ML Fundamentals

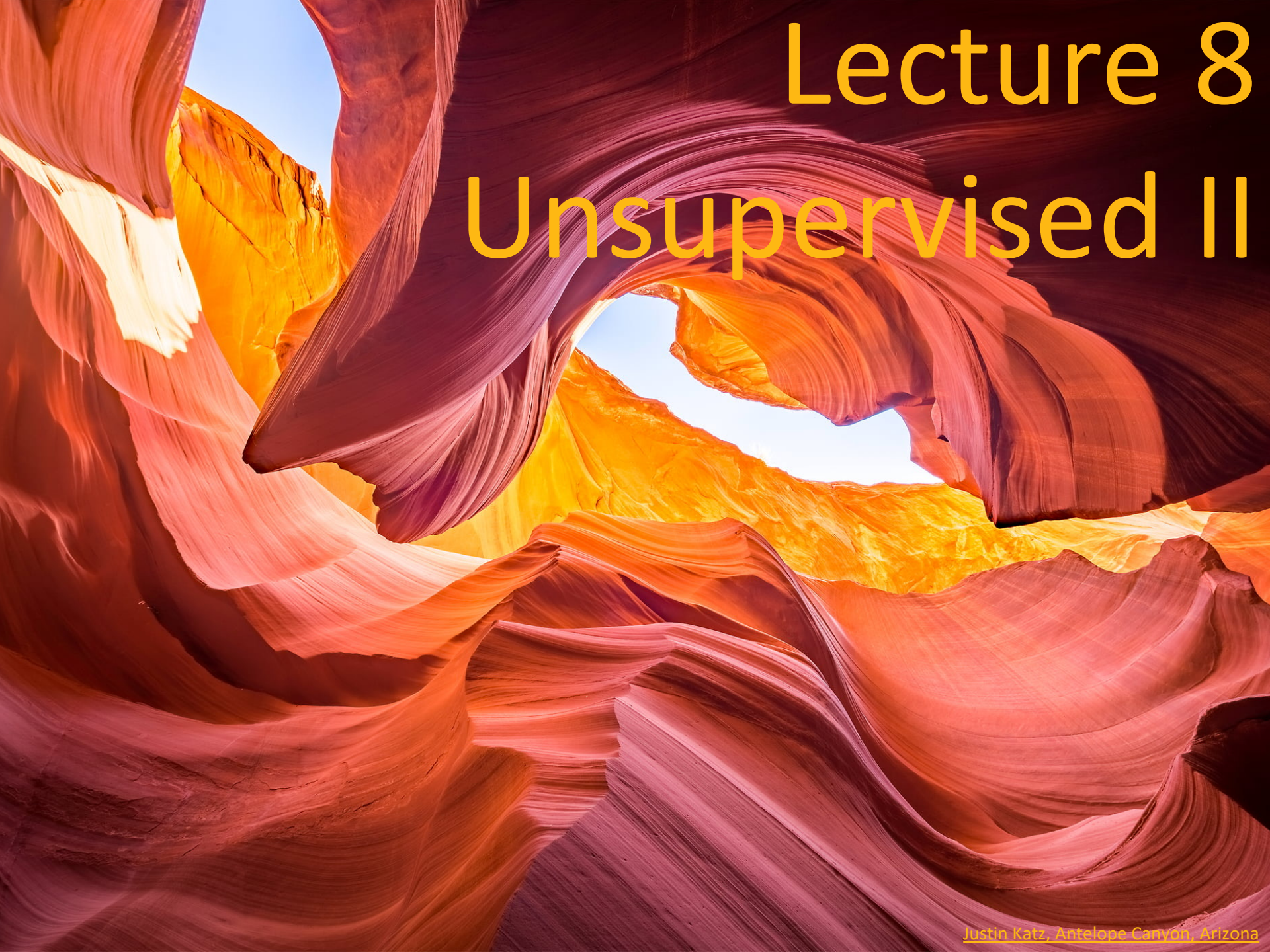
Instituto Balseiro

Instituto Balseiro
13/09/2022

Announcements

- Se subió la guía 3 de aprendizaje no-supervisado (opcional).
- Se movió el deadline del 1er informe a mañana 14, a las 23:59. Debe presentarse en el Google Classroom.
  - Si hay alguien con dificultades para acceder, avise. También escriban a estudiantes@ib.edu.ar.
- El lunes que viene tenemos clase para compensar la que no tuvimos el 02/09.
  - Lunes 19/09 de 14:00 a 18:00, acá.
- Tema dosel
- Tema pickle

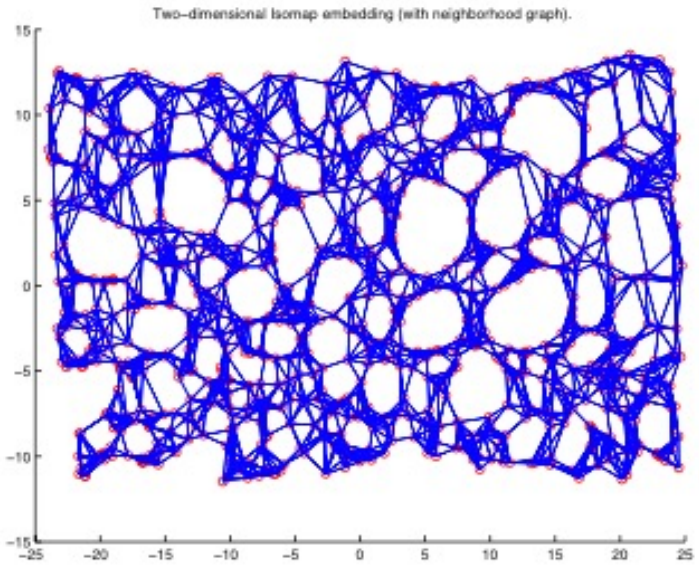
Lecture 8
Unsupervised II

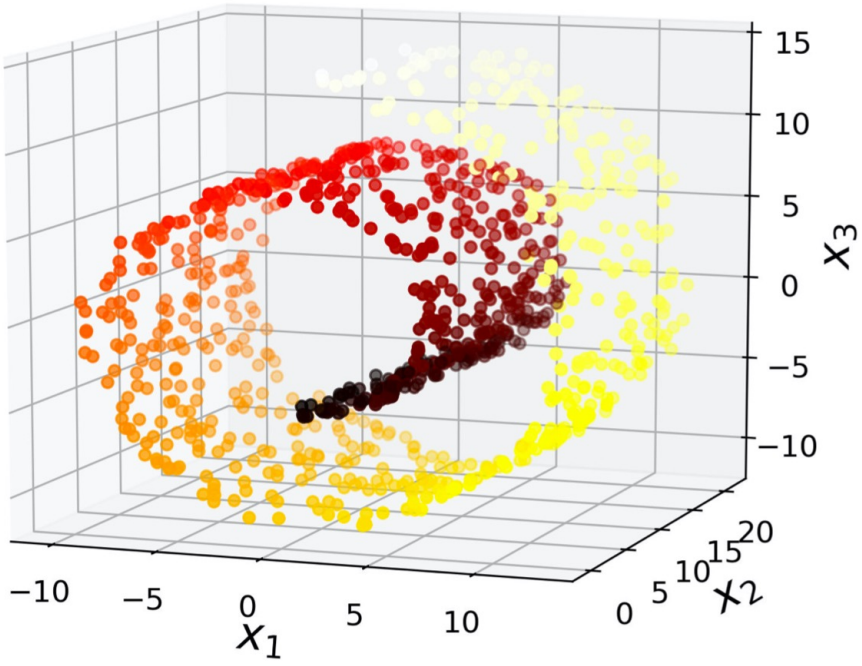


Justin Katz, Antelope Canyon, Arizona

ML Fundamentals – Lecture 8

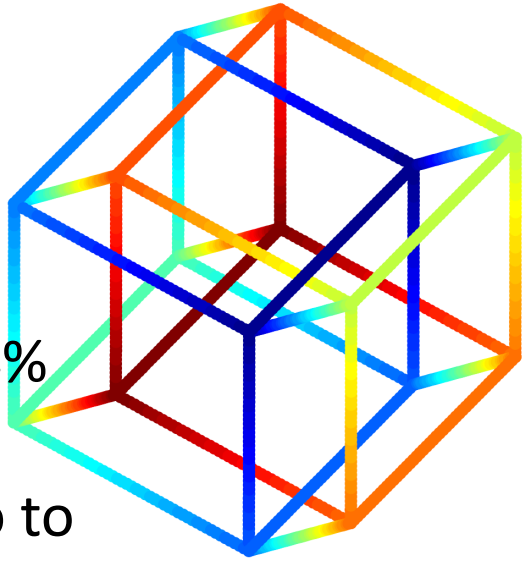
- Curse of dimensionality
- Representation learning
- Manifold learning
  - Isomap
  - LLE
- Visualization
  - t-SNE
  - UMAP




15
10
5
0
−5
−10
x_3
−10
−5
0
5
10
x_1
0
5
10
15
20
x_2

The curse of dimensionality

(Ballman, 1961)



- 2D-3D intuition **fails** at higher dimensions
- E.g. One random point in 1x1 square less than 4% chance of closer than 10^{-3} from border, i.e. an extreme value in a given dimension. Chances go to 99.999999% for d=10^4.
- Two random points in a unit square: average distance of .52. In a unit cube, 0.66. In a 10^6 unit hypercube, distance is 408.25.
- Think of a sphere and a cube containing it, such that L = 2R. It follows that:

$$V_{\text{sphere}}(r) = \frac{\pi^{D/2} r^D}{\Gamma(1 + D/2)} \quad , \quad \lim_{D \to \infty} \frac{V_{\text{sphere}}(r)}{V_{\text{cube}}(r)} = 0$$

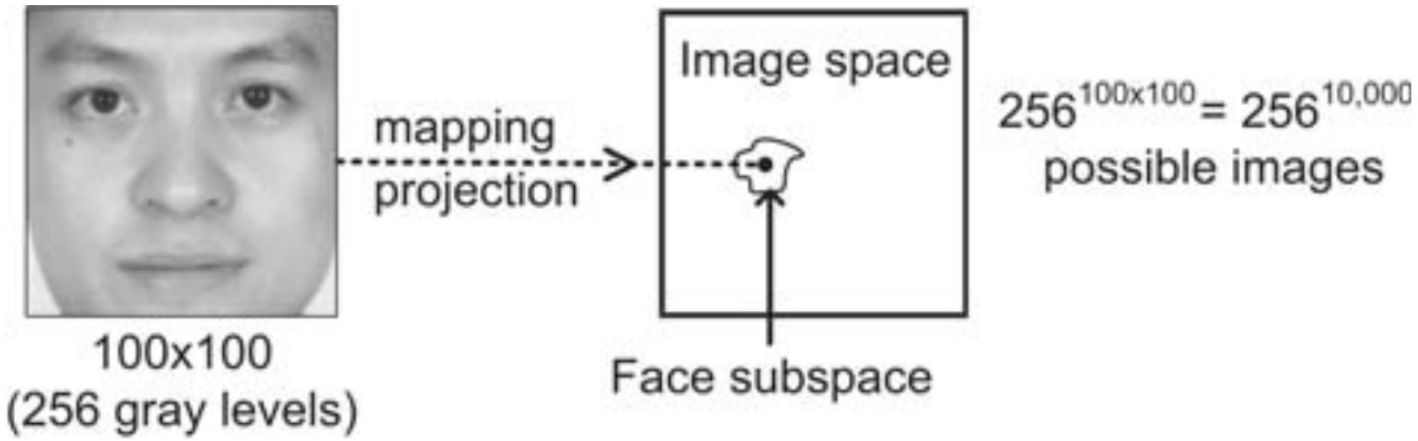
$$V_{\text{cube}}(r) = (2r)^D \quad ,$$

and $\lim_{D \to \infty} V_{\text{sphere}}(r) = 0$.

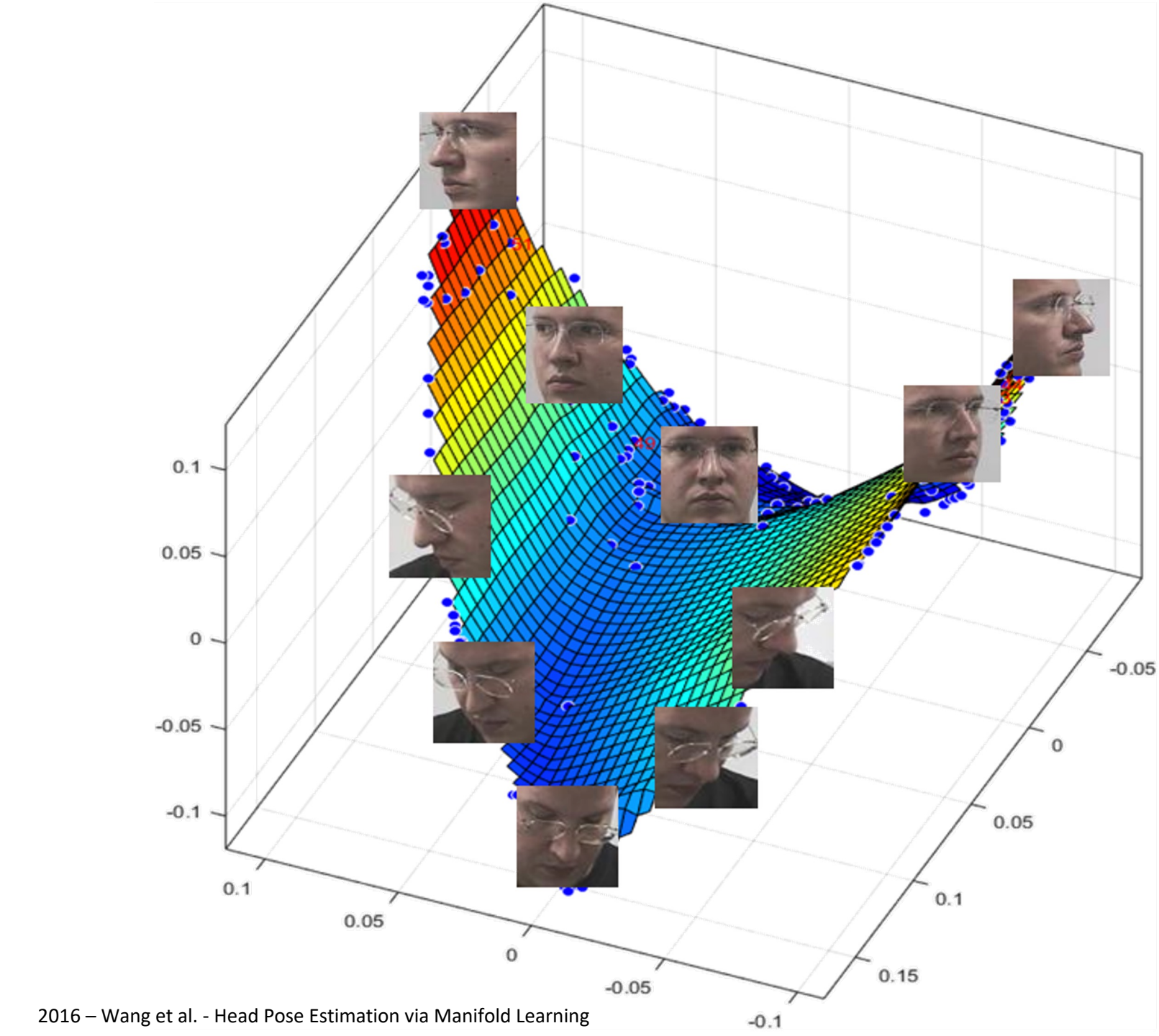
when r = 1/2

The curse of dimensionality

- High-dimensional spaces are inherently sparse



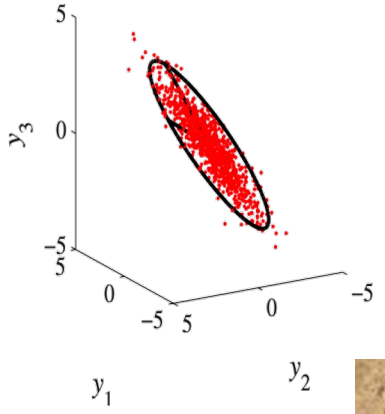
- Adding additional signal features that are truly associated with the response will improve the fitted model
- Adding noise features that are not truly associated with the response will lead to a deterioration in the fitted model, exacerbating the risk of overfitting

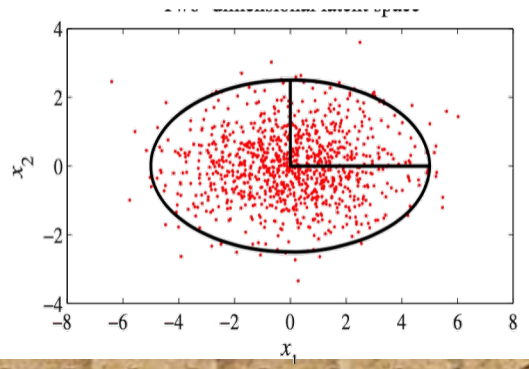


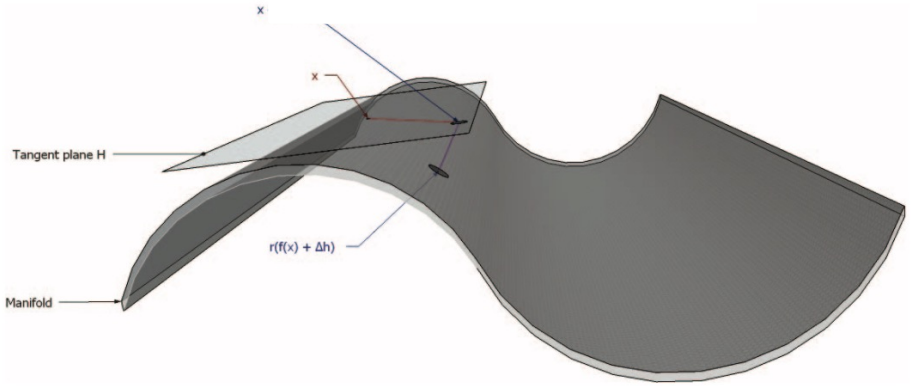
2016 – Wang et al. - Head Pose Estimation via Manifold Learning

Representations: Why?

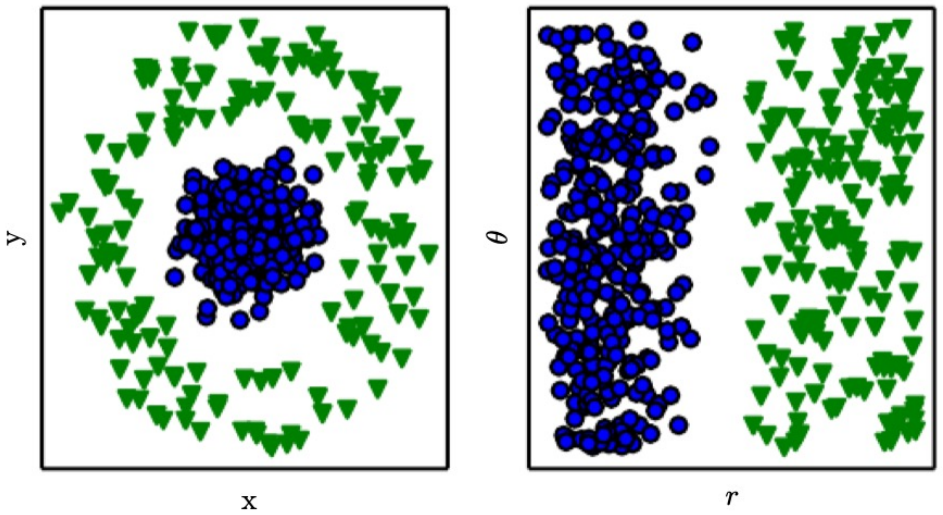
The concept of representation learning ties together all of the many forms of deep learning.
"Deep Learning", Goodfellow, Bengio, Courville (2016)

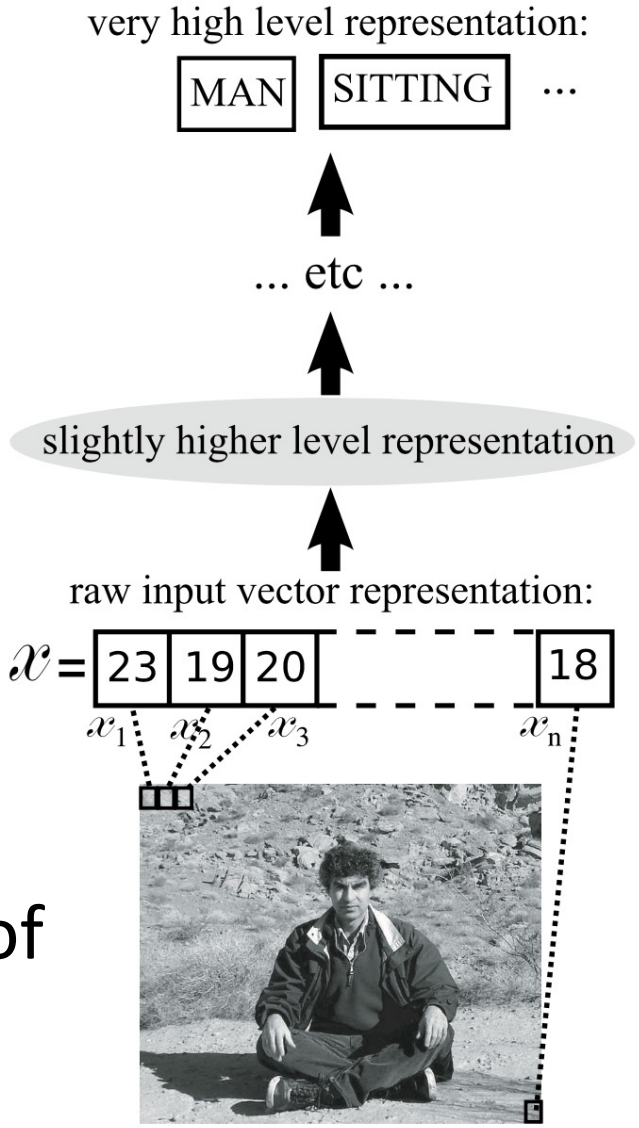






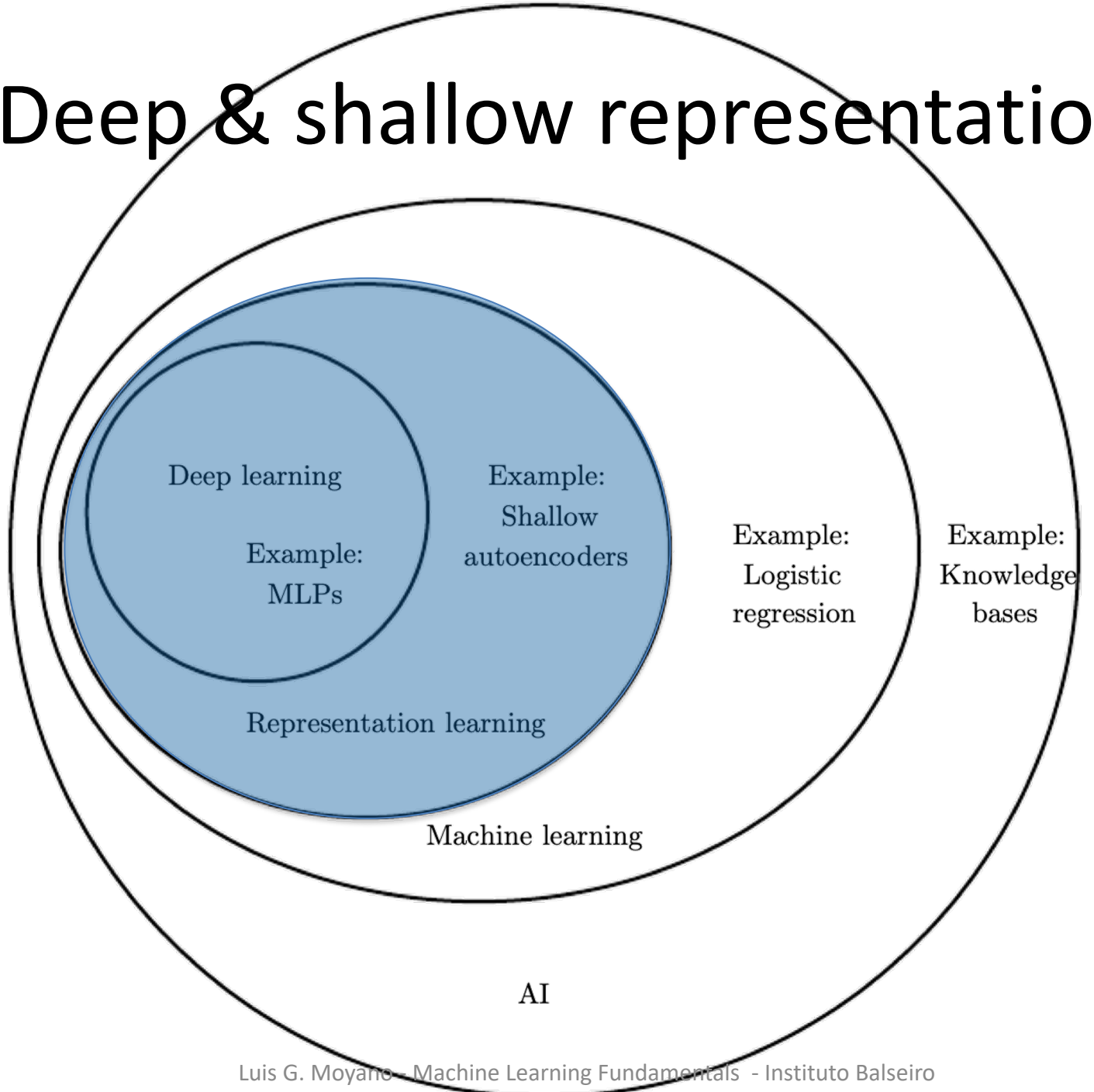
What is a representation?



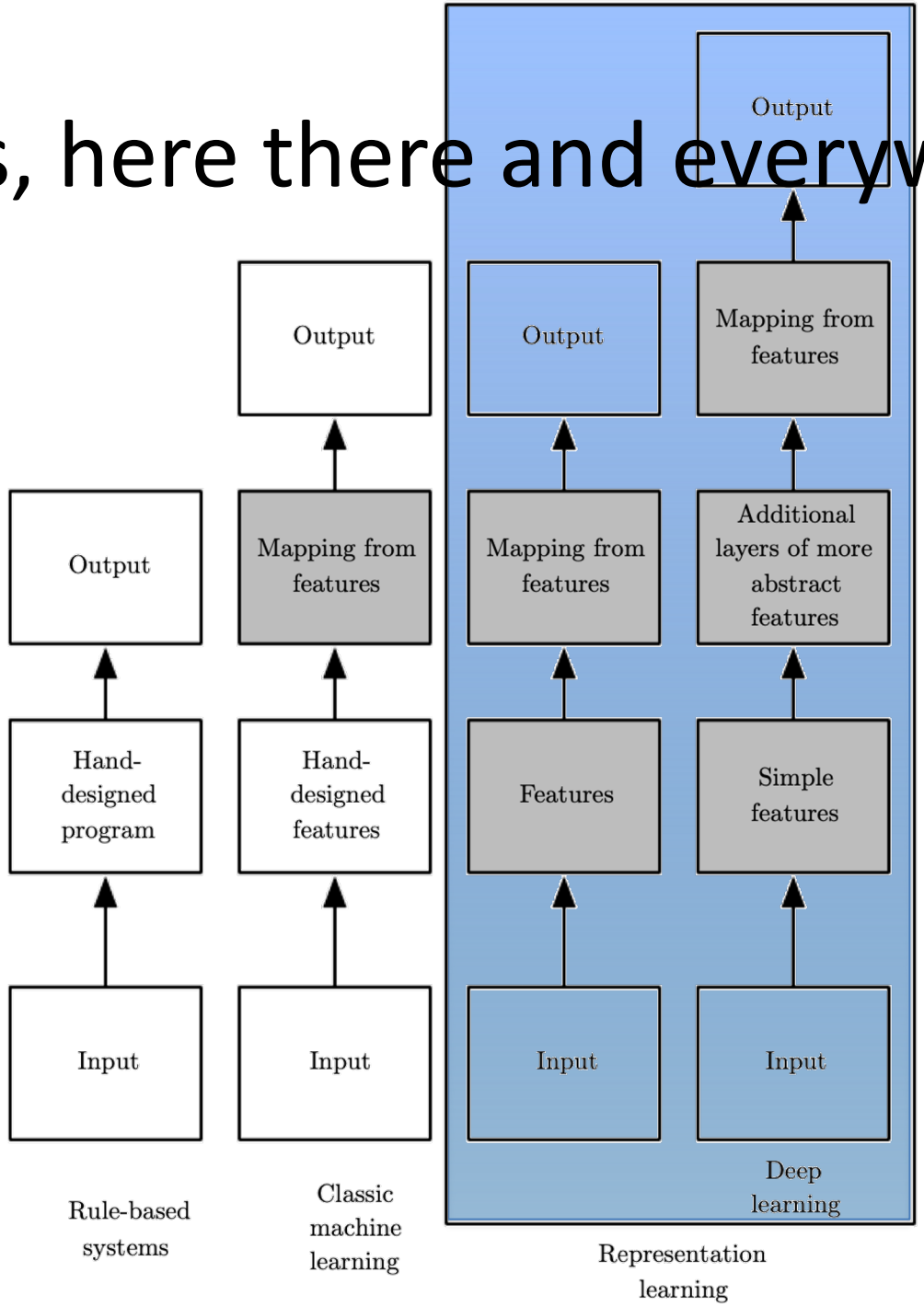


- **Shallow learning**: one level of representation
- **Deep learning**: multiple levels of increasing complexity & abstraction

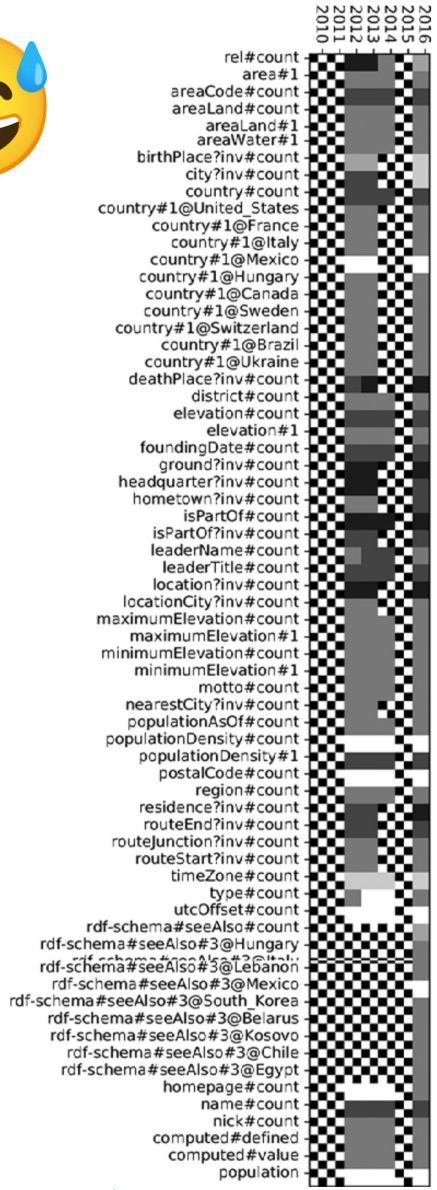
Deep & shallow representations

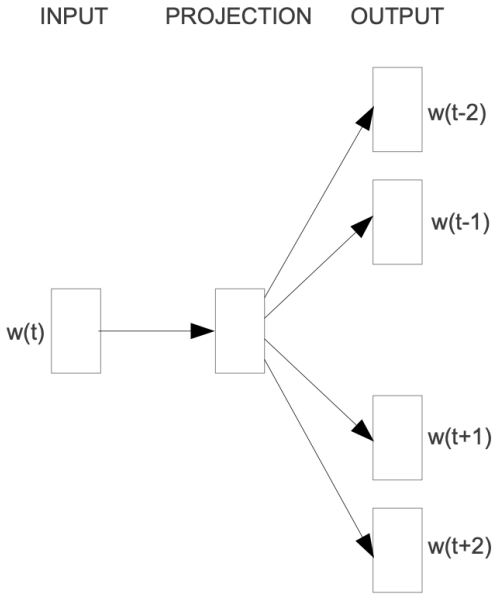


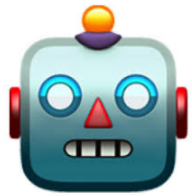
Features, here there and everywhere



Feature engineering vs. representation learning

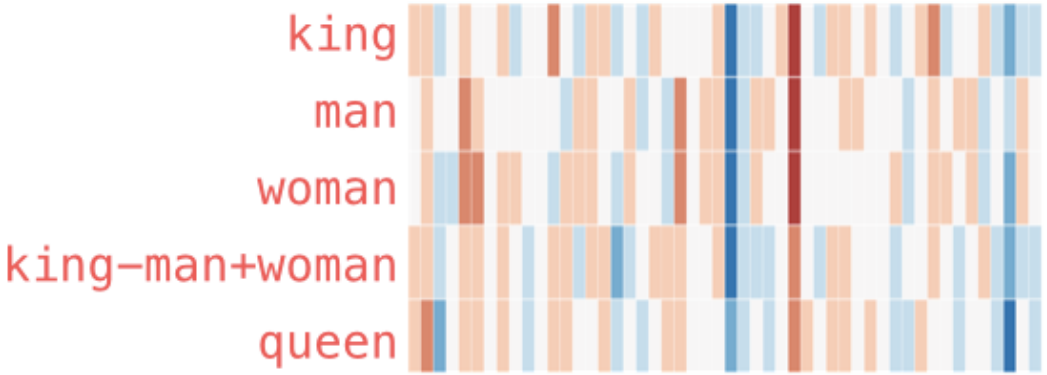






2013 - Mikolov - Efficient estimation of word representations in vector space

king − man + woman ~= queen

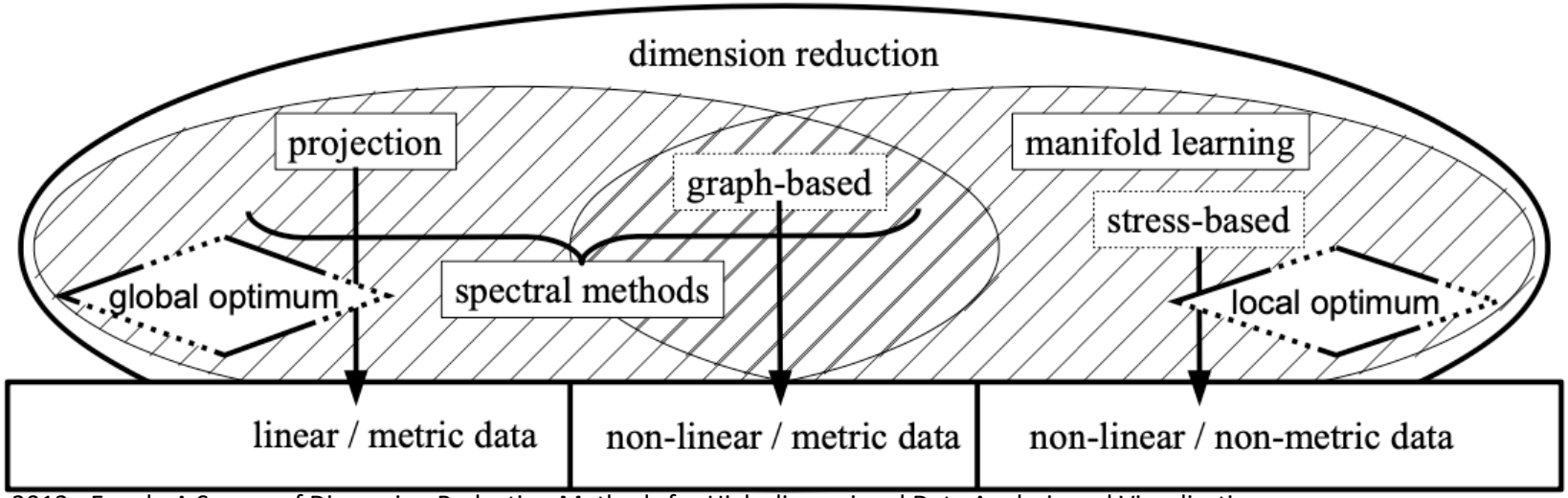


Alammar, Jay (2018). [The Illustrated Transformer]

2020 - Duboue - The Art of Feature Engineering= Essentials for Machine Learning

Dimensional reduction - Aims

- Capture essential structure in data - Inference
- Exploratory analysis – Visualization
- Preprocess data for supervised learning – Capturing latent variables
  - explanatory quantities not accessible directly
  - computing an embedding, a map between objects and vectors
- Noise reduction and compression – Redundancy reduction

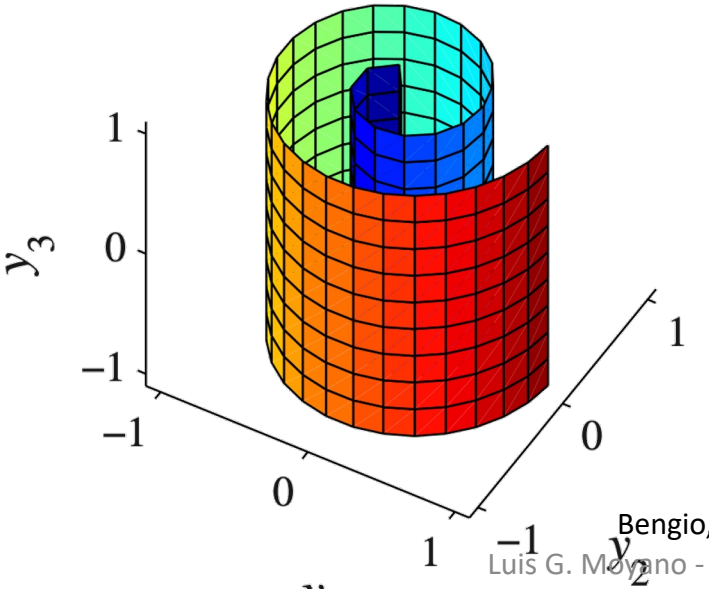


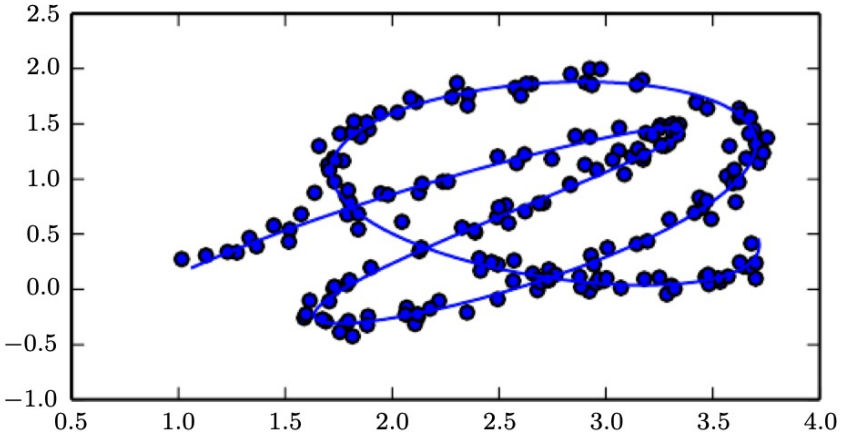
2012 - Engel - A Survey of Dimension Reduction Methods for High-dimensional Data Analysis and Visualization

Manifold Learning

Non-linear dimensionality reduction

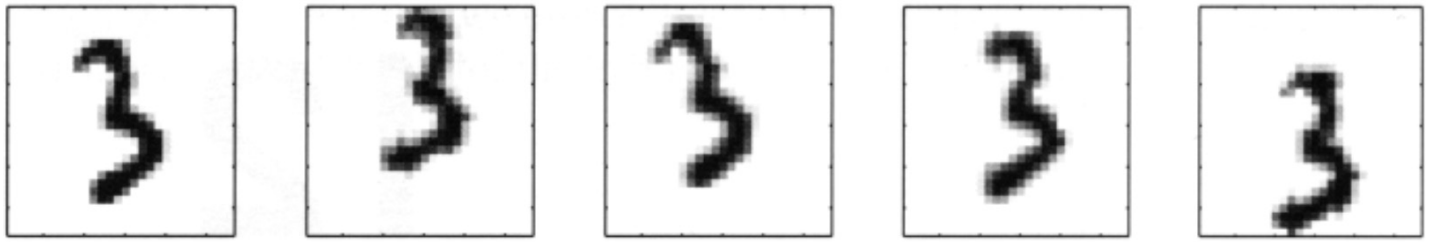
- Manifolds are topological objects which are locally flat, and (preferably) differentiable
- ***Manifold hypothesis:*** real-world data in high dimensional spaces may be expected to concentrate probability mass in the vicinity of a manifold *M* of much lower dimensionality d(M), embedded in the high dimensional input space $R^{d(x)}$.



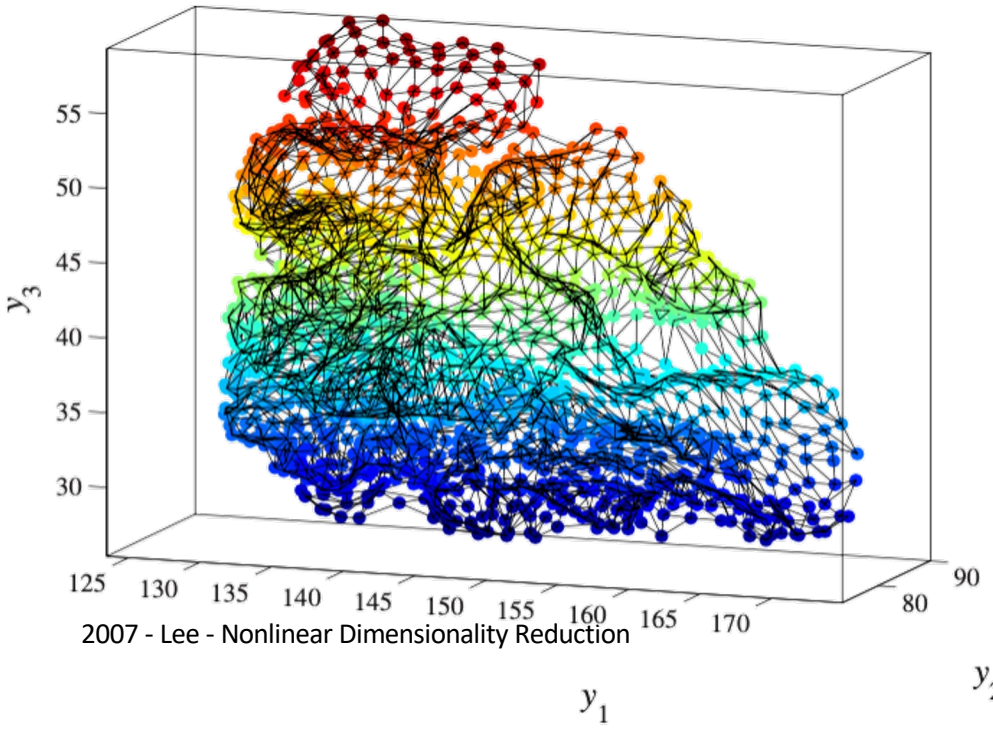


Bengio, Courville and Vincent, Representation Learning=A Review and New Perspectives (2013)

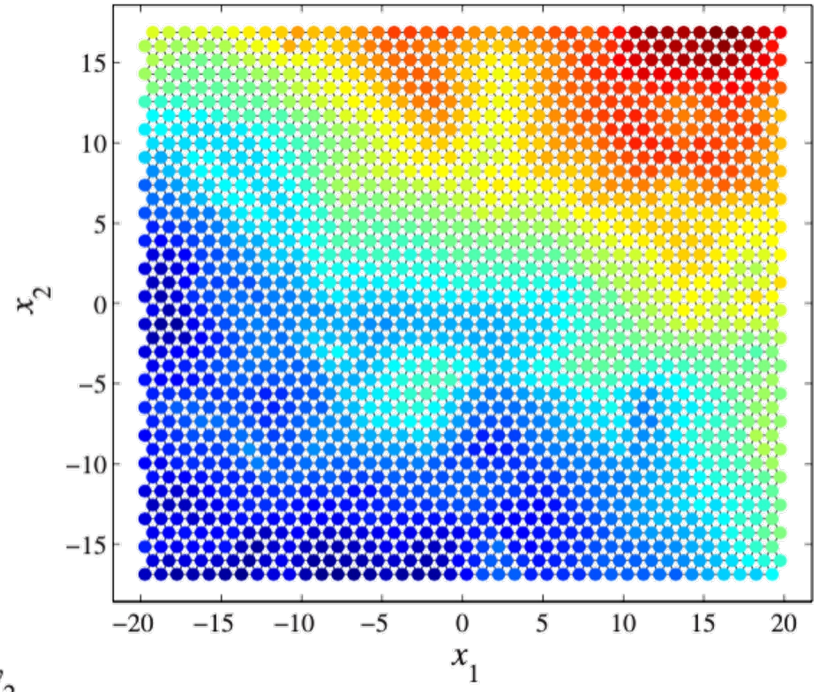
Manifold Learning



2006 - Bishop - Pattern Recognition And Machine Learning



2007 - Lee - Nonlinear Dimensionality Reduction

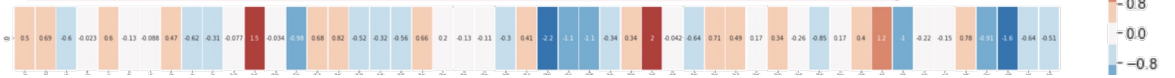


Embeddings: distributed representations

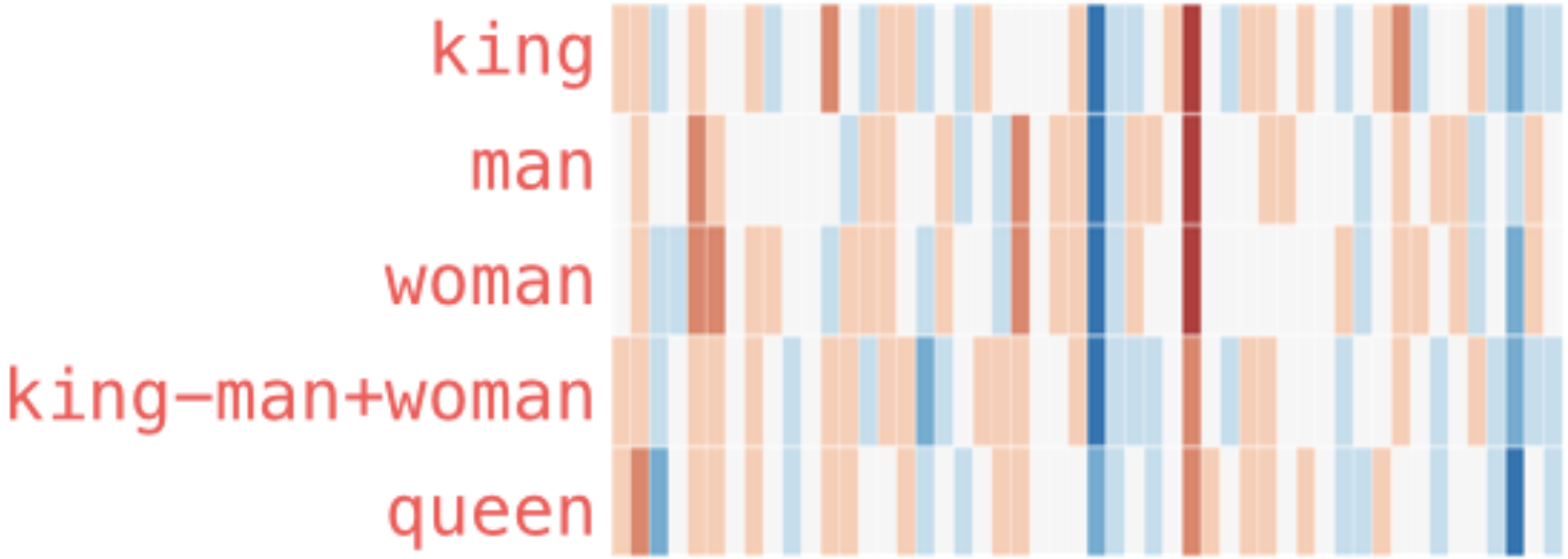
Alammar, Jay (2018). The Illustrated word2vec

"king"

[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 ,
-0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961
, -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 ,
-0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 ,
-1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]



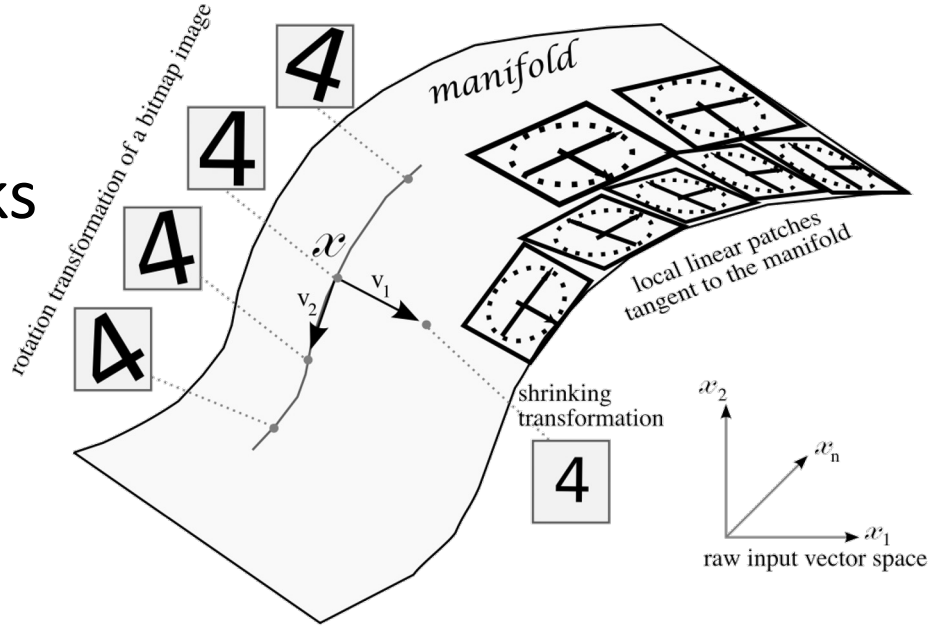
king − man + woman ~= queen



WHAT MAKES A REPRESENTATION GOOD?

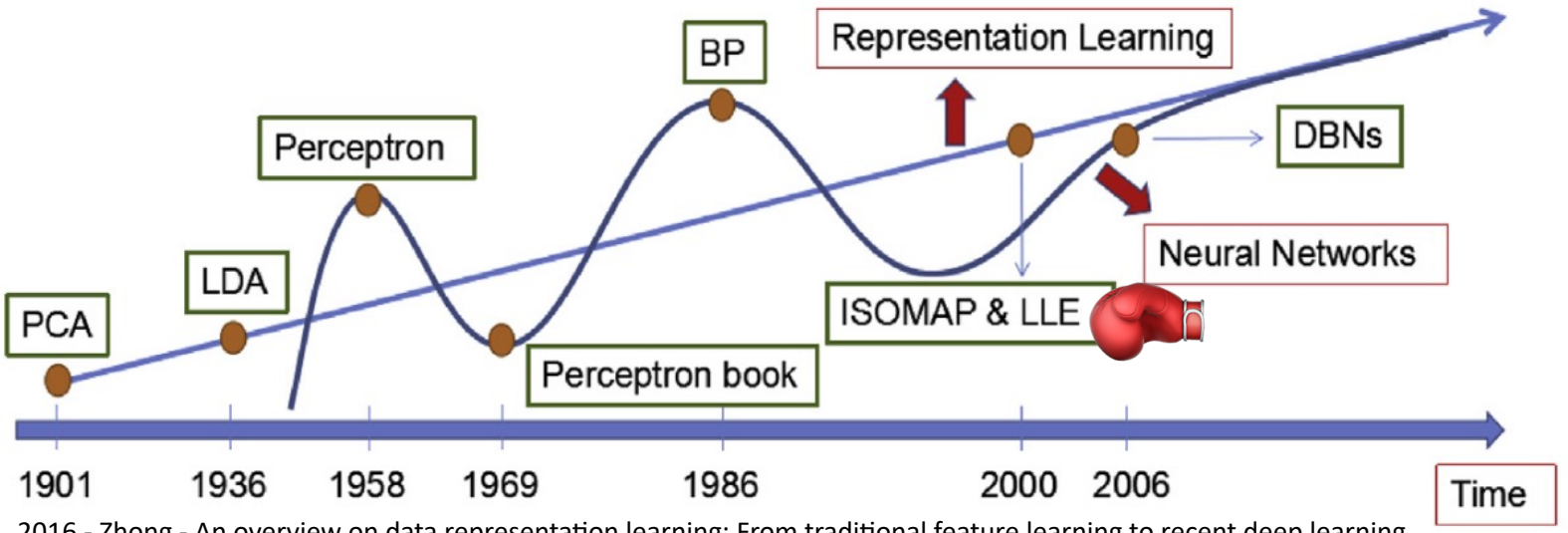
Desirable characteristics for representations, across ML tasks

- Smoothness
- Multiple explanatory factors
- A hierarchical organization of explanatory factors
- Semi-supervised learning
- Shared factors across tasks
- Manifolds
- Sparsity



2013 - Bengio - Representation learning A review and new perspectives
2016 - Goodfellow - Deep learning

Representation learning timeline



2016 - Zhong - An overview on data representation learning: From traditional feature learning to recent deep learning

- Isometric feature mapping – ISOMAP (2000)
- Locally linear embedding – LLE (2000)
- Independent Component Analysis – ICA (2001)
- Laplacian eigenmaps – LE (2002)
- Hinton's (2006)
  - "DL kickoff"paper

Multidimensional Scaling revisited

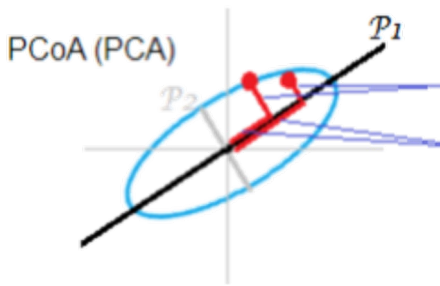
- Family of dimensionality reduction algorithms based on spectral structure
- Representation of data in low dimensionality aiming at preserving pairwise dissimilarities (distance):

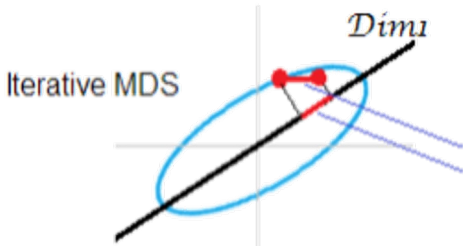
$$d(\mathbf{y}(i), \mathbf{y}(j)) = \sqrt{\langle (\mathbf{y}(i) - \mathbf{y}(j)) \cdot (\mathbf{y}(i) - \mathbf{y}(j)) \rangle}$$

$$D := \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,M} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,M} \\ \vdots & \vdots & & \vdots \\ d_{M,1} & d_{M,2} & \cdots & d_{M,M} \end{pmatrix}.$$

Find vectors such that $\|x_i - x_j\| \approx d_{i,j}$

- Norm may be Euclidian (PCA) or other (metric/classic MDS)
- Distance -> dissimilarities (inner product) -> SDU eigens
- Target dimensions chosen beforehand
- sklearn.manifold.MDS
- much slower than PCA
- As PCA, can't capture local nonlinear structure



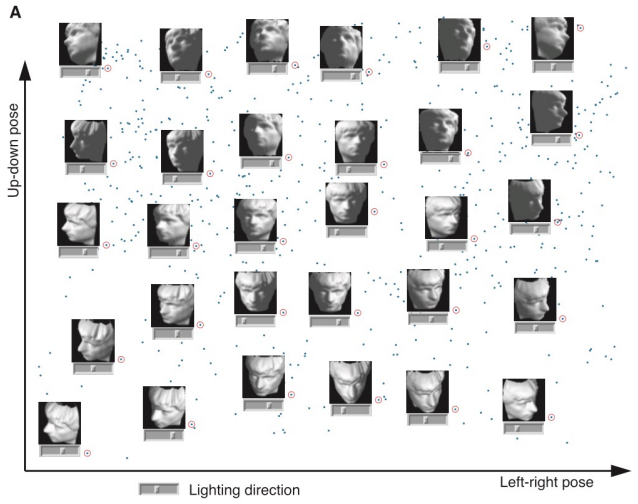


https://scikit-learn.org/stable/auto_examples/manifold/plot_manifold_sphere.html

ISOMAP - Isometric feature mapping

Tenembaum et al. (2000)

- Try to preserve local Euclidian distances in the embedding
- ISOMAP Recipe:
  - Build graph from k Nearest Neighbors.
  - Run MDS

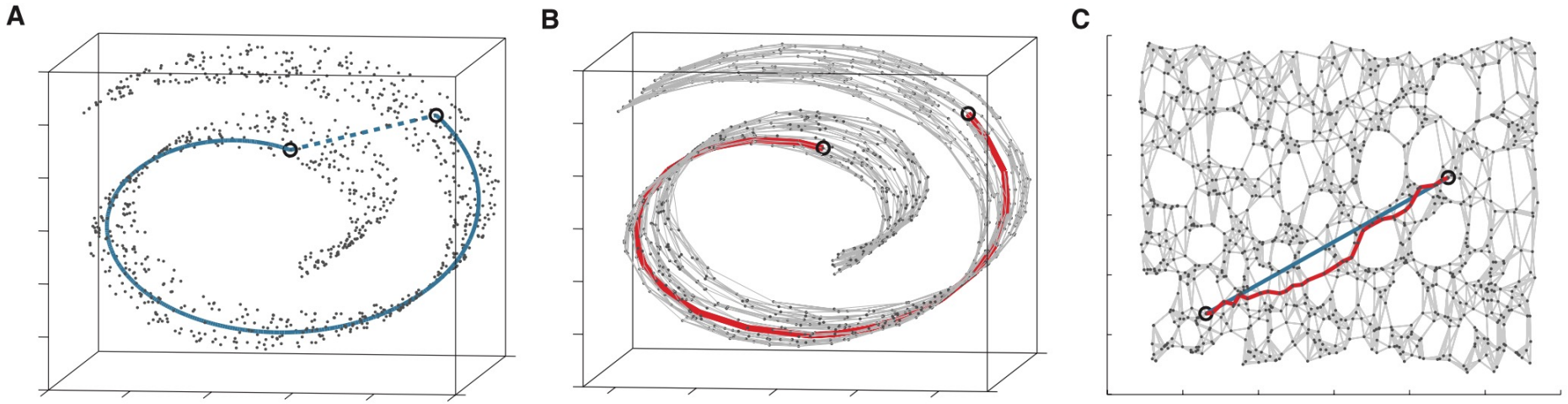


- Since MDS is slow, ISOMAP is also very slow.
- Need estimate of k...
- Assumes data set is convex (no holes).

ISOMAP - Isometric feature mapping

Tenembaum et al. (2000)

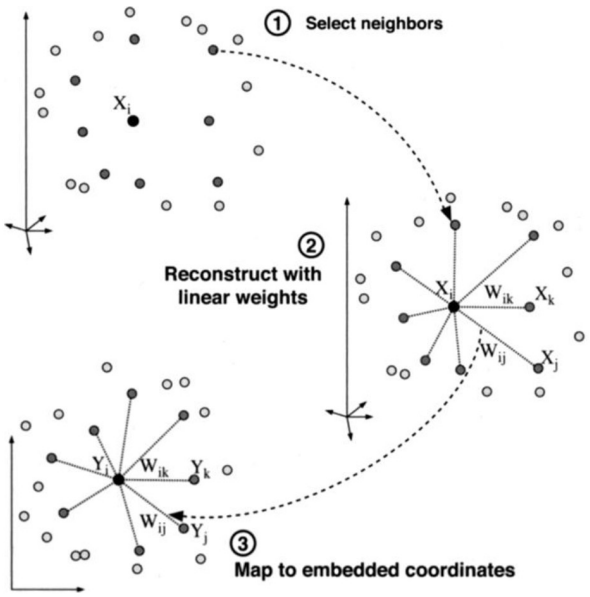
- Build a **neighborhood graph** (via K or ε)
  - Compute *weights* through Euclidian distance
- Graph distance as a proxy for geodesical distance
  - shortest path via **Dijkstra's algorithm**
- Spectral decomposition of pairwise distance matrix. Keep better components.
  - The graph distance metric makes it non-linear
- Good approximation provided enough & dense points

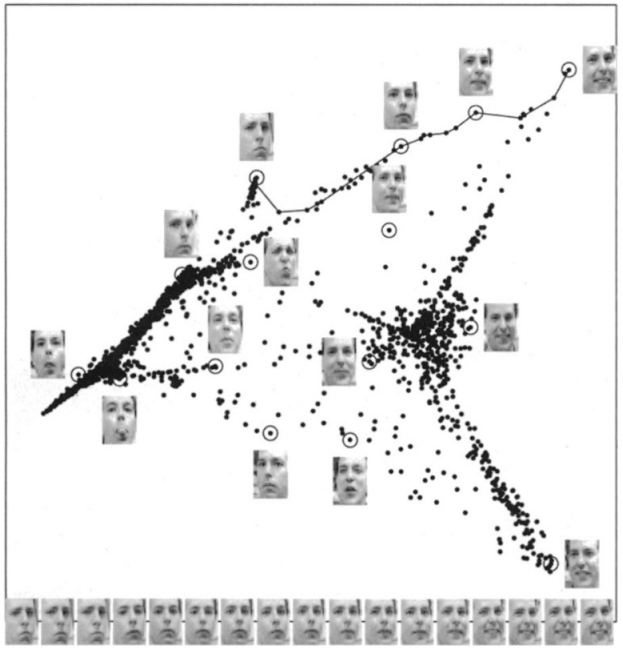


LLE - Locally linear embedding

Roweis & Saul (2000)

- LLE recovers global nonlinear structure from locally linear fits.
- Replace each point y(i) with a linear combination of its neighbors. $x_i = \sum_{x_j \in N_i} W_{i,j} x_j$
- Preserved topology i.e. transformations that preserves local angles (invariance to rotations, translations, rescaling).
-





Reconstruction error

$$\Phi(\hat{\mathbf{X}}) = \sum_{i=1}^{N} \left\| \hat{\mathbf{x}}(i) - \sum_{j \in \mathcal{N}(i)} w_{i,j} \hat{\mathbf{x}}(j) \right\|^2$$

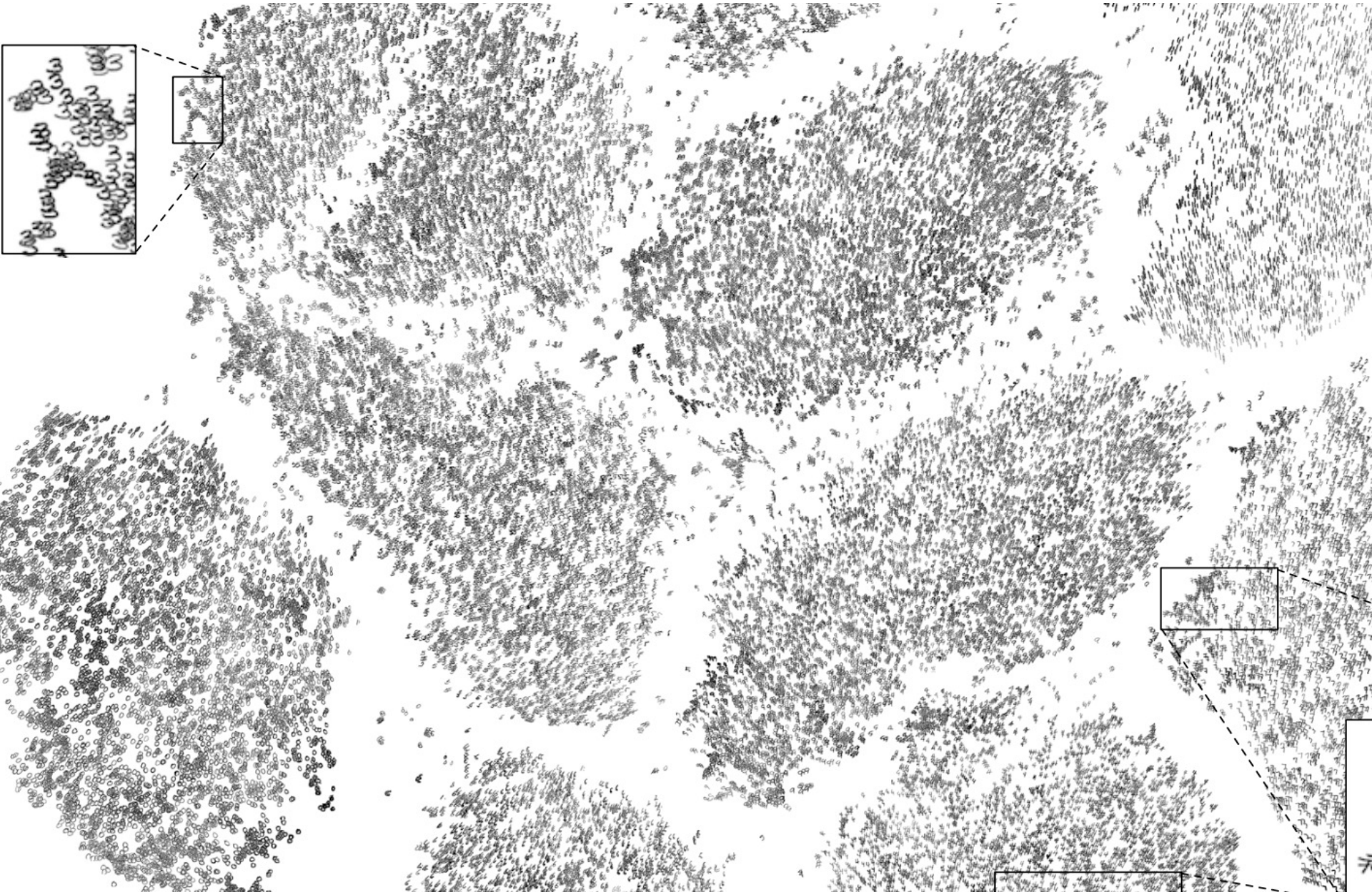
Embedding cost function

$$\mathcal{E}(\mathbf{W}) = \sum_{i=1}^{N} \left\| \mathbf{y}(i) - \sum_{j \in \mathcal{N}(i)} w_{i,j} \mathbf{y}(j) \right\|^2$$

Distance vs. Neighborhoods

- These methods show disadvantages in higher dimensions
- Distances in high-dimensions tend to concentrate, and tend to be almost identical ("The Curse")
- Raw distances don't imply neighborhood structure (i.e., points that are neighbors of each other in high dimensions are not always neighbors in low dimensions).
- Modern methods aim to preserve graph structure (neighbors).


@txolerfx



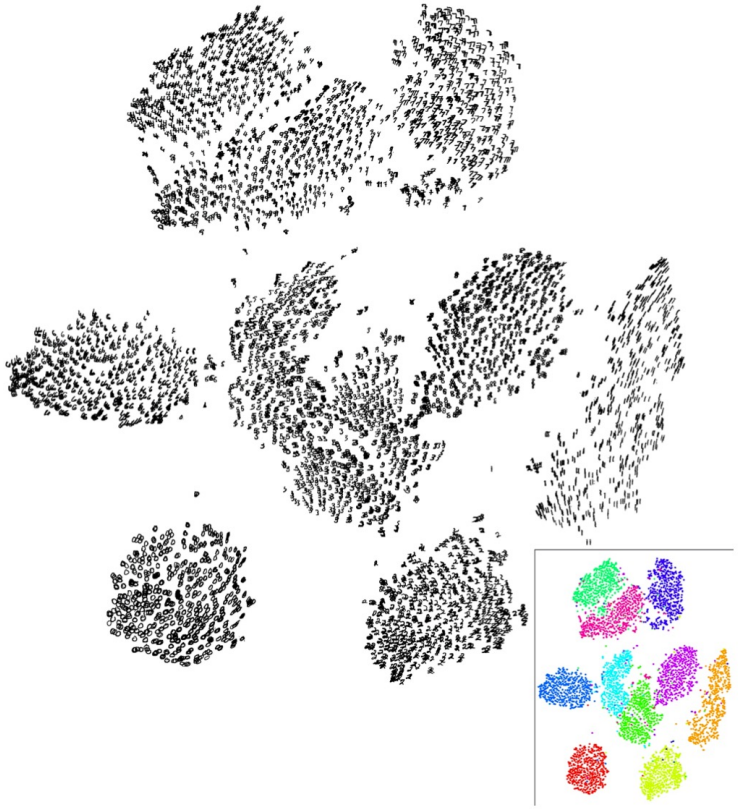
t-SNE (van der Maaten, 2014)

- t-Distributed Stochastic Neighbor Embedding
- Visualization of high-dimensional datasets - o(10^7) and o(10^{2-3}) dimensions.
- *t-SNE solves the Crowding problem*. The extent to which this problem occurs depends on the ratio between the intrinsic data dimensionality and the embedding dimensionality.
- *Accelerating t-SNE using Tree-Based Algorithms (*van der Maaten, 2014*).*

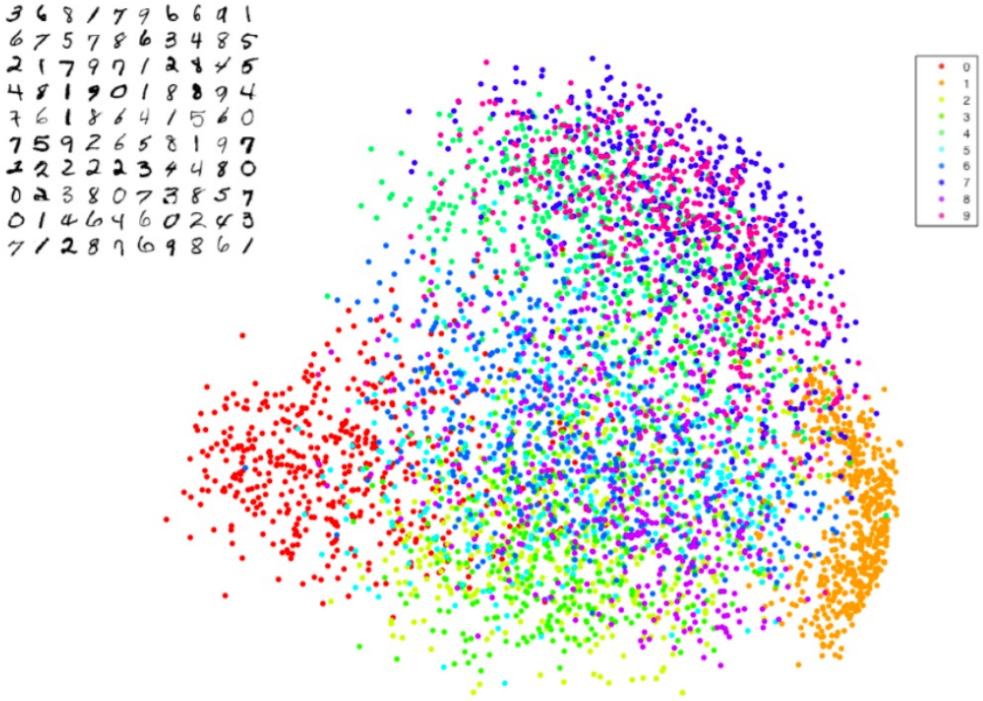
sklearn.manifold.TSNE

https://lvdmaaten.github.io/tsne/

Comparison on the MNIST dataset



t-SNE



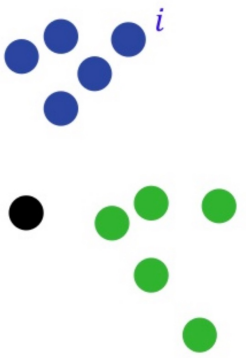
PCA

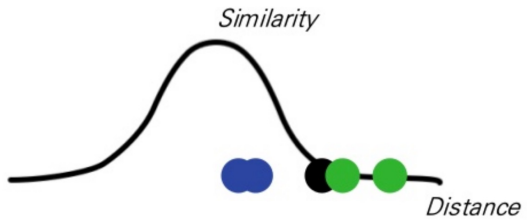
SNE (Hinton and Roweis, 2003)

- **SNE basic idea**: ”Encode” high dimensional neighborhood information as a distribution.
- Intuition: Random walk between data points.
  - High probability to jump to a close point.
- Find low dimensional points such that their neighborhood distribution is similar.
- How do you measure distance between distributions?
  - Most common measure: **KL divergence**

Modeling Neighborhood Distributions

- Consider the neighborhood around an input data point $x_i \in R^d$ Imagine that we have a Gaussian distribution centered around x_i.
- Then the probability that x_i chooses some other datapoint x_j as its neighbor is proportional to the density under this Gaussian.
- A point closer to x_i will be more likely than one further away.



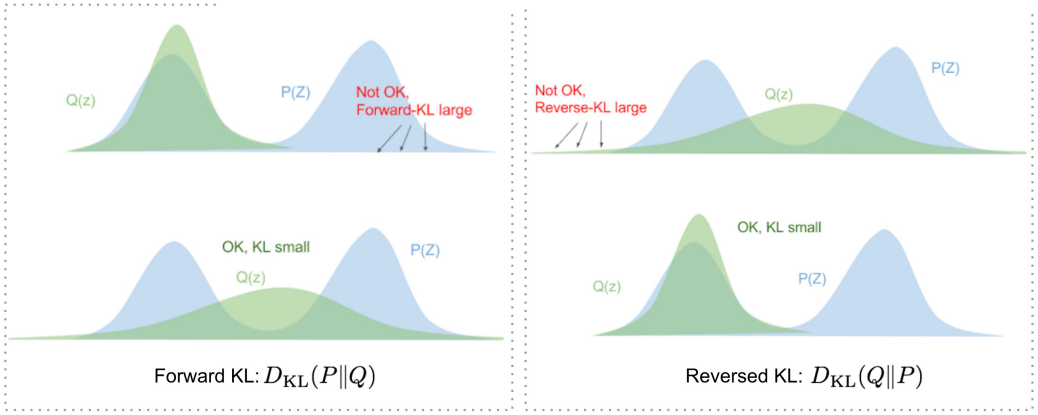


$$p_{j|i} = \frac{e^{-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}}}$$

SNE

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)}, \qquad q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k\neq i}\exp\left(-\|y_i - y_k\|^2\right)}.$$

Minimize Kullback-Leibler divergence $C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}},$



The gradient is: $$\frac{\delta C}{\delta y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j).$$

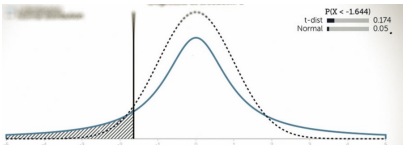
t-SNE attacks "the crowding problem"

- In high dimension we have more room, so points can have a lot of different neighbors
  - In 2D a point can have a few neighbors at distance 1 all far from each other. What happens when we embed in 1D?
- This is the ”crowding problem” – not enough room to accommodate all neighbors.
- This is one of the biggest problems with SNE.
- t-SNE solution: Change the Gaussian in Q to a heavy tailed distribution: the t-Student

t-SNE

Good ol' Student

$$p(x) \propto \left(1 + \frac{x^2}{v}\right)^{-(v+1)/2}$$



William Sealy Gosset

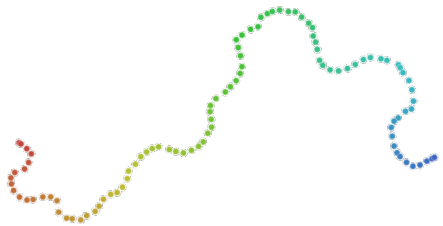
- Replace Gaussian with **t-Student**
  - to avoid 'crowding problem' (loosely, clusters tendency to overlap in lower d, give more probability to points farther apart).
- Exponentially decreasing tails vs. 'heavy' power-law tails

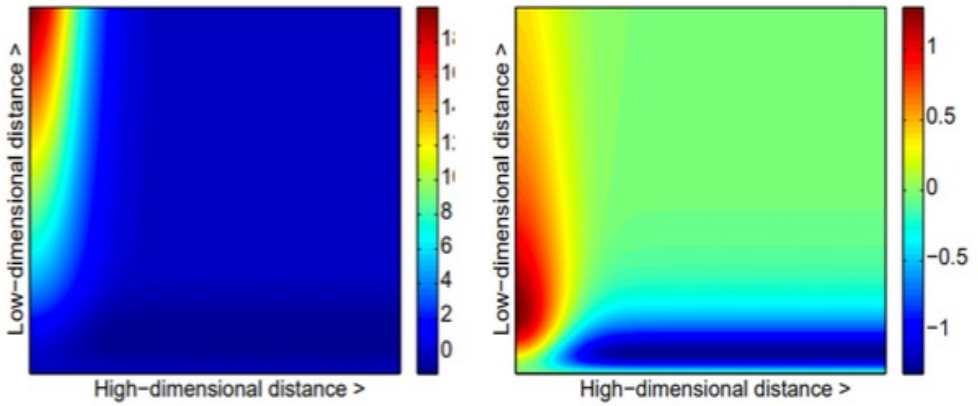
$$Q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}}$$

$$\frac{\partial L}{\partial \mathbf{y}^{(i)}} = \sum_j (P_{ij} - Q_{ij})(\mathbf{y}^{(i)} - \mathbf{y}^{(j)})(1 + ||y_i - y_j||^2)^{-1}$$

SNE

- Optimization with graphs & random walks, for speed

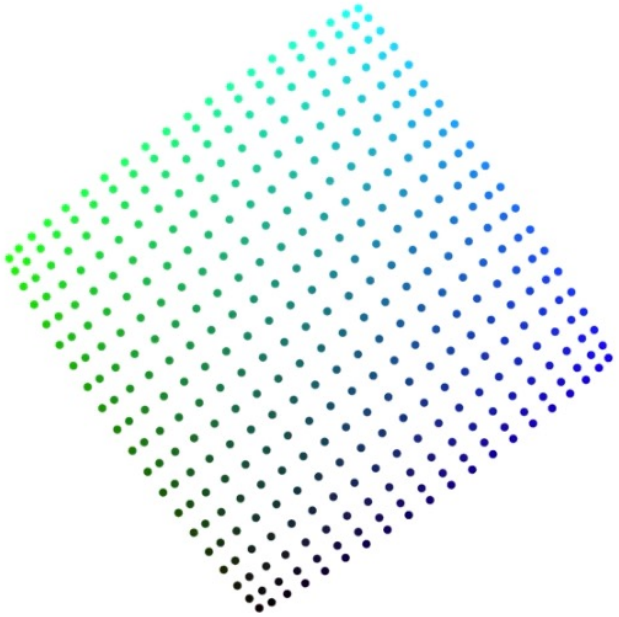




(a) Gradient of SNE. (c) Gradient of t-SNE.

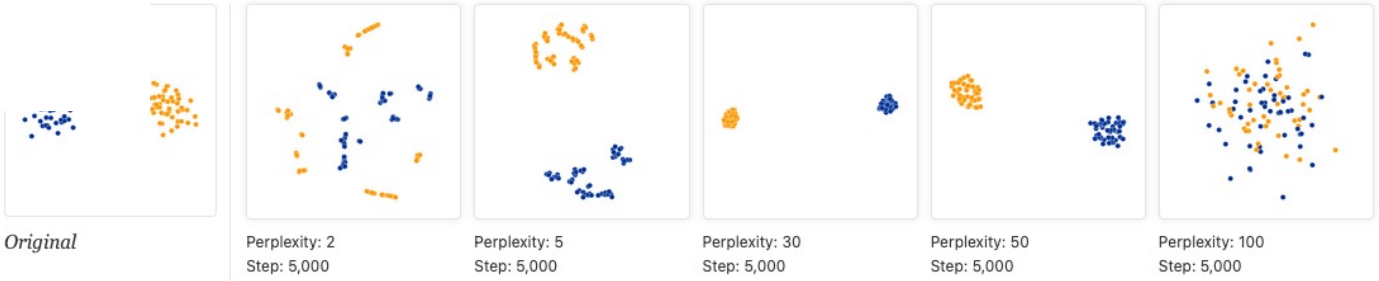
How to Use t-SNE Effectively

Although extremely useful for visualizing high-dimensional data, t-SNE plots can sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.



perplexity parameter: balance between local and global. Usually in [5, 50]. Should be lower than number of points.
epsilon parameter: learning rate. Should be let to converge.

- Cluster sizes and distances mean nothing.
- Random noise doesn't always look random.
- t-SNE tends to expand denser regions of data (lookout for ghost patterns)
- For topology, you may need more than one plot



How to Use t-SNE Effectively

2017 – Cao - Automatic Selection of t-SNE Perplexity

UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction

- Uniform Manifold Approximation and Projection (McInnes & Healy, 2018).
- Dimension reduction technique
  - Visualization of high-dimensional datasets
  - Also as a NLDR method, for preprocess downstream tasks (clustering, classification, etc.)
- sklearn-compatible 👍

```
conda install -c conda-forge umap-learn
```

```
pip install umap-learn
```

https://github.com/lmcinnes/umap

UMAP assumptions

1. The data is uniformly distributed on a Riemannian manifold;
2. The Riemannian metric is locally constant (or can be approximated as such);
3. The manifold is locally connected.

UMAP constructs a high dimensional graph representation of the data then optimizes a low-dimensional graph to be as structurally similar as possible

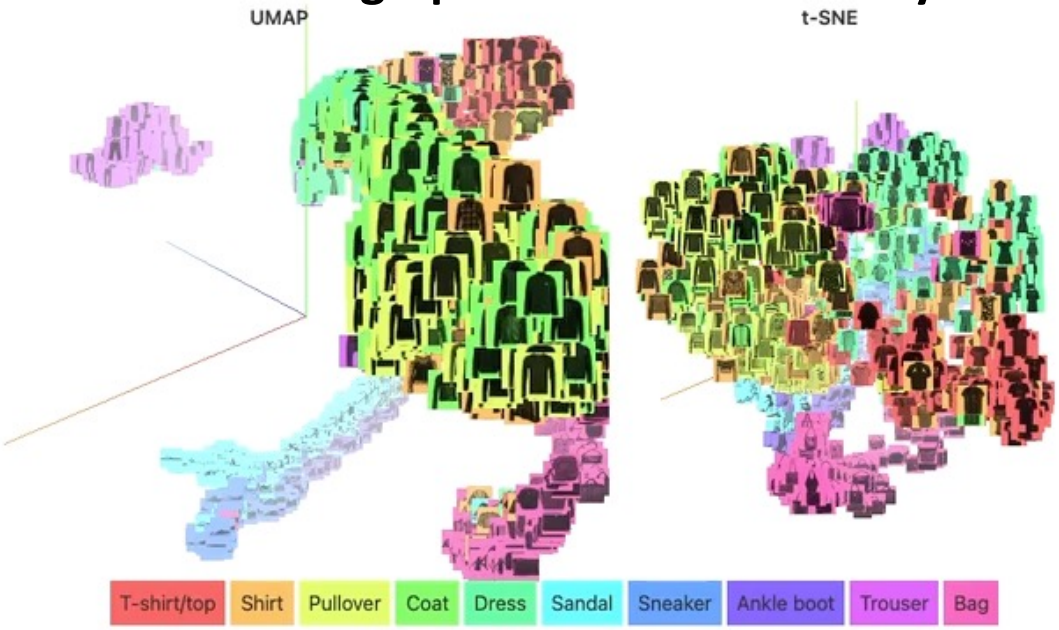


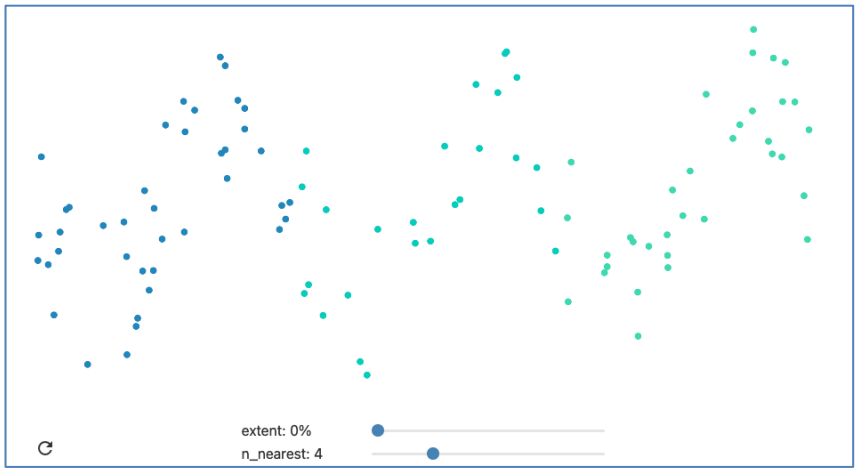
Figure 2: Dimensionality reduction applied to the Fashion MNIST dataset. 28x28 images of clothing items in 10 categories are encoded as 784-dimensional vectors and then projected to 3 using UMAP and t-SNE.

Fashion MNIST
subset:
28x28=784 dimensions
projected into 3
(4' vs 27')

Balance between
Local vs global
structure

Understanding UMAP
Andy Coenen, Adam Pearce | *Google PAIR*

How UMAP works: By "fuzzy simplicial complex"



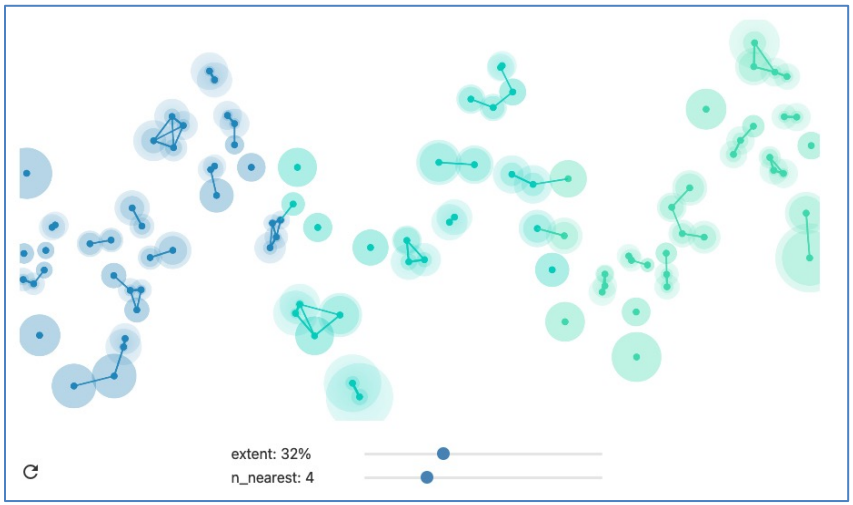
Recipe:

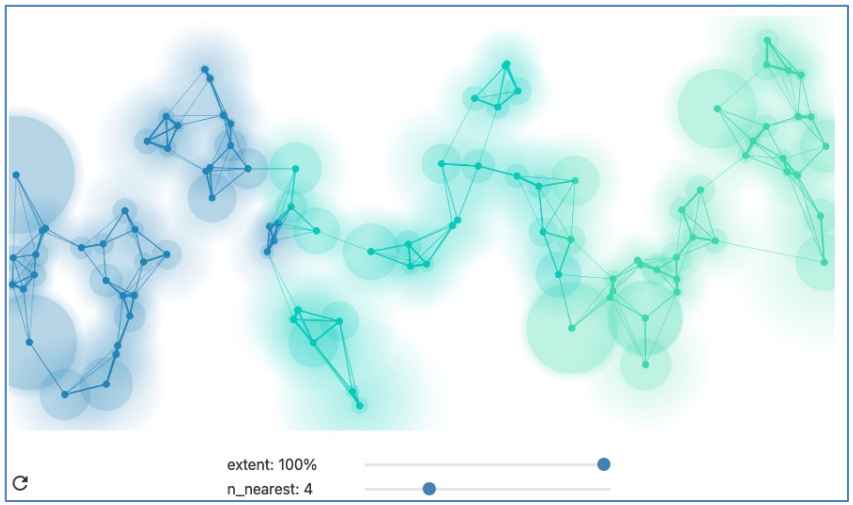
1. Build a high dimensional weighted graph
   (weights measure distance betw points)
2. Optimize graph projection in low dimension
   (via force-directed algorithm)

Main parameters:

- `n_neighbors`
- `min_dist`
- `n_components`
- `metric`

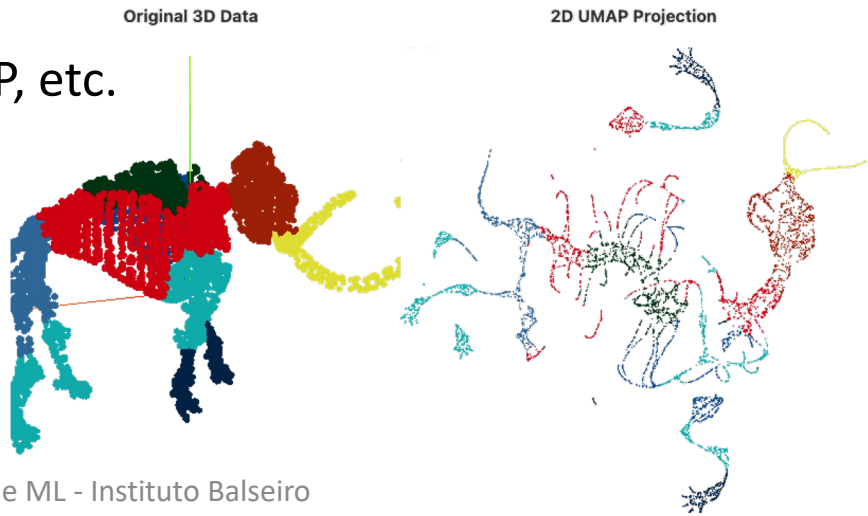
https://umap-learn.readthedocs.io/en/latest/parameters.html





Benefits of UMAP

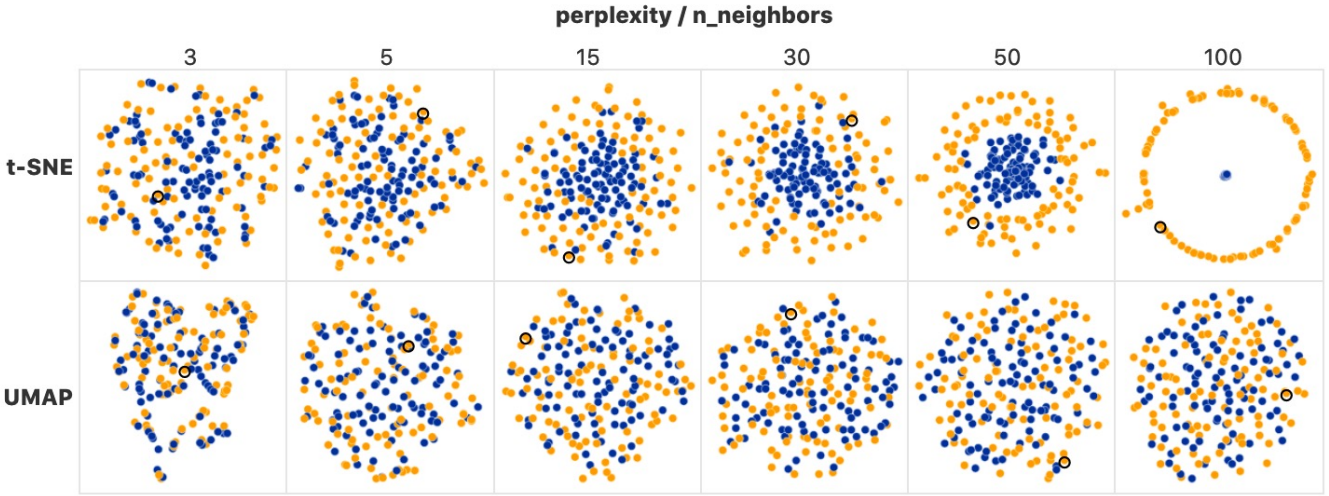
1. Fast!
2. Scales well in embedding dimension, good for upstream tasks (e.g. clustering methods such as HDBSCAN, a fancy improvement to DBScan)
3. Better "big picture" view while preserving local neighbours
4. Provides several (possibly non-metric) distances (e.g. for words)
5. Adding new points to an existing embedding (t-SNE can't)
6. Supports both supervised and semi-supervised dimension reduction
7. Possibility of using non-euclidian spaces (e.g. hyperbolic)
8. Strong mathematical foundations
9. Extensions: densMAP, parametric UMAP, etc.



UMAP vs t-SNE

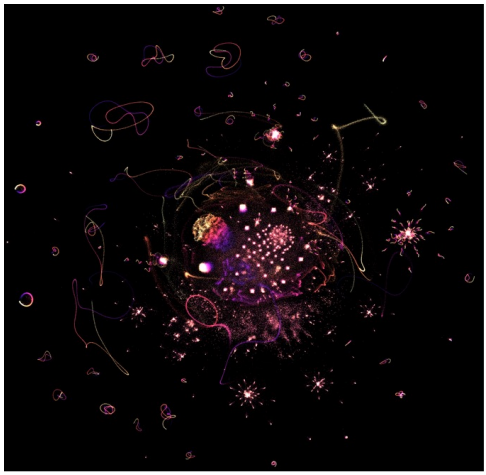
From https://github.com/lmcinnes/umap:

- UMAP is often better at preserving global structure in the final projection ("big picture" + neighborhoods).
- UMAP is faster than t-SNE*: 3' vs 45'* to project MNIST.
- UMAP scales well in embedding dimensions (not just for 2d).
  - UMAP supports supervised and semi-supervised dimension reduction (with care!)
- More understandable parameters than t-SNE
- Extensions: densMAP, parametric UMAP, etc.
- But, not always superior! E.g.:



From https://pair-code.github.io/understanding-umap/

How to (mis)read UMAP



One million integers embedded into 2D space with UMAP

1. Hyperparameters really matter!
   - fast means more trials
2. Cluster sizes in a UMAP plot mean nothing
   - local distance used to build the graph
3. Distances between clusters might not mean anything
   - idem previous
4. Random noise doesn't always look random.
   - specially with low values of number of neighbours
5. You may need more than one plot
   - Due to stochasticity and hyperparameter search

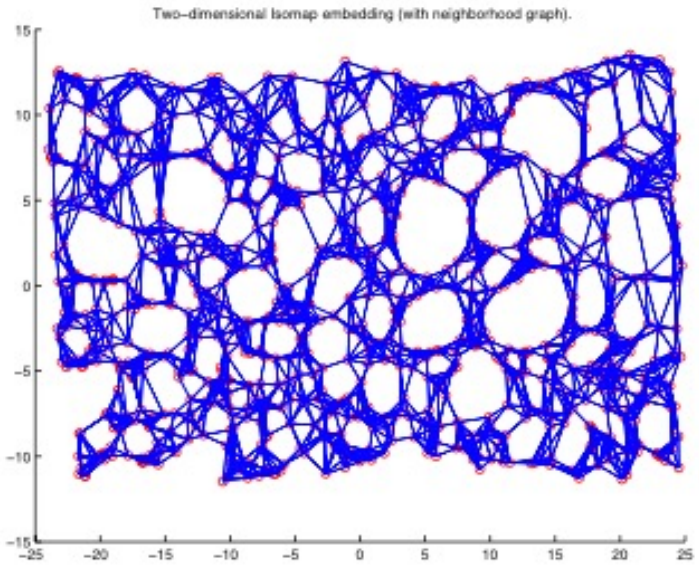
Some tips for effective dimensionality reduction

2019 - Nguyen - Ten quick tips for effective dimensionality reduction

1. Choose an appropriate method (linear or not, supervised or not)
2. Preprocess continuous and count input data
3. Handle categorical input data appropriately
4. Use embedding methods for reducing similarity and dissimilarity input data
5. Consciously decide on the number of dimensions to retain
6. Apply the correct aspect ratio for your visualizations
7. Understand the meaning of the new dimensions
8. Find the hidden signal
9. Take advantage of multidomain data
10. Check the robustness of your results and quantify uncertainties

ML Fundamentals – Lecture 8

- Curse of dimensionality
- Representation learning
- Manifold learning
  - Isomap
  - LLE
- Visualization
  - t-SNE
  - UMAP



Next:
Deep learning