Collaborative Deep Learning for Recommender Systems

Sampath Chanda
Carnegie Mellon University
schanda@andrew.cmu.edu

Suyin Wang
Carnegie Mellon University
suyinw@andrew.cmu.edu

Xiaoou Zhang
Carnegie Mellon University
xiaouz@andrew.cmu.edu

Introduction

The enormous number of services provided by the e-commerce websites, such as Amazon, Netflix and Yahoo!, feed the customers with abundant choices, but also lead to the problem that the right information may not go to the right customer. An efficient recommender system is required to benefit both the customers by delivering the services in need, and the companies by providing advertisement strategies. While a lot of attention has been paid to recommender systems based on the explicit ratings in both academia and industry, we find that the explicit ratings from users to products are not always available in real business world, where consumers reveal their preferences implicitly through purchasing, browsing and searching behaviors. Moreover, with an explosive increase of the side information about users and products in the age of big data, how to efficiently incorporate such information into recommender systems is an active field of study in recent years.

In this project, we combine the stacked denoising autoencoder (SDAE) with matrix factorization (MF) algorithm. We utilize this model to predict the customer purchase behavior in the following month according to the purchase history and side information in the Santander dataset [1]. We use the hidden layer with the smallest dimension in the SDAE, i.e., the "bottleneck", as the representation of the user information, and send it to the MF algorithm.

Related Work

Collaborative filtering (CF) is a powerful method for recommender systems. CF starts from the rating history or, more implicitly, the past preferences of users. It has reached high accuracy using MF (Koren *et al.* (2009)) and restricted Boltzmann machines (Salakhutdinov *et al.* (2007)). However, the cold start problem arises when the rating information is sparse, in which case CF cannot lead to an accurate prediction. One way to get rid of the cold start problem is combining the content based algorithm with CF. These are called hybrid methods. In some earlier papers (such as Ma *et al.* (2011)), the side information is just regarded as a regularization. This method is less helpful when the side information is sparse. A more efficient way to exploit the side information is using deep learning algorithms, which are powerful in terms of extracting high level representations. A recent paper (Wang *et al.* (2015)) encodes the user information with SDAE and feed it to the MF algorithm; while another paper (Strub *et al.* (2016)) feeds the encoding to a collaborative filtering neural network.

Methods

Dataset and pre-processing

The goal of the data pre-processing is extracting the rating matrix and the side information of users.

[1] https://www.kaggle.com/c/santander-product-recommendation/data

10707 Class Project (2017).

The rating matrix is extracted from the product subscription information of around 1M customers in Santander's dataset. For each user and each month, 24 binary indicators are provided which correspond to 24 different products and indicate whether or not the user has subscribed to the product in that month. This information is provided for 17 months with the time line from 2015-01-28 to 2016-05-28, with an interval of one month. The total number of such entires is 13.5M, which will go through the data cleaning process.

During the data cleaning process, we try to drop as little data as possible. Instead of dropping all the rows with NaN values, we create an UNKNOWN category for most of the columns. For some special product indicators like payroll and pension, it is more reasonable to replace the NaN values with 0. One the other hand, we are under the opinion that the column "renta", that indicates the gross income of an user, could be a key factor in the purchasing power of any user and hence decided to drop all the entries in the dataset having NaNs in "renta".

After cleaning up the data, we extract a 3-column rating table — ncodpers (customer id), prodIdx (product id), rating (number of months the customer has subscribed to a given product). We divide the rating table into three parts: the first 15 months as the training set, the 16th month as the validation set and the last month as the test set. We end up with 1167015 users in the training set. This rating table is converted into a rating matrix r_{ij}, where each row corresponds to a user, and each column corresponds to a product.

Another step in the data processing is to produce the inputs of the denoising autoencoder. Firstly, some user features that doesn't seem to affect the user details much are dropped. Then, to get the user information, entries corresponding to each user at the latest date is kept. Each field of user information is converted to categorical type to make sure that they can be converted to one-hot encoding. Some variables like annual gross income and age have too many possible values to be directly converted into categorical. The variables are binned manually such that the number of entries distributes evenly in each bin. Having the type of each column to be categorical, all the values in these columns are converted into one hot encodings separately.

Fig. 1 shows the number of users subscribed to each product in the month of 2015-01-28. The y-axis shows different products offered by the bank like loans, pensions, savings account, credit card, etc. (represented by their corresponding product codes). The product "ind_cco_fin_ult1" has a huge user base which refers to current account. It is followed by particular account (ind_ctop_fin_ult1) and direct debit account (ind_recibo_fin_ult1) which also have a good user base.

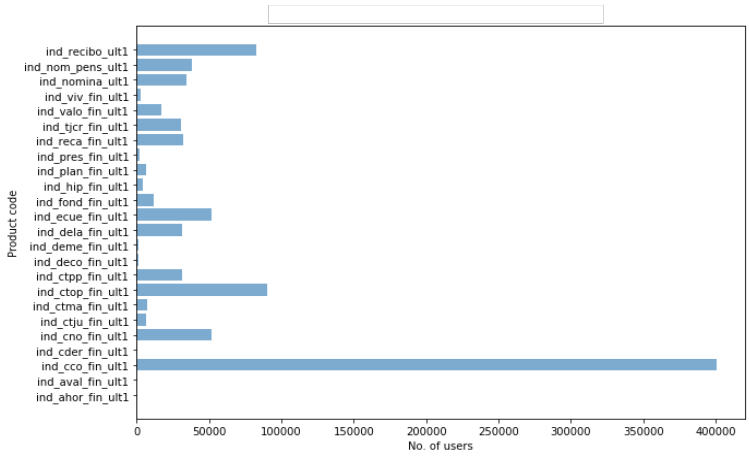


Figure 1: Product-wise user base for month 2015-01-28.

Stacked Denoising Autoencoder

We employ the SDAE to incorporate the users' side information into the recommender system (Wang *et al.* (2015)). The user information is converted into 18 categorical variables, each of which is represented by a one-hot vector. The final representation of each user's side information is the concatenation of the one-hot vectors of the 18 variables, and hence makes a 314-dimension vector.

The one-hot encoding in the input layer is followed by a corruption layer, where a Gaussian noise is added to the input. The structure of a typical SDAE is shown in the upper part of fig. 2. X_0 is the original user information using one-hot encoding, and X_c is the corrupted input with Gaussian noise. Note that we use tied weight in the SDAE, such that the SDAE has a symmetric structure. The X_{encode} layer with the least number of hidden units is the encoding of the user information, which is also called "bottleneck" in this paper.

We use mini-batch gradient descent algorithm to train our SDAE, with the batch size = 500. We have compared the training performance of this mini-batch algorithm with stochastic gradient descent algorithm, and the mini-batch approach converges faster and has lower loss. Hence we choose to stick to the mini-batch gradient descent for the rest of the paper.

We expect to collect two outputs from SDAE: 1) the prediction, i.e. the reconstruction of the input data; 2) the "bottleneck", which is the encoding of the user information. The bottleneck would enter the MF as an additional source of information to optimize our recommendation.

Matrix Factorization for Implicit Feedbacks

The rating matrix r_{ij} gives the user behavior history, which are implicit feedbacks. We cannot simply apply the MF algorithm for the explicit rating, since there are two fundamental differences between the implicit and explicit feedback:

1. A small value in the rating matrix does not always imply that the user dislike the service. The zero entries may due to the user's unawareness to the service.
2. A large value in the rating matrix does not always imply that the user like the service. For example, a customer may have used the loan service for years. But once the customer paid it off, he/she may not use it any more.

For the recommender systems with implicit feed backs, we can construct preference p_{ij} and confidence c_{ij} according to the rating matrix (Hu *et al.* (2008)). If a customer has used a certain service before, it is possible that the user has preference to this service. Accordingly, the definition of preference p_{ij} is given by

$$p_{ij} = \begin{cases} 1 & \text{if } r_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{1}$$

On the other hand, we are not completely sure about the preference. Therefore, we need to define the confidence

$$c_{ij} = 1 + \alpha r_{ij} , \tag{2}$$

where α describes how the confidence grows with the history of using the service j.

The MF algorithm for implicit feed back is applied in the following way. We define user matrix $U_{i,:}$ and item matrix $V_{j,:}$, where each row, written as $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$, is the vector in the latent factor representation for each customer and service, respectively. We predict the preference $p_{i,j}$ by $\boldsymbol{u}_i \cdot \boldsymbol{v}_j$. The preferences with different levels of confidence are not treated equally in the loss function, which is given by

$$l = \sum_{i,j} c_{i,j}(p_{i,j} - \boldsymbol{u}_i \cdot \boldsymbol{v}_j)^2 + \lambda(\sum_i ||\boldsymbol{u}_i||^2 + \sum_j ||\boldsymbol{v}_j||^2) . \tag{3}$$

From a probability point of view, the confidence $c_{i,j}$ is the standard deviation of the prediction to the preference.

Note that the matrix being predicted is p_{ij}, which has no missing values. In this case, gradient decent may be a slow way to learn the parameters. Instead, we can set the derivatives of $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ to zero and alternatively update this two quantities in the following way:

$$U_{i,:} \to P^i C^i V(\lambda I + V^T C^i V)^{-1} , \tag{4}$$

where $C^i_{jj} = c_{ij}$ is a diagonal matrix, and $P^i = \{p_{ij}\}_{\text{all j}}$ is the preference vector for customer i;

$$V_{i,:} \to \tilde{P}^j \tilde{C}^j U(\lambda I + U^T C^j U)^{-1} , \tag{5}$$

where $\tilde{C}^j_{ii} = c_{ij}$ is a diagonal matrix, and $\tilde{P}^j = \{p_{ij}\}_{\text{all i}}$ is the preference vector for service j.

In the prediction process, for each user i, we mask out the services that have been chosen before. Then we assign a percentile-ranking for the remaining services for the user i, according to the value of $\boldsymbol{u}_i \cdot \boldsymbol{v}_j$. A ranking of 100% means the item is predicted to be the least favorable for user i, while 0% means the the item is the most favorable. A random guess should have a ranking of 50%. We evaluate our model by calculating the average percentile-ranking of the user-item pairs that do not appear in the purchase history.

MF-SDAE Hybrid Model

The matrix factorization algorithm is not good at predicting the preferences for the users whose purchasing history is not on file. In the MF-SDAE hybrid model, the predictions for these new users rely on the user information. Fig. 2 illustrates a hybrid model where three hidden layers are used. The shortest hidden layer X_{encode} for each user i is sent to the MF algorithm as the user matrix $\boldsymbol{U}_i$. Note that the SDAE not only reconstructs the input X_0 at the out put layer X_{out}, but also looks for the best encoding X_{encode} such that it minimizes the loss function for the MF in eq. (3).

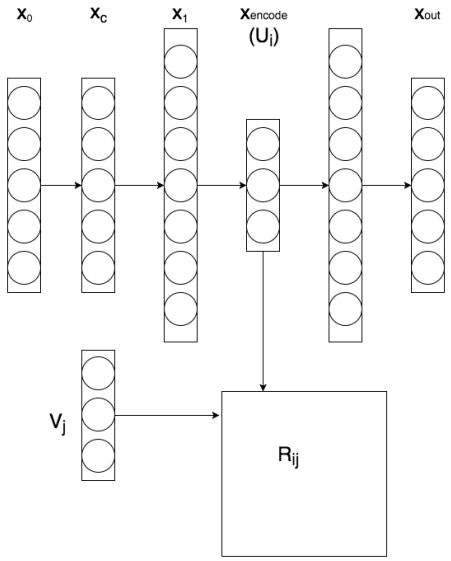


Figure 2: The structure of the MF-SDAE hybrid model

The complete loss function for the hybrid model is given by

$$\begin{aligned} l = &\sum_{i,j} c_{i,j}(p_{i,j} - \boldsymbol{u}_i \cdot \boldsymbol{v}_j)^2 + \lambda(\sum_i ||\boldsymbol{u}_i||^2 + \sum_j ||\boldsymbol{v}_j||^2) \\ &+ \lambda_u(||\boldsymbol{U} - X_{\text{encode}}||^2) + \lambda_n(||X_0 - X_{\text{out}}||^2) + \sum_{\text{layers}} \lambda_W ||W||^2 + \sum_{\text{layers}} \lambda_b ||b||^2 \ . \end{aligned} \tag{6}$$

The first line is just the loss function for the MF part. In the second line, the first term minimizes the difference between the encoding from the SDAE and the user matrix; and the second term is the reconstruction error. The regularization terms for the weights and biases are summed over all hidden layers and the output layer.

The loss function is minimized by block coordinate decent. In each training step, we update $\boldsymbol{U}$ and $\boldsymbol{V}$ using eq. (4) and eq. (5). Given the updated $\boldsymbol{U}$ and $\boldsymbol{V}$, the parameters in the SDAE are updated using gradient decent with mini-batch. The gradient for each weight and bias includes the contribution from reconstructing both the input and the encoding.

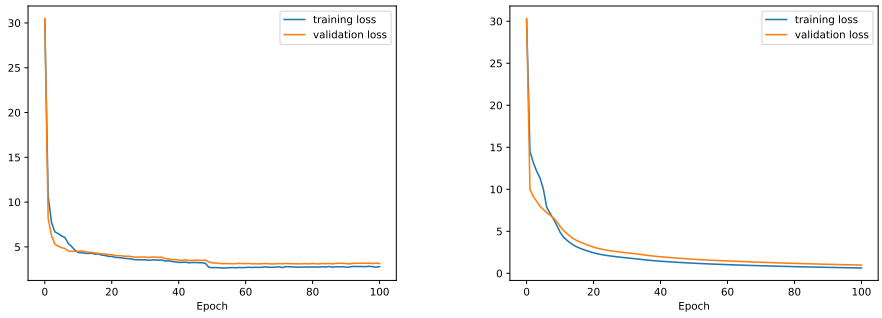
Note that our hybrid model is different from the unsupervised pre-training in neural networks. In the unsupervised pre-training, the restricted Boltzmann machine only provides the initial values of the weights in the neural network, while the parameters in the restricted Boltzmann machine do not receive any feedback from the neural network. However, the two components in our model, i.e., the SDAE and MF part, are updated alternatively. Each part receives the feedback from the other. This

two-way feedback can be interpreted as a mutual regularization. The MF puts a restriction on the SDAE such that the encoding of the user information must help to reconstruct the rating matrix. On the other hand, the SDAE requires that the latent vectors of the users must be related to the user information. This mutual regularization has two advantages. First, the user matrix is not arbitrary, but can be constructed according to the user information even when no purchase history presents. Second, the user information restrains the possible values of the user matrix such that the user matrix will not be trapped in the local minimums which cannot reflect the user information.

Results

We start from the bare MF algorithm, where the SDAE is not applied. We are not able to handle the entire dataset for the time being, since it will be too time consuming to train and requires a giant RAM. We take a subset of 50000 users from the 1167015 users in the training set. We set $\alpha = 40$ and use a 20-dimensional latent vector for each user and service. A strong regularization $\lambda = 100$ is applied to prevent over-fitting. The training error is estimated by the 2-norm of $p_{i,j} - \boldsymbol{u}_i \cdot \boldsymbol{v}_j$, which is 187.76 using the parameters mentioned above. This error is quite small since the matrix size is 50000×24. The percentile-ranking is 11.70%, which is way below 50%. Note that when the regularization is turned off, the prediction is very poor (percentile-ranking 44.07%). Now the MF algorithm can provide a good prediction even without using SDAE.

We train the bare SDAE model using gradient decent with mini-batches of 500 samples, without connecting to the MF algorithm. In addition to the user information of 50000 users in the subset mentioned in the last paragraph, we also use another subset of 20000 users as the validation set. The cross entropy loss for the SDAE using one and three hidden layers are shown in fig. 3. We use 20 hidden units in the one-hidden-layer model, while the number of hidden units in the three-hidden-layer model is 30-20-30. In both cases, the dimension of the bottle neck is the same as that of the latent vector representation in the MF algorithm. The three-hidden-layer model has a lower reconstruction cross entropy loss (0.63) compared to that of the one-layer model (2.79). In both cases, the validation loss is just slightly higher that the training loss, which implies that our SDAE model has good generalization properties. Therefore, there is no need to include regularization terms for the weights and biases in SDAE.



(a) One-hidden-layer SDAE

(b) Three-hidden-layer SDAE

Figure 3: Cross entropy loss on training (blue line) and validation set against epochs

Given the separate bare SDAE and MF model, we can combine them together and train the hybridized model. The result usually converges after 30 such iterations.

We find that a simple SDAE with only one hidden layer can provide good predictions. The best result appears when $\lambda_u/\lambda_n = 300$. In this case, the percentile-ranking for the users with purchase history is 10.41%. This percentile-ranking is lower than that of the bare MF algorithm, which means the SDAE enhances the performance of our collaborative filtering model. We also construct a subset of users whose purchase history is not fed to our model. We first generate the user matrix of these new users, which is given by the value of the bottle neck of the SDAE. After that, the preference to new products of these users is given by multiplying the user matrix and the product matrix. The

percentile-ranking of the new items purchased by these users is 12.37%. As a comparison, we use a random user matrix to mimic the situation that the information from SDAE is missing. In this case, the percentile-ranking rises to about 20%. Therefore, we can conclude that the SDAE successfully solves the cold start problem in the MF algorithm. The percentile ranking for other λ_u/λ_n values for the users with and without purchase history is shown in fig. 4.

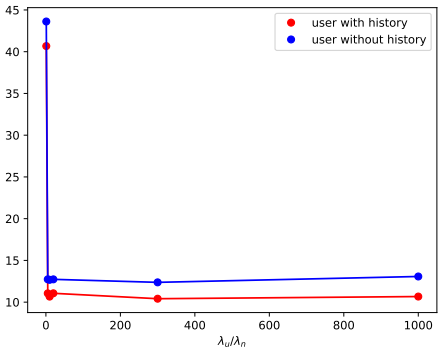


Figure 4: The percentile-ranking with different λ_u/λ_n values for the users with (red) and without (blue) purchase history.

We also applied the three-hidden layer SDAE in the hybrid model. However, the performance of this more complicated model is not as good as that using the model with one hidden layer, although the bare model with three hidden layers has a significantly lower reconstruction loss. The percentile-ranking of the three-hidden-layer model with $\lambda_u/\lambda_n = 10$ is 15.98% for users with purchase history, and 16.83% without purchase history.

Discussion and Analysis

The Relative Importance Between MF and SDAE

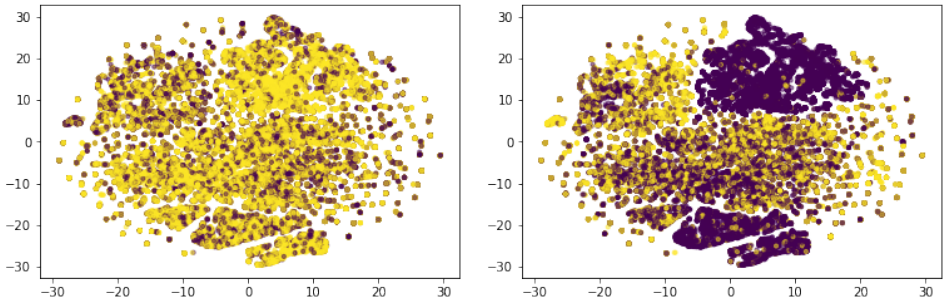
The relative importance between different components is a key factor for most hybrid models. In our case, the relative importance between MF and SDAE is controlled by λ_u/λ_n. When $\lambda_u/\lambda_n = 1$, the percentile ranking is over 40%, which is not evidently better than random guess. In this case, the SDAE is more focused on reducing the reconstruction loss. Further investigation shows that the gradient from the reconstruction loss is usually larger that that from fitting the encoding by 10 to 100 times. Accordingly, the SDAE cannot provide a good fitting for the encoding in the case of $\lambda_u/\lambda_n = 1$. On the other hand, when $\lambda_u/\lambda_n = 1000$, the percentage-ranking for the users without purchase history increases to 13.07% (compared to 12.07% for $\lambda_u/\lambda_n = 300$), although the prediction for the users with purchase history is still low. In this case, the SDAE tends to fit the latent vector of the users and ignore the reconstruction loss. In other words, the regularization effect from the reconstruction loss is too small. Therefore, the prediction for the new users will suffer from the overfitting to the latent vectors of the users in the training set.

Include More Hidden Layers

There is well accepted opinion that deeper models should have better performance. However, the advantage of deeper SDAE is not seen in our work. One reason is that we should apply a stronger regularization since there are more trainable parameters in a deep model. However, we haven't found the proper regularization for the deeper model to surpass the performance of the model using one hidden layer due to the limited amount of time. Another reason is that deeper models are usually harder to train. We believe that the performance of the model using deep SDAE can be enhanced using batch normalization and adjustable learning rates.

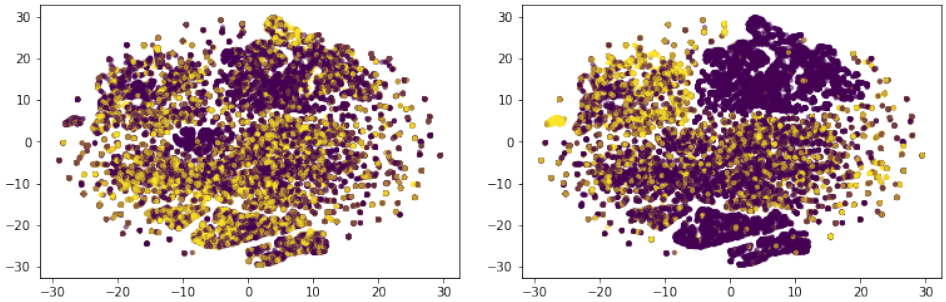
Visualization of the Latent Vectors

We use t-SNE method to project the 20 hidden units on the two-dimensional plane, which is shown in fig. 5. Each point represents one user, and the highlighted yellow dots label the users that have selected a certain product. We show the top 4 most popular products. The most popular product covers most users and does not show obvious feature. One the other hand, the second, third and fourth most popular products are clustered. As a comparison, we show the t-SNE using the encoding from the bare SDAE model in fig. 6. The distribution of the users for different products do not show any feature. They are not clustered and are just randomly scattered among other users. Therefore, in the MF-SDAE hybrid model, the SDAE is not just learning an arbitrary encoding for the user information, but is actually learning the features that determines the purchase behavior of users.



(a) Users subscribed to the most popular product

(b) Users subscribed to the second most popular product



(c) Users subscribed to the third most popular product

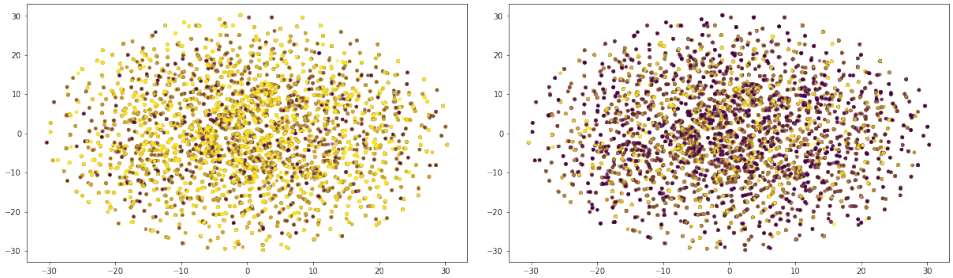
(d) Users subscribed to the fourth most popular product

Figure 5: t-SNE visualization for the encoding in the MF-SDAE hybrid model. In each subgraph, the users who subscribed to a certain product are highlighted.

Future Work

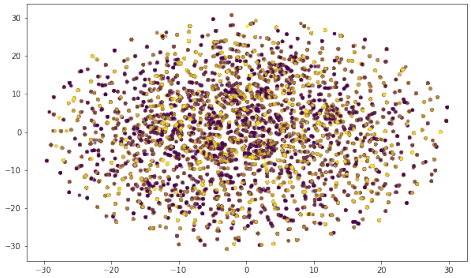
We believe that the performance of our model could be further improved with deep autoencoders by searching for better parameters, including the normalization to the user and product matrix and λ_u/λ_n. In addition, we can use batch normalization and adaptive learning rate to beat the difficulty of training deep models.

We can also extend our model by including the side information of products, such as textual descriptions of the financial products on the Santander website and Wikipedia. The product information can be encoded by SDAE, such that we have both user and product side information in our recommender system. There is a similar work done by Li *et al.* (2015) where the side information of both the users and the products is used. The large amount of information from the textual description makes it possible to apply more complicated while powerful models.

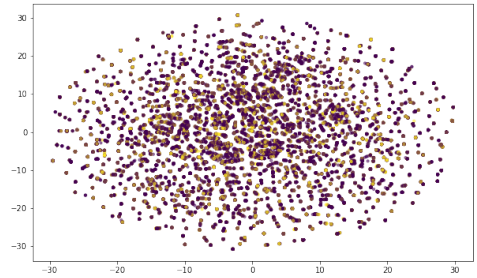


(a) Users subscribed to the most popular product

(b) Users subscribed to the second most popular product



(c) Users subscribed to the third most popular product



(d) Users subscribed to the fourth most popular product

Figure 6: t-SNE visualization for the encoding in the bare SDAE model. In each subgraph, the users who subscribed to a certain product are highlighted.

References

Y. Koren, R. Bell, and C. Volinsky, Computer **42** (2009).

R. Salakhutdinov, A. Mnih, and G. Hinton, in *Proceedings of the 24th international conference on Machine learning* (ACM, 2007) pp. 791–798.

H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, in *Proceedings of the fourth ACM international conference on Web search and data mining* (ACM, 2011) pp. 287–296.

H. Wang, N. Wang, and D.-Y. Yeung, in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2015) pp. 1235–1244.

Y. Hu, Y. Koren, and C. Volinsky, in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on* (Ieee, 2008) pp. 263–272.

S. Li, J. Kawale, and Y. Fu, in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (ACM, 2015) pp. 811–820.

F. Strub, J. Mary, and R. Gaudel, CoRR (2016).