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$x \in \mathbb{R}$，其类别w_i的先验概率为$P(w_i) = \frac{1}{c}$，$i = 1,2,\ldots,c$，且有：

$$p(x|w_i) = \begin{cases} 1, & 0 \le x \le \dfrac{cr}{c-1} \\ 1, & i \le x \le i + 1 - \dfrac{cr}{c-1} \\ 0, & \textit{其他} \end{cases}$$

其中$0 < r < \frac{c-1}{c}$，证明：

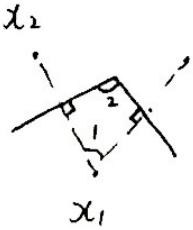
1）贝叶斯误差率为：$P^* = r$；

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。

1. 由于欧式距离下，分类面必然垂直于两个样本点之间的连线，

对于任意相交的两个分类面，考虑其对应的三个样本点所在的平面。



两个分类面都垂直于该平面，

所以 $\angle 2$ 即为两个分类面的夹角，

对于 $\angle 1$，$\angle 2$ 所在的四边形，

$\angle 1+\angle 2+2\times 90^\circ=360^\circ$

$\therefore \angle 1+\angle 2=180^\circ$

$\therefore \angle 2<180^\circ$

$\therefore$ 任意两个分类面夹角均小于 180°

$\therefore$ 划分的分区对为凸。

2. 1) 先考虑 $P(w_i|x)=\dfrac{P(x|w_i)\cdot P(w_i)}{P(x)}$

$$=\begin{cases}\frac{1}{c}\cdot\frac{1}{P(x)}, & 0\le x\le \frac{cr}{c-1}\\ \frac{1}{c}\cdot\frac{1}{P(x)}, & i\le x\le i+1-\frac{cr}{c-1}\\ 0, & \text{其他}\end{cases}$$

考虑到归一性，在 $0\le x\le\frac{cr}{c-1}$ 时

$\sum_{i=1}^{c} P(w_i|x)=c\times\frac{1}{c}\cdot\frac{1}{P(x)}=1$，$\therefore P(x)=1$

$P(e|x)=\frac{c-1}{c}$

在 $j\le x\le j+1-\frac{cr}{c-1}$

$\sum_{i=1}^{c} P(w_i|x)=P(w_j|x)=\frac{1}{c}\cdot\frac{1}{P(x)}=1$ $\therefore P(x)=\frac{1}{c}$

$P(e|x)=0$

$$\therefore P^*=\int_0^{\frac{cr}{c-1}} P(e|x)\cdot P(x)\cdot dx$$

$$=\frac{cr}{c-1}\cdot\frac{c-1}{c}=r$$

2). 对于 $0\le x\le\frac{cr}{c-1}$，

$P(w_i|x)=\frac{1}{c}$，$P(x)=1$

$$\therefore P_0=\int_0^{\frac{cr}{c-1}}(1-\sum_{i=1}^{c}\frac{1}{c}(\frac{1}{c})^2)\cdot P(x)\,dx$$

$$=\int_0^{\frac{cr}{c-1}}\frac{c-1}{c}P(x)\,dx$$

$$=\frac{cr}{c-1}\cdot\frac{c-1}{c}=r$$

对于 $j\le x\le j+1-\frac{cr}{c-1}$

$P(w_j|x)=1$，$P(w_k|x)_{k\ne j}=0$

$$P_j=\int(1-1)P(x)\,dx=0$$

$\therefore P=P_0+\sum P_j=r$

$\therefore P=P^*$

3. Minkowski 距离的定义为

$D(\vec{X},\vec{Y})=(\sum_{i=1}^{n}|x_i-y_i|^p)^{1/p}$

① 由于 $\forall|x_i-y_i|\ge 0$，$\therefore D(\vec{X},\vec{Y})\ge 0$

② 由于 $|x_i-y_i|=|y_i-x_i|$

$\therefore D(\vec{x},\vec{y})=D(\vec{y},\vec{x})$

③ 现证明 $D(\vec{x},\vec{z}) \le D(\vec{x},\vec{y}) + D(\vec{y},\vec{z})$

该问题等价于

$$D(\vec{a}+\vec{b},0) \le D(\vec{a},0) + D(\vec{b},0)$$

$$\Rightarrow \left(\sum_{i=1}^{n} |a_i+b_i|^p\right)^{1/p} \le \left(\sum_{i=1}^{n}|a_i|^p\right)^{1/p} + \left(\sum_{i=1}^{n}|b_i|^p\right)^{1/p}$$

即 Minkowski不等式

$$\left(\sum_{i=1}^{n} |a_i+b_i|^p\right)^{1/p} = \left[\sum_{i=1}^{n} |a_i+b_i|^{p-1}\cdot|a_i+b_i|\right)^{1/p}$$

两边同求 p 次幂

$$\sum_{i=1}^{n} |a_i+b_i|^p = \sum_{i=1}^{n} |a_i+b_i|^{p-1}|a_i+b_i| \quad ①$$

由三角不等式

① 式 $\le \sum |a_i|\,|a_i+b_i|^{p-1} + \sum |b_i|\,|a_i+b_i|^{p-1}$ ②

由 Holder不等式，设 q 满足 $\frac{1}{p}+\frac{1}{q}=1$，即有

② 式 $\le \left(\sum |a_i|^p\right)^{1/p}\left(\sum |a_i+b_i|^{q(p-1)}\right)^{1/q}$

$$+ \left(\sum |b_i|^p\right)^{1/p}\left(\sum |a_i+b_i|^{q(p-1)}\right)^{1/q} \quad ③$$

考虑到 $q(p-1)=p$

③ 式 $= \left[\left(\sum |a_i|^p\right)^{1/p} + \left(\sum |b_i|^p\right)^{1/p}\right]\left(\sum |a_i+b_i|^p\right)^{\frac{1}{p}\cdot\frac{p}{q}}$

$$\Rightarrow D(\vec{a}+\vec{b},0)^p \le \left(D(\vec{a},0)+D(\vec{b},0)\right)\cdot D(\vec{a}+\vec{b},0)^{p/q}$$

$$D(\vec{a}+\vec{b},0)^{p-\frac{p}{q}} \le D(\vec{a},0)+D(\vec{b},0)$$

$\because \frac{1}{p}+\frac{1}{q}=1 \quad \therefore p-\frac{p}{q}=1$

$$\Rightarrow D(\vec{a}+\vec{b},0) \le D(\vec{a},0)+D(\vec{b},0)$$

即证.

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
```