10-605/805 – ML for Large Datasets Lecture 4: Nonlinear Dimensionality Reduction

Henry Chai

9/8/22

Front Matter

- HW1 released 8/30, due 9/13 at 11:59 PM
- HW2 released 9/8 (today!), due 9/22 at 11:59 PM
  - **We strongly encourage you to start the programming portion early and refer back to HW1 programming if you get stuck!**
- Recitations on Friday, 11:50 – 1:10 (**different from lecture**) in GHC 4401 (**same as lecture**)
  - Recitation 2 on 9/9: Review of linear algebra

PCA Algorithm

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{n}, r$

1. Center the data
   A. Optionally, normalize the data by features so that all features are of the same scale
2. Compute the covariance matrix $C_X = X^T X$
3. Collect the top r eigenvectors (corresponding to the r largest eigenvalues), $P \in \mathbb{R}^{k \times r}$
4. Project the data into the space defined by P, $Z = XP$

- Output: Z, the latent representation ("PCA scores")

PCA Example: MNIST Digits

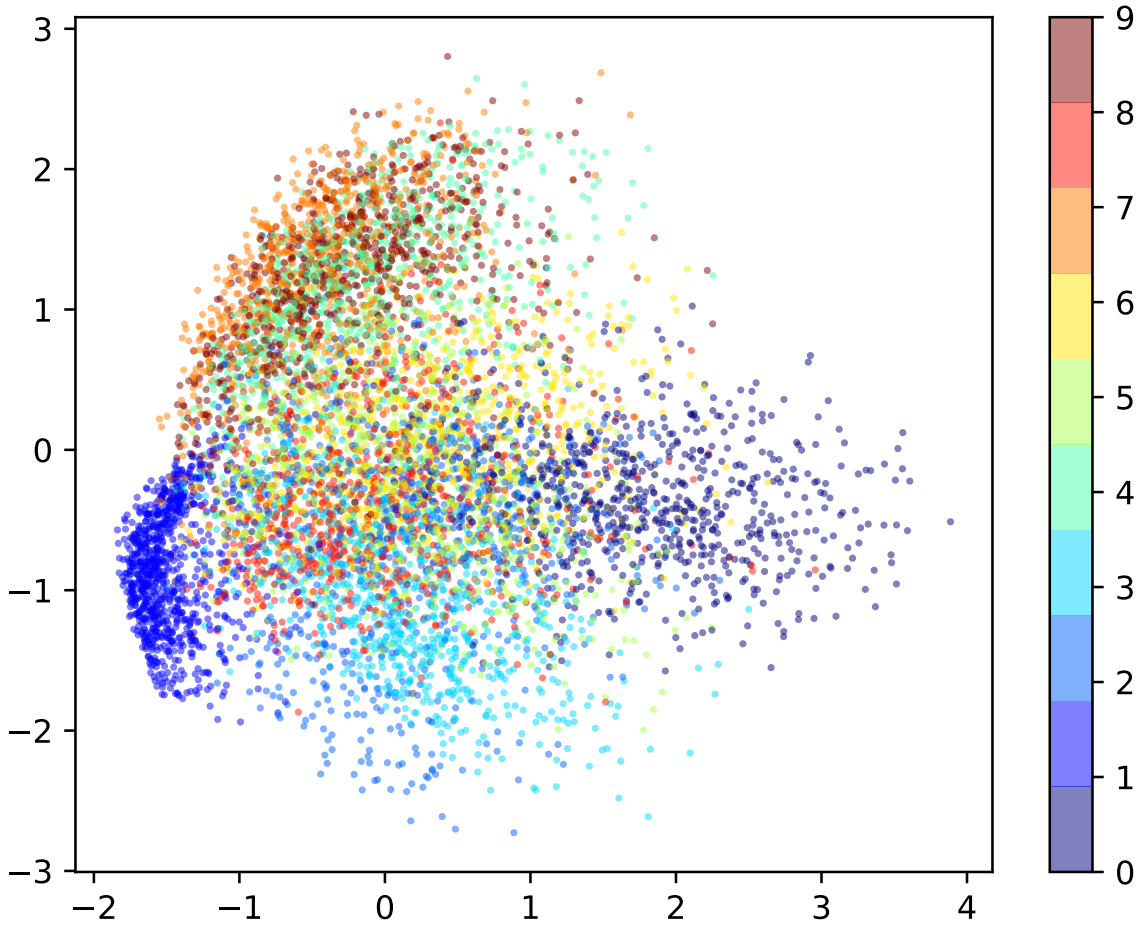
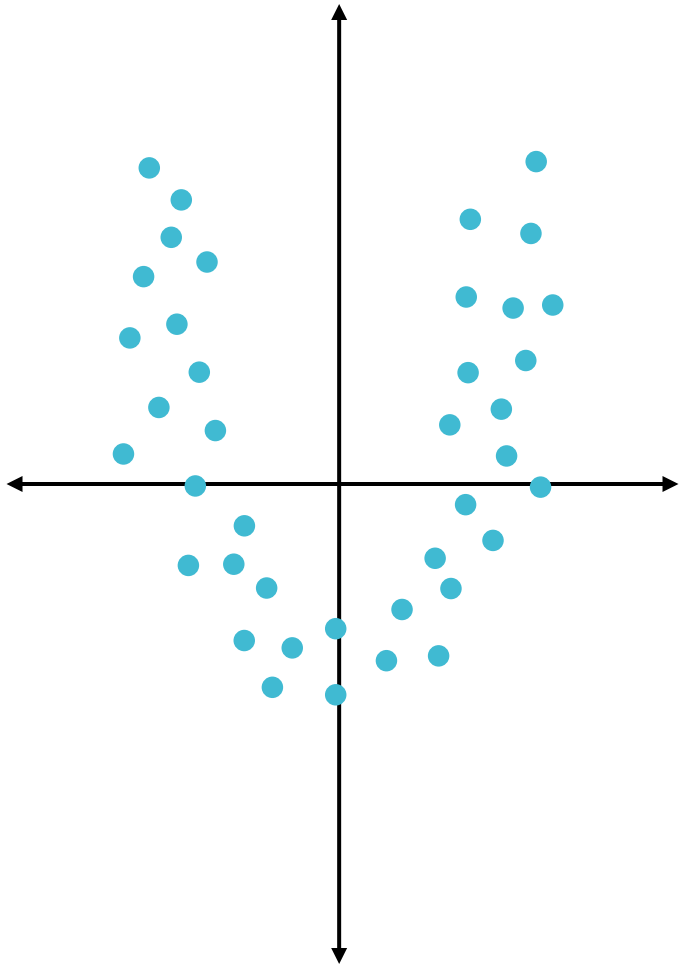


Figure courtesy of Matt Gormley

Shortcomings of PCA

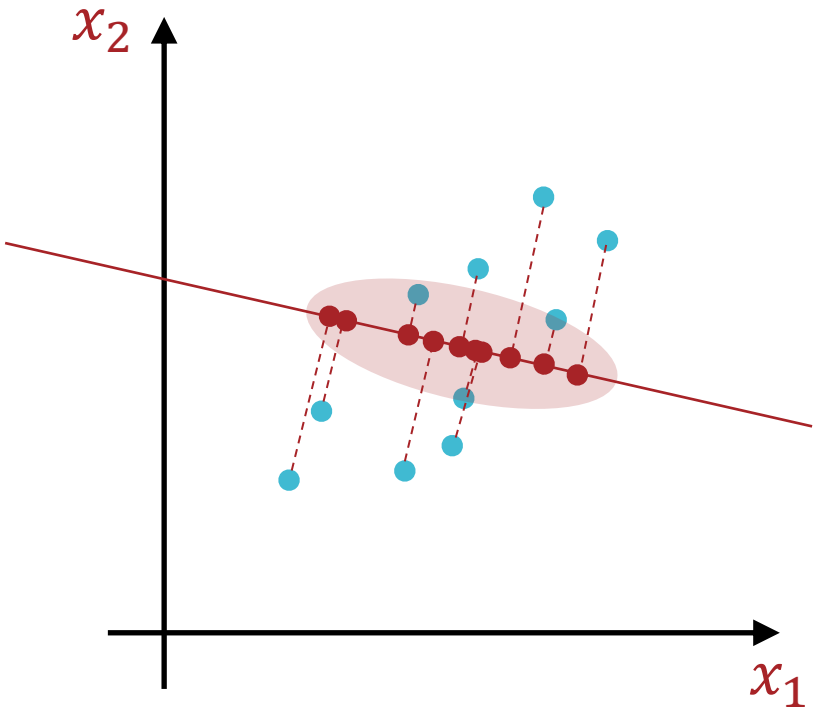


- Principal components are orthonormal
- Principal components are linear combinations of the features
- Principal components are expensive to compute

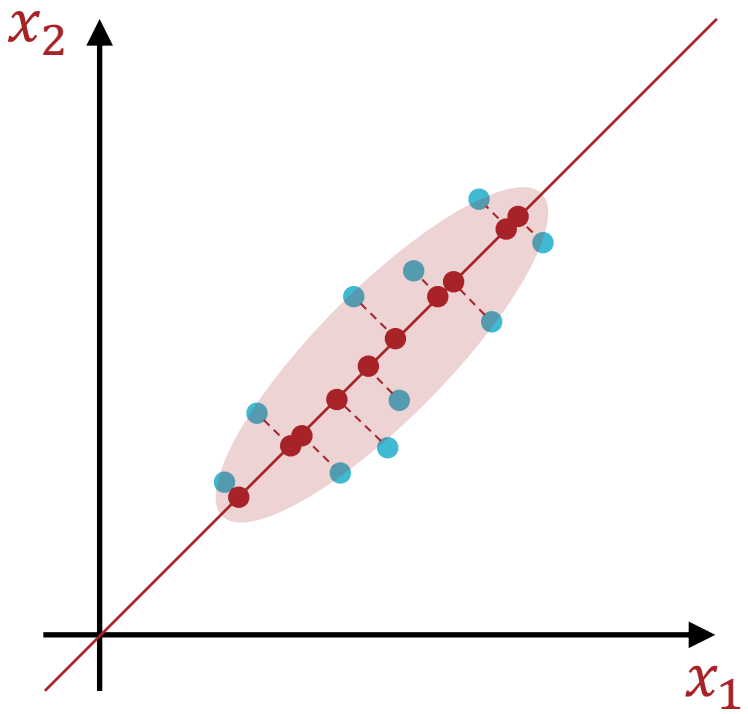
PCA Algorithm: Computational Cost

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{n}, r$

1. Center the data
   A. Optionally, normalize the data by features so that all features are of the same scale
2. Compute the covariance matrix $C_X = X^T X$ $(O(nk^2))$
3. Collect the top r eigenvectors (corresponding to the r largest eigenvalues), $P \in \mathbb{R}^{k \times r}$ $(O(k^3))$
4. Project the data into the space defined by P, $Z = XP$ $(O(nkr))$



Option A



Option B

Maybe Option A isn't so bad?

Random Projections?

- Issue: when k is very large, computing principal components can be intractable
  - PCA also requires the entire dataset to be available, which might not be possible (e.g., online learning)
- Idea: instead of rigorously minimizing the reconstruction error/maximizing the projections' variance, just pick a random set of vectors to project onto!
  - Each element of P is sampled from a standard Gaussian distribution, $P_{i,j} \sim N(0,1)$
  - Projections are still given by $Z = XP$
- Inquiry: is this a good idea? And if so, how good?

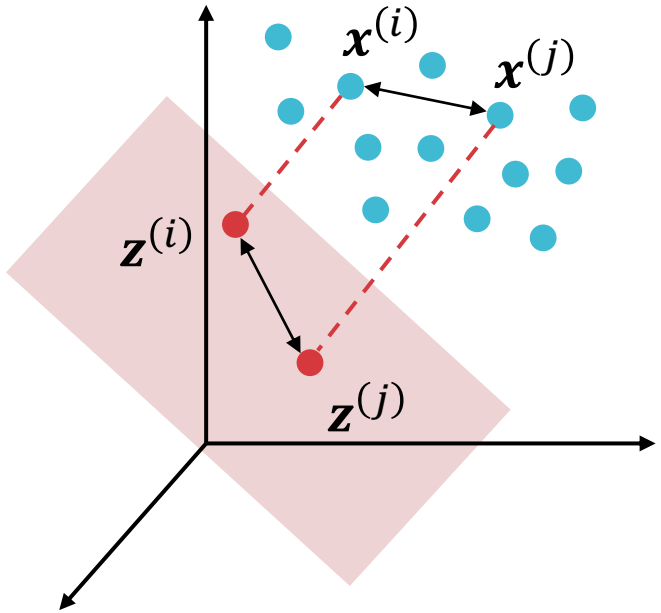
Random Projections: Computational Cost

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{n}, r$

1. ~~Center the data~~
   A. ~~Optionally, normalize the data by features so that all features are of the same scale~~
2. ~~Compute the covariance matrix $C_X = X^T X$ ($O(nk^2)$)~~
3. Generate P by sampling each element from a standard Gaussian distribution ($O(kr)$)
4. Project the data into the space defined by P, $Z = XP$ ($O(nkr)$)

Preserving Relative Distances

- Reasonable desideratum: distances between points are similar before and after being projected



- Formally:

$$\left\|\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}\right\|_2^2 \in \left[(1-\epsilon)\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\|_2^2, (1+\epsilon)\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\|_2^2\right]$$

$$\forall\, \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \in \mathcal{D}$$

Johnson-Lindenstrauss Lemma

- Given a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)})\}_{i=1}^{n}$ and $0 < \epsilon < 1$, if

$$r \geq \frac{4\ln(n)}{\epsilon^2/2 - \epsilon^3/3}$$

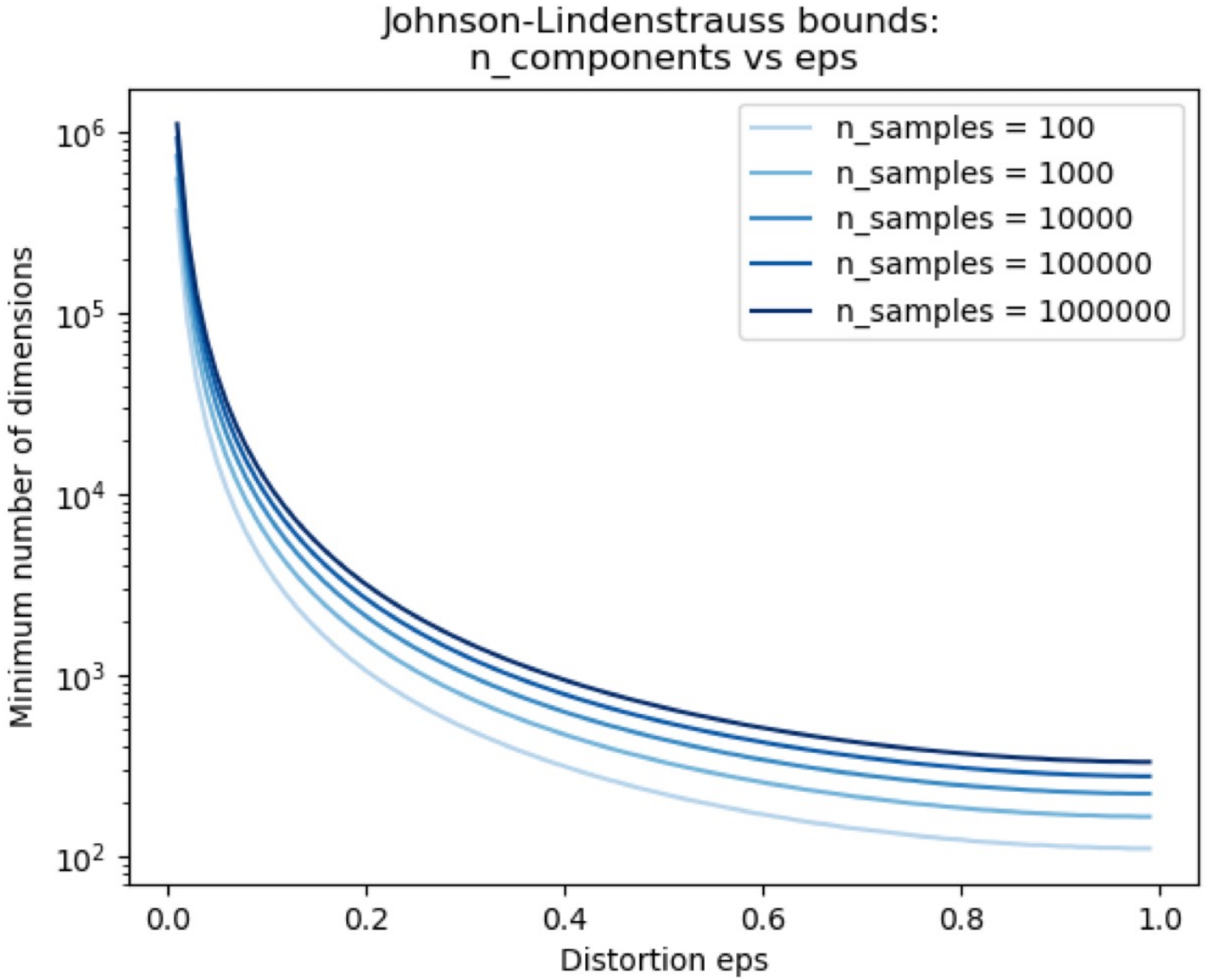
then ∃ a linear map $f\colon \mathbb{R}^k \to \mathbb{R}^r$ s.t.

$$\|\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}\|_2^2 \in \left[(1-\epsilon)\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|_2^2, (1+\epsilon)\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|_2^2\right]$$

$$\forall\, \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \in \mathcal{D}$$

- The bound on r does not depend on k!

Johnson-Lindenstrauss Lemma



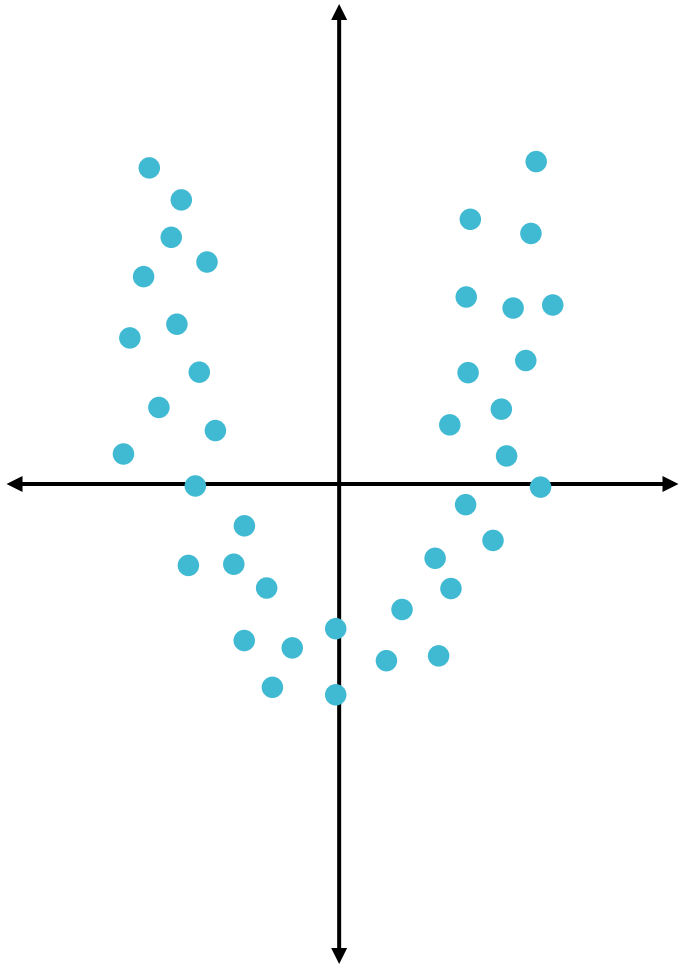
Source: https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_johnson_lindenstrauss_bound.html

Johnson-Lindenstrauss Lemma: Proof (Sketch)

1. Construct f randomly by sampling from a standard Gaussian
2. Show that the probability that f satisfies the desired condition is > 0 (specifically, $> {}^{1}/_{n}$)
   - Use concentration of measure: compute the expected distance between projected points and then bound the probability that the actual distance deviates from this expected value using Markov's inequality
3. Conclude that such a map must exist and also, that we can find such a map in randomized polynomial time simply by repeated sampling

Source: https://cseweb.ucsd.edu/~dasgupta/papers/jl.pdf

Shortcomings of PCA



- Principal components are orthonormal
- Principal components are linear combinations of the features
- Principal components are expensive to compute

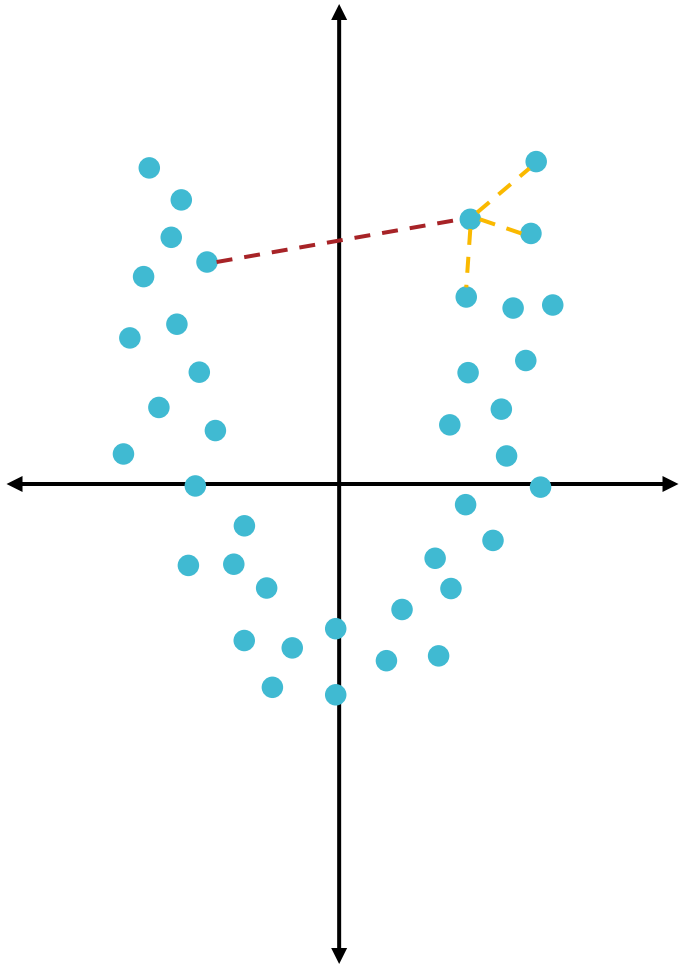
Nonlinear Dimensionality Reduction

- Autoencoders (1987)
- Kernel PCA (1999)
- Locally linear embedding (2000)
- Isomap (2000)
- Laplacian Eigenmaps (2003)
- t-SNE (2008)
- *… many others*

Nonlinear Dimensionality Reduction

- Autoencoders (1987)
- Kernel PCA (1999)
- Locally linear embedding (2000)
- Isomap (2000)
- Laplacian Eigenmaps (2003)
- **t-SNE (2008)**
- *… many others*

Shortcomings of PCA

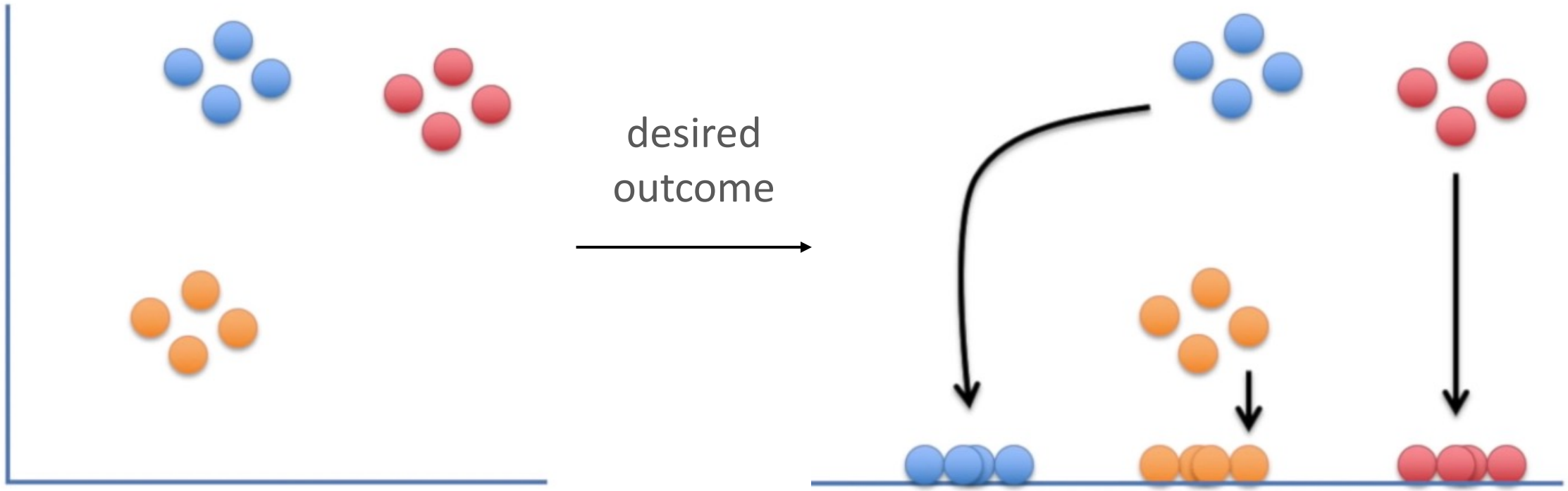


- Insight: PCA assumes that all distances between pairs of points are equally important
- Idea: Focus primarily on preserving “distances” between points in a small, local neighborhood!

t-distributed Stochastic Neighbor Embedding

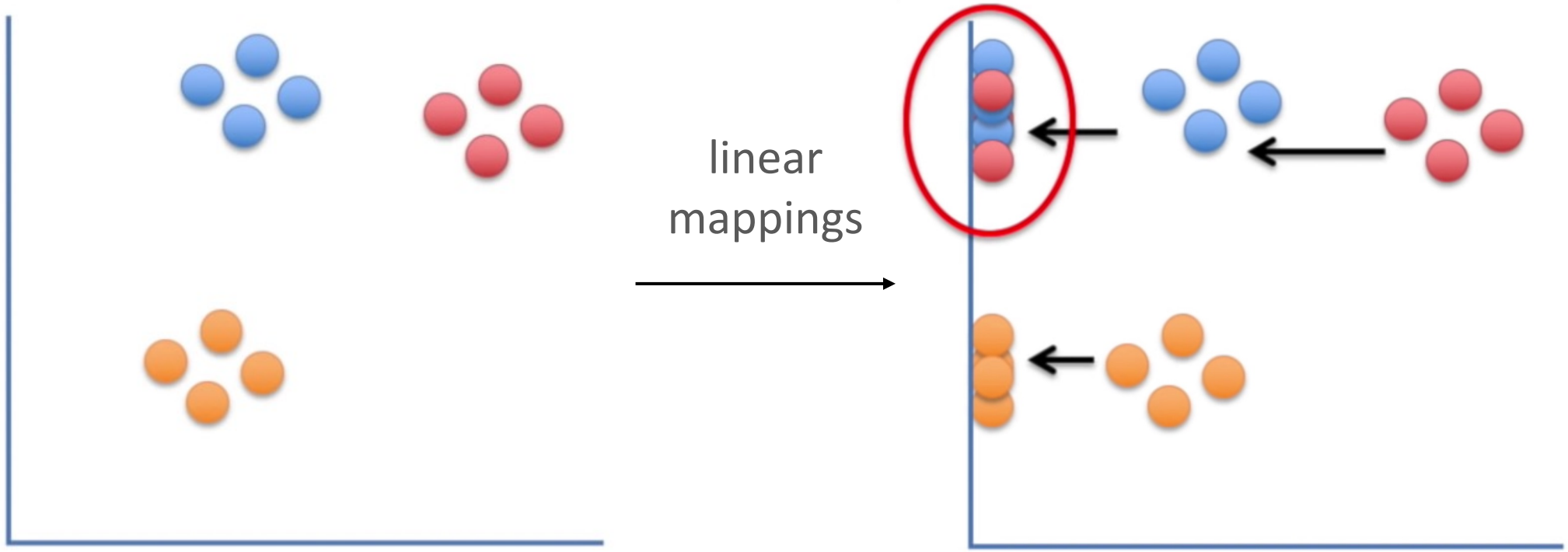
- Main idea: use the probability that two points are similar as the quantity to be preserved and *cleverly choose the distributions that we use*

1. Compute the probability that two points are similar in the original space
2. Randomly sample a projection for each point
3. Compute the probability that two points are similar in the projected space
4. Move points around in the projected space so that these two probabilities are “close” for all pairs of points



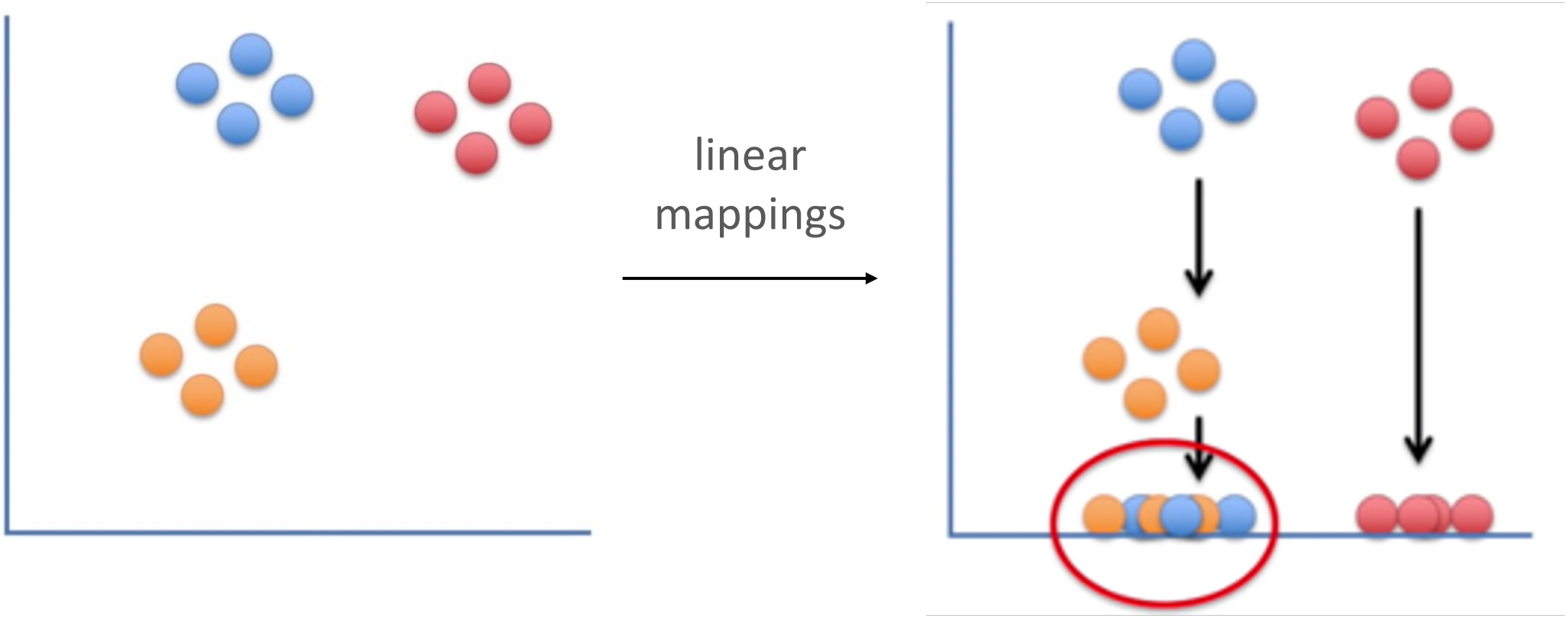
t-SNE: Example

Source: https://www.youtube.com/watch?v=NEaUSP4YerM



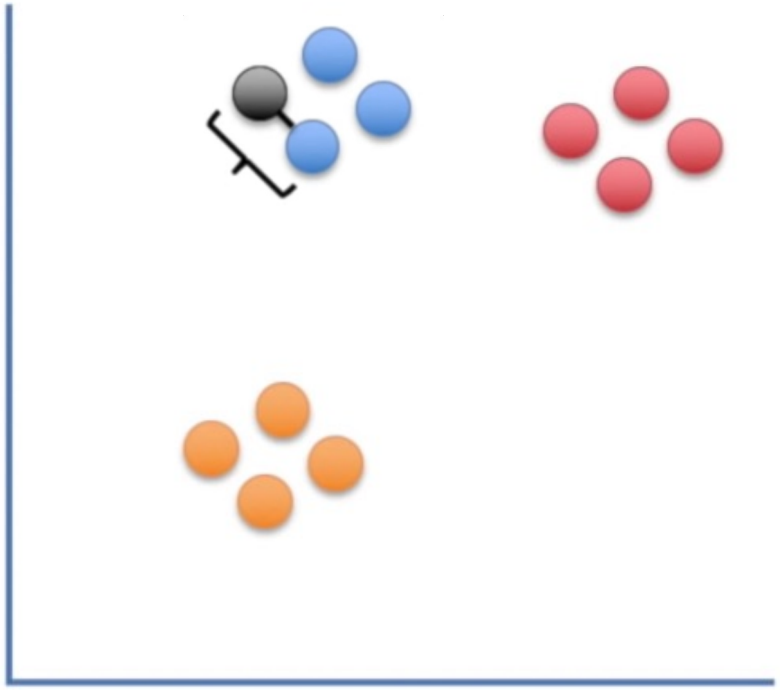
t-SNE: Example

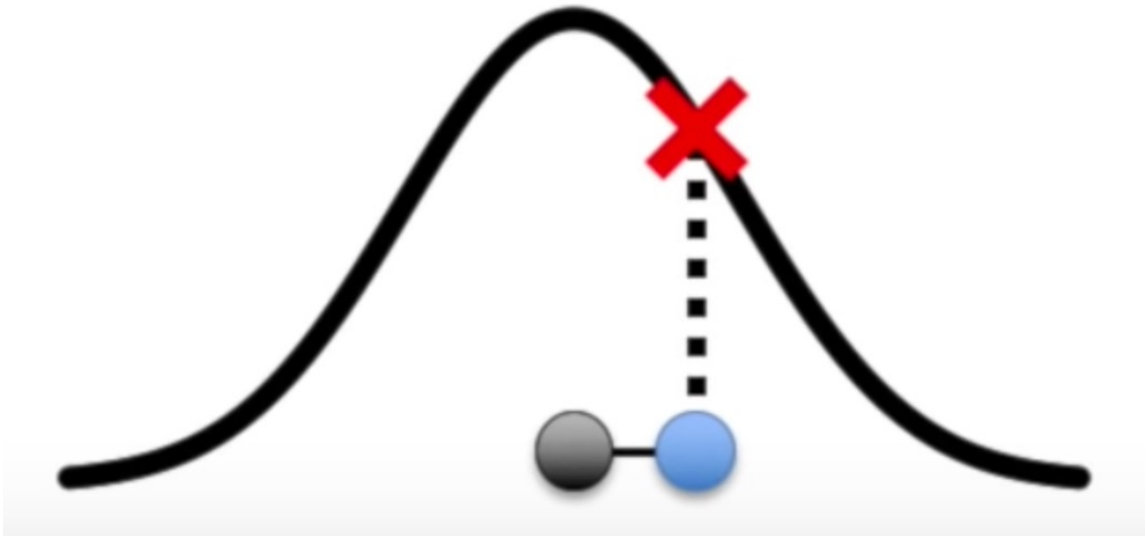
Source: https://www.youtube.com/watch?v=NEaUSP4YerM



t-SNE: Example

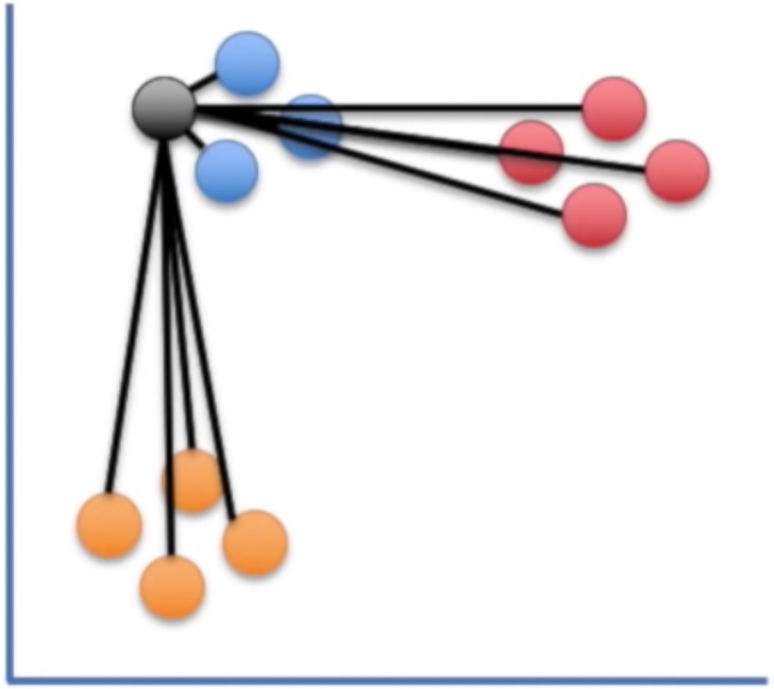
Source: https://www.youtube.com/watch?v=NEaUSP4YerM

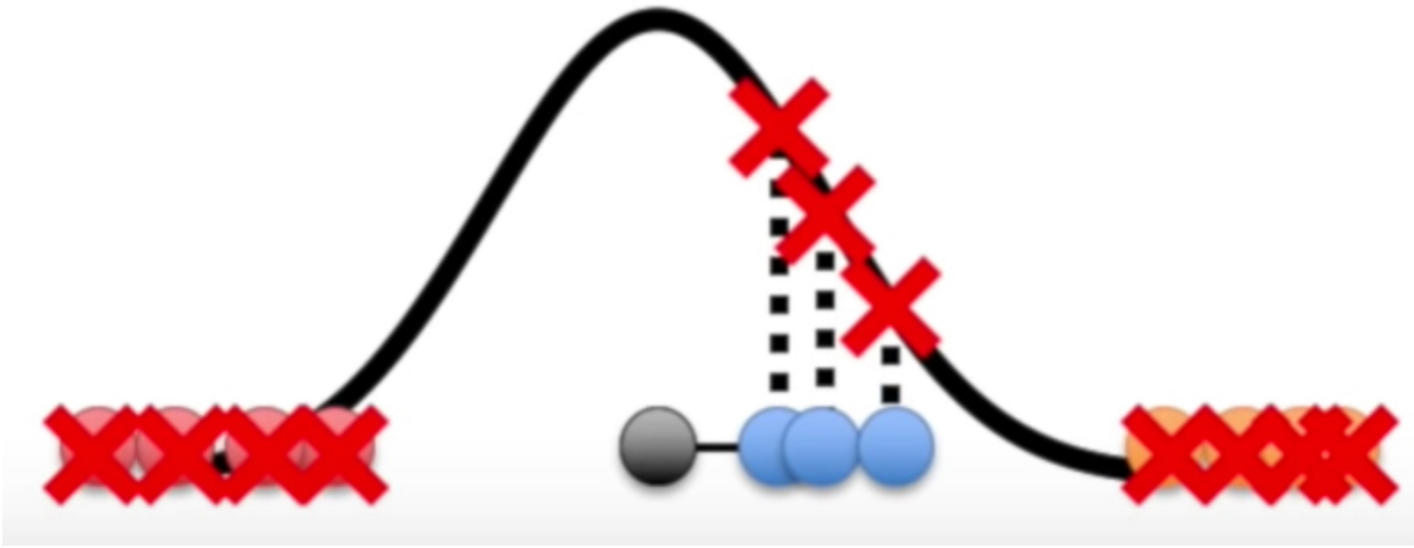




t-SNE: Example

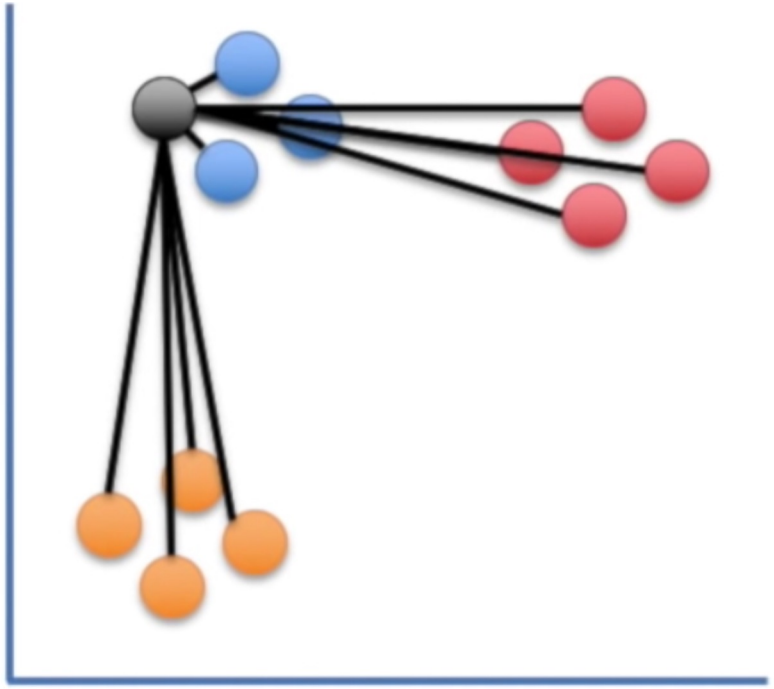
Source: https://www.youtube.com/watch?v=NEaUSP4YerM



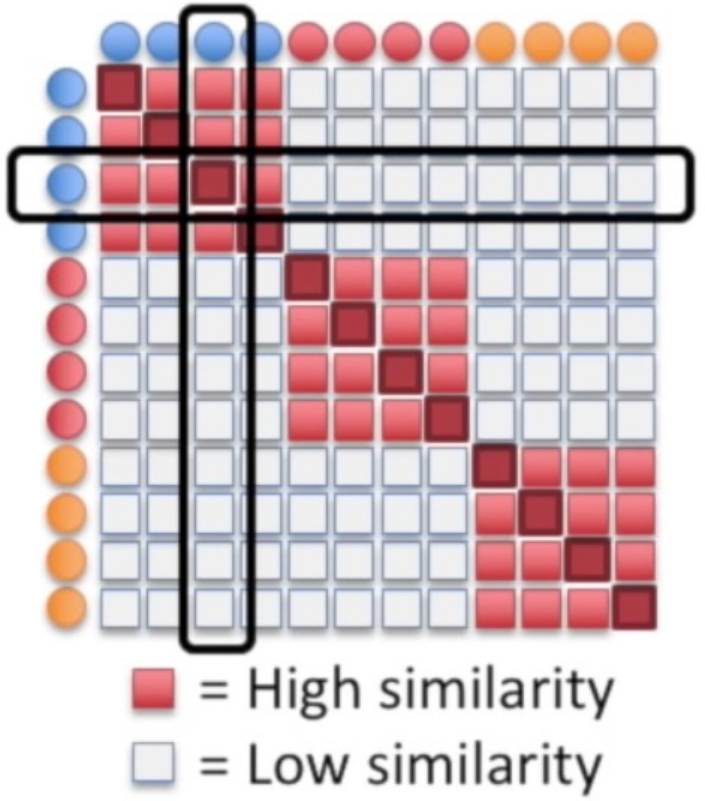


t-SNE: Example

Source: https://www.youtube.com/watch?v=NEaUSP4YerM

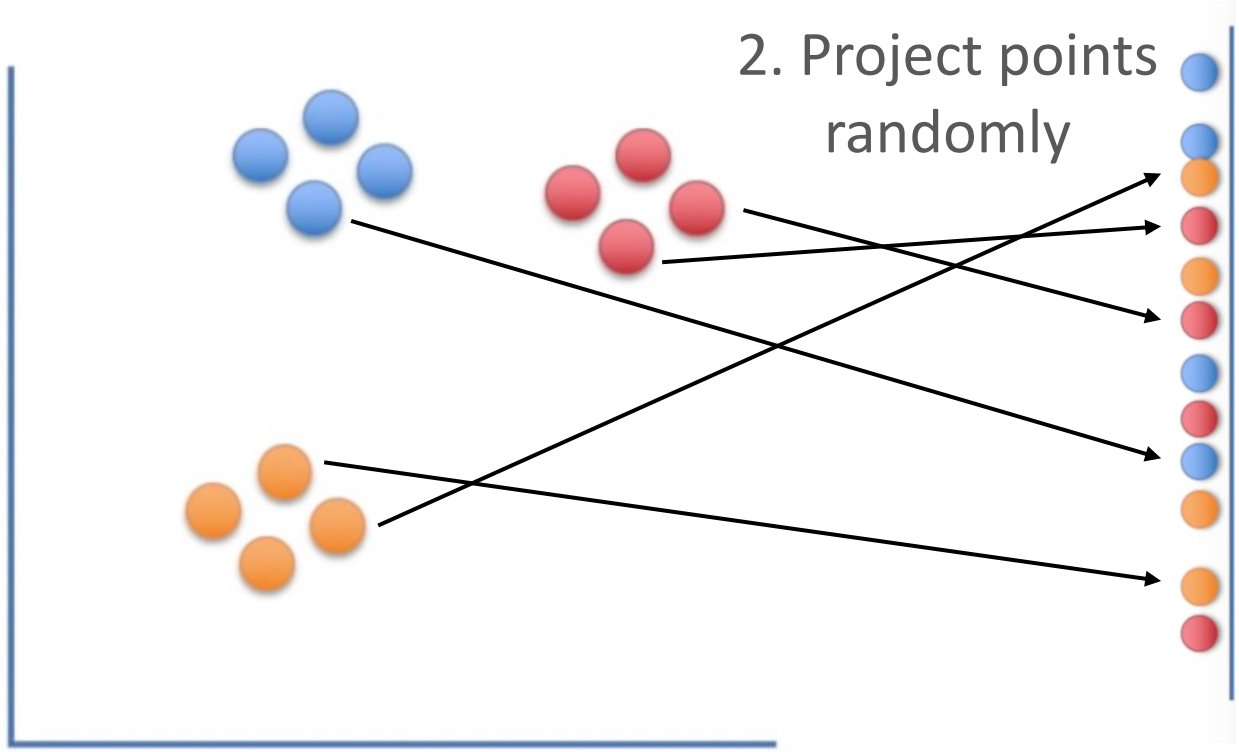


1. Compute probability of similarity in original space



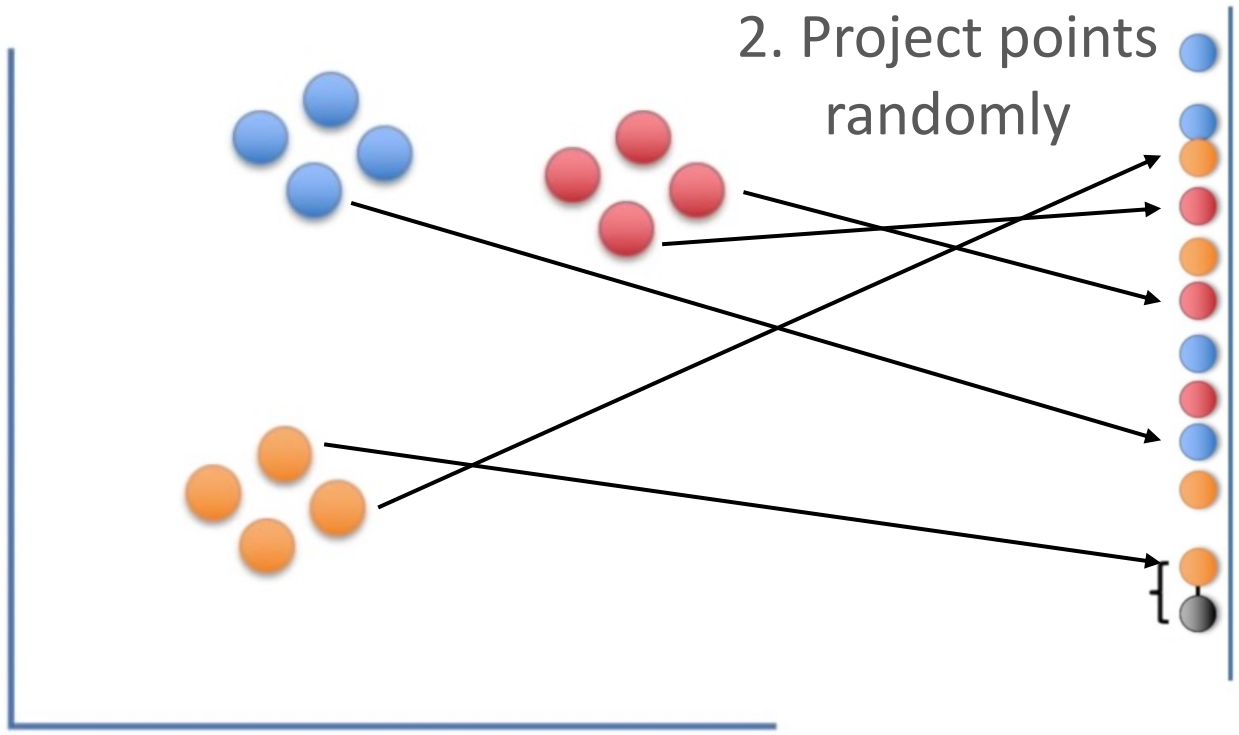
t-SNE: Example

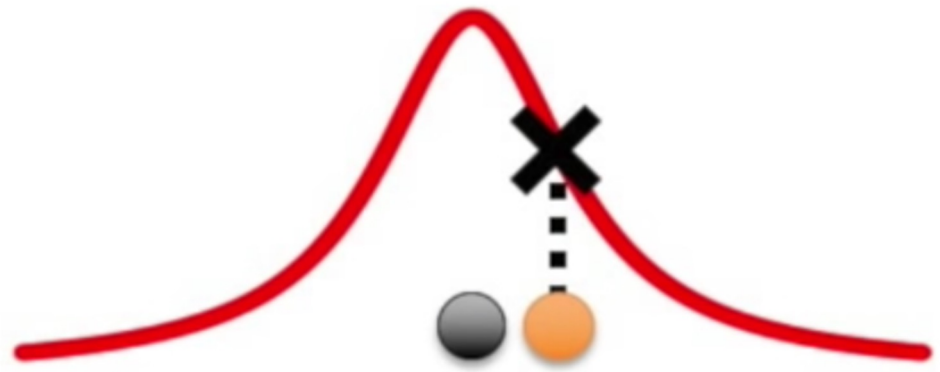
Source: https://www.youtube.com/watch?v=NEaUSP4YerM



t-SNE: Example

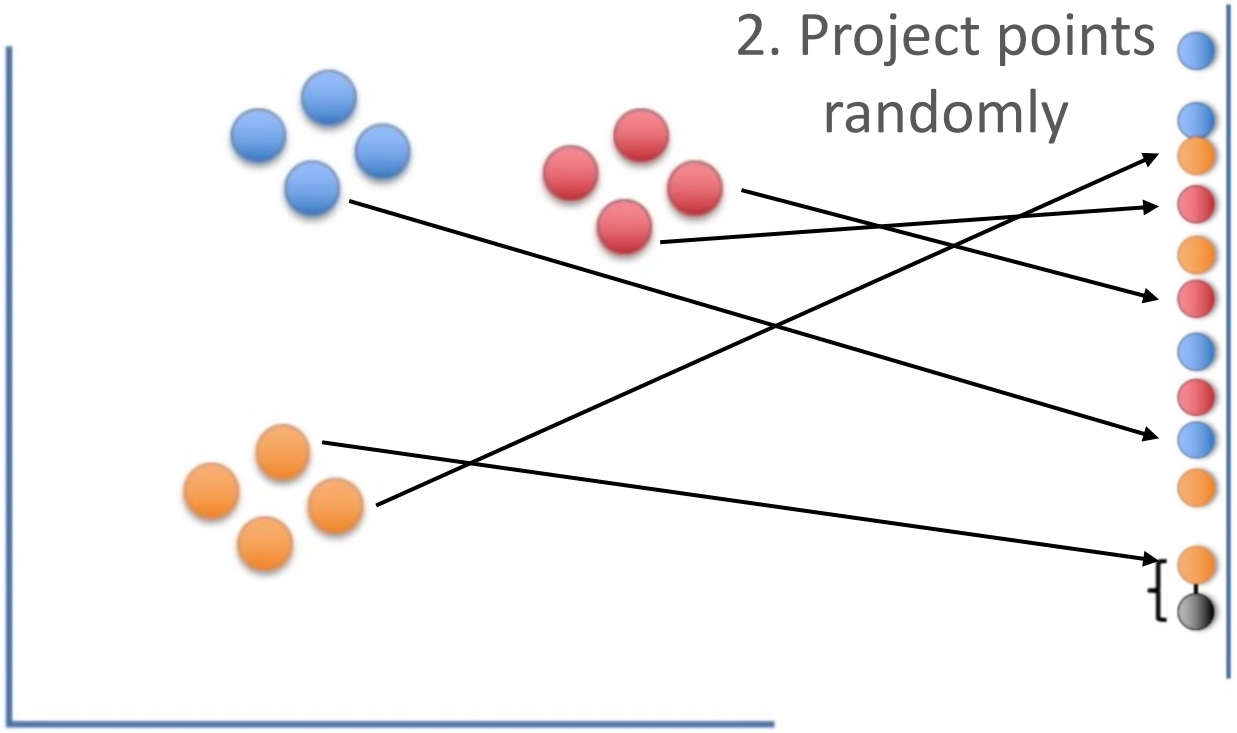
Source: https://www.youtube.com/watch?v=NEaUSP4YerM

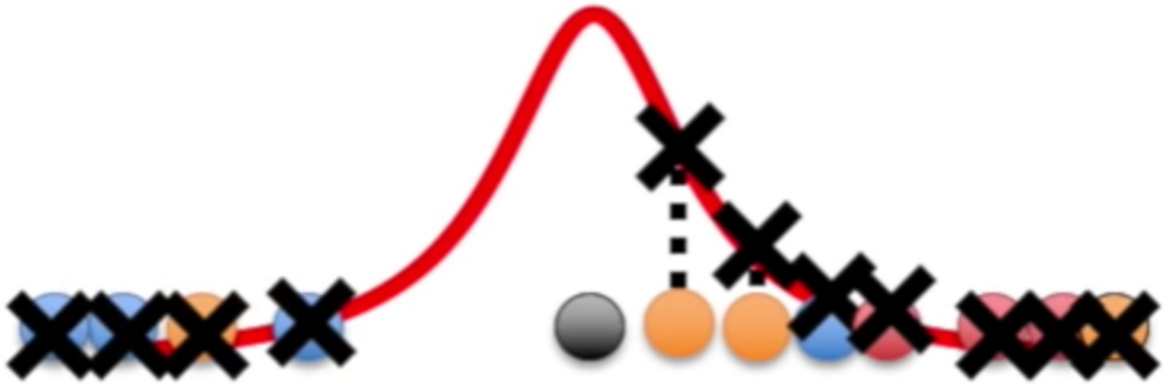




t-SNE: Example

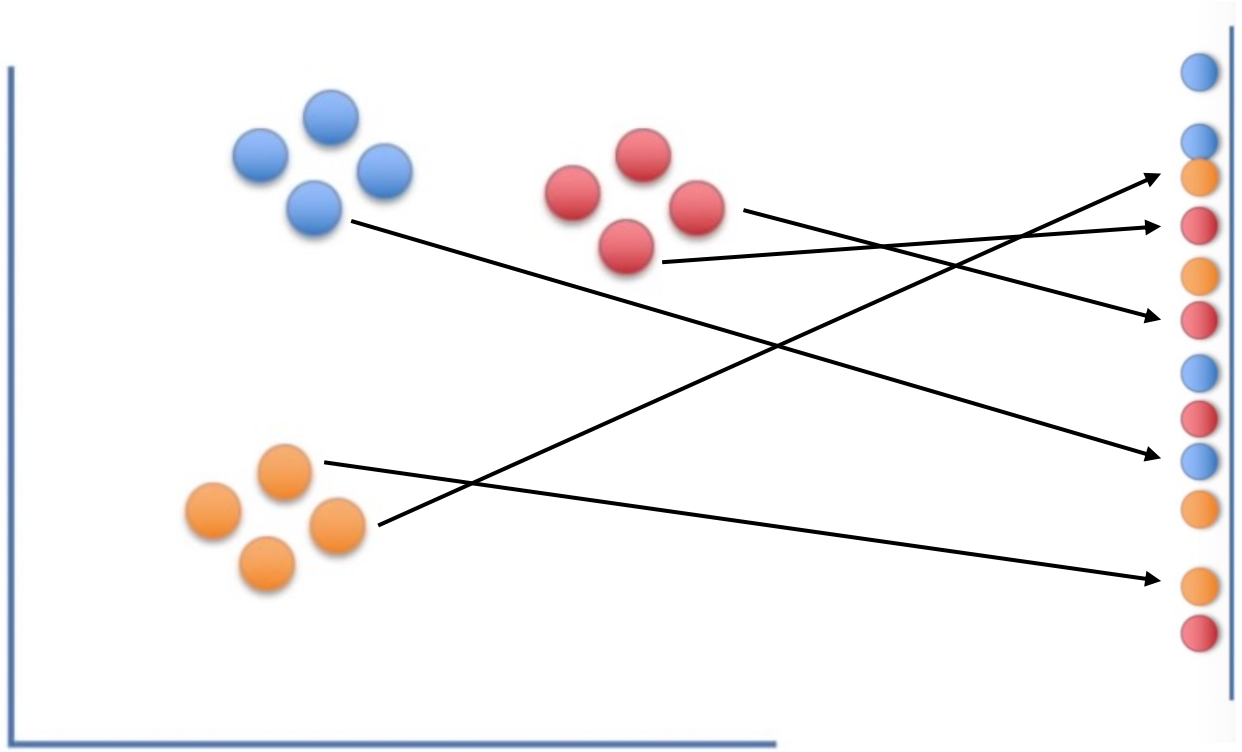
Source: https://www.youtube.com/watch?v=NEaUSP4YerM



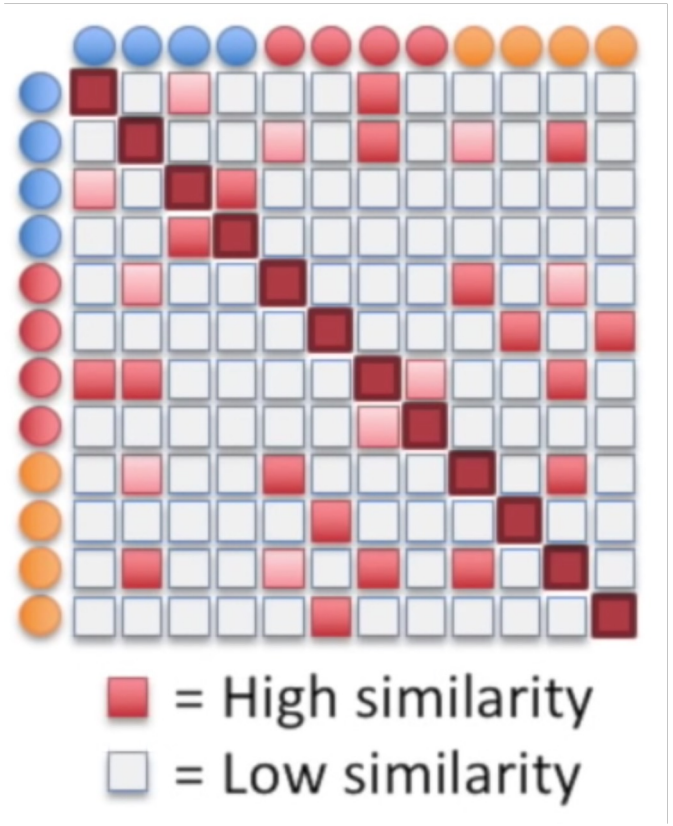


t-SNE: Example

Source: https://www.youtube.com/watch?v=NEaUSP4YerM

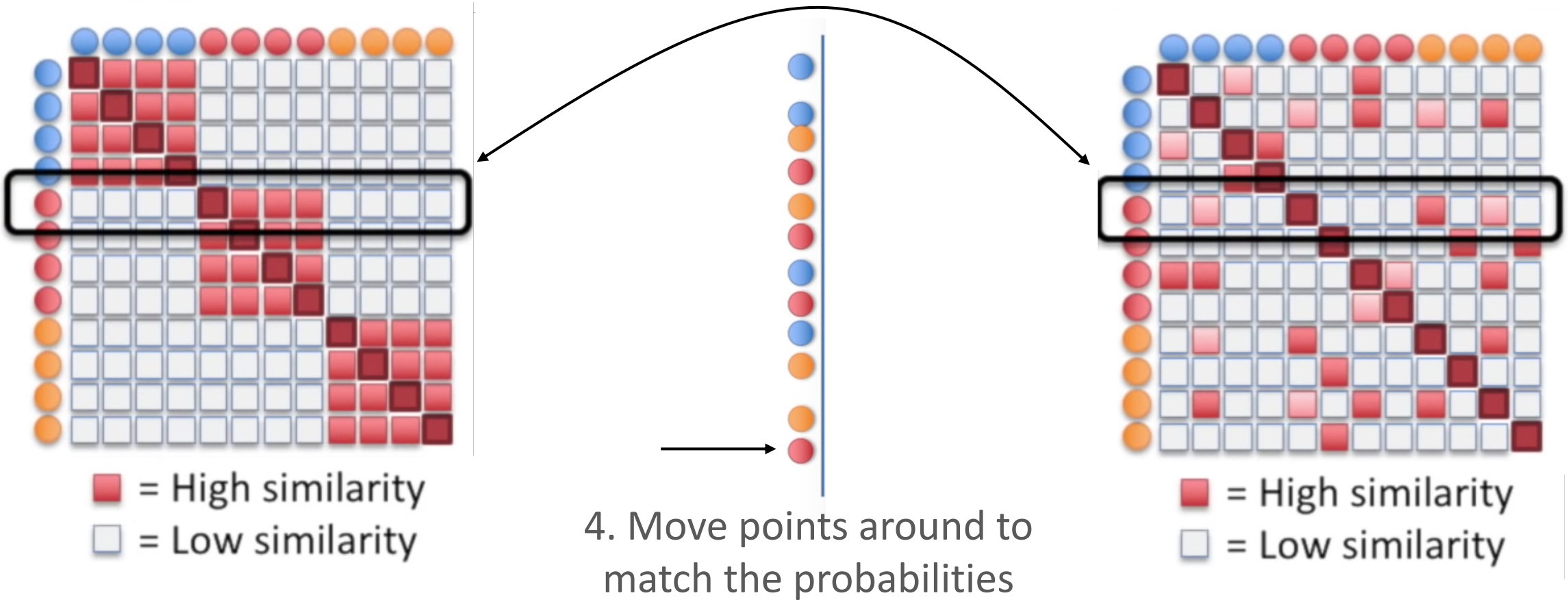


3. Compute probability of similarity in new space



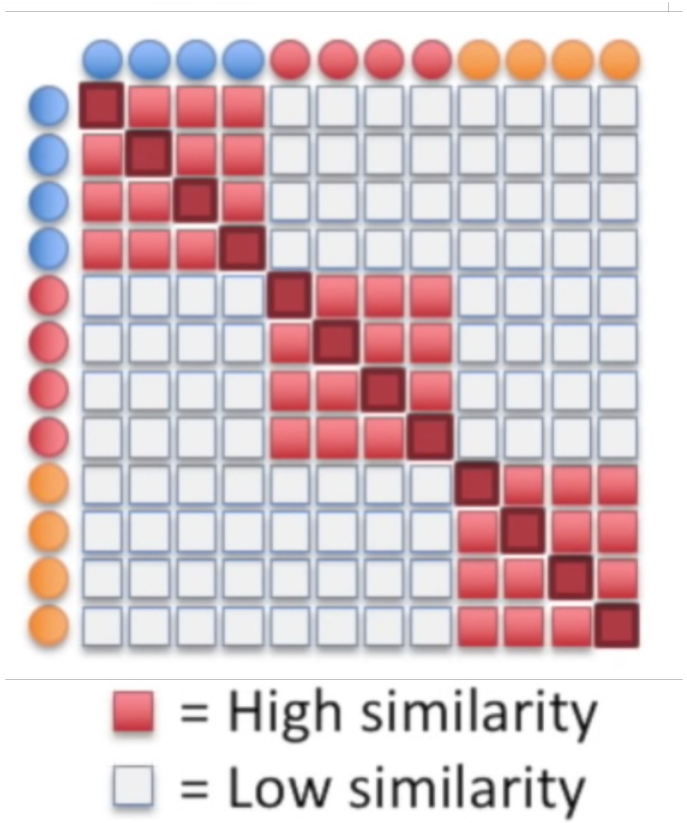
t-SNE: Example

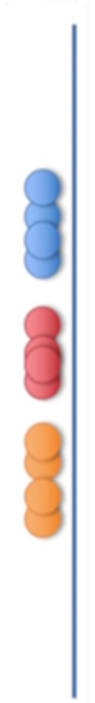
Source: https://www.youtube.com/watch?v=NEaUSP4YerM



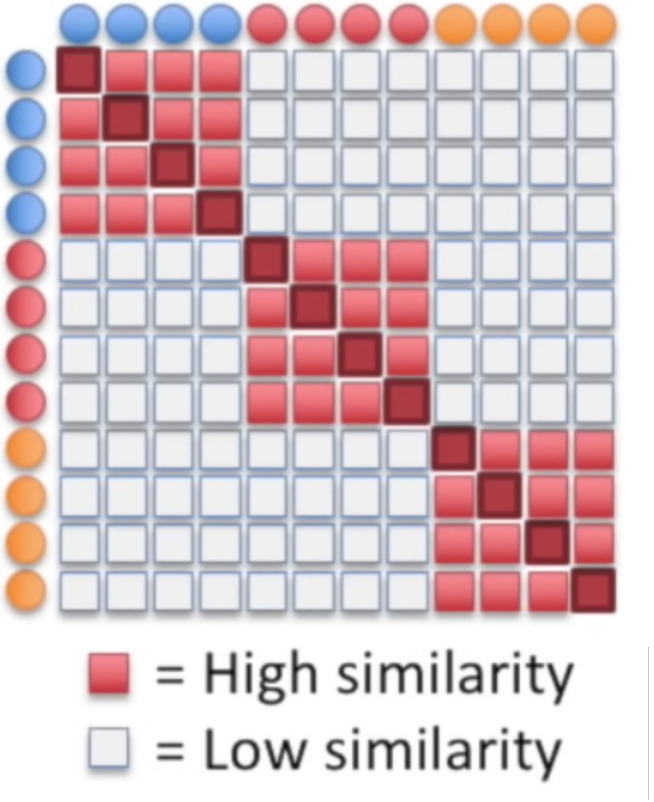
t-SNE: Example

Source: https://www.youtube.com/watch?v=NEaUSP4YerM





4. Move points around to match the probabilities



t-SNE: Example

Source: https://www.youtube.com/watch?v=NEaUSP4YerM

t-distributed Stochastic Neighbor Embedding

- Main idea: use the probability that two points are similar as the quantity to be preserved and *cleverly choose the distributions that we use*

1. Compute the probability that two points are similar in the original space
2. Randomly sample a projection for each point
3. Compute the probability that two points are similar in the projected space
4. Move points around in the projected space so that these two probabilities are “close” for all pairs of points

t-distributed Stochastic Neighbor Embedding

- Main idea: use the probability that two points are similar as the quantity to be preserved and *cleverly choose the distributions that we use*

1. Compute the probability that two points are similar in the original space

Let $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ be two points in the original space

$$p_{j|i} = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(\|\boldsymbol{x}_i - \boldsymbol{x}_k\|_2^2 / 2\sigma_i^2)}$$

$$p_{i|i} := 0$$

model hyperparameters

Define a new distribution as $p_{ij} = (p_{j|i} + p_{i|j})/(2n)$

$$\left(\sum_{i,j} p_{ij} = 1 \right)$$

t-distributed Stochastic Neighbor Embedding

- Main idea: use the probability that two points are similar as the quantity to be preserved and *cleverly choose the distributions that we use*

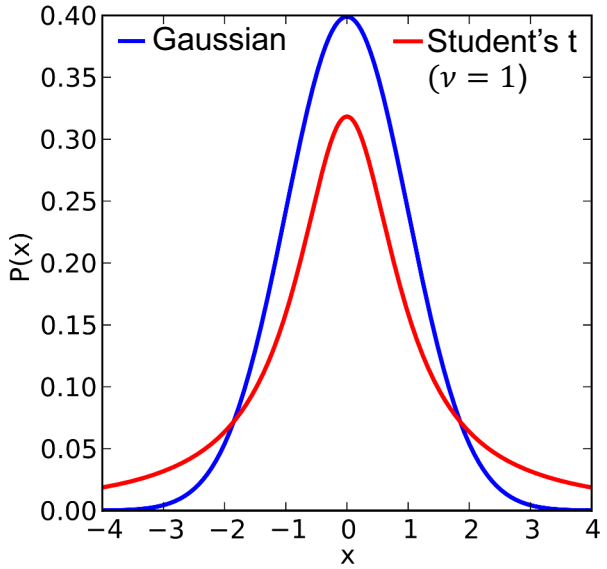
3. Compute the probability that two points are similar in the projected space

Let $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ be two points in the projected space

$$q_{ij} = \frac{(1 + \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2)^{-1}}{\sum_{k \neq i}(1 + \|\boldsymbol{z}_i - \boldsymbol{z}_k\|_2^2)^{-1}}$$

$$q_{ii} \coloneqq 0$$

Encourages similar points to clump and dissimilar points to spread out



Source: https://en.wikipedia.org/wiki/Student%27s_t-distribution#/media/File:T_distribution_1df_enhanced.svg

t-distributed Stochastic Neighbor Embedding

- Main idea: use the probability that two points are similar as the quantity to be preserved and *cleverly choose the distributions that we use*

4. Move points around in the projected space so that these two probabilities are “close” for all pairs of points

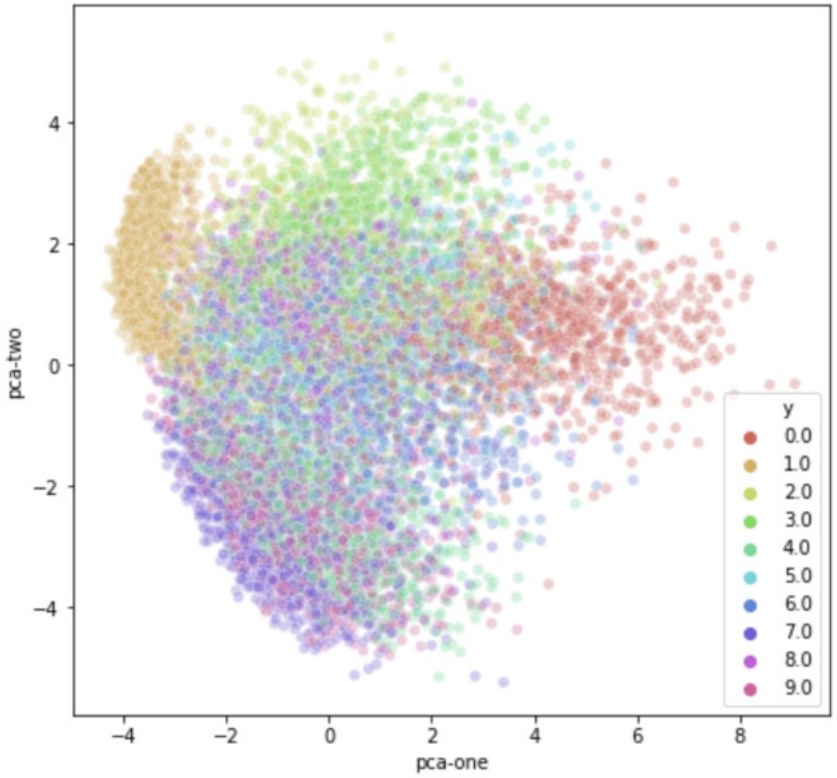
Use the KL-divergence between the two probability distributions

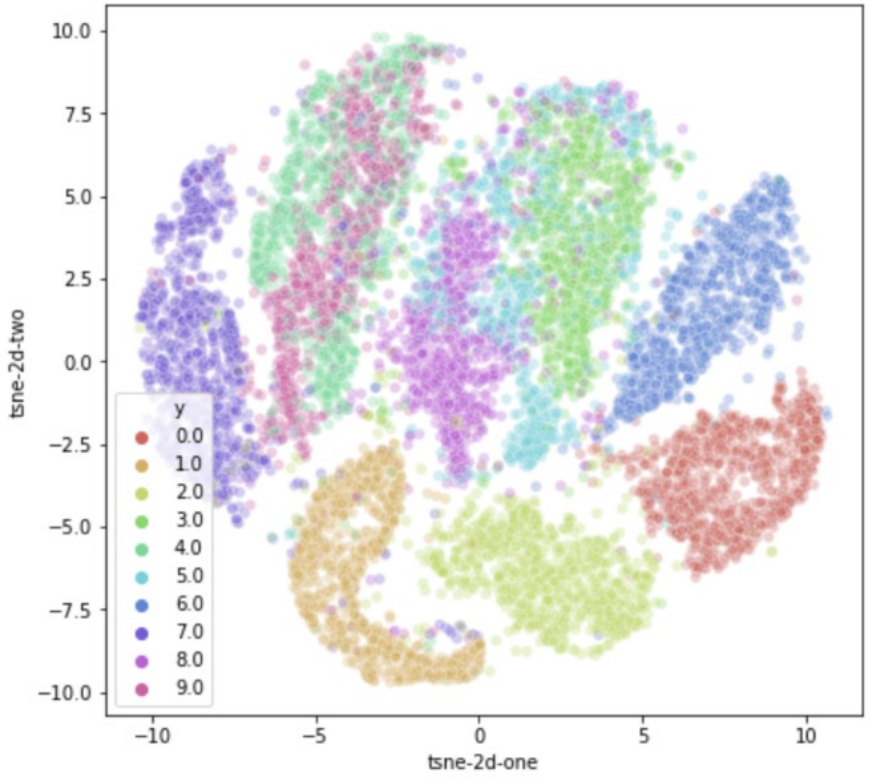
$$C = KL(P \parallel Q) = \sum_i \sum_{j \neq i} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

$$\frac{\partial C}{\partial \boldsymbol{z}_i} = 4 \sum_j (p_{ij} - q_{ij})(1 + \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2)^{-1}(\boldsymbol{z}_i - \boldsymbol{z}_j)$$

Optimize using (stochastic) gradient descent!

Source: https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf?fbcl





t-SNE vs. PCA

Source: https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

t-SNE vs. PCA

- PCA
  - ✓ Simple to use: no hyperparameters

    Computationally expensive: $O(k^3)$
    - ✓ Stay tuned for distributed PCA!

    Principal components are constrained to be *orthonormal* vectors (linear in the existing features)
- t-SNE
  - ✓ Can learn non-linear representations

    Computationally expensive: $O(n^2)$ per iteration

    Involves non-convex optimization

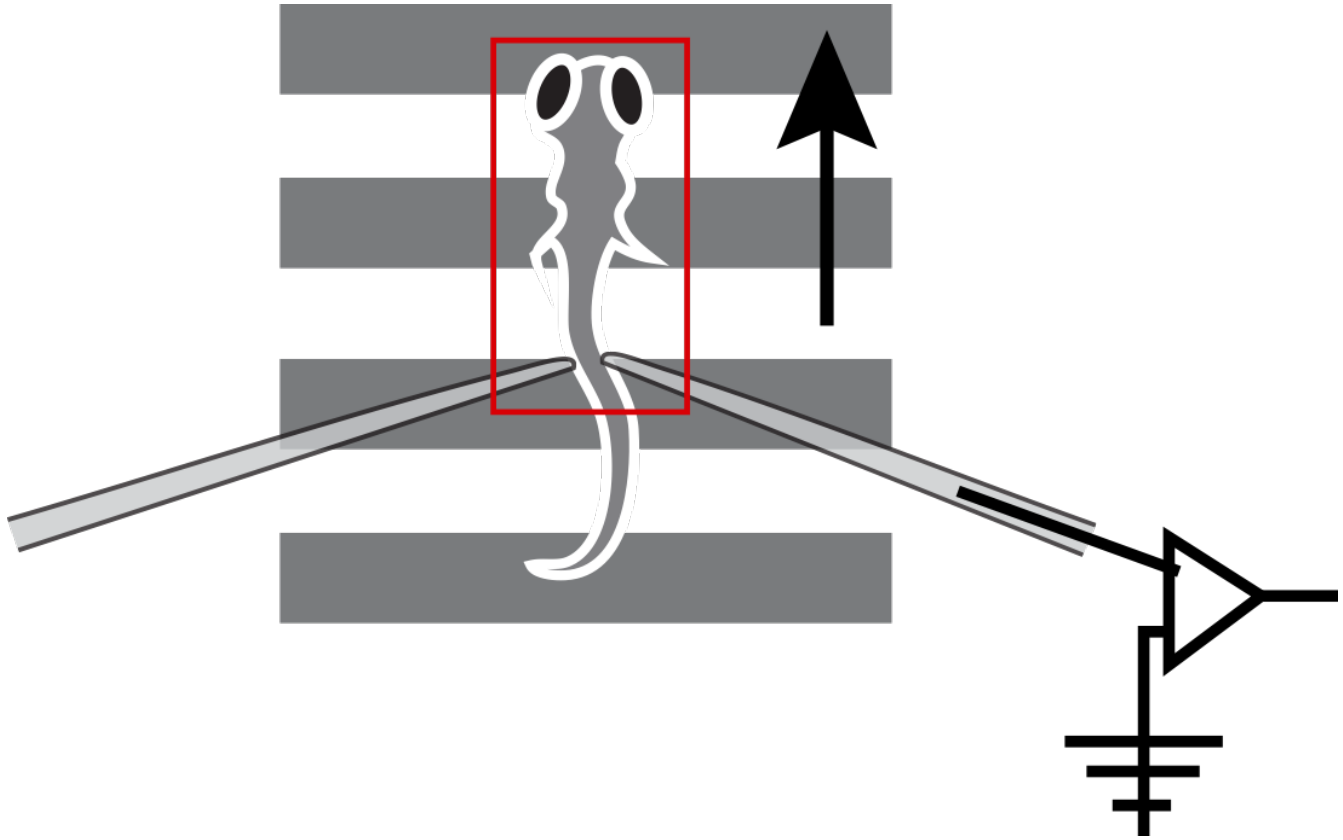
    Sensitive to hyperparameters (https://distill.pub/2016/misread-tsne/)

Key Takeaways

- Random sampling as an efficient alternative to PCA for finding linear projections
  - Theoretical guarantees regarding preserving relative distances via the Johnson-Lindenstrauss lemma
- t-SNE learns non-linear latent representations by preserving "distances" or matching probability distributions
  - Using the Student's t distribution encourages spread in the latent space
  - KL-divergence can be minimized using (stochastic) gradient descent

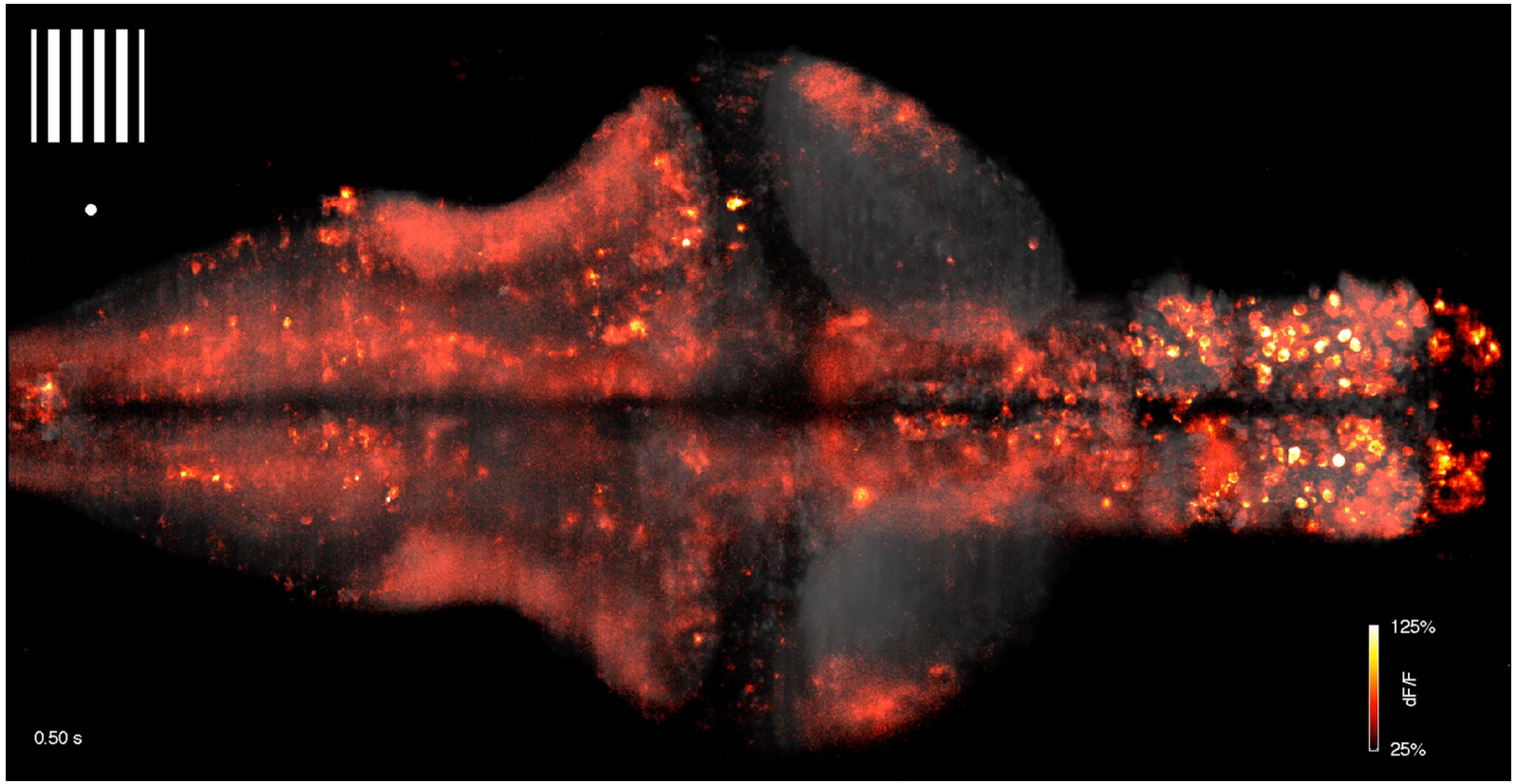
HW2 Preview

- Visualizing neural stimuli in larval zebrafish



HW2 Preview

- Visualizing neural stimuli in larval zebrafish

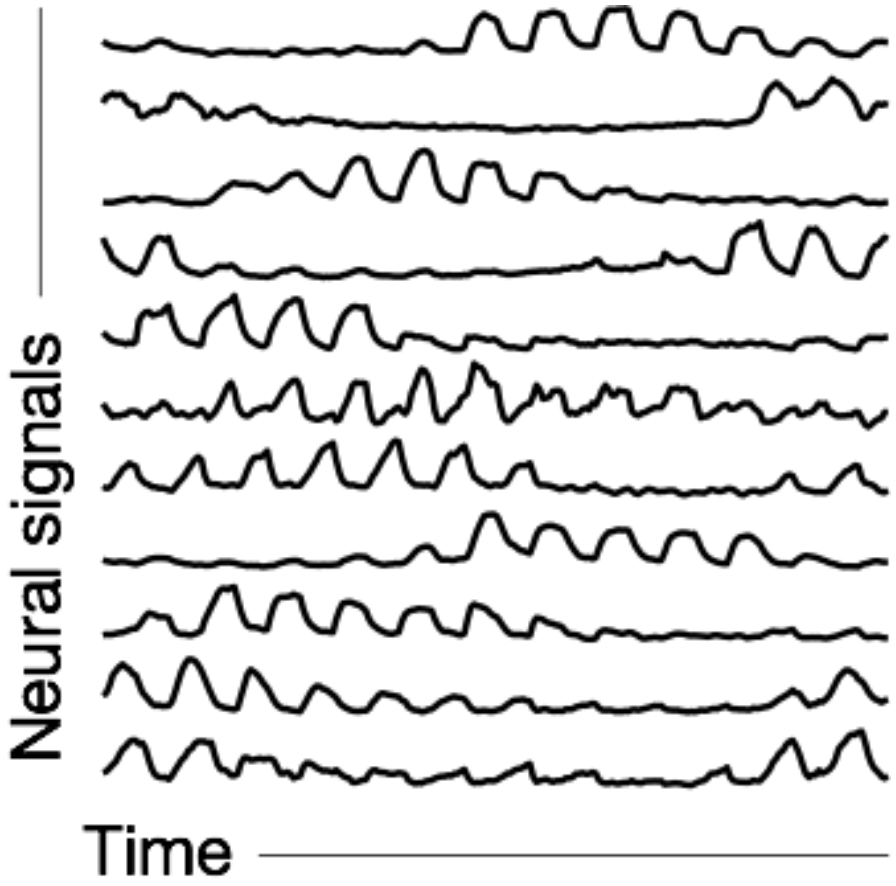


HW2 Preview

- Visualizing neural stimuli in larval zebrafish
- Mouse somatosensory cortex:
  - ~1,000 neurons
  - 0.1 TB/experiment
- Larval zebrafish brain:
  - ~100,000 neurons
  - 1 TB/experiment
- Mouse brain:
  - ~80,000,000 neurons
  - > 100 TB/experiment

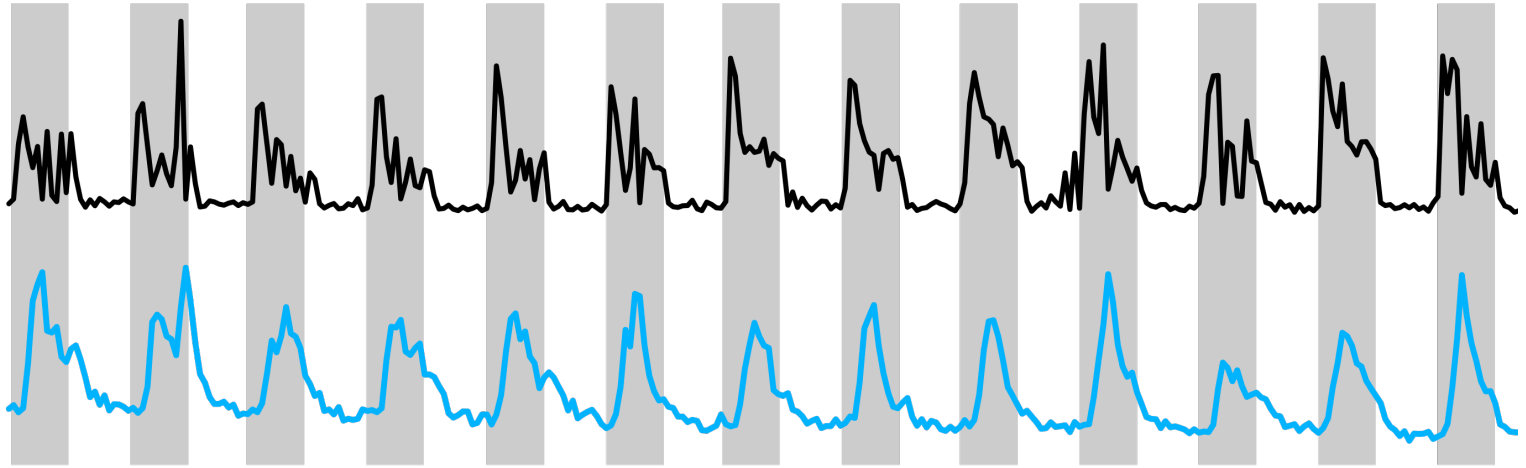
HW2 Preview

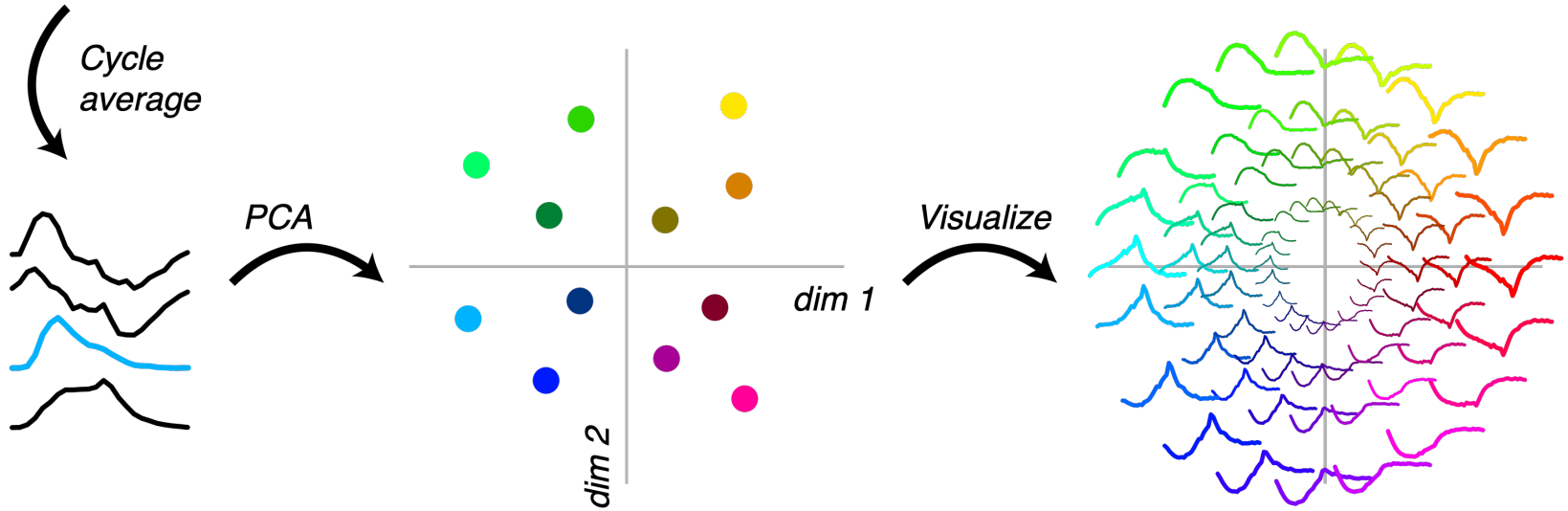
- Visualizing neural stimuli in larval zebrafish



HW2 Preview

- Visualizing neural stimuli in larval zebrafish





HW2 Preview

- Visualizing neural stimuli in larval zebrafish