EECS 491: Artificial Intelligence:Probabilistic Graphical Models Assignment #1

Boning Zhao, bxz213

February 14, 2018

1 Problem Definition

Now we have a set of news(more than ten thousands) which could be divided into tens of categories(like science. Medicine, computer. Graphics). With the set of already known news how could we classify unknown news into a correct group?

I got the data set from http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html. It contains 20000 news which are divided into 20 groups.

2 Method

This problem is actually a classification problem and naive Bayes classifier is just one of highly pratical classify method based on Bayes rules. [2] So let's look at the problem firstly. Consdering that a specific news is consit of a set of words in different locations in the text. For Example: "I love Bayes rule". if this is one news belongs to group computer. So we could treat this news as 'I' in location 0, 'love' in location 1, 'Bayes' in location 2, and 'rule' in location 3. Here we actually transform a news to a word flow. So now we want to find is the probabilities of this words flow belongs to differnt groups. The group with max probability is just the predict answer.Let's denote the location's information as a_n and the group as $news_i$

$$Result_{map} = argmaxP(news_i|a_1, a_2.....a_n) \tag{1}$$

So this is a conditional probability. I apply Bayes rule here and transform the equation to

$$Result_{map} = argmax[P(a_1, a_2.....a_n|news_i)P(news_i)/P(a_1, a_2.....a_n)] \tag{2}$$

We could notice that $P(a_1, a_2.....a_n)$ is a constant$(1/news'sum)$ So we now just need to consider:

$$Result_{map} = argmaxP(a_1, a_2.....a_n|news_i)P(news_i) \tag{3}$$

Look at the $P(a_1, a_2.....a_n|news_i)$, in order to calculate it we need to introduce an assumtion here.

2.1 assumption 1

We know that a word's probability to occur in a specific position is actually affected by what the prior is. For example, if the prior is word 'artificial' and the next word will have greater probability to be 'intelligence'. But if we take this into consideration, it will be very complex. Here the assumption comes: the locations' value are conditionally independent goven the target value, which means

$P(a_1, a_2.....a_n|news_i) = \prod P(a_n|news_i)$. Actually this does not affect much of the function of the classifer, which has been proved by Pedro et al. [1]
So now we transform the euqtion to:

$$Result_{map} = argmaxP(news_i) \prod P(a_n|news_i) \tag{4}$$

Thus if we could calculate $P(news_i)$ and $\prod P(a_n|news_i)$.We could got what we want. The prior is easy to find, which is the ratio of the group in all groups. But for the later, a problem comes up. We need the numer of groups, the sum of locations and sum of words to calculate it. But it could be a huge work. For example if we have 5 groups, 50000 words and 20 locations, we must calculate $5x50000x20 = 5000000$ times in training data.

2.2 assumption 2

In order to solve the problem before, here we need another assumption: a specific word k's probability is independent with the locations, which means:$P(a_j = k|news_i) = P(a_p = k|news_i)$. So now for the same condition mention before, we just need to calculate $2x50000$ times. One advance of this assumption is it increases the number of examples available to estimate each of the required probabilities,therefore increasing the reliability of the estimates. Then I aplly a m-estimate of probability here with uniform priors and with m equal to the size of the word vocabulary. So $P(a_p = k|news_i)$ will be:

$$n_k + 1/n + |Vocabulary| \tag{5}$$

where n is the total nmber of word locations in all training examples whose taget is $news_i$; n_k is the number of times word k is found among these n word positions, and —Vocabulary— is the total number of distinct words.

3 Implementation

I use Java to implement this algorithm.

3.1 prework

I import Je analyser and lucene for word segmentation and built up the word flow to collect information of vocabulary and word.

```
Vector<String> readFile(String fileName) throws IOException, FileNotFoundException{
    File f=new File(fileName);
    InputStreamReader isr=new InputStreamReader(new FileInputStream(f),"GBK");
    char[] cbuf=new char[(int) f.length()];
    isr.read(cbuf);
    Analyzer analyzer=new MMAnalyzer();
    TokenStream tokens=analyzer.tokenStream("Contents", new StringReader(new String(cbuf)));
    Token token=null;
    Vector<String> textList=new Vector<String>();
    while((token=tokens.next(new Token()))!=null){
        textList.add(token.term());
    }
    return textList;
}
```

Figure 1:

The code above just made all the words in the list into a String vector except symbols like comma.
In order to save the information, I create two classes: the first is the WordInfor class:

```java
class WordInfor{//This class for word information, including frequency and number
    double fre;//words frequency which should be calculated after the vocabulary has been set
    double count;//number of words
    public WordInfor(double count) {
        this.fre=-1;
        this.count=count;
    }
    public void setFrequency(double frequency){
        this.fre=frequency;
    }
}
```

Figure 2:

It is used to store information including count and frequency of a specific word.Because we could only calculate the frequency after the vocabulary is set up, thus this attribute will has a default value of -1 when initialization.

Another class is label, it contaions information of words sum of a specific label, text sum and a hash map for mapping the word with its information.

```java
class Label{//For Text's label: computer, social
    //map for storing every single word and its calculation
    Map<String,WordInfor> map=new HashMap<String,WordInfor>();
    double wordCount;//The sum of words of one specific label
    double textCount;//The sum of texts of one label
    public Label() {
        this.map=null;
        this.wordCount=-1;
        this.textCount=-1;
    }
    public void set(Map<String,WordInfor> m,double wordCount,double documentCou
        this.map=m;
        this.wordCount=wordCount;
        this.textCount=documentCount;
    }
}
```

Figure 3:

3.2 train

In order to get the correct Bayes learning set, the first job is to got all the words in one specific label in to a set. then made a sort of it for letting the same word together. Then compare the word from begining got the count of word and add it to vocabulary. Finally use a hash map here to connet the label with word information. Finally using m-estimate to calculate the frequency($P(a_p = k|news_i)$) of words(there also using a logarithm to treat the frequency value, which is in order to map the process in test. This function's aim will be explained in next section). The code is shown below:

```java
public void train() {
    long startTrainTime=System.currentTimeMillis();
    File folder=new File(trainingFilePath);
    labelsName=folder.list();
    labelsList=new Vector<Label>();
    for(int i=0;i<labelsName.length;i++){
        labelsList.add(new Label());
        File subFolderList=new File(trainingFilePath+"\\"+labelsName[i]);
        String[] filesList=subFolderList.list();
        System.out.println("Now processing:"+labelsName[i]);
        GUI.setTextArea("Now processing:"+labelsName[i]);
        Vector<String> text=new Vector<String>();
        for(int j=0;j<filesList.length;j++){
            //System.out.print(files[j]+" ");
            try {
                text.addAll(readFile(trainingFilePath+"\\"+labelsName[i]+"\\"+filesList[j]));
            } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        //put all word from current label to the set
        //get the size of vocabulary
        //sort for all words.
        //in order to calculate the word information
        String[] allWords=new String[text.size()];
        for(int j=0;j<text.size();j++)
            allWords[j]=text.elementAt(j);
        sort(allWords,0,text.size()-1);

        String previous=allWords[0];
        double count=1;
        Map<String,WordInfor> trainMap=new HashMap<String, WordInfor>();
        for(int j=1;j<allWords.length;j++){
            if(allWords[j].equals(previous))
                count++;
            else{
                vocabulary.add(previous);
                trainMap.put(previous, new WordInfor(count));
             previous=allWords[j];
             count=1;
             }
       }
       labelsList.elementAt(i).set(trainMap, text.size(),filesList.length);
       long endTrainTime=System.currentTimeMillis();
       trainingTime=endTrainTime-startTrainTime;
   }
   //Got the size of vocabulary, then calculate the frequency
   for(int i=0;i<labelsList.size();i++){
       Iterator iter=labelsList.elementAt(i).map.entrySet().iterator();
       while(iter.hasNext()){
           Map.Entry<String, WordInfor> entry=(Map.Entry<String, WordInfor>)iter.next();
           WordInfor item=entry.getValue();
           item.setFrequency(Math.log10((item.count+1)/(labelsList.elementAt(i).wordCount+vocabulary.size())));//using m-estimate here
       }
   }
}
```

Figure 4:

3.3 test

Testing section is somewhat similar with train.Getting the word flow first and then to compare every word in every label's vocabulary, if exists then apply the frequency of that word, if not the use the m-estimate function to calculate the frequency.In order to decrease the answer size, I apply a logrithm here which all change the product to the sum, the code is shown below:

```java
public void test(){
    long startTestTime=System.currentTimeMillis();
    File folder=new File(testingFilePath);
    String []ln;
    ln=folder.list();
    for(int x=0;x<ln.length;x++){
    Vector<String> v=null;
    try {
        v = readFile(testingFilePath+"\\"+ln[x]);
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    double values[]=new double[labelsName.length];
    for(int i=0;i<labelsList.size();i++){
        double tempValue=labelsList.elementAt(i).textCount;
        for(int j=0;j<v.size();j++){
            if(labelsList.elementAt(i).map.containsKey(v.elementAt(j))){
                tempValue+=labelsList.elementAt(i).map.get(v.elementAt(j)).fre;
            }else{
                tempValue+=Math.log10(1/(double)(labelsList.elementAt(i).wordCount+vocabulary.size()));
            }
        }
        values[i]=tempValue;
    }
    //for(int i=0;i<values.length;i++)
        //System.out.println(labelsName[i]+"'s probability is"+values[i]);
    int maxIndex=findMax(values);
    System.out.println(testingFilePath+" belongs to "+labelsName[maxIndex]);
    GUI.setTextArea(testingFilePath+" belongs to "+labelsName[maxIndex]);
    labelofnews=labelsName[maxIndex];
}
    long endTestTime=System.currentTimeMillis();
    testingTime=endTestTime-startTestTime;
}
```

Figure 5:

3.4 UI

I made a simple user interface with Jpanel which contains title, button to train, button to test and a background with Mr.Bayes's photo.The code is shown below(I don't include the button for test because it is similar with the button for train):

```java
    ((JPanel)this.getContentPane()).setOpaque(false);
    ImageIcon img = new ImageIcon
            ("C:\\Users\\13269\\Desktop\\Java\\Bayes\\Bayes.jpg");
    JLabel background = new JLabel(img);
    this.getLayeredPane().add(background, new Integer(Integer.MIN_VALUE));
    background.setBounds(0,0,img.getIconWidth(),img.getIconHeight());

getContentPane().setLayout(null);
setSize(new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT));



final JButton button0 = new JButton();
button0.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        JFileChooser chooser = new JFileChooser();
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        int   n = chooser.showOpenDialog(getContentPane());
        if(n == JFileChooser.APPROVE_OPTION){
            bayes.setTrainPath(chooser.getSelectedFile().getPath());
            flagT=true;
            run();
        }
    }
});

button0.setText("Train");
button0.setBounds(93, 64, 106, 28);
getContentPane().add(button0);
this.getContentPane().setBackground(Color.yellow);

JTextField text = new JTextField();
text.setText("Bayes Classify(╯'□')╯︵┻┻");
text.setBounds(180,30,190,30);
getContentPane().add(text);
```

Figure 6:

The UI is like this:

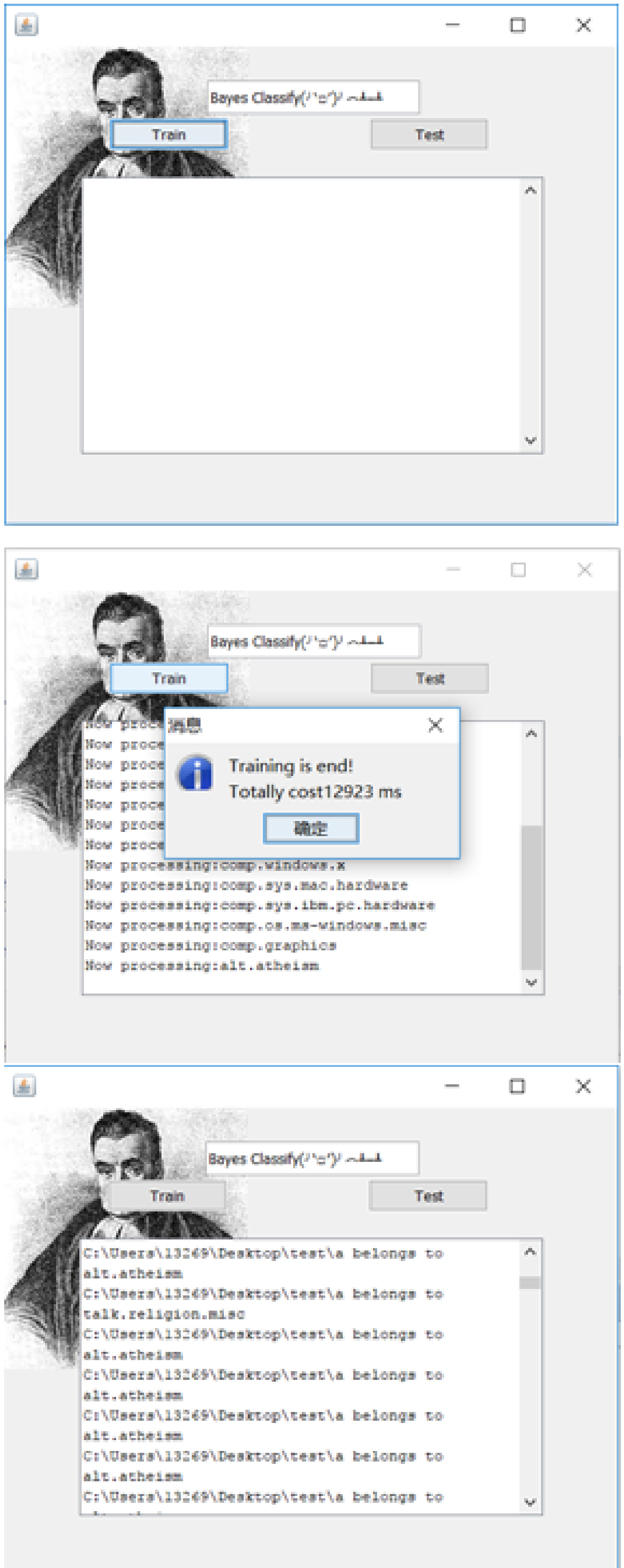
Bayes Classify
Train
Test
Bayes Classify
Train
Test
Training is end!
Totally cost12923 ms
确定
Now processing:comp.windows.x
Now processing:comp.sys.mac.hardware
Now processing:comp.sys.ibm.pc.hardware
Now processing:comp.os.ms-windows.misc
Now processing:comp.graphics
Now processing:alt.atheism
Bayes Classify
Train
Test
C:\Users\13269\Desktop\test\a belongs to
alt.atheism
C:\Users\13269\Desktop\test\a belongs to
talk.religion.misc
C:\Users\13269\Desktop\test\a belongs to
alt.atheism
C:\Users\13269\Desktop\test\a belongs to
alt.atheism
C:\Users\13269\Desktop\test\a belongs to
alt.atheism
C:\Users\13269\Desktop\test\a belongs to
alt.atheism
C:\Users\13269\Desktop\test\a belongs to

Figure 7:

4 Results

I used 13400 news for training and 6600 for testing. In all the testing document, 5610 has been correctly classified. The accurracy is $5610/6600 = 85$

References

[1] P. Domingos and M. Pazzani. Simple bayesian classifiers do not assume independence. 1996.

[2] T. Mitchell and M. Hill. *Machine Learning*. March 1,1997.