

Machine Learning for Data Science (CS4786)

Lecture 15

Review + Probabilistic Modeling

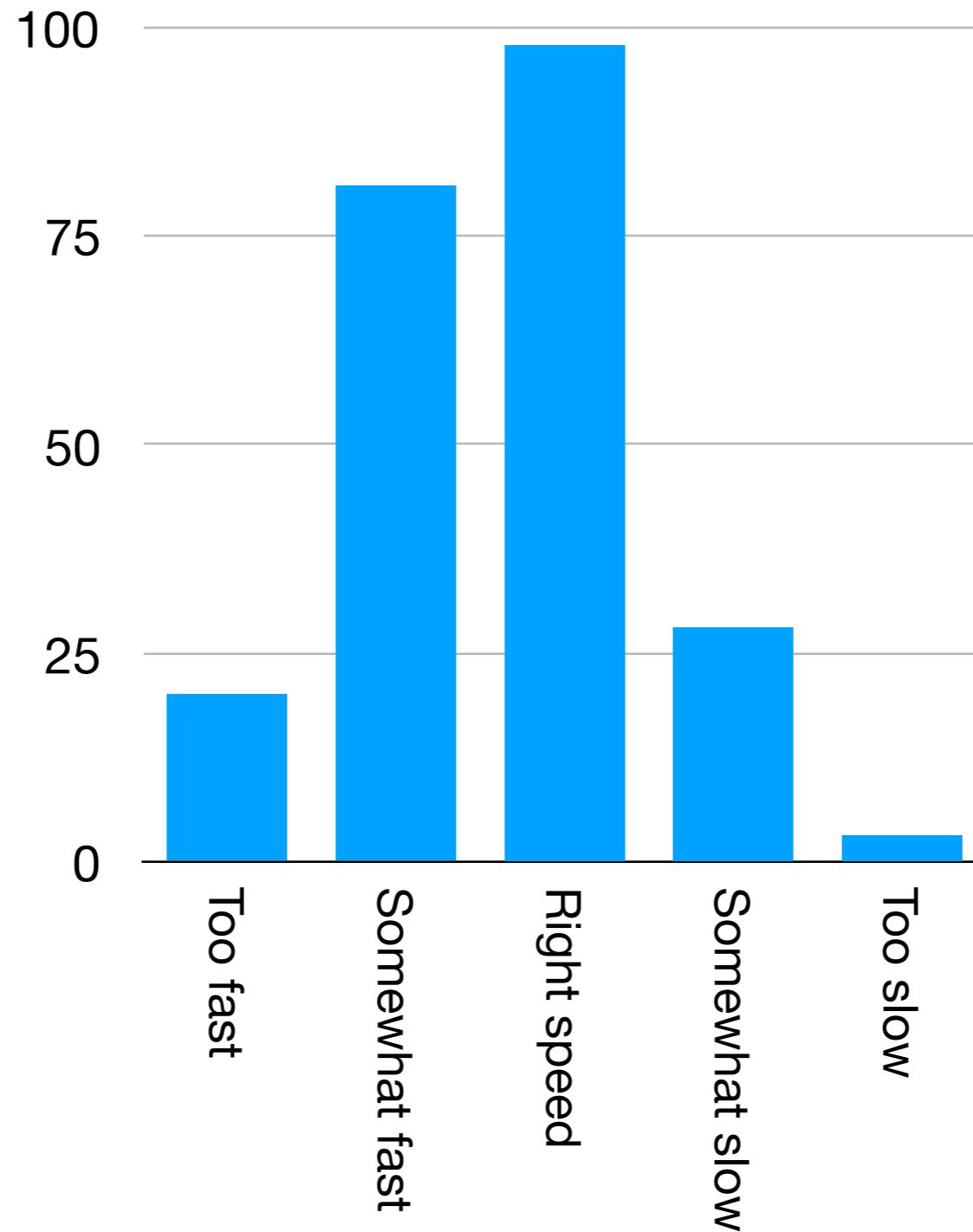
Course Webpage :

<http://www.cs.cornell.edu/Courses/cs4786/2017fa/>

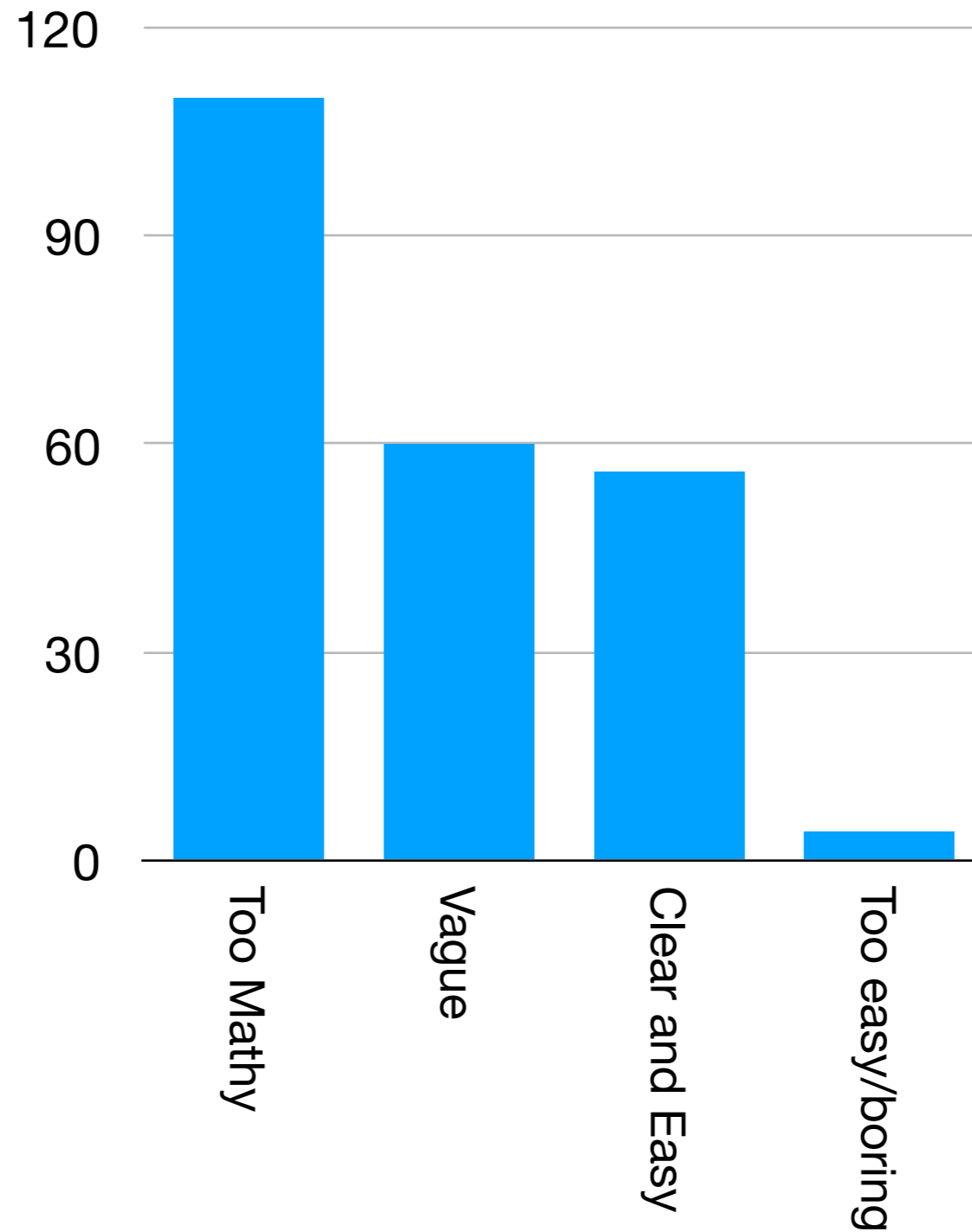
Announcements

- In-class Kaggle link is up
 - only one registration per group
 - 5 submissions per day allowed
 - Start early so you get more submissions
- Survey: Participation 95.44% ! Kudos!

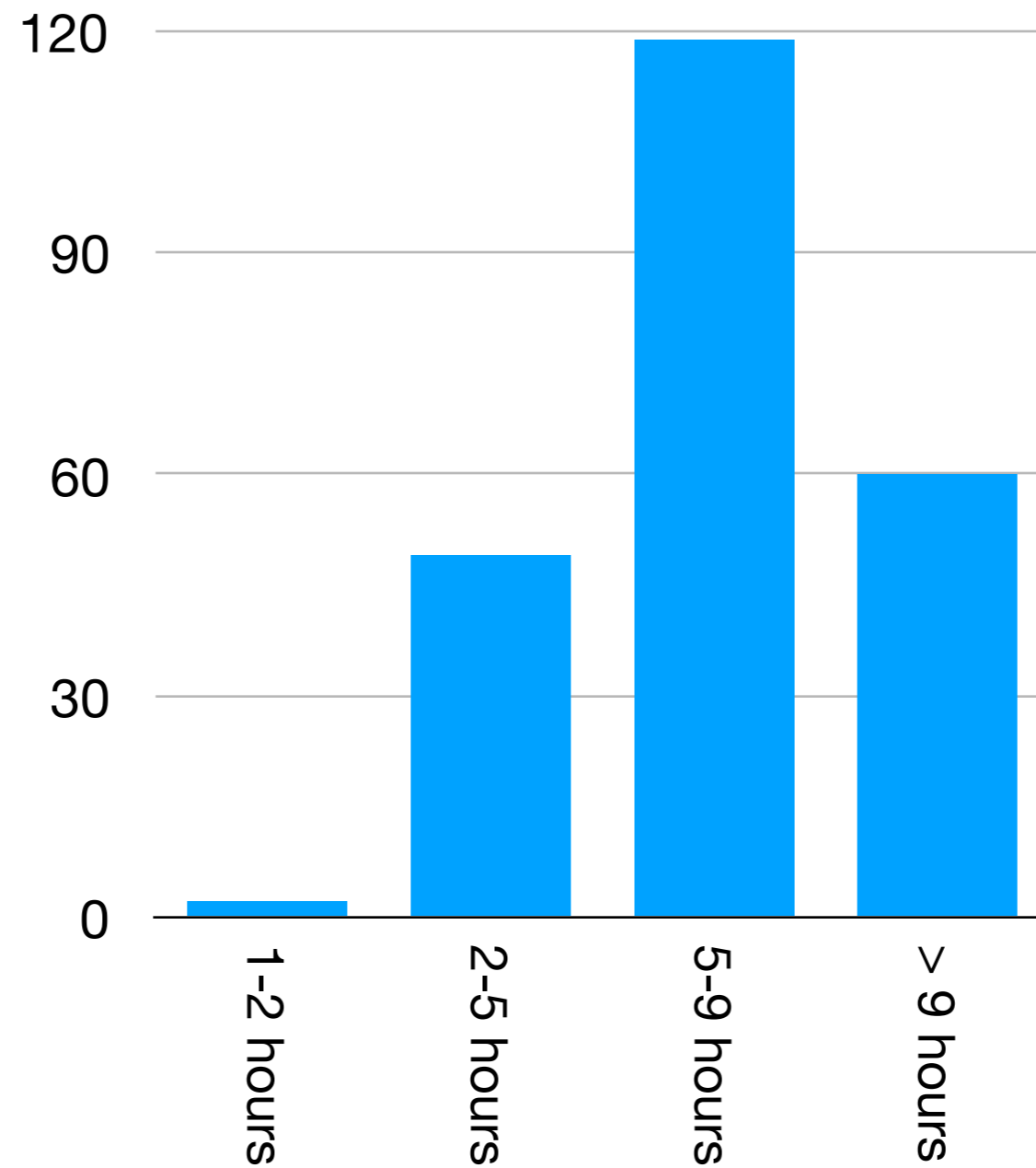
Lecture Speed



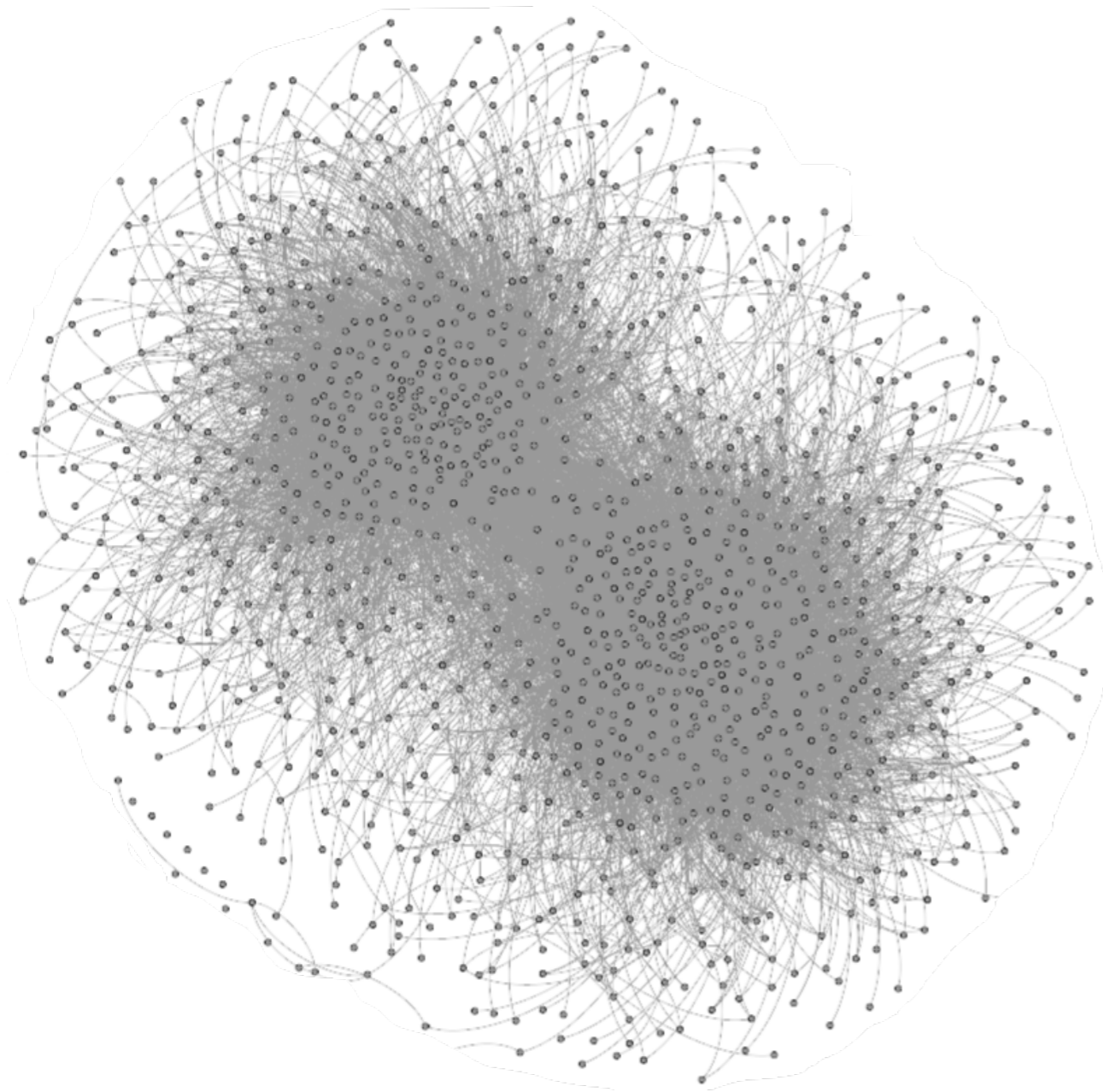
Lecture Style/Clarity



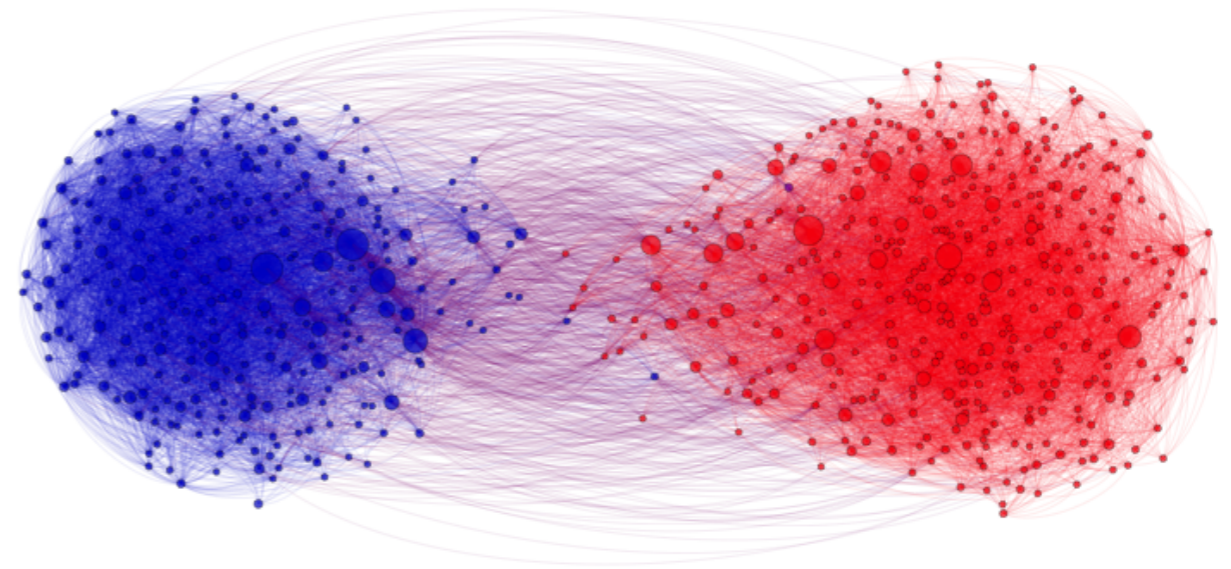
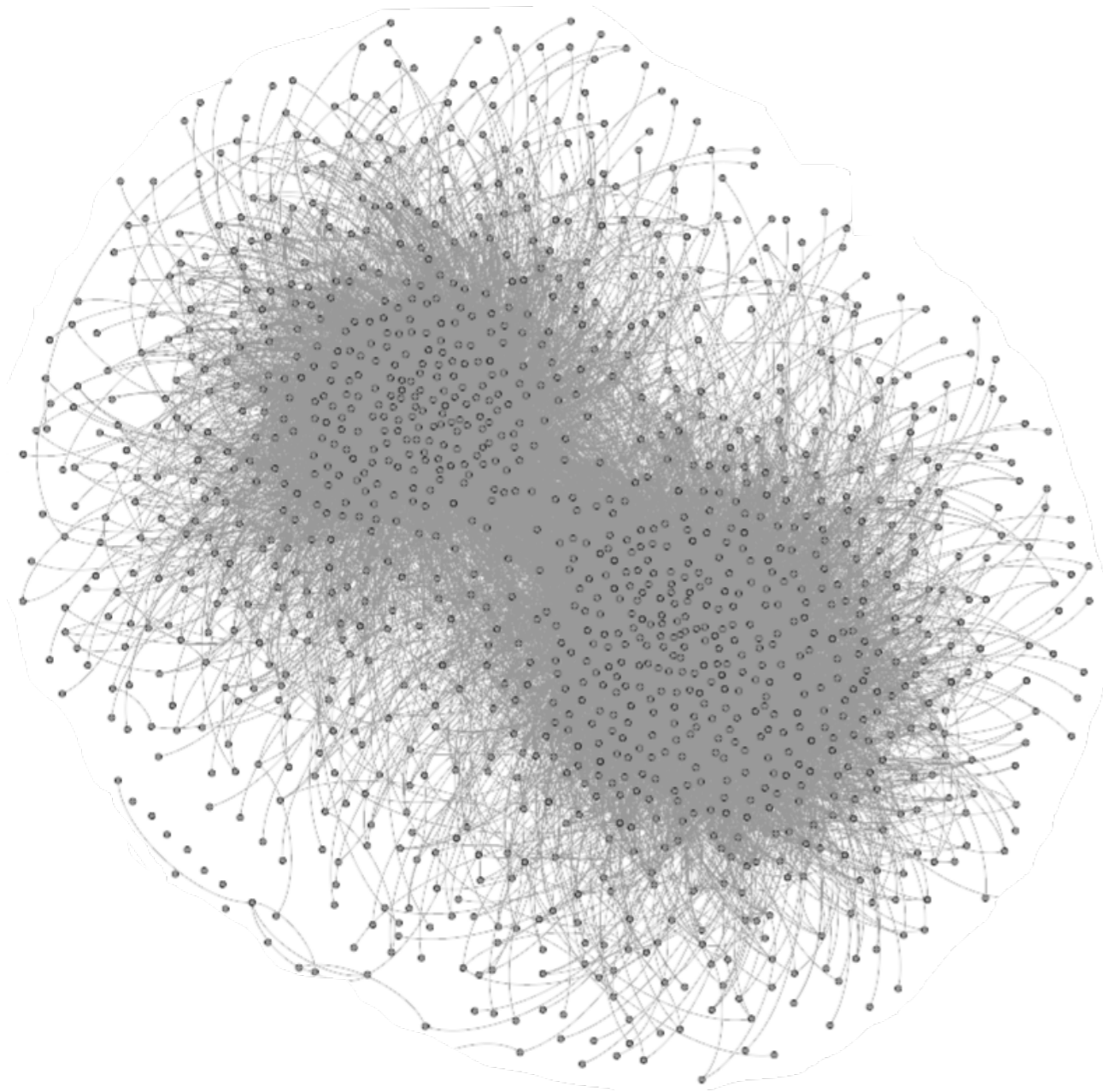
Assignment Load



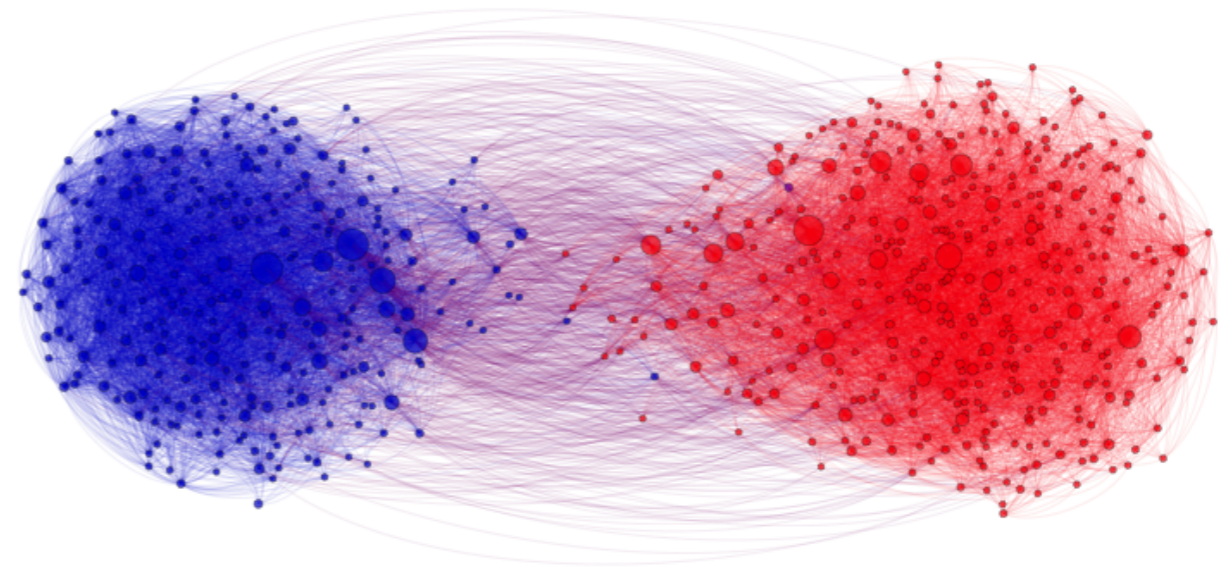
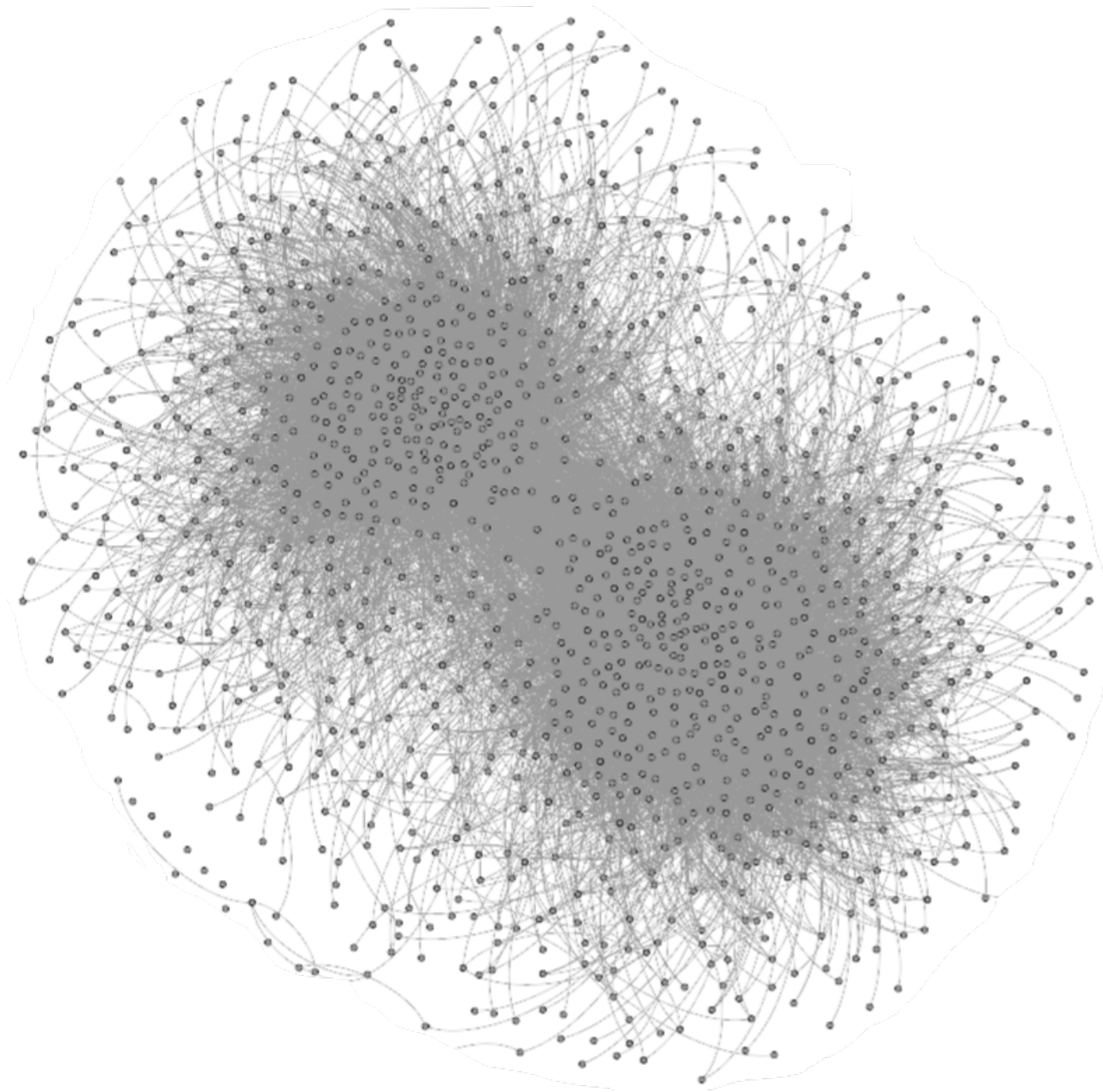
SPECTRAL CLUSTERING



SPECTRAL CLUSTERING



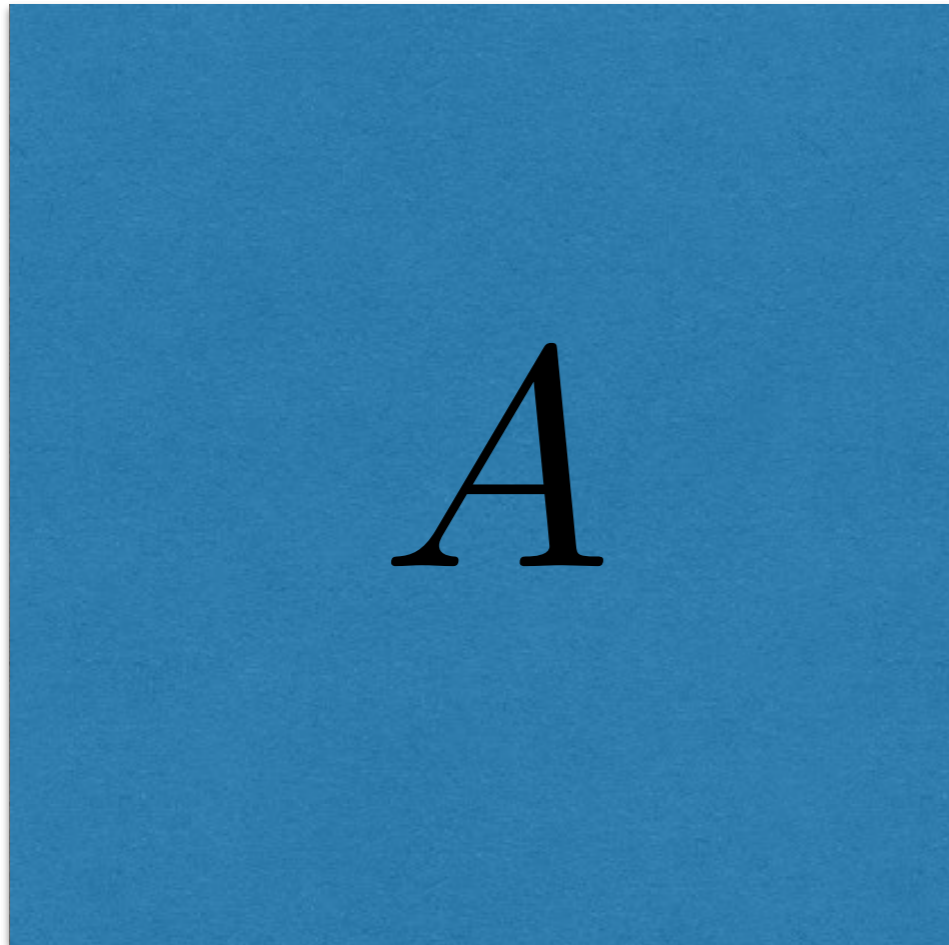
SPECTRAL CLUSTERING



- Cluster nodes in a graph.
- Analysis of social network data.

Steps

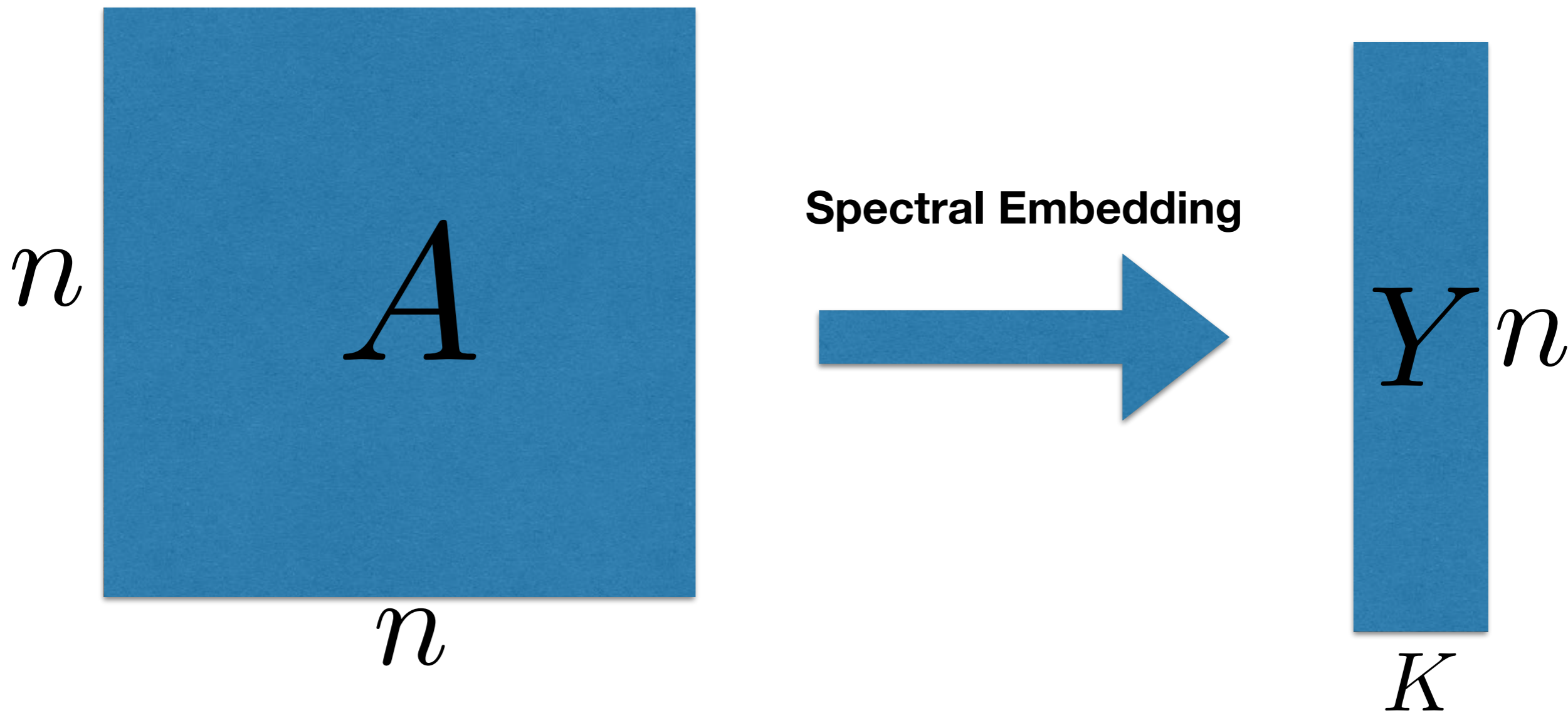
n



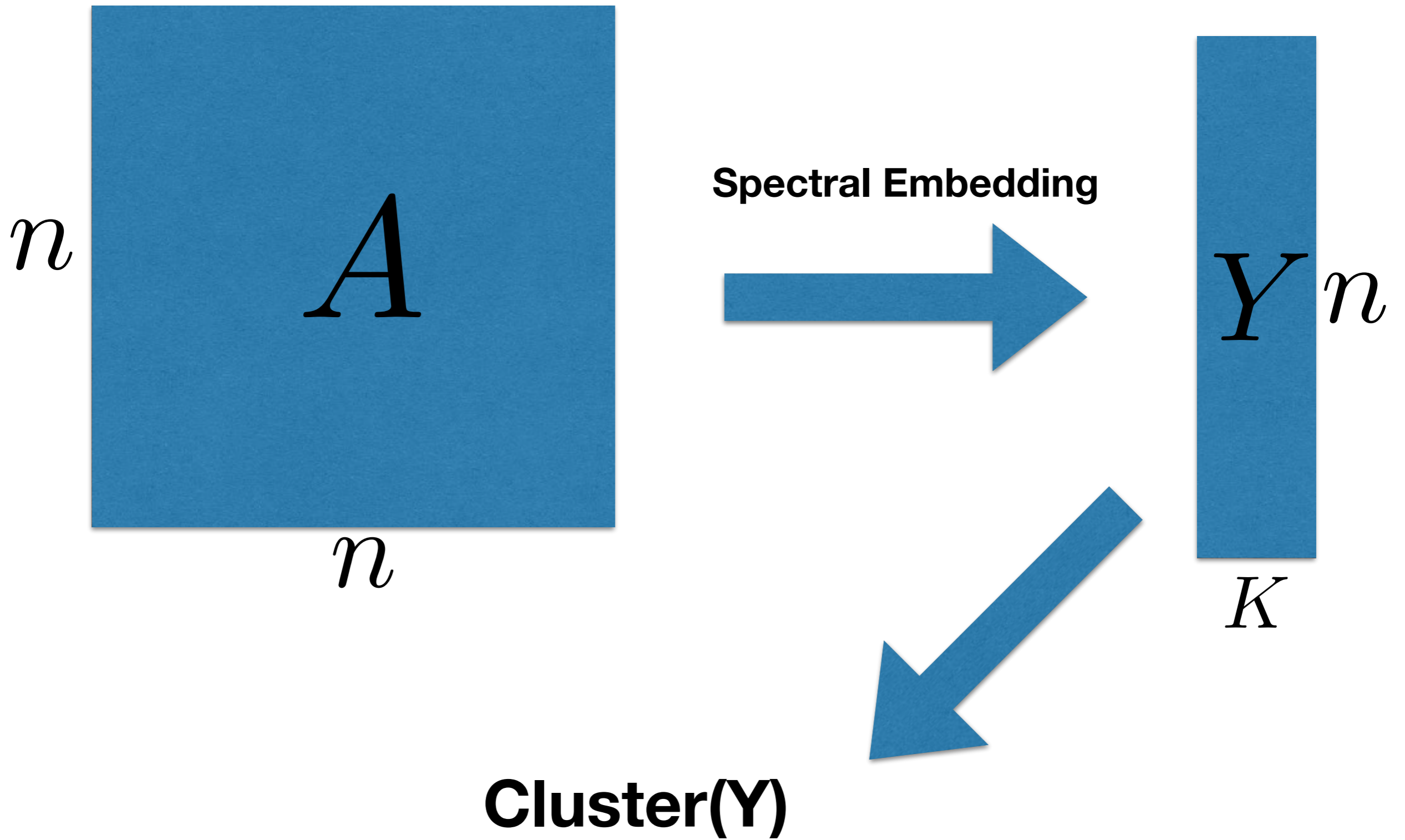
A

n

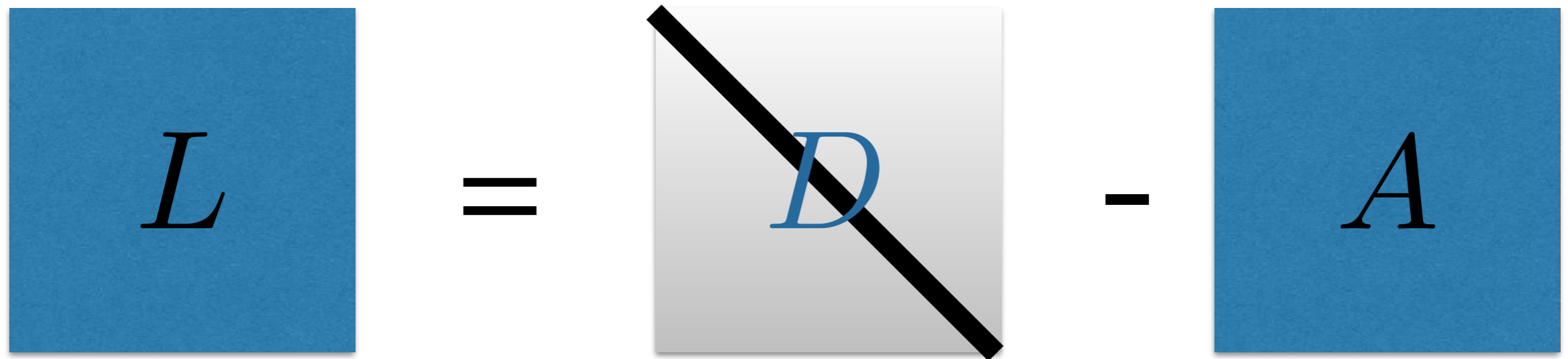
Steps



Steps



SPECTRAL CLUSTERING

$$L = D - A$$
The diagram illustrates the equation $L = D - A$ using three square boxes. The first box on the left is blue and contains the letter L . To its right is an equals sign. The second box is gray and contains the letter D , with a thick black diagonal line crossing it from the top-left to the bottom-right. To its right is a minus sign. The third box on the right is blue and contains the letter A .

SPECTRAL CLUSTERING

A diagram illustrating the relationship between the Laplacian matrix L , the degree matrix D , and the adjacency matrix A . On the left is a blue square containing the letter L . To its right is an equals sign. Next is a light gray square containing the letter D , which is crossed out with a thick black diagonal line. To the right of this is a minus sign, followed by a blue square containing the letter A .

$$D_{i,i} = \sum_{j=1}^n A_{i,j}$$

Spectral Embedding

- Nodes linked to each other are close in embedded space

SPECTRAL CLUSTERING ALGORITHM (UNNORMALIZED)

- 1 Given matrix A calculate diagonal matrix D s.t. $D_{i,i} = \sum_{j=1}^n A_{i,j}$
- 2 Calculate the Laplacian matrix $L = D - A$
- 3 Find eigen vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of L (ascending order of eigenvalues)
- 4 Pick the K eigenvectors with smallest eigenvalues to get $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$
- 5 Use K-means clustering algorithm on $\mathbf{y}_1, \dots, \mathbf{y}_n$

SPECTRAL CLUSTERING ALGORITHM (UNNORMALIZED)

- 1 Given matrix A calculate diagonal matrix D s.t. $D_{i,i} = \sum_{j=1}^n A_{i,j}$
- 2 Calculate the Laplacian matrix $L = D - A$
- 3 Find eigen vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of L (ascending order of eigenvalues)
- 4 Pick the K eigenvectors with smallest eigenvalues to get $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$
- 5 Use K-means clustering algorithm on $\mathbf{y}_1, \dots, \mathbf{y}_n$

$\mathbf{y}_1, \dots, \mathbf{y}_n$ are called spectral embedding

SPECTRAL CLUSTERING ALGORITHM (UNNORMALIZED)

- 1 Given matrix A calculate diagonal matrix D s.t. $D_{i,i} = \sum_{j=1}^n A_{i,j}$
- 2 Calculate the Laplacian matrix $L = D - A$
- 3 Find eigen vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of L (ascending order of eigenvalues)
- 4 Pick the K eigenvectors with smallest eigenvalues to get $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$
- 5 Use K-means clustering algorithm on $\mathbf{y}_1, \dots, \mathbf{y}_n$

$\mathbf{y}_1, \dots, \mathbf{y}_n$ are called spectral embedding

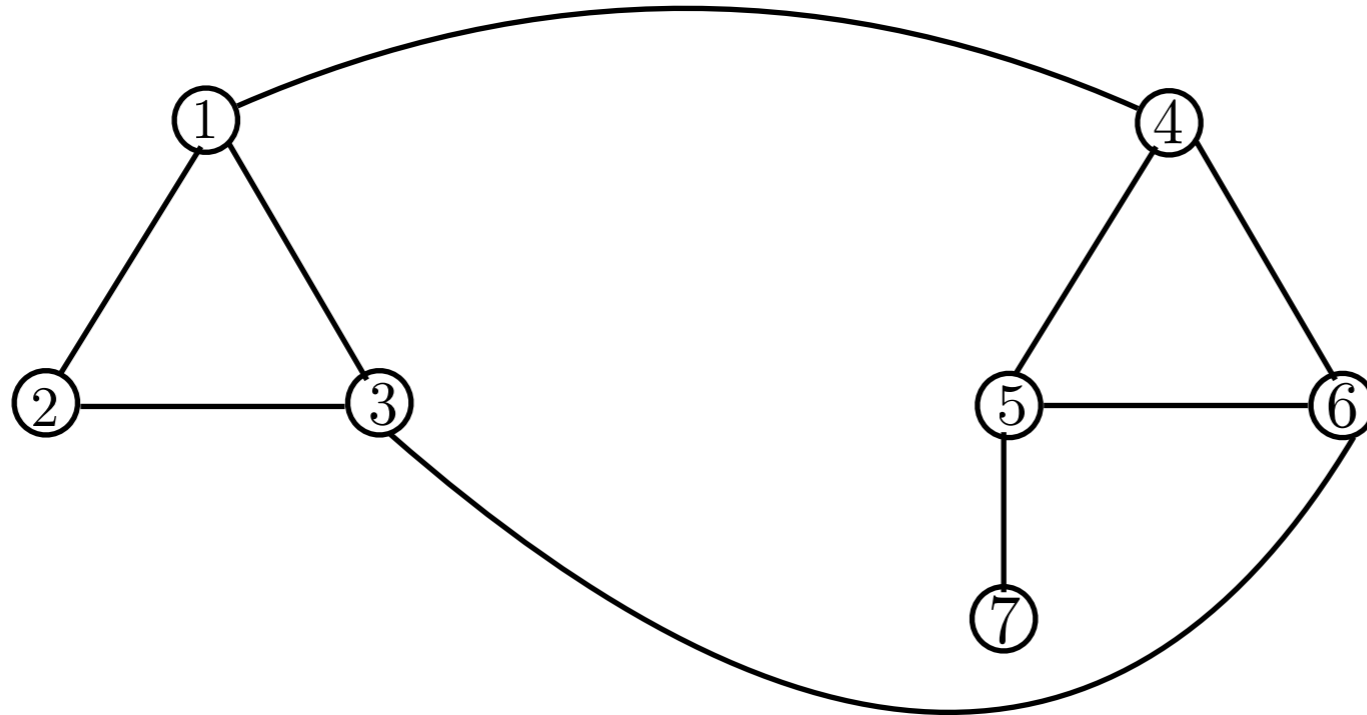
Embeds the n nodes into $K-1$ dimensional vectors

NORMALIZED CUT

- Why cut is perhaps not a good measure?

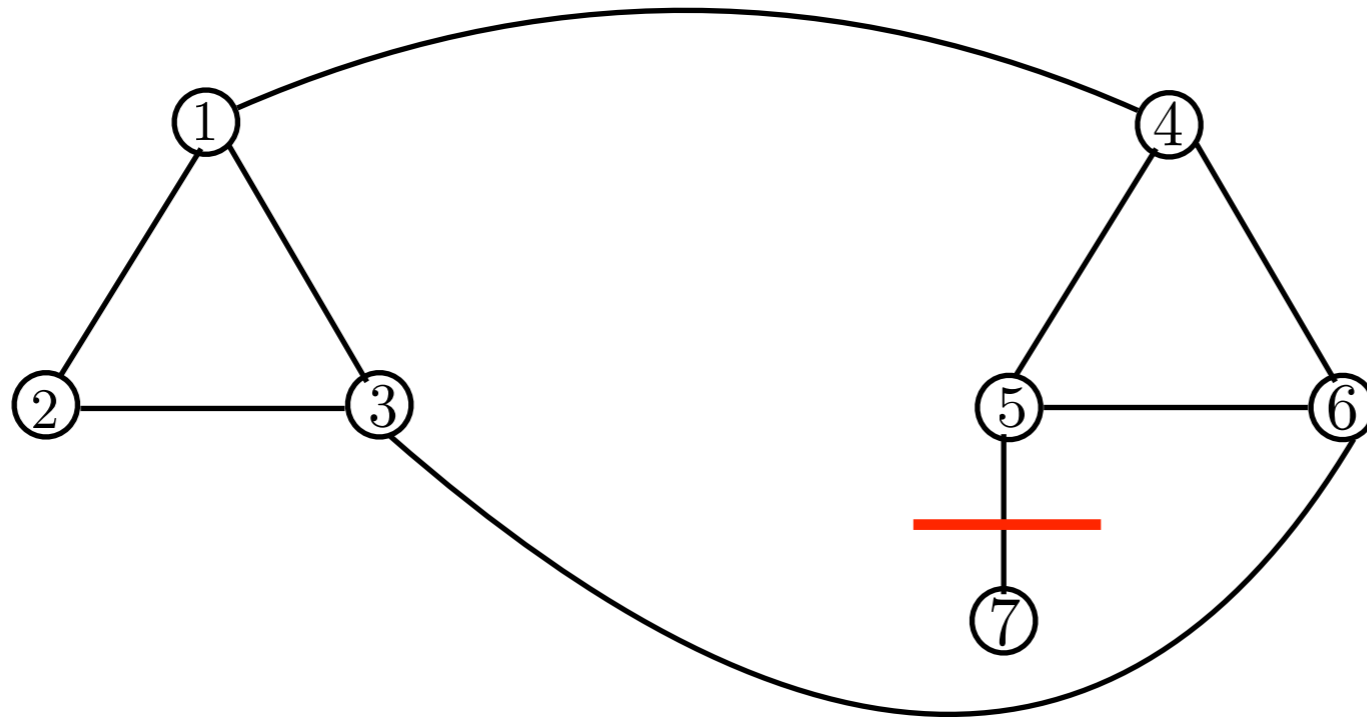
NORMALIZED CUT

- Why cut is perhaps not a good measure?



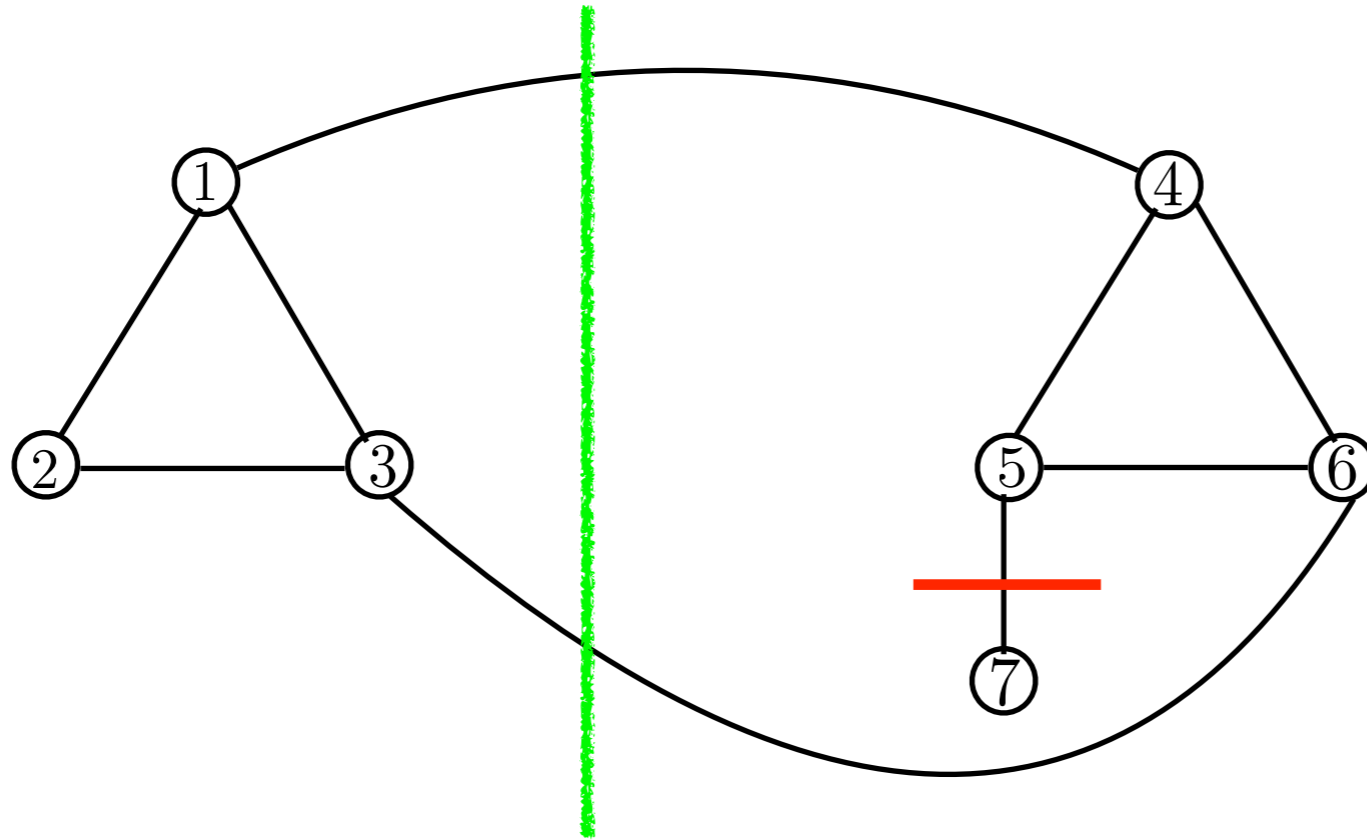
NORMALIZED CUT

- Why cut is perhaps not a good measure?



NORMALIZED CUT

- Why cut is perhaps not a good measure?



NORMALIZED CUT

- Normalized cut: Minimize sum of ratio of number of edges cut per cluster and number of edges within cluster

$$\text{NCUT} = \sum_j \frac{\text{CUT}(C_j)}{\text{Edges}(C_j)}$$

NORMALIZED SPECTRAL CLUSTERING

- As before, we want to minimize $\sum_{(i,j) \in E} (c_i - c_j)^2 = c^\top Lc$

NORMALIZED SPECTRAL CLUSTERING

- As before, we want to minimize $\sum_{(i,j) \in E} (c_i - c_j)^2 = c^T L c$
- But we also want to weight the values of c_i 's based on degree. We want high degree nodes to have larger c magnitude

NORMALIZED SPECTRAL CLUSTERING

- As before, we want to minimize $\sum_{(i,j) \in E} (c_i - c_j)^2 = c^\top Lc$
- But we also want to weight the values of c_i 's based on degree. We want high degree nodes to have larger c magnitude
- That is we want to simultaneously maximize $\sum_{i=1}^n c_i^2 D_{i,i} = c^\top Dc$

NORMALIZED SPECTRAL CLUSTERING

- As before, we want to minimize $\sum_{(i,j) \in E} (c_i - c_j)^2 = c^\top Lc$
- But we also want to weight the values of c_i 's based on degree. We want high degree nodes to have larger c magnitude
- That is we want to simultaneously maximize $\sum_{i=1}^n c_i^2 D_{i,i} = c^\top Dc$
- Find c so as to:

$$\begin{aligned} & \text{minimize } \frac{c^\top Lc}{c^\top Dc} \\ & \equiv \text{minimize } c^\top Lc \text{ subject to } c^\top Dc = 1 \end{aligned}$$

NORMALIZED SPECTRAL CLUSTERING

- As before, we want to minimize $\sum_{(i,j) \in E} (c_i - c_j)^2 = c^\top Lc$
- But we also want to weight the values of c_i 's based on degree. We want high degree nodes to have larger c magnitude
- That is we want to simultaneously maximize $\sum_{i=1}^n c_i^2 D_{i,i} = c^\top Dc$
- Find c so as to:

$$\text{minimize } \frac{c^\top Lc}{c^\top Dc}$$

$$\equiv \text{minimize } c^\top Lc \text{ subject to } c^\top Dc = 1$$

$$\equiv \text{minimize } u^\top D^{-1/2} L D^{-1/2} u \text{ subject to } \|u\| = 1$$

SPECTRAL CLUSTERING

Minimize $c^\top \tilde{L}c$ s.t. $c \perp \mathbf{1}$

SPECTRAL CLUSTERING

Minimize $c^\top \tilde{L}c$ s.t. $c \perp \mathbf{1}$

Approximately Minimize normalized cut!

SPECTRAL CLUSTERING

Minimize $c^\top \tilde{L}c$ s.t. $c \perp \mathbf{1}$

Approximately Minimize normalized cut!

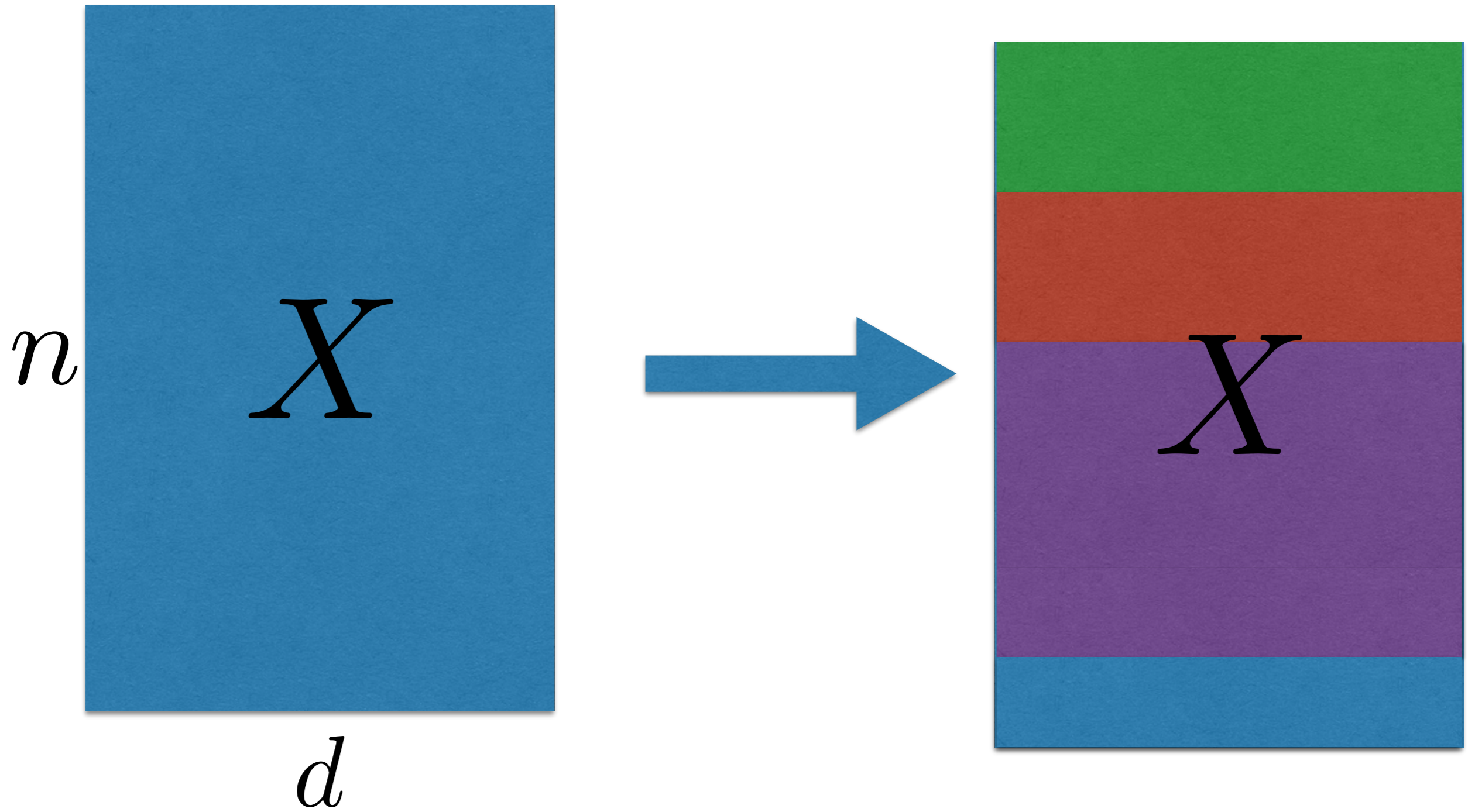
- Solution: Find second smallest eigenvectors of $\tilde{L} = I - D^{-1/2}AD^{-1/2}$

SPECTRAL CLUSTERING ALGORITHM (NORMALIZED)

- 1 Given matrix A calculate diagonal matrix D s.t. $D_{i,i} = \sum_{j=1}^n A_{i,j}$
- 2 Calculate the normalized Laplacian matrix $\tilde{L} = I - D^{-1/2}AD^{-1/2}$
- 3 Find eigen vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of \tilde{L} (ascending order of eigenvalues)
- 4 Pick the K eigenvectors with smallest eigenvalues to get $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$
- 5 Use K-means clustering algorithm on $\mathbf{y}_1, \dots, \mathbf{y}_n$

Review

CLUSTERING



K-means

- K-means algorithm: (wishful thinking)
 - Fix parameters (the k means) and compute new cluster assignments (or probabilities) for every point
 - Fix cluster assignment for all data points and re-evaluate parameters (the k-means)

Single-Link Clustering

- Start with all points being their own clusters
- Until we get K -clusters, merge the closest two clusters

When to Use Single Link

- When we have dense sampling of points within each cluster
- When not to use: when we might have outliers

When to use K-means

- When we have nice spherical round equal size clusters or cluster masses are far apart
- Handles outliers better

PRINCIPAL COMPONENT ANALYSIS

1. $\Sigma = \text{COV}(X)$

2. $W = \text{eigs}(\Sigma, K)$

3. $Y = (X - \mu) \times W$

When to use PCA

- Great when data is truly low dimensional (on a hyperplane (linear))
- Or **approximately** low dimensional (almost lie on plane Eg. very flat ellipsoid)
- Eg. Dimensionality reduction for face images, for multiple biometric applications as preprocessing...

CCA ALGORITHM

$$1. \quad X = \begin{pmatrix} n & \begin{matrix} X_1 \\ d_1 \end{matrix}, & \begin{matrix} X_2 \\ d_2 \end{matrix} \end{pmatrix}$$

$$2. \quad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = \text{COV} \left(\begin{matrix} X \end{matrix} \right)$$

$$3. \quad W_1 = \text{eigs} \left(\Sigma_{11}^{-1} \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}, K \right)$$

$$4. \quad Y_1 = (X_1 - \mu_1) \times W_1$$

When to use CCA?

- CCA applies for problems where data can be split into 2 views $X = [X_1, X_2]$
- CCA picks directions of projection (in each view) where data is maximally correlated
- Maximizes correlation coefficient and not just covariance so is **scale free**

When to use CCA

- Scenario 1: You have two feature extraction techniques.
 - One provides excellent features for dogs Vs cats and noise on other classes
 - Other method provides excellent features for cars Vs bikes and noise for other classes
- What do we do?
 - A. Use CCA to find one common representation
 - B. Concatenate the two features extracted

When to use CCA

- Scenario 2: You have two cameras capturing images of the same objects from different angles.
- You have a feature extraction technique that provides feature vectors from each camera.
- You want to extract good features for recognizing the object from the two cameras
- What do we do?
 - A. Use CCA to find one common representation
 - B. Concatenate features provides excellent features for

PICK A RANDOM W

$$Y = X \times \left[\begin{array}{ccc} +1 & \dots & -1 \\ -1 & \dots & +1 \\ +1 & \dots & -1 \\ & \cdot & \\ & \cdot & \\ & \cdot & \\ +1 & \dots & -1 \\ & K & \end{array} \right] \Big/ \sqrt{K}$$

When to use RP?

- When data is huge and very large dimensional
- For PCA, CCA typically you think of K (no. of dimensions we reduce to) in double digits
- For RP think of K typically in 3-4 digit numbers
- RP guarantees preservation of inter-point distances.
 - RP unlike PCA and CCA does not project using unit vectors. (What does this mean?)

KERNEL PCA

1. \tilde{K} $n = \text{compute} \left(k, X \right)$

2. $\left[A, \lambda \right] = \text{eigs} \left(\tilde{K}, K \right)$

3. $P = \begin{bmatrix} \vdots & \vdots \\ \frac{A_1}{\sqrt{\lambda_1}} & \frac{A_K}{\sqrt{\lambda_K}} \\ \vdots & \vdots \end{bmatrix}$

KERNEL PCA

4. $Y = \tilde{K} \times P$

When to use Kernel PCA

- When data lies on some non-linear, low dimensional subspace
- Kernel function matters. (Eg. RBF kernel, only points close to a given point have non-negligible kernel evaluation)

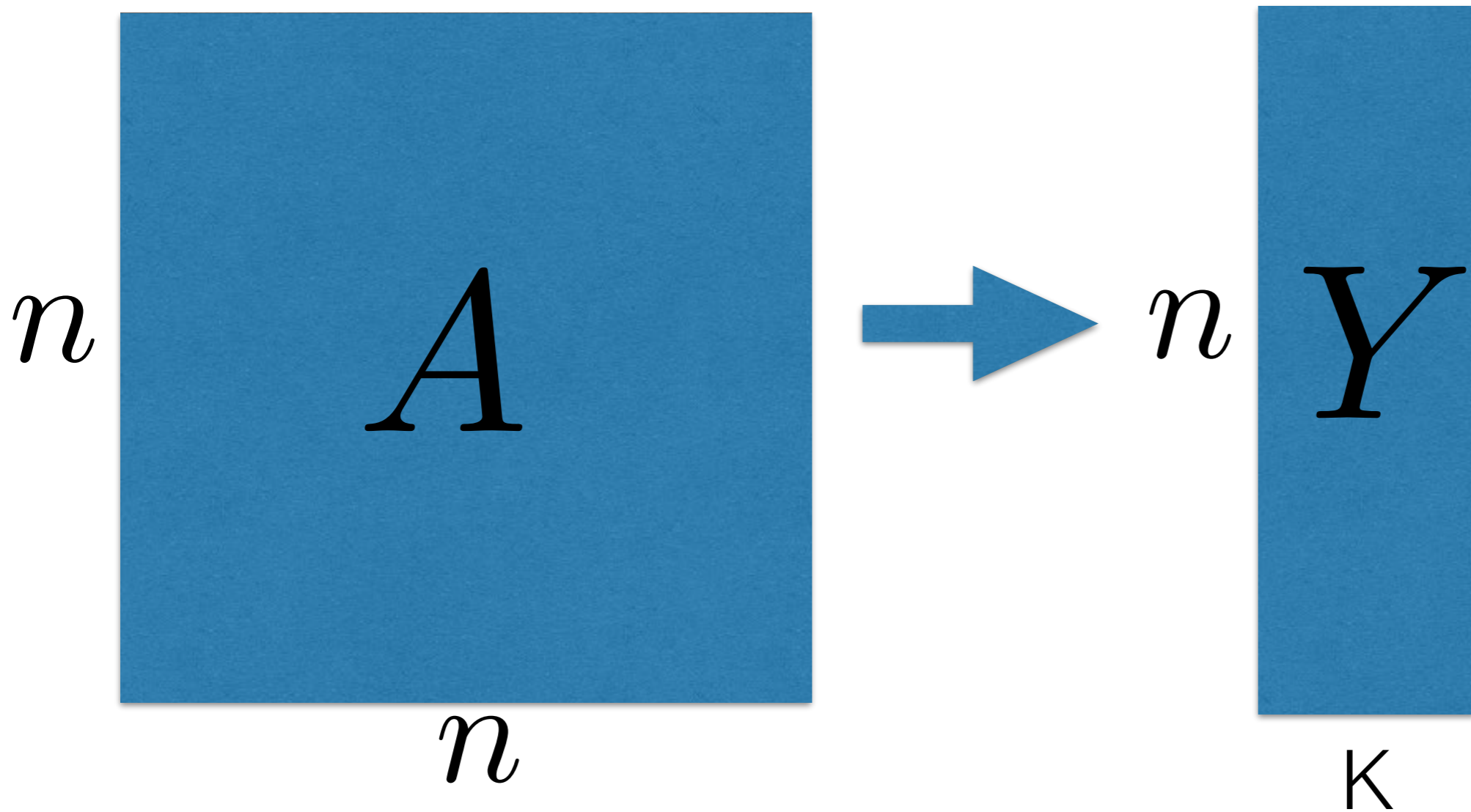
Spectral Clustering

- You want to cluster nodes of a graph into groups based on connectivity
- Unnormalized spectral clustering: divide into groups where as few edges between groups are cut

SPECTRAL CLUSTERING ALGORITHM (UNNORMALIZED)

- 1 Given matrix A calculate diagonal matrix D s.t. $D_{i,i} = \sum_{j=1}^n A_{i,j}$
- 2 Calculate the Laplacian matrix $L = D - A$
- 3 Find eigen vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of L (ascending order of eigenvalues)
- 4 Pick the K eigenvectors with smallest eigenvalues to get $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$
- 5 Use K-means clustering algorithm on $\mathbf{y}_1, \dots, \mathbf{y}_n$

Spectral Embedding



Use K-means on Y

Normalized Spectral Clustering

- Unnormalized spectral embedding encourages loner nodes to be pushed far away from rest
- This is indeed the min-cut solution to cut off loners
- Instead form clusters that minimize ratio of edges cut to number of edges each cluster has
 - (busy groups tend to form clusters)
- Algorithm, replace Laplacian matrix by normalized one

SPECTRAL CLUSTERING ALGORITHM (NORMALIZED)

- 1 Given matrix A calculate diagonal matrix D s.t. $D_{i,i} = \sum_{j=1}^n A_{i,j}$
- 2 Calculate the normalized Laplacian matrix $\tilde{L} = I - D^{-1/2}AD^{-1/2}$
- 3 Find eigen vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of \tilde{L} (ascending order of eigenvalues)
- 4 Pick the K eigenvectors with smallest eigenvalues to get $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$
- 5 Use K-means clustering algorithm on $\mathbf{y}_1, \dots, \mathbf{y}_n$

When to use Spectral Clustering

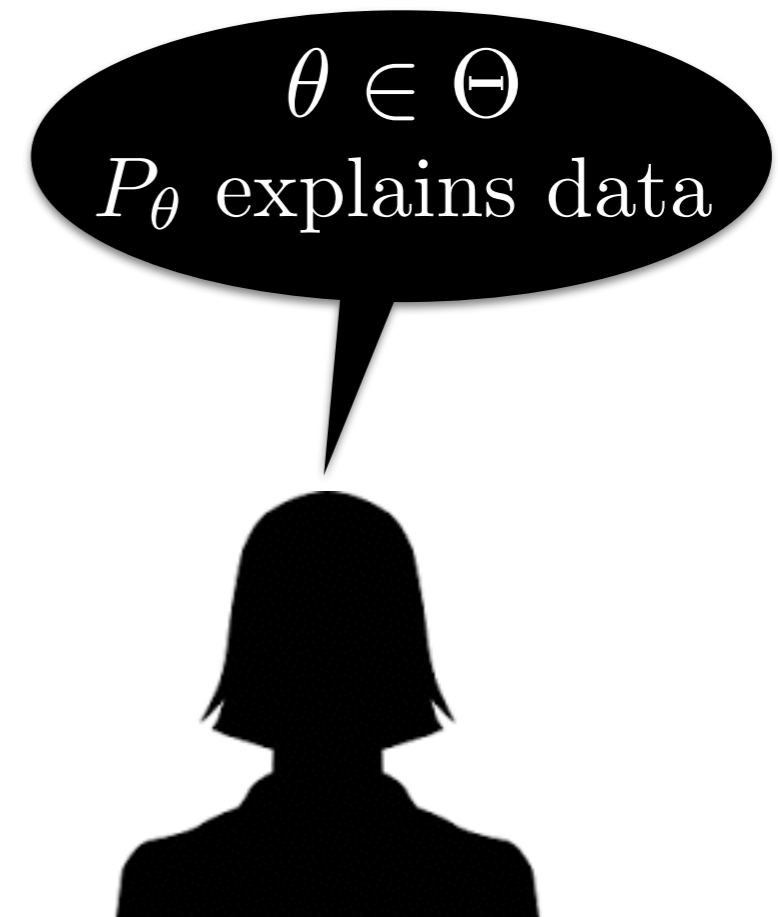
- First, even works with weighted graph, where weight of edge represents similarity
- When knowledge about how clusters should be formed is solely decided by similarity between points, there is no underlying prior knowledge

Probabilistic Modeling

PROBABILISTIC MODEL

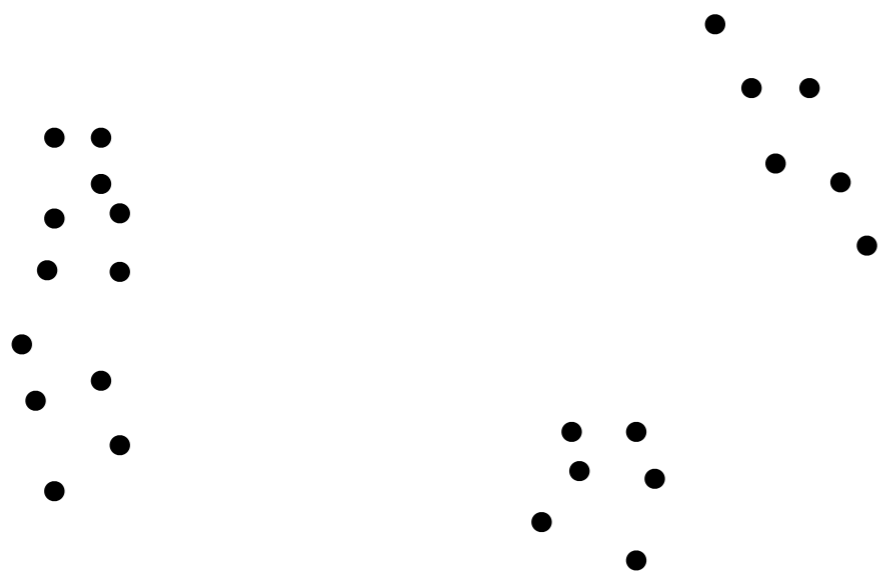
Data: $\mathbf{x}_1, \dots, \mathbf{x}_n$

PROBABILISTIC MODEL

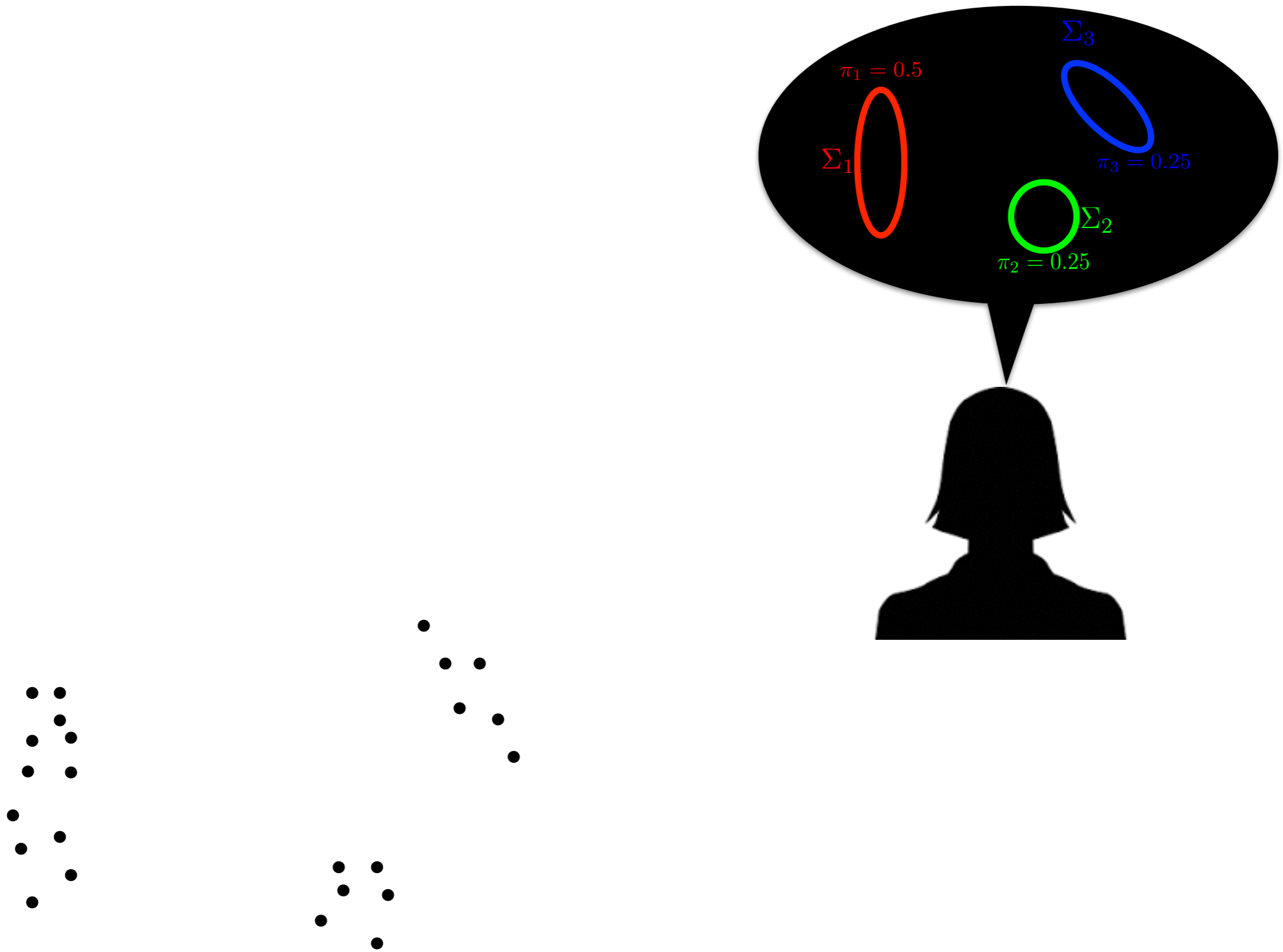


Data: $\mathbf{x}_1, \dots, \mathbf{x}_n$

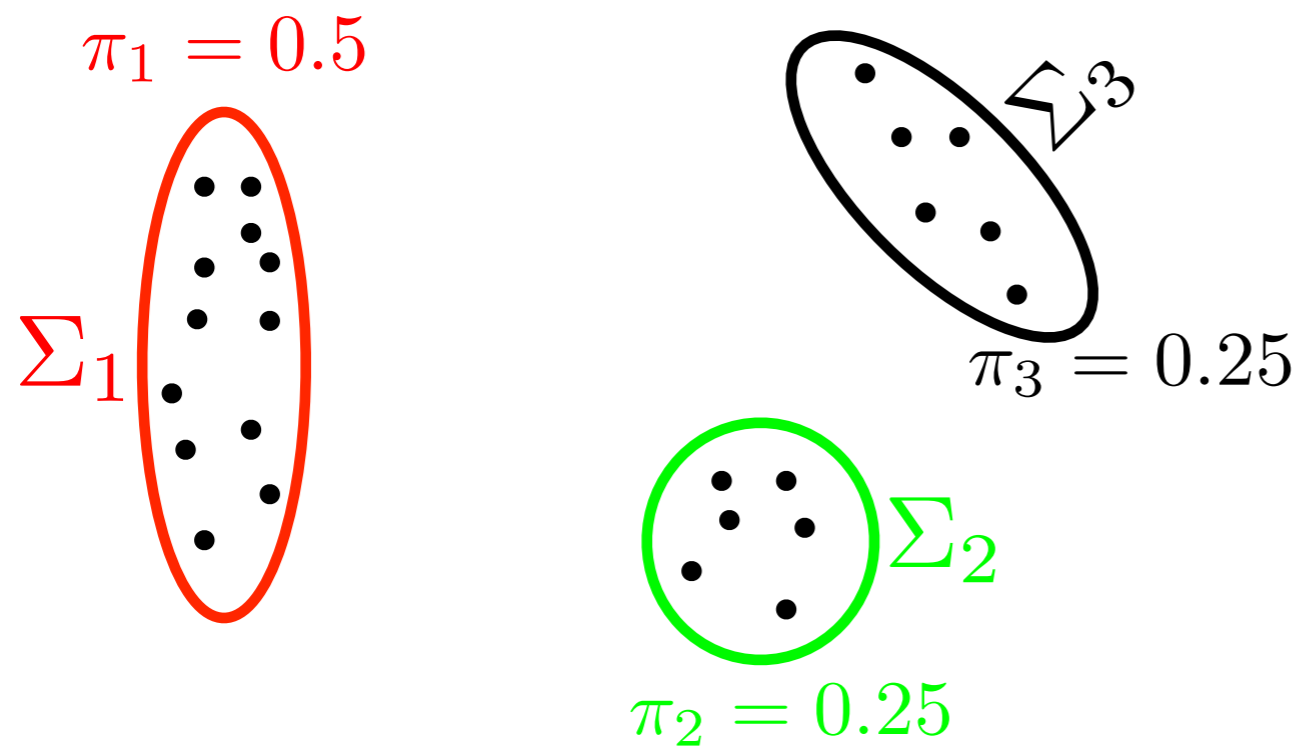
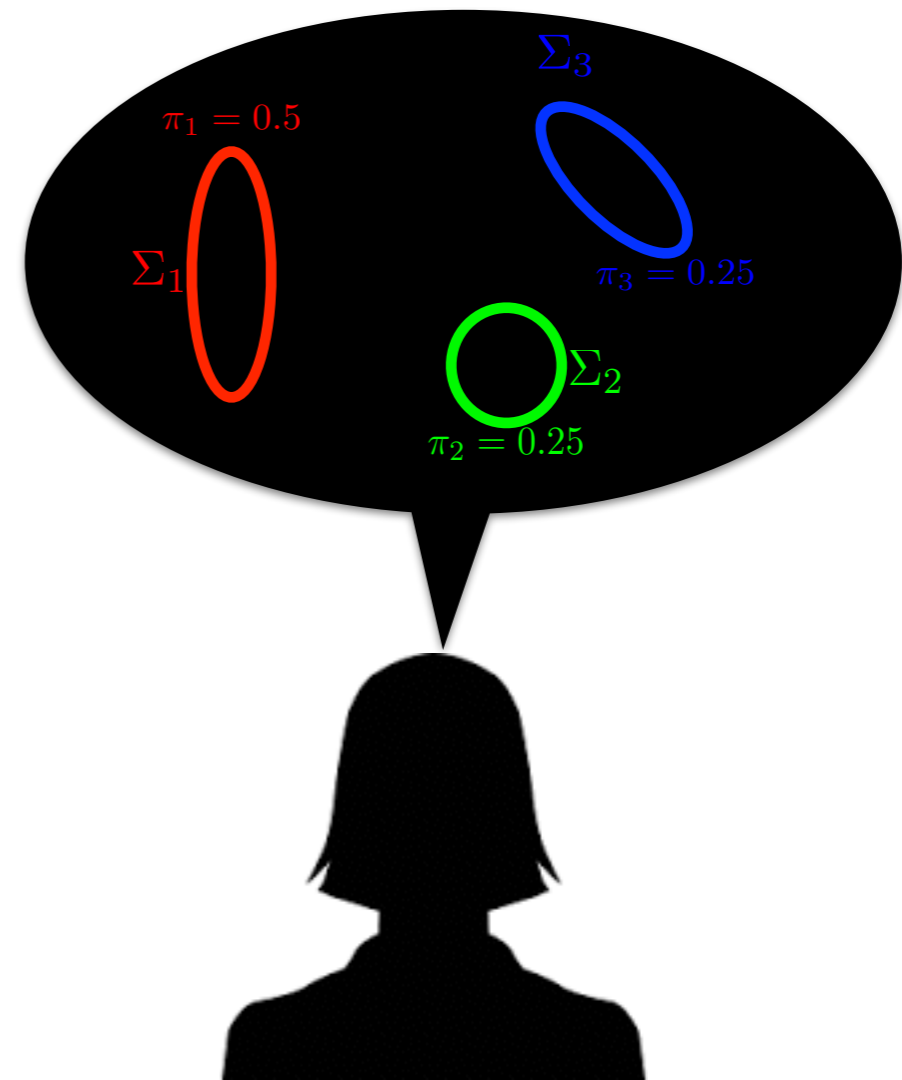
PROBABILISTIC MODEL



PROBABILISTIC MODEL



PROBABILISTIC MODEL



PROBABILISTIC MODELS

- Set of models Θ consists of parameters s.t. P_θ for each $\theta \in \Theta$ is a distribution over data.
- Learning: Estimate $\theta^* \in \Theta$ that best models given data

MAXIMUM LIKELIHOOD PRINCIPAL

Pick $\theta \in \Theta$ that maximizes probability of observation

MAXIMUM LIKELIHOOD PRINCIPAL

Pick $\theta \in \Theta$ that maximizes probability of observation

Reasoning:

- One of the models in Θ is the correct one

MAXIMUM LIKELIHOOD PRINCIPAL

Pick $\theta \in \Theta$ that maximizes probability of observation

Reasoning:

- One of the models in Θ is the correct one
- Given data we pick the one that best explains the observed data

MAXIMUM LIKELIHOOD PRINCIPAL

Pick $\theta \in \Theta$ that maximizes probability of observation

Reasoning:

- One of the models in Θ is the correct one
- Given data we pick the one that best explains the observed data
- Equivalently pick the maximum likelihood estimator,

$$\theta_{MLE} = \operatorname{argmax}_{\theta \in \Theta} \log P_{\theta}(x_1, \dots, x_n)$$

MAXIMUM LIKELIHOOD PRINCIPAL

Pick $\theta \in \Theta$ that maximizes probability of observation

Reasoning:

- One of the models in Θ is the correct one
- Given data we pick the one that best explains the observed data
- Equivalently pick the maximum likelihood estimator,

$$\theta_{MLE} = \operatorname{argmax}_{\theta \in \Theta} \log P_{\theta}(x_1, \dots, x_n)$$

Often referred to as frequentist view

MAXIMUM LIKELIHOOD PRINCIPAL

Pick $\theta \in \Theta$ that maximizes probability of observation

$$\theta_{MLE} = \operatorname{argmax}_{\theta \in \Theta} \underbrace{\log P_{\theta}(x_1, \dots, x_n)}_{\text{Likelihood}}$$

- A priori all models are equally good, data could have been generated by any one of them

MAXIMUM A POSTERIORI

Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

MAXIMUM A POSTERIORI

Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:

- Models are abstractions that capture our belief

MAXIMUM A POSTERIORI

Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:

- Models are abstractions that capture our belief
- We update our belief based on observed data

MAXIMUM A POSTERIORI

Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:

- Models are abstractions that capture our belief
- We update our belief based on observed data
- Given data we pick the model that we believe the most

MAXIMUM A POSTERIORI

Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:

- Models are abstractions that capture our belief
- We update our belief based on observed data
- Given data we pick the model that we believe the most
- Pick θ that maximizes $\log P(\theta|x_1, \dots, x_n)$

MAXIMUM A POSTERIORI

Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:

- Models are abstractions that capture our belief
- We update our belief based on observed data
- Given data we pick the model that we believe the most
- Pick θ that maximizes $\log P(\theta|x_1, \dots, x_n)$

I want to say : Often referred to as Bayesian view

MAXIMUM A POSTERIORI

Say you had a prior belief about models provided by $P(\theta)$

Pick $\theta \in \Theta$ that is most likely given data

Reasoning:

- Models are abstractions that capture our belief
- We update our belief based on observed data
- Given data we pick the model that we believe the most
- Pick θ that maximizes $\log P(\theta|x_1, \dots, x_n)$

I want to say : Often referred to as Bayesian view

There are Bayesian and there Bayesians

MAXIMUM A POSTERIORI

Pick $\theta \in \Theta$ that is most likely given data

Maximize a posteriori probability of model given data

$$\theta_{MAP} = \operatorname{argmax}_{\theta \in \Theta} P(\theta | x_1, \dots, x_n)$$

MAXIMUM A POSTERIORI

Pick $\theta \in \Theta$ that is most likely given data

Maximize a posteriori probability of model given data

$$\begin{aligned}\theta_{MAP} &= \operatorname{argmax}_{\theta \in \Theta} P(\theta | x_1, \dots, x_n) \\ &= \operatorname{argmax}_{\theta \in \Theta} \frac{P(x_1, \dots, x_n | \theta) P(\theta)}{P(x_1, \dots, x_n)}\end{aligned}$$

MAXIMUM A POSTERIORI

Pick $\theta \in \Theta$ that is most likely given data

Maximize a posteriori probability of model given data

$$\theta_{MAP} = \operatorname{argmax}_{\theta \in \Theta} P(\theta | x_1, \dots, x_n)$$

$$= \operatorname{argmax}_{\theta \in \Theta} \frac{P(x_1, \dots, x_n | \theta) P(\theta)}{P(x_1, \dots, x_n)}$$

$$= \operatorname{argmax}_{\theta \in \Theta} \underbrace{P(x_1, \dots, x_n | \theta)}_{\text{likelihood}} \underbrace{P(\theta)}_{\text{prior}}$$

MAXIMUM A POSTERIORI

Pick $\theta \in \Theta$ that is most likely given data

Maximize a posteriori probability of model given data

$$\theta_{MAP} = \operatorname{argmax}_{\theta \in \Theta} P(\theta | x_1, \dots, x_n)$$

$$= \operatorname{argmax}_{\theta \in \Theta} \frac{P(x_1, \dots, x_n | \theta) P(\theta)}{P(x_1, \dots, x_n)}$$

$$= \operatorname{argmax}_{\theta \in \Theta} \underbrace{P(x_1, \dots, x_n | \theta)}_{\text{likelihood}} \underbrace{P(\theta)}_{\text{prior}}$$

$$= \operatorname{argmax}_{\theta \in \Theta} \log P(x_1, \dots, x_n | \theta) + \log P(\theta)$$

THE BAYESIAN CHOICE

Don't pick any $\theta^* \in \Theta$

- Model is simply an abstraction
- We have a prosteriori distribution over models, why pick one θ ?

$$P(X|\text{data}) = \sum_{\theta \in \Theta} P(X, \theta|\text{data}) = \sum_{\theta \in \Theta} P(X|\theta)P(\theta|\text{data})$$