02_sentiment_analysis_logistic_regression_SKlearn

August 22, 2021

1 Logistic Regression

Welcome to week one of this specialization. You will learn about logistic regression. Concretely, you will be implementing logistic regression for sentiment analysis on tweets. Given a tweet, you will decide if it has a positive sentiment or a negative one. Specifically you will:

- Learn how to extract features for logistic regression given some text
- Implement logistic regression from scratch
- Apply logistic regression on a natural language processing task
- Test using your logistic regression
- Perform error analysis

We will be using a data set of tweets. Hopefully you will get more than 99% accuracy.
Run the cell below to load in the packages.

1.1 Import functions and data

[1]:
```
# run this cell to import nltk
import nltk
from os import getcwd
```

[2]:
```
import numpy as np
import pandas as pd
from nltk.corpus import twitter_samples

from utils import process_tweet, build_freqs
```

[3]:
```
# select the set of positive and negative tweets
all_positive_tweets = twitter_samples.strings('positive_tweets.json')
all_negative_tweets = twitter_samples.strings('negative_tweets.json')
```

[4]:
```
# split the data into two pieces, one for training and one for testing␣
 ↪(validation set)
test_pos = all_positive_tweets[4000:]
train_pos = all_positive_tweets[:4000]
test_neg = all_negative_tweets[4000:]
train_neg = all_negative_tweets[:4000]
```

```
train_x = train_pos + train_neg
test_x = test_pos + test_neg
```

[5]:
```
# combine positive and negative labels
train_y = np.append(np.ones((len(train_pos), 1)), np.zeros((len(train_neg),␣
 ↪1)), axis=0)
test_y = np.append(np.ones((len(test_pos), 1)), np.zeros((len(test_neg), 1)),␣
 ↪axis=0)

# Print the shape train and test sets
print("train_y.shape = " + str(train_y.shape))
print("test_y.shape = " + str(test_y.shape))
```

```
train_y.shape = (8000, 1)
test_y.shape = (2000, 1)
```

[6]:
```
# create frequency dictionary
freqs = build_freqs(train_x, train_y)

# check the output
print("type(freqs) = " + str(type(freqs)))
print("len(freqs) = " + str(len(freqs.keys())))
```

```
type(freqs) = <class 'dict'>
len(freqs) = 11340
```

2 Logistic regression from SKlearn

[7]:
```
def extract_features(tweet, freqs):
    word_l = process_tweet(tweet)
    x = np.zeros((1, 3))
    x[0,0] = 1
    for word in word_l:
        x[0,1] += freqs.get((word, 1.0),0)
        x[0,2] += freqs.get((word, 0.0),0)
    return x
```

[8]:
```
# collect the features 'x' and stack them into a matrix 'X'
X = np.zeros((len(train_x), 3))
for i in range(len(train_x)):
    X[i, :]= extract_features(train_x[i], freqs)

# training labels corresponding to X
Y = train_y
Y = Y.flatten()
```

[9]:

```
print ("X.shape", X.shape, ", Y.shape:", Y.shape)
```

```
X.shape (8000, 3) , Y.shape: (8000,)
```

[10]:

```
from sklearn.linear_model import LogisticRegression
```

[11]:

```
model = LogisticRegression()
model.fit(X, Y)
acc_train=model.score(X, Y)
print ("train accuracy:", acc_train)
```

```
train accuracy: 0.990375
```

[12]:

```
acc_test=model.score(X, Y)
```

[13]:

```
theta=model.coef_[0]
theta
```

[13]:

```
array([ 0.49750595,  0.00903714, -0.01025992])
```

[14]:

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))
```

[15]:

```
# UNQ_C4 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def predict_tweet(tweet, freqs, theta):
    x = extract_features(tweet,freqs)
    y_pred = sigmoid(np.dot(x,theta))
    return y_pred
```

[16]:

```
# Run this cell to test your function
for tweet in ['I am happy',
              'I am bad',
              'this movie should have been great.',
              'great', 'great great',
              'great great great',
              'great great great great']:
    print( '%s -> %f' % (tweet, predict_tweet(tweet, freqs, theta)))
```

```
I am happy -> 0.854185
I am bad -> 0.517490
this movie should have been great. -> 0.822561
great -> 0.825575
great great -> 0.931608
great great great -> 0.975126
great great great great -> 0.991214
```

```
[17]: # Feel free to check the sentiment of your own tweet below
      my_tweet = 'I am learning :)'
      predict_tweet(my_tweet, freqs, theta)
```

```
[17]: array([1.])
```

```
[18]: def predict_test_set():
          X_test = np.array([extract_features(tw, freqs) for tw in test_x]).
       ↪reshape(2000, 3)
          Y_test = test_y.flatten()
          Y_pred_test = model.predict(X_test)
          acc_test = (Y_pred_test==Y_test).sum()/len(X_test)
          return acc_test

      acc_test = predict_test_set()
      print (acc_test)
```

```
0.9915
```

```
[19]: def prediction(tweet, model=model):
          print("processed tweet:", process_tweet(tweet))
          X_new = extract_features(tweet, freqs).reshape(1,3)
          Y_out=int(model.predict(X_new)[0])
          sents = {1:'Positive', 0:'Negative'}
          print ("Sentiment:", sents[Y_out])
```

```
[20]: # Feel free to change the tweet below
      my_tweet = 'This is a ridiculously good movie. The plot was okay!'
      prediction(my_tweet)
```

```
processed tweet: ['ridicul', 'good', 'movi', 'plot', 'okay']
Sentiment: Positive
```

```
[21]: # Feel free to change the tweet below
      my_tweet = 'This was a great movie. story was compelling.'
      prediction(my_tweet)
```

```
processed tweet: ['great', 'movi', 'stori', 'compel']
Sentiment: Positive
```

```
[22]: # Feel free to change the tweet below
      my_tweet = 'A great movie.'
      prediction(my_tweet)
```

```
processed tweet: ['great', 'movi']
Sentiment: Positive
```

[24]:

```
# Feel free to change the tweet below
my_tweet = 'This is a ridiculously bright movie.\
            The plot was terrible and I was sad until the ending!'
prediction(my_tweet)
```

```
processed tweet: ['ridicul', 'bright', 'movi', 'plot', 'terribl', 'sad', 'end']
Sentiment: Negative
```

[ ]:

```
```