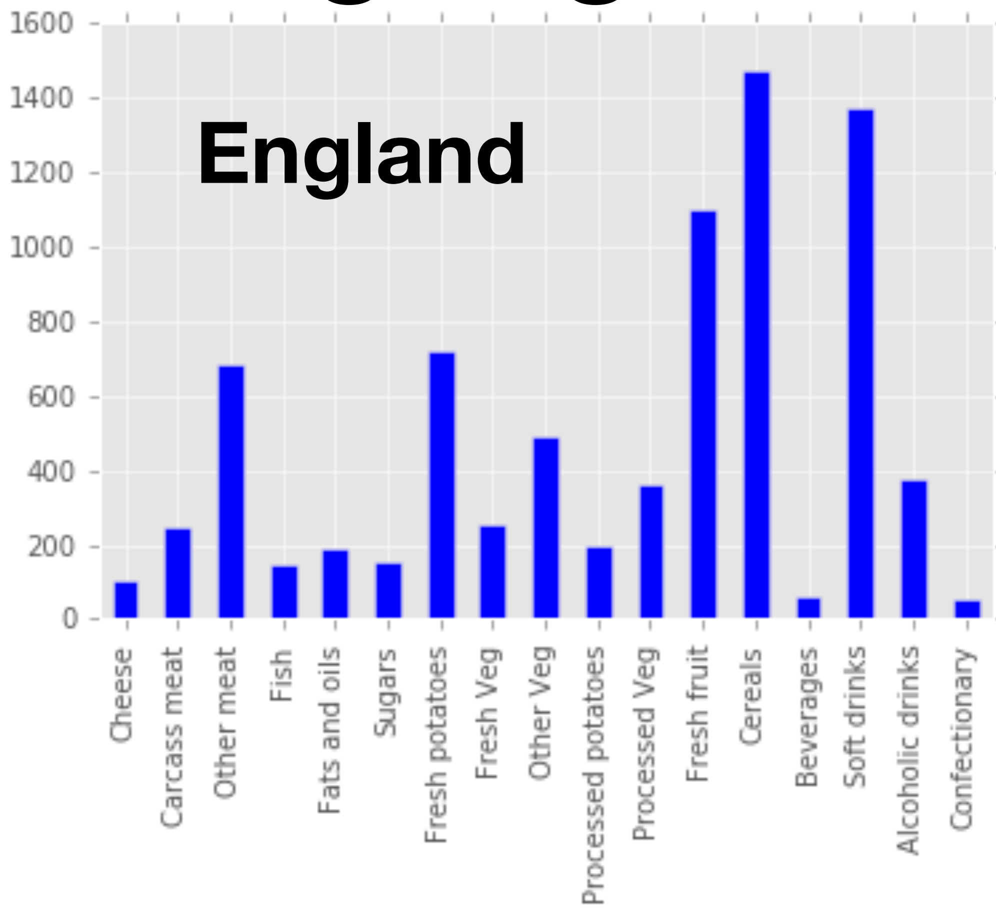
Visualizing High-Dimensional Vectors

Visualizing High-Dimensional Vectors

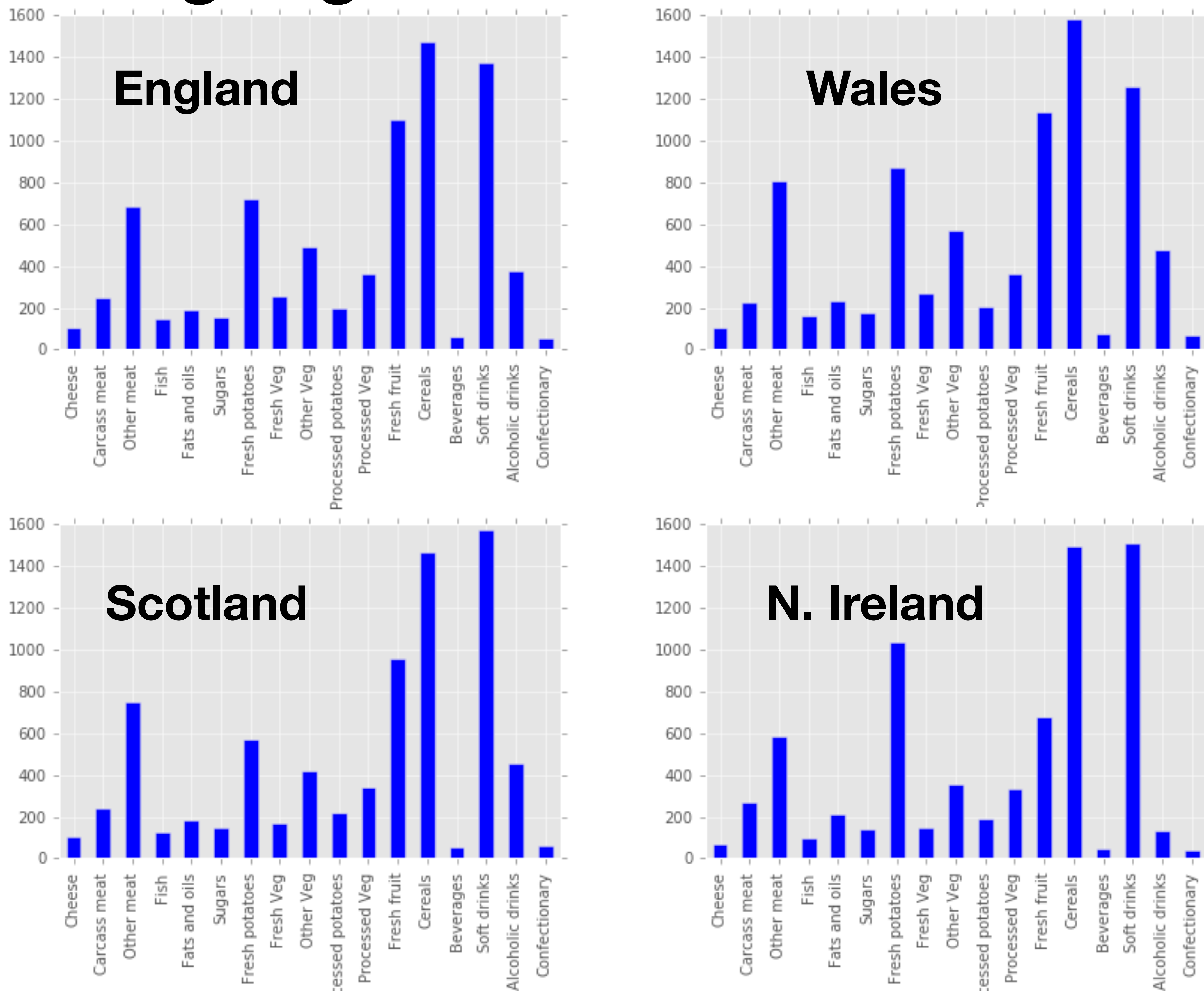
The next two examples are drawn from:
http://setosa.io/ev/principal-component-analysis/

Visualizing High-Dimensional Vectors

Visualizing High-Dimensional Vectors

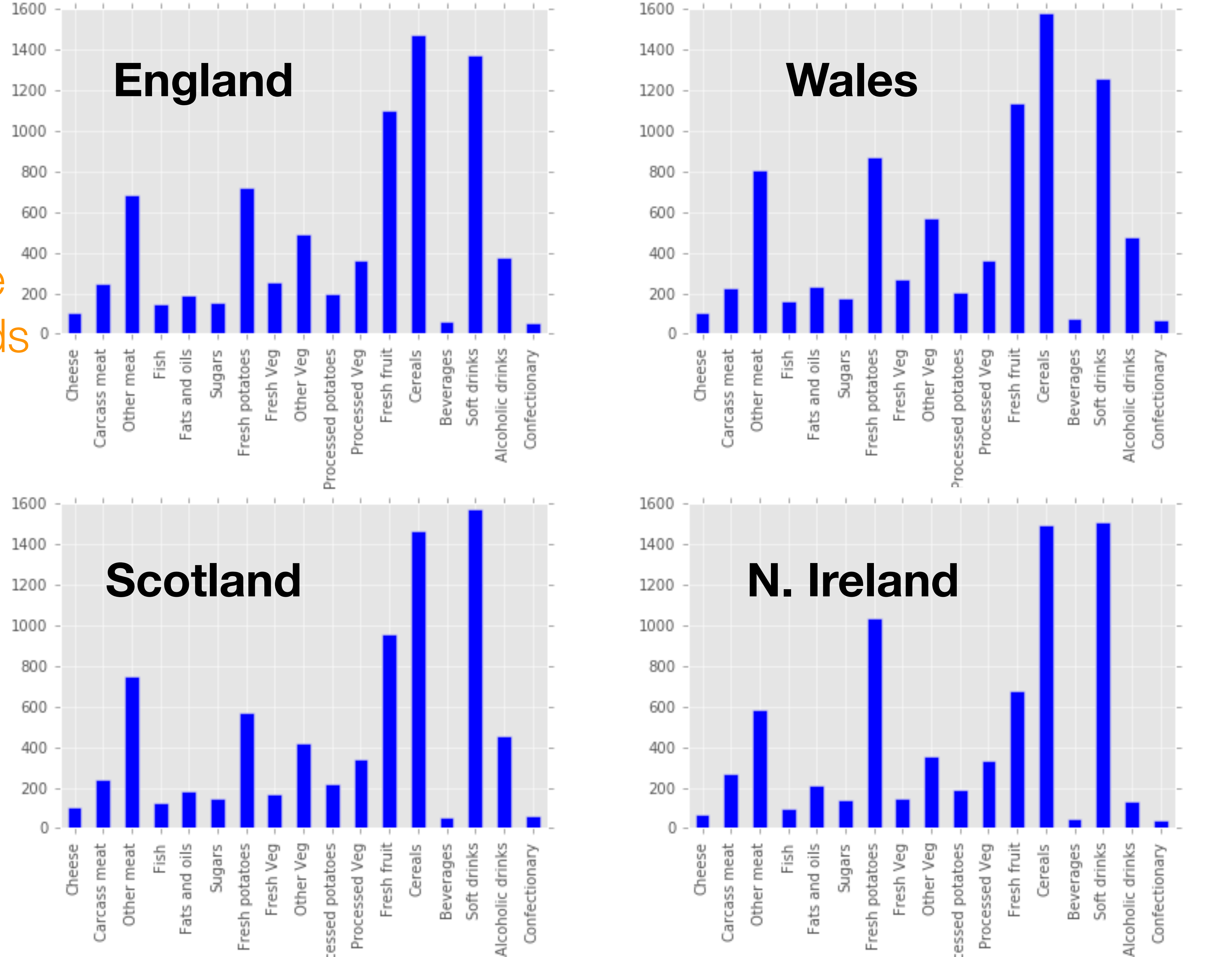


Visualizing High-Dimensional Vectors

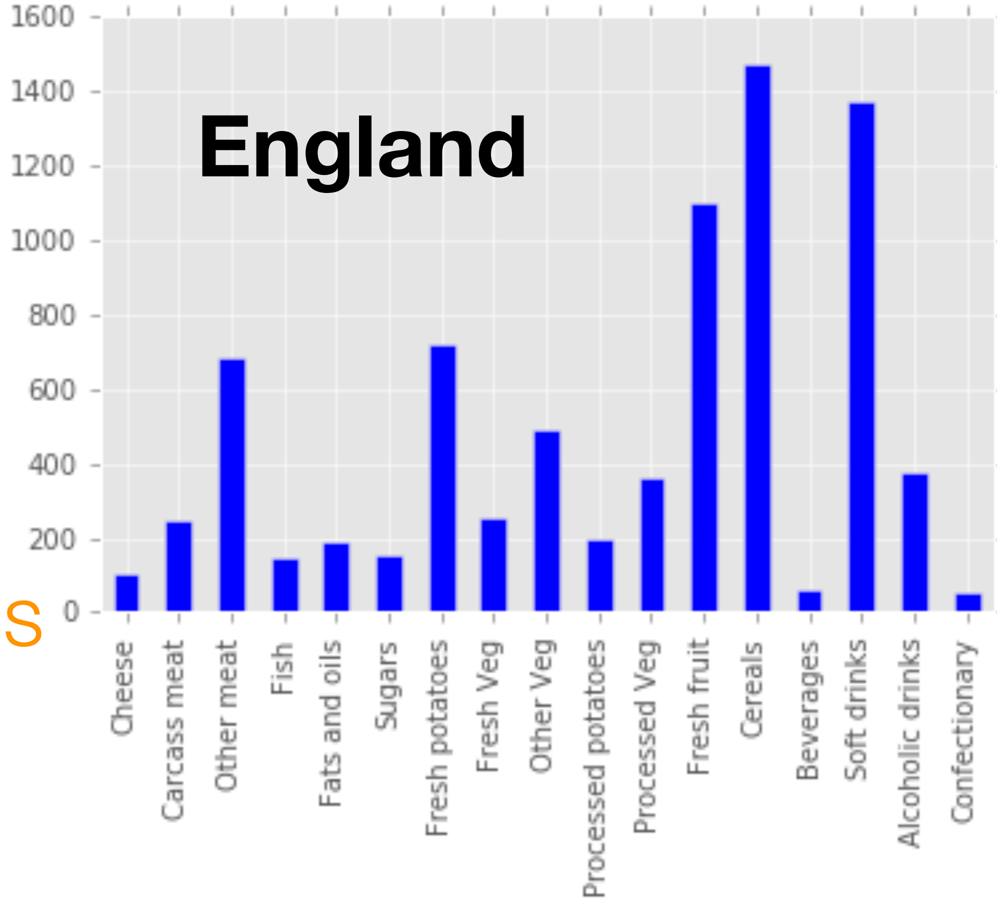


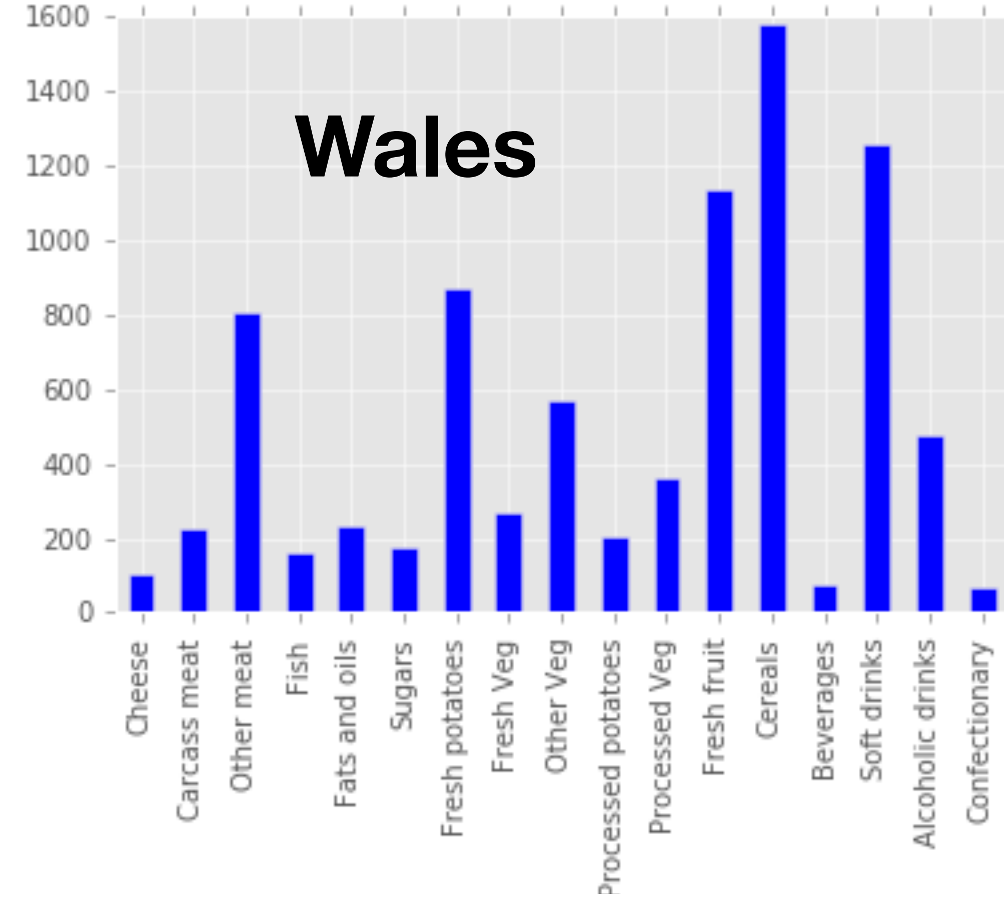
Visualizing High-Dimensional Vectors

Imagine we had hundreds of these

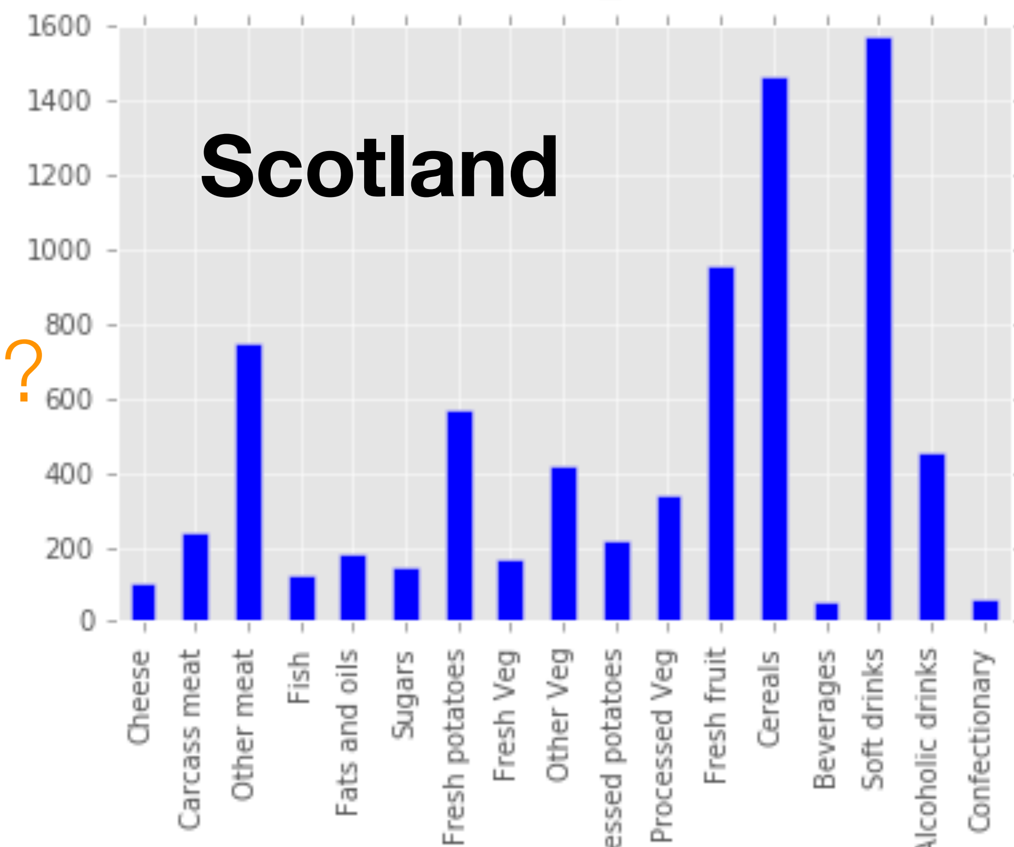


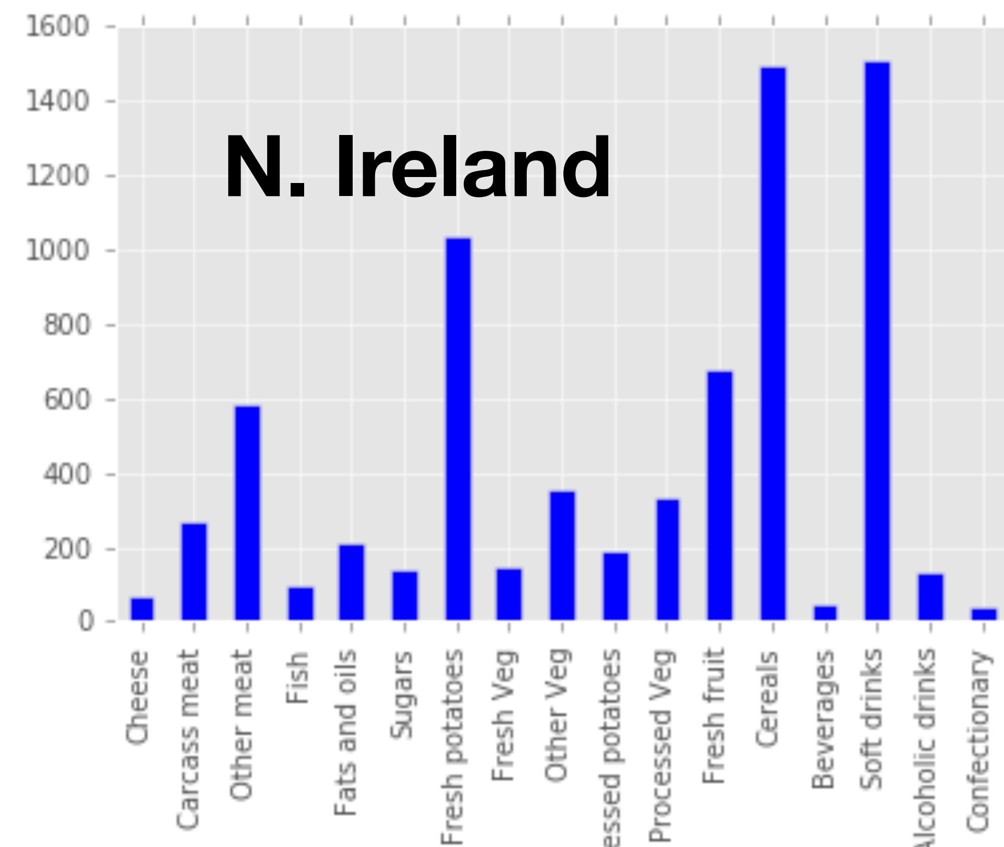
Visualizing High-Dimensional Vectors





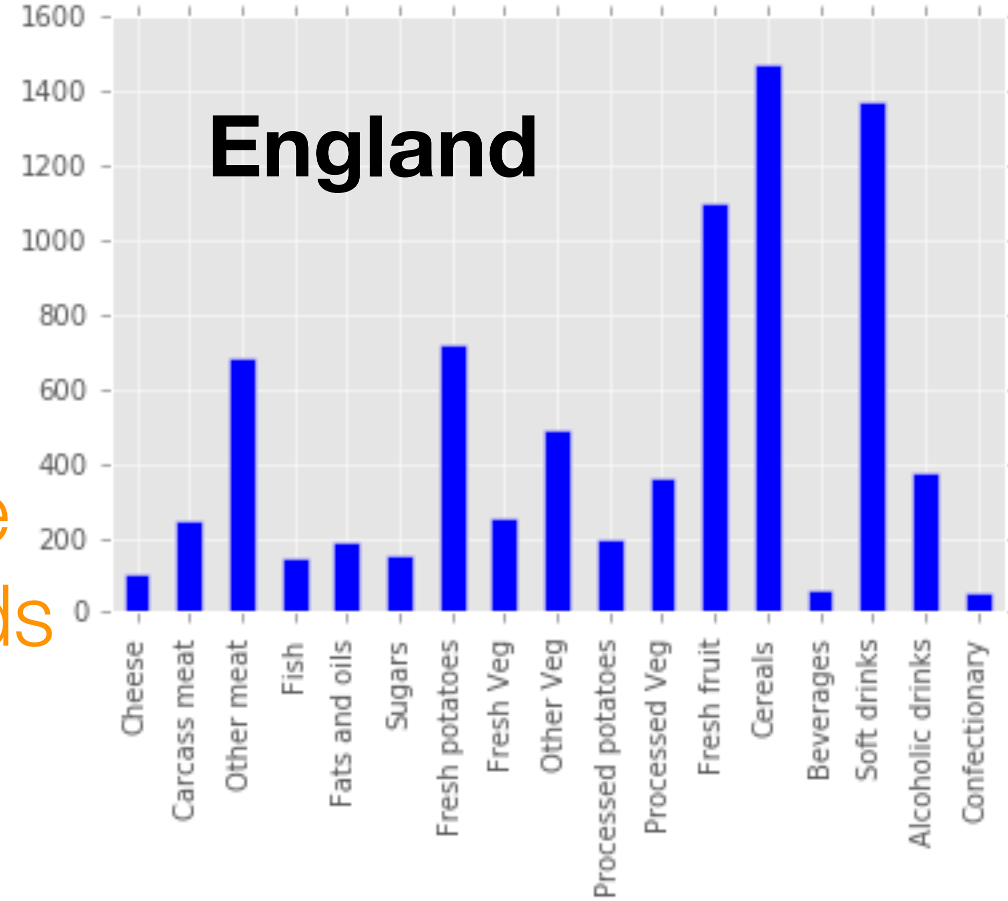
Imagine we had hundreds of these

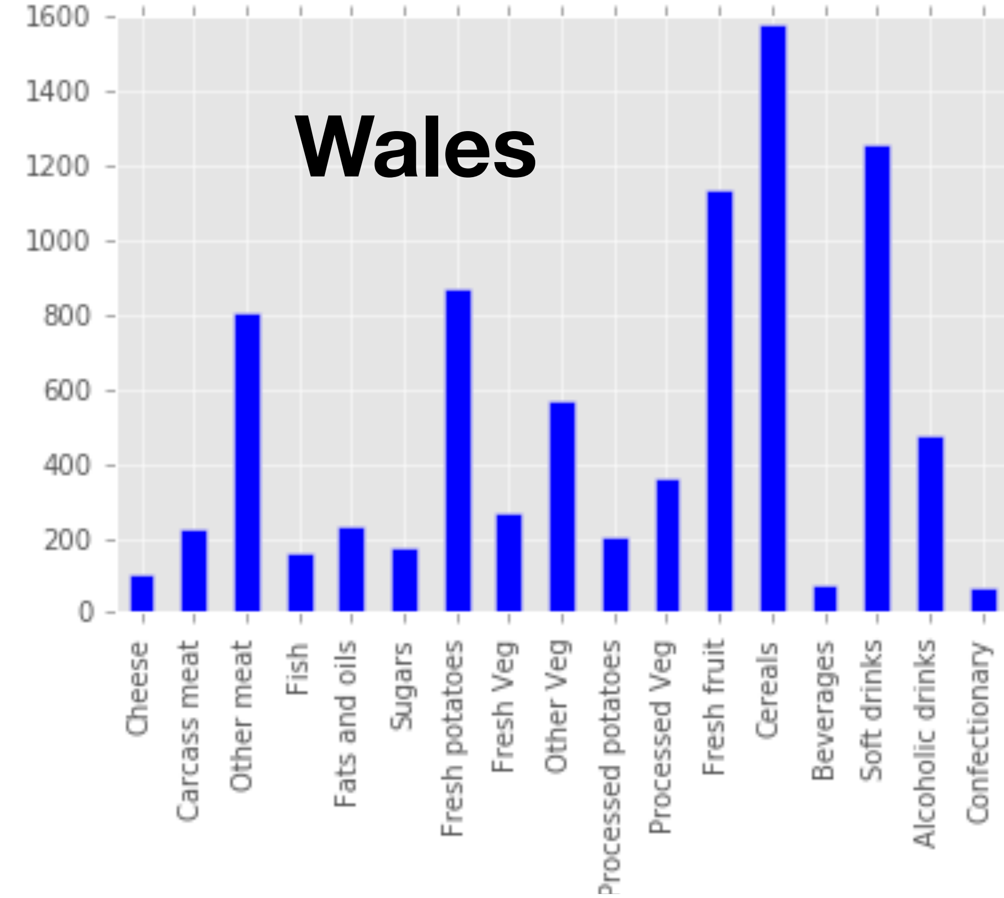




How to visualize these for comparison?

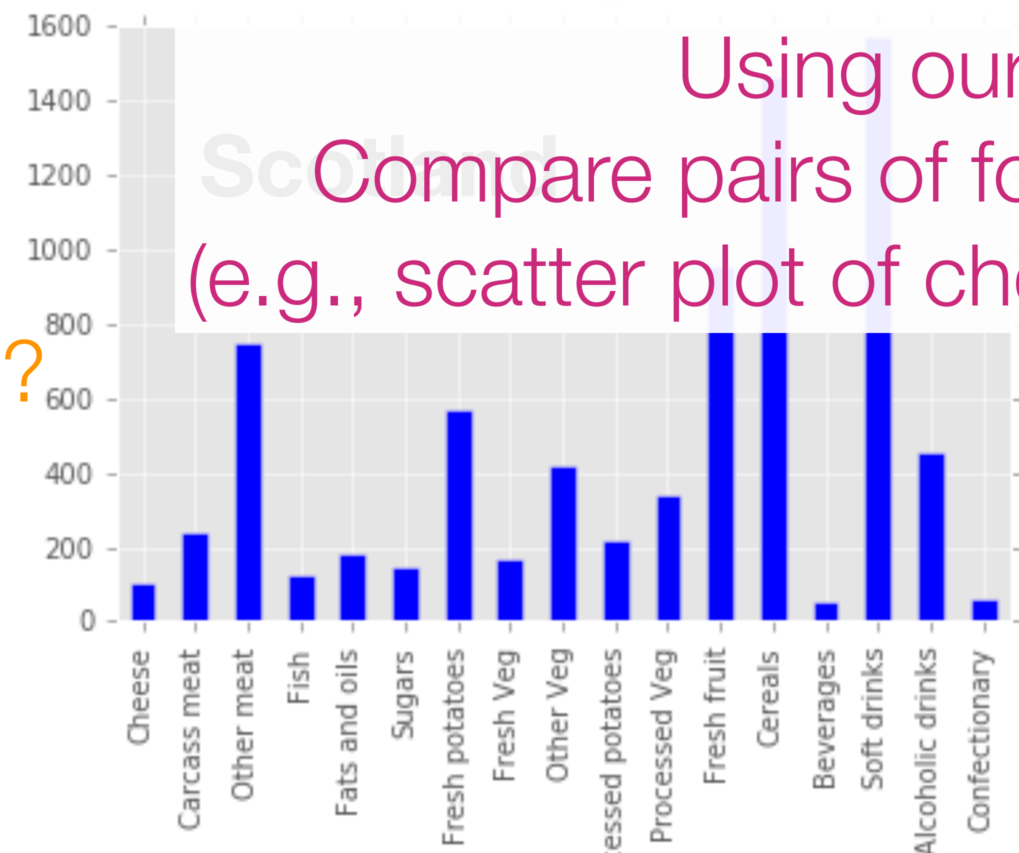
Visualizing High-Dimensional Vectors

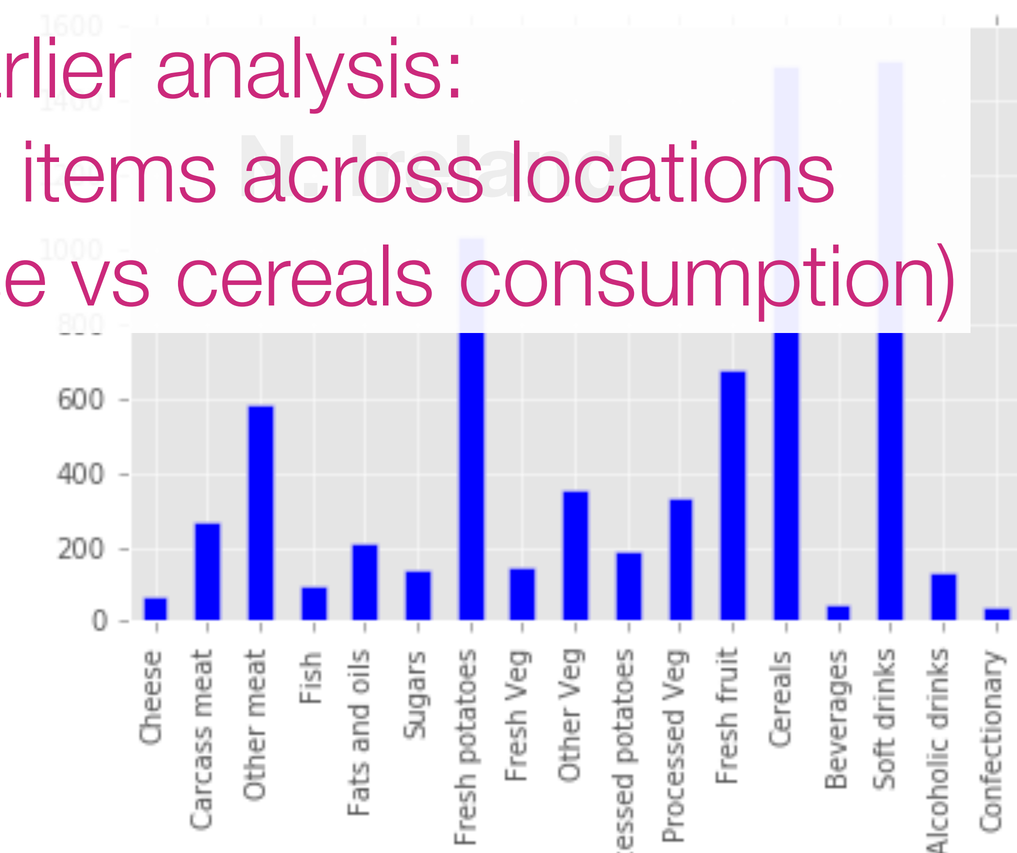




Imagine we had hundreds of these

How to visualize these for comparison?





Using our earlier analysis:
Compare pairs of food items across locations
(e.g., scatter plot of cheese vs cereals consumption)

Visualizing High-Dimensional Vectors

Imagine we had hundreds of these

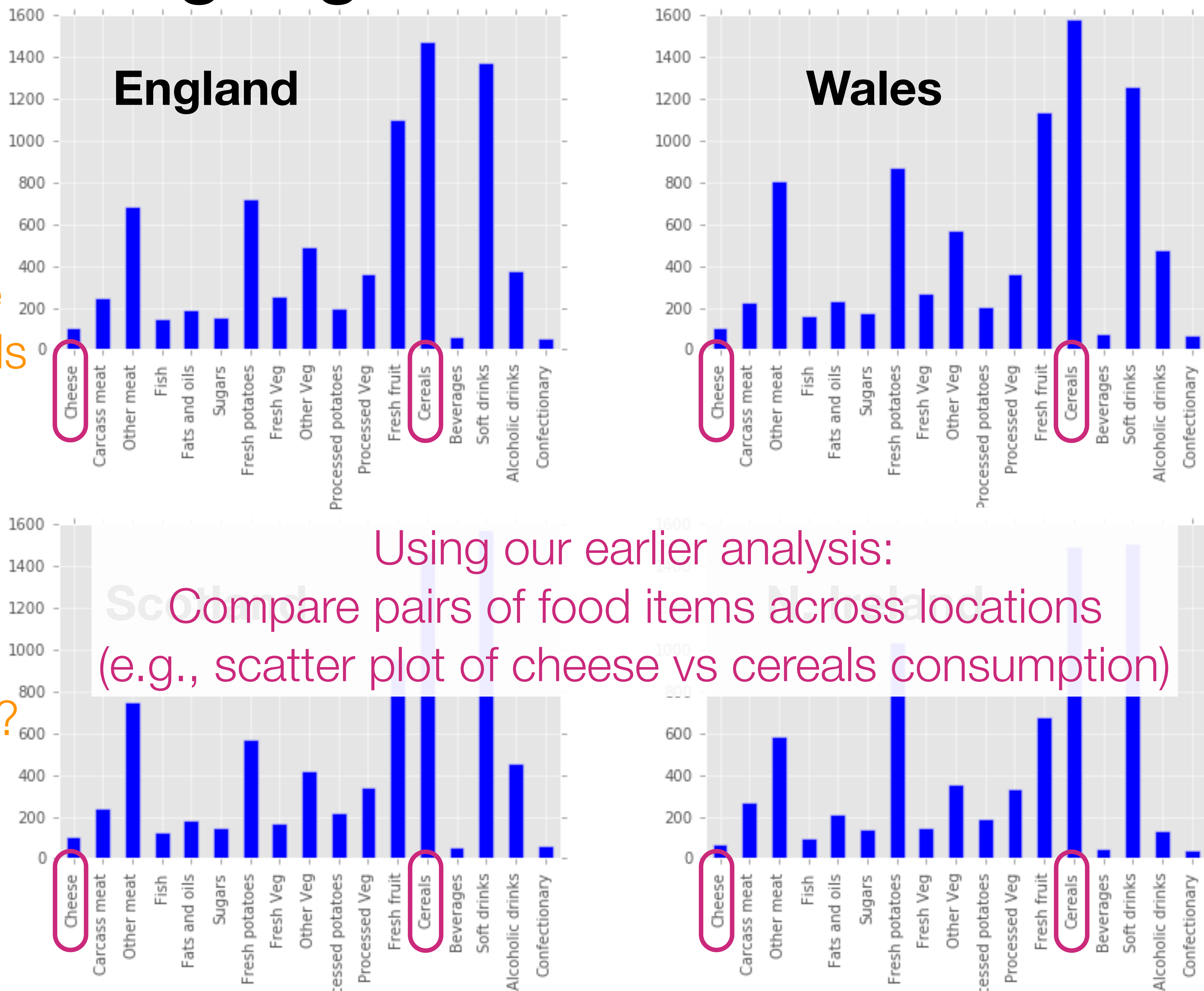
How to visualize these for comparison?

Using our earlier analysis:
Compare pairs of food items across locations
(e.g., scatter plot of cheese vs cereals consumption)

Visualizing High-Dimensional Vectors

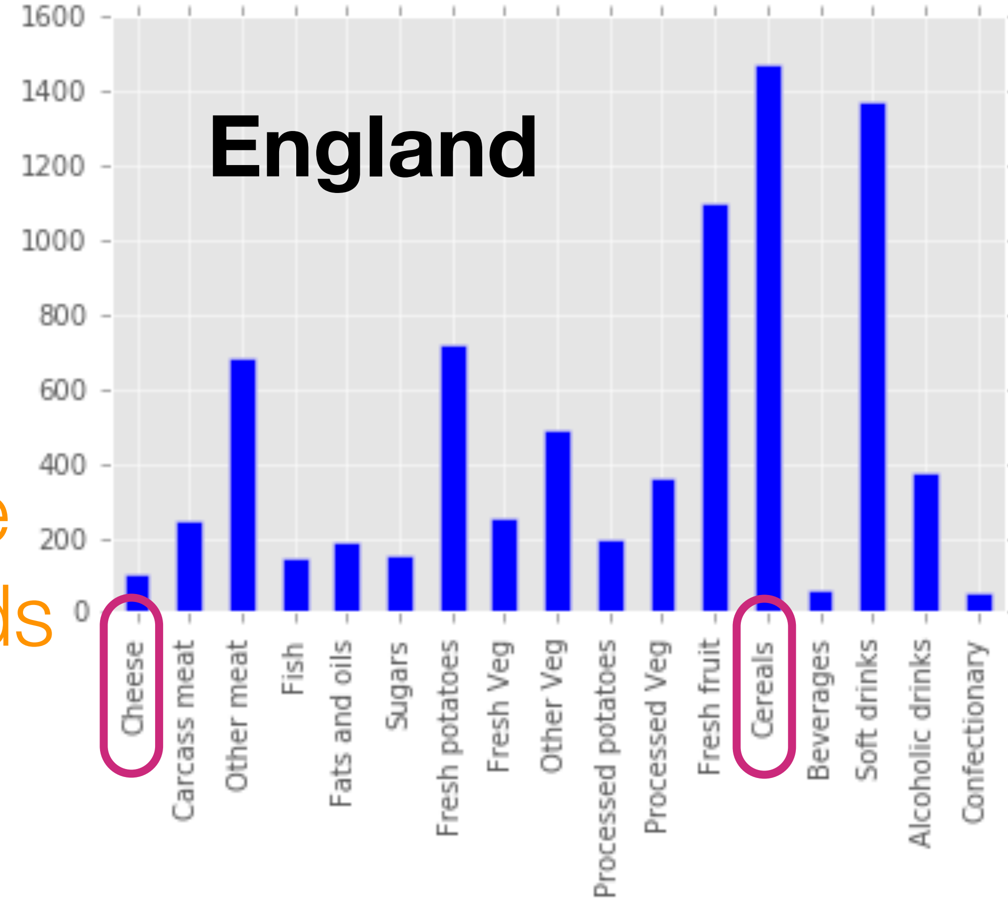
Imagine we had hundreds of these

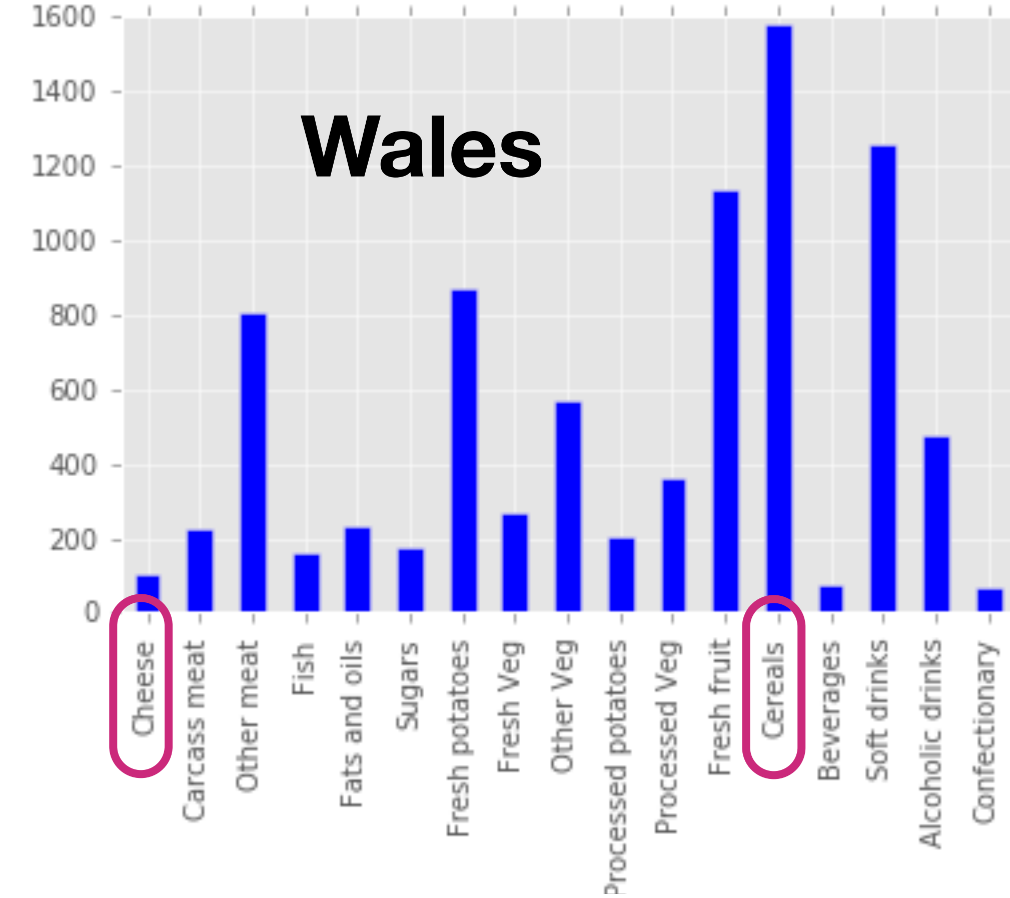
How to visualize these for comparison?



Using our earlier analysis:
Compare pairs of food items across locations
(e.g., scatter plot of cheese vs cereals consumption)

Visualizing High-Dimensional Vectors





Imagine we had hundreds of these

How to visualize these for comparison?

Using our earlier analysis:
Compare pairs of food items across locations
(e.g., scatter plot of cheese vs cereals consumption)

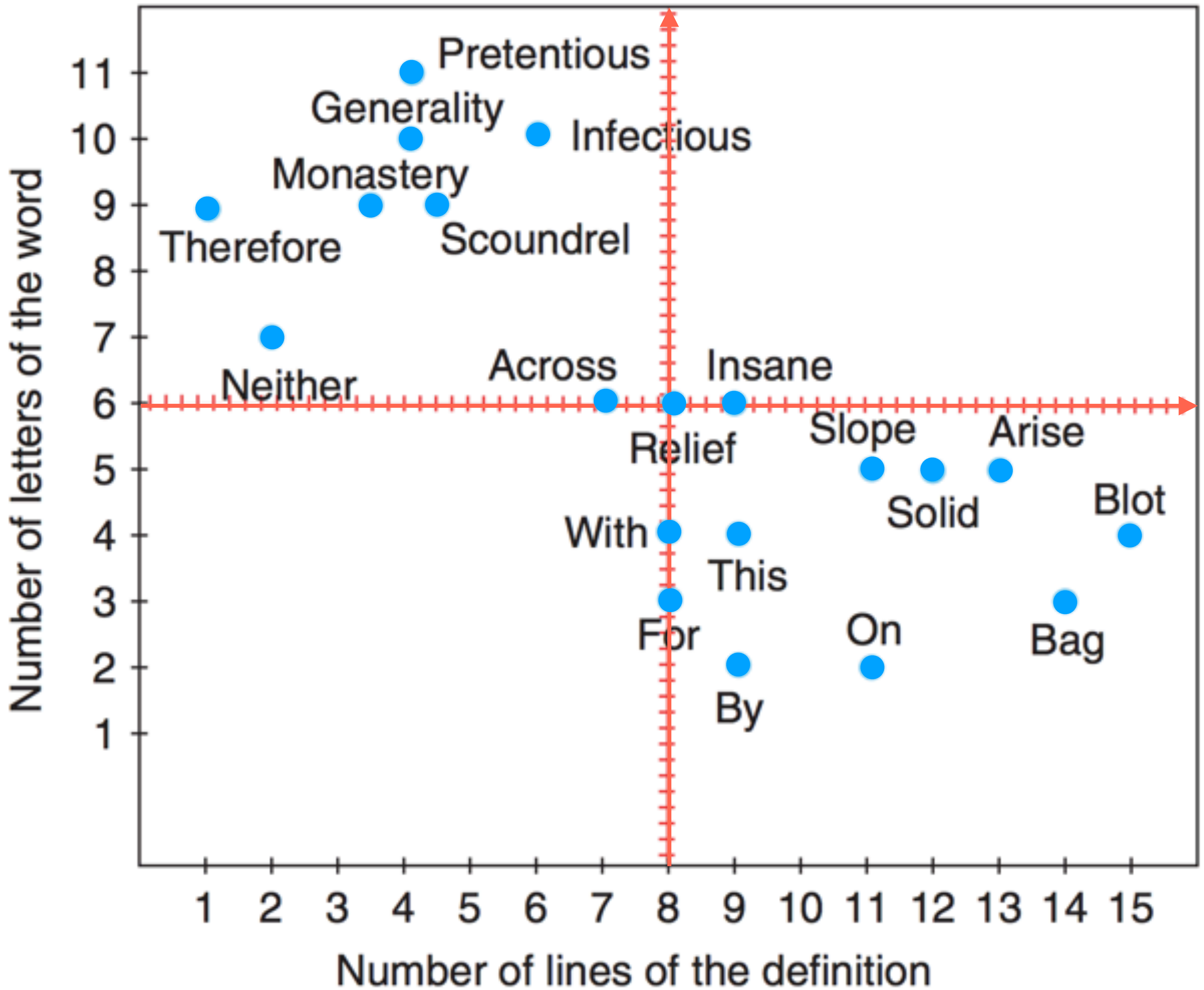
But unclear how to compare the locations (England, Wales, Scotland, N. Ireland)!

The issue is that as humans we can only really visualize up to 3 dimensions easily

Goal: Somehow reduce the dimensionality of the data preferably to 1, 2, or 3

Principal Component Analysis (PCA)

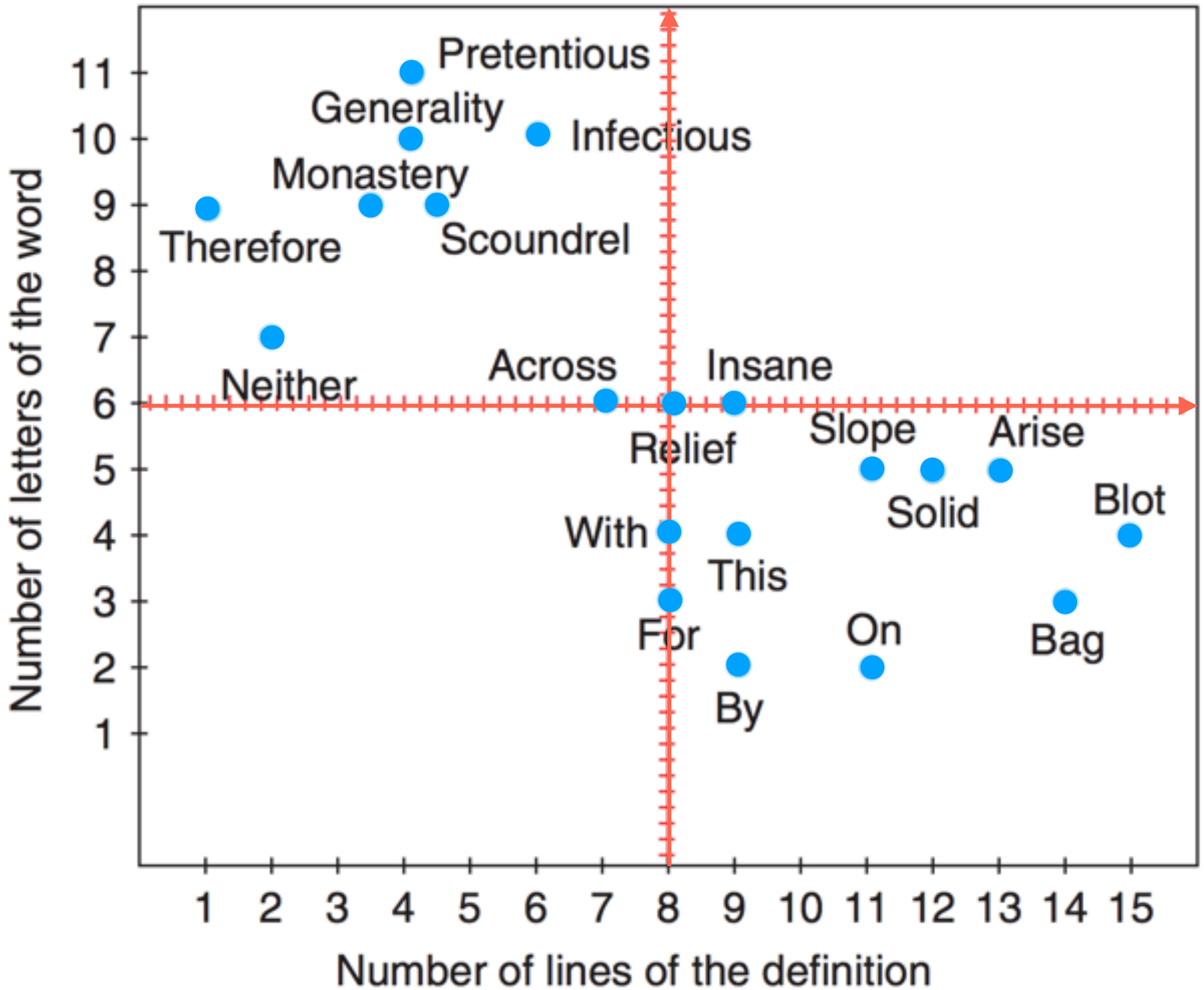
Principal Component Analysis (PCA)



Hervé Abdi and Lynne J. Williams. Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics. 2010.

Principal Component Analysis (PCA)

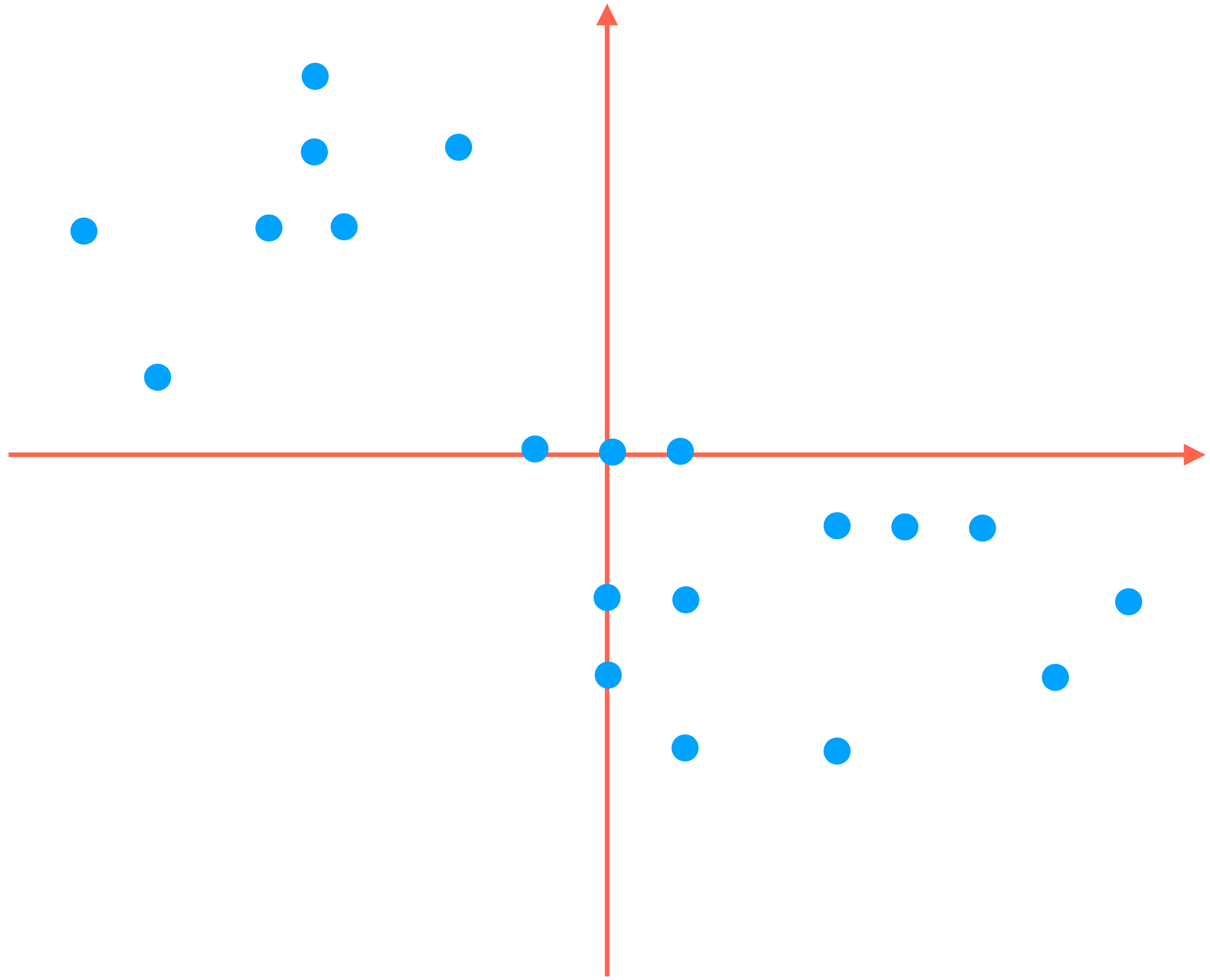
How to project 2D data down to 1D?



Hervé Abdi and Lynne J. Williams. Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics. 2010.

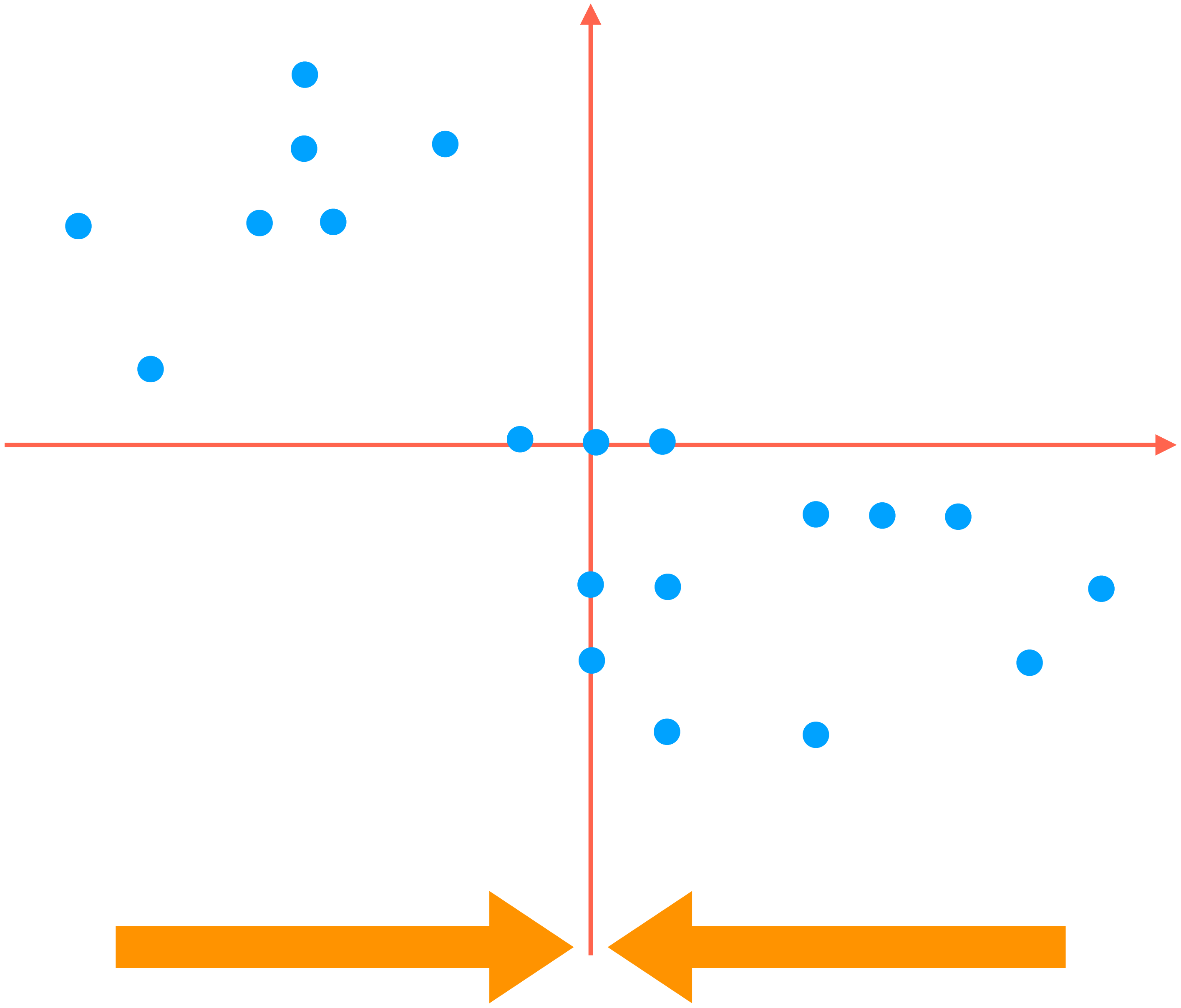
Principal Component Analysis (PCA)

How to project 2D data down to 1D?



Principal Component Analysis (PCA)

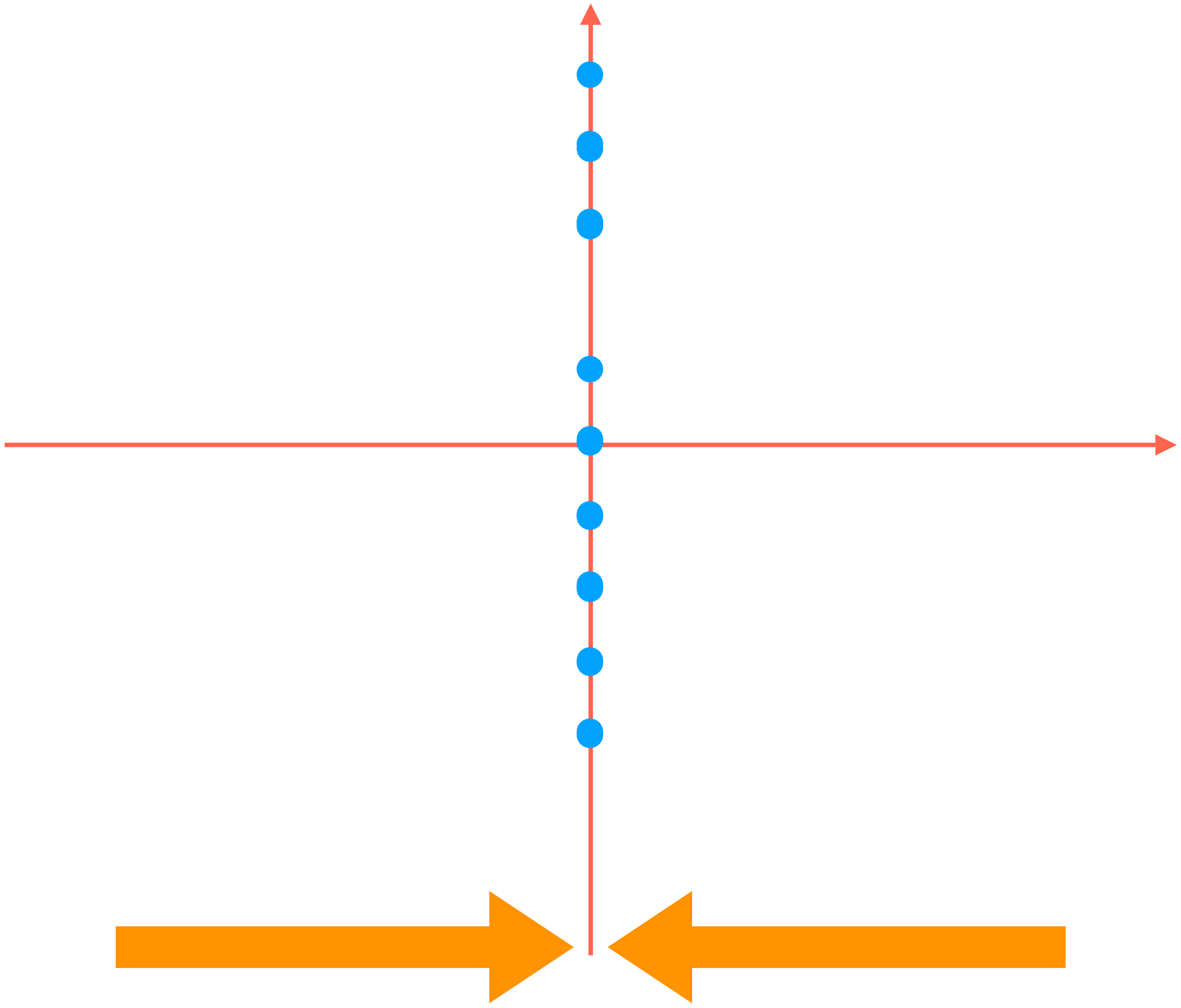
How to project 2D data down to 1D?



Simplest thing to try: flatten to one of the red axes

Principal Component Analysis (PCA)

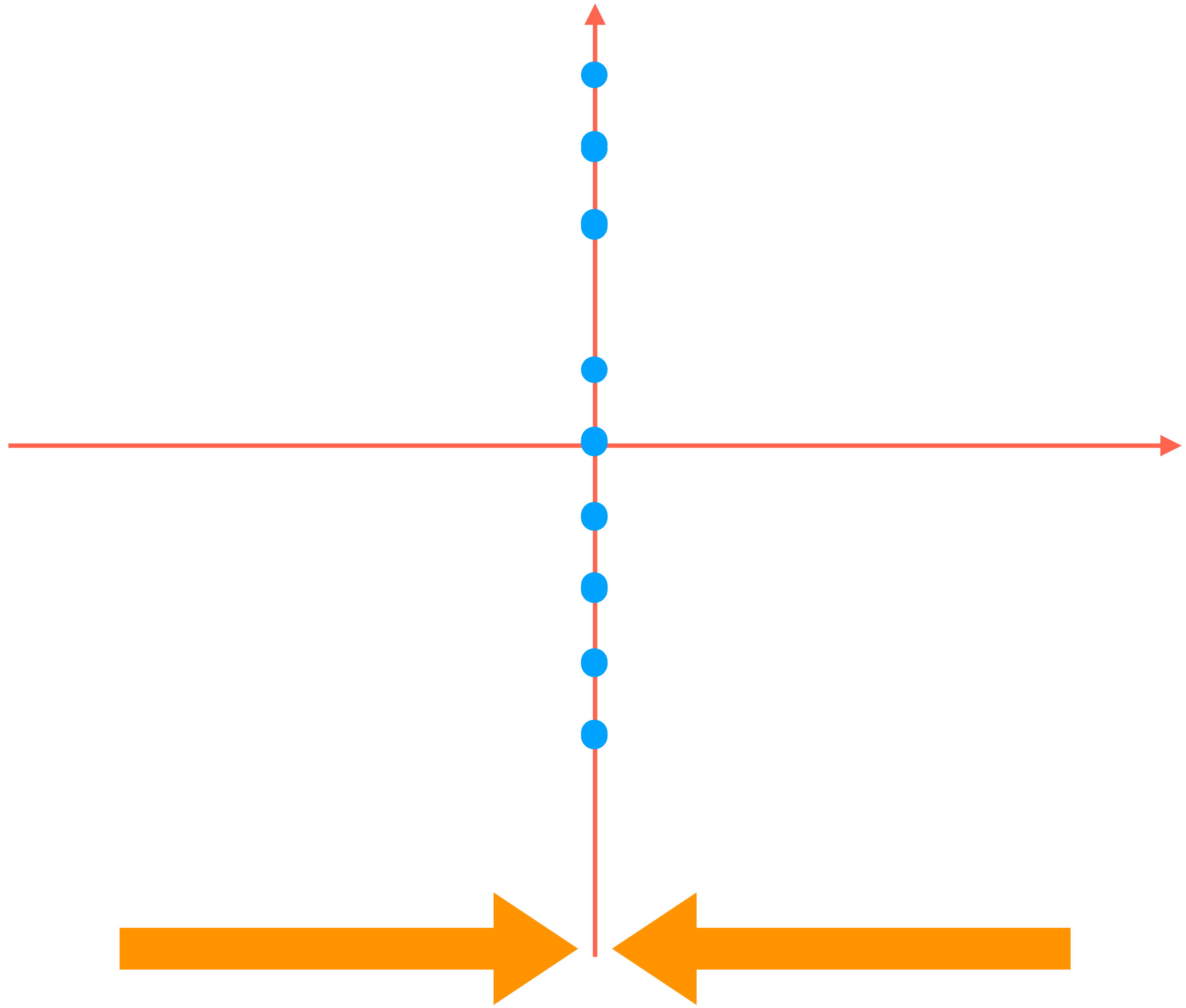
How to project 2D data down to 1D?



Simplest thing to try: flatten to one of the red axes

Principal Component Analysis (PCA)

How to project 2D data down to 1D?

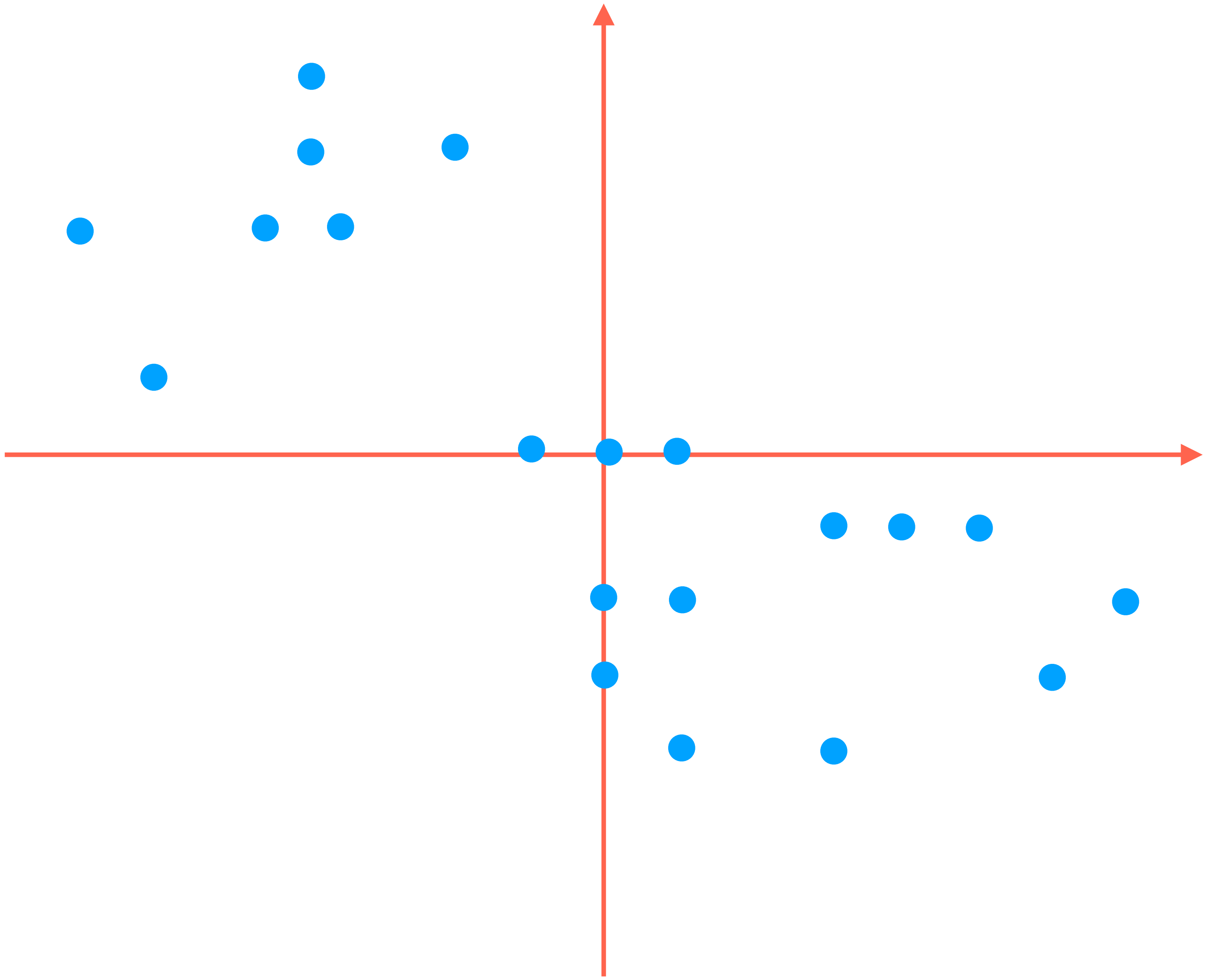


Simplest thing to try: flatten to one of the red axes

(We could of course flatten to the other red axis)

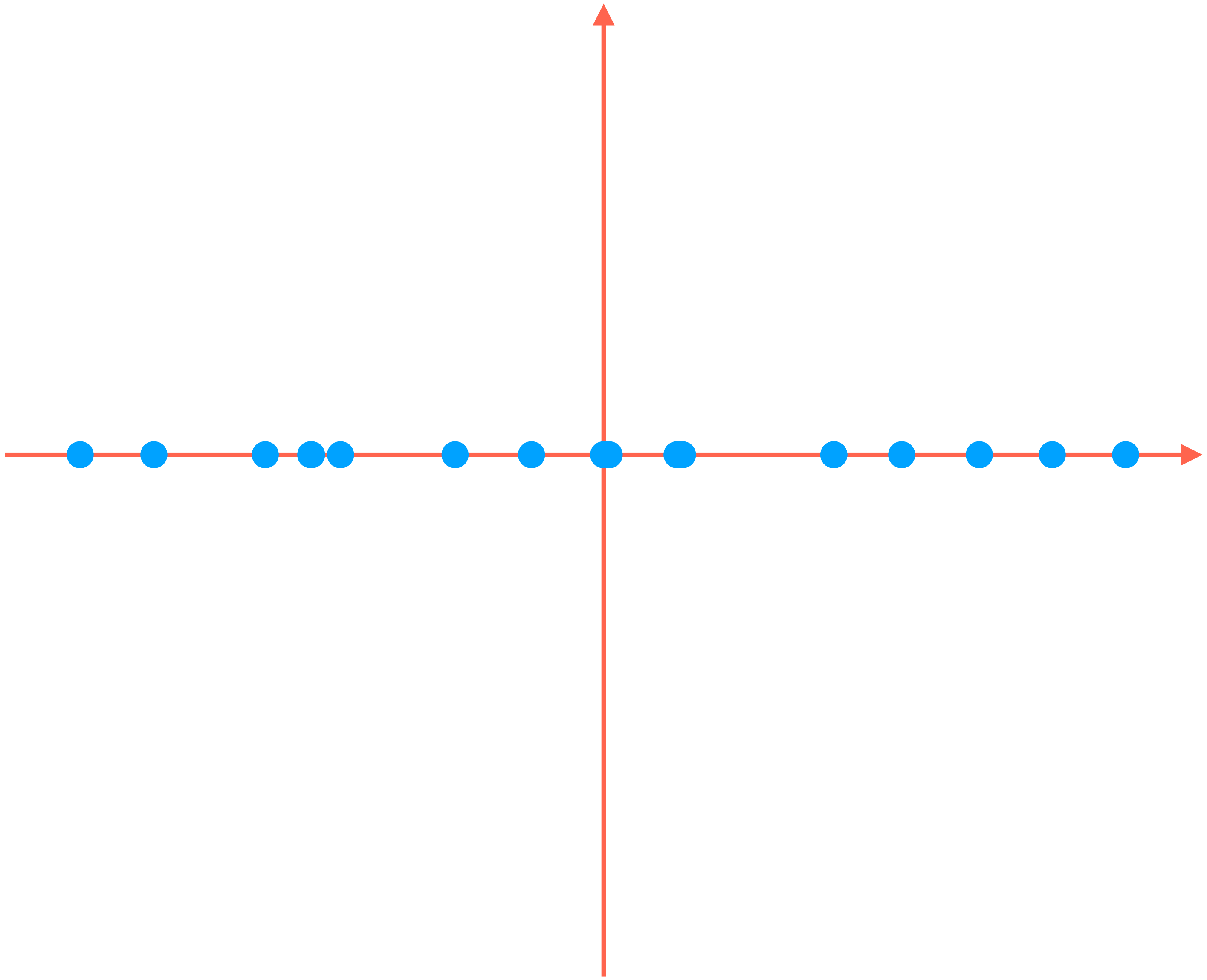
Principal Component Analysis (PCA)

How to project 2D data down to 1D?



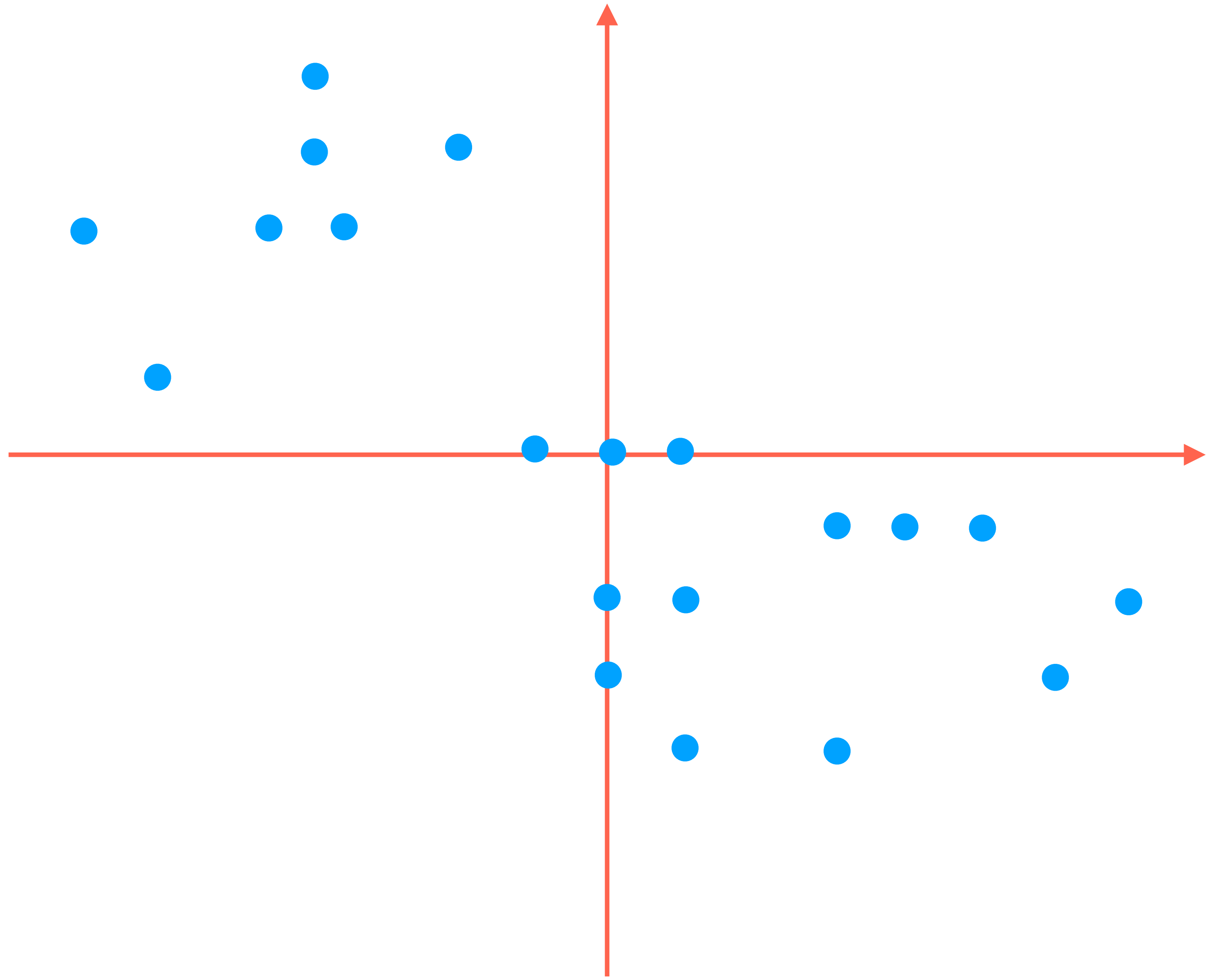
Principal Component Analysis (PCA)

How to project 2D data down to 1D?



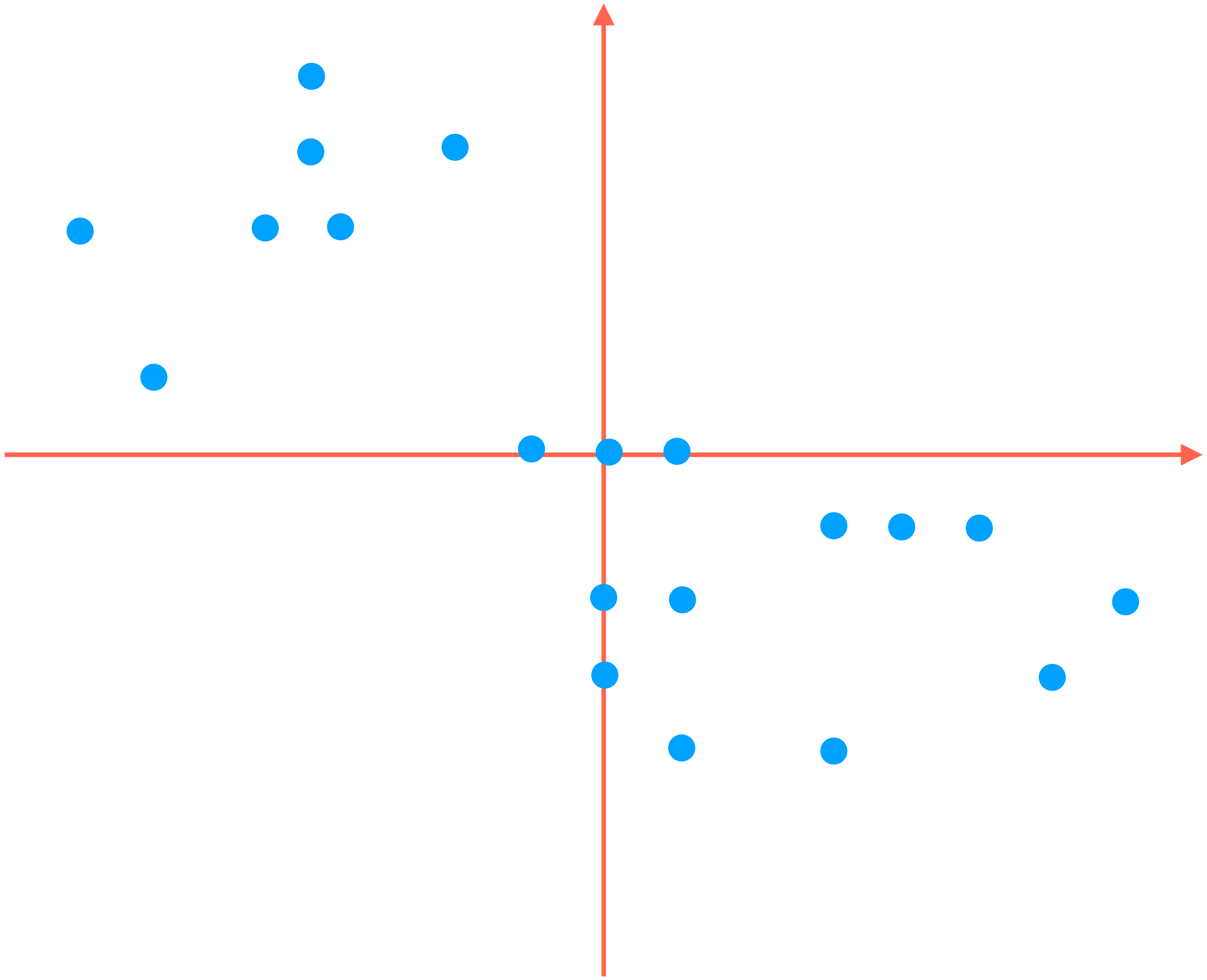
Principal Component Analysis (PCA)

How to project 2D data down to 1D?



Principal Component Analysis (PCA)

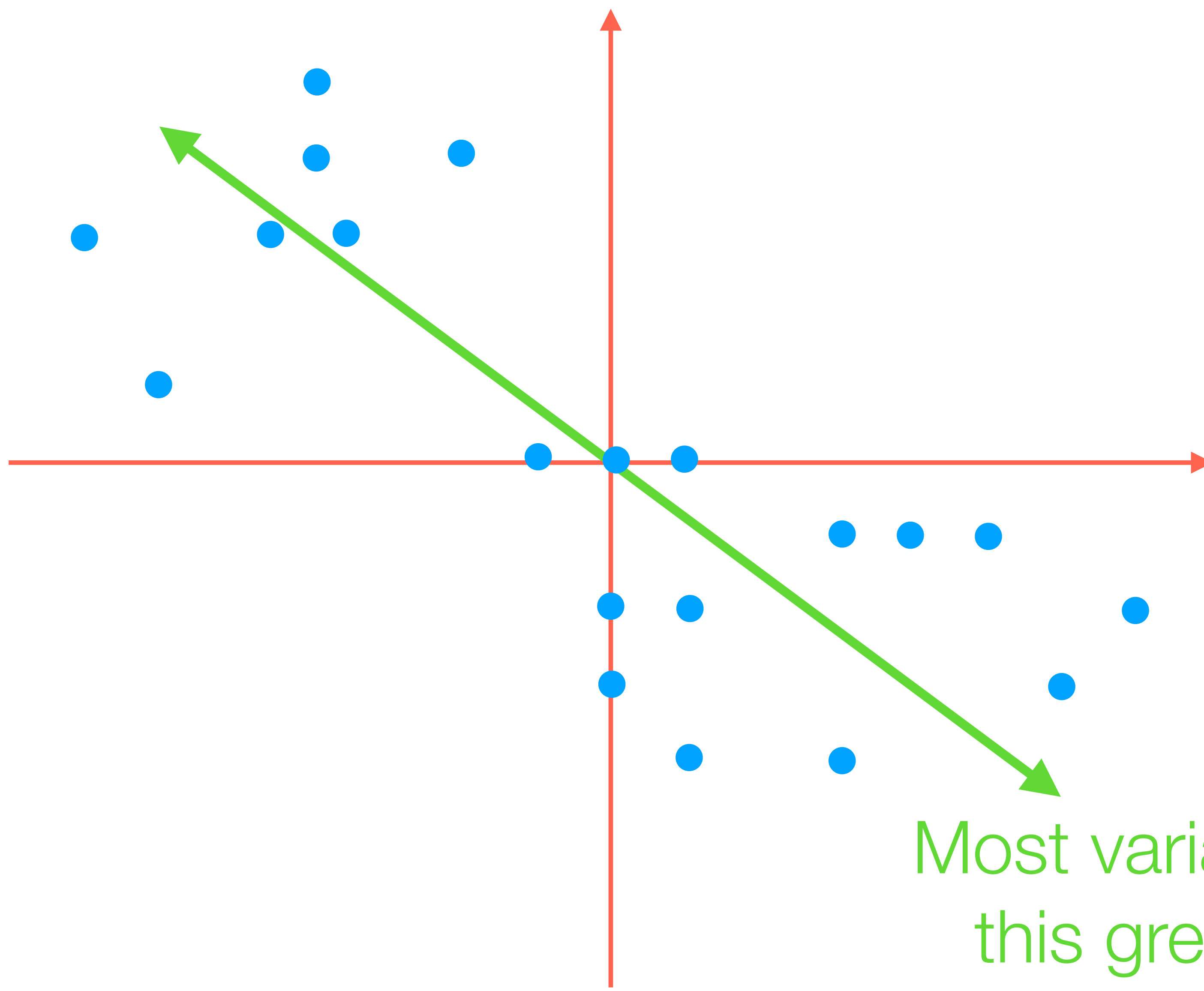
How to project 2D data down to 1D?



But notice that most of the variability in the data is *not* aligned with the red axes!

Principal Component Analysis (PCA)

How to project 2D data down to 1D?

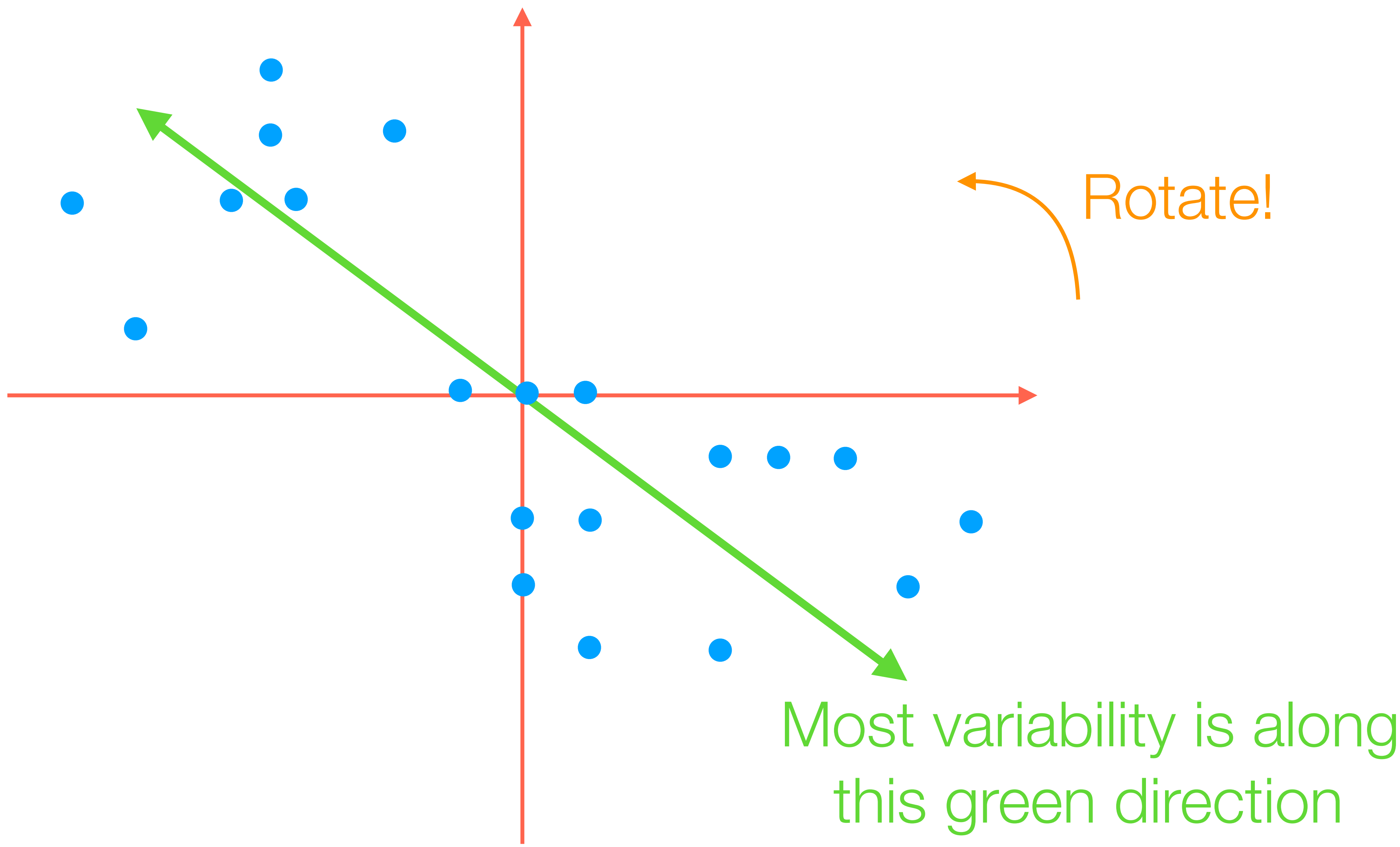


Most variability is along this green direction

But notice that most of the variability in the data is *not* aligned with the red axes!

Principal Component Analysis (PCA)

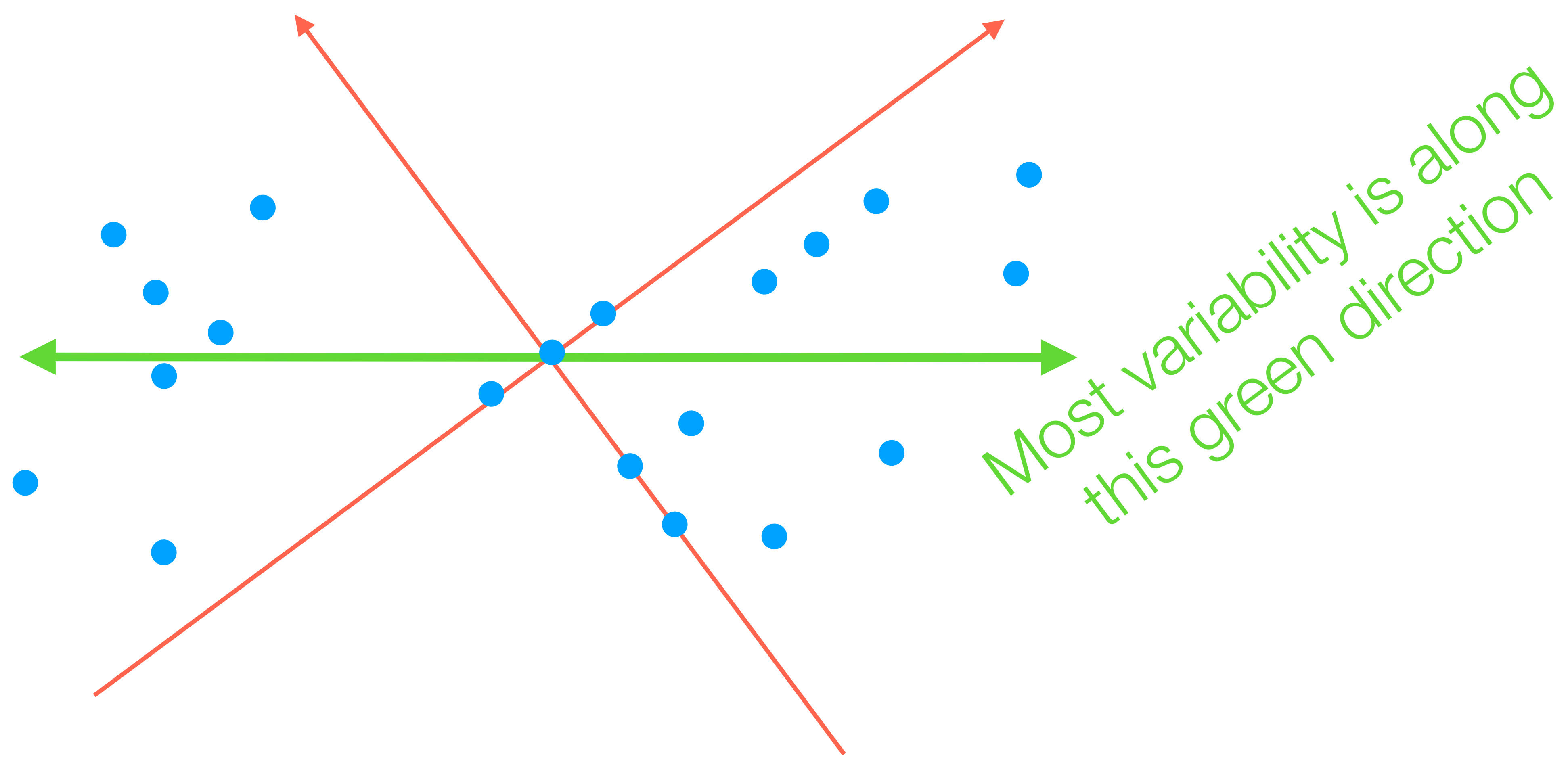
How to project 2D data down to 1D?



But notice that most of the variability in the data is *not* aligned with the red axes!

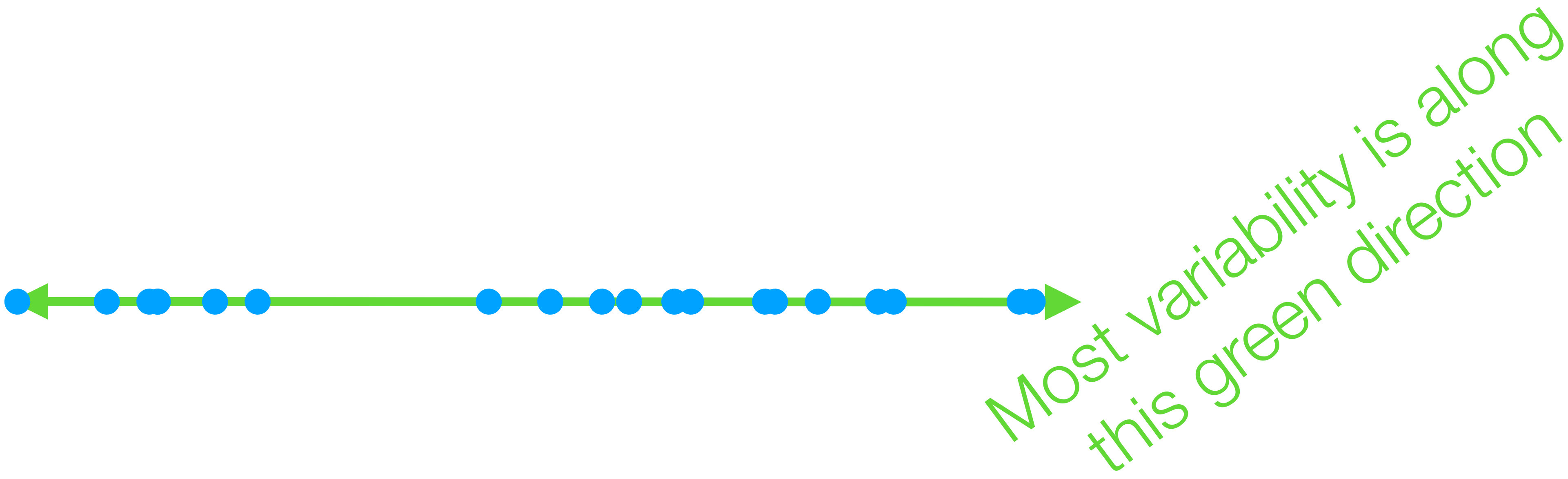
Principal Component Analysis (PCA)

How to project 2D data down to 1D?



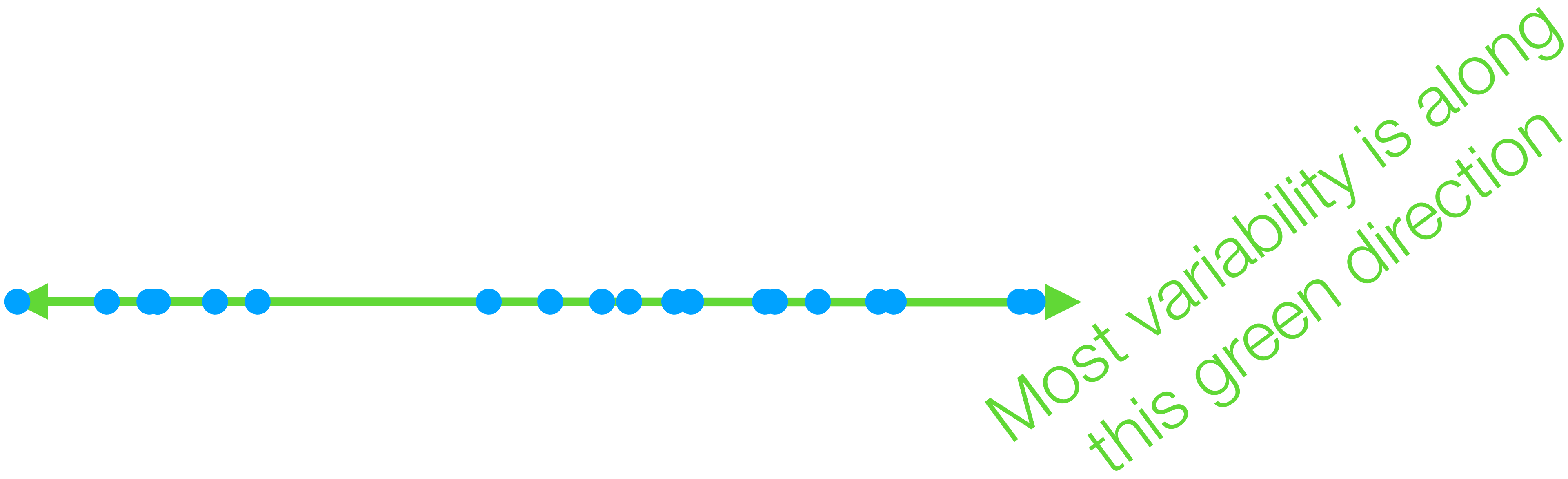
Principal Component Analysis (PCA)

How to project 2D data down to 1D?



Principal Component Analysis (PCA)

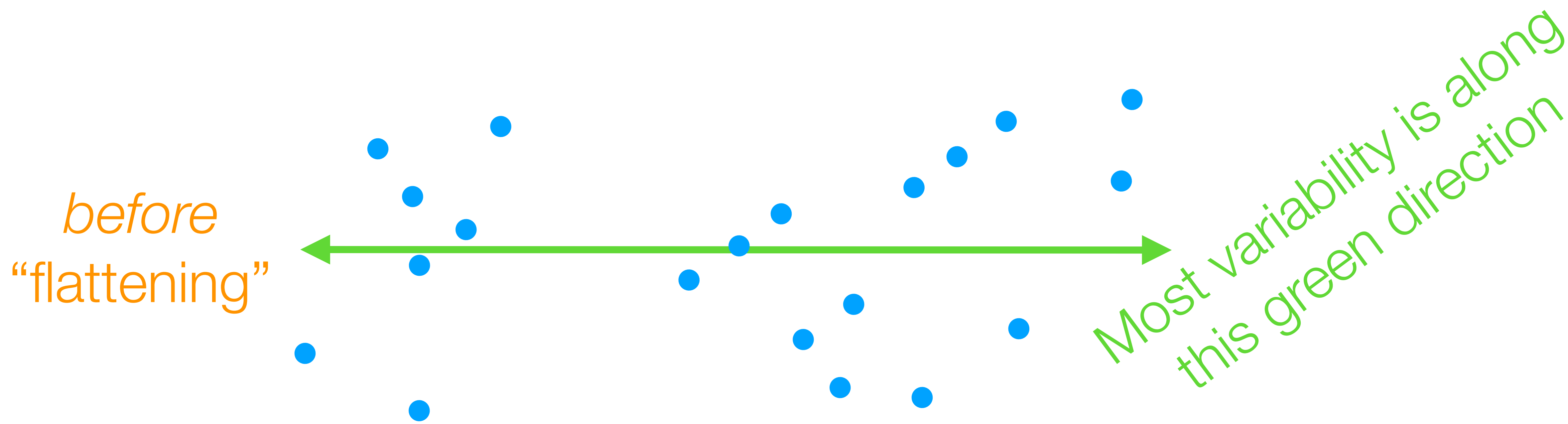
How to project 2D data down to 1D?



The idea of PCA actually works for 2D ➔ 2D as well
(and just involves rotating, and not “flattening” the data)

Principal Component Analysis (PCA)

How to project 2D data down to 1D?

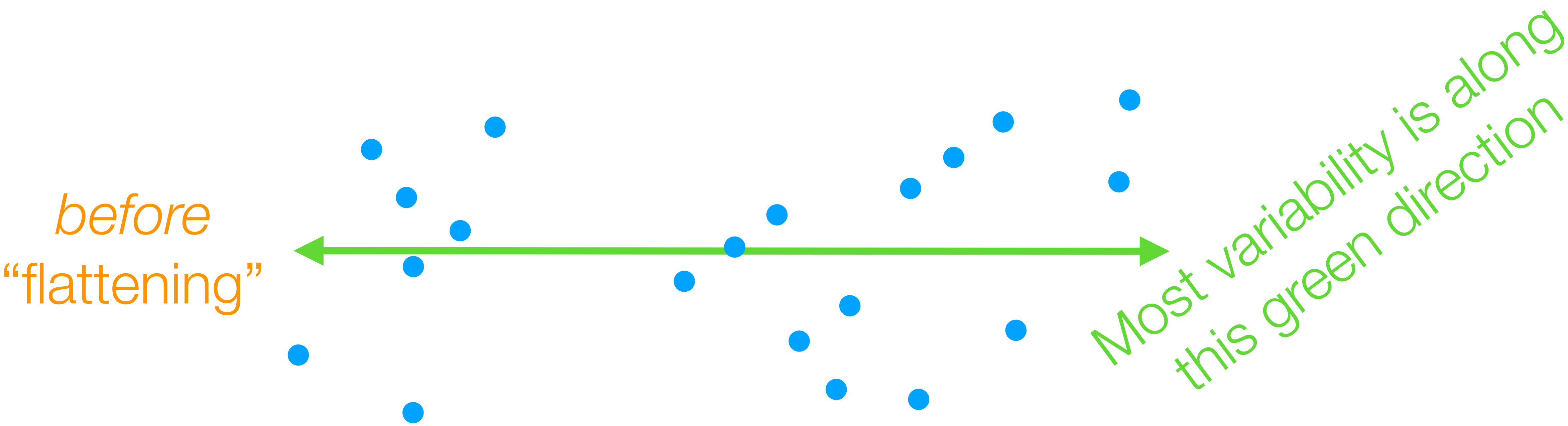


The idea of PCA actually works for 2D ➔ 2D as well
(and just involves rotating, and not "flattening" the data)

Principal Component Analysis (PCA)

~~How to project 2D data down to 1D?~~

How to rotate 2D data so 1st axis has most variance

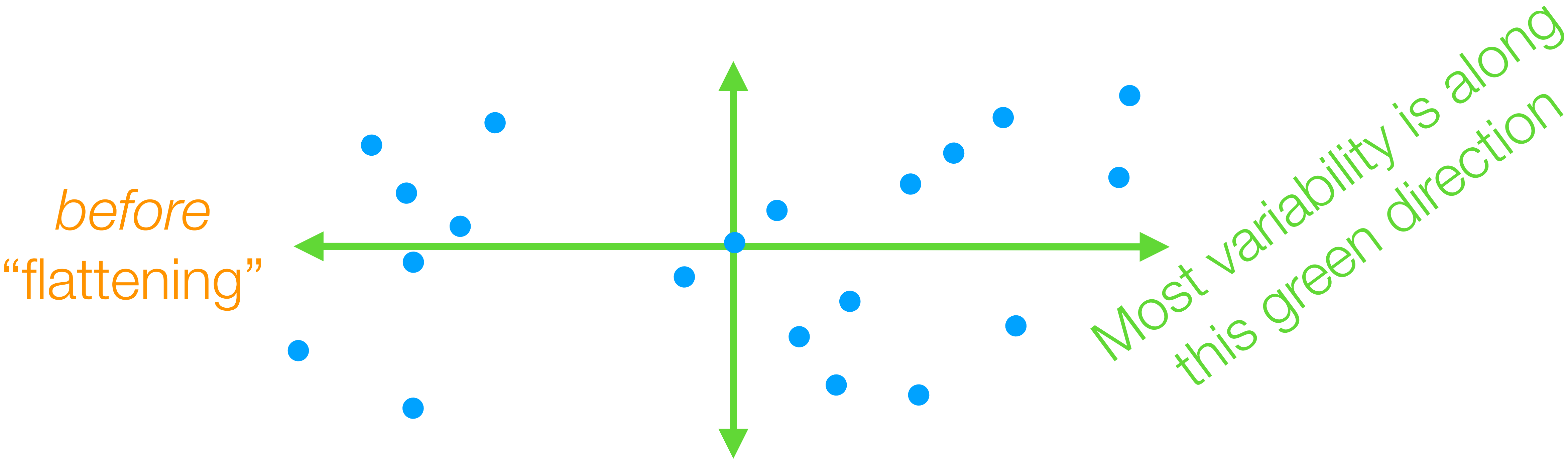


The idea of PCA actually works for 2D ➔ 2D as well
(and just involves rotating, and not “flattening” the data)

Principal Component Analysis (PCA)

~~How to project 2D data down to 1D?~~

How to rotate 2D data so 1st axis has most variance



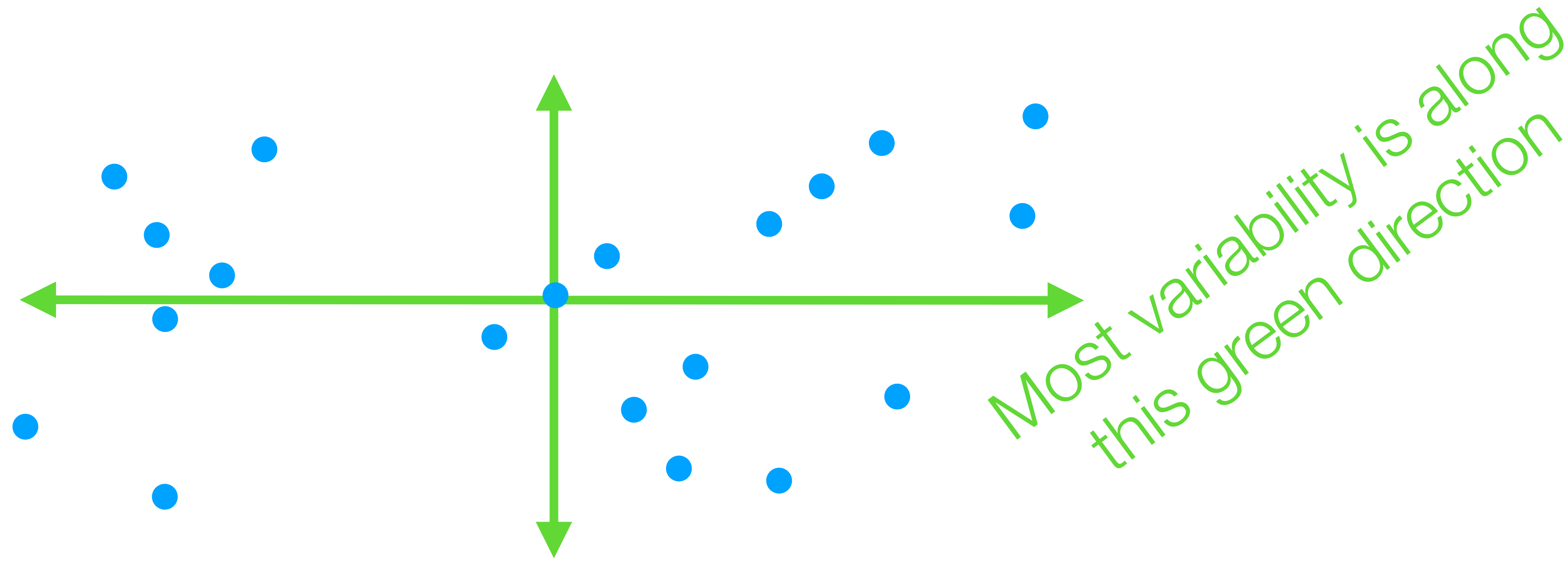
The idea of PCA actually works for 2D ➔ 2D as well
(and just involves rotating, and not “flattening” the data)

Principal Component Analysis (PCA)

~~How to project 2D data down to 1D?~~

How to rotate 2D data so 1st axis has most variance

before
"flattening"



The idea of PCA actually works for 2D ➔ 2D as well
(and just involves rotating, and not "flattening" the data)

2nd green axis chosen to be 90° ("orthogonal") from first green axis

Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

- Finds top *k* orthogonal directions that explain the most variance in the data

Principal Component Analysis (PCA)

- Finds top *k* orthogonal directions that explain the most variance in the data
    - 1st component: explains most variance along 1 dimension

Principal Component Analysis (PCA)

- Finds top *k* orthogonal directions that explain the most variance in the data
    - 1st component: explains most variance along 1 dimension
    - 2nd component: explains most of remaining variance along next dimension that is orthogonal to 1st dimension

Principal Component Analysis (PCA)

- Finds top *k* orthogonal directions that explain the most variance in the data
  - 1st component: explains most variance along 1 dimension
  - 2nd component: explains most of remaining variance along next dimension that is orthogonal to 1st dimension
  - …

Principal Component Analysis (PCA)

- Finds top *k* orthogonal directions that explain the most variance in the data
  - 1st component: explains most variance along 1 dimension
  - 2nd component: explains most of remaining variance along next dimension that is orthogonal to 1st dimension
  - …
- “Flatten” data to the top *k* dimensions to get lower dimensional representation (if $k <$ original dimension)

Principal Component Analysis (PCA)

3D example from:
http://setosa.io/ev/principal-component-analysis/

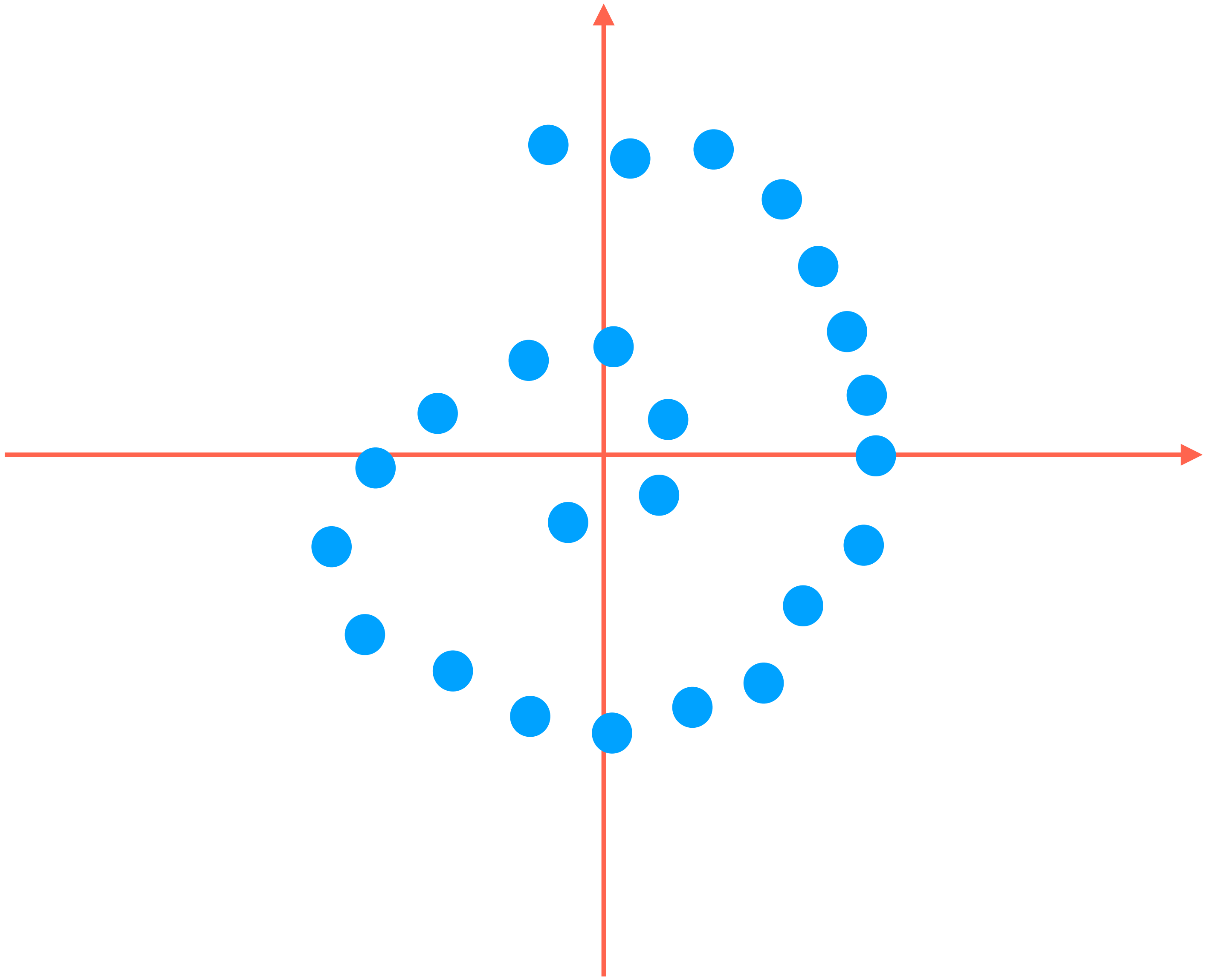
Principal Component Analysis (PCA)

Demo

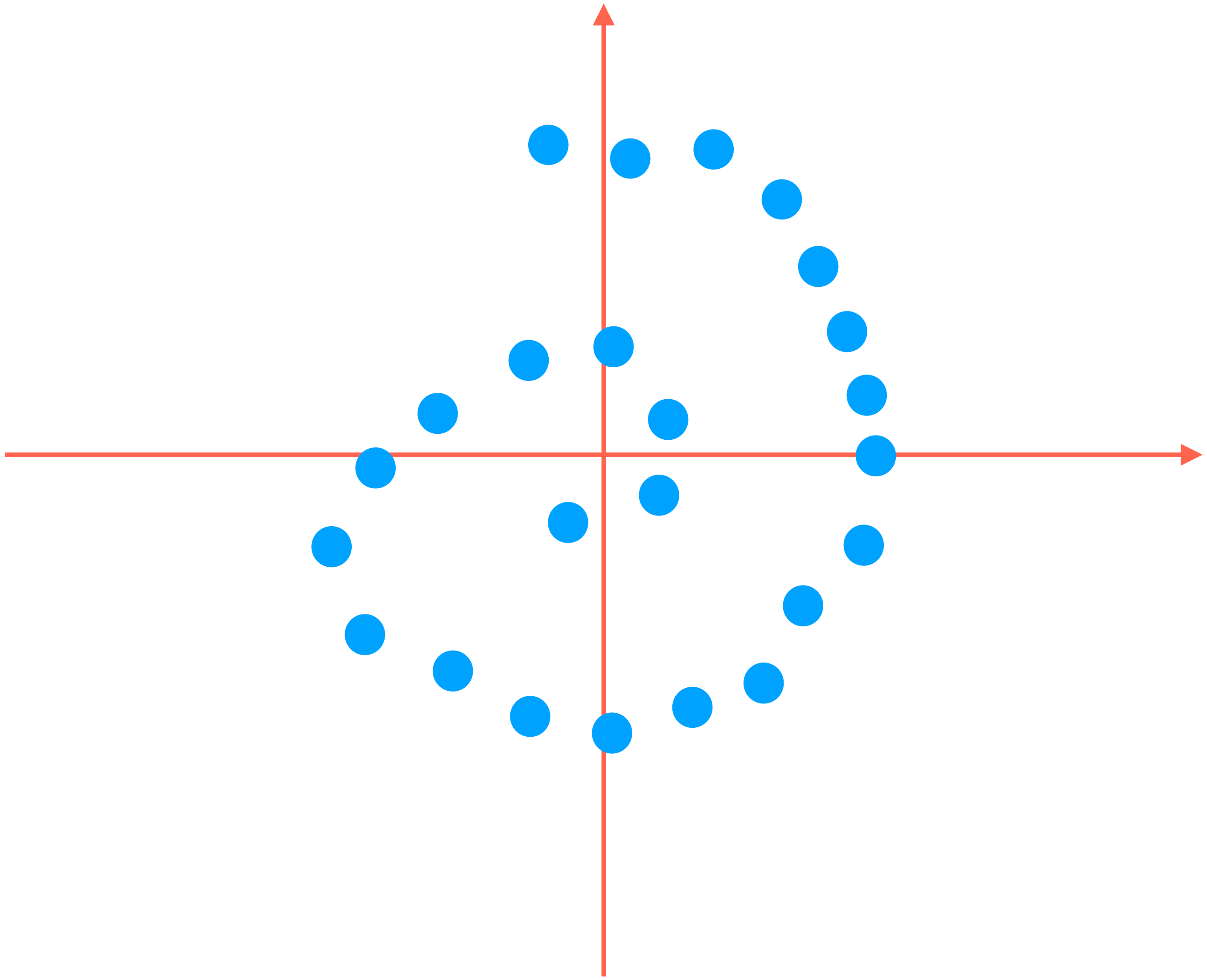
PCA reorients data so axes explain variance in "decreasing order"
➔ can "flatten" (*project*) data onto a few axes that captures most variance

Image source: http://4.bp.blogspot.com/-USQEgoh1jCU/VfncdNOETcI/AAAAAAAAGp8/Hea8UtE_1c0/s1600/Blog%2B1%2BIMG_1821.jpg

2D Swiss Roll



2D Swiss Roll



PCA would just flatten this thing and
lose the information that the data actually lives on a 1D line that has been curved!

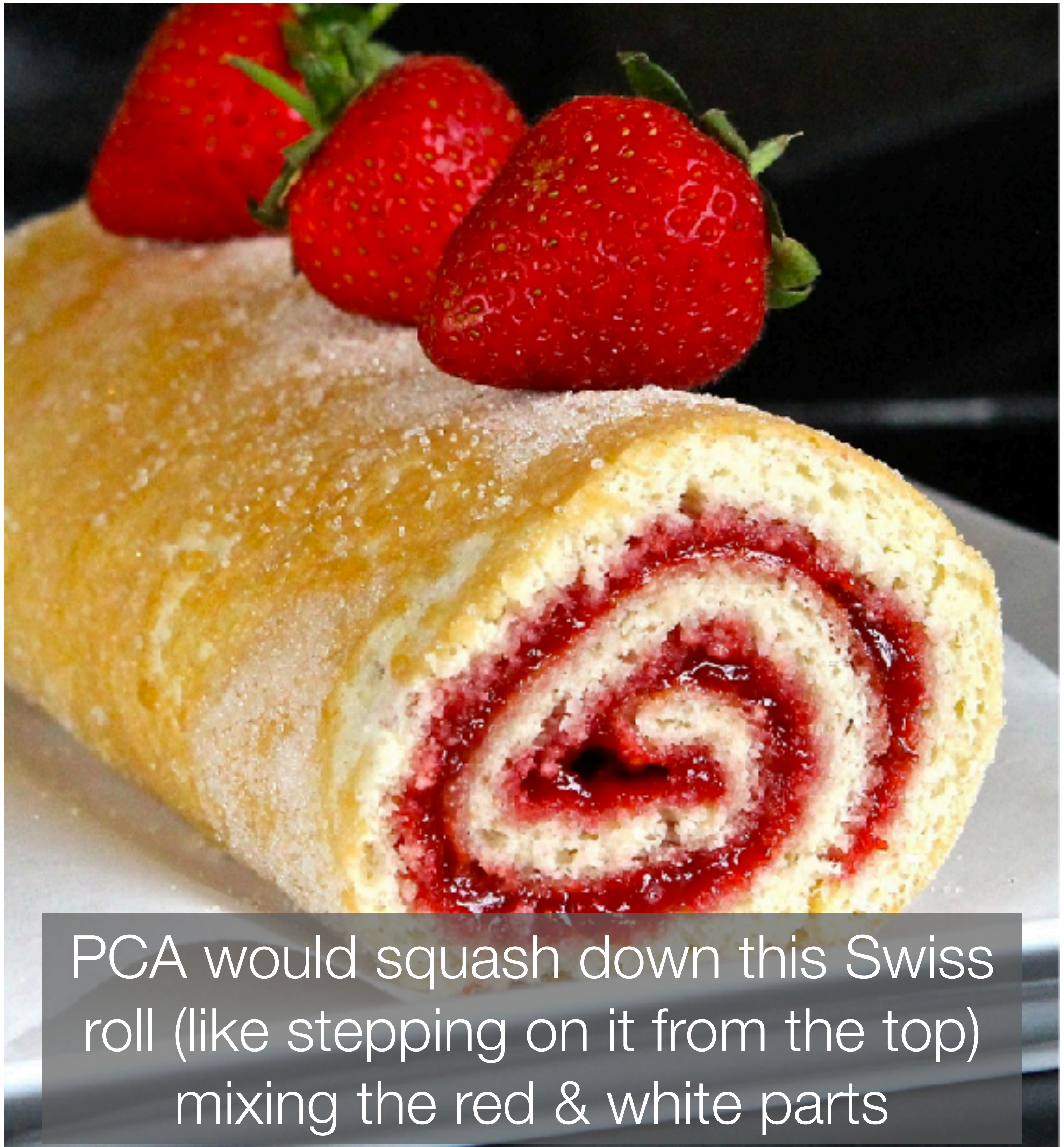
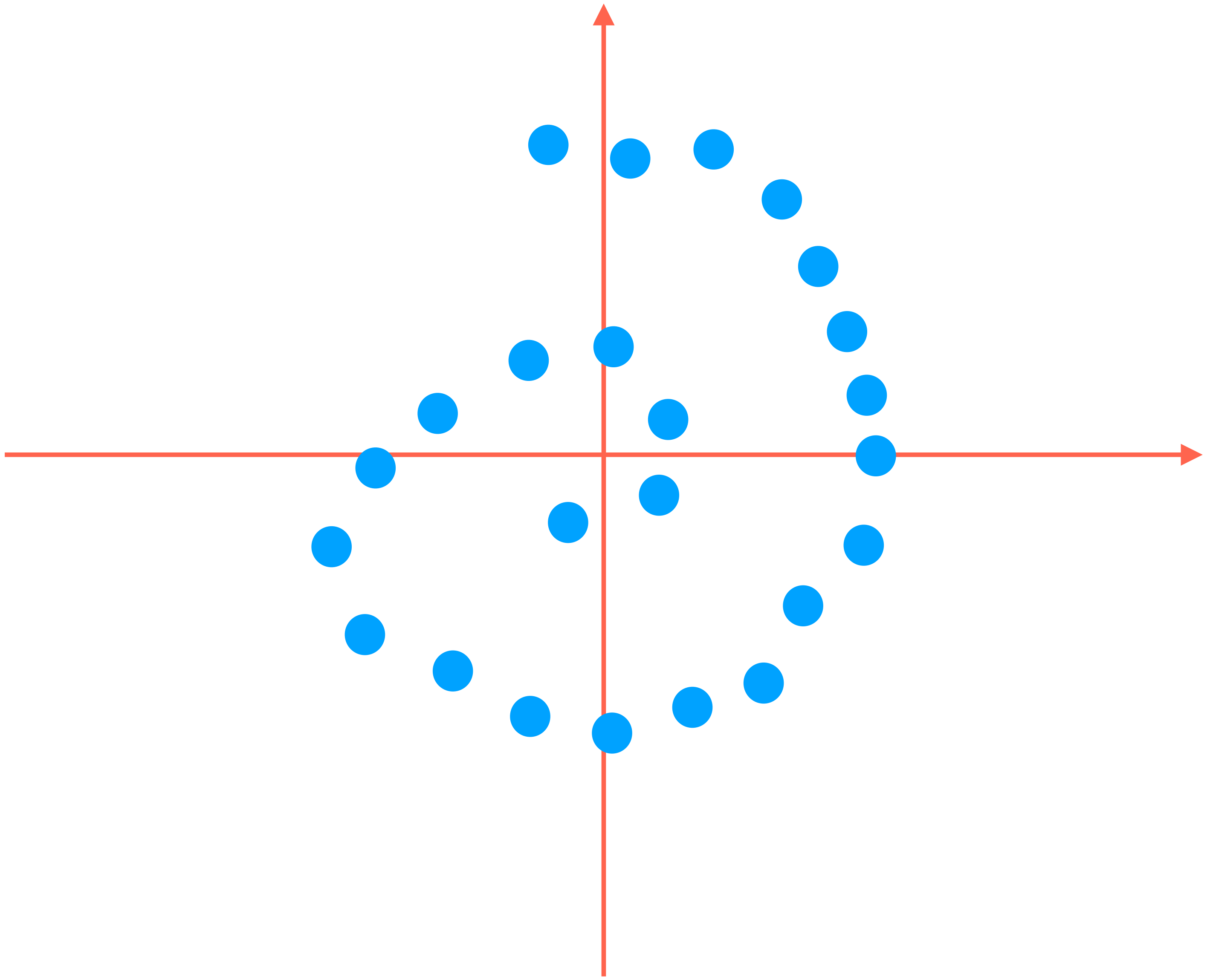
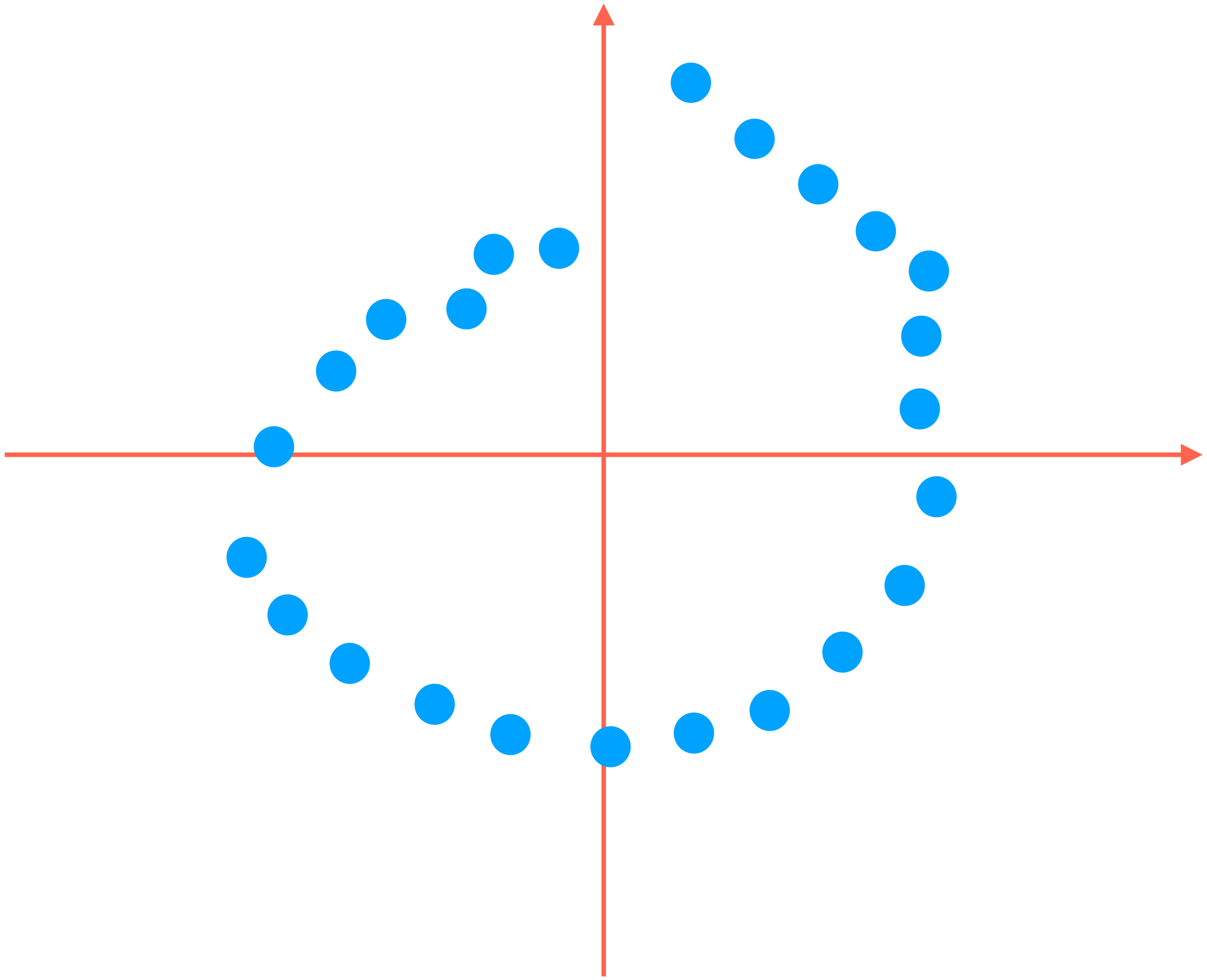


Image source: http://4.bp.blogspot.com/-USQEgoh1jCU/VfncdNOETcI/AAAAAAAAGp8/Hea8UtE_1c0/s1600/Blog%2B1%2BIMG_1821.jpg

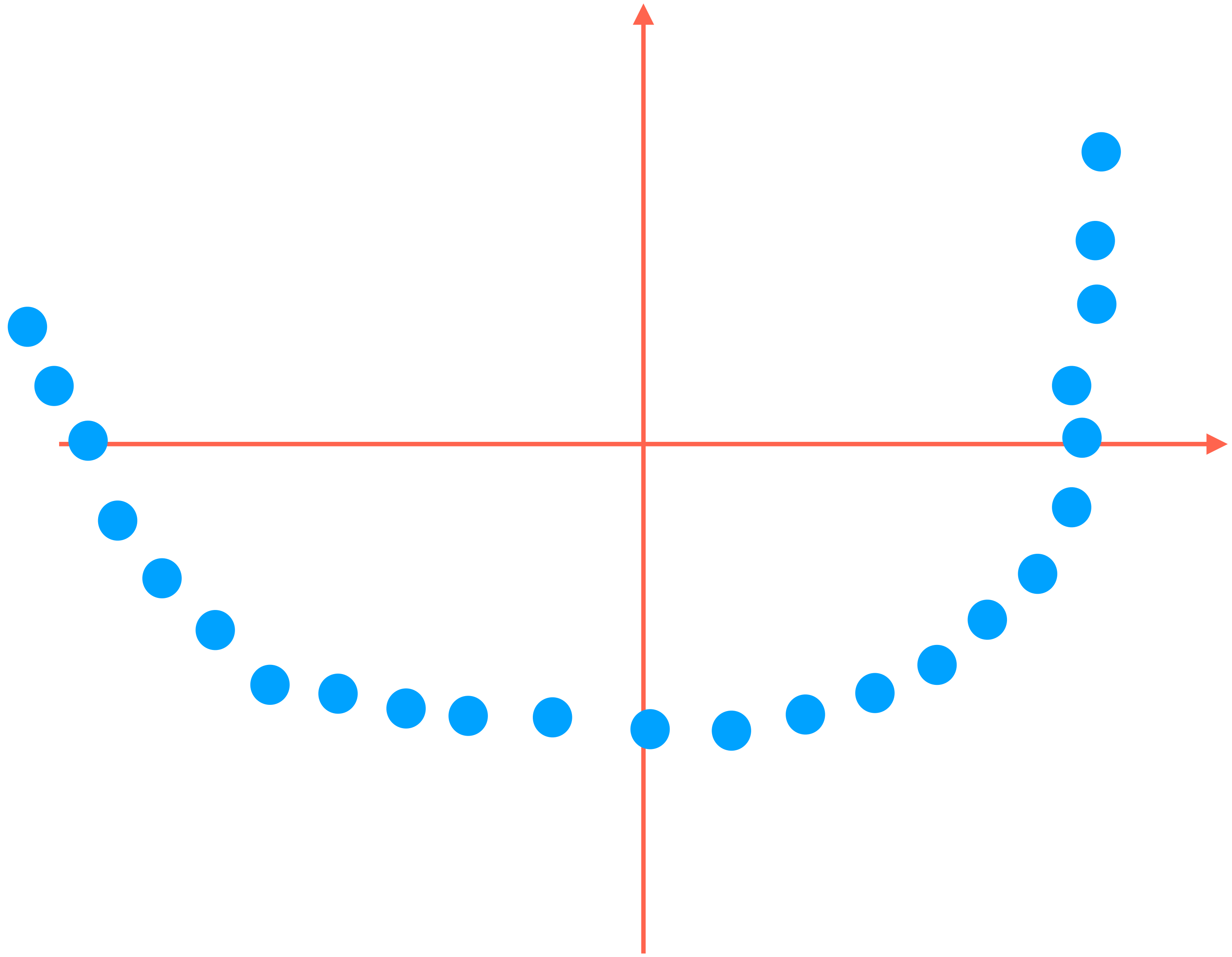
2D Swiss Roll



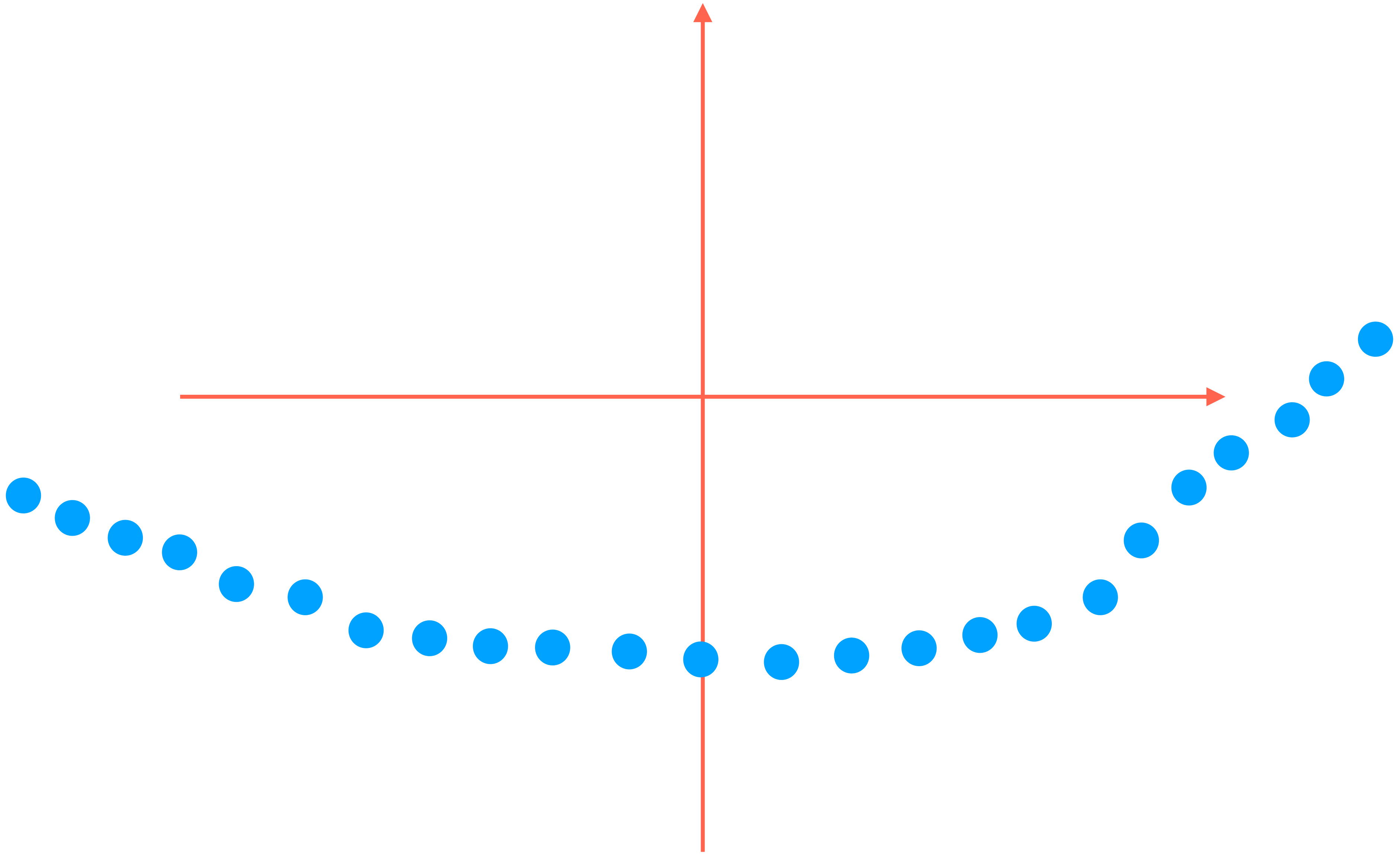
2D Swiss Roll



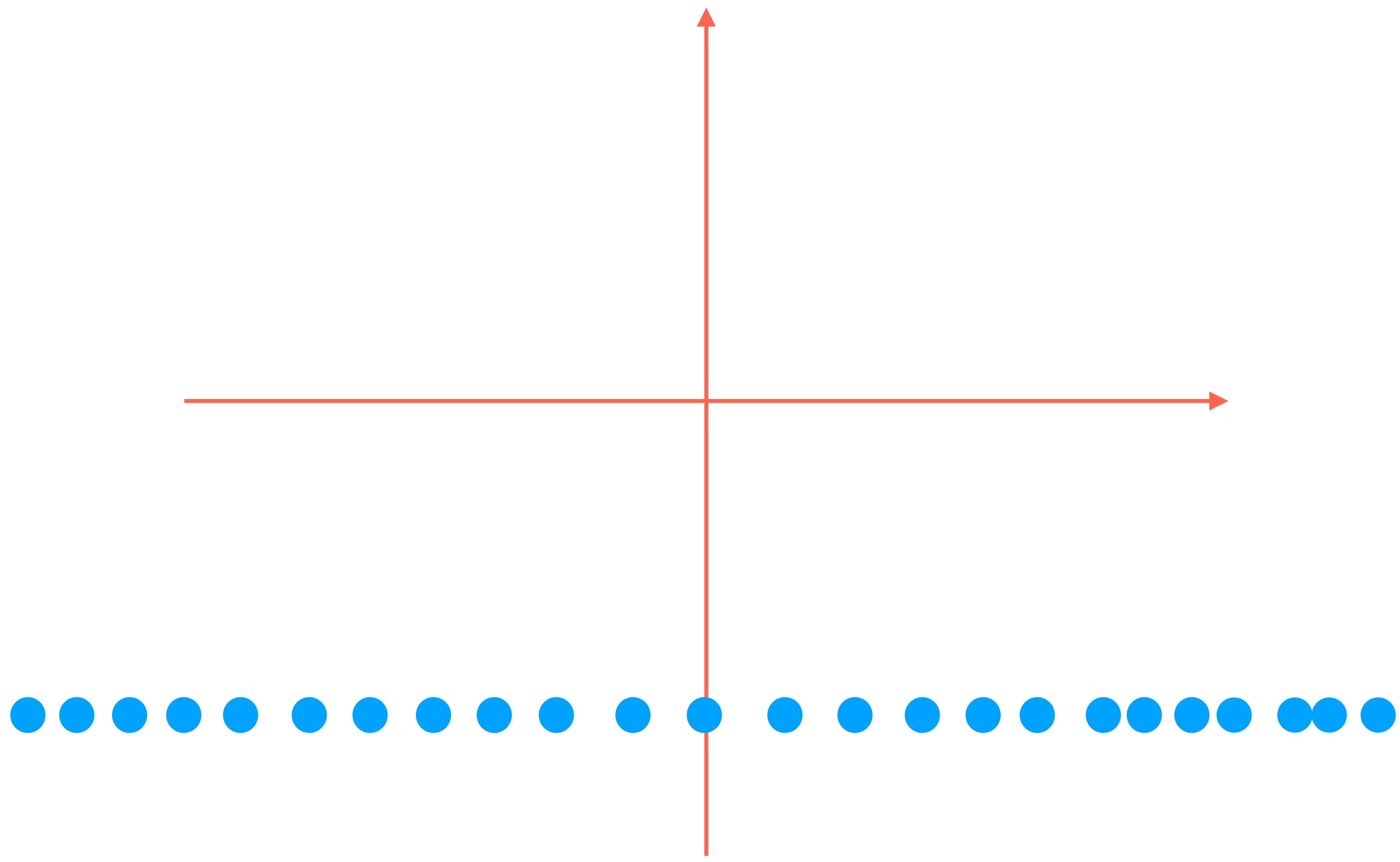
2D Swiss Roll



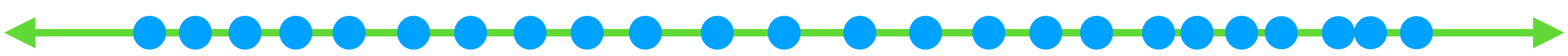
2D Swiss Roll



2D Swiss Roll

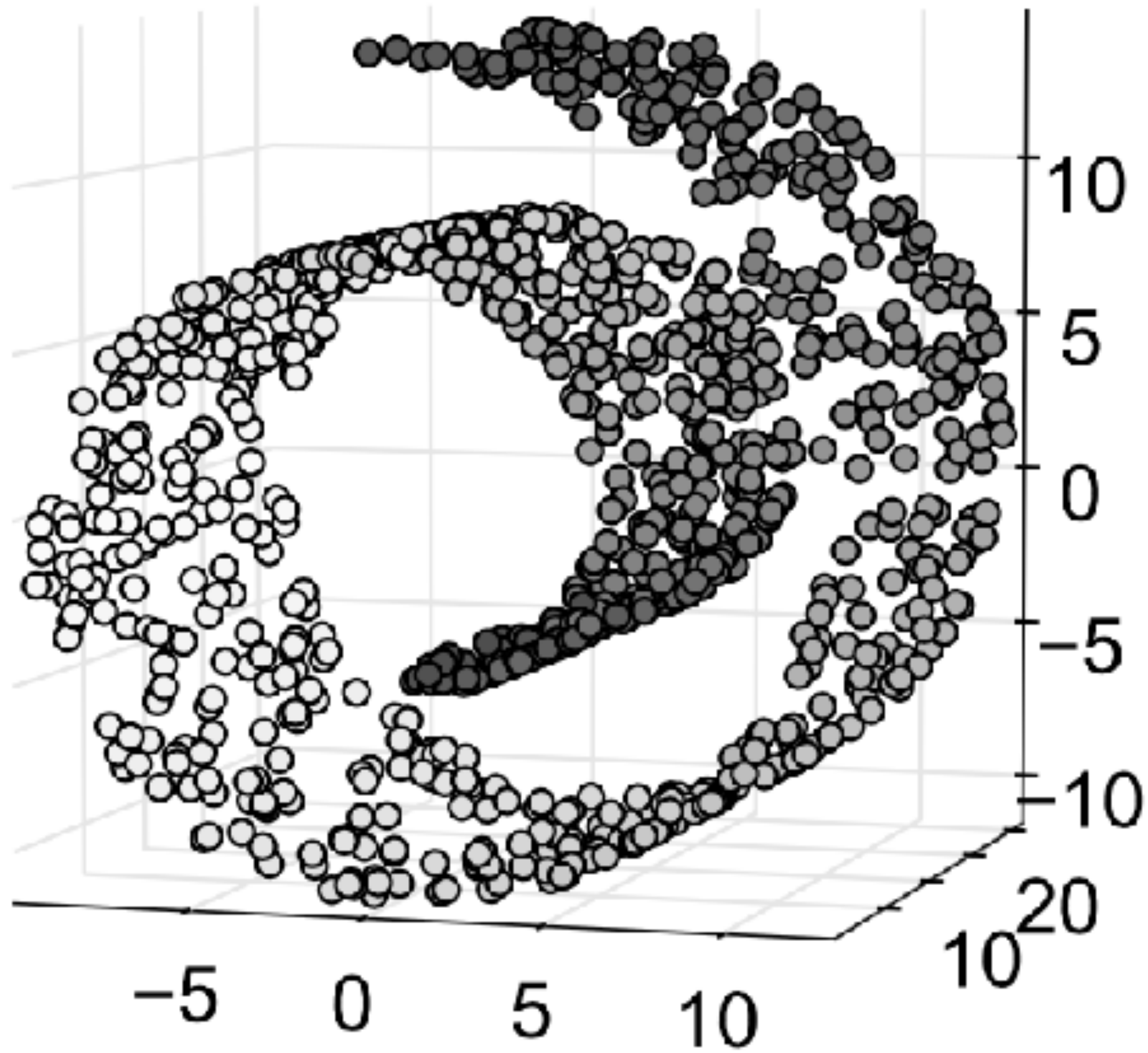


2D Swiss Roll



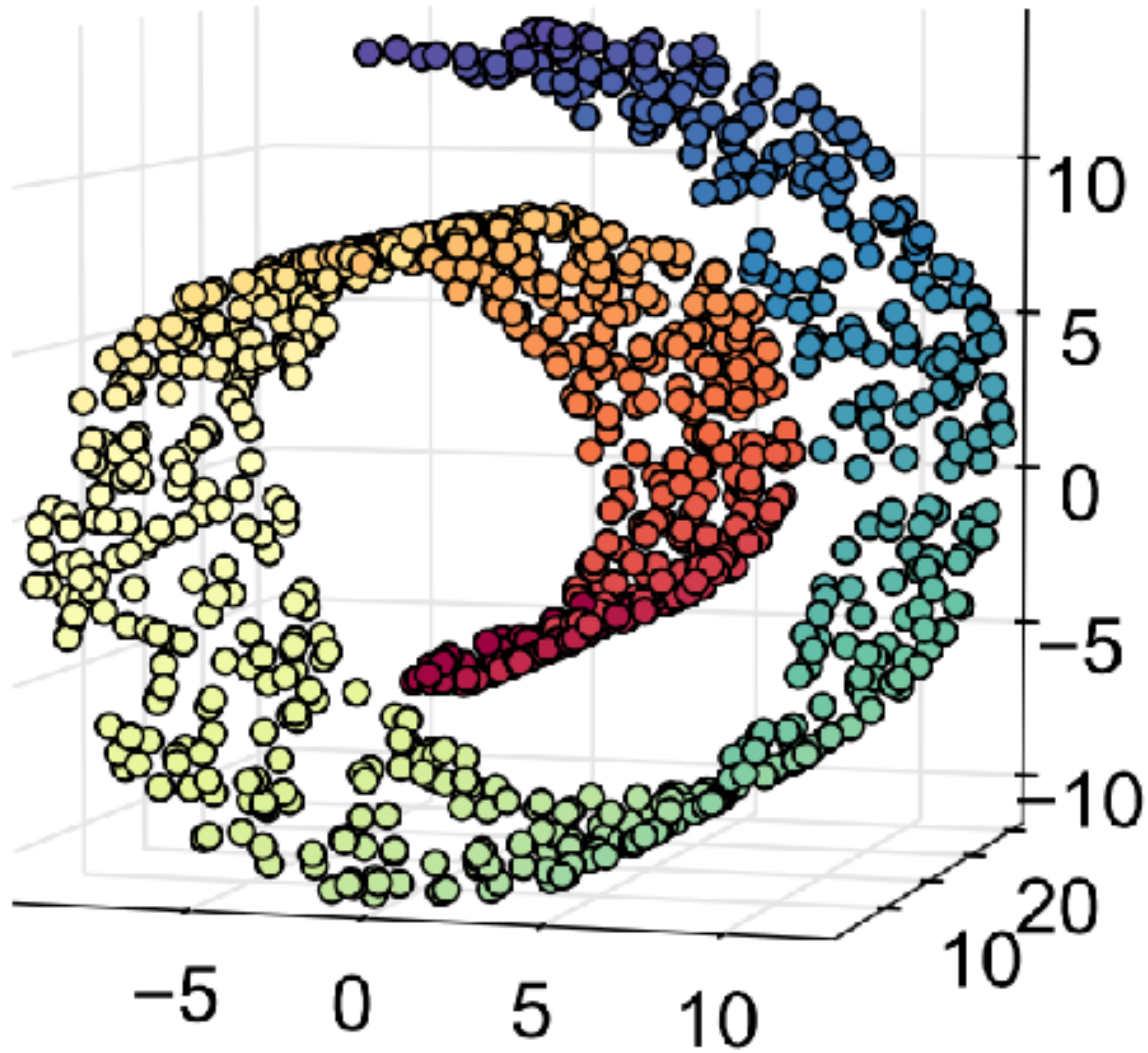
This is the desired result

3D Swiss Roll



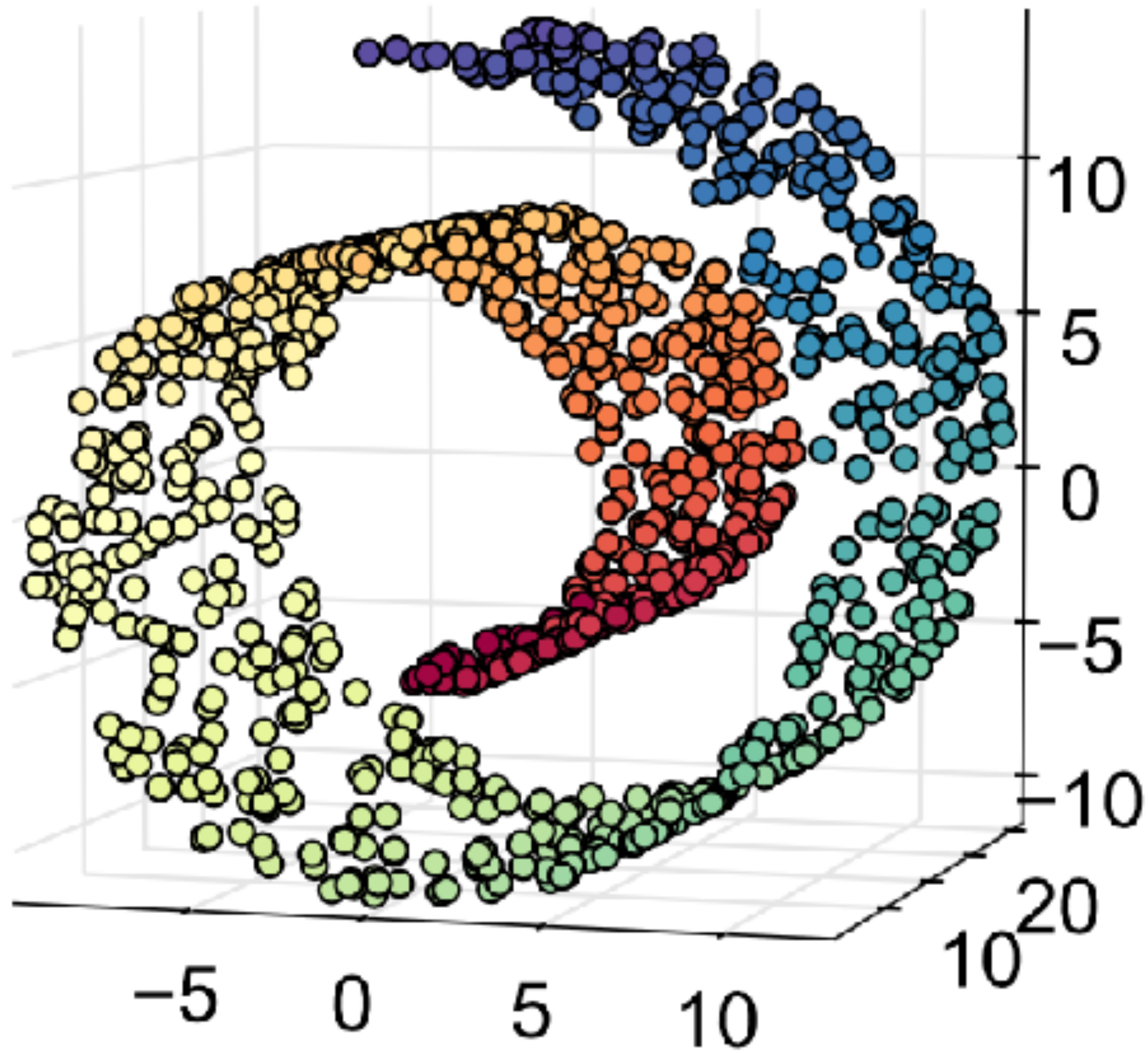
Projecting down to any 2D plane puts points
that are far apart close together!

3D Swiss Roll



Projecting down to any 2D plane puts points
that are far apart close together!

3D Swiss Roll



Projecting down to any 2D plane puts points that are far apart close together!

Goal: Low-dimensional representation where similar colored points are near each other (we don't actually get to see the colors)

Manifold Learning

Manifold Learning

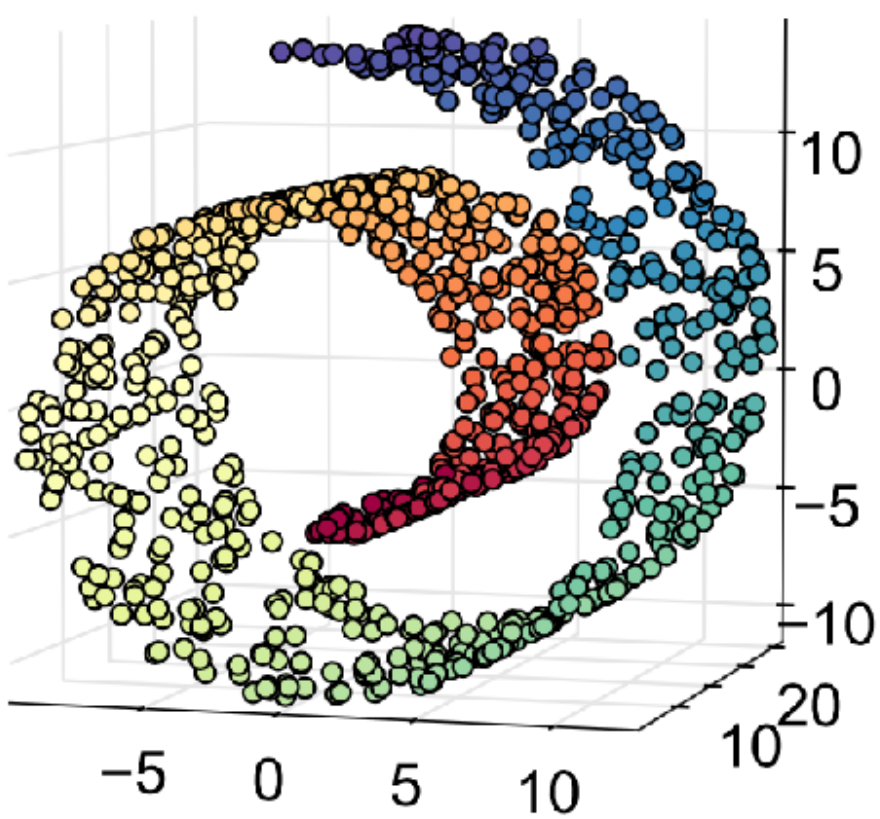
- Nonlinear dimensionality reduction (in contrast to PCA which is linear)

Manifold Learning

- Nonlinear dimensionality reduction (in contrast to PCA which is linear)
- Find low-dimensional “manifold” that the data live on

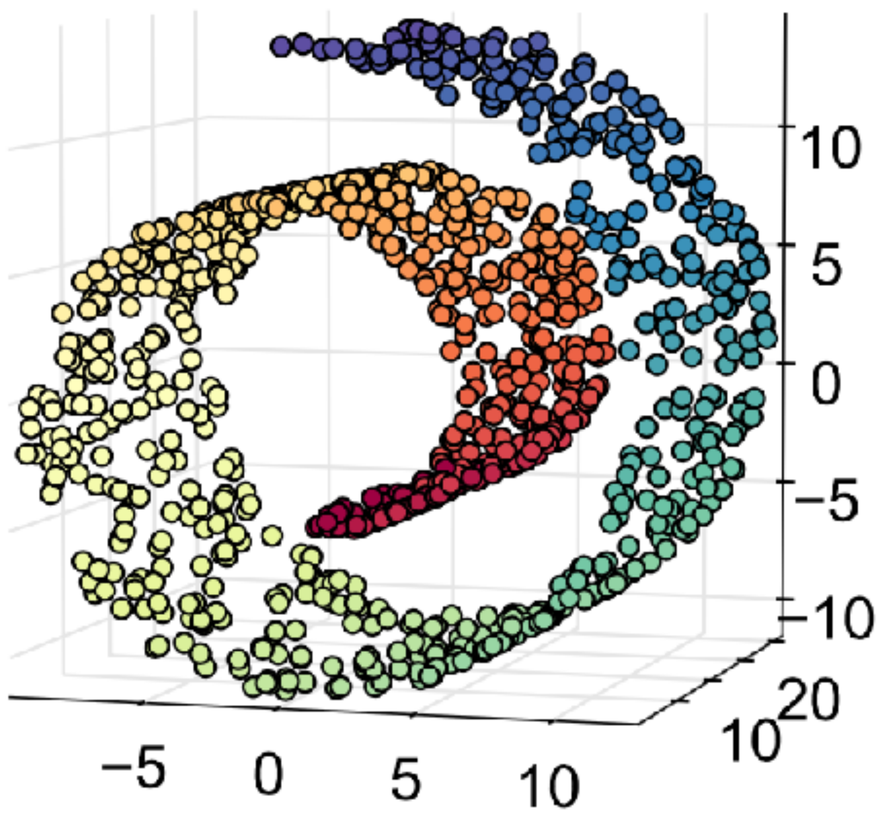
Manifold Learning

- Nonlinear dimensionality reduction (in contrast to PCA which is linear)
- Find low-dimensional "manifold" that the data live on



Manifold Learning

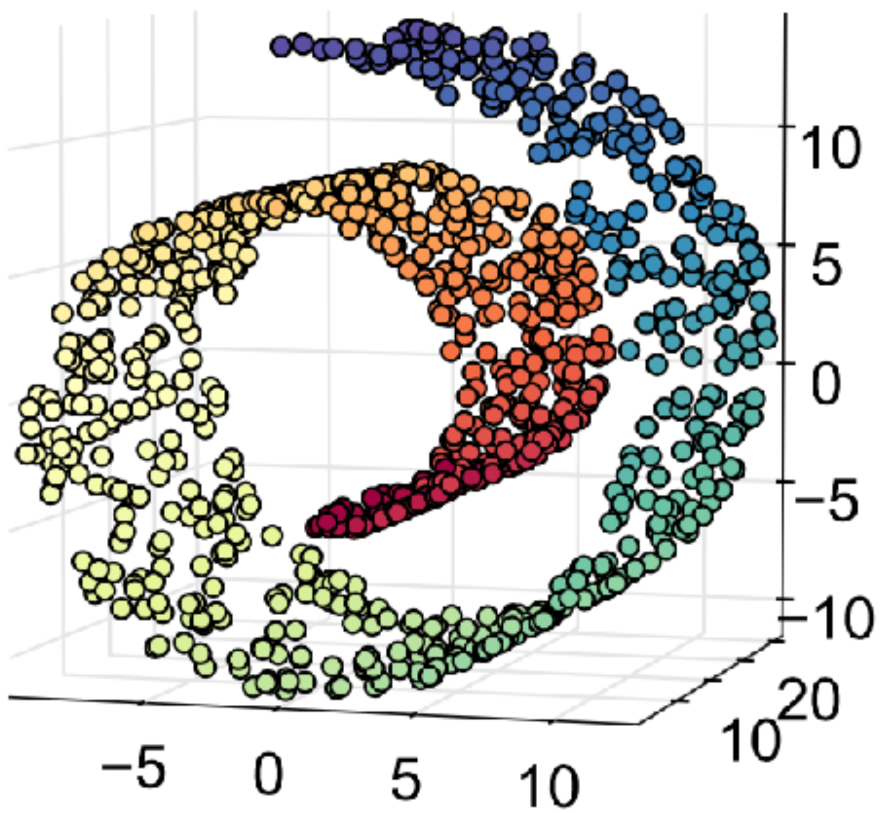
- Nonlinear dimensionality reduction (in contrast to PCA which is linear)
- Find low-dimensional "manifold" that the data live on



Basic idea of a manifold:

Manifold Learning

- Nonlinear dimensionality reduction (in contrast to PCA which is linear)
- Find low-dimensional "manifold" that the data live on

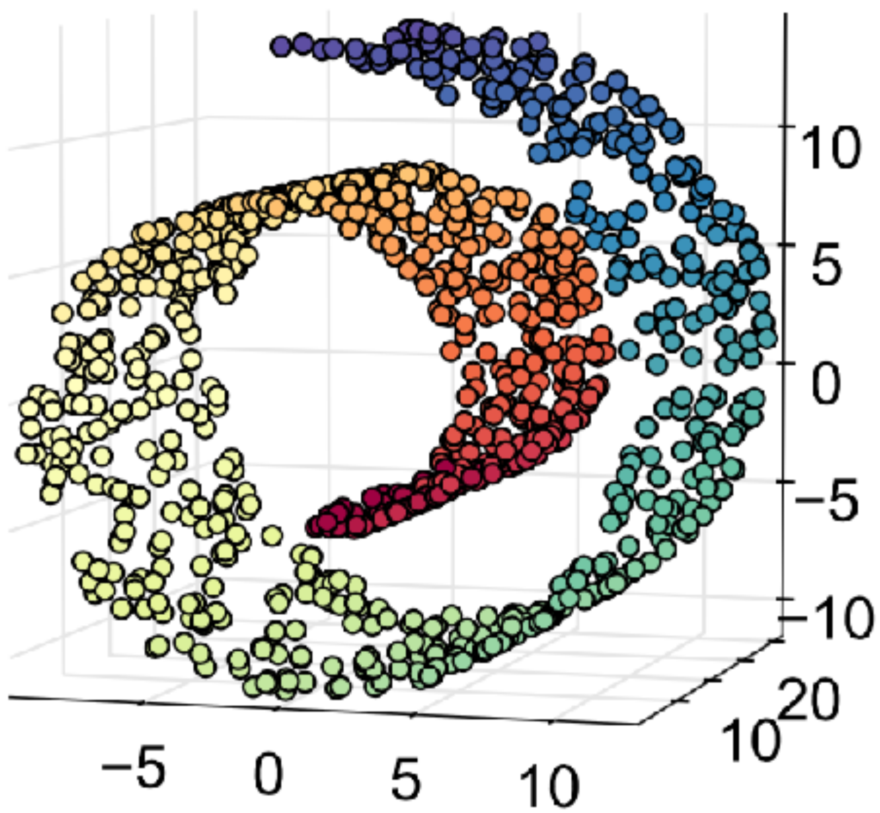


Basic idea of a manifold:

1. Zoom in on any point (say, *x*)

Manifold Learning

- Nonlinear dimensionality reduction (in contrast to PCA which is linear)
- Find low-dimensional "manifold" that the data live on



Basic idea of a manifold:

1. Zoom in on any point (say, *x*)

2. The points near *x* look like they're in a lower-dimensional Euclidean space (e.g., a 2D plane in Swiss roll)

Do Data Actually Live on Manifolds?

Do Data Actually Live on Manifolds?

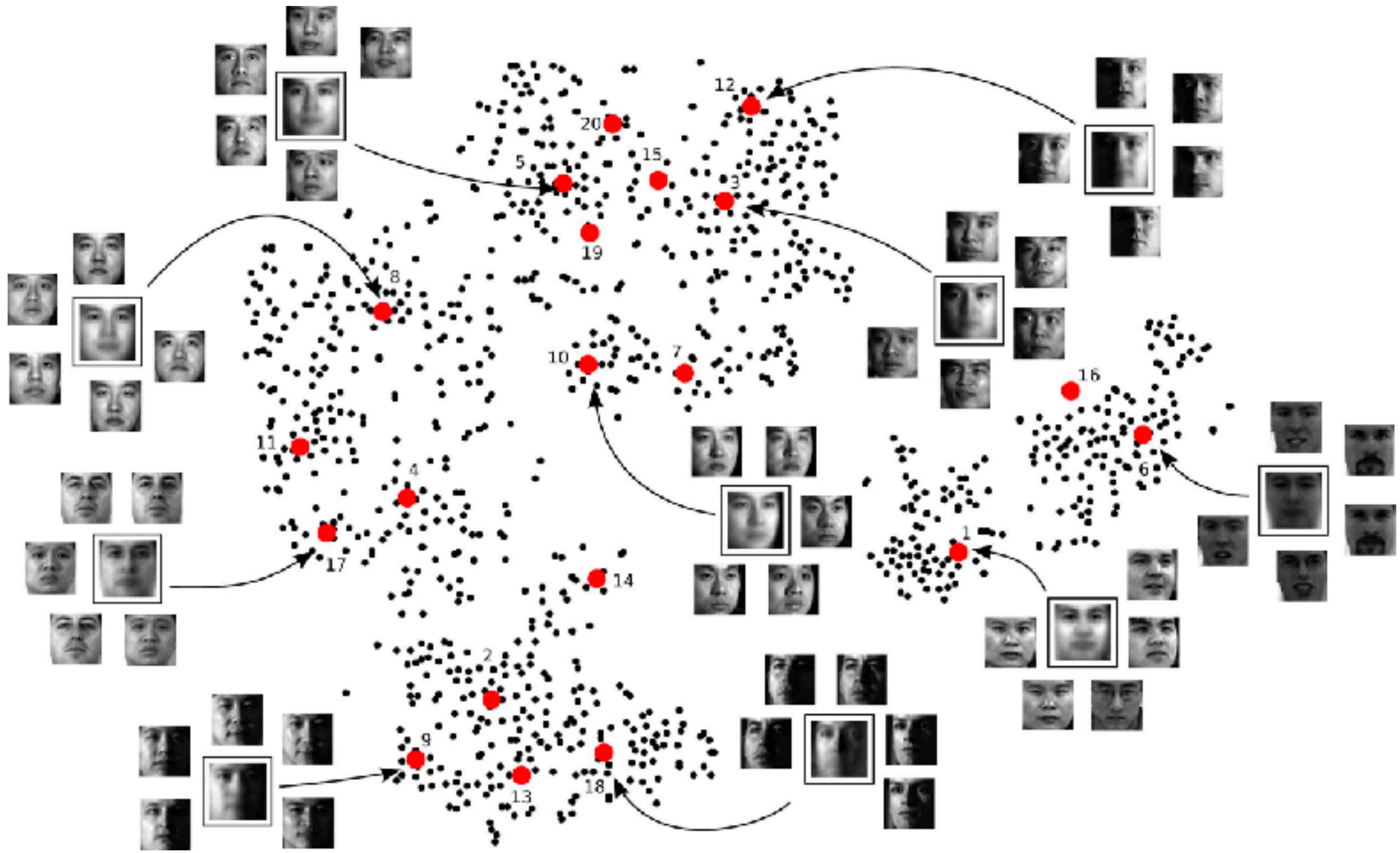


Image source: http://www.columbia.edu/~jwp2128/Images/faces.jpeg

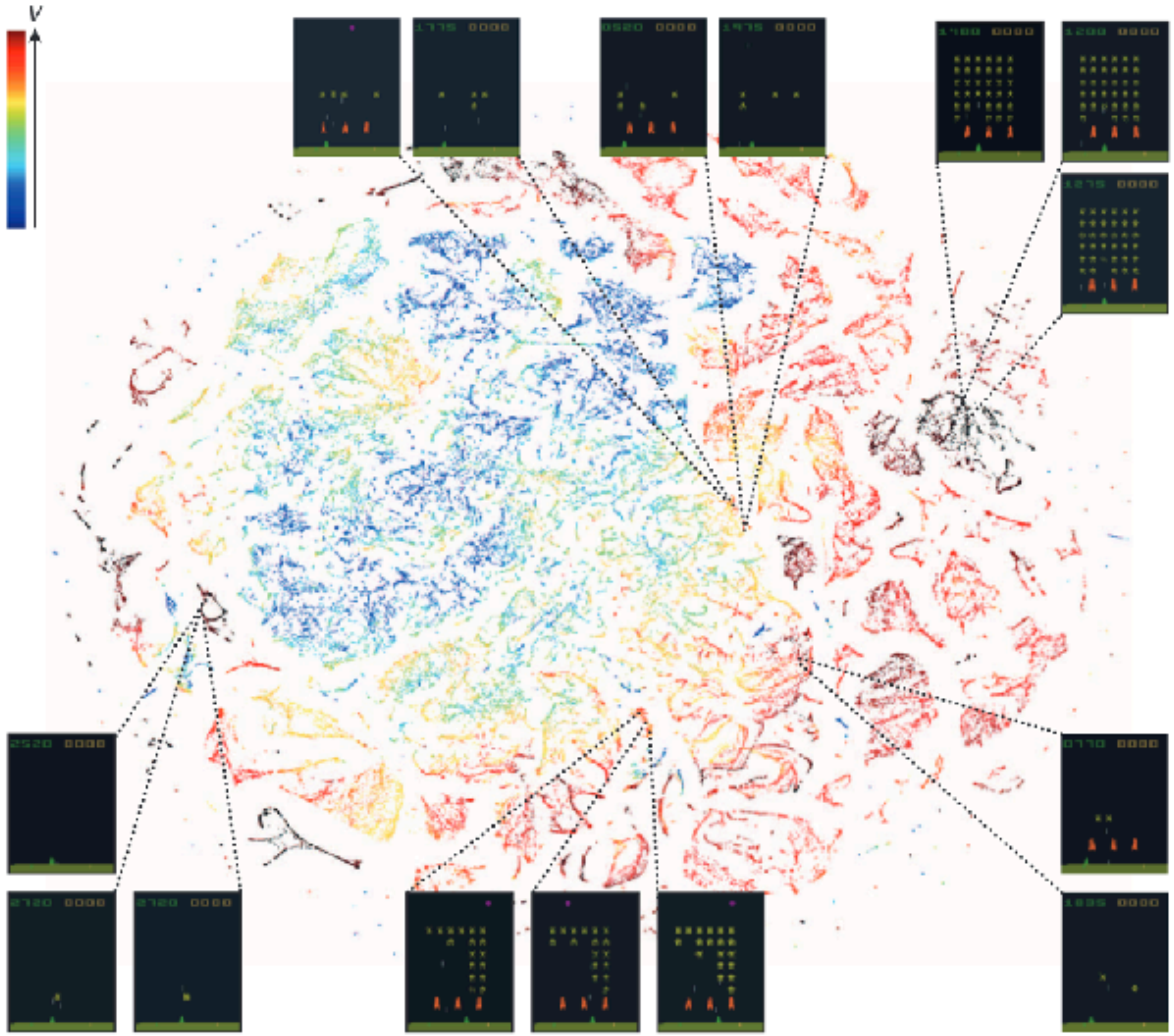
Do Data Actually Live on Manifolds?

Phillip Isola, Joseph Lim, Edward H. Adelson. Discovering States and Transformations in Image Collections. CVPR 2015.

Do Data Actually Live on Manifolds?

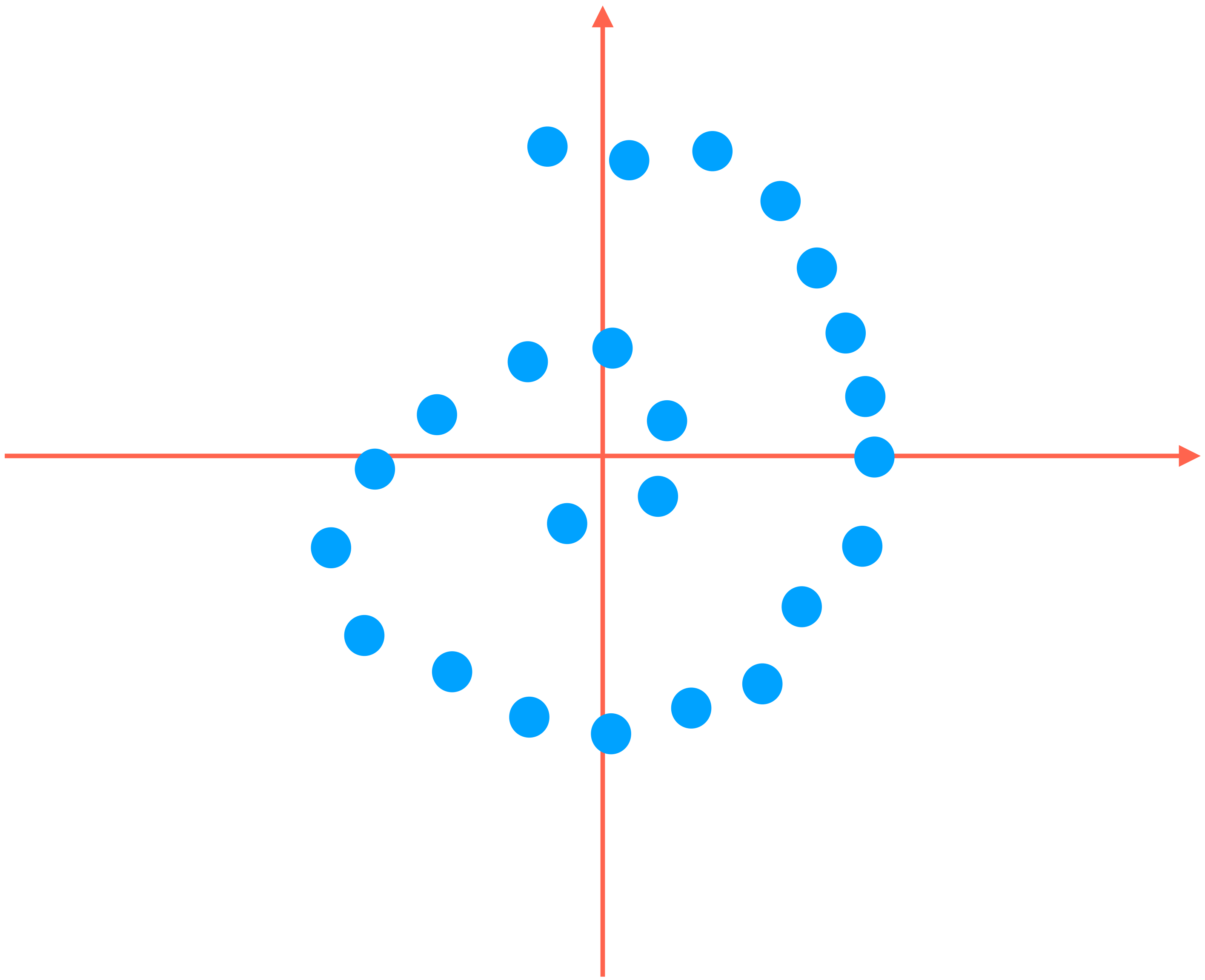
Image source: http://www.adityathakker.com/wp-content/uploads/2017/06/word-embeddings-994x675.png

Do Data Actually Live on Manifolds?



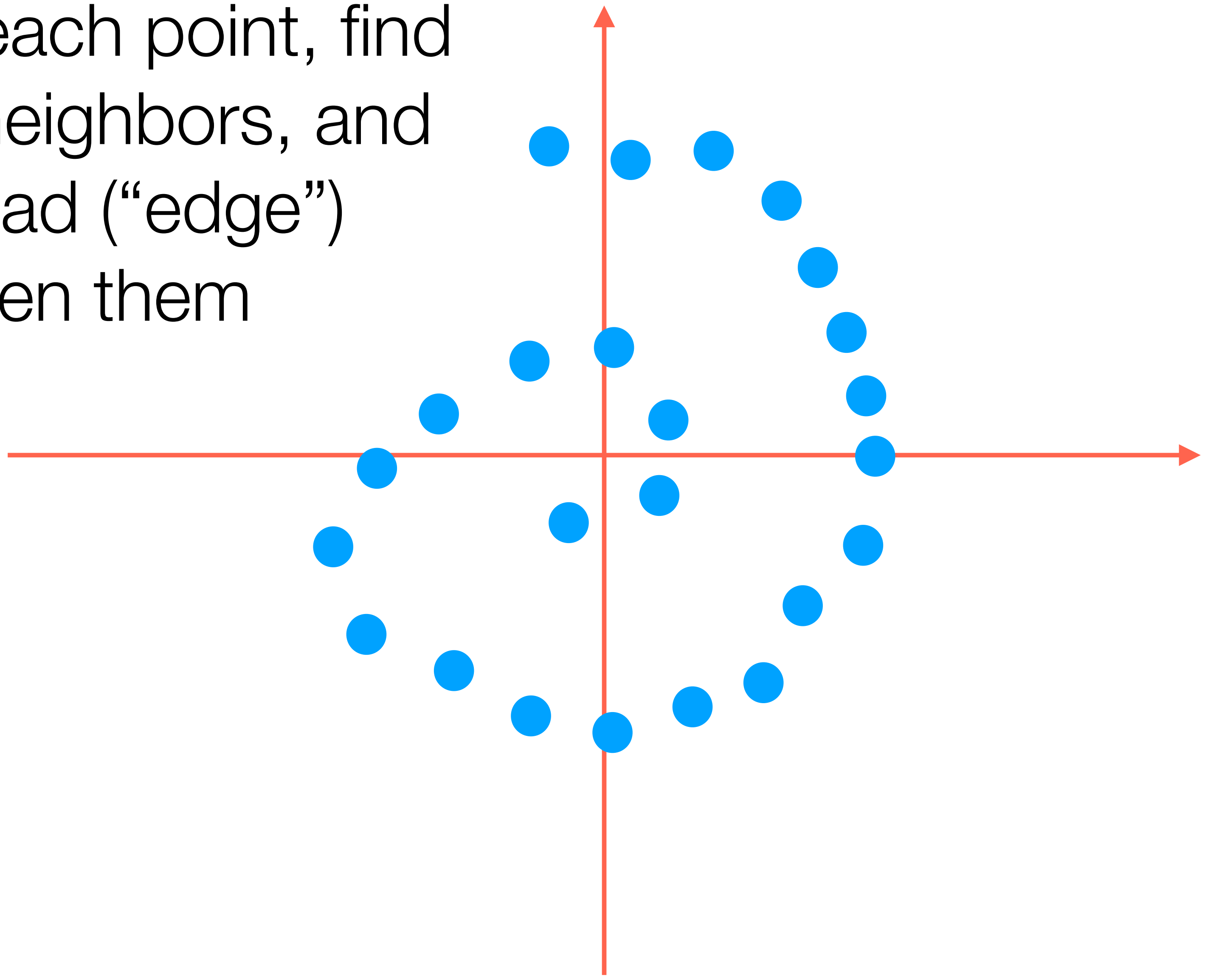
Mnih, Volodymyr, et al. Human-level control through deep reinforcement learning. Nature 2015.

Manifold Learning with Isomap



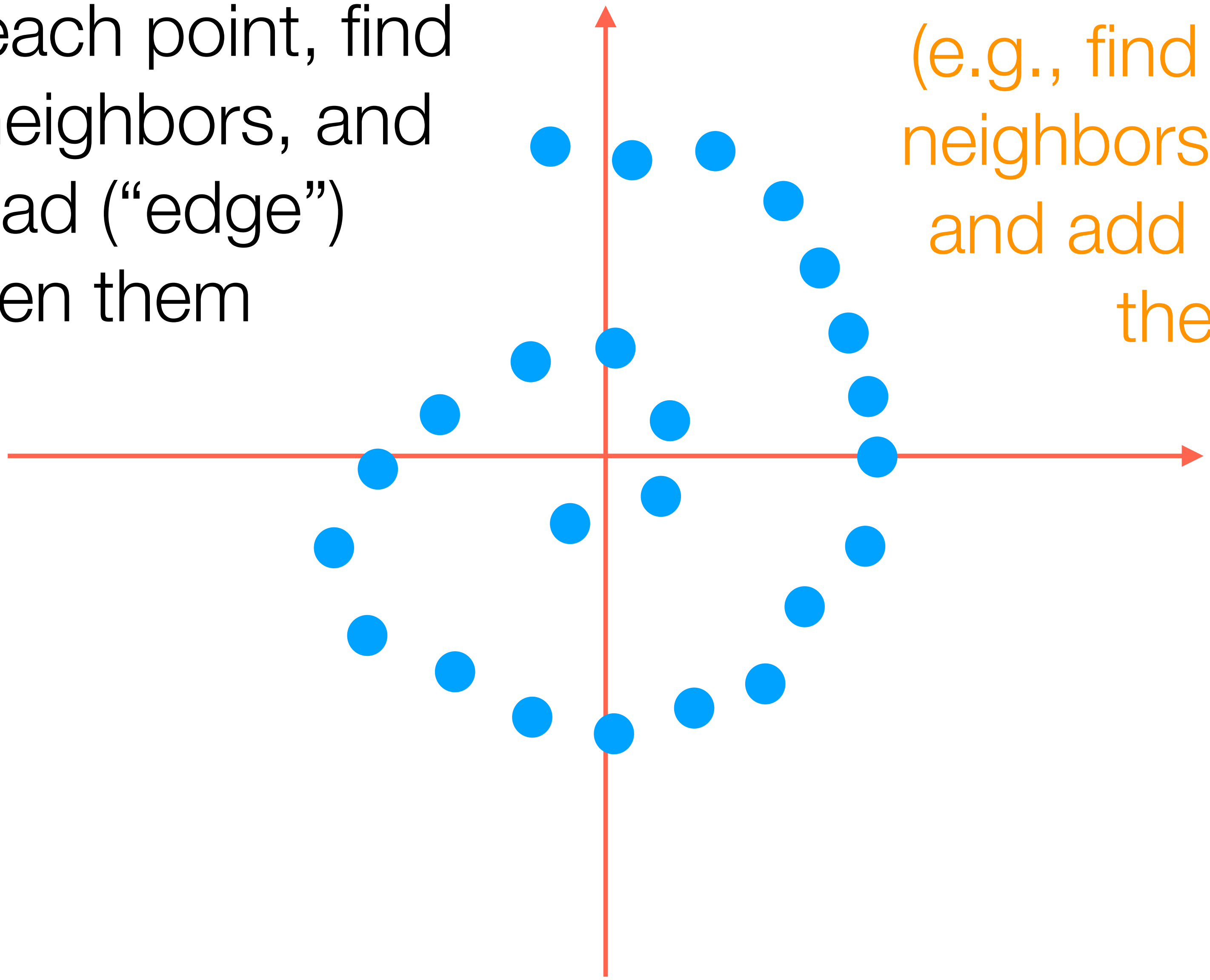
Manifold Learning with Isomap

Step 1: For each point, find its nearest neighbors, and build a road (“edge”) between them



Manifold Learning with Isomap

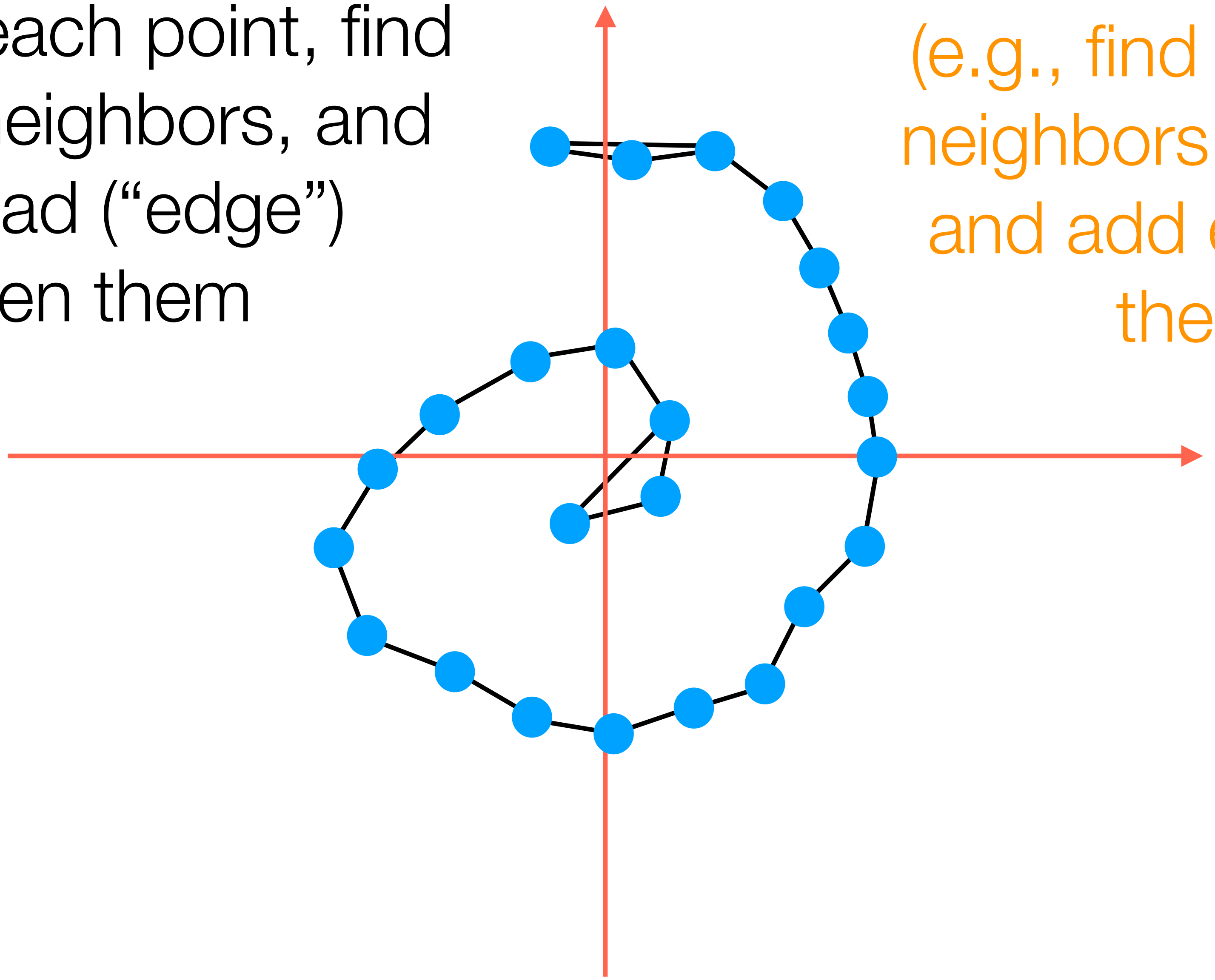
Step 1: For each point, find its nearest neighbors, and build a road ("edge") between them



(e.g., find closest 2 neighbors per point and add edges to them)

Manifold Learning with Isomap

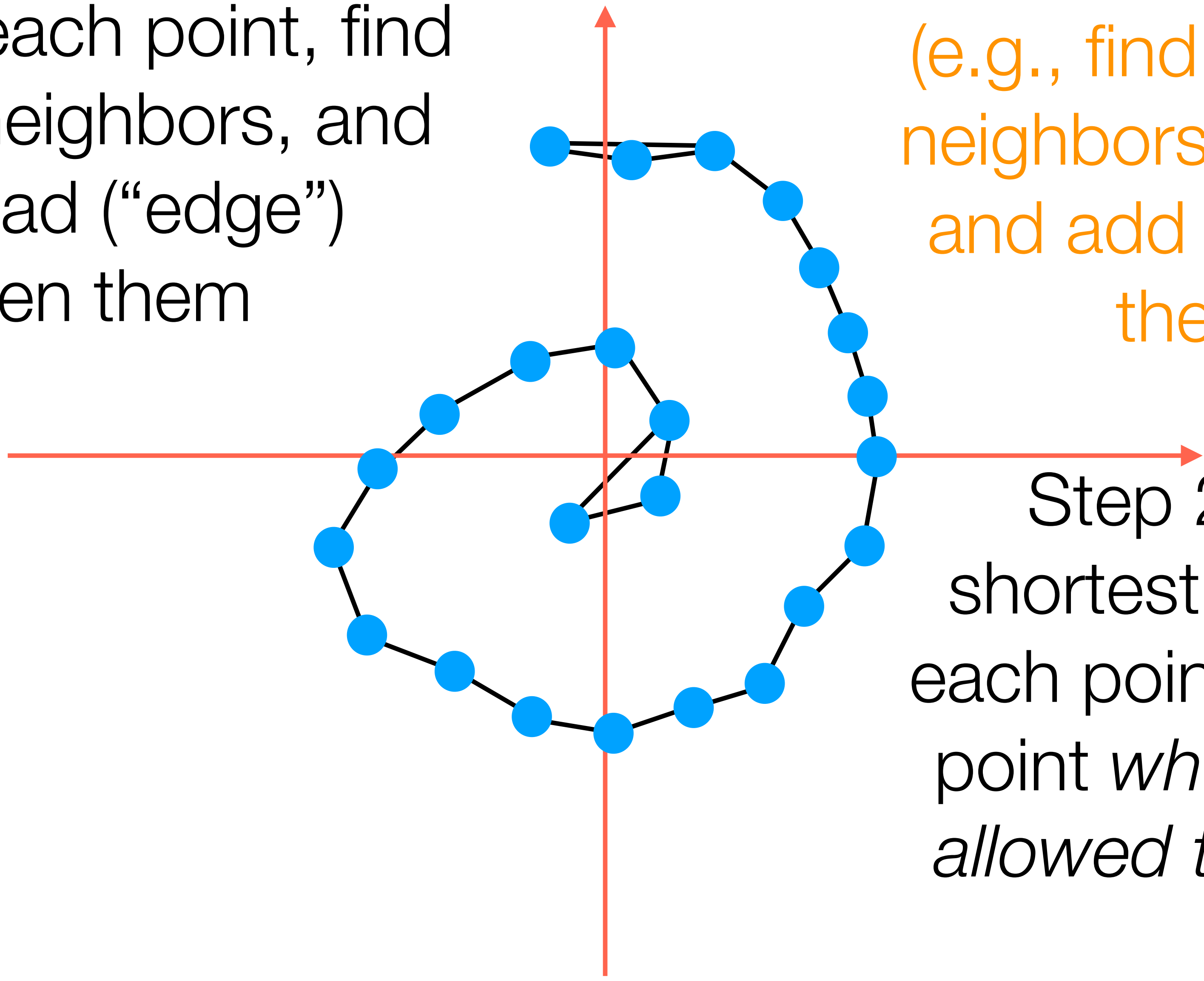
Step 1: For each point, find its nearest neighbors, and build a road ("edge") between them



(e.g., find closest 2 neighbors per point and add edges to them)

Manifold Learning with Isomap

Step 1: For each point, find its nearest neighbors, and build a road ("edge") between them

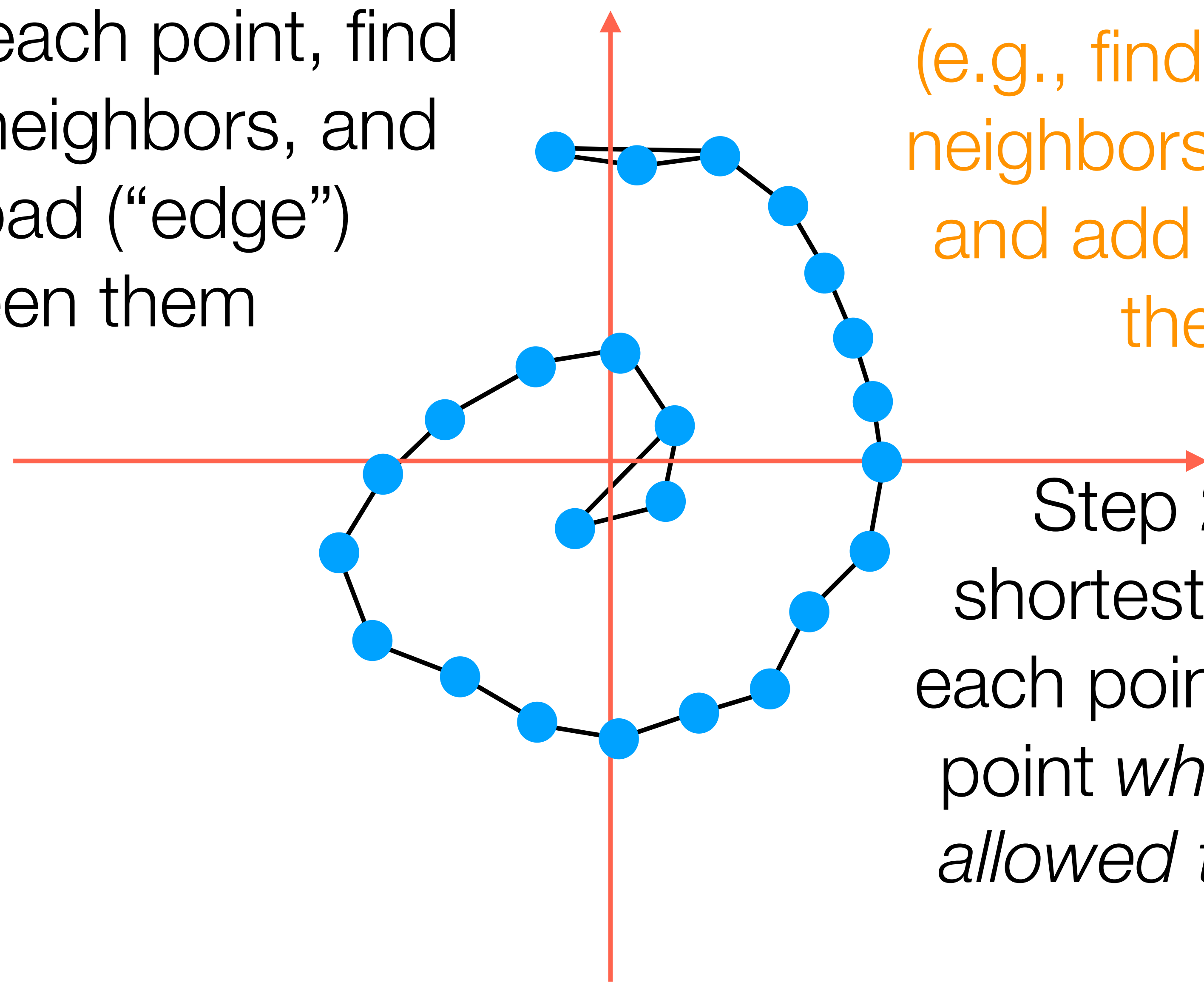


(e.g., find closest 2 neighbors per point and add edges to them)

Step 2: Compute shortest distance from each point to every other point *where you're only allowed to travel on the roads*

Manifold Learning with Isomap

Step 1: For each point, find its nearest neighbors, and build a road (“edge”) between them



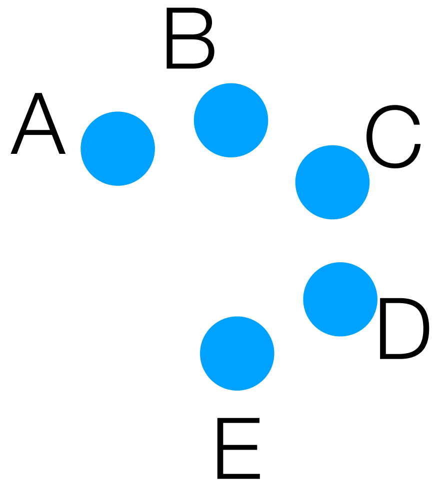
(e.g., find closest 2 neighbors per point and add edges to them)

Step 2: Compute shortest distance from each point to every other point *where you’re only allowed to travel on the roads*

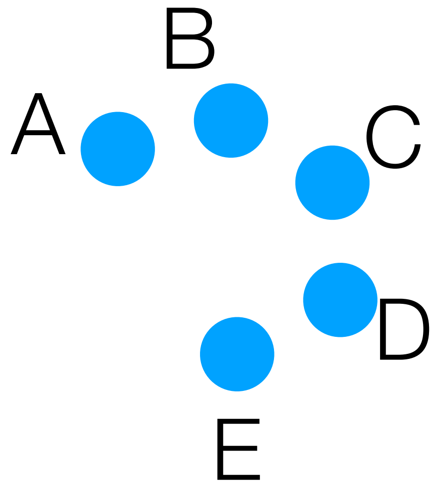
Step 3: It turns out that given all the distances between pairs of points, we can compute what the points should be (the algorithm for this is called *multidimensional scaling*)

Isomap Calculation Example

Isomap Calculation Example

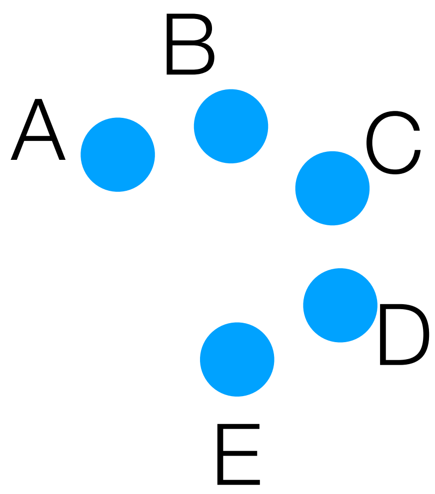


Isomap Calculation Example



Build "symmetric 2-NN" graph
(add edges for each point to
its 2 nearest neighbors)

Isomap Calculation Example

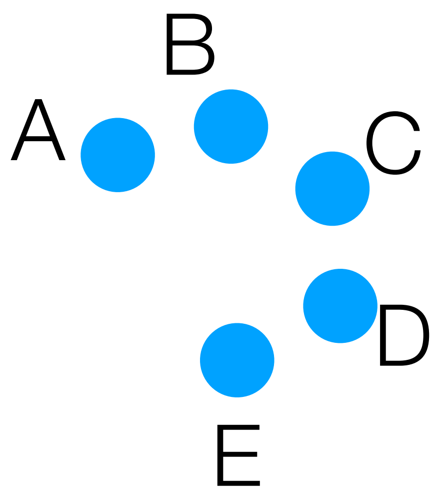


2 nearest neighbors of A:

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

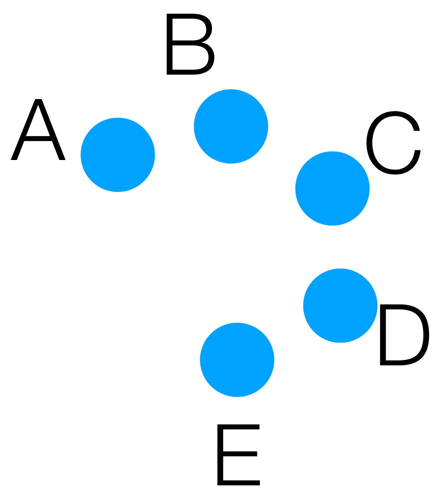
Isomap Calculation Example

2 nearest neighbors of A: B, C



Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example

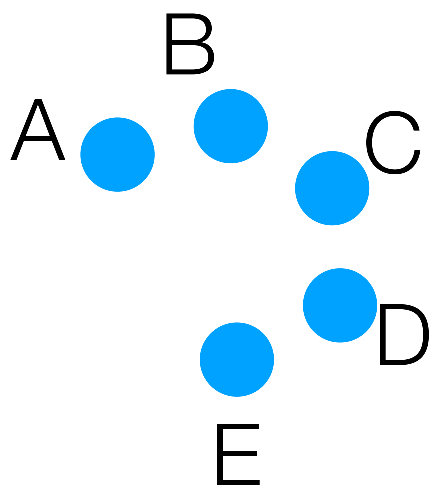


2 nearest neighbors of A: B, C

2 nearest neighbors of B:

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example

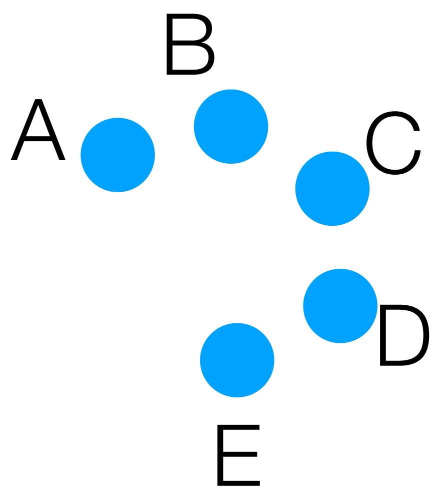


2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

Build "symmetric 2-NN" graph
(add edges for each point to
its 2 nearest neighbors)

Isomap Calculation Example



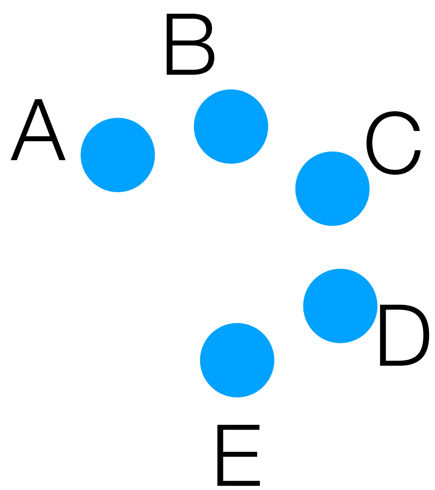
2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C:

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example



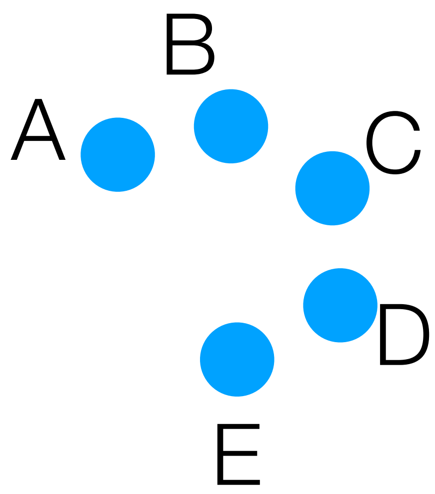
2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example



2 nearest neighbors of A: B, C

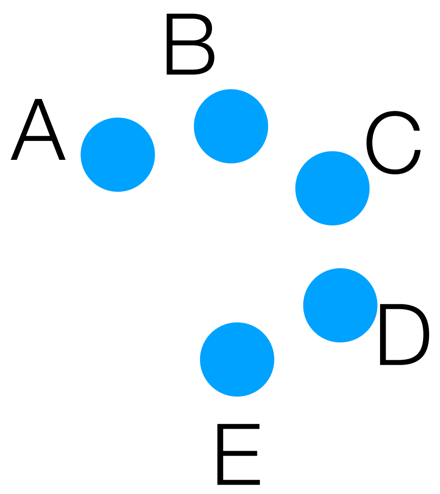
2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D:

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example



2 nearest neighbors of A: B, C

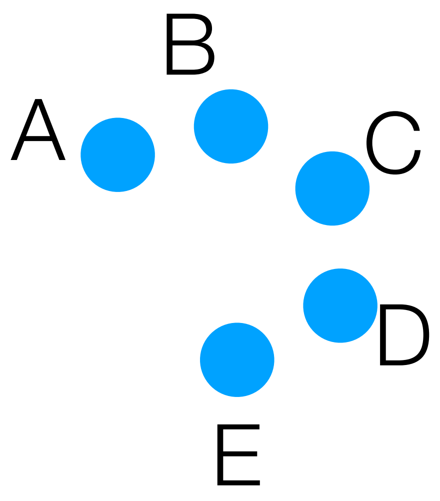
2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

Build "symmetric 2-NN" graph
(add edges for each point to
its 2 nearest neighbors)

Isomap Calculation Example



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

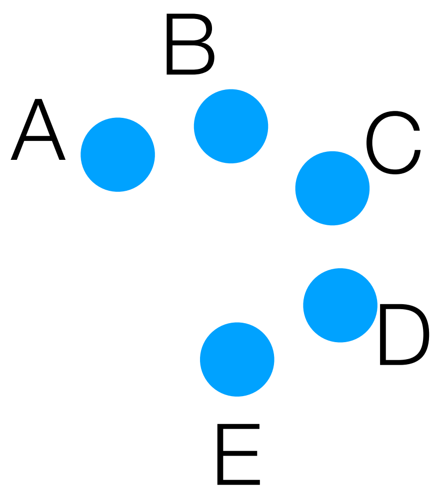
2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E:

Build "symmetric 2-NN" graph
(add edges for each point to
its 2 nearest neighbors)

Isomap Calculation Example



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

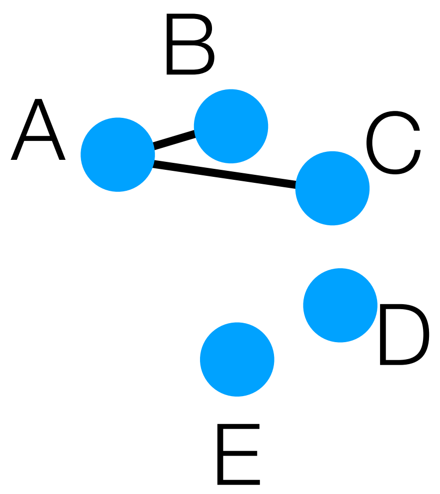
2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

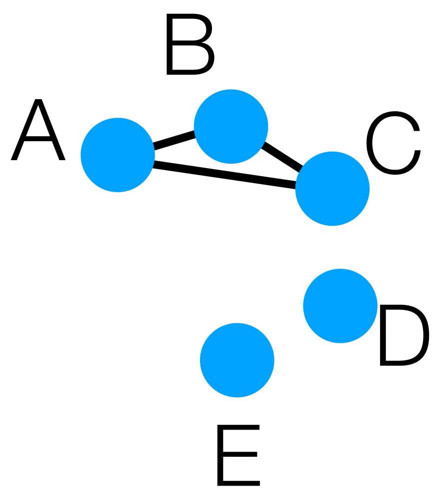
2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph
(add edges for each point to
its 2 nearest neighbors)

Isomap Calculation Example



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

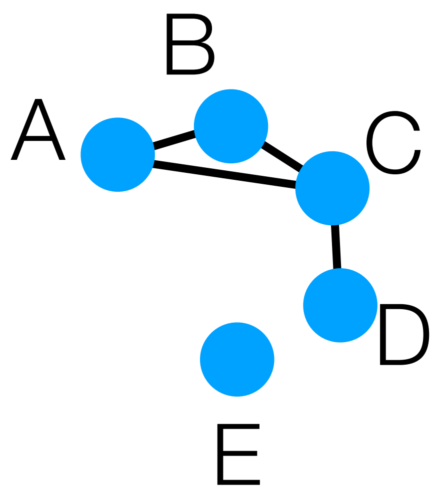
2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph
(add edges for each point to
its 2 nearest neighbors)

Isomap Calculation Example



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

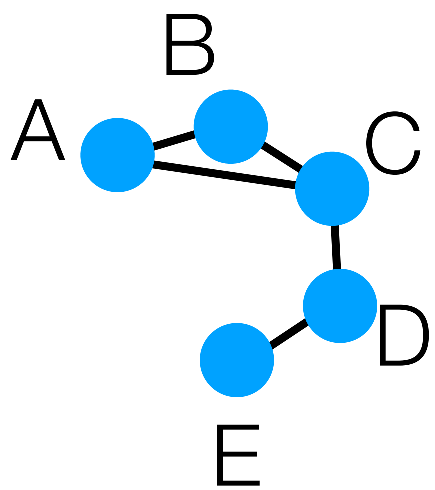
2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph
(add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

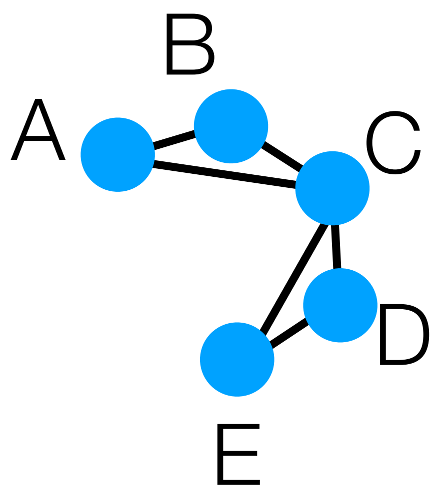
2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph
(add edges for each point to
its 2 nearest neighbors)

Isomap Calculation Example



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

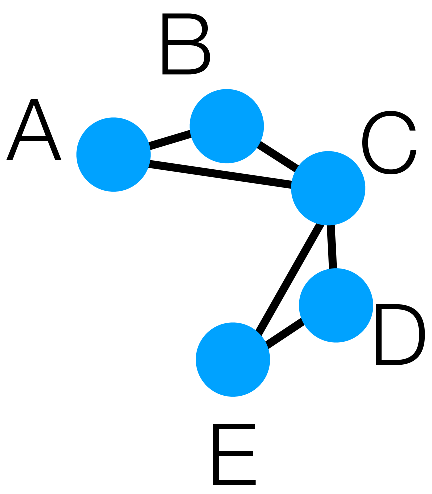
2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

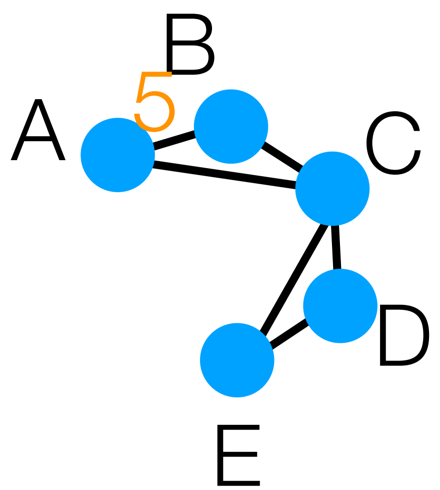
2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

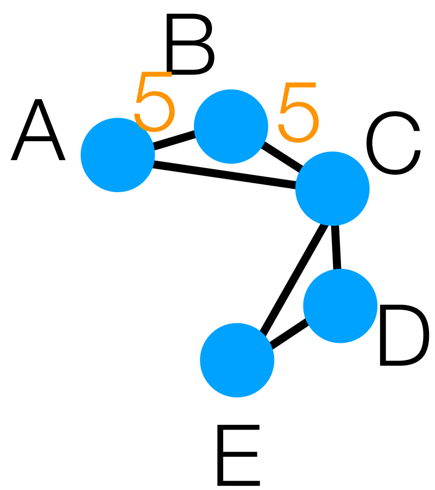
2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

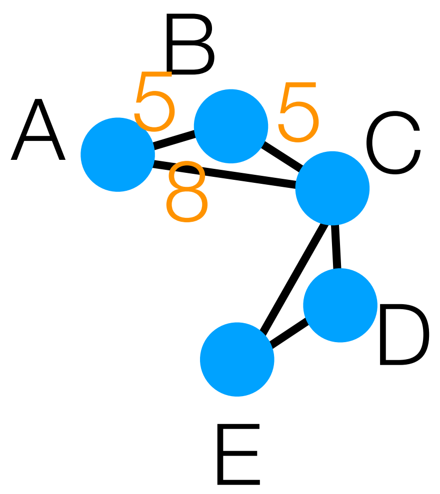
2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph
(add edges for each point to
its 2 nearest neighbors)

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

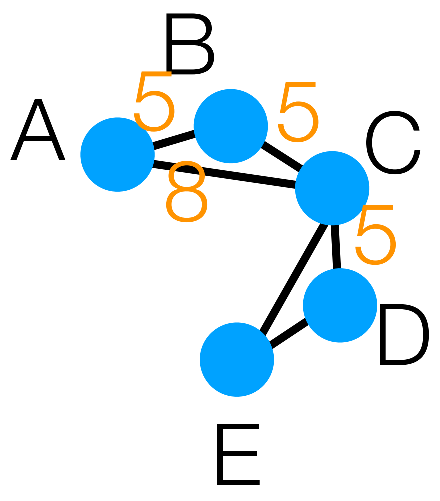
2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

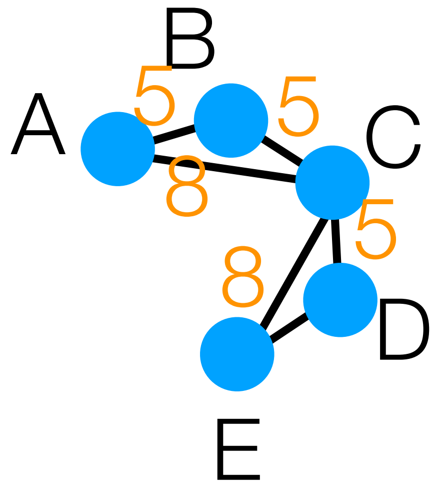
2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

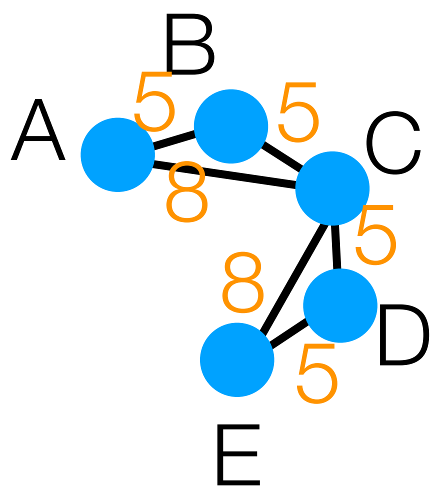
2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

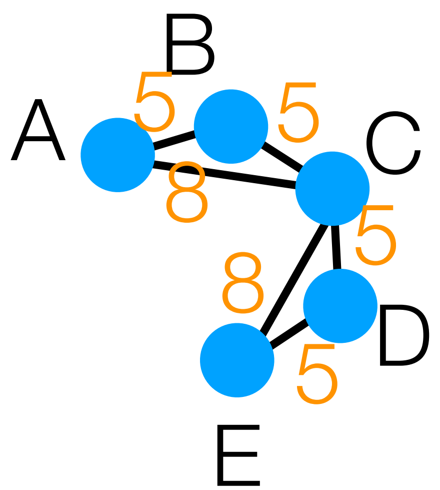
2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

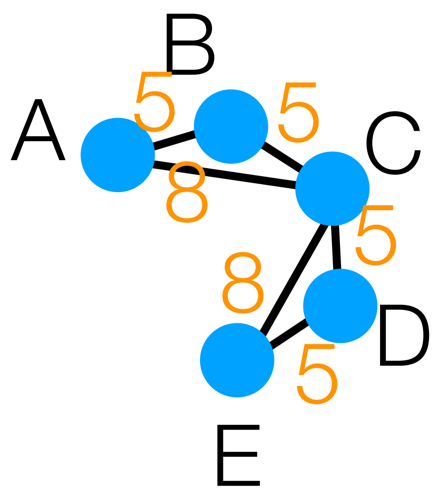
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A					
B					
C					
D					
E					

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

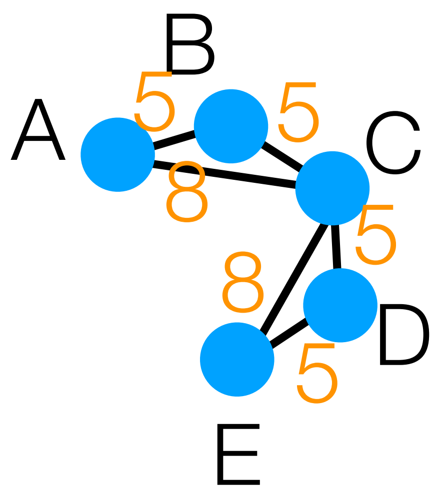
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0				
B		0			
C			0		
D				0	
E					0

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

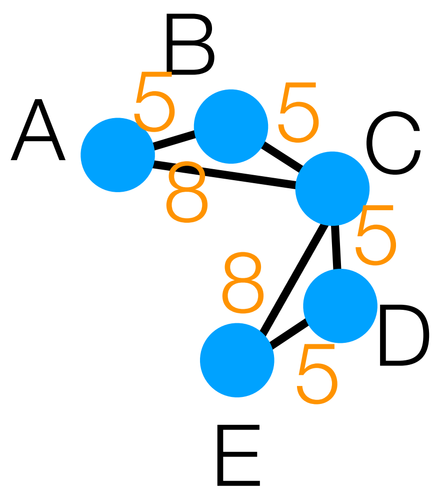
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5			
B		0	5		
C			0	5	
D				0	5
E					0

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

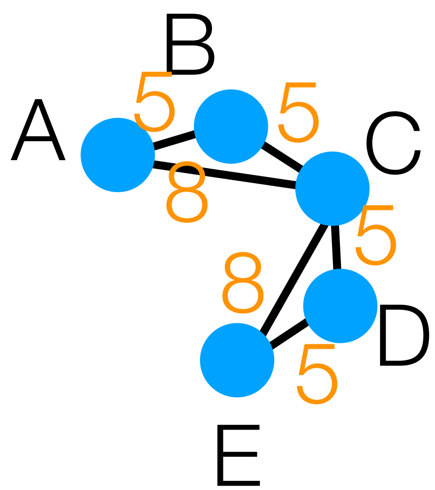
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8		
B		0	5		
C			0	5	
D				0	5
E					0

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

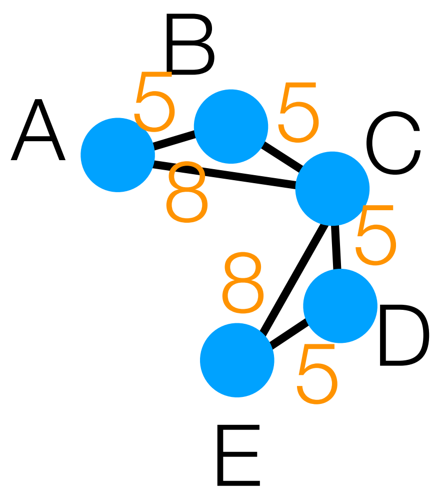
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8	13	
B		0	5		
C			0	5	
D				0	5
E					0

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

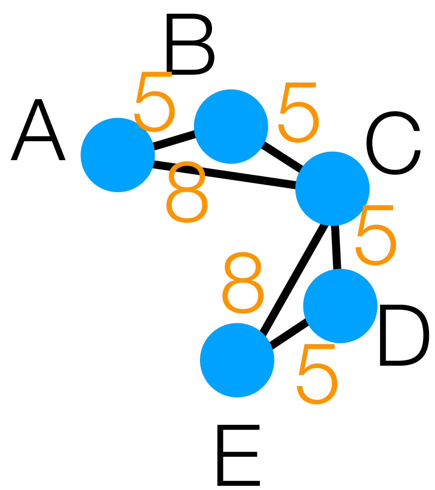
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8	13	16
B		0	5		
C			0	5	
D				0	5
E					0

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

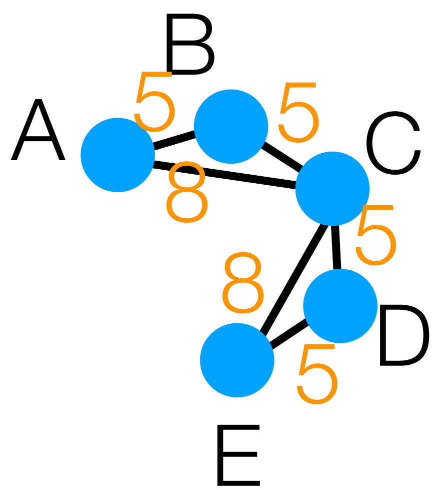
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8	13	16
B		0	5	10	
C			0	5	
D				0	5
E					0

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

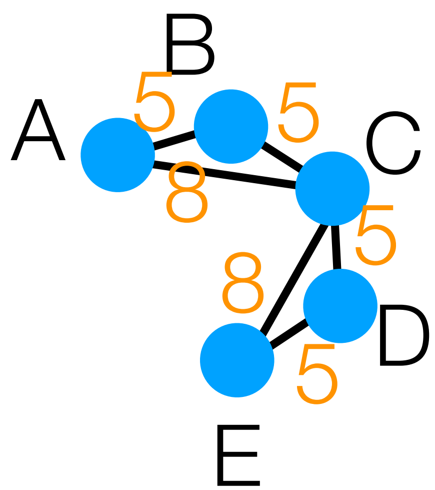
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8	13	16
B		0	5	10	13
C			0	5	
D				0	5
E					0

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

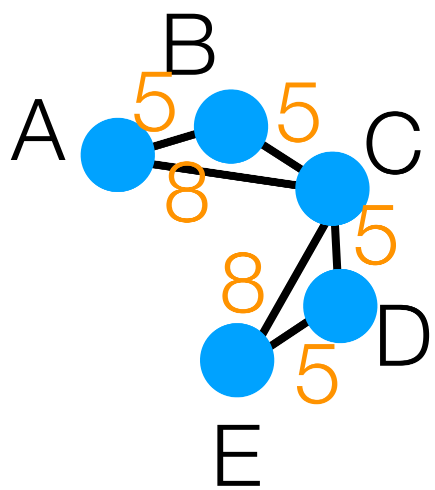
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8	13	16
B		0	5	10	13
C			0	5	8
D				0	5
E					0

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

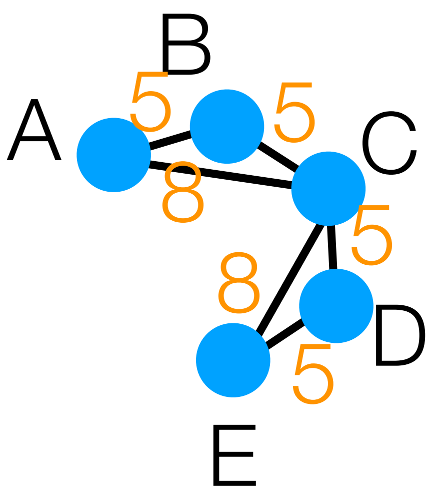
Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8	13	16
B	5	0	5	10	13
C	8	5	0	5	8
D	13	10	5	0	5
E	16	13	8	5	0

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

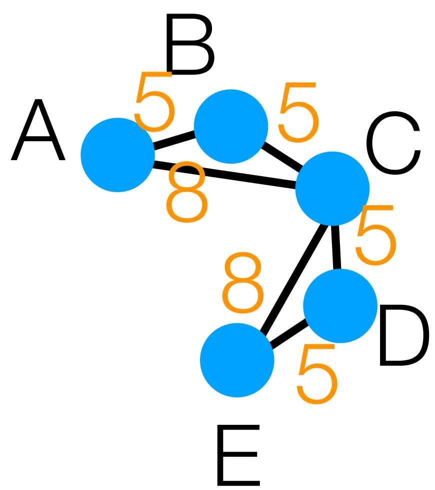
Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8	13	16
B	5	0	5	10	13
C	8	5	0	5	8
D	13	10	5	0	5
E	16	13	8	5	0

This matrix gets fed into *multidimensional scaling* to get 1D version of A, B, C, D, E

Isomap Calculation Example

In orange: road lengths



2 nearest neighbors of A: B, C

2 nearest neighbors of B: A, C

2 nearest neighbors of C: B, D

2 nearest neighbors of D: C, E

2 nearest neighbors of E: C, D

Build "symmetric 2-NN" graph (add edges for each point to its 2 nearest neighbors)

Shortest distances between every point to every other point *where we are only allowed to travel along the roads*

	A	B	C	D	E
A	0	5	8	13	16
B	5	0	5	10	13
C	8	5	0	5	8
D	13	10	5	0	5
E	16	13	8	5	0

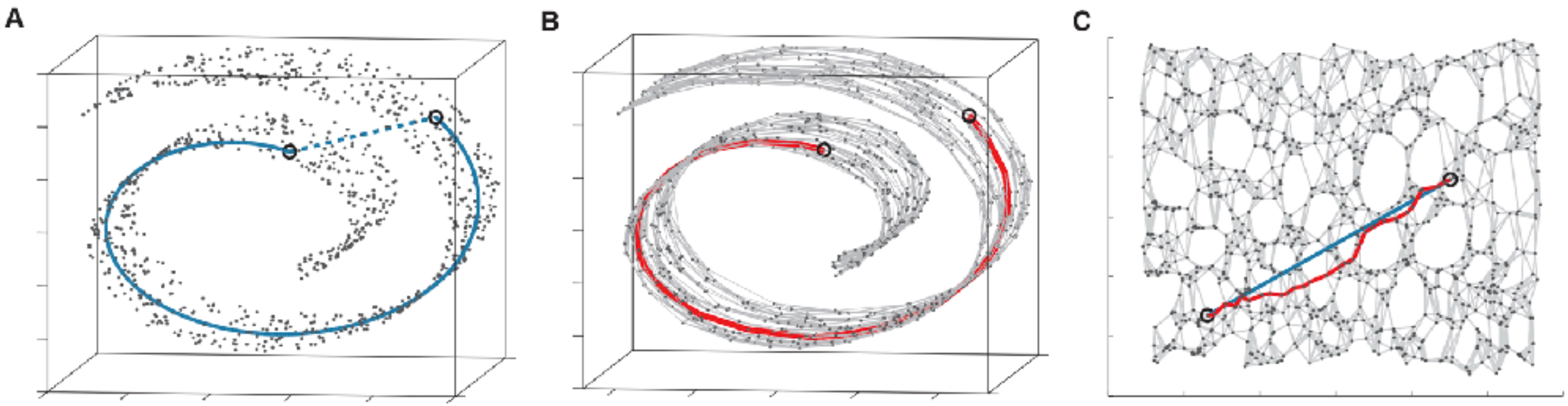
This matrix gets fed into *multidimensional scaling* to get 1D version of A, B, C, D, E

The solution is not unique!

Isomap Calculation Example

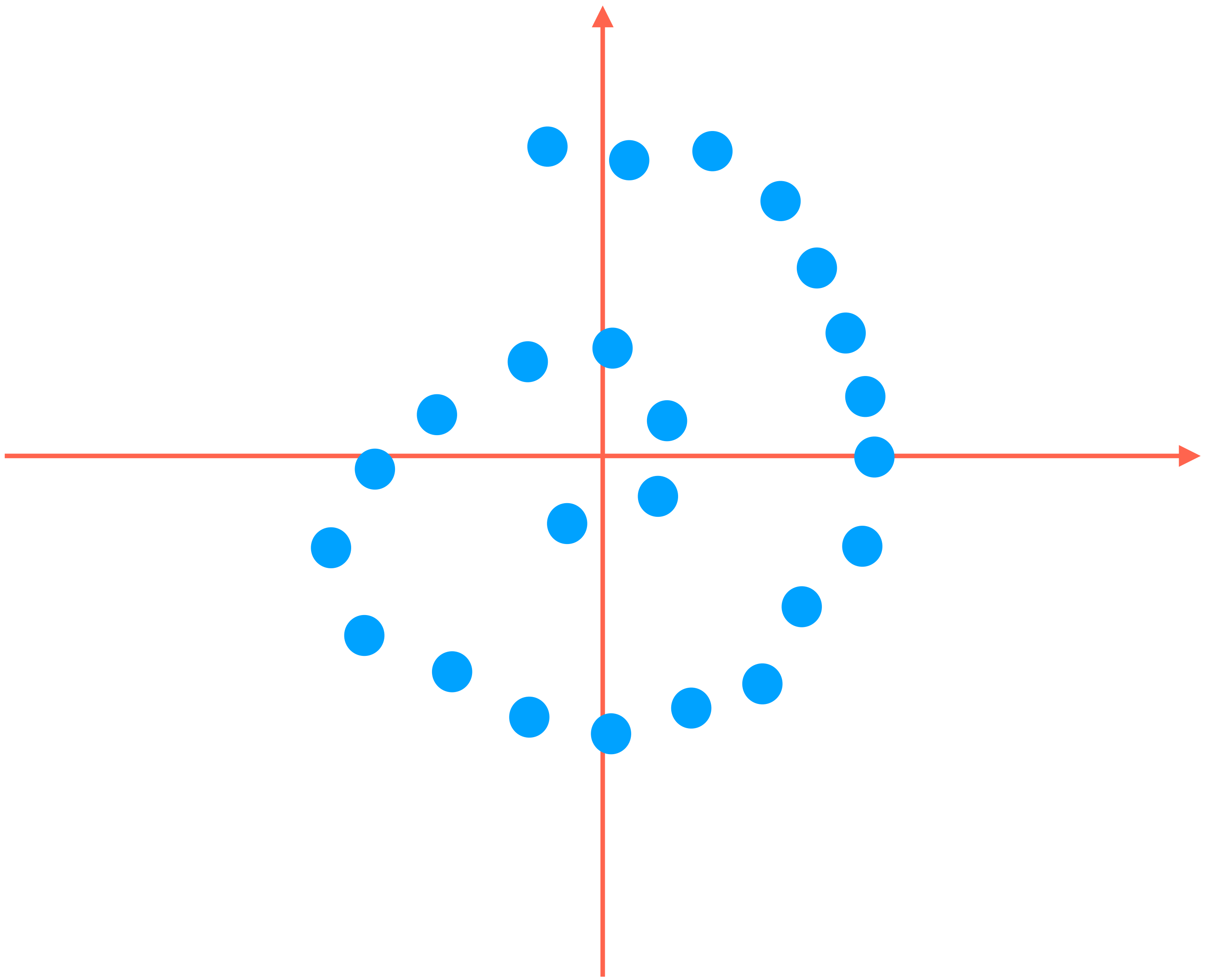
Multidimensional scaling demo

3D Swiss Roll Example

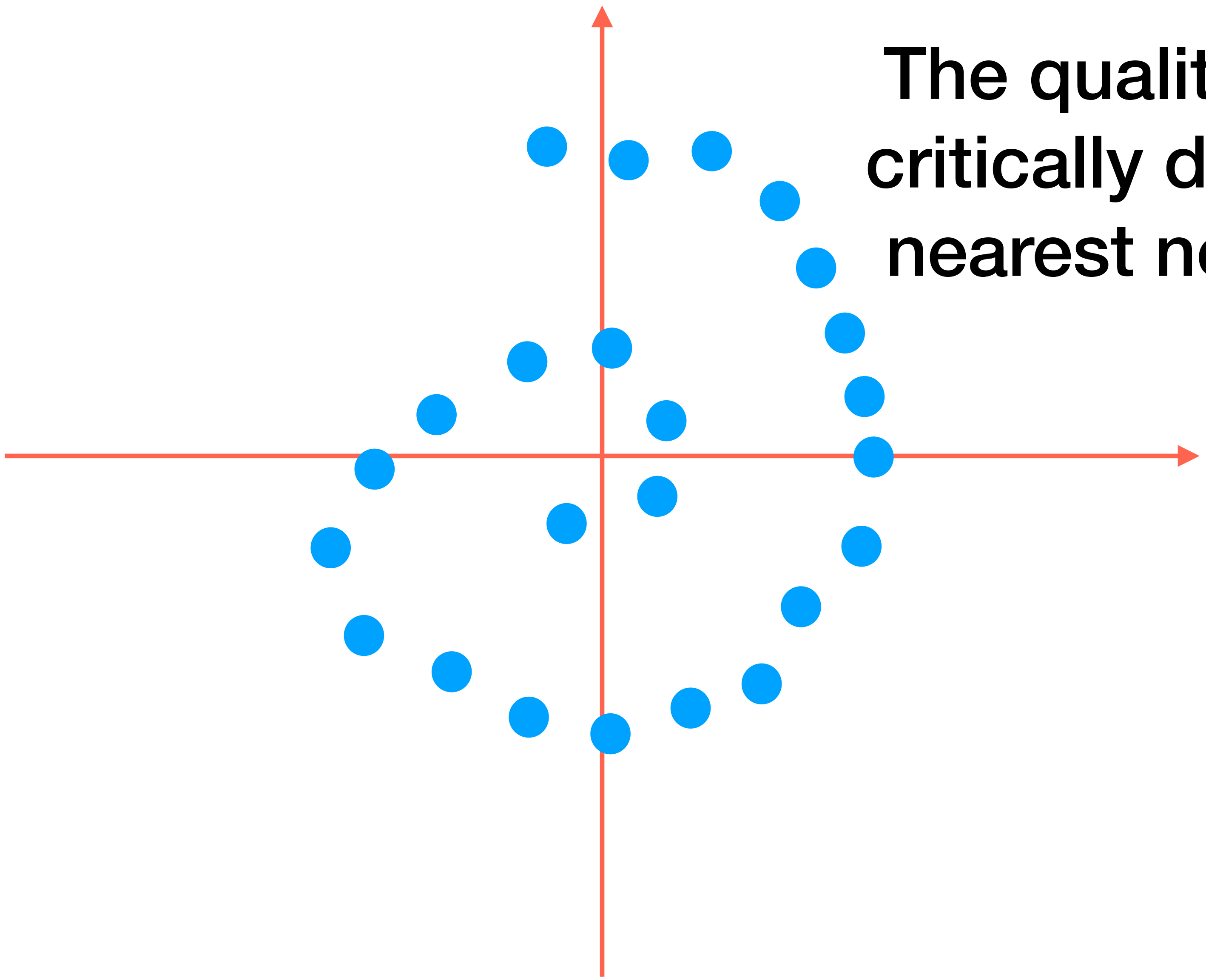


Joshua B. Tenenbaum, Vin de Silva, John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 2000.

Some Observations on Isomap

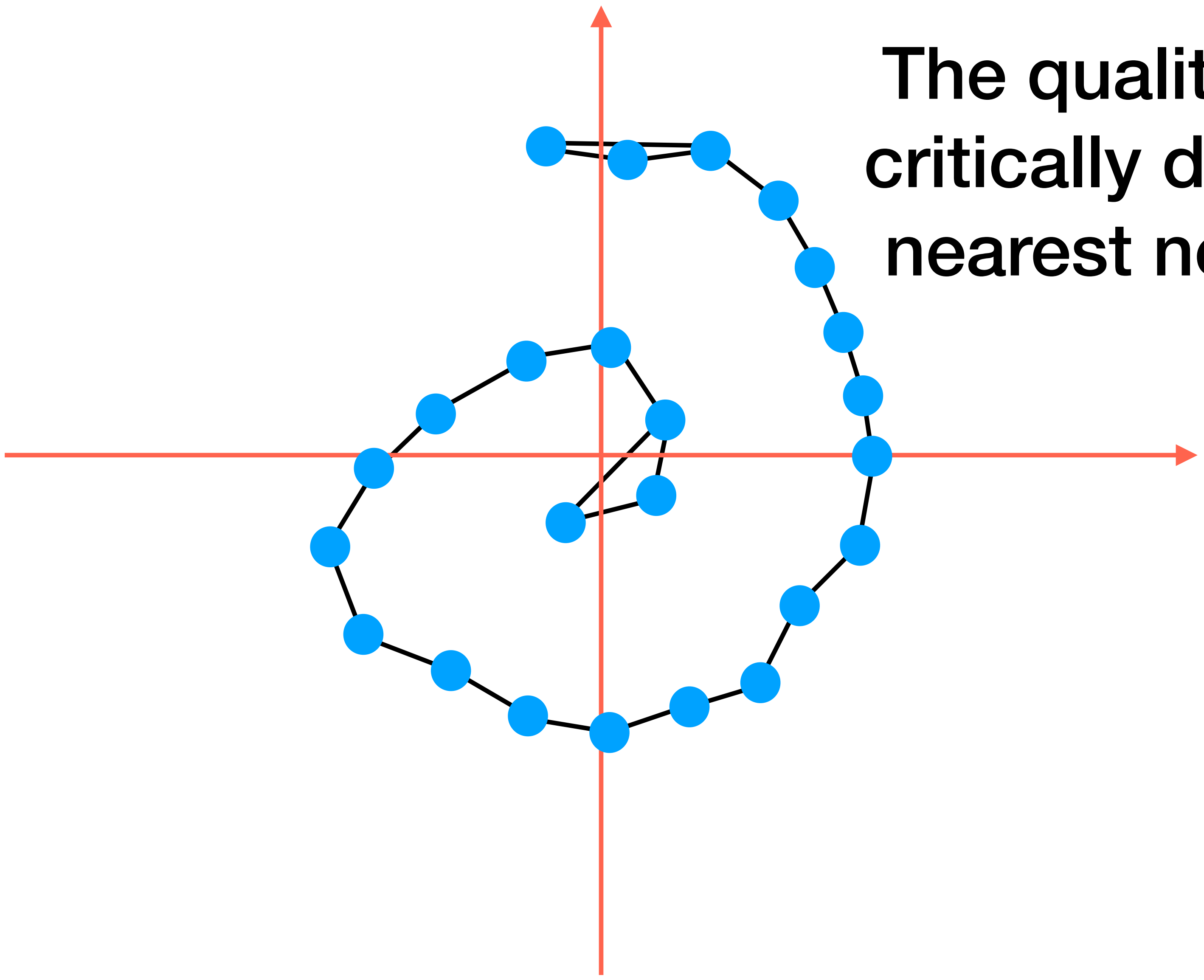


Some Observations on Isomap



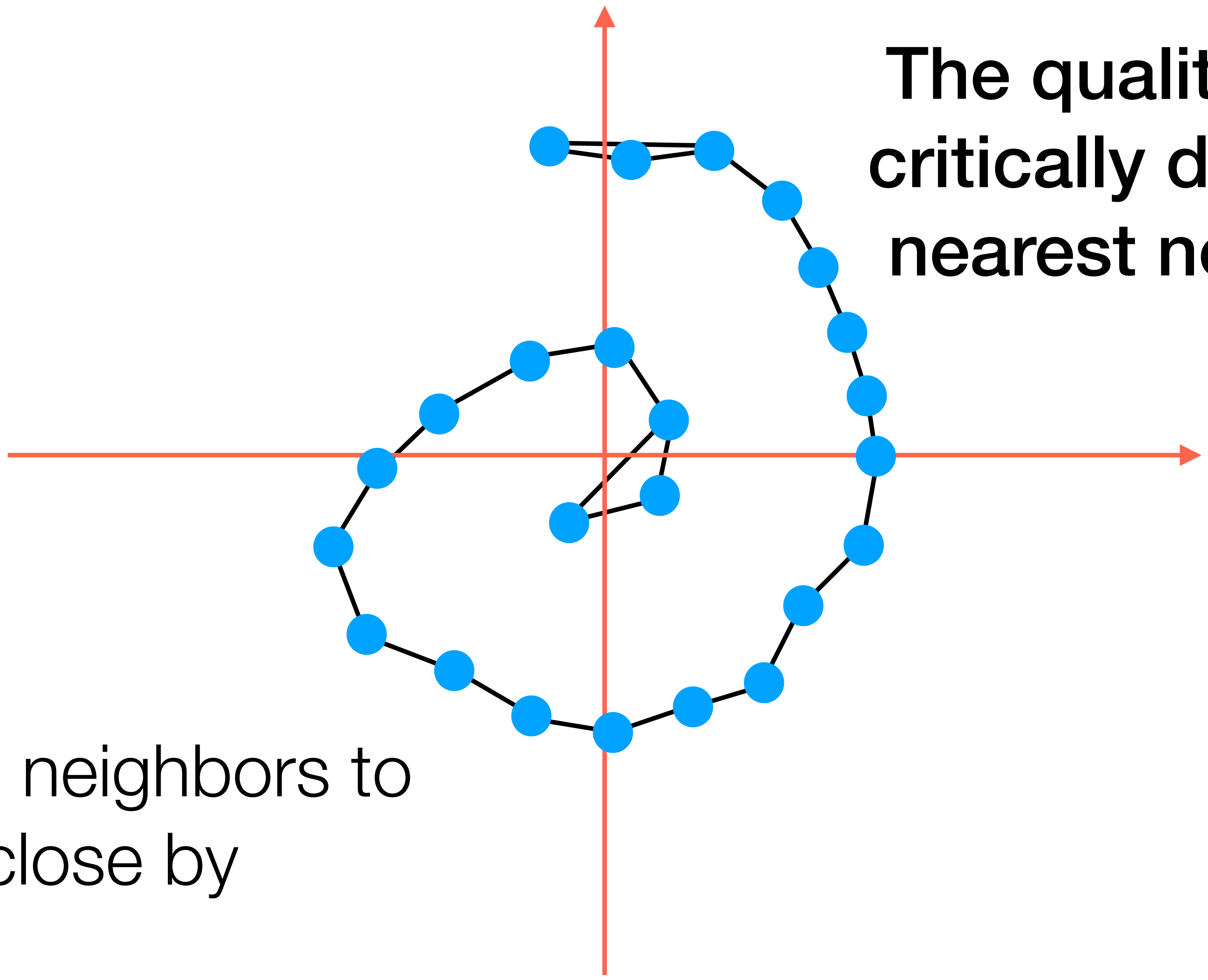
The quality of the result critically depends on the nearest neighbor graph

Some Observations on Isomap



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

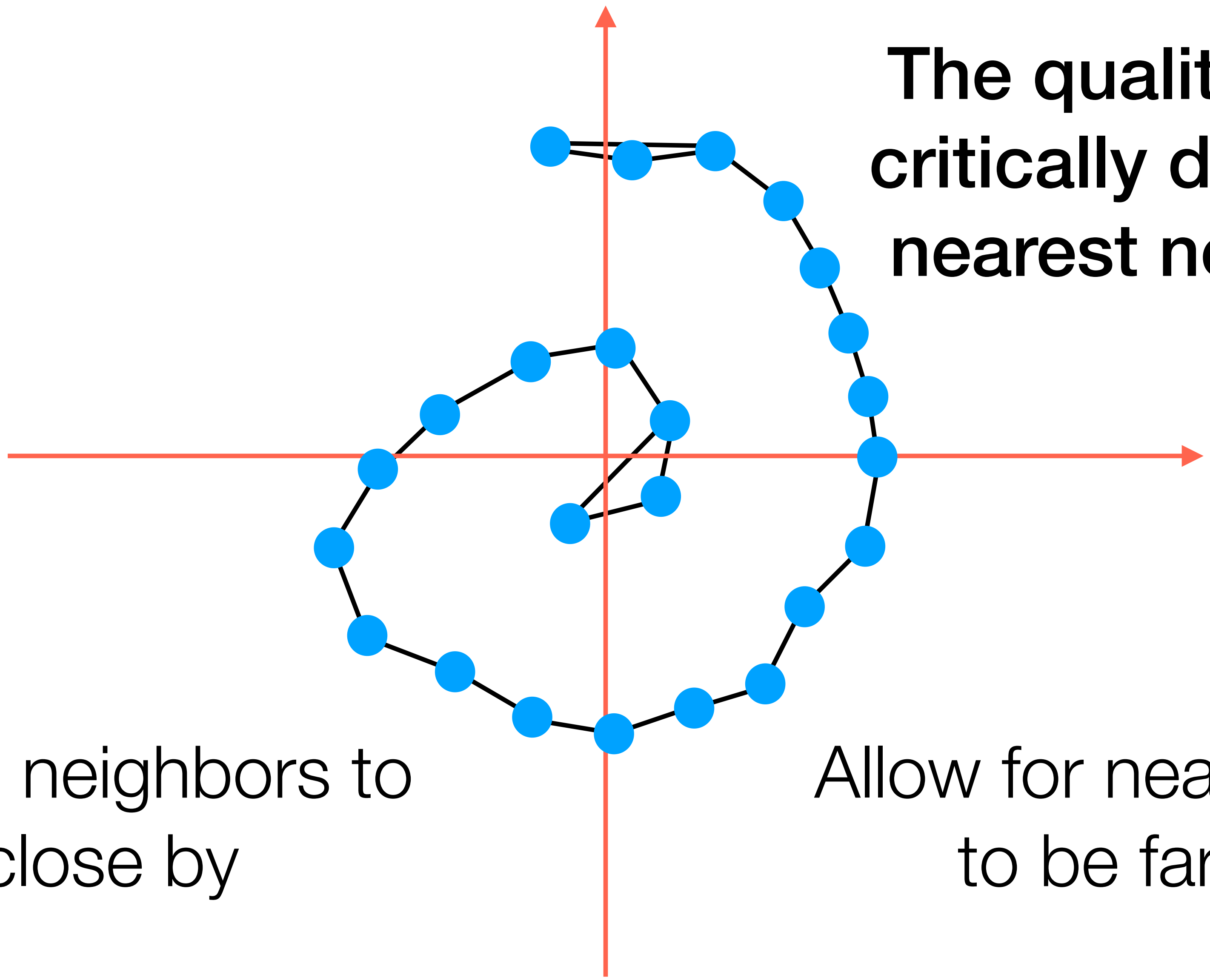
Some Observations on Isomap



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

Ask for nearest neighbors to be really close by

Some Observations on Isomap

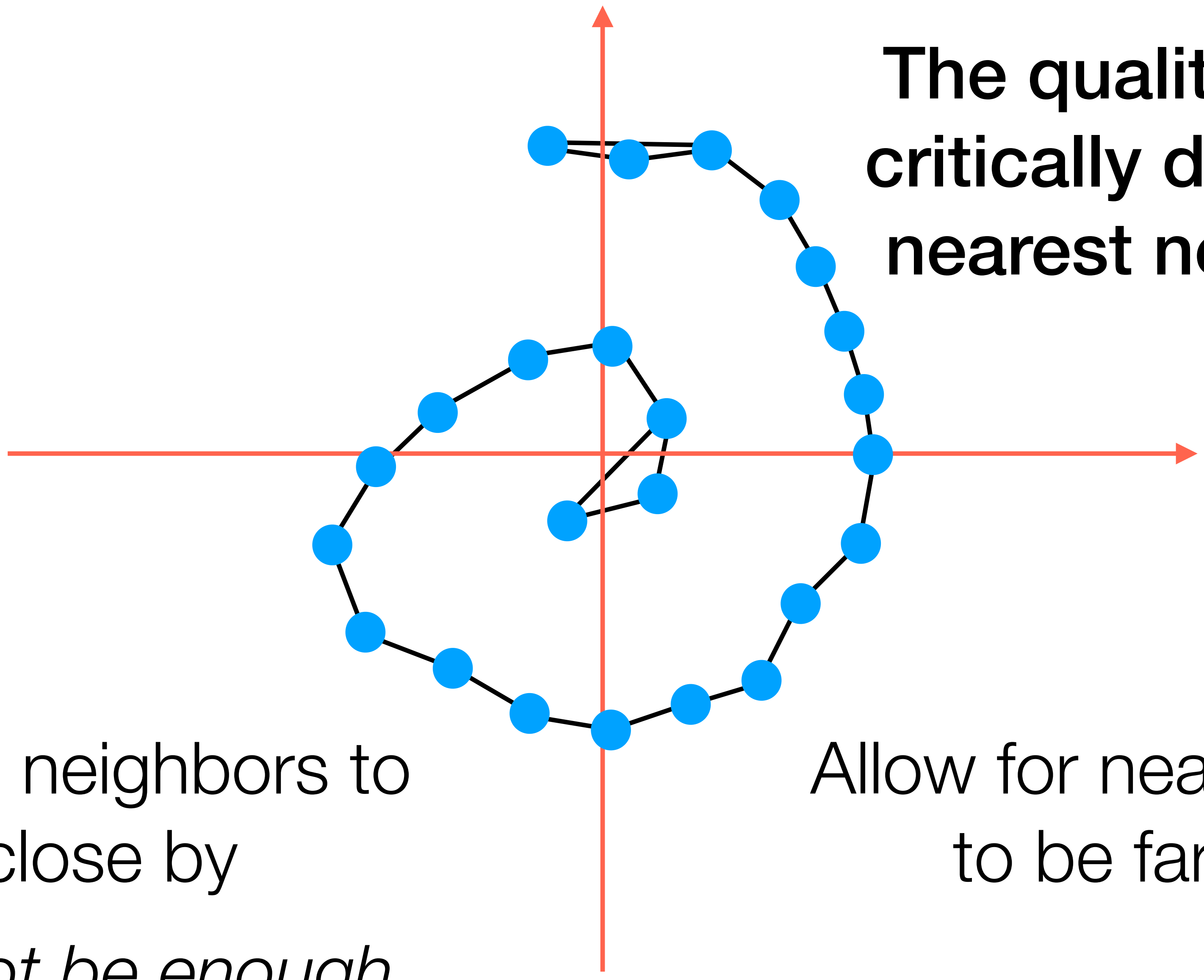


The quality of the result critically depends on the nearest neighbor graph

Ask for nearest neighbors to be really close by

Allow for nearest neighbors to be farther away

Some Observations on Isomap



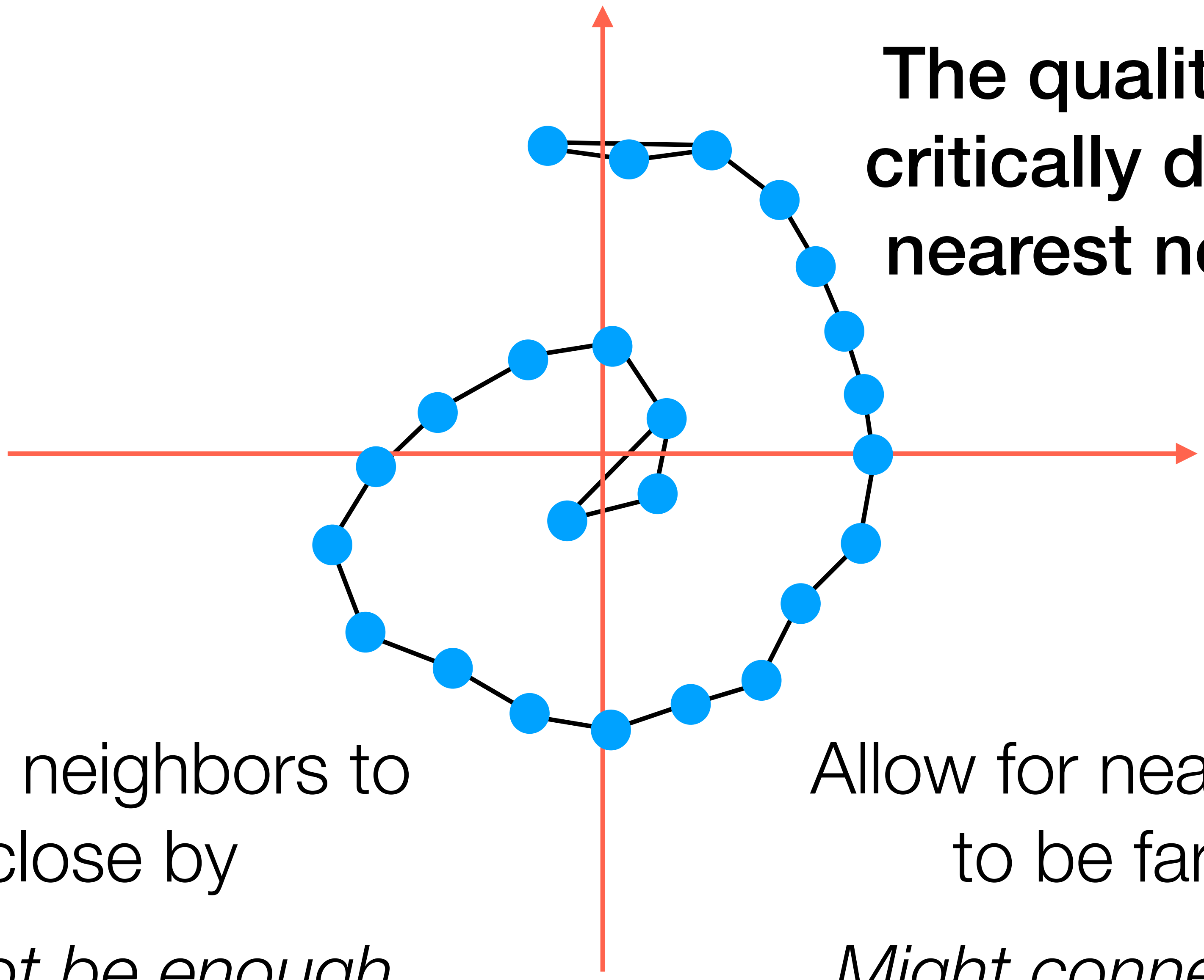
The quality of the result critically depends on the nearest neighbor graph

Ask for nearest neighbors to be really close by

There might not be enough edges

Allow for nearest neighbors to be farther away

Some Observations on Isomap



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

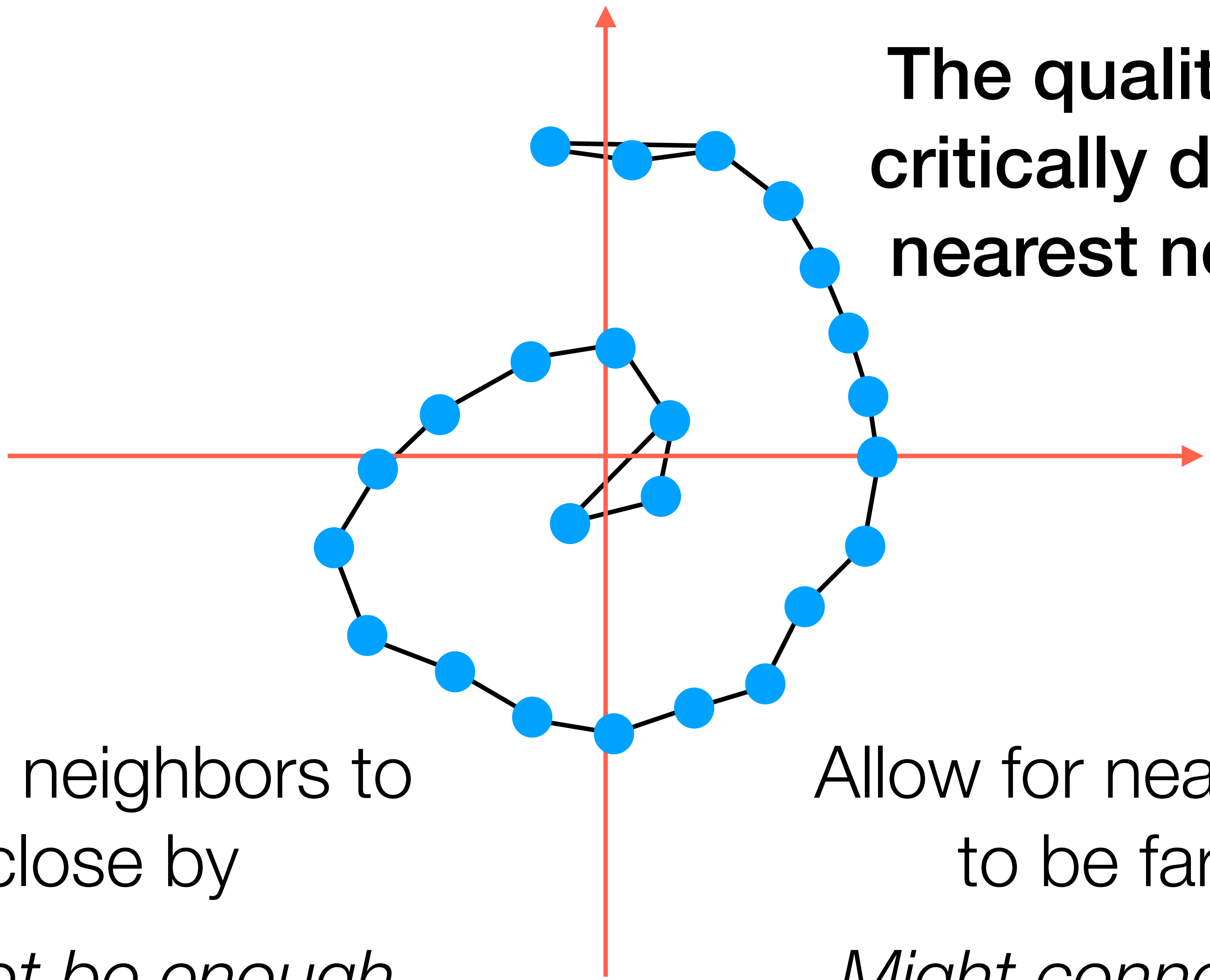
Ask for nearest neighbors to be really close by

There might not be enough edges

Allow for nearest neighbors to be farther away

Might connect points that shouldn't be connected

Some Observations on Isomap



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

Ask for nearest neighbors to be really close by

There might not be enough edges

Allow for nearest neighbors to be farther away

Might connect points that shouldn't be connected

In general: try different parameters for nearest neighbor graph construction when using Isomap + visualize

t-SNE
(t-distributed stochastic neighbor embedding)

t-SNE High-Level Idea #1

t-SNE High-Level Idea #1

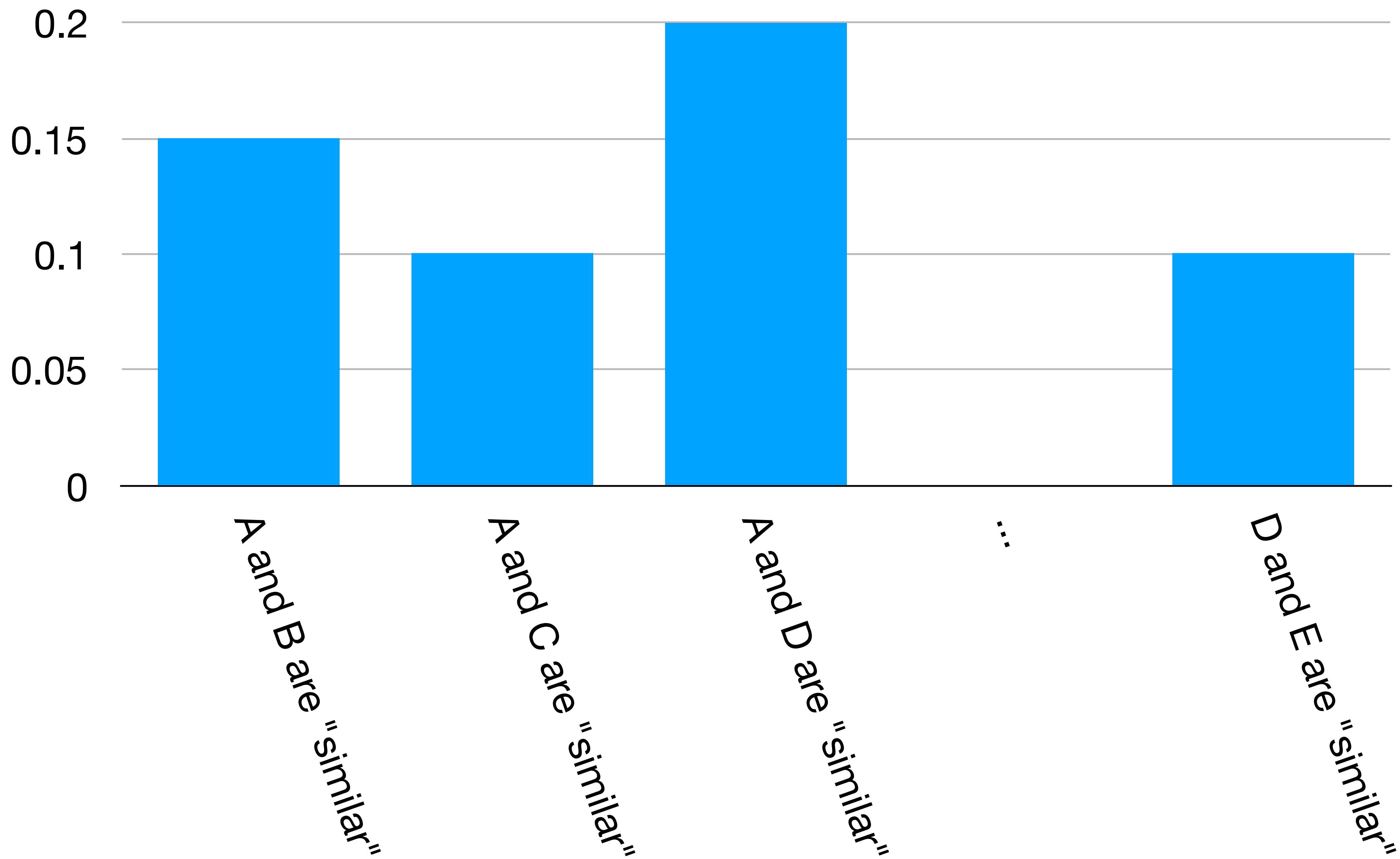
- Don't use deterministic definition of which points are neighbors

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 #1

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



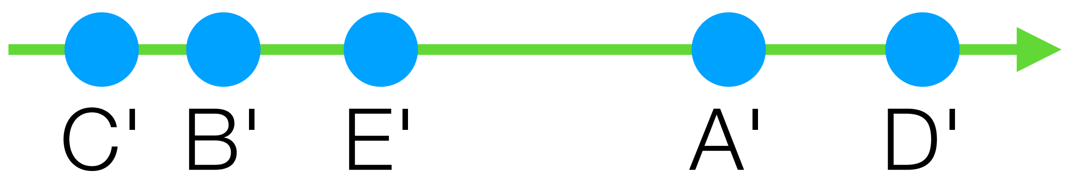
t-SNE High-Level Idea #2

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):

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):



t-SNE High-Level Idea #2

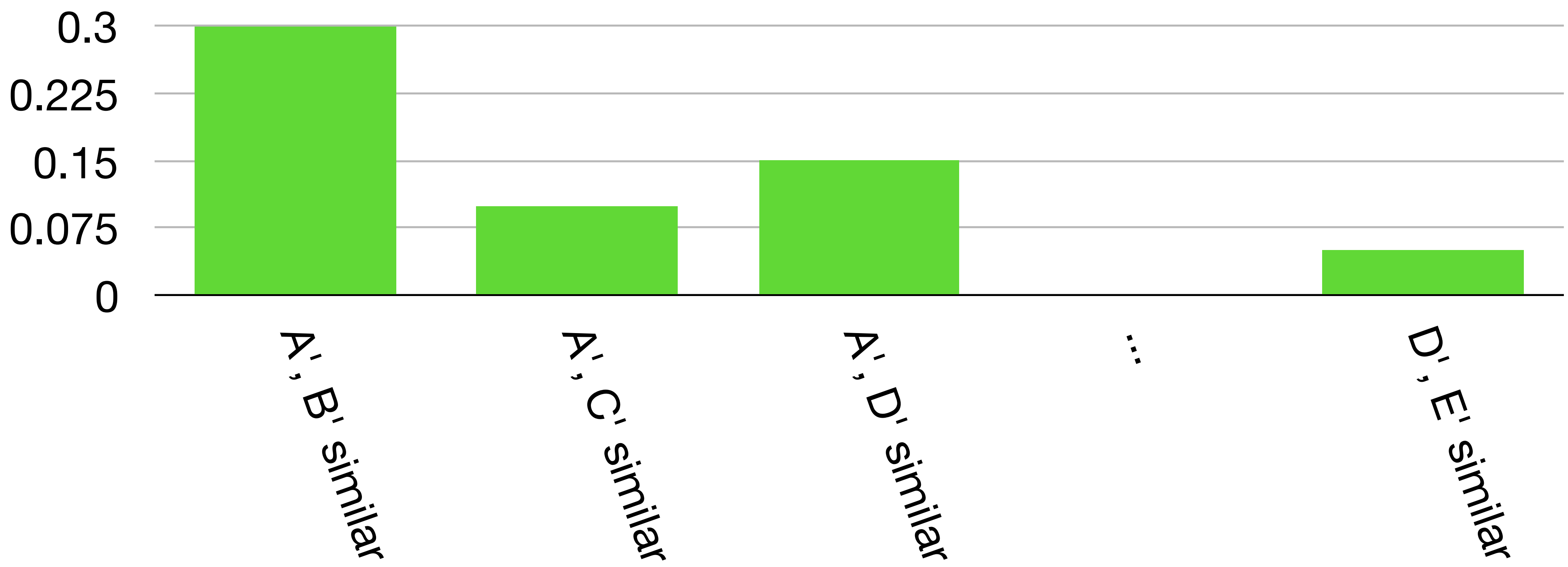
- In low-dim. space (e.g., 1D), suppose we just randomly assigned coordinates as a candidate for a low-dimensional representation for A, B, C, D, E (I'll denote them with primes):

- 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 #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



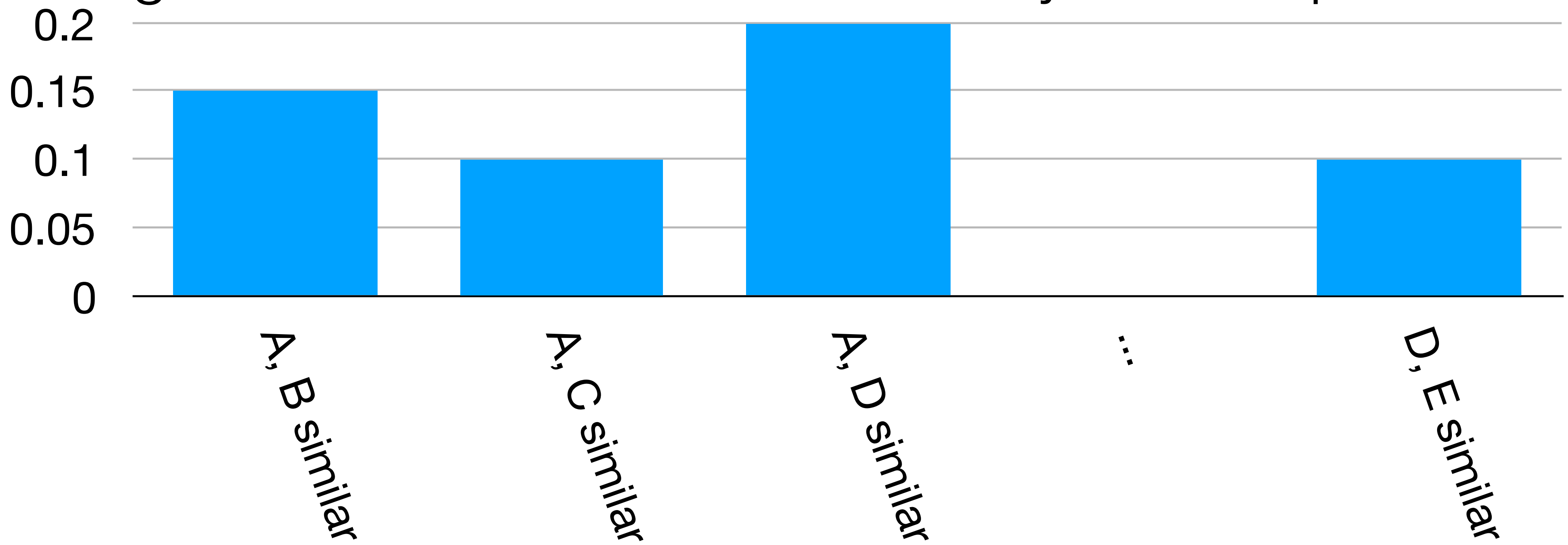
t-SNE High-Level Idea #3

t-SNE High-Level Idea #3

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

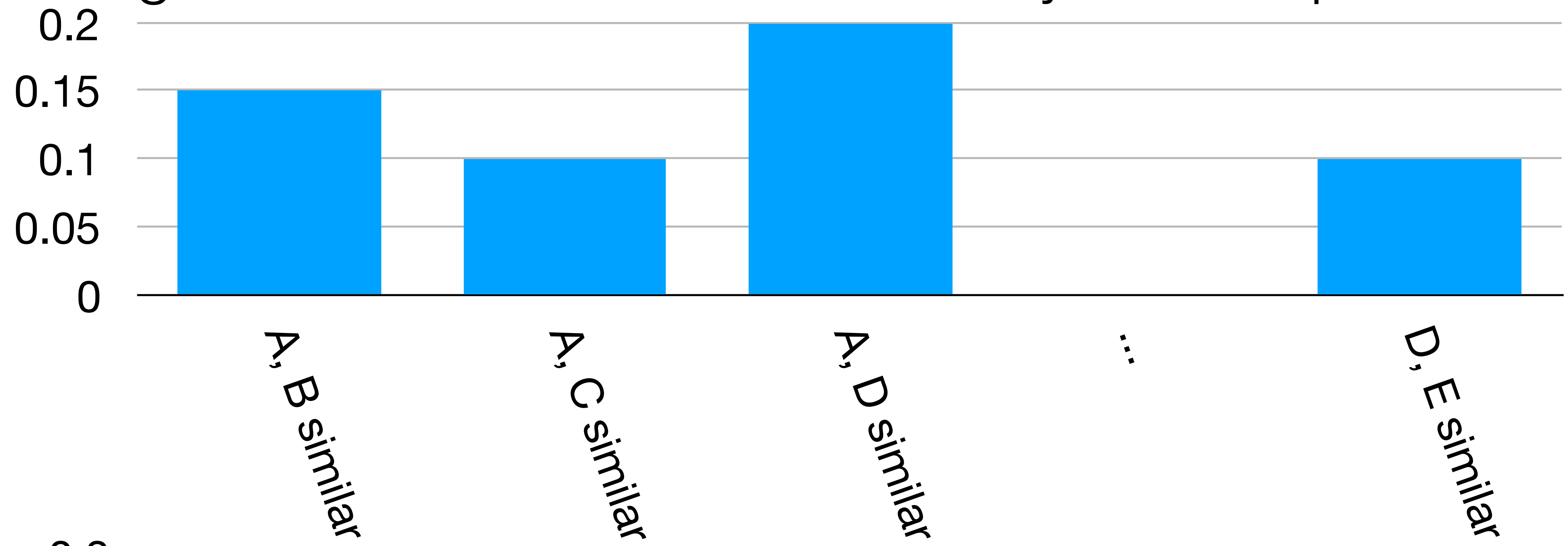
t-SNE High-Level Idea #3

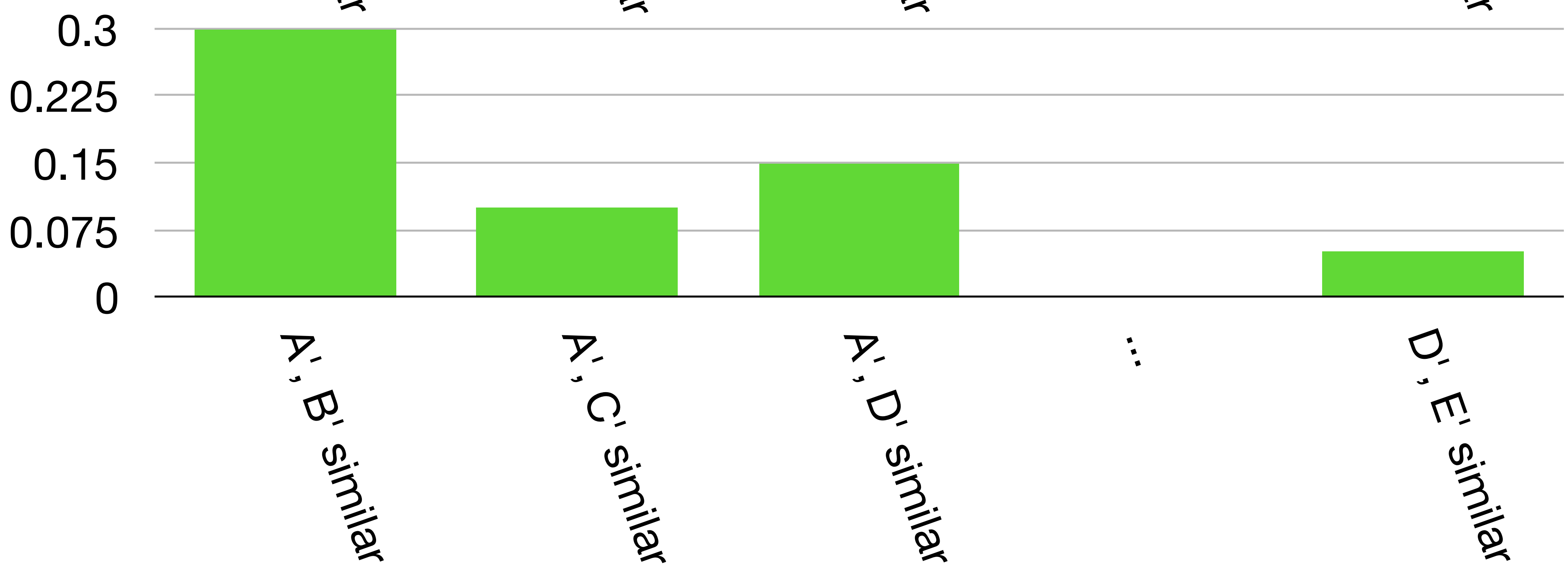
- Keep improving low-dimensional representation to make the following two distributions look as closely alike as possible



t-SNE High-Level Idea #3

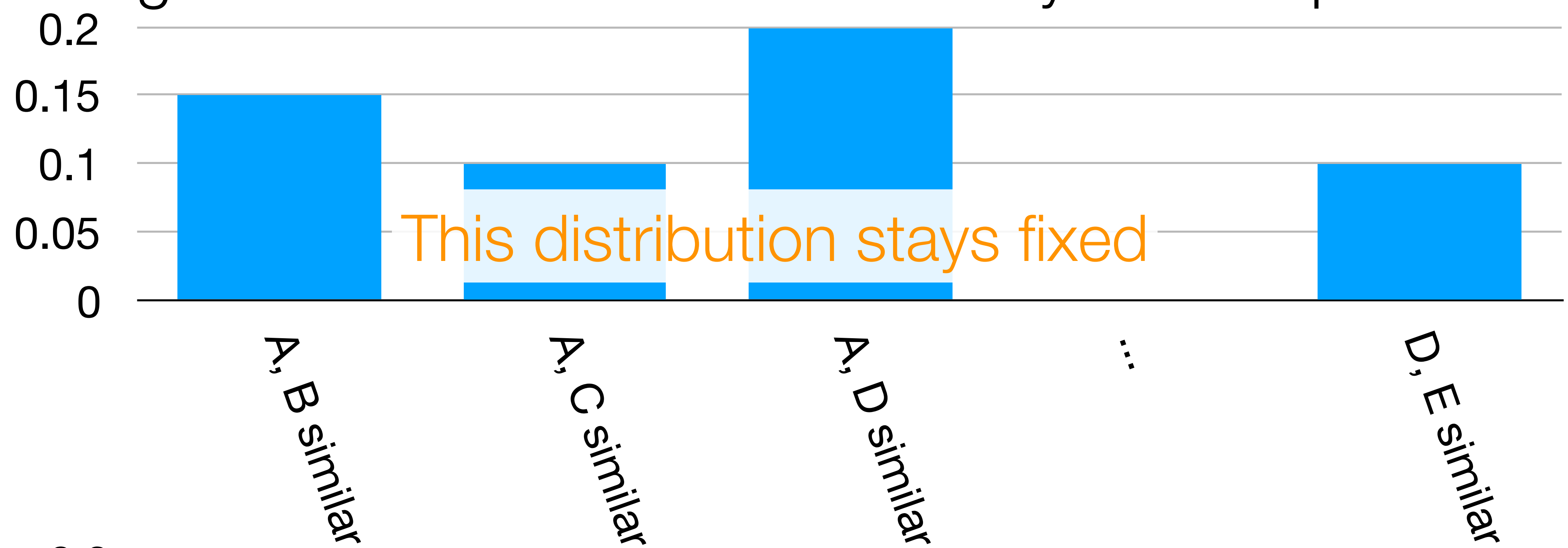
- Keep improving low-dimensional representation to make the following two distributions look as closely alike as possible

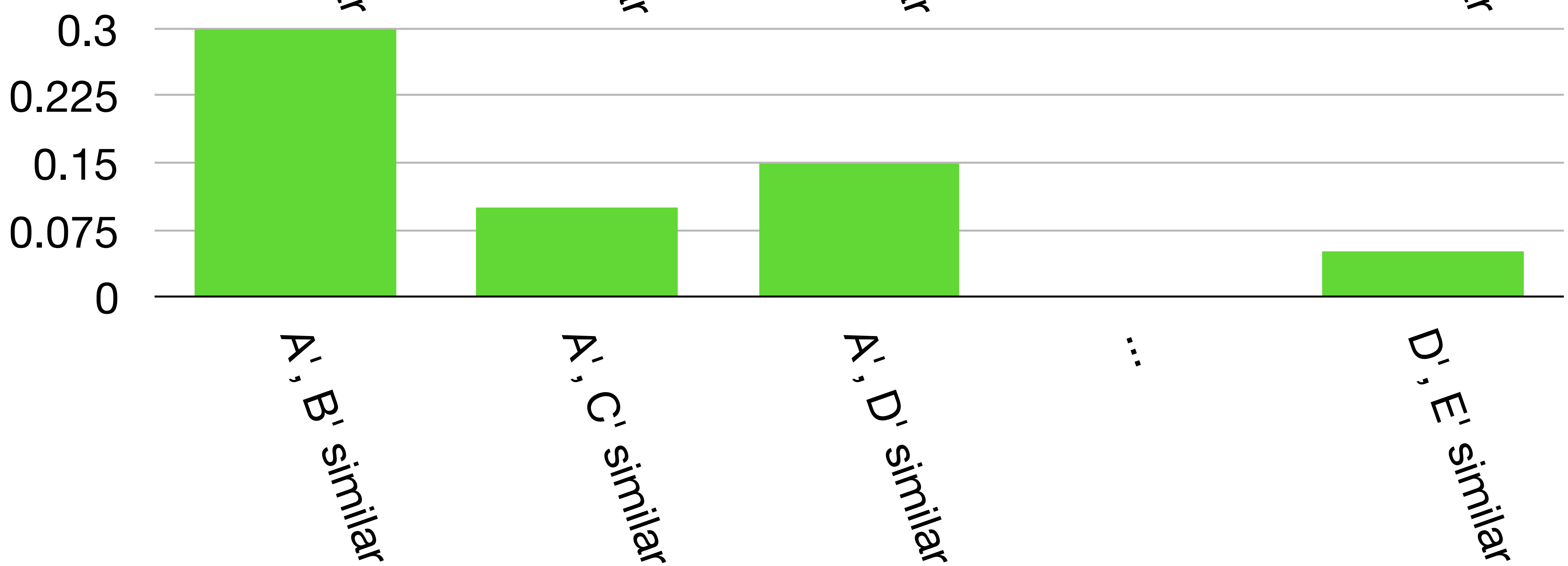




t-SNE High-Level Idea #3

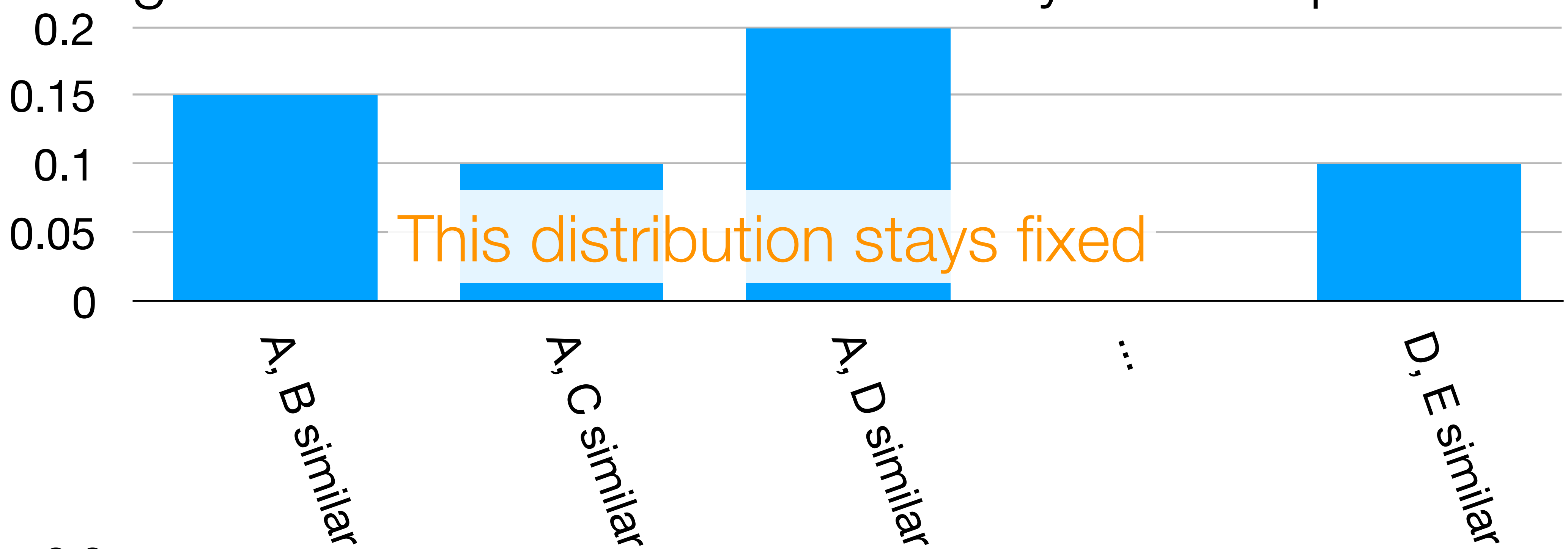
- Keep improving low-dimensional representation to make the following two distributions look as closely alike as possible

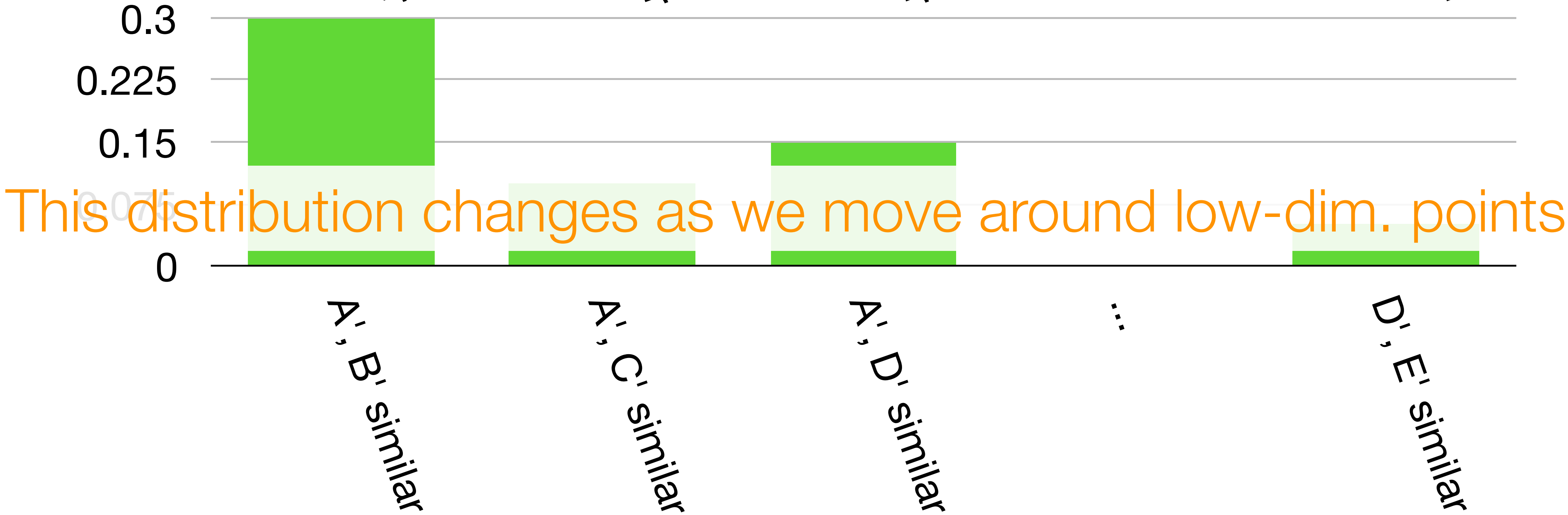




t-SNE High-Level Idea #3

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





Manifold Learning with t-SNE

Demo

Technical Detail for t-SNE

Fleshing out high level idea #1

Suppose there are n high-dimensional points $x_1, x_2, \ldots, x_n$

For a specific point i, point i picks point j ($\neq i$) to be a neighbor with probability:

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

σ_i (depends on i) controls the probability in which point j would be picked by i as a neighbor (think about when it gets close to 0 or when it explodes to ∞)

σ_i is controlled by a knob called 'perplexity'
(rough intuition: it is like selecting small vs large neighborhoods for Isomap)

Points i and j are "similar" with probability: $p_{i,j} = \dfrac{p_{j|i} + p_{i|j}}{2n}$

This defines the earlier blue distribution

Technical Detail for t-SNE

Fleshing out high level idea #2

Denote the n low-dimensional points as x_1', x_2', …, x_n'

Low-dim. points i and j are "similar" with probability: $q_{i,j} = \dfrac{\frac{1}{1+\|x_i' - x_j'\|^2}}{\sum_{k \neq m} \frac{1}{1+\|x_k' - x_m'\|^2}}$

This defines the earlier green distribution

Fleshing out high level idea #3

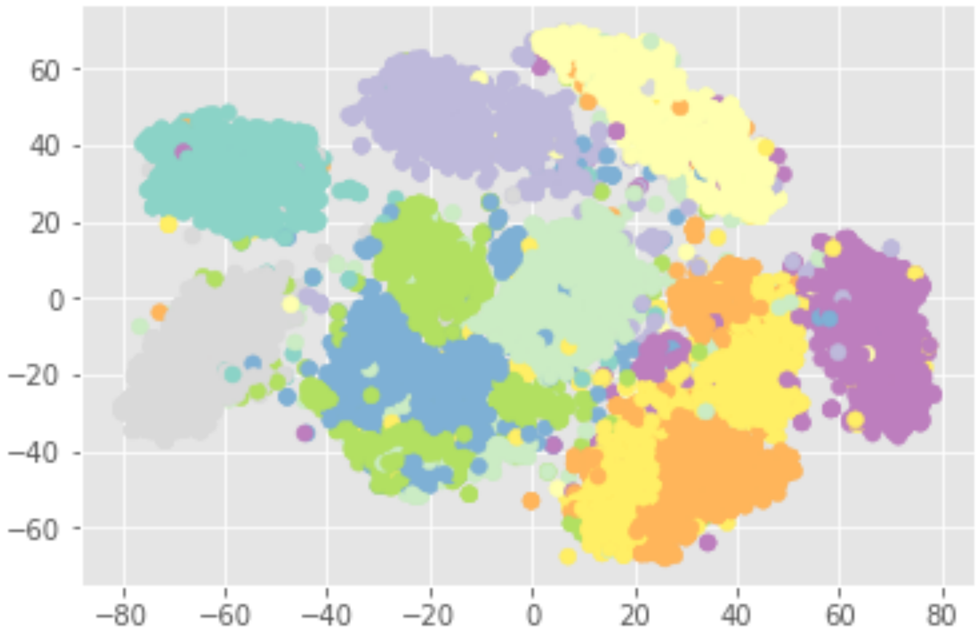
Use gradient descent (with respect to $q_{i,j}$) to minimize:

$$\sum_{i \neq j} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}$$

This is the KL-divergence between distributions $\boldsymbol{p}$ and $\boldsymbol{q}$

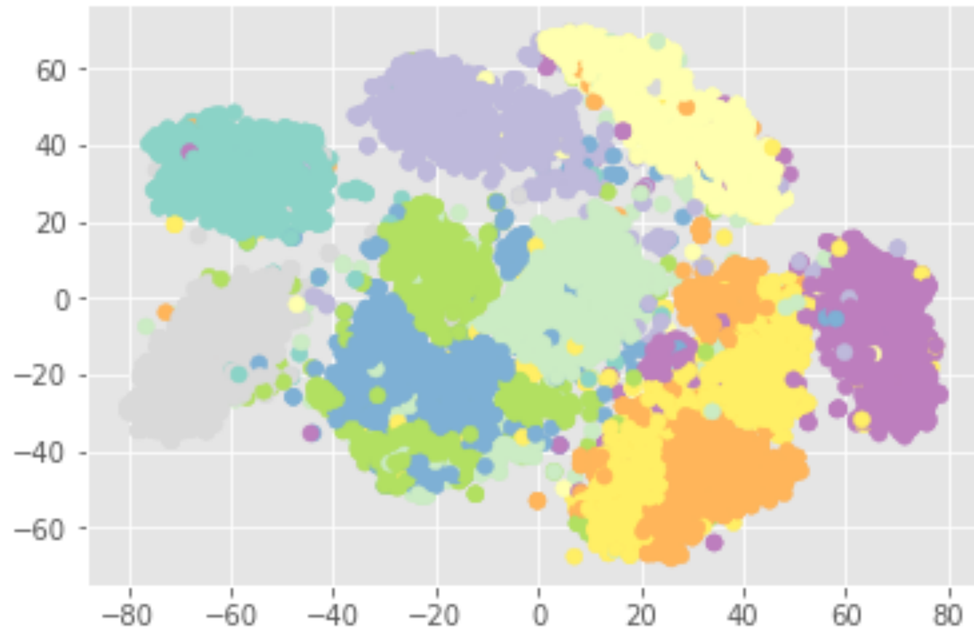
Visualization

Visualization



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

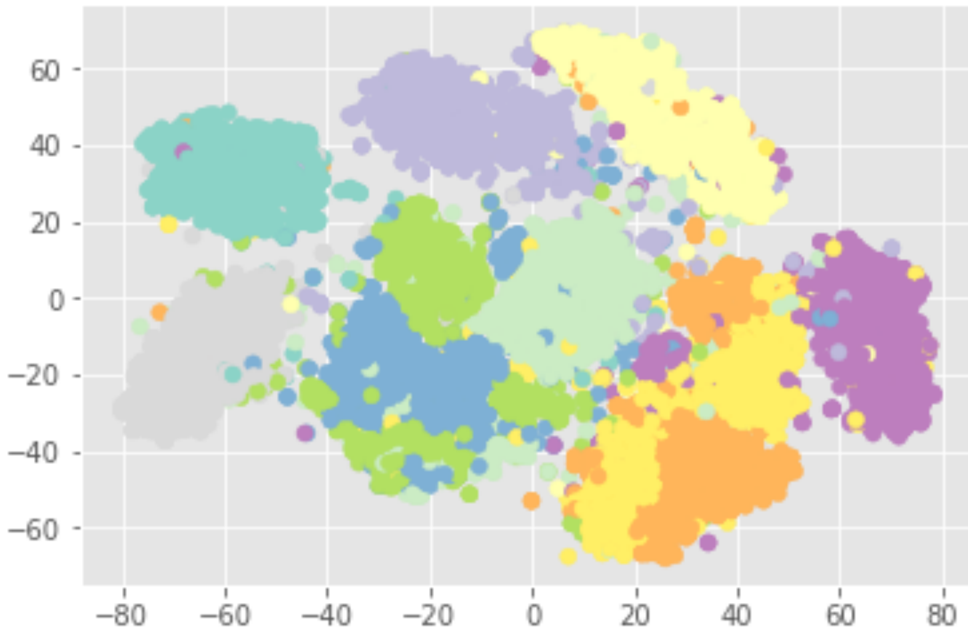
Visualization

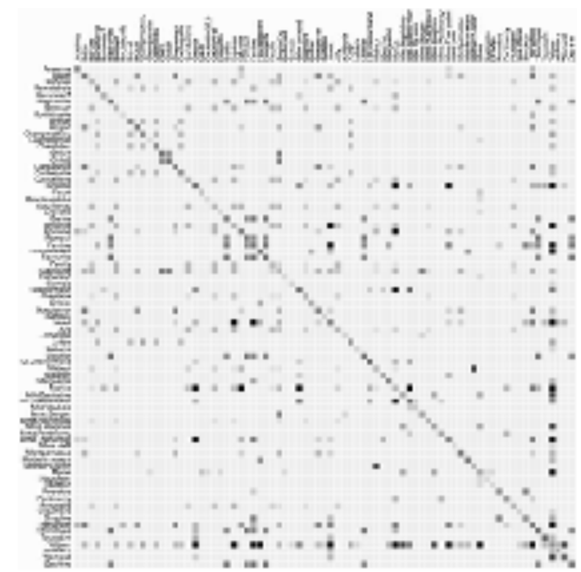


Important:
Handwritten digit demo was a **toy example** where we know which images correspond to digits 0, 1, … 9

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

Visualization



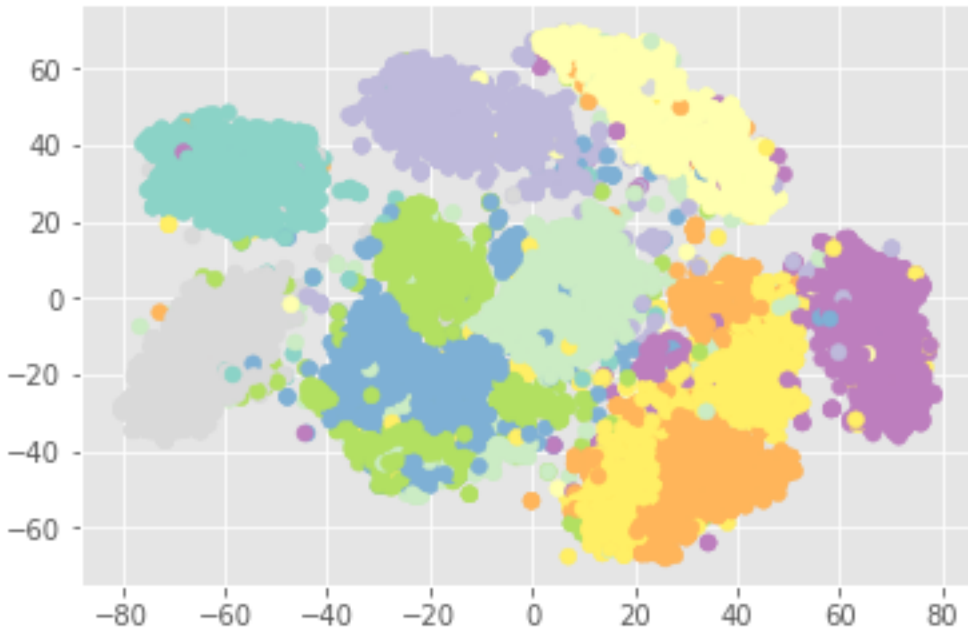


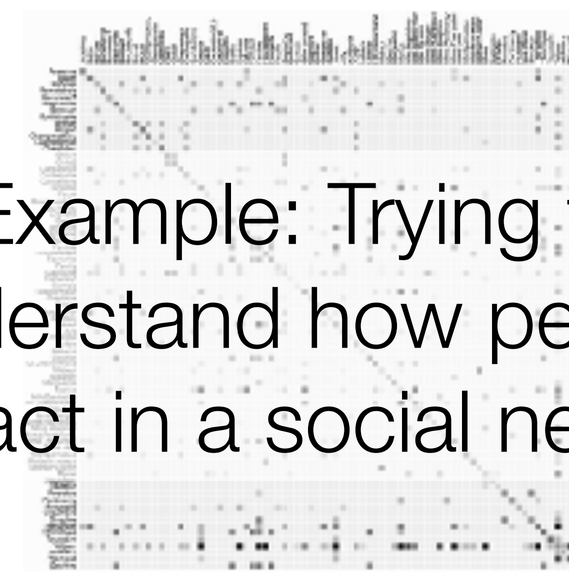
Important:
Handwritten digit demo was a **toy example** where we know which images correspond to digits 0, 1, … 9

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

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

Visualization





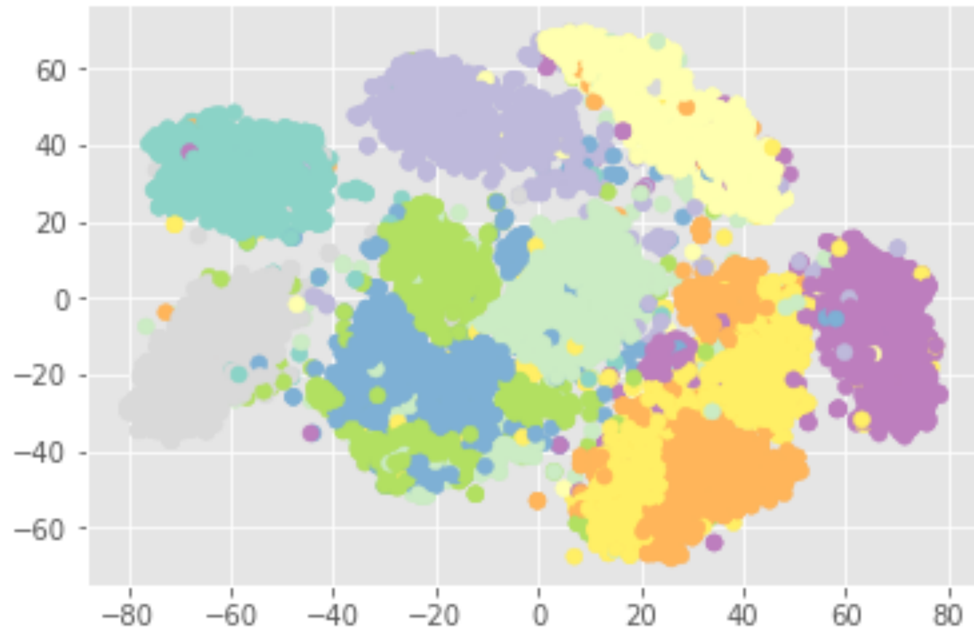
Example: Trying to understand how people interact in a social network

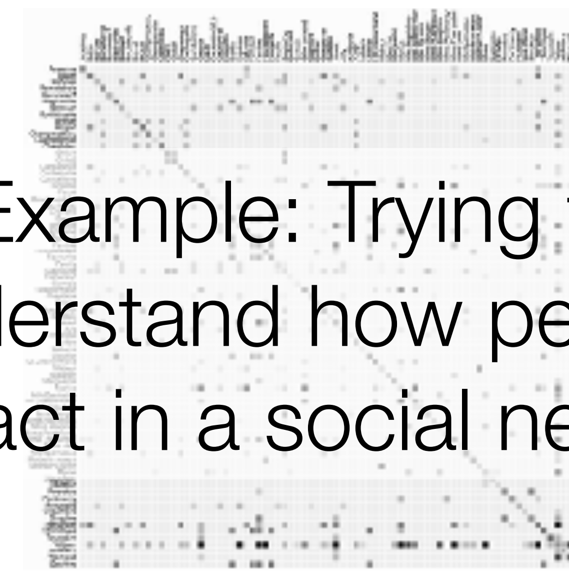
Important:
Handwritten digit demo was a **toy example** where we know which images correspond to digits 0, 1, … 9

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

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

Visualization





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

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

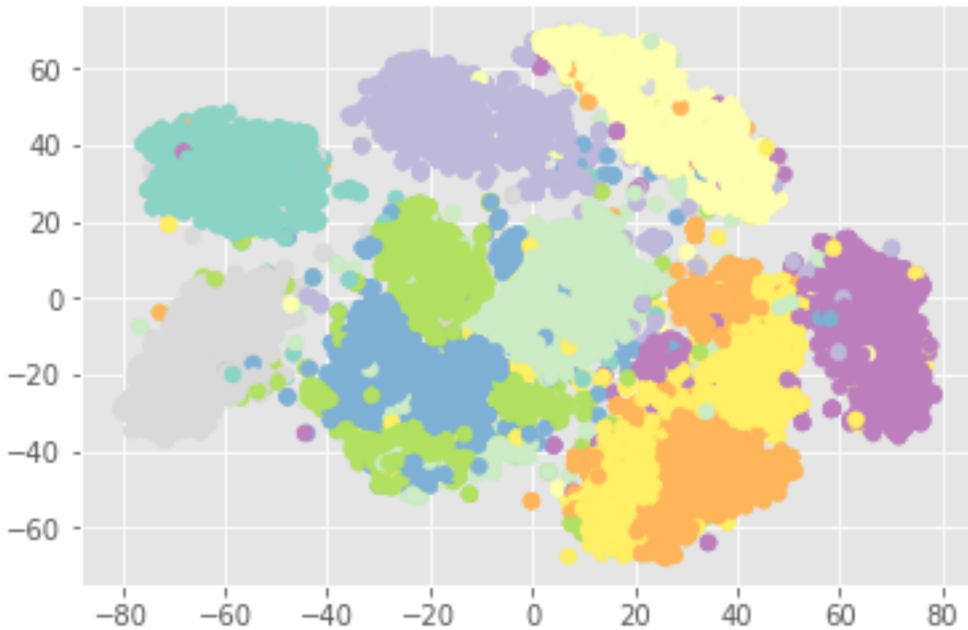
Many real UDA problems:
The data are **messy** and it's not obvious what the "correct" labels/answers look like, and "correct" is ambiguous!

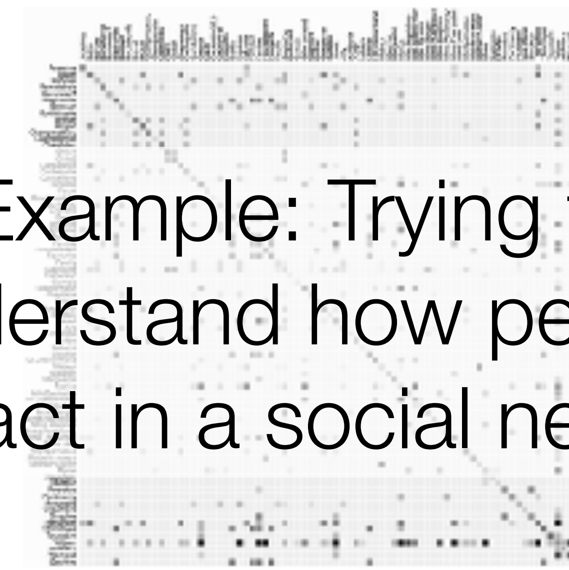
This is largely why I am covering "supervised" methods (require labels) *after* "unsupervised" methods (don't require labels)

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

Visualization

is a way of debugging data analysis!





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

Important:
Handwritten digit demo was 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!

This is largely why I am covering "supervised" methods (require labels) *after* "unsupervised" methods (don't require labels)

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

Dimensionality Reduction for Visualization

Dimensionality Reduction for Visualization

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

Dimensionality Reduction for Visualization

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

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

Dimensionality Reduction for Visualization

- There are *many* methods (I've posted a link on the course webpage to a scikit-learn Swiss roll 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)

Dimensionality Reduction for Visualization

- There are *many* methods (I've posted a link on the course webpage to a scikit-learn Swiss roll 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

Dimensionality Reduction for Visualization

- There are *many* methods (I've posted a link on the course webpage to a scikit-learn Swiss roll 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!