数字系统设计实验

**实验二：软件编程**

# K-NN算法

## update\_knn函数实现

为使后续计算方便，确保training\_set中每个x的k个距离均由小到大排列。因此，计算出距离dis后，将dis与training\_set\_x中的距离作比较，若dis比最大距离小，则将其插入2个距离之间或最前面，同时dis后面的距离依次后移一位，最大距离舍去，完成更新。

## knn\_vote函数实现

定义数组mindis\_num[10]，表示每个x得到的投票数。由于training\_set中每个x的k个距离均由小到大排列，故每次筛选最小距离时，都将10个knn\_set[mindis\_num[x]]（即各个类别中剩下的最小距离）相互比较，求最小者，设最小者所属类别为x0，再让mindis\_num[x0]++，实现投票。最后统计得票最多者作为最终判定结果。若出现多个得票最多者，则任选其一（程序中选取x小者）。

# 代码测试

## FPGA-Windows文件互传

编写完成k-NN算法代码后，接下来的任务是将代码部署到FPGA上的Xillinux系统中，在本地编译并运行k-NN算法代码。为此，首先需要搭建PC(Windows)---Xillinux文件传输渠道。

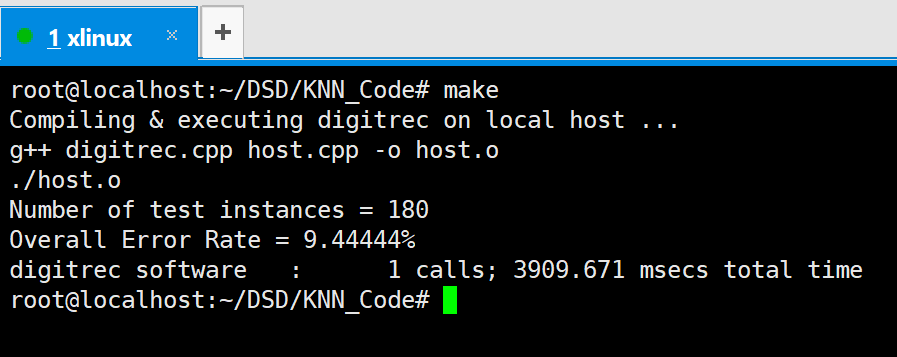
Step1: 在网上下载lrzsz程序，并将gz压缩文件放置在SD卡的FAT分区中，和xillydemo.bit放在一起。

Step2: 用tar命令解压lrzsz.tar.gz，并且本地编译。

Step3: 打开XShell，将文件拖拽到shell控制台上即可传输文件。

## KNN算法测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| **k** | **测试实例数** | **耗时** | **错误率** |
| 1 | 180 | 3167ms | 9.44% |
| 2 | 180 | 3299ms | 12.78% |
| 3 | 180 | 3207ms | 7.22% |
| 4 | 180 | 3521ms | 7.78% |
| 5 | 180 | 3387ms | 7.22% |
| 6 | 180 | 3479ms | 6.67% |
| 7 | 180 | 3619ms | 7.77% |
| 8 | 180 | 3855ms | 8.33% |
| 9 | 180 | 3911ms | 9.44% |



图：编译执行KNN代码