**人工智能 – 朴素贝叶斯分类器**

1. 学习部分：

计算公式：

n 是在类别vj中词出现的位置总数（或者说词频总数）

nk是词wk出现的位置总数（或者说词频）

1. 分类过程：

目标函数为：



1. 实验结果

|  |
| --- |
| 训练数据 |
| No Sunny Hot High Weak No Sunny Hot High Strong Yes Overcast Hot High Weak  Yes Rain Mild High Weak Yes Rain Cool Normal Weak No Rain Cool Normal Strong  Yes Overcast Cool Normal Strong No Sunny Mild High Weak Yes Sunny Cool Normal Weak  Yes Rain Mild Normal Weak Yes Sunny Mild Normal Strong Yes Overcast Mild High Strong  Yes Overcast Hot Normal Weak |
| 测试数据 |
| Rain Mild High Strong |

训练结果：

|  |
| --- |
| wk,vj: Sunny,No: 0.06557377049180328  wk,vj: Hot,No: 0.04918032786885246  wk,vj: High,No: 0.06557377049180328  wk,vj: Weak,No: 0.04918032786885246  wk,vj: Strong,No: 0.04918032786885246  wk,vj: Overcast,No: 0.018032786885245903  wk,vj: Rain,No: 0.03278688524590164  wk,vj: Mild,No: 0.03278688524590164  wk,vj: Cool,No: 0.03278688524590164  wk,vj: Normal,No: 0.03278688524590164  wk,vj: Sunny,Yes: 0.04838709677419355  wk,vj: Hot,Yes: 0.04838709677419355  wk,vj: High,Yes: 0.06451612903225806  wk,vj: Weak,Yes: 0.11290322580645161  wk,vj: Strong,Yes: 0.06451612903225806  wk,vj: Overcast,Yes: 0.08064516129032258  wk,vj: Rain,Yes: 0.06451612903225806  wk,vj: Mild,Yes: 0.08064516129032258  wk,vj: Cool,Yes: 0.06451612903225806  wk,vj: Normal,Yes: 0.11290322580645161 |

测试结果：

percent:1.3921873847094822E-5 VNB:Yes

代码：

|  |
| --- |
| package machineLearning;  import java.io.BufferedReader;  import java.io.FileReader;  import java.util.ArrayList;  import java.util.HashSet;  import java.util.Hashtable;  import java.util.Iterator;  public class naiveBayes {  public static final String train\_input = "d://examples.txt";  public static final String classify\_input = "d://classify.txt";  public ArrayList<String> vocabulary;  public ArrayList<texts> examples;  public ArrayList<String> V;  public Hashtable<String, Double> P;  ArrayList<texts> doc;  public naiveBayes() {  P = new Hashtable<String, Double>();  V = new ArrayList<String>();  vocabulary = new ArrayList<String>();  examples = new ArrayList<texts>();  doc = new ArrayList<texts>();  }  public void testInput() {  try {  BufferedReader br = new BufferedReader(new FileReader(  classify\_input));  String lines = null;  while ((lines = br.readLine()) != null) {  String[] items = lines.split(" ");  doc.add(new texts(items));  }  } catch (Exception ex) {  ex.printStackTrace();  }  }  public void formatInput() {  try {  BufferedReader br = new BufferedReader(new FileReader(train\_input));  String lines = null;  while ((lines = br.readLine()) != null) {  String[] items = lines.split(" ");  String word = items[0].trim();  if (!V.contains(word) && word != null)  V.add(word);  examples.add(new texts(items));  }  br.close();  } catch (Exception ex) {  ex.printStackTrace();  }  }  public void learn\_text(ArrayList<texts> Example, ArrayList<String> v) {  for (texts txt : Example) {  for (String s : txt.items) {  vocabulary.add(s);  }  }  for (int i = 0; i < v.size(); i++) {  String vj = v.get(i);  if (vj.length() > 0) {  ArrayList<String> docsj = new ArrayList<String>();  int count = 0;  for (texts txt : Example) {  if (txt.aim.equals(vj)) {  count++;  for (String s : txt.items)  docsj.add(s);  }  }  P.put(vj, count \* 1.0 / Example.size());  Hashtable<String, Integer> table = new Hashtable<String, Integer>();  Iterator<String> it\_doc = docsj.iterator();  int n = 0;  while (it\_doc.hasNext()) {  String s = it\_doc.next();  if (!table.containsKey(s)) {  table.put(s, 1);  n++;  } else {  table.put(s, table.get(s) + 1);  }  }  Iterator<String> it\_voca = vocabulary.iterator();  while (it\_voca.hasNext()) {  String wk = it\_voca.next();  if (wk.length() > 0) {  double nk = 0.1;  if (table.get(wk) != null) {  nk = table.get(wk);  }  System.out.println("wk,vj: " + wk + "," + vj + ": "  + (nk + 1.0) / (n + vocabulary.size()));  P.put(wk + "," + vj,  (nk + 1.0) / (n + vocabulary.size()));  }  }  }  }  }  public void classify\_text(ArrayList<texts> Doc) {  Iterator<texts> it = Doc.iterator();  while (it.hasNext()) {  texts txt = it.next();  Iterator<String> its = V.iterator();  double maxn = 0.0;  String Vnb = "result";  while (its.hasNext()) {  String vj = its.next();  if (vj.length() > 0) {  double sum = 1;  if (P.get(vj) != null)  sum = P.get(vj);  for (int i = 0; i < txt.items.size(); i++) {  String ai = txt.items.get(i);  if (vocabulary.contains(ai)) {  if(P.get(ai + "," + vj)!=null){  sum \*= P.get(ai + "," + vj);  }  }  }  if (maxn <= sum) {  maxn = sum;  Vnb = vj;  }  }  }  System.out.println("maxn:" + maxn + " VNB:" + Vnb);  }  }  public static void main(String[] args) {  naiveBayes bys = new naiveBayes();  bys.formatInput();  bys.learn\_text(bys.examples, bys.V);  bys.testInput();  bys.classify\_text(bys.doc);  }  } |

|  |
| --- |
| package machineLearning;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.HashSet;  public class texts {  public String aim;  public ArrayList<String> items;  public texts(String[] data){  items=new ArrayList<>();  aim=data[0].trim();  for(int i=1;i<data.length;i++){  items.add(data[i].trim());  }  }  } |



44