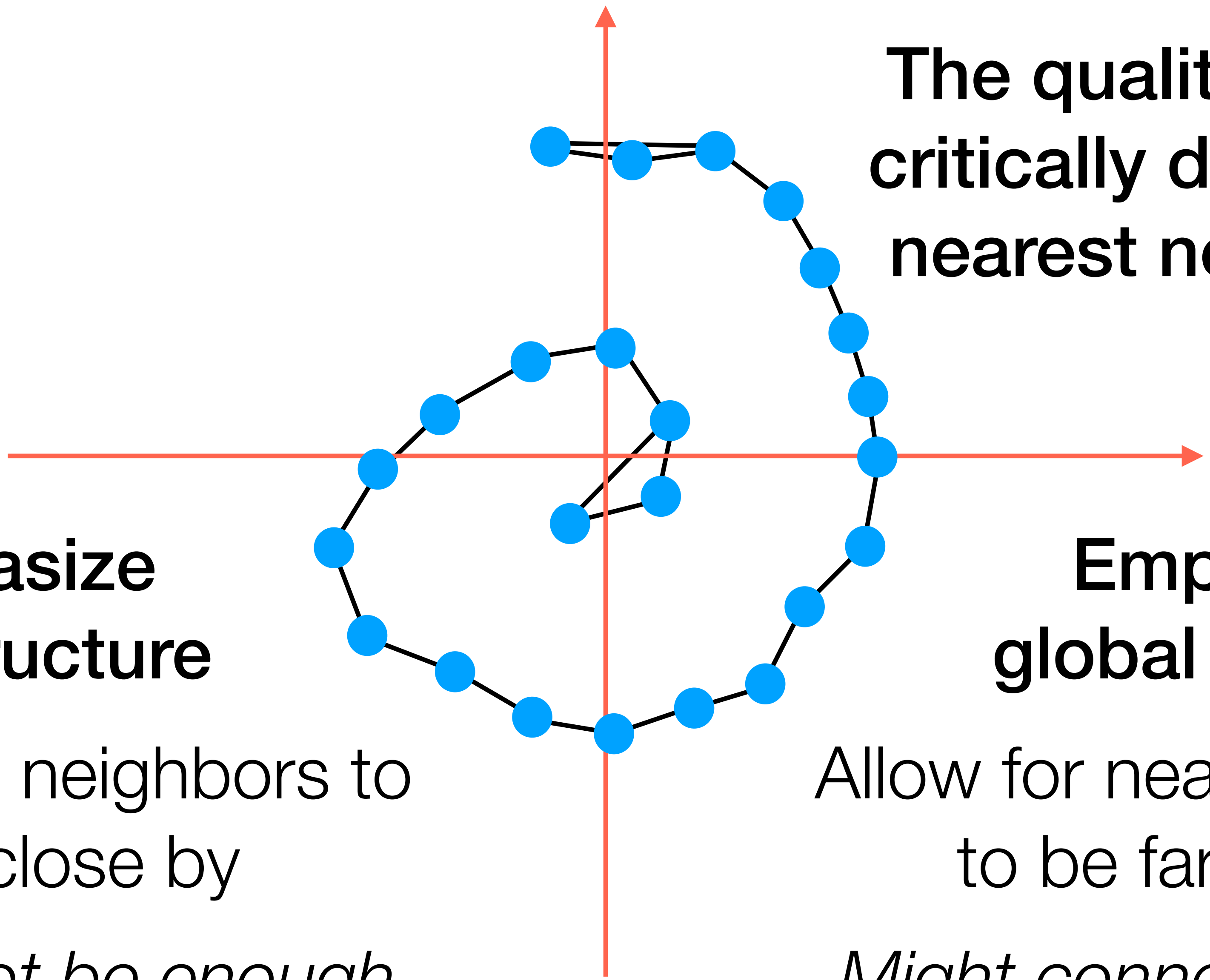
Carnegie Mellon University
Heinz College

Unstructured Data Analysis

Lecture 6: wrap up manifold learning (t-SNE), a first look at analyzing images, and an introduction to clustering phenomena

George Chen

(Flashback) 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

(Flashback) Isomap

Build k-NN graph, computed shortest distances

If k is set too large and we connect everything: Isomap just becomes MDS

Original high-dim. data

Distance table (for high-dim. points)

Make these two as close as possible

(Euclidean dist)

Low-dim. data

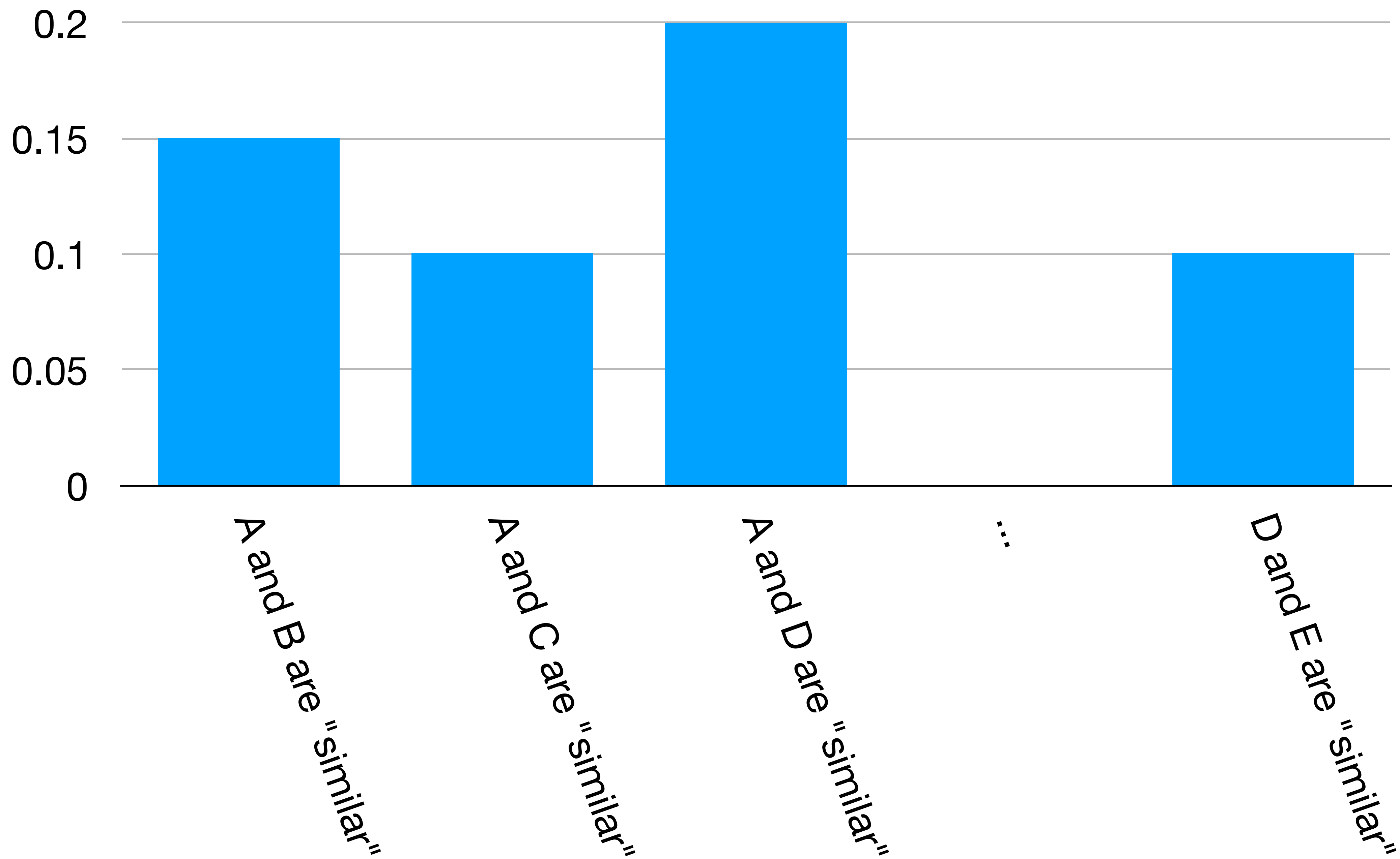
Distance table (for low-dim. points)

Compute Euclidean distances between all pairs of low-dimensional points

t-SNE
(t-distributed stochastic neighbor embedding)

t-SNE High-Level Idea #1

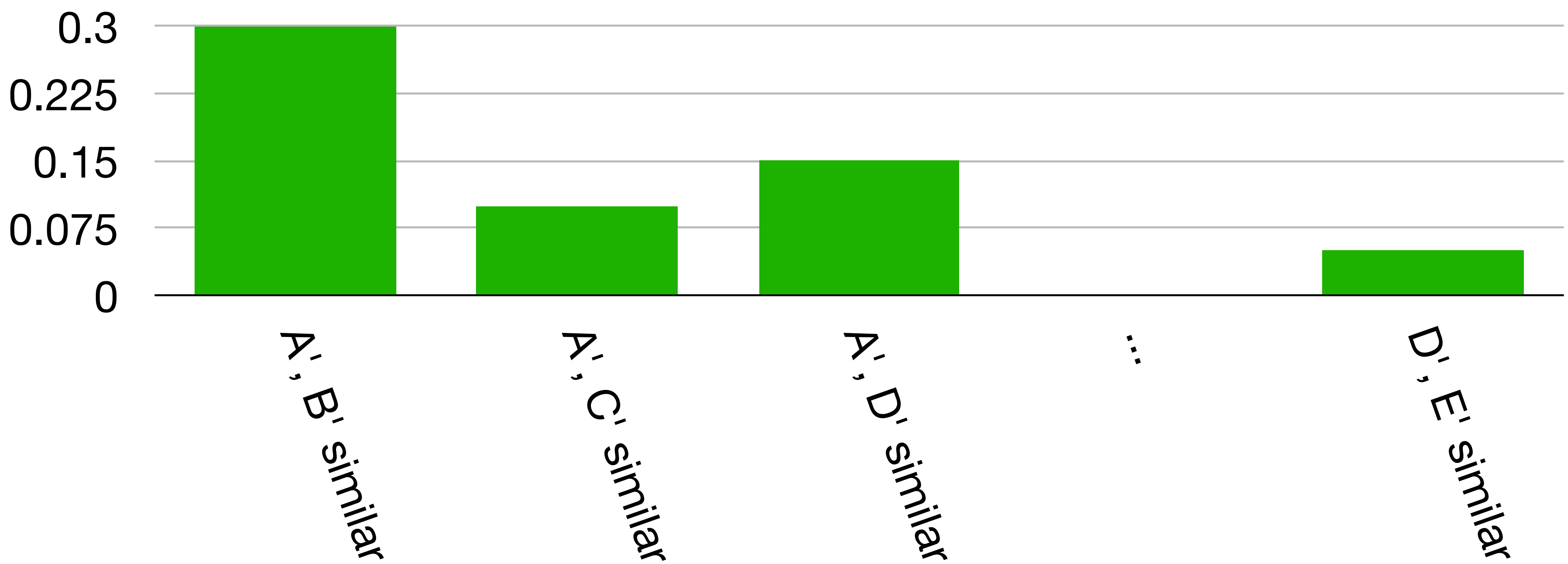
- Don't use deterministic definition of which points are neighbors
- Use probabilistic notation instead



t-SNE High-Level Idea #2

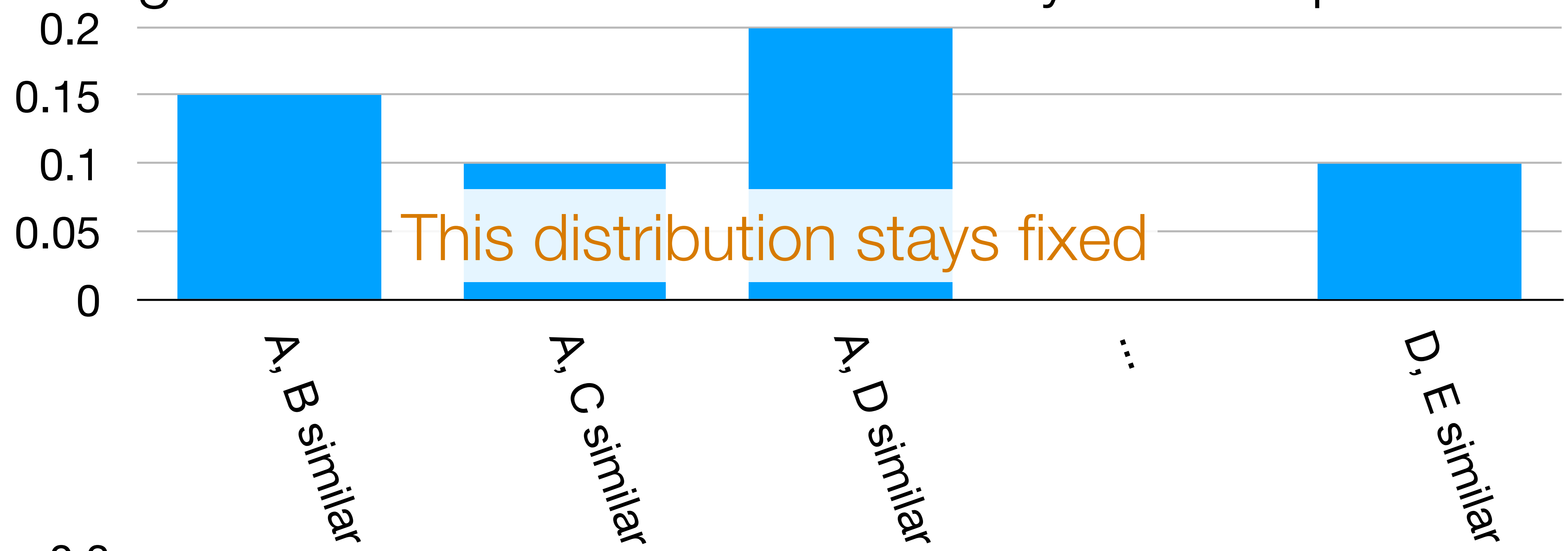
- In low-dim. space (e.g., 1D), suppose we just randomly assigned coordinates as a candidate for a low-dimensional representation for A, B, C, D, E (I'll denote them with primes):

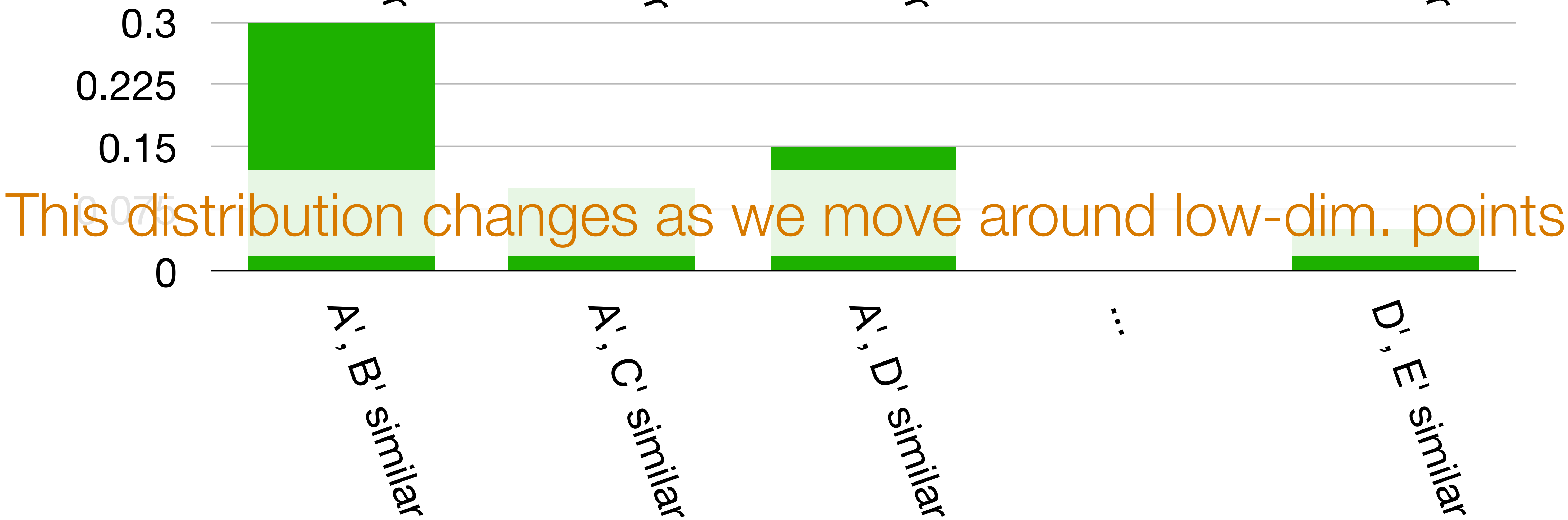
- With any such candidate choice, we can define a probability distribution for these <u>low-dimensional</u> points being similar



t-SNE High-Level Idea #3

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





t-SNE

Technical detail: creates probabilities
based on Gaussian distribution

Original high-dim. data

Probability table
(for high-dim. points)

Make these two as
close as possible

(Technical detail:
KL divergence)

Low-dim. data

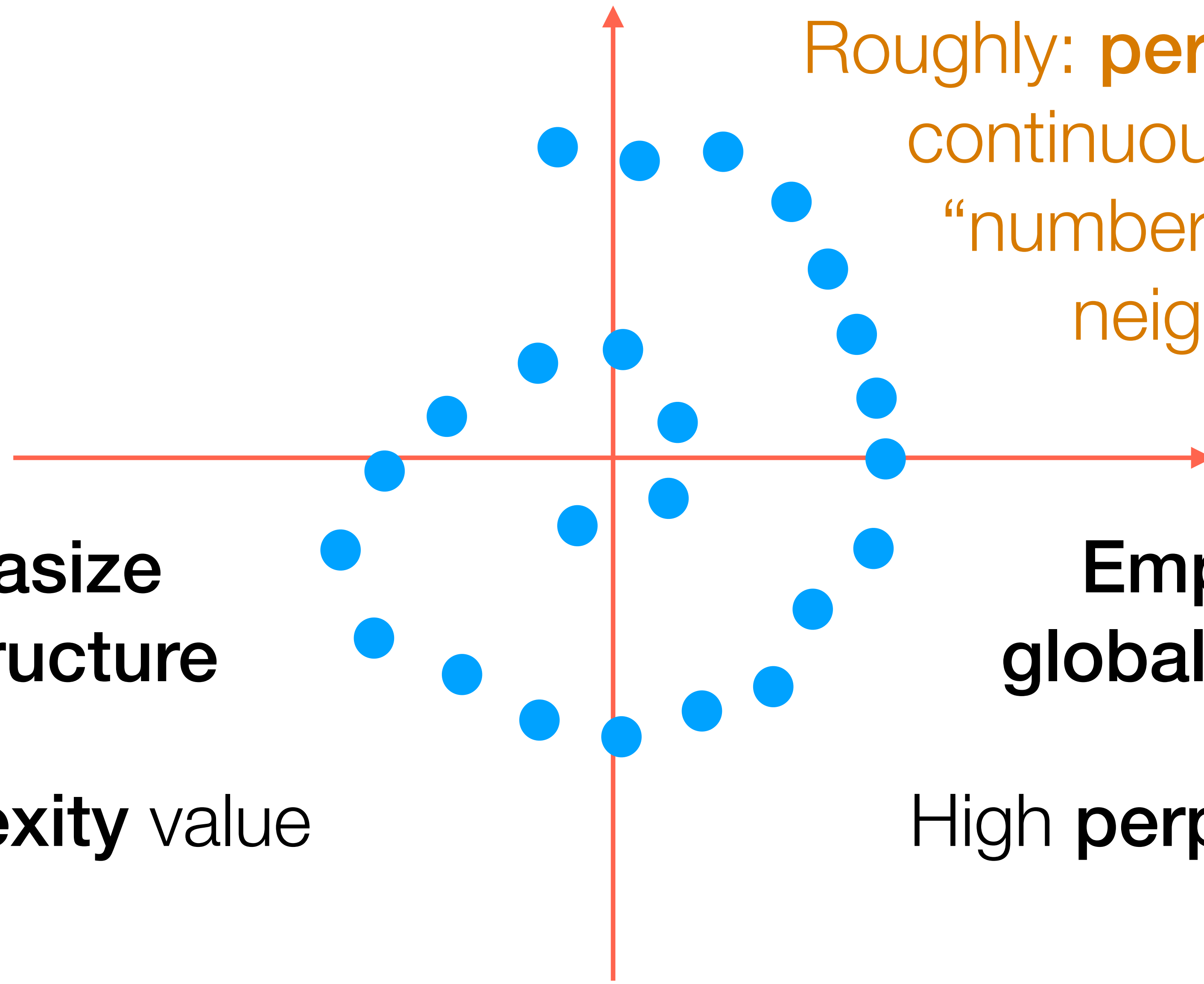
Probability table
(for low-dim. points)

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

Low **perplexity** value

Emphasize global structure

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

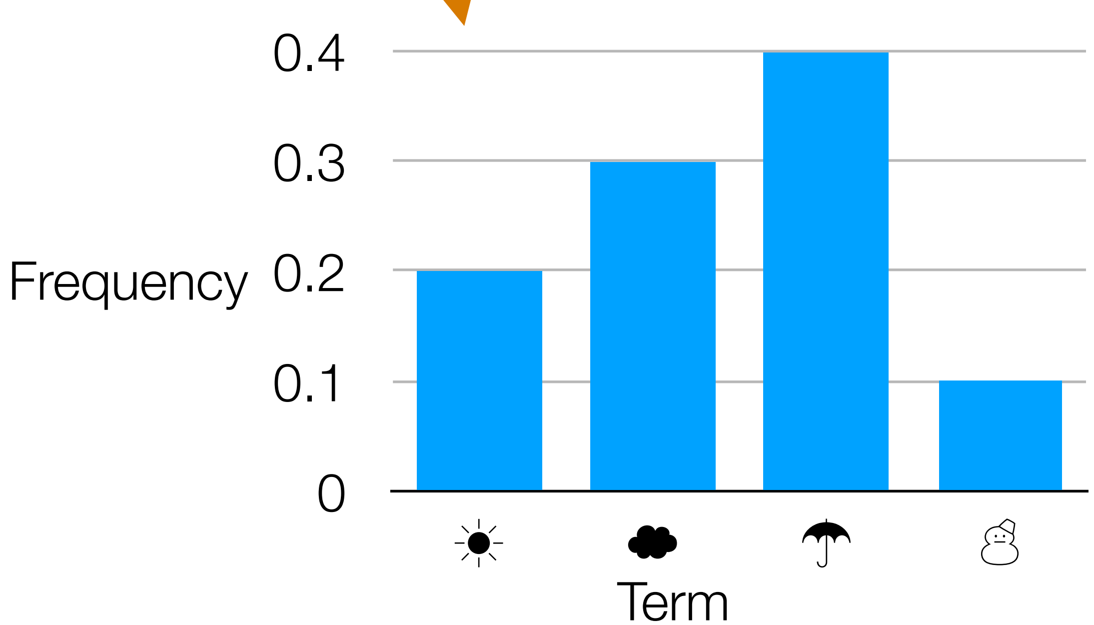
- There are *many* methods (I've posted a link on the course webpage to a scikit-learn example using ~10 methods)
- PCA is very well-understood; the new axes can be interpreted
- Nonlinear dimensionality reduction: new axes may not really be all that interpretable (you can scale axes, shift all points, 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": ☀☂☁☁☁☂☃☂☂☀



$$\begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.1 \end{bmatrix}$$

This is a point in 4-dimensional space, $\mathbb{R}^4$

dimensions = number of terms

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

Example: Representing an Image

Go row by row and look at pixel values

0: black
1: white

$$\begin{bmatrix} 0 & & \\ & & \\ & & \end{bmatrix}$$

Image source: *The Mandalorian*

Example: Representing an Image

0: black
1: white

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ \\ \\ \\ \end{bmatrix}$$

Go row by row and look at pixel values

Image source: *The Mandalorian*

Example: Representing an Image

0: black
1: white

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0.9 \\ \vdots \\ \\ \\ \end{bmatrix}$$

Go row by row and look at pixel values

Image source: *The Mandalorian*

Example: Representing an Image

0: black
1: white

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0.9 \\ \vdots \\ \\ \\ 0.1 \end{bmatrix}$$

Go row by row and look at pixel values

dimensions = image width × image height

Very high dimensional!

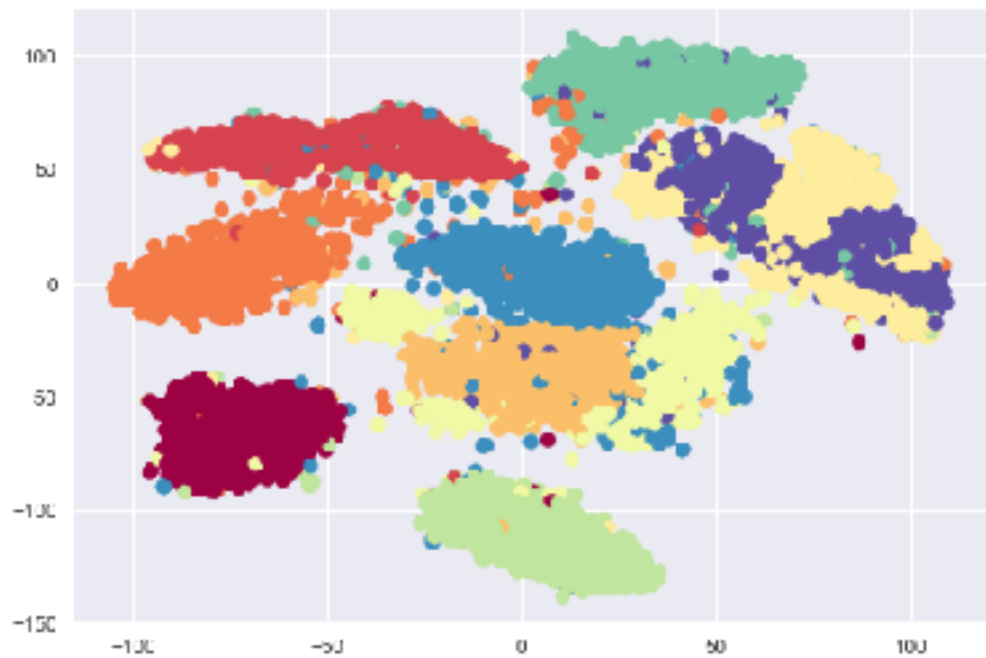
Image source: *The Mandalorian*

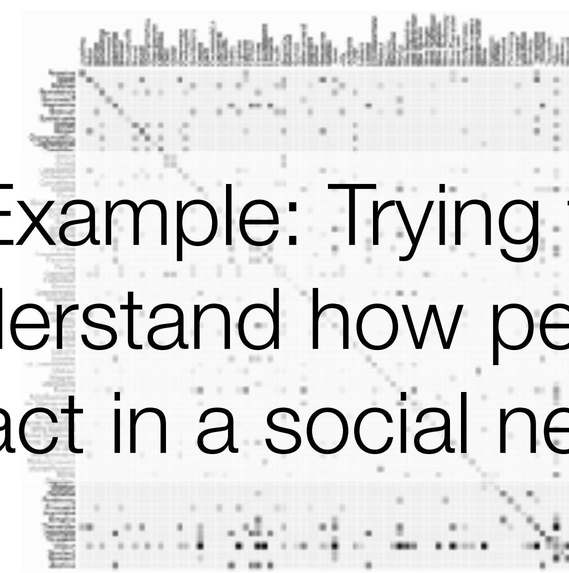
Dimensionality Reduction for Images

Demo

Visualization

is a way of debugging data analysis!





Example: Trying to understand how people interact in a social network

Important:
Handwritten digit demo is a **toy example** where we know which images correspond to digits 0, 1, …, 9

Many real UDA problems:
The data are **messy** and it's not obvious what the "correct" labels/answers look like, and "correct" is ambiguous!

Later on in the course (when we cover predictive analytics), we look at how to take advantage of knowing the true "correct" answers

Top right image source: https://bost.ocks.org/mike/miserables/

Let’s look at a *structured* dataset
(easier to explain clustering):
drug consumption data

Drug Consumption Data

Demo