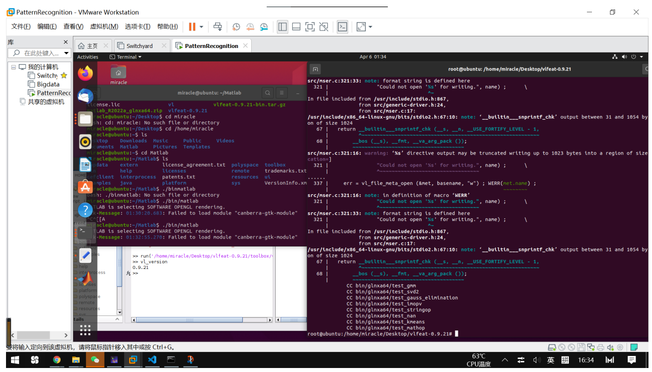
徐思源 191220133 计算机系计算机系选修课程 本科生

3.1

(a)



(b) 直接遍历所有的点，然后计算其他所有点的欧几里得距离，找出距离最小的点返回

(c) 首先用 $vlkdtreebuild$ 函数构建一棵 kd 树，之后调用 $vl_kdtreequery$ 返回前 k 个最近点，这里我们排除点自身的情况，返回最近的 2 个点，这时候第二个最近的点就是我们要求的最近邻点。利用 matlab 内置的 cputime 记录运行的时间，其中要省略 build 的时间，所以在 build 前后记录 cputime，两个时间相减得到结果。相比而言，利用 vlfeat 函数运行时间较快，大概相差五倍。

(d) 因为按照朴素算法计算的结果计算的结果一定是正确的，只需要比较 vlfeat 函数和朴素算法的结果就可以，本题中比较的结果是完全正确的，和我们的预期相符，因为函数当中设计的 MaxNumConparisions 是 6000，总样本量是 5000，理论上不会出现错误

(e) 错误率上升，运行速率下降

(f) 数据集大小从 5000 变成 500，正确率并不会下降，因为数据集大小仍然小于 MaxNumComparisions，正确率应该是 1，但是当数据集大小超过 6000 的时候正确率出现下降，50000 的时候正确率有所下降，但依然很高，随机结果大概在 0.9995 左右，但是相比之下，如果是 (e) 当中我们参数改小成 50，正确率下降的更加明显，大概是 0.8 几

4.5

(a)

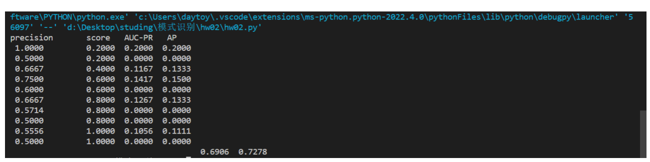
index	label	score	precision	recall	AUC-PR	AP
0			1.0000	0.0000	–	–
1	1	1.0	1.0000	0.2000	0.2000	0.2000
2	2	0.9	0.5000	0.2000	0.0000	0.0000
3	1	0.8	0.6667	0.4000	0.1167	0.1333
4	1	0.7	0.7500	0.6000	0.1417	0.1500
5	2	0.6	0.6000	0.6000	0.0000	0.0000
6	1	0.5	0.6667	0.8000	0.1267	0.1333
7	2	0.4	0.5714	0.8000	0.0000	0.0000
8	2	0.3	0.5000	0.8000	0.0000	0.0000
9	1	0.2	0.5556	1.0000	0.1056	0.1111
10	2	0.1	0.5000	1.0000	0.0000	0.0000
					0.6906	0.7278

(b) 是相似的，因为他们只是在计算 PR 曲线下面积时，用了不同插值计算近似方法，但是 AP 使用矩形近似而 AUC-PR 用梯形近似，大部分情况应该是 AUC-PR 更加精确一点，因为曲线大部分地方都是有一定斜率的

(c) 调换位置之后，数据只改变下标 9，10 的部分，如下:

9	2	0.2	0.4444	0.8000	0.0000	0.0000
10	1	0.1	0.5000	1.0000	0.0944	0.1000
					0.6794	0.7167

(d)



```
ftware\PYTHON\python.exe' 'c:\Users\daytoy\.vscode\extensions\ms-python.python-2022.4.0\pythonFiles\lib\python\debugpy\launcher' '5
6097' '--' 'd:\Desktop\studing\模式识别\hw02\hw02.py'
precision       score   AUC-PR   AP
 1.0000         0.2000  0.2000  0.2000
 0.5000         0.2000  0.0000  0.0000
 0.6667         0.4000  0.1167  0.1333
 0.7500         0.6000  0.1417  0.1500
 0.6000         0.6000  0.0000  0.0000
 0.6667         0.8000  0.1267  0.1333
 0.5714         0.8000  0.0000  0.0000
 0.5000         0.8000  0.0000  0.0000
 0.5556         1.0000  0.1056  0.1111
 0.5000         1.0000  0.0000  0.0000
                                         0.6906  0.7278
```

结果和自己计算的结果相符

4.8

(a) 出现了 5 次

(b) 一次出现概率是 $p = \frac{4}{9}$,10 次出现概率为 $1-(1-p)^{10} \approx 0.997$

(c)(0,2) 会导致样本分布不均衡，假设用第一类样本为 0 的组作为训练集，那么就会导致模型对于第 1 类的识别能力很差，而验证集当中存在第 1 类样本，模型泛化能力较差。

(d) 因为分层采样的时候要对每类样本进行训练/测试划分，而 D 中由两个 1 类样本，8 个 2 类样本，所以 1 类样本的分布一定是 (1,1)

5.1

(a)$XX^T = U\Sigma\Sigma^T U^T$,$XX^T$ 的非零特征值是 Σ 的非零元素的平方（$\Sigma\Sigma^T$ 对角线的非零值），特征向量是 U 的各列向量（即 X 的左奇异向量）

(b) 类似的，X^TX 的非零特征值是 $\Sigma^T\Sigma$ 对角线的非零元素，特征向量是 V 的各列向量（即 X 的右奇异向量）

(c) 因为 Σ 是一个对角阵，所以 $\Sigma^T\Sigma$ 和 $\Sigma\Sigma^T$ 的对角线非零值相同，X^TX 和 XX^T 的非零特征值相等

(d)X 的非零奇异值就是 XX^T 和 X^TX 非零特征值的算术平方根

(e) 此时 X^TX 是一个 100000x100000 的矩阵，直接计算特征值很麻烦，由前面的结论知道 X^TX 和 XX^T 特征值相等，所以可以计算 XX^T 的特征值。

5.2

第一个特征向量差不多就是平均向量

e1 随 scale 变化由较大的变化，$scale \geq 0.05$ 的时候，corr1 基本上是-1，特征向量和样本均值方向相反；scale=0.01，corr1 为 0.4544，随后随着 scale 改变而减小，而 new_e1 基本不变正确的第一个特征向量是

eigenvector =

-0.0068	0.2711	-0.4699	0.7081	-0.1738	-0.1484	0.1298	-0.5053	-0.2212
0.0231	-0.3025	0.3072	-0.5945	-0.2654	0.3062	-0.0648	0.4579	-0.6057
0.1128	0.2805	0.3117	-0.2714	0.1172	0.0838	0.1328	-0.2049	0.0691
0.6496	-0.4849	0.4091	-0.1457	-0.2615	-0.6770	0.4791	-0.0568	0.2573
-0.4993	0.2501	-0.1848	0.0165	0.1026	0.0259	0.4842	0.3546	0.2713
-0.3812	0.3148	-0.1475	0.0949	0.0185	0.0908	0.0175	-0.2661	0.1804
-0.2913	-0.3260	0.3197	0.1363	0.6275	0.3310	-0.4576	0.5093	0.5330
0.0800	-0.0483	-0.3257	-0.0872	-0.2475	-0.2855	-0.5098	-0.0761	-0.3315
0.2785	0.3761	0.1469	0.1191	0.4583	0.4379	-0.1578	-0.1585	-0.0510
0.0345	-0.3309	-0.3667	0.0241	-0.3759	-0.1647	-0.0534	-0.0541	-0.1018