EECS 553 Lecture 19
Machine Learning (ECE)
[class version]

Jeff Fessler

University of Michigan
©Laura Balzano and Jeff Fessler, UM, November 5, 2026, 12:49

F26

Logistics

© 2026 Fessler

- HW 6 due Wed. Nov. 4
- PA3: Fri. Nov. 6: draft reproducibility report
  Overleaf invites essential!!
  Must have "553" and "team (#)" in overleaf project name
- HW 7 due Wed. Nov. 11
  Some non-graded problems for timing flexibility
  Last HW before Exam 2
- Exam 2: Mon. Nov. 16, 6-8PM; 1013 Dow; 2 sheets
  W26 practice exam released on Canvas

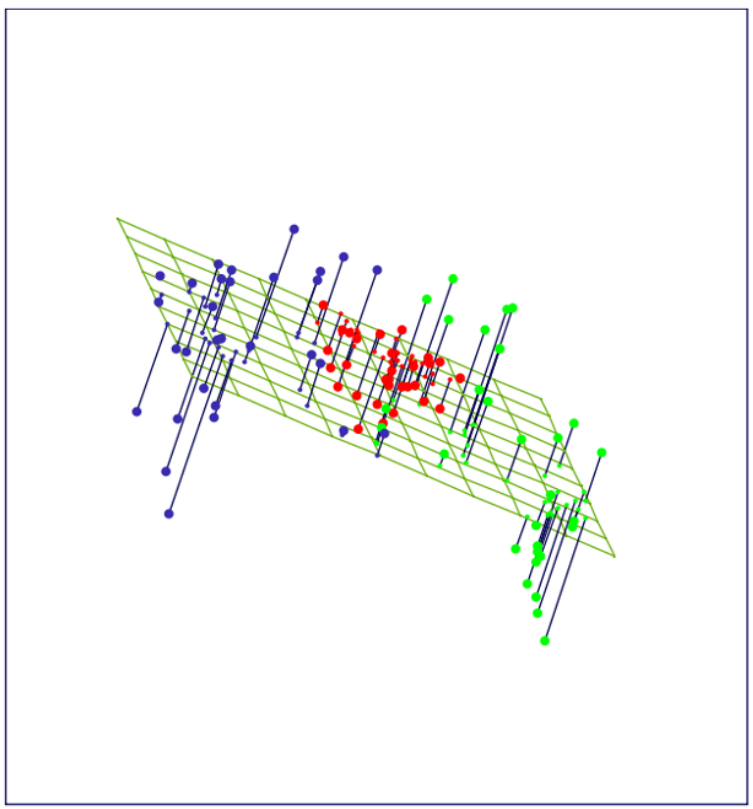
Recap/Outline

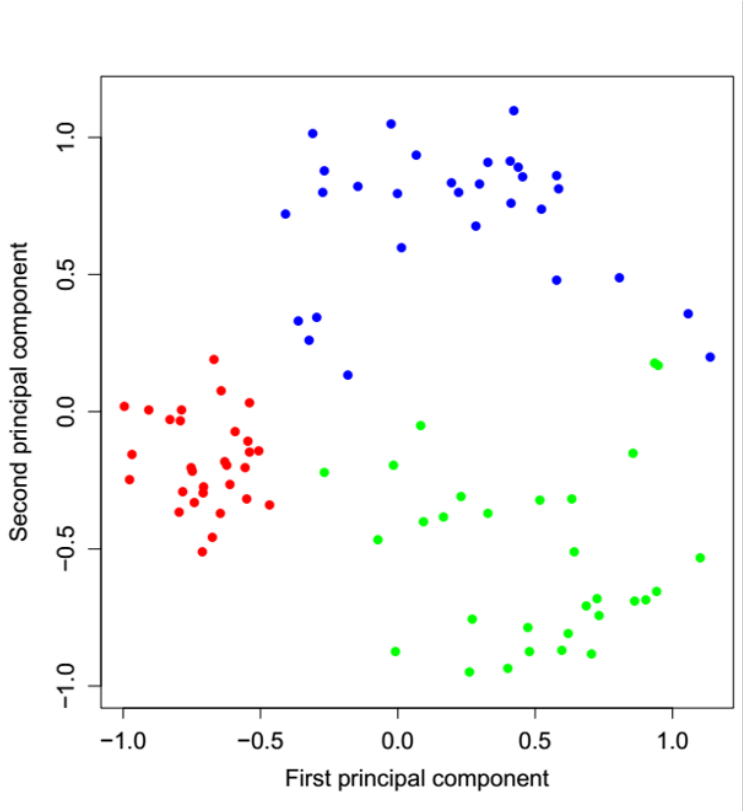
© 2026 Fessler

- Recap
  - Unsupervised learning / embedding / PCA
- Today
  - PCA recap
  - Embedding and dimensionality reduction
  - Nonlinear dimensionality reduction
    - t-Stochastic Neighbor Embedding (t-SNE)
  - Clustering
    - k-Means
    - Lloyd's algorithm

Example of applying PCA with $d = 3, k = 2$ for visualization.

$$\hat{\boldsymbol{x}}_i \approx \boldsymbol{Q} \underbrace{\boldsymbol{Q}^\top \boldsymbol{x}_i}_{\text{PCs: } \boldsymbol{\theta}_i \in \mathbb{R}^2}$$





label axes:

EOM

Embedding and dimensionality reduction: Goal © 2026 Fessler

The goal of **embedding** aka **dimensionality reduction** methods is to map a (potentially) high-dimensional dataset into a low-dimensional space in a way that preserves global and/or local

The term as used in ML may be somewhat counter-intuitive compared to the English meaning of embed.

Embedding vs dimensionality reduction

© 2026 Fessler

- **Embedding** in ML generally means transforming data into a lower-dimensional Euclidean-space representation.
  - The initial data may be real-valued, but often is not.
  - The goal is typically to find a useful task-oriented representation.

- **Dimensionality reduction** also means to transform to a lower-dimensional representation.
  - It typically assumes the initial data are at least numerical and sometimes real-valued.
  - It can provide data compression and can reduce computation.

Embedding and dim. reduction: Examples

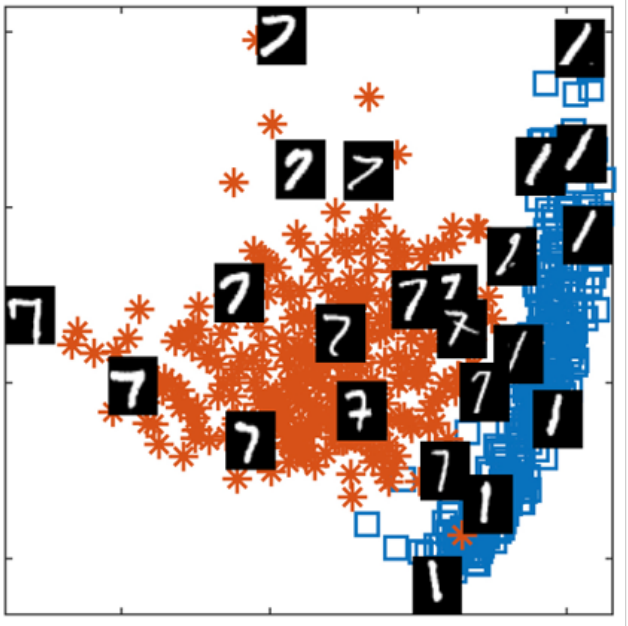
© 2026 Fessler

- Embed words into $\mathbb{R}^d$ for $d \approx 100 - 1000$ in natural language processing (*e.g.*, **Word2vec**).
- Embed commercial products (categorical data) in $\mathbb{R}^d$ for $d \approx 20 - 100$ in recommender systems.
- Embed songs into $\mathbb{R}^d$ for $d \approx 20 - 50$ for playlist prediction [2]. Songs have different lengths so raw audio waveforms are unsuitable for finding similarities.

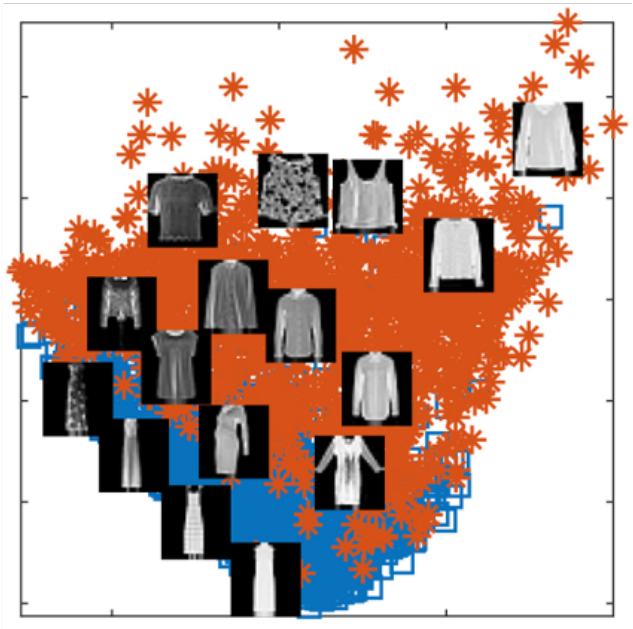
- Embed images in $\mathbb{R}^d$ where $d \ll$ number of pixels.
  Could also be called dimension reduction, especially if new features are learned by a generic process.

- Reduce the dimension of data by subsampling it.
- Reduce the dimension of data for visualization purposes.
- Reduce the dimension of a large state-space representation of a dynamical system to facilitate computation.

Applications of embedding: image classification © 2026 Fessler



First 2 PCs for MNIST 1s and 7s



First 2 PCs for FMNIST shirts and dresses

Figure courtesy [3].

Every digit classification example all semester long has used PCA to embed MNIST images into 2D ($k = 2$) to facilitate visualization.
Right: Fashion-MNIST: [4]

Applications of embedding: image denoising

© 2026 Fessler

- Image denoising is a key component of **diffusion models** for image generation.
- Here $16 \times 16 \times 3$ image patches ($d = 768$) are "bottleneck" embedded in $\approx$ 64-dimensional space
- Impressive image generation results

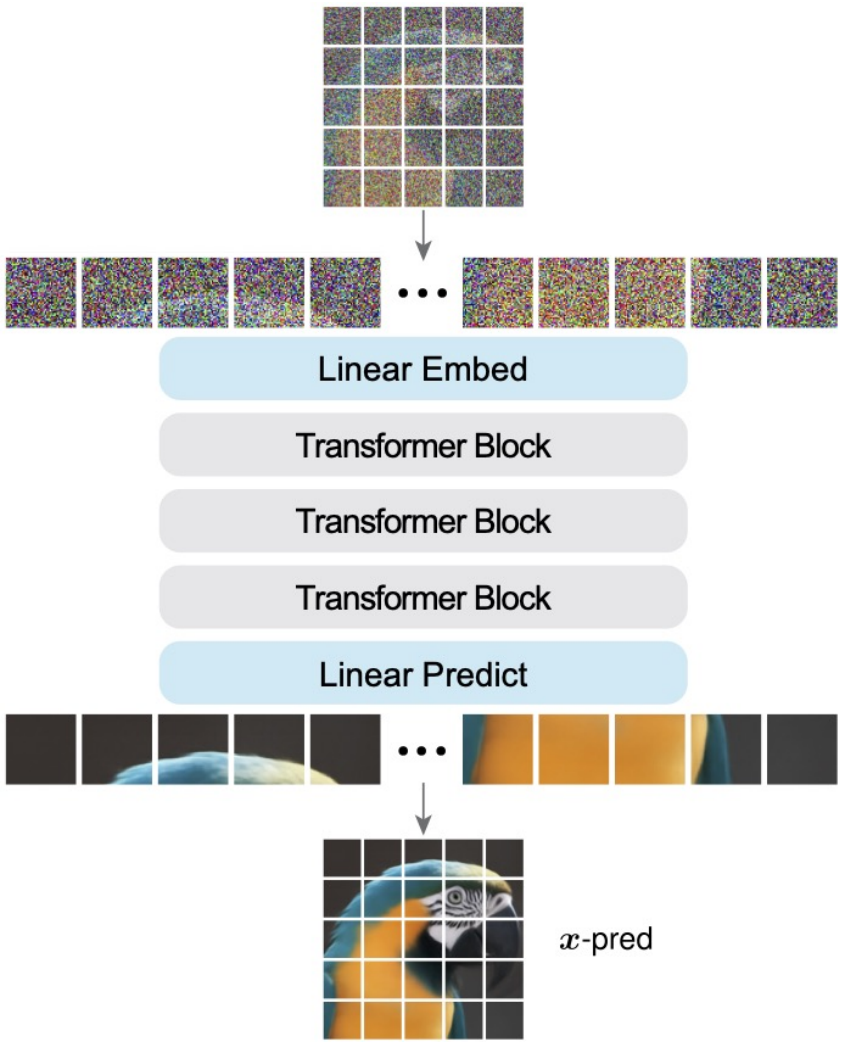


Figure from [5]

Applications: genetics

© 2026 Fessler

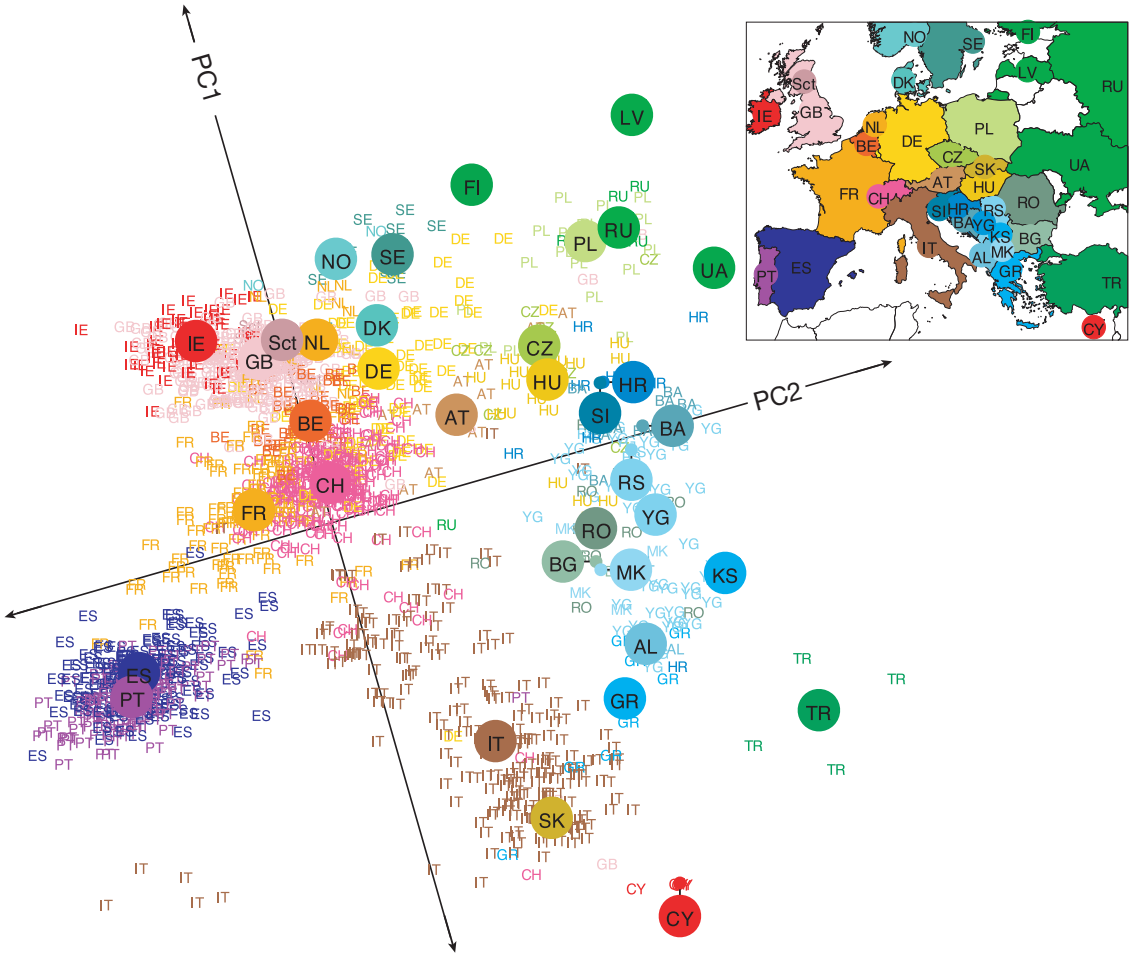


Figure courtesy [6].

PCA with on human genotype data from $d = 197,146$ loci/SNPs in $n = 1,387$ European individuals.

Applications: cell biology

© 2026 Fessler

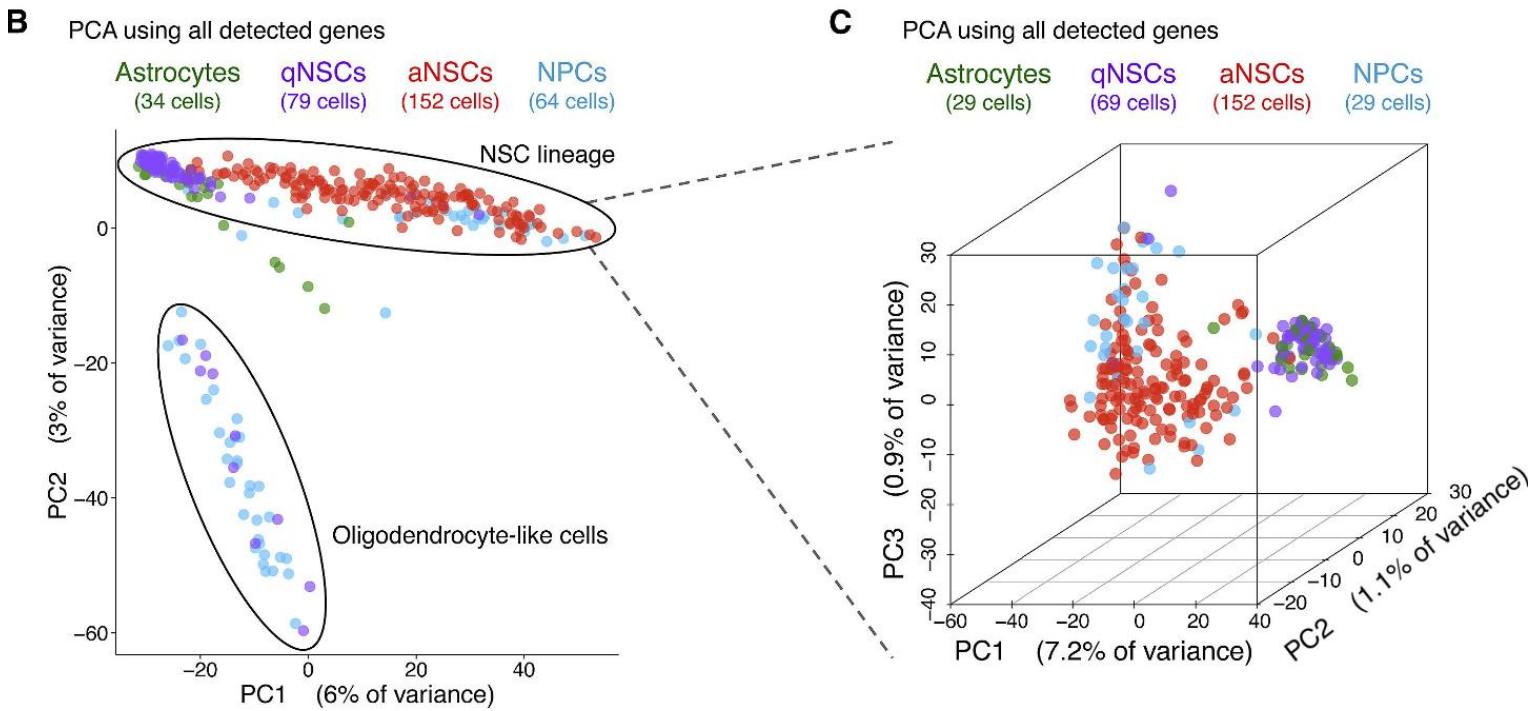


Figure courtesy [7]. DOI

Exploratory analysis via PCA with $k = 2$ and $k = 3$ of single-cell gene data:

- Quiescent neural stem cells (qNSC)
- Activated neural stem cells (aNSC)
- Neural progenitor cells (NPC)

Gemini said $d = 14,551$ genes, but when pressed it admitted that it had hallucinated and claimed that a typical number is 12,000 to 18,000 genes.

- PCA is a linear transformation of the original data.
- Some datasets have nonlinear structures that are not captured by linear transformations.

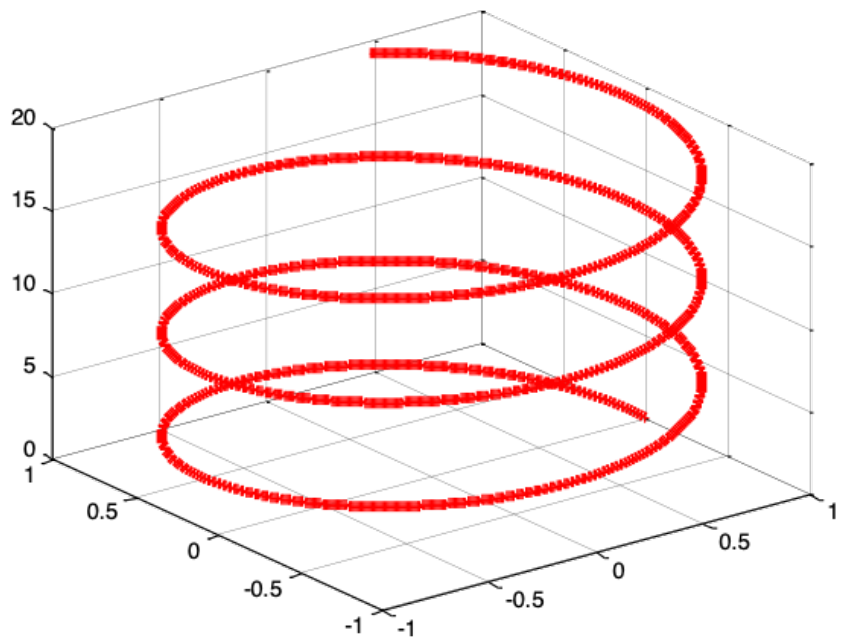


Figure from [8].

- This is a 1D 1-dimensional **manifold** in $d = 3$
- PCA with $k = 1$ would be terrible (globally).
- label axes

- There are many **nonlinear dimensionality reduction** methods including **independent component analysis** (**ICA**) and **multidimensional scaling** (**MDS**) [9, p. 569].

Intuition: how does your brain represent these images?



Figure from [8].

Given the 3D face surface geometry, we can generate all of these images from: the (point source) lighting direction, the vertical and horizontal pose $\implies$ images are on low-dimensional **manifold** [10, 11, 12].

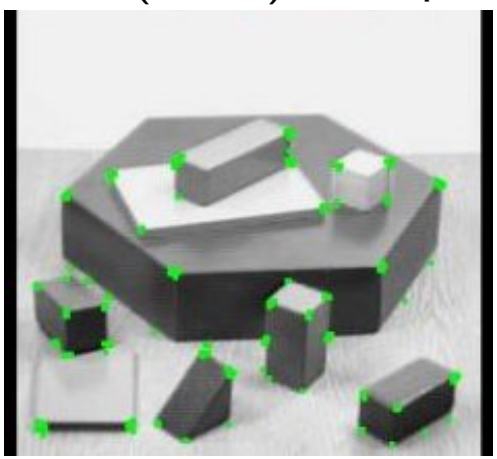
- Canny edges



Image courtesy pyimagesearch.com.

- Canny edges
- Harris corners : important "keypoints" for image matching.
- SIFT keypoints : scale invariant feature transform.

  This combines Harris corners with HOG (histogram of gradients) and corner detection with scale-invariance so that a keypoint can be traced across images despite scale (zoom). Computed for only the keypoints.



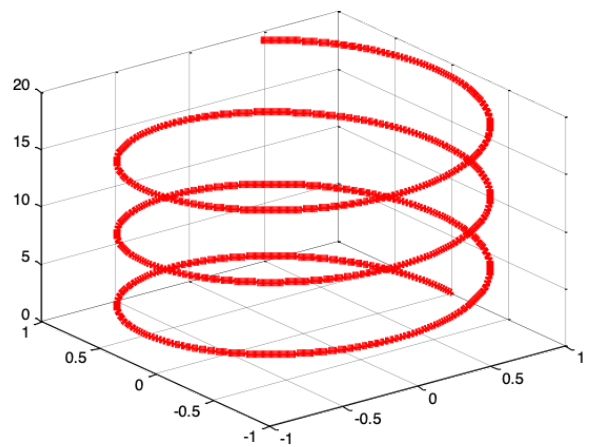


Harris image courtesy opencv.org; SIFT image courtesy opencv.org.

- Canny edges
- Harris corners : important "keypoints" for image matching.
- SIFT keypoints : scale invariant feature transform.
  This combines Harris corners with HOG (histogram of gradients) and corner detection with scale-invariance so that a keypoint can be traced across images despite scale (zoom). Computed for only the keypoints.
- SURF (faster SIFT)
- BRIEF (compressed memory SIFT)
- ORB (open source version of above; SIFT patent expired in 2020)

What if we don't know what nonlinear functions to apply for nonlinear features - and we want to learn them from data? Foundations: **Laplacian eigenmaps** and **manifolds**.

- **Locally linear embedding**
- **t-Stochastic neighbor embedding** (**t-SNE**)
- **Uniform manifold approximation and projection** (**UMAP**)
- Pairwise Controlled Manifold Approximation (PaCMAP) [13]



t-stochastic neighbor embedding (**t-SNE**) is a nonlinear dimensionality reduction method for visualization that attempts to match the distribution of "pairwise affinities" in the dataset from high- to low-dimensions.
Building on **stochastic neighbor embedding** (**SNE**) [14], t-SNE was proposed in [15].

- SNE converts Euclidean distances between high-dimensional data points $\boldsymbol{x}_i \in \mathbb{R}^d$ into conditional probabilities representing "affinity" values between *pairs*: [14] [16, Eqn. 1]:

$$p_{\boldsymbol{x}_j|\boldsymbol{x}_i} \triangleq \frac{\exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2/(2\sigma_i^2)\right)}{\sum_{l \neq i}} \mathbb{I}_{\{i \neq j\}}, \quad \forall i, j = 1, \ldots, n$$

- σ_i is a selected via **peplexity**; think "smooth number of neighbors."
- We want to choose embedded points $\boldsymbol{y}_i = \boldsymbol{\Phi}(\boldsymbol{x}_i) \in \mathbb{R}^k$ for $k \ll d$, without actually constructing $\boldsymbol{\Phi}$.
- Define a *similar* distribution in the low-dimensional space:

$$q_{\boldsymbol{y}_j|\boldsymbol{y}_i} = \frac{}{\sum_{l \neq i} \exp\left(-\|\boldsymbol{y}_i - \boldsymbol{y}_l\|_2^2\right)} \mathbb{I}_{\{i \neq j\}}, \quad \forall i, j = 1, \ldots, n.$$

- SNE chooses the embedded points $\{\boldsymbol{y}_i\}$ to "best" match these defined probability distributions $p_{\boldsymbol{x}_j|\boldsymbol{x}_i}$ and $q_{\boldsymbol{y}_j|\boldsymbol{y}_i}$.
- These are probabilities on pairwise distances (not on the original $\boldsymbol{x}$ values or new $\boldsymbol{y}$ values).
- How to define "best" for distributions here? Euclidean norm is unsuitable.
- A natural cost function for comparing distributions is the **KL-divergence**:

$$D_{\mathrm{KL}}(P_{j|i} \| Q_{j|i}) \triangleq \sum_i \sum_j p_{x_j|x_i} \log \frac{p_{x_j|x_i}}{\quad}.$$

- In t-SNE, a key contribution was using the **Student-t distribution** for the joint probabilities q_{y_j,y_i} in the reduced space:

$$q_{\boldsymbol{y}_j|\boldsymbol{y}_i} = \frac{\left(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2\right)^{-1}}{\sum_k \sum_{l \neq k} \left(1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|_2^2\right)^{-1}} \mathbb{I}_{\{i \neq j\}}, \quad \forall i, j = 1, \ldots, n.$$

- Uses t with 1 degree of freedom, with is same as **Cauchy distribution**.
- Avoids the "crowding" problem, preventing moderate-distance points from collapsing into a dense, jumbled center.
- t-SNE also uses a KL divergence of symmetricized distribution:

$$D_{\mathrm{KL}}(P \| Q) \triangleq \sum_i \sum_j p_{ij} \log \frac{pij}{q_{ij}}, \quad p_{ij} = \qquad .$$

  - Helps keep outliers isolated, due to .
  - Improves GD convergence

The KL-divergence of two distributions defined on the same probability space $\mathcal{X}$ is defined as

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right).$$

Q19.1 This divergence

A: is symmetric in P, Q

B: is zero if $P = Q$

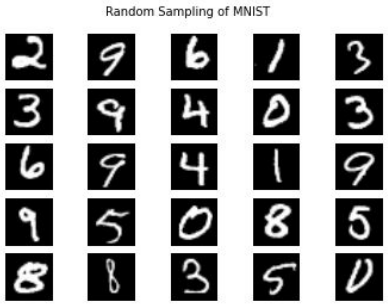
C: allows the support of P to be larger than the support of Q

D: B and C

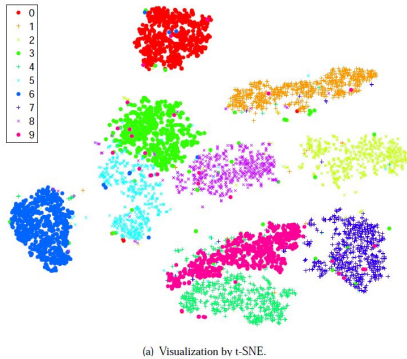
E: all of these

Example in JULIA TSne.jl package

Uses an efficient **quadtree** data structure implementation of the n^2 distances.



Visualizing Data using t-SNE



(a) Visualization by t-SNE.

Laurens van der Maaten, and Geoffrey Hinton
Tilburg University. The Netherlands
University of Toronto. Canada

Journal of Machine Learning Research (2008)

Presented to MIAL by: Jeremy Kawahara (2015)
slides from→ http://kawahara.ca/visualizing-data-using-t-sne-slides/

1

slides from [17]

t-SNE algorithm overview

© 2026 Fessler

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$
  for *t=1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
  end
end

The basic algorithm …

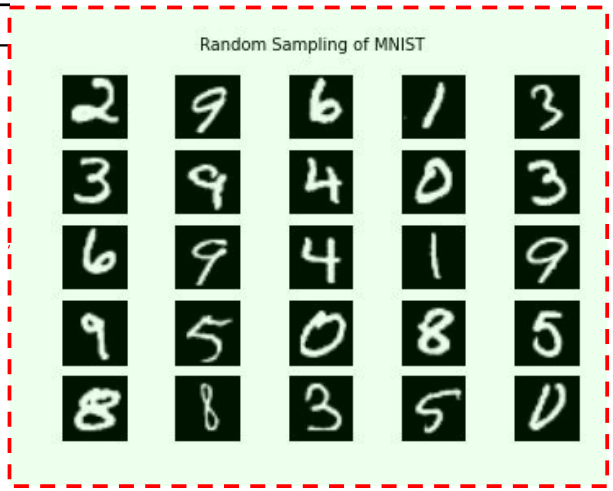
13

t-SNE inputs

© 2026 Fessler

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
  end
end



15

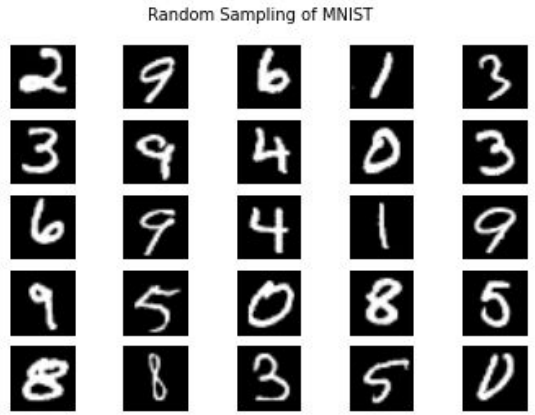
The **perplexity** hyperparameter here is our

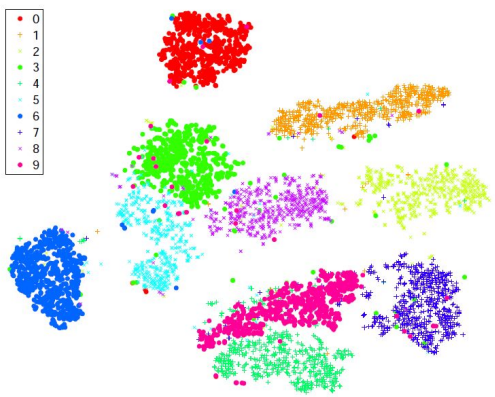
t-SNE output

© 2026 Fessler

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for *t=1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end





(a) Visualization by t-SNE.

16

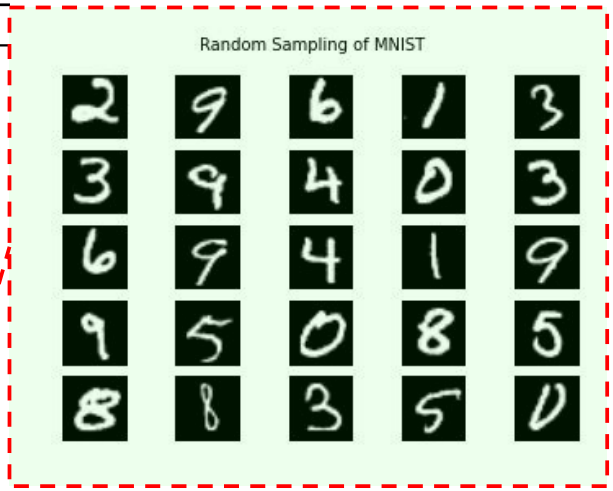
t-SNE affinities

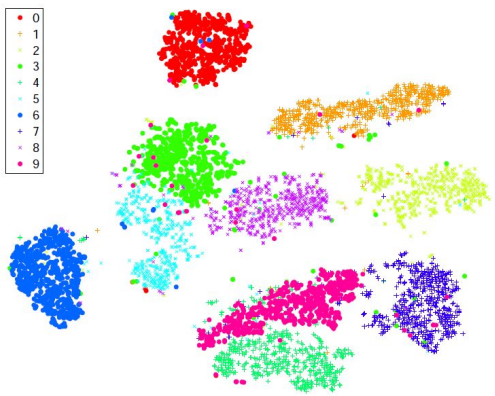
© 2026 Fessler

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Compute probabilities ***P*** that ***xi*** and ***xj*** are neighbors (based on Euclidian distance in **high-d** space)





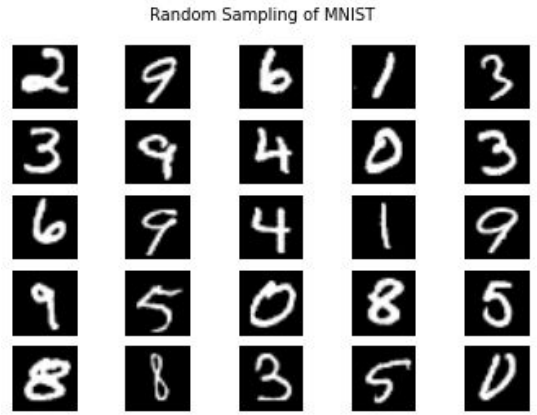
(a) Visualization by t-SNE.

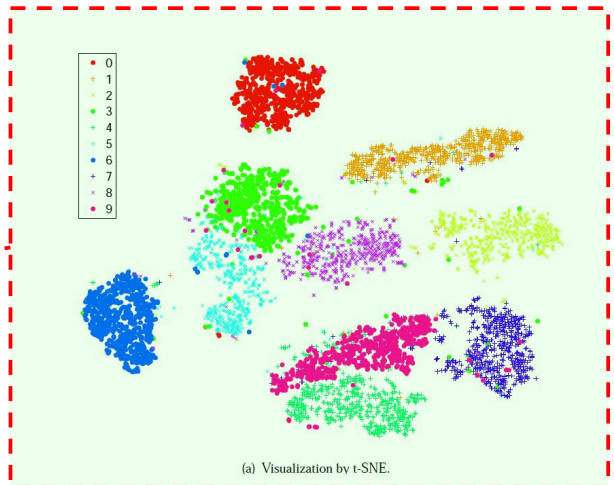
17

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$
  for $t = 1$ **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
  end
end

Compute probabilities ***Q*** that ***yi*** and ***yj*** are neighbors (corresponding to *xi, xj*) (based on Euclidian distance in **low-d** space)





18

t-SNE principle

© 2026 Fessler

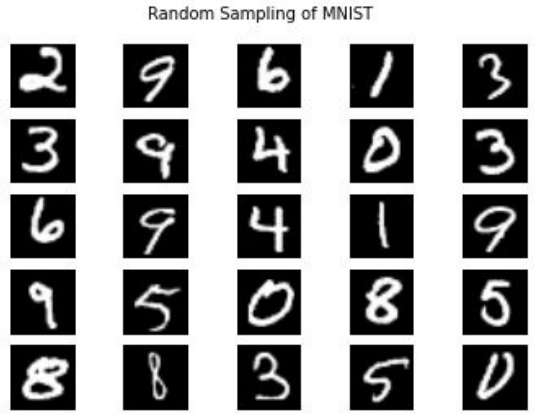
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

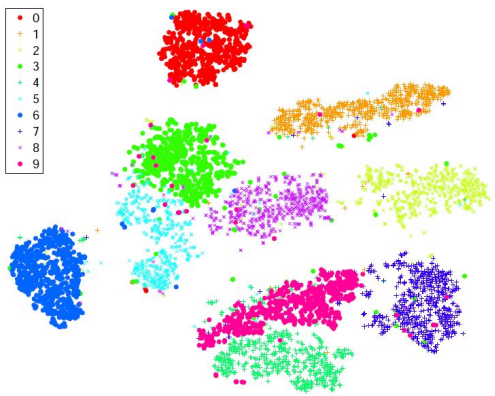
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.

begin
- compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
- set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
- sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$
- **for** t=1 **to** T **do**
  - compute low-dimensional affinities q_{ij} (using Equation 4)
  - compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
  - set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
- **end**

end

Key assumption is that the **high-d** *P* and the **low-d** *Q* probability distributions should be the same





(a) Visualization by t-SNE.

19

t-SNE cost gradient

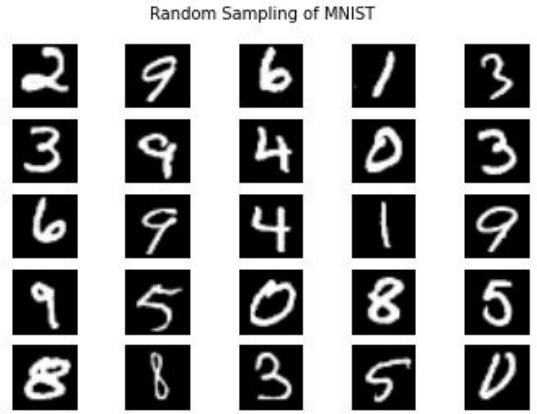
© 2026 Fessler

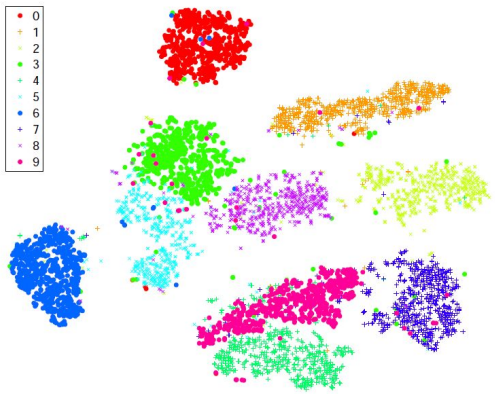
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$
  for *t=1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Find a **low-d** map that minimizes the difference between the ***P*** (high-d) and ***Q*** (low-d) distributions

(if *xi,xj* has high probability of being neighbors in **high-d**, then *yi,yj* should have high probability in **low-d**)





(a) Visualization by t-SNE.

20

The gradient of the Kullback-Leibler divergence between P and the Student-t based joint probability distribution Q (computed using Equation 4) is derived in Appendix A, and is given by

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}. \quad (5)$$

t-SNE momentum

© 2026 Fessler

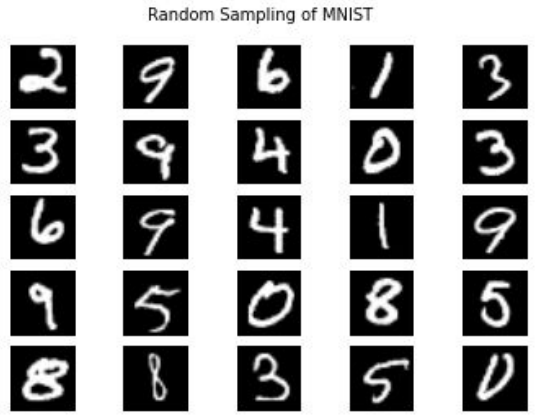
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

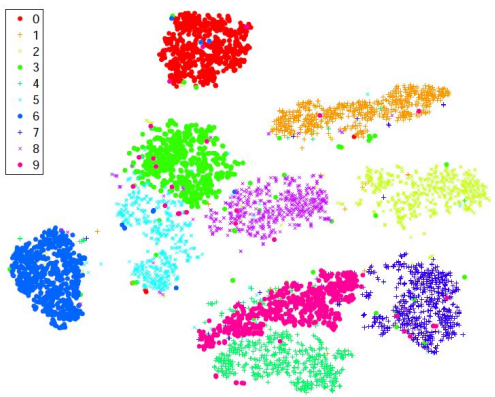
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.

begin
- compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
- set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
- sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$
- **for** t=*1* **to** T **do**
  - compute low-dimensional affinities q_{ij} (using Equation 4)
  - compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
  - set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
- **end**

end

We will minimize the difference between the **high-d** and **low-d** maps using **gradient descent**



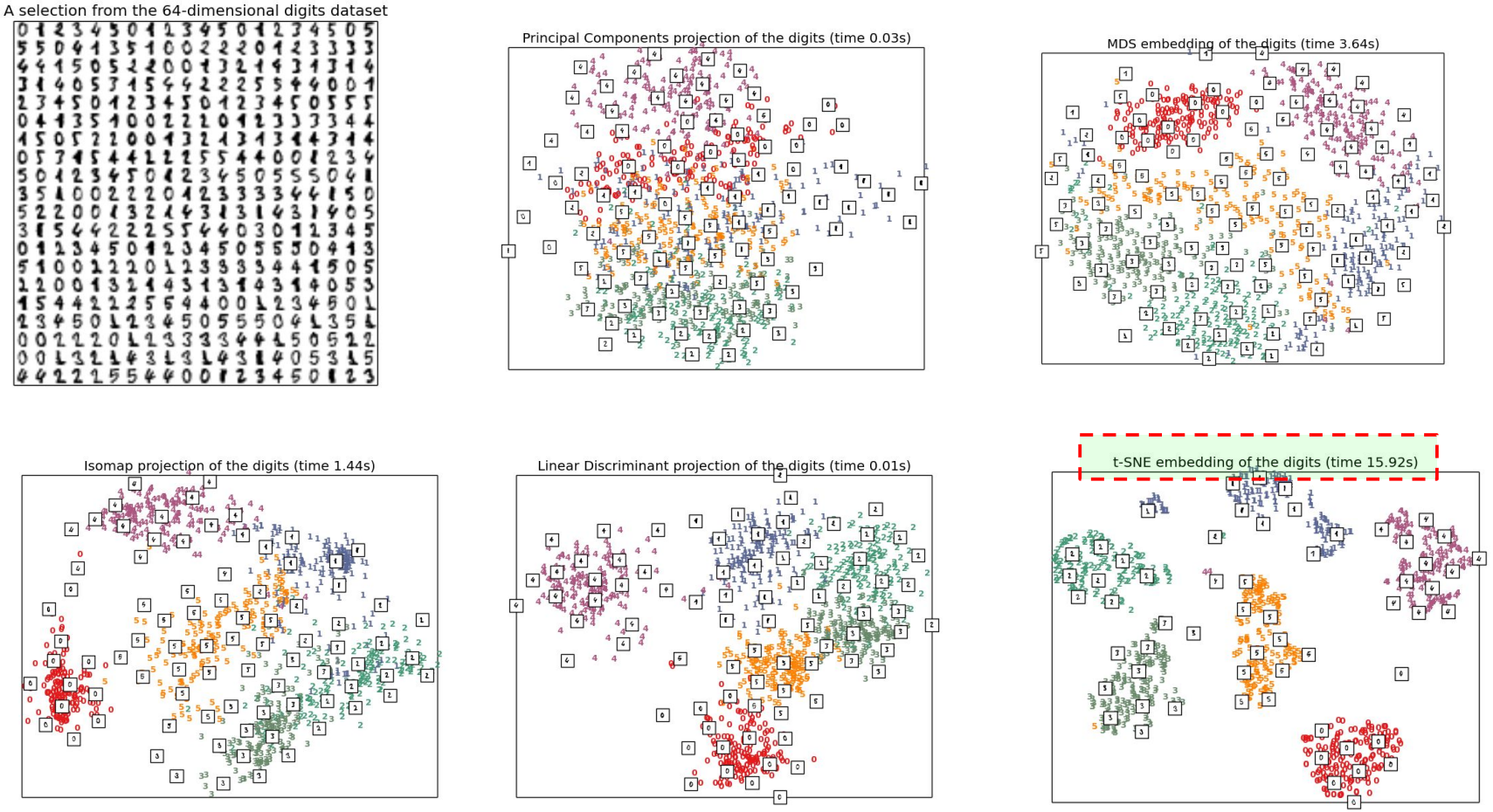


(a) Visualization by t-SNE.

21

t-SNE MNIST example

© 2026 Fessler



http://scikit-learn.org/stable/auto_examples/manifold/plot_lle_digits.html 7

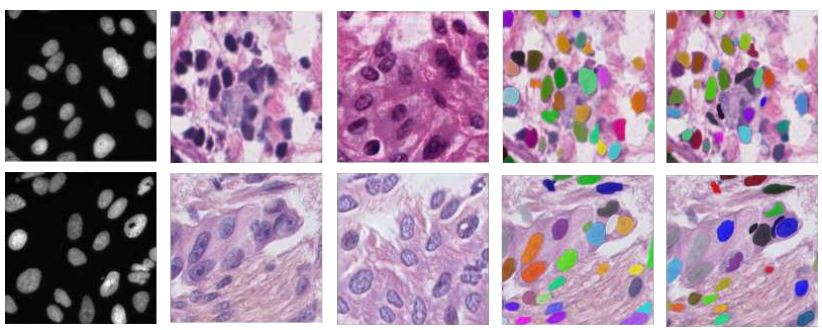
$K = 6$ for digits 0-5

t-SNE summary

© 2026 Fessler

- No Φ in t-SNE algorithm!
- Just move around embedded points $\{\boldsymbol{y}_i\}$
- Conceptually $\boldsymbol{y}_i = \Phi(\boldsymbol{x}_i), \ i = 1, \dots, n$
- But, cannot map new points
- Useful for visualization, but maybe less useful for downstream classification tasks
- In constrast, PCA learns the mapping

$$\boldsymbol{x} \mapsto \boldsymbol{\mu} + \boldsymbol{Q} \underbrace{\boldsymbol{Q}^\top (\boldsymbol{x} - \boldsymbol{\mu})}_{\text{embedding } \Phi(\boldsymbol{x})}.$$



Figures courtesy of [18] and [19].

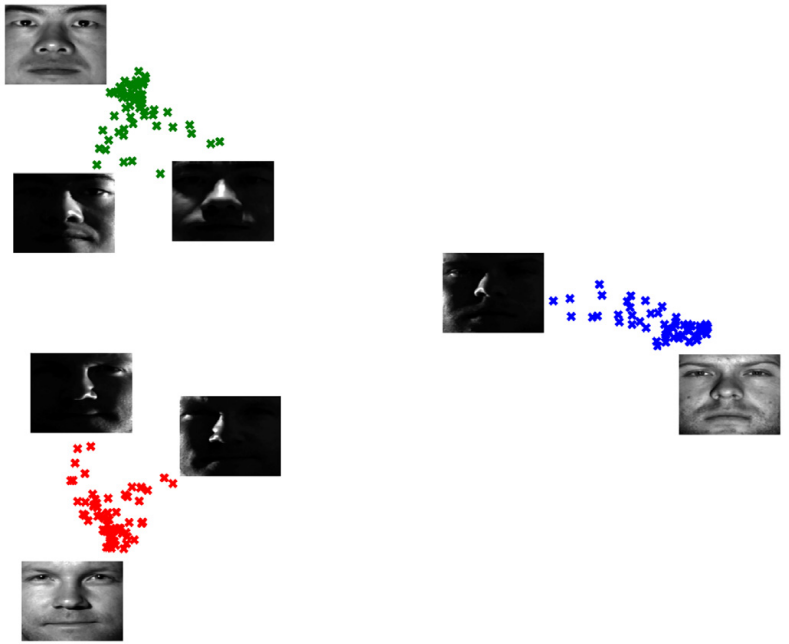


Fig. 2. Two-dimensional embedding of points in Extended Yale Face Database B, subjects 13, 26, and 38 using multidimensional scaling on the proposed point-to-point distance. The proposed distance provides an indication of which points lie near the estimated subspace and groups outliers with similar forms of shadow.

Clustering overview

© 2026 Fessler

- Given data $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ with $\boldsymbol{x}_i \in \mathbb{R}^d$.
- **Clustering** aims to partition $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ into disjoint subsets called **clusters** such that points in the same cluster are more similar to each other than to
- Want intra-class similarities higher than
- A partition of the data points, *i.e.*, a "clustering," can be represented by a **cluster map** function

$$C : $$

where k is the number of clusters. C is learned from all of $\mathcal{D}$.

- This definition of clustering hinges on a notion of similarity.

Clustering and applications

© 2026 Fessler

- The main goal of clustering is to
- Examples:
  - Internet network traffic analysis: clustering users by what content they request
  - Clinical trials: clustering outcomes
  - Marketing/sales: clustering customers
  - Finance: cluster stocks
  - Bioinformatics: clustering cell types
  - ...
- Clustering is often not the end of data analyses.
- Clusters serve to generate hypotheses for subsequent analysis.

Simplest clustering approach: K-Means

© 2026 Fessler

Let $\boldsymbol{x}_i \in \mathbb{R}^d$, $i = 1, \ldots, n$ denote data points. The **k-means method** groups data into k clusters by "solving"

- $\text{dist}(\boldsymbol{x}, \boldsymbol{y})$ is a often/typically Euclidean distance squared.
- $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ is a partition of $\{1, \ldots, n\}$,
- $\mathcal{M} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k)$ are called
- This optimization problem is NP-hard in general.

Let $\boldsymbol{x}_i \in \mathbb{R}^d, \ i = 1, \ldots, n$ denote data points that we wish to cluster into k clusters.

$$\arg\min_{\mathcal{C}, \mathcal{M}} \sum_{j=1}^{k} \sum_{i \,:\, \boldsymbol{x}_i \in \mathcal{C}_j} \text{dist}(\boldsymbol{x}_i, \boldsymbol{m}_j).$$

Finding the optimal solution to the k-means clustering problem for n observations in d dimensions is:

- NP-hard in general Euclidean space (of d dimensions) even for two clusters [20, 21].
- NP-hard for a general number of clusters k even for $d = 2$ [22].
- There are k^n possible partitions of the n points into k groups (including partitions where some groups are empty).
- Many k-means heuristics give good approximations to the solution of the k-means objective.

Assume we are using Euclidean distance (squared) in the k-means method: $\text{dist}(\boldsymbol{x}_i, \boldsymbol{m}_j) = \|\boldsymbol{x}_i - \boldsymbol{m}_j\|_2^2$. So we wish to solve:

$$\arg\min_{\mathcal{C}, \mathcal{M}} \sum_{j=1}^{k} \sum_{i \,:\, \boldsymbol{x}_i \in \mathcal{C}_j} \|\boldsymbol{x}_i - \boldsymbol{m}_j\|_2^2 .$$

Q19.2 If we are given the partition (the assignment of points to clusters) then the computation of the centers is closed-form.
Similarly, if we are given the centers (assume distinct), then the computation of each point's assignment is closed form.
A: (T,T) B: (T,F) C: (F,T) D: (F,F) **??**

Lloyd's algorithm for K-Means with Euclidean distance ©

2026 Fessler

Input: $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, k

Initialize $\boldsymbol{m}_1, \ldots . \boldsymbol{m}_k \in \mathbb{R}^d$

Repeat

  For $i = 1, \ldots, n$

    $C(i) = \arg\min_{\ell} \|\boldsymbol{x}_i - \boldsymbol{m}_\ell\|$

  End

  For $\ell = 1, \ldots, k$

    $\boldsymbol{m}_\ell = \frac{1}{|\{i : C(i) = \ell|} \sum_{i:C(i)=\ell} \boldsymbol{x}_i$

  End

Until the clusters don't change.

Output: C

Illustration: Old Faithful exploration

© 2026 Fessler

- **Old Faithful** is a geyser in **Yellowstone National Park** that erupts spectacularly for several minutes, every 35-120 minutes.
- It is only quasi-periodic.



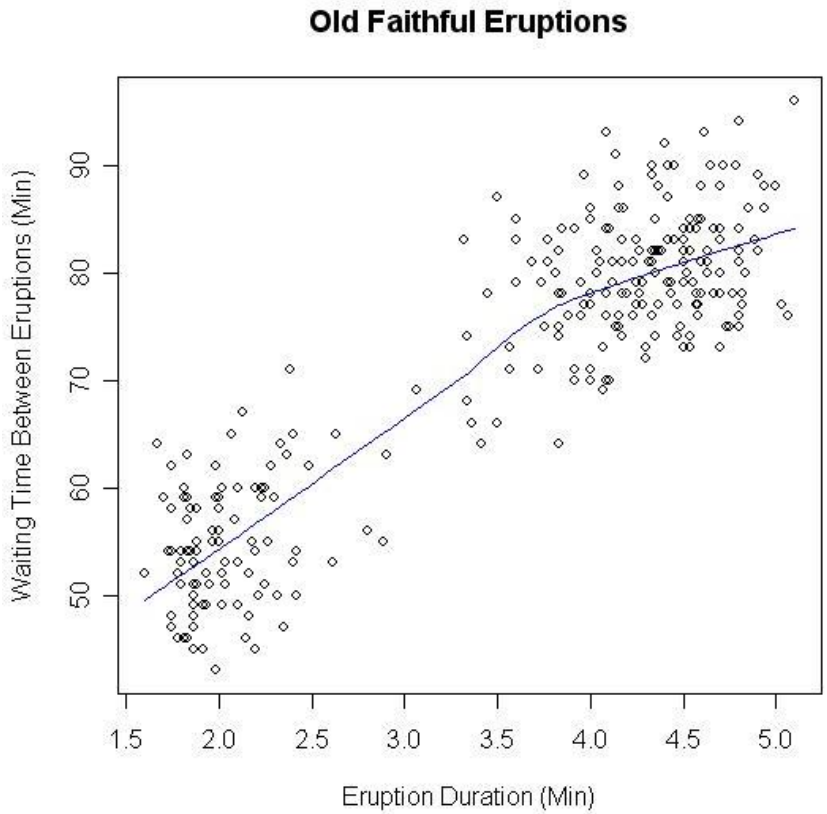
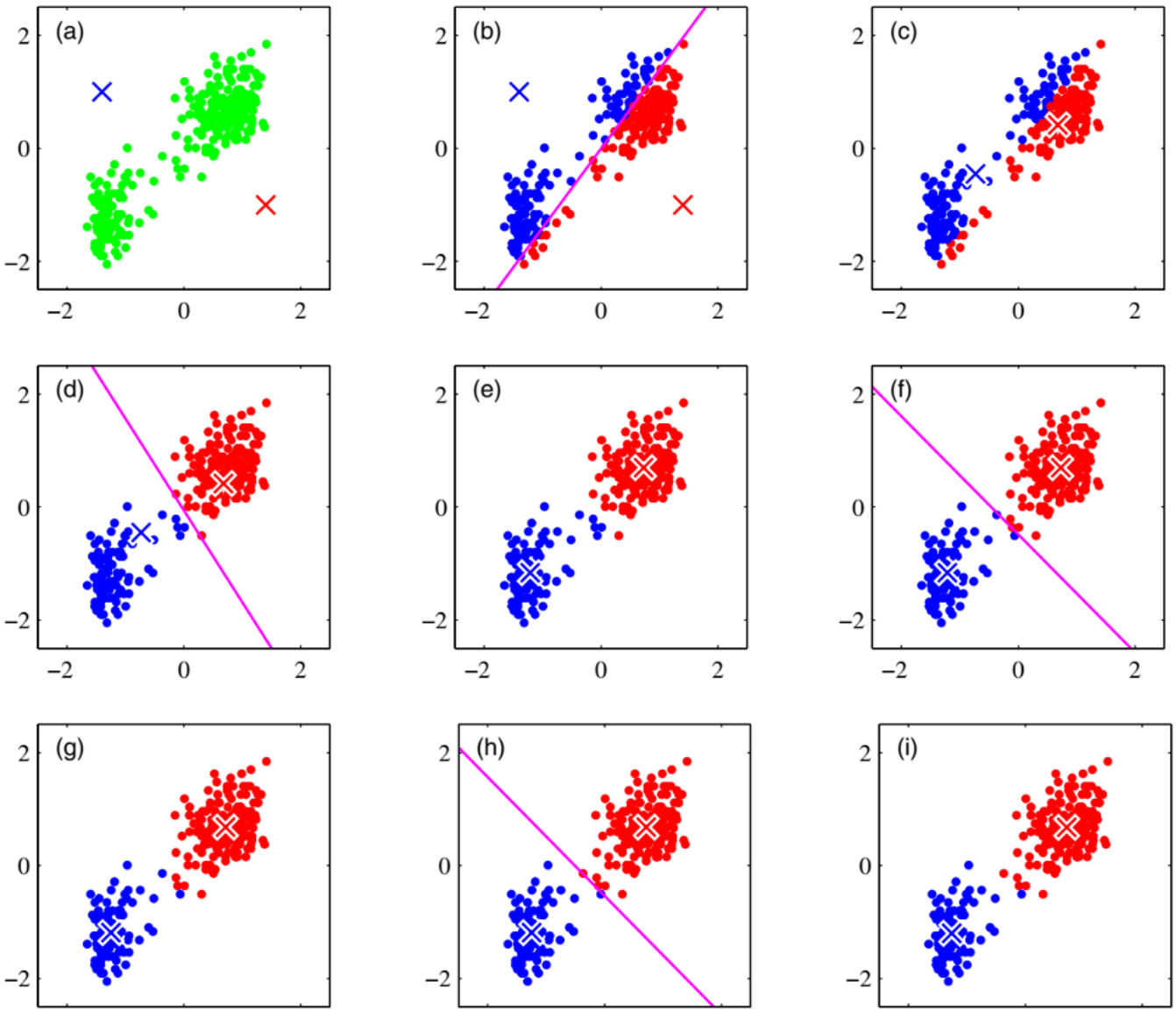


Illustration: Old Faithful results

© 2026 Fessler



Example from [23].

CQ: clustering

© 2026 Fessler

Recall clustering goal:
"Want intra-class similarities higher than inter-class similarities."

Q19.3 Did k-means method applied to Old Faithful data accomplish this goal?

A: Yes B: Mostly yes C: Maybe not D: No

Even though k-Means is a very classic clustering method, it has been used in modern ML for learnable sparse attenation [1, p. 531].

Clustering is a huge field

© 2026 Fessler

- We discussed
  - centroid based (*e.g.*, k-means)
  - distribution based (*e.g.*, GMM)

  See [9, p. 526] for one discussion of k-means vs GMM.
- Many other methods and research topics
  - hierarchical clustering
  - density-based clustering
  - graph clustering
  - clustering non-real-valued data
  - model selection
  - clustering quality metrics

References I

[1] K. P. Murphy.
Probabilistic machine learning: An introduction.
MIT Press, 2022.

[2] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims.
Playlist prediction via metric embedding.
In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 714–722, 2012.

[3] Alexander Ritchie, Laura Balzano, Daniel Kessler, Chandra S Sripada, and Clayton Scott.
Supervised pca: A multiobjective approach.
arXiv preprint arXiv:2011.05309, 2020.

[4] H. Xiao, K. Rasul, and R. Vollgraf.
Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms, 2017.

[5] T. Li and K. He.
Back to basics: Let denoising generative models denoise.
In *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, 2026.

[6] John Novembre, Toby Johnson, Katarzyna Bryc, Zoltán Kutalik, Adam R Boyko, Adam Auton, Amit Indap, Karen S King, Sven Bergmann, Matthew R Nelson, et al.
Genes mirror geography within europe.
Nature, 456(7218):98–101, 2008.

[7] Ben W Dulken, Dena S Leeman, Stéphane C Boutet, Katja Hebestreit, and Anne Brunet.
Single-cell transcriptomic analysis defines heterogeneity and transcriptional dynamics in the adult neural stem cell lineage.
Cell reports, 18(3):777–790, 2017.

References II

[8] M.R. Azimi.
Ece656-machine learning and adaptive systems lectures 23 & 24, 2015.
https://www.engr.colostate.edu/ECE656/lectures/lectures23-24.pdf.

[9] R. O. Duda, P. E. Hart, and D. G. Stork.
Pattern classification.
Wiley, New York, 2001.

[10] H. Murase and S. K. Nayar.
Illumination planning for object recognition using parametric eigenspaces.
IEEE Trans. Patt. Anal. Mach. Int., 16(12):1219–27, December 1994.

[11] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman.
From few to many: illumination cone models for face recognition under variable lighting and pose.
IEEE Trans. Patt. Anal. Mach. Int., 23(6):643–60, June 2001.

[12] S. Barsky and M. Petrou.
The 4-source photometric stereo technique for three-dimensional surfaces in the presence of highlights and shadows.
IEEE Trans. Patt. Anal. Mach. Int., 25(10):1239–52, October 2003.

[13] Y. Wang, H. Huang, C. Rudin, and Y. Shaposhnik.
Understanding how dimension reduction tools work: an empirical approach to deciphering t-SNE, UMAP, triMap, and PaCMAP for data visualization.
J. Mach. Learning Res., 22(201):1–73, 2021.

[14] Geoffrey E Hinton and Sam Roweis.
Stochastic neighbor embedding.
Advances in neural information processing systems, 15, 2002.

References III

[15] Laurens Van der Maaten and Geoffrey Hinton.
Visualizing data using t-sne.
Journal of machine learning research, 9(11), 2008.

[16] L. van der Maaten and G. Hinton.
Visualizing data using t-SNE.
J. Mach. Learning Res., 9(86):2579–605, 2008.

[17] Jeremy Kawahara.
Visualizing data using t-SNE, 2015.
http://kawahara.ca/visualizing-data-using-t-sne-slides/.

[18] Dongnan Liu, Donghao Zhang, Yang Song, Fan Zhang, Lauren O'Donnell, Heng Huang, Mei Chen, and Weidong Cai.
Unsupervised instance segmentation in microscopy images via panoptic domain adaptation and task re-weighting.
In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4243–4252, 2020.

[19] John Lipor and Laura Balzano.
Clustering quality metrics for subspace clustering.
Pattern Recognition, 104:107328, 2020.

[20] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat.
Np-hardness of euclidean sum-of-squares clustering.
Machine learning, 75(2):245–248, 2009.

[21] Sanjoy Dasgupta.
The hardness of k-means clustering.
Technical report, University of California, San Diego, 2008.

References IV

[22] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan.
The planar k-means problem is np-hard.
Theoretical computer science, 442:13–21, 2012.

[23] Christopher M Bishop and Nasser M Nasrabadi.
Pattern recognition and machine learning, volume 4.
Springer, 2006.