Report

November 23, 2020

0.0.1 Kaggle Team - Severus Snape

0.0.2 Task : Play Prediction

Different approaches experimented -

1. Popularity based metrics - Baseline
2. Jaccard similarity - Performs worse than strong baseline
3. Cosine similarity - Performs worse than strong baseline
4. Popularity + similarity based model - Strong baseline
5. Features extracted (similarity, popularity) + Training a classifier

   5.1 Features extracted = (similarity between users, popularity of the given game)

   5.2 Classifiers used - 5.2.1 (Logistic regression (Value of C was found to be 10^{-7} for the validation set accuracy = 0.69) 5.2.2 (AdaBoost - To imitate ensemble learning)

   - 5 fold cross validation was used for computing the hyperparameters for the above mentior
   The model was later discarded due to a loss in performance.

   - The accuracy of classifiers was lower by 1.6% on the validation set and thus, the origir

Methodology and Evaluation -

For all approaches that involve classification, data duplication was performed for the entire dataset to produce negative samples on the dataset for the classifier to be trained.

1. Accuracy was not the only metric used for evaluation. Just like in the homeworks, Balanced Error Rate (BER) was also used for evaluation purposes.
2. In this scenario, both the BER and accuracy were the best for a similarity threshold of 0.04 and popularity threshold = 80 (number of users who played the same game as the query game)
3. There's a chance of overfit with this model as these are customized values for this dataset. There's a good chance that this model doesn't perform well on another dataset. So, I used a classifier to train the features obtained from this similarity, popularity approach. It didn't give better performance and thus discarded.

The model that's finally submitted is the one that's providing better performance on leaderboard. The strong baseline model with thresholds empirically computed on the validation and test sets.

The biggest learning is the principle of Occam's razor (simplest explanation is often the better performing one)

The code for the same has been provided below.

0.0.3 Task : Time played prediction

Different approaches -

1. Time played $= \alpha + \beta_u + \beta_i$
2. Time played $= \alpha + \beta_u + \beta_i + \gamma_u.\gamma_i$

Data preparation and pre-processing -

- 80% and 20 % for this example for both the models (140000 training and 35000 validation samples)

Methodology and evaluation -

- Implemented a customized version of early stopping for the complete latent factor model to avoid overfitting
- Used 5 fold cross validation for estimation of hyperparameters.
- Grid search was used (and was very runtime consuming) for deciding on the hyper-parameters for the model.
- Evaluated models based on MSE of validation set.

Results and Conclusion -

- Simulated a customized version of mini-batch gradient descent with 5 fold cross validation to prevent overfitting of parameters for the training set.
  - Stop SGD after 5 iterations and check whether the following condition is satisfied -
    - (prev MSE - current MSE > 0) (prev MSE and current MSE are values on the validation set)
    - Break out of the loop if it's not satisfied. This is very similar to the early stopping criterion that's commonly used to avoid overfitting.
    - Choose the best model that results in the minimum value of MSE.
    - For this problem, the following values were obtained for the complete latent factor model
      - Number of latent factors = 3
      - Value of regression coefficient (λ) = 10 / 140000
      - Overall MSE on test set = 3.0985
- Achieved a slightly worse performant model (MSE = 3.098) than the strong baseline set by the first model (3.078).
- But, considering the fact that there are more variables whose values impact the performance of the model, I choose to proceed with the simpler model because it has fewer variables and the problem is a convex optimization problem and gradient descent provides the optimal solution to that problem given those values.

- Also, the complete latent factor model has a lot of chances of variance (depending on the initialized values of γ_u and γ_i, values of K, Gradient descent step size)
- Similar to the above approach, I am using Occam's razor and continuing to use the simpler version of the latent factor model.

[ ]: