ML Lecture 15:

Unsupervised Learning - Neighbor Embedding

> 臺灣大學人工智慧中心 科技部人工智慧技術暨全幅健康照護聯合研究中心 http://aintu.tw

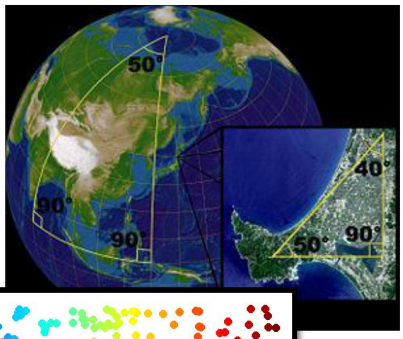
Neighbor Embedding

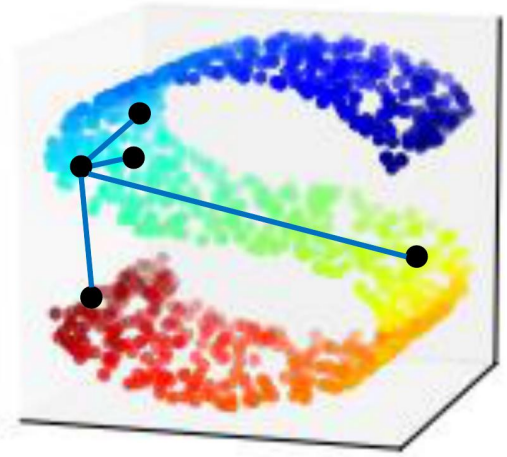
- t-SNE
- T-distributed Stochastic **Neighbor Embedding**

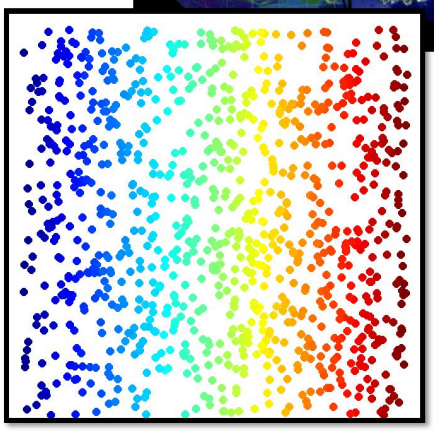
Dimension Reduction

- 「非線性」的降維
- 在高維空間裡面的一個 Manifold
- Ex: 地球
  - 表面就是一個 Manifold
  - 塞到了一個三維的空間裡面
  - Euclidean distance (歐式幾何) 只有在很近的距離的情況下才會成立
- Ex: 附圖之S形空間
  - 藍色區塊的點距離近 $\Rightarrow$ 他們比較像
  - 距離比較遠(ex: 藍色跟紅色) $\Rightarrow$ 無法直接以 Euclidean distance 計算相似度

Manifold Learning







Suitable for clustering or following supervised learning

Manifold Learning

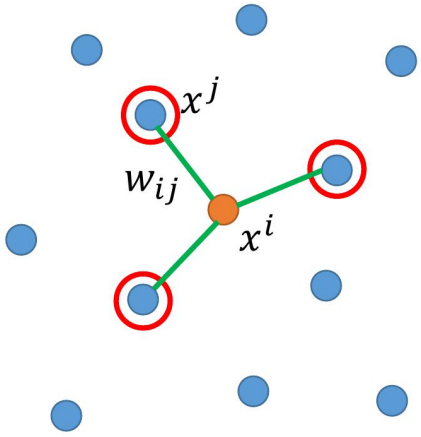
- 把 S 形的這塊東西展開
- 把塞在高維空間裡面的低維空間「攤平」，也就是降維
- Pros
  - 可以用 Euclidean distance 來計算點和點之間的距離
  - 對 clustering 有幫助
  - 對 supervised learning 也會有幫助

Locally Linear Embedding (LLE)

Setting

- 本來有某一個點，叫做 x_i
- 然後選出這個 x_i 的 k 個 neighbors
- 假設其中一個叫做 x_j ，w_{ij} 代表 x_i 和 x_j 的關係
- 表示所有的 k 個 neighbors Neighbor 之線性組合要跟 x_i 越像越好
- Minimize $\sum_i \|x^i - \sum_j w_{ij} x^j\|_2$

Locally Linear Embedding (LLE)



w_{ij} represents the relation between x^i and x^j

Find a set of w_{ij} minimizing

$$\sum_i \left\| x^i - \sum_j w_{ij} x^j \right\|_2$$

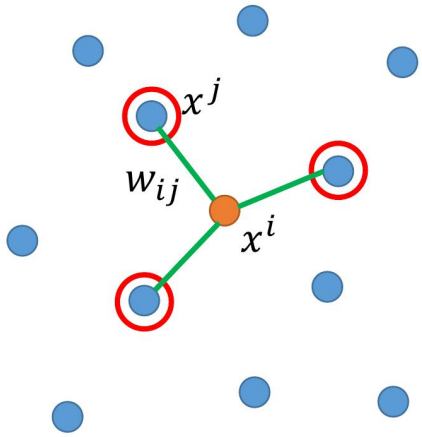
Then find the dimension reduction results z^i and z^j based on w_{ij}

Dimension Reduction

- 將所有的 x_i 跟 x_j 轉成 z_i 和 z_j ，而中間的關係 w_{ij} 是**不變**的
- 首先 w_{ij} 在原來的 space 上面找完以後，就 fix 住
- **沒有**一個明確的 function 說怎麼做 dimension reduction
- 憑空找出來降維後的 z_i 跟 z_j ，可能原本100維(x)，降到2維(z)
- Minimize $\sum_i \|z^i - \sum_j w_{ij} z^j\|_2$

LLE

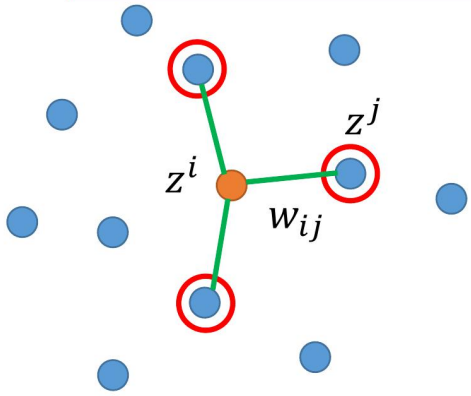
Keep w_{ij} unchanged



Original Space

Find a set of z^i minimizing

$$\sum_i \left\| z^i - \sum_j w_{ij} z^j \right\|_2$$



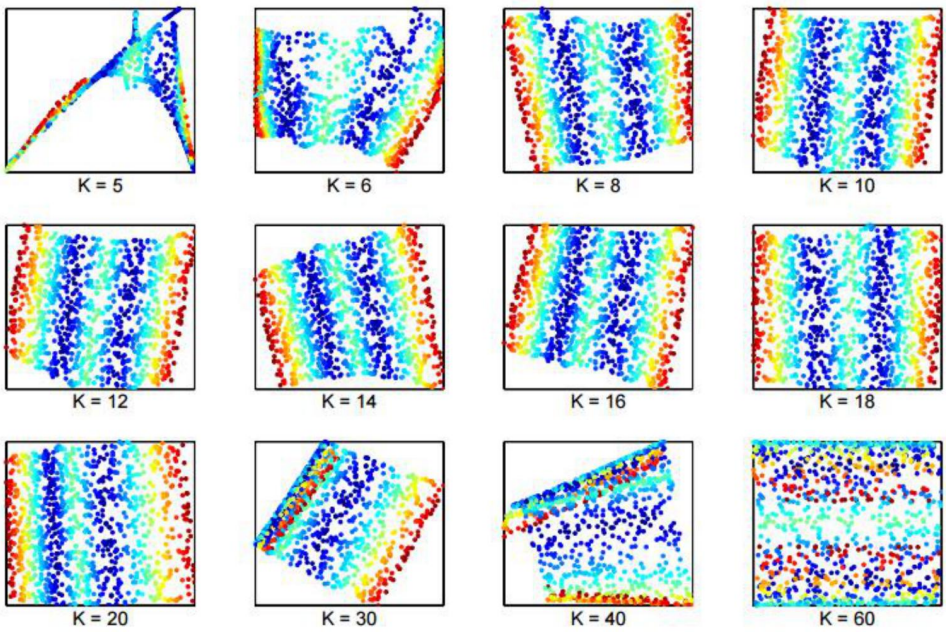
New (Low-dim) Space

Something about Numbers of Neighbors (k)

- neighbor 選的數目要剛剛好才會得到好的結果
- Reference paper: “Think Globally, Fit Locally”
- k 太小，就不太robust，表現不太好
- k 太大，會考慮到一些距離很遠的點，這些點被 transform 以後，relation 沒有辦法 keep 住

LLE

Lawrence K. Saul, Sam T. Roweis, “Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds”, JMLR, 2013



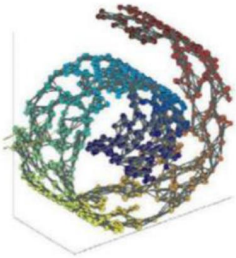
Laplacian Eigenmap

- 考慮到先前提過的 Smoothness assumption
- 只算它的 Euclidean distance 來比較點跟點之間的距離是不足夠的
- 要看在這個 High density 的 region 之間的 distance
  - 有 high density 的 connection 才是真正的接近
  - 可以用 graph 描述
- Graph Construction
  - 計算 data point 兩兩之間的相似度，超過一個 thereshold 就 connect 起來

Laplacian Eigenmaps

$$w_{i,j} = \begin{cases} \text{similarity} & \text{If connected} \\ 0 & \text{otherwise} \end{cases}$$

- *Review in semi-supervised learning*: If x^1 and x^2 are close in a high density region, $\hat{y}^1$ and $\hat{y}^2$ are probably the same.



$$L = \sum_{x^r} C(y^r, \hat{y}^r) + \lambda S$$

As a regularization term

$$S = \frac{1}{2}\sum_{i,j} w_{i,j}\left(y^i - y^j\right)^2 = \boldsymbol{y}^T L \boldsymbol{y}$$

L: (R+U) x (R+U) matrix

S evaluates how smooth your label is

Graph Laplacian

$$L = D - W$$

- Review
  - 如果 x_1 跟 x_2 在 high density 的 region 上面是相近的，那它們的 label y_1, y_2，很有可能是一樣的
- 考慮 smoothness 的距離，可以被這個 graph 上面的 connection 來 approximate
- 有一項是考慮有 label 的 data 的項 (L)
  - graph 的 laplacian
  - $E - W$
- 另外一項是跟 labeled data 沒有關係的 (S)
  - 那這一項的作用它要考慮說我們現在得到的 label 是不是 smooth 的 regularization 的 term
  - y_i 跟 y_j 的這個距離乘上 $w_{i,j}$
  - $w_{i,j}$ 就是 x_i 跟 x_j 的相似成度
  - evaluate 現在得到的 label 有多麼的 smooth
  - $y^T L\, y$

Laplacian Eigenmaps

- *Dimension Reduction*: If x^1 and x^2 are close in a high density region, z^1 and z^2 are close to each other.

$$S = \frac{1}{2}\sum_{i,j} w_{i,j}\left(z^i - z^j\right)^2$$

Any problem? How about $z^i = z^j = \mathbf{0}$?

Giving some constraints to z:

If the dim of z is M, Span{z^1, z^2, … z^N} = R^M

Spectral clustering: clustering on z

Belkin, M., Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems* . 2002

- 如果 x1 跟 x2 在 high density 的 region 他們是 close 的，那我們就會希望， z1 跟 z2 他們也是相近的

- 如果今天 x_i 跟 x_j，這個 i 跟 j 的這兩個 datapoint 他們之間的 $w_{i,j}$ 很像，那 z_i 跟 z_j 做完 dimension reduction 以後他們的距離就很近
- 反之，如果他們的 $w_{i,j}$ 很小，那他們的距離要怎樣就都可以

- Problem
  - minimize只要把所有的 z_i 跟 z_j 通通設一樣的值就好了? 直接通通變0?
  - 光這個式子是不夠的
  - 少了 supervise 的項，需要給一些 constraint
    - $Span\{z^1, z^2 \ldots, z^N\} = R^M$
    - z 不是活在一個比 M 維更低維的空間裡面
    - z 就是 graph laplacian 的 eigenvector
    - Laplacian Eigenmap
  - Spectral clustering
    - 先找出 z，再用 K-means 做 clustering

Laplacian Eigenmaps

- *Dimension Reduction*: If x^1 and x^2 are close in a high density region, z^1 and z^2 are close to each other.

$$S = \frac{1}{2}\sum_{i,j} w_{i,j}\left(z^i - z^j\right)^2$$

Any problem? How about $z^i = z^j = \mathbf{0}$?

Giving some constraints to z:

If the dim of z is M, Span{z^1, z^2, … z^N} = R^M

Spectral clustering: clustering on z

Belkin, M., Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems* . 2002

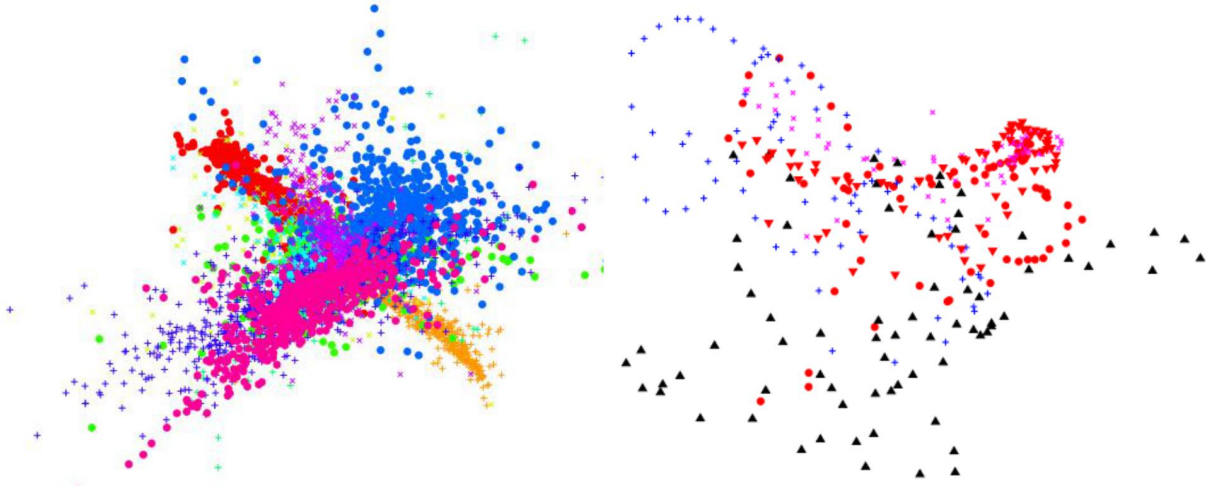
t-SNE (T-distributed Stochastic Neighbor Embedding)

Problems above

- 先前只假設相近的點應該要是接近的，但沒有假設**不相近的點不要接近、要分開**
- 確實會把同個 class 的點都聚集在一起，但也會把不同的class混雜
- Example: MINIST (手寫辨識)、 COIL-20 (Image Corpus)

T-distributed Stochastic Neighbor Embedding (t-SNE)

- Problem of the previous approaches
  - Similar data are close, but different data may collapse



LLE on MNIST　　LLE on COIL-20

t-SNE

- 做 t-SNE，一樣是要做降維
  - 把原來的 data point x 變成比較 low dimension 的 vector z
  - 在原來的 x 這個 space 上面會計算所有的點的 pair，x_i 和 x_j 之間的 similarity
    - 寫成 $S(x_i, x_j)$
    - 做一個 normalization，計算一個 $P(x_j|x_i)$
    - $P(x_j|x_i)$，他是從 x_i 跟 x_j 的 similarity 來的
      - 分子的地方是 x_i 跟 x_j 的 similarity
      - 分母的地方是 summation over 除了 x_i 以外，所有其他的點和 x_i 之間所算出來的距離

發現

- xi 對其他所有的 data point，它所算出來的這個 $P(x_j|x_i)$ 的 summation 應該要是 1。
- 另外假設我們今天已經找出了一個 low dimension 的 representation 就是 z_i 跟 z_j 的話，我們已經把 x 變成 z 的話，那我們也可以計算 z_i 和 z_j 之間的 similarity。
- 這邊 similarity 我們寫成 S'，那一樣你可以計算一個 $Q(z_j|z_i)$，他的分子的地方就是 $S'(z_i, z_j)$，分母的地方就是 $\sum z_i$，跟所有 database 裡面的 data point z_k 之間的距離。

Normalization

那今天有做這個 normalization 其實感覺是必要的，因為你不知道在高維空間中算出來的距離，$S(x_i, x_j)$ 跟 $S'(z_i, z_j)$，它們的 scale 是不是一樣的

如果，你今天有做這個 normalization，那你最後就可以把他們都變成機率，那這個時候他們值都會介於 0 到 1 之間，他們的 scale 會是一樣的

找z

我們現在還不知道 z_i 跟 z_j 他們的值到底是多少，那麼希望找一組 z_i 跟 z_j，它可以讓這一個 distribution 跟這一個 distribution 越接近越好

我們要讓原來在這個根據 similarity 在 S 這個原來的 space 算出來 distribution，跟在這個 dimension reduction 以後的 space 算出來的 distribution **越接近越好**。

怎麼衡量兩個 distribution 之間的相似度呢？

- KL divergence
  - 可以很直覺的衡量兩個 distribution 之間的相似度的方法
  - 也就是找一組 z，可以讓 x_i 的這個 distribution，跟 x_i 對其他 point 的 distribution。跟 z_i 對其他 point 的 distribution，這兩個 distribution 之間的 KL divergence 越小越好
  - Summation over 所有的 data point，然後你要使得這一項，它的值越小越好

t-SNE x → z

Compute similarity between all pairs of x: $S(x^i, x^j)$

$$P(x^j|x^i) = \frac{S(x^i, x^j)}{\sum_{k \neq i} S(x^i, x^k)}$$

Compute similarity between all pairs of z: $S'(z^i, z^j)$

$$Q(z^j|z^i) = \frac{S'(z^i, z^j)}{\sum_{k \neq i} S'(z^i, z^k)}$$

Find a set of z making the two distributions as close as possible

$$L = \sum_i KL\left(P(*|x^i)||Q(*|z^i)\right)$$

$$= \sum_i \sum_j P(x^j|x^i) log \frac{P(x^j|x^i)}{Q(z^j|z^i)}$$

Problem

- 因為要計算所有 data point 之間的 similarity，所以運算量很大

Solution

- 常見的做法是先做降維
  - 先用比較快的方法比如說 PCA，比如說，先降到 50 維
  - 再用 t-SNE 從 50 維降到 2 維

Feature

- 少用來training
- 多拿來做 visualization
  - visualize 他們在二維空間的分佈上是什麼樣子

t-SNE 之 similarity 的選擇

- 原來: 選擇 RBF 的 function
  - 要在 graph 上算 similarity 的話，這種方法比較好
  - 可以確保說只有非常相近的點才有值

- SNE
  - 在 data point 原來的 space 上用這個 evaluation 的 measure
- t-SNE
  - 在 dimension reduction 以後的 space，選擇的 measure 跟原來的 space 是不一樣的
  - 選的 space 是這個 t-distribution 的其中一種

Why different measure

假設橫軸代表了在原來 space 上的 Euclidean distance，紅線是 RBF function，藍線是 t-distribution。

- 如果本來距離比較近
  - 他們的影響是比較小的
  - 做完 transform 以後他還是很近
- 如果本來就已經有一段距離，那從原來的這個 distribution 變到 t-distribution 以後
  - t-distribution 他的尾巴特別長，他會被拉得很遠
  - 做完 transform 以後他就會被拉得很遠

Ignore σ for simplicity

t-SNE –Similarity Measure

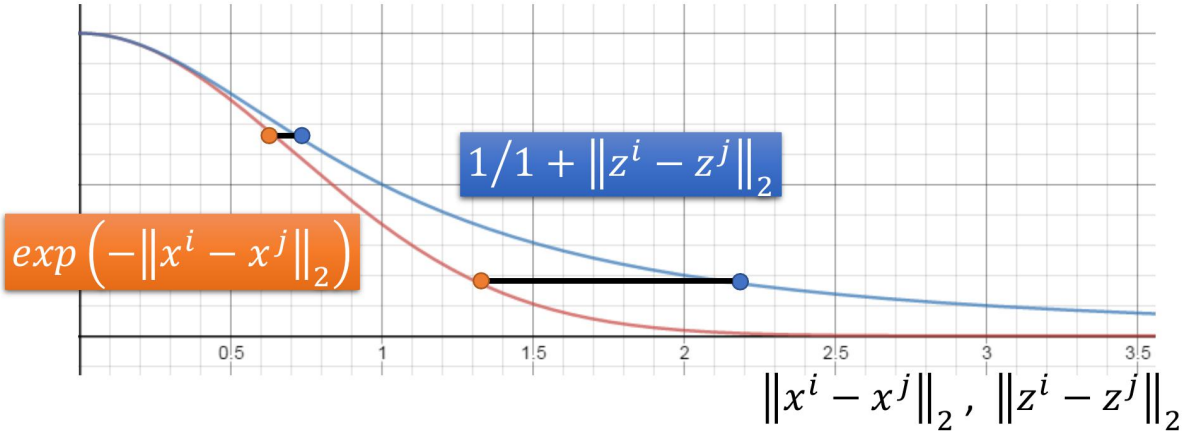
$$S(x^i, x^j) = exp\left(-\left\|x^i - x^j\right\|_2\right)$$

SNE:

$$S'(z^i, z^j) = exp\left(-\left\|z^i - z^j\right\|_2\right)$$

t-SNE:

$$S'(z^i, z^j) = 1/1 + \left\|z^i - z^j\right\|_2$$



T-SNE Visualization Example

發現

t-SNE 畫出來的圖往往長得像這樣，他會把你的 data point 聚集成一群、一群、一群

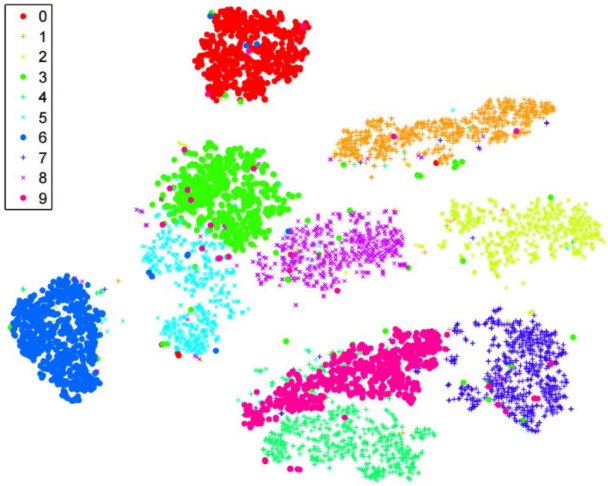
因為 data point 之間本來只要有一個 gap，做完 t-SNE 以後，它就會**把 gap 強化**，gap 就會變得特別明顯

Example

- MNIST
  - 先對 pixel 做 PCA 降維以後，再做 t-SNE
  - 直接做 t-SNE 效果不好
- COIL-20
  - 每一個圈圈就代表一個 object，他只是圖片轉了一圈以後的結果
  - 有一些被扭曲的圈，是因為有一些東西其不同角度特徵差很多，比如說杯子
  - 左下角這很多條線，好像是說，有 4 部都是小汽車，4 部小汽車看起來是蠻像的。所以這邊有四條線是被 align 在一起的

t-SNE

- Good at visualization



t-SNE on MNIST

t-SNE on COIL-20

Appendix

To learn more …

- Locally Linear Embedding (LLE): [Alpaydin, Chapter 6.11]
- Laplacian Eigenmaps: [Alpaydin, Chapter 6.12]
- t-SNE
  - Laurens van der Maaten, Geoffrey Hinton, “Visualizing Data using t-SNE”, JMLR, 2008
  - Excellent tutorial: https://github.com/oreillymedia/t-SNE-tutorial