ECE532: Spooky Author NLP and LSR

Table of Contents

By Thomas Hansen and Junior Quintero 1
Categorizing their text 2
Testing on small sample set 3
Lasso use 7
References 8

By Thomas Hansen and Junior Quintero

In our project we've decided to use our dataset from the Kaggle Spooky Author Identification competition[0]

Here we're given a large dataset and testing set of data for comparing work done by different horror authors, and our goal will be to tell if we can differentiate the authors from one another just based off analyzing different words and styles they use.

Initially we'll test the idea with linear regression, which we expect to get poor rates for, and then we'll use different techniques to read in the data, such as changing the number of senteces we read in at a time to improve accuracy, or trying new tequniques not used in class.

```
% For this project you may want to have matlabNLP installed for some
% versions.

% read in file, set training data to td
filename = 'train.csv';
file = readtable(filename);
td = table2array(file);

% Shorten td to reasonable size, can remove later
% td = td(1:10000,:);

[n,~] = size(td);

eap_occurance = 0;
hpl_occurance = 0;
mws_occurance = 0;
for i = 1:n
    if strcmp(td(i,3),'EAP')
        eap_occurance = eap_occurance + 1;
    elseif strcmp(td(i,3),'HPL')
        hpl_occurance = hpl_occurance + 1;
    elseif strcmp(td(i,3),'MWS')
        mws_occurance = mws_occurance + 1;
    else
        fprinf('Didnt work on line %i\n', i);
    end
end
```

```
% This total should now equal n
if (n == (eap_occurance + hpl_occurance + mws_occurance))
    fprintf('number of names categorized correctly.\n');
end
```

```
number of names categorized correctly.
```

Categorizing their text

Now that we have the size of each required array, we can enter each word into an array for the three authors, and then compare them directly that way.

Recall our end game is we want to calculate the weights, so w = X\y, and X will be a matrix of the given words/sentences, while y will be the estimate for each author.

Another way to write this is that X would be the number of times a word comes up in a sentence, so that times a weight would be the likelihood of it being a specific author.

author = of word per sentence * it's an author

So we would see X as perhapse a large matrix, where each row is a different sentence, each column is the rate at which a word shows up, and each row in the w is the weight for the corrisponding word column in the X matrix. From here we produce a y matrix, this y matrix will be a column with a height equal to the number of sentences, where each sentence is a weight.

Additionally I'll note that the words chosen are to some extent arbitrary. We've been basing words by each author off of other research into each author based off of word use frequency. Additionally we've been avoiding nouns and names as even though some authors use them more often than others it can still throw off the data significantly. Lastly after taking this into account, we've use/thrown out words with high or low weights, where high weights suggest it's more relevant than a low weight.

MWS [1] ELP [2] HPL [3][4]

```
% List of words we've chosen, we've limited the size so that the
 matrix
% multiplication may still run quickly.
words =
 {'ascertain','lay','my','surcingle','hand','thus','to','nor','subject','suffer','
[wn,wm] = size(words);

data = zeros(wm,n);
% Need to collect word data for 100 sentences
for i = 1:wm
    for j = 1:n
        wordLoc = strfind(td{j,2}, words{i});
        data(i,j) = length(wordLoc);
    end
end

% Flipping matrix as it was accidentally built upsidown.
X = transpose(data);

% Now we must build the y matrix, it would be best to incorporate this
 in
% with the above algo later on, though the O(N^2) time complexity
 remains
```

```
% the same. This will be the true value y matrix.
y = zeros(n,1);
for i = 1:n
    if (length(strfind(td{i,3}, 'EAP')) >= 1)
        y(i,1) = +1;
    else
        y(i,1) = -1;
    end
end

% So we can build the weight setup as
w = X\y;

% And then we'll test it for EAP in the next 10 matricies
```

```
Warning: Rank deficient, rank = 64, tol =  1.942057e-09.
```

Testing on small sample set

here we're testing our results on a sample set of 1000.

```
test = table2array(file);
test = test(10000:10999,:);
n = length(test);

% We need to set up the environment by copying and pasting the code
 from
% above but with a test
X_test = zeros(wm,n);
for i = 1:wm % for each word
    for j = 1:n % for each sentence
        wordLoc = strfind(test{j,2}, words{i}); % if the word exists
 in the sentence
        X_test(i,j) = length(wordLoc); % how many times the word shows
 up
    end
end
X_test = transpose(X_test); % corrects the matrix orientation

% So the predicted vector is
y_hat = X_test*w;

% And we can compare this to the expected output, so
y_expected = zeros(n,1);
for i = 1:n
    if (length(strfind(test{i,3}, 'EAP')) >= 1)
        y_expected(i) = +1;
    else
        y_expected(i) = -1;
    end
end

% Now we test it to see easily how many were right
vals = zeros(20,2);
```

```
for i = 1:n
    vals(i,1) = y_hat(i);
    vals(i,2) = y_expected(i);
end

% sum up right answers
sum = 0;
for i = 1:n
    if (sign(vals(i,1)) == sign(vals(i,2)))
        sum = sum + 1;
        vals(i,1);
        vals(i,2);
    end
end

fprintf('There were %i right answers out of %i, which equals a %2.2d
 correct percentage.\n', sum, n, (sum/n)*100);
```

```
There were 596 right answers out of 1000, which equals a 5.96e+01
 correct percentage.
```

Here the goal is to run the same algorithm we ran in the least squares section, however now we'll want to input multiple sentences to see if we can improve our ability to distinguish one author from another. Eventually we'll be putting this in a makeshift decision tree, where it'll determine which of the three authors wrote it.

First we count and split the test data into each of the authors

```
filename = 'train.csv';
file = readtable(filename);
td = table2array(file);
td = td(18000:end,:); % starts at quasi random point near the end

[n,m] = size(td);

sentence_size = 5;

eap_occurance = 0;
eap_sentences = cell(1,sentence_size);
hpl_occurance = 0;
hpl_sentences = cell(1,sentence_size);
mws_occurance = 0;
mws_sentences = cell(1,sentence_size);
for i = 1:n
    if strcmp(td(i,3),'EAP')
        eap_occurance = eap_occurance + 1;
        if (eap_occurance <= sentence_size)
            eap_sentences{eap_occurance} = td(i,2);
        end
    elseif strcmp(td(i,3),'HPL')
        hpl_occurance = hpl_occurance + 1;
        if (hpl_occurance <= sentence_size)
            hpl_sentences{hpl_occurance} = td(i,2);
        end
    elseif strcmp(td(i,3),'MWS')
```

```
        mws_occurance = mws_occurance + 1;
        if (mws_occurance <= sentence_size)
            mws_sentences{mws_occurance} = td(i,2);
        end
    else
        fprinf('Didnt work on line %i\n', i);
    end
end

% This total should now equal n
if (n == (eap_occurance + hpl_occurance + mws_occurance))
    fprintf('number of names categorized correctly.\n');
end

% Now that we have cell arrays of each of the 3 authors, we can show
 that
% this will more readily prove

tmp = zeros(1,length(eap_sentences));
for i = 1:length(eap_sentences)
   % we run the function on each sentence
   tmp(i) = isAuthor(words, eap_sentences{i}, w);
end

newSum = 0;
denom = 0;
for i = 1:length(eap_sentences);
    denom = denom + 1;
    if (sign(y_expected(i)) == sign(tmp(1,i)))
        newSum = newSum + 1;
    end
end
fprintf('So on average it correctly guesses EAP sentences %2.1d
 percent of the time', 100*(newSum/denom));
```

```
number of names categorized correctly.
So on average it correctly guesses EAP sentences 60 percent of the
 time
```

Using a decision tree to improve LSR sentence multiple times it'll decide it's either EAP, HPL or MWS.

```
eap_w = w;
hpl_w = 2*rand(size(w)) - 1; % generates author weights
mws_w = 2*rand(size(w)) - 1;
% hpl_w = generate_w();
% mws_w = generate_w();

% And given the sets of sentences from each of the authors, we can use
 a
% for loop and the isAuthor function to determine whether each
 sentence is
% part of the author or not. If it isn't, the decision tree will break
 down
% and try another author. If it recieves a positive result, it'll
 continue
```

```
% on to say who it is.
%                        input sentences
%                         isAuthor(EAP)
%                      no /  < 50% <   \ yes
%                        /              \
%                isAuthor(HLP)          EAP
%            no /  < 50% <   \ yes
%              /              \
%          isAuthor(MWS)      HLP
%      no /  < 50% <   \ yes
%  use highest          \
%   saved %             MWS
%  /    |   \
% EAP  HPL  MWS
%
% Decision Tree implementation:

% X_auth_sent = eap_sentences'; % n = 50; decision here is arbitrary
X_auth_sent = cell(sentence_size,3);
tmp_cell = eap_sentences';
X_auth_sent(:,2) = tmp_cell(:,1);

y_auth_expected = ones(sentence_size,3); % This just means that
 they're all the
% correct author, since X is the same

eap_percent = isAuthors(X_auth_sent, words, eap_w, y_auth_expected);
if (eap_percent > .5)
    % Then it's EAP
    fprintf('Sentences were by EAP\n');
else
    hpl_percent = isAuthors(X_auth_sent, words, hpl_w,
 y_auth_expected);
    if (hpl_percent > .5)
        % Then it's HPL
        fprintf('Sentences were by HPL\n');
    else
        mws_percent = isAuthors(X_auth_sent, words, mws_w,
 y_auth_expected);
        if (mws_percent > .5)
            % Then it's MWS
            fprintf('Sentences were by MWS\n');
        else
            % Here they're all below our margin of error
            if (eap_percent > hpl_percent && eap_percent >
 mws_percent)
                % EAP had the most correct
                fprintf('Sentences were by EAP, caught on second
 round.\n');
            elseif (hpl_percent > eap_percent && hpl_percent >
 mws_percent)
                % It's HPL
                fprintf('Sentences were by HPL, caught on second
 round.\n');
```

```
            else
                % It's MWS or they all match percentages
                fprintf('Sentences were by MWS, caught on second
 round.\n');
            end
        end
    end
end

% So the function
%                isAuthors(test, words, w, y_expected)
% Is used to return the percentage of results. Additionally it's done
 in a
% fashion where each if is embeded in an else, and this is done to
 reduce
% the number of computations needed, while still saving the percentage
% values.
```

```
ans =

   -0.0261   -0.7846


ans =

    4.3343    8.3253

Sentences were by HPL
```

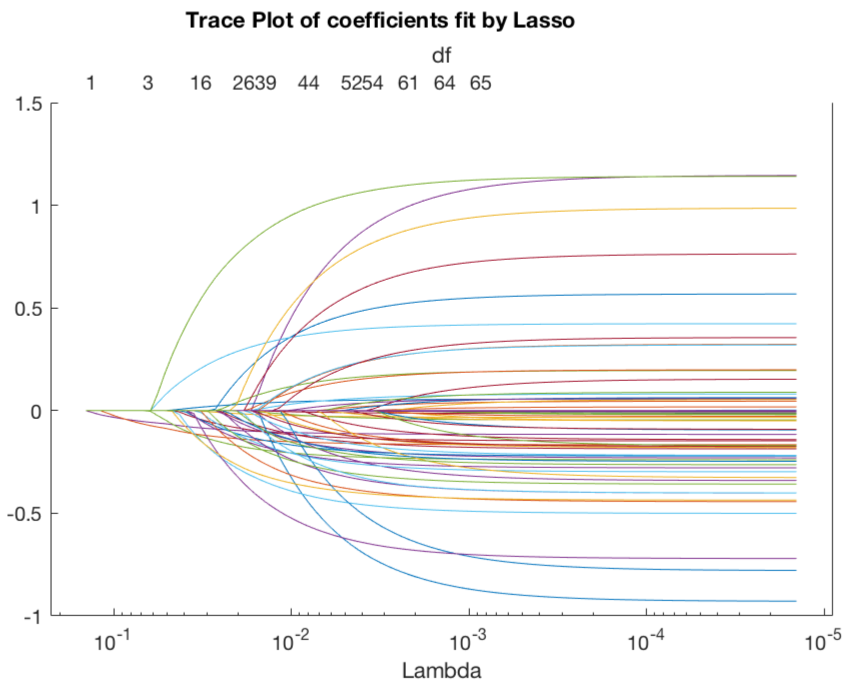
Lasso use

For comparison here is the result of the built in LASSO regression on the data

```
[B, FitInto] = lasso(X,y);
lassoPlot(B,FitInto,'PlotType','Lambda','XScale','log');

% The plot shows the nonzero coefficients in the regression for
 various
% values of the Lambda regularization parameter. Larger values of
 Lambda
% appear on the left side of the graph, meaning more regularization,
% resulting in fewer nonzero regression coefficients.
%
% The dashed vertical lines represent the Lambda value with minimal
 mean
% squared error (on the right), and the Lambda value with minimal mean
% squared error plus one standard deviation. This latter value is a
 recommended
% setting for Lambda. These lines appear only when you perform cross
 validation.
% Cross validate by setting the 'CV' name-value pair. This example
 uses
% 10-fold cross validation.
```

```
%
% The upper part of the plot shows the degrees of freedom (df),
 meaning the
% number of nonzero coefficients in the regression, as a function of
 Lambda.
% On the left, the large value of Lambda causes all but one
 coefficient to
% be 0. On the right all five coefficients are nonzero, though the
 plot shows
% only two clearly. The other three coefficients are so small that you
 cannot
% visually distinguish them from 0.
%
% For small values of Lambda (toward the right in the plot), the
% coefficient values are close to the least-squares estimate.
%
```



References

[0] Spooky Author Identification | Kaggle, www.kaggle.com/c/spooky-author-identification.

[1] November 22, 2013 By Andy H. (NY). Mary Shelley's Frankenstein. Vocabulary.com, www.vocabulary.com/lists/344129.

[2] (NY), July 25 2013 By Vocabulary.com. Poe's Favorite Words, collected by Charles Harrington Elster. Vocabulary.com, www.vocabulary.com/lists/285259.

[3] H.P. Lovecrafts 10 Favorite Words and a Free Lovecraft eBook. Tor.com, 14 Dec. 2014, www.tor.com/2011/03/01/lovecraft-favorite-words-free-ebook/.

[4] Wordcount for Lovecraft's Favorite Words. The Arkham Archivist Wordcount for Lovecrafts Favorite Words Comments, arkhamarchivist.com/wordcount-lovecraft-favorite-words/.

[5] Green, Rachel M, and John W Sheppard.#Comparing Frequency- and Style-Based Features for Twitter Author Identification.# Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference, www.aaai.org/ocs/index.php/FLAIRS/FLAIRS13/paper/viewFile/5917/6043.

[6] Zhao, Peng, and Bin Yu.#On Model Selection Consistency of Lasso. Journal of Machine Learning Research, vol. 7, Nov. 2006, pp. 2541#2563., www.jmlr.org/papers/volume7/zhao06a/zhao06a.pdf.

Published with MATLAB® R2016a