UTD 2016 Spring

CS6301.5U1 Advanced Computational Methods for Data Science

Assignment 8 – Bag of Words

Name: Yu Zhang, Mingyue Sun

NetID: yxz141631, mxs151730

1. Overview
2. Creating DTM and scaling with TF-IDF
3. Classification with glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models
4. Using Cross Validation to Evaluate Model Performance
5. Conclusion

1. Overview

In this assignment, we are required to build a classification model for SMS text messages using the Bag of Words technique. Our approach will be formalized as following: 1. use Bag of Words to create a DTM by removing a list of stop words of our choice and scale this DTM with TF-TDF technique. 2. build a ? classification model and perform the cross validation for this model to get the accuracy of this model. 3. build a lasso and elastic-net regularized generalized linear model and compare the accuracy with the previous model.

2. Creating DTM and scaling with TF-IDF

In order to use the Bag of Words technique, we first installed the "text2vec" package and imported it into our workspace. Then a stop-word list which contains around 500 frequently appearing terms was selected and introduced into the vocabulary for all message text body. The work was finished as follow:

```
> sms <- read.csv(file = "./smsspamcollection/SMSSpamCollection", h
eader = FALSE, quote ="", sep = "\t")
> names(sms)<-c("Class","text")
> sms$text<-as.character(sms$text)
> set.seed(42L)
> it<-itoken(sms$text,process_function = tolower, tokenizer = word_
tokenizer)
> sw <- c( "wif","a","about","above","across","after","again","agai
nst","all","almost","alone","along","already","also","although","al
ways","am","among","an","and","another","any","anybody","anyone","a
nything","anywhere","are","area","areas","aren't","around","as","as
k","asked","asking","asks","at","away","b","back","backed","backin
g","backs","be","became","because","become","becomes","been","befor
e","began","behind","being","beings","below","best","better","betwe
en","big","both","but","by","c","came","can","cannot","can't","cas
e","cases","certain","certainly","clear","clearly","come","coul
d","couldn't","d","did","didn't","differ","different","differentl
y","do","does","doesn't","doing","done","don't","down","downed","do
wning","downs","during","e","each","early","either","end","ende
d","ending","ends","enough","even","evenly","ever","every","everybo
dy","everyone","everything","everywhere","f","face","faces","fac
t","facts","far","felt","few","find","finds","first","for","fou
r","from","full","fully","further","furthered","furthering","furthe
rs","g","gave","general","generally","get","gets","give","given","g
ives","go","going","good","goods","got","great","greater","greates
t","group","grouped","grouping","groups","h","had","hadn't","ha
s","hasn't","have","haven't","having","he","he'd","he'll","her","he
re","here's","hers","herself","he's","high","higher","highest","hi
m","himself","his","how","however","how's","i","i'd","if","i'l
l","i'm","important","in","interest","interested","interesting","in
terests","into","is","isn't","it","its","it's","itself","i'v
e","j","just","k","keep","keeps","kind","knew","know","known","know
```

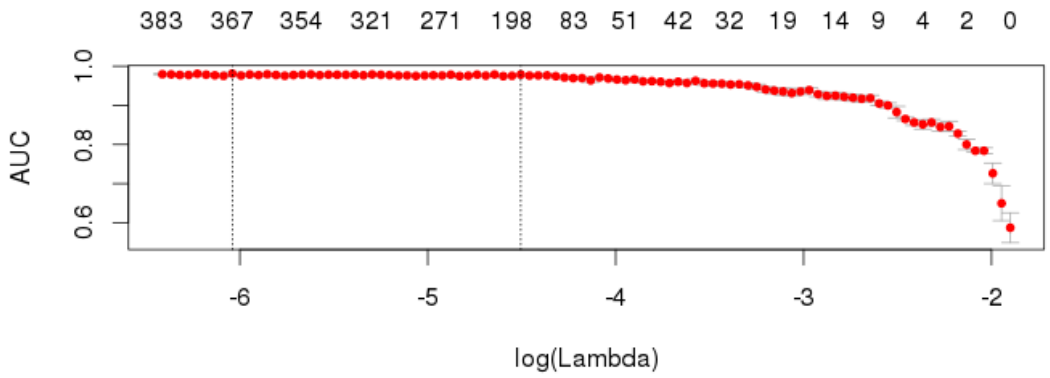
```
e","seem","seemed","seeming","seems","sees","several","shall","sha
n't","she","she'd","she'll","she's","should","shouldn't","show","sh
owed","showing","shows","side","sides","since","small","smaller","s
mallest","so","some","somebody","someone","something","somewher
e","state","states","still","such","sure","t","take","taken","tha
n","that","that's","the","their","theirs","them","themselves","the
n","there","therefore","there's","these","they","they'd","they'l
l","they're","they've","thing","things","think","thinks","this","th
ose","though","thought","thoughts","three","through","thus","to","t
oday","together","too","took","toward","turn","turned","turning","t
urns","two","u","under","until","up","upon","us","use","used","use
s","v","very","w","want","wanted","wanting","wants","was","was
n't","way","ways","we","we'd","well","we'll","wells","went","wer
e","we're","weren't","we've","what","what's","when","when's","wher
e","where's","whether","which","while","who","whole","whom","wh
o's","whose","why","why's","will","with","within","without","wo
n't","work","worked","working","works","would","would
n't","x","y","year","years","yes","yet","you","you'd","you'll","you
ng","younger","youngest","your","you're","yours","yourself","yourse
lves","you've")
> vocab <- create_vocabulary(it, stopwords = sw)
  |============================================================| 100%
> it <- itoken(sms$text, tolower, word_tokenizer)
> Vectorizer <- vocab_vectorizer(vocab)
> dtm <- create_dtm(it,Vectorizer)
  |============================================================| 100%
> dim(dtm)
[1]  5574 10949
> dtm_tfidf = transform_tfidf(dtm)
idf scaling matrix not provided, calculating it form input matrix
> dtm_tfidf
5574 x 10949 sparse Matrix of class "dgCMatrix"
```

As we can see here, by calling the create_dtm and transform_tfidf functions respectively, we were able to create the document-term matrix (5574 docs X 10949 terms after removing stopwords) and scale this matrix with TF-IDF.

3. Classification with glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models

In the meanwhile, we also tried to build a lasso and elastic-net regularized generalized linear model for these text messages and check the accuracy for this model as well. First of all, we just tried this model with the same vocabulary and the result is as follow:

```
> fit <- cv.glmnet(x = dtm, y = sms[['Class']],family = 'binomial',
type.measure = "auc",alpha = 1,nfolds = 5,thresh = 1e-3,maxit = 1e
3)
> plot(fit)
> round(max(fit$cvm), 4)
[1] 0.981
> bestlam = fit$lambda.min
> bestlam
[1] 0.002381186
>
```

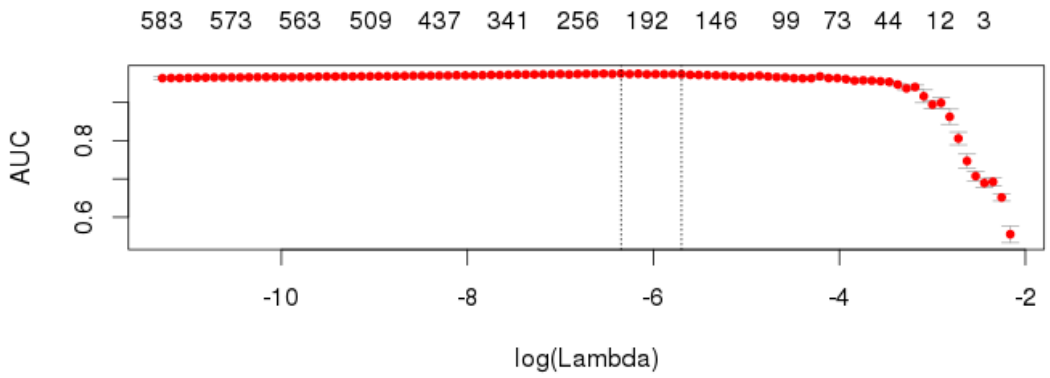


And we can see that for this basic model, the max mean-cross-validated error was 0.981 and from the chart above, we can find the best lambda value to be 0.002381186.

After that we pruned the vocabulary by setting some requirement for the terms in the vocabulary: 1. each term's minimum count has to be at least 10. 2. each term's maximum occurrence rate across all messages has to be at most 0.5. 3. each term's minimum occurrence rate across all messages has to be at least 0.001. And by doing that, we were able to filter off most of the terms in the original vocabulary and leave 933 terms (by default, they could be either monogram, bigram or trigram) that we think are important in determining the message to be ham or spam. The work is as follow:

```
> pruned_vocab <- prune_vocabulary(vocab, term_count_min = 10, doc_
proportion_max = 0.5, doc_proportion_min = 0.001)
> tokens <- sms$text %>%
+ tolower() %>%
+ word_tokenizer()
> it <- itoken(tokens)
> vectorizer <- vocab_vectorizer(pruned_vocab)
> dtm <- create_dtm(it, vectorizer)
  |======================================================| 100%
> dim(dtm)
[1] 5574  933
> dtm <- dtm %>% transform_tfidf()
idf scaling matrix not provided, calculating it form input matrix
> fit <- cv.glmnet(x = dtm, y = sms[['Class']],family = 'binomial',
type.measure = "auc",alpha = 1,nfolds = 5,thresh = 1e-3,maxit = 1e
3)
> plot(fit)
> round(max(fit$cvm), 4)
[1] 0.975
> bestlam = fit$lambda.min
> bestlam
[1] 0.001748646
> |
```

And we can see that for this improved model, the max mean-cross-validated error was reduced to 0.975 which is a good indication and from the chart below, we can find the best lambda value for this model to be 0.001748646.



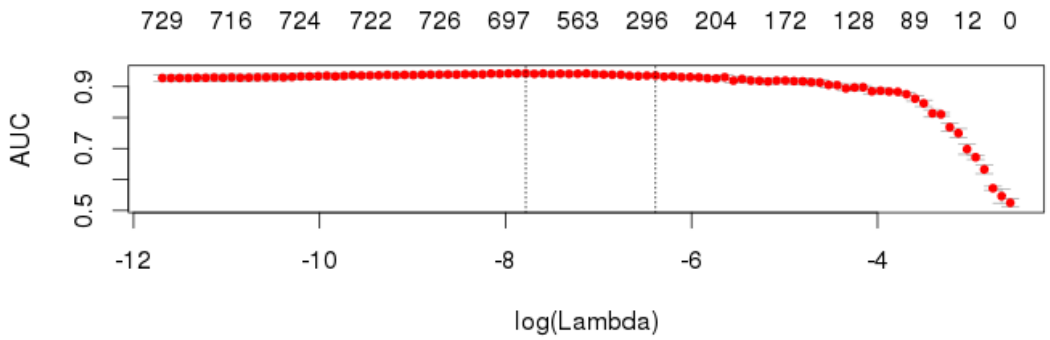
Right after that, we just divided the original dataset into a training and testing subset, built the model again with training set and test it against the testing set. The work is shown as below:

```
> train = sample(1:5574, 4574)
> x.train = dtm[train,]
> y_vals = sms[['Class']]
> y.train = y_vals[train]
> x.test = dtm[-train,]
> y.test = y_vals[-train]
> lasso.mod = glmnet(x.train,y.train,alpha=1,family = "binomial")
> lasso.probs = predict(lasso.mod,s=bestlam,newx = x.test, response
= "response")
> lasso.pred = rep(0, 1000)
> lasso.pred[lasso.probs > .5] = "spam"
> lasso.pred[lasso.probs <= .5] = "ham"
> table(lasso.pred, y.test)
          y.test
lasso.pred ham spam
      ham  853   27
      spam  10  110
> mean(lasso.pred == y.test)
[1] 0.963
>
```

And we can see that this model could make a 96.3% right predication on the testing set which is a quite acceptable accuracy rate.

Then since we think the bigram term might play a more important role in the vocabulary than any only terms in ham/spam determination, we then created a vocabulary with only bigram inside as below:

```
> it <- itoken(tokens)
> vocab <- create_vocabulary(it, ngram = c(2L, 2L)) %>%
+ prune_vocabulary(term_count_min = 10,doc_proportion_max = 0.5,doc
_proportion_min = 0.001)
  |======================================================| 100%
> vectorizer <- vocab_vectorizer(vocab)
> dtm <- tokens %>%
+ itoken() %>%
+ create_dtm(vectorizer) %>%
+ transform_tfidf()
  |======================================================| 100%
idf scaling matrix not provided, calculating it form input matrix
> dim(dtm)
[1] 5574  814
> fit <- cv.glmnet(x = dtm, y = sms[['Class']],family = 'binomial',
alpha = 1,type.measure = "auc",nfolds = 5,thresh = 1e-3,maxit = 1e
3)
> plot(fit)
> round(max(fit$cvm), 4)
[1] 0.9424
> bestlam = fit$lambda.min
> bestlam
[1] 0.0004168682
>
```



This time we can see, we further removed 119 terms from the filtered vocabulary and now it only contained 814 bigram terms which we regarded as important. And to our delight, we noticed a huge drop of the max mean-cross-validated error to 0.9424 with this approach. And from the chart above, we can find the best lambda value for this model to be 0.0004168682.

Finally, we used the previously-divided dataset in building the model with new vocabulary on training set and use this model to make predication on the testing set again. The work is shown below:

```
> x.train = dtm[train,]
> x.test = dtm[-train,]
> lasso.mod = glmnet(x.train,y.train,alpha=1,family = "binomial")
> lasso.probs = predict(lasso.mod,s=bestlam,newx = x.test, response
= "response")
> lasso.pred = rep(0, 1000)
> lasso.pred[lasso.probs > .5] = "spam"
> lasso.pred[lasso.probs <= .5] = "ham"
> table(lasso.pred, y.test)
          y.test
lasso.pred ham spam
      ham  846   33
      spam  17  104
> mean(lasso.pred == y.test)
[1] 0.95
>
```

And we can see that accuracy for this model was only 95% although still an acceptable one but slightly worse than the previous model.

4. Using Cross Validation to Evaluate Model Performance

In this part, we will use 10-folds cross validation to evaluate performance of Lasso and Elastic-Net Regularized Generalized Linear Model, feeding with DTM and DTM scaled with TF-DTF. And compare the performances between feeding with different DTMs.

a) Feeding The Model with DTM

```
> k=10
> err=seq(1:10)
> set.seed(1)
> folds=sample(1:k,nrow(dtm),replace = TRUE)
> for(i in 1:k)
+ {
+   train=dtm[folds!=i,]
+   test=dtm[folds==i,]
+   train.y=sms[folds!=i,]$Class
+   test.y=sms[folds==i,]$Class
+   fit=cv.glmnet(x=train, y=train.y,family = 'binomial', alpha = 1,
 type.measure = "auc", nfolds = 5, thresh = 1e-3, maxit = 1e3)
+
+   pred=predict(fit,test)
+   pred[pred>0]="spam"
+   pred[pred<=0]="ham"
+
+   countequal=sum(pred==test.y)
+   n=length(test.y)
+   accuracy=countequal/n
+   err[i]=accuracy
+ }
> err
 [1] 0.9604811 0.9763206 0.9763206 0.9572650 0.9756098 0.9652015
 [7] 0.9731959 0.9721707 0.9597198 0.9781145
> ave=mean(err)
> ave
[1] 0.9694399
```

When doing 10-folds cross validation on DTM in glmnet model, the overall accuracy is 0.9694399. This performance is pretty good in terms of accuracy. In the following we will scale the DTM using TF-IDF and using 10-folds cross validation to test the accuracy.

2) Feeding The Model with Scaled DTM in TF-IDF

```
> k=10
> err=seq(1:10)
> set.seed(1)
> folds=sample(1:k,nrow(dtm),replace = TRUE)
> for(i in 1:k)
+ {
+   train=dtm_tfidf[folds!=i,]
+   test=dtm_tfidf[folds==i,]
+   train.y=sms[folds!=i,]$Class
+   test.y=sms[folds==i,]$Class
+   fit=cv.glmnet(x=train, y=train.y,family = 'binomial', alpha = 1,
 type.measure = "auc", nfolds = 5, thresh = 1e-3, maxit = 1e3)
+
+   pred=predict(fit,test)
+   pred[pred>0]="spam"
+   pred[pred<=0]="ham"
+
+   countequal=sum(pred==test.y)
+   n=length(test.y)
+   accuracy=countequal/n
+   err[i]=accuracy
+ }
> err
 [1] 0.9484536 0.9526412 0.9690346 0.9555556 0.9581882 0.9615385
 [7] 0.9731959 0.9795918 0.9492119 0.9579125
> ave=mean(err)
> ave
[1] 0.9605324
```

When doing cross validation on model built with scaled DTM in TF-IDF, the overall accuracy is 0.9605324. This is a little bit lower than the performance before scaled.

5. Conclusion

Given the analysis above we can see that the bag of words technique performs pretty well in this case. Almost all the accuracies are over 95%. We can also notice that, the performance is closely related with the DTM and the way vocabulary is created. After the vocabulary has only bigram case inside the result has a lower performance. Also, if the DTM is scaled by TF-IDF, the accuracy also will go down. Thus, to improve the performance, we need to take care of the choosing of DTM building and scaling strategies. One more important thing is that, in this case, we have 4827 ham but only 747 ham. The performance of this experiment may a little bit high comparing with real world tasks. Because there are only two categories and one of them possesses a large portion.