knn 算法改进实验报告

实验环境：

操作系统 ubuntu14.04
语言：python2.7.6

代码：

knn.py（包含数据读取，原始 knn 算法实现，改进算法的辅助程序，算法测试部分）
主要函数

image2vect	读入图像数据
get_class	原始的 knn 分类
get_class_mdist	采用马氏距离的 knn 分类
get_class_kdtree	使用了 kd_tree 的 knn 分类
handwritingclasstest	主程序 1——测试原始 knn 算法
handwritingclasstest_mdist	主程序 2——测试马氏距离
handwritingclasstest_kdtree	主程序 3——测试 kd_tree

kd_tree.py（包含 kd_tree 的构造和搜索部分的实现）
主要函数

kd_tree	构造 kd_tree
search_kd_tree	原始的 kd_tree 搜索——只能搜索一个最邻近的样本点
search_kd_tree_n	改进后的 kd_tree 搜索——能一次搜索出 n 个最近邻

m_distance.py（包含马氏距离的计算）
主要函数

get_distance1_1	求解向量集中两个向量的马氏距离
get_distance1_n	求解测试向量与训练向量集中所有向量的马氏距离
get_distance1_n_covfix	求解测试向量与训练向量集中所有向量的马氏距离(提前输入伪逆矩阵)

代码测试方法：在 python 命令行下导入 **knn** 模块(knn.py)，分别运行 knn.handwritingclasstest()， knn.handwritingclasstest_mdist()，knn.handwritingclasstest_kdtree()即可，其中 knn.handwritingclasstest_mdist()运行时间较长（30min 左右）。

实验结果

实验	原始 knn 算法	kd_tree	马氏距离
error_num 错误个数	12.0	11.0	534.0
test_num 测试数据总数	946	946	946
rate 错误率	0.0126849894292	0.0116279069767	0.564482029598
time_used 用时(s)	13.291287899	57.335518837	2098.61799812

参数 k=3

注：原理上在 k 值相同的情况下，采用 kd_tree 后分类结果不变，但存在多个训练样本与测试样本距离相同的情况，这里的具体情况在下面的实验过程中会介绍。

实验过程：

1. 原始算法

在 knn.py 中实现了原始算法之后运行，最后的结果是：
在 946 个测试集中，分类错误的个数为 12 个，运行时间是 13s。

2. kd_tree 对比实验

参考《统计学习方法》在 kd_tree.py 中实现 kd 树的基本算法，包括 kd 树的构造和搜索，其中书上只介绍了在 kd 树中搜索唯一一个最近邻的方法，选择 k=1 对算法做一个简单的测试，这样算法就成了最近邻算法，测试结果如下：

error_num: 12.0
test_num: 946
rate: 0.0126849894292
time_used: 32.8754899502

分类的结果碰巧与上一步的实验相同，但用时反而是原始算法的两倍多，应该是数据量太少的问题，理论上只有训练样本数 n 与样本维度 k 满足关系 n=2^k 时算法才能取得比较好的效果，而这里的 n 与 k 的大小相差不大。

改进 kd_tree 搜索算法使得每次搜索能搜出多个最邻近的点，算法描述如下：

输入：kd_tree 根节点，测试点 t，最大距离值 Md，搜索的近邻个数 n。

(1)构造一个长度为 n 的**列表 list**，用于存放最近邻信息，初始化；

(2)从根节点开始深度优先搜索直到根节点，把搜索过程中遇到的节点依次压栈；

(3)如果栈为空，则返回 None；

(4)通过依次出栈回溯，对每一个遇到的节点，首先判断该节点与 t 的距离是否小于 list 中的最大值，如果是，则插入 list 替换掉最大值。

(5)判断 list 中的**最大距离**是否大于 t 与当前遇到节点的分割面的距离：

如果是，则递归的搜索另一分支（传给递归函数的参数中 Md 为 list 中的**距离最大值**），把递归函数返回的 list 与当前 list 合并，获得两个 list 并集的前 k 个最小点，然后回到第 4 步继续回溯

否则直接回到第 4 步继续回溯。

(6)回溯完成之后，如果 list 中的最小值大于 Md，那么返回 None，否则返回 list。

采用上述算法的 knn 分类运行结果如下（k=3）：

error_num: 11.0

test_num: 946

rate: 0.0116279069767

time_used: 57.335518837

首先，运行时间比 k=1 的情况更长了，达到了近 1 分钟，另外，其中分类错误个数不等于原始 knn 算法，把两个算法错误分类样本的信息打印出来：

原始 knn（12）	kd_tree（11）
96:5_42.txt(3)	96:5_42.txt(3)
242:8_68.txt(1)	
267:9_60.txt(7)	267:9_60.txt(7)
311:9_14.txt(1)	311:9_14.txt(1)
471:8_11.txt(6)	471:8_11.txt(6)
519:8_45.txt(1)	519:8_45.txt(1)
548:1_86.txt(7)	548:1_86.txt(7)
646:5_43.txt(6)	646:5_43.txt(6)
671:3_55.txt(9)	671:3_55.txt(9)
869:8_23.txt(3)	869:8_23.txt(3)
874:8_36.txt(1)	874:8_36.txt(1)
900:3_11.txt(9)	900:3_11.txt(9)

数据格式：样本索引号:文件名(实际分类)

二者区别是样本 242:8_68.txt，原始 knn 分类错误，样本本身是类别“8”被分成了类别“1”，从程序中找出更详细信息。

原始 knn 算法线性查找的 top4 个近邻是(训练样本与测试点距离，训练样本类型)：

(10.8627804912,8) (10.9087121146,1) (11.0905365064,1) (11.0905365064,8)

返回的分类结果是 1

kd 树查找过程中返回的 3 个最近邻是：

(10.8627804912,8) (10.9087121146,1) (11.0905365064,8)

返回的分类结果是 8

可以看出由于第三近邻有多个距离相同的样本点，两个算法因为查找顺序的问题导致产生了不同的结果。

3. 马氏距离对比实验

训练集的协方差矩阵是个奇异矩阵，所以最后用的伪逆矩阵，计算过程比欧氏距离慢很多，计算结果如下：

error_num: 534.0

test_num: 946

rate: 0.564482029598

time_used: 2098.61799812

时间花费是欧氏距离的 100 倍以上，并且不知道是不是用伪逆矩阵的问题，最后的错误率达到了 56%，这个如果要改进看到效果的话只能换数据集了，因为时间问题这个也没往下做了。

在实现过程中前期没有考虑到先把训练集协方差矩阵的伪逆算出来而是对每个测试数据都算一遍导致速度不能忍受(get_distance1_n)，最后改进了之后(get_distance1_n_covfix)计算过程在半个小时完成了。