Predicting Implicit Ratings

Group 15
William Guenneugues
Brendan Jenkins

Project Overview

- In this project we are trying to predict the likelihood that a given user will interact with a given item.
- To do so we leverage user history data that tells us which items each user has interacted with in the past. This type of data is known as implicit user feedback.

Data Overview

We used data gathered from 3 csv files:

- Train.csv: train data with columns user_id, item_id and context_feature_id
- Items.csv: data with categorical features for each item with columns: item_id, item_feature_id
- Test.csv: test data with columns user_id, item_id and context_feature_id

Matrix Factorization

- The model we used was a Matrix Factorization model with embedding size of 50.
- The features we used were user_id and item_id.
- We added two bias vectors to account for user and item bias and it decreased the validation loss.

DataLoader

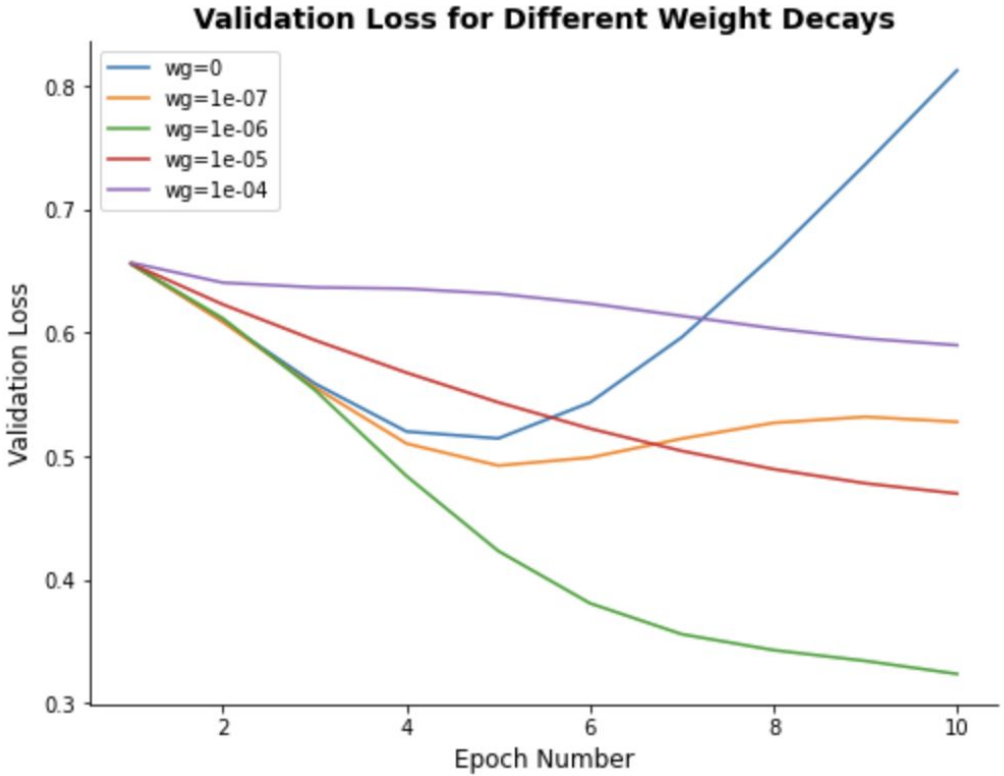
- With so many rows of data, our model might overfit quickly.
- We implemented a DataLoader on one of our models to counteract this.
- A DataLoader divides the training data into smaller batches and trains the model on one batch at a time.
- This can be used to prevent overfitting because your model is taking multiple, less accurate steps per epoch, instead of just one step per epoch.
- Since the DataLoader is updating the optimizer more, our best learning

Negative Sampling

- When dealing with implicit feedback, we do not have any negative feedback from that user. We only have information about the users' behavior, like their purchase history or the browsing history.
- The problem with this is that there is no negative feedback.
- Therefore, we must negative sample. This is where we fill empty spaces with zeros.
- Filling all unseen combinations with zeros is expensive, so we limited it to about a 1:1 ratio.

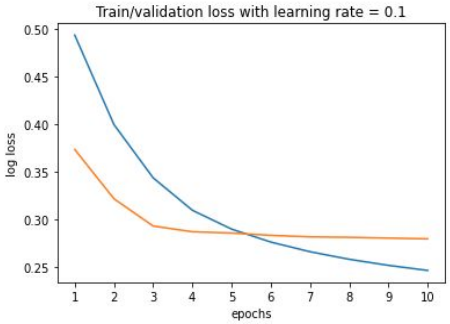
Weight Decay Hyperparameter Search

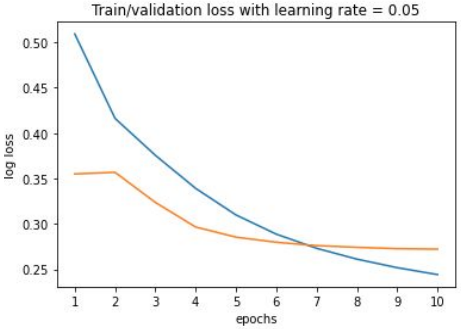
- We ran an experiment to find the best weight decay.
- We fixed the learning rate to 0.1 and the number of epochs to 10 and trained models using varying weight decays.
- We found that the weight decay that gave us the best validation loss was 1e-06.

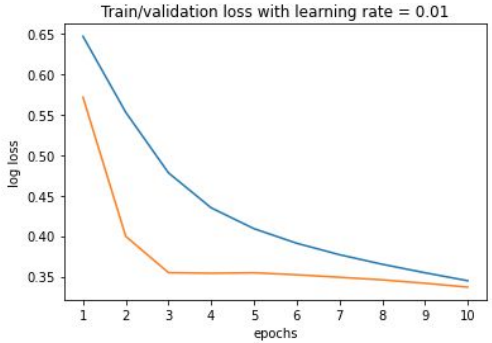


Learning Rate Hyperparameter Search

- Picking a Learning Rate can be challenging.
- We explored various learning rates and epoch combinations.
- We found that the best learning rate was a combination of different rates.
- We settled on 10 epochs at a learning rate of 0.1, followed by 5 epochs at a learning rate of 0.01.
- The larger learning rate at the beginning allowed us to approach the minimum faster.
- While the smaller learning rate at the end ensured us that we did not overstep the minimum.







Final Model

- Learning Rate = 0.1 for 10 epochs.
- Then, Learning Rate = 0.01 for 5 epochs.
- Weight Decay = 1e-06.
- DataLoader with batch size of 100,000.
- Negative Sampling of about 1:1 to about 1.2:1.

Lessons Learned

- When working with Recommendation Systems and only implicit feedback, we must Negative Sample.
- When working with so much data, using a DataLoader can be an effective way to avoid overfitting.
- Try a range of possibilities for your hyperparameters.
- Use a random seed so your results are replicable.
- Chart your progress. We spent a lot of time in the beginning trying random hyperparameters, and when we wanted to go back to a certain model we could not replicate that model because we did not keep track of our results.