Outline

- Independent Component Analysis
- Sparse Coding
- Multidimensional Scaling
- Isomap
- t-distributed stochastic neighbor embedding (t-SNE)
- Autoencoder

Examples of Embedding methods

Learning Embedding

Problem definition:

- Given set X of input samples, find a mapping from input to embedding X⟶ Y such that Y reflects semantics or similarity/neighborhood structures.

$$\mathcal{X} = \left\{x^{(1)}, x^{(2)}, \ldots, x^{(N)} \in \mathbb{R}^h\right\} \rightarrow \mathcal{Y} = \left\{y^{(1)}, y^{(2)}, \ldots, y^{(N)} \in \mathbb{R}^l\right\}$$

- The embedding may be a parameterized function (e.g., y = f(x) by a neural network), or alternatively, you can directly optimize Y (i.e., all embedding coordinates) corresponding to the entire input dataset X.
- The specific formulation differs depending on the objective function.

Learning Embedding

Examples of methods for learning embedding.

- Multidimensional Scaling
- ISOMAP
- tSNE (t-Distributed Stochastic Neighbor Embedding)
- Autoencoder

Here, we will primarily cover methods used for visualization via dimensionality reduction of the original data

Multidimensional scaling

Multidimensional scaling (MDS)

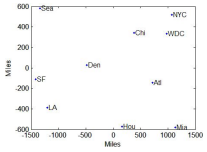
- Given pairwise distances between data points, MDS tries to "recover" the linear (Euclidean) embedding that can preserve the original distances (as much as possible).
  - i.e., Any pairwise distance -> Euclidean space
- Often used for data visualization, or dimensionality reduction
- Can be used for nonlinear dimensionality reduction (ISOMAP; covered later)

Multidimensional scaling (MDS)

- Given pairwise distances between data points (cities in the USA), can you locate the points (cities) in a 2D map?

	Atl	Chi	Den	Hou	LA	Mia	NYC	SF	Sea	WDC
Atl	0	587	1212	701	1936	604	748	2139	2182	543
Chi	587	0	920	940	1745	1188	713	1858	1737	597
Den	1212	920	0	879	831	1726	1631	949	1021	1494
Hou	701	940	879	0	1374	968	1420	1645	1891	1220
LA	1936	1745	831	1374	0	2339	2451	347	959	2300
Mia	604	1188	1726	968	2339	0	1092	2594	2734	923
NYC	748	713	1631	1420	2451	1092	0	2571	2408	205
SF	2139	1858	949	1645	347	2594	2571	0	678	2442
Sea	2182	1737	1021	1891	959	2734	2408	678	0	2329
WDC	543	597	1494	1220	2300	923	205	2442	2329	0

- Desired output:



Algorithm for MDS

- Given pairwise distances *D* (note: x is unknown)

$$\mathbf{D}_{ij} = \left\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\right\|^2$$

- Compute pairwise inner products (Gram matrix)

$$\mathbf{B} = -\frac{1}{2}\mathbf{HDH}$$

$$\text{where } \mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$$

1: a Nx1 vector of all ones
1 **1**$^\top$: outer product of **1** and **1**$^\top$, which is a NxN matrix of all ones

  - Then it can be shown that $\mathbf{B} = \mathbf{X}^\top\mathbf{X}$. where $\mathbf{X} : N_{\text{dim}} \times N_{\text{examples}}$
- Embed with *Y* (i.e. approximate $\mathbf{B} \approx \mathbf{Y}^\top\mathbf{Y}$)
  - SVD: $\mathbf{B} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$
- Solution: $\mathbf{Y} = \sqrt{\mathbf{\Lambda}}\mathbf{V}^\top$
  - Often truncate with top-K eigenvectors/eigenvalues

Detailed Derivation for MDS

$$HDH = \left(I - \frac{1}{N}11^T\right)D\left(I - \frac{1}{N}11^T\right) = D - \frac{1}{N}11^TD - D\frac{1}{N}11^T + \frac{1}{N}11^TD\frac{1}{N}11^T$$

$$\begin{aligned}
(HDH)_{ij} &= D_{ij} - \frac{1}{N}\left(1^TD\right)_j - \frac{1}{N}\left(D1\right)_i + \frac{1}{N^2}\sum_{kl}D_{kl} \\
&= \left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) \\
&\quad -\frac{1}{N}\sum_i\left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) - \frac{1}{N}\sum_j\left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) \\
&\quad +\frac{1}{N^2}\sum_{ij}\left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) \\
&= \left(||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)}\right) - \frac{1}{N}\sum_i\left(||x^{(i)}||^2 + ||x^{(j)}||^2\right) - \frac{1}{N}\sum_j\left(||x^{(i)}||^2 + ||x^{(j)}||^2\right) \\
&\quad +\frac{1}{N^2}\sum_{ij}\left(||x^{(i)}||^2 + ||x^{(j)}||^2\right) \\
&= ||x^{(i)}||^2 + ||x^{(j)}||^2 - 2x^{(i)T}x^{(j)} - \frac{2}{N}\sum_i||x^{(i)}||^2 - ||x^{(j)}||^2 - ||x^{(i)}||^2 + \frac{2}{N}\sum_i||x^{(i)}||^2 \\
&= -2x^{(i)T}x^{(j)}
\end{aligned}$$

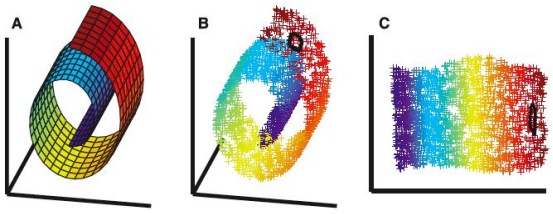
Summary: MDS

- MDS is a "linear" dimensionality reduction method
  - If the original distance is defined over the Euclidean space, then MDS can at best recover the original space (or approximation with reduced dimensionality)
- However, MDS can be combined with nonlinear distance metric to discover "manifold" structure in the data (ISOMAP)

ISOMAP

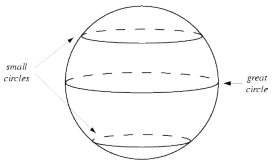
Nonlinear Manifolds

- PCA fails on seriously nonlinear manifolds like the "Swiss roll"

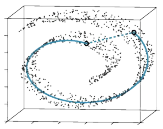


Isometric Feature Mapping (ISOMAP)

- Geodesic: the shortest curve on a manifold that connects two points on the manifold
  - e.g. on a sphere, geodesics are great circles



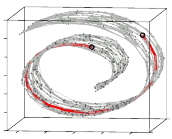
- Geodesic distance: length of the geodesic



- Points far apart measured by geodesic dist. appear close measured by Euclidean dist.

ISOMAP

- Take a distance matrix as input
- Construct a weighted graph G based on neighborhood relations
- Estimate pairwise geodesic distance by "a sequence of short hops" on G
- Apply MDS to the geodesic distance matrix
  - MDS "unfolds" the manifold into Euclidean space

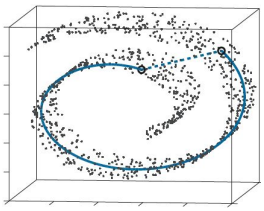


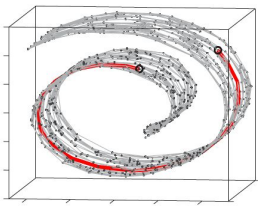
Isomap

- Define local neighborhoods by small distance or *k*-nearest-neighbors.
  - Link each pair of points in a neighborhood with their distance in the neighborhood.
- Distance between all other pairs is shortest path distance in the connectivity graph.
- Compute eigenvalues; select dimension.
- Do multidimensional scaling into desired low-dimensional space.

Isomap: the Swiss roll

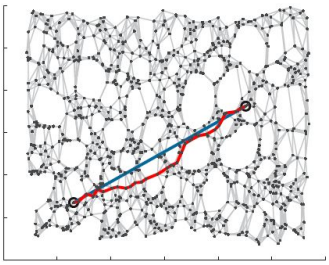
- Distance may be longer along a geodesic in the manifold than in the embedding space.





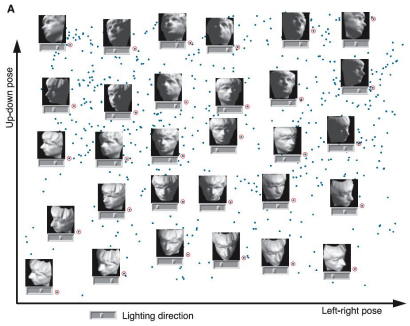
Unrolling the Swiss Roll

- The resulting 2D structure reflects the geodesic distances along the manifold.



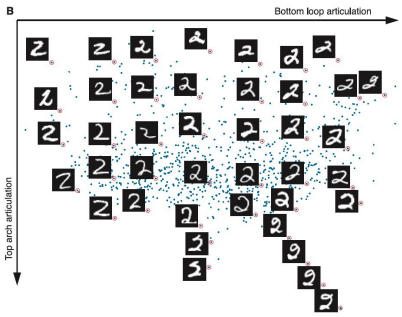
Faces: pose x illumination

- 64x64 = 4096 dimensions



Hand-written "2"s

- Mapping groups the digits by "style"



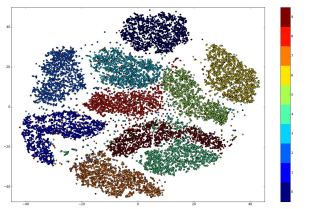
Summary: ISOMAP

- Nonlinear dimensionality reduction methods can be used to find intrinsic manifold structure from the data; also useful for visualization
- Limitation:
  - computationally quite expensive; doesn't scale to very large data
  - Requires dense data points for good estimates
  - Sensitive to noise
- Other related algorithms:
  - Hessian LLE, Laplacian eigenmap, etc.

tSNE (t-Distributed Stochastic Neighbor Embedding)

Data visualization with embeddings

- Embeddings are vector representations that reflect semantic meaning in feature space (i.e., feature vectors live in neighborhoods based on meaning)
- We can visualize these with techniques such as t-SNE

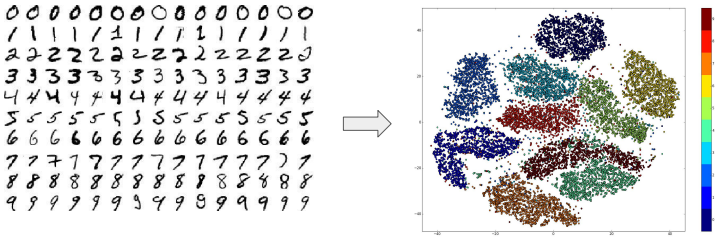


tSNE visualization of handwritten digits (MNIST) in 2d space.

Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov), 2579-2605.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

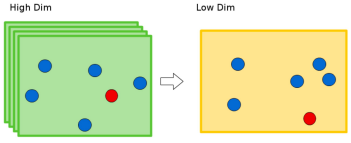
- Given a collection of N high-dimensional data $x^{(1)}, x^{(2)}, \ldots x^{(N)}$, how can we get a sense of how they are arranged in data space?



Slide credit: Kai-Wen Zhao

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Compress the high dimensional data into a lower dimensional space
- We want to preserve distance and neighborhood structure



Slide credit: Kai-Wen Zhao

Preliminary: Stochastic Neighbor Embedding (SNE)

SNE converts euclidean distances to similarities, that can be interpreted as probabilities P^i over neighbors of x^i. Then we find embedding Q^i that approximates P^i.

$P^i = \{p^{1|i}, p^{2|i}, ..., p^{N|i}\}$ and $Q^i = \{q^{1|i}, q^{2|i}, ..., q^{N|i}\}$ are the distributions on the neighbors (e.g., transition prob.) of datapoint *i*.

$$p^{j|i} = \frac{\exp\left(-\left\|x^{(i)} - x^{(j)}\right\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\left\|x^{(i)} - x^{(k)}\right\|^2 / 2\sigma_i^2\right)}$$

$$q^{j|i} = \frac{\exp\left(-\left\|y^{(i)} - y^{(j)}\right\|^2\right)}{\sum_{k \neq i} \exp\left(-\left\|y^{(i)} - y^{(k)}\right\|^2\right)}$$

$$p^{i|i} = 0, \; q^{i|i} = 0$$

SNE minimizes the following cost:

$$C = \sum_i KL\left(P^i \| Q^i\right) = \sum_i \sum_j p^{j|i} \log \frac{p^{j|i}}{q^{j|i}}$$

Slide credit: Simon Carbonnelle

t-Distributed Stochastic Neighbor Embedding (t-SNE)

SNE

Modelisation:

$$p^{j|i} = \frac{\exp\left(-\|x^{(i)} - x^{(j)}\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x^{(i)} - x^{(k)}\|^2 / 2\sigma_i^2\right)}$$

$$q^{j|i} = \frac{\exp\left(-\|y^{(i)} - y^{(j)}\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y^{(i)} - y^{(k)}\|^2\right)}$$

Cost Function:

$$C = \sum_i KL(P_i||Q_i)$$

Derivatives:

$$\frac{dC}{dy} = 2\sum_j \left(p^{j|i} - q^{j|i} + p^{i|j} - q^{i|j}\right)\left(y^{(i)} - y^{(j)}\right)$$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

SNE ⇨ Symmetric SNE

Modelisation:

$$p^{j|i} = \frac{\exp\left(-\|x^{(i)} - x^{(j)}\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x^{(i)} - x^{(k)}\|^2 / 2\sigma_i^2\right)}$$

$$q^{j|i} = \frac{\exp\left(-\|y^{(i)} - y^{(j)}\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y^{(i)} - y^{(k)}\|^2\right)}$$

Cost Function:

$$C = \sum_i KL(P_i||Q_i)$$

Derivatives:

$$\frac{dC}{dy} = 2\sum_j \left(p^{j|i} - q^{j|i} + p^{i|j} - q^{i|j}\right)\left(y^{(i)} - y^{(j)}\right)$$

Modelisation:

$$p^{ij} = \frac{p^{j|i} + p^{i|j}}{2N}$$

$$q^{ij} = \frac{\exp\left(-\|y^{(i)} - y^{(j)}\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y^{(k)} - y^{(l)}\|^2\right)}$$

Cost Function:

$$C = KL(P||Q)$$

Derivatives:

$$\frac{dC}{dy} = 4\sum_j \left(p^{ij} - q^{ij}\right)\left(y^{(i)} - y^{(j)}\right)$$

→ Faster optimization!

t-Distributed Stochastic Neighbor Embedding (t-SNE)

SNE ⇨ Symmetric SNE ⇨ t-SNE

Modelisation:

$$p^{j|i} = \frac{\exp\left(-\|x^{(i)} - x^{(j)}\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x^{(i)} - x^{(k)}\|^2 / 2\sigma_i^2\right)}$$

$$q^{j|i} = \frac{\exp\left(-\|y^{(i)} - y^{(j)}\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y^{(i)} - y^{(k)}\|^2\right)}$$

Cost Function:

$$C = \sum_i KL(P_i||Q_i)$$

Derivatives:

$$\frac{dC}{dy} = 2\sum_j \left(p^{j|i} - q^{j|i} + p^{i|j} - q^{i|j}\right)\left(y^{(i)} - y^{(j)}\right)$$

Modelisation:

$$p^{ij} = \frac{p^{j|i} + p^{i|j}}{2N}$$

$$q^{ij} = \frac{\exp\left(-\|y^{(i)} - y^{(j)}\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y^{(k)} - y^{(l)}\|^2\right)}$$

Cost Function:

$$C = KL(P||Q)$$

Derivatives:

$$\frac{dC}{dy} = 4\sum_j \left(p^{ij} - q^{ij}\right)\left(y^{(i)} - y^{(j)}\right)$$

Modelisation:

$$p^{ij} = \frac{p^{j|i} + p^{i|j}}{2N}$$

$$q^{ij} = \frac{\left(1 + \|y^{(i)} - y^{(j)}\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y^{(k)} - y^{(l)}\|^2\right)^{-1}}$$

Cost Function:

$$C = KL(P||Q)$$

Derivatives:

$$\frac{dC}{dy^{(i)}} = 4\sum_j \left(p^{ij} - q^{ij}\right)\left(y^{(i)} - y^{(j)}\right)\left(1 + \|y^{(i)} - y^{(j)}\|^2\right)^{-1}$$

→ Faster optimization!

→ Improved robustness (to noise/uncertainties)

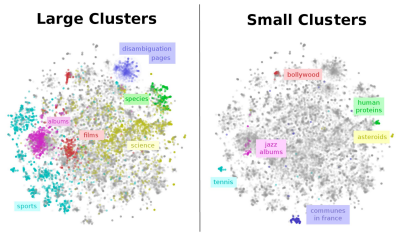
Embedding Methods Basics

Using t-SNE to explore word embeddings

Slide credit: Simon Carbonelle

Embedding Methods Basics

Using t-SNE to explore Wikipedia article embeddings

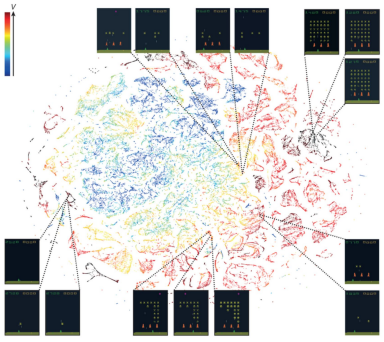


Slide credit: Simon Carbonelle

Embedding Methods Basics

Using t-SNE to explore game state representations

- DQN network that learns to play video games learns embeddings that group game states that look similar



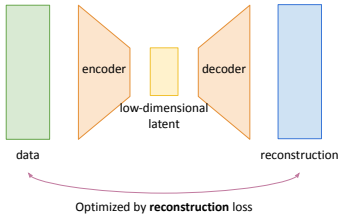
Human-level control through deep reinforcement learning, V. Mnih et Al. (Nature,2015)

Slide credit: Simon Carbonelle

Autoencoder

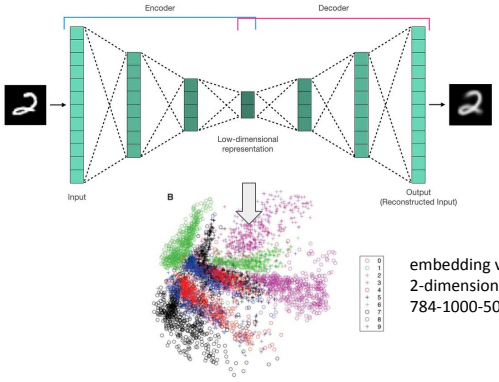
Autoencoders

- Compress the data and learn data reconstruction.
  - Reconstruction loss can take L2 loss (for real-valued inputs) or cross-entropy loss (for binary inputs)
- The learned features reduce dimension, the reconstruction will not be exactly the same as inputs.



Problem: Autoencoder may overfit if # parameters > # data

Learning Autoencoder with handwritten digits

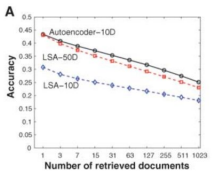


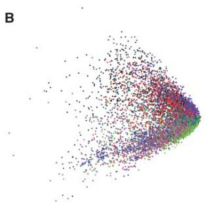
embedding visualization:
2-dimensional codes found by a 784-1000-500-250-2 autoencoder

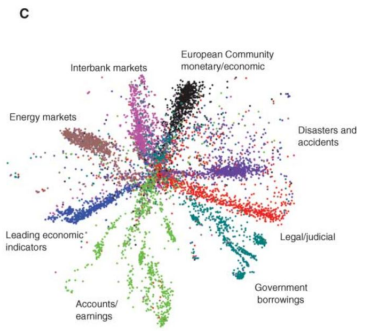
Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.

Learning Autoencoder with documents

Fig. 4. **(A)** The fraction of retrieved documents in the same class as the query when a query document from the test set is used to retrieve other test set documents, averaged over all 402,207 possible queries. **(B)** The codes produced by two-dimensional LSA. **(C)** The codes produced by a 2000-500-250-125-2 autoencoder.



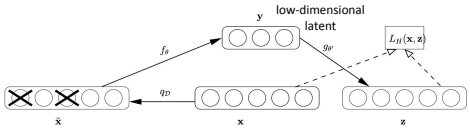




Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.

Denoising autoencoders (DAEs)

- Problem: Autoencoders can memorize training data too closely, reducing their effectiveness on unseen data.
- Denoising: reconstruct from **corrupted**, partially destroyed data
  - Binary masking noise: masked a part of the component to 0
  - Additive isotropic Gaussian noise: $\widetilde{x} = x + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I)$



- DAE can be used as building blocks for deep neural networks
  - greedy layer-wise pre-training, followed by fine-tuning with task objective

Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *Proceedings of the 25th international conference on Machine learning*. 2008.