Web Mining and Recommender Systems

Dimensionality Reduction

Learning Goals

In this section we want to:

- Introduce **dimensionality reduction**
- Explore different interpretations of low-dimensional structures
- Discuss the relationship between supervised and unsupervised learning

This section

How can we build **low dimensional** representations of **high dimensional** data?

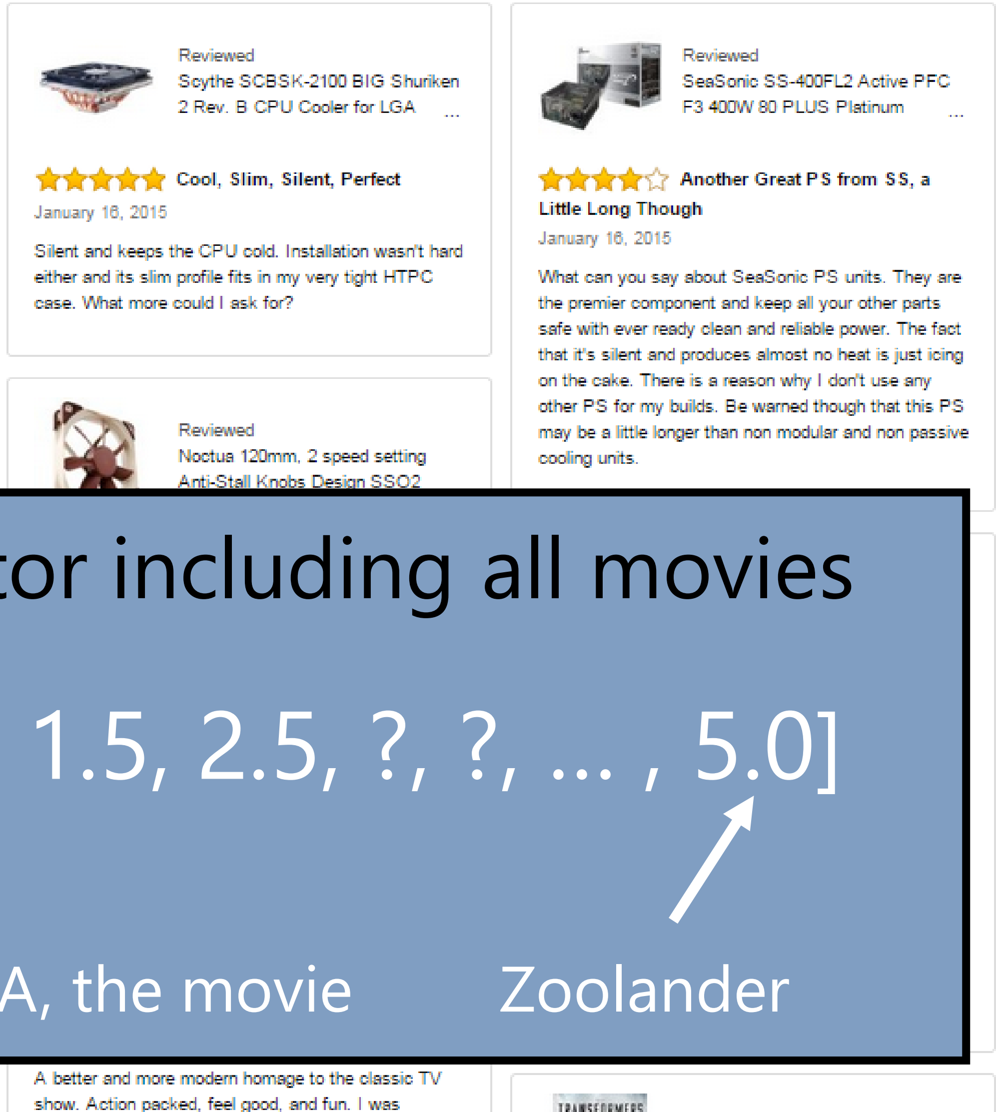
e.g. how might we (compactly!) represent

1. The ratings I gave to every movie I’ve watched?
2. The complete text of a document?
3. The set of my connections in a social network?

Dimensionality reduction

Q1: The ratings I gave to every movie I've watched

(or product I've purchased)



A1: A (sparse) vector including all movies

F_julian = [0.5, ?, 1.5, 2.5, ?, ?, ... , 5.0]

A-team — ABBA, the movie — Zoolander

Dimensionality reduction

A1: A (sparse) vector including all movies

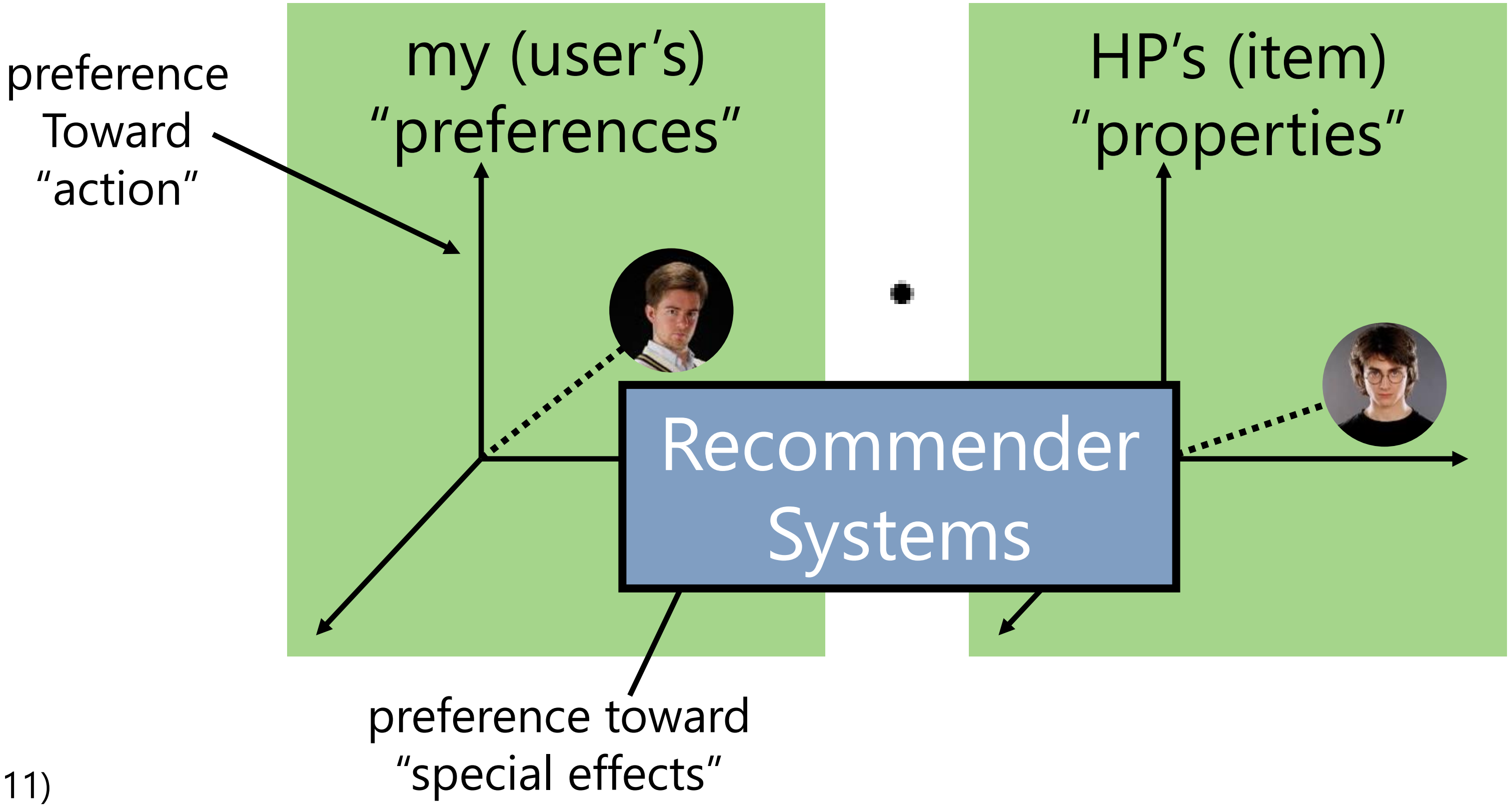
F_julian = [0.5, ?, 1.5, 2.5, ?, ?, … , 5.0]

Incredibly high-dimensional

- Costly to store and manipulate
- Not clear how to add new dimensions
- Missing data
- Many dimensions are associated with obscure products
- Not clear how to use this representation for prediction

Dimensionality reduction

A2: Describe my preferences using a **low-dimensional** vector



e.g. Koren & Bell (2011)

Dimensionality reduction

Q2: How to represent the complete text of a document?

The Peculiar Genius of Bjork

CULTURE | BY EMILY WITT | JANUARY 23, 2015 11:30 AM

A1: A (sparse) vector counting all **words**

F_text = [150, 0, 0, 0, 0, 0, … , 0]

a aardvark zoetrope

concepts and feelings; and Bjork the producer and curator, who seeks out

Dimensionality reduction

A1: A (sparse) vector counting all **words**

F_text = [150, 0, 0, 0, 0, 0, … , 0]

Incredibly high-dimensional…

- Costly to store and manipulate
- Many dimensions encode essentially the same thing
- Many dimensions devoted to the “long tail” of obscure words (technical terminology, proper nouns etc.)

Dimensionality reduction

A2: A low-dimensional vector describing the **topics in the document**

87 of 102 people found the following review helpful

☆☆☆☆☆ **You keep what you kill**, December 27, 2004

By Schtinky "Schtinky" (Washington State) - See all my reviews

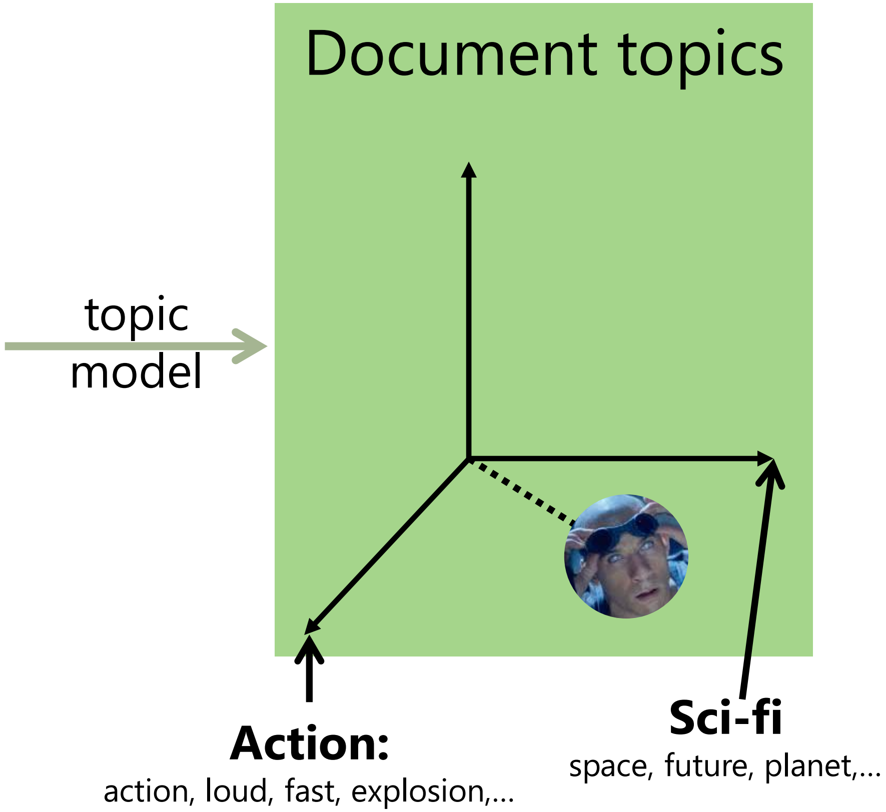
VINE™ VOICE

This review is from: The Chronicles of Riddick (Widescreen Unrated Director's Cut) (DVD)

Even if I have to apologize to my Friends and Favorites, and my family, I have to admit that I really liked this movie. It's a Sci-Fi movie with a "Mad Maxx" appeal that, while changing many things, left Riddick from `Pitch Black' to be just Riddick. They did not change his attitude or soften him up or bring him out of his original character, which was very pleasing to `Pitch Black' fans like myself.

First off, let me say that when playing the DVD, the first selection to come up is Convert or Fight, and no explanation of the choices. This confused me at first, so I will mention off the bat that they are simply different menu formats, that each menu has the very same options, simply different background visuals. Select either one and continue with the movie.

(review of "The Chronicles of Riddick")



Dimensionality reduction

Q3: How to represent connections in a social network?

A1: An adjacency matrix!

$$A = \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}$$

Dimensionality reduction

A1: An adjacency matrix

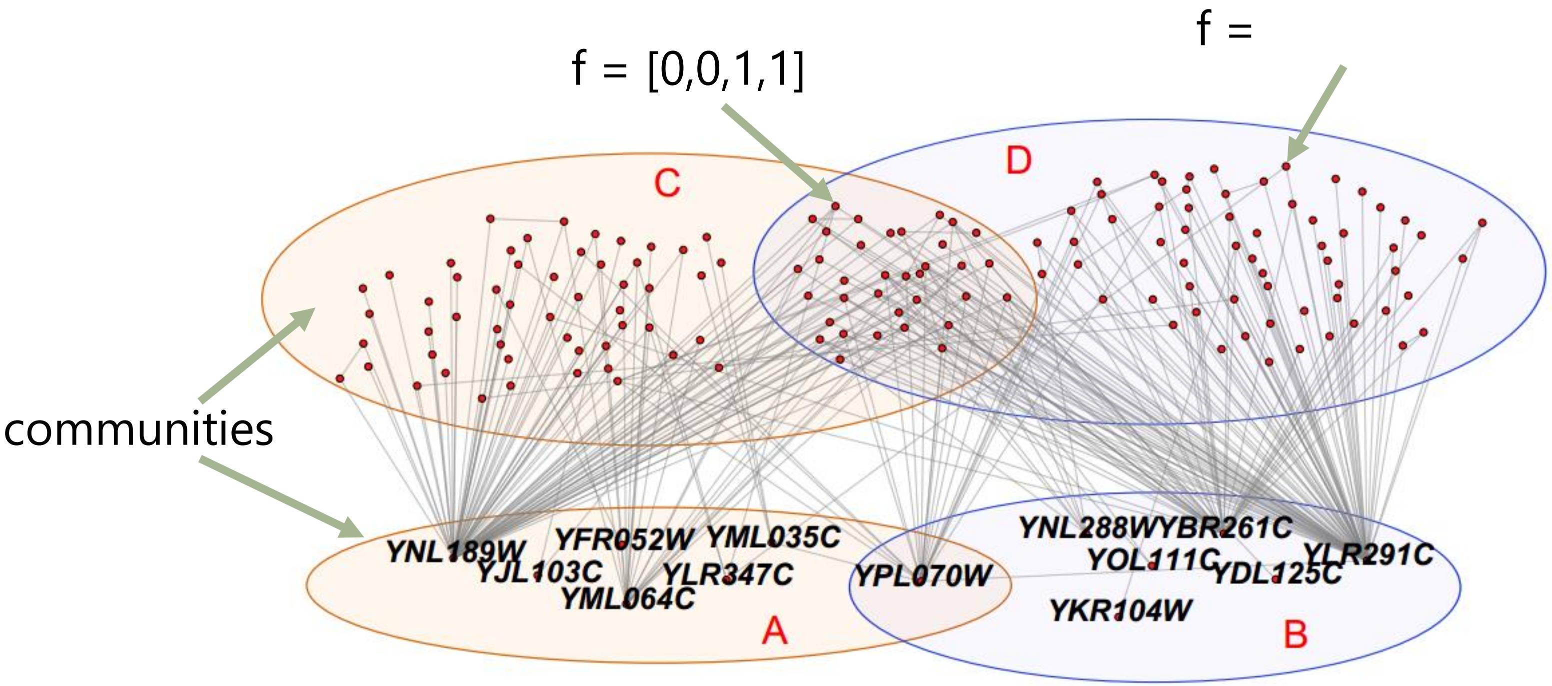
$$A = \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}$$

Seems almost reasonable, but…

- Becomes very large for real-world networks
- Very fine-grained – doesn't straightforwardly encode which nodes are similar to each other

Dimensionality reduction

A2: Represent each node/user in terms of the **communities** they belong to



e.g. from a PPI network; Yang, McAuley, & Leskovec (2014)

Why dimensionality reduction?

Goal: take **high-dimensional** data, and describe it compactly using a small number of dimensions

Assumption: Data lies (approximately) on some l**ow-dimensional** manifold

(a few dimensions of opinions, a small number of topics, or a small number of communities)

Why dimensionality reduction?

Unsupervised learning

- Today our goal is not to solve some specific predictive task, but rather to **understand** the important features of a dataset
- We are not trying to understand the process which generated labels from the data, but rather the process which generated the data itself

Why dimensionality reduction?

Unsupervised learning

- **But!** The models we learn will prove useful when it comes to solving predictive tasks later on, e.g.

- **Q1:** If we want to predict which users like which movies, we need to understand the important dimensions of opinions
- **Q2:** To estimate the category of a news article (sports, politics, etc.), we need to understand topics it discusses
- **Q3:** To predict who will be friends (or enemies), we need to understand the communities that people belong to

Coming up…

Dimensionality reduction, clustering, and community detection

- Principal Component Analysis
- K-means clustering
- Hierarchical clustering
- Later: Community detection
  - Graph cuts
  - Clique percolation
  - Network modularity

Web Mining and Recommender Systems

Principal Component Analysis

Learning Goals

- Present Principal Components Analysis

Principal Component Analysis

Principal Component Analysis (PCA) is one of the oldest (1901!) techniques to understand which dimensions of a high-dimensional dataset are "important"

Why?

- To **select** a few important features
- To **compress** the data by ignoring components which aren't meaningful

Principal Component Analysis

Motivating example:
Suppose we rate restaurants in terms of:
[value, service, quality, ambience, overall]

- Which dimensions are highly correlated (and how)?
- Which dimensions could we "throw away" without losing much information?
- How can we find which dimensions can be thrown away automatically?
- In other words, how could we come up with a "compressed representation" of a person's 5-d opinion into (say) 2-d?

Principal Component Analysis

Suppose our data/signal is an MxN matrix

N = number of observations

$$X = \begin{pmatrix} 5 & 3 & \cdots & 1 \\ 4 & 2 & & 1 \\ 3 & 1 & & 3 \\ 2 & 2 & & 4 \\ 1 & 5 & & 2 \\ \vdots & & \ddots & \vdots \\ 1 & 2 & \cdots & 1 \end{pmatrix}$$

M = number of features
(each **column** is a data point)

Principal Component Analysis

We'd like (somehow) to recover this signal using as few dimensions as possible

signal

$$X \in \mathbb{R}^{M \times N}$$

compressed signal (K < M)

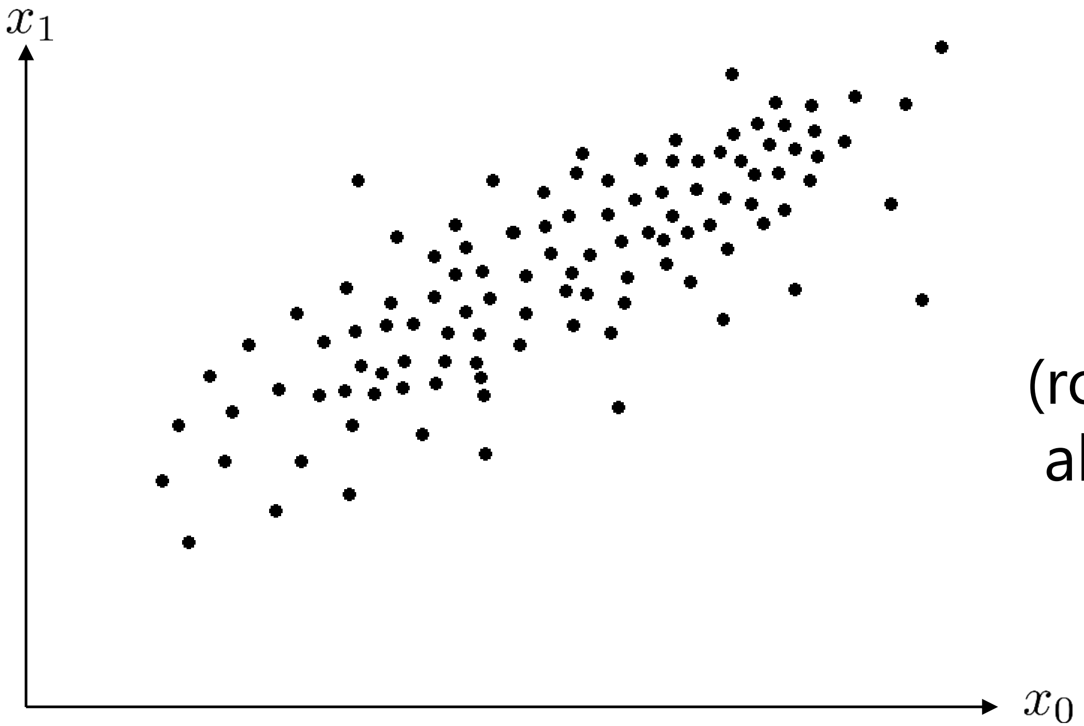
$$Y' \in \mathbb{R}^{K \times N}$$

$$f(Y') \simeq X$$

(approximate) process to recover signal from its compressed version

Principal Component Analysis

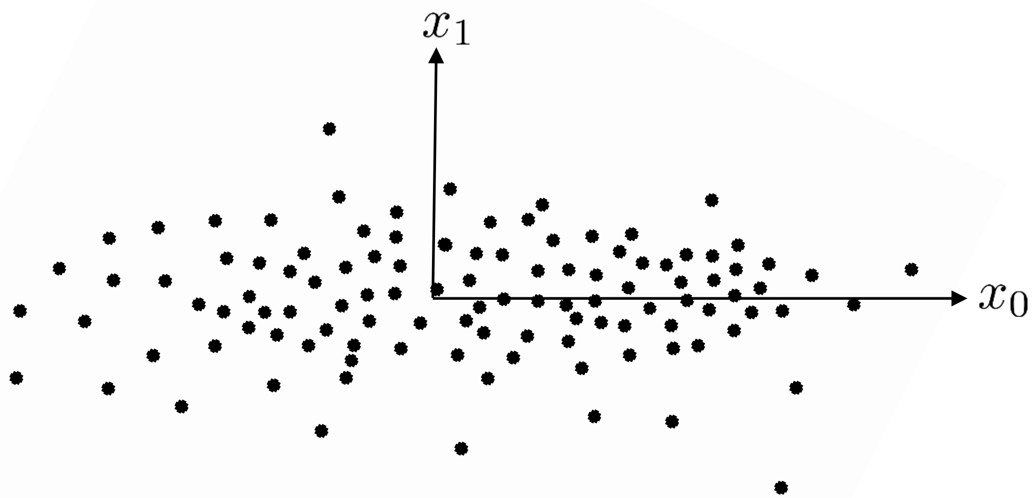
E.g. suppose we have the following data:



The data (roughly) lies along a line

Idea: if we know the position of the point on the line (1D), we can approximately recover the original (2D) signal

But how to find the important dimensions?



Find a new basis for the data (i.e., rotate it) such that

- **most** of the variance is along x0,
- most of the “leftover” variance (not explained by x0) is along x1,
- most of the leftover variance (not explained by x0,x1) is along x2,
- etc.

Principal Component Analysis

But how to find the important dimensions?

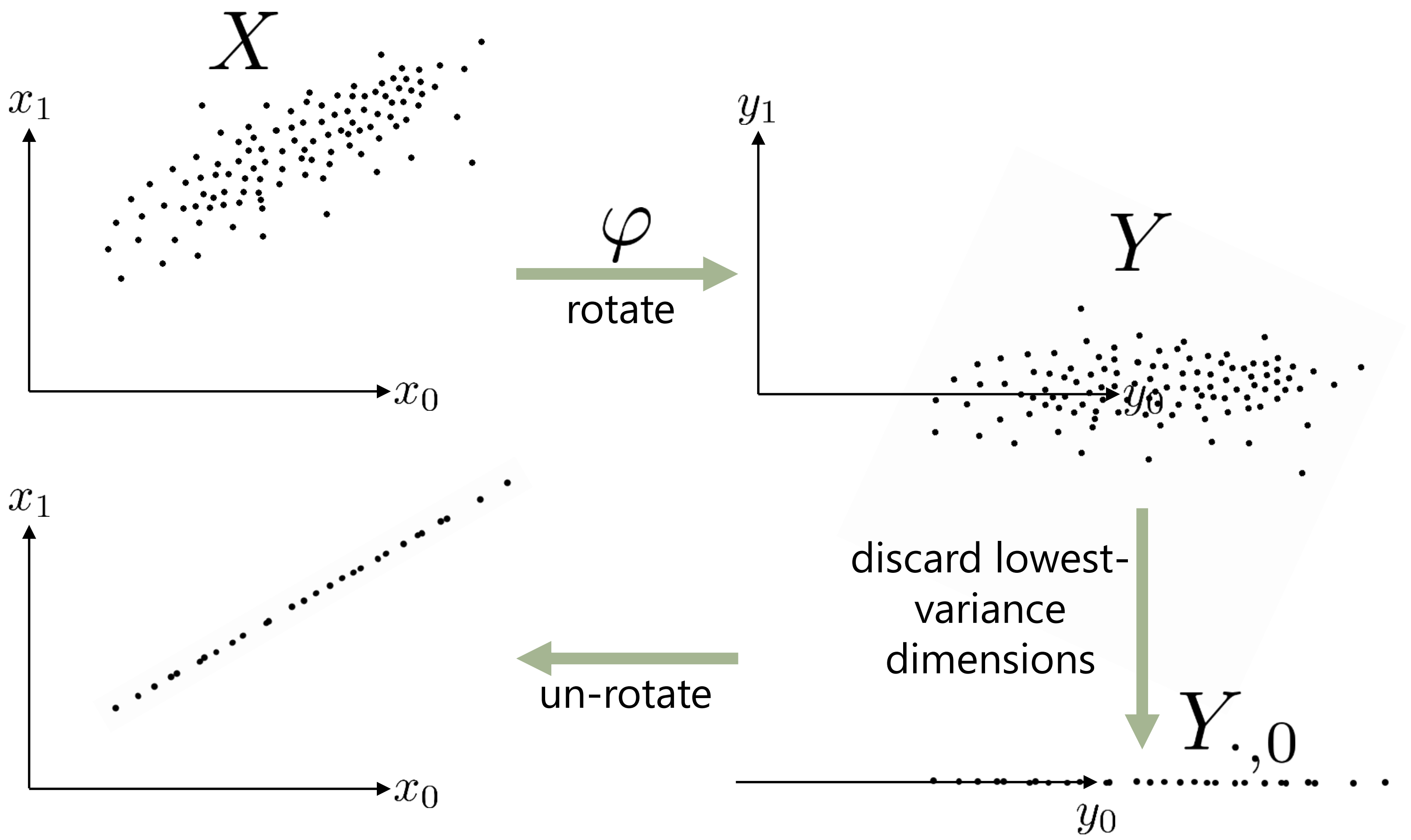
- Given an input $X \in \mathbb{R}^{M \times N}$
- Find a basis $\varphi \in \mathbb{R}^{M \times M}$

Principal Component Analysis

But how to find the important dimensions?

- Given an input $X \in \mathbb{R}^{M \times N}$
- Find a basis $\varphi \in \mathbb{R}^{M \times M}$
- Such that when X is rotated $(Y = \varphi X)$
  - Dimension with highest variance is y_0
  - Dimension with 2nd highest variance is y_1
  - Dimension with 3rd highest variance is y_2
  - Etc.

Principal Component Analysis



Principal Component Analysis

For a single data point: $y = \varphi x \quad x = \varphi^{-1} y = \varphi^T y$

$$x =$$

Principal Component Analysis

$$x = \varphi_1 y_1 + \varphi_2 y_2 + \ldots + \varphi_M y_M = \sum_{j=1}^{M} \varphi_j y_j$$

$$x \simeq$$

Principal Component Analysis

For a single data point: $y = \varphi x \quad x = \varphi^{-1} y = \varphi^T y$

$$x = \varphi_1 y_1 + \varphi_2 y_2 + \ldots + \varphi_M y_M = \sum_{j=1}^{M} \varphi_j y_j$$

$$x \simeq \sum_{j=1}^{K} \varphi_j y_i + \sum_{j=K+1}^{M} \varphi_j b_j$$

Keep K dimensions of *y*

And replace the others by constants

Principal Component Analysis

We want to fit the "best" reconstruction:

$$x = \underbrace{\varphi^T y}_{\text{"complete" reconstruction}} \qquad x_i \simeq \underbrace{\sum_{j=1}^{K} \varphi_j y_j + \sum_{j=K+1}^{M} \varphi_j b_j}_{\text{approximate reconstruction}}$$

i.e., it should minimize the **MSE**:

Principal Component Analysis

$$\min_{\varphi,b} \frac{1}{N} \sum_{y} \left\| \sum_{j=1}^{K} \varphi_j y_j + \sum_{j=K+1}^{M} \varphi_j b_j - \varphi^T y \right\|_2^2$$

Simplify…

Principal Component Analysis

$$\min_{\varphi,b} \frac{1}{N} \sum_{y} \left\| \sum_{j=K+1}^{M} \varphi_j (y_j - b_j) \right\|_2^2$$

Expand…

Principal Component Analysis

This simplifies to:

$$\min_{\varphi,b} \frac{1}{N} \sum_{y} \left\| \sum_{j=K+1}^{M} \varphi_j (y_j - b_j) \right\|_2^2$$

(due to orthonormality of φ) –
expand and convince ourselves

$$\min_{\varphi,b} \frac{1}{N} \sum_{y} \sum_{j=K+1}^{M} (y_j - b_j)^2$$

Principal Component Analysis

$$\min_{\varphi,b} \frac{1}{N} \sum_{y} \sum_{j=K+1}^{M} (y_j - b_j)^2$$

Principal Component Analysis

$$\min_{\varphi} \underbrace{\frac{1}{N} \sum_{y} \sum_{j=K+1}^{M} (y_j - \bar{y}_j)^2}$$

Equal to the **variance** in
the discarded dimensions

Principal Component Analysis

PCA: We want to keep the dimensions with the highest variance, and discard the dimensions with the lowest variance, in some sense to maximize the amount of "randomness" that gets preserved when we compress the data

Principal Component Analysis

$$\min_{\varphi} \frac{1}{N} \sum_{y} \sum_{j=K+1}^{M} (y_j - \bar{y}_j)^2$$

(subject to φ orthonormal)

Expand in terms of X

$$\min_{\varphi} \frac{1}{N} \sum_{j=K+1}^{M} \varphi_j (X - \bar{X})(X - \bar{X})^T \varphi_j^T$$

(subject to φ orthonormal)

Principal Component Analysis

$$\min_{\varphi} \frac{1}{N} \sum_{j=K+1}^{M} \varphi_j \overbrace{(X - \bar{X})(X - \bar{X})^T}^{\mathrm{Cov}(X)} \varphi_j^T$$

(subject to φ orthonormal)

Lagrange multiplier

$$\min_{\varphi} \frac{1}{N} \sum_{j=K+1}^{M} \varphi_j \mathrm{Cov}(X) \varphi_j^T - \lambda_j (\varphi_j \varphi_j^T - 1)$$

Lagrange multipliers:
Bishop appendix E

Principal Component Analysis

Solve:

$$\frac{\partial}{\partial \varphi_j} \sum_{j=K+1}^{M} \varphi_j \mathrm{Cov}(X) \varphi_j^T - \lambda_j (\varphi_j \varphi_j^T - 1) = 0$$

(Cov(X) is symmetric)

$$2(\mathrm{Cov}(X)\varphi_j^T - \lambda_j \varphi_j^T) = 0$$

- This expression can only be satisfied if phi_j and lambda_j are an **eigenvectors/eigenvalues** of the covariance matrix
- So to minimize the original expression we'd discard phi_j's corresponding to the **smallest** eigenvalues

Principal Component Analysis

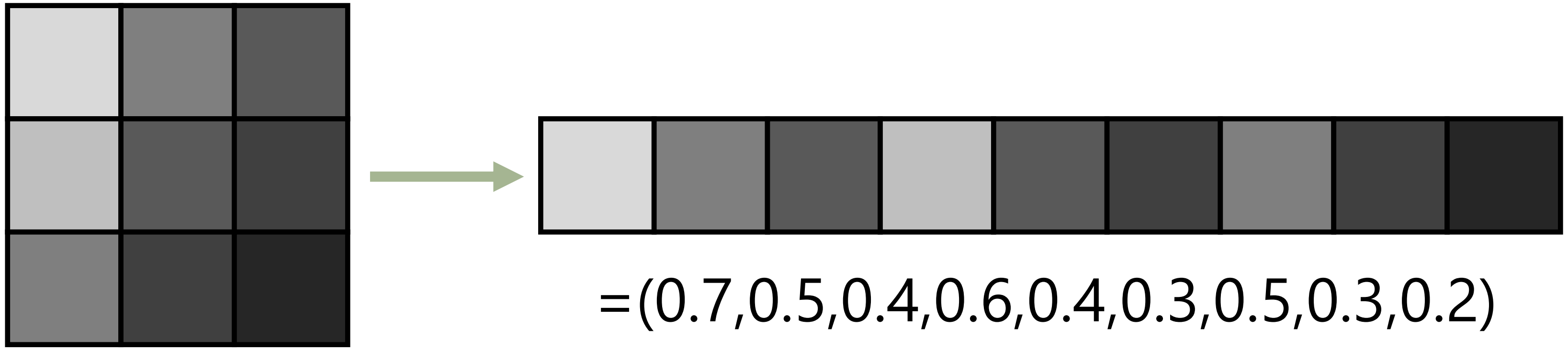
Moral of the story: if we want to optimally (in terms of the MSE) project some data into a low dimensional space, we should choose the projection by taking the **eigenvectors** corresponding to the **largest eigenvalues of the covariance matrix**

Example 1:
What are the principal components of people's opinions on beer?

(code available on course webpage)

Principal Component Analysis

Example 2:
What are the principal dimensions of image patches?

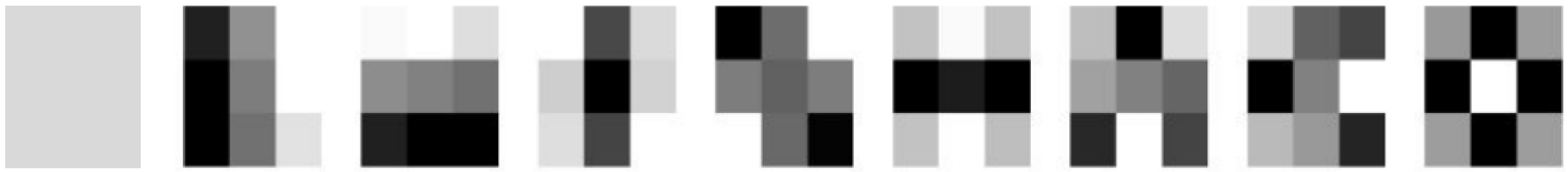


=(0.7,0.5,0.4,0.6,0.4,0.3,0.5,0.3,0.2)

Principal Component Analysis

Construct such vectors from 100,000 patches from real images and run PCA:

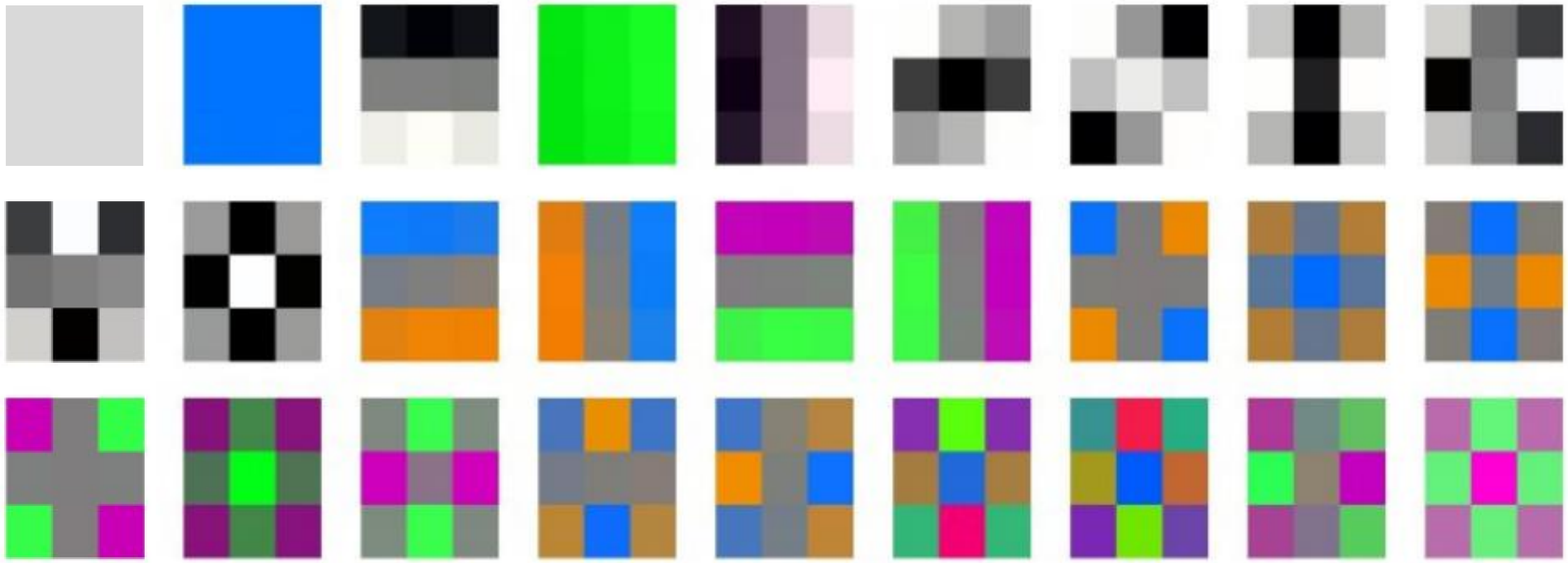
Black and white:



Principal Component Analysis

Construct such vectors from 100,000 patches from real images and run PCA:

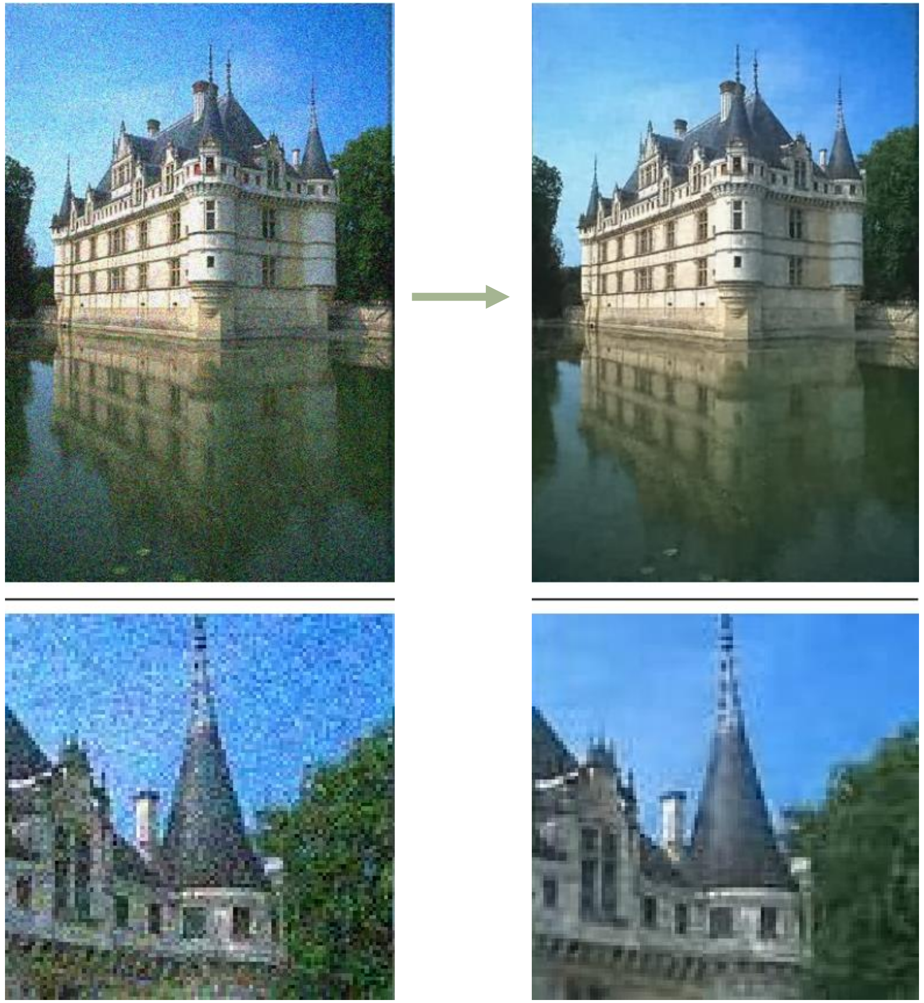
Color:



Principal Component Analysis

From this we can build an algorithm to “denoise” images

Idea: image patches should be **more like** the high-eigenvalue components and **less like** the low-eigenvalue components



McAuley et. al (2006)

Principal Component Analysis

- We want to find a low-dimensional representation that best compresses or “summarizes” our data
- To do this we’d like to keep the dimensions with the highest variance (we proved this), and discard dimensions with lower variance. Essentially, we’d like to capture the aspects of the data that are “hardest” to predict, while discard the parts that are “easy” to predict
- This can be done by taking the eigenvectors of the covariance matrix

Learning Outcomes

- Introduced and derived PCA
- Explained how dimensionality reduction can be cast as **describing patterns of variation** in datasets

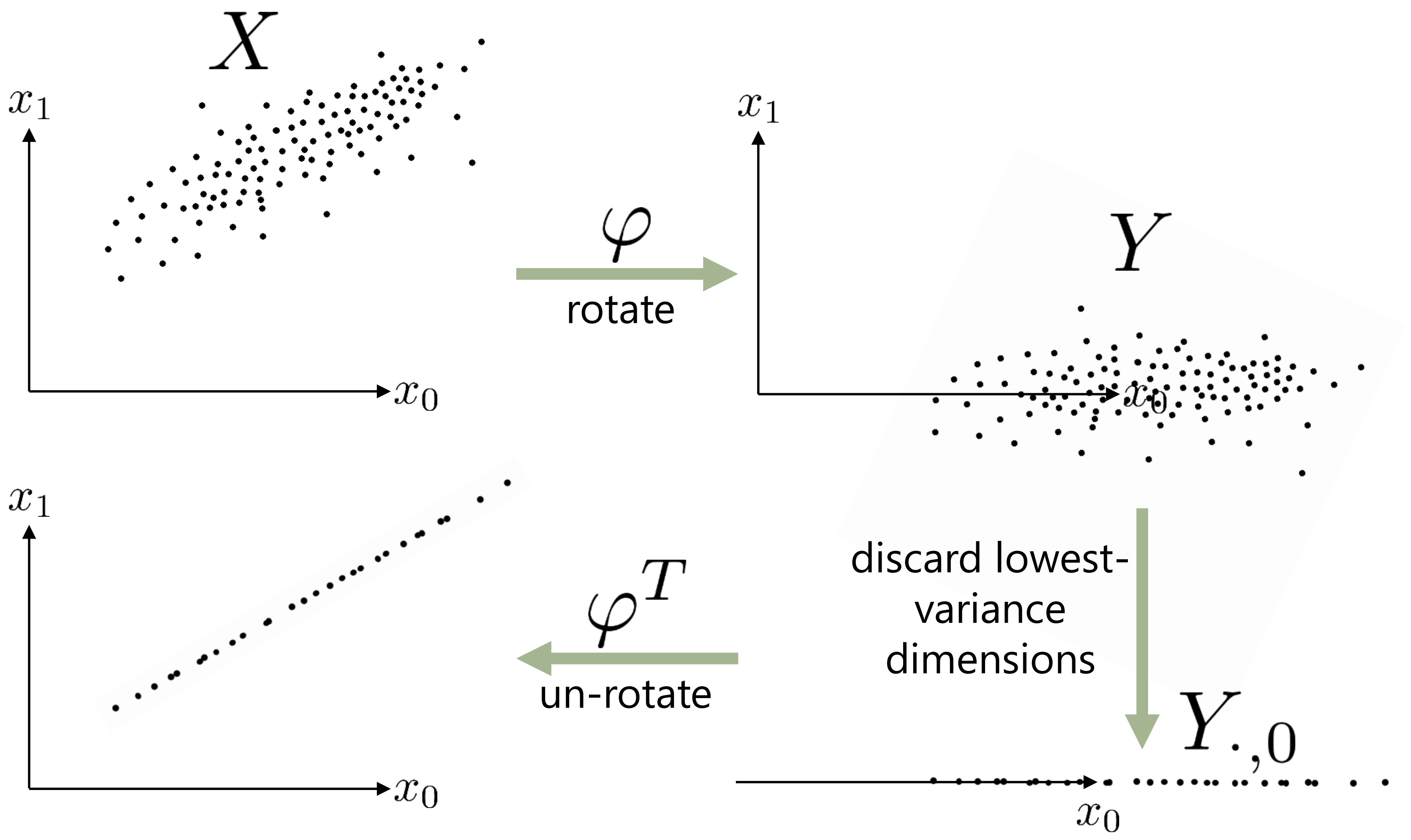
Web Mining and Recommender Systems

Clustering – K-means

Learning Goals

- Introduce the K-means classifier
- Explain how the notion of "low-dimensional" can mean different things for different datasets

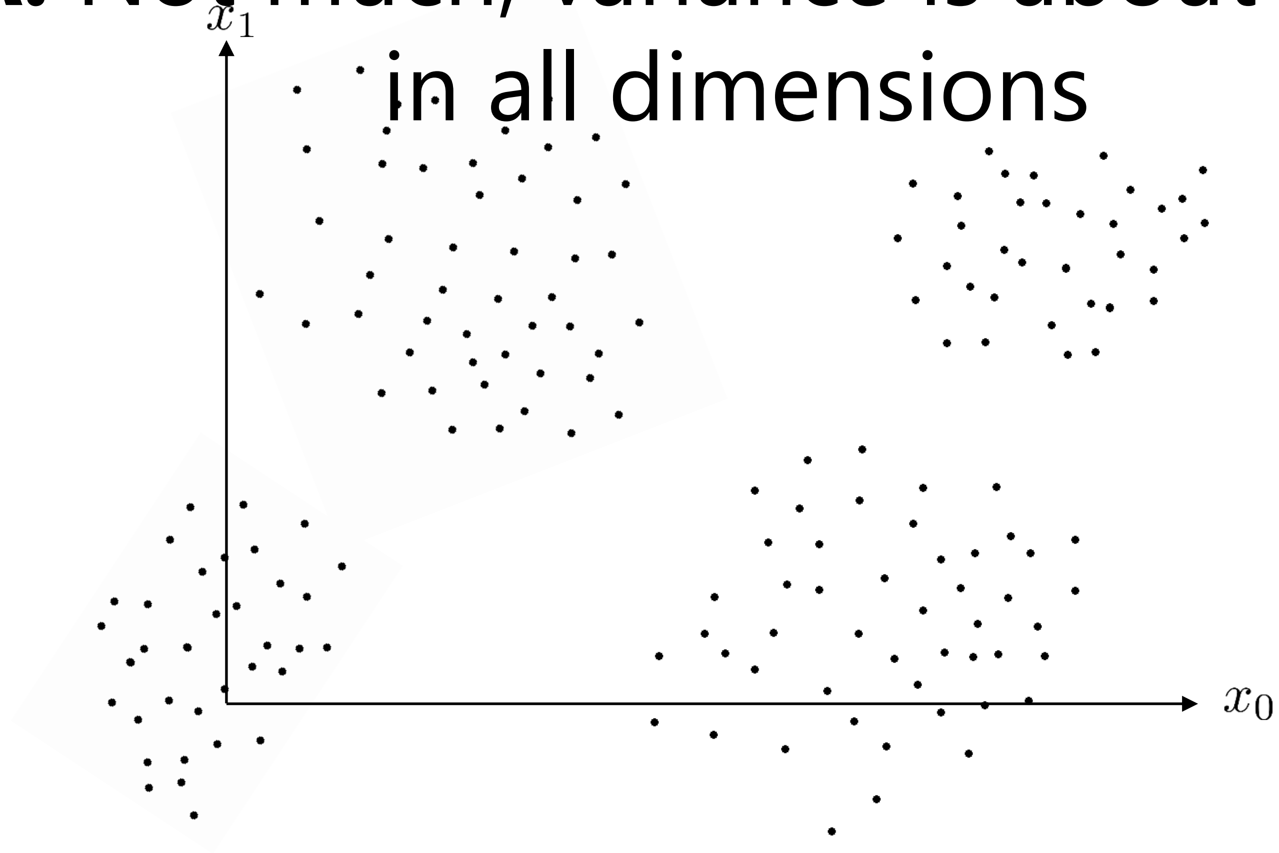
Principal Component Analysis



Clustering

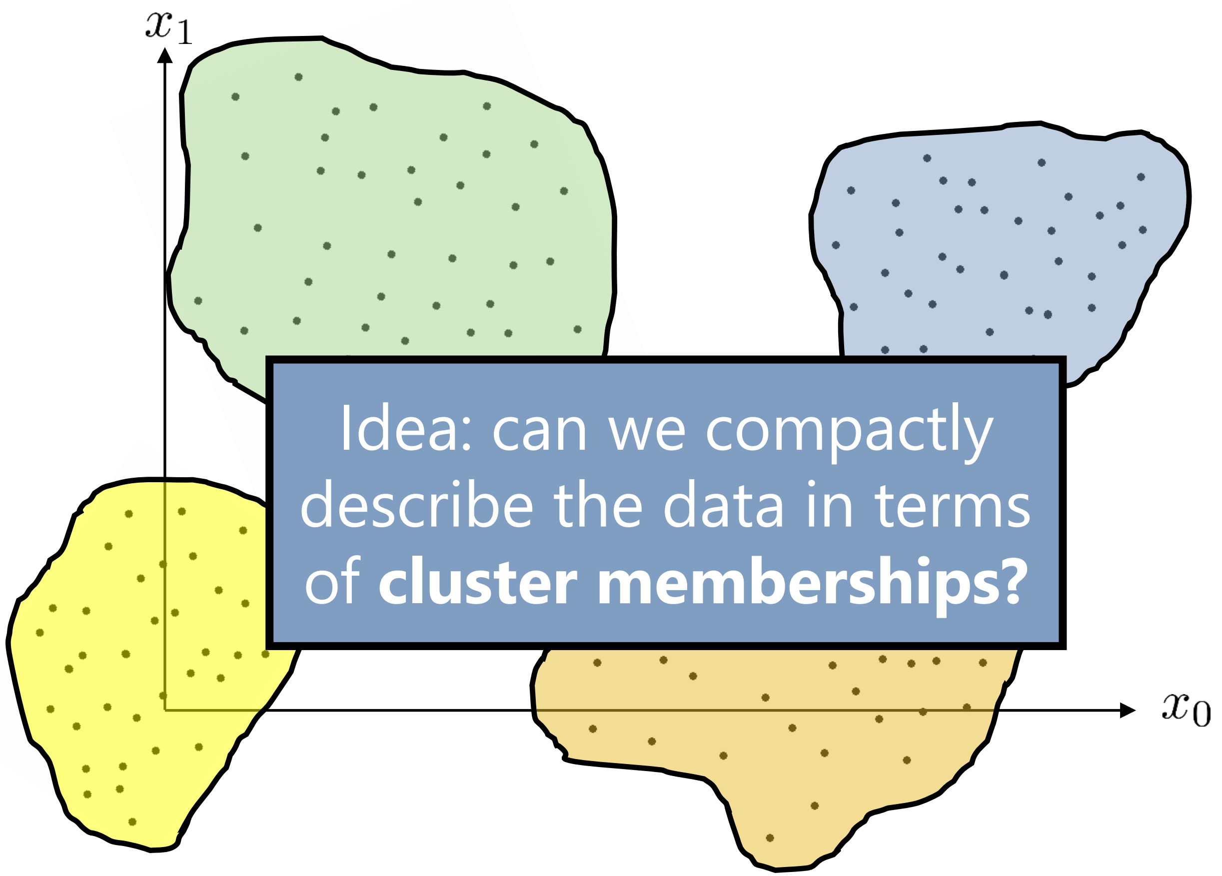
Q: What would PCA do with this data?

A: Not much, variance is about equal in all dimensions



Clustering

But: The data are highly **clustered**



Idea: can we compactly describe the data in terms of **cluster memberships?**

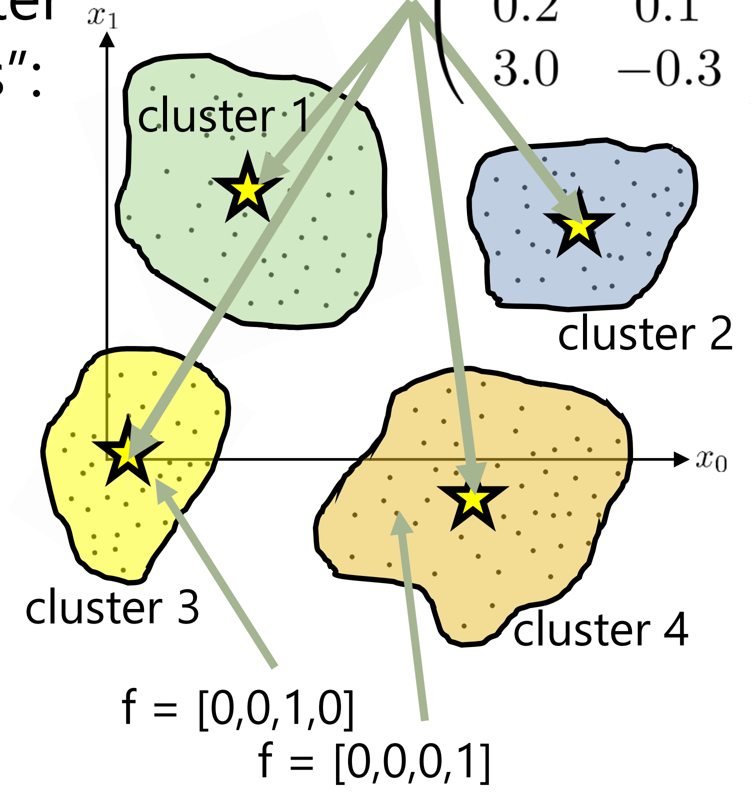
K-means Clustering

1. Input is still a matrix of features:

$$X = \begin{pmatrix} 5 & 3 & \cdots & 1 \\ 4 & 2 & & 1 \\ 3 & 1 & & 3 \\ 2 & 2 & & 4 \\ 1 & 5 & & 2 \\ \vdots & & \ddots & \vdots \\ 1 & 2 & \cdots & 1 \end{pmatrix}$$

2. Output is a list of cluster "centroids":

$$\text{centroids} = \begin{pmatrix} 1.1 & 2.1 \\ 3.5 & 1.8 \\ 0.2 & 0.1 \\ 3.0 & -0.3 \end{pmatrix}$$



3. From this we can describe each point in X by its cluster membership:

$$Y = (1, 2, 4, 3, 4, 2, 4, 2, 2, 3, 3, 2, 1, 1, 3, \ldots, 2)$$

K-means Clustering

Given **features** (X) our goal is to choose K **centroids** (C) and **cluster assignments** (Y) so that the reconstruction error is minimized

Number of data points

$$X \in \mathbb{R}^{N \times M}$$

Feature dimensionality

Number of clusters

$$C \in \mathbb{R}^{K \times M}$$

$$Y \in \{1 \ldots K\}^N$$

$$\text{reconstruction error} = \sum_i \|X_i - C_{y_i}\|_2^2$$

(= sum of squared distances from assigned centroids)

K-means Clustering

Q: Can we solve this optimally?

$$\min_{C,y} \sum_i \|X_i - C_{y_i}\|_2^2$$

A: No. This is (in general) an **NP-Hard** optimization problem

See *"NP-hardness of Euclidean sum-of-squares clustering"*, Aloise et. Al (2009)

K-means Clustering

Greedy algorithm:

```
1. Initialize C (e.g. at random)
2. Do
3.       Assign each X_i to its nearest centroid
4.       Update each centroid to be the mean
         of points assigned to it
5. While (assignments change between iterations)
```

$$y_i = \arg\min_k \|X_i - C_k\|_2^2$$

$$C_k = \frac{\sum_i \delta(y_i{=}k) X_i}{\sum_i \delta(y_i{=}k)}$$

(also: reinitialize clusters at random should they become empty)

Learning Outcomes

- Introduced the K-means classifier
- Gave a greedy solution for the K-means algorithm

K-means Clustering

Further reading:

- K-**medians**: Replaces the mean with the meadian. Has the effect of minimizing the 1-norm (rather than the 2-norm) distance
- **Soft** K-means: Replaces "hard" memberships to each cluster by a proportional membership to each cluster

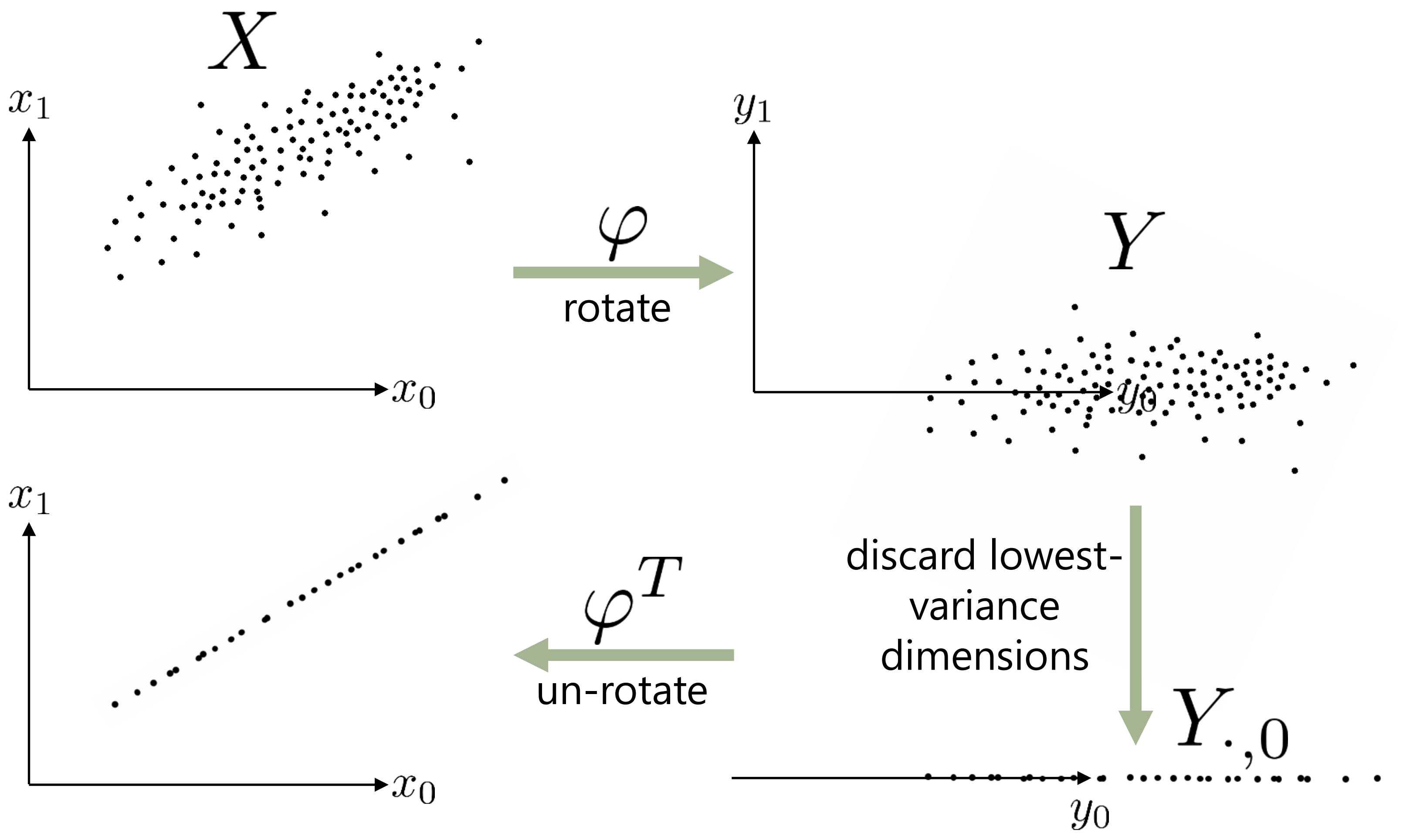
Web Mining and Recommender Systems

Clustering – Hierarchical Clustering

Learning Goals

- Introduce hierarchical clustering

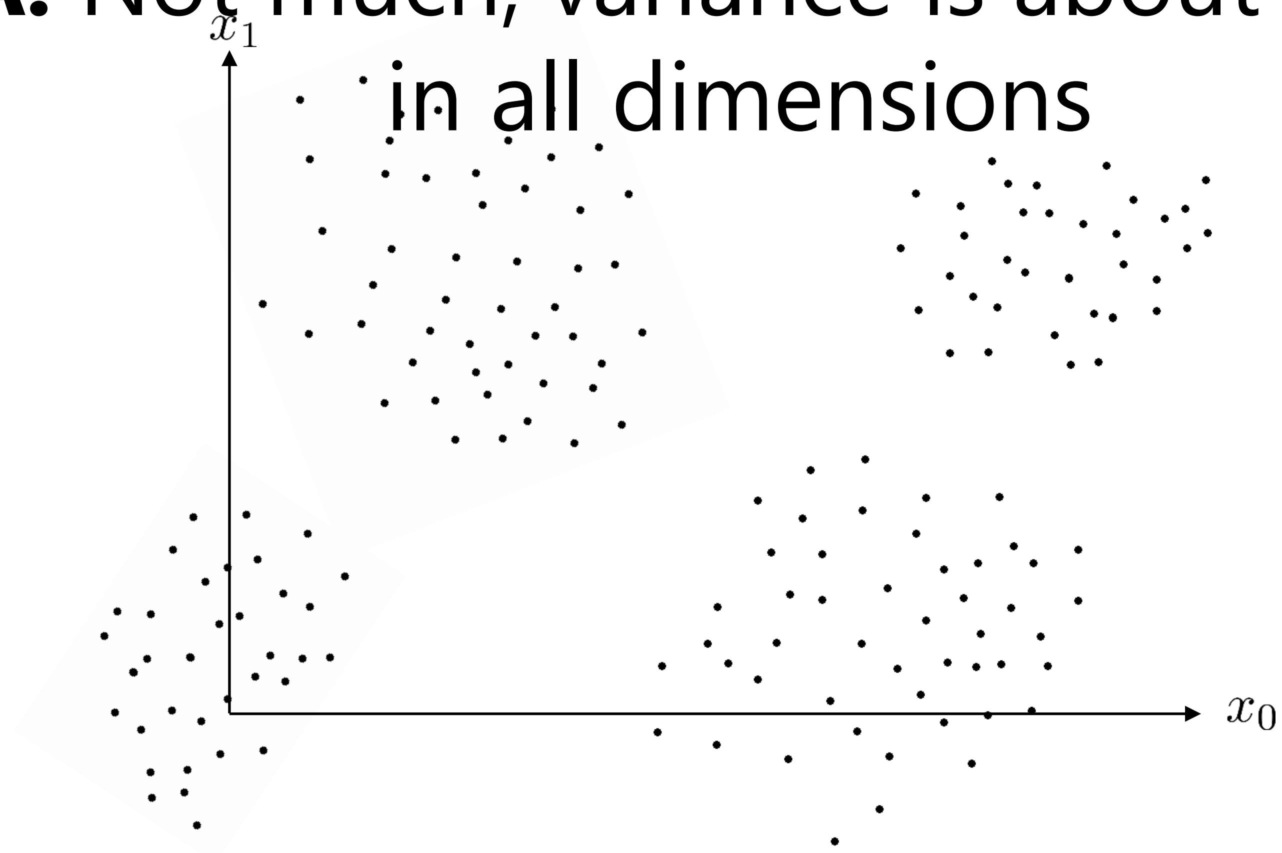
Principal Component Analysis



Principal Component Analysis

Q: What would PCA do with this data?

A: Not much, variance is about equal in all dimensions



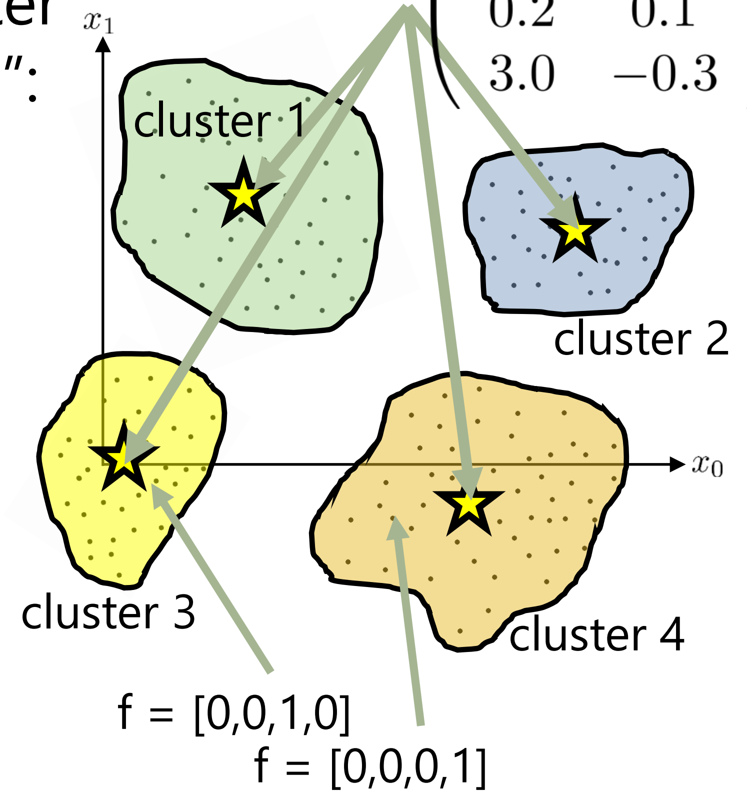
K-means Clustering

1. Input is still a matrix of features:

$$X = \begin{pmatrix} 5 & 3 & \cdots & 1 \\ 4 & 2 & & 1 \\ 3 & 1 & & 3 \\ 2 & 2 & & 4 \\ 1 & 5 & & 2 \\ \vdots & & \ddots & \vdots \\ 1 & 2 & \cdots & 1 \end{pmatrix}$$

2. Output is a list of cluster "centroids":

$$\text{centroids} = \begin{pmatrix} 1.1 & 2.1 \\ 3.5 & 1.8 \\ 0.2 & 0.1 \\ 3.0 & -0.3 \end{pmatrix}$$

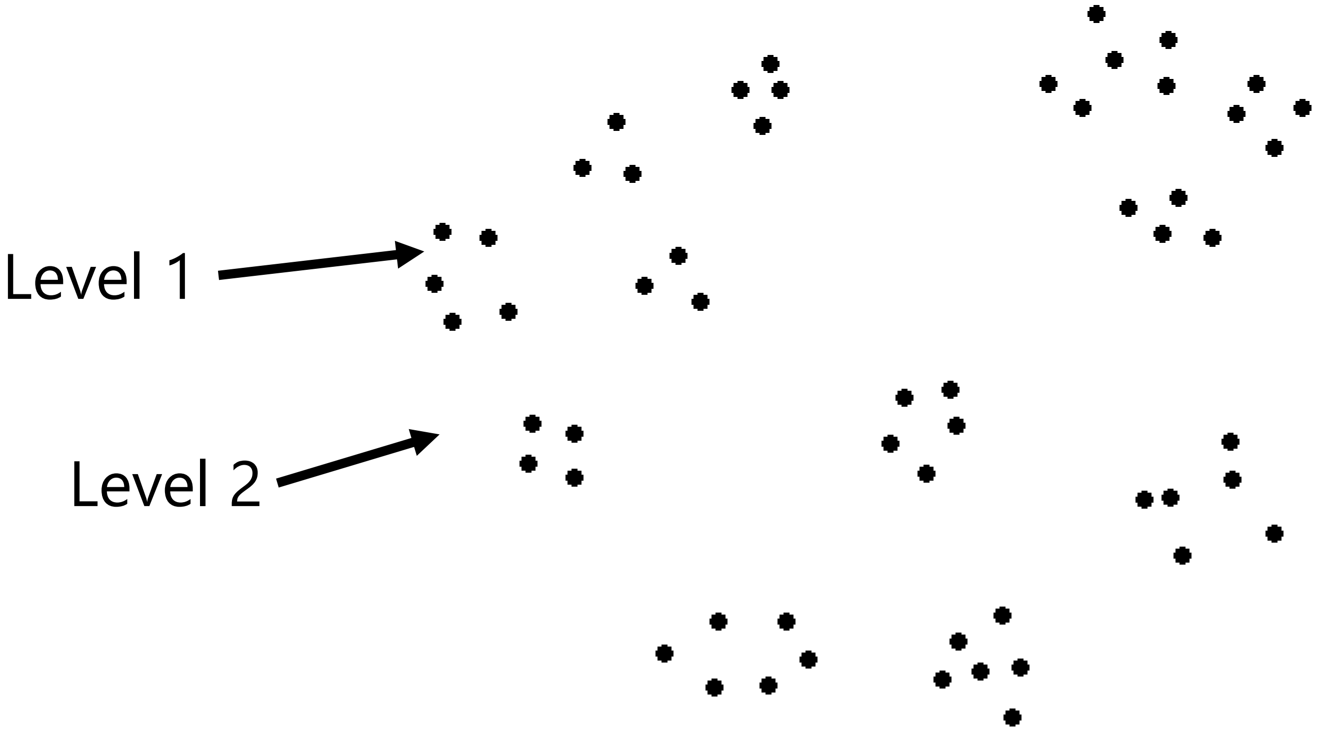


3. From this we can describe each point in X by its cluster membership:

$$Y = (1, 2, 4, 3, 4, 2, 4, 2, 2, 3, 3, 2, 1, 1, 3, \ldots, 2)$$

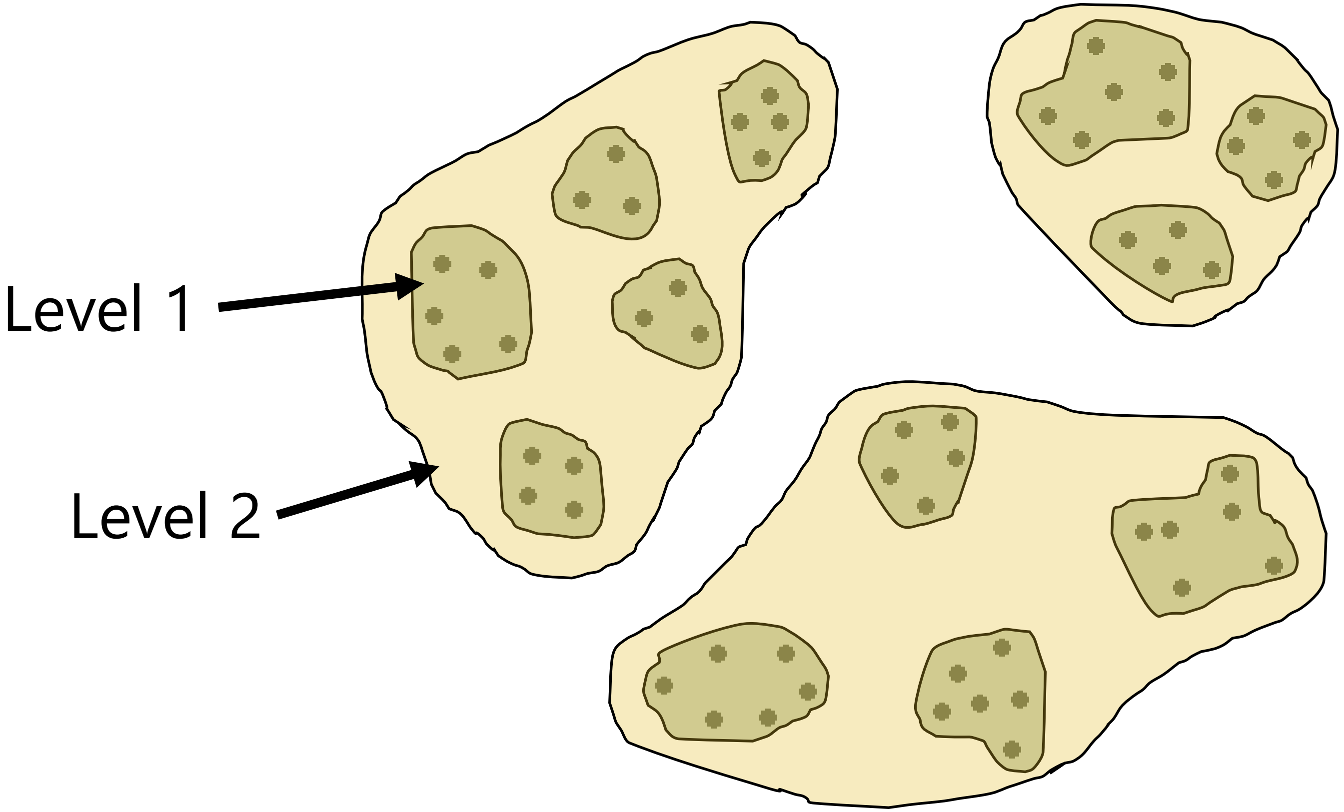
Hierarchical clustering

Q: What if our clusters are **hierarchical?**

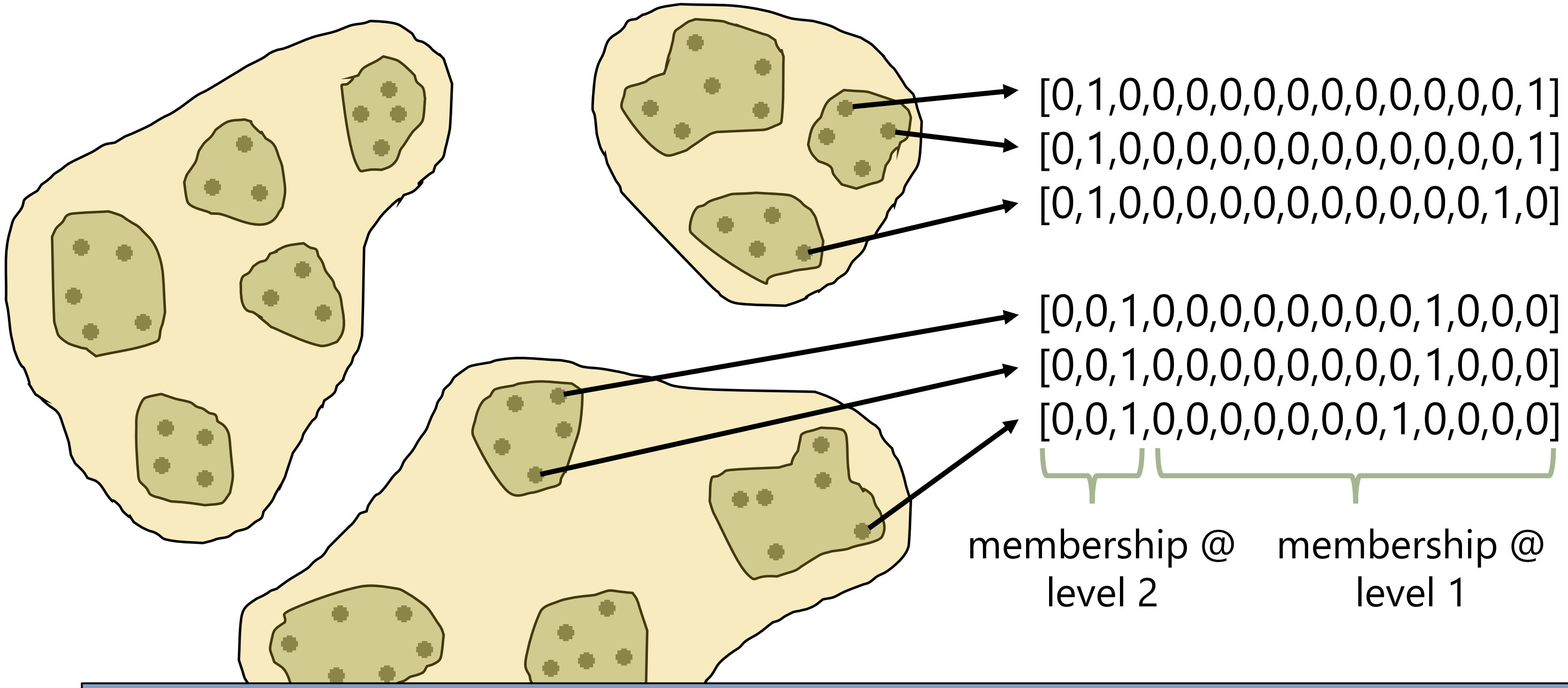


Hierarchical clustering

Q: What if our clusters are **hierarchical?**



Q: What if our clusters are **hierarchical?**



A: We'd like a representation that encodes that points have **some features** in common but not others

Hierarchical clustering

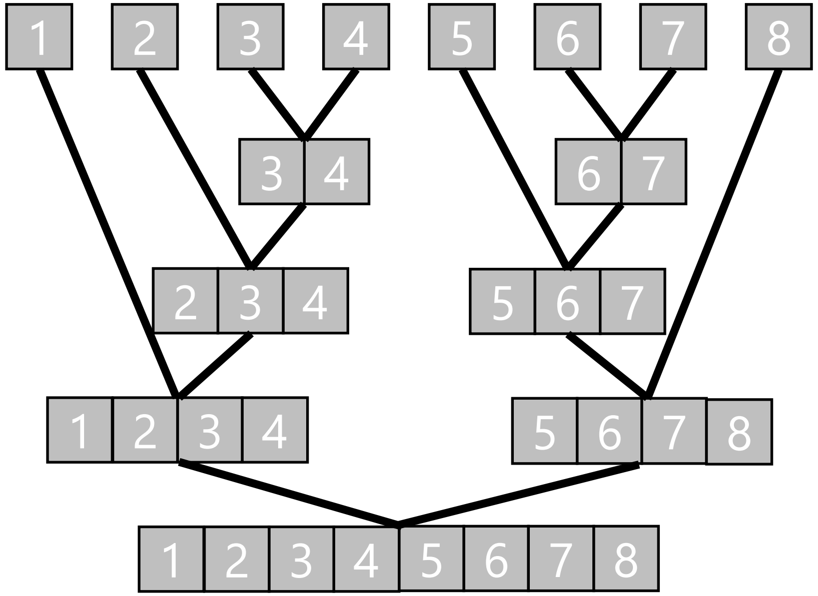
Hierarchical (agglomerative) clustering works by gradually fusing clusters whose points are closest together

```
Assign every point to its own cluster:
Clusters = [[1],[2],[3],[4],[5],[6],…,[N]]
While len(Clusters) > 1:
      Compute the center of each cluster
      Combine the two clusters with the nearest centers
```

Example

Hierarchical clustering

If we keep track of the order in which clusters were merged, we can build a "hierarchy" of clusters



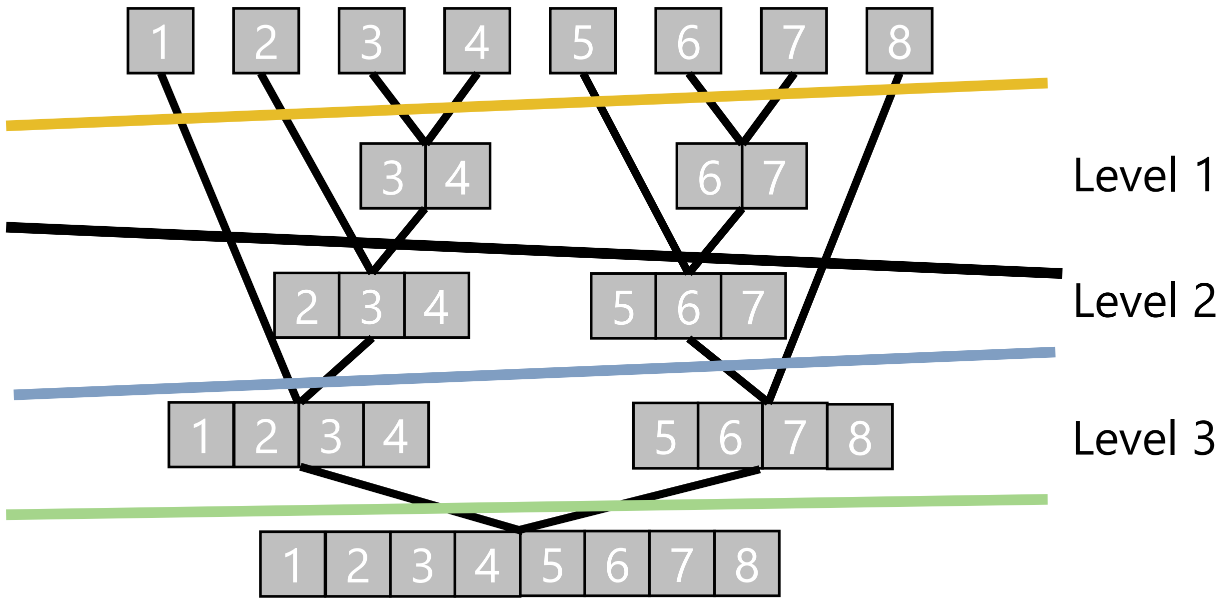
("dendrogram")

Hierarchical clustering

Splitting the dendrogram at different points defines cluster "levels" from which we can build our feature representation

L1, L2, L3

1: [0,0,0,0,1,0]
2: [0,0,1,0,1,0]
3: [1,0,1,0,1,0]
4: [1,0,1,0,1,0]
5: [0,0,0,1,0,1]
6: [0,1,0,1,0,1]
7: [0,1,0,1,0,1]
8: [0,0,0,0,0,1]



Model selection

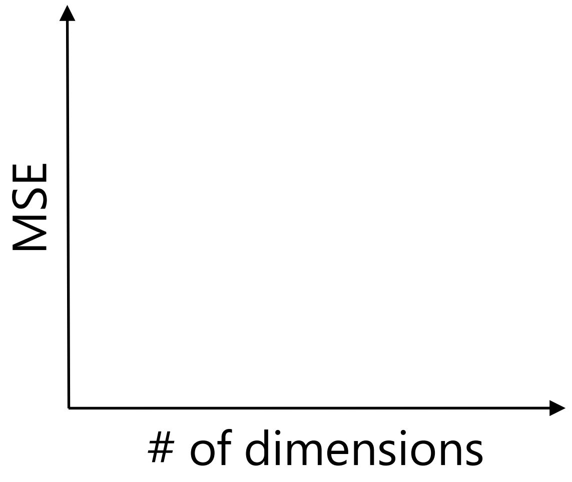
- **Q:** How to choose K in K-means?

(or:

- How to choose how many PCA dimensions to keep?
- How to choose at what position to "cut" our hierarchical clusters?
- (later) how to choose how many communities to look for in a network)

Model selection

1) As a means of "compressing" our data
   - Choose however many dimensions we can afford to obtain a given file size/compression ratio
   - Keep adding dimensions until adding more no longer decreases the reconstruction error significantly



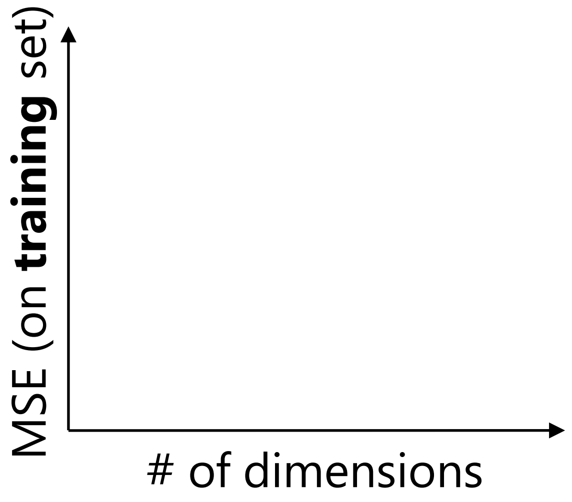
Model selection

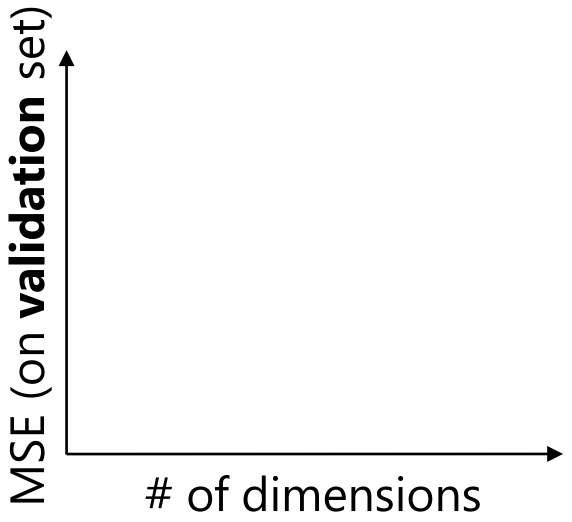
2) As a means of generating potentially useful features for some other predictive task (which is what we're more interested in in a predictive analytics course!)

- Increasing the number of dimensions/number of clusters gives us additional features to work with, i.e., a longer feature vector
- In some settings, we may be running an algorithm whose complexity (either time or memory) scales with the feature dimensionality (such as we saw last week!); in this case we would just take however many dimensions we can afford

Model selection

- Otherwise, we should choose however many dimensions results in the best prediction performance **on held out data**





Learning Outcomes

- Introduced hierarchical clustering
- Discussed how validation sets can be used to choose hyperparameters (besides just for regularization)

References

Further reading:

- Ricardo Gutierrez-Osuna's PCA slides (slightly more mathsy than mine):
http://research.cs.tamu.edu/prism/lectures/pr/pr_l9.pdf
- Relationship between PCA and K-means:
http://ranger.uta.edu/~chqding/papers/KmeansPCA1.pdf
http://ranger.uta.edu/~chqding/papers/Zha-Kmeans.pdf

Web Mining and Recommender Systems

Community Detection: Introduction

Learning Goals

- Introduce community detection
- Explain how it is different from clustering and other forms of dimensionality reduction

Community detection versus clustering

So far we have seen methods to reduce the dimension of points based on their **features**

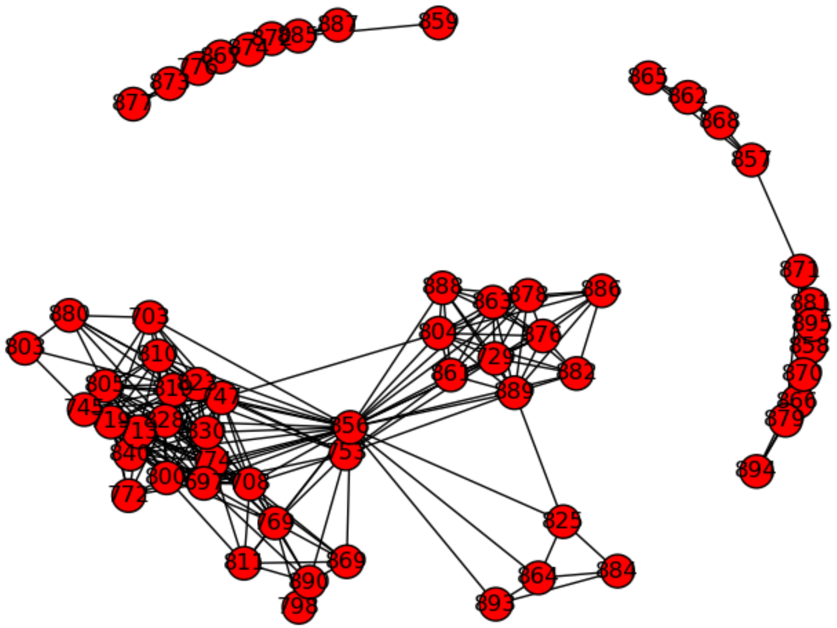
Community detection versus clustering

So far we have seen methods to reduce the dimension of points based on their **features**

What if points are not defined by features but by their **relationships** to each other?

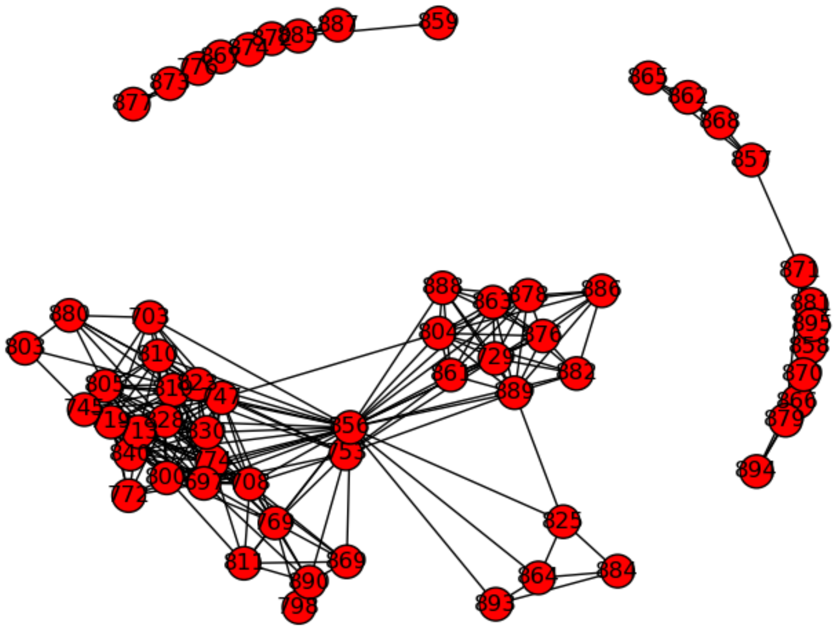
Community detection versus clustering

Q: how can we compactly represent the set of relationships in a graph?



Community detection versus clustering

A: by representing the nodes in terms of the **communities** they belong to



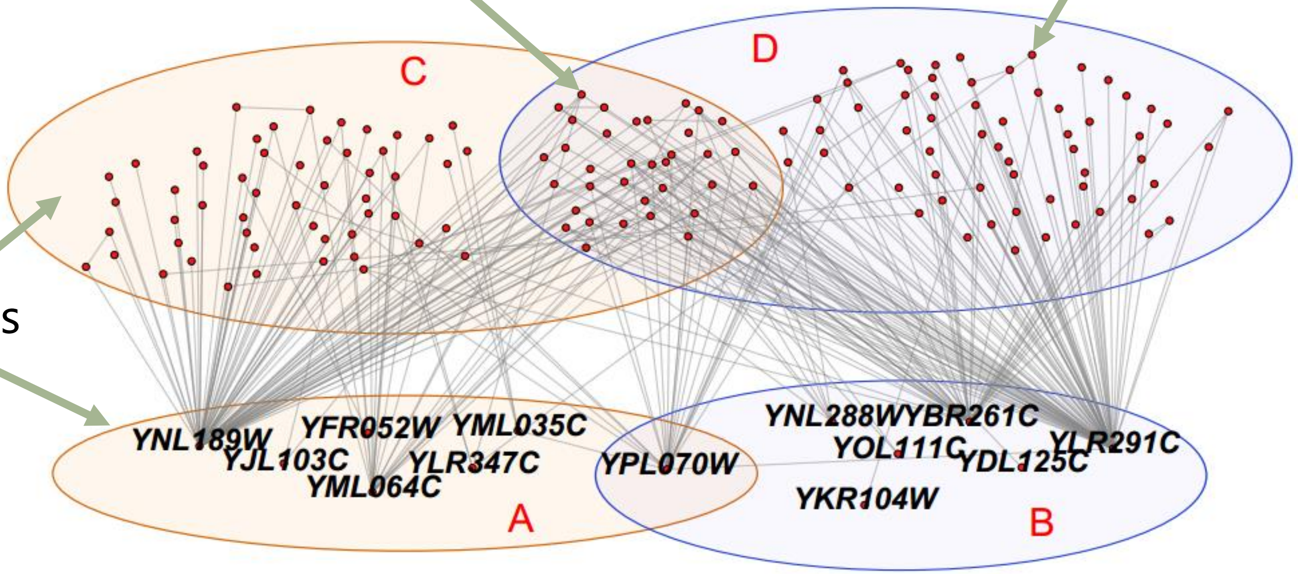
Community detection

(from previous lecture)

f = [0,0,1,1] (A,B,C,D)

f = [0,0,0,1] (A,B,C,D)

communities



e.g. from a PPI network; Yang, McAuley, & Leskovec (2014)

Community detection versus clustering

Part 1 – Clustering
Group sets of points based on their **features**

Part 2 – Community detection
Group sets of points based on their **connectivity**

Warning: These are **rough** distinctions that don't cover all cases. E.g. if I treat a row of an adjacency matrix as a "feature" and run hierarchical clustering on it, am I doing clustering or community detection?

Community detection

How should a "community" be defined?

- Similar behavior / interests?
- Geography?

Common interests

- Mutual friends?
- Cliques / social groups?
- Frequency of interaction?

Common bonds

Community detection

How should a “community” be defined?

1. Members should be connected
2. Few edges between communities
3. “Cliqueishness”
4. Dense inside, few edges outside

Coming up…

1. **Connected components**
(members should be connected)
2. **Minimum cut**
(few edges between communities)
3. **Clique percolation**
("cliqueishness")
4. **Network modularity**
(dense inside, few edges outside)

Web Mining and Recommender Systems

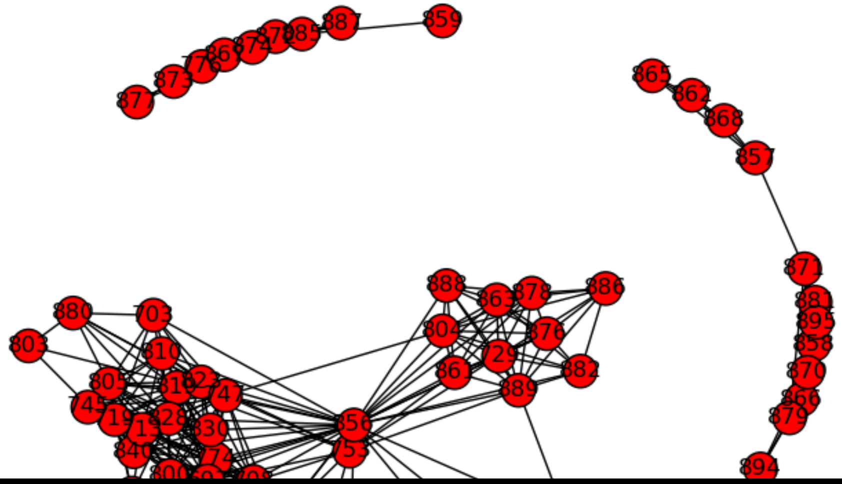
Community Detection: Graph Cuts

Learning Goals

- Introduce community detection algorithms based on Graph Cuts
- (also introduce connected components as a point of contrast)

1. Connected components

Define communities in terms of sets of nodes which are reachable from each other



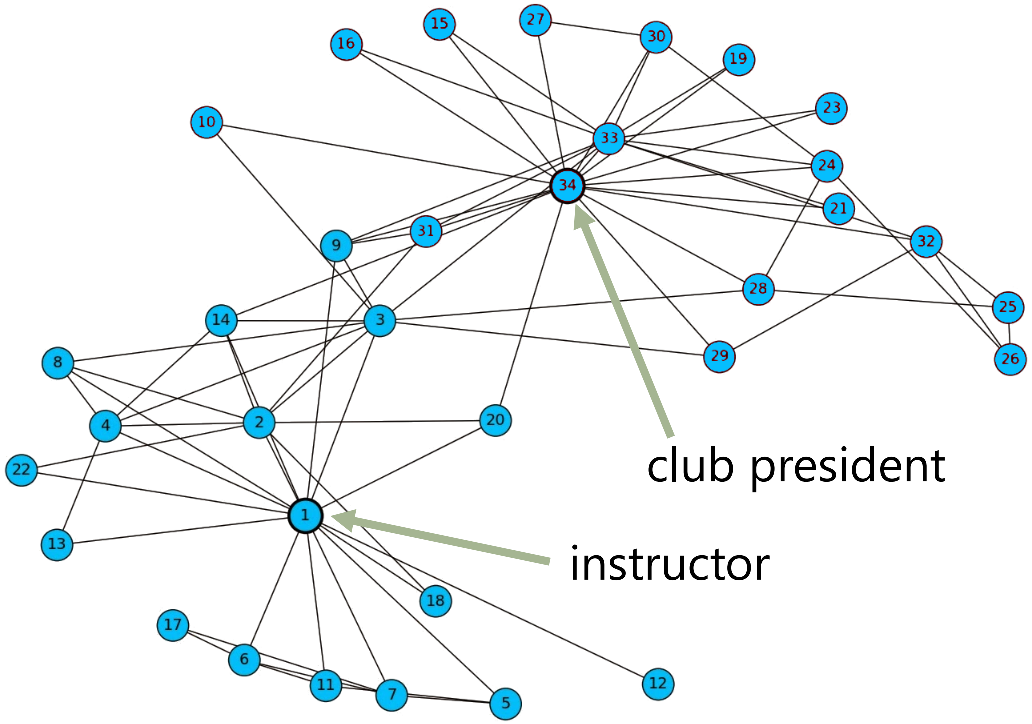
- If *a* and *b* belong to a **strongly connected component** then there must be a path from a → b and a path from b → a
- A **weakly connected component** is a set of nodes that **would be** strongly connected, if the graph were undirected

1. Connected components

- Captures about the roughest notion of "community" that we could imagine
- Not useful for (most) real graphs: there will usually be a "giant component" containing almost all nodes, which is not really a community in any reasonable sense

2. Graph cuts

What if the separation between communities isn't so clear?



e.g. “Zachary’s Karate Club” (1970)

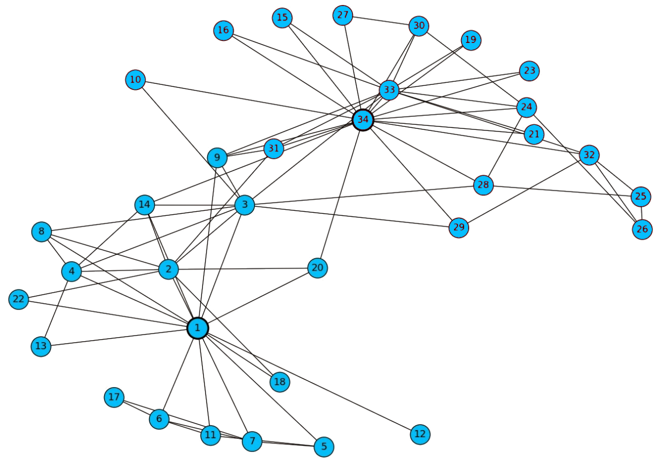
Picture from http://spaghetti-os.blogspot.com/2014/05/zacharys-karate-club.html

Aside: Zachary's Karate Club Club

http://networkkarate.tumblr.com/

2. Graph cuts

Cut the network into two partitions such that the number of edges crossed by the cut is minimal



Solution will be degenerate – we need additional constraints

2. Graph cuts

We'd like a cut that favors **large** communities over small ones

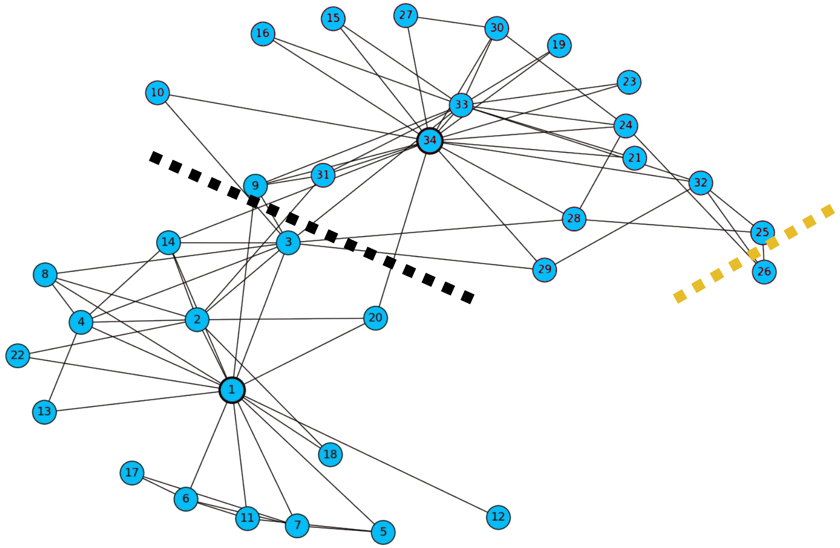
#of edges that separate *c* from the rest of the network

$$\text{Ratio Cut}(C) = \frac{1}{|C|} \sum_{c \in C} \frac{cut(c,\bar{c})}{|c|}$$

Proposed set of communities

size of this community

What is the **Ratio Cut** cost of the following two cuts?

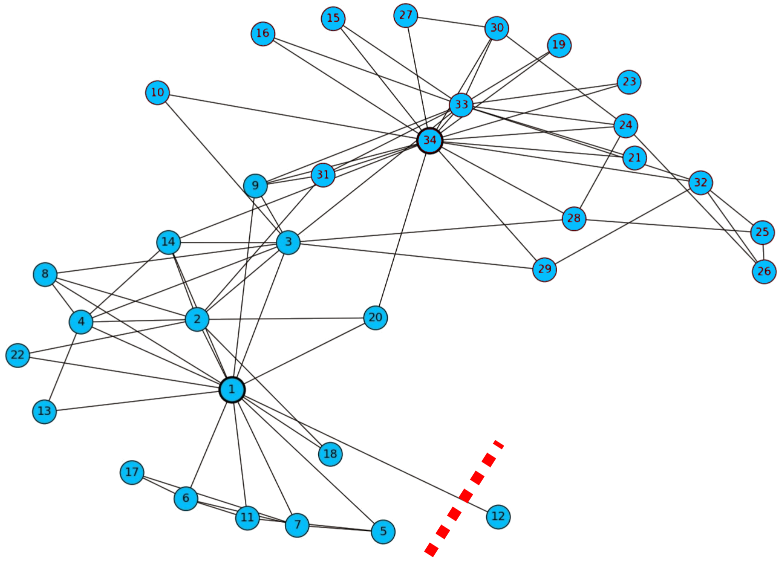


$$\text{Ratio Cut}(\text{yellow cut}) = \frac{1}{2}\left(\frac{3}{33} + \frac{3}{1}\right) = 1.54545$$

$$\text{Ratio Cut}(\text{black cut}) = \frac{1}{2}\left(\frac{9}{16} + \frac{9}{18}\right) = 0.53125$$

2. Graph cuts

But what about…



$$\text{Ratio Cut}(\ddots) = \frac{1}{2}\left(\frac{1}{33} + \frac{1}{1}\right) = 0.51515$$

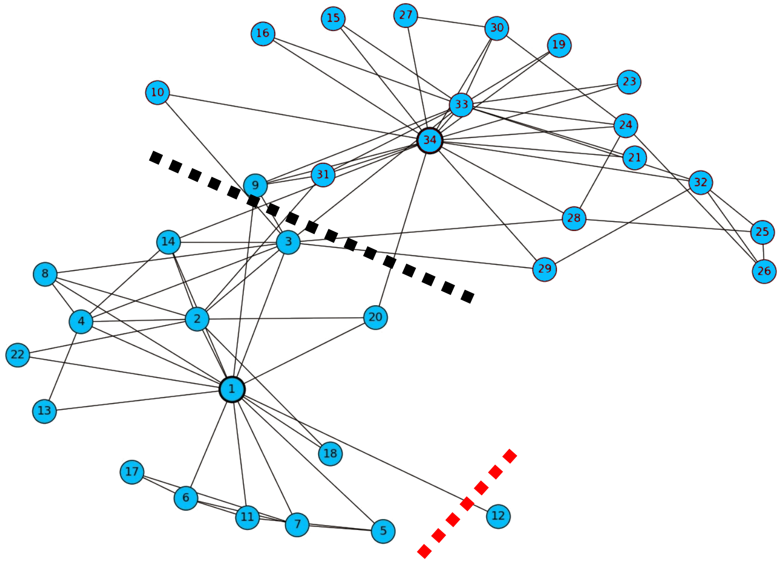
2. Graph cuts

Maybe rather than counting all nodes equally in a community, we should give additional weight to “influential”, or high-degree nodes

$$\text{Normalized Cut}(C) = \frac{1}{|C|} \sum_{c \in C} \frac{cut(c, \bar{c})}{\sum \text{degrees in } c}$$

nodes of high degree will have more influence in the denominator

What is the **Normalized Cut** cost of the following two cuts?



$$\text{Norm. Cut}(\ddots) = \frac{1}{2}\left(\frac{1}{155} + \frac{1}{1}\right) = 0.50322$$

$$\text{Norm. Cut}(\ddots) = \frac{1}{2}\left(\frac{9}{76} + \frac{9}{80}\right) = 0.11546$$

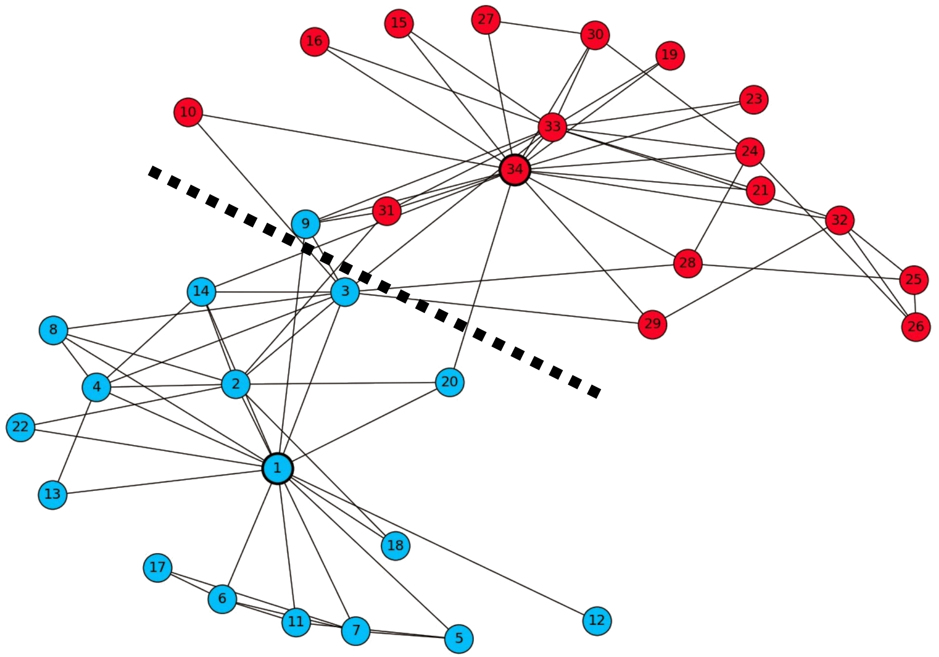
2. Graph cuts

Code:

```
>>> Import networkx as nx
>>> G = nx.karate_club_graph()
>>> c1 = [1,2,3,4,5,6,7,8,11,12,13,14,17,18,20,22]
>>> c2 = [9,10,15,16,19,21,23,24,25,26,27,28,29,30,31,32,33,34]
>>> Sum([G.degree(v-1) for v in c1])
76
>>> sum([G.degree(v-1) for v in c2])
80
```

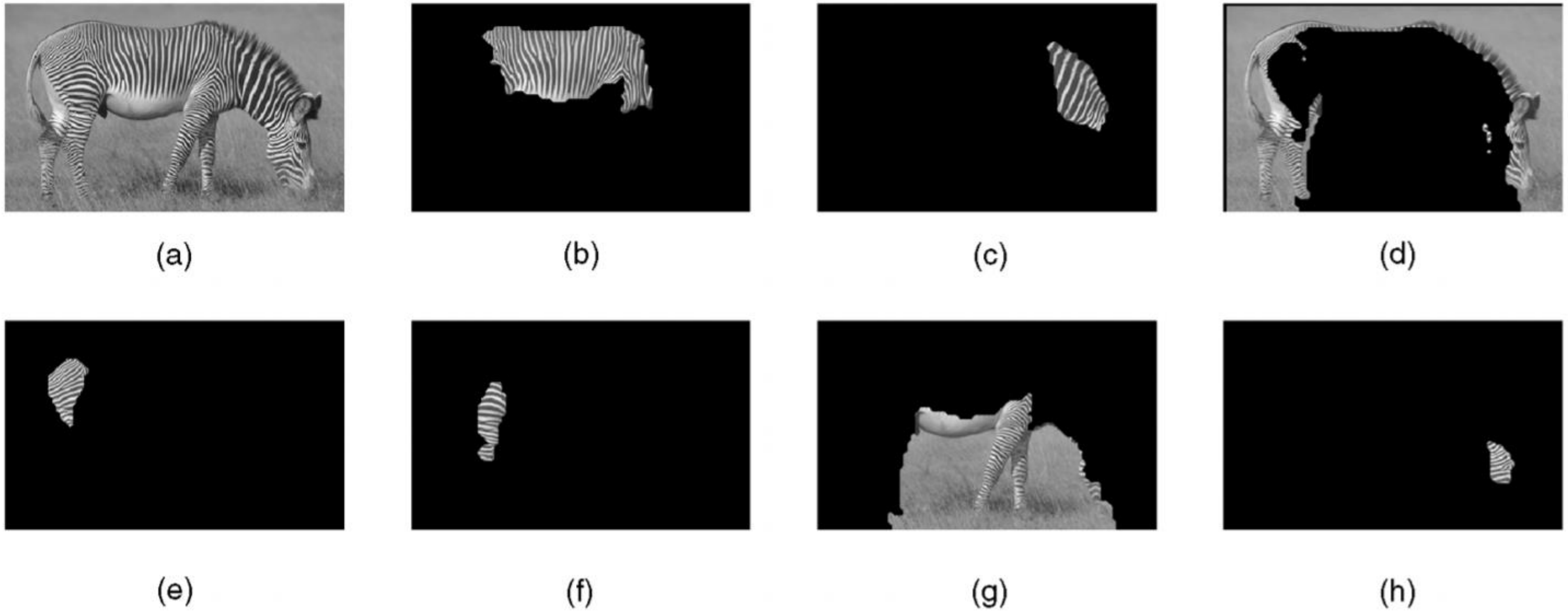
Nodes are indexed from 0 in the networkx dataset, 1 in the figure

So what actually happened?



- ⋱ = Optimal cut
- Red/blue = actual split

Normalized cuts in Computer Vision



“Normalized Cuts and Image Segmentation”
Shi and Malik, 1998

Learning Outcomes

- Introduced graph cuts-based community detection algorithms
- Showed some of the challenges in designing a community detection algorithm based on this concept
- Discussed the history of the community detection problem a little

Web Mining and Recommender Systems

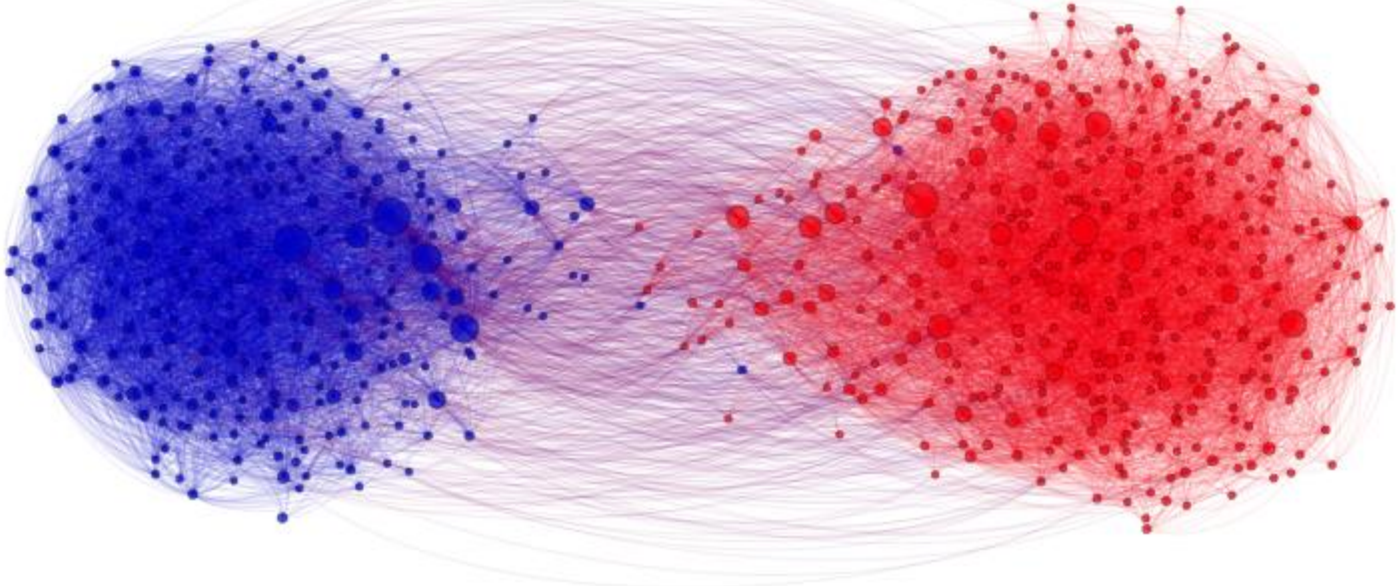
Community Detection: Clique Percolation

Learning Goals

- Introduce the Clique Percolation community detection algorithm

Disjoint communities

Separating networks into disjoint subsets seems to make sense when communities are somehow "adversarial"



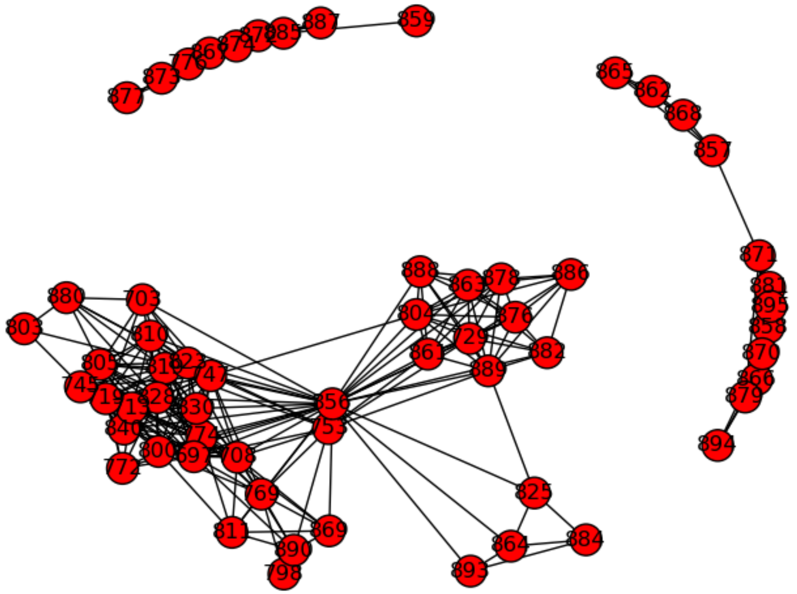
E.g. links between democratic/republican political blogs (from Adamic, 2004)

Graph data from Adamic (2004). Visualization from allthingsgraphed.com

Social communities

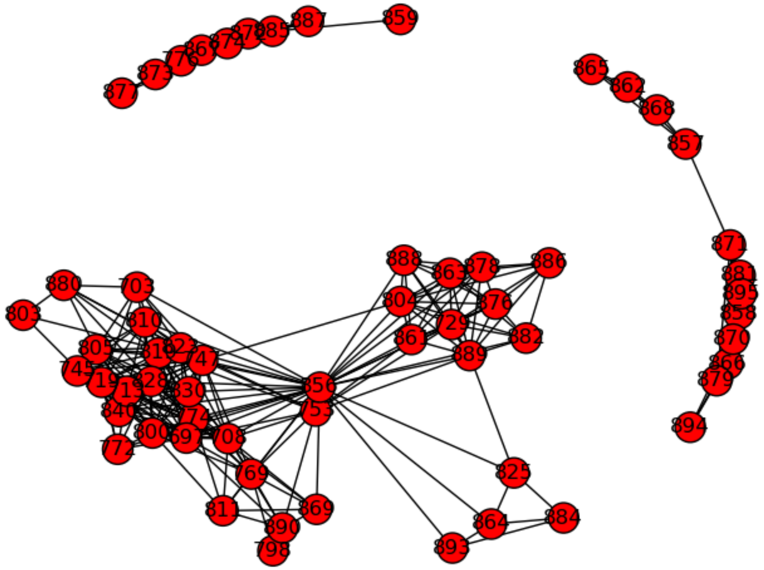
But what about communities in social networks (for example)?

e.g. the graph of my facebook friends:



http://jmcauley.ucsd.edu/cse258/data/facebook/egonet.txt

Social communities



Such graphs might have:

- **Disjoint** communities (i.e., groups of friends who don't know each other)
  e.g. my American friends and my Australian friends
- **Overlapping** communities (i.e., groups with **some** intersection)
  e.g. my friends and my girlfriend's friends
- **Nested** communities (i.e., one group within another)
  e.g. my UCSD friends and my CSE friends

3. Clique percolation

How can we define an algorithm that handles all three types of community (disjoint/overlapping/nested)?

Clique percolation is one such algorithm, that discovers communities based on their “cliqueishness”

3. Clique percolation

- Clique percolation searches for “cliques” in the network of a certain size (K). Initially each of these cliques is considered to be its own community
- If two communities share a (K-1) clique in common, they are merged into a single community
- This process repeats until no more communities can be merged

```
Given a clique size K
Initialize every K-clique as its own community
While (two communities I and J have a (K-1)-clique in common):
    Merge I and J into a single community
```

3. Clique percolation

Learning Outcomes

- Introduced Clique Percolation
- Discussed some of the underlying assumptions made by different community detection algorithms

Web Mining and Recommender Systems

Community Detection: Network Modularity

Learning Goals

- Introduce Network Modularity

What is a “good” community algorithm?

- So far we’ve just defined algorithms to match some (hopefully reasonable) intuition of what communities should “look like”
- But how do we know if one definition is better than another? I.e., how do we evaluate a community detection algorithm?

- Can we define a **probabilistic model** and evaluate the **likelihood** of observing a certain set of communities compared to some null model

4. Network modularity

Null model:
Edges are equally likely between any pair of nodes, regardless of community structure
(“Erdos-Renyi random model”)

4. Network modularity

Null model:
Edges are equally likely between any pair of nodes, regardless of community structure
("Erdos-Renyi random model")

Q: How much does a proposed set of communities **deviate** from this null model?

4. Network modularity

4. Network modularity

$$e_{kk} = \frac{\#\ \text{edges with both endpoints in community } k}{\#\ \text{edges}}$$

$$a_k = \frac{\#\ \text{edge endpoints in community } k}{\#\ \text{edge endpoints}}$$

$$Q = \sum_{k=1}^{K} (e_{kk} - a_k^2)$$

Fraction of edges in community k

Fraction that we would expect if edges were allocated randomly

4. Network modularity

$$e_{kk} = \frac{\#\text{ edges with both endpoints in community } k}{\#\text{ edges}}$$

$$a_k = \frac{\#\text{ edge endpoints in community } k}{\#\text{ edge endpoints}}$$

$$Q = \sum_{k=1}^{K} (e_{kk} - a_k^2)$$

4. Network modularity

4. Network modularity

$$e_{kk} = \frac{\#\text{ edges with both endpoints in community } k}{\#\text{ edges}}$$

$$a_k = \frac{\#\text{ edge endpoints in community } k}{\#\text{ edge endpoints}}$$

$$Q = \sum_{k=1}^{K} (e_{kk} - a_k^2)$$

$$-\frac{1}{2} \leq Q < 1$$

Far fewer edges in communities than we would expect at random

Far more edges in communities than we would expect at random

4. Network modularity

Algorithm: Choose communities so that the deviation from the null model is maximized

$$Q = \sum_{k=1}^{K} (e_{kk} - a_k^2)$$

$$\arg\max_{\text{communities}} Q(\text{communities})$$

That is, choose communities such that **maximally** many edges are within communities and **minimally** many edges cross them
(NP Hard, have to approximate, e.g. choose greedily)

Summary

- Community detection aims to summarize the structure in networks
  (as opposed to clustering which aims to summarize feature dimensions)
- Communities can be defined in various ways, depending on the type of network in question
  1. Members should be connected (connected components)
  2. Few edges between communities (minimum cut)
  3. "Cliqueishness" (clique percolation)
  4. Dense inside, few edges outside (network modularity)

Learning Outcomes

- Introduced network modularity
- Briefly summarized our discussion of community detection

References

Further reading:

Just on modularity: http://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/modularity.pdf

Various community detection algorithms, includes spectral formulation of ratio and normalized cuts: http://dmml.asu.edu/cdm/slides/chapter3.pptx