欧式距离计算

注：欧式距离时间复杂度非常高 例如：假如colLen里面数字分别位5，6，7，4 则会计算:`2^5 + 2^6 + 2^7 + 2^4`次

计算欧式距离涉及的函数：

```
void countOneAndZero(map<int, int>& oneCount, map<int, int>& zeroCount)
```

函数作用：统计每一位的0和1的个数

`oneCount`统计每一位的0的map

`zeroCount`统计每一位的1的map

传入引用，实现in place修改，即计算公式三中的`n1`和`n0`

$$Pr(s_i = 0) = \frac{(P_{diff})^{n_1} \times (P_{same})^{n_0}}{(P_{diff})^{n_1} \times (P_{same})^{n_0} + (P_{diff})^{n_0} \times (P_{same})^{n_1}}$$

```
vector<double> getParamQ(int i)
```

函数作用：计算第`i`个属性的`Q`参数

返回参数：一系列的Q，用于计算公式五

如图，即图中的`Q`参数

$$E(\mathbf{s},\mathbf{t}) = \sqrt{\sum_{i=0}^{2^{m_1}-1} (i - t_1^d)^2 \times Q_{1i} + \ldots + \sum_{i=0}^{2^{m_n}-1} (i - t_n^d)^2 \times Q_{ni}}$$

```
double getDistance(NDB nDB, string dValue)
```

函数作用：计算nDB和dValue之间的距离

返回参数：距离

如图，即图中的`E(s,t)`

$$E(\mathbf{s},\mathbf{t}) = \sqrt{\sum_{i=0}^{2^{m_1}-1} (i - t_1^d)^2 \times Q_{1i} + \ldots + \sum_{i=0}^{2^{m_n}-1} (i - t_n^d)^2 \times Q_{ni}}$$

```
double EuclidDis::getAttrDvalue(string s, int i)
```

函数作用：计算二进制串`s`第`i`个属性的十进制值

返回参数：对应的属性的十进制值

此函数是为了计算如图的参数：

$$E(\mathbf{s},\mathbf{t}) = \sqrt{\sum_{i=0}^{2^{m_1}-1} (i - t_1^d)^2 \times Q_{1i} + \ldots + \sum_{i=0}^{2^{m_n}-1} (i - t_n^d)^2 \times Q_{ni}}$$

```
pDiff = 0.0;
for (int i = 1; i <= K0; i++) {
pDiff += P[i] * i;
}
pDiff = pDiff / K0;
pSame = 1 - pDiff;
```

代码作用：计算公式三中的`Pdiff`和`Psame`

kmeans

`kmeans`没有使用类

使用方法：

```
vector<Cluster> kmeans(nDBs, iteration, k, colLen)
```

返回结果：

`vector<Cluster> cluster`：聚类中心，每个`Cluster`对象中包含这一类的样本

参数：

`nDBs`：负数据库

`iteration`：最大迭代次数

`k`：`kmeans`中的参数`k`

`colLen`：代码里面的`colLen`变量

注：

`kmeans`中更新聚类中心部分存在问题：老师所给代码中的计算方法在论文中没有提及

knn

`knn`使用面向对象的思想

使用方法：

```
EuclidDis dis(colLen);  //同样可以使用HammingDis dis(colLen);
KNN knn(3,nDBs,labels); //labels和nDBs一一对应
knn.setDistance(&dis);
map<string,string> testResult = knn.testData(testdata);
```

返回结果：

`map<string,string> testResult`利用 `string` 来索引，获取 `label`

参数：

`testdata` 为 `vector<string>` 类型，每一个 `string` 代表一个测试数据对应的二进制串

读取文件

```
void readDataWithLabel()
```

文件格式：

```
floatN,floatN,floatN,1
floatN,floatN,floatN,0
floatN,floatN,floatN,0
floatN,floatN,floatN,1
floatN,floatN,floatN,0
...,...,...,...,...
```

最后一列0，1或者A，B都可以

文件更新部分

`calDistance.cpp`，`calDistance.h` 文件用于距离计算（面向过程思想）

`kmeans.cpp`，`kmeans.h` 文件用于实现`kmeans`算法（面向过程思想）

`Distance.h` 将距离进行抽象，归为了一个父类`Distance`，`EucildDis` 和 `HammingDis` 是父类，这样写是为了将来在距离上实现多态

`Distance.cpp` 实现`Distance.h`中的方法，与`calDistance.cpp`的方法冲突

注： `kmeans`方法因为没有使用面向对象的思想实现，所以只能使用`calDistance.cpp`中的方法

文件读取部分新增加三个函数，用于实现带有标签的文件的读取

后续

1. Hamming距离实现
2. kmeans改成面向对象方式，便于提供统一的对外接口
3. 欧式距离时间复杂度较高要找老师寻求解决方案
4. kmeans更新聚类中心打算找老师问清楚