Udacity Machine Learning Engineer

YongKyung Oh

Project Proposal

Domain Background: Sentiment classification, detecting if a piece of text is positive or negative, is a common NLP task that is useful for understanding feedback in product reviews, user's opinions, etc. Sentiment can be expressed in natural language in both trivial and non-trivial ways. Sometimes sentiment lexicon words such as "Good" and "Bad" are explicitly mentioned in the text and make the task easier. However, very often it could be much more subtle than that.

Problem Statement: In this project, I will build two sentiment classifiers based on naïve Bayes and neural networks. I will use the movie review data and use the percentage of correct predictions as evaluation metric.

Datsets and Inputs: dev_text.txt contains reviews, one per line, and dev_label.txt contains their labels. I use these files for training and testing models, by splitting them. heldout_text.txt contains reviews that the models will be evaluated on the unknown label data. In this project, I will just show the performance of model on the dev data.

Solution Statement: I will build a RNN model for the sentiment classification. Specifically, I will build model using pytorch and torchtext. Pretrained word embedding will be used.

Benchmark model: There are two different benchmark models. First, naïve Bayes model will be the baseline model for comparison. Second, Word-to-Vector model will be compared with pretrained embedding model (Glove6B).

Evaluation metrics: For simplicity, the accuracy on the text and label will be the metric for model comparison.

Project Design

1. Preprocess: Train data is consist with 1000 pos-labeled review and 1000 neg-labeled review. Through preprocess, I divided sentence into tokens and remove uninformed text using nltk. Also, I removed all numeric text. After preprocess, dataset only contain useful tokens. Average token per sentence is 216.31 where min is 15 and max is 1654.

2. Resampling: The number of train data is not big. So, I implemented permutation resampling to select more samples. Through this step, I randomly select the same number of samples with original dev data. So, the total number of train data is 4000 and the train/validation ratio is different in each methods.

3. Strategy for data usage: After data clean-up, I split the train data into train set and validation set. I selected 20% of data. So, 1600 for train set and 400 for validation set. This data is used to evaluate the model. After that, I conducted random sampling for train set. Train set is not enough for each class. So, I shuffle the each class set and select random samples as follow. Through this process, I can get 3600 train data with 1800 pos-labled samples and 1800 neg-labeled samples. I use the validation for 400 samples.

4-1. Naïve Bayes model: Most of all, I build a vocabulary dictionary from both classes. I selected top 2000 words from each classes and use the unique 2500 words as dictionary. Main model is multinomial NB which is suggested by J. Rennie et al. (2003. Also, I applied 10-fold validation to select the best model.

4-2. RNN model: Main model is bi-directional LSTM, which is appropriate for the sequential data such as text. Model only has two hidden layer with 100 nodes and each layer represent forward and backward characteristics. Output node is 2 and I use the logit for the output value. Rather than other activation function, I used logit and cross entropy to evaluate loss.