CSE 258 – Recommender systems, WI 2017

ASSIGNMENT 1, 02-28-2017

SRINATH NARAYANAN

A53213478

Kaggle username - "vnnsrk"

Email: srn001@ucsd.edu

Helpfulness prediction –

In this task we are given a training data set of 200,000 amazon reviews. We are needed to train a regressor model that predicts how helpful a given review will be for other users, based on the reviews the user has given, and that item has received. Used the first 100000 for training and the rest for validation.

Model –

I used a simple linear regressor model and selected feature elements by trial and error. I tried to find linearly dependent features and removed them from the model parameters. I tried to use PCA to reduce the feature matrix size, but it yielded worse results hence I skipped it.

Model used: $nHelpful = \theta_0 + \sum_{i=n}^{n} \theta_i * [feature_i]$; $y = \theta * X$

Features used :

1. Price – if present in review, else average price
2. ReviewText –
    a. Number of fully capitalized words
    b. Length of text
    c. Number of sentences
    d. Number of exclamations
    e. Number of punctuations
    f. Number of question marks
3. Summary – Length, number of punctuations and capitalized words
4. Unix Time
5. outOf value
6. rating for the item
7. category length
8. RACU / RACI – Reviewer tendency to get more nHelpful per item
9. OOCU / OOCI – Reviewer average outOf per item
10. RCU / RCI – ratings given per user/item pair

I used sklearn.linear_model.LinearRegression model under scikit-learn to train the model with a bias term. Surprisingly elastic regression, ridge regression, lasso regression, SVM regressor and random forests gave worse results and hence I stuck with the most basic model and worked on improving the feature quality.

Final MAE - 0.17771 (private test data)
Final MAE on public test data - 0.15971

Rating prediction –

For the rating prediction I used the simple latent factor model after pre-filtered the training data. I used the test data from test_helpful.json as it provided more rich data and it had other distinct review tuples containing new users and items than the train.json.gz file. Hence I appended the test data from helpfulness prediction with the training data set to make my data more robust and enriched it.

I used the simple latent factor model : Rating = $\alpha + \beta_{user} + \beta_{item} + \gamma_{user} \cdot \gamma_{item} \cdot$

$\alpha -$ Global average rating
β_{user} – Inclination of user to give better rating
β_{item} – Inclination of item to receive better rating
$\gamma_{user} \cdot \gamma_{item}$ – The latent factor term.

I used a simple MAE regularizer, as trying an elastic regularizer yielded worse results. Surprisingly MAE regularizer worked better than MSE regularizer and I used that.

Regularizer - $\lambda * [\ |\beta_{user}|^2 + |\beta_{item}|^2 + |\gamma_{user}|^2 + |\gamma_{item}|^2]$

The model was trained on the whole test_helpful data and half of the training data. I used the vanilla gradient descent model for optimization. It converged pretty fast and was quite accurate. The hyper-parameter values I used were :

$$\lambda = 5.8, \qquad \kappa = 0.1$$

It converged in nearly 112 iterations, with a good mse error rate.

The final result was rounded off to the nearest 0.5 digit, since I believed that will decrease the probability of making rounding error. It predicted = 4.3, it is more likely to be a 4, and if it is 4.9 it is more likely to be a 5, so rounding helps. The value of λ was critical to my approach, and I selected the value by trail and error.
The model provided good results for λ ranging from 5 to 6.5, of which 5.8 yielded the best results.

MSE (Private data) : 1.08339
MSE (Public data) : 1.13183

The model has clearly favoured the test_helpful data as adding those reviews while training yielded a much better result than by underfitting the current model with only train.json reviews.

Same kaggle username : vnnsrk