如何判断两条轨迹（或曲线）的相似度？

[图片] 比如上图，虽然它们的重合度不高，但是它们的结构是一样的，应当认为它们是很相似的，请问如何量化的判断呢？有什么好的算法？...显示全部

15 个回答 默认排序

匿名用户

42 人赞同了该回答

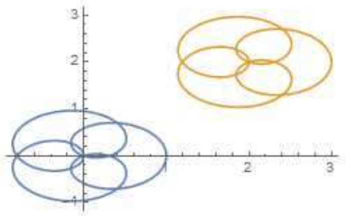
有这么个距离度量...叫

时间翘曲距离(Canonical Warping Distance)

一个点集经过平移,旋转,放缩后得到的点集与原来的点集的距离为0(时间翘曲意义下).

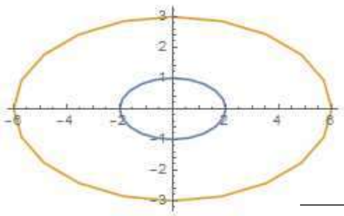
也就是具有平移不变性,旋转不变性,标度不变性

平移不变性:



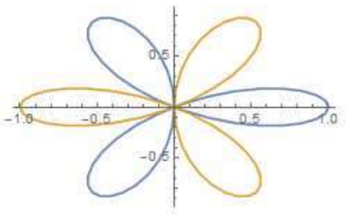
```
CanonicalWarpingDistance[s1, s2]
典型的时间翘曲距离
2.29375×10^-29
```

标度不变性:



```
CanonicalWarpingDistance[s1, s2
典型的时间翘曲距离
0.
```


旋转不变性:



```
CanonicalWarpingDistance[s1, s2]
典型的时间翘曲距离
0.
```

虽然这个名字取得很科幻,但是思想很简单

从1维情况说起:有两个时间序列比如:

A:{1, 2, 3, 5, 8, 13, 21, 34, 55, 89}

B:{0, 1, 1, 2, 3, 5, 8, 13, 21, 34}

相关问题

割圆曲线是什么样的曲线？ 1 个回答

几何拓扑（geometric topology）这个数学分支，主要包含哪些内容？ 3 个回答

有哪些定理在高维情况下与三维情况下培养出来的直觉不符？ 24 个回答

高维空间点的旋转问题？ 6 个回答

轴对称图形只能看出来么，不能证明么？ 20 个回答

```
In[874]:= A = {1, 2, 3, 5, 8, 13, 21, 34, 55, 89};
          B = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34};
          CanonicalWarpingDistance[A, B]
          典型的时间翘曲距离
          N@EuclideanDistance[A, B]
          欧几里得距离
Out[876]= 0.197573
Out[877]= 69.9643
```

从欧几里得距离上看不知差到哪边去了
但是实际上这两个曲线极其相似(本来就是同一个)...
什么情况下这两个时间序列才能相似呢?
时间扭曲
A从1-5花了3时间,B花了5时间..
所以只要扭曲下A在t=1-3时的时间标度就能对齐了!

一维情景大概相当于语音识别中用到的DTW(DynamicTimeWarping)动态时间规划算法,用于判断不同音长,不用音高,不同音调的声音是否是同一个音节...

二维情景复杂得多,不过也类似看成存在一种时空扭曲使得两个点集相似就行了...
当然也能作用于更高维度...

发布于 2017-02-06

THU Zoey
对现在有耐心，对未来有信心

43 人赞同了该回答

谢瑶。。。 @Macrofuns果然验证了你没有看我paper的method。。。

做过一个小研究是关于判断曲线的相似性的，点赞答案1已经基本上解决了题主的问题。使用归一化后的Fréchet
distance即可。另外也可使用Hausdorff distance，但是使用效果不如前者。
推荐几篇论文仅供参考：
Alt H, Godau M (1995) Computing the Fréchet
distance between two polygonal curves International Journal of Computational
Geometry & Applications 5:75-91. doi:10.1142/S0218195995000064

Fréchet MM (1906) Sur quelques points du
calcul fonctionnel Rendiconti del Circolo Matematico di Palermo (1884-1940)
22:1-72. doi:10.1007/BF03018603

此外，可以使用离散化的方法使用Fréchet distance，以便于实际编程计算，同推荐一篇离散化的文章供参考，里面有该方法的伪代码，另matlab也有现成的package，搜搜即可。

Eiter T, Mannila H (1994) Computing discrete
Fréchet distance See Also

最后推荐一个应用该方法的实例，为了解决Fréchet distance中阈值的问题对其进行了标准化从而得到一个相似指数。

Wang J, Xu C, Tong S, Chen H, Yang W (2013)
Spatial dynamic patterns of hand-foot-mouth disease in the People's Republic of
China Geospatial health 7:381-390

了解有限，对于该方法的代码实现问题我也存在疑虑，如有问题和了解欢迎探讨。

编辑于 2015-01-06

▲ 43 ▼ 12 条评论 分享 收藏 感谢 ···

Chao Xu
理论CS博士生，关注组合优化，开始学习AI，会写代码

理论CS博士生. 关注组合优化. 开始学习AI. 会写代码.

10 人赞同了该回答

可以使用Fréchet distance 特别是如果你已经有了trajectory.
当然了不会是纯粹的Fréchet distance. 要先normalize一下再使用...

发布于 2015-01-05

▲ 10 ▼ 1 条评论 分享 收藏 感谢 ···

Di Yao

26 人赞同了该回答

判断两条轨迹的相似性方法有很多

基于点方法： EDR，LCSS，DTW等

基于形状的方法： Frechet， Hausdorff

基于分段的方法：One Way Distance, LIP distance

基于特定任务的方法：TRACLUS， Road Network，grid等

附上本人总结的Trajectory Distance slides：

Relation Between EDR and LCSS

10

- They are both count-based
  - LCSS counts the number of matched pairs
  - EDR counts the cost of operations needed to fix the unmatched pairs
- Higher LCSS, lower EDR
  - If cost(replace) = cost(insert) = cost(delete)
  - EDR(X,Y) = L(X)+L(Y) – 2LCSS(X,Y)

Outline

11

- Point based Distance
  - Euclidean, DTW, LCSS, EDR
- Shape based Distance
  - Hausdorff distance, Frechet distance
- Segment based Distance
  - One Way Distance, LIP distance
- Task Specific Distance
  - TRACLUS, road, semantic, grid
- Conclusion

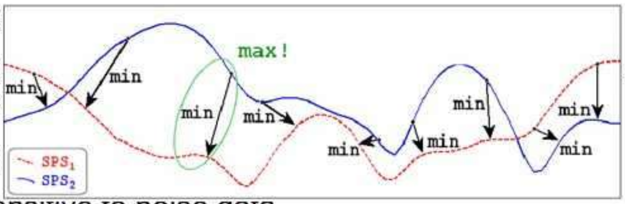
Hausdorff distance

12

- Used to measure how far two trajectories are from each other

$$D_H\left(L_i, L_j\right) = \max(h(L_i, L_j), h(L_j, L_i))$$
$$\text{where } h(L_i, L_j) = \max_{a \in L_i}(\min_{b \in L_j}(dist(a, b)))$$

- Pros
  - M
    tw veen
- Con
  - Sensitive to noise data

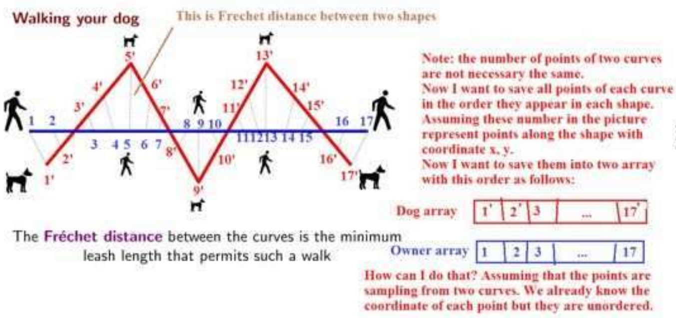


H. A lt, "The computational geometry of comparing shapes, " in EffiCient Algorithms. Springer, 2009, pp. 235-248
Xie D, Li F, Phillips JM. Distributed Trajectory Similarity Search. *Pvldb*. 2017:1478-1489.

Frechet Distance

13

- Sh
- Dc
  - pg
  -



where, $\|C\| = \max_{k=1}^{K} \text{dist}(a_i^k, b_j^k)$

$D_F(L_i, L_j)$ is the Fréchet distance between trajectory segments L_i and L_j. Here, L_i and L_j are the trajectory segments whose lengths are m and n respectively. $K = min(m, n)$. a_i^k and b_j^k are the kth points on trajectory segments L_i and L_j respectively. $dist(a_i^k, b_j^k)$ is the Euclidean distance between a_i^k and b_j^k.

H. A lt, "The computational geometry of comparing shapes, " in EffiCient Algorithms. Springer, 2009, pp. 235-248
Xie D, Li F, Phillips JM. Distributed Trajectory Similarity Search. *Pvldb*. 2017:1478-1489.

Comparison

14

Measurement	Parameters	Applicable scope	Anti-noise property	Computational complexity
Euclidean distance	Parameter-free	The length of two trajectories must be the same	Weakest	O(n)
PCA + Euclidean distance	Parameter-free	The length of two trajectories must be the same	Weaker	O(n)
Hausdorff distance	Parameter-free	It is applicable for most of trajectory data	Weaker	O(m*n) Can be optimized to O(m+n) in a convx setting
LCSS distance	σ and ε (distance threshold of x and y direction)	It is applicable for most of trajectory data except the discrete trajectory data	Strong	O(m*n)
DTW distance	Parameter-free	Trajectory must be continuous and there does not exist completely dissimilar trajectory range in trajectories	Weaker	O(m*n)
Fréchet distance	Parameter-free	Trajectory data is discrete or continuous	Weaker	O(m*n) $O(mn \cdot \log(mn))$

Comparison

15

- Evaluation Method
  - Given a set of labeled trajectories T
  - Utilize one nearest neighbor (1NN) classifier to evaluate
    - Underlying distance metric is critical to the performance of 1NN classifier
    - 1NN classifier is parameter free

Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E. Querying and Mining of Time Series Data : Experimental Comparison of Representations and Distance Measures. 2008.

Comparison

16

- Evaluation Method

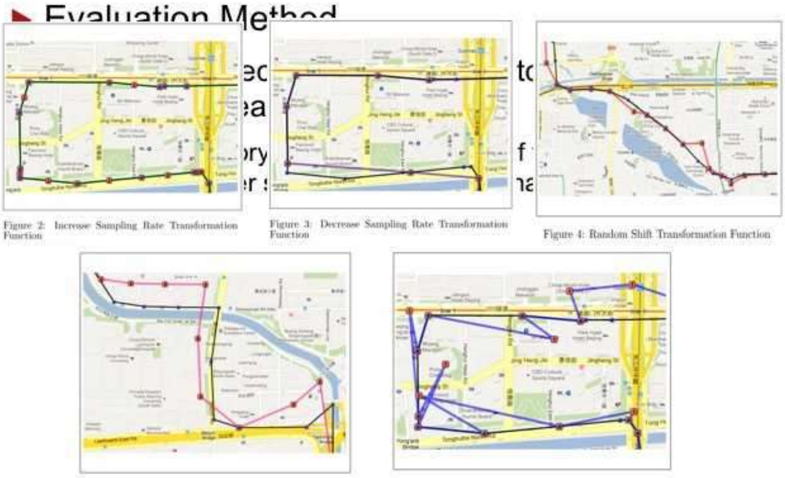


Figure 2: Increase Sampling Rate Transformation Function

Figure 3: Decrease Sampling Rate Transformation Function

Figure 4: Random Shift Transformation Function

Figure 5: Synchronized Shift Transformation Function

Figure 6: Add Noise Transformation Function

Wang H, Sadiq S. An Effectiveness Study on Trajectory Similarity Measures(ADC 2013).pdf. 2013;137(February):13-22.

Outline

17

- Accumulation based Distance
  - Euclidean, DTW, LCSS, EDR
- Point based Distance
  - Hausdorff distance, Frechet distance
- Segment based Distance
  - One Way Distance, LIP distance
- Task Specific Distance
  - TRACLUS, road, semantic, grid
- Conclusion

One Way Distance

18

- One Way Distance of trajectory T1 and T2:
  - Integral of the distance from points of T1 to T2, divided by the length of T1

$$D_{\text{owd}}(T_1, T_2) = \frac{1}{|T_1|}\left(\int_{p\in T_1} D_{\text{point}}(p, T_2)\, dp\right)$$

$$D_{\text{point}}(p, T) = \min_{q\in T} D_{\text{Euclid}}(p, q)$$

  - Symmetric measure

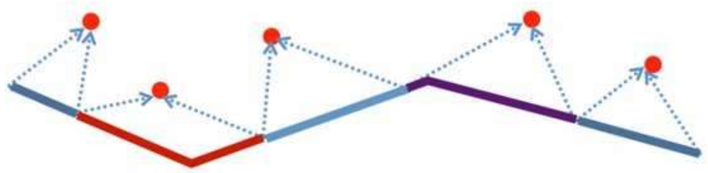
$$D(T_1, T_2) = \frac{1}{2}\left(D_{\text{owd}}(T_1, T_2) + D_{\text{owd}}(T_2, T_1)\right)$$

Bin Lin, Jianwen Su, One Way Distance: For Shape Based Similarity Search of Moving Object Trajectories. In Geoinformatica (2008)

One Way Distance

19

- Consider one trajectory as piece-wise line segment, and the other as discrete samples



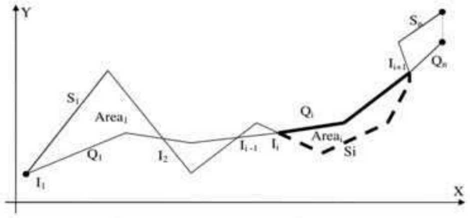
LIP distance

20

- Locality In-between Polylines

$$LIP(Q,S) = \sum_{\forall\, polygon_i} Area_i \cdot w_i \qquad w_i = \frac{Length_Q(I_i, I_{i+1}) + Length_S(I_i, I_{i+1})}{Length_Q + Length_S}$$

  - *Polygon* is the set of polygons formed between intersection points(Only work for 2-dimensional trajectories)



Nikos Pelekis et al, Similarity Search in Trajectory Databases. Symposium on Temporal Representation and Reasoning 2007

Outline

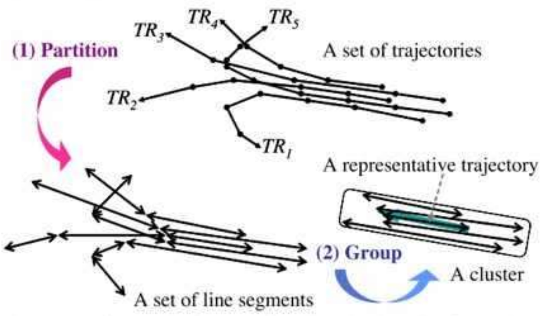
21

- Accumulation based Distance
  - Euclidean, DTW, LCSS, EDR
- Point based Distance
  - Hausdorff distance, Frechet distance
- Segment based Distance
  - One Way Distance, LIP distance
- Task Specific Distance
  - TRACLUS, road, clue, semantic, grid
- Conclusion

TRACLUS

22

- A trajectory clustering approach that considers *sub-trajectories*
  - Parts of trajectories might match even when the trajectory as a whole does not

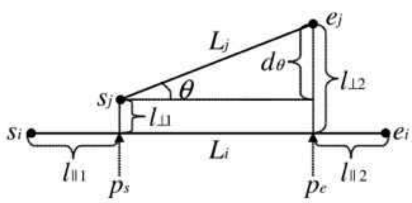


Lee J-G, Han J, Whang K-Y. Trajectory clustering. *Proc 2007 ACM SIGMOD Int Conf Manag data - SIGMOD '07*. 2007:593. doi:10.1145/1247480.1247546.

TRACLUS

23

- Define perpendicular, parallel and angle distances for segment lines
- Then, weight sum the distances as the distance of segment



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

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

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

TRACLUS

24

- Partition: Trajectory to Sub-trajectory
  - minimum description length (MDL)

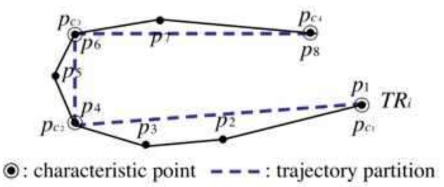
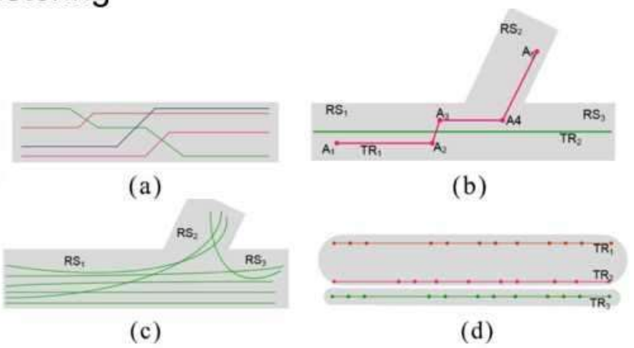


Figure 6: An example of a trajectory and its trajectory partitions.

NEAT-road network aware

25

- Pervious works on trajectory do not consider the road network factor which leads to bad trajectory clustering



Han B, Liu L, Omiecinski E. Road-network aware trajectory clustering: Integrating locality, flow, and density. *IEEE Trans Mob Comput*. 2015;14(2):416-429. doi:10.1109/TMC.2013.119.

NEAT-road network aware

26

- Three Phase trajectory clustering
  - Base Cluster Formation
  - Flow cluster formation
  - Flow Cluster Refinement

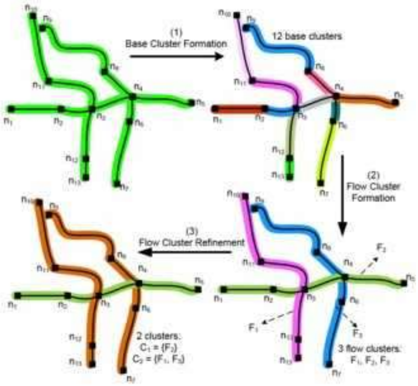


Fig. 3. Example of three phase clustering in the NEAT framework.

NEAT-road network aware

27

- Three Phase trajectory clustering
  - Base Cluster Formation
    - Segment trajectory by road network
    - Assign each fragment to a road segment by map matching
  - Flow cluster formation
    - Choose the density core as initial cluster S
    - Merge the neighbor cluster into S by computing flow factor q, density factor k and speed limit factor v
    - Weight sum above factors and using a threshold to merge cluster
  - Flow Cluster Refinement
    - Calculate the shortest path base Hausdorff distance between clusters and then Optimization clusters by DBSCAN

NEAT-road network aware

28

Definition 5 (netflow). *The netflow between two base clusters S_i and S_j, denoted by $f(S_i, S_j)$, is the number of trajectories participating in both clusters: $f(S_i, S_j) = |PTr(S_i) \cap PTr(S_j)|$. The function netflow between two base clusters computes the number of common objects traveled on both representative road segments e^{S_i} and e^{S_j}.*

Definition 6 (f-neighborhood). *Let B denote a set of base clusters, S_i denote a base cluster and n_u denote one endpoint of e^{S_i}. The f-neighborhood of S_i wrt. n_u, denoted by $N_f(S_i, n_u)$, is the set of base clusters that have at least one common participating trajectory, and is formally defined as: $N_f(S_i, n_u) = \{S_j \mid e^{S_j} \in L_{n_u}(e^{S_i})\ \&\ f(S_i, S_j) > 0\}$. Let n_v be the other endpoint of e^{S_i}. We define the f-neighborhood of S_i wrt. e^{S_i} as: $N_f(S_i) = N_f(S_i, n_u) \cup N_f(S_i, n_v)$. Each $S_j \in N_f(S_i)$ is called the f-neighbor of S_i. Note that the f-neighbor is a symmetric relation.*

Definition 9. *Given a base cluster S and n_u as one endpoint of road segment e^S, the flow factor q, density factor k and speed limit factor v of a base cluster $S_j \in N_f(S, n_u)$ wrt. S are defined respectively as follows:*

$$q = f\left(S, S_j\right) / |PTr(S)| \tag{1}$$

$$k = d\left(S_j\right) / (d(S) + \sum_{S_i \in N_f(S, n_u)} d\left(S_i\right)) \tag{2}$$

$$v = speed\left(S_j\right) / \sum_{S_i \in N_f(S, n_u)} speed\left(S_i\right) \tag{3}$$

where $speed(S_j)$ is the speed limit of e^{S_j}.

Definition 10. *Given a base cluster S and n_u as one endpoint of e^S, the merging selectivity of a base cluster $S_j \in N_f(S, n_u)$ is defined as:*

$$SF(S, S_j) = w_q \times q + w_k \times k + w_v \times v \tag{4}$$

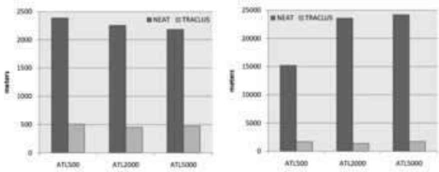
where the coefficients w_q, w_k and w_v determine the weights of q, k, v respectively. The weights $w_q \geq 0$, $w_k \geq 0$ and $w_v \geq 0$ satisfy $w_q + w_k + w_v = 1$.

NEAT-road network aware

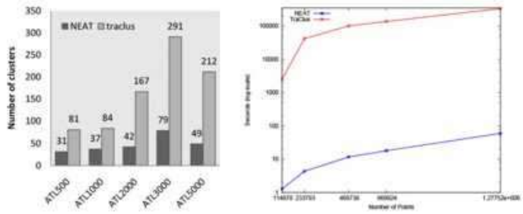
29

(a) (b)

Fig. 7. TraClus. (a) 81 clusters for ATL500 (ε=10m, *MinLns*=30). (b) 460 clusters for ATL500 (ε=1m, *MinLns*=1).



(a) (b)

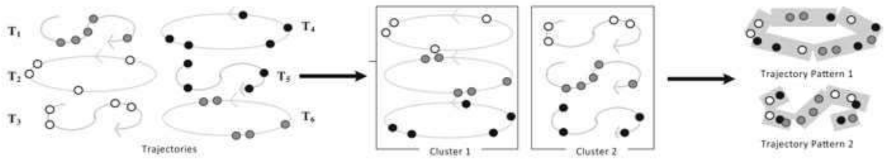


(c) (d)

Clue-aware trajectory similarity

30

- "clues" referring to those spatially and temporally co-located data points among trajectories
- The concept of clues is to evaluate how many such co-located points exist between two trajectories
  - closer points → stronger clues
  - co-located points reveal clues → occurrence times close(tolerate such temporal shifting)



Hung CC, Peng WC, Lee WC. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. VLDB J. 2015;24(2):169-192. doi:10.1007/s00778-011-0262-6.

Clue-aware trajectory similarity

31

- Clue-aware trajectory similarity

Definition 4 Spatial Decaying Function: Given a spatial threshold ϵ and two data points $p_{i,\ell} = (l_{i,\ell}, t_{i,\ell})$ and $p_{j,k} = (l_{j,k}, t_{j,k})$ from two trajectories (i.e., T_i and T_j), a spatial decaying function for two points $p_{i,\ell}$ and $p_{j,k}$ is defined as

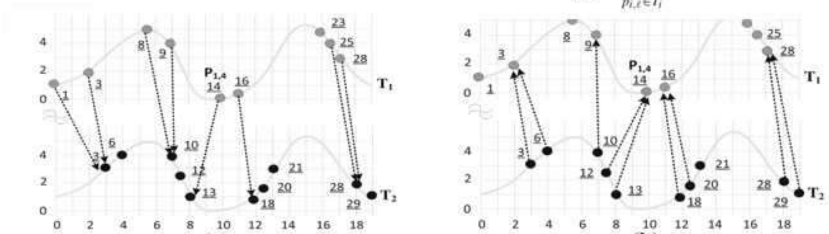
$$f_\epsilon(p_{i,\ell}, p_{j,k}) = \begin{cases} 0 & \text{, if } \mathrm{dist}(p_{i,\ell}, p_{j,k}) > \epsilon \\ 1 - \frac{\mathrm{dist}(p_{i,\ell}, p_{j,k})}{\epsilon} & \text{, otherwise} \end{cases}$$

where $\mathrm{dist}(\cdot,\cdot)$ denotes Euclidean distance between two data points.

Definition 5 Clue score of data points: Given a point $p_{i,\ell}$, a reference trajectory T_j, a spatial threshold ϵ, and a temporal threshold τ, the clue score of data point $p_{i,\ell}$ to trajectory T_j is defined as $score_{\epsilon,\tau}(p_{i,\ell}, T_j) = \max\{f_\epsilon(p_{i,\ell}, p_{j,k}) | p_{j,k} \in T_j \text{ and } t_{j,k} \in [t_{i,\ell} - \tau, t_{i,\ell} + \tau]\}$.

Definition 6 Clue-aware trajectory similarity: Given a spatial threshold ϵ and a temporal threshold τ, the clue-aware trajectory similarity from T_i to T_j is defined below:

$$CATS_{\epsilon,\tau}(T_i, T_j) = \frac{1}{|T_i|} \times \sum_{p_{i,\ell} \in T_i} score_{\epsilon,\tau}(p_{i,\ell}, T_j).$$

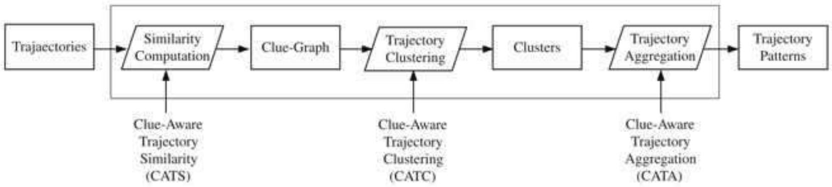


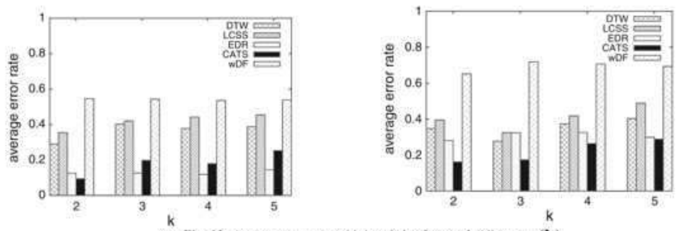
(a) (b)

Clue-aware trajectory similarity

32

- Clue-aware trajectory clustering





(a) Fig. 12 Average error rates with k varied. a 3 types. b All types (b)

Semantic Trajectory

33

- Semantic Trajectory: trajectory sequence includes spatial, temporal, semantic information
  - Semantic information→ a set of key words
- Similarity Computation
  - Combine the text similarity and geographical distance together
- Experiment
  - Time cost per round (seconds)

Zheng B, Yuan NJ, Zheng K, Xie X, Sadiq S, Zhou X. Approximate keyword search in semantic trajectory database. In: *Proceedings - International Conference on Data Engineering*. Vol 2015-May. ; 2015:975-986. doi:10.1109/ICDE.2015.7113349.

编辑于 2017-12-06

▲ 26 ▼ 2 条评论 分享 收藏 感谢 ··· 收起

张华
沉默的大多数

3 人赞同了该回答

两条轨迹是图像还是3D加速度传感器的数据？

图像的话通常做法是归一化、提取图像特征，然后计算图像距离，这个距离不是基于拓扑的，但是通常使用效果非常好。

3D加速度传感器的数据就可以按照拓扑分类。也是通过距离。

发布于 2015-09-06

▲ 3 ▼ 3 条评论 分享 收藏 感谢 ···

球面平行线
自由职业

基础是圆弧的相似的判断。

发布于 2016-10-11

▲ 0 ▼ 添加评论 分享 收藏 感谢 ···

刘皮皮
哦在这停顿

5 人赞同了该回答

卷积神经网络 LeNet5
LeCun教授的很多论文是做这个的
1998-Gradient-based learning applied to document recognition

发布于 2015-01-01

▲ 5 ▼ 1 条评论 分享 收藏 感谢 ···

Uncle Leon

2 人赞同了该回答

你的轨迹data是什么样的？比如每条轨迹由多少个离散的点构成，然后画出来这条轨迹，这样的话可以用procrustes distance来计算相似度，从变化角度的方面考虑，当然也可以加入cosine similarity，自己简单的做个linear的加权。。

发布于 2015-09-06

▲ 2 ▼ 添加评论 分享 收藏 感谢 ···

匿名用户

1 人赞同了该回答

本科毕设做过类似的工作，基于加速度传感器的手势识别，看过一个方法，用的是旋转特征，假设每条曲线n个点的坐标你有了，两两相连，你就有了n-1条线段了，每两条线段之间做叉乘，根据正负你就知道线段之间的旋转趋势了，共有n-2个值，然后再让正值为1负值为-1就有了n-2个1，-1串，比如有两段线段，你算出来分别是：
1,1,1,1,1,1,-1,-1,-1,-1,-1,1,1,1,-1,-1,-1
1,1,1,1,-1,-1,1,1,-1-1
然后你在想办法衡量这两个的相似性。。。

以前看的那篇文献中好像标记的特征是从第i象限转向第j象限这个二元特征{i,j}，然后用编辑距离来衡量。。好像是

我自己的研究最后用的是DTW，动态时间规整，可以衡量两端持续时间不一样的曲线的相似性的算法，前提是你特征要选的好。。

编辑于 2015-01-01

▲ 1 ▼ 6 条评论 分享 收藏 感谢 ···

beanfrog

2 人赞同了该回答

或许可以参照签名验证的方法，通过训练Siamese Network来学习一个距离度量，或者尝试一下其他的度量学习（Metric Learning）的方法。

图片摘自论文：Signature verification using a “Siamese” time delay neural network

编辑于 2015-01-07

▲ 2 ▼ 添加评论 分享 收藏 感谢 ···

韩松

1 人赞同了该回答

类聚问题和模式识别问题会讲到。
不知道别人会怎么处理：我的方式是曲线离散化，然后通过计算Hausdor distance评价其匹配度。当然参数设置可能引起评价结果的不同，但是只要能将相似的图形从一大堆图形库中跳出来，目的就达到了。

发布于 2015-09-22

▲ 1 ▼ 添加评论 分享 收藏 感谢 ···

Greene
江南无所有，聊赠一枝春

目前还不是专业的
但是从非专业角度来看看

题主想问的是拓扑结构相似？还是变化趋势相似？
如果是拓扑结构相似的话，可以试试把端点和交点标记出来，环点特殊标记，然后对应着去比较，至于相似程度可以用最大连续相同拓扑结点数目来衡量。（例如每个点对应一个向量（是否端点，是否环点，邻接点个数，上级邻接点向量，下级邻接点向量），然后依次比较）
如果是变化趋势相似的话，可以试试用同一组平行线去切割两个曲线，从而得到两组点，然后比较这两组点构成的拓扑结构的相似程度，最后综合用于切割的平行线的密度来判断相似程度。

编辑于 2018-01-01

▲ 0 ▼ 添加评论 分享 收藏 感谢 ···

匿名用户

Shape context is enough.

Ref: en.m.wikipedia.org/wiki...

编辑于 2017-02-06

▲ 0 ▼ 添加评论 分享 收藏 感谢 ···

匿名用户

先给个定义

发布于 2015-01-06

▲ 0 ▼ 添加评论 分享 收藏 感谢 ···

微笑笑笑
本科设计留学生，会点Java，自己捣鼓MMORPG中

专业细节方面不懂，只是单纯开脑洞，砸个砖头等玉丶(￣д￣;)ノ

所谓的比较结构……我就理解成题主是想要比较线条的走向，但是忽略线条的长度吧？
如果是比较两个函数，那么可以考虑比较特定区间的斜率增与减和x/y的变化量

手机没办法po图，如果表述不清……我努力改(#ﾟДﾟ)

就说楼主说的这个图吧，很明显这不是个函数，但是可以将它拆成复数个函数来看
比如那个圈圈之前的部分比较一次，圈圈的下半部分比较一次，以此类推，最后将它量化

只适用于矢量图像，别的情况可以参考google的以图搜图，如果是用于手写的话还可以将笔画的顺序算入量化计算

具体细节就别问惹，我不懂[○・｀Д´・○]
已经做好了被点没有帮助的准备了

编辑于 2015-01-01

▲ 0 ▼ 1 条评论 分享 收藏 感谢 ···

写回答

2 个回答被折叠（为什么？）