Training Naive Bayes

02/01/2015 WEI PENG

a) Complete the function $logProd(x)$ with takes as input a vector of numbers in logspace (i.e., $x_i = \log p_i$) and returns the product of those numbers in logspace-i.e.$logProd(x) = \log(\Pi_i p_i)$.

```
function [log_product] = logProd(x)
% LOGPROD Given a vector of numbers in log-space
% (i.e. x(i) = log p(i)), this computes the log of the
% product of the numbers: log \product_i p(i).
%
% inputs:
%   x - A vector of length n containing the factors in
%     log-space: x(i) = log p(i).
%
% output:
%   log_product - A scalar containing the log of the
%     product of the numbers: log \product_i p(i).

% TODO: implement me!
log_product = 0;
log_product=sum(x);
end
```

b) Complete the fucntion $[D] = NB_X GivenY(XTrain, yTrain)$. The output D is a $2 \times V$ matrix, where for any word index $\omega \in 1, \ldots, V$ and class index $y \in 0, 1$, the entry $D(y, w)$ is the MAP estimator of $\theta_{yw} = P(X_w = 1 | Y = y)$ with a Beta(1,2) prior distribution.

```
function [D] = NB_XGivenY(XTrain, yTrain)
% NP_XGIVENY Estimates the probability that a word is
% observed given (conditioned on) the class label.
%
% inputs:
%   XTrain - An [n x V] matrix where each row describes
%     which words are present in a particular document.
%     XTrain(i, j) is 1 if word j appears in the i-th
%     document.
%   yTrain - An [n x 1] vector containing the class labels
%     for the training documents. yTrain(i) is 0 if the
%     i-th document belongs to The Economist or 1 if it
%     belongs to The Onion.
%
% output:
%   D - A [2 x V] matrix, where D(i, j) is an estimate of
%     the probability that word j is in a document with class
%     label i-1: P[X_j = 1 | Y = i-1].

V = size(XTrain, 2);
D = zeros(2, V);
n=size(XTrain,1);
N0=0;N1=0;
for i=1:n
if yTrain(i)==0
  N0=N0+1;
  for j=1:V
    if XTrain(i,j)==1
       D(1,j)=D(1,j)+1;
     end
   end
```

```
   else
         N1=N1+1;
          for j=1:V
             if XTrain(i,j)==1
            D(2,j)=D(2,j)+1;
             end
          end
  end
   end
D(1,:)=(D(1,:)+1)/(N0+1);
D(2,:)=(D(2,:)+1)/(N1+1);
% TODO: implement me!
end
```

c) Complete the function $[p] = NB_Y Prior(yTrain)$. The output p is the MLE for $\rho = P(Y = 1)$.

```
function [p] = NB_YPrior(yTrain)
%NP_YPRIOR Computes the prior probability that the class
% label is 1: P[Y = 1].
%
% inputs:
%   yTrain - An [n x 1] vector containing the class labels
%      for the training documents. yTrain(i) is 1 if the
%      i-th document belongs to The Economist or 2 if it
%      belongs to The Onion.
%
% output:
%   p - The prior probability that the class label is 1.

% TODO: implement me!
p = 0;
n=size(yTrain,1);
N0=0; N1=0;
for i=1:n
    if yTrain(i)==0
         N0=N0+1;
       else N1=N1+1;
      end
end
 p(1)=N0/n;
 p(2)=N1/n;
 p=transpose(p);
end
```

d) Complete the function $[yHat] = NB_c LASSIFY(D, P, X)$. The input X is an $n \times V$ matrix containing n feature vectors(stored as rows). The output $yHat$ is an $n \times 1$ vector of predicted class labels, where $yHat(i)$ is the predicted label for the i^{th} row of X.(Hint: In this function, you will want to use the $logProd$ function to avoid numerical problems).

```
function [yHat] = NB_Classify(D, p, XTest)
% NP_CLASSIFY Predicts the class labels for some given
% test input.
%
% inputs:
%   D - A [2 x V] matrix, where D(i, j) is an estimate of
%      the probability that word j in a document with class
%      label i-1 (the output of NB_XGivenY).
%   p - A scalar characterizing the prior probability that
%      the class label is 1 (the output of NB_YPrior).
%   XTest - An [m x V] matrix where each row describes
%      which words are present in a particular *test*
```

```
%       document. XTest(i, j) is 1 if word j appears in the
%       i-th document.
%
% output:
%    yHat - An [m x 1] vector of predicted class labels,
%       where yHat(i) is the predicted class label for the
%       i-th document (the i-th row of XTest).
m = size(XTest, 1);
V=size(XTest,2);
% TODO: implement me!
yHat = zeros(m, 1);
for i=1:m
    C0=zeros(V+1,1);
    C1=zeros(V+1,1);
    C0(V+1)=log(p(1));
    C1(V+1)=log(p(2));
    for j=1:V
        if XTest(i,j)==1
            C0(j)=log(D(1,j));
            C1(j)=log(D(2,j));
         else
            C0(j)=log(1-D(1,j));
            C1(j)=log(1-D(2,j));
         end
     end
   if  logProd(C0)>logProd(C1)
       yHat(i)=0;
       else
       yHat(i)=1;
   end
end
end
```

e) Complete the function $[error] = ClassificationError(yHat, yTruth)$, which takes two vectors of equal length and returns the proportion of entries that they agre on.

```
function [error] = ClassificationError(yHat, yTruth)
% CLASSIFICATIONERROR Computes the classification error.
%
% inputs:
%    yHat - An [m x 1] vector of predicted class labels,
%       where yHat(i) is the predicted class label for the
%       i-th document (the i-th row of XTest).
%    yTruth - An [m x 1] vector of the true class labels.
%
% output:
%    error - A scalar containing the proportion of entries
%       that the given vectors disagree on.
% TODO: implement me!
error = 0;
m=size(yTruth,1);
for i=1:m
if yHat(i)~= yTruth(i)
error=error+1;
end
end
error=error/m;
end
```

Questions f) Train your classifier on the data contained in ***XTrain***, and $yTrain$ by running

$D = NB_XGivenY(Xtrain, yTrain)$
$P = NB_YPrior(yTrain)$

Use the learned classifier to predict the labels for the article feature vectors in $\boldsymbol{XTrain}$ and $\boldsymbol{XTest}$ by running

$yHatTrain = NB\_Classify(D, P, XTrain)$
$yHatTest = NB\_Classify(D, P, XTest)$

Use the function $ClassificationError$ to measure and report the training and testing error by running

$trainError = ClassificationError(yHatTrain, yTrain)$
$testError = ClassificationError(yHatTest, yTest)$

(1) How do the train and test errors compare? Explain ang significant difference.
A:
trainError=0, testError=0.0207.

g) Repeat the steps from part f), but this time use the smaller trainning set $XTrainSmall$ and $yTrainSmall$. Explain any difference between the train and test error in the question and in part (f)[Hint: When we have less training data, does the prior have more or less impact on our classifier?]
(1) Explain any differences between the train errors in this questions and in part (f).
(2) Explain any differences between the test errors in this questions and in part (f).
A:
(1) trainErrorSmall=0.0966>0=trainError
(2) testErrorSmall=0.2759>0.0207=testError

h) Finally we will try to interpret the learned parameters. Train your classifier on the data contained in $XTrain$ and $yTrain$. For each class label $y \in 0, 1$, list the six words that the model says are most likely to occur in a document from class y. Also for each class labekl $y \in 0, 1$, list the six words ω that maximize the following quantity:

$$\frac{P(X_\omega = 1|Y = y)}{P(X_\omega = 1|Y \neq y)}$$

```
function [W] = Max8(Vocabulary,D)
W=cell(4,8);
A1=zeros(1,26048);
A2=zeros(1,26048);
[~,A1]=sort(D(1,:));
[~,A2]=sort(D(2,:));
[~,A3]=sort(D(1,:)./D(2,:));
[~,A4]=sort(D(2,:)./D(1,:));
for i=1:1:8
    W(1,i)=Vocabulary(A1(26040+i));
    W(2,i)=Vocabulary(A2(26040+i));
    W(3,i)=Vocabulary(A3(26040+i));
    W(4,i)=Vocabulary(A4(26040+i));
    end
end
```

(1) Give one word in the top eight highest probabilityies $P(X_\omega = 1|Y = 0)$(i.e. words most likely to appear in Economist articles)
A: 'is' 'that' 'a' 'and' 'of' 'in' 'the' 'to'

(2) Give one word in the top eight highest probabilityies $P(X_\omega = 1|Y = 1)$(i.e. words most likely to appear in Onion articles).

A: 'a' 'and' 'the' 'to' 'of' 'said' 'in' 'for'

(3) Give one word in the top eight that maximize the quantity in Euqation for $y = 0$.
A: 'parliamentari' 'neighbour' 'labour' '1990s' 'centr' 'favour' 'reckon' 'organis'

(4) Give one word in the top eight that maximize the quantity in Euqation for $y = 1$.
A: 'favorit' 'tuesday' 'coach' 'realiz' 'percent' 'monday' '5enlarg' '4enlarg'

(5) Which list of words describels the two classes better?
A: The second list of words describes the two classes better.