模式识别作业2

张祎扬 181840326 匡亚明学院本科生

1. 第三章讲义习题2

(a)$\arg\min_{\gamma_{ij},\mu_i} \sum_{i=1}^K \sum_{j=1}^M \gamma_{ij}||x_j-\mu_i||^2$means finding the value of γ_{ij},μ_i that minimize the value of the formula. Note that $\gamma_{ij}=1$ if x_j is assigned to the i-th group, otherwise 0. So $\sum_{i=1}^K \gamma_{ij}||x_j-\mu_i||^2=||x_j-\mu_c||^2$(given a fix j, and xj belogs to group c).So $\sum_{i=1}^K \sum_{j=1}^M \gamma_{ij}||x_j-\mu_i||^2=\sum_{j=1}^M D_j$($D_j$ means the distance between x_j and the center of the group which x_j belogs to), so the optimization means to minimize the total distance between each x and their group center, which is the goal of K-means.

(b) i.for fixed μ_i,$J(\gamma,\mu)=\sum_{i=1}^K \sum_{j=1}^M \gamma_{ij}||x_j-\mu_i||^2$

for each j, exactly one of the following terms is nonzero:

$$\gamma_{1j}||x_j-\mu_1||^2,\gamma_{2j}||x_j-\mu_2||^2,\ldots,\gamma_{Kj}||x_j-\mu_K||^2$$

Take $\gamma_{cj}=1(c=\arg\min_i ||x_j-\mu_i||^2)$, that is to assign x_j to cluster c with minimum distance

$$||x_j-\mu_c||^2$$

ii. For fixed γ,

$$J(\gamma,\mu)=\sum_{i=1}^K \sum_{j=1}^M \gamma_{ij}||x_j-\mu_i||^2=\sum_{i=1}^K J_c$$

$$J_c(\mu_c)=\sum ||x_i-\mu_c||^2(i|x_i\ belongs\ to\ cluster\ c)$$

J_c is minimized by $\mu_c=mean(\{x_i|x_i\ belongs\ to\ cluster\ c\})$

(c)Note that the objective value never increases in an update(worst case:everything stays the same).

Consider the sequence of objective values, which are monotonously decreasing and bounded below by zero. Therefore, k-Means objective value converges to $inf_t J_t$.

Reminder: This is convergence to a local minimum.

注：因为买的中文版教材到了，所以后面的解答都用中文

2.第四章讲义习题2

(a)线性回归任务即找到最优的β，使代价最小化，即求 $\arg\min_\beta \frac{1}{n}\sum_{i=1}^n \epsilon_i^2=\arg\min_\beta \frac{1}{n}\sum_{i=1}^n (y_i-x_i^T\beta)^2$

(b)用X和y重写可以表示为，求$\arg\min\limits_{\beta} \frac{1}{n}(y-X\beta)^T(y-X\beta)$

(c)

$$\begin{aligned} J(\beta) &= \frac{1}{n}(y-X\beta)^T(y-X\beta) \\ &= \frac{1}{n}(y^T-\beta^TX^T)(y-X\beta) \\ &= \frac{1}{n}(y^Ty-y^TX\beta-\beta^TX^Ty+\beta^TX^TX\beta) \end{aligned}$$

对上式取β的偏导，

$$\begin{aligned} \frac{\partial J(\beta)}{\partial \beta} &= \frac{1}{n}(2X^TX\beta-X^Ty-y^TX) \\ &= \frac{1}{n}(2X^TX\beta-2X^Ty)\ (X^Ty=y^TX) \end{aligned}$$

令上式为0，

$$\beta=(X^TX)^{-1}X^Ty(\text{假设}X^TX\text{可逆})$$

(d)不可逆

(e)该正则项会对参数β作一个惩罚，解决线性回归可能出现的过拟合问题以及在通过正规方法求解β过程中出现的X^TX不可逆的情况。同时，通过确定λ的值，可以在方差和偏差之间达到平衡。

(f)岭回归的优化问题是求$\arg\min\limits_{\beta}(\frac{1}{n}(y-X\beta)^T(y-X\beta)+\lambda\beta^T\beta)$,同样求导并令导数为0，可得

$$\beta=(X^TX+n\lambda I)^{-1}X^Ty$$

(g)岭回归在X^TX的基础上加一个较小的λ扰动，使行列式不再为0，可求逆。

(h)$\lambda=0$时，即为普通线性回归，$\beta=(X^TX)^{-1}X^Ty.\lambda=\infty$时，欠拟合，相当于截距回归

(i)可以。可以对于不用的λ值进行岭回归，再通过交叉验证在训练集上选出最佳的λ值。还可以用梯度下降迭代找到最优解。

3.第四章讲义习题5

(a)

下标	类别标记	得分	查准率	查全率	AUC-PR	AP
0			1.0000	0.0000	-	-
1	1	1.0	1.0000	0.2000	0.2000	0.2000
2	2	0.9	0.5000	0.2000	0.0000	0.0000
3	1	0.8	0.6667	0.4000	0.1167	0.1333
4	1	0.7	0.7500	0.6000	0.1417	0.1500
5	2	0.6	0.6000	0.6000	0.0000	0.0000
6	1	0.5	0.6667	0.8000	0.1267	0.1333
7	2	0.4	0.5714	0.8000	0.0000	0.0000
8	2	0.3	0.5000	0.8000	0.0000	0.0000
9	1/2	0.2	0.5556/0.4444	0.8000	0.1056/0.0000	0.1111/0.0000
10	2/1	0.1	0.5000	1.0000	0.0000/0.0944	0.0000/0.1000
					0.6907/0.6795	0.7277/0.7166

(b)它们的值应该相差不大，但AUC-PR将面积近似为梯形，更为精确一些。因为在通常情况下，曲线是有斜率的。

(c)见上表最后几行斜杠后的数据

(d)代码如下

```python
import numpy as np
#input the array [class, point]
data = np.array(([1,1.0],[2,0.9],[1,0.8],[1,0.7],[2,0.6],[1,0.5],[2,0.4],[2,0.3],[1,0.2],[2,0.1]))
#calculate precision and recall
precision = [0]*11
recall=[0]*11
for i in range(1,11):
    predict_class = [0]*11
    TP=0
    for j in range(1,11):
        predict_class[j]=1 if data[j-1][1]>= data[i-1][1] else 2
        if predict_class[j]==1 and data[j-1][0]==1:
            TP+=1
    precision[i]=TP/predict_class.count(1)
    recall[i]=TP/5
precision[0]=1.0
recall[0]=0.0
#calculate AUC-PR and AP
auc=[0]*11
pr=[0]*11
for i in range(1,11):
    auc[i]=(recall[i]-recall[i-1])*(precision[i]+precision[i-1])/2
    pr[i]=(recall[i]-recall[i-1])*precision[i]
```

```python
print(auc)
print(sum(auc))
print(pr)
print(sum(pr))
```

输出结果如下,和手算结果一致（可能有一点点精度的差别）

```
/usr/local/bin/python3 /Users/zhangyiyang/workspace/python_practice/AUCPR
[0, 0.2, 0.0, 0.11666666666666665, 0.1416666666666666, 0.0, 0.1266666666666667, 0.0, 0.0, 0.10555555555555554, 0.0]
0.6905555555555555
[0, 0.2, 0.0, 0.13333333333333333, 0.14999999999999997, 0.0, 0.13333333333333336, 0.0, 0.0, 0.11111111111111109, 0.0]
0.7277777777777777

Process finished with exit code 0
```

修改第九个和第十个的类别标记后结果如下，仍然和手算基本一致（有一点精度差别）

```
/usr/local/bin/python3 /Users/zhangyiyang/workspace/python_practice/AUCPR
[0, 0.2, 0.0, 0.11666666666666665, 0.1416666666666666, 0.0, 0.1266666666666667, 0.0, 0.0, 0.0, 0.09444444444444441]
0.6794444444444444
[0, 0.2, 0.0, 0.13333333333333333, 0.14999999999999997, 0.0, 0.13333333333333336, 0.0, 0.0, 0.0, 0.09999999999999998]
0.7166666666666667

Process finished with exit code 0
```

4.第四章讲义习题6

(a)

$$
\begin{gathered}
E[(y-f(x;D))^2]=E_D[(F(x)-f(x;D)+\epsilon)^2] \\
=E_D[(F(x)-f(x;D))^2+\epsilon^2+2\epsilon(F(x)-f(x;D))] \\
=E_D[(F(x)-f(x;D))^2]+E[\epsilon^2]+2E_D[\epsilon(F(x)-f(x;D))] \\
\text{又}\, E[\epsilon^2]=(E[\epsilon])^2+Var(\epsilon)=\sigma^2 \\
E_D[\epsilon(F(x)-f(x;D))]=E_D[F(x)-f(x;D)]E[\epsilon]=0 \\
so\ E[(y-f(x;D))^2]=E_D[(F(x)-f(x;D))^2]+\sigma^2 \\
=(E_D[F(x)-f(x;D)])^2+Var(F(x)-f(x;D))+\sigma^2 \\
because\ E_D[F(x)-f(x;D)]=F(x)-E_D[f(x;D)] \\
and\ Var(F(x)-f(x;D))=Var(f(x;D))=E_D[(f(x;D)-E_D[f(x;D)])^2] \\
so\ E[(y-f(x;D))^2]=(F(x)-E_D[f(x;D)])^2+E_D[(f(x;D)-E_D[f(x;D)])^2]+\sigma^2 \\
=bias^2+variance+noise
\end{gathered}
$$

(b)

$$
\begin{gathered}
E[f]=E[\frac{1}{k}\sum_{i=1}^{k}y_{nn}(i)]=E[\frac{1}{k}\sum_{i=1}^{k}F(x_{nn}(i))+\epsilon] \\
=E[\frac{1}{k}\sum_{i=1}^{k}F(x_{nn}(i))]+E[\epsilon]=\frac{1}{k}E[\sum_{i=1}^{k}F(x_{nn}(i))]=\frac{1}{k}\sum_{i=1}^{k}F(x_{nn}(i))
\end{gathered}
$$

(c)

$$E[(y-f(x;D))^2]=(F(x)-E[f])^2+E[(f-E[f])^2]+\sigma^2$$
$$=(F-\frac{1}{k}\sum_{i=1}^{k}F(x_{nn}(i)))^2+E[\frac{1}{k}\sum_{i=1}^{k}(y_{nn}(i)-F(x_{nn}(i)))^2]+\sigma^2$$
$$=(F-\frac{1}{k}\sum_{i=1}^{k}F(x_{nn}(i)))^2+E[\frac{1}{k}\sum_{i=1}^{k}(\epsilon_{nn}(i))^2]+\sigma^2$$

(d)方差项是$E[\frac{1}{k}\sum_{i=1}^{k}(\epsilon_{nn}(i))^2]$,当k减小，方差变大；当k增大，方差变小

(e)偏置的平方项是$(F-\frac{1}{k}\sum_{i=1}^{k}F(x_{nn}(i)))^2$，当k变小时，它也随着变小；当k变大，偏置也变大

5.第五章讲义习题5

(a)要证范数相等，即证$X^TX=(GX)^T(GX),X^TX=(G^TX)^T(G^TX)$,因为G是正交矩阵，所以$G^TG=GG^T=I$

$$(GX)^T(GX)=X^TG^TGX=X^T(G^TG)X=X^TIX=X^TX$$
$$(G^TX)^T(G^TX)=X^T(G^T)^TG^TX=X^TGG^TX=X^T(GG^T)X=X^TIX=X^TX\text{证毕}$$

(b)

$$||X||_F=\sqrt{tr(XX^T)}$$
$$||G^TXG||_F=\sqrt{tr(G^TXG(G^TXG)^T)}=\sqrt{tr(G^TXGG^TX^TG)}=\sqrt{tr(G^TX(GG^T)X^TG)}$$
$$=\sqrt{tr(G^TXX^TG)}=\sqrt{tr(G^TX(G^TX)^T)}=\sqrt{tr((G^TX)^TG^TX)}$$
$$=\sqrt{tr(X^TGG^TX)}=\sqrt{tr(X^T(GG^T)X)}=\sqrt{tr(X^TX)}$$
$$so\ ||X||_F=||G^TXG||_F$$

(c)对于实对称矩阵X，可以分解为$X=J^TAJ$,其中J是正交矩阵，A是对角矩阵，且对角线上的元素是特征值。所以计算主成分就是要将X近似对角化。根据(b)的证明可以知道实矩阵的F范数是正交不变量，因此要使X近似对角化，就应该使对角元的平方和尽可能增大，非对角元的平方和尽可能减小，即$off(X)$尽可能减小。所以这一基本步骤很有用。

(d)令$J(i,j,\theta)$为一个$Givens$旋转矩阵，因为第(i,j)项和第(j,i)项都是0，代入得

$$x_{ii}sin^2\theta-2x_{ij}cos\theta sin\theta+x_{jj}cos^2\theta=0$$
$$\frac{1}{2}(x_{jj}-x_{ii})sin2\theta+x_{ij}cos2\theta=0$$

解得

$$tan2\theta=\frac{2x_{ij}}{x_{ii}-x_{jj}}$$
$$if\ x_{ii}=x_{jj},\theta=\frac{\pi}{4}sign(x_{ij})$$

所以令J为$Givens$旋转矩阵，按照上述解取θ的值。

(e)根据F范数不变性，变换前后$x_{ii}^2+x_{jj}^2+2x_{ij}^2$不变。由于旋转变化将第$(i,j)$项和第$(j,i)$项都置0，所以对角元的模平方和最多增加$2x_{ij}^2$，所以一次迭代不会增加$off(X)$.

(f)根据经典$Jacobi$法的迭代步骤，可以保证对角元的模平方和不断增加，$off(X)$广义单调减。所以总可以在有限次迭代后，使X成为严格对角占优阵。根据定理：若X为严格对角占优阵，则$Jacobi$迭代法收敛。