altius

Recommender Engines with Altius

WHITEPAPER

Contents:

Background, problem and challenge

Recommender Systems (RS) seek to predict which previously unseen products a customer is likely to have a preference for and advertise or recommend those to the customer via different media. This includes modelling of customer demand, product preferences, similar customer groups and similar product groups. A typical scenario is that businesses send product recommendations to customers which are similar, or accessory, to historical purchases, reacting to their recent customer data.

The most popular technique for Recommender Systems is Collaborative Filtering (CF), which has been around since the early 1990s. The first CF algorithm, "user-user" collaborative filtering, was proposed in 1992 but failed to scale up to millions of users. In 2001 'item-item' collaborative filtering was developed [1], and Amazon has implemented their systems based on this idea of item similarity [2]. These are memory-based algorithms. CF has since gained wide attention after 2006 because of the Netflix Prize, and a series of new algorithms has been invented for CF to achieve personalization in Recommender Systems. There are many variations in the ways to implement a Recommender System because different businesses have different purposes and requirements.

For example, some want to recommend new products to the customers, while some want to improve customers' fidelity and satisfaction, and thus reward them for repeat buying. The recommenders in practice often combine both memory-based and machine learning approaches (hybrid approach) with specific rules tailored to the business.

Benefit cases

Recommender Systems create a positive user experience while driving incremental revenue. The primary goal is to increase the conversion rate, i.e. the number of users that receive the recommendations and act on them, and more importantly consume the recommended items assuming they fit the customer's needs and preferences. The personalised system increases the cart/basket value, either by adding the recommended items that a user does not normally buy or by replacing an item with a similar premium product.

75% of Netflix customers watch movies/TV series from their recommender system, which saves Netflix $1 billion each year [4]. Amazon credited the different recommenders they implemented for a 29% increase in total sales ($2.93 billion) in 2016 [3]. Best Buy also reported 23.7% sales increase in 2015 [5]. YouTube uses Recommender Systems to gain revenue; the more repeating viewing they can obtain, the more advertising opportunities they have.

The benefits from a Recommender System can also be non-commercial, such as improving the customer experience and increasing customer satisfaction. Some online supermarkets help customers shorten their shopping time by recommending relevant products, especially for new customers.

Recommenders also help improve the diversity of sales, promoting the products that a user may not have originally considered. Another important requirement is to encourage customer feedback. One challenge of building a successful recommender is the lack of explicit signals from customers. Collecting explicit feedback from the customers not only allows the business better understanding of their customers but also improves the recommender itself.

Data Science approach to RS

The chances of finding an off-the-shelf recommender that can achieve the business goal and satisfy all the requirements is slim. Most of the Recommender Systems are hybrid, which combine different algorithms to complement each other's limitations. It normally starts with a baseline model and is then modified to meet the requirements of the user.

A good choice of baseline models is Neural Collaborative Filtering [7] that generalises the traditional matrix factorization via a non-linear neural architecture. It uses minimal data of user-item interactions that a recommender normally requires and is easy to implement. It inherits the advantage of handling the data sparsity from the matrix factorization method by compressing the user-item matrix into a low dimensional representation.

While it has many advantages, it also has two main disadvantages: it cannot handle new users/ items and has low explainability of the results. This can be improved or overcome by adopting a hybrid approach. Integrating it with memory-based algorithms makes it possible to handle new users/ items and improve the model explainability.

Data requirement and challenge

The minimal requirement of data is the interactions between users and items either explicitly or implicitly. For movie recommendations, users' ratings on the movies are the explicit interactions, while for supermarket shopping, the buying of the products is an implicit interaction. Deep learning models like neural collaborative filtering will benefit significantly from high volumes of data. However, when the numbers of users and items grow, most CF algorithms will face a scalability problem. One solution is to use higher capacity machines. With resource limits, more creative solutions are needed, for example, the segmentation of users based on user features and then applying one recommender per user segment.

Another common data challenge is the lack of negative interactions between users and items. In this case, negative sampling can be used to enrich the data set with implicit negative feedback (non-action of not buying). It cannot automatically be assumed that products which were not purchased by the customer are the products they do not like, because the customer may not be aware of their existence. A reasonable assumption will be that if the products are generally popular, but the customer does not buy them, then this customer does not like them. Thus, for each user, it randomly selects the negative items from a pool of items that he/she has not bought in the context of the popularity of the items.

Fall-Back Model for cold start

The cold start challenge is also very common when building a Recommender System. This can be for new users or new items. New users happen more often because new customers join services all the time. Some recommenders are even targeted for attracting new customers. Similarly companies use marketing campaigns to promote new products during the year. For example, some products are only available during Christmas or Easter time and every year they may be different.

There are solutions at two levels depending on whether feature data is available on the new users/items and on whether personalization is consented to.

If the feature data is available and personalization is consented to, the memory-based CF algorithms can be integrated into the hybrid model. Based on the user/item features, we can find similar users/items and recommend to new users the items consumed by similar users or recommend the new items to those who bought similar items.

When the new items are added for periodic or temporary sales and no feature data is available, an arbitrary weight is applied to promote them over regular popular items.

For new users, a general Fall-Back model can be introduced to produce a global recommendation based on the global item popularity, with some advanced weighting methods. One example of a weighting method for retail companies is the reversed exponential decay weighting, where the furthest day/week/month has the highest weighting on the item popularity. The assumption is that users are more likely to buy the product that they bought a while ago than those they just bought yesterday.

More advanced Fall-Back models are sometimes needed. For instance, Market Basket Analysis (MBA) is an association analysis used by large retailers to uncover associations between items.

It works by looking for combinations of items that occur together frequently in carts/baskets, and allows retailers to identify relationships between the items that customers buy. It involves three main association rules.

1. Support: Frequency that a pair of items occurs in the same cart/basket. In many instances, high support means a useful relationship. However, there may be instances where a low support is useful if you are trying to find “hidden” relationships.
2. Confidence: Measure of the reliability of the rule, the probability that a user buys item B given he/she has bought item A.
3. Lift: Ratio of the observed support to that expected if buying two items were independent. A Lift of 1 means that the occurrences of A and B are independent.

When recommending the popular item A, item B (which is often bought together with it) is also recommended. MBA aims to increase the cart/basket size, i.e. cross sell. It can also be used to identify the promotions of popular combination of items.

As MBA is designed for the retail industry, it may not work for other industries. For example, watching videos on YouTube or Netflix, a normal person only watches one video at a time, so the support measure no longer applies. The confidence or even lift may still be feasible. Confidence here is the probability that a user will watch video B shortly after video A. Lift can measure the independence of video A and B within a short period. However, the ‘short’ period needs to be defined properly.

Personalised model

Once the user-item interactions are available, we can create the personalised Recommender System. In most circumstances, a user only consumes a small subset of items out of the full item space. Thus, in the user-item interaction matrix, many entries are empty, i.e. these interactions never occurred or have not been recorded. This feature of the data is called sparsity.

The data sparsity can be solved by the matrix factorization (MF) method which approximates the user-item interaction matrix by the product of lower rank matrices i.e. the rating matrix (R) can be decomposed into an m by k user (where k is the number of latent factors) and transposed n by k Item matrix. Once the latent factors are determined, they can be used to predict the missing user-item interactions. These factors loosely correspond to the hidden user or item characteristics, e.g. the first latent factor may represent how healthy an item is, correspondingly the first user latent factor represent how healthy oriented the user is. An example is shown in Figure 1.

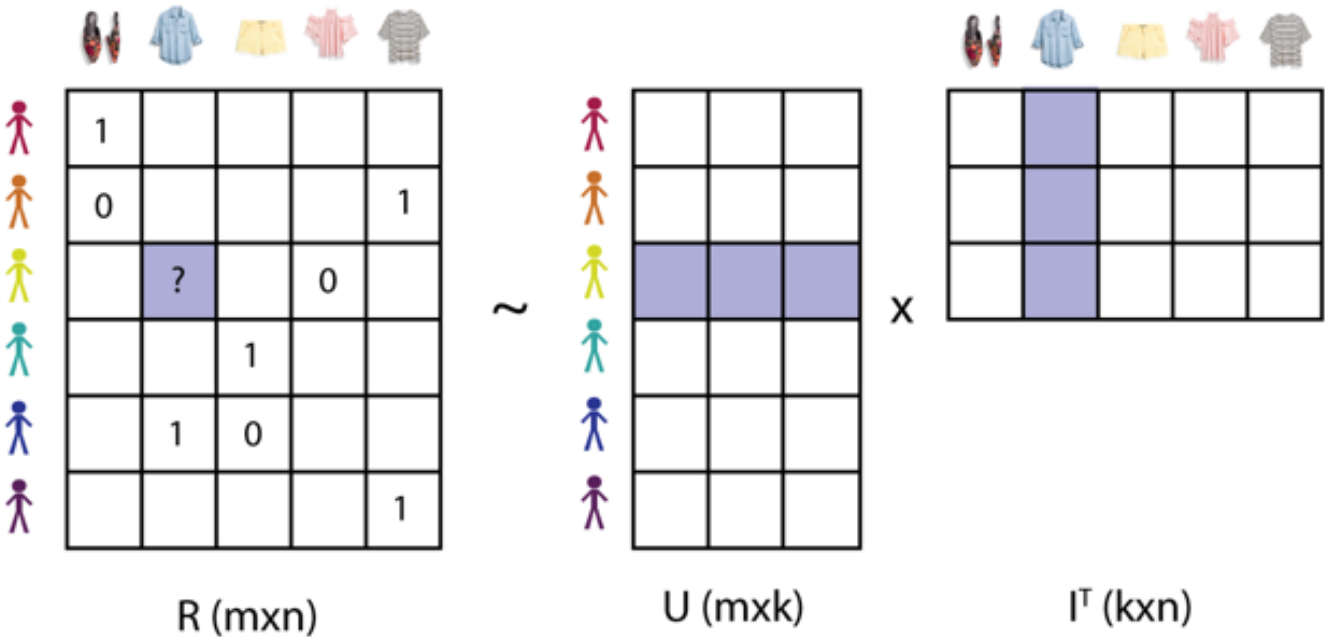

Figure 1: **UV decomposition of user-item matrix**
Image from: https://multithreaded.stitchfix.com/blog/2018/06/28/latent-style/)

The matrix factorization described above is called UV decomposition, which is a generalised or non-linear version of Singular Value Decomposition (SVD). UV decomposition is the underlining method of Neural Collaborative Filtering.

SVD factorises the user-item interaction matrix into three parts: user and item hidden characteristics and in the middle an identity matrix. This identity matrix ensures that the identified hidden characteristics are orthogonal, i.e. independent of each other.

UV does not impose the restriction of the identity matrix in the middle, thus the user and item vectors are not orthogonal. Also, for this reason, the UV decomposition has multiple local optima. It does not restrict the solution space, only reduces the error (e.g. RMSE).

The baseline model structure is shown in Figure 2. The input demonstrated here is the one-hot encoding vector: 1 represents the interaction between user and item has occurred, while 0 for when it has not. This can differ for different problems.

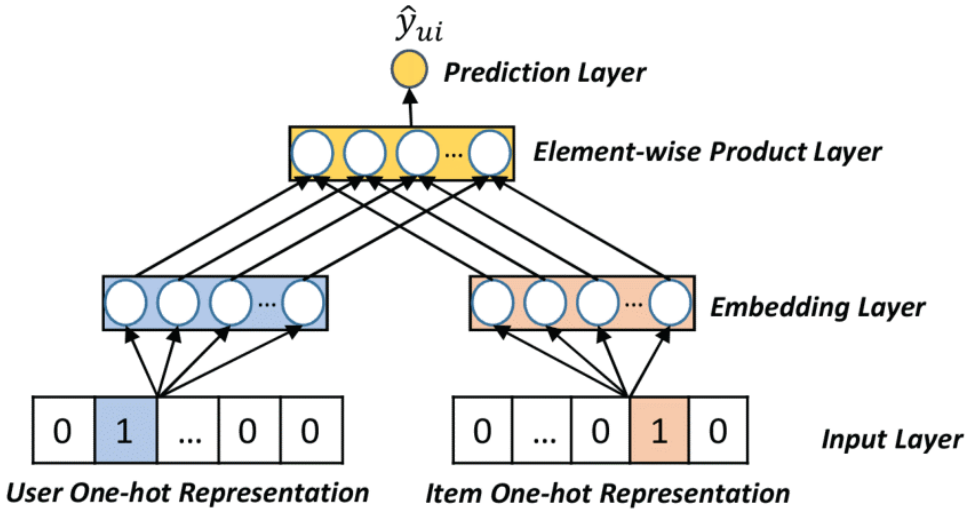

Figure 2: **Neutral Collaborative Filtering UV decomposition model.**

For example, in the case of movie recommenders, the entries are the ratings. The output are the probabilities of every user-item pair, representing how likely the interaction will occur.
The UV decomposition can be easily implemented to derive the embeddings instead of implementing a matrix factorisation. Using this approach has several benefits; it is more computationally efficient and flexible when extending the model. A simple extension is to add sample weighting to quantify the level of item importance/ discount. More elements can be added to accommodate additional data sources. For example, to differentiate the recommendations between morning and afternoon, or between weekdays and weekends.

In the context of morning versus afternoon time segments, items can either be positive, negative or neutral so it would be sufficient to use the already present levels in the time segment categorical variable and set the number of latent factors to 1.

The framework is extendable in that each additional data source such as holiday versus weekday flag or time-of-day segmentation can be represented as a distinct number of latent factors which interact specifically with either users or items in a defined way. As before, the interactions are modelled as a dot product layer and then summed up into a scaler before being fed into the last layer which applies a sigmoid transformation to obtain a probability between 0 and 1.

Model evaluation and testing

Offline model evaluations

There are two general ways to evaluate a model offline: hold-out and cross-validation. The former is more suitable for evaluating Recommender Systems because it avoids the problem of variation in training data. Cross-validation can lead to unjustified decisions by pooling the results across partitions that are not independent for ranking algorithms [9]. To apply the hold-out method, the historical data is split into three sets: training, validation and test sets. The validation set is used to tune the model/models. The test set is used to evaluate the final model/models.

There are many evaluation metrics available to assess different aspects of a recommender, and based on different metrics, the success of the system can be measured very differently. The following are example metrics of recall, novelty, and diversity.

- Recall: The proportion of the 10 products with the highest propensity as predicted by the model which appear in the ground-truth (i.e. held-back) set of things actually purchased by the customer.
- Novelty: Items which appear in the test set but not the training set. Once all items have been flagged on a customer-by-customer basis, the ratio of all top-10 recommended lists which have at least one novel item within them is the metric we use for novelty.

- Diversity: Standard deviation of each item's purchase propensity rank across all users. It indicates the average spread of item ranks across customers and hence the level of personalization.

A/B testing

Offline evaluation is not the most ideal estimate of accuracy because the purpose of building a recommender is to change user behavior by recommending things that they otherwise would not see or buy. In an online system, the user reactions are measured with respect to the presented recommendations. Therefore, user participation is essential in online systems. For example, one might measure the conversion rate of users clicking on items that were recommended. Such testing methods are referred to as A/B testing whereby users are directed to one of two groups: A denoting control and B the random experimental group. The objective of the experiment is to measure and compare a KPI of interest, for example conversion rate or basket size in both groups, to assess the performance of the recommender against no recommendations being given.

Technology and architecture

As outlined above, an "out of the box" Recommender System is unlikely to meet the needs of a particular business, especially as the uses of recommenders span multiple industries. A hybrid approach of reusing pre-built and tested components whilst developing the models to drive the business's aims will provide greater flexibility and reach production faster. In this scenario standard components such as data ingestion, cleaning, feature engineering, monitoring and model management are pre-built and well tested, thus reducing development costs and risk.

Various techniques/tools are available to improve the efficiency of the pipeline for the different components above. For example, the data ingestion, cleaning and processing can be implemented in a distributed fashion using Spark/Databricks. The matrix factorization CF model can be built in Spark using alternative least squares, thus training and prediction of the model are parallelized. Training can also be distributed across multiple GPUs.

Overall the choices of the technologies depend on the business requirements and framework, as well as based on the data volume and frequency of the model training and consumption.

Three possible scenarios are:

1. Batch train/batch predict: In this scenario the model is trained periodically, and the output is also returned in batches. This is suitable for large data volume and low frequency of model training and consumption. The model training and predicting can done using Kubernetes or Spark when it is feasible.
2. Batch train/live predict: In the event where the recommendations need to be returned dynamically, but the model can still be trained periodically. This is suitable when the training data is rich or the model is complex, the recommendations need to be live and frequent, or only required for a medium number of people. The trained model parameters can be hosted on the cloud and triggered by a live request.
3. Live train/live predict: Both the model training and predicting are required for each incoming request. This is suitable when the data volume is small or the model is simple. This scenario must make full use of cloud technologies for handling streaming events.

Applications of Recommender Systems

Immediate recommendation

Some businesses require dynamic recommendations. For instance, the recommendations are built based on the live browsing histories or the products the customer is adding to the basket, then the live train/live predict deployment fits in place. This is more challenging because it is constrained by time. For example, when browsing a product on Amazon, the relevant recommended products are immediately suggested under the chosen product. The input data must be manageable in size and the model training and prediction need to be completed in seconds.

Daily recommendations

When the recommendation is triggered by logging into the account or stepping into the physical store, the batch deployment can be applied either on model training or on both training and predicting of the model. The recommendations are not required to build on the current activities, but the historical ones. When logging into Netflix, a list of movies 'Because you watched X' will appear on the front page. The model is pre-trained. The prediction may also be prepared and stored waiting for the customer to login.

If the pre-completion of prediction for all the users is too costly, then the prediction may be done during the log-in.

Long-term campaigns

Discount or voucher advertising from retailers can be deployed in batch for both training and predicting. It is suitable when the business plans the marketing campaigns or wants to enforce some rules.

Suggesting alternatives

Another type of recommendation suggests alternatives, for example when the products you want to buy are out of stock. The model will be pre-trained, and the prediction can also be ready in advance, but extra filtering is required on the fly, i.e. checking the availability of the original product and selecting the alternatives.

References

[1] The Economist (2005, Mar 12). United we find. Retrieved from https://www.economist.com/technology-quarterly/2005/03/12/united-we-find

[2] G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," in IEEE Internet Computing, vol. 7, no. 1, pp. 76-80, Jan.-Feb. 2003. https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

[3] Sigmoidal. Recommendation Systems – How Companies are Making Money. Retrieved from https://sigmoidal.io/recommender-systems-recommendation-engine/

[4] Carlos A Gomez-Uribe and Neil Hunt. 2016. The netflix recommender system: Algorithms, business value, and innovation. TMIS 6, 4 (2016), 13. https://www.academia.edu/33391850/The_Netflix_Recommender_System_Algorithms_Business_Value_and_Innovation

[5] N. Rajendra, A. Dewan and M. C. Colakoglu. 2012. CS229 Project - Best Buy Recommendation System. http://cs229.stanford.edu/proj2012/RajendraDewanColakoglu-Best-BuyRecommendationSystem.pdf

[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings of the WWW. 173--182. https://www.comp.nus.edu.sg/~xiangnan/papers/ncf.pdf

[7] Dawen Liang, Rahul G Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. arXiv preprint arXiv:1802.05814 (2018). https://arxiv.org/abs/1802.05814

[8] Bengio and Y. Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. Journal of Machine Learning Research, 5, 2004. http://www.jmlr.org/papers/volume5/grandvalet04a/grandvalet04a.pdf

[9] Bengio and Y. Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. Journal of Machine Learning Research, 5, 2004. http://www.jmlr.org/papers/volume5/grandvalet04a/grandvalet04a.pdf

info@altiusdata.com

https://linkedin.com/company/altius-data/

www.altiusdata.com