COMP642 Scribe of March 9,2023

Andrew Chu, Rocky Wang, Xuwei Tan, Kyle Sneed

March 16, 2023

1 Collaborative Filtering

1.1 Concept

Collaborative filtering is a type of recommendation algorithm used in machine learning that makes predictions about a user's preferences or interests based on the opinions and behavior of other similar users. The basic idea behind collaborative filtering is that people who share similar interests and preferences are likely to have similar opinions about a particular item.

1.2 Netflix Problem

The Netflix recommendation challenge was a competition launched by Netflix in 2006 with the aim of improving the accuracy of its recommendation algorithm. The challenge involved providing participants with a large data set of movie ratings and challenging them to develop an algorithm that could predict the ratings that users would give to movies they had not yet seen.

The data set provided by Netflix contained over 100 million movie ratings provided by more than 480,000 users. The data set was divided into training and testing sets, with the training set containing about 100 million ratings and the testing set containing about 4 million ratings. Participants were required to develop an algorithm that could predict the ratings for the movies in the testing set as accurately as possible.

The challenge offered a prize of 1 million to the team or individual that could improve the accuracy of Netflix's recommendation algorithm by at least 10%. The competition attracted thousands of participants from around the world, including data scientists, researchers, and machine learning enthusiasts.

The challenge continued for three years, with the winning team being announced in 2009. The winning algorithm, developed by a team of researchers from ATT Labs, achieved a 10.05% improvement in the accuracy of Netflix's recommendation algorithm.

1.3 Collaborative Filtering and Netflix

Collaborative filtering can be used by Netflix to predict the taste of the users on different movies and videos. One example of the data attributes could be shown in Table 1:

	UID(user ID)	MID(movie ID)	No.stars
1			
2			

Table 1: Example of data attributes.

In this example, the UID and MID are categorical data.We will first vectorize these two attributes. We will then train the vectorized data attributes in a neural network. The resulting gradients can be used as a score function. The predicted No.stars the user would give on a certain movie can be generated by $Score(UID(vectorized), MID(vectorized))$.

1.4 Problem with Collaborative Filtering for Recommendation

Although collaborative filtering can be a relatively accurate model for recommendation systems, it can not deal with timestamp. In other words, it cannot weight the importance of data points based on their timestamps. For example, if a Netflix user used to watch a a lot of action movies, but recently switch to watch cartoons because he spent more time with his daughter. Without consideration of timestamps, the system will continue to recommend action movies to the user for a long time. However, in reality, the more recent history data is more relevant to the prediction. Hence we need machine learning models that can analyze time series data for recommendation systems. Timestamp will also be treated as a data attribute as in Table 2:

	UID(user ID)	MID(movie ID)	No.stars	timestamp
1				
2				

Table 2: Example of data attributes.

2 Autoregressive model

2.1 Definition

An autoregressive (AR) model is a statistical model used to analyze time series data in various fields including statistics, econometrics, and signal processing. It is used to represent a type of random process that describes time-varying phenomena in natural, economic, or behavioral systems. The autoregressive model assumes that the output variable is linearly related to its own past values, and also incorporates a stochastic term which accounts for unpredictable noise in the system. The model can be expressed as a stochastic difference equation or recurrence relation, where the current value of the output variable depends on its previous values and the stochastic term.[**?**]

2.2 Linear Function

The order of the AR model, denoted by t, refers to the number of past values of the variable used to predict its current value. The most common autoregressive model is the AR(p) model, where p represents the order of the model.

$$P_t = \alpha P_{t-1} + b$$

P_t: represents the value of the dependent variable (also called the endogenous variable) at time t.
P_{t-1}: represents the value of the dependent variable at time $t-1$, which is the previous time period. This is called the lagged value of the dependent variable.
α: represents the coefficient or parameter that quantifies the strength and direction of the linear relationship between P_t and P_{t-1}. This is called the autoregressive coefficient.
b: represents the intercept or constant term in the model. It captures the impact of all other factors that affect the dependent variable besides the lagged value of the dependent variable.

$P_0, P_1, ..., P_t$ can be used to predict $\hat{P_{t+1}}$. We will continue to use the predicted value as we assume that our prediction is accurate.

2.3 Expansion

The autoregressive model is a special case of more complex time series models, such as the autoregressive-moving-average (ARMA) model and the autoregressive integrated moving average (ARIMA) model. These models have a more complicated stochastic structure and incorporate additional components such as moving averages and differencing to improve their performance on more complex time series data. The autoregressive model is also a special case of the vector autoregressive model (VAR), which models systems with multiple evolving random variables and interlocking stochastic difference equations.

3 Recurrent Neural Networks(RNNs)

3.1 Why do we need RNN?

After training the neural network model, given an x in the input layer, it is possible to get a specific y in the output layer after passing through the network. they can only handle one input individually, and the previous input is completely unrelated to the latter ones. However, some tasks require better processing of sequential information, which means, the previous inputs are related to the later ones. For example, when we are comprehending the meaning of a sentence, it is not enough to understand each word of the sentence in isolation; we need to process the whole sequence of words connected together. It is obvious that the previous word in a sentence actually has a great influence on the lexical prediction of the current word. To solve some similar problems and to be able to process the information of the sequence better, Recurrent Neural Network(RNN) was born.

3.2 The structure of RNN

Figure 1 shows a simple Recurrent Neural Network structure. It can be seen that the output value P_{t+1} of the hidden layer of the recurrent neural network depends not only on the current input X_t but also on the output of the last previously hidden layer.

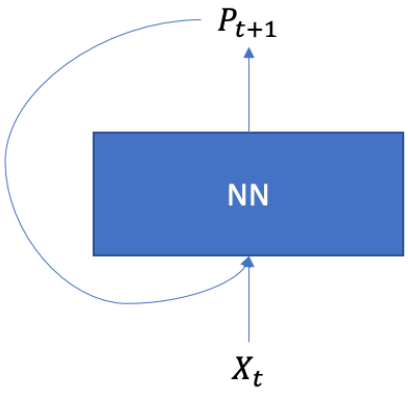

Figure 1: A simple structure of RNN

The RNN model based on the timeline expansion is shown in Figure 2.

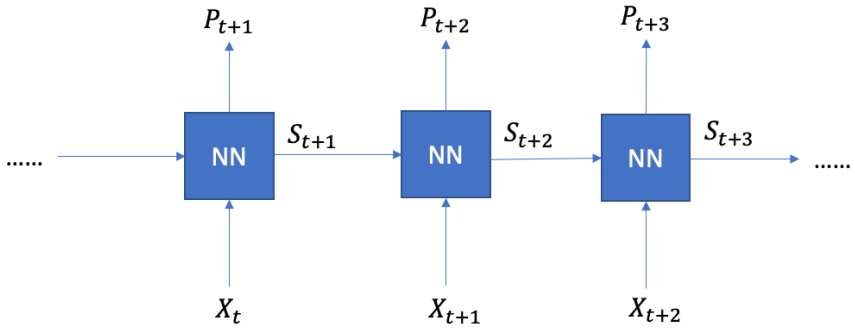

Figure 2: A simple structure of RNN based on timeline

The training method of the Recurrent Neural Network (with known X_t, X_{t-1}, X_{t-2}, to predict

X_{t+2}) can be represented by the following equation:

$$\hat{X_{t+2}} = f(\hat{X_{t+1}}, X_t, X_{t-1}) = f(f(X_t, X_{t-1}, X_{t-2}), X_t, X_{t-1})$$

3.3 Combining Static and Dynamic Features

Referencing Table 2 we can let the UserID and MovieID be our Static Features while sentiment (Star rating) and timestamps be our Dynamic Features. From our motivating example we are interested in assigning a dynamic sentiment vector which combines the sentiments (Sent1, Sent2) and timestamps for a truncated history. This is shown below in Figure 3

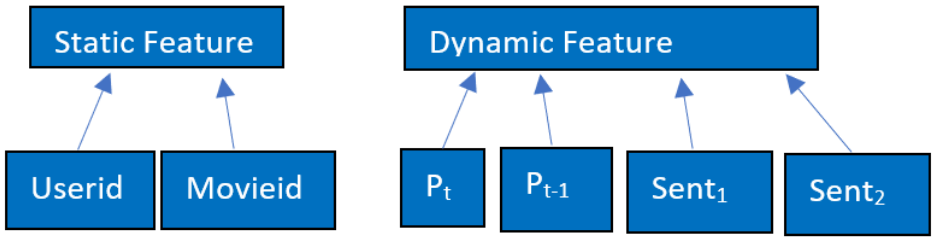

Figure 3: Static and Dynamic Feature

This feature combination would be fed to a Neural Network and trained to predict the next movieid to suggest a given user. Referencing the above figure, the model would relate across the static features for similar dynamic features.

Although the sentiment is aggregated daily, the bottleneck is the sentiment classification which takes increasing amount of time as targeted accuracy increases. From the netflix example this leads to re-training the model every 1-2 weeks.

One boundary case is how to suggest movie's for a new user who has no viewing history. Additionaly if the user gives no supporting metadata beyond the UserId, one solution is to suggest movie's at random until a dynamic history has built up. Otherwise for the initial movie's the model can relate across given metadata (zipcode, age, gender) to suggest initial movies to the user.

Another novel application to combining static and dynamic features has been found in applying RNN and single layer perceptron's to improve cardiac arrest prediction algorithms[3]. The authors of the article concluded that,"the addition of the static features improves the performance of the RNN than would otherwise by using the sequentioal and static features alone."[3]

4 Reference

1. https://ml.berkeley.edu/blog/posts/AR_intro/
2. https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn
3. Tule, S., Oguntayo, S. (2022, June 12). Time Series Classification tutorial with LSTM Recurrent Neural Networks. Omdena. https://omdena.com/blog/time-series-classification-model-tutorial