Deep Learning for Natural Language Processing Assignment 1

ZHOU, Xinrui | email: xzhouaz@connect.ust.hk | SID: 20493318

1 Data Preprocessing and Analysis

1.1 Data Statistics

Data statistic before cleaning is as follows:

```
{'file': './twitter-sentiment.csv', 'clean': False, 'max_vocab': -1}
Number of words:  266282
Max sentence length:  34
Number of sentences:  20000
Number of vocabulary:  34389
Top 10 most frequently words ['@', 'i', 'to', 'the', 'a', 'and', 'my', 'is', 'you', 'for']
```

1.2 Data Cleaning

Main methods for data cleaning:

1) Convert all characters into lowercase
2) Expand words (eg. “I’m” converted to “I am")
3) Removing the word “url”
4) Removing all irrelevant punctuations (!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~)
5) Removing all non-English words (such as 锟、铐 in the dataset)
6) Tokenization
7) Removing Stopwords
8) Stemming
9) Convert the list of tokens into back to the string

Data statistic before cleaning is as follows:

```
Number of words:  138601
Max sentence length:  26
Number of sentences:  20000
Number of vocabulary:  18167
Top 10 most frequently words ['go', 'get', 'work', 'day', 'good', 'like', 'today', 'love', 'time', 'want']
```

2 Logistic Regression

2.1 Formula Derivation

The probability to generate the whole dataset D is:

$$L(W,b) = \prod P_i \,, P_i = \begin{cases} f(x_i), & C_i == C_{pos} \\ 1 - f(x_i), & C_i == C_{neg} \end{cases}$$

Let’s assume C_{pos}= 1 and C_{neg}= 0. Then P_i *and* $L(W,b)$ can be expressed as

$$P_i = f(x_i)^{C_i} * \left(1 - f(x_i)\right)^{1-C_i} \qquad\qquad L(W,b) = \prod \left(f(x_i)^{C_i} * (1 - f(x_i))^{1-C_i}\right)$$

We want to find the parameter W*, b* that can maximize the probability, which is equivalent to minimize the $-logL(W,b)$. First, we derive $logP_i$:

$$logP_i = C_i * \log(f(x_i)) + (1 - C_i) * log(1 - f(x_i))$$

Thus, Loss = $-logL(W,b) = -\log(\prod P_i) = -\sum[C_i * \log(f(x_i)) + (1 - C_i) * log(1 - f(x_i))]$

Now we have our loss, and we need to minimize it. Compute the derivative of $-logL(W,b)$ over W:

$$\frac{\partial Loss}{\partial W} = \frac{\partial}{\partial W} - \sum[C_i * \log(\sigma(Wx_i + b)) + (1 - C_i) * log(1 - \sigma(Wx_i + b))]$$

$$= -\sum\frac{\partial}{\partial W}[C_i * \log(\sigma(Wx_i + b)) + (1 - C_i) * log(1 - \sigma(Wx_i + b))$$

$$= -\sum[\frac{\partial}{\partial W}C_i * \log(\sigma(Wx_i + b)) + \frac{\partial}{\partial W}(1 - C_i) * log(1 - \sigma(Wx_i + b))$$

Next, using the chain rule, and relying on the derivative of log:

$$\frac{\partial Loss}{\partial W} = \sum - \frac{C_i}{\sigma(Wx_i + b)}\frac{\partial}{\partial W}\sigma(Wx_i + b) - \frac{1 - C_i}{1 - \sigma(Wx_i + b)}\frac{\partial}{\partial W}(1 - \sigma(Wx_i + b))$$

Rearranging terms:

$$\frac{\partial Loss}{\partial W} = \sum -[\frac{C_i}{\sigma(Wx_i + b)} - \frac{1 - C_i}{1 - \sigma(Wx_i + b)}]\frac{\partial}{\partial W}\sigma(Wx_i + b)$$

Plug in the derivative for the sigmoid function:

$$\frac{\partial Loss}{\partial W} = \sum -\left[\frac{C_i}{\sigma(Wx_i + b)} - \frac{1 - C_i}{1 - \sigma(Wx_i + b)}\right]\sigma(Wx_i + b)(1 - \sigma(Wx_i + b))\frac{\partial(Wx_i + b)}{\partial W}$$

$$= \sum -\left[\frac{C_i - \sigma(Wx_i + b)}{\sigma(Wx_i + b)(1 - \sigma(Wx_i + b))}\right](Wx_i + b)(1 - \sigma(Wx_i + b))x_i$$

$$= \sum -(C_i - \sigma(Wx_i + b))x_i$$

$$= \sum(\sigma(Wx_i + b) - C_i)x_i$$

Then compute the derivative of $-logL(W,b)$ over b:

$$\frac{\partial Loss}{\partial b} = \frac{\partial}{\partial b} - \sum[C_i * \log(\sigma(Wx_i + b)) + (1 - C_i) * log(1 - \sigma(Wx_i + b))]$$

$$= -\sum[\frac{\partial}{\partial b}C_i * \log(\sigma(Wx_i + b)) + \frac{\partial}{\partial b}(1 - C_i) * log(1 - \sigma(Wx_i + b))$$

$$= \sum - \frac{C_i}{\sigma(Wx_i+b)}\frac{\partial}{\partial b}\sigma(Wx_i + b) - \frac{1-C_i}{1-\sigma(Wx_i+b)}\frac{\partial}{\partial b}(1 - \sigma(Wx_i + b))$$

$$= \sum - \left[\frac{C_i}{\sigma(Wx_i+b)} - \frac{1-C_i}{1-\sigma(Wx_i+b)}\right]\frac{\partial}{\partial b}\sigma(Wx_i + b)$$

$$= \sum - \left[\frac{C_i-\sigma(Wx_i+b)}{\sigma(Wx_i+b)(1-\sigma(Wx_i+b))}\right](Wx_i + b)(1 - \sigma(Wx_i + b))$$

$$= \sum(\sigma(Wx_i + b) - C_i)$$

2.2 Implementation

1) What is the effect of the learning rate? What is the best learning rate you found?

Learning rate	0.0003	0.001	0.003	0.01	0.03	0.1	0.3	1
Best dev acc	71.125	71.175	71.2	71.2	71.2	71.25	71.225	71.225

When learning rate is too small, the accuracy improves very slowly, while when learning rate is too large, the accuracy will decrease in the later stages of training. The best learning rate is around 0.1 according to my trails.

2) What is the best accuracy you can get on the training set? how about on the development set?

The best accuracy on the training set (lr = 1) is 92.2188%.

The best accuracy on the development set (lr = 0.1) is 71.25%, while best accuracy on the training set (lr = 0.1) is 90.5625%.

3) Describe whether the data normalization helps. Why or why not?

Learning rate		0.0003	0.001	0.003	0.01	0.03	0.1	0.3	1
Best dev acc%	w/ normalization (200 iterations)	71.125	71.175	71.2	71.2	71.2	71.25	71.225	71.225
	w/o normalization (200 iterations, not converge)	60.2	60.4	60.7	61.775	64.5	69.275	70.05	71.325
	w/o normalization (500 iterations)	67.575	67.6	68.125	69.05	69.05	69.025	70.625	71.7

From the above data we can see, the data normalization will improve the final accuracy and make the model converge much faster. For data without normalization, there will be columns that have a larger scale than others. Since $dw = (\sigma(Wx_i + b) - C_i)x_i$, some weights will update much lower than others, so it will converge slower. It also improves the accuracy by mapping data to the same scale and avoid dominating features. However, it will not affect much since the model will assign smaller weights for the data with large scale when converges.

4) List at least 10 sentences in the dataset that your model gives you a wrong prediction (sentence should be positive but the prediction is negative, and vice versa). Try to explain why.

ID	Label	Prediction	Sentences
75	pos	neg	my brother is sick too lol. quite weird
285	pos	neg	today i performed for the first and last time at the grebel talent show. 4 exams left until undergrad is over
360	pos	neg	rocking out to no doubt and gonna start up some more neverwinter nights
444	pos	neg	i wonder how long i can stay up until i pass out and wake up again at 6
2803	pos	neg	im soo tired. but i have my hat now, graduation hat mom started crying when i had it on love u mom, and my tiger
3046	pos	neg	@ sweet... must go home shortly and make a death ray
1637	pos	neg	@ good job! yeah i feel you. working out can be annoying sometimes but its definitely worth it in the end
12846	neg	pos	@ i love how i can - could - easily read and tweed while working. with twitter.com i'm reloading all the time
12931	neg	pos	marley and me. i am crying like a kid, but a good movie. have a good night people
13057	neg	pos	@ nicee... we ran out of filter coffee art work , not good , forgot my bread to make toast bad start to the day
13077	neg	pos	@ hope so - i don't have an iphone (@lauraoliver)
13136	neg	pos	i can't enjoy the weather

Potential reasons for prediction errors:

1) The model takes the tokenized word as input, and doesn't consider any context meaning

2) When cleaning the data, we deleted the word like "not" that will critically affect the meaning of the sentence. For example, in sentence 13136, the model will take in word "enjoy" instead of understanding "can't enjoy", thus affecting the prediction.

3) There are too many noises in the data even after cleaning that will affect the prediction. e.g. Many spelling errors

4) Some sentences use turning words like "but", "despite", and are expressing positive emotions using negative words,

or vice versa (e.g., sarcasm), that could not be comprehended by the model.

3 Bonus

3.1 Bi-gram feature

Implemented bigram with BoW feature in *preprocess_bigram.py*. The result is as shown below:

Learning rate	Iterations	Best dev accuracy	
0.1	200	BoW	Bi-gram with BoW
		72.15 %	70.325 %

From the data above, we can observe the bi-gram feature doesn't improve the validation accuracy. The underlying cause may be there are too many noises in the dataset, so many critical bi-gram features only appear very few times and affect the accuracy.

3.2 Gradient descent with momentum

Implemented bigram with BoW feature in *logistic_regression_momentum.py*.

By calculating and adding the momentum term to the gradient, the gradient with momentum is averaging previous gradients. The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, the model gains faster convergence and reduced oscillation.

3.3 Training phenomenon and tips

Phenomenon:

1. The cost and accuracy improve rapidly in the first few iterations, and the change rate get slower and slower in later iterations.
2. The accuracy will oscillate when learning rate is too large
3. Sometimes the dev accuracy is unchanged while dev cost is still decreasing, this may because of the problem nature of a binary classification. e.g. the predictions for each sample is rarely changed, but sigmoid prediction value is more close to 1 or 0, leading to a smaller loss.

Tips:

1. Always pay attention to the shape of each input and output variable
2. If the accuracy and cost are still decreasing obviously at the end if the training → may have not converged and need more iterations
3. Read and dig in raw data carefully before preprocessing to find the most suitable cleaning method