**THU-70250043 Pattern Recognition (Spring 2018) Homework: 5**

稀疏学习，距离度量，最近邻方法

*Lecturer: Changshui Zhang* zcs@mail.tsinghua.edu.cn

*Student:聂浩 2017312153*

1.证明利用欧式距离的最近邻规则将空间划分成的区域（Voronoi网格）是凸的。

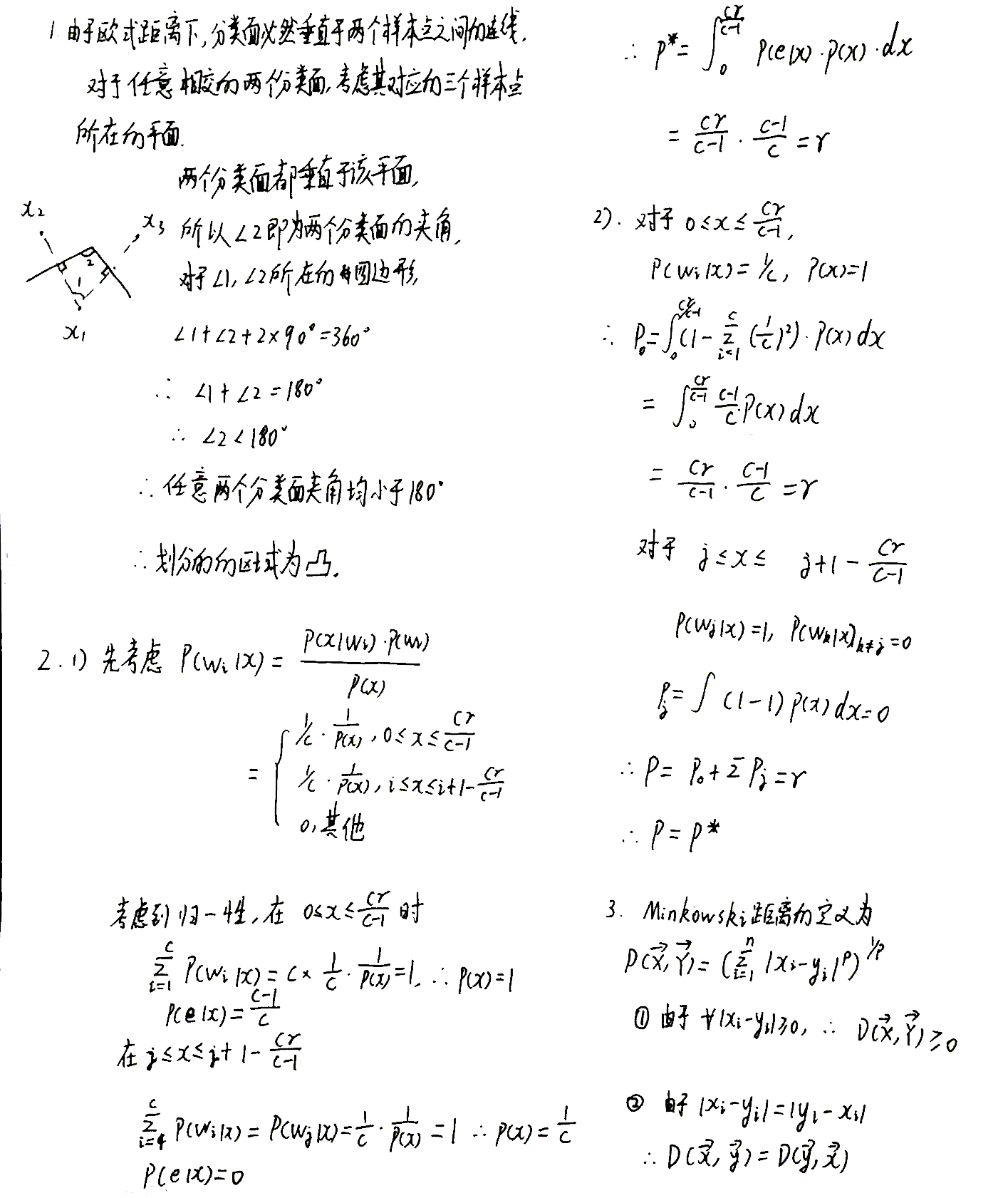
2.假设数据，其类别的先验概率为，且有：

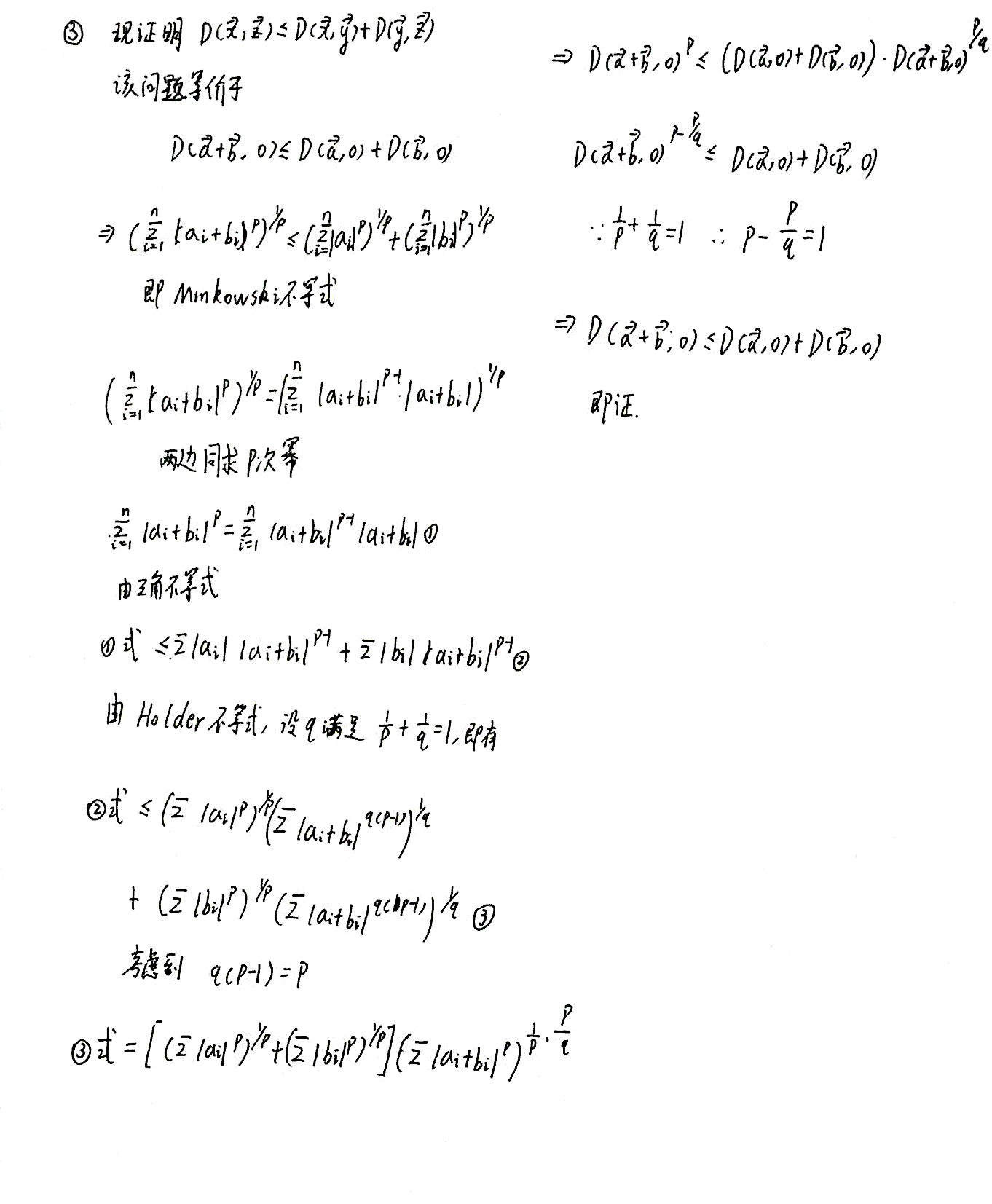
其中，证明：

1）贝叶斯误差率为：；

2）最近邻规则的误差率等于贝叶斯误差率。

3.证明Minkowski距离是一个距离度量。





4.编程实现最近邻及K近邻方法，在MNIST数据集上测试，并撰写实验报告。

要求：

1）使用不同规模的训练样本，比较最近邻分类器的性能变化，包括正确率，时间和空间复杂度等；

这里使用1000个测试集（前1000个）进行实验

|  |  |  |  |
| --- | --- | --- | --- |
| 训练集大小 | 错误率 | 运行时间 | 内存使用峰值 |
| 5000 | 9.8% | 14.068s | 38.04MB |
| 10000 | 8.0% | 20.346s | 69.34MB |
| 20000 | 7.0% | 40.022s | 102.49MB |

代码如下

|  |
| --- |
| clear**;**close all**;**clc**;**  profile on **-**memory  trainsize**=**15000**;**  testsize**=**1000**;**  trainData**=**loadimg**(**'train-images.idx3-ubyte'**,**trainsize**);**  trainLabel**=**loadlabel**(**'train-labels.idx1-ubyte'**,**trainsize**);**  testData**=**loadimg**(**'t10k-images.idx3-ubyte'**,**testsize**);**  testLabel**=**loadlabel**(**'t10k-labels.idx1-ubyte'**,**testsize**);**  trainL**=**length**(**trainLabel**);**  testL**=**length**(**testLabel**);**  testResult**=**zeros**(**testL**);**  tic**;**  **for** count**=**1**:**testL  tmpImg**=**repmat**(**testData**(:,:,**count**),**1**,**1**,**trainL**);**  tmpImg**=(**trainData**-**tmpImg**).^**2**;**  comp**=**sum**(**sum**(**tmpImg**));**  **[**m**,**Index**]=**min**(**comp**);**  testResult**(**count**)=**trainLabel**(**Index**);**  **end**  toc**;**  profile viewer**;**  error**=**0**;**  **for** i**=**1**:**testL  **if** **(**testResult**(**i**)** **~=** testLabel**(**i**))**  error**=**error**+**1**;**  **end**  **end** |

2）使用不同的k值，分析对性能的影响；

使用1000个测试集（前1000个），10000个训练集进行实验

|  |  |  |  |
| --- | --- | --- | --- |
| k大小 | 错误率 | 运行时间 | 内存使用峰值 |
| 5 | 8.4% | 35.36s | 69.30MB |
| 20 | 9.3% | 37.68s | 68.20MB |
| 50 | 12.3% | 35.46s | 69.41MB |

代码如下

|  |
| --- |
| clear;close all;clc;  profile on -memory  trainsize=10000;  testsize=1000;  k=5;  trainData=loadimg('train-images.idx3-ubyte',trainsize);  trainLabel=loadlabel('train-labels.idx1-ubyte',trainsize);  testData=loadimg('t10k-images.idx3-ubyte',testsize);  testLabel=loadlabel('t10k-labels.idx1-ubyte',testsize);  trainL=length(trainLabel);  testL=length(testLabel);  testResult=zeros(testL);  leastk=zeros(1,k);  tic;  for count=1:testL  tmpImg=repmat(testData(:,:,count),1,1,trainL);  tmpImg=(trainData-tmpImg).^2;  comp=sum(sum(tmpImg));  [m,Index]=sort(comp);  leastk=trainLabel(Index(1:k));  testResult(count)=mode(leastk);  end  toc;  profile viewer;  error=0;  for i=1:testL  if (testResult(i) ~= testLabel(i))  error=error+1;  end  end |

3）使用不同的距离度量，分析对性能的影响；

使用1000个测试集（前1000个），5000个训练集进行实验

|  |  |  |  |
| --- | --- | --- | --- |
| 距离度量 | 错误率 | 运行时间 | 内存使用峰值 |
| 切比雪夫距离 | 39.8% | 15.16s | 38.16MB |
| 欧氏距离 | 9.8% | 11.03s | 30.63MB |
| 3阶马氏距离 | 8.4% | 53.84s | 37.4MB |

各种距离对应的代码如下

|  |
| --- |
| %%欧氏距离  %tmpImg=(trainData-tmpImg).^2;  %comp=sum(sum(tmpImg));  %%切比雪夫距离  %tmpImg=abs(trainData-tmpImg);  %comp=max(tmpImg);  %%3阶Minkovski  %tmpImg=(abs(trainData-tmpImg)/255).^3;  %comp=(sum(tmpImg)); |

4）是否存在一组不全为0的系数，使得数据经过的变换之后，使用最近邻分类器的效果得到提升？如果存在设计一组。

利用PAC主成分分析的方法，得到相应矩阵，分别提取测试数据和训练数据的主成分，之后再进行比较，这里使用1000组测试数据和10000组训练数据。可以看出效果要好于提取前。

|  |  |  |  |
| --- | --- | --- | --- |
| 距离度量 | 错误率 | 运行时间 | 内存使用峰值 |
| 欧氏距离 | 7.0% | 8.545s | 61.17MB |

其中，PAC映射矩阵的生成代码如下（pac.m）：

|  |
| --- |
| trainsize**=**10000**;**  trainData**=**loadimg**(**'train-images.idx3-ubyte'**,**trainsize**);**  trainData**=**trainData**-**repmat**(**mean**(**trainData**,**2**),**1**,**trainsize**);**  su**=**zeros**(**784**,**784**);**  **for** c**=**1**:**trainsize  su**=**su**+**trainData**(:,**c**)\***trainData**(:,**c**)';**  **end**  su**=**su**./**trainsize**;**  **[**U**,**S**,**V**]=**svd**(**su**);**  **[**re**,**I**]=**sort**(**diag**(**S**),**'descend'**);**  Utran**=**U**(:,**I**(**re**>**1000**));**  save**(**'pac.mat'**,**'Utran'**);** |

PAC映射及求距离的代码如下(nn\_pac.m)

|  |
| --- |
| clear**;**close all**;**clc**;**  profile off**;**  profile on **-**memory  trainsize**=**10000**;**  testsize**=**1000**;**  trainData**=**loadimg**(**'train-images.idx3-ubyte'**,**trainsize**);**  trainLabel**=**loadlabel**(**'train-labels.idx1-ubyte'**,**trainsize**);**  testData**=**loadimg**(**'t10k-images.idx3-ubyte'**,**testsize**);**  testLabel**=**loadlabel**(**'t10k-labels.idx1-ubyte'**,**testsize**);**  trainL**=**length**(**trainLabel**);**  testL**=**length**(**testLabel**);**  testResult**=**zeros**(**1**,**testL**);**  load pac**.**mat**;**  trainData**=**Utran**'\***reshape**(**trainData**,**28**\***28**,**trainL**);**  testData**=**Utran**'\***reshape**(**testData**,**28**\***28**,**testL**);**  tic**;**  **for** count**=**1**:**testL  disp**(**count**);**  tmpImg**=**repmat**(**testData**(:,**count**),**1**,**trainL**);**  %%欧氏距离  tmpImg**=(**trainData**-**tmpImg**).^**2**;**  comp**=(**sum**(**tmpImg**));**  **[**m**,**Index**]=**min**(**comp**);**  testResult**(**count**)=**trainLabel**(**Index**);**  **end**  toc**;**  profile viewer**;**  error**=**0**;**  **for** i**=**1**:**testL  **if** **(**testResult**(**i**)** **~=** testLabel**(**i**))**  error**=**error**+**1**;**  **end**  **end** |

5）在最近邻分类器中，设计切线距离代替欧氏距离，叙述计算方法，并比较MNIST上分类器性能的变化。

这里使用了开源的tangent distance的C(见distance.c文件)实现，编译为mex来进行调用。其中使用到的切线模式有x平移、y平移、缩放、旋转。由于该方法消耗时间长，仅测试300个测试数据，使用5000组训练数据。

在仅使用缩放切线和旋转切线求解时，错误率为7%，耗时127s，在仅使用x平移和y平移时，错误率为4%，耗时133.4s。可见x平移和y平移这种最简单的切线函数的计算量更小。

调用代码为：

|  |
| --- |
| for k=1:trainL  comp(k)=distance(trainData(:,:,k)/255,testData(:,:,count)/255);  end |

distance中接口为

|  |
| --- |
| void mexFunction**(**int nlhs**,** mxArray **\***plhs**[],** int nrhs**,** const mxArray **\***prhs**[])**  **{**  plhs**[**0**]** **=** mxCreateDoubleMatrix**(**1**,** 1**,**mxREAL**);**  double **\***outData **=** mxGetPr**(**plhs**[**0**]);**  outData**[**0**]** **=** 1**;**  //获取输入变量的数值大小  //获取输出变量的指针  **if** **(**nrhs **==** 2**){**  int m **=** mxGetM**(**prhs**[**0**]);**  int n **=** mxGetN**(**prhs**[**0**]);**    outData**[**0**]** **=** distance**(**mxGetPr**(**prhs**[**0**]),**mxGetPr**(**prhs**[**1**]));**    **}else**  **{**  outData**[**0**]** **=** 0**;**  **}**  **}** |

以下为MNIST的解析函数：

图片读取loadimg.m

|  |
| --- |
| **function** **[** images **]** **=** loadimg **(**filename**,**imsize**)**  fp **=** fopen**(**filename**,**'rb'**);**  magic**=**fread**(**fp**,**1**,**'int32'**,**0**,**'ieee-be'**);**  nimg**=**fread**(**fp**,**1**,**'int32'**,**0**,**'ieee-be'**);**  Row**=**fread**(**fp**,**1**,**'int32'**,**0**,**'ieee-be'**);**  Col**=**fread**(**fp**,**1**,**'int32'**,**0**,**'ieee-be'**);**  images**=**fread**(**fp**,**Col**\***Row**\***imsize**,**'unsigned char'**);**  images **=** reshape**(**images**,** Col**,** Row**,** imsize**);**  images **=** permute**(**images**,[**2 1 3**]);**%roate  fclose**(**fp**);**  %images=reshape(images,Col\*Row,imsize);  **end** |

Label读取loadlabel.m

|  |
| --- |
| **function** **[** label **]** **=** loadlabel **(**filename**,**lsize**)**  fp **=** fopen**(**filename**,**'rb'**);**  magic**=**fread**(**fp**,**1**,**'int32'**,**0**,**'ieee-be'**);**  nlabels **=** fread**(**fp**,** 1**,** 'int32'**,** 0**,** 'ieee-be'**);**  label **=** fread**(**fp**,** lsize**,** 'unsigned char'**);**  fclose**(**fp**);**  %label=label(1:lsize);  **end** |