

DATA TRANSFORMATION

DATA TRANSFORMATION

- Our data is provided in a given form
 - ▶ Tabular (vectors)
 - ▶ Network
 - ▶ Time series
 - ▶ Text
 - ▶ Images
 - ▶
- To use the full potential of data mining, you might want to study it from multiple angles
 - ▶ How to convert from tabular to graph?
 - ▶ From Graph to Tabular?
 - ▶ From images/text to tabular (embedding)?

DIMENSIONALITY REDUCTION

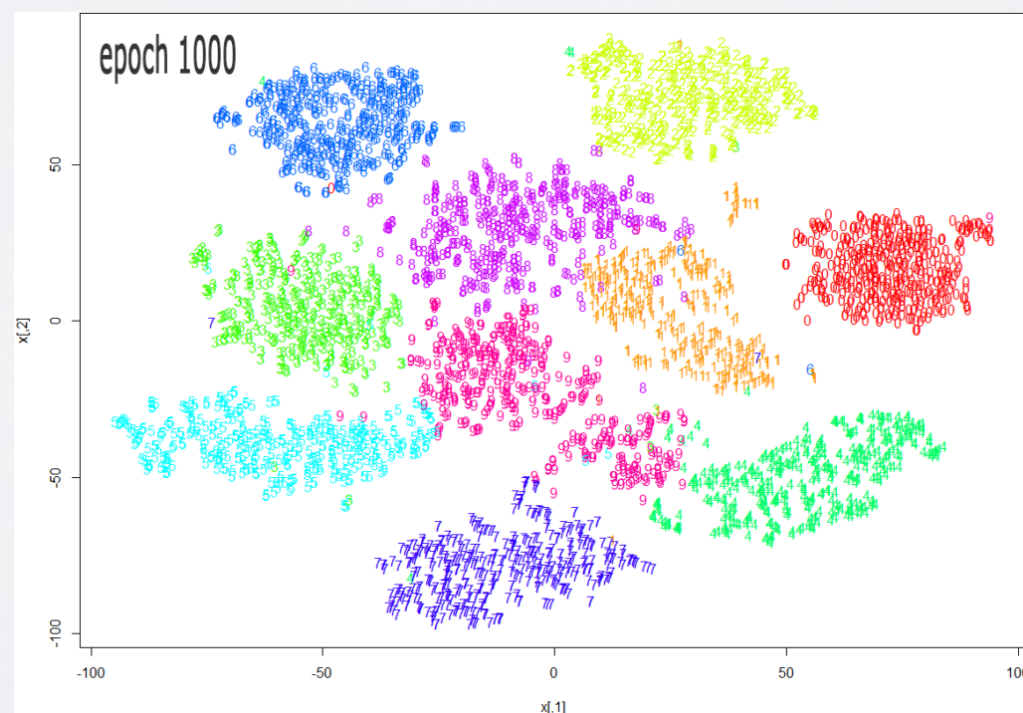
Low dimensionality embedding

DIMENSIONALITY REDUCTION

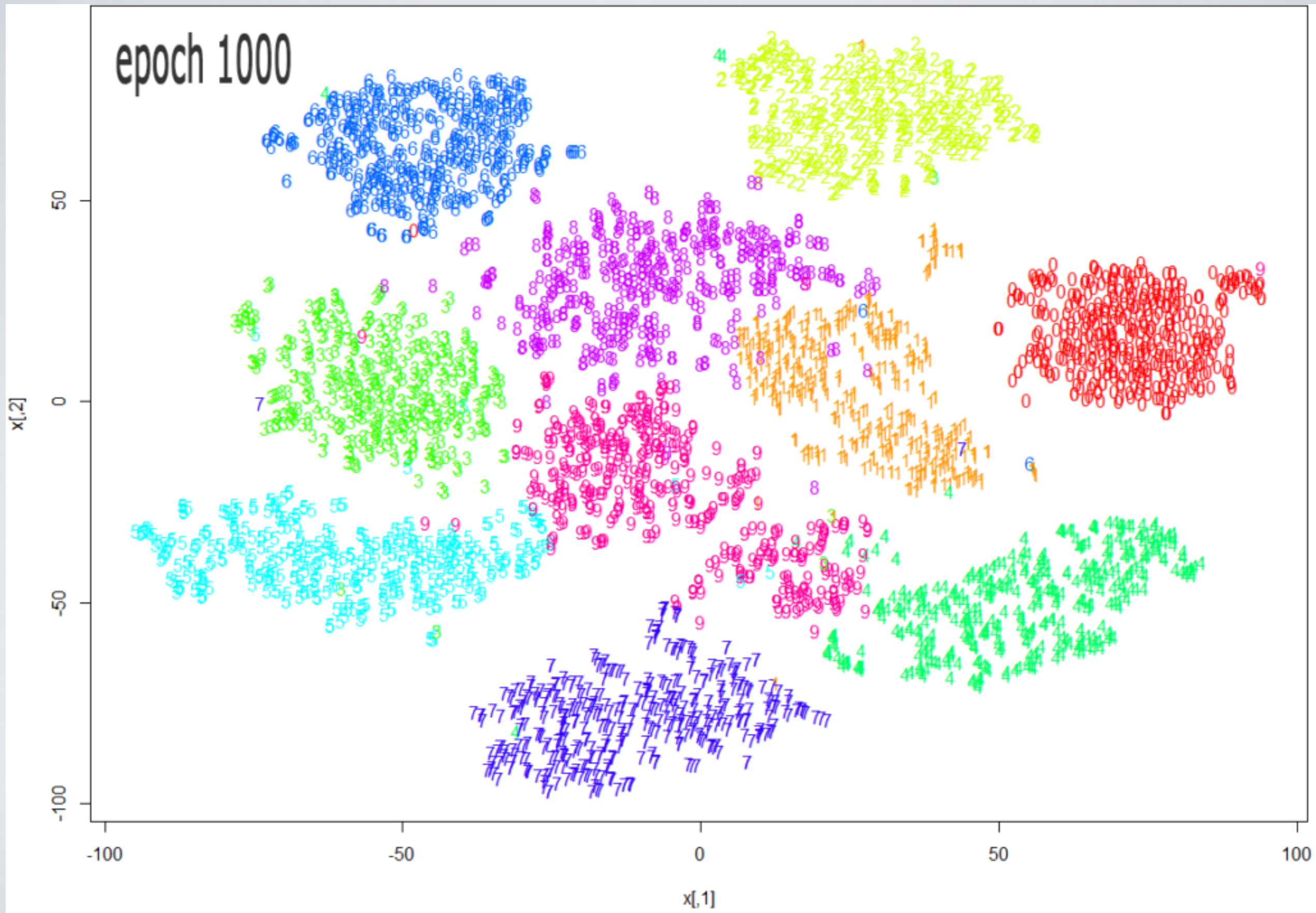
- Data Mining objective: understand our data
 - ▶ We get a dataset composed of many features
 - Or worst, complex object (image, sound, graph...)
 - ▶ How to understand the organization of our data?
 - ▶ How to perform clustering?

VISUALIZATION

- Your data is perfectly fine, but you want to intuitively understand how it is organized
 - Are there groups of similar objects?
 - Are my clusters meaningful?
 - Is my classification/clustering on some types of elements and not others.



epoch 1000



CURSE OF DIMENSIONALITY

- Having hundreds/thousands of attributes is a problem for data analysis.
 - e.g.: medicine: blood analysis, genomics.....
 - e.g.: cooking recipes: each column an ingredient...
- We want to reduce number of attributes while keeping most of the information
- Scalability

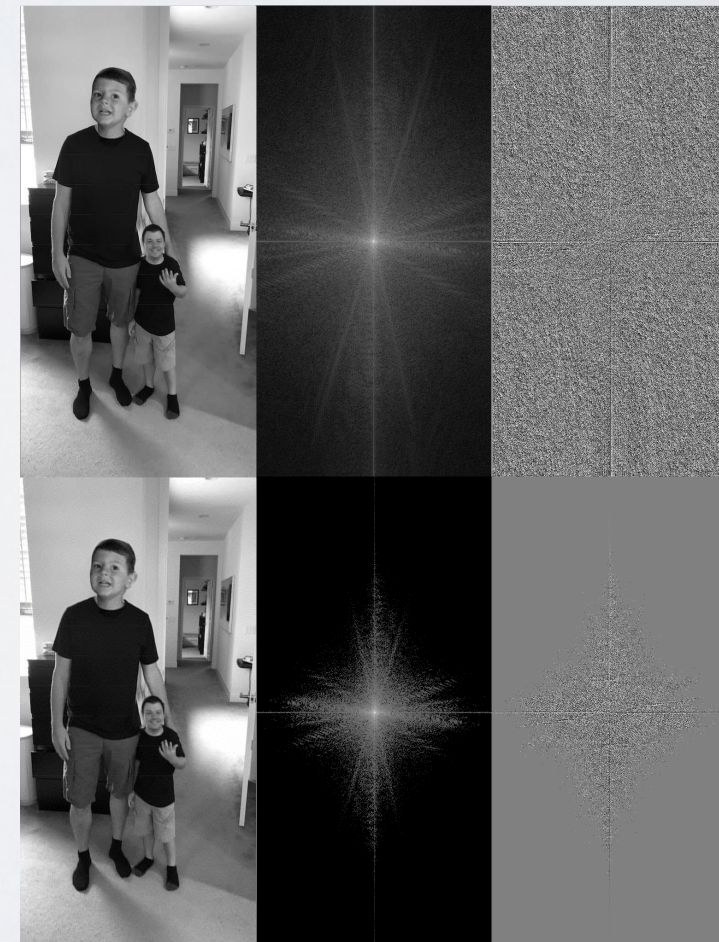
CORRELATION

- Assume that you have correlated features such as age, height and weight.
 - Linear regression will attribute the coefficients somewhat arbitrarily between them
 - Decision tree will spend a lot of time choosing between them for no reason
- Dimensionality reduction can create a single variable to capture what is common
 - The rest can be lost or captured by another feature,
 - i.e., height - average height for that age, “residuals”

PCA

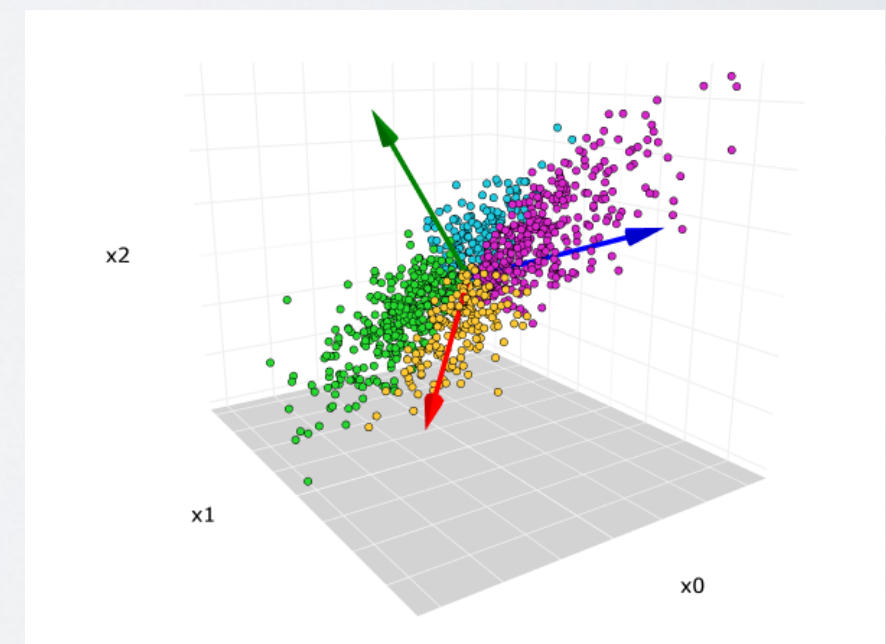
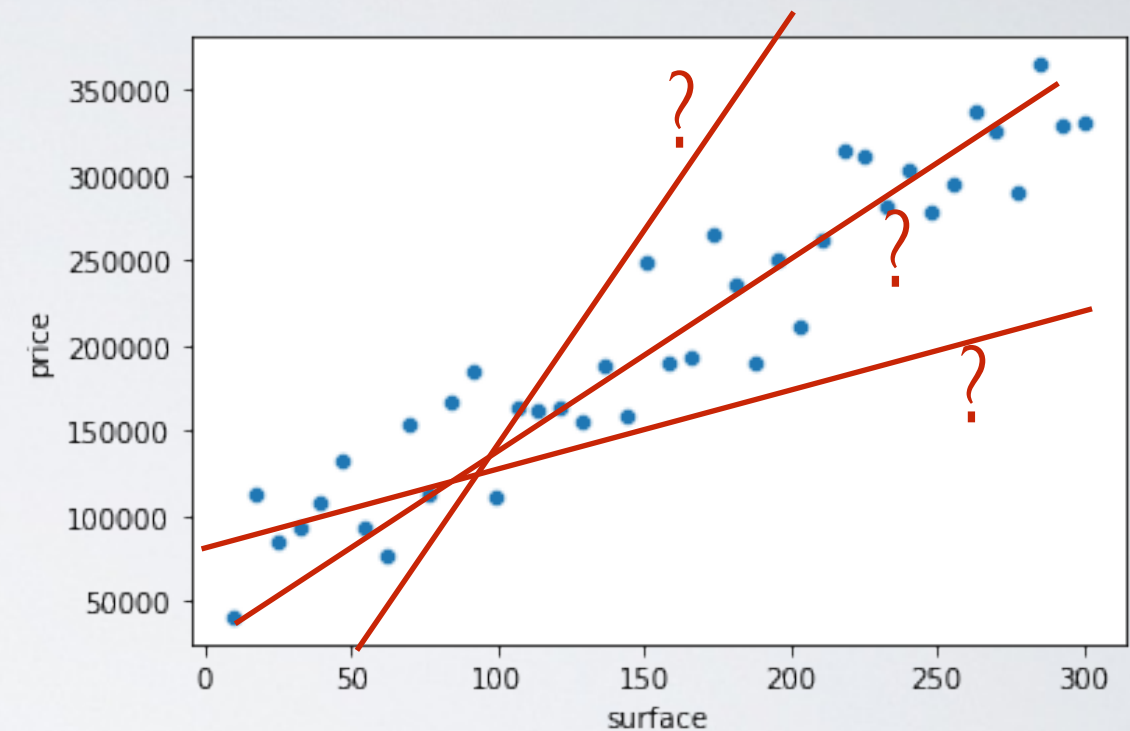
PCA

- PCA: Principal Component Analysis
- Defines new dimensions that are linear combinations of initial dimensions
 - Objective: concentrate the **variance** on some dimensions
 - So that we can keep only these ones.
 - Those we remove contain low variance, thus low information
- Similar principle than the Fourier transform technique for image compression

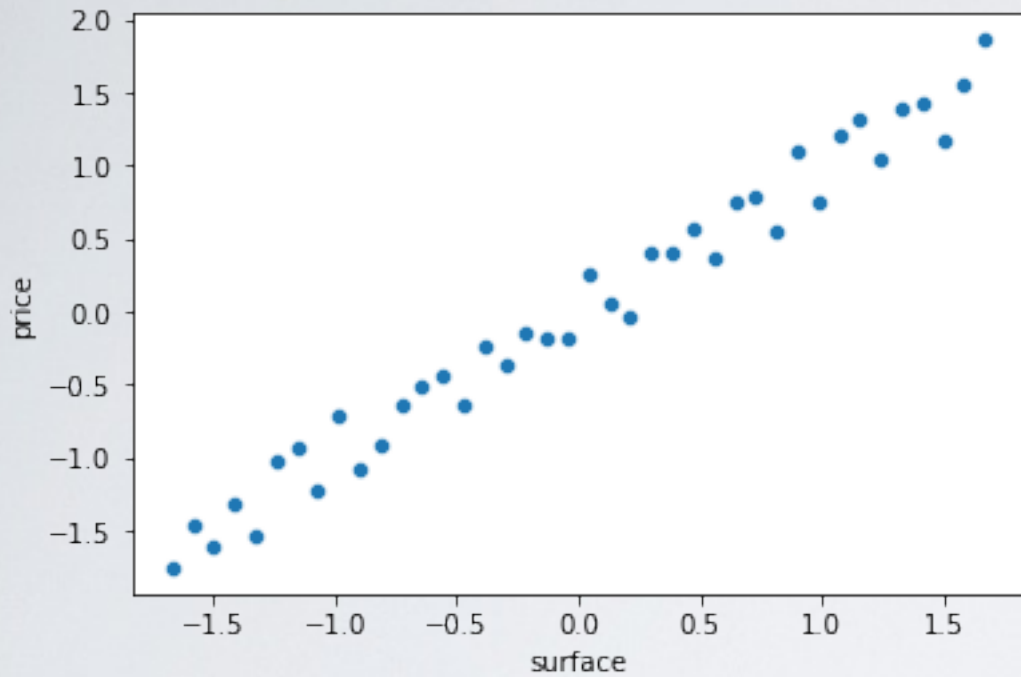


PCA

- Algorithm:
 - ▶ 1) Find an “axis”, a unit vector defining a line in the space
 - That minimizes the variance \Rightarrow the squared distance from all points to that line
- 2) For d in $2-(\text{initial_}d)$
 - ▶ Find another axis, with two constraints:
 - Orthogonal to all previous axis
 - Among those, minimizing the variance
- 3) At the end, keep the first k dimensions
 - ▶ Some information is lost



EXAMPLE PCA 2D

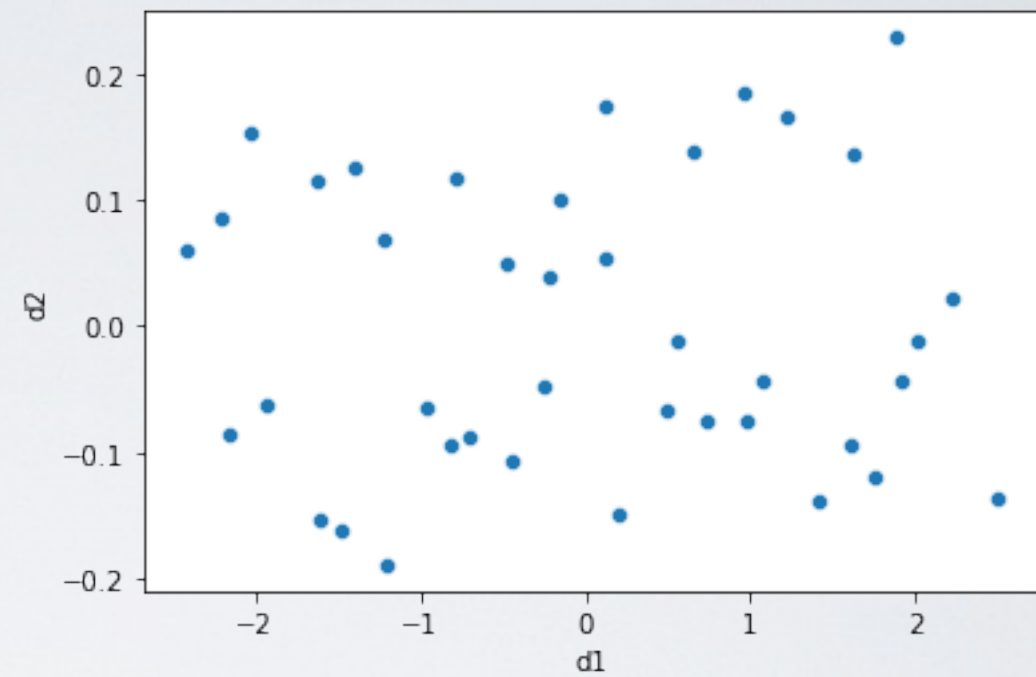


Covariance matrix (original)

```
[1.          , 0.98675899],  
 [0.98675899, 1.          ]
```

Sum of variance
2

Variance by dimension
1 1



Covariance matrix (pca)

```
[ 1.98675899e+00, 0],  
 [0, 1.32410092e-02]
```

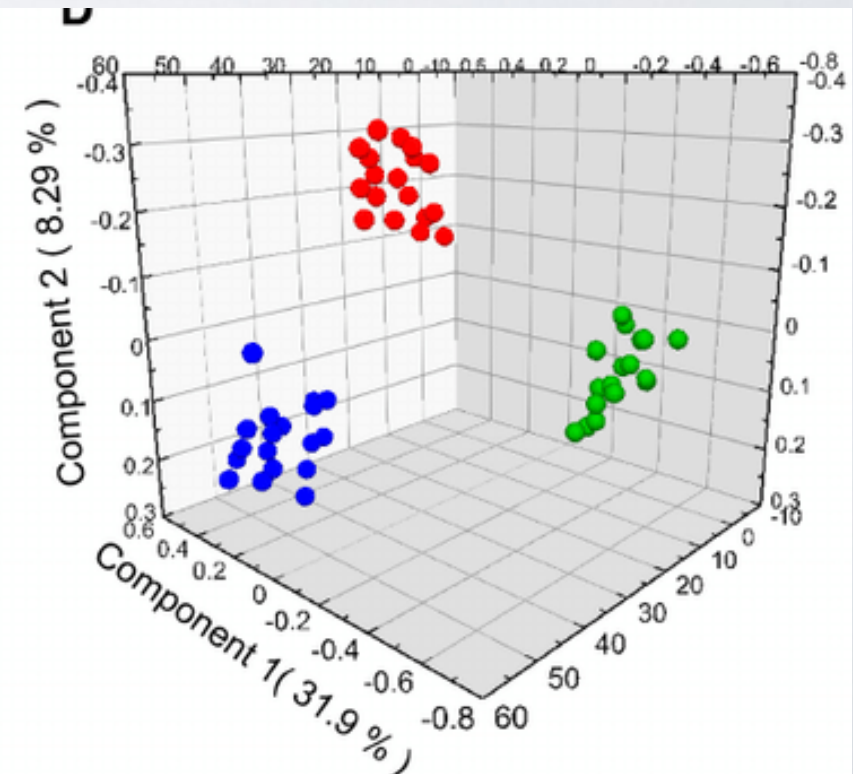
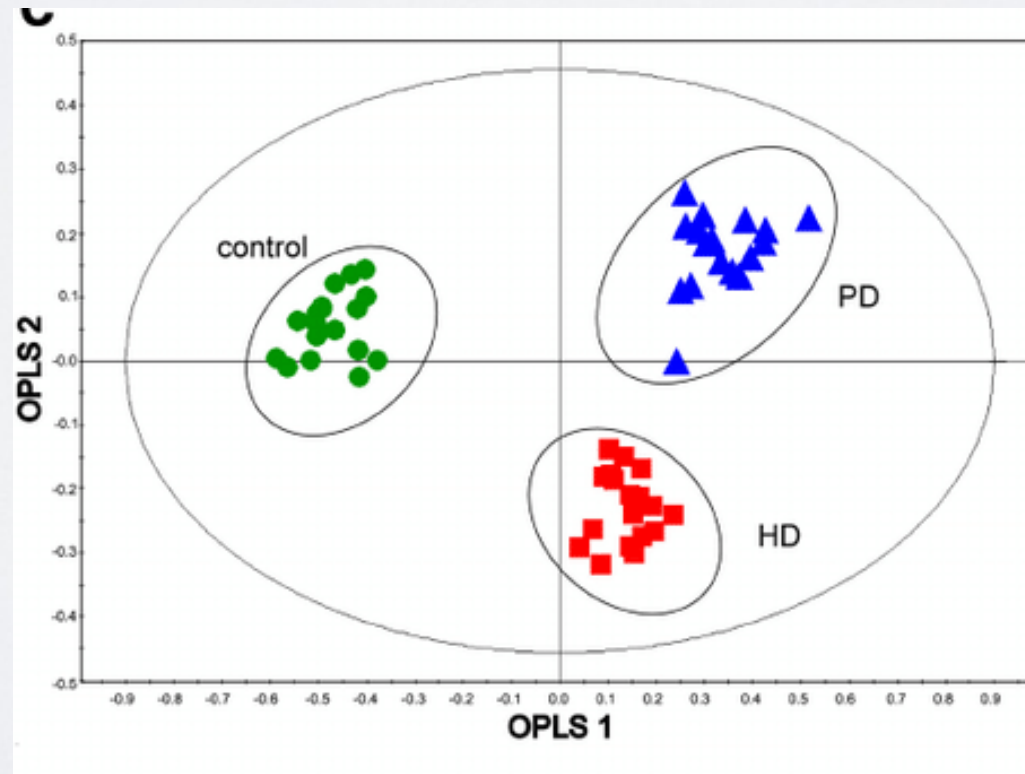
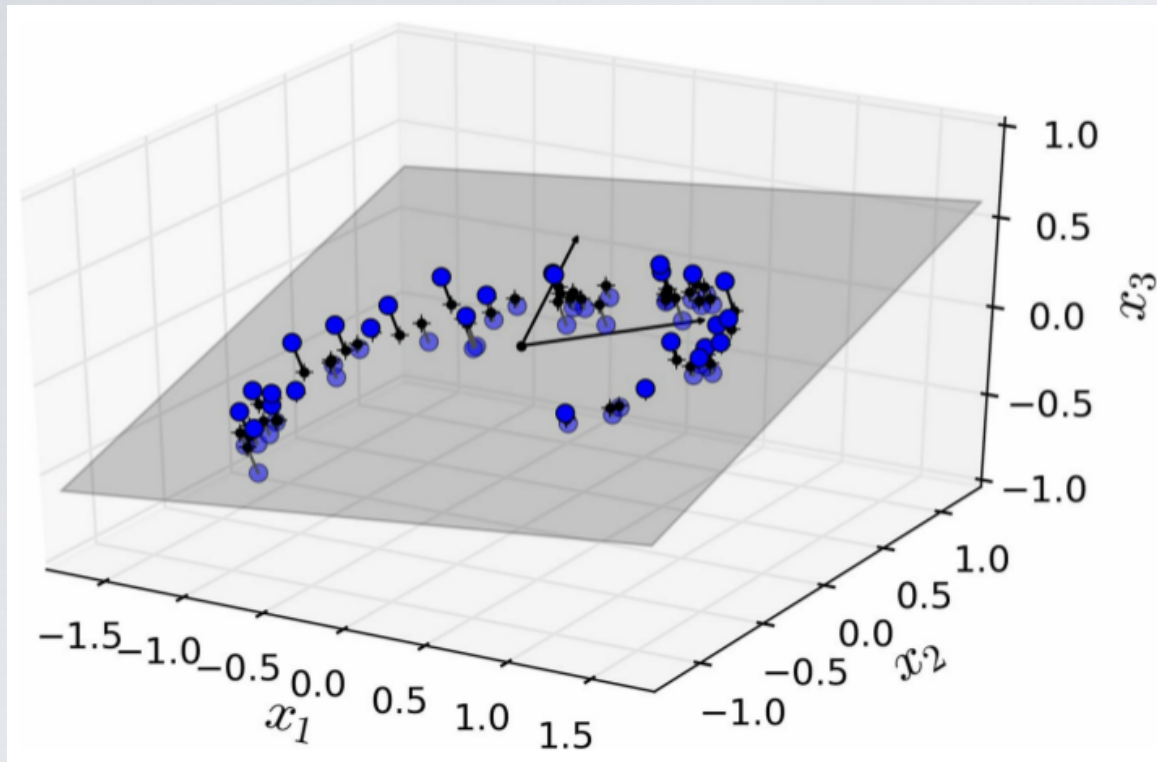
Sum of variance
2

Variance by dimension
1.98675899 0.01324101

Explained variance(ratio)

```
[0.9933795, 0.0066205]
```

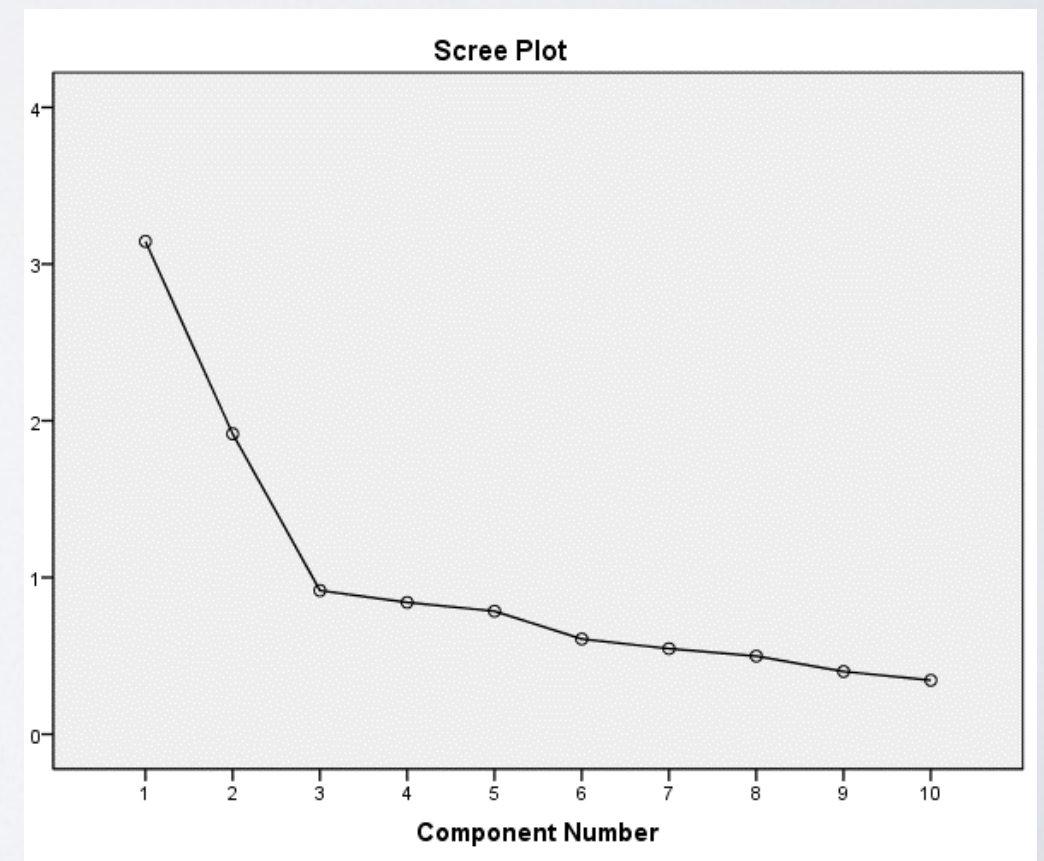

3D \Rightarrow 2D



CHOOSING COMPONENTS

- How to choose k?
 - ▶ Elbow method
 - ▶ OR fix beforehand a min threshold of explained variance, e.g.: 80%
 - We are fine with losing 20% of information

Explained
variance



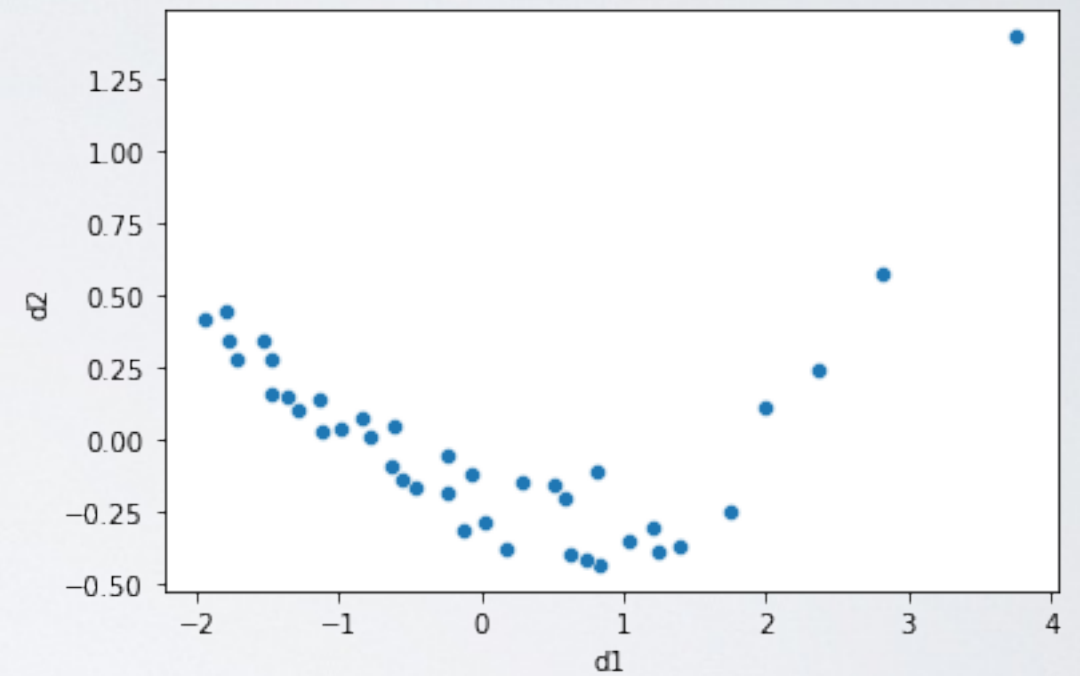
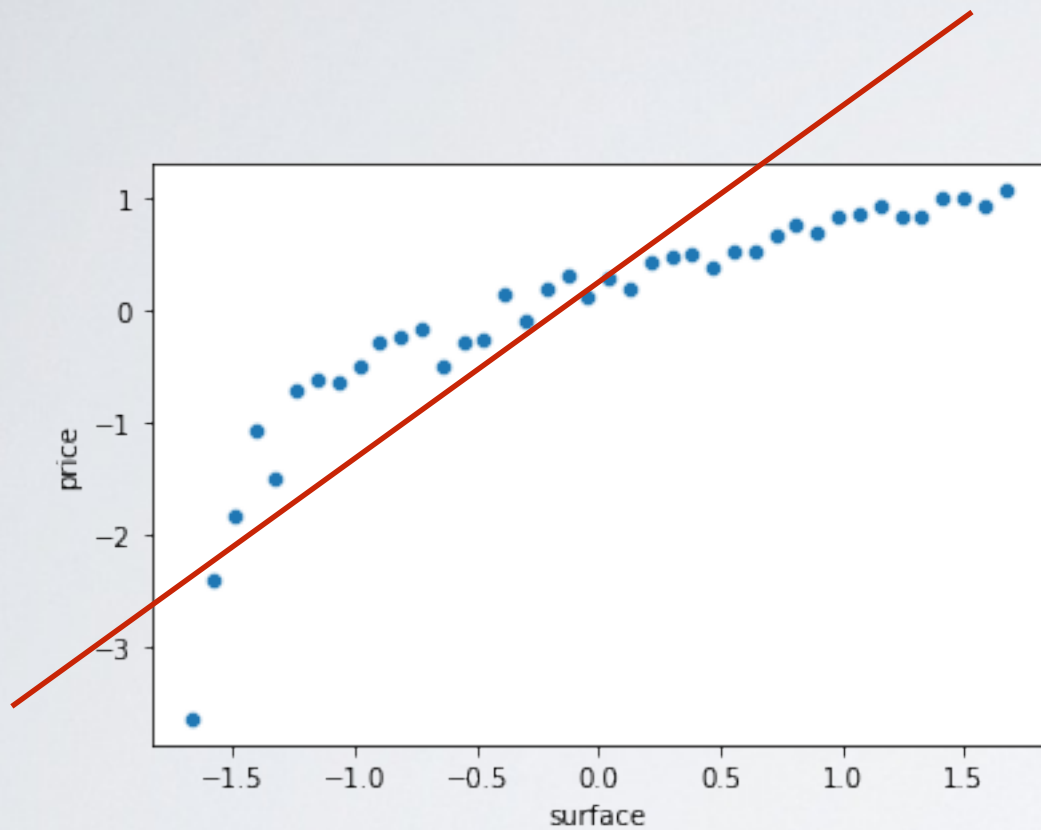
COMPUTATION IN PRACTICE

- Find the eigenvectors of the covariance matrix of centered data
- If you want k new dimensions, pick the k eigenvectors associated with the k largest eigenvalues
 - Eigenvalues = explained variance
- The eigenvectors corresponding to the top eigenvalues are coefficients of the linear transformation

PCA POPULARITY

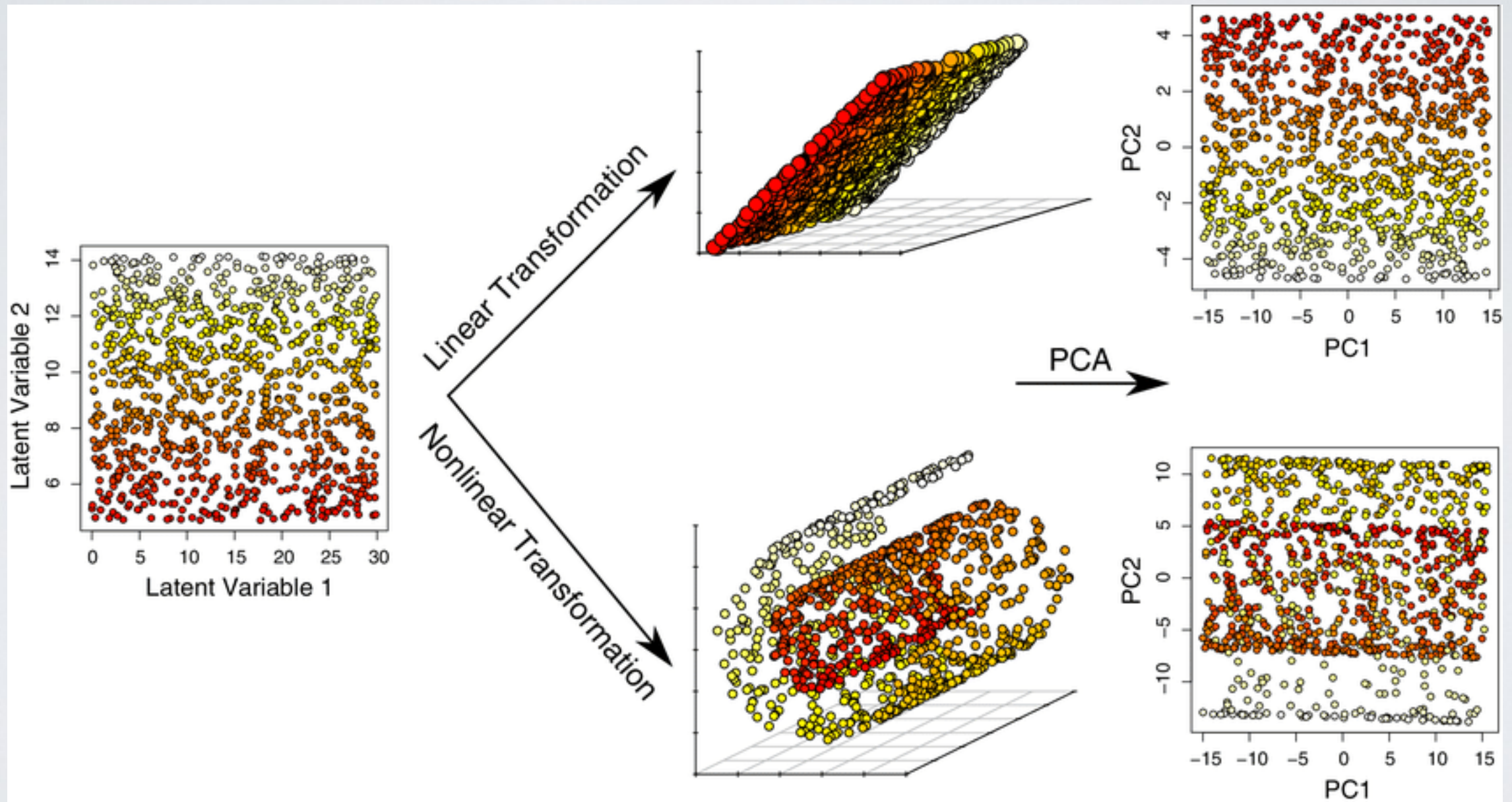
- Why is PCA popular?
- Similar reasons than linear regression:
 - ▶ Useful
 - Eliminate correlations
 - ▶ Analytical solutions
 - Guarantee to find the global minimum of the objective
 - Could be done before modern computers
 - ▶ Interpretable solution
 - ▶ Intuitively pleasant
- No reason to consider it “better” than other methods for dimensionality reduction...

NON-LINEAR SITUATIONS



Pearson correlation(d1,d2): 0

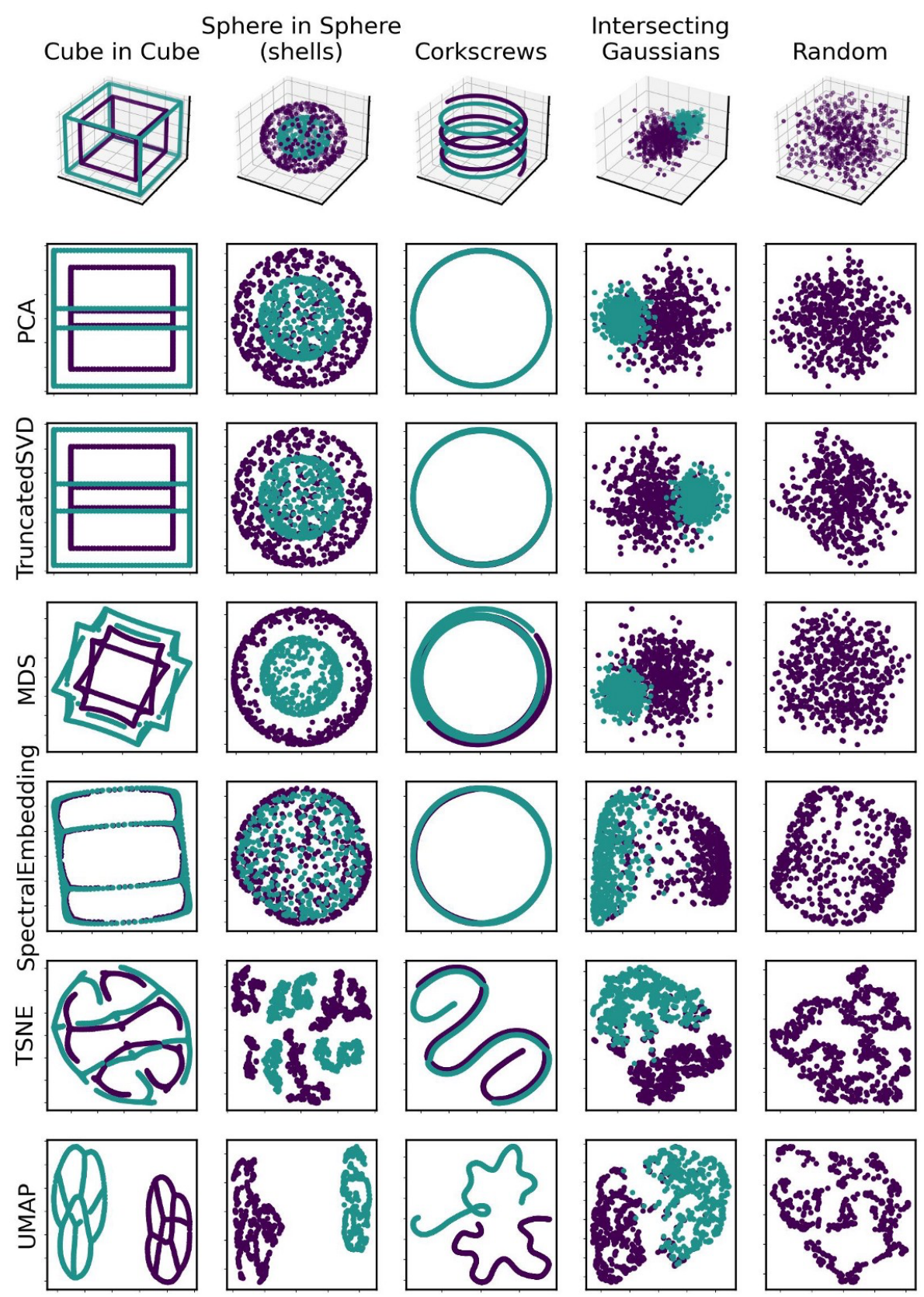
NONLINEAR DATA



MANIFOLDS

MANIFOLDS

- Manifolds are another approach to dimensionality reduction
- The general principle is to
 - 1) Define a notion of distance between elements in the original space
 - 2) Define a notion of distance between elements in a reduced, target space
 - 3) Minimize the difference between distances in original and target space
- In many cases, the process is nonlinear, i.e., we choose distances such as
 - We care more about preserving the distance for items “close” in space than for those “far” from each other

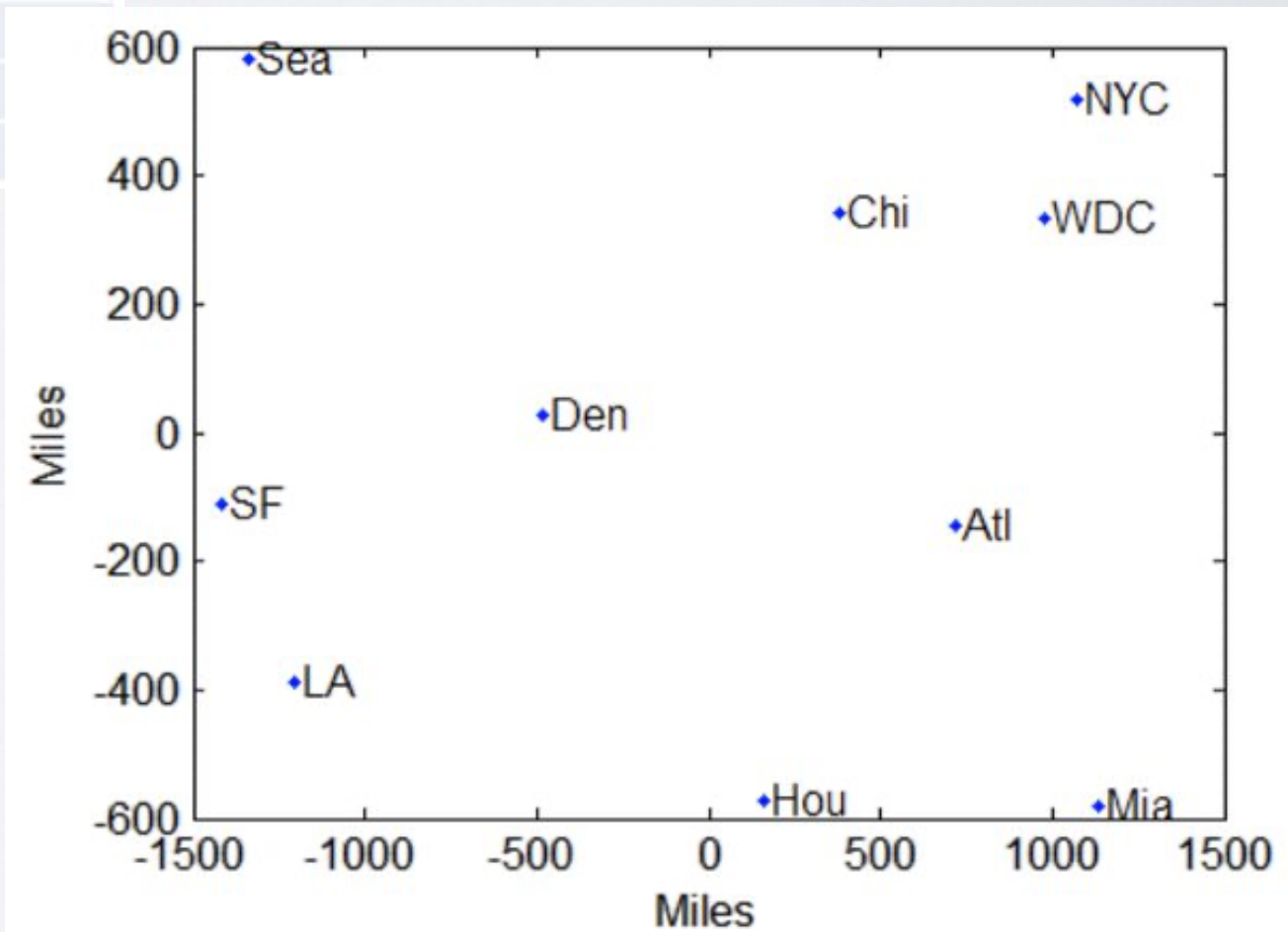


MDS

- MDS: Multi-dimensional Scaling:
 - Simply minimize distance between original space and target space
 - e.g., d-dimensional forced to 2-dimensional
- How to do it?
 - 1) Compute all (squared Euclidean) pairwise distances between items => similarity matrix
 - $n \times f$ matrix => $n \times n$ matrix
 - Apply double-centering
 - 2) Compute PCA on this similarity matrix
 - PCA preserves column information => preserves distance on a similarity matrix
- Problems:
 - Very costly (nb features=nb elements), n^2
 - Try to preserve all distances, therefore extremely constrained

MDS

	Atl	Chi	Den	Hou	LA	Mia	NYC	SF	Sea	WDC
Atl	0	587	1212	701	1936	604	748	2139	2182	543
Chi	587	0	920	940	1745	1188	713	1858	1737	597
Den	1212	920	0	879	831	1726	1631	949	1021	1494
Hou	701	940	879	0	1374	968	1420	1645	1891	1220
LA	1936	1745	831	1374	0	2339	2451	347	959	2300
Mia	604	1188	1726	968	2339	0	1092	2594	2734	923
NYC	748	713	1631	1420	2451	1092	0	2571	2408	205
SF	2139	1858	949	1645	347	2594	2571	0	678	
Sea	2182	1737	1021	1891	959	2734	2408	678	0	
WDC	543	597	1494	1220	2300	923	205	2442	2329	



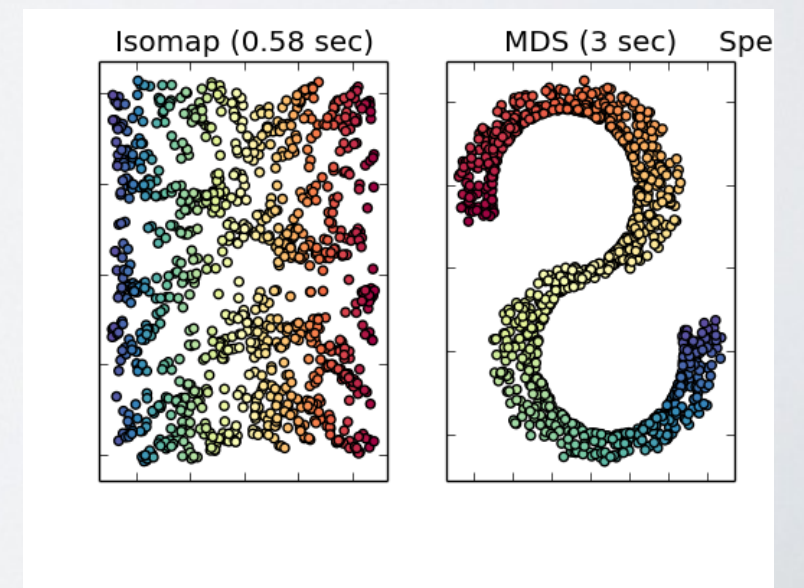
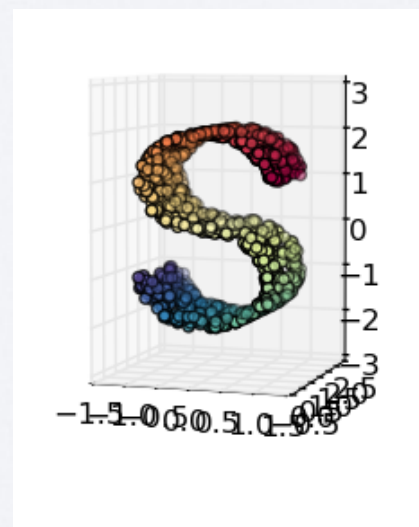
ISOMAP

- Variation of MDS

- ▶ 1) We define a graph such as two elements are connected if they are at distance < threshold. (Alternative: fixed number of neighbors)
 - Put a weight on edges = euclidean distance
- ▶ 2) Compute a similarity matrix, such as distance = weighted shortest path distance
- ▶ 3) Apply MDS on it

- Computing shortest paths on a graph is (relatively) fast

- ▶ Floyd–Warshall algorithm



T-SNE

T-SNE

- t-SNE : t-distributed stochastic neighbor embedding
- Non-linear dimensionality reduction
- Currently the most popular method for visualizing data in low dimensions

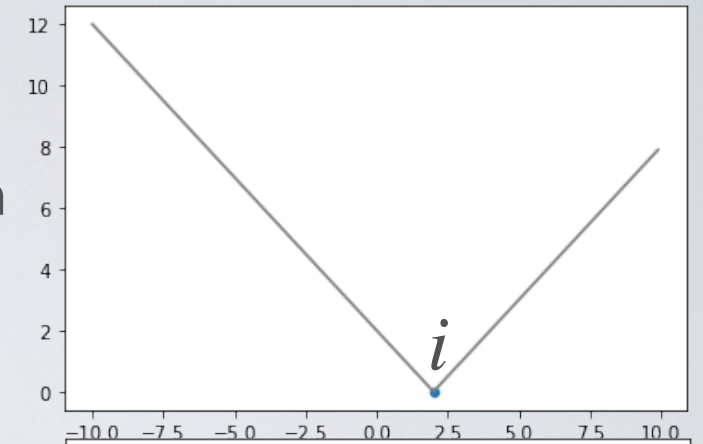
T-SNE

- General principle:

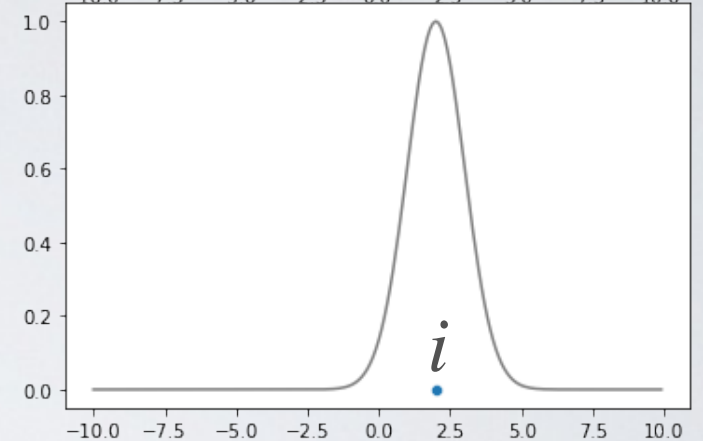
- ▶ Define a notion of similarity $p_{j|i}$ in the high dimensional space P
 - Based on normal distribution
- ▶ Define a notion of similarity $q_{j|i}$ in the low dimensional space Q
 - Based on student-t distribution, tends to “exaggerate” differences
- ▶ For each point of initial coordinates x_i , find a new coordinate y_i in the lower dimensional space, such as to minimize the difference between P and Q
 - $\forall_{i,j} p_{j|i} \approx q_{j|i}$

SNE

Euclidean



Normal



- Distance in the original space P

- ▶ To compute how far j is from i , consider a normal distribution centered in j with variance σ

- ▶ Mathematically: the raw distance is given as $s_{j|i}^P = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$

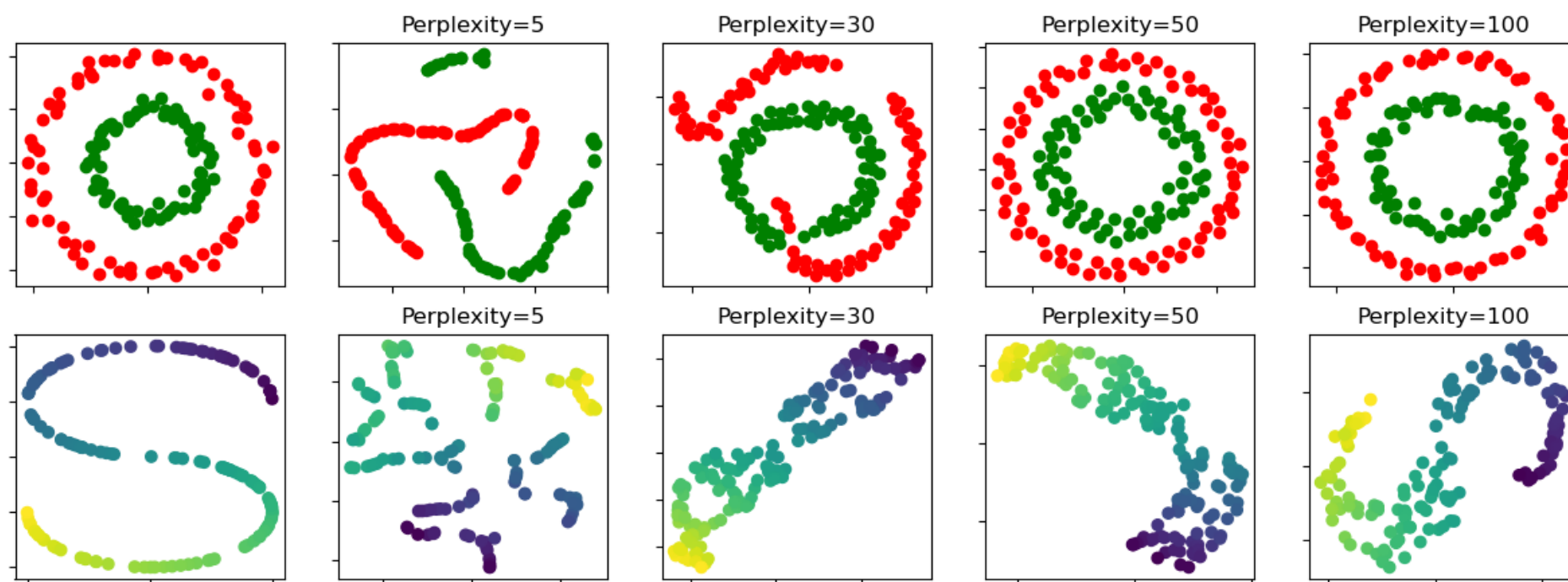
- ▶
$$p_{j|i} = \frac{s_{j|i}^P}{\sum_{k \neq i} s_{j|k}^P}$$

- Normalizes the similarity by sum of similarity to all other points.
- With proper σ , local definition of similarity

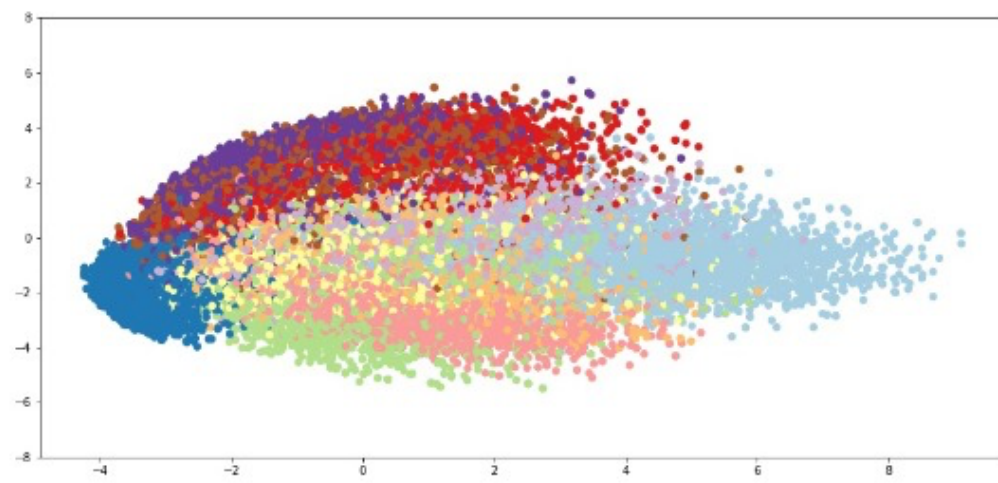
T-SNE: PERPLEXITY

- There is a perplexity parameter σ : it controls how much each point cares more about close neighbors compared with farther neighbors
 - Low σ : Preserve mostly local distances
 - High σ : Give more importance to long-range distances
 - More expensive, more similar to MDS

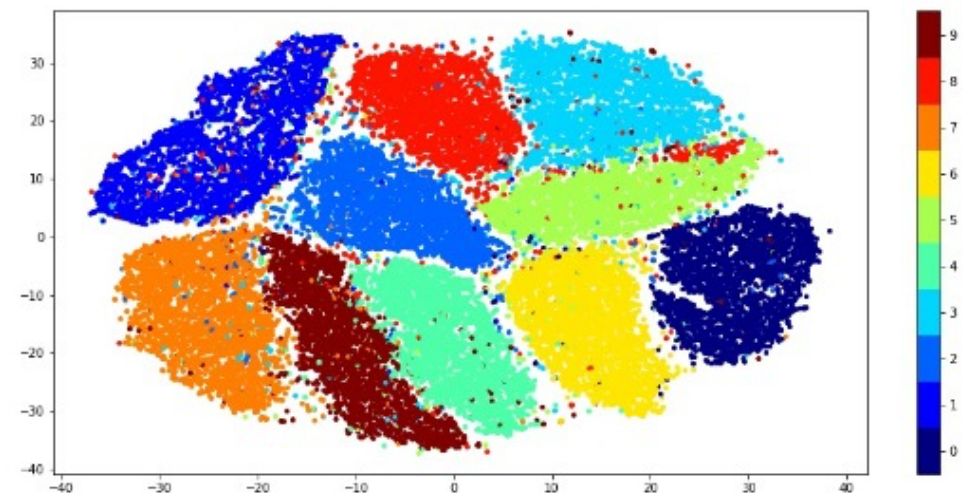
INFLUENCE OF PERPLEXITY



MNIST - PCA



MNIST - TSNE



LOW DIMENSIONAL EMBEDDINGS

EMBEDDINGS

- A recent usage of low dimensional embeddings is to encode complex objects as vectors
 - ▶ Words as Vector \Rightarrow Word2Vec
 - ▶ Nodes (of graph) as Vectors \Rightarrow Node2Vec
 - ▶ Documents as Vectors \Rightarrow Doc2Vec
 - ▶

WORD EMBEDDING

WORD EMBEDDING

- Words can be understood as a (very) high dimensional space
 - Using One Hot encoding: vocabulary of 1000 words=> 1000 columns
- Could we assign a vector in “low dimension”, encoding the “semantic” of a word?
 - Two words with similar meanings should be close

SKIPGRAM

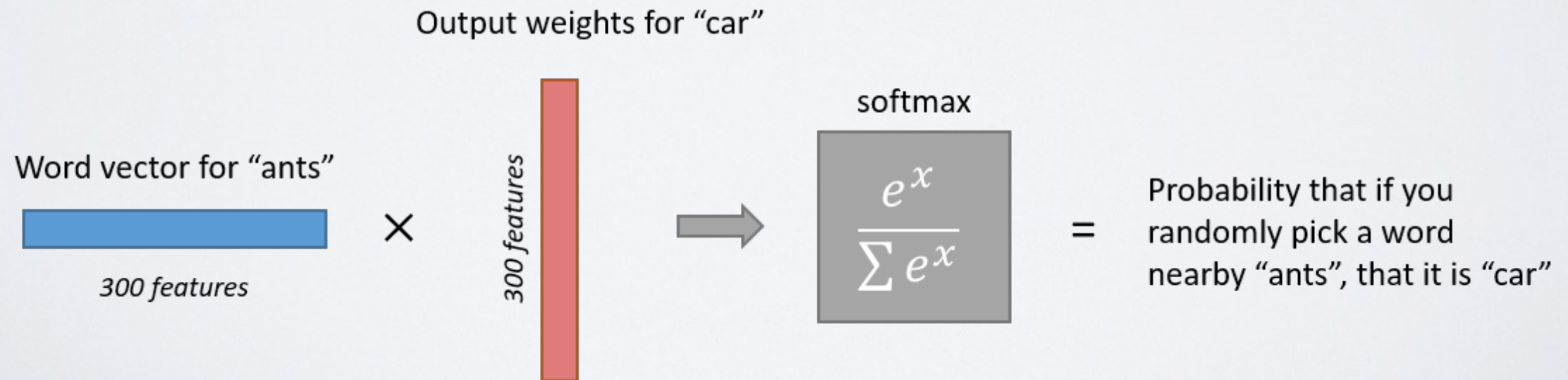
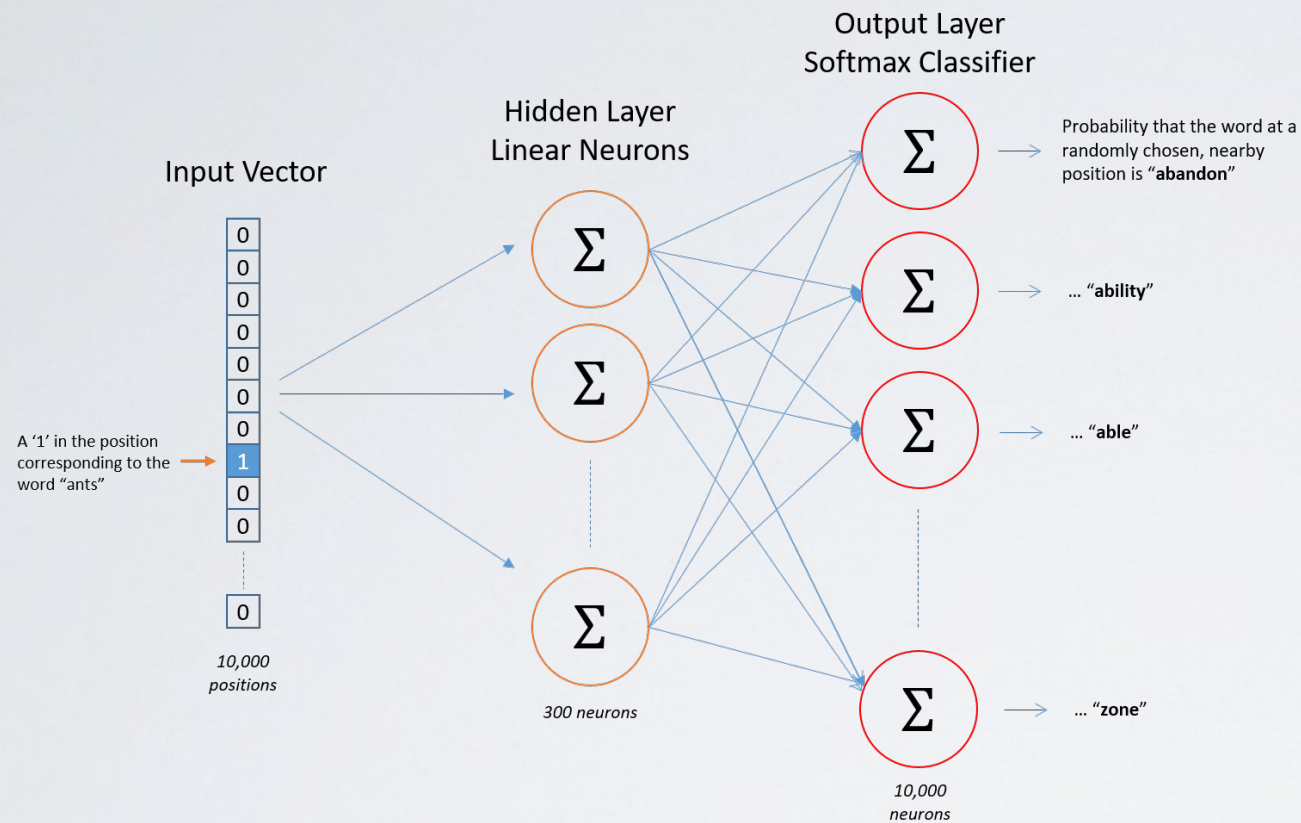
Word embedding

Corpus => Word = vectors

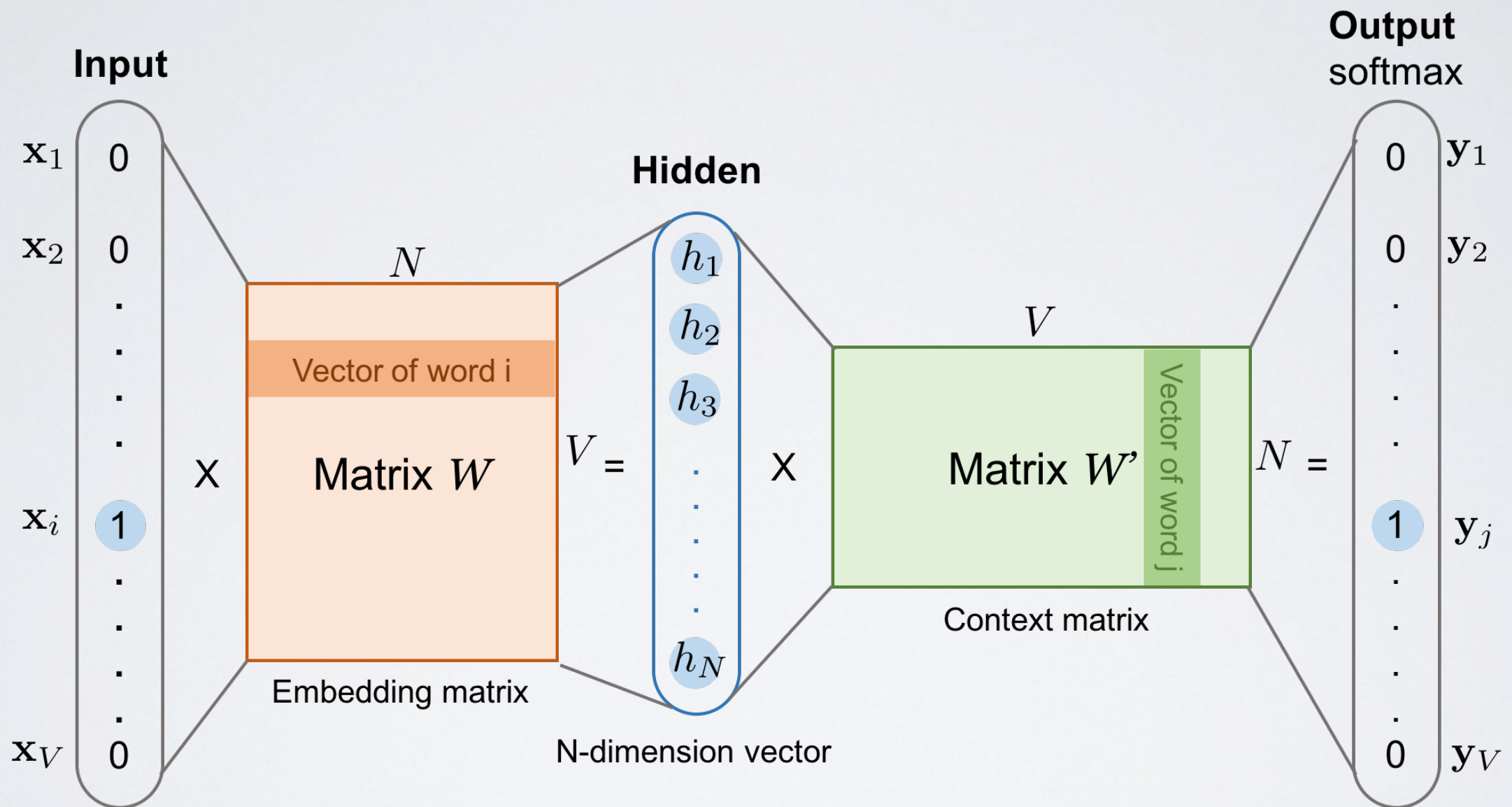
Similar embedding = similar **context**

Source Text	Training Samples							
<table border="1"><tr><td>The</td><td>quick</td><td>brown</td><td>fox jumps over the lazy dog.</td></tr></table> →	The	quick	brown	fox jumps over the lazy dog.	(the, quick) (the, brown)			
The	quick	brown	fox jumps over the lazy dog.					
<table border="1"><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td><td>jumps over the lazy dog.</td></tr></table> →	The	quick	brown	fox	jumps over the lazy dog.	(quick, the) (quick, brown) (quick, fox)		
The	quick	brown	fox	jumps over the lazy dog.				
<table border="1"><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td><td>jumps</td><td>over the lazy dog.</td></tr></table> →	The	quick	brown	fox	jumps	over the lazy dog.	(brown, the) (brown, quick) (brown, fox) (brown, jumps)	
The	quick	brown	fox	jumps	over the lazy dog.			
<table border="1"><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td><td>jumps</td><td>over</td><td>the lazy dog.</td></tr></table> →	The	quick	brown	fox	jumps	over	the lazy dog.	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
The	quick	brown	fox	jumps	over	the lazy dog.		

SKIPGRAM



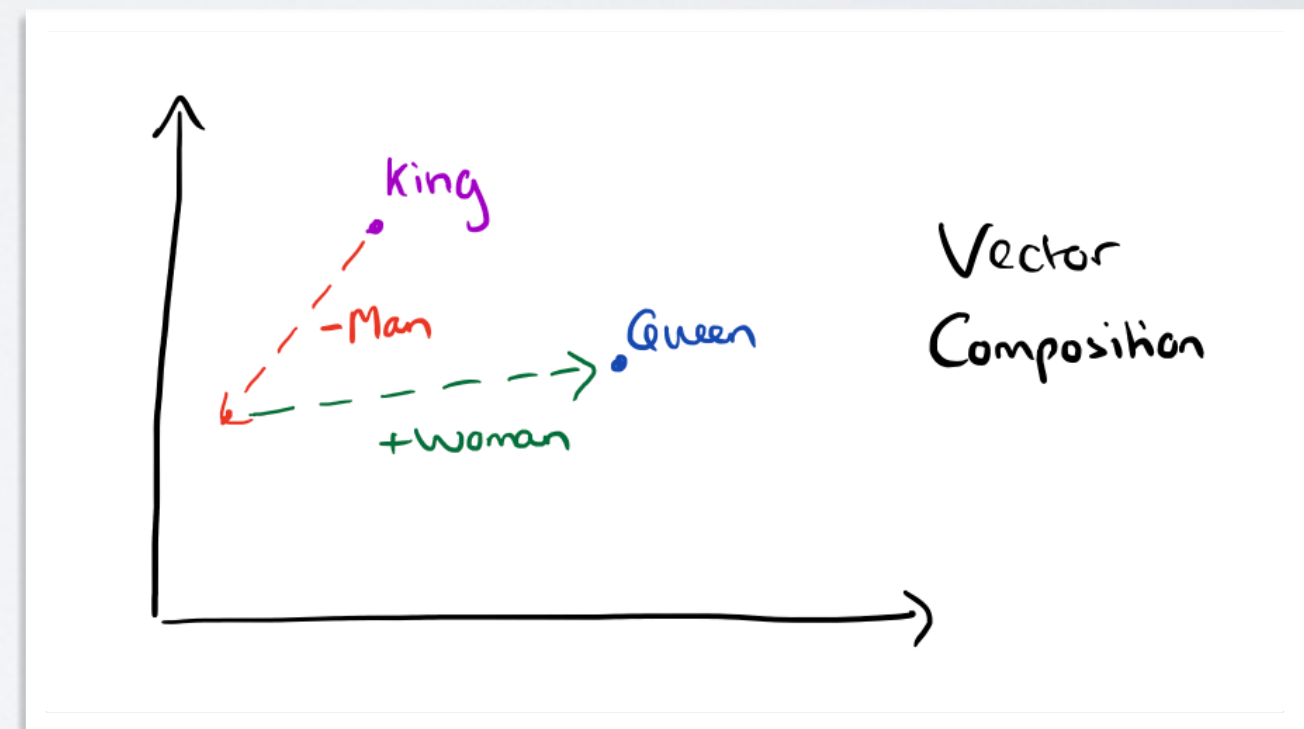
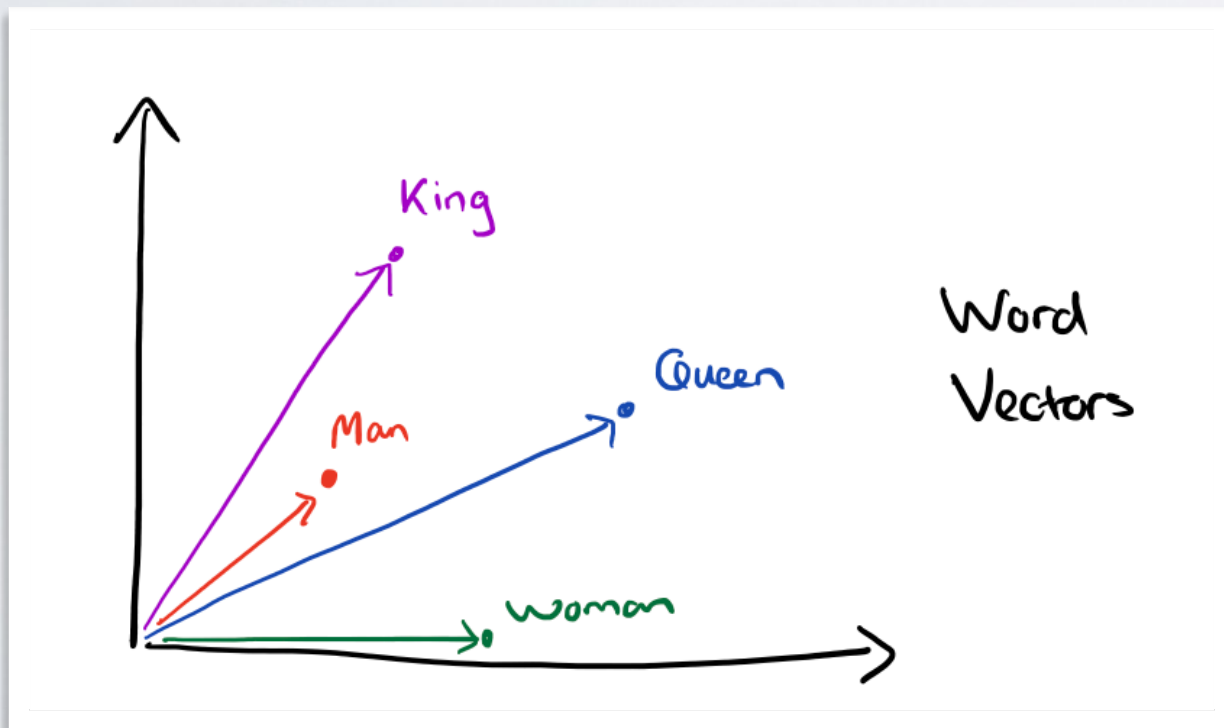
SKIPGRAM



\mathbf{N} =embedding size. \mathbf{V} =vocabulary size

SKIPGRAM

	King	Queen	Woman	Princess	...
Royalty	0.99	0.99	0.02	0.98	
Masculinity	0.99	0.05	0.01	0.02	
Femininity	0.05	0.93	0.999	0.94	
Age	0.7	0.6	0.5	0.1	
...	⋮				



[<https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/>]

SKIPGRAM

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[<https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/>]

PRE-TRAINED

- You can train word2vec on your own dataset, but it needs to be large enough (and its costly)
 - <https://radimrehurek.com/gensim/models/word2vec.html>
- You can use pre-trained embeddings, trained on enormous corpus (Twitter, Wikipedia...)
 - e.g., Glove: <https://nlp.stanford.edu/projects/glove/>

USAGE

- Single words=> Use directly vectors
- Short texts=> Weighted average vectors (more weights to more important words, e.g., rare words: TF-IDF...)
- Long texts=> More tricky. Need other approaches (Doc2vec, RNN)

USAGE

- Parameters:
 - ▶ Embedding dimensions d
 - ▶ Context size

GRAPH EMBEDDING

GENERIC “SKIPGRAM”

- Algorithm that takes an input:
 - The element to embed
 - A list of “context” elements
- Provide as output:
 - An embedding with interesting properties
 - Works well for machine learning
 - Similar elements are close in the embedding
 - Somewhat preserves the overall structure

DEEPWALK

- Skipgram for graphs:
 - 1) Generate “sentences” using random walks
 - 2) Apply Skipgram
- Parameters:
 - Same as Skipgram
 - Embedding dimensions d
 - Context size
 - Parameters for “sentence” generation: length of random walks, number of walks starting from each node, etc.

NODE2VEC

- Use biased random walk to tune the context to capture *what we want*
 - ▶ “Breadth first” like RW => local neighborhood (edge probability ?)
 - ▶ “Depth-first” like RW => global structure ? (Communities ?)
 - ▶ 2 parameters to tune:
 - **p**: bias towards revisiting the previous node
 - **q**: bias towards exploring undiscovered parts of the network

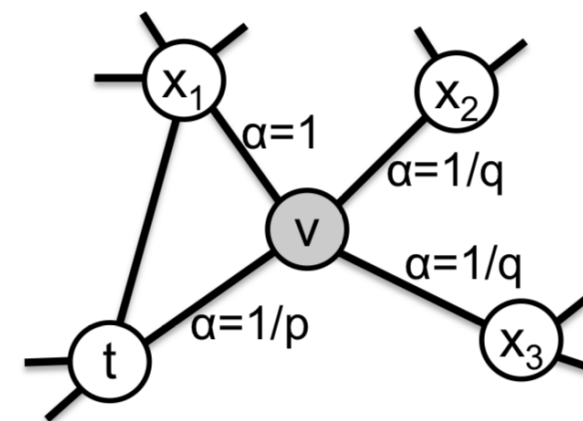
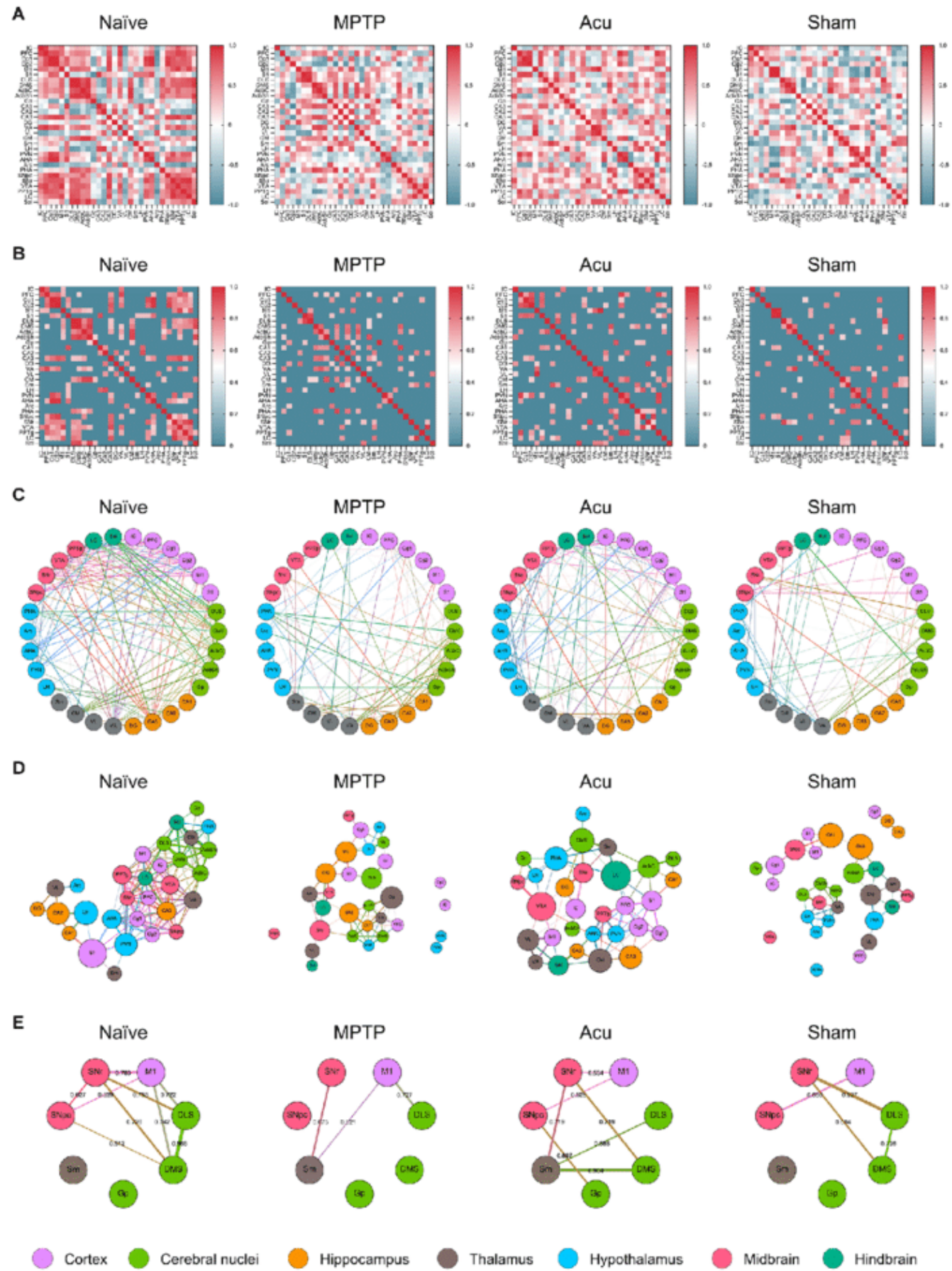


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v . Edge labels indicate search biases α .

OBJECTS/VECTORS
TO
GRAPHS

GRAPH \leftrightarrow VECTORS

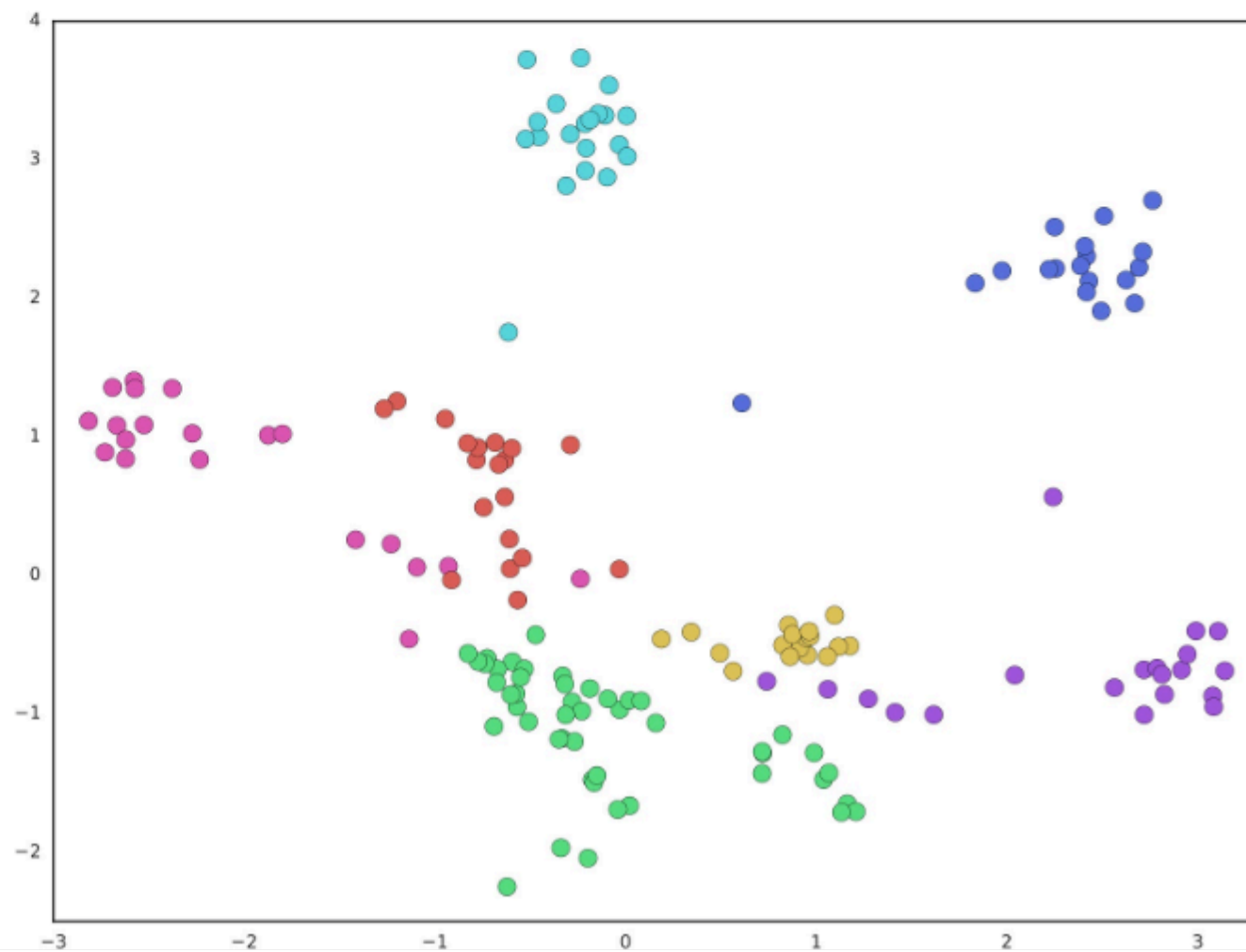
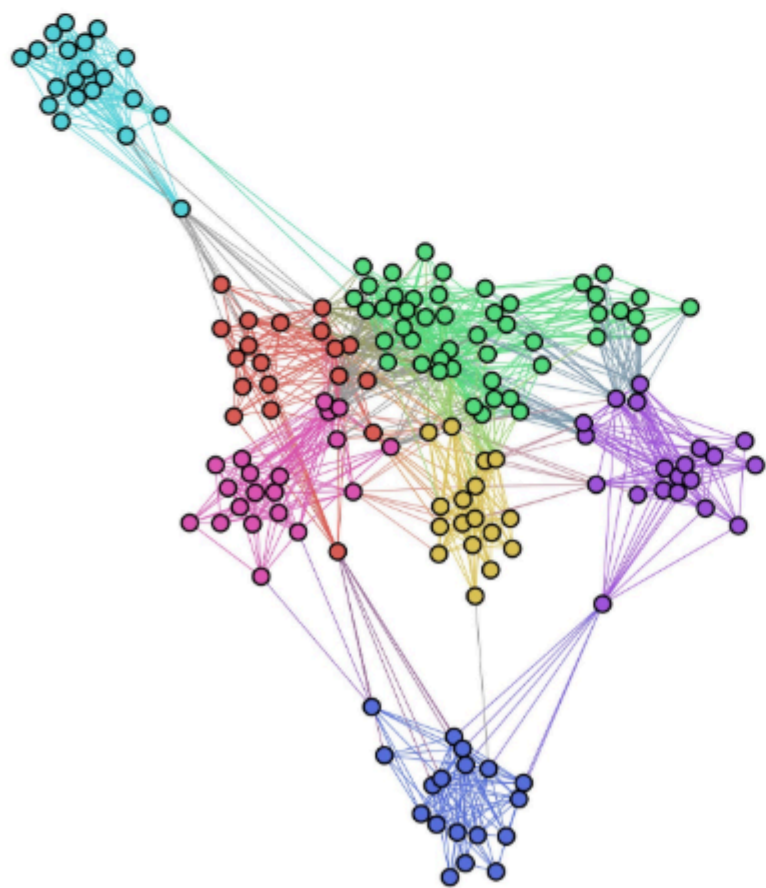
- Graph Embedding: Graph \rightarrow Vectors
- What about Vectors \rightarrow Graphs
 - Simple approach: Correlation matrix
 - \Rightarrow Represent the relations between features in a dataset
 - 1) Compute the correlation between all variables (spearman/Pearson)
 - 2) Keep only correlations above a threshold
 - 3) Correlation values can be represented as weights



ITEM-ITEM GRAPH

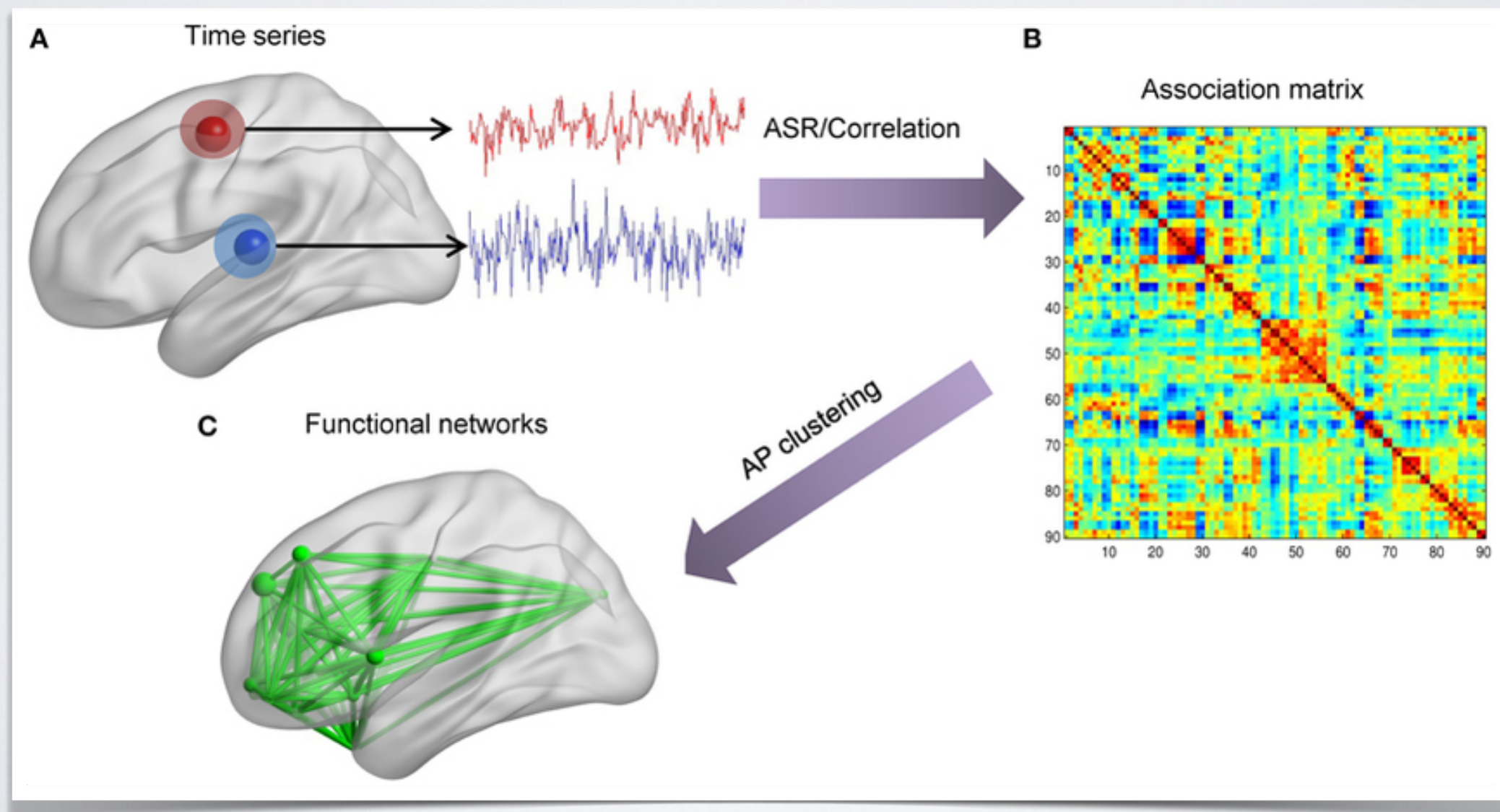
- We can use graphs as an alternative to dimensionality reduction for visualization
 - PCA / tSNE: project items in 2D, close items are similar
 - Some impossibilities, e.g., multiple semantics for words (“palm”: part of the hand, tree)
 - Networks can also be viewed in 2D and preserve the similarity information
- Approach:
 - 1) Compute the distance between elements
 - Euclidean
 - Cosine (in recommendation settings for instance)
 - 2) Keep as an edge values above a threshold

ITEM-ITEM GRAPH



ITEM-ITEM GRAPH

- Typical application case: Brain signal analysis
 - Distance is computed as signal correlation on fMRI, i.e., regional brain activity
 - => Time series to graph

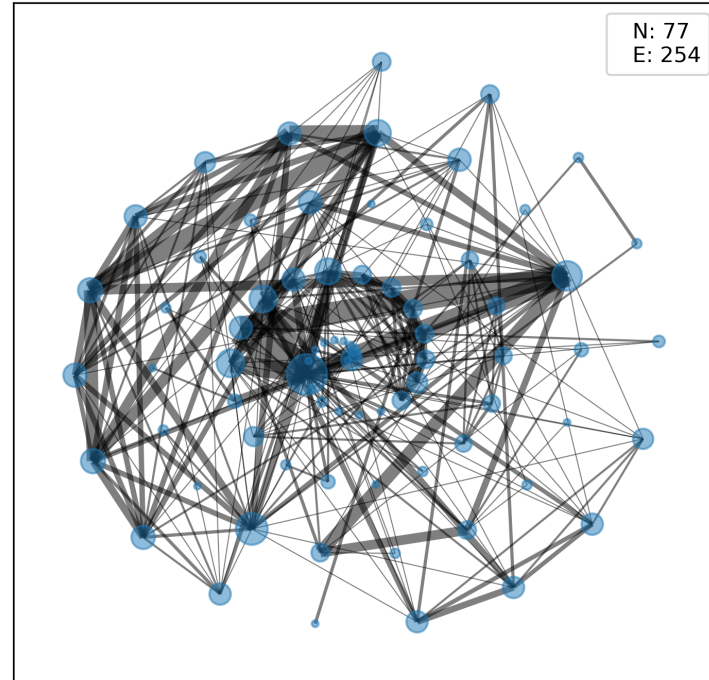


BACKBONE EXTRACTION

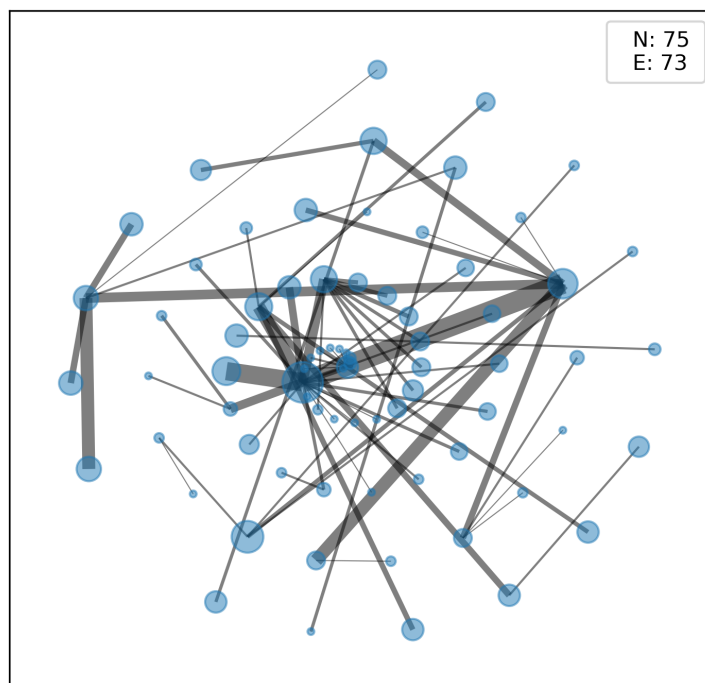
- In some cases, the network created might be too dense to be analyzed properly
 - Too low threshold: everything is connected
 - Too high: disconnected graph, most elements removed
- A solution is to use Backbone extraction
 - Methods that retain only the most important edges, based on different principles
 - e.g., <https://gitlab.liris.cnrs.fr/coregraphie/netbone>

BACKBONE EXTRACTION

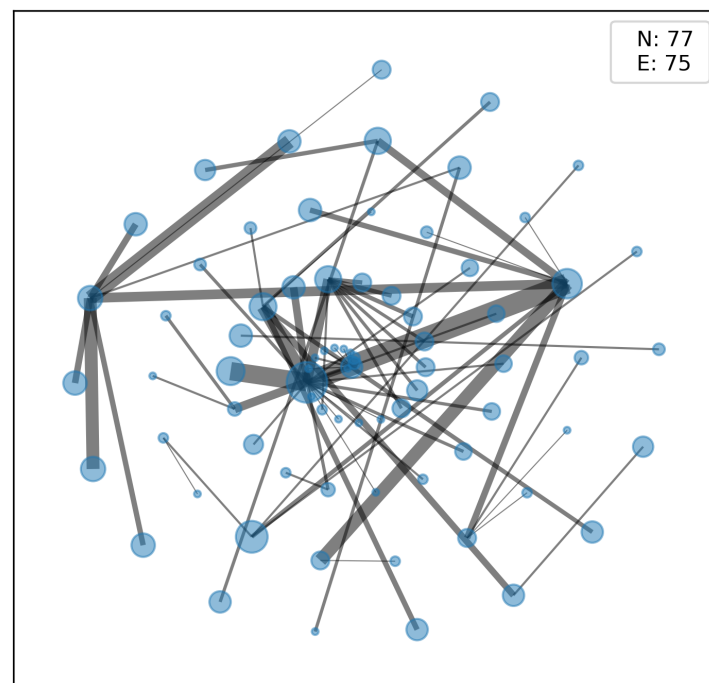
Les Misérables Original Network



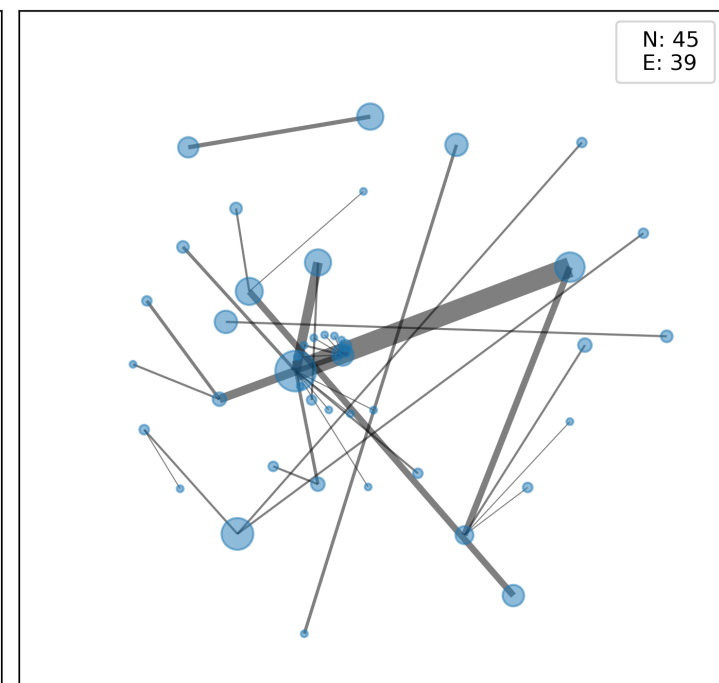
Boolean Filter



Threshold Filter



Fraction Filter



PROJET

CRÉER UN “DASHBOARD”

Avec des visualizations sur mesure

DASH

- Librairie python pour créer des dashboards d'analyse de données
 - Il existe des outils ne nécessitant pas de code (Power BI, Tableau, Google data Studio, OBIEE...)
- Dash permet potentiellement de faire des choses beaucoup plus personnalisées
 - Mais requiert de programmer :)

OBJECTIF

- Créer un Dashboard de données réelles
 - Choisissez un jeu de données simple.
 - Faites une visualisation sous forme de dashboard de quelques aspects
- Prochain cours: tutoriel dash
 - Faire un graphe
 - Une carte avec des points
 - Une distribution
 - Un clustering