features as the input of our next fully connected network, when we training we can update CNN and FCN's weight together.

The most popular category contains over 2,7000 out of 40,000 articles in training set. The article data are extremely biased in terms of categories. It would be better if we can reclassify the articles with some hierarchical clustering methods. Besides the original articles information that had already been used, some additional features for reclassifying can be used are the number of times each article is being read, the characteristics for the readers (by keep tracking of the readers profile data), etc.

4. Evaluation Metric and Results

4.1 **Experimental step**

-run the model prediction on our validation set and got precision and recall scores.

-use KNN algorithm to pair similar test data and train data.

-run the model prediction on test set, and get feedback from teaching assistant.

4.2 **performance metrics**

We used precision and recall for the top 5 recommendations in our model.

Precision at k is the proportion of recommended articles in the top-k set that are liked. The mathematical definition is:

$$\text{precision@k} = \frac{\text{\# of recomended articles @k liked}}{\# \, of \; recommended \; articles \; @k}$$

Recall at k is the proportion of favored articles found in the top-k recommendations. The mathematical definition is:

$$\text{Recall@k} = \frac{\text{\# of recomended articles @ liked}}{total \text{ \# of liked articles by user}}$$

4.3 **experimental results and analysis**

The performance of model 3 on training set is no better than the second model. It is as expected because there are too many things can be tuned to get a better performance. The method for selecting a user's favorite, the embedding methods, and the parameters as well as the activation functions can all be modified. We would fine tune all of them if we could have more time to work on the project, but for now we just leave it as is. Thus, we did not bother to test the performance of it on test set. It is similar for model 4. Therefore, we will report the testint result for model 1 and 2, as well as the test performance for randomly selected samples.

Models	Validation set		Testing set	
	Precision	recall	precision	recall
Random selection	N/A	N/A	0.00061	0.00061
Model1 (Baseline)	0.00166	0.00057	0.00067	0.00087
Model2 (CMF)	0.00373	0.00129	0.00074	0.00044

5. Conclusion and Future Work

We have experimented on multiple models at various complication level – traditional simple supervised learning model (i.e. random forest), pattern transfer learning (i.e. collective matrix factorization), domain transfer learning, and a two-staged deep recommender system.

There a lot of improvements can be made in each model. The computation speed in Model 2 can be improved by using some other loss functions, which is described earlier. It is possible to try out different metric for defining a user's "favorite article" in model 3. At stage 1 in model 4, we can reclassify articles using hierarchical clustering methods, instead of predicting the categories that a user might be interested in. The embedding methods for users and articles can also be modified for both model 3 and model 4. For example, we can encode the information using autoencoder.

Since model 3 and model 4 were based on the result from model 2, model 2 is essentially a part of the latter models. Thus, it is highly expectable to get a bad result from the latter models, due to lack of fine tuning for complex models.

6. REFERENCES

[1] Qiang Yang, Sinno Jialin Pan, "A Survey on Transfer Learning", IEEE Transactions on Knowledge & Data Engineering, vol. 22, no., pp. 1345-1359, October 2010, doi:10.1109/TKDE.2009.191

[2] W. Dai, Q. Yang, G. Xue, and Y. Yu, “Boosting for transfer learning,” in Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, USA, June 2007, pp. 193–200.

[3] A. Argyriou, T. Evgeniou, and M. Pontil, “Multi-task feature learning,” in Proceedings of the 19th Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 2007, pp. 41–48.

[4] Bobadilla J, Ortega F, Hernando A, et al. Recommender systems survey[J]. Knowledge-based systems, 2013, 46: 109-132.

[5] Bobadilla J, Ortega F, Hernando A, et al. Recommender systems survey[J]. Knowledge-based systems, 2013, 46: 109-132.

[6] KIM, Bu Sung, et al. Improving a recommender system by collective matrix factorization with tag information. In: Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on. IEEE, 2014. p. 980-984.

[7] He X, Liao L, Zhang H, et al. Neural collaborative filtering[C]//Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2017: 173-182.

[8] Cheng H T, Koc L, Harmsen J, et al. Wide & deep learning for recommender systems[C]//Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 2016: 7-10.

[9] Guo H, Tang R, Ye Y, et al. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction[J]. arXiv preprint arXiv:1703.04247, 2017.

[10] COVINGTON, Paul; ADAMS, Jay; SARGIN, Emre. Deep neural networks for youtube recommendations.
In: Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 2016. p. 191-198.

[11] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In Proceedings of the 24th International Conference on World Wide Web. ACM, 111–112.

[12] Florian Strub, Romaric Gaudel, and J´er´emie Mary. 2016. Hybrid Recommender System based on Autoencoders. In Proceedings of the 1st, Workshop on Deep Learning for Recommender Systems. ACM, 11–16.

[13] Florian Strub and Jeremie Mary. 2015. Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs. In NIPS Workshop on Machine Learning for eCommerce.

[14] Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014). APA

About the authors:

Author 1 Yuxiang Chen is a masters of Big Data Technology student in Hong Kong University of Science and Technology. He obtained his bachelor degree in Mechatronics in Zhejiang University. He had summer internship about image processing and object detection.

Author 2 He Lin is a masters of Big Data Technology student in Hong Kong University of Science and Technology. He obtained his bachelor’s degree in Computer Science from Macau University of Science and Technology. He had several project experience and internships about image segmentation and object detection.

Author 3 Mengdi Xu is a masters of Big Data Technology student in Hong Kong University of Science and Technology. She graduated from the University of Toronto with an honors Bachelor of Science degree (double-major in Mathematics and Computer Science) in summer 2017.

Author 4 Dingyi Zheng currently studies in Hong Kong University of Science and Technology, is working on her Master degree of Science in Big Data Technology program. She graduated from Wuhan University with a Bachelor’s degree in Information Management and Information System in 2016, and worked as a full-time Image Algorithm Engineer for a year before she came to Hong Kong.