# Predicting Gender From Twitter Online Data

Team: twitte-tree

Members: Zhengxuan Wu, Peiqi Wang

## Project Overview

This project focuses on the relationship between genders and their twitter profile information, such as twitts, profile picture, and profile picture features. Twitts data of 5,000 frequent words were provided for 4,998 users. Besides, the profile pictures contain 30,000 feature per user, which is a 100x100x3 RGB pictures. Other than this, we also have profile picture features data, which contain 7 instinct image features that were extracted before-hand. The final goal of this project is using those data to predict the gender of the user as accurate as possible, and do some analysis based on the result. Besides, we also compare different models.

## Model #1

Due to the large number features, and the difficulties of handle the image data, the first attempt was to use the training data of individual twitts. Because we are having a big amount of features, we started native functions in Matlab toolboxes, such as stepwise regression, OLS models, and LASSO. And the accuracy is compared between those three models. Apparently, LASSO is leading the board. It is simply because in our training data of words, there are large number of un-used or non-informative features which will be filtered by LASSO regression or Step-wise. The training data accuracy is displayed

below in Figure 1.

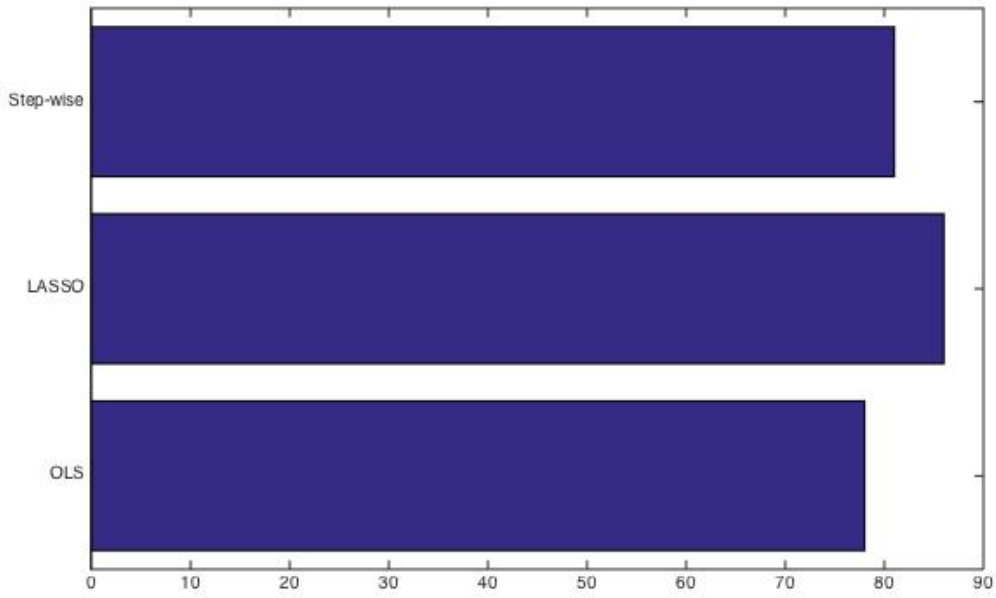



**Figure 1.** Accuracy of Step-wise, LASSO, and OLS Model On Words Data

Besides, using 10-folds cross validation, we were able to determine the best lambda for LASSO regression, which is $\lambda = 0.0269$ as shown in Figure 2. The actual testing data accuracy is 86.06, which beats the first base-line.

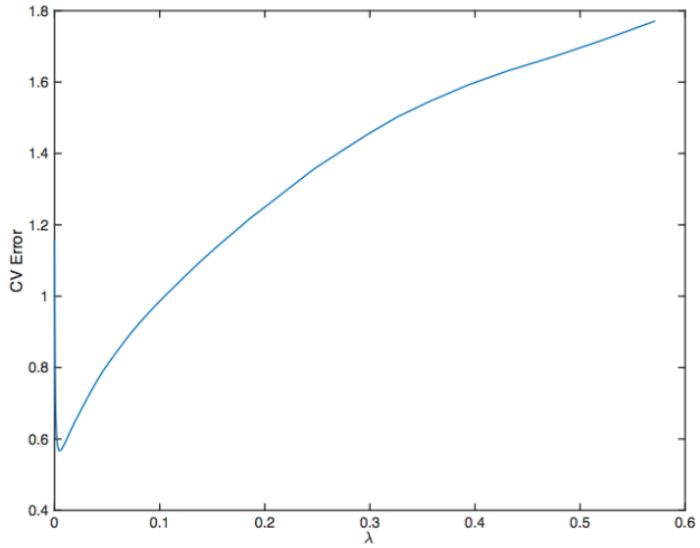



**Figure 2.** CV Error vs. Lambda For LASSO

## Model #2

On the contrast to the LASSO regression, we also use logistic regression model with L1-morm penalty. The reason is that logistic regression is a non-linear model, and in this case, the data might be non-linearly separable. We used cross validation for getting the penalty coefficient as well. The best lambda is 0.03. The testing data accuracy is 86.69, which also beat the first base-line. Unfortunately, the output test accuracy is 73.22% which is lower than the normal LASSO or normal L2 norm logistic regression.

## Model #3

In order to refine our model, we attempted glmet followed by Naïve Bayes Classification. The idea behind this is assuming every feature point of the words is independent, and different properties in different ranges of models may be different. And by classifying the properties in different group first, we may be able to tune our model to achieve a better RMSE.

To conduct the Naïve Bayes on this dataset, we first need to choose an online or built-in Naïve-Bayes source code. The procedure will be, we would use the train data to thrain our Naïve-Bayes' train. However, it turned out will increase RSME dramatically to 0.9877. Basically, it means our model is incorrect. The reason why our data is not fit with Naïve Bayes is because the words are somehow dependent on each other. In other words, we could not use this as a part of our final project. The actual testing accuracy for this method is 78.05%.

## Model #4

Other than the training data of words, we also have the profile image. However. The original image is too big to compile it in a reasonable time, so we resize and gray scale the profile image. Specifically, we resize the image to 50*50, and gray scale the image once. Thus, the feature is reduced down from 30,000 to 2500 for each person. And the new run PCA for this image data, and we plot the re-construction err(accuracy), as shown below.

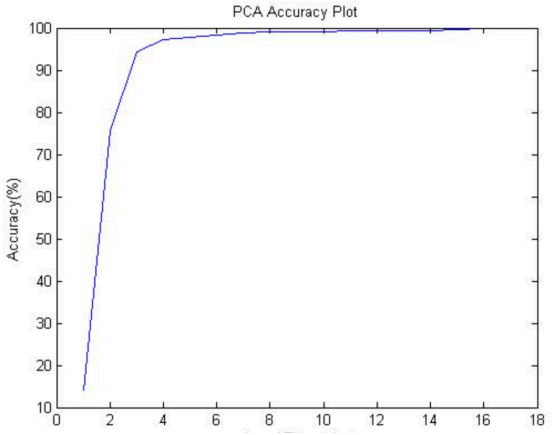
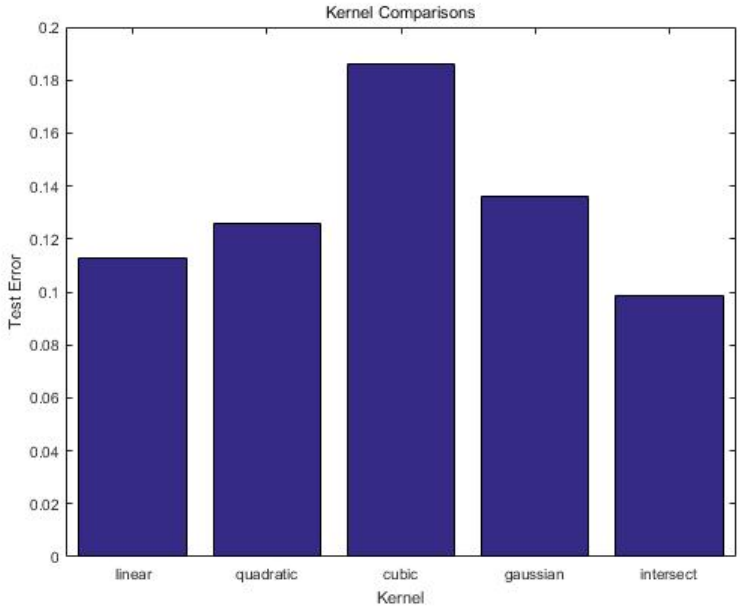



**Figure 3.** PCA Accuracy For Image

As we can see that jus the top ten PCA-ed feature will already bring us a 90% percent of reconstruction error of the original data. Based on this result, we used top 80 features from the PCA-ed data, and run KNN-Method. The actual testing error for this method is 73.07%.

## Model #5

We use an online library libsvm to do the calculation about the SVM method.
First thing is to choose a good kernel, in one of the homeworks we calculate different types of kernels when doing documents classification. the test error looks like this:




From this we can see that the best kernel to use when doing documents classification is the intersect kernel. This kernel is summing the minimum value of each features with 2 people.This kernel is good here because that when two people have the same peak of using some words, the “distance” between this two people becomes pretty close since the value of the kernel is getting large.

Here we are doing the sexual classification using words people said in twitter, this looks quite like the document classification, so I choose to use this kernel here to do the SVM. when doing SVM using intersect kernel, a parameter C is to be decided, here we are using

nu-SVM to decide the penalty term since it has a good character that it varies from 0 to 1. So we use a line space from 0 to 1 and divide it into 10 pieces and using each value of C to do the cross validation to get the best C, from the calculation we know that when C = 0.3, CV gives out the minimum value. After we get the best value of C, we can use this parameter and do a training using the full training set and get a better model. the training error of C = 0.3 is about 90.32%

Here I also try different types of other kernels, like using a linear kernel can give a testing error as big as 88.39%, which means that linear kernel is also not a bad choice for

this twitter sexual classification. I also test a Gaussian kernel, which seems to be a really bad idea because the default value of the Gaussian kernel gives out an accuracy about only 60% percent, but since there are two parameters in a Gaussian kernel, sigma and C, we need to do a two dimensional gradient descent to find the local minimum, actually the minimum happens when C = 0.0022 and sigma = 0.1. here I use a logspace to try different values of C and sigma. This can gives out a training accuracy about 85.12%.

Model combination:

To get better result we always combine different model together and gives them different weight, since if we using our predict label to do this model combination is useless(high accuracy model will dominate the predict label, it's better for us to do the model combination using the probability

and weight each of due to their test accuracy.

We tried to combine the logistic regression model and the intersect kernel SVM model and using the online library calculate the probability of each observation's label. then we weight the probability by 30%-logistic regression and 70%-SVM (since SVM has the higher test accuracy about 89% and logistic regression's accuracy is lower, about 86%) and then we can a new model. This model changes 124 labels of the SVM predicted labels of the training set and after check the true label we found that the most of these changed labels are good.

So finally we using this combined model and get an accuracy about 89% on the leaderboard.

| Method | Parameters | Test accuracy |
|---|---|---|
| Linear regression with L1 | lambda = 0.05 | 86.06% |
| SVM with Gaussian kernel | C = 0.0022 sigma = 0.1 | 84.98% |
| SVM with Linear kernel | C = 0.003 | 86.78% |

| | | |
|---|---|---|
| SVM with Intersect kernel | nu = 0.3 | 87.98% |
| logistic regression with L1 | lambda = 0.038 | 86.69% |
| Combine model | Logistic regression with L1<br>SVM with intersect kernel | 89.09% |
| K-means | apply to images features<br>K = 15 | 72.30% |
| Naive bayes | | 78.05% |
| PCA image | 80 features remains | |
| K-NN | apply to PCAed data<br>K = 15 | 73.07% |