Assignment 1

Wenjun Zhang

A53218995

Task 1 Helpfulness prediction

Description:

This task is to predict whether a user's review of an item will be considered helpful. Accuracy will be measured in terms of the total absolute error, i.e., you are penalized one according to the difference |nHelpful – prediction|, where 'nHelpful' is the number of helpful votes the review actually received, and 'prediction' is the prediction of this quantity.

Solution:

Fitting the 'nHelpful' variable directly may not make sense, since its scale depends on the total number of votes received. Instead, let's try to fit $\frac{\text{nHelpful}}{\text{outOf}}$ (which ranges between 0 and 1).

We design a model based on **Logistic Regression** for this task. When $\frac{\text{nHelpful}}{\text{outOf}}$ is greater than 0.75, we consider the review is 'helpful', and when $\frac{\text{nHelpful}}{\text{outOf}}$ is equal or lower than 0.75, we consider the review is 'not helpful'.

We choose the data from the first half of 'train.json.gz' whose outOf is greater than 0 as the training data and compute their $\frac{\text{nHelpful}}{\text{outOf}}$'s. Then we determine the training labels y_i to be:

$$y_i = \begin{cases} 1, & \text{if } \dfrac{\text{nHelpful}}{\text{outOf}} > 0.75 \\ 0, & \text{if } \dfrac{\text{nHelpful}}{\text{outOf}} \leq 0.75 \end{cases}$$

For the features of the model, we choose four features in the data, which are:

1. the number of words in review
2. review's rating in star
3. review's date from today in days
4. number of the item's categories in the review

So the feature vector will be:

$$X_i = [1, \text{\# words in review}, \text{\# ratings}, \text{\# days from review'date to today}, \text{\# categories}]$$

A sigmoid function is defined as below:

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Using the method in Logistic Regression, $X_i \cdot \theta$ should be maximized when y_i is positive and minimized when y_i is negative. Combining the regularizer, we define:

$$l_\theta(y|X) = \sum_i -\log\left(1 + e^{-X_i \cdot \theta}\right) + \sum_{y_i=0} -X_i \cdot \theta - \lambda \, || \, \theta \, ||_2^2$$

Then the derivative will be:

$$\frac{dl}{d\,\theta_k} = \sum_i X_{ik}(1 - \sigma(X_i \cdot \theta)) + \sum_{y_i=0} -X_{ik} - \lambda\, 2\theta_k$$

Using the formula above by setting $\lambda = 1$, we solve using gradient ascent until θ finally converge.

From the (user, item) pairs contained in 'pairs_Helpful.txt', we obtain the feature vectors X from 'test_Helpful.json.gz', then using the equation below we obtain the $y_{prediction}$ data.

$$y_{prediction} = X \cdot \theta$$

$y_{prediction}$ is the confidence we get to say if the review is 'helpful'. We first find the maximum and minimum confidence in the $y_{prediction}$ data. And call them y_{max} and y_{min}. Clearly $y_{max} > 0$ and $y_{min} < 0$.

Next, we predict the $\frac{\text{nHelpful}}{\text{outOf}}$ using the formula below:

$$\frac{\text{nHelpful}}{\text{outOf}} = \begin{cases} 0.75 + 0.25 * \dfrac{y_{prediction}}{y_{max}}, & if\ y_{prediction} > 0 \\ 0.75 - 0.75 * \dfrac{y_{prediction}}{y_{min}}, & if\ y_{prediction} \leq 0 \end{cases}$$

Finally, we can predict 'nHelpful' by multiplying the predicted $\frac{\text{nHelpful}}{\text{outOf}}$ with outOf. To be more precise, we round the 'nHelpful' to interger.

Result:

Since the statistic $X \cdot \theta$ is the confidence that the review is considered 'helpful', it's reasonable to predict $\frac{\text{nHelpful}}{\text{outOf}}$ from the confidence by putting some weights to it.

The result score in Kaggle is 0.16543.

Task 2 Ratings prediction

Description:

This task is to predict people's star ratings as accurately as possible, for those (user, item) pairs in 'pairs_Rating.txt'. Accuracy will be measured in terms of the (root) mean-squared error (RMSE).

Solution:

We choose a **Latent factor model** for this task. In general, we fit a predictor of the form

$$\mathrm{rating(u, i)} = \alpha + \beta_u + \beta_i$$

by fitting the mean and the two bias terms using a regularization parameter of $\lambda = 1$.

In this model, β_u is the bias that how much does this user tend to rate things above the mean, and β_i is the bias that does this item tend to receive higher ratings than others. It's easy to see that this is a linear model, and the optimization problem is:

$$\arg\min_{\alpha,\beta} \sum_{u,i} (\alpha + \beta_u + \beta_i - R_{u,i})^2 + \lambda \left[\sum_u \beta_u^{\ 2} + \sum_i \beta_i^{\ 2}\right]$$

This equation is jointly convex in β_u, β_i and can be solved by iteratively removing the mean and solving for β. We differentiate the equation and obtain the iterative procedure to be as below:

$$\alpha = \frac{\sum_{u,i \in \mathrm{train}} (R_{u,i} - (\beta_u + \beta_i))}{N_{\mathrm{train}}}$$

$$\beta_u = \frac{\sum_{i \in I_u} R_{u,i} - (\alpha + \beta_i))}{\lambda + |I_u|}$$

$$\beta_i = \frac{\sum_{u \in U_i} R_{u,i} - (\alpha + \beta_u))}{\lambda + |U_i|}$$

To find more precise $\left(\alpha, \beta_u, \beta_i\right)$, we use the whole 200,000 reviews as the training data. Using $\lambda = 1$, we do this iteration one at a time until the $(\alpha, \beta_u, \beta_i)$ finally converge.

Finally, we can predict the ratings using $\mathrm{rating(u, i)} = \alpha + \beta_u + \beta_i$ and round the predicted ratings to one decimal place.

Result:

Though we're fitting a function that treats users and items independently, this model actually looks good and works well in the test data.

The result score in Kaggle is 1.13516.