朱文韬 1600012785

计算概论（实验班）

2016年11月8日

首先按照公式写两个评价指标Purity&F值的计算函数。

numcate,numClu分别存放每个Category和每个Cluster所含元素个数

prec表示(i,r)对的Precision值，rec表示(i,r)对的Recall值

再对Purity取平均得AveragePurity

```
242  void CountIndex()
243  {
244      for(int i = 0 ; i < numTexts; i ++)
245      {
246          numcate[texts[i].categories]++;
247          numClu[texts[i].cluster]++;
248      }
249      //For(i,0,numCategories-1) printf("%d\n", numcate[i]);
250  }
251
252
253  void CalcF()
254  {
255      CountIndex();
256      int tmp,ori,cur;
257      double rec, prec, max1, F, tmpp, ans;;
258      int ni,nr;
259      F = 0;
260      ans = 0;
261      for (int i=0; i<numClusters; i++)  //按每个类
262      {
263          tmp = 0;
264          memset(Pcount,0,sizeof(Pcount));
265          for (int k=0; k<clusters[i].elementsCount; k++) //所包含的文档编号
266          {
267              cur = clusters[i].elements[k];
268              Pcount[texts[cur].categories]++;
269              N[texts[cur].categories][i]++;
270
271          }
272              for (int j=0; j<numCategories; j++)
273              if(Pcount[j]>tmp)
274              {
275                  tmp = Pcount[j];
276                  ori = j;
277              }
278
279          Pur[i] = tmp/(double)numcate[ori];
280          ans+=Pur[i];
281          //printf("TP=%d,original=%d,times=%d,Purity =%lf\n", i, ori, tmp, Pur[i]);
294                  tmpp = ((2*rec*prec)/(prec+rec));
295                  if(tmpp>max1) max1 = tmpp;
296            }
297            if(max1!=-1) F += ni*max1;
298
299      }
300      //printf("F = %lf\n", F);
301      F = F/(double)numTexts;
302      ans = ans/(double)numClusters;
303      printf("F = %lf, Average Purity = %lf\n", F, ans);
304
305  }
```

测试数据进行100次随机取质心测试结果（部分）：

```
F = 0.748359, Average Purity = 0.700000
F = 0.596229, Average Purity = 0.484615
F = 0.540260, Average Purity = 0.428571
F = 0.722098, Average Purity = 0.671429
F = 0.780478, Average Purity = 0.671429
F = 0.739615, Average Purity = 0.707317
F = 0.882401, Average Purity = 0.814286
F = 0.792052, Average Purity = 0.571429
F = 0.626716, Average Purity = 0.571429
F = 0.707831, Average Purity = 0.571429
F = 0.718504, Average Purity = 0.634615
F = 0.727846, Average Purity = 0.685714
F = 0.643226, Average Purity = 0.498901
F = 0.810022, Average Purity = 0.814286
F = 0.774874, Average Purity = 0.714286
F = 0.636777, Average Purity = 0.428571
F = 0.852187, Average Purity = 0.664460
F = 0.852187, Average Purity = 0.664460
F = 0.751561, Average Purity = 0.653746
F = 0.751561, Average Purity = 0.653746
```

对100次结果求平均值，**F = 0.711491, Average Purity = 0.595555**

调整1:调整距离计算公式

Block距离

$$\sum_{i=1}^{p}\left|x_i - y_i\right|$$

F = 0.688567, Average Purity = 0.461377

```
    double dist = 0;

    for (int i=0; i<numFeatures-1; i++)
    {
        dist += (abs(features1[i] - features2[i]));
    }
    return dist;
}
```

Minkovski距离

$$\sqrt[q]{\sum_{i=1}^{p}\left|x_i - y_i\right|^q}$$

q = 3 **F = 0.660712, Average Purity = 0.465940**

q = 4 **F = 0.714800, Average Purity = 0.585549**

q = 7 **F = 0.698908, Average Purity = 0.500682**

调整2:调整随机策略

“躲避”策略

```
void Calcdis()
{
    For(i,0,numTexts-1)
    For(j,0,numTexts-1)
    dis[i][j] = getSquaredEuclideanDist(texts[i].feature, texts[j].feature);
//  For(i,0,numTexts-1) For(j,0,numTexts-1) printf("%lf\n", dis[i][j]);
}

//从n篇文档(文档号：0~n-1)中随机选择k个(不重复的数)作为初始聚类种子
void selectRandomSeeds(int n, int k, int seeds[])
{
    Calcdis();
    srand(time(NULL));   //生成伪随机数种子
    int r = rand() % n;   //生成 0到index-1之间的一个随机数
    int max2;
    seeds[0] = r;
    For(i,0,numTexts-1) tot[i] += dis[r][i];
    tot[r] = -1;
//  printf("Random Seeds: ");
    for (int i=1; i<k; i++)
    {
        max2 = 0;
        For(j,0,numTexts-1)
        if(tot[j]>max2)
        {
            max2 = tot[j];
            r = j;
        }
        seeds[i] = r;
        For(j,0,numTexts-1) tot[j] += dis[r][j];
        tot[r] = -1;
    }
//  For(i,0,k-1) printf("Seed%d:%d\n", i, seeds[i]);

}
```

先随机一个初始中心，然后取出到该点距离最小的点作为第二个，再取出到前两个点距离和最小的点作为第三个……以此类推

F = 0.810532,

Average Purity = 0.672344

“躲避”策略的思考 — 取出到前n个点距离积最小的点作为第(n+1)个点

加强较近距离的影响，使初始点更分散（几何平均代替算术平均）

F = 0.836144, Average Purity = 0.732039

用平方平均代替算术平均时，评价指标明显下降（较近距离影响被减弱）

F = 0.755942, Average Purity = 0.585127

用调和平均代替算术平均时，由于较大、较小的距离都能得到合理的反馈，评价指标依旧较高

F = 0.816975, Average Purity = 0.707677

总结

聚类算法的两个主要评价指标Purity和F-Value可以较好地反映聚类的准确程度。本文主要探索了调整距离计算公式和调整随机策略对聚类算法准确率的影响，前者本质是通过不同的距离计算方式改变归类，后者本质是通过改变初始选取点影响算法的准确率。

实验表明，对于题目数据的纯01串，简单修改EuclideanD距离公式为Block距离或Minkovski距离对评价指标的提高没有明显作用。

在初始点选取过程中，我们的目标是使初始点彼此远离，避免距离过近产生重叠。为此，自然产生了朴素的贪心算法——“躲避”法，使后续质心的选取受到前面质心选取的影响。在定量表出这种影响时，应该尽量加大较小距离对“躲避”程度的影响，即使两个相近点之间的“排斥力”急剧增大。在此思想指导下对距离的四种平均值表达方式进行了试验，得到的实验指标符合预期。

进一步的可以考虑层次聚类以改进聚类算法的准确度，由于时间原因没有进行代码实现。