Averaging the GPS trajectories

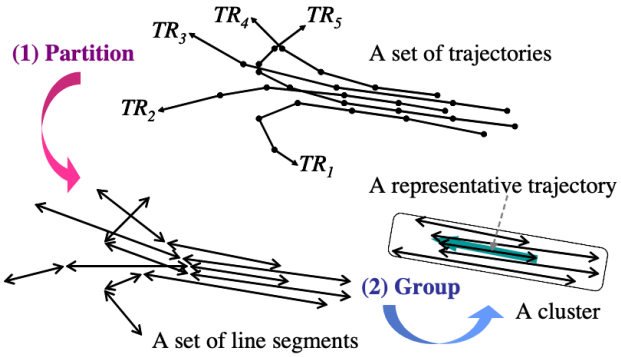
1753837 陈柄畅

1750926 周添龙

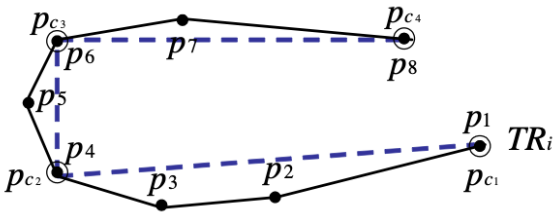
算法

我们的算法主要参考了`TRACLUS`，由Jae-Gil Lee, Jiawei Han 和 Kyu-Young Wang 发表于SIGMOD'07。并根据这次的题目做出了改动和调整。

我们的算法主要可以分为三个阶段：划分、聚类、取代表线段。前两个阶段与论文方法比较相似，第三个阶段根据实际情况做出了不同的尝试。



首先是划分阶段。



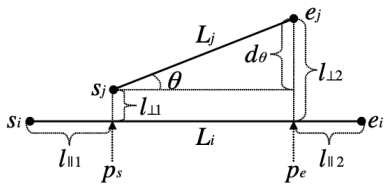
对于一条轨迹来说，它的划分有两个重要的评判标准，准确度和连续性，表现在轨迹里就是划分更长的segment并且平衡segment与原轨迹的偏差。信息论中的最小描述距离可以很好地表示这两方面。

$$L(H) = \sum_{j=1}^{par_i - 1} \log_2(len(p_{c_j} p_{c_{j+1}}))$$

$$L(D|H) = \sum_{j=1}^{par_i - 1} \sum_{k=c_j}^{c_{j+1}-1} \{\log_2(d_\perp(p_{c_j} p_{c_{j+1}}, p_k p_{k+1})) + \log_2(d_\theta(p_{c_j} p_{c_{j+1}}, p_k p_{k+1}))\}$$

具体的论证在论文中有详细描述，这里不作赘述。简单描述就是取得能取到的相距最远的两个特征点，保证这样划分优于不划分。

接下来是聚类阶段，我们采用了论文中类似DBSCAN的聚类算法和其中定义的距离



$$d_\perp = \frac{l_{\perp 1}^2 + l_{\perp 2}^2}{l_{\perp 1} + l_{\perp 2}}$$

$$d_\parallel = \mathrm{MIN}(l_{\parallel 1}, l_{\parallel 2})$$

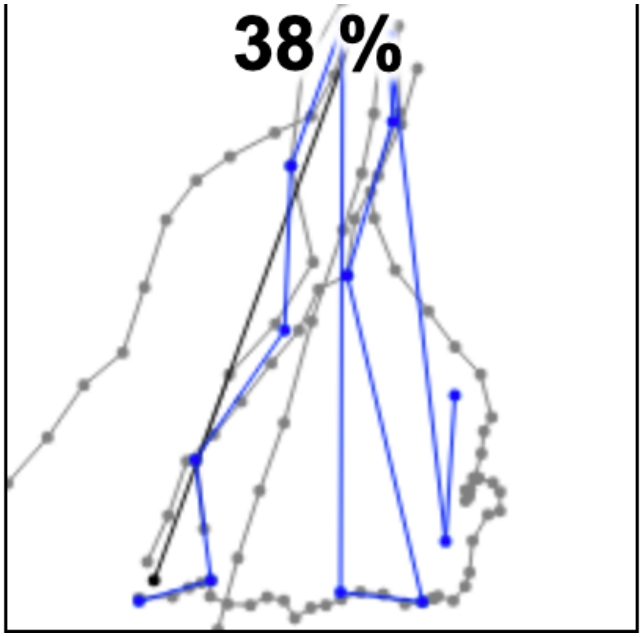
$$d_\theta = \|L_j\| \times \sin(\theta)$$

这个自定义距离分别从长度，远近，角度偏差计算了两条线段的距离。与DBSCAN不同的是，该聚类算法考虑到一个类是否可以代表足够多的不同轨迹。但初期错误认为本题测试样例中，有些数据样本轨迹过少，与论文所设想的情况稍有不同，加以限制可能会导致无聚类。所以初期没有尝试。但其实这一限制更是为了去除噪音。后期加入后对聚类效果有所改善。

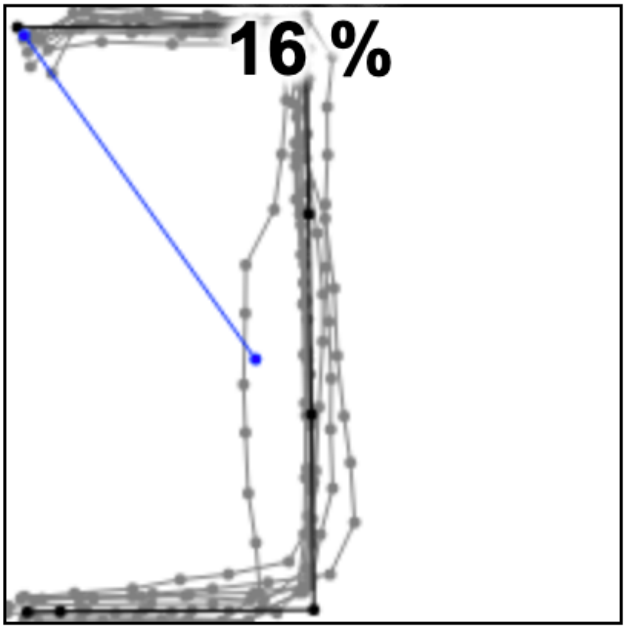
聚类过后，需要对类进行取代表轨迹。算法根据类求取平均向量，再旋转坐标轴，在X轴与平均方向向量平行的坐标系中根据当前垂直于X轴的线段是否满足要求，从而取平均点，最后组成代表轨迹。

论文中对于聚出的不同的类并没有进行连接处理，所以与本题不同。我们接下来在不同的方面进行了尝试：

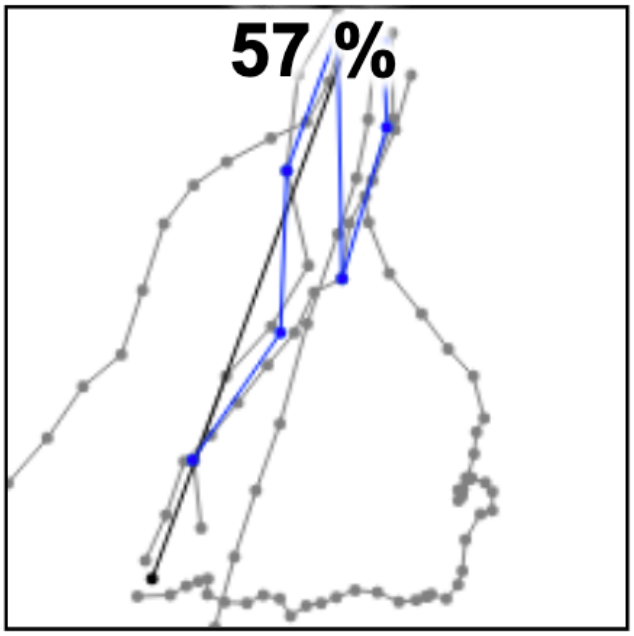
- 首先，我们先定义轨迹总体走向，即x*y的大小关系，确定轨迹的起始点，为所有轨迹的所有特征点进行时序编号。最后对聚出的不同类的代表轨迹按时序进行连接。但是测试数据中同一样本中，有些轨迹点过多，有些又很少，所以这种方法做出的效果并不好。数据也没有给出明确的时序信息可以利用。最后结果准确度不到59%。最严重的误差情况如下所示。但后期加入聚类限制后，异常点的干扰减小

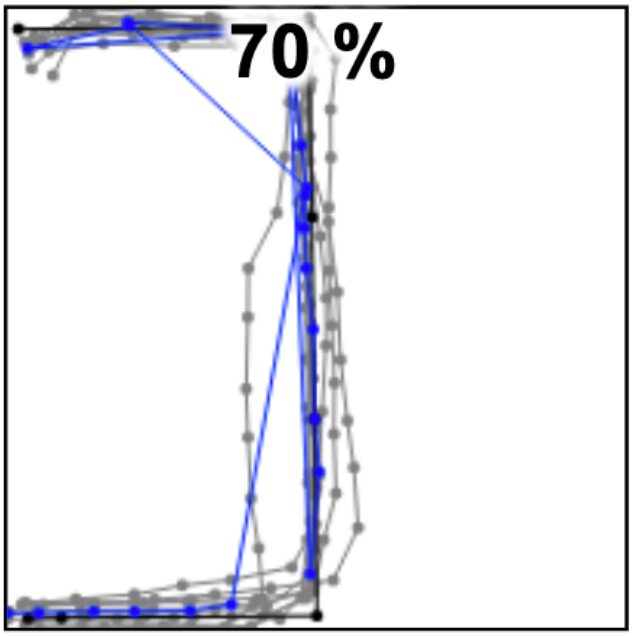


- 接着，我们观察了大多数测试数据的情况，发现大多数情况下，都为单条轨迹。所以我们将所有聚类片段轨迹都归入一个类中，求取代表轨迹。由于测试集偏向于直线，所以这种方法反而效果有所提高，达到62%。但直接归为一类的最大问题在于，对于一些角度变动较大的轨迹，实际上是由多个类合并而成。直接归为一类会使其聚类效果极差。如下图所示



- 最后，由于时间限制，我们就再进行了一种尝试。首先对不同聚类的代表向量进行求夹角，如果夹角大于一定角度，则直接进行按时序连接。如果夹角小于设定的超参数，那么就对所有的聚类结果进行二次聚类，这样可以有效的减少不同轨迹点数偏差导致的来回曲折现象。而且对于轨迹拐弯角度较大的样本，也具有一定的准确度。同时，我们加上了之前一直错误认为的聚类限制，从而对异常样本的处理更加合理。但是由于测试数据的特殊性，准确度仍只有60%，不如直接聚成一类。





代码

`Point`：轨迹点

`EndPoint`：对Segment进行represent时的点

`Segment`：划分的片段

`calEuclideanDistance(point1: Point, point2: Point)`：计算欧式距离

`calPoint2LineDistance(point, start_point, end_point)`：计算点到直线距离

`calThreeDistance(point1: Point, point2: Point, point3: Point, point4: Point)`：计算三个线段之间距离

`calMdlPar(points_i2j)`：计算MDL_{par}

`calMdlNoPar(points_i2j)`：计算MDL_{no_par}

`partition(trajectorySet)`：划分轨迹

`calENeighbor(segment: Segment, segments: np.ndarray)`：计算N近邻

`expandCluster(queue: list, segments, cluster_id, segment_cluster, clusters)`：拓展聚类

`group(segments)`：聚类

`calAverageDirectionVector(cluster)`：计算平均向量

`represent(cluster)`：取代表点

`ParameterSelectFunc(segments)`：参数选择

`calIncluedAngle(x1, x2, y1, y2)`：计算向量夹角

改进

- 该算法较为适合轨迹点多的聚类情况。对于测试样本，稍微调整参数，就有可能导致数据无聚类结果。所以可以考虑更换算法。
- 对于不同聚类的连接仍存在不足，来回曲折的现象仍然存在。后期可能考虑根据不同轨迹的点的个数为不同轨迹的时序赋予不同权重。