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 2
Using a decision tree to improve LSR 4
Lasso use 5
References 6

By Thomas Hansen and Junior Quintero

In our project we've decided to use our dataset from the Kaggle Spooky Author Identification competition[0] to predict whether a given sentence belongs to Edgar Allen Poe (EAP), Howard Philipps Lovecraft (HPL), May W. Shelley (MWS)

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, but as we progress we'll use a decision tree[7] to figure out which author wrote which sentences, and we'll look into the use of lasso and different lambdas as an alternative as well.

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

```
filename = 'train.csv'; % read in file, set training data to td
file = readtable(filename); td = table2array(file); [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); % error
    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. X represents the number of times a word comes up in a sentence, so that the product would be the likelihood of it being a specific author.

Additionally note that the words chosen are to some extent arbitrary. We've been basing words by each author off of other research into word frequency of these authors[1][2][3][4]. We've also been avoiding nouns and names even though some authors use them more (i.e. EAP used the name elizabeth frequently)[2] still it can throw off the data significantly if a different author uses that name. Lastly 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]

We've limited the size so that the 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
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
w = X\y; % building the weight
% Now we'll test it for EAP in the next 10 matrices
```

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

Testing on small sample set

here we're cross validating the data with 1000 points from the data.

```
test = table2array(file);
test = test(10000:10999,:);  n = length(test);
% Need to set up the environment by copying the code 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;
    end
end
fprintf('There were %i right answers out of %i, which equals a %2.2d
 correct percentage.\n', sum, n, (sum/n)*100);
% Goal: Run the same LS algorithm but with multiple sentences in order
% to improve our ability to distinguish authors
% 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,:); % start point was
 arbitrary
[n,m] = size(td);       sentence_size = 100;
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 % so running over each sentence we check the author
    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); % error finding author
    end
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));
```

```
There were 596 right answers out of 1000, which equals a 5.96e+01
 correct percentage.
So on average it correctly guesses EAP sentences 66 percent of the
 time
```

Using a decision tree to improve LSR

We're using a decsion tree here to figure out which author the set of sentences belong to, though we'll have to generate each new weight.

In our tree it is built of conditional statements and will flow down checking each author for likely authorship, if it is likely. As the tree grows, we could use more complex trees, setting each author as a leaf and nodes would be able to distinguish between similar authors [7].

```
eap_w = w; hpl_w = w; mws_w = w;
%                        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 = cell(sentence_size,3);
tmp_cell = eap_sentences';
X_auth_sent(:,2) = tmp_cell(:,1);
y_auth_expected = ones(sentence_size,3);
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)
                fprintf('Sentences were by EAP\n');
            elseif (hpl_percent > eap_percent && hpl_percent >
 mws_percent)
                fprintf('Sentences were by HPL\n');
            else
                % It's MWS or they all match percentages
                fprintf('Sentences were by MWS\n');
            end
        end
    end
end
```

Sentences were by MWS

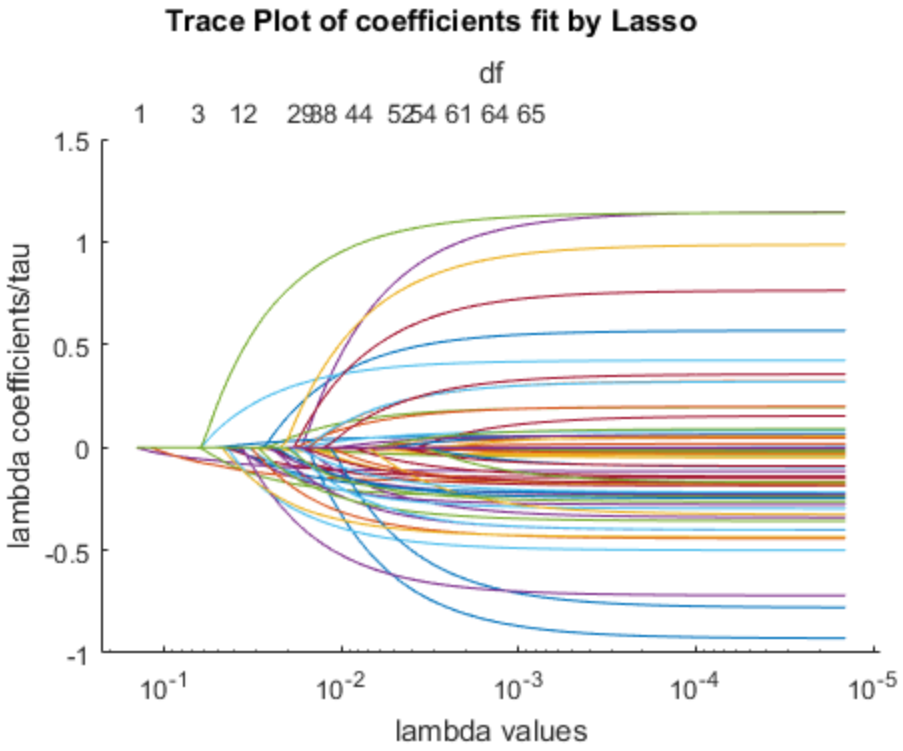
Lasso use

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

The upper part of the plot shows the degrees of freedom (df). For small values of Lambda (toward the right in the plot), the coefficient values are close to the least-squares estimate.

```
[B, FitInto] = lasso(X,y);
lassoPlot(B,FitInto,'PlotType','Lambda','XScale','log');
ylabel('lambda coefficients/tau'); xlabel('lambda values');
```



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 Lovecrafts 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.

[7] Ben-Haim, Yael, and Elad Tom-Tav. "A Streaming Parallel Decision Tree Algorithm.” Journal of Machine Learning Research, vol. 11, Feb. 2010, pp. 849–872.

Published with MATLAB® R2017a