Task 3 Report

Joshua Crain
DATA 471
June 5th, 2025

1 Task

This report is on task 3, Predicting the rating a user gives a product based on the term-frequency vector of their review of the product. This is a regression task. The lead group member who worked on this task is Joshua Crain.

2 Methods

The methods used for this task were linear regression through a neural network and SVM implementation through the help of the libraries PyTorch and Scikit Learn. The SVM implementation was through the use of the LinearSVR method. Due to the training and dev data size (being over 3 billion data points), dimensionality reduction was used, with the number of components to reduce to being chosen based on the number of components vs. cumulative explained variance. The amount of components chosen was selected based on what amount was close to a 1.0 value. This was found by arbitrarily picking an amount of components for the training set, then using TruncatedSVD. A plot was made afterwards using matplotlib.pyplot to determine what amount comes close enough to 1.0, and that was used for all future tests with the neural network. For the LinearSVR implementations, the default values were used (epsilon=0.0, tol=1e-4, C=1.0, loss='epsilon_insensitive', fit_intercept=True, intercept_scaling=1.0, dual="auto", verbose=0, random_state=None, max_iter=1000). The neural network implementation contained 3 relu-activated hidden layers, utilized a learning rate of 0.02, and ran for 2000 updates.

3 Submission Model Details

The best model I have come up with was using LinearSVR, as I ran into issues calculating R^2 values for the neural network implementation. In order to make and run this model, the data, which was in sparseX format, was parsed and converted into a matrix of the appropriate dimensions and values. This matrix is returned as a coo_matrix, and then dimensionality reduction was performed using TruncatedSVD, where I found that 6000 components seemed appropriate based on figure 1. Then, the LinearSVR method from Scikit Learn was used with C=1.0 and another time with C=0.5, the data was fit to the model, then predictions ran for the training and development sets. Afterwards, a graph of the training set predictions and expected values were plotted in figures 2 and 3, for C=1.0 and C=0.5 respectively.

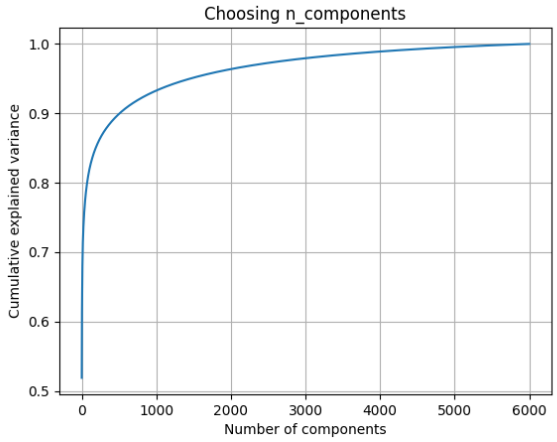

Figure 1: Graph of component number vs. cumulative explained variance to decide on using 6000 components.

4 Results

R^2 metrics for the neural network implementation were unsuccessful, however, mean squared error was able to be gathered despite being insignificant as a performance metric. When running for 2000 updates, an error of 0.0711379 was gathered for the training set, and 2.1676576 for development set error. For the LinearSVR performance, R^2 values of 0.374222 for the training set and -0.445172 was obtained for the development set when C=0.5, and 0.355853 for the training set and -0.478020 for the development set when C=1.0.

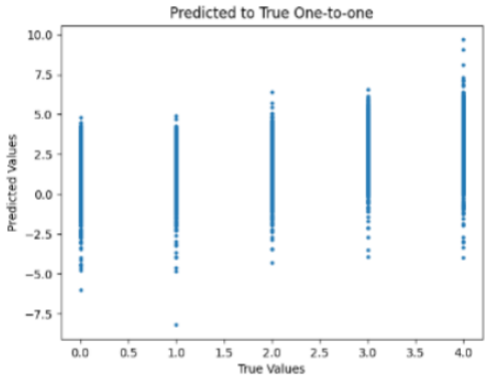

Figure 2: LinearSVR with C=1.0

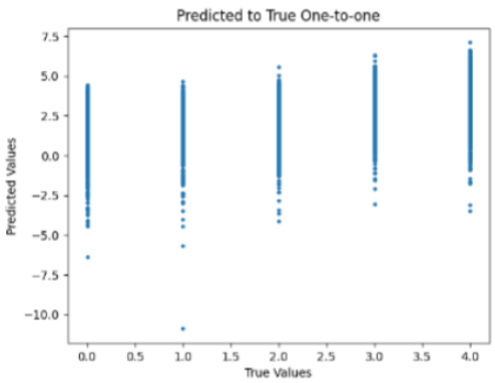

Figure 3: LinearSVR with C=0.5

5 Distribution of work

Andy Ngo contributed through his method for parsing the sparse data into it's appropriate format for the training and dev sets. Jenny Sims contributed with the structure of the neural network implementation and some bug fixing, and Io Paul for some bug fixing as well.