COMP4901I Assignment 1 Twitter Sentiment Analysis For Sentence-Level Text

Cheng Chi Fung
cfchengac@connect.ust.hk

Abstract

In this assignment, we used supervising learning to train an Artificial Intelligence to perform sentiment analysis for sentence-level text. We used logistic regression to estimate the probability that de-scribes the likelihood of a positve sentence. Since its accuracy was not good enough. We performed serveral data pre-processing procedures such as Bi-gram and SGD with momentum to further improve its convergence.

1 Data Pre-Processing

1.1 Data Cleaning

For this assignment, we first cleaned the data by **expanding all the contradictions**. For example, replace "it's" by "it is". And followed by **removing all the integers and punucations**. We found out that after performing these kinds of data cleaning , the development accuracy had rised. Finally, the data were tokenized by white space.

1.2 Feature Extraction

To contruct the data matrix, we had tried both **Bi-gram** and **Uni-gram** model. The performace (development accuracy) of using Bi-gram and Uni-gram model for feature contruction were the following.

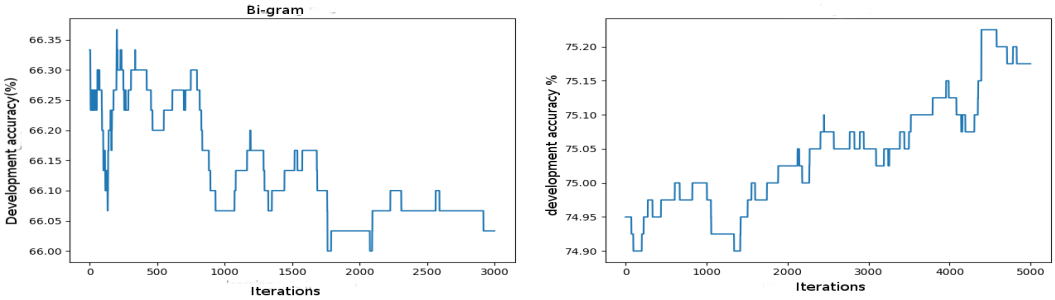

Figure 1: The Development accuracy of using Bi-gram and Uni-gram

We found that the development accuracy of using Uni-gram was higher than using Bi-gram model. Thus, Uni-gram performed better on this dataset. This might due to the increasing of training features when the n-gram length is increased using the Bi-gram model caused the model **overfit** the data. Moreover, extra features when using Bi-gram slower the model's convergence.

1.3 Feature Normalization

The data matrix was normalized by the following formula so that each dimension of the feature was **zero mean** and **standard deviation** was 1.

$$data - data.mean(axis = 0)/data.std(axis = 0)$$

We found out that training with normalization had a little bit enhancement of in terms of the development accuracy when it is compared to the training without normalization. This might because the normalization bring different dimension in the data matrix into the same scale which might contribute to **faster convergence**.

Table 1: Training Results before and after normalization with same parameters

Name	Normalized	Without Normalized
Development Accuracy	~ 73.4	~ 72.5
Training Accuracy	~ 89.5	~ 87.5

2 Prediction Model

2.1 Logistic Regression

In this assignment, we adopted **Logistic Regression** as the prediction model. The formula we used in our prediction model were the followings.

Formula of logistic regression $f(z) = \frac{1}{1+e^{-z}}$

Formula of cost function $L(x) = \frac{1}{m}\sum_{i=1}^{m}[-y_i log(f_\beta(x_i) - (1-y_i)log(1-f_\beta(x_i))$

2.2 Batch Gradient Descent

To minimize the cost function, we used **Batch Gradient Descent** (BGD) to iterately update the weights using the following rules,

$$\theta^* = \theta - \alpha\nabla L(x)$$

where θ^* is the new weights, θ is the old weights and $\nabla L(x)$ is the cost function

However, to faster the convergence, we tried to adopt **Momentum** in our Batch Gradient Descent. Following was the updated rule of BGD with momentum that we implmented in our program.

$$v^* = v - \alpha\nabla L(x)$$

$$\theta^* = \theta + v^*$$

where v^* is the new momentum, v is the old momentum

We had compared the performance of using BGD and BGD with momentum.

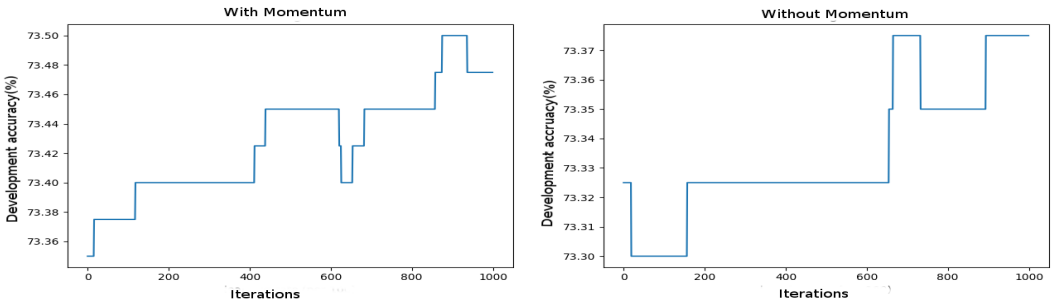

Figure 2: Development accuracy of BGD with and without momentum

The results shown that using BGD with momentum, the model converged faster and gave slightly better results in terms of the development accuracy. This might because after adding the momentum term in the objective funtion, it increased the size of the steps taken towards the minimum by trying to **jump from a local minima**.

3 Training Results, Phenomenons and Problems

3.1 HyperParameters Tuning and Training Results

To obtain the best parameters, we implemented an extra program tester.py to perform hyperparameter optimization using **grid search**. And after serveral times of trainings, the following was the best parameters we had obtained.

Table 2: Best HyperParameters

Name	Size
Learning Rate	$\sim$0.001
Lambda	$\sim$0.95
Vocabulary Size	18000

After serveral times of training, we found out that, for **higher learning rate**, model can converge faster to local minima. However, it might easily get overshoot. Therefore, sometimes, the accuracy would suddenly dropped when using higher learning rate. For **lower learning rate**, although the convergence might be very slow, the accuracy increased steadily without sudden dropped.

And, we used the best hyperparameters to train the model in 5000 iterations which produced the following results.

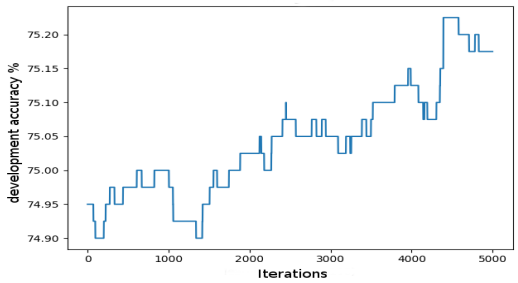

Figure 3: Development accuracy with best hyper-parameters

Table 3: Training Results

Name	Percentage
Highest Development Accuracy	~ 75
Highest Training Accuracy	~ 96

3.2 Training Phenomenons and Tips

Automatically fine-tune the hyperparamters using program: Rather than using human hand to change the hyper-parameters every time. It is better to write a program to automatically train the model serveral times with different hyperparameters. And find out the best hyperparameters with highest accuracy.

Perform data pre processing according to data's features: Different dataset may suit for different data pre procssing technique. We had tried stopwords filtering but we found out that the development accuracy was lower after we did it.

Early stopping: When the accuracy has a trends to drop, it is better the stop the traininig earlier.

3.3 Wrong Predictions

Some sentences in the training dataset were difficult to predict by our model because of various reasons. Here ,we list out some with examples.

Having non-English words or uncommon words: e.g. "tak at to prezijes" (neg)

Too ambiguous for its sentiment: e.g. "i didn't say it was correct, just what i thought" (pos), "someone come out with me" (neg), "ffmpeg hall of shame URL" (pos), "going to class." (neg), "here too" (neg)

Wrong original label: e.g. "i love carbs!! ...too much" (neg), "URL - i love you, buck" (neg)

Bad formating sentence: e.g "i am soo t i r e d in every sense of the w o r d need a tune to pick me up" (neg), "wow, l-o-v-e your new geeky avatar" (pos)