CMSC 25025 / STAT 37601

Machine Learning and Large Scale Data Analysis

LSD PROJECT 2

Due: Tuesday, May 7, 2013

This is the second of four large scale data projects. In this project, you will work with the Twitter "wishes" dataset. This dataset is made up of tweets from around New Year's Day, 2013.

You will implement stochastic gradient descent on the Twitter dataset to learn a logistic regression model that differentiates between tweets primarily referring to "wishes" and "hopes" and everything else. For more about the study of wishes on Twitter, you may want to check out `http://pages.cs.wisc.edu/~jerryzhu/pub/wish.pdf`. This provides an interesting analysis of how to build a "wish detector" for NLP text using topic models. See also `http://xkcd.com/1086/`.

1. *Generating features* (20 points)

   Your task will be to classify tweets. To do so, you will have to represent tweets in terms of a vector of features (covariates). You are free to use and experiment with any representation or combination of representations that you deem appropriate for achieving low classification error.

   One effective representation is to use membership in a fixed vocabulary. Suppose that $\mathcal{D}$ is a dictionary of words. For a given tweet, you normalize the text, and separate it into space-delimited tokens. For each of the tokens, if it is in the dictionary you have a 1 for the corresponding word in the feature vector, and you ignore it otherwise.

   For example, suppose the tweet is

   `"I wish I had another wish! #silly"`

   and after normalization you process this into the list

   `["i","wish","i","had","another","wish","#silly"]`.

   If `"had"` and `"#silly"` are not in your dictionary, but the other words are, you have three active features

   `{"i","wish","another"}`.

   So, this tweet would have three binary features set to 1, and the rest set to zero. Your model would then assign a probability to class $Y = 1$ as

   $$\mathbb{P}(Y = 1 \,|\, \texttt{"I wish I had another wish! \#silly"}) = \frac{\exp(b + \beta_{\text{i}} + \beta_{\text{wish}} + \beta_{\text{another}})}{1 + \exp(b + \beta_{\text{i}} + \beta_{\text{wish}} + \beta_{\text{another}})}.$$

   Note that most of your features are going to very sparse as each tweet has very few words.

This is just an example of a simple representation. Feel free to use it, with an appropriately chosen dictionary. Or, you can use another set of features that you think is more effective.

You can use any of the data before December 20, 2012 to select your representation. For the data beginning on December 20, 2012, you will make predictions and update your model using online stochastic gradient descent.

Give a detailed description of your feature representation, and how you compute it. Enter this description into the iPython notebook you build for this project, together with your code that implements it.

2. *Stochastic gradient descent* (80 points)

Your representation above gives a vector $x(t) \in \mathbb{R}^d$ for an arbitrary tweet t. The labelled set of tweets is in the dataset `wishes-labelled`. Each tweet consists of the text of the tweet, a unique global id, and the label of the tweet. The unlabelled tweets are marked by a '?'. Your job is to train an ℓ_2-regularized logistic regression classifier on the entire set of tweets using stochastic gradient descent.

Recall the stochastic gradient descent framework for logistic regression that was covered in class. As you process each tweet, use your existing model to predict the label of that tweet. If the tweet is labelled, note whether or not you make an error and update your model using the stochastic gradient descent update on that tweet. The online stochastic gradient descent begins with the data on December 20, 2012. Initialize the model with $\beta = 0$ (uniform).

If the tweet is unlabelled, you should use your existing model to predict the label. Print the id of the unlabelled tweet and the label of the tweet in a file `predict_unlabelled.txt` in the format

`<tweet_id> <predicted_label>`

Thus, there should be one per line for every unlabelled tweet. Note that you do not make different passes over the data to work on the labelled and unlabelled examples. You should simultaneously predict over the unlabelled data points and update your model corresponding to the labelled data points.

The process the tweets more quickly, following three lines of code should be evaluated. This is just a trick to pre-fetch the data:

```
for s in wishes.subsets():
   for l in wishes.iter(s):
      break
```

Implement stochastic gradient descent, and calculate the error rate (defined as the ratio of the number of mistakes you have made to the total number of tweets encountered up to the current point of time) on the labelled tweets. Plot the error rate after every 1,000 tweets and plot a curve of the error rate as a function of the total number of tweets encountered.

Remember the role the learning rate plays in SGD. You should try the following values of the learning rate $\eta \in \{0.1, 0.5, 1.0, 0.3/t, 1/t, 2/t, 0.3/\sqrt{t}, 1/\sqrt{t}, 5/\sqrt{t}\}$ where t is the number of iterations so far. Feel free to try other values of η as well.

Submit an iPython notebook called `lsd_project2.ipynb` in your `submissions/lsdproj2` directory. Include plots corresponding to each of the different values of η and one test file corresponding to the model that you think has the lowest error rate.

Not that since you are using stochastic gradient descent, the code is not parallelized. So, you should use a single node cluster.