算法项目实验报告

寻找最近邻对

08 组　　黄建武 骆铭涛 童云钊 陈海涛

目录

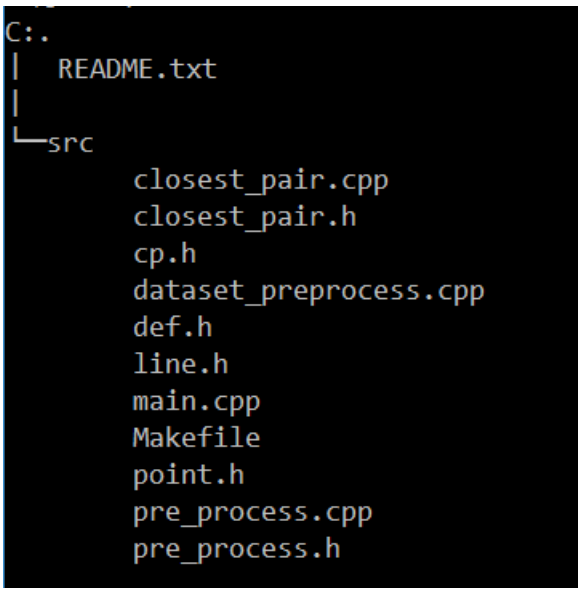
一、实验目的

实现一个带有随机投影的近似算法，在高维空间中找出一组对象中的最近邻对，并使用欧氏距离作为距离度量。

二、实验环境

编程语言：C++（C++11 标准）
操作系统：Linux
编译器：g++5.4.1

三、文件目录结构



四、实验实现

1. 首先，先生成均匀分布，再使用该均匀分布生成 100 个单位向量，读入文件将图片的像素信息并存进数组，将每个点分别投影到生成的 100 个向量上，形成 100 根线；
2. 将投影形成的 100 根线拷贝一份，将这两份作为后面中位数最近邻算法和随机最近邻算法的输入，控制变量法，保证两个算法的投影相同；

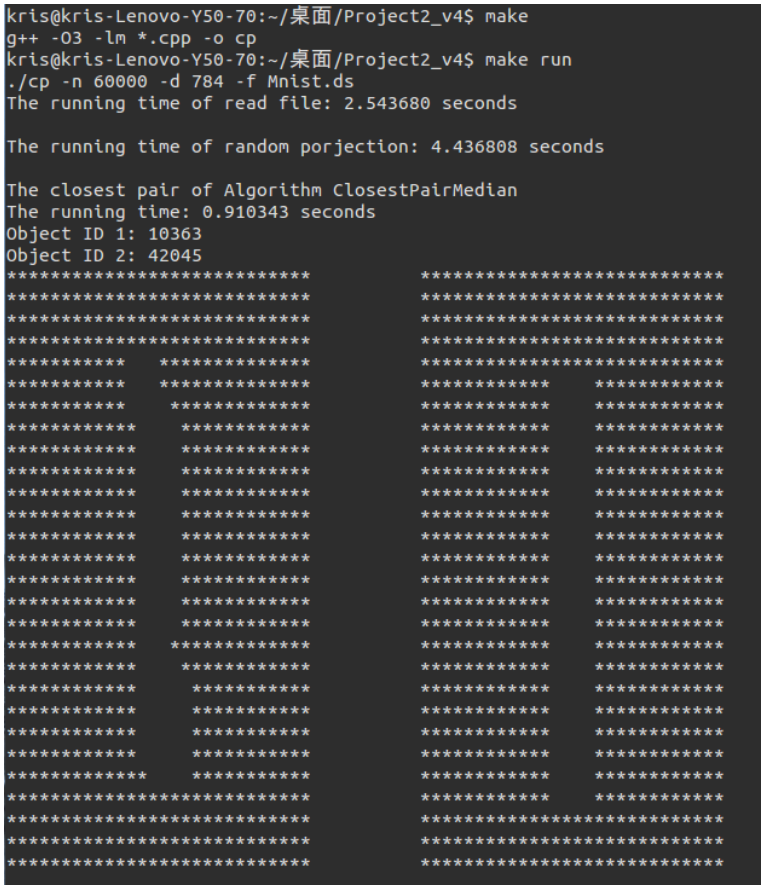
3. 实现找第 K 大的点，在一根线上比较投影值，随机选一个点，交换该点左右两边的点，使得左边的点都比它小，右边的点都比它大，递归左右两边，该点与左右相邻两个点形成点对，左边的最近点对，右边的最近点对，返回这 4 个点对中的最小点对；

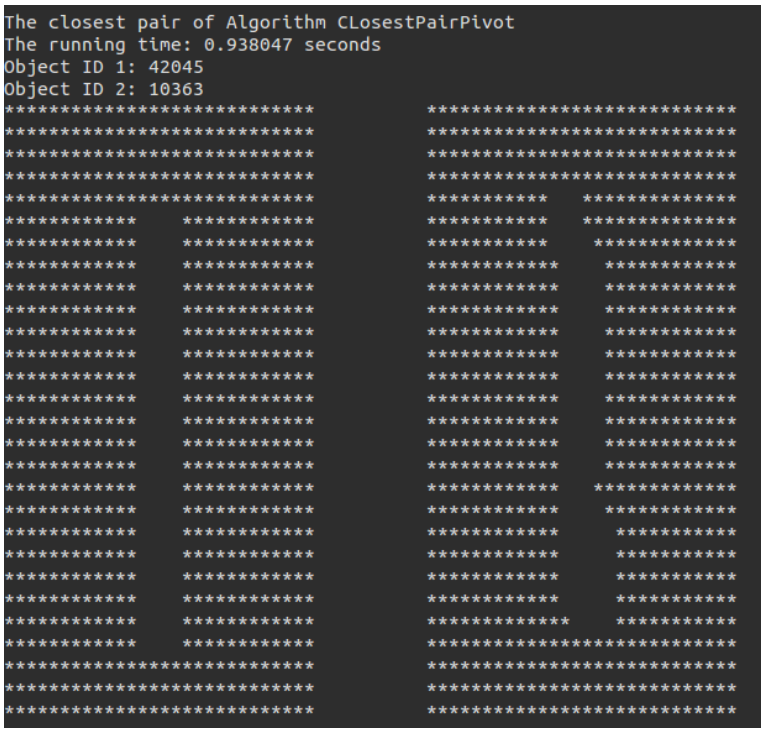
4. 对 100 根线运用 3 中的方法，可以得到 100 个点对，分别计算这 100 个点对的欧氏距离的平方，找出最小的点对（欧式距离的平方小的点对欧式距离也小）即为算法得到的最近对；

5. 复用寻找第 K 大的函数，根据中位数和随机两种算法，改变 K 的值即可。

五、实验结果与结论

1、实验结果





2、评测指标及结果分析

如图所示，读文件的时间为 2.54s，投影的时间为 4.44s。两个算法分别是中位数和随机算法。中位数算法得到的两个点的 ID 为 10363 和 42045，运行的时间为 0.91s；随机算法得到的两个点的 ID 分别为 42045 和 10363，运行时间为 0.94s。

经过多次运行，可以知道随机算法总是比中位数算法慢约 0.04-0.05s。因为随机算法有可能随机到最差的情况，这时候的时间复杂度接近 $O(n^2)$，所以随机算法比中位数算法慢。

六、心得总结

骆铭涛：

这次实验让我深入地了解如何使用分而治之的方法寻找最近邻对，并对平均分割和随机分割的两种方法进行比较。实验结果出乎我的意料，随机地以一个点为标准将一条线上的所有点分割成两个集合比以中点为标准将一条线上的所有点分割成两个集合，这说明实践非常重要，有时候能够推翻猜想。

实验过程中也遇到不少 bug，例如访问越界和函数栈溢出，用 gdb 调试大大加快我们定位 bug 的速度，以及了解 bug 的错误信息。对于代码优化，我们输出每部分花费的时间再进行优化，其中线上的点比较距离的时候，刚开始我们使用欧氏距离比较，大约是 22s 左右，后来我们改为点在线上投影的距离进行比较，

大约是 6s 左右，效率提升明显。

童云钊：

总体来说，这次项目考验了我们的编码能力、学习能力，更锻炼了我们的团队协作能力和软件工程能力，对我来说有很大的收获。

在团队里我主要负责了预处理模块和实验报告的编写，总体来说工作量并不大。这个实验最有价值的地方在于锻炼了分而治之的思想，比起暴力求解最近邻，分而治之的思想效率更高也更加优雅。虽然这部分代码的编写我没有参与，但看着队友编写这个模块并自己思考理解，还是有了很大的收获。

虽然我负责的工作看上去完成难度并不大，但过程中还是遇到了一些问题，暴露了太久没打 c++，知识遗忘得比较严重。c++编码太过生疏导致了程序运行起来出现了一些 bug，花了很多时间去调试。这些问题都值得我去认真地思考和总结。

总而言之，这次项目很好地锻炼了我的工程能力，积累了宝贵的经验教训，也给我提了个醒，就是要加强 c++编码的练习，基本功不能忘。

黄建武：

很久没用 C++写过程序，用惯了 Python，没有注意到 C++中的传递参数都是传值，在预处理的函数参数中，一开始直接传进 data_objects 和 lines 当成了 Python 和 Java 里面的传对象,在调用预处理函数后访问 data_objects,发生段错误,使用 gdb 调试后,才发现非法访问内存,于是改成传递引用。

此次实验也先后修改了很多地方,一开始在线上找最近对时是比较两点之间的欧氏距离的平方而不是投影差的绝对值,程序运行时间长,每次都相当于暴力求 60000 个点中最近的两个点,根本没有使用到投影值，误解了算法的意思。

在比较两种算法的运行时间时，一开始都是随机算法运行时间少，后来才发现两个算法使用的是相同的投影线，因为调用了中位数的算法后，100 根投影线的投影已经有序了，然后在有序的基础上使用随机算法，随机算法根本就没有交换的过程，因此运行时间少，改用拷贝投影线后，再次运行，中位数算法的运行时间更少。

陈海涛：

这次实验我负责的是从一百对最近邻中找出最近的一对，并输出这两个点的图像内容。由于这一部分并没有涉及到重要算法内容，所以实现起来很简单。

一开始与队友讨论后的结果是他们返回给我的是每条线上最近邻对的 objectID，然后从数据集中读取对应这两个 objectID 的数据，于是我就先按照文

件读取的方法去实现了。后来与队友进一步讨论后，决定改为一开始就把数据都存放在一个 60000*784 的二维数组里面，直接通过数组下标就可以访问对应 objectID 的数据，因此最终我的实现也变得更简单。

在一条线上找出欧几里得距离最近的两个点，最容易想到的方法是对线上的点进行排序，然后一次遍历即可找到最近的两个点。但是我们可以使用分而治之算法，使得在排序的过程中找到最近的两个点。算法实现过程是先找出中位数，然后将这条线分成两部分，递归寻找左右两部分的最近邻点对，所以这条线上的最近邻点对就是左边最近、右边最近以及中位数最近这三者中的一个。至于寻找中位数，先随机选取一个数作为分界点，然后将线上的点分为大于它和小于它的两部分，根据两部分的数据集大小再进一步递归划分，直至找到目标中位数。

通过这次项目，我加深了对寻找最近邻算法和分而治之算法的理解，与此同时，我深刻体会到了队友实力的强大，以及自己距离他们还有一大段差距，接下来的日子还是要好好努力，一起加油吧。

七、小组成员分工

小组成员	分工
骆铭涛	负责找每条线上的最近邻对，实现中位数和随机算法
童云钊	负责预处理部分和实验报告
黄建武	负责找每条线上的最近邻对，实现中位数和随机算法
陈海涛	负责数据集的处理，找出候选对中的最近邻对