Twitter Popularity and Machine Learning

Dante Tam, 5/23/2017

Twitter is one of the greatest media in learning to understand the human condition. In this article we investigate how we can leverage Twitter to become popular. I collect tweets, create mathematical models to predict their impact, and finally, interpret these abstract results for real-world, social applications.

First, I used Twitter's API, a web interface between the company and my computer, to collect popular tweets by trending topics. Then, these tweets were converted into vectors, which simply encode the tweet and its statistics such as favorites and retweets.

A linear regression was run. This returns a hyperplane, or a function that can be used to predict the popularity of any tweet.

For the first trial, I manually determined the category and emotion of every tweet. I used four broad categories: sports, culture (pop culture), politics (including anything politically charged), and Twitter (anything related to Twitter culture) — in addition, five emotions: neutrality (including adverts and promotions); anger (or passive-aggressiveness, cynicism, bitter sarcasm); sadness (and grief); happiness (also hopefulness, wonder); and humor (positive sarcasm, parody).

Tweets were grouped into six time blocks, and a popularity score was computed. This is quite important since we already know that great tweets will come from popular users talking about hot, trending topics. This score ensures that we are not biased by already existing popularity.

The results were quite intuitive, and still fun to figure out. Most notably,

- Tweets about sports are likely to be more popular than users themselves. This topic is more "decentralized", and often in the hands of many distinct, but united users.
- Anger does actually spread much faster on the Internet, like a contagion. So does happiness. Sad and neutral tweets were the least popular.
- The best time for stardom is around 2-6 AM, or the evenings. As expected, most people don't tweet in the early morning.
- Users with too many statuses risk losing attention.

The technical equation for the score is defined as

$$popularity\_score = 100000 \cdot \frac{2(retweets) + favorites}{0.01(followers) + friends}$$

while in the linear regression we minimize squared error, i.e.

$$minarg_w ||Xw - y||_2^2$$

The first function is very much like a loss function. A normal loss function can take on certain values to penalize certain types of error. In the case of regression, we can reward or punish certain attributes of the tweets.

Using this knowledge of tweets and popularity, we write the linear regression model in our IPython notebook code:

In [1]:
```
import numpy as np

import sklearn
```

```
from sklearn.preprocessing import normalize
from sklearn import datasets, linear_model

from random import shuffle
```

In [8]:
```
def quoteCustomSplit(text):
    firstIndex, secondIndex = -1,-1
    for i in range(len(text)):
        c_i = text[i]
        c_l = text[i-1] if i > 0 else None
        c_r = text[i+1] if i < len(text) - 1 else None
        if c_i == '"' and c_l != "\\" and firstIndex == -1:
            firstIndex = i
        elif c_i == '"' and c_r == ',' and firstIndex != -1:
            secondIndex = i
            newText = text[0:firstIndex] + text[firstIndex:secondIndex].re
place(",", "") + text[secondIndex:]
    return newText.split(",")

def readTwitterData(fname):
    parsedX = []
    parsedY = []

    with open(fname) as f:
        content = f.readlines()
    content = [x.strip() for x in content]
    contentParsed = [text for text in content if len(text) > 0]

    for line in contentParsed:
        data = quoteCustomSplit(line)
        label = float(data[len(data) - 1])
        parsedY.append(label)

        newPoint = [float(x) for x in data[2:(len(data) - 4)]]

        if data[1] == "N": newPoint = [1,0,0,0,0] + newPoint
        if data[1] == "A": newPoint = [0,1,0,0,0] + newPoint
        if data[1] == "S": newPoint = [0,0,1,0,0] + newPoint
        if data[1] == "H": newPoint = [0,0,0,1,0] + newPoint
        if data[1] == "F": newPoint = [0,0,0,0,1] + newPoint

        if data[0] == "S": newPoint = [1,0,0,0] + newPoint
        if data[0] == "C": newPoint = [0,1,0,0] + newPoint
        if data[0] == "P": newPoint = [0,0,1,0] + newPoint
        if data[0] == "T": newPoint = [0,0,0,1] + newPoint

        parsedX.append(newPoint)

    f.close()

    return parsedX, parsedY
```

In [10]:
```
dataX, dataY = readTwitterData("vectorized_tweets.txt")

dataX = np.array(dataX)
dataY = np.array(dataY)
```

```
dataX = sklearn.preprocessing.normalize(dataX, axis=1)

bestX, bestY = None, None

regr = linear_model.LinearRegression()
regr.fit(dataX, dataY)

np.set_printoptions(suppress=True)

linRegColumns = ["Topic: Sports", "Topic: Culture", "Topic: Politics", "To
pic: Twitter/Misc.",
          "Emotion: Neutral", "Emotion: Angry", "Emotion: Sad", "Emotion:
Happy/Hopeful", "Emotion: Funny/Satirical",
          "TIME2_6","TIME6_10","TIME10_14","TIME14_18","TIME18_22","TIME22
_2",
          "DATE_SUN","DATE_MON","DATE_TUE","DATE_WED","DATE_THU","DATE_FRI
","DATE_SAT",
          "PHOTO","VIDEO","ANIMATED_GIF",
          "LOG10_USER_FAV","LOG10_USER_STATUS_COUNT"]

# The coefficients
print('Coefficients: \n')

for i in range(len(linRegColumns)):
    print(linRegColumns[i] + " -> %.2f" % regr.coef_[i])

print('\n')

# The mean squared error
print("Mean squared error: %f"
      % np.mean((regr.predict(dataX) - dataY) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %f' % regr.score(dataX, dataY))
```

```
Coefficients:

Topic: Sports -> 32.90
Topic: Culture -> 20.60
Topic: Politics -> 22.98
Topic: Twitter/Misc. -> 0.00
Emotion: Neutral -> 11.78
Emotion: Angry -> 20.04
Emotion: Sad -> 10.99
Emotion: Happy/Hopeful -> 16.87
Emotion: Funny/Satirical -> 16.80
TIME2_6 -> 18.05
TIME6_10 -> 7.23
TIME10_14 -> 9.28
TIME14_18 -> 15.45
TIME18_22 -> 11.71
TIME22_2 -> 14.75
DATE_SUN -> 9.24
DATE_MON -> -1.45
DATE_TUE -> 28.30
DATE_WED -> 0.00
DATE_THU -> 14.28
```

```
DATE_FRI -> 7.49
DATE_SAT -> 18.61
PHOTO -> -2.40
VIDEO -> -1.28
ANIMATED_GIF -> 0.19
LOG10_USER_FAV -> -5.14
LOG10_USER_STATUS_COUNT -> -28.08


Mean squared error: 12.636830
Variance score: 0.297129
```

In [ ]:

In the second trial, we automate this process of categorizing tweets. CoreNLP, a Stanford research project in Natural Language Processing, is used to determine the emotion of every tweet. Topic categories are found from the main topic hashtag. We also add more information to the vector.