Amazon ML Engineer Hiring Challenge

Girish Ewoorkar

---

1)What are Pros and Cons of recommendation by this approach

Pros are:-

- The "**Implicit Data**" is easily available unlike explicit Data.
- This ensures "**scalability**" allowing us to tackle bigger datasets which is one of the drawback of Collaborative filtering (CF) technique
- Since CF based recommender system require Computations that are very expensive and grows exponentially with number of users and items, now by dividing customers into their respective cluster we can apply recommendation algorithm separately to each cluster there by **reducing the time complexity**.
- Partially resolves issue of "**Sparsity**" , since we have classified the customer to their similar category the of user-items pair in matrix is reduced
- Partial "**personalizable**" as the goal is to understand user's preference which indirectly reflect the opinion through observing user behavior by classifying them to similar group

Cons are:-

- As we are using implicit data for classifying user and among the feature was "**click through rate**" which was top important feature (according to my solution) in classifying the customer, the dataset is **heavily biased** since the CTR is heavily dependent on position of link on the page.
- Implicit feedback may not be the right choice to classify the user as the click on link might just teaching us about how clickbaity the link is rather than the relevance of actual preference of user.
- "**Cold Start**" w.r.t to user, if user is new has not interacted with the system much then its not possible to classify and recommend the items.
- Partially "**Not personalized**" since recommendation are per cluster not per user as in CF.
- Classifying the user and then recommending the items may not work for longer run since
    - user preference/trends changes over time hence the category which user belongs at time T may not be same at time T+k
    - clustering/classifying is bit weak technique because it identify user groups and recommend each user in this group same items.

2)Propose an architecture that will work more efficiently when building a recommendation engine for an e-commerce platform.

- In real scenario the user preference in past will not be much relevant for the present need of user especially in ecommerce platform like amazon where user buy variety of products ex during festivals user may buy gifts for his close ones which is not users choice of product hence recommending the similar items which user wants is intuitive only during that session and will be not be relevant for future sessions.

- There are times most of ecommerce websites don't have user authentication system or incase of Amazon where it allows users to search the products without asking user to login or authenticate themselves and users only trusts the platform until they find the items they need before logging in or authenticating themselves. Hence during these scenarios the platform needs to recommend the valid relevant products else user may not be satisfied and may not consider to interact with platform again.

So based on above scenarios the most efficient and intuitive algorithm is to provide "***session based recommendation using deep learning***".

Let's see how Session based recommendation is better than the recommendation through current approach…

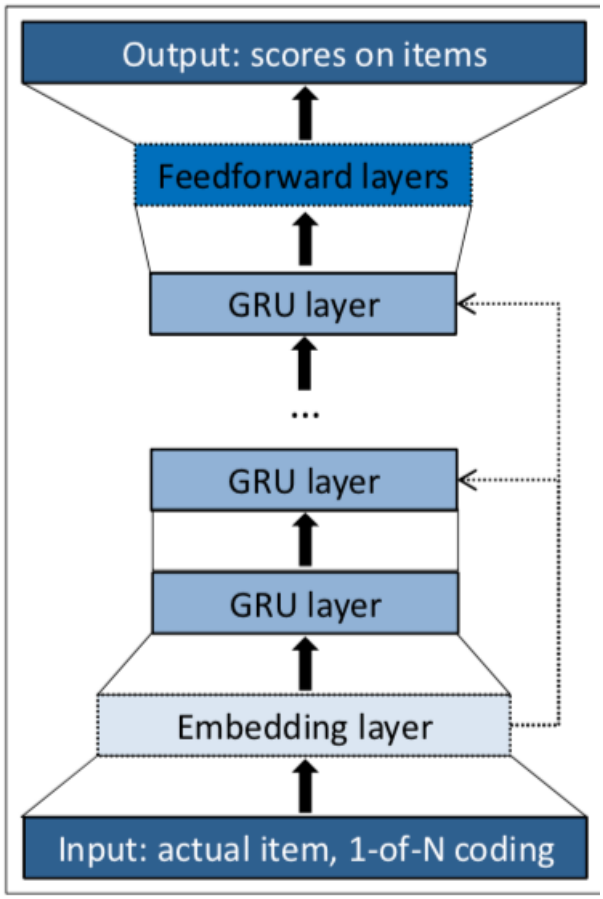
- It reduces the "**cold start**" problem since session based recommender system rely one sequence of recent interactions without requiring additional details (historical like past clicks, orders etc and demographics) of the user.
- It takes into account of users need or preferences/trends in that session
- Does recommend variety of products unlike the current approach (as mentioned in the cons section).
- It also provides personalize recommendations.

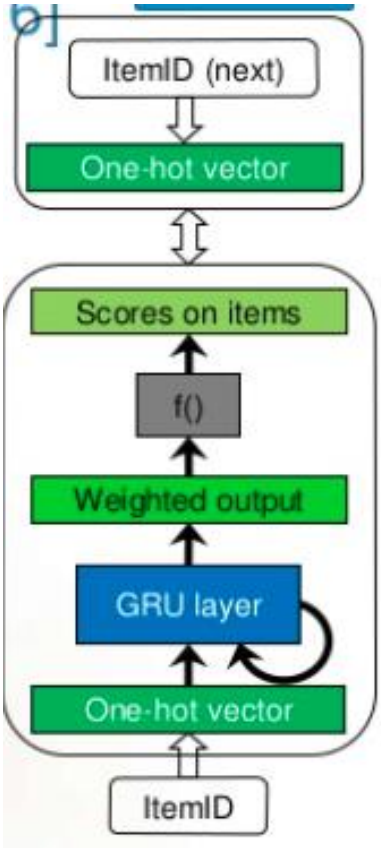
Brief intro of session based recommendation.

- This approach utilizes each session of users by feeding it into GRU where input is current state of session and output is the item of next event in session

As in above image user go through series of similar items in each session model has to predict next similar item until user lands on desired product.

Architecture:



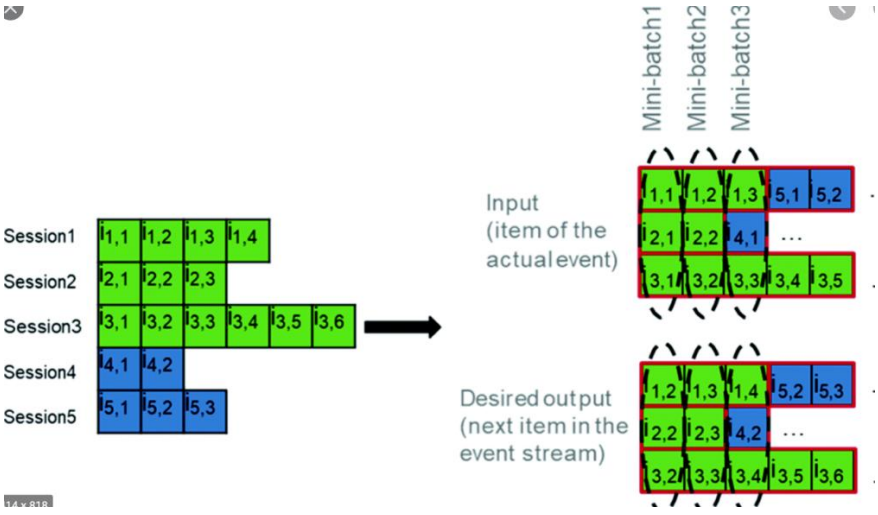


Input: one hot encoded vector of current item in actual event

Output: Probability score on items for being the next event in steam

If sessions are relatively small then we can stack only one layer of GRU would be enough

To increase performance session parallel mini patches is found to be beneficial.



Since GRUs are sequential model which works well on longer sequence of data and sessions are relatively smaller to get better performance sessions are concatenated and in mini batches the first item of each session is considered for first batch and 2nd item of each session is considered for second batch and so on. The output is next item for each session.

The goal is to capture how sessions evolve based on high variance in length of session

Loss function:

Instead of cross entropy loss it was found that using Pair wise loss functions such as "Bayesian personalized ranking loss function" works better where w tries to optimize function such a way that positive sample data(item that user has clicked) has higher ranking than negative sampled data (item that user has not clicked) and that we put penalty if that's not the case

- BPR
  - Bayesian personalized ranking
  - $L = -\frac{1}{N_S}\sum_{j=1}^{N_S} \log\left(\sigma\left(\hat{r}_{s,i} - \hat{r}_{s,j}\right)\right)$
- TOP1
  - Regularized approximation of the relative rank of the positive item
  - $L = \frac{1}{N_S}\sum_{j=1}^{N_S} \sigma\left(\hat{r}_{s,i} - \hat{r}_{s,j}\right) + \sigma\left(\hat{r}_{s,j}^2\right)$

By adding regularization term on negative sampling on BPR get TOP 1

Sampling Data:

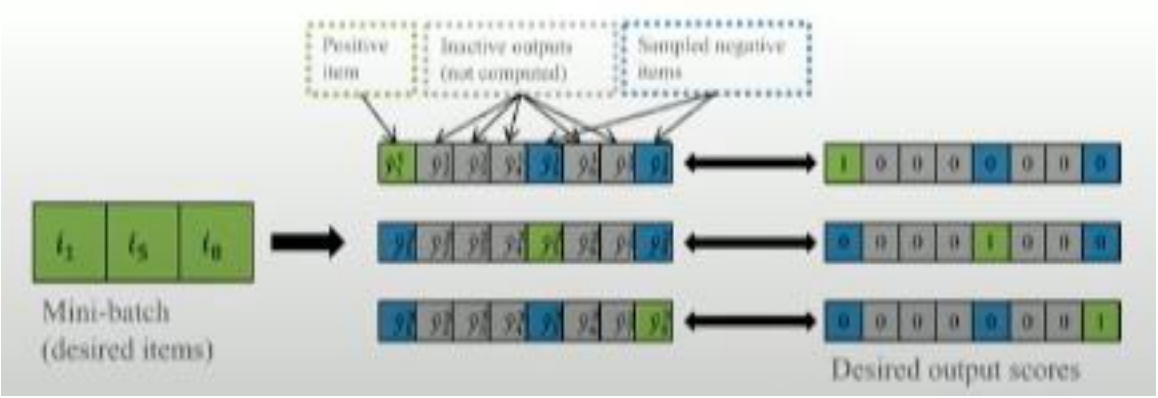
To compute loss on all pairs of positive and negative items will not feasible as number of items high and model needs to be quick and trained frequently so we have to sample the negative data based on others positive data for current user
We have to sample data out of mini batches in session

EX: lets say there are three 5 items (i1,i2,i3,i4,i5) and 3 users (A,B,C)

User A click on i1 and user B click on i4 and user C click on i5

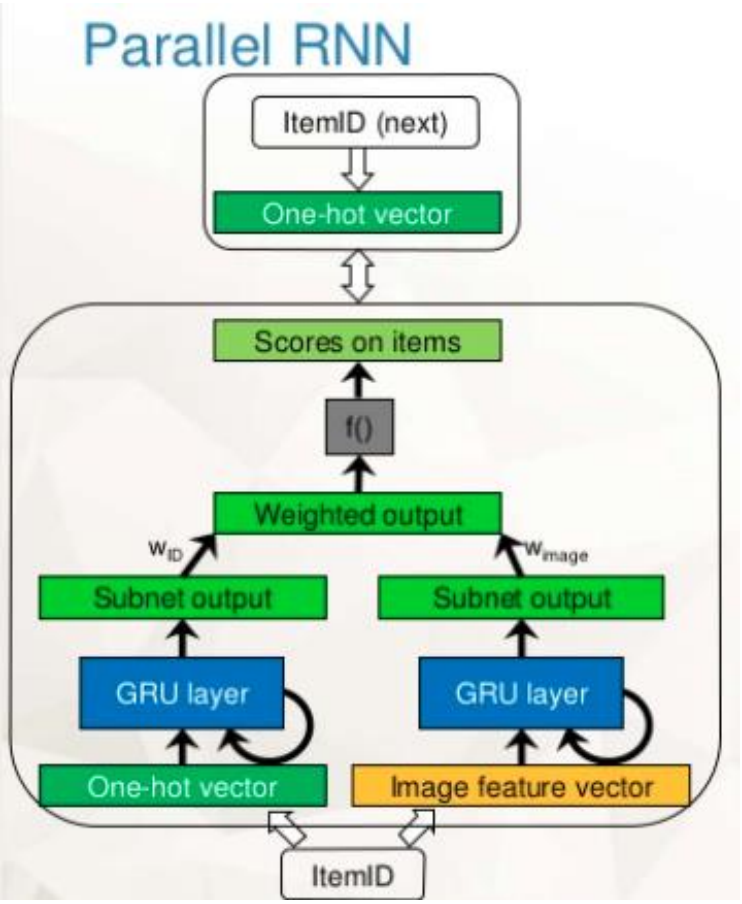
Then for user A positive data = i1 and negative data= i4 and i5
User B positive data =i4 and negative data = i1 and i5
User C positive data = i5 and negative data =i1 and i4

To improve again sample additional data from as well



Parallel GRUs (RNN):

This is just based on item id but we have not considered yet item features like images, price etc. so for that training Parallel GRUs works better like first training Item ID a parallelly with item feature separately and then concatenate both vectors and then calculate the scores.



To get image vector from images of items we can any apply transfer learning I.e apply pretrained images with model like VGG16, Google net etc.

The loss can be updated with techniques like residual training like train one subset NN fully then train other Subset NN or Alternating training per epoch or interleaving training per mini batch based on accuracy w.r.t dataset choose technique

Now we can also include text features instead of image feature and we can get vector by word2vec or tf-idf or using bert etc

Personalized session based recommender system:

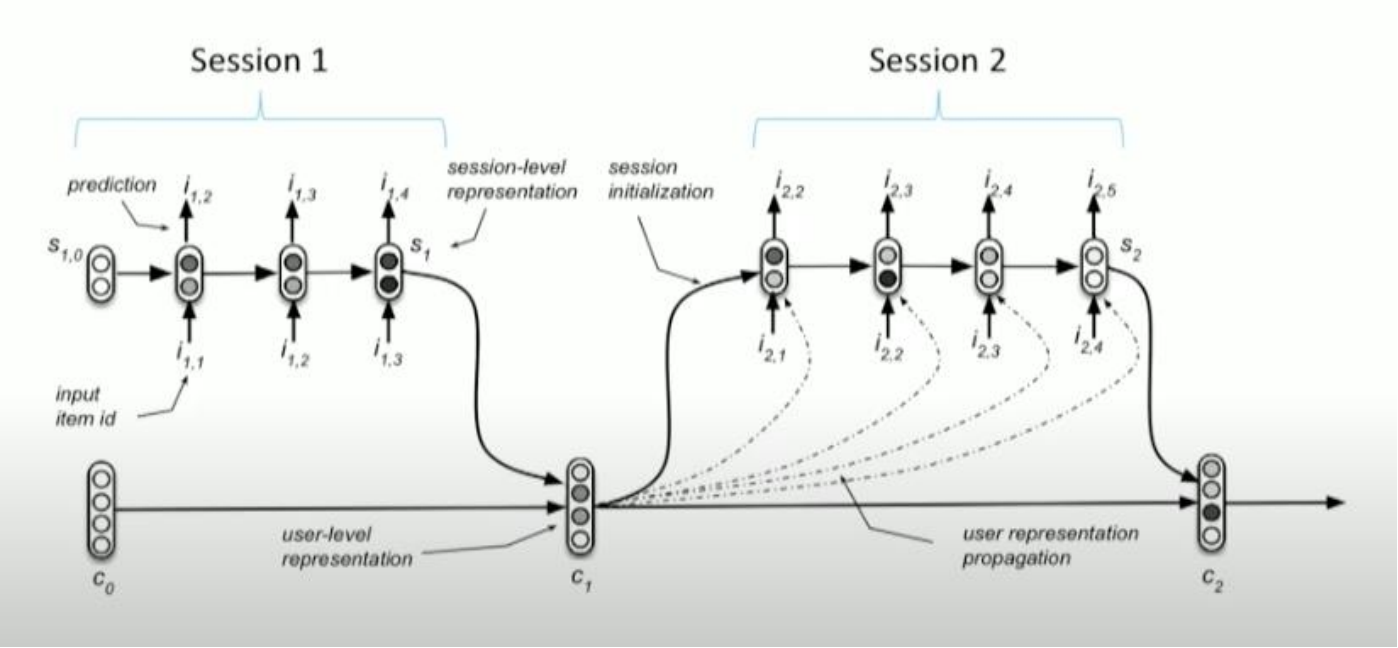
We provide more personalized session based recommendation i.e based on previous sessions we can recommend items to current session to begin with.

Here hierarchical neural networks GRU is applied.

Where previous session is used to initialize the hidden state of next session.

Initially for fist session the hidden state is initialized randomly for second session the hidden state is initialized/updated by previous session.

As can see in below pic for particular user how previous sessions are used for next sessions. I that way two identical sessions from different users will produce different recommendations



Some cons are:

- Limited GPU memory since the input layer dimension is very high.
- Slow training and testing times.

So to improve the performance bit it is suggested to use bloom filter on input and output by researchers to reduce the dimensionality and increase the efficiency.

References:

- The research paper I followed for the above approach: https://arxiv.org/pdf/1511.06939v4.pdf
- Blog link https://medium.com/amaro-tech/session-based-fashion-item-recommendation-with-aws-personalize-part-1-8410e85ccbae
- And some youtube videos to understand about recommendation and session based recommendations