Terrance Taubes

2017951160

Data Science Homework 4

2017/5/20

Supplementary Document

Kaggle Username: **Terry\_Taubes**

My source code has been built and separated into several modules in order to deal with preprocessing the massive amounts of data. The preprocessing takes place in modules ‘*preprocess\_business.py*’, ‘*preprocess\_review.py*’, and ‘*build\_user\_dict.py*’. In my preprocessing, I decided to use dictionaries as the data type to store the processed information how I want, as I have experimented and found dictionaries to have quicker access to values over DataFrames and thus has significantly sped up my preprocessing and prediction time.

Preprocess\_business builds a dictionary ‘X\_dict’ that takes the form:

*X\_dict = {* ***‘business\_id’*** *: [ feature values ] }*

‘business\_id’ being a business’s ID number as a key, and its corresponding array of feature values. I created my entire list of features by finding all unique values in the attributes column of business.csv. The value for each feature of a business will either be 1 or 0, corresponding to whether or not the particular business had the particular attribute and if it was marked as True. This is a similar method as to creating feature vectors for email spam. This dictionary was then written using the json library to *X\_dict.txt* to later be retrieved for further use.

Preprocess\_review builds a dictionary ‘review\_dict’ that takes the form:

*review\_dict = {* ***‘user\_id’*** *: { ‘reviews’: { ‘business\_id’ : stars }, ‘avg’: float } }*

‘user\_id’ being a key for review\_dict, ‘reviews’ itself being a dictionary with its value being the pairs of ‘business\_id’s and corresponding ratings that the user has given each business. ‘avg’ corresponds to the calculated average of the ratings of the businesses the current ‘user\_id’ has given. This average is then used to normalize the values in { ‘business\_id’ : stars }. This dictionary ‘review\_dict’ is saved in *review\_dict.txt* to be used later.

Build\_user\_dict continues to work on review\_dict by adding keys ‘bus\_matrix’, which contains an vector of the businesses the user has reviewed, and ‘rate\_matrix’ which contains a vector of the corresponding ratings the user has given. The dictionary is then re-saved into *‘user\_dict.txt’*.

The training and predictions of the model occur in the file ‘*predict\_functions.py*’. Here we load in *user\_dict.txt*, *X\_dict.txt*, and *sample.csv*. The functions included are *reshape*, which unrolls a vector containing both theta and X values into two separate vectors, *reg\_cost\_function*, which is my regularized cost function, and *reg\_gradient*, which is my regularized gradient descent function. All theta values of each user are initialized to an array of small random values. We use the *fmin\_cg* function in the Scipy.optimize library to perform the iterations of training to compute the optimal values for theta and X. We take a similar approach to ‘online learning’ by taking in a single user at a time, and then training the parameters for that user and then using the parameters to then predict the value for the given ‘user-id\_business-id’ pair as to keep run-time optimal while dealing with such massive data. The user-business pair and rating are then appended to the DataFrame ‘*predict\_csv*’. At the end of the predictions, ‘*predict\_csv*’ is then saved to ‘predict.csv’ which is my submission.