CSE 291-D: Project Proposal

Po-Ya Hsu [A53202971] **Yang Liu [A53206488]** **Hao-en Sung [A53204772]**

1 Project Goal and Motivation

Goal: Implement temporal dynamics to do rating predictions. Try to seek the causality of the temporal variance, or the hidden temporal pattern.

Motivation: Prediction of customer's ratings is an important part of a recommending system. Based on the potential rating that a customer will give to a specific product or place, recommending system can improve its recommendation behavior and make potential commerce deal happen. Also, accurate recommendation can provide user with better experience and thus gain user's loyalty.
However, people's preferences or some hidden factors can vary with time, so we should add time factor to make a better prediction. With temporal information, the prediction become beneficial not only to the restaurants but the users.

2 Dataset

We decide to use the Yelp dataset, which is available online [1]. They consist of text-based data (1.8G compressed, 4.6G uncompressed), and image-auxiliary data (5.8G compressed). Text-based data include basic business information, i.e. location and business hour, customer reviews, encrypted customer information, check-in information, and amount of tip.

3 Proposed Model

Latent variable model with temporal dynamics

$$f(u, i, t) = \alpha + \beta_u(t) + \beta_i(t) + \gamma_u(t) * \gamma_i(t),$$

where $f(u, i, t)$ is the predicted rating, $\beta_u(t)$ is a vector associated with users and $\beta_i(t)$ is a vector associated with items; γ_u and γ_i map users and items into the latent space. We plan to use stochastic gradient descent (SGD) to optimize our model and use mean squared error (MSE) as the evaluation metric.

4 Experiment Design

Step 1. Preprocess the data. For instance, remove bad data.
Step 2. Implement the proposed model to different time sections. For time sections, they can be decided based on autoregressive models or self-defined priors.
Step 3. Evaluate the model with different sliding windows and well-split train-test set. Find the causality of the temporal changes with the usage of text-based data.

5 Reference

[1]Moore, Joshua L., et al. "Taste Over Time: The Temporal Dynamics of User Preferences." ISMIR. 2013.

[2]Reuning, Kevin, Michael R. Kenwick, and Christopher J. Fariss. "Exploring the Dynamics of Latent Variable Models." (2016).

[3]Koren, Yehuda. "Collaborative filtering with temporal dynamics." Communications of the ACM 53.4 (2010): 89-97.

[4]Talmon, Ronen, et al. "Manifold learning for latent variable inference in dynamical systems." IEEE Transactions on Signal Processing 63.15 (2015): 3843-3856.

[1]https://www.yelp.com/dataset_challenge/dataset

CSE 291-D: Progress Report

Po-Ya Hsu [A53202971] **Yang Liu [A53206488]** **Hao-en Sung [A53204772]**

1 Project Goal

Our goal is to come up with a model that can properly describe dynamice time series Yelp dataset.

2 Progress Made

Big Picture

1. Yelp Dataset Exploration
2. Simulations on Dynamic Time Warping (DTW) Algorithm with time domain signals
3. Matrix Factorization on user / business ratings

Details attached on the following pages

3 Results Obtained

1. Made decisions on the useful features provided, such as date, 'Restaurant' category, and rating
2. Able to tell how sampling rate, phase shift and magnitude change the optimal time warping path and cost

4 Challenges Faced

1. Some businesses have a small number of reviews, which may not be reliable
2. To process Yelp businesses' time clustering, the cost function / distance distribution has to be redefined

5 Proposed Solutions

1. Remove businesses by setting reviews' number threshold, plus take only the users that have written sufficiently large number of reviews into account
2. Modify the cost function to binarize elementwise product of the signals, and paper study on DTW paths

6 Related Works

6.1 [Paper 0]

Senin, Pavel. "Dynamic time warping algorithm review." Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA 855 (2008): 1-23.

7 Data Analysis

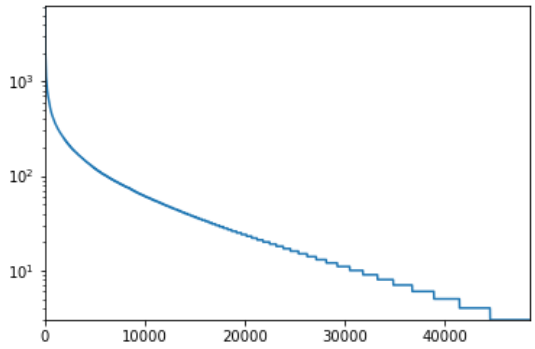
In this section, we are curious about the importance of time factor to Yelp review dataset. Before starting the analysis, we filter out those reviews for restaurant with qualified reviews. After that, we first observe the review distribution. Then, we plot one representative restaurant and representative user associated with most reviews against the time factor.

7.1 Preprocessing

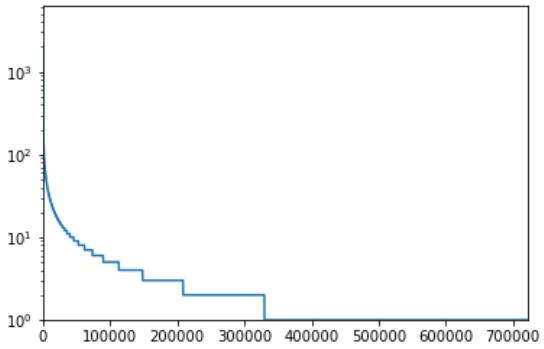
1. Filter out non-restaurant business. After reducing the number of businesses from 144072 to 48485, the overall number of reviews decreases from 4153150 to 2577298.
2. Filter out restaurants with less than 10 reviews, which reduces the number of restaurants decreases from 48485 to 31792.
3. Filter out users providing leas than 5 reviews, which reduces the number of users from 707751 to 109160.
4. Remove those unqualified restaurants and users, which reduces the number of reviews from 2577298 to 1532164.
5. Separate training and testing sets with time threshold — year 2015. At the end, there are 1206403 reviews in training set and 325761 reviews in testing set.

7.2 Review Distribution

From Fig. 1a and Fig. 1b, we know that the distribution of number of reviews for both businesses and users are log-linear. It means that few restaurants and users are associated many reviews, while the rest of them only are only related few reviews.



(a) Review Distribution for Restaurants



(b) Review Distribution for Users

7.3 Representative Restaurant

Here, we are interested in the restaurant with most reviews — *4JNXUYY8wbaaDmk3BPzlWw*, which receives 6414 reviews. From Fig. 1, one can tell that there is a pattern for the rating across months. In other words, adding temporal dynamics information might be helpful for modeling this restaurant.

7.4 Representative User

Similar to the restaurant, Fig, 2 shows that there is also a clear pattern here.

8 Experiments

In our experiment, hashed user identity and hashed business identity are provided as features; while our target score is the rating from one user identity to one business identity, which is a real number ranging between 0 and 5.

In view of the dataset property, we decide to use latent variable based model to solve Before building up a complex model with temporal information, we would like to first run some simpler model as baseline algorithms, including matrix factorization with and without latent vectors.

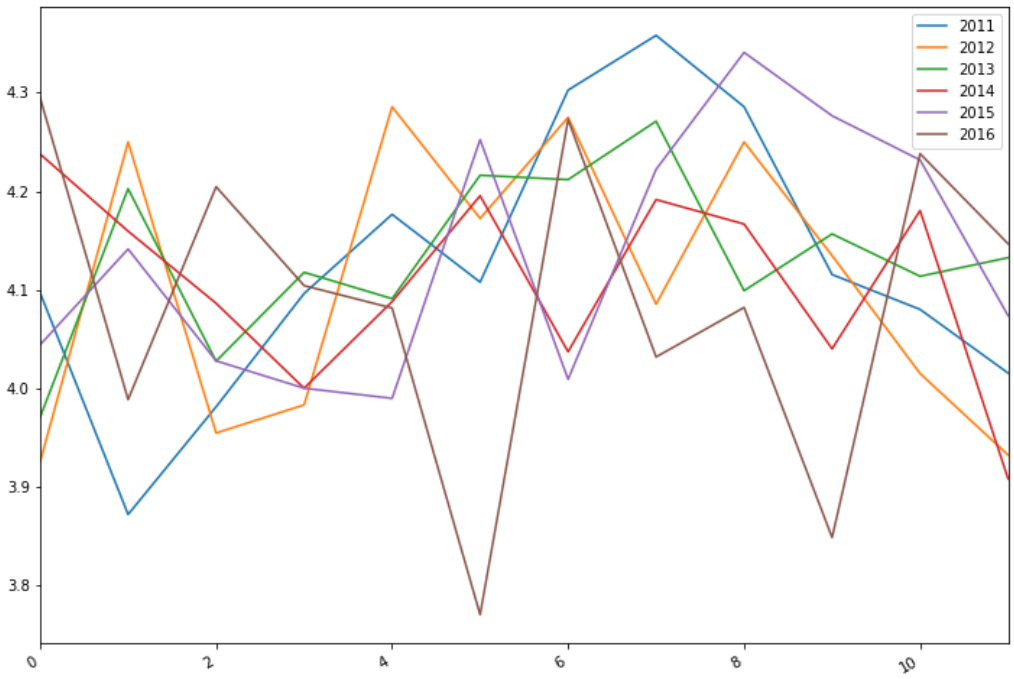


Figure 1:

8.1 Baseline Algorithm: Matrix Factorization with Biased Terms only

The model is in the form of

$$r_{u,i} \approx a + \boldsymbol{b}_u + \boldsymbol{b}_i,$$

where

u is the user identity
i is the business identity
$r_{u,i}$ is the rating from u to i
a is the general biased term
$\boldsymbol{b}_u$ is the biased term for u
$\boldsymbol{b}_i$ is the biased term for i

Since we do not plan to run rigorous parameter tuning for baseline algorithm currently. We directly run the baseline model on both training and testing data and the results are shown in Table 1.

8.2 Baseline Algorithm: Matrix Factorization with Additional Latent Vectors

The model is in the form of

$$r_{u,i} \approx a + \boldsymbol{b}_u + \boldsymbol{b}_i + \lambda * \boldsymbol{P}_u \boldsymbol{Q}_i^{\intercal},$$

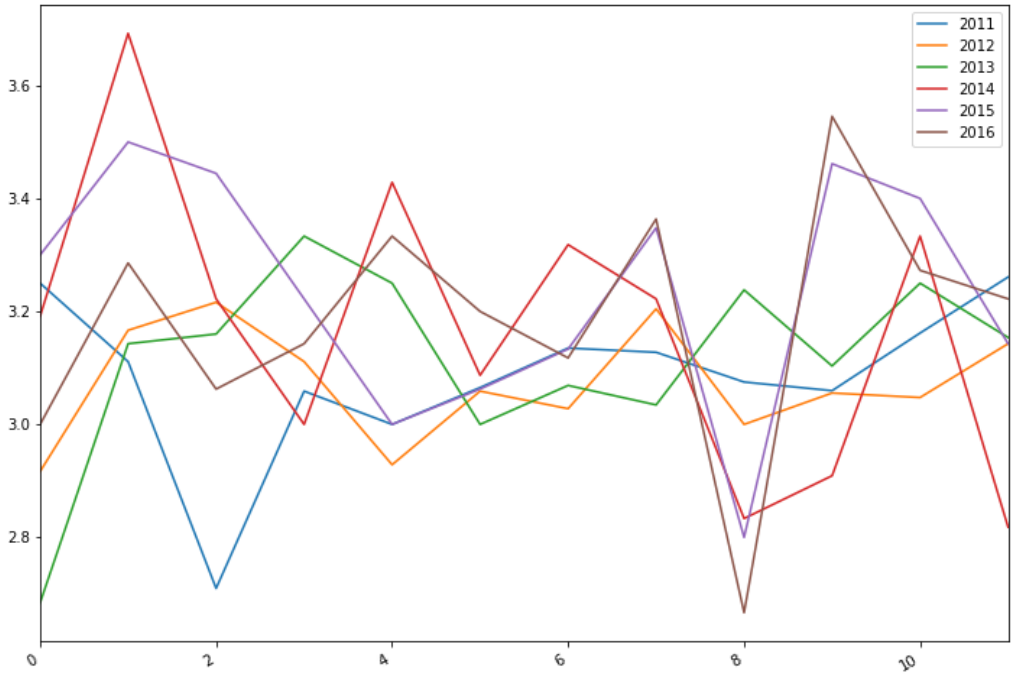


Figure 2: Review Distribution for Users

Table 1: Results for Matrix Factorization with Biased Term Only

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.438498	1.602756	1735408.498650	522115.405951
3	1.007219	1.379669	1307312.460167	541642.764160
6	0.994037	1.371987	1286111.631646	533842.139729
9	0.993159	1.372219	1285061.203351	533926.725984
12	0.992963	1.372622	1284690.702078	533924.146053
15	0.992879	1.373039	1284442.179888	533912.316899
18	0.992826	1.373450	1284257.141350	533925.016696
21	0.992786	1.373846	1284114.588497	533960.292843
24	0.992753	1.374224	1284002.574620	534011.157873

where

λ is the meta parameter for latent vectors

$\boldsymbol{P}_u \in \mathbb{R}^K$ is the latent vector for u

$\boldsymbol{Q}_i \in \mathbb{R}^K$ is the latent vector for i

K is the meta parameter for latent vector length

Similar to what we show in previous sections, we now work on Matrix Factorization with addition latent vectors. The results can be found in Table 2

Table 2: Results for Matrix Factorization with Additional Latent Vectors

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.440934	1.604356	2208263.176994	992552.624725
5	1.020058	1.390369	1381831.668316	604158.455929
10	1.007811	1.383391	1340408.290566	575236.520401
15	0.994730	1.383236	1330056.456926	580615.442759
20	0.987506	1.383797	1326401.451475	585858.703958
25	0.983849	1.384413	1324877.259138	588946.289266
30	0.981746	1.384997	1324018.593771	590815.710580
35	0.980341	1.385546	1323427.806758	592097.778377
40	0.979304	1.386056	1322976.763243	593064.602866

8.3 Dynamic Time Warping Algorithm Simulation Results

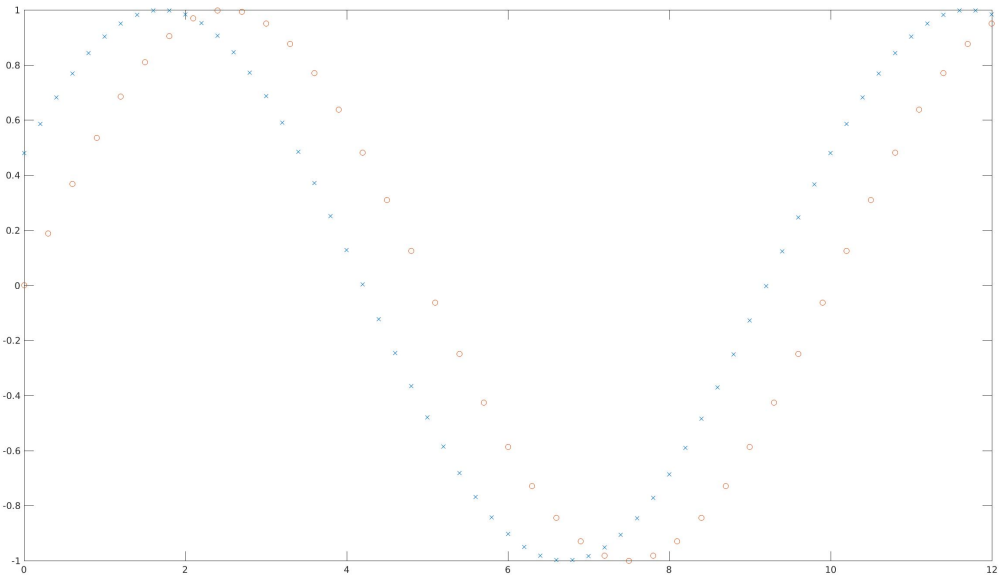


Figure 3: Simulated Signals : Phase Shift

References

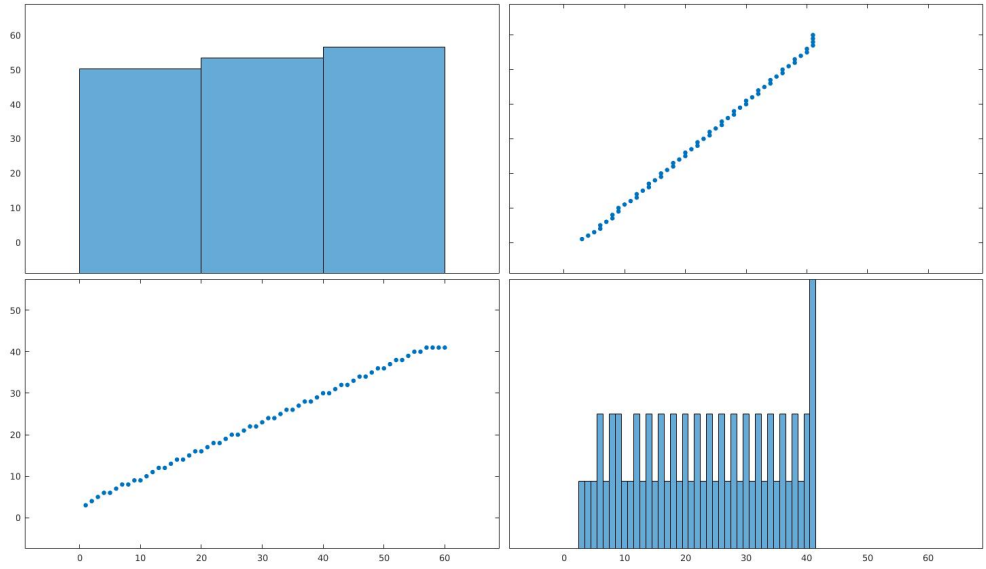


Figure 4: Dynamic Time Warping Path : Phase Shift

CSE 291-D: Final Report

Po-Ya Hsu [A53202971] **Yang Liu [A53206488]** **Hao-en Sung [A53204772]**

1 Abstract

This paper focuses on the predictions of customers' ratings of restaurants on Yelp using matrix factorization with temporal dynamic. We do clustering on the data by first using dynamic time warping algorithm and then perform matrix factorization with simple temporal dynamic, i.e temporal dynamic term relating only to time instead of a term for one specific user or item. We hope clustering data based on pattern changing over time plus simple temporal dynamic term can be as powerful as complex temporal dynamic term, yet more flexible and easy to scale with a larger number of customers.

2 Introduction

Nowadays, customers are flooded by the multiple choices provided by retailers. Matching customers with the most appropriate products is essential to give user better experience, gain user's loyalty and thus make profits so building robust and accurate recommender systems is very important.

Rating predictions is at the core of recommending systems because it reveals and gives a reasonable expectation of the attitude of a customer towards a certain product. It can provide valuable information to help improve the behavior of recommender system.

The matrix factorization with temporal dynamic is powerful for rating predictions yet we need to have a lot of extra parameters to exploit such power especially we have large scale of users and items. In such scenario, the extra parameters for encoding temporal dynamic increase linearly as the number of user and item increase. We want to take advantage of temporal dynamic to encode time-varied information but we want to encode the information in a simple, clean way to make it easy to scale. So we come up with a new method to avoid the tedious parameter choosing and expression design process of temporal dynamic while still achieve the power of temporal dynamic in rating prediction. We first use Dynamic Time Warping (DTW) Algorithm with time domain signals to cluster the dataset and then perform matrix factorization on each cluster to train the model.

In **section 3**, we give some background knowledge about dynamic time warping method and matrix factorization; we do basic data analysis on Yelp data in **section 4**; and we come up with our model in **section 5**; in **section 6**, we give the result of our experiments; we discuss about the results in **section 7** and some improvements and future work are given in **section 8**.

3 Background Knowledge

3.1 Dynamic Time Warping Method

To handle pattern recognition in temporal data, focused back-propagation algorithm is a technique used in speech recognition [4]. Another approach mentioned in [1] is using hidden Markov model approach. In this project, we choose to implement dynamic time warping algorithm to find patterns in our data. DTW method, according to [5], redefines the distance distribution between two signals and finds an optimal time warping path to interpret the mutual patterns. The algorithm first calculates the pairwise cost of each data point, and then searches an optimal time warping path by dynamic programming.

3.2 Matrix Factorization

In this paper, we do rating prediction for restaurants on Yelp. We do not find any related document that has performed the same experiment but we do find one about rating prediction for movies on Netflix. Paper[3] does rating predictions for the Netflix Prize, which relates closely to our task. In this paper, a basic matrix factorization model maps users and

items into a joint latent factor space and use inner dot products to represent user-item interactions, i.e $\boldsymbol{r}_{ui} = \boldsymbol{Q}_i^T \boldsymbol{P}_u$. Then they improve this method by adding biases, i.e $\boldsymbol{r}_{ui} = a + \boldsymbol{b}_i + \boldsymbol{b}_u + \boldsymbol{Q}_i^\intercal \boldsymbol{P}_u$. The model is improved again by adding temporal dynamics, making item biases $\boldsymbol{b}_i$, user biases $\boldsymbol{b}_u$ and user preferences $\boldsymbol{p}_u$ vary over time. Then the model can be represented as $\boldsymbol{u}_{ui} = a + \boldsymbol{b}_i(t) + \boldsymbol{b}_u(t) + \boldsymbol{Q}_i^\intercal \boldsymbol{P}_u(t)$. From the experiment on Netflix data, temporal does bring benefits to rating predictions.

Paper [2] gives some methods to encode the varying terms. Item biases changing over time is addressed by split the item biases into time-based bins and then item biases can be represented as a stationary part and a time changing part of a given bin, i.e $\boldsymbol{b}_i(t) = \boldsymbol{b}_i + \boldsymbol{b}_{i,Bin(t)}$. For user biases, term associated time deviation is introduced,i.e $\textbf{dev}_u(t) = \text{sign}(t - \boldsymbol{t}_u) \cdot |t - \boldsymbol{t}_u|^\beta$ where β is chosen by cross-validation. Then a time-dependent user-bias is expressed as $\boldsymbol{b}_u(t) = \boldsymbol{b}_u + \boldsymbol{\alpha}_u \cdot \textbf{dev}_u(t) + \boldsymbol{b}_{ut}$ where α is new parameter for each user and $\boldsymbol{b}_{ut}$ is a single parameter per user and day to represent the day-specific variability. The same methodology can be used to capture more effects.

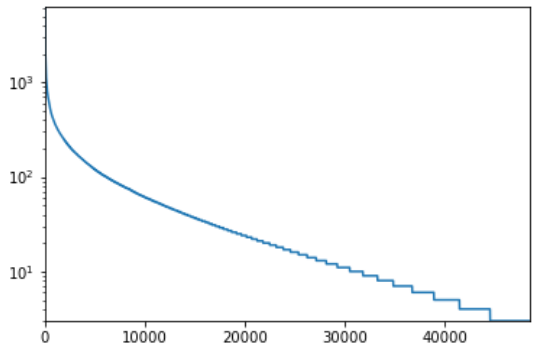
4 Data Analysis

4.1 Data Source

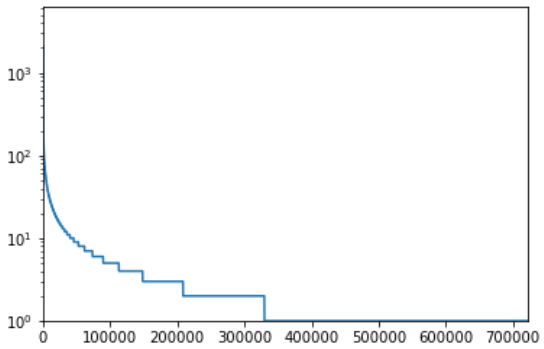
The data we use for our project is the data provided by Yelp for Round 9 Of The Yelp Dataset Challenge[1]. We focus on only restaurant related data in yelp_academic_dataset_business.json, yelp_academic_dataset_review.json and yelp_academic_dataset_user.json. More information about these data can be found on the webpage.

4.2 Analysis on Data

First, we filter out the non-restaurant data and plot the number of total reviews received by each restaurant and each user separately. The result is as following. From Fig. 1a and Fig. 1b, we know that the distribution of number of reviews for both businesses and users are log-linear. It means that few restaurants and users are associated many reviews, while the rest of them only are only related few reviews.



(a) Review Distribution for Restaurants



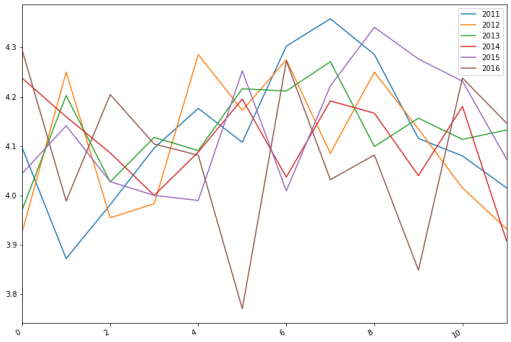
(b) Review Distribution for Users

We also filter out restaurants and users with few reviews.

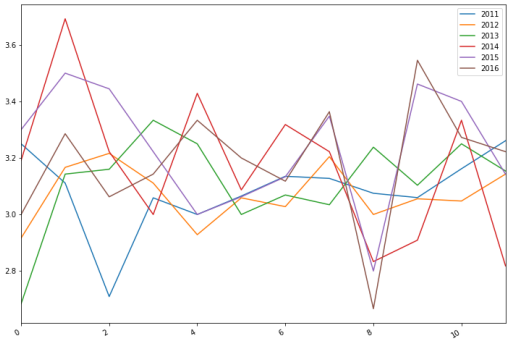
1. Filter out non-restaurant business. After reducing the number of businesses from 144072 to 48485, the overall number of reviews decreases from 4153150 to 2577298.
2. Filter out restaurants with less than 10 reviews, which reduces the number of restaurants decreases from 48485 to 31792.
3. Filter out users providing leas than 5 reviews, which reduces the number of users from 707751 to 109160.
4. Remove those unqualified restaurants and users, which reduces the number of reviews from 2577298 to 1532164.
5. Separate training and testing sets with time threshold — year 2015. At the end, there are 1206403 reviews in training set and 325761 reviews in testing set.

After all these steps, we try to find some pattern about the data by heuristic. We plot average month ratings of a representative restaurant and user. From the plots, we can see there is a pattern for the rating across months

[1] `https://www.yelp.com/dataset_challenge`



(a) Month ratings for Restaurants



(b) Month ratings for Users

5 Method and Model

5.1 Method

We use DTW method to cluster the restaurants based on their rating changing with time pattern and build a matrix factorization with simple temporal dynamic for each cluster. By saying simple temporal dynamic, we mean we do not assign each user or each restaurant a temporal parameter, we use a dynamic term only relating to time. Since the clustering already includes some information related with time, we hope our method can achieve a result at least as good as the complex way of time encoding mentioned in [2].

We use matrix factorization with biased term only model and matrix factorization with additional latent variables as baselines.

5.2 Clustering

To do clustering, we have made attempts on DTW. Plus, we endeavored to evaluate the clustering by other means such as latent Dirichlet Allocation (LDA) and data prediction by running DTW.

5.2.1 Dynamic Time Warping Approach

Our goal is to categorize the items with similar trend into the same group, and thus, we design an algorithm to capture this property. We come up with a modified DTW algorithm, DTW-MAG-COST, to exclude the magnitude's impact. On top of that, we run DTW algorithm as a subroutine to calculate the difference between the two items. If the difference

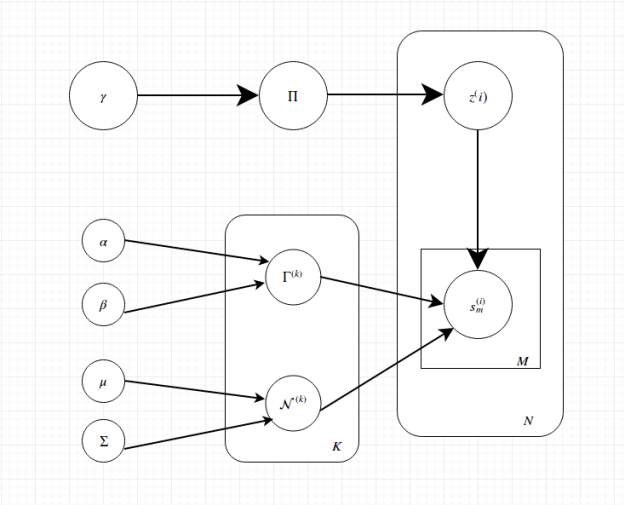
Algorithm 1 Find the difference between 2 items

procedure DTW-ITEM($S1$,$S2$)
Input: Two Signals $S1$, $S2$
Output: Difference between $S1$ and $S2$
$dtw0 \leftarrow$ DTW-COST$(S1, S2)$
$P0 \leftarrow$ OPT-DTW-PATH$(dtw0)$
$dtw1 \leftarrow$ DTW-MAG-COST$(S1, S2)$
$P1 \leftarrow$ OPT-DTW-PATH$(dtw1)$
$d \leftarrow$ DTW-PATH-DIFF$(P0, P1)$

found between two items is 0, we put them into the same group.

5.2.2 Clustering Evaluation

We modify the model in Homework 2 from sampling each word count per document to sampling each month's rating per item. Suppose the number of clusters found by DTW is K, then we assume the latent variable to have K hidden state. The graphical model is shown in the figure below.



(a) Graphical Model for DTW Clustering Evaluation

We use Gibbs sampling to find the distribution of the hidden variables. Given priors are $\Pi(\gamma)$,$\Gamma(\alpha,\beta)$, $\mathcal{N}(r,\eta)$ The update equations are:

$$\Pi_k^{(t+1)} = \text{SampleDirichlet}\left(\beta^{t+1}\right), \gamma^{t+1} = \gamma + \sum_{i=1}^{N} \mathbf{1}\left(z^{i(t)} = k\right)$$

$$z^{(d)(t+1)} = \text{SampleDiscrete}\left(\frac{q_k}{\sum\limits_k q_k}\right), q_k = \pi_k \prod_{m=1}^{12} \mathcal{N}_m^{(k)} \Gamma_m^{(k)}$$

$$\Gamma^{(k)(t+1)} = \text{SampleGamma}\left(\Gamma\left(\alpha + \frac{N^{(k)(t)}}{2}, \beta + \sum_{i=1}^{N^{(i)(t)}} \frac{s^{(i)} - \mu}{2}\right)\right)$$

$$\mathcal{N}^{(k)(t+1)} = \text{SampleGaussian}\left(\frac{\tau N \vec{x} + \eta r}{\tau N + \eta}, \tau N + \eta\right), N = N_m^k$$

5.3 Baseline Algorithm and Our Algorithm

5.3.1 Baseline Algorithm: Matrix Factorization with Biased Terms only

Firstly, we would like to work on the simplest model that only takes user, item, and rating into consideration.

According to [2], a very handy latent variable model is Matrix Factorization with only biased terms, i.e.

$$r_{u,i} \approx a + \boldsymbol{b}_u + \boldsymbol{b}_i,$$

where

u is the user identity
i is the business identity
$r_{u,i}$ is the rating from u to i
a is the general biased term
$\boldsymbol{b}_u$ is the biased term for u
$\boldsymbol{b}_i$ is the biased term for i.

5.3.2 Baseline Algorithm: Matrix Factorization with Additional Latent Vectors

Besides biased terms for users and items, we can also introduce two latent variable matrices: one of them is for user latent features, and the other is for item latent features, i.e.

$$r_{u,i} \approx a + \boldsymbol{b}_u + \boldsymbol{b}_i + \lambda \times \boldsymbol{P}_u \boldsymbol{Q}_i^{\intercal},$$

where

$$\begin{aligned} \lambda &\text{ is the meta parameter for latent vectors} \\ \boldsymbol{P}_u \in \mathbb{R}^K &\text{ is the latent vector for } u \\ \boldsymbol{Q}_i \in \mathbb{R}^K &\text{ is the latent vector for } i \\ K &\text{ is the meta parameter for latent vector length} \end{aligned}$$

This model is especially powerful in capturing latent, or said hidden, information from the whole dataset, and is widely used in different applications.

5.3.3 Our Algorithm:Matrix Factorization with User, Item, and Time Information

In view of the strong correlation between data and time factor, we also want to take time information into concern. Based on [3], an effective way is to regard $\boldsymbol{b}_u$ and $\boldsymbol{b}_i$ as functions of t, i.e. $\boldsymbol{b}_u(t)$ and $\boldsymbol{b}_i(t)$, instead of using purely constants for each user and item. In the paper, they further assume that the latent user preference will vary through time. Thus, their final model is like:

$$r_{u,i}(t) \approx a + \boldsymbol{b}_u(t) + \boldsymbol{b}_i(t) + \lambda \times \boldsymbol{P}_u(t) \cdot \boldsymbol{Q}_i^\intercal.$$

However, since we do not want to over-complicate the model and DTW can help encode some information, we decide to introduce time information as a trainable biased term only relates to time, i.e. either

$$r_{u,i}(t) \approx a + \boldsymbol{b}_u + \boldsymbol{b}_i + \boldsymbol{b}_t.$$

or

$$r_{u,i}(t) \approx a + \boldsymbol{b}_u + \boldsymbol{b}_i + \boldsymbol{b}_t + \lambda \times \boldsymbol{P}_u \boldsymbol{Q}_i^\intercal.$$

It is noticeable that the t here is a general time domain feature. It can represent for year, month, date, or day.

6 Experiments and Results

In our experiment, hashed user identity and hashed business identity are provided as features; while our target score is the rating from one user identity to one business identity, which is a real number ranging between 0 and 5.

In view of the dataset property, we decide to use latent variable based model to solve Before building up a complex model with temporal information, we would like to first run some simpler model as baseline algorithms, including matrix factorization with and without latent vectors.

6.1 Dynamic Time Warping Algorithm Results

6.1.1 EXP 1. DTW Clustering Results

We found 11 clusters 1 after running DTW.

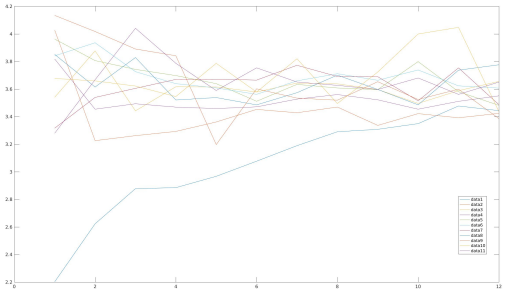
Table 1: Dynamic Time Warping Clustering Results

Cluster	1	2	3	4	5	6	7	8	9	10	11
item number	2898	1222	1044	1928	745	170	367	150	87	25	48

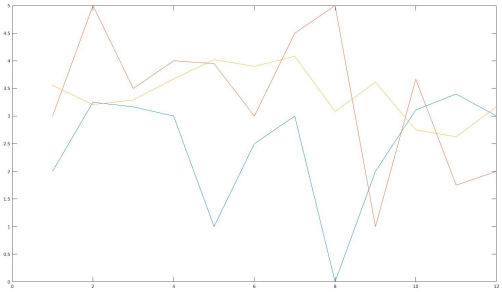
The average curves from each cluster is shown in 1a.

6.1.2 EXP 2. Clustering Evaluation : DTW and LDA

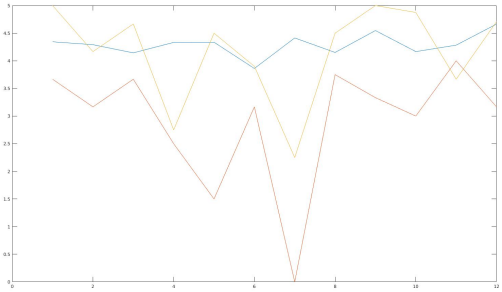
We treat 2010-2015 data as training set and 2016 data as testing set for DTW and LDA. For LDA, we have tried setting different priors: from DTW results or our belief. Besides, we have tried on different features such as the difference of the ratings. However, the results are still eventually one cluster.



(a) Graphical Model for DTW Clustering Evaluation



(a) Representative Patterns for Cluster 1



(b) Representative Patterns for Cluster 8

Table 2: Clustering Evaluation Results

Evaluation Method	DTW	LDA
training results (2010-2015)	11 clusters	1 cluster
testing results (2016)	mainly 2 clusters (Cluster 8 and 9)	No need to do

6.2 Baseline Algorithm: Matrix Factorization with Biased Terms only

It is the simplest matrix factorization model that regards each user and item with consistent score tendency, i.e. biased term $\boldsymbol{b}_i$ and $\boldsymbol{b}_i$. We put our experiments on two different data settings: one without time information and one with time information.

To fulfill the data convergence, I implement two different learning algorithms in *C++*: Alternate Least Square (ALS) and Stochastic Gradient Descent (SGD). For ALS, I set the *MAXIMUM_ITER* as 100, because of its ability to efficiently converge to a poorer local minimum. For SGD, I specify the learning rate to be 1 (after normalized with the number of training instances), learning rate decay to be 0.9999 and momentum to be 0.9. Both of them use regularization term $\lambda = 5$ to prevent model from overfitting. However, due to the similar performance and space limitation of this report, we decide to omit the detailed results here but provide codes on GitHub.

In implementation, I only implement one algorithm but set a scaler *MONTH_SCALE* and switch between 0 and 1 to disable or enable the model capability of dealing with time information.

6.2.1 Model with User, Item, and Rating Information

In this experiment, *MONTH_SCALE* is set to 0. The corresponding results are listed as Table 3.

6.2.2 Model with User, Item, Time, and Rating Information

In this experiment, *MONTH_SCALE* is set to 1. The corresponding results are listed as Table 4.

Table 3: Matrix Factorization with Biased Term Only (User + Item)

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.354851	1.490561	1196592.520015	336896.683485
20	0.962545	1.290251	902088.126999	343599.334280
40	0.962532	1.290547	902023.440762	343613.282293
60	0.962530	1.290829	902003.961840	343659.635690
80	0.962529	1.291062	901994.086963	343703.357421
100	0.962528	1.291247	901988.421500	343740.169698

(a) Target on all eleven clusters

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.343741	1.478029	345642.494682	95563.437035
20	0.907083	1.311304	258489.482181	109949.569648
40	0.907076	1.311617	258481.983893	109964.229485
60	0.907075	1.311877	258478.192905	109977.536769
80	0.907074	1.312084	258475.908304	109988.829458
100	0.907073	1.312248	258474.523220	109998.178617

(b) Target on one of eleven clusters — cluster 0

Table 4: Matrix Factorization with Biased Term Only (User + Item + Month)

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.354851	1.490561	1196592.520015	336896.683485
20	0.962511	1.290219	902056.982273	343591.155361
40	0.962498	1.290514	901992.298558	343604.966090
60	0.962496	1.290797	901972.853402	343651.191632
80	0.962494	1.291028	901963.005858	343694.799980
100	0.962494	1.291213	901957.361928	343731.522458

(a) Target on all eleven clusters

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.343741	1.478029	345642.494682	95563.437035
20	0.906999	1.311232	258464.121395	109941.167054
40	0.906992	1.311543	258456.660855	109955.780844
60	0.906990	1.311802	258452.901746	109969.024600
80	0.906990	1.312007	258450.641811	109980.265720
100	0.906989	1.312171	258449.275801	109989.574221

(b) Target on one of eleven clusters — cluster 0

6.2.3 Summary

From Table 3 and 4, we can easily tell that the improvement of considering time information — month for both all eleven clusters and cluster 0 are not significant, though experiment on cluster 0 improves slightly more than experiment on all clusters.

In view of this, we want to do more rigorous results and will discuss in more details in later sections.

6.3 Baseline Algorithm: Matrix Factorization with Additional Latent Vectors

On top of the simplest model, one can also introduce $\boldsymbol{P}_u$ and $\boldsymbol{Q}_i$ for user and item latent vectors. Similar to previous section, we also put experiments on two different data settings: one without time information and one with time information.

To fulfill the data convergence, I implement two different learning algorithms in *C++*: Alternate Least Square (ALS) and Stochastic Gradient Descent (SGD). For ALS, I set the *MAXIMUM_ITER* as 100, because of its ability to efficiently converge to a poorer local minimum. For SGD, I specify the learning rate to be 1 (after normalized with the number of training instances), learning rate decay to be 0.9999 and momentum to be 0.9. Both of them use regularization term $\lambda = 10$ to prevent model from overfitting. The results for SGD is omitted here because of space limitation.

In implementation, I only implement one algorithm but set a scaler *MONTH_SCALE* and switch between 0 and 1 to disable or enable the model capability of dealing with time information.

6.3.1 Model with User, Item, and Rating Information

In this experiment, *MONTH_SCALE* is set to 0. The corresponding results are listed as Table 3.

Table 5: Matrix Factorization with Additional Latent Vectors (User + Item)

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.360202	1.495980	1518716.149541	655518.949408
20	0.950662	1.295037	924805.022578	377893.286941
40	0.937487	1.296810	920279.158249	385404.216356
60	0.934217	1.298023	919268.085931	387555.283252
80	0.932584	1.298614	918699.179108	388562.210034
100	0.931337	1.299051	918128.837900	389191.500756

(a) Target on all eleven clusters

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.349893	1.483521	617228.254168	365921.868825
20	0.877999	1.316569	263659.644270	122941.444936
40	0.864460	1.318197	262486.234981	125355.696591
60	0.858914	1.319384	262040.635133	126413.504131
80	0.855698	1.320032	261787.731370	127029.639177
100	0.853458	1.320532	261585.087671	127435.447882

(b) Target on one of eleven clusters — cluster 0

6.3.2 Model with User, Item, Time, and Rating Information

In this experiment, *MONTH_SCALE* is set to 1. The corresponding results are listed as Table 4.

Table 6: Matrix Factorization with Additional Latent Vectors (User + Item + Month)

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.360202	1.495980	1518716.149541	655518.949408
20	0.950895	1.295183	925066.817898	377981.765980
40	0.937704	1.297128	920463.964101	385469.368121
60	0.934473	1.298135	919468.865431	387554.819742
80	0.932873	1.298716	918920.472895	388551.099182
100	0.931664	1.299252	918371.819923	389191.635135

(a) Target on all eleven clusters

Iteration	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
0	1.349893	1.483521	617228.254168	365921.868825
20	0.877981	1.316571	263674.729793	122961.199212
40	0.864520	1.318387	262496.974283	125363.359022
60	0.859003	1.319390	262054.307004	126404.530325
80	0.855794	1.320032	261803.876299	127021.134728
100	0.853562	1.320617	261603.915407	127433.005515

(b) Target on one of eleven clusters — cluster 0

6.3.3 Summary

From Table 5 and 6, we find out that time information does not help this time. In addition to the previous experiment results, it makes us more curious about the clustering result. Thus, more rigorous experiment are provided in later section.

Besides that, when we compare the results between model with biased term and model with additional latent vector, one can tell that the results from model with additional information performs worse. We find out this issue from the start of the quarter and have tried many different ways to verify it, including running our algorithm on some toy-example, verifying the results in terms of training error and training loss. Unfortunately, we still cannot find out any bug in our implementation. Here, we would like to provide an alternative explanation for it: we can see that the training error is much lower in latent variable model than in biased-term model. In other words, our model indeed learns better representative way in terms of training data; nevertheless, it is very difficult and tedious in grid-search the proper parameter for model.

7 Discussion

Our model does not work very well, which motivates us to think about the reliability of our clustering. DTW algorithm gives 11 clusters of our data while Gibbs sampling only results in one cluster. We then perform our models on the whole data set and on each cluster separately. The result is as following. As we can see from Table 7, clustering helps

Table 7: Clustering vs Non-clustering

Dataset	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
All	0.931337	1.299051	918128.837900	389191.500756
Cluster 0	0.906989	1.312171	258449.275801	109989.574221
Cluster 1	0.878571	1.320322	103693.580562	49112.810403
Cluster 2	0.849380	1.326759	87129.571813	41000.215517
Cluster 3	0.902417	1.344746	173022.338026	77410.022732
Cluster 4	0.875140	1.371026	64225.518483	30300.297385
Cluster 5	0.835407	1.398496	12504.257415	7352.344659
Cluster 6	0.843513	1.428097	28029.138189	15266.850285
Cluster 7	0.836292	1.411318	12224.821136	6363.273979
Cluster 8	0.828668	1.395997	7270.716190	3828.859834
Cluster 9	0.858364	1.357204	2189.973273	1137.282753
Cluster 10	0.679948	1.278873	3728.566350	2361.248985

(a) Matrix Factorization with Biased Term only (User + Item + Month)

Dataset	TRAIN MSE	TEST MSE	TRAIN COST	TEST COST
All	0.931664	1.299252	918371.819923	389191.635135
Cluster 0	0.853562	1.320617	261603.915407	127433.005515
Cluster 1	0.809677	1.327427	104921.600985	57638.888261
Cluster 2	0.784600	1.331241	88568.194953	48320.613698
Cluster 3	0.848912	1.350294	176637.421633	90384.350555
Cluster 4	0.811024	1.375340	65795.578797	35993.269489
Cluster 5	0.758452	1.398042	12997.843835	8803.696071
Cluster 6	0.787111	1.434129	29255.025828	18116.342246
Cluster 7	0.750922	1.413847	12540.236465	7730.330813
Cluster 8	0.718678	1.394789	7421.095533	4779.580812
Cluster 9	0.682766	1.355447	2226.399794	1544.111935
Cluster 10	0.676778	1.277333	4015.616131	2660.607904

(b) Matrix Factorization with Latent Vectors (User + Item + Month)

with MSE for training data for both cases but it does not help much with MSE on test data.

One very possible reason is the sparsity of Yelp data. Given a certain number of zeros, our algorithm can wrongly fill in the trend of the missing data and eventually put the item into a wrong group.

8 Conclusion and Future Works

Our temporal dynamic handling tool does not work well to make predictions. As for clustering, we believe our tactics of capturing temporal dynamic features is correct based on the simulations. Nevertheless, we do not take the intrinsic properties of the data into consideration. For example, we naively take 2010-2015 data as training set instead of selecting the representative temporal data for each item. Given the sparsity of the data, every non-representative trend acts as the noise to our training set. To make improvements on our clustering method, we are eager to add a procedure to select representative trends and treat those as our targeted training set. As for the matrix factorization part, we have not implemented the complex version of temporal dynamic to compare it with our simple temporal dynamic, and we want to do it in the future to see how powerful our model is. Also, we want to try such comparison on several other datasets, for example Amazon review data[2] to further evaluate the power of our model.

9 Appendix: Dynamic Time Warping Algorithm

Algorithm 2 Find the optimal cost for each dynamic time warping path

procedure DTW-COST($S1$,$S2$)
Input: Two Signals $S1$, $S2$
Output: Optimal Cost Matrix for each pairwise distance D
$n \leftarrow |S1|,\ m \leftarrow |S2|$
Create a matrix C of size $n \times m$ with all 0s
for $i = 1 : n$ **do**
  for $k = 1 : m$ **do**
    $C(i,k) = |S1(i) - S2(k)|$ ▷ modify targeted cost here, $---L5$
Create a matrix D of size $n \times m$ with all 0s
for $i = 2 : n$ **do**
  $D(i,1) = D(i-1,1) + C(i,1)$
for $k = 2 : m$ **do**
  $D(k,1) = D(k-1,1) + C(1,k)$
for $i = 2 : n$ **do**
  for $k = 2 : m$ **do**
    $D(i,k) = C(i,k) + min(D(i-1,k), D(i-1,k-1), D(i,k-1))$
Return D

Algorithm 3 Find the optimal dynamic time warping path

procedure OPT-DTW-PATH(D)
Input: Cost matrix D of two signals of interest
Output: Optimal Dyanmic Time Warping Path P based on D
$[i,j] \leftarrow Size(D)$
Create an empty matrix P
while $i > 1$ **OR** $j > 1$ **do**
  if $i = 1$ **then**
    $j \leftarrow j - 1$
    elseif $j = 1$
    $i \leftarrow i - 1$
  else
    $choices \leftarrow [D(i-1,j), D(i,j-1), D(i-1,j-1)]$
    if $D(i-1,j) = min(choices)$ **then**
      $i \leftarrow i - 1$
      elseif $D(i,j-1) = min(choices)$
      $j \leftarrow j - 1$
    else
      $i \leftarrow i - 1$
      $j \leftarrow j - 1$
    $P \leftarrow [P; (i,j)]$
Return P

[2] http://jmcauley.ucsd.edu/data/amazon/

Algorithm 4 Find the magnitude-defined optimal cost for each dynamic time warping path

procedure DTW-MAG-COST($S1$,$S2$)
Input: Two Signals $S1$, $S2$
Output: Optimal Cost Matrix for each pairwise distance D
Exactly the same Procedure as **Algorithm 1** but define the cost as $C(i,k) = |sign(S1(i)) - sign(S2(k))|$

Algorithm 5 Quantify the path difference between two dynamic time warping paths

procedure DTW-PATH-DIFF($P1$,$P2$)
Input: Two Dynamic Time Warping Paths $P1$, $P2$ ▷ Suppose they are 2-dimension
Output: Quantified difference d between $P1$ and $P2$
$d1 \leftarrow DTW - Cost(P1.dim1, P2.dim1)$
$d2 \leftarrow DTW - Cost(P1.dim2, P2.dim2)$
$d \leftarrow d1.end + d2.end$
Return : d

References

[1] Kenneth H Fielding and Dennis W Ruck. Spatio-temporal pattern recognition using hidden markov models. *IEEE Transactions on Aerospace and Electronic Systems*, 31(4):1292–1300, 1995.

[2] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[4] Michael C Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381, 1989.

[5] Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855:1–23, 2008.