Constructing Maps to Visualize Big Data

Laurens van der Maaten

TU Delft

Introduction

- As a machine learner, the first thing I tell all my students is:

"Have a look at your data!"

Introduction

- As a machine learner, the first thing I tell all my students is:

“Have a look at your data!”

- Good visualizations leverage the human visual system to:
  - Recognize patterns, spot trends, and identify outliers
  - Replace cognitive calculations by perceptual inferences
  - Engage more diverse audiences, *etc.*

Introduction

- Creating a good visualization requires a series of nuanced judgements:
    - What is the task? What type of data do I have? What visual encoding to use?

Introduction

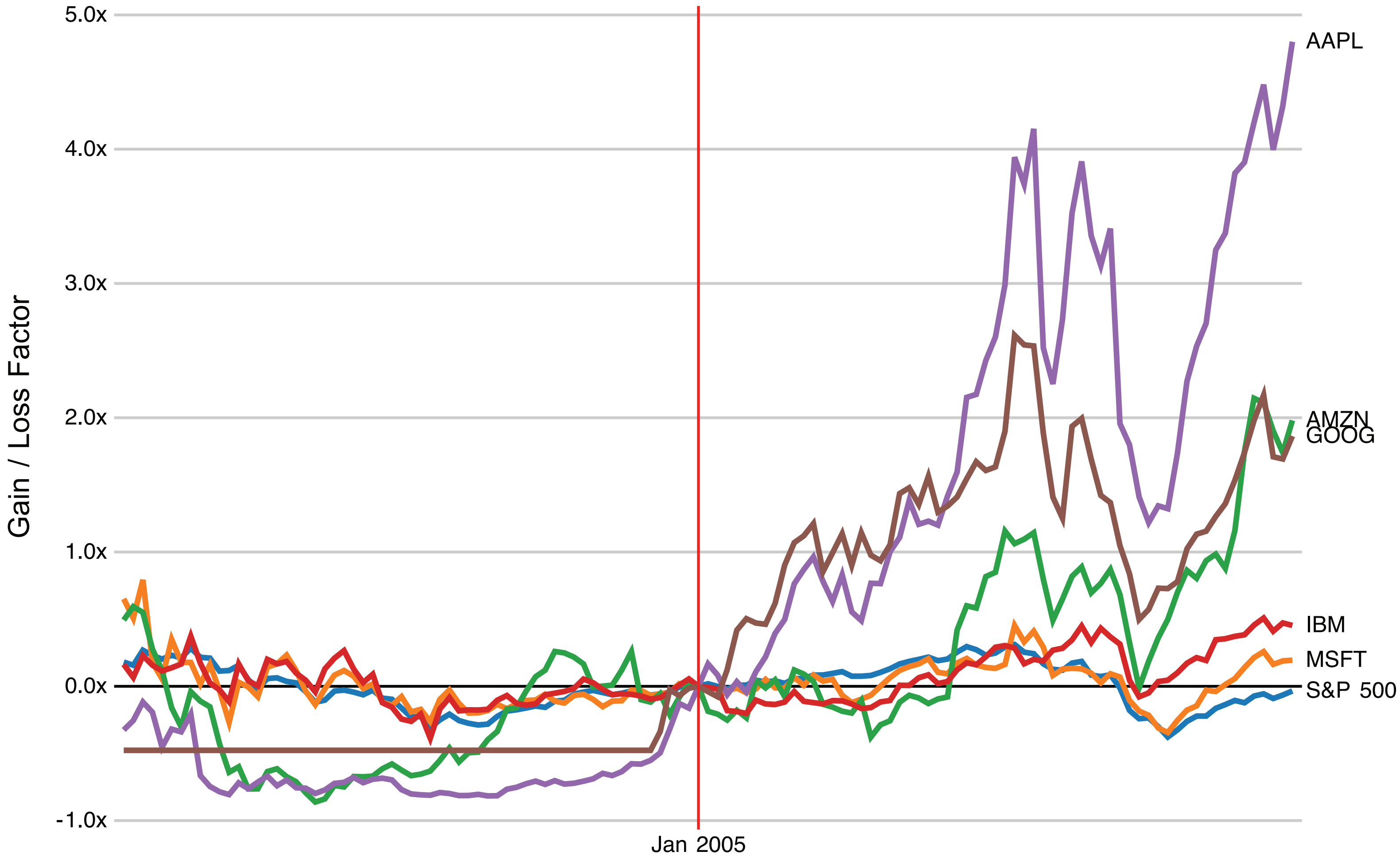
- Creating a good visualization requires a series of nuanced judgements:
  - What is the task? What type of data do I have? What visual encoding to use?
- There are some rules of thumb that can help in selecting visual encodings:
  - For instance, spatial position leads to most accurate coding of numeric data

Introduction

- Creating a good visualization requires a series of nuanced judgements:
  - What is the task? What type of data do I have? What visual encoding to use?
- There are some rules of thumb that can help in selecting visual encodings:
  - For instance, spatial position leads to most accurate coding of numeric data
- Making good visualizations requires a number of iterations

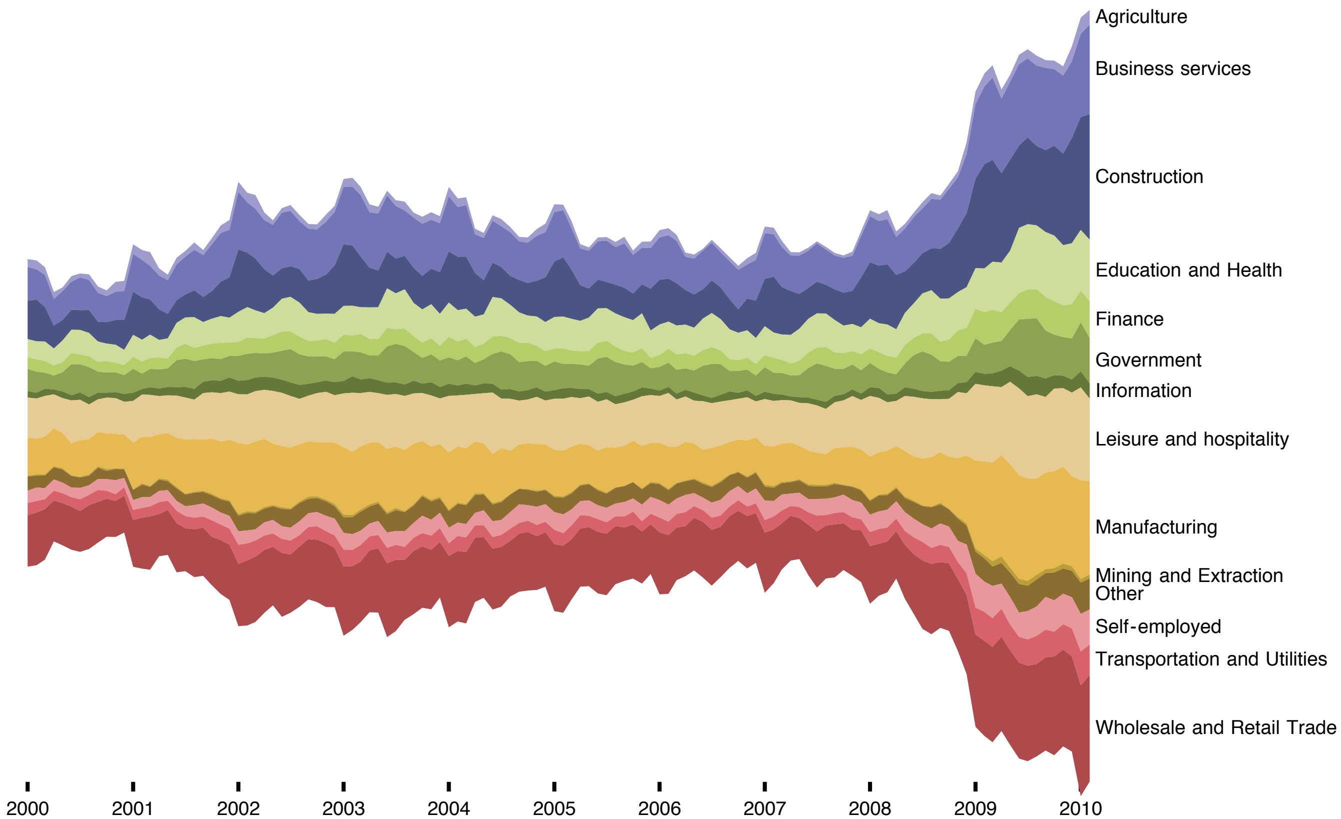
Time series: Index chart

- Displays *relative changes* instead of actual values:



Time series: Stacked graph

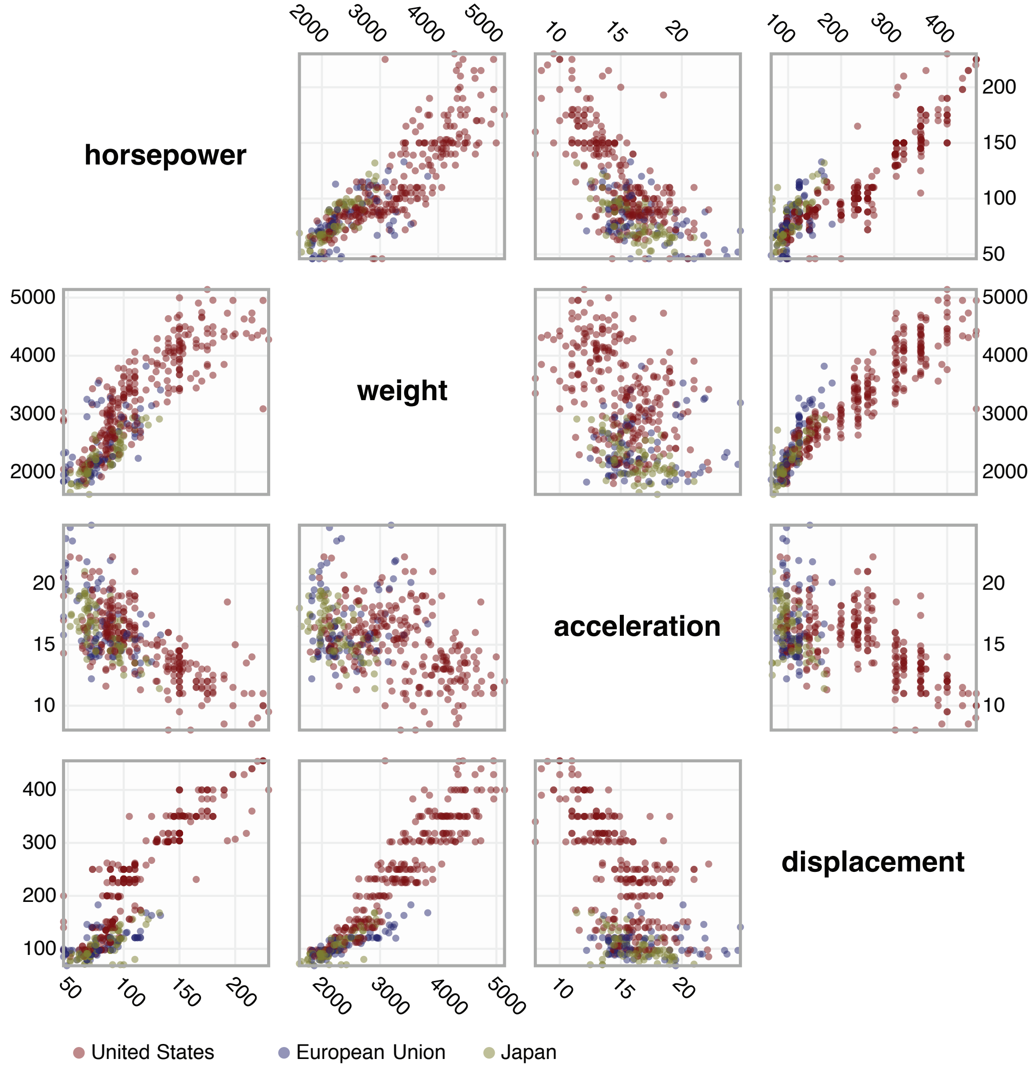
- Shows data in *aggregate*, highlighting *relative changes* in variables:



- Cannot show negative values; hard to interpret trends on top of other curves

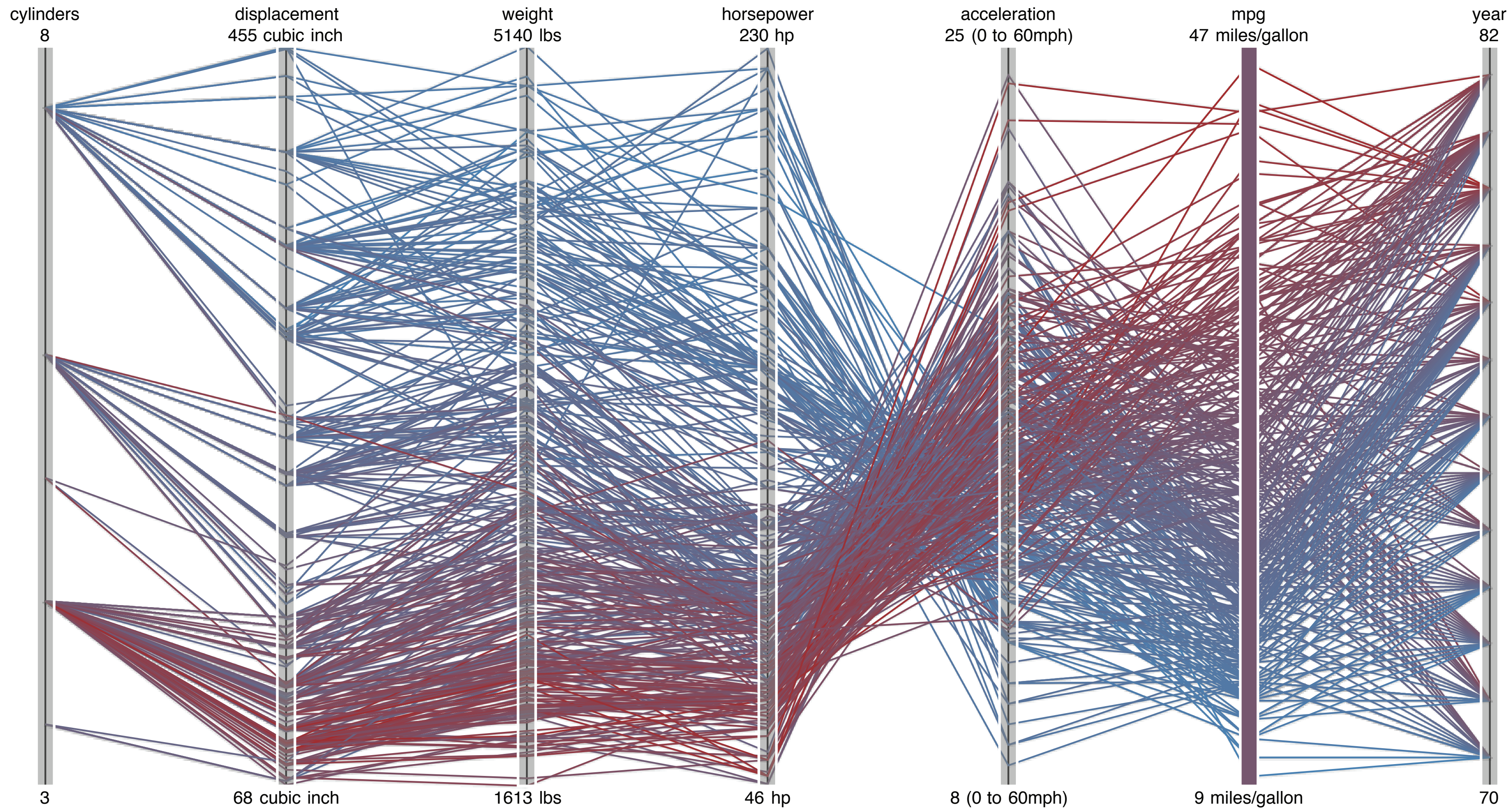
Real-valued data: Scatter plot matrix

- Allows one to quickly spot correlations (if number of variables limited):



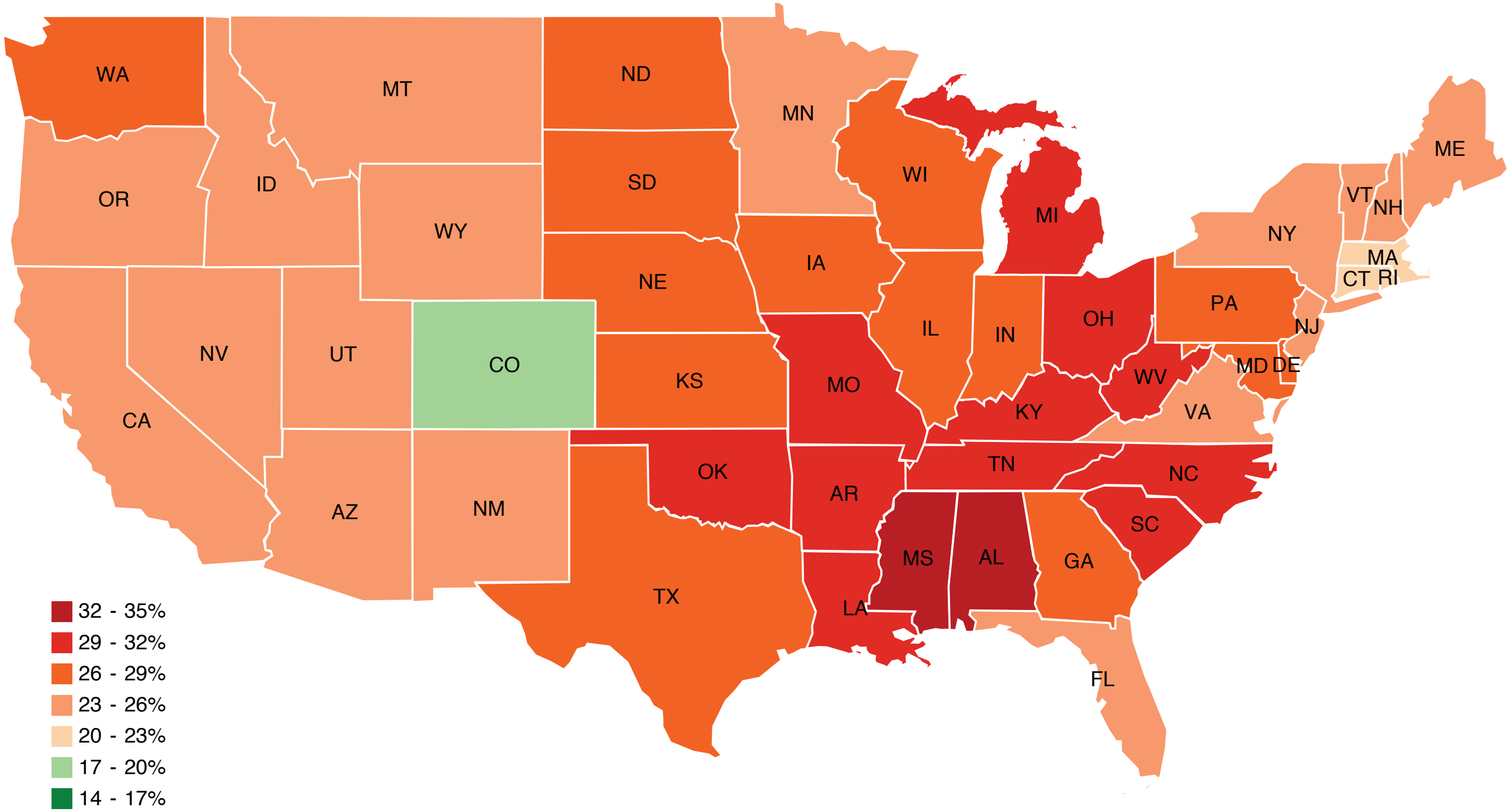
Real-valued data: Parallel coordinates

- Each vertical line corresponds to a variable:



Geographical data: Chloropleth map

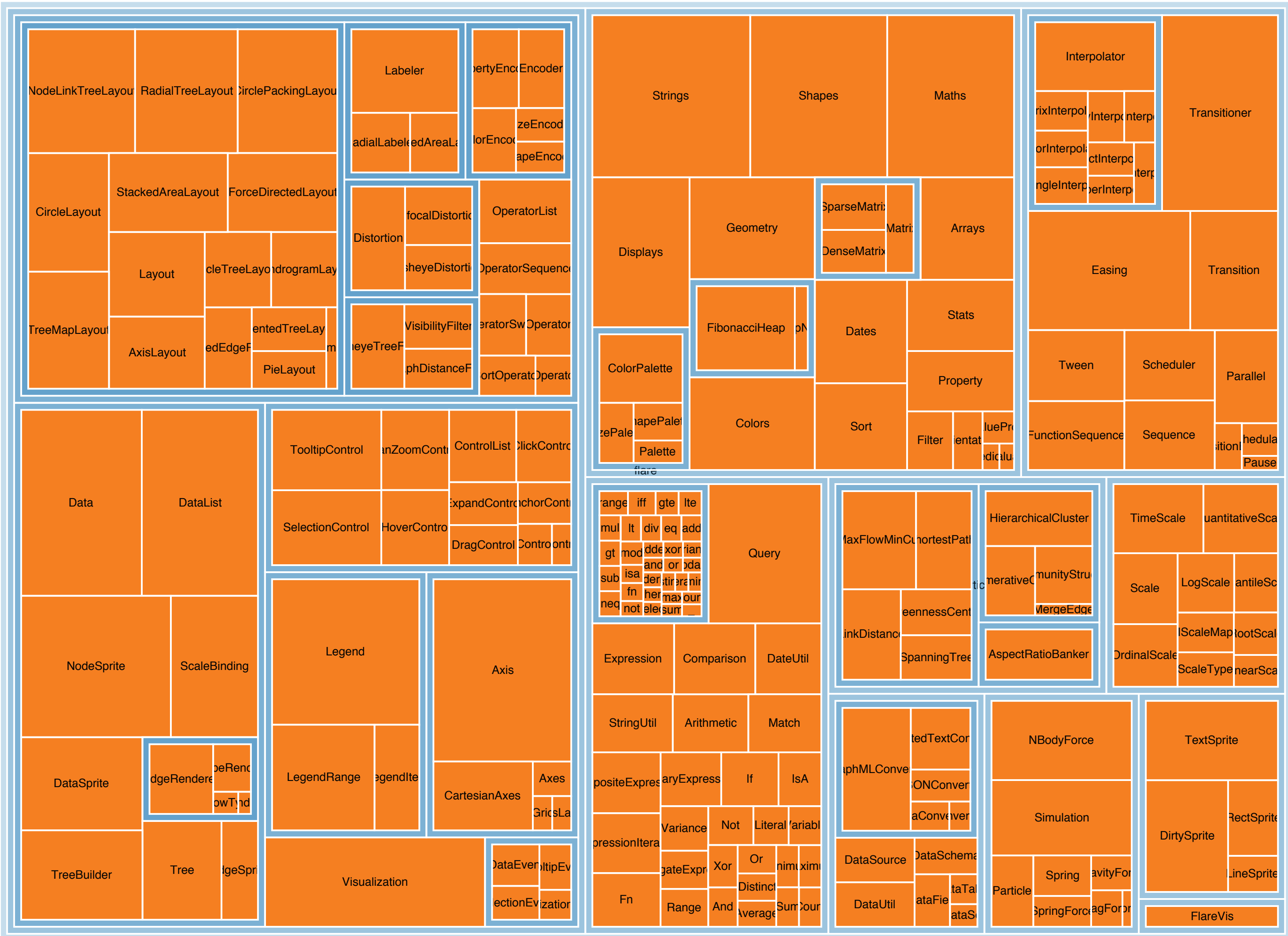
- Visualizes data aggregated by geographic region:



- Disadvantage: perception may be altered by area of geographic region

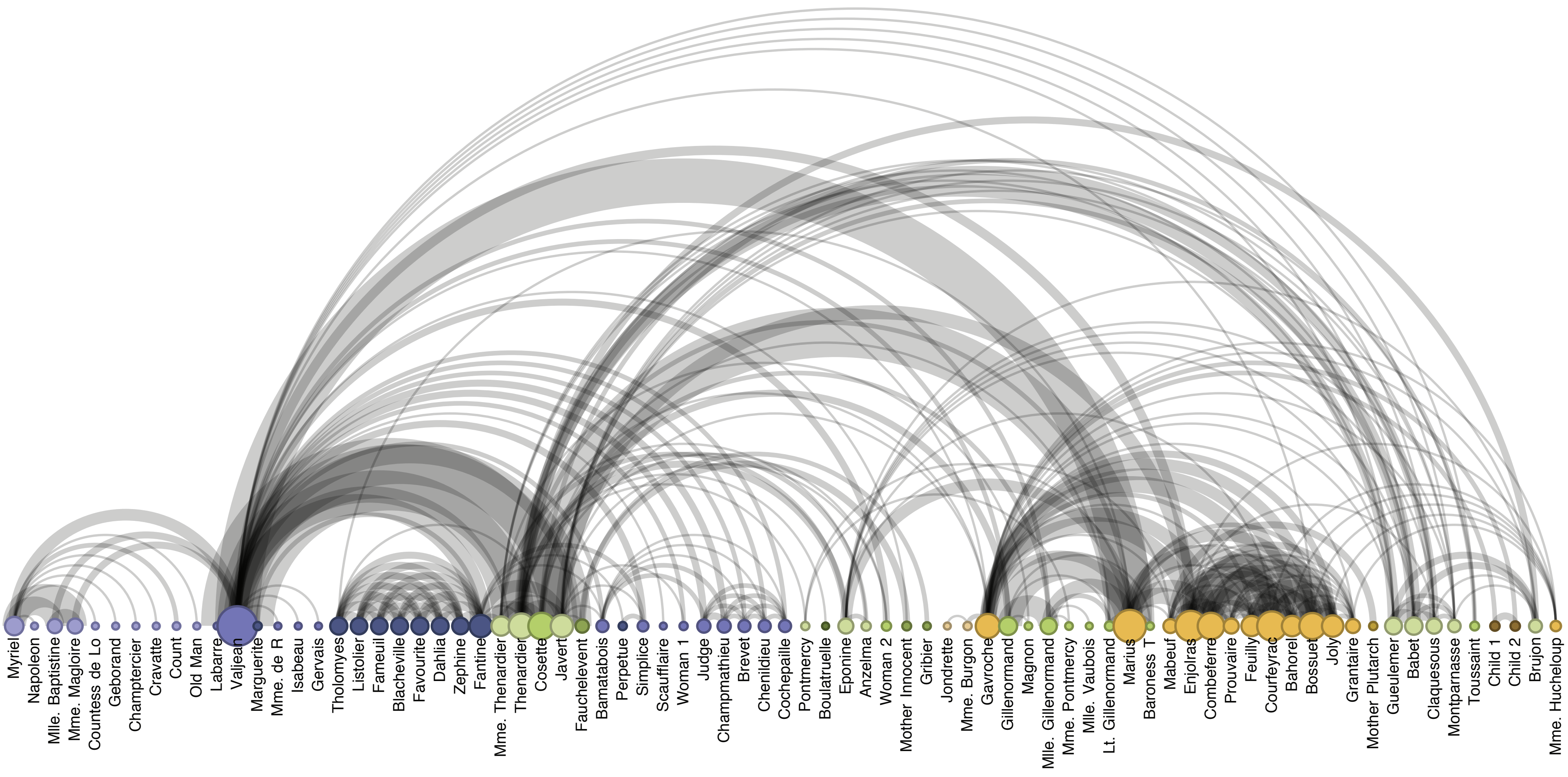
Hierarchies: Treemap / Enclosure diagram

- Effective way to visualize tree with a single variable at each node:



Networks: Arc diagram

- Easy to verify cliques and bridges, but has seriation problem:



Introduction

- Nice overview of techniques is given in *“A Tour of the Visualization Zoo”*

Introduction

- Nice overview of techniques is given in *"A Tour of the Visualization Zoo"*

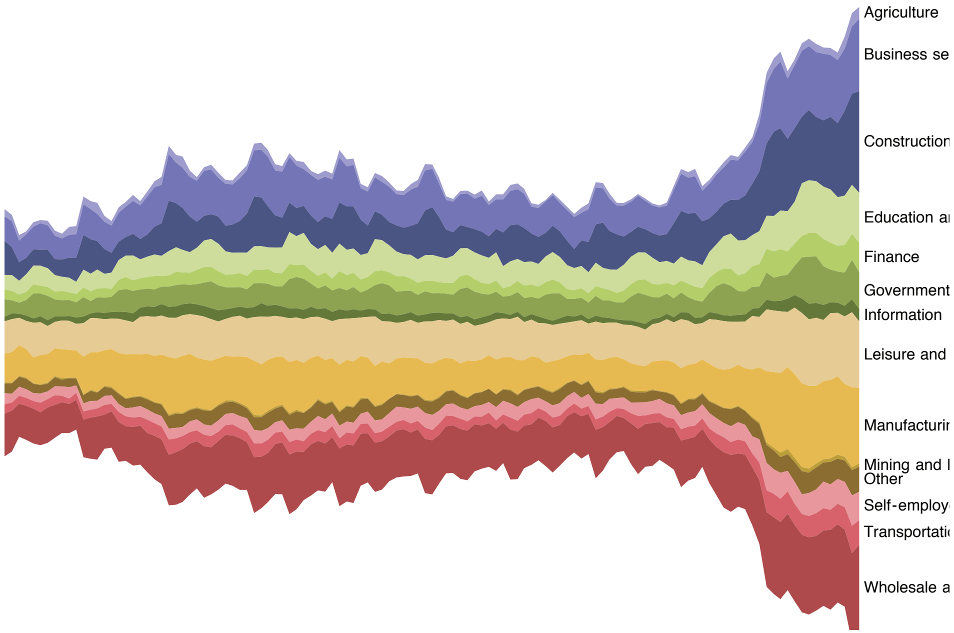
- A plethora of visualization tools / frameworks exist, for instance:
  - d3.js is a popular framework for building web-based visualization
  - VTK is commonly used for 3D and scientific visualization
  - Tools like Matlab, R, Mathematica, and SPSS also provide various visualization tools
  - SynerScope is a Dutch visual-analytics product you will hear more about soon

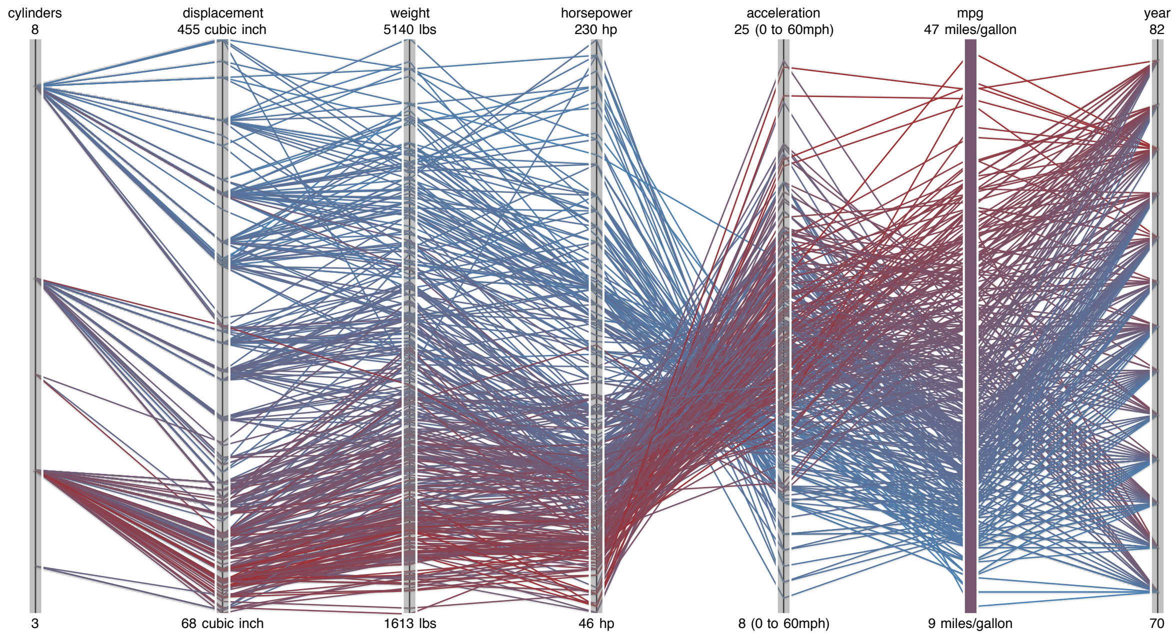
Introduction

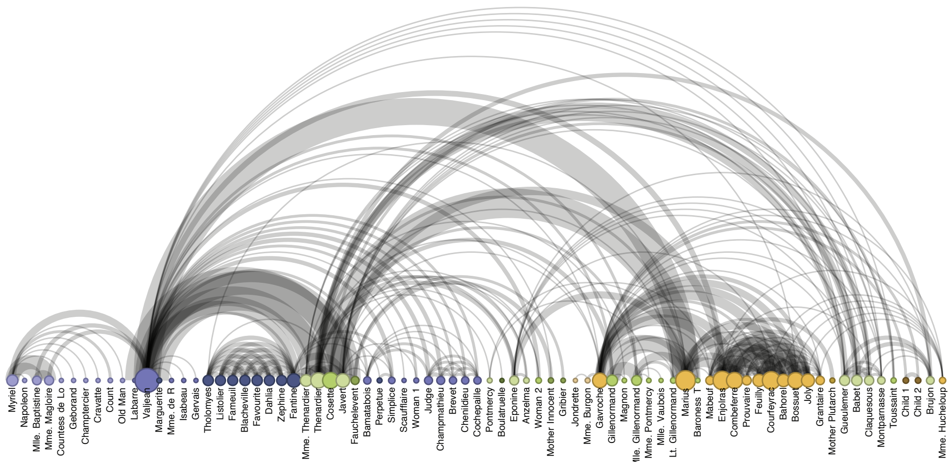
- What does a machine learner see when he looks at this?

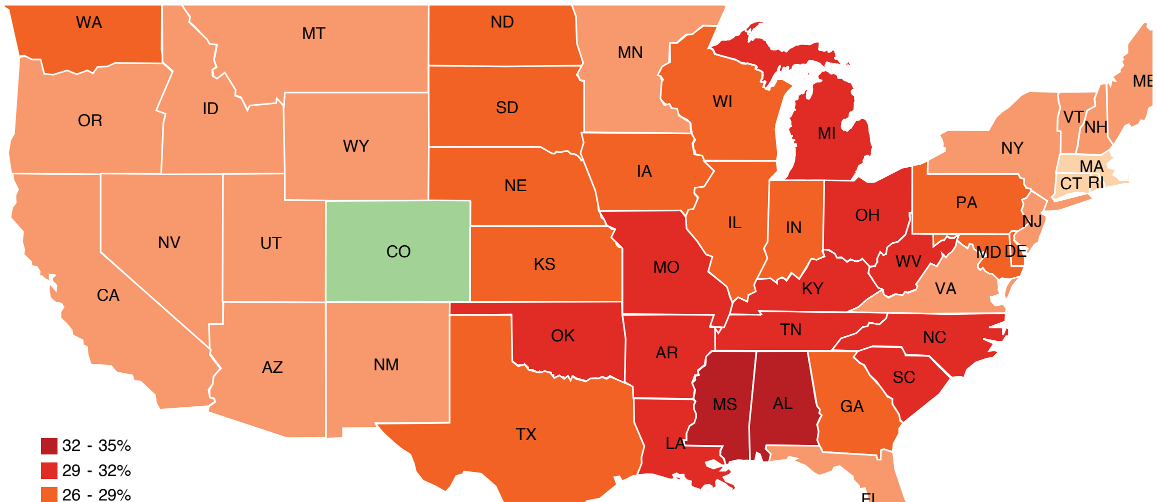
Introduction

- What does a machine learner see when he looks at this?



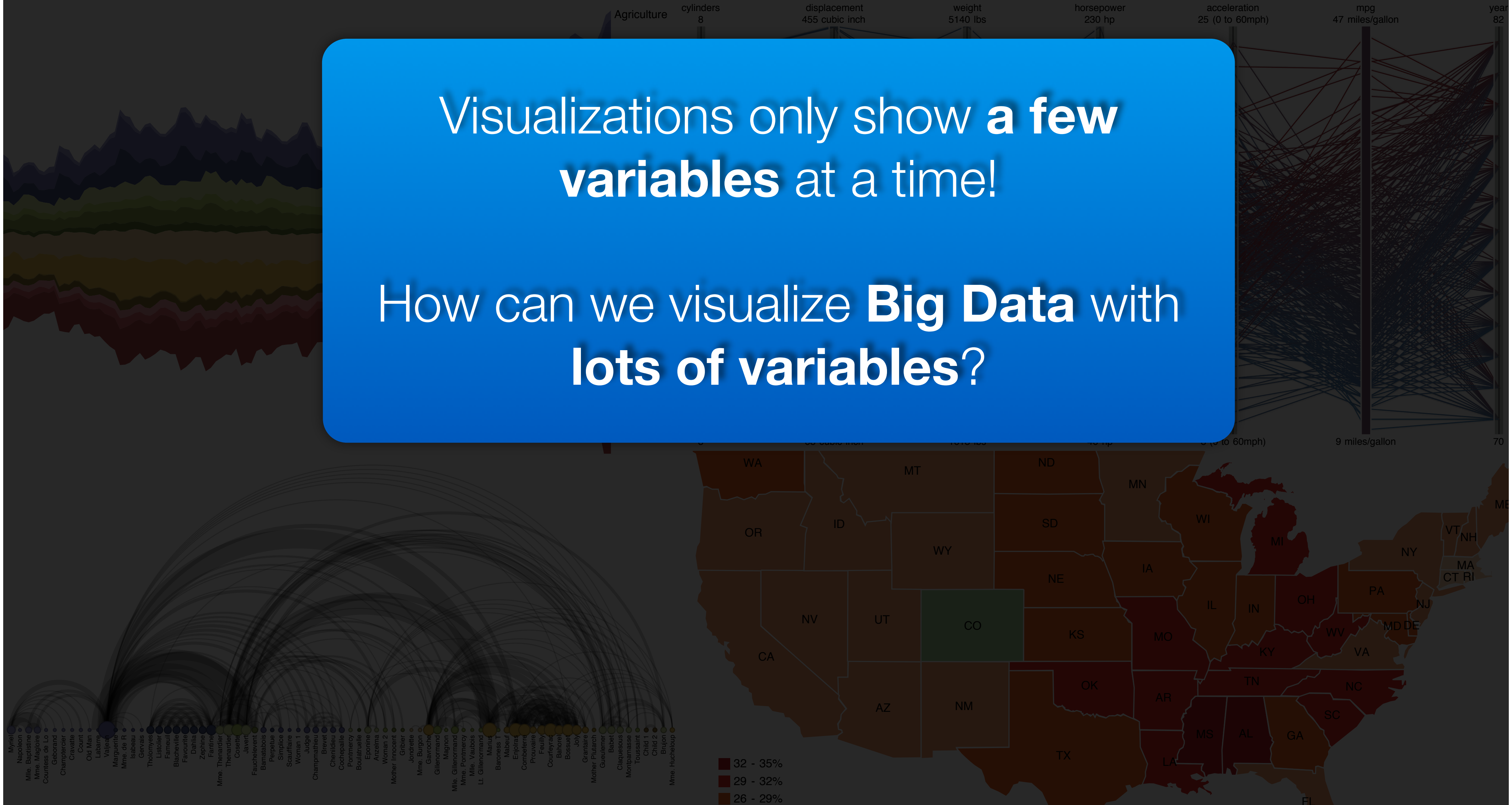






Introduction

- What does a machine learner see when he looks at this?



Visualizations only show **a few variables** at a time!

How can we visualize **Big Data** with **lots of variables**?

Visualizing data by constructing maps

Introduction

- Compute *dissimilarity* of all pairs of records in the database:

Ams	800	1	1	0
Rot	700	3	0	1
Gro	200	1	0	0
Maa	100	0	0	1
Zwo	100	1	0	1

Introduction

- Compute *dissimilarity* of all pairs of records in the database:

Ams	800	1	1	0
Rot	700	3	0	1
Gro	200	1	0	0
Maa	100	0	0	1
Zwo	100	1	0	1

→

	Ams	Rot	Gro	Maa	Zwo
Ams	0	58	147	178	82
Rot		0	202	146	128
Gro			0	270	85
Maa				0	187
Zwo					0

Introduction

- Compute *dissimilarity* of all pairs of records in the database:

Ams	800	1	1	0
Rot	700	3	0	1
Gro	200	1	0	0
Maa	100	0	0	1
Zwo	100	1	0	1

→

	Ams	Rot	Gro	Maa	Zwo
Ams	0	58	147	178	82
Rot		0	202	146	128
Gro			0	270	85
Maa				0	187
Zwo					0

- Exact way in which dissimilarities are computed depends on problem at hand

Introduction

- Build *map* in which each point represents a database record, and distances between points reflect similarities in the data:

	Ams	Rot	Gro	Maa	Zwo
Ams	0	58	147	178	82
Rot		0	202	146	128
Gro			0	270	85
Maa				0	187
Zwo					0

Introduction

- Build *map* in which each point represents a database record, and distances between points reflect similarities in the data:

	Ams	Rot	Gro	Maa	Zwo
Ams	0	58	147	178	82
Rot		0	202	146	128
Gro			0	270	85
Maa				0	187
Zwo					0

How to build a map from a distance matrix?

Principal components analysis

- Principal components analysis (PCA) maps the data onto a *linear subspace*, such that the *variance* of the projected data is maximized

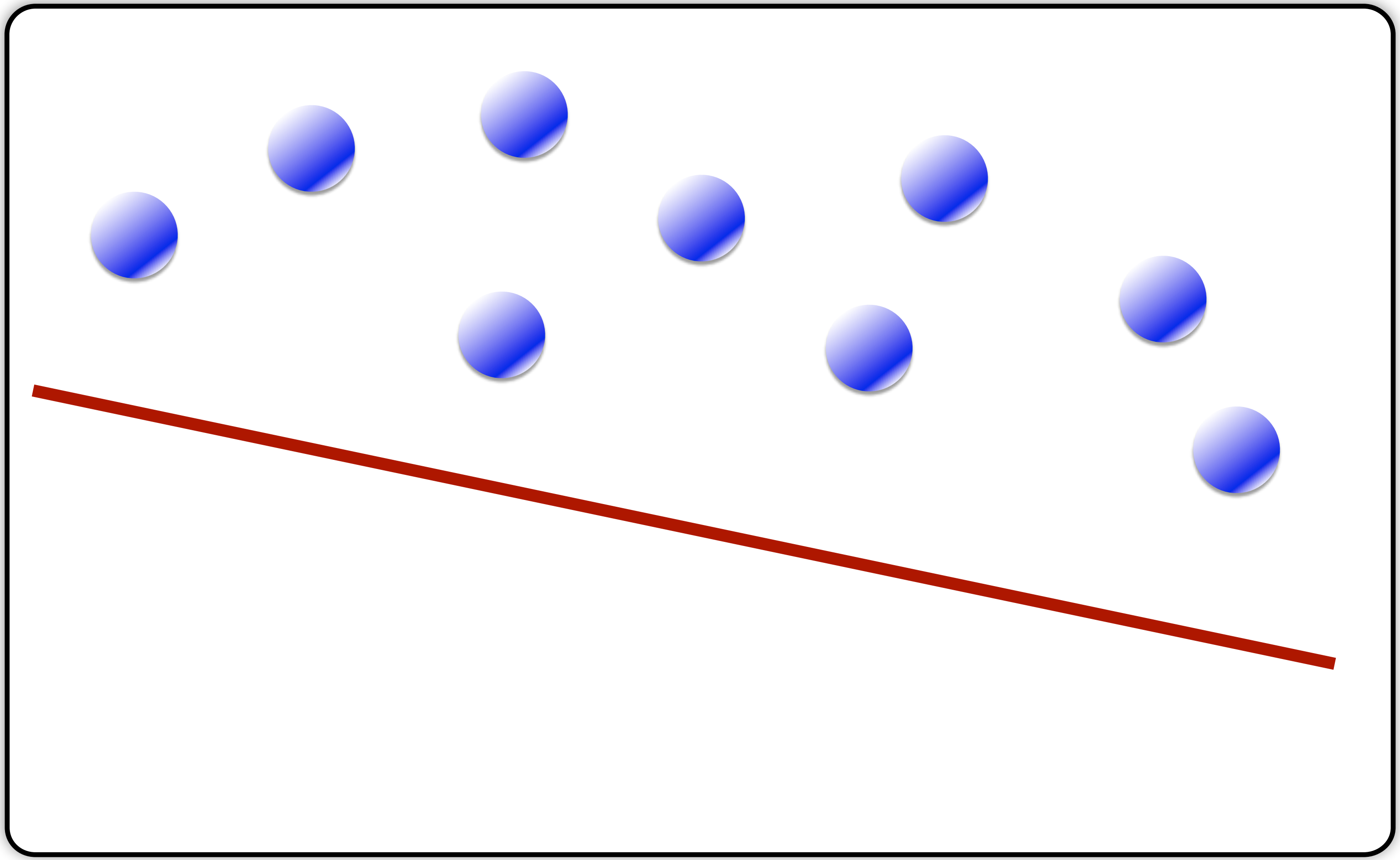
Principal components analysis

- Principal components analysis (PCA) maps the data onto a *linear subspace*, such that the *variance* of the projected data is maximized

- So PCA performs the following maximization: $\max_{\|\mathbf{w}\|^2=1} var(\mathbf{w}^T \mathbf{X})$

Principal components analysis

- Principal components analysis (PCA) maps the data onto a *linear subspace*, such that the *variance* of the projected data is maximized



$\mathbf{w}^{\mathrm{T}}\mathbf{x}$

Principal components analysis

- The objective is to maximize variance: $\max_{\|\mathbf{w}\|^2=1} var(\mathbf{w}^T\mathbf{X})$

Principal components analysis

- The objective is to maximize variance: $\max_{\|\mathbf{w}\|^2=1} var(\mathbf{w}^T\mathbf{X})$
- Assuming zero-mean data: $var(\mathbf{w}^T\mathbf{X}) = [\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}] = [\mathbf{w}^T\mathbf{C}\mathbf{w}]$

Principal components analysis

- The objective is to maximize variance: $\max_{\|\mathbf{w}\|^2=1} var(\mathbf{w}^T\mathbf{X})$
- Assuming zero-mean data: $var(\mathbf{w}^T\mathbf{X}) = [\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}] = [\mathbf{w}^T\mathbf{C}\mathbf{w}]$
- Enforce constraint using *Lagrange multiplier*:

$$\max_{\|\mathbf{w}\|^2=1} var(\mathbf{w}^T\mathbf{X}) = \max_{\mathbf{w},\lambda} \mathbf{w}^T\mathbf{C}\mathbf{w} - \lambda(1 - \mathbf{w}^T\mathbf{w})$$

Principal components analysis

- The objective is to maximize variance: $\max_{\|\mathbf{w}\|^2=1} var(\mathbf{w}^T\mathbf{X})$
- Assuming zero-mean data: $var(\mathbf{w}^T\mathbf{X}) = [\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}] = [\mathbf{w}^T\mathbf{C}\mathbf{w}]$
- Enforce constraint using *Lagrange multiplier*:

$$\max_{\|\mathbf{w}\|^2=1} var(\mathbf{w}^T\mathbf{X}) = \max_{\mathbf{w},\lambda} \mathbf{w}^T\mathbf{C}\mathbf{w} - \lambda(1 - \mathbf{w}^T\mathbf{w})$$

- Set gradient with respect to $\mathbf{w}$ to zero: $\mathbf{C}\mathbf{w} - \lambda\mathbf{w} = 0$

$$\mathbf{C}\mathbf{w} = \lambda\mathbf{w}$$

Classical (multi-dimensional) scaling

- PCA is identical to the following classical scaling algorithm:
  - Obtain a (squared) Euclidean distance matrix for your data

Classical (multi-dimensional) scaling

- PCA is identical to the following classical scaling algorithm:
  - Obtain a (squared) Euclidean distance matrix for your data
  - Perform a “centering” operation that essentially computes the inner product matrix of the original data (note that $\|\mathbf{x}-\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\mathbf{x}^\top\mathbf{y}$)

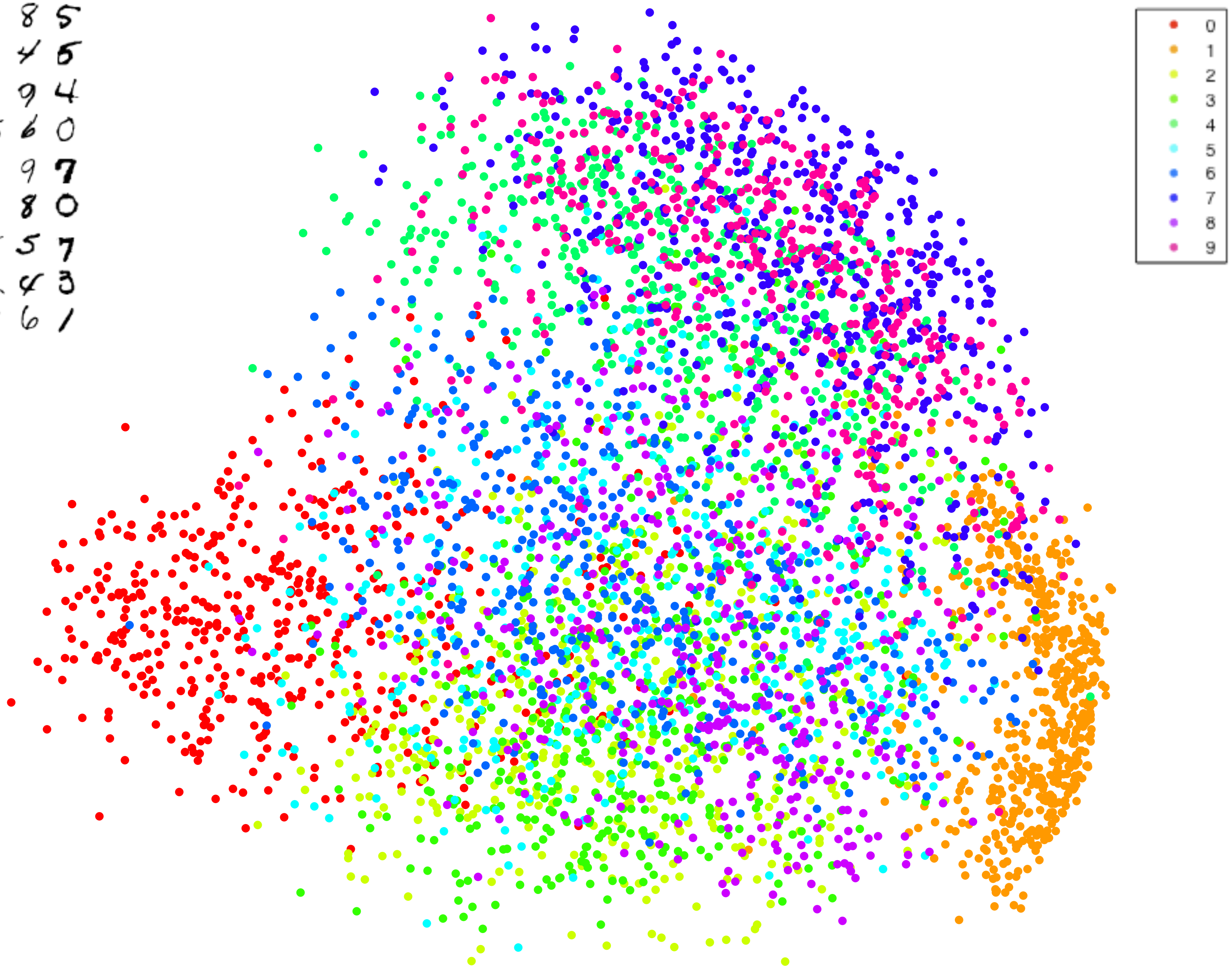
Classical (multi-dimensional) scaling

- PCA is identical to the following classical scaling algorithm:
  - Obtain a (squared) Euclidean distance matrix for your data
  - Perform a "centering" operation that essentially computes the inner product matrix of the original data (note that $\|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\mathbf{x}^\top \mathbf{y}$)
  - Compute the principal eigenvectors of the resulting matrix

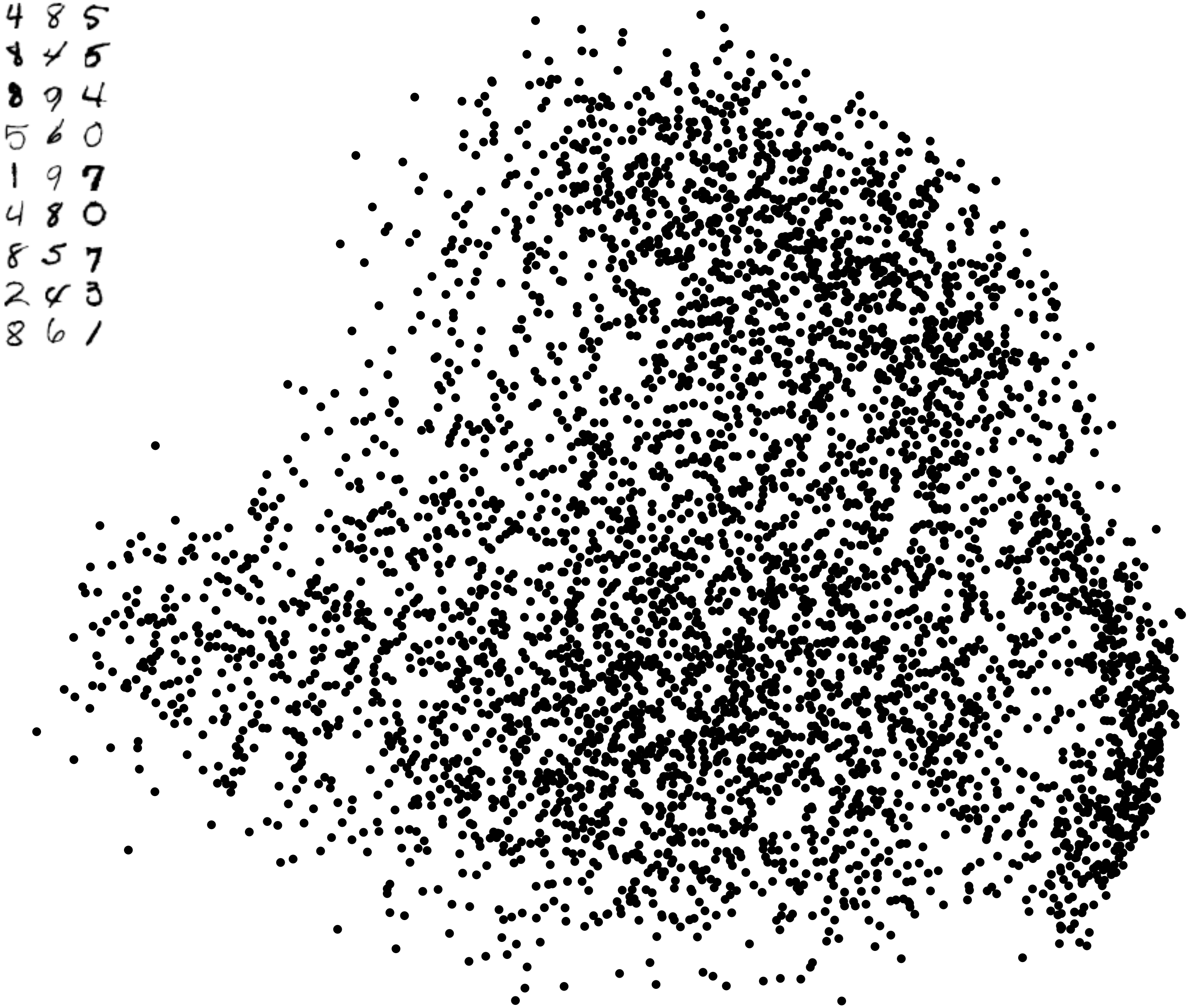
Classical (multi-dimensional) scaling

- PCA is identical to the following classical scaling algorithm:
  - Obtain a (squared) Euclidean distance matrix for your data
  - Perform a "centering" operation that essentially computes the inner product matrix of the original data (note that $\|\mathbf{x}-\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\mathbf{x}^\top\mathbf{y}$)
  - Compute the principal eigenvectors of the resulting matrix
- These eigenvectors are identical to the projected data computed by PCA:
  - Identity is due to a relation between eigenvectors of inner and outer products

Principal components analysis



Principal components analysis



What distances to preserve?

- PCA is mainly concerned with preserving *large* pairwise distances in the map

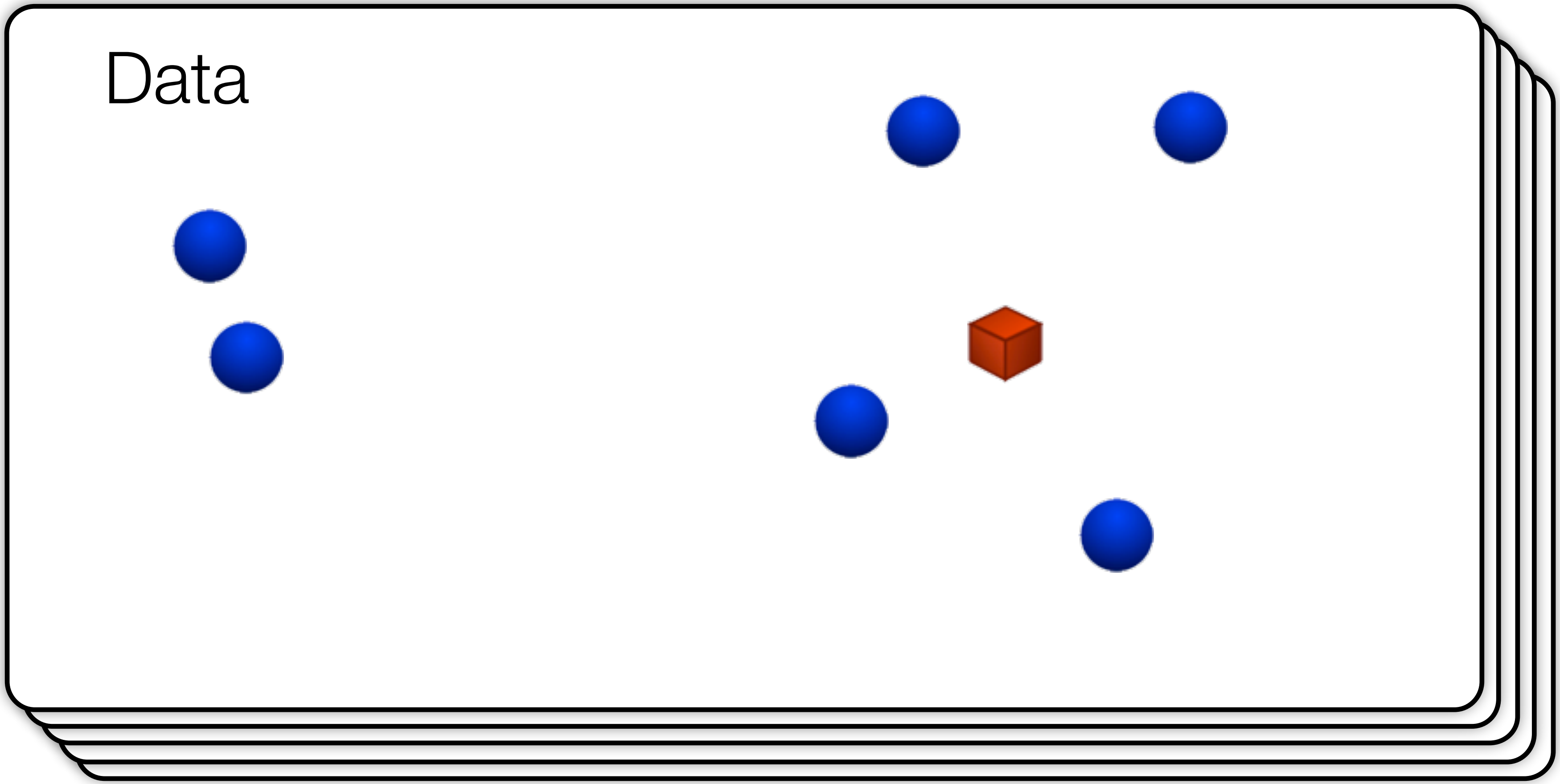
What distances to preserve?

- PCA is mainly concerned with preserving *large* pairwise distances in the map

- Large pairwise distances, however, are relatively unimportant in visualizations
    - Do we really care exactly how dissimilar zeros and ones are?

t-Distributed Stochastic Neighbor Embedding

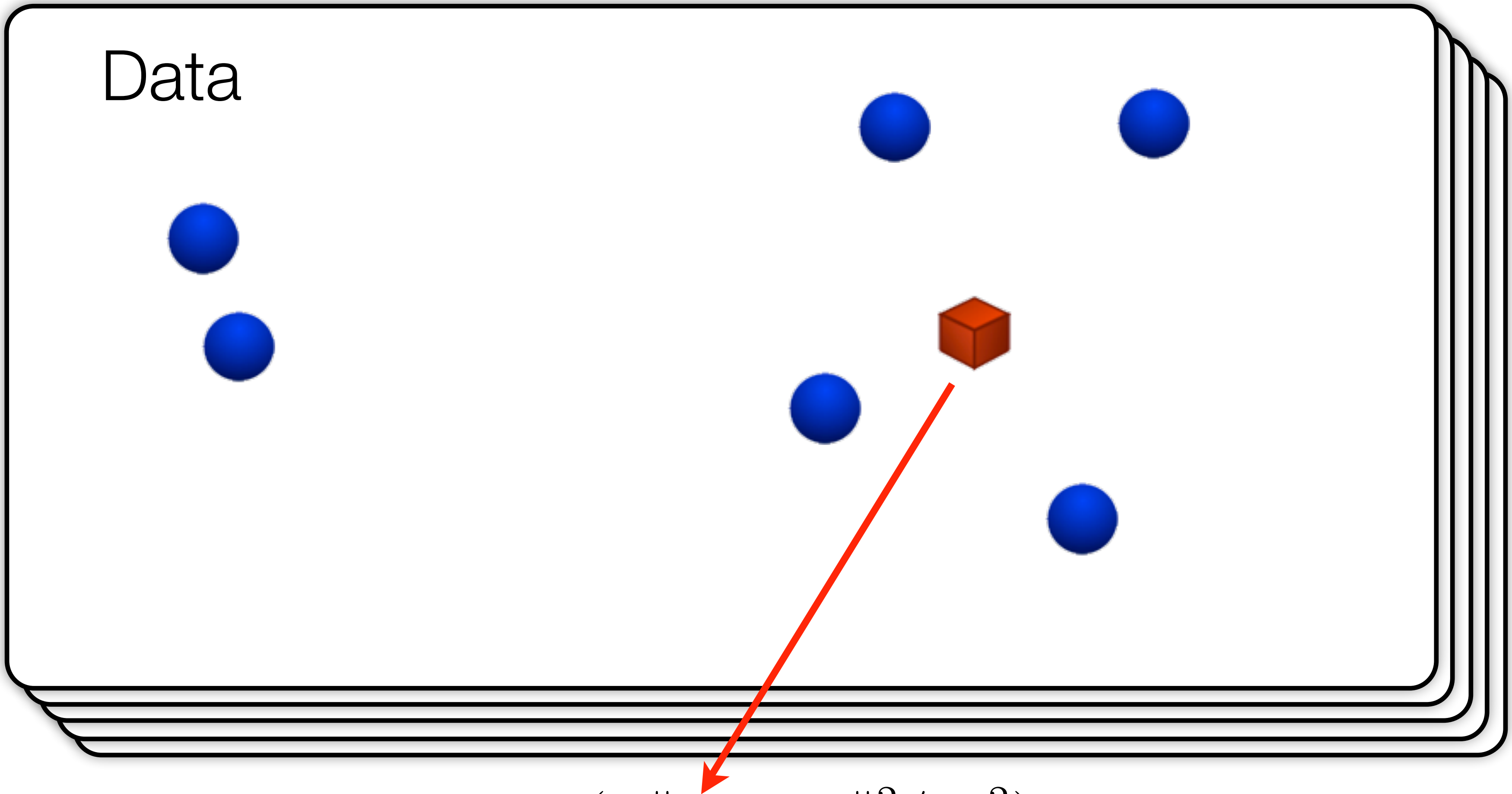
- Compute "local" pairwise similarities based on pairwise distances:



$$p_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{x}_k - \mathbf{x}_l\|^2/2\sigma^2)}$$

t-Distributed Stochastic Neighbor Embedding

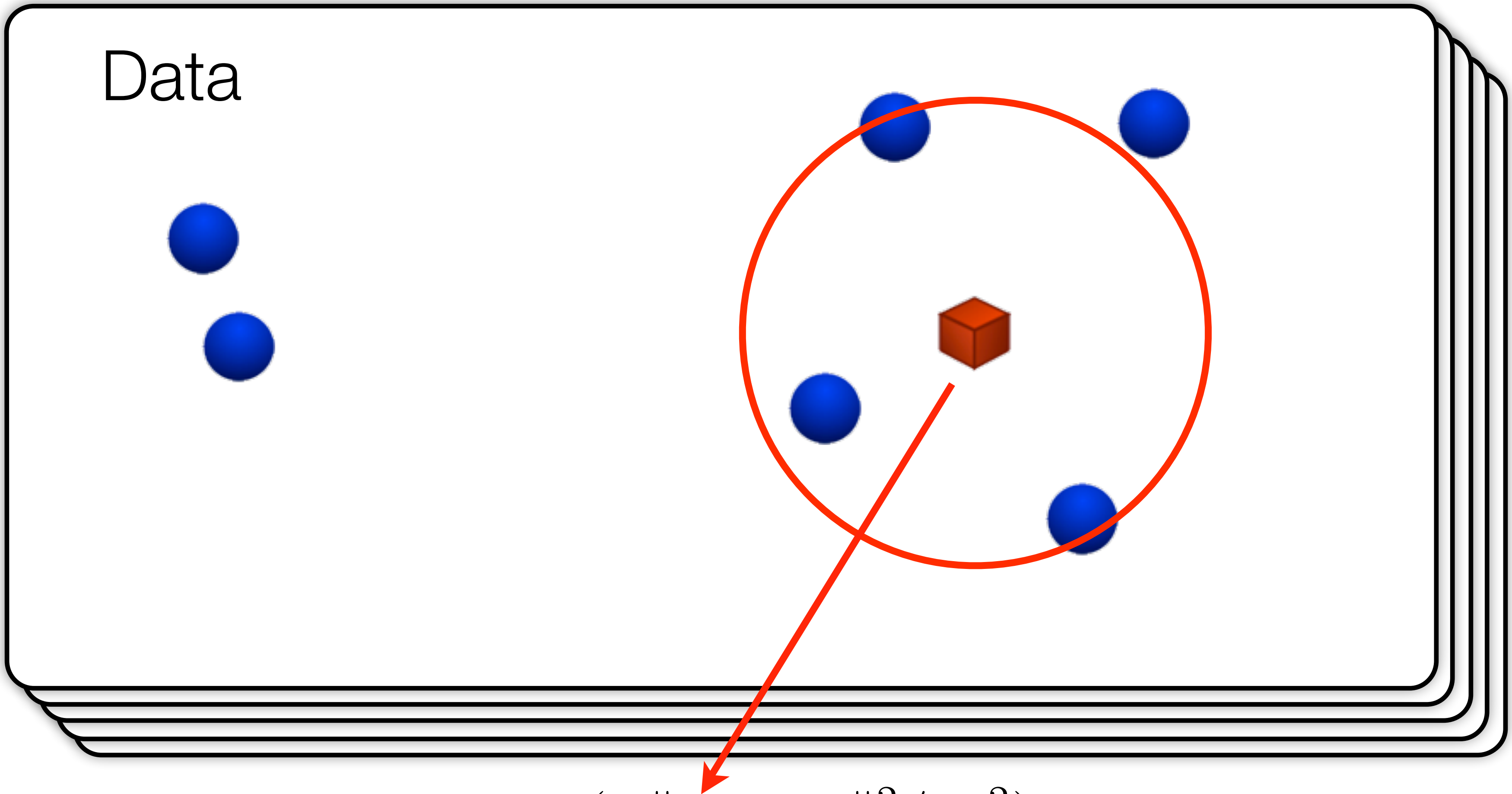
- Compute "local" pairwise similarities based on pairwise distances:



$$p_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{x}_k - \mathbf{x}_l\|^2/2\sigma^2)}$$

t-Distributed Stochastic Neighbor Embedding

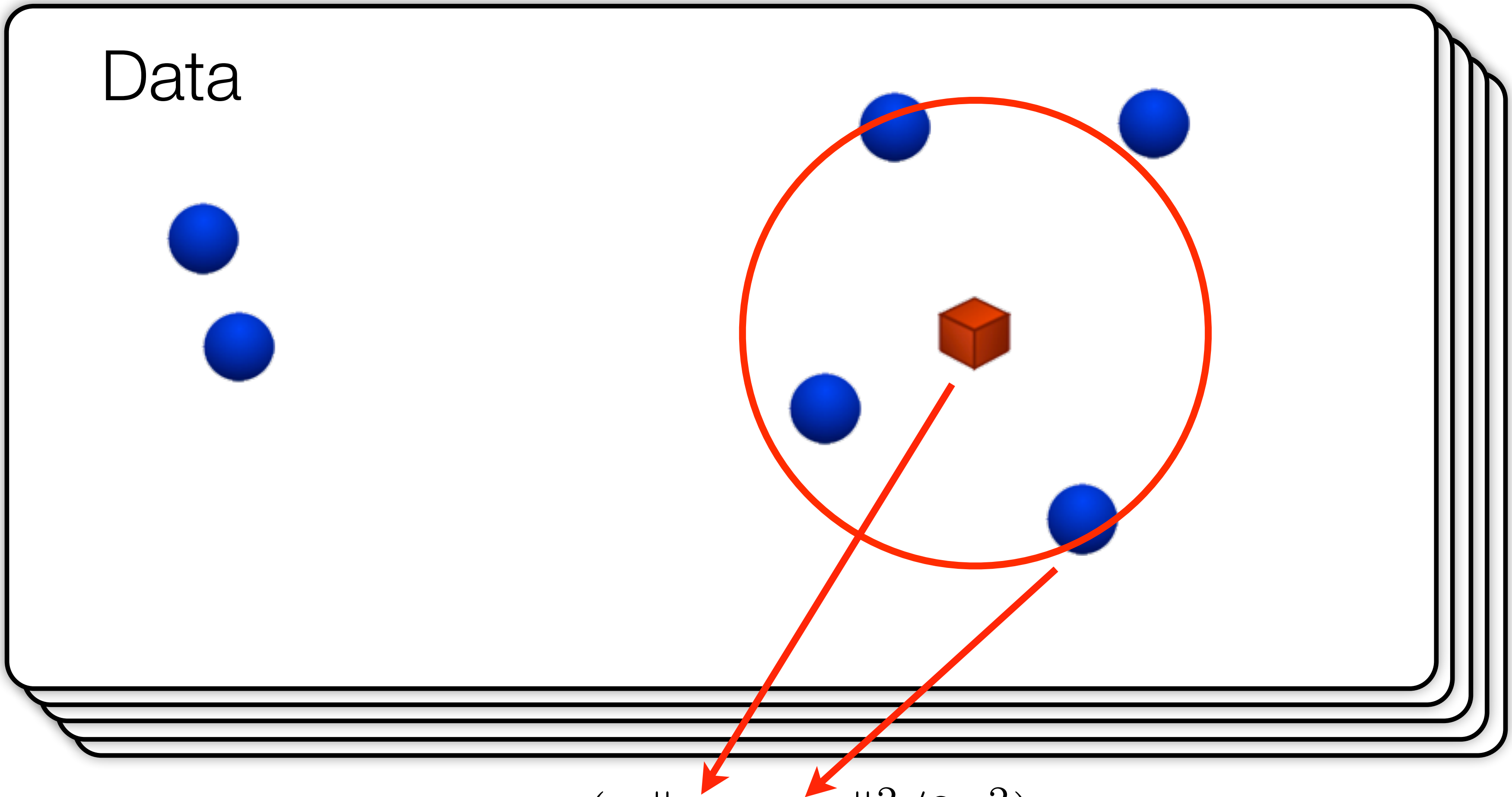
- Compute "local" pairwise similarities based on pairwise distances:



$$p_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{x}_k - \mathbf{x}_l\|^2/2\sigma^2)}$$

t-Distributed Stochastic Neighbor Embedding

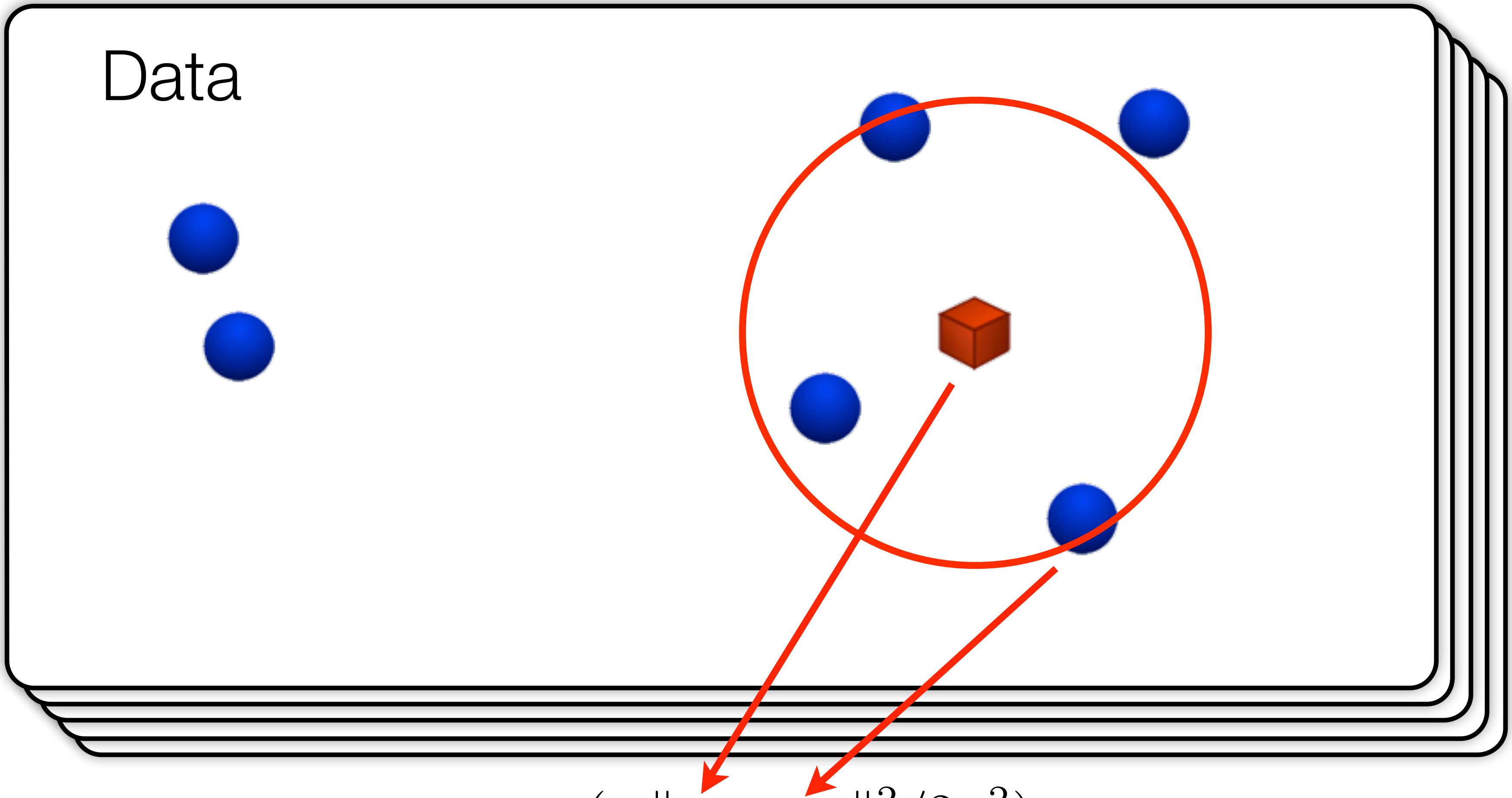
- Compute “local” pairwise similarities based on pairwise distances:



$$p_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{x}_k - \mathbf{x}_l\|^2 / 2\sigma^2)}$$

t-Distributed Stochastic Neighbor Embedding

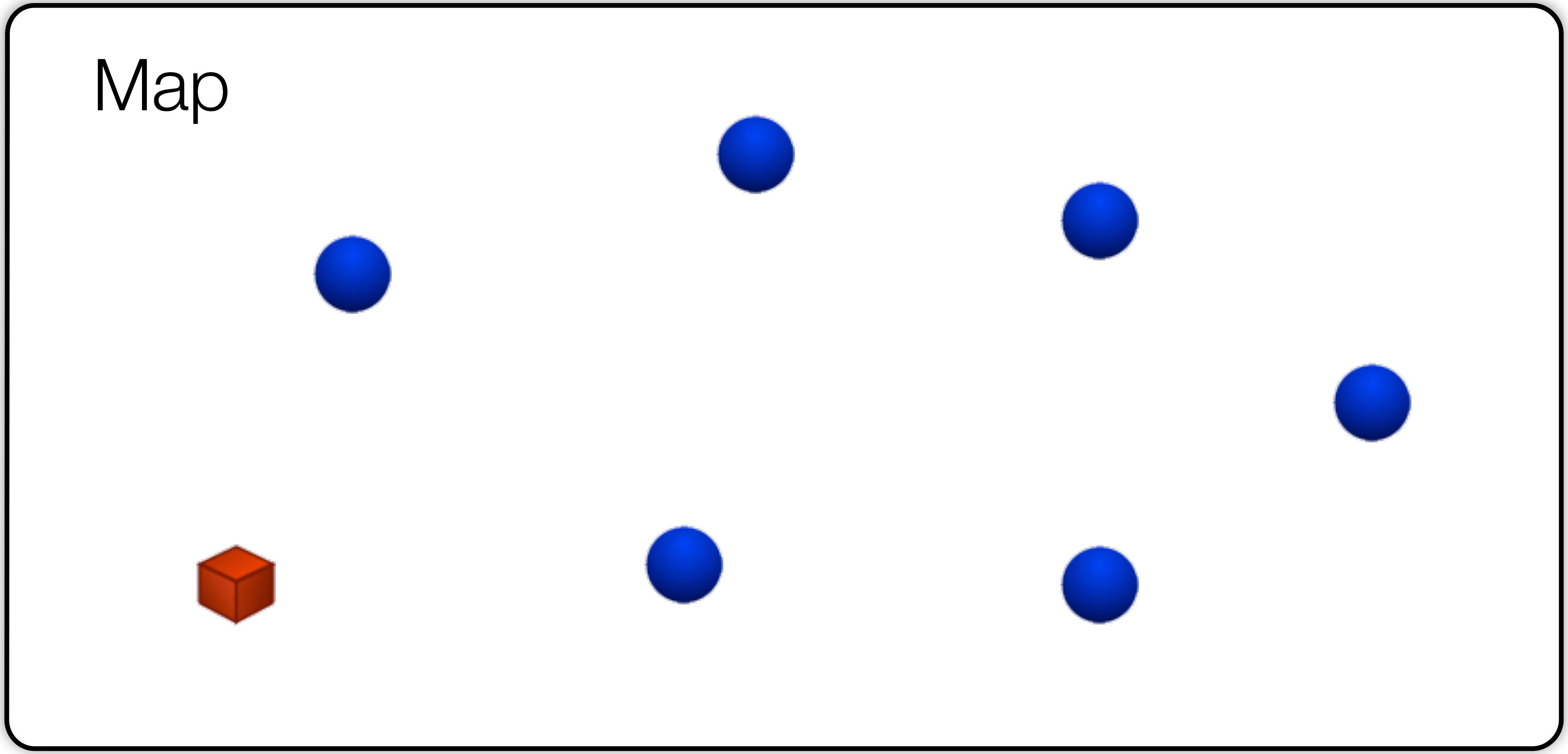
- Compute "local" pairwise similarities based on pairwise distances:



$$p_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)}{\sum_k \sum_{l \neq k} \exp(-\|\mathbf{x}_k - \mathbf{x}_l\|^2/2\sigma^2)}$$

t-Distributed Stochastic Neighbor Embedding

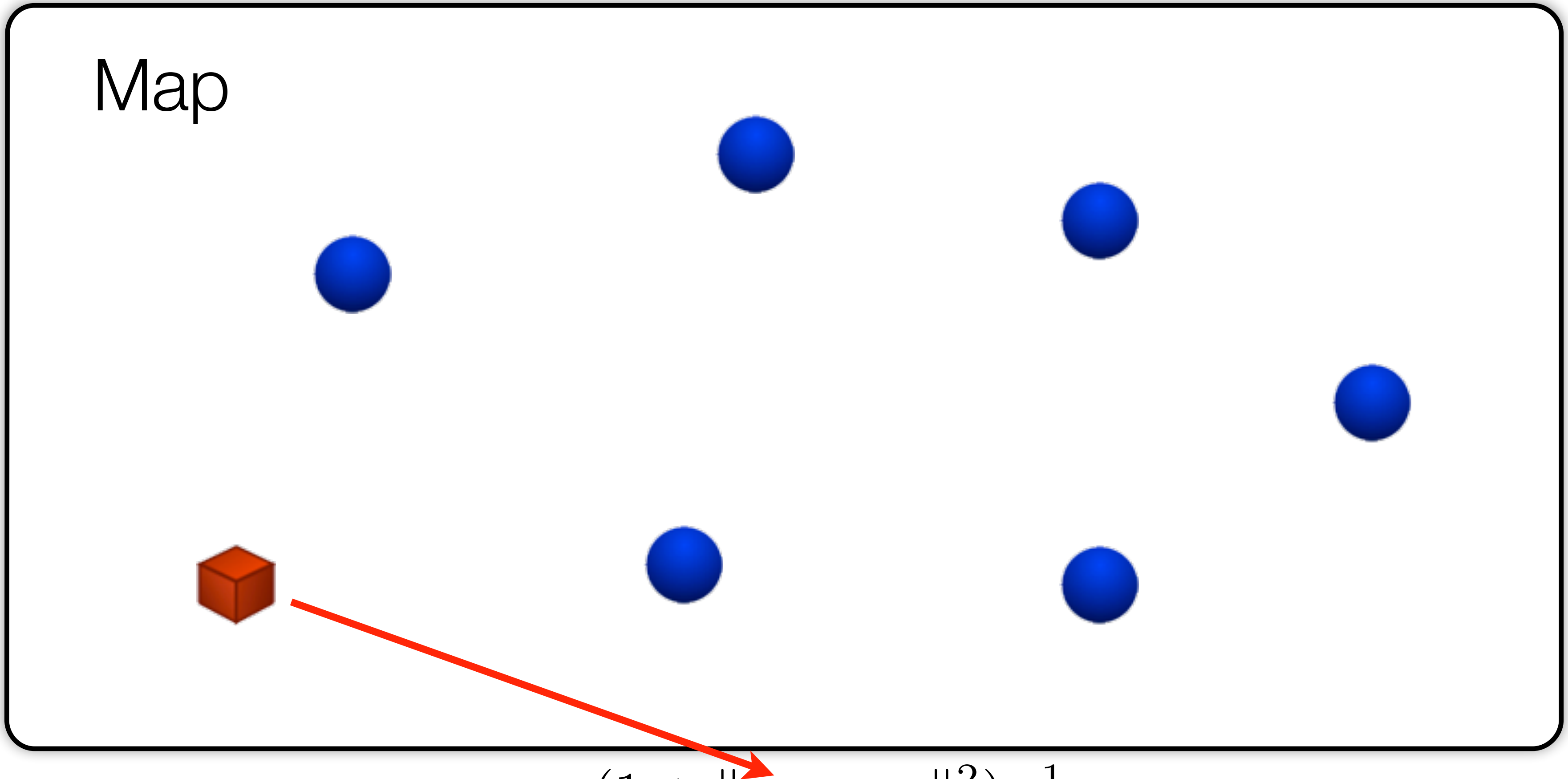
- Measure pairwise similarities between corresponding points in the map:



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

t-Distributed Stochastic Neighbor Embedding

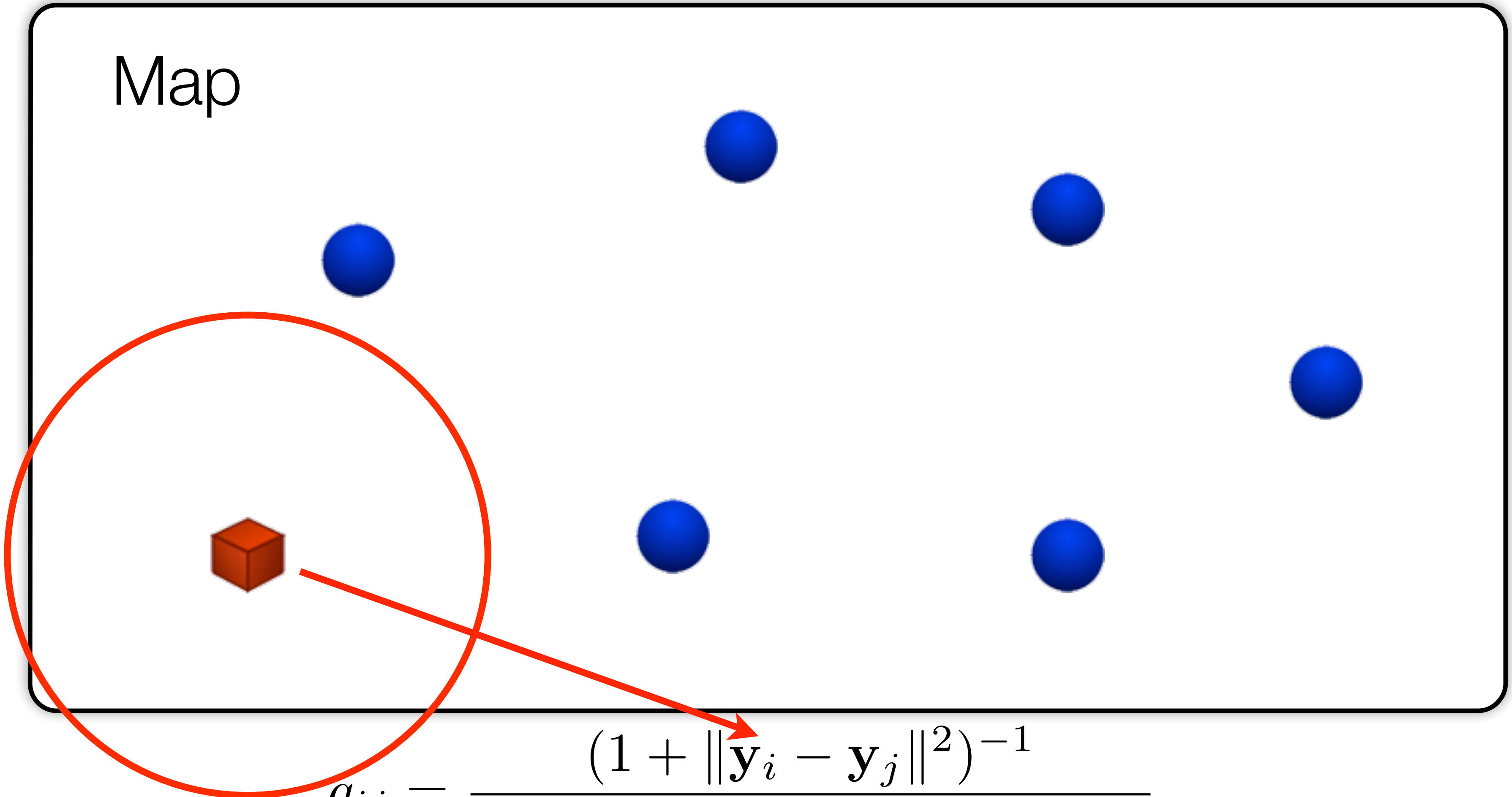
- Measure pairwise similarities between corresponding points in the map:



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

t-Distributed Stochastic Neighbor Embedding

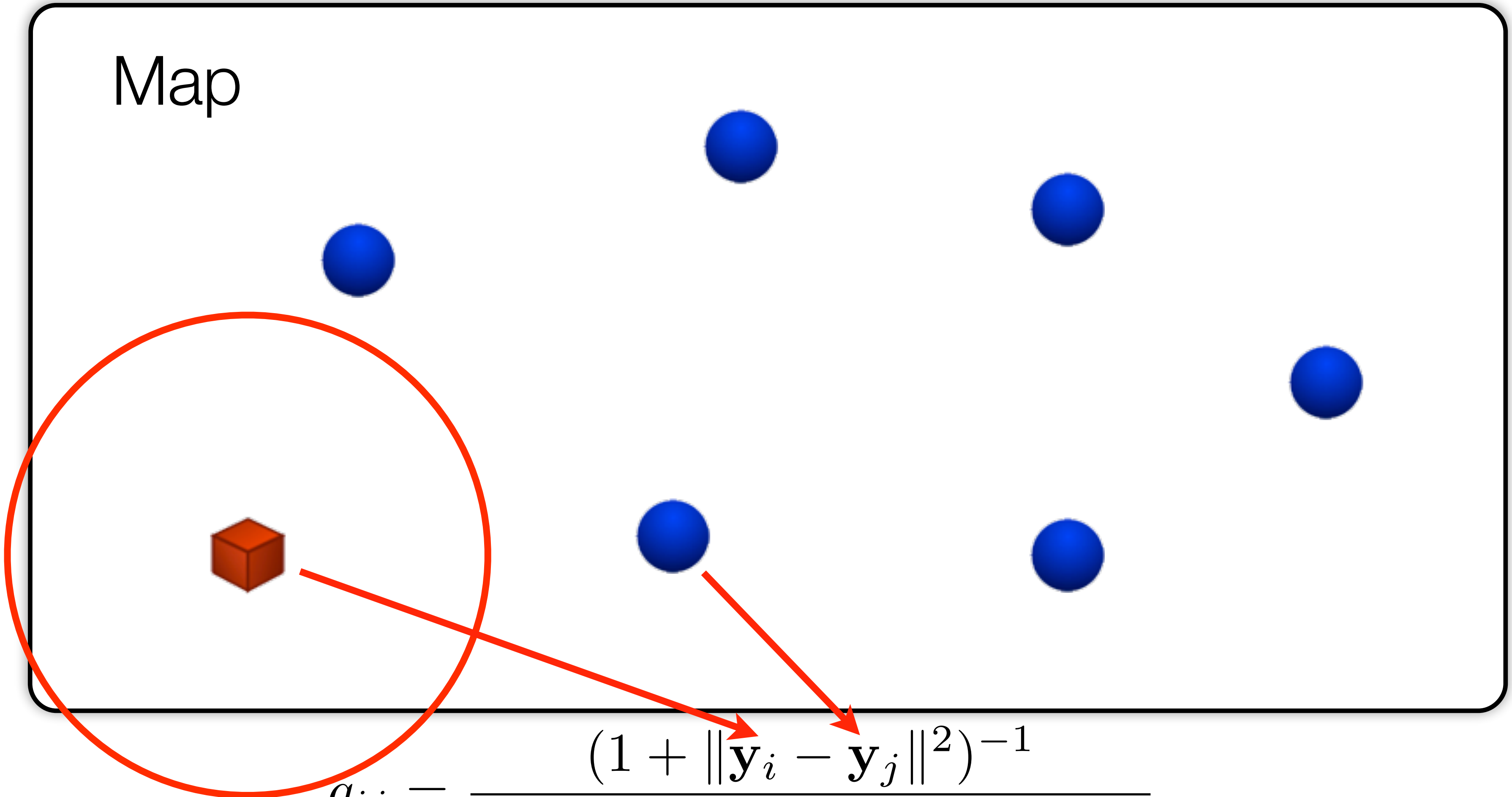
- Measure pairwise similarities between corresponding points in the map:



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

t-Distributed Stochastic Neighbor Embedding

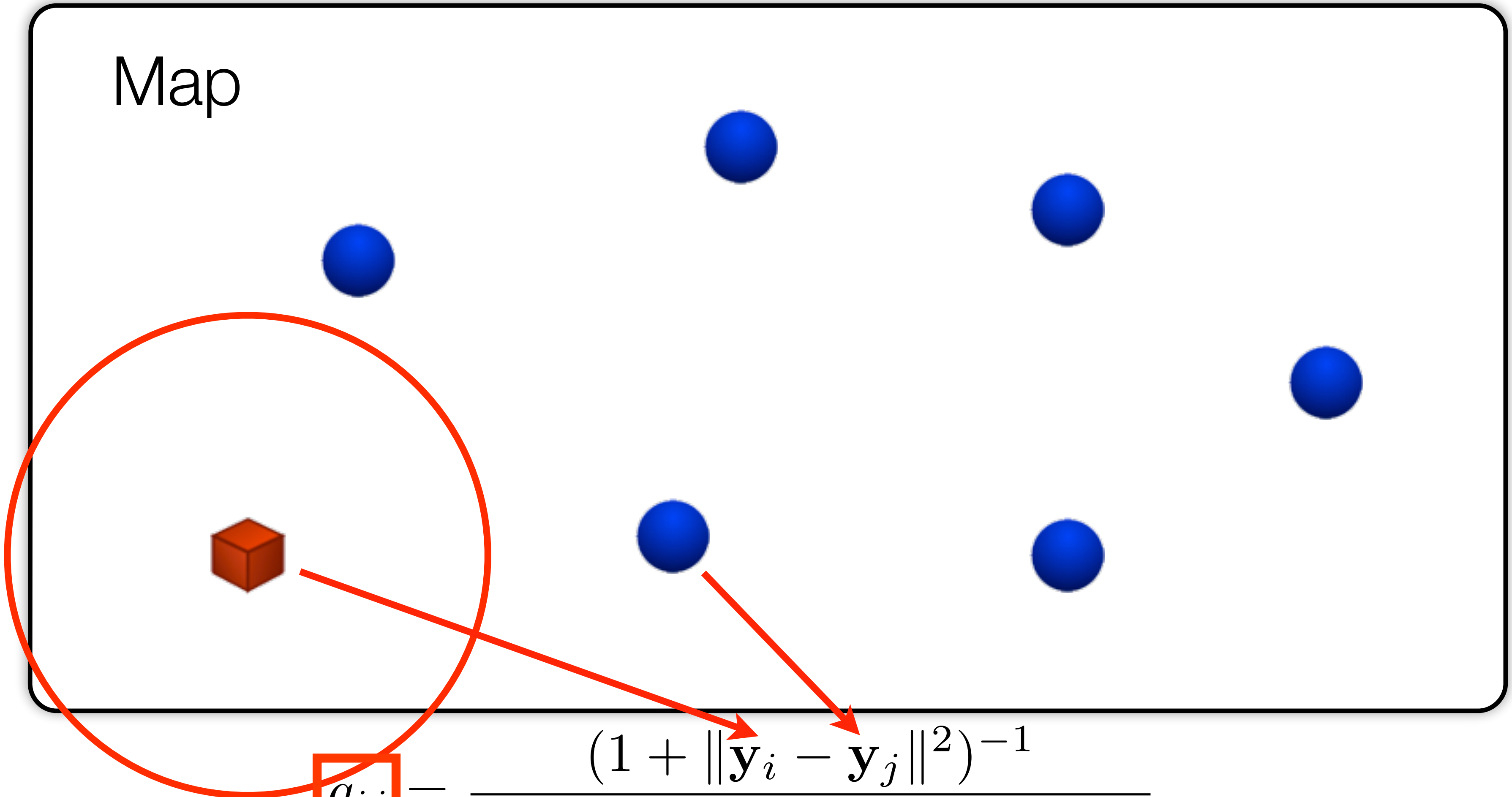
- Measure pairwise similarities between corresponding points in the map:



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

t-Distributed Stochastic Neighbor Embedding

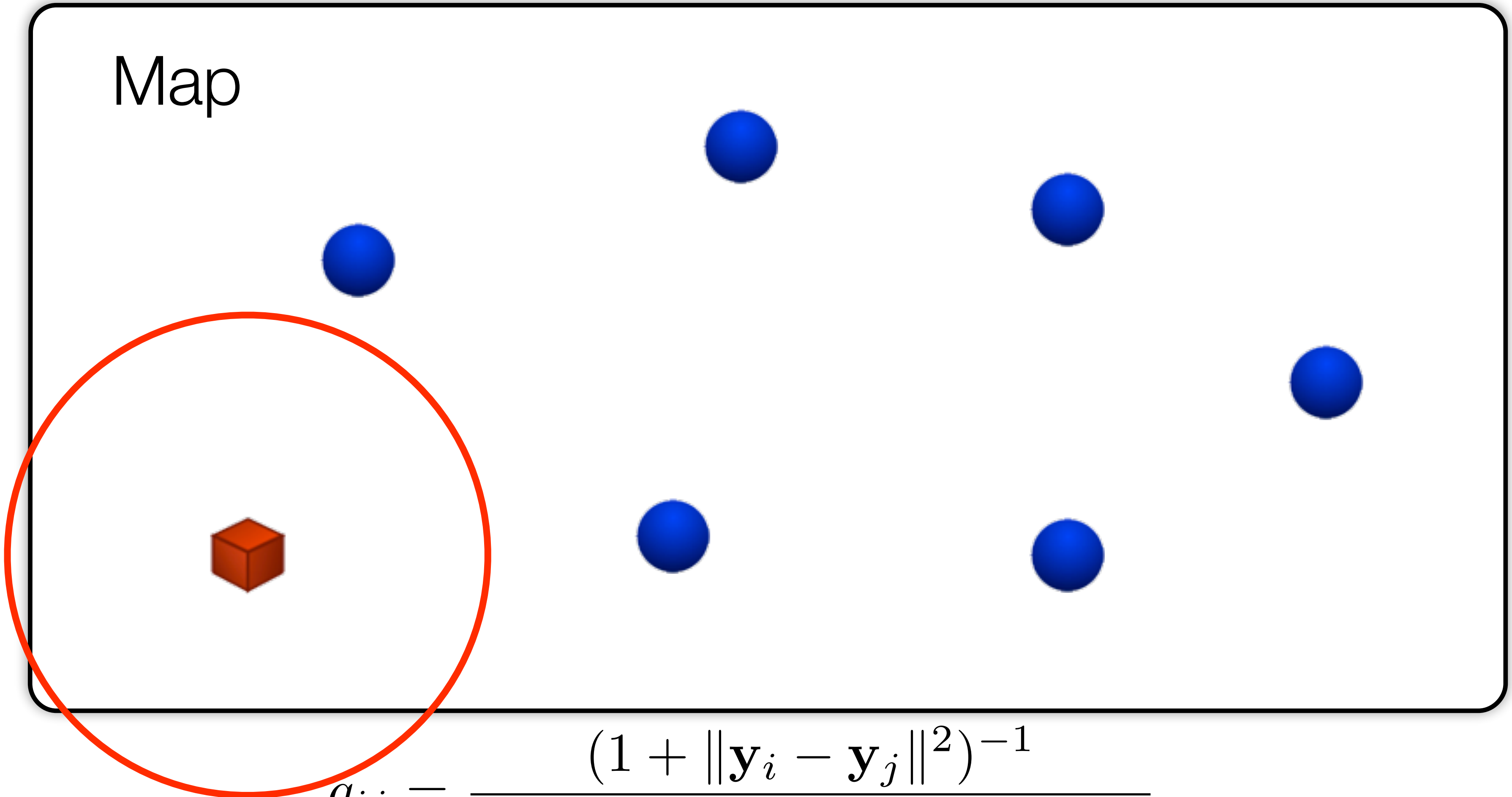
- Measure pairwise similarities between corresponding points in the map:



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

t-Distributed Stochastic Neighbor Embedding

- Move points around to minimize: $KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$



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

t-Distributed Stochastic Neighbor Embedding

- Kullback-Leibler divergence: $KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$

t-Distributed Stochastic Neighbor Embedding

- Kullback-Leibler divergence: $KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$

  - Large p_{ij} modeled by small q_{ij}? Big penalty!

t-Distributed Stochastic Neighbor Embedding

- Kullback-Leibler divergence: $KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$

    - Large p_{ij} modeled by small q_{ij}? Big penalty!

    - Small p_{ij} modeled by large q_{ij}? Small penalty!

t-Distributed Stochastic Neighbor Embedding

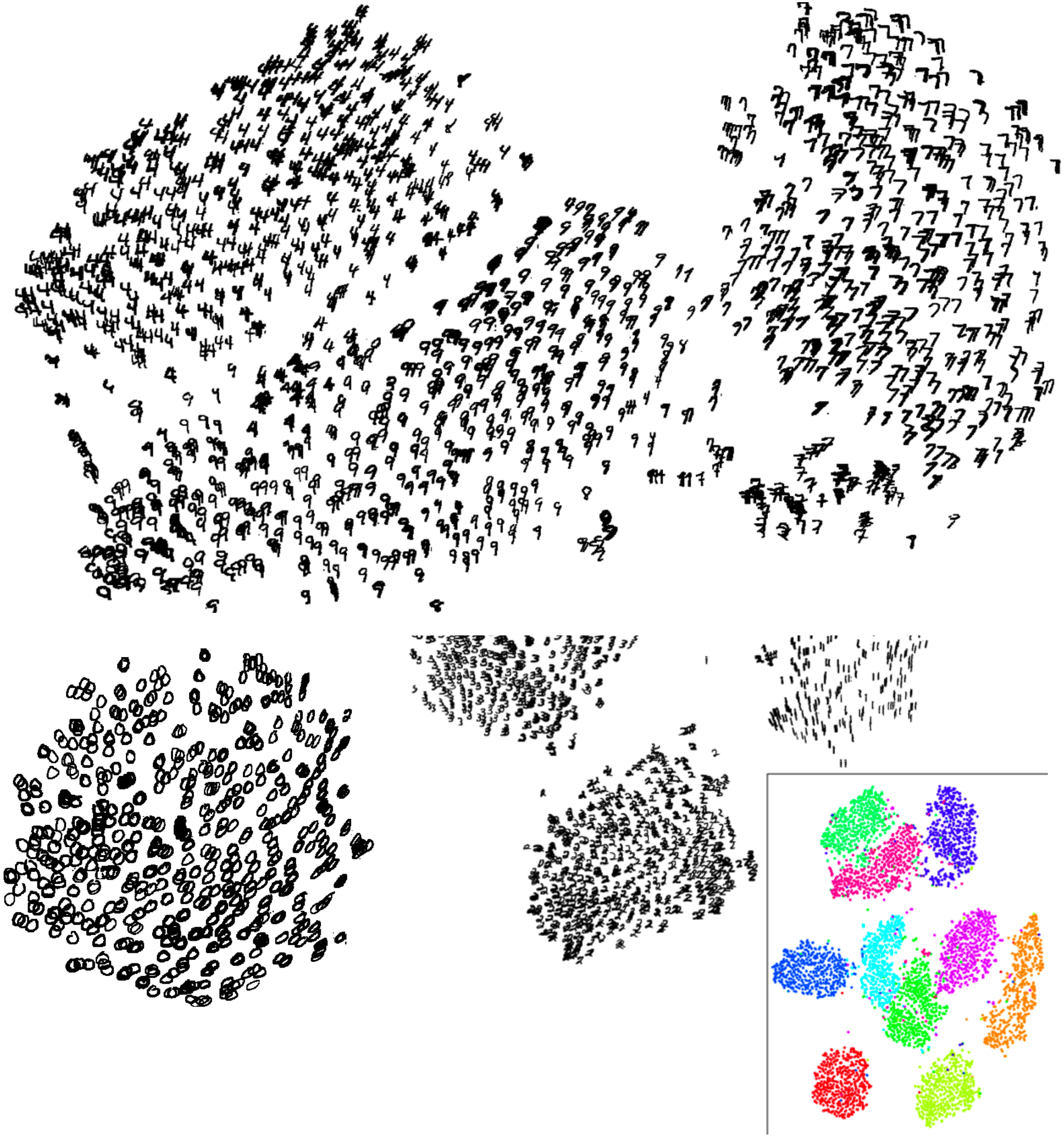
- Kullback-Leibler divergence: $KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$

  - Large p_{ij} modeled by small q_{ij}? Big penalty!

  - Small p_{ij} modeled by large q_{ij}? Small penalty!

- Hence, t-SNE mainly preserves *local similarity structure* of the data


0
1
2
3
4
5
6
7
8
9



Why a Student-t distribution?

- Why do we define map similarities as $q_{ij} \propto (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$?

Why a Student-t distribution?

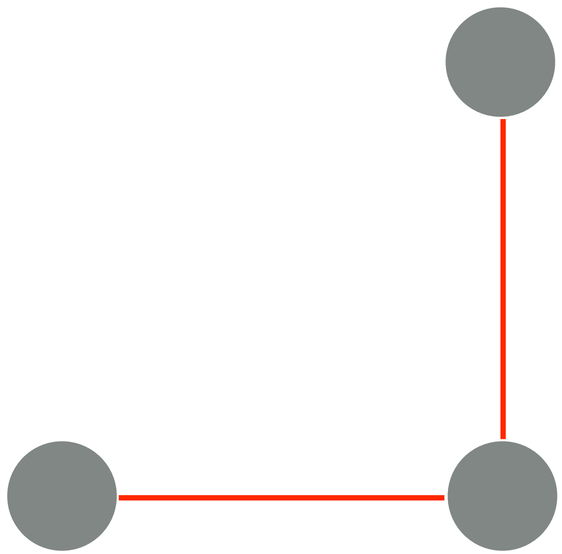
- Why do we define map similarities as $q_{ij} \propto (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$?
- Suppose data is *intrinsically* high-dimensional

Why a Student-t distribution?

- Why do we define map similarities as $q_{ij} \propto (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$?
- Suppose data is *intrinsically* high-dimensional
- We try to model the *local structure* of this data in the map

Why a Student-t distribution?

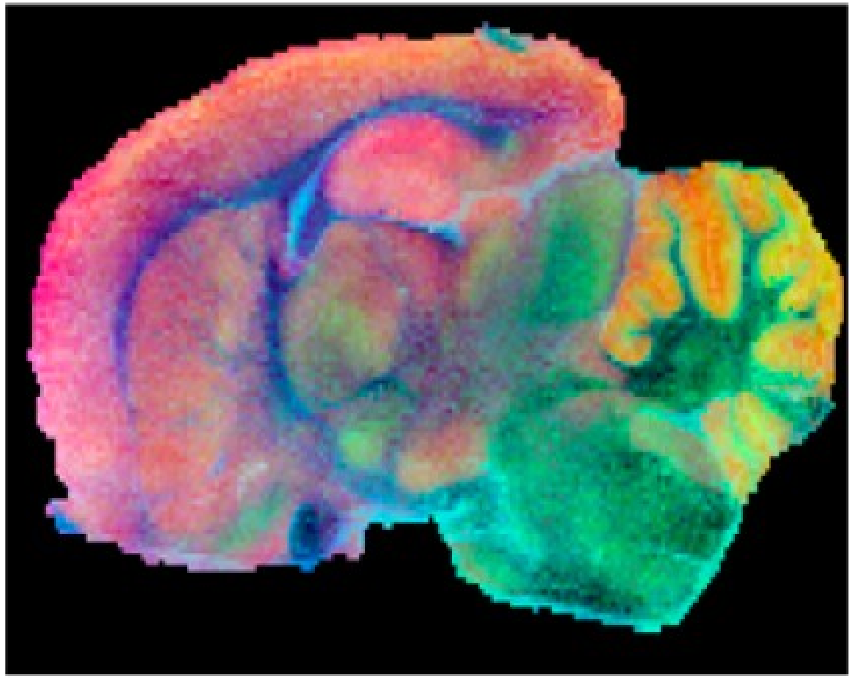
- Why do we define map similarities as $q_{ij} \propto (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$?
- Suppose data is *intrinsically* high-dimensional
- We try to model the *local structure* of this data in the map
- Result: dissimilar points have to be modeled as too far apart in the map!



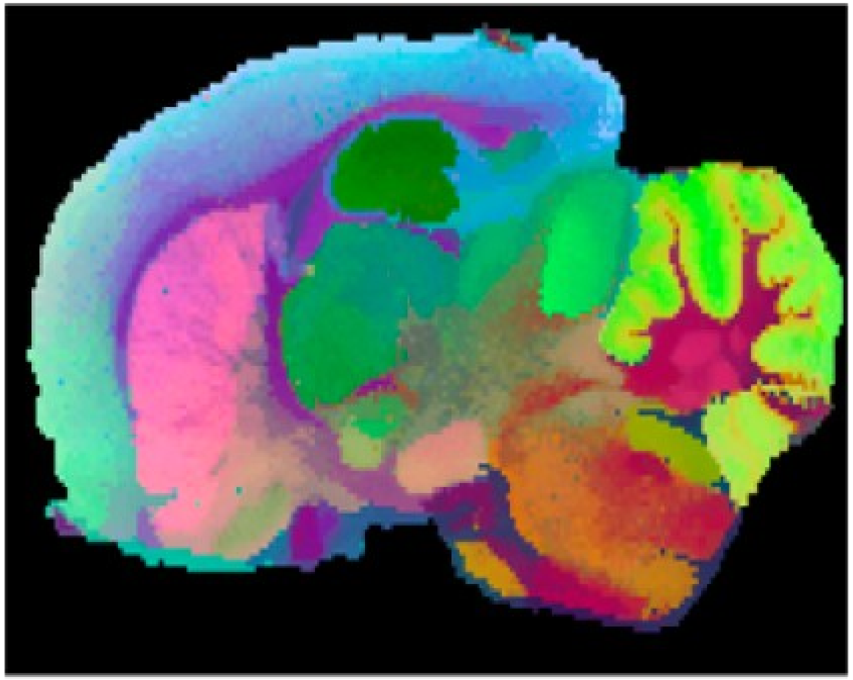
Visualizing mass spectrometry data

A mass spectrum is a plot of the ion signal as a function of the mass-to-charge ratio. It helps identify the amount and type of chemicals present in a sample.

PCA

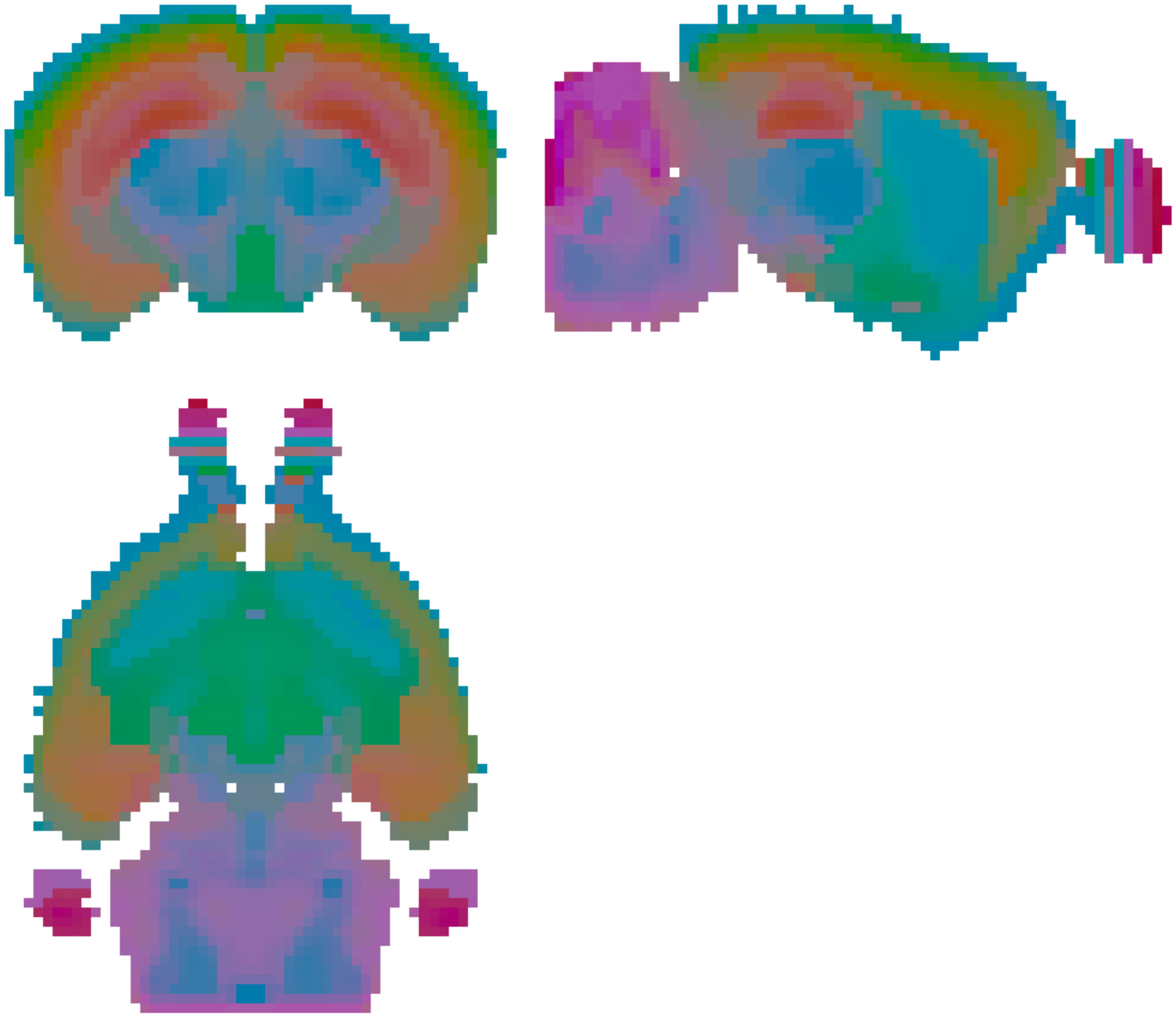


t-SNE



* Figure adopted from: J. Fonville *et al.*, *Anal. Chem.* 85(3), 2013.

Visualizing gene expression data



* Joint work with the LUMC and the Allen Brain Institute.

Human embryo data

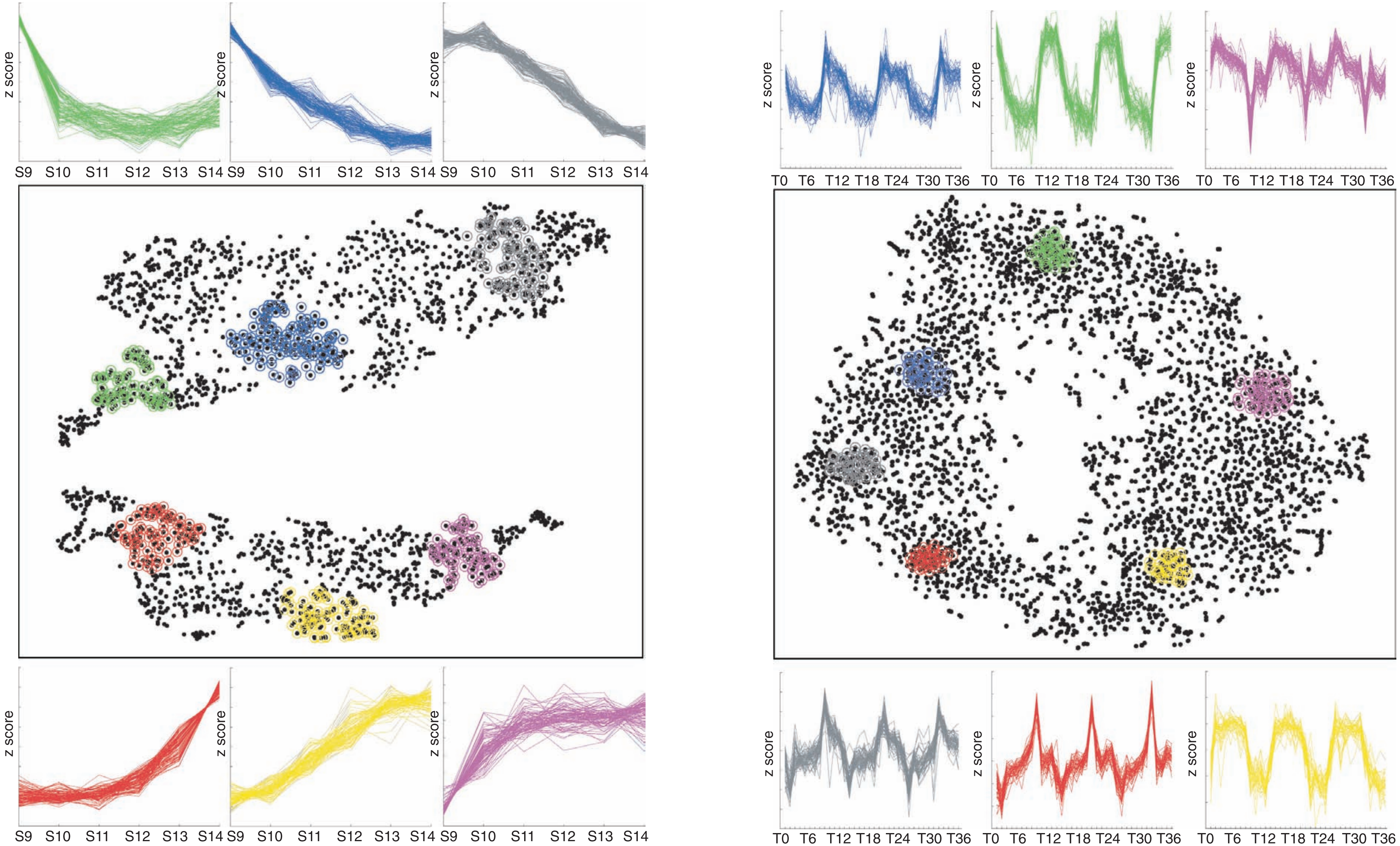
Two groups: one set of genes gets up-regulated and one set of genes gets down-regulated. (6D)

g gene expression data

Yeast metabolic cycle data

Cyclic behavior causes ring-like structure. (36D)

- Visualizing human embryo data and yeast metabolic cycle data:

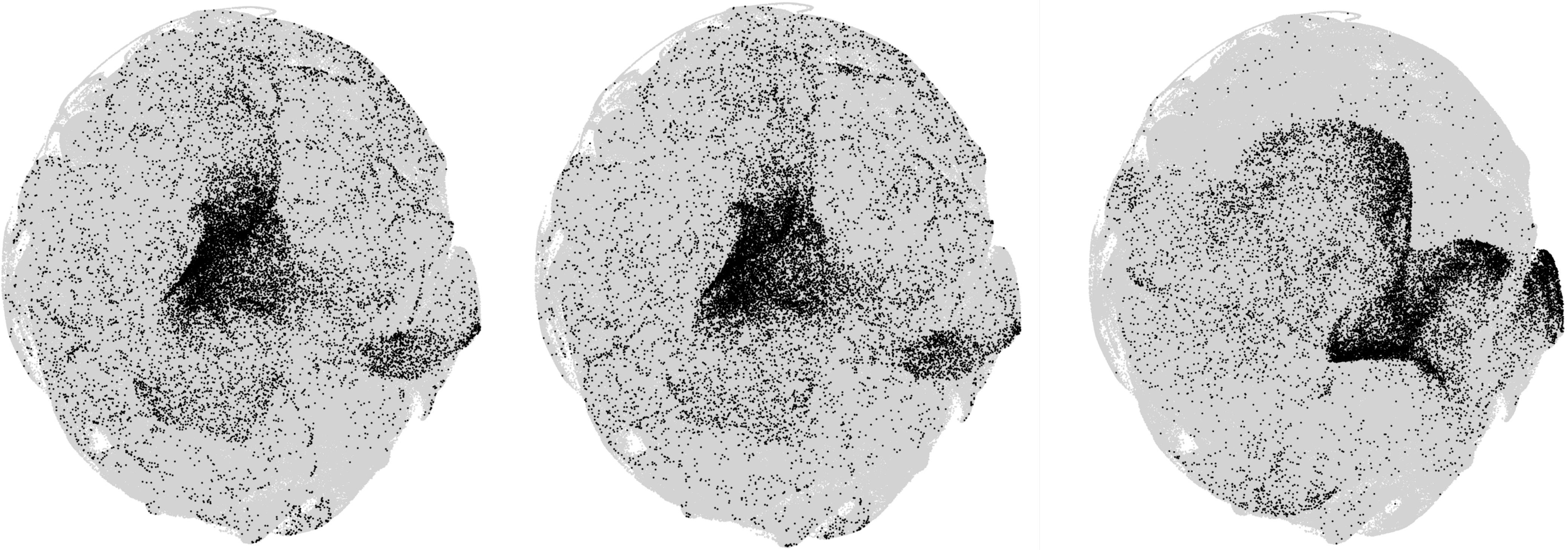


* Figure adopted from: N. Bushati *et al.*, *Nucleic Acids Res.* 39(17), 2011.

Visualizing flowcytometry data

- Flowcytometry data (= blood cell measurements) of leukemic and healthy kids:

Healthy | Healthy | Leukemia

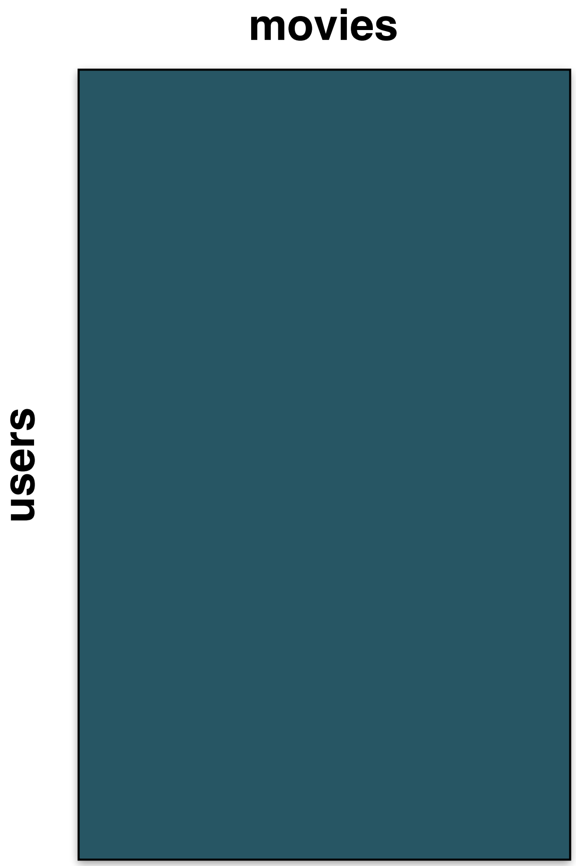


* Joint work with SickKids Hospital Toronto.


COFFEE!

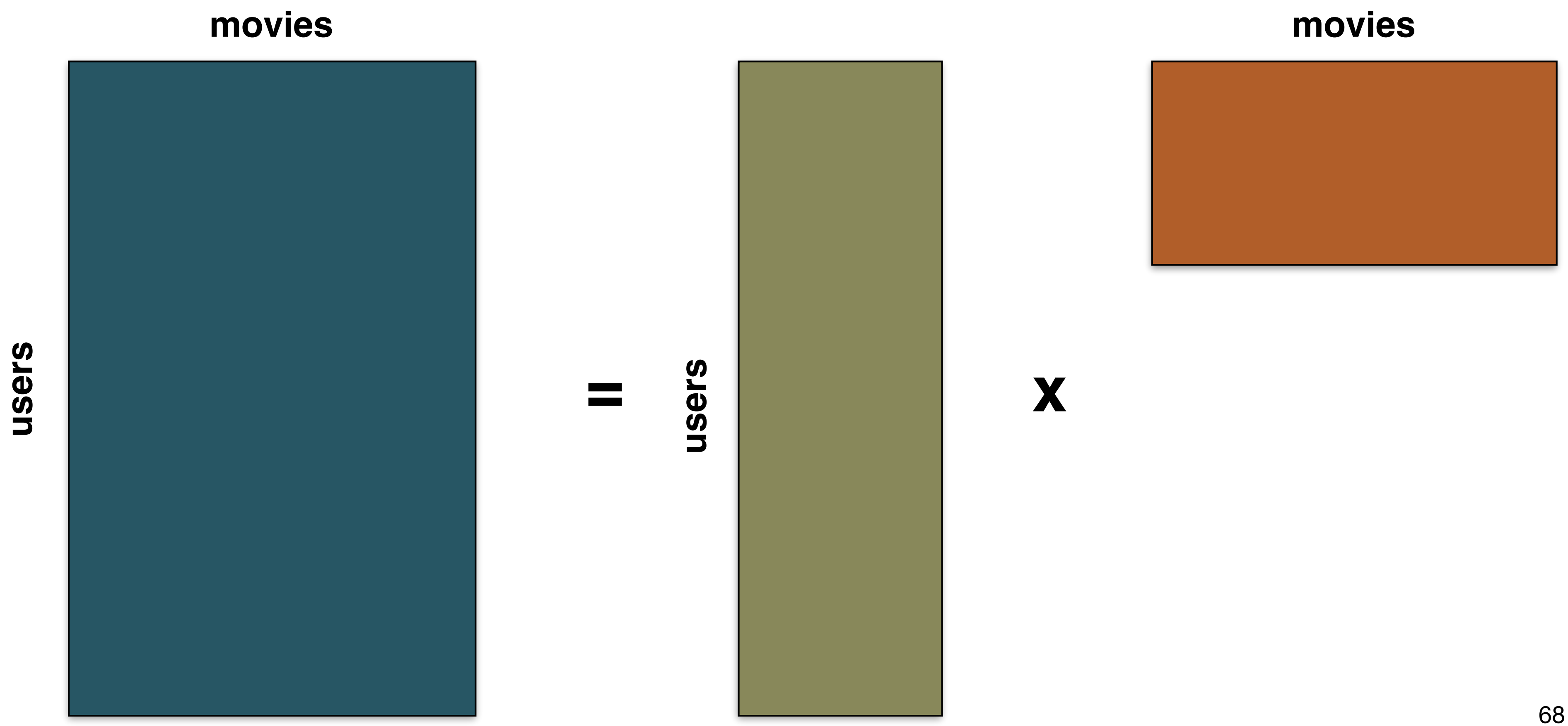
Visualizing movies

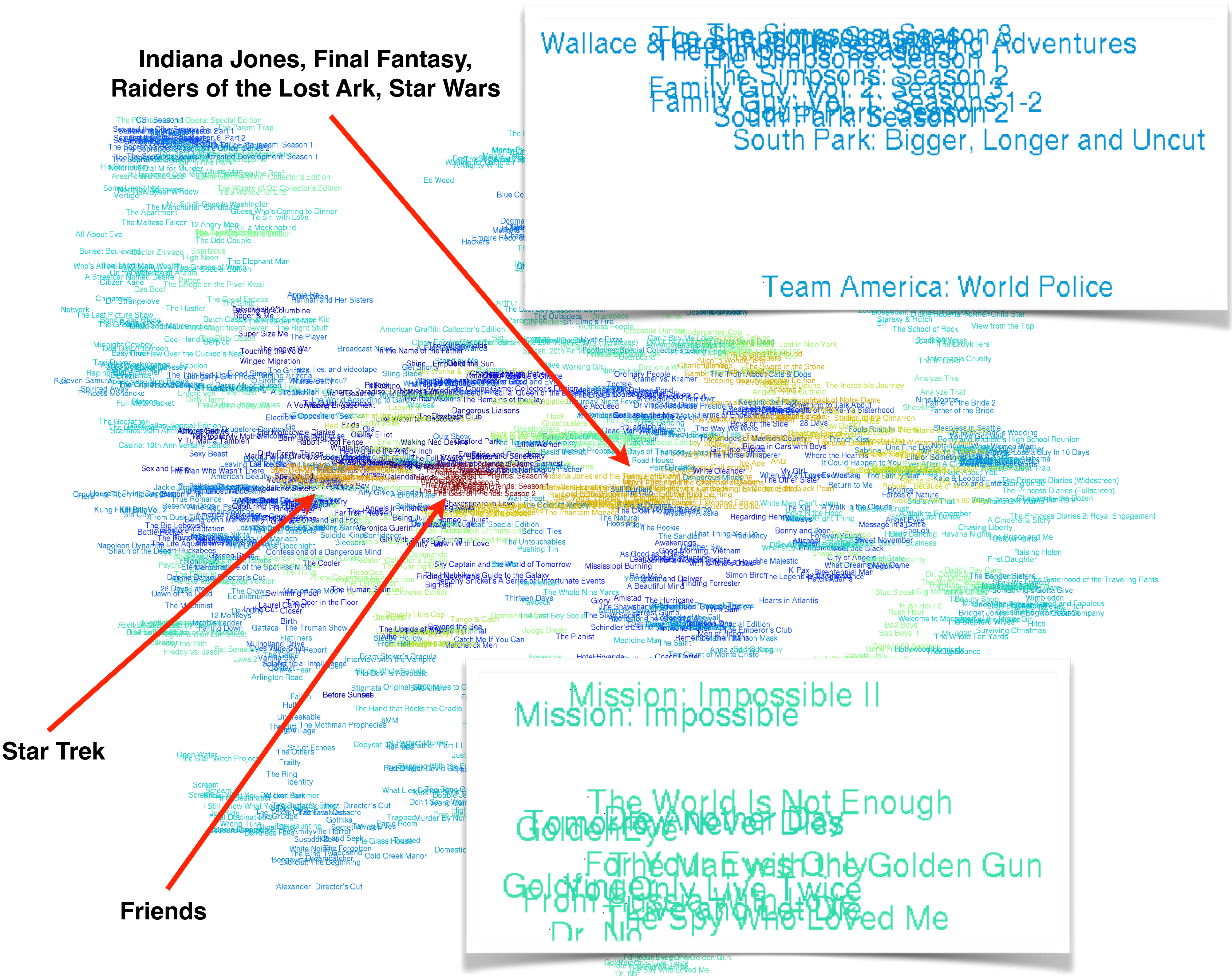
- Netflix has a large collection of user-movie ratings stored in a *rating matrix*



Visualizing movies

- Netflix has a large collection of user-movie ratings stored in a *rating matrix*
- *Decompose* the rating matrix to obtain *user features* and *movie features*:




Indiana Jones, Final Fantasy, Raiders of the Lost Ark, Star Wars
Star Trek
Friends
South Park: Bigger, Longer and Uncut
Team America: World Police
Mission: Impossible II
Mission: Impossible
The World Is Not Enough
The Spy Who Loved Me

What about Big Data?

- All the approaches I presented so far scale quadratically (or worse):
  - How do you make maps of data with lots of instances / records?

- Trick 1: Construct sparse matrix (approximate) input similarities

- Trick 2: Approximate interactions between points in map during learning

Computing high-dimensional similarities efficiently

Finding nearest neighbors

- What is the most common value in the input similarity matrix *P*?

Finding nearest neighbors

- What is the most common value in the input similarity matrix *P*? Infinitesimal!

Finding nearest neighbors

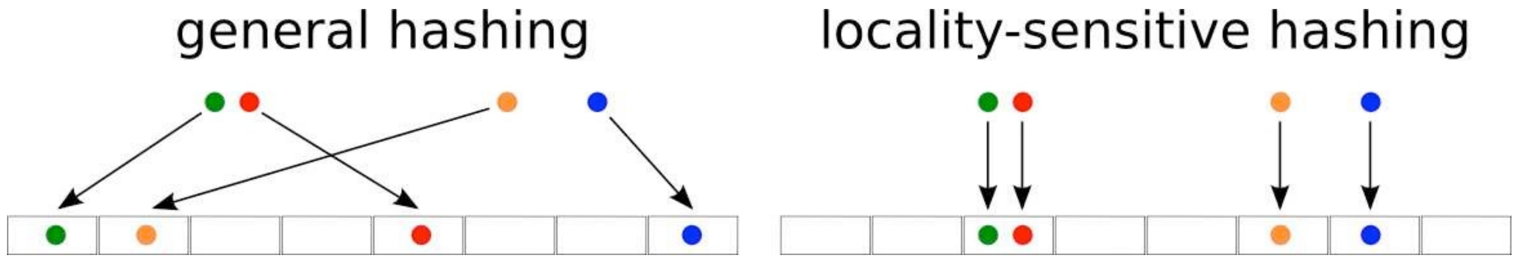
- What is the most common value in the input similarity matrix *P*? Infinitesimal!

- Good approximation: only compute p_{ij} for pairs of near neighbors

- Finding near neighbors (approximately) can be performed very efficiently:
    - Using a trick called *locality-sensitive hashing*
    - Using clever data structures such as *vantage-point trees*

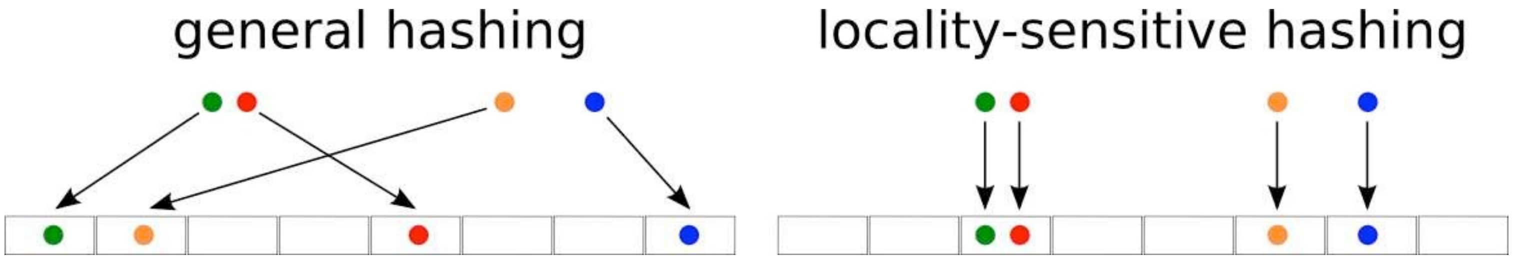
Locality-sensitive hashing

- LSH uses hashing functions that take “location” of object in consideration:

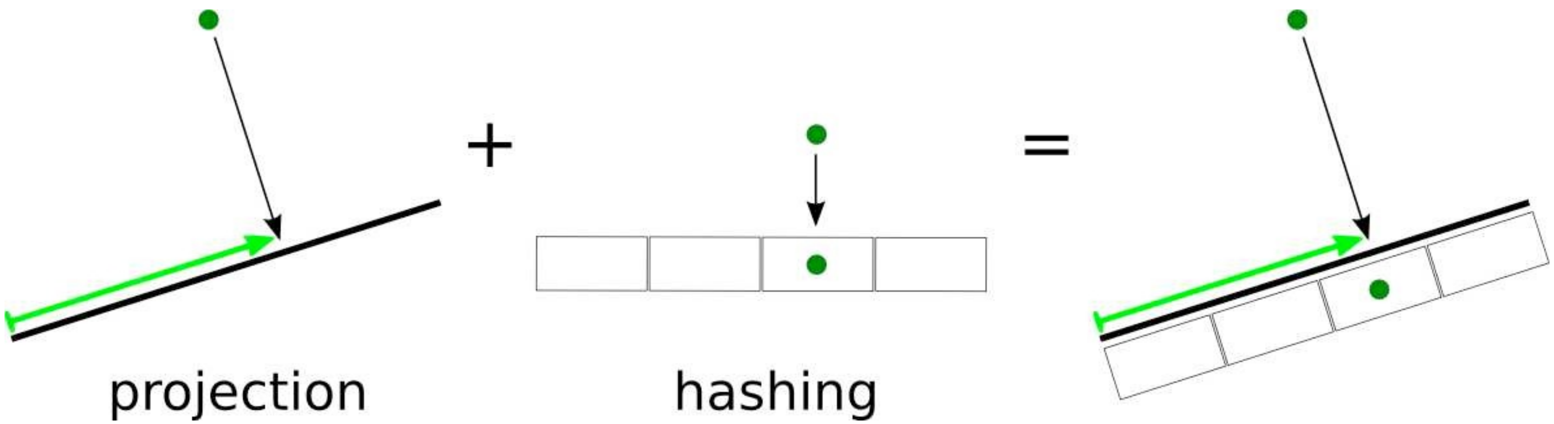


Locality-sensitive hashing

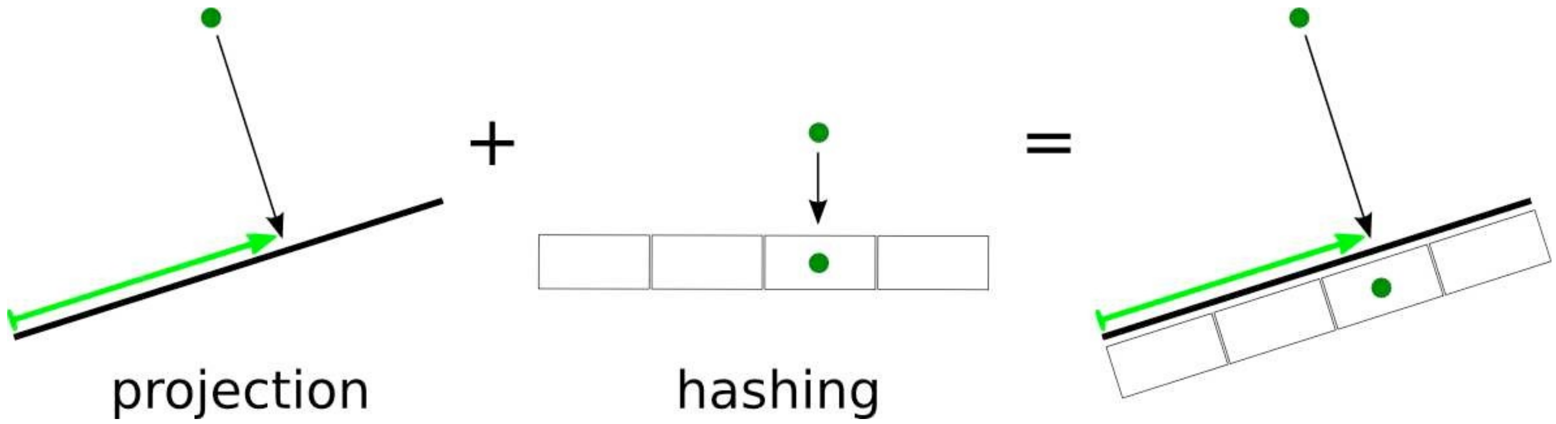
- LSH uses hashing functions that take “location” of object in consideration:



- Example of a locality-sensitive hashing function for points in a space:
  - Project the point onto a random subspace; divide result into 4 buckets (2 bits)



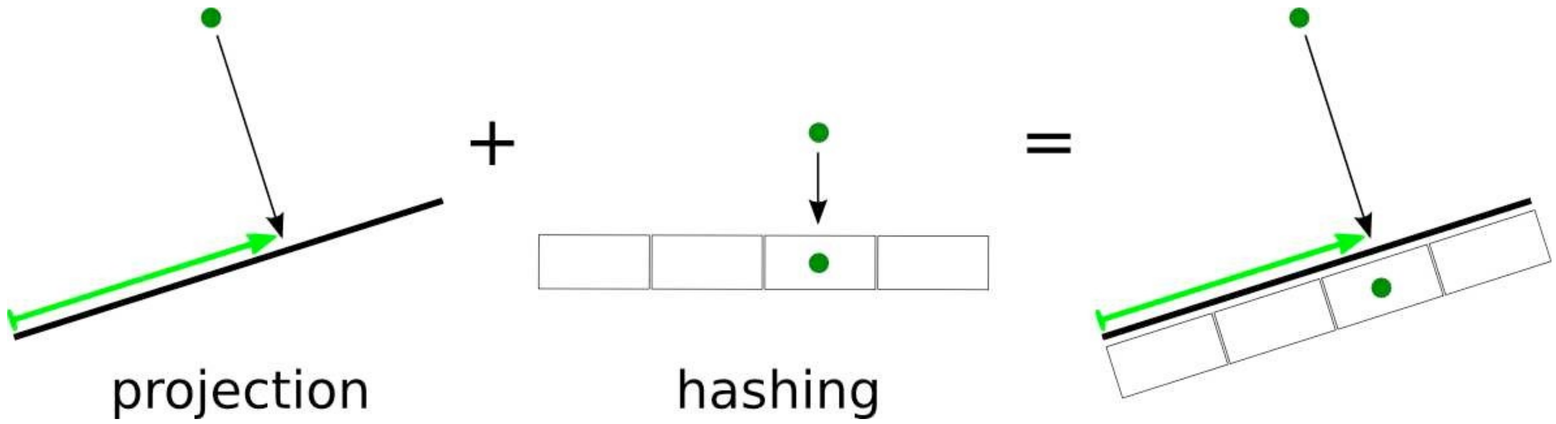
Locality-sensitive hashing



- Mathematically, we could express this locality sensitive hash function as:

$$h(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad \mathbf{w}^\top \mathbf{x} \leq -\tau \\ 1 & \text{if} \quad -\tau < \mathbf{w}^\top \mathbf{x} \leq 0 \\ 2 & \text{if} \quad 0 < \mathbf{w}^\top \mathbf{x} \leq \tau \\ 3 & \text{if} \quad \mathbf{w}^\top \mathbf{x} > \tau \end{cases}$$

Locality-sensitive hashing

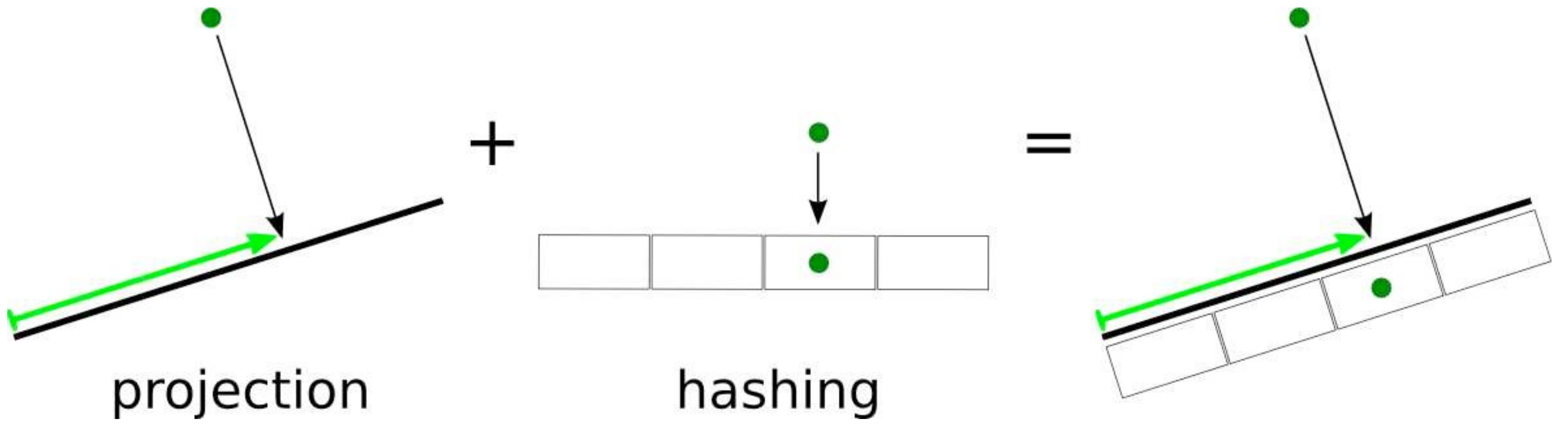


- Mathematically, we could express this locality sensitive hash function as:

$$h(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad \mathbf{w}^\top \mathbf{x} \leq -\tau \\ 1 & \text{if} \quad -\tau < \mathbf{w}^\top \mathbf{x} \leq 0 \\ 2 & \text{if} \quad 0 < \mathbf{w}^\top \mathbf{x} \leq \tau \\ 3 & \text{if} \quad \mathbf{w}^\top \mathbf{x} > \tau \end{cases}$$

random projection

Locality-sensitive hashing



- Mathematically, we could express this locality sensitive hash function as:

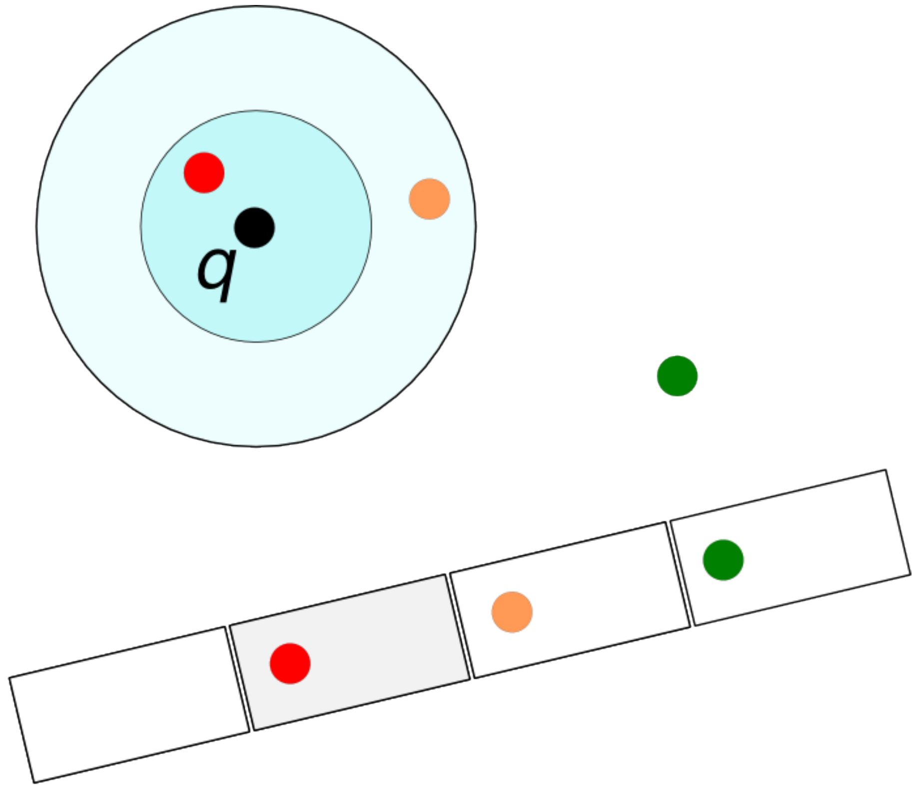
$$h(\mathbf{x}) = \begin{cases} 0 & \text{if} \quad \mathbf{w}^\top \mathbf{x} \leq -\tau \\ 1 & \text{if} \quad -\tau < \mathbf{w}^\top \mathbf{x} \leq 0 \\ 2 & \text{if} \quad 0 < \mathbf{w}^\top \mathbf{x} \leq \tau \\ 3 & \text{if} \quad \mathbf{w}^\top \mathbf{x} > \tau \end{cases}$$

random projection

threshold parameter

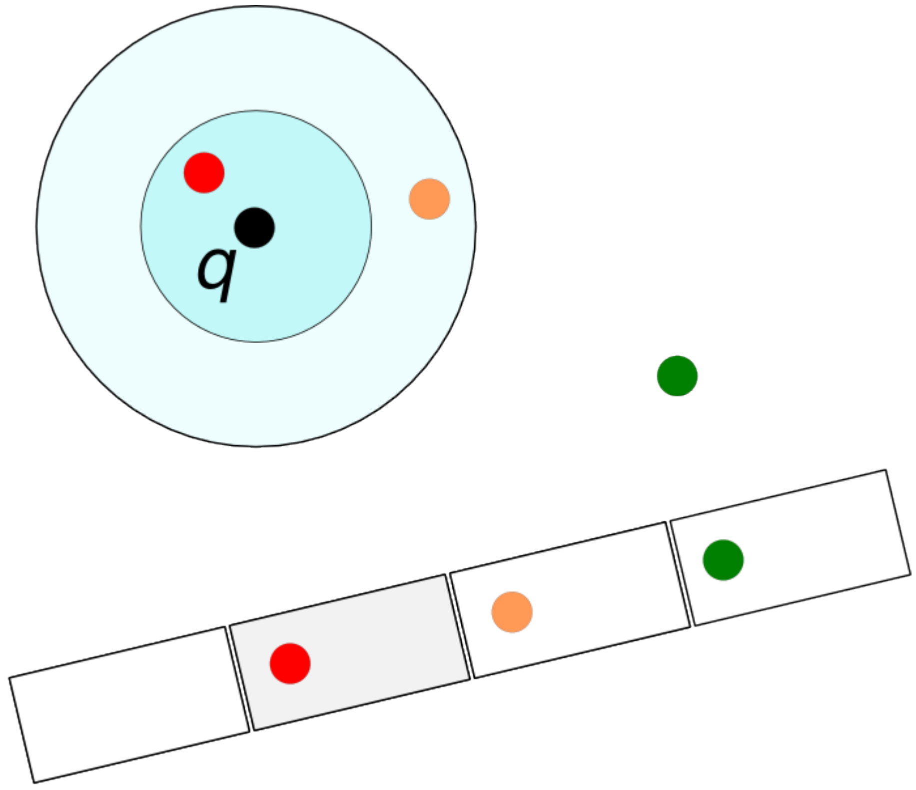
Locality-sensitive hashing

- Retrieval of nearest neighbors of a query point *q* using LSH works as follows:



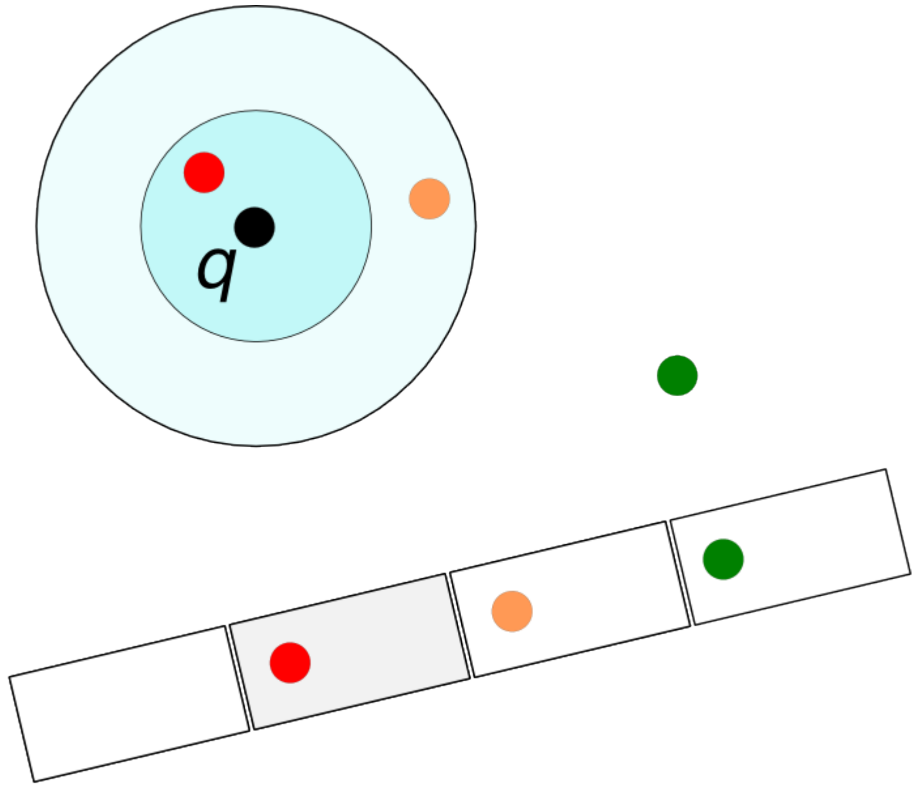
Locality-sensitive hashing

- Retrieval of nearest neighbors of a query point *q* using LSH works as follows:
  - Hash all data points using locality-sensitive hash



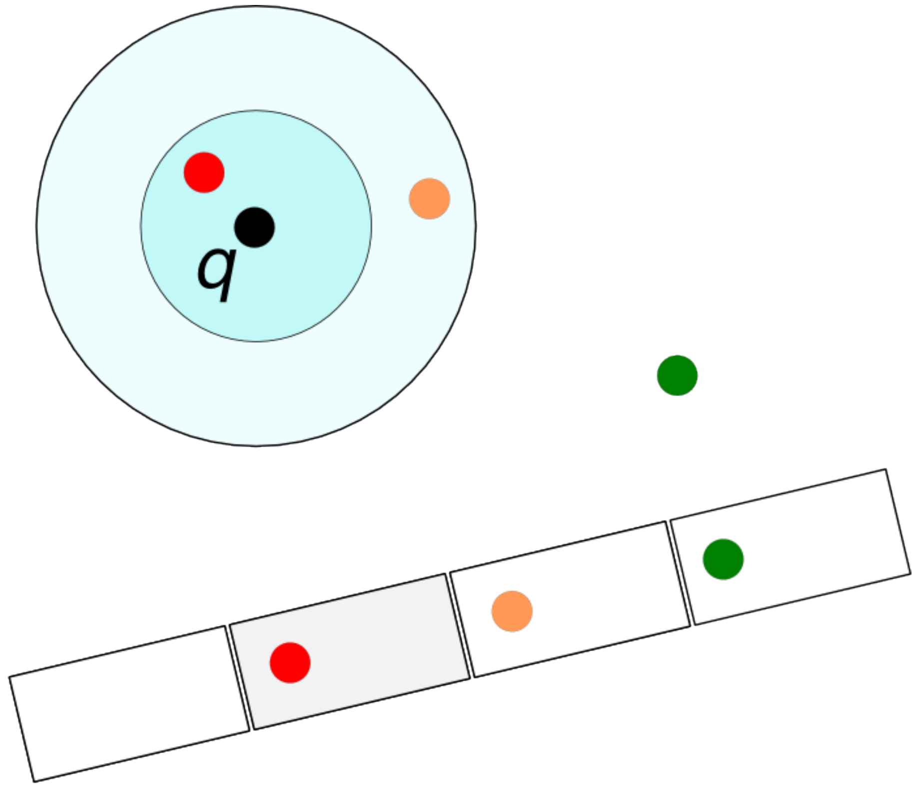
Locality-sensitive hashing

- Retrieval of nearest neighbors of a query point *q* using LSH works as follows:
    - Hash all data points using locality-sensitive hash
    - Compute locality-sensitive hash of query point



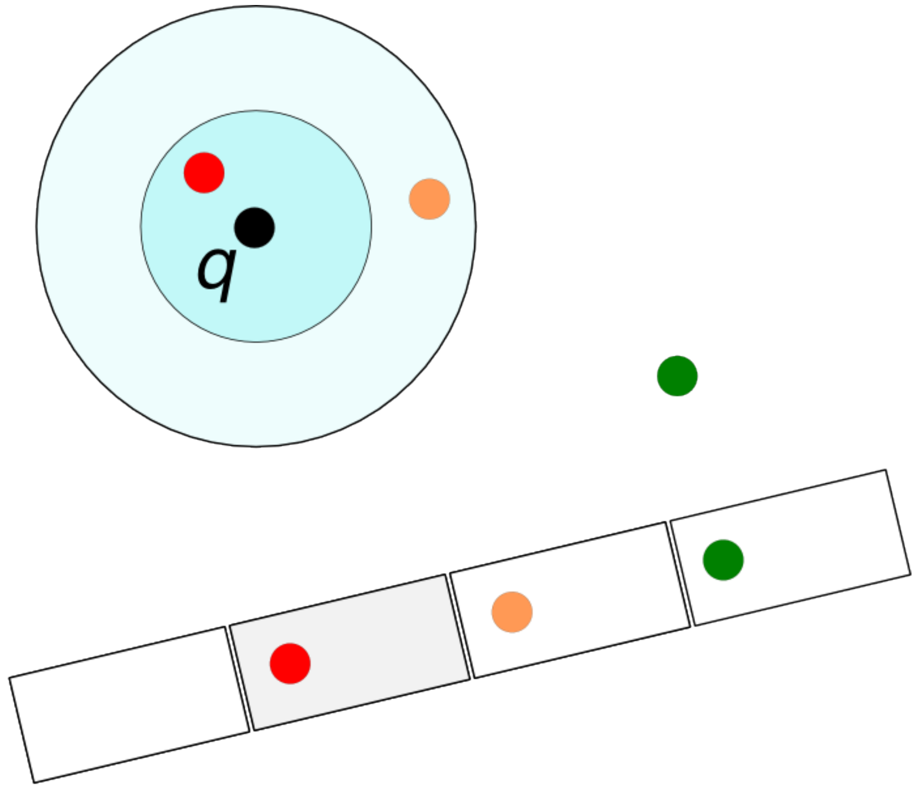
Locality-sensitive hashing

- Retrieval of nearest neighbors of a query point *q* using LSH works as follows:
  - Hash all data points using locality-sensitive hash
  - Compute locality-sensitive hash of query point
  - All data points in the bucket are candidate near neighbors



Locality-sensitive hashing

- Retrieval of nearest neighbors of a query point *q* using LSH works as follows:
  - Hash all data points using locality-sensitive hash
  - Compute locality-sensitive hash of query point
  - All data points in the bucket are candidate near neighbors
  - Compute distances to candidate points to find true nearest neighbors

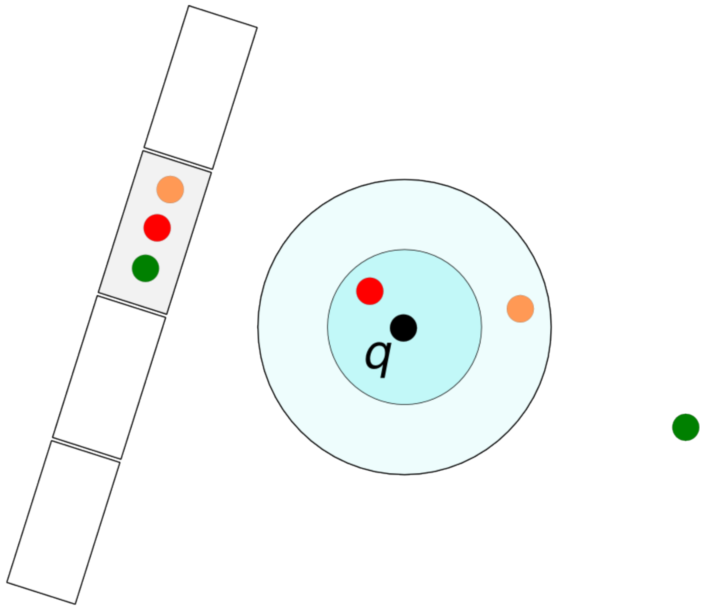


Locality-sensitive hashing

- LSH projections may be “unlucky” in two main ways:

Locality-sensitive hashing

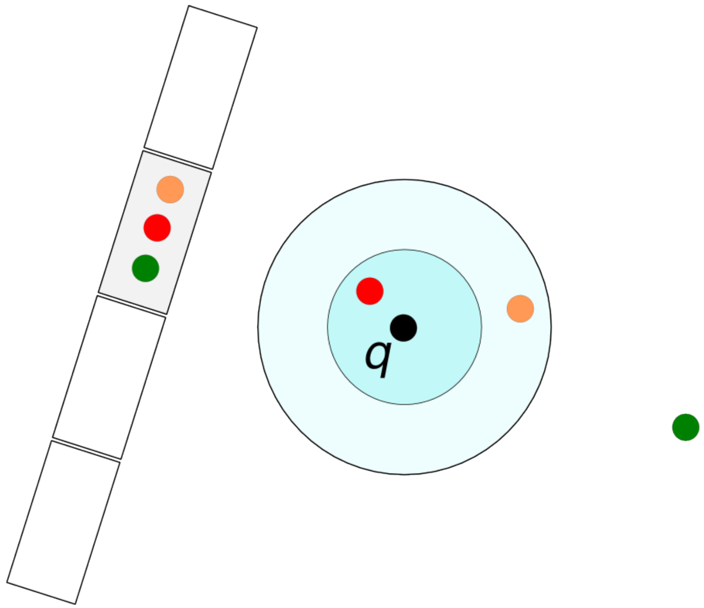
- LSH projections may be “unlucky” in two main ways:



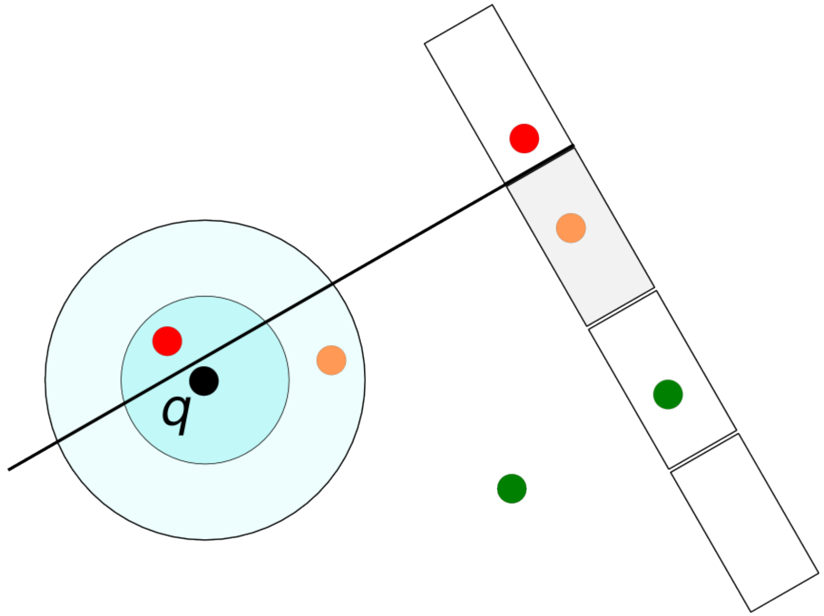
“Collision”: Distant points
hashed in the same bucket

Locality-sensitive hashing

- LSH projections may be “unlucky” in two main ways:



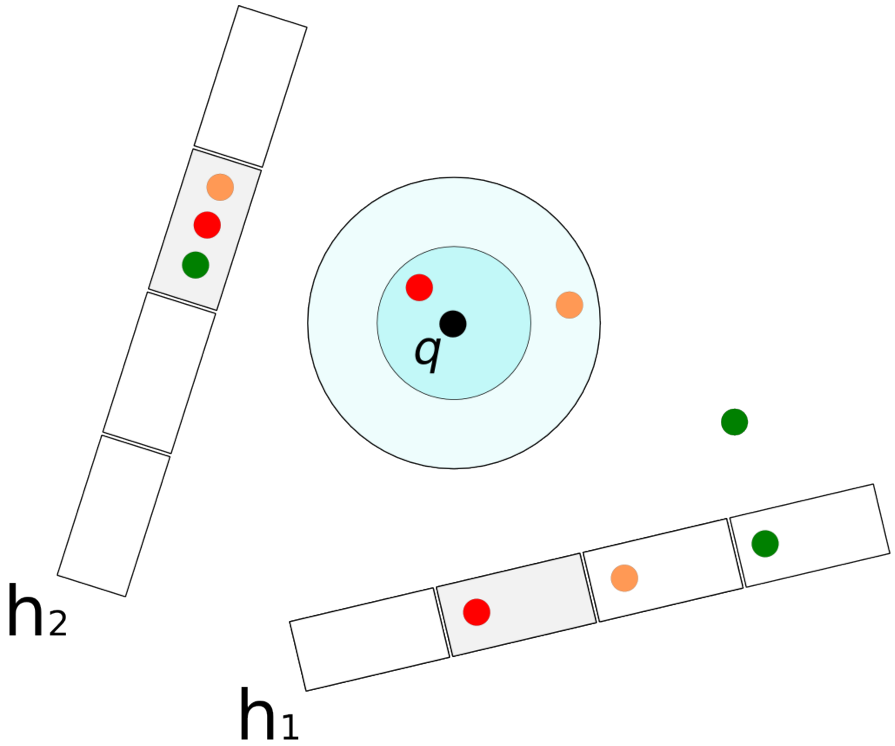
“Collision”: Distant points
hashed in the same bucket



“Split”: Nearby points
hashed in different buckets

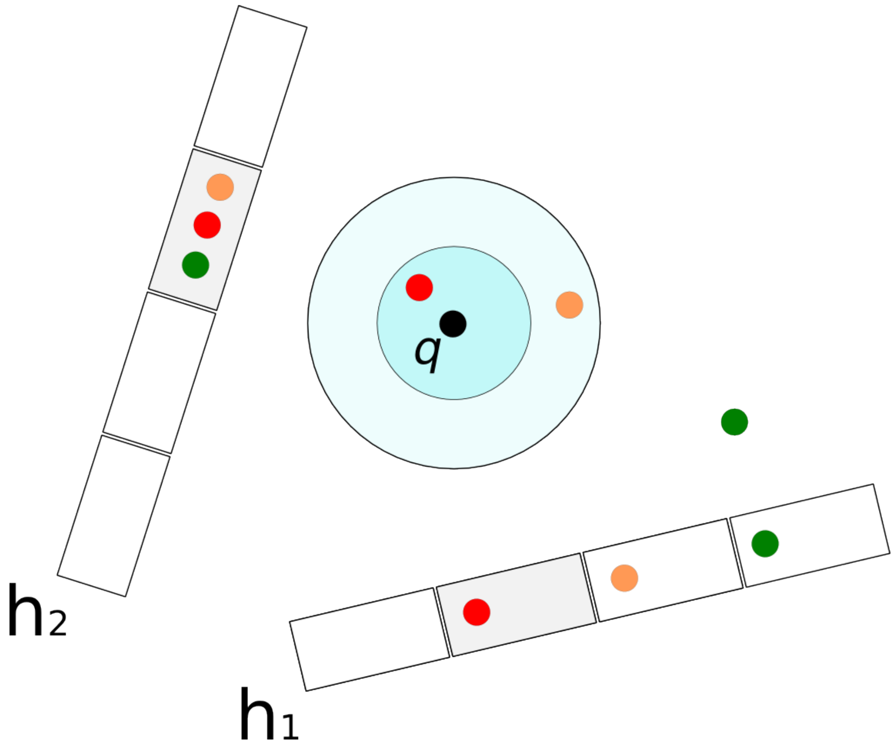
Locality-sensitive hashing

- Using multiple projections in an LSH resolves “collisions”:



Locality-sensitive hashing

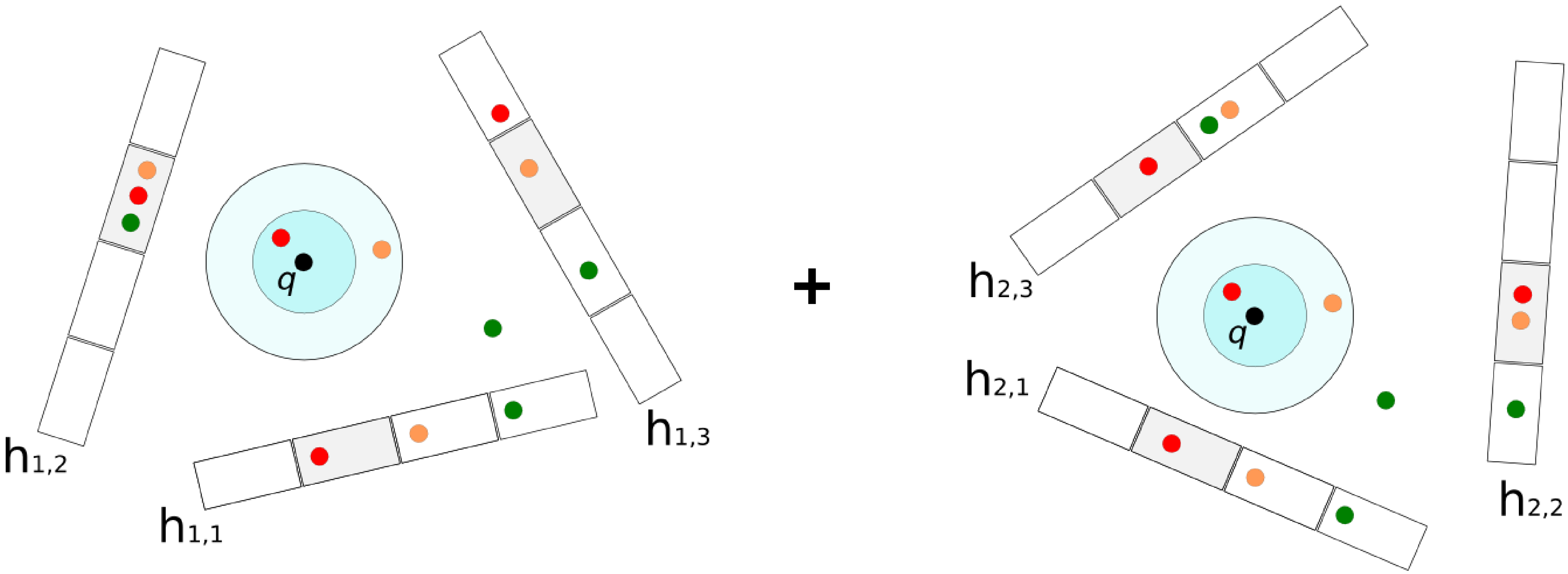
- Using multiple projections in an LSH resolves “collisions”:



- The LSH is given by a concatenation of all individual buckets

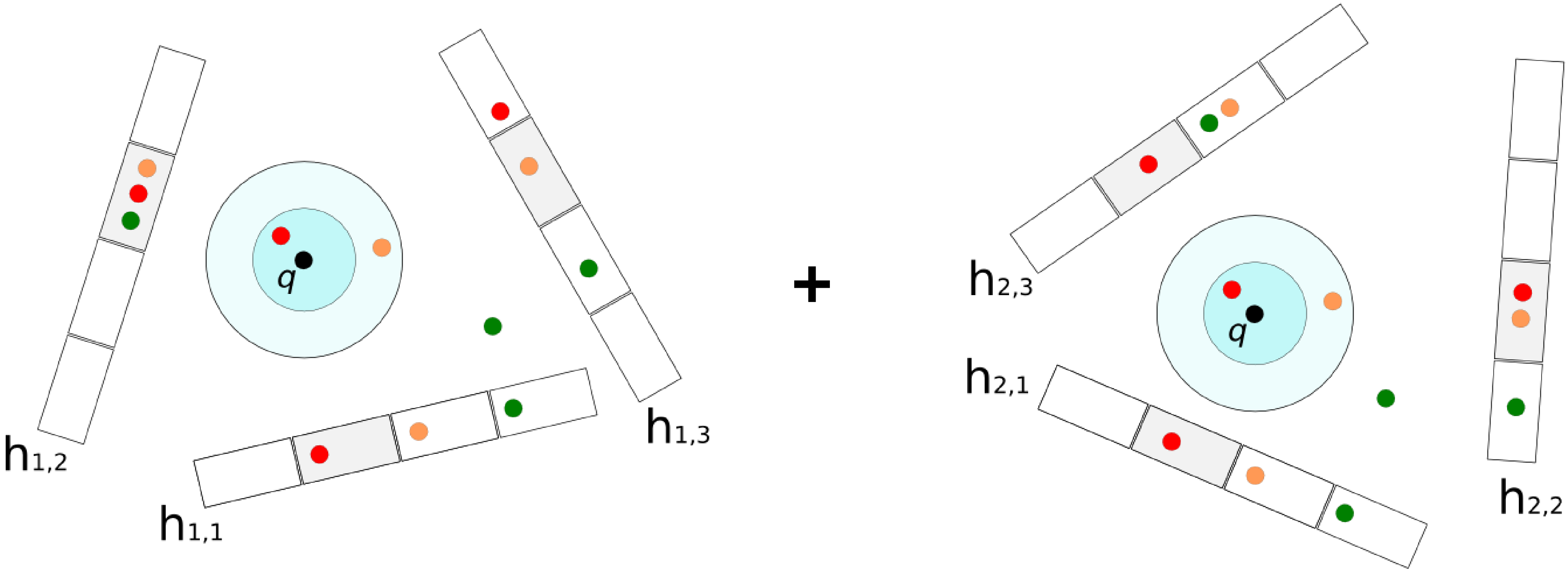
Locality-sensitive hashing

- Using multiple separate hash tables when doing LSH resolves “splits”:



Locality-sensitive hashing

- Using multiple separate hash tables when doing LSH resolves “splits”:



- Points are candidate neighbors if candidate in *any* of the hash tables

Locality-sensitive hashing

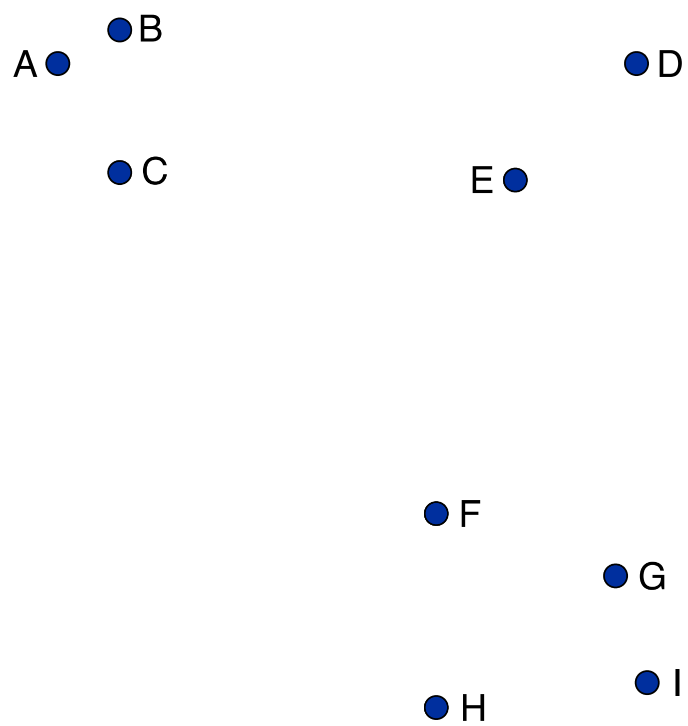
- Efficient algorithm to compute a sparse *P*-matrix:
    - Load all data in a locality-sensitive hash
    - For each data point, retrieve the *candidate near neighbors* from the LSH
    - For these candidates, compute the *P*-value using a Gaussian kernel

Constructing maps efficiently

Gradient interpretation

- We can interpret building a t-SNE map as a simulation of an *N-body system*:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}(\mathbf{y}_i - \mathbf{y}_j)$$

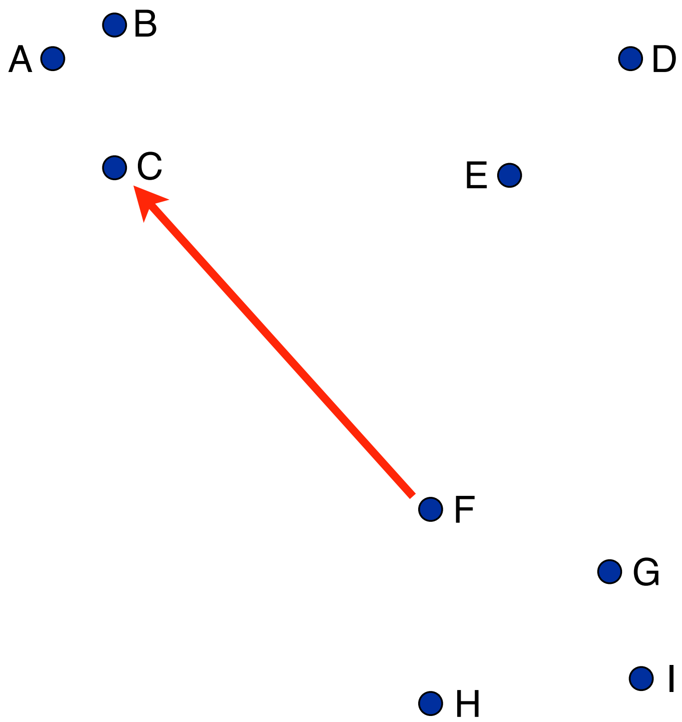


Gradient interpretation

- We can interpret building a t-SNE map as a simulation of an *N-body system*:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1} \boxed{(\mathbf{y}_i - \mathbf{y}_j)}$$

spring

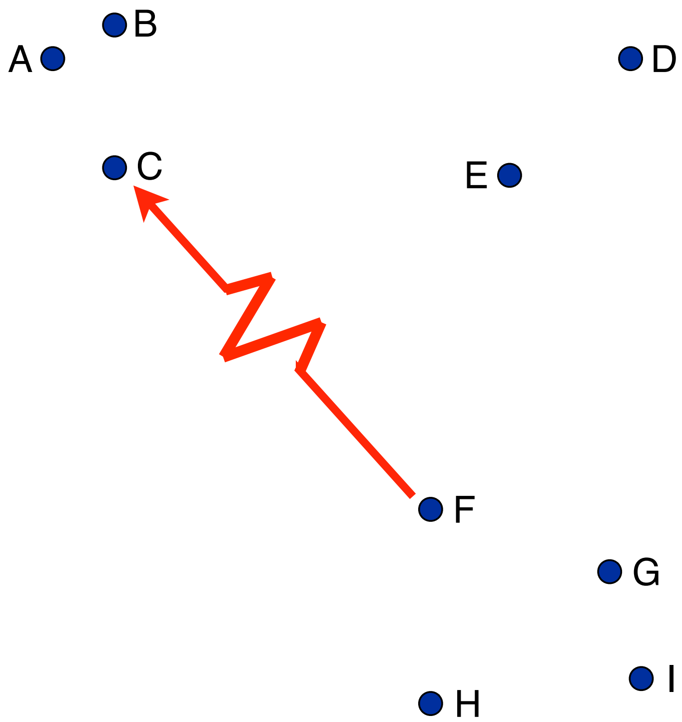


Gradient interpretation

- We can interpret building a t-SNE map as a simulation of an *N-body system*:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1} (\mathbf{y}_i - \mathbf{y}_j)$$

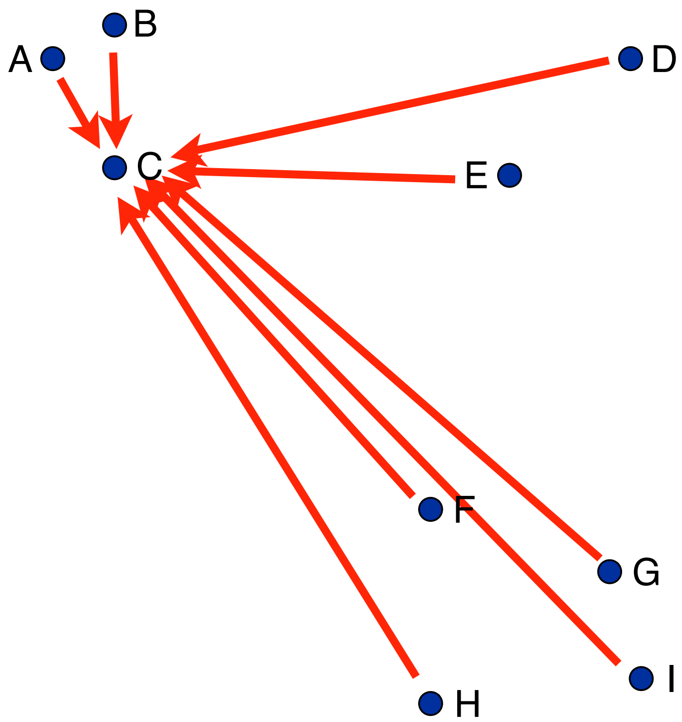
exertion / compression



Gradient interpretation

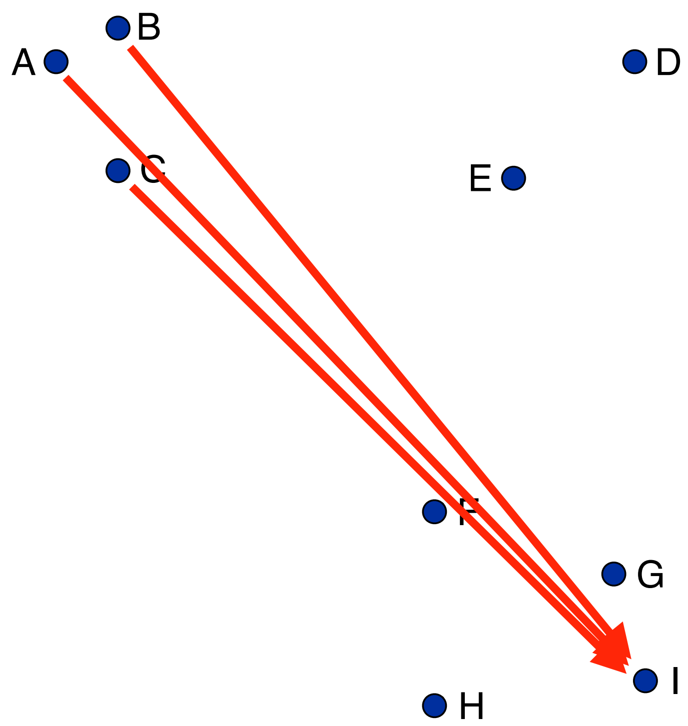
- We can interpret building a t-SNE map as a simulation of an *N-body system*:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1} (\mathbf{y}_i - \mathbf{y}_j)$$



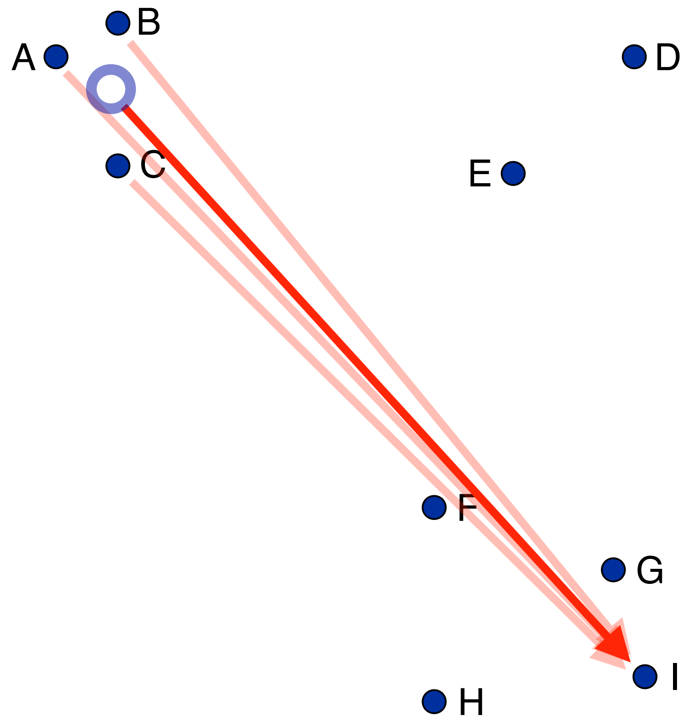
Barnes-Hut approximation

- Many of the pairwise interactions between points are very similar:



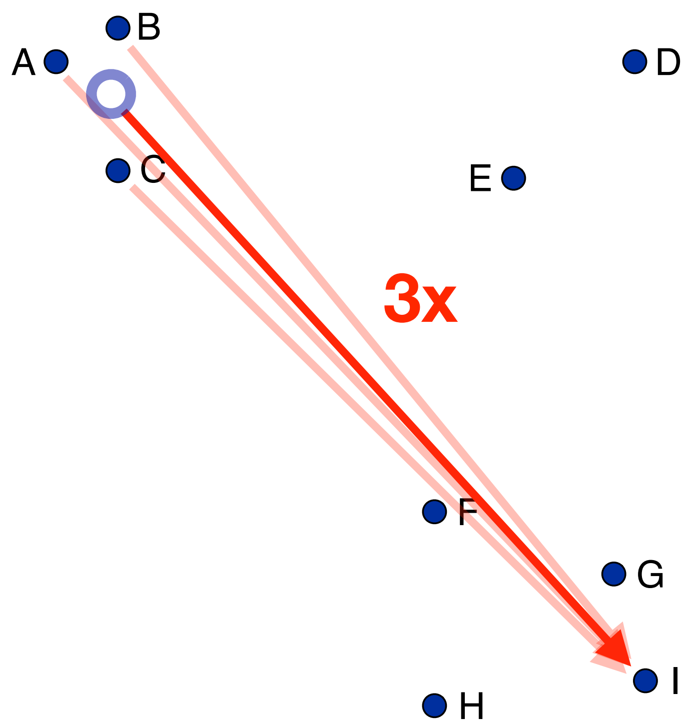
Barnes-Hut approximation

- Approximate such similar interactions by a *single* interaction:



Barnes-Hut approximation

- Approximate such similar interactions by a *single* interaction:



Barnes-Hut-SNE

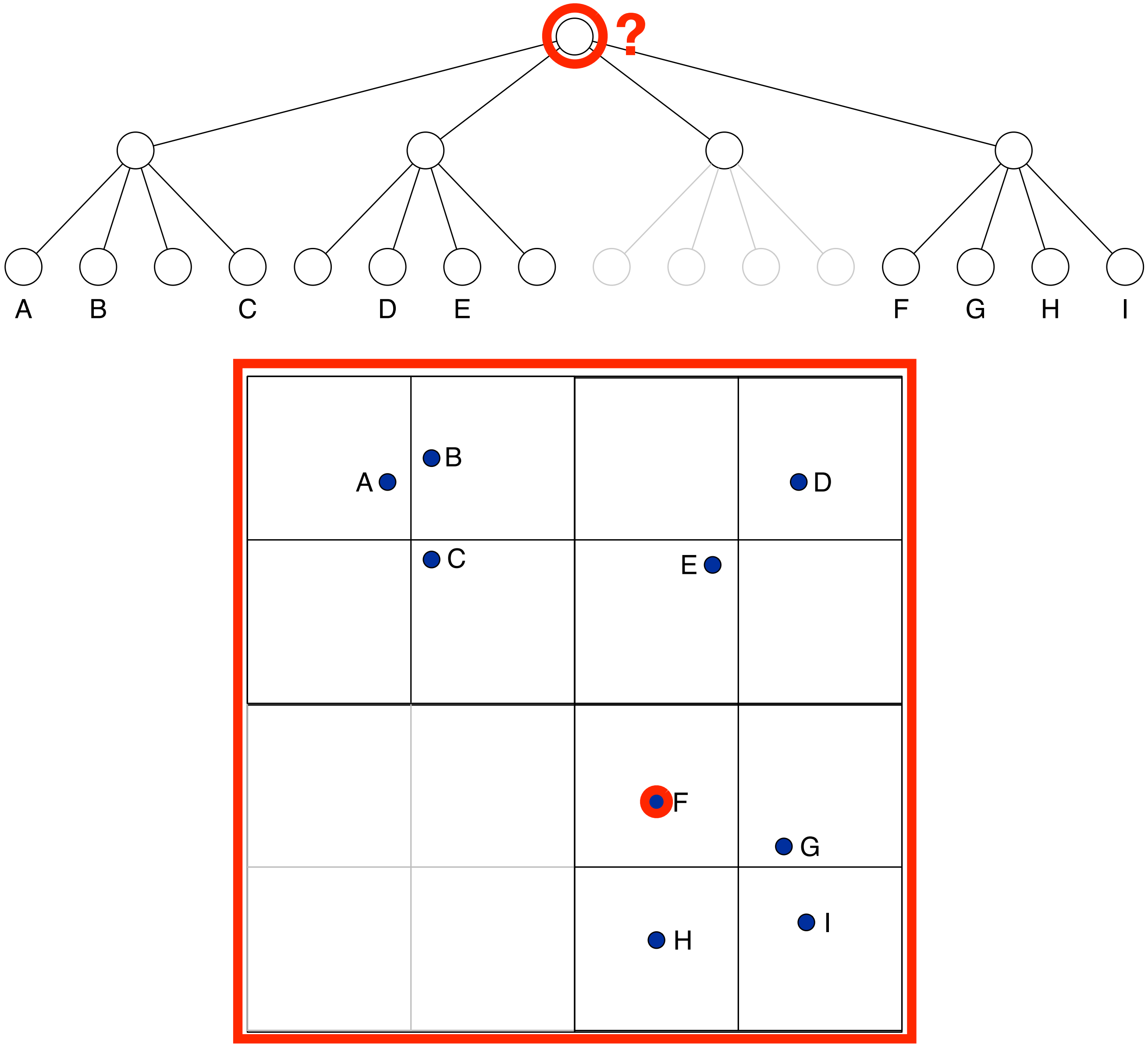
- Split up the t-SNE gradient into two main parts:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4(F_{attr} - F_{rep}) = 4\left(\sum_{j \neq i} p_{ij} q_{ij} Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_{j \neq i} q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)\right)$$

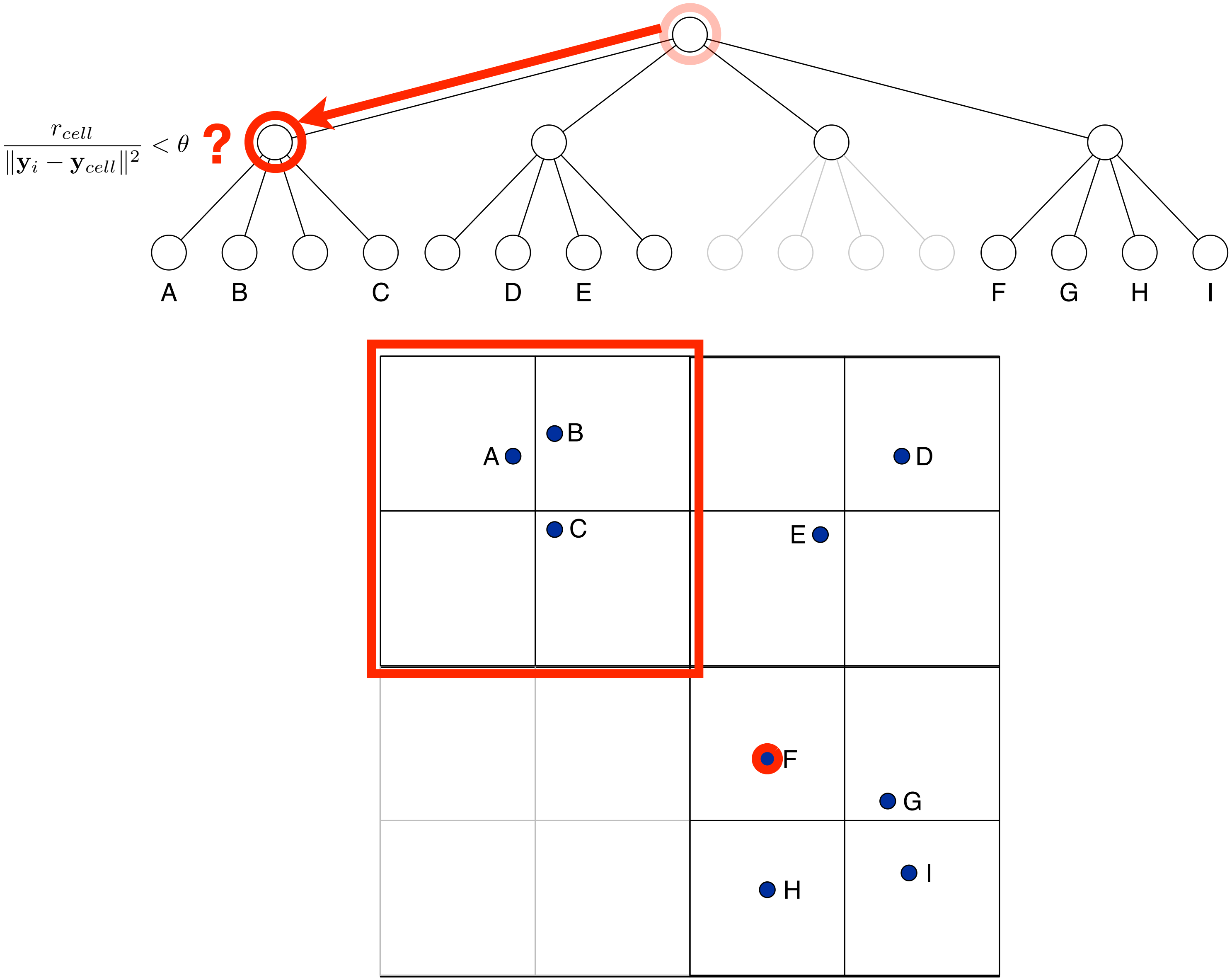
- Compute $\sum_{j \neq i} p_{ij} q_{ij} Z(\mathbf{y}_i - \mathbf{y}_j)$ exactly (possible because *P*-values are sparse)

- Approximate $\sum_{j \neq i} q_{ij}^2 Z^2(\mathbf{y}_i - \mathbf{y}_j)$ and Z with two Barnes-Hut algorithms

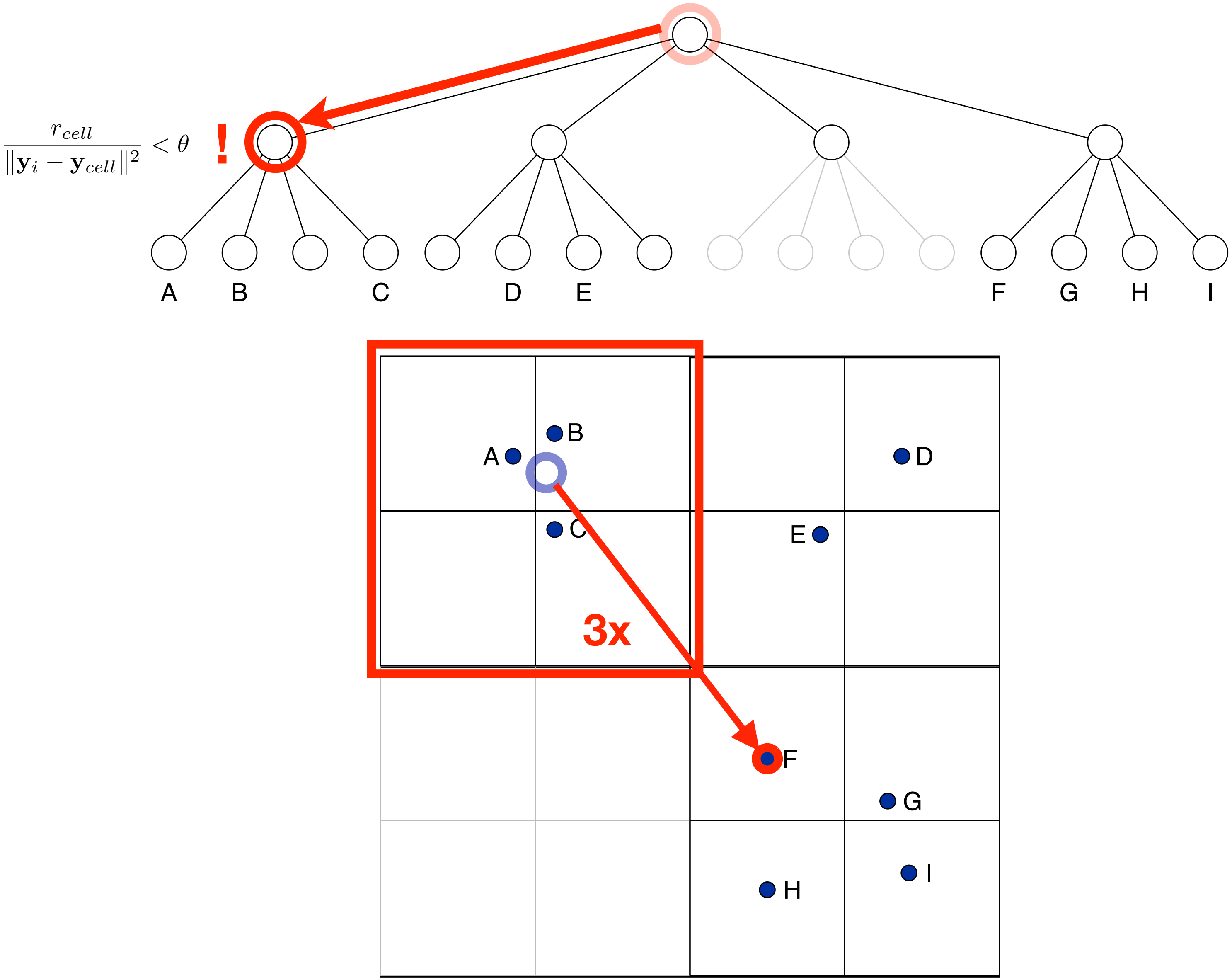
Barnes-Hut-SNE



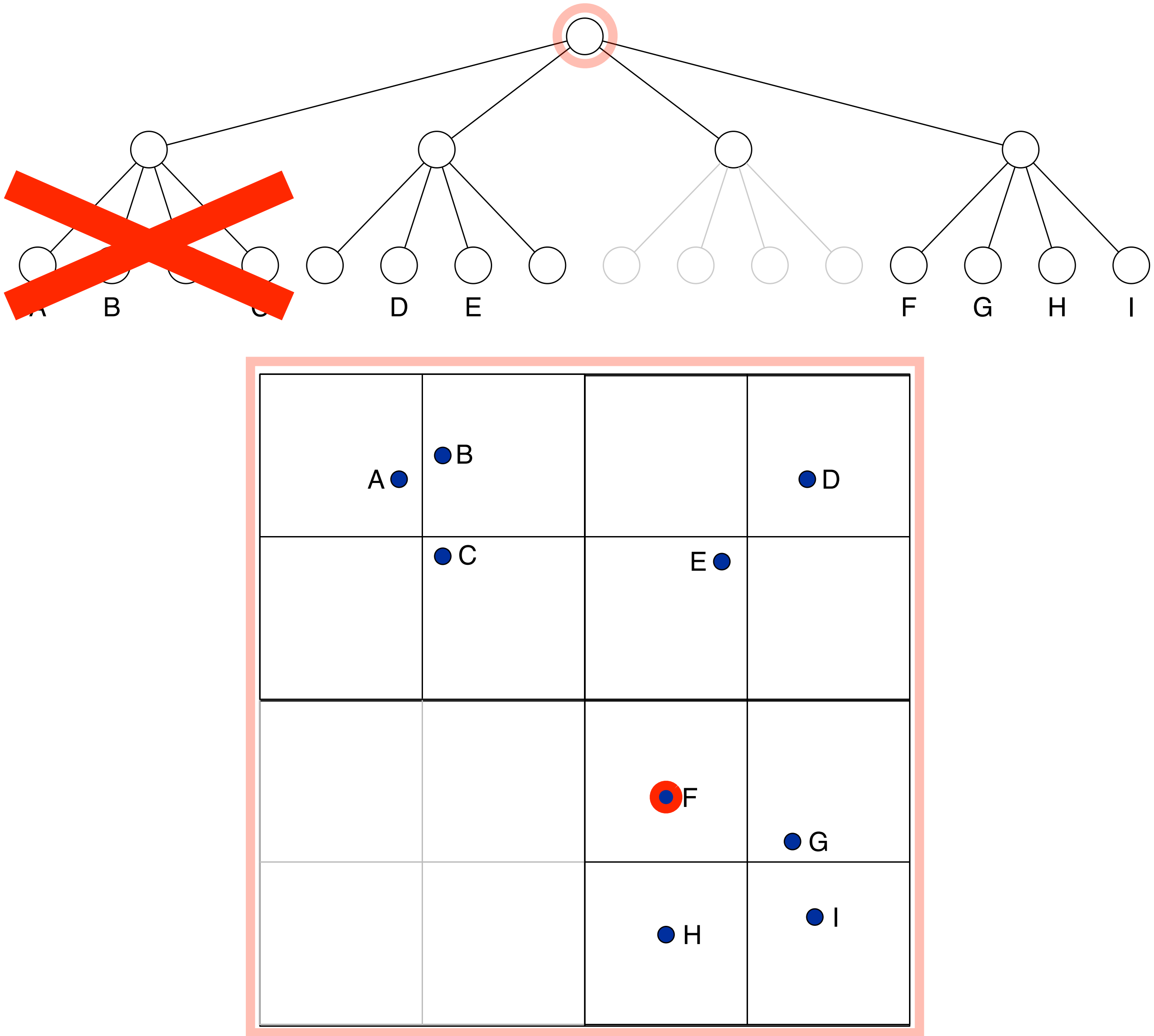
Barnes-Hut-SNE



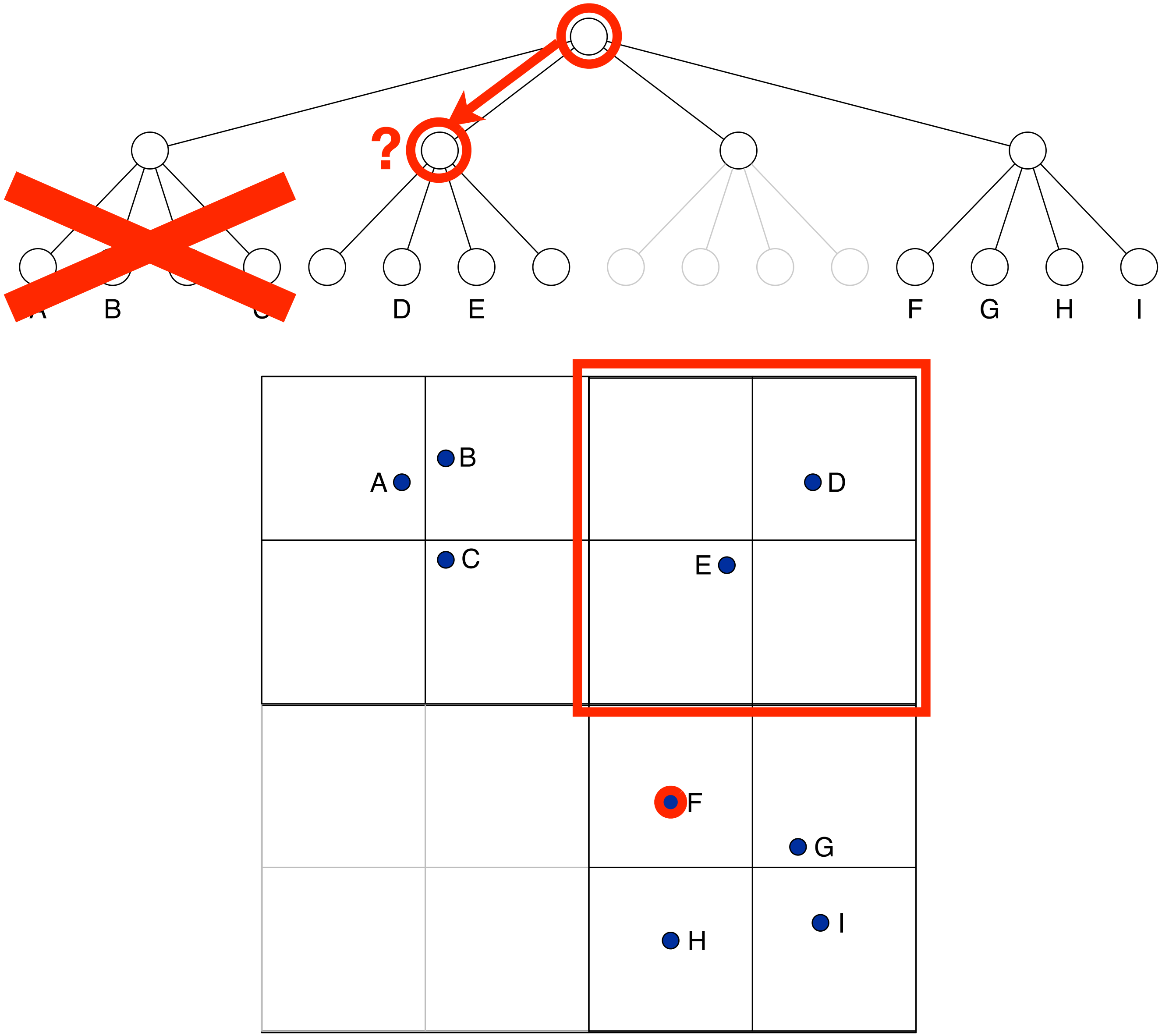
Barnes-Hut-SNE



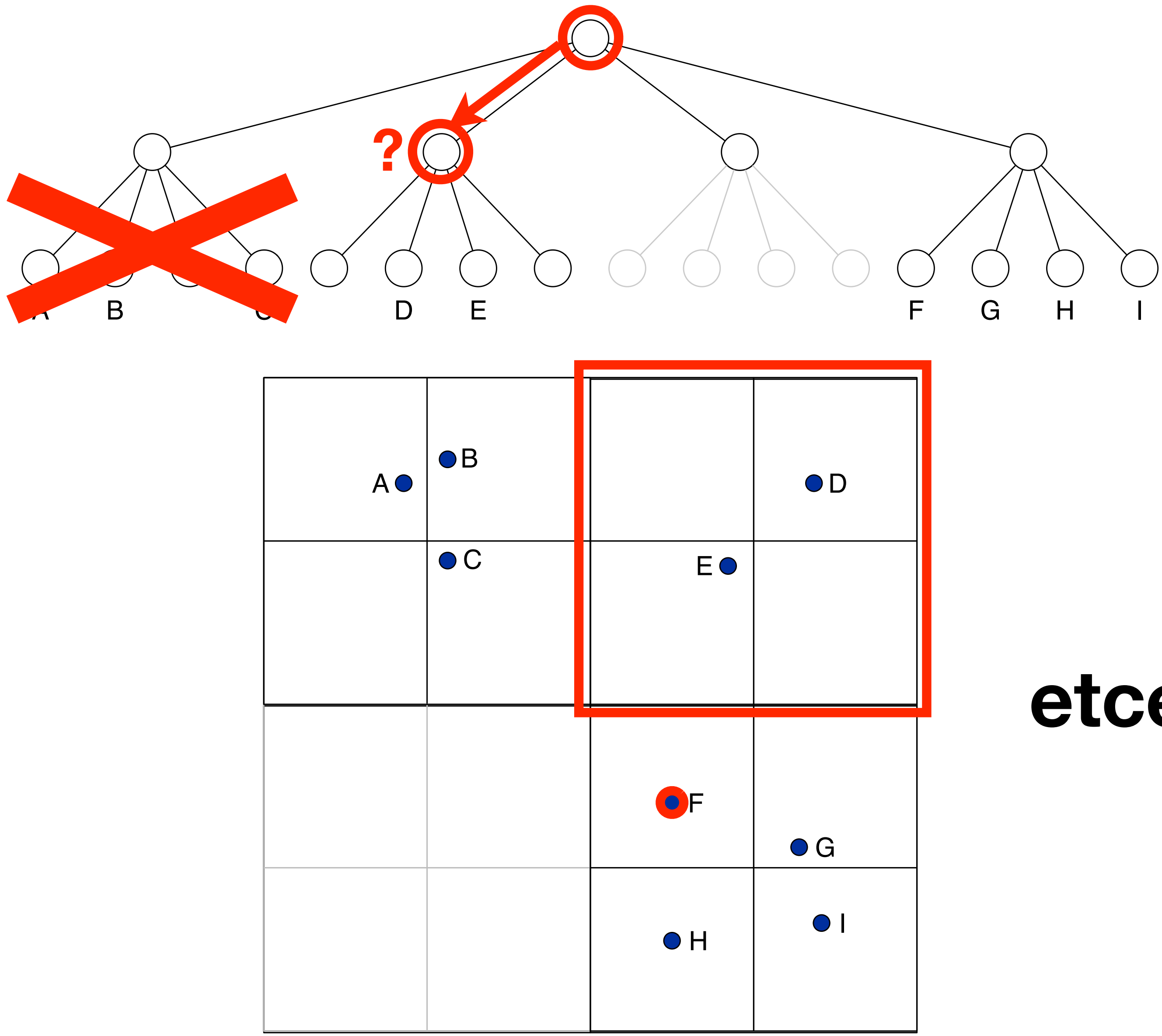
Barnes-Hut-SNE



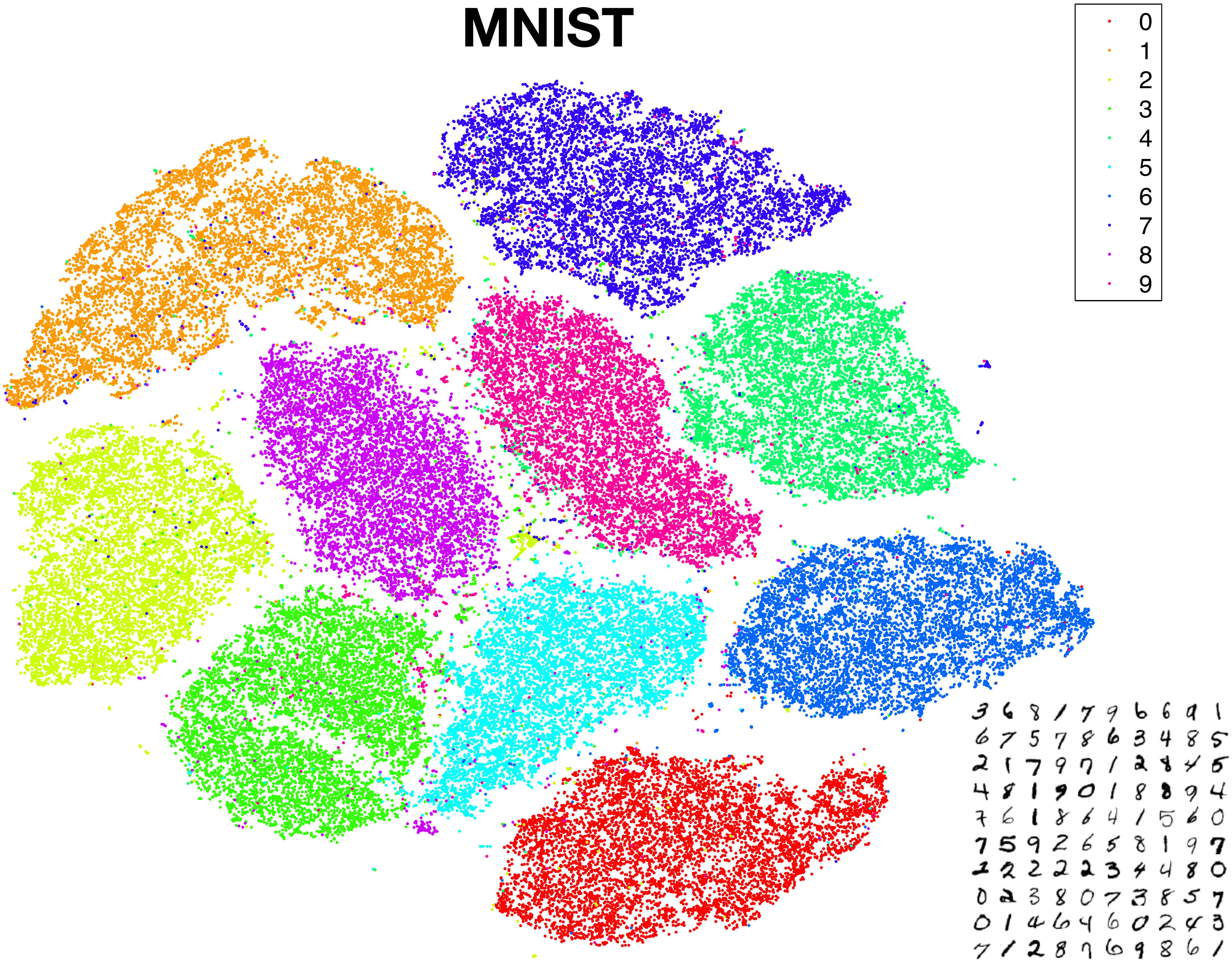
Barnes-Hut-SNE

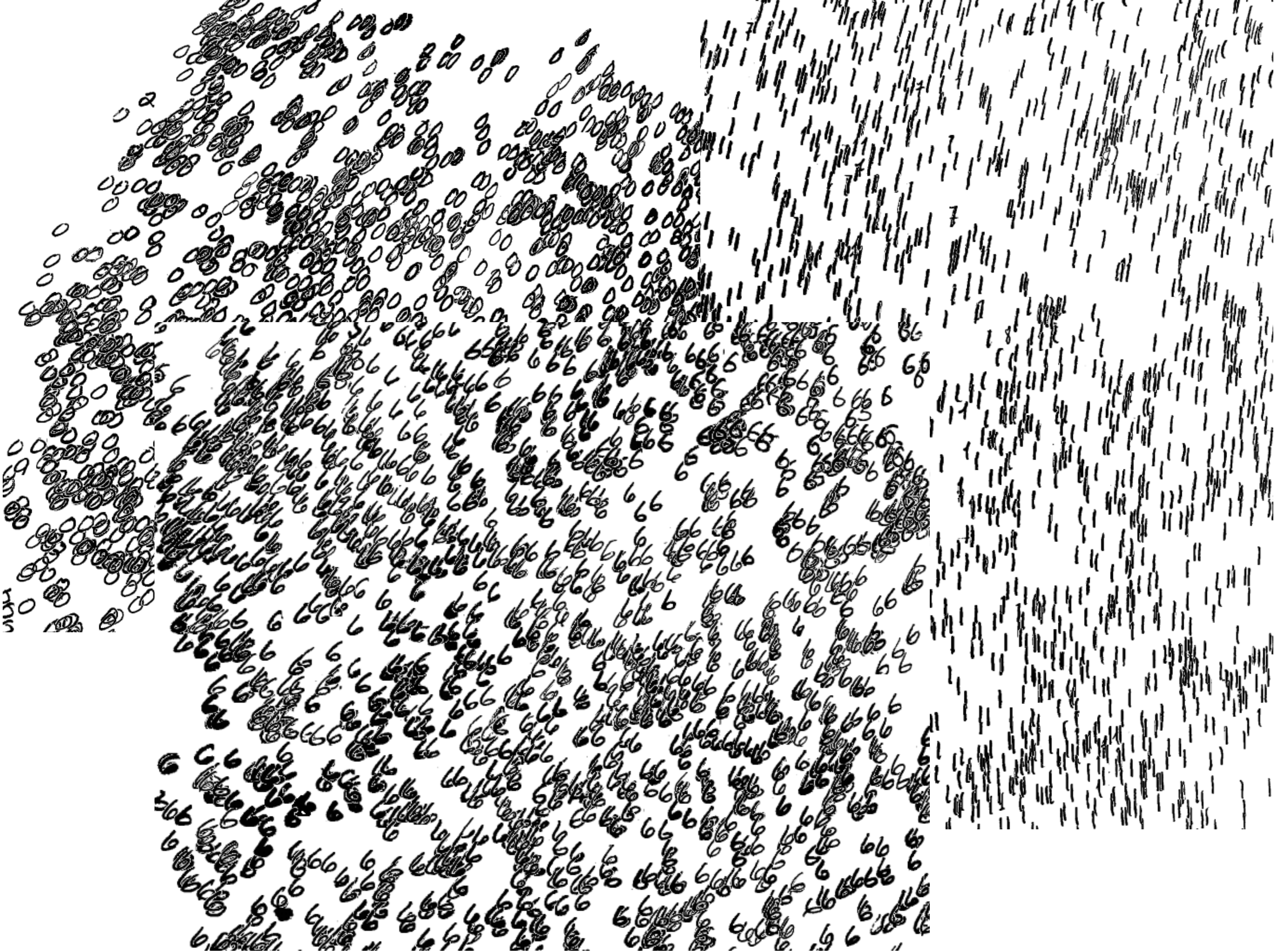


Barnes-Hut-SNE



etcetera...


MNIST
0
1
2
3
4
5
6
7
8
9



* Word map made by Joseph Turian at University of Montreal.

Conclusions

- Visualizing high-dimensional data in maps may lead to insight into “Big Data”

Conclusions

- Visualizing high-dimensional data in maps may lead to insight into “Big Data”

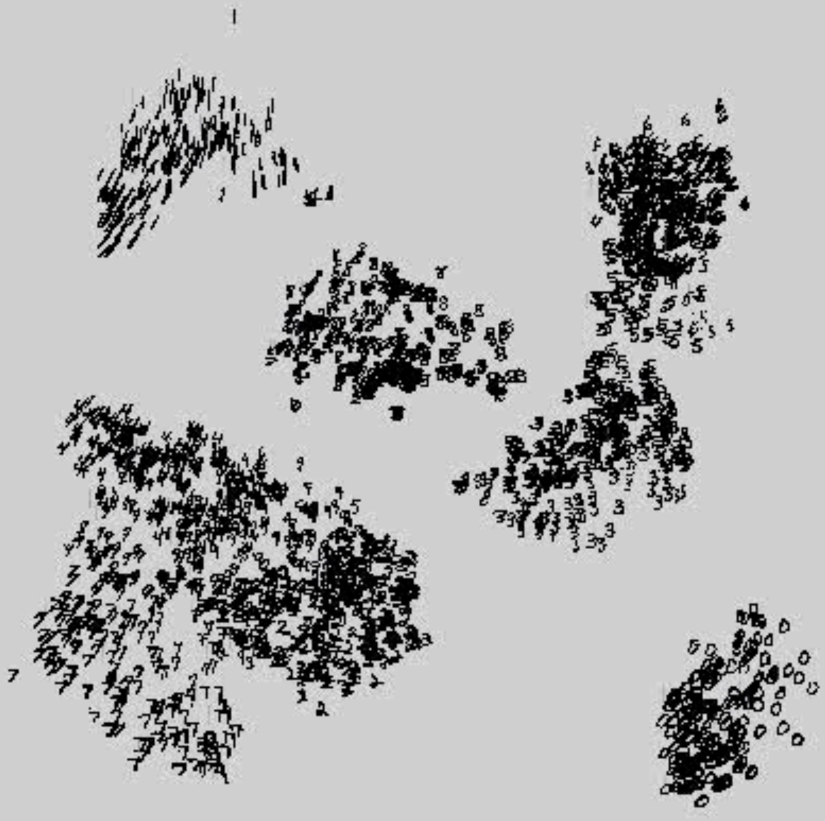
- t-SNE is an effective and efficient algorithm to make such maps

Conclusions

- Visualizing high-dimensional data in maps may lead to insight into “Big Data”

- t-SNE is an effective and efficient algorithm to make such maps

- t-SNE has already been successfully applied in a range of domains:
    - Bioinformatics, computer security, climate research, cancer research, *etc.*



QUESTIONS?

Try it out yourself! Code and papers are available on http://lvdmaaten.github.io/tsne
Shorter version of this talk is available on: http://www.youtube.com/user/GoogleTechTalks/videos