Unstructured Data Analysis

Lecture 5: Manifold learning

George Chen
Isomap

- Build k-NN graph, computed shortest distances
- Compute Euclidean distances between all pairs of low-dimensional points
- Make these two as close as possible (Euclidean dist)

Original high-dim. data → Distance table (for high-dim. points)

Low-dim. data → Distance table (for low-dim. points)
Isomap Calculation Example

Demo
3D Swiss Roll Example

Key idea: true distance on manifold is the blue line

We’re approximating the blue line with the red line (poor choice of # nearest neighbors can make approximation bad)

Some Observations on Isomap

The quality of the result critically depends on the nearest neighbor graph.

- Emphasize local structure:
  - Ask for nearest neighbors to be really close by
  - There might not be enough edges

- Emphasize global structure:
  - Allow for nearest neighbors to be farther away
  - Might connect points that shouldn’t be connected

In general: try different parameters for nearest neighbor graph construction when using Isomap + visualize.
t-SNE
(t-distributed stochastic neighbor embedding)
t-SNE High-Level Idea #1

- Don't use deterministic definition of which points are neighbors
- Use probabilistic notation instead

A and B are "similar"
A and C are "similar"
A and D are "similar"
D and E are "similar"
t-SNE High-Level Idea #2

- In low-dim. space (e.g., 1D), suppose we just randomly assigned coordinates as a candidate for a low-dimensional representation for A, B, C, D, E (I'll denote them with primes):

  \[
  \begin{align*}
  &A', B', C', D', E' \\
  \end{align*}
  \]

- With any such candidate choice, we can define a probability distribution for these low-dimensional points being similar:

\[
\begin{array}{c}
A', B' \text{ similar} \\
A', C' \text{ similar} \\
A', D' \text{ similar} \\
\vdots \\
D', E' \text{ similar}
\end{array}
\]
t-SNE High-Level Idea #3

- Keep improving low-dimensional representation to make the following two distributions look as closely alike as possible.

This distribution stays fixed

This distribution changes as we move around low-dim. points
t-SNE

Original high-dim. data → Probability table (for high-dim. points)

Low-dim. data → Probability table (for low-dim. points)

Technical detail: creates probabilities based on Gaussian distribution

Make these two as close as possible (Technical detail: KL divergence)

Technical detail: creates probabilities based on Student’s t-distribution

Technical details are in separate slides (posted on webpage)
t-SNE

Roughly: **perplexity** is like a continuous version of “number of nearest neighbors”

Emphasize local structure

Emphasize global structure

Low **perplexity** value  

High **perplexity** value

Also: play with learning rate, # iterations

In practice, often people initialize with PCA
Manifold Learning with t-SNE

Demo
t-SNE Interpretation

https://distill.pub/2016/misread-tsne/
Dimensionality Reduction for Visualization

- There are many methods (I've posted a link on the course webpage to a scikit-learn example using ~10 methods)

- PCA is very well-understood; the new axes can be interpreted

- Nonlinear dimensionality reduction: new axes may not really be all that interpretable (you can scale axes, shift all points, etc)

- PCA and t-SNE are good candidates for methods to try first

- If you have good reason to believe that only certain features matter, of course you could restrict your analysis to those!
Let’s look at images
(Flashback) Recap: Basic Text Analysis

- Represent text in terms of “features” (such as how often each word/phrase appears)
- Can repeat this for different documents: *represent each document as a “feature vector”*

"Sentence": ☀️ 🌧️ ⛈️ 🌡️ ☂️ ☂️ ☂️ ☂️

In general (not just text): first represent data as feature vectors

This is a point in 4-dimensional space, $\mathbb{R}^4$
Example: Representing an Image

Go row by row and look at pixel values

Image source: starwars.com
Example: Representing an Image

Go row by row and look at pixel values

Image source: starwars.com
Example: Representing an Image

Go row by row and look at pixel values

Image source: starwars.com
Example: Representing an Image

Go row by row and look at pixel values

# dimensions = image width × image height

Very high dimensional!

Image source: starwars.com
Dimensionality Reduction for Images

Demo