https://github.com/tedinburgh/ads2023

# GMMs and spectral clustering

**Tom Edinburgh**
**te269**

# Today: GMMs and spectral clustering

- Self-organising maps
- Gaussian mixture models
- Spectral clustering

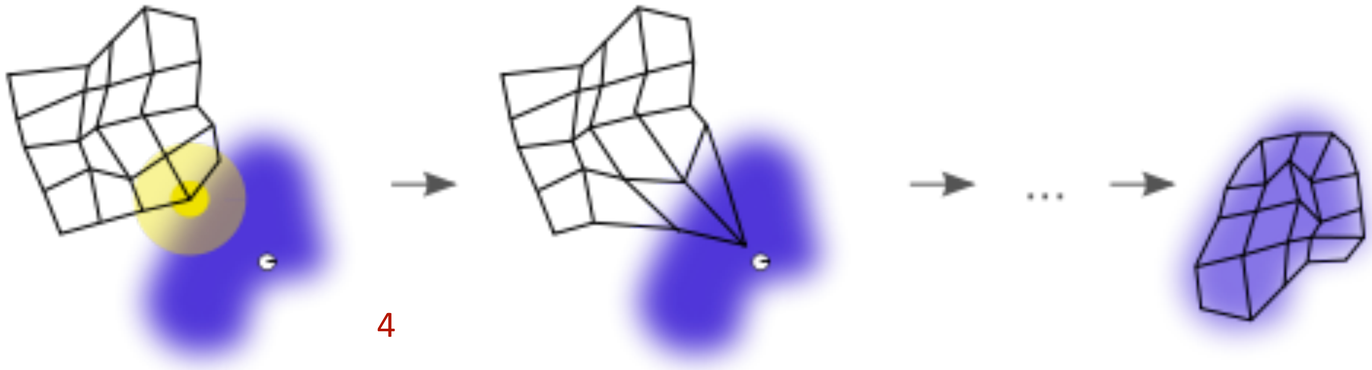
Questions: halfway through, at the end, or by email (te269)

# Resources

- Slides adapted from:
  - Ethan Fetaya/James Lucas/Emad Andrews, Toronto
  - Andrew Ng, Stanford
  - Thomas Sauerwald, Cambridge

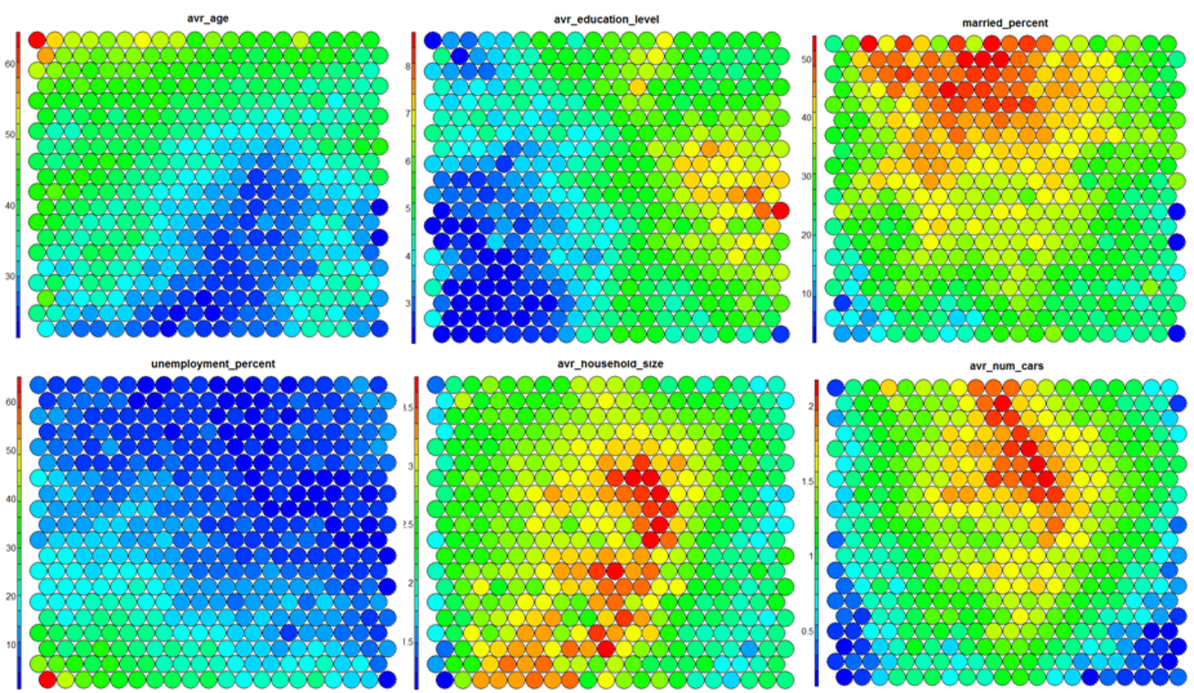
# Self-organising maps

- This is a dimensionality reduction technique, rather than clustering
- These are neural networks with nodes arranged in a grid pattern, each has a weight (the position of the node in the feature space)
- The SOM is distorts over the data manifold, while preserving the grid structure
- Training is competitive, the most similar node to a training input observation is called the best matching unit (BMU)
- The BMU and it's neighbours are pulled towards this input



# Self-organising maps

- After training, each node represents a variable-size region in the input space (i
- Nearby nodes within the grid represent nearby regions
- Data features can be overlayed onto the grid structure to visualise show groups of observations that have similar values of that feature
- We can also perform clustering (e.g. k-means or hierarchical clustering) on the nodes
- SOMs are also called Kohonen maps



# Generative modelling

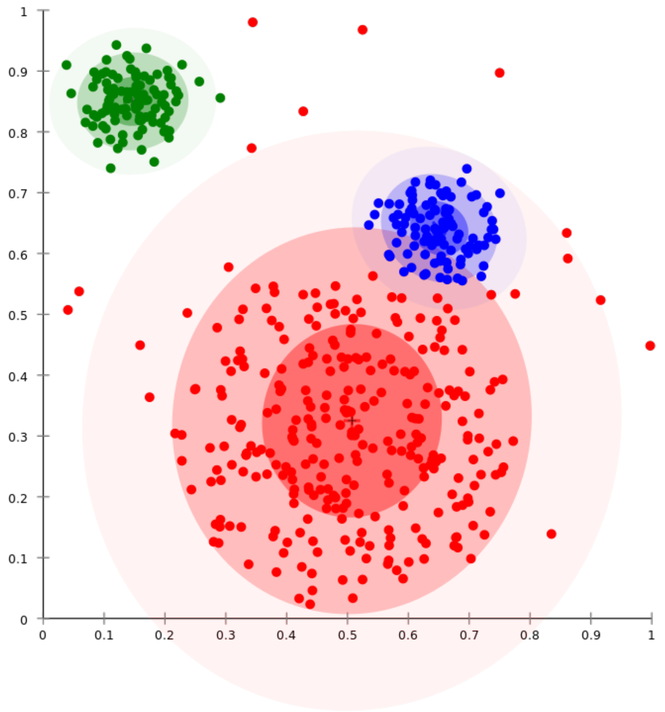
- We assume the data comes from a (probabilistic) generative model
- How do we learn what this generative model is?
- We could assume that the data comes from a known distribution, with parameters $\theta$, i.e. $x \sim P_\theta$

# Generative modelling

- We assume the data comes from a (probabilistic) generative model
- How do we learn what this generative model is?
- We could assume that the data comes from a known distribution, with parameters $\theta$, i.e. $x \sim P_\theta$
- Then we only need to learn the parameters $\theta$, i.e. guess starting $\theta$ and then adjust it to maximise the probability that the generative model produces the observed data
- This sounds a bit like maximum likelihood estimation (or a Bayesian posterior)

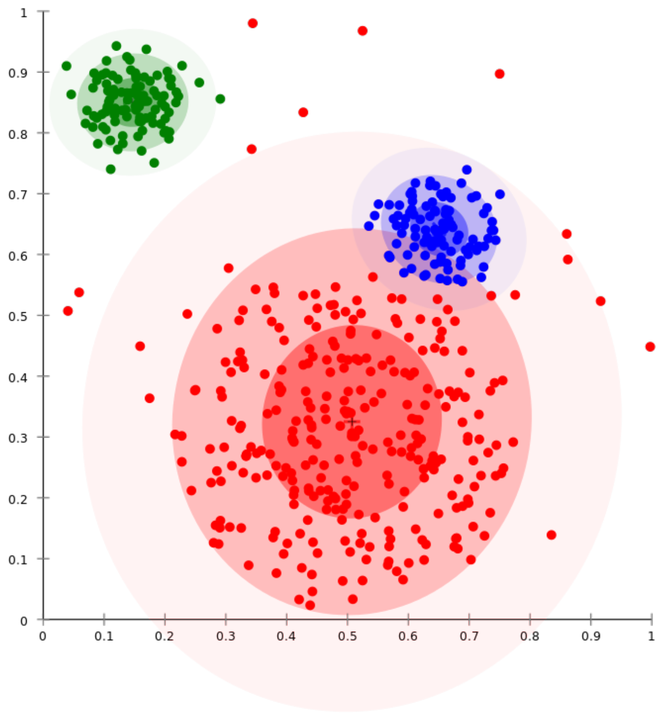
# Distribution-based clustering

- Now we assume the data comes from a number of different classes
- What if we modelled each of the (unknown) data-generating classes using different known distributions?
- Each point is space is then modelled as a mixture of these distributions
- What are the pros and cons?



# Cluster membership in distribution-based clustering

- Each point is space is then modelled as a mixture of these distributions
- But each observations also has probability under each of the distributions (which gives soft cluster membership)
- We can easily construct hard clusters by assigning an observation to distribution under which it has the highest probability



# Latent variable models

- We could model the joint distribution of an observation $x$ and its class $z$ as $p(x, z \mid \theta) = p(x \mid z, \theta)p(z \mid \theta)$, for some model parameters $\theta$
- But we don't know the class labels $z$, so instead we use a mixture model $p(x) = \sum_z p(x, z) = \sum_z p(x \mid z)p(z)$
- (dropping $\theta$ temporarily, for readability)
- This is called a **latent variable model (LVM)**, and $z$ is called a discrete-valued **latent variable** (latent means hidden, we can infer $z$ but we don't observe it directly)

# Distribution-based clustering

- If we assume there are $K$ classes, then $p(x) = \sum_{k=1}^{K} p(x \mid z = k)p(z = k)$
- So we need to specify $K$ distributions $p_1(x) = p(x \mid z = 1), \ldots, p_K(x) = p(x \mid z = K)$, plus another distribution over the discrete-valued latent variable $p(z = k)$
- Divide-and-conquer approach: use simple parts to build complex models

# Gaussian mixture models

- One simple solution is to use a fixed number of Gaussian distributions
- This is then called a Gaussian mixture model (GMM), defined as:

  $p(x \mid \theta) = \sum_{k=1}^{K} \phi_k f(x \mid \mu_k, \Sigma_k)$, where $f$ is the density of $N(\mu_k, \Sigma_k)$

- The full parameters for this GMM are $\theta = (\mu_1, \Sigma_1, \phi_1, \ldots, \mu_K, \Sigma_K, \phi_K)$
- $\phi_k$ are mixture weights or mixing coefficients, with $\phi_k \geq 0, \ \sum_{k=1}^{K} \phi_k = 1$

# GMMs and LVMs

- The GMM is $p(x \mid \theta) = \sum_{k=1}^{K} \phi_k f(x \mid \mu_k, \Sigma_k)$, where $f$ is the density of $N(\mu_k, \Sigma_k)$
- How does this relate to LVM

$$p(x \mid \theta) = \sum_{k=1}^{K} p(x \mid z = k, \theta) p(z = k \mid \theta) = \sum_{k=1}^{K} p_k(x \mid \theta) p(z = k \mid \theta)?$$

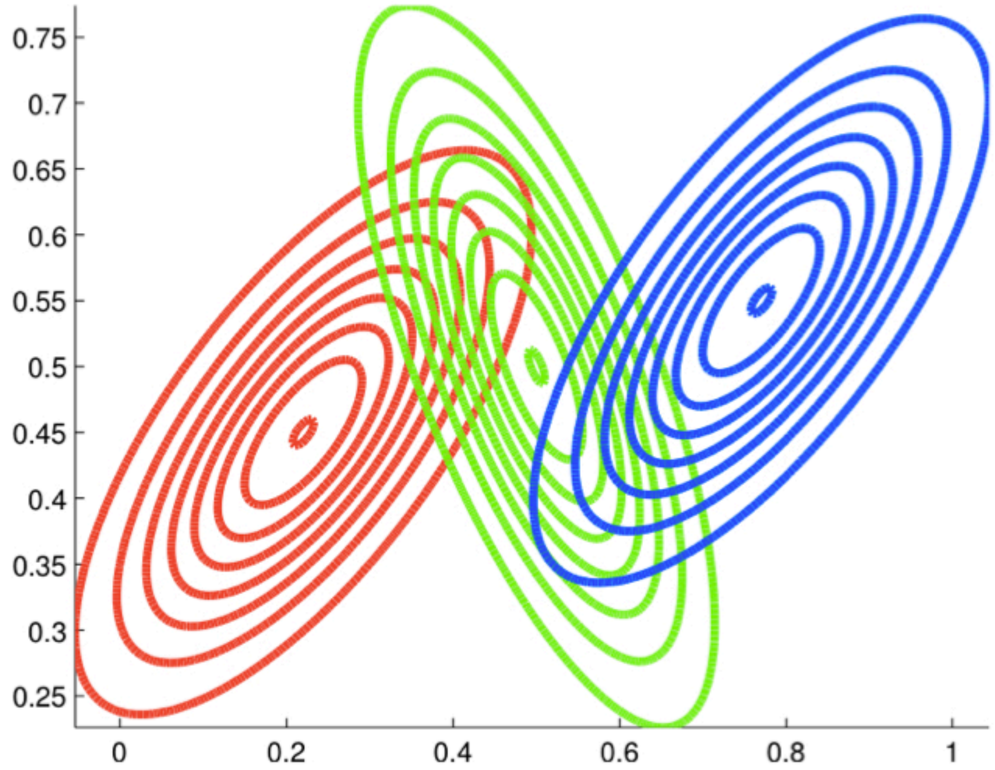
# GMMs and LVMs

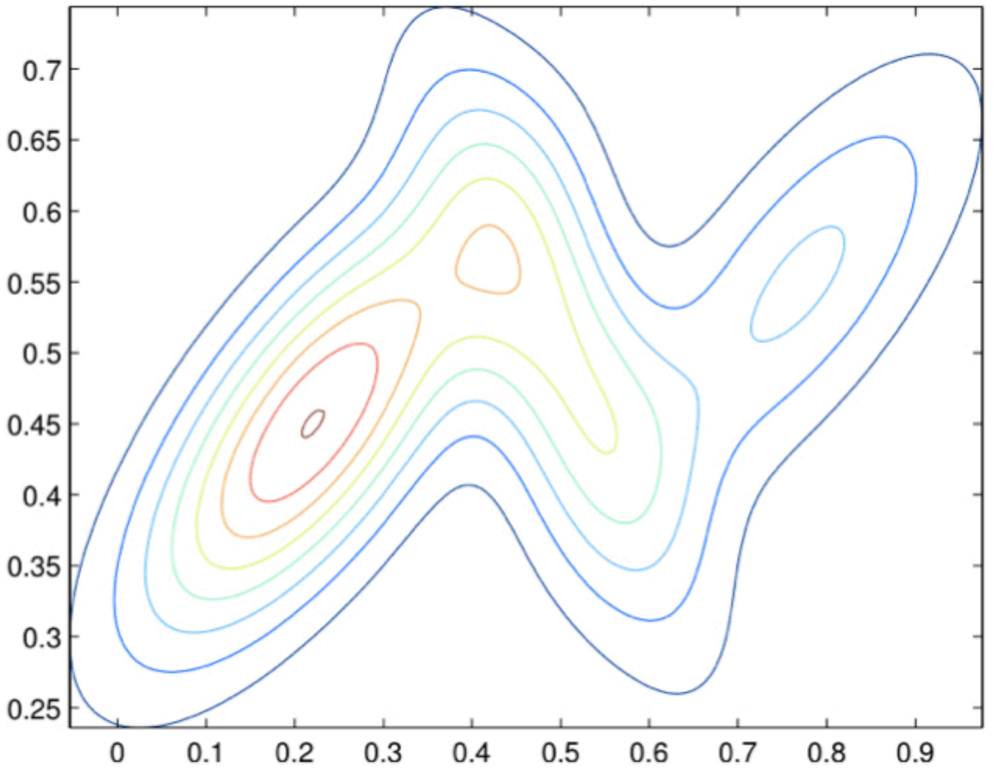
- The GMM is $p(x \mid \theta) = \sum_{k=1}^{K} \phi_k f(x \mid \mu_k, \Sigma_k)$, where $f$ is the density of $N(\mu_k, \Sigma_k)$
- How does this relate to the LVM

  $p(x \mid \theta) = \sum_{k=1}^{K} p(x \mid z = k, \theta) p(z = k \mid \theta) = \sum_{k=1}^{K} p_k(x \mid \theta) p(z = k \mid \theta)$?
- Suppose distribution $p_k(x \mid \theta)$ is a Gaussian with mean $\mu_k$ and covariance $\Sigma_k$
- Also suppose that $z \sim$ Categorical$((\phi_1, \ldots, \phi_K))$, then $p(z = k \mid \theta) = \phi_k$
- Then we get the GMM, $p(x \mid \theta) = \sum_{k=1}^{K} \phi_k f(x \mid \mu_k, \Sigma_k)$

# Gaussian mixture models

- GMMs are **universal approximations of densities** (if you have enough diagonal Gaussians, you can model the data very well) so they are powerful



# Gaussian mixture models

- GMMs are **universal approximations of densities** (if you have enough diagonal Gaussians, you can model the data very well) so they are powerful
- But they are hard to optimise
- In general, assuming data is normally-distributed is quite a strong assumption
- Overfitting may be a problem (without some constraints on model complexity)

Small $K$ → Large $K$

Assumption not valid? Overfitting?

# Maximum likelihood estimation

- We want to find an optimal value of $\theta = (\mu_1, \Sigma_1, \phi_1, \ldots, \mu_K, \Sigma_K, \phi_K)$
- We want to perform maximum likelihood estimation for parameters of a statistical model, but the model depends on unobserved latent variables
- Generally, maximum likelihood estimation involves solving a set of differential equations (e.g. take the derivative of the likelihood with respect to every unknown and solve)
- With latent variables this becomes difficult

# Expectation-maximisation algorithm

- To do this, we use a two-step iterative method called expectation-maximisation
- Basic idea: two sets of unknowns (model parameters $\theta$ and latent variables $z$)
  - Pick arbitrary values for one set of unknowns and use them to estimate the other
  - Use these estimates for the second set of unknowns to find a better estimate for the first set, then use those to improve the estimates of the first second set
  - Keep alternating and repeat until there is convergence

# Expectation-maximisation algorithm

- This process is very similar to the Lloyd's algorithm for k-means clustering
- We can show that this converges to a fixed point but there is no guarantee that this is a global optimum
- Note: it is possible that the solution has a **singularity** e.g. one of the Gaussians is a spike at one of the observations (zero variance), which is not good

# GMM maximum likelihood estimation

- The log-likelihood in the GMM model is

$$L(\theta; x, z) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} p(x_i, z_i = k \mid \theta)$$

$$= \sum_{i=1}^{n} \log \sum_{k=1}^{K} p(x_i \mid \mu_k, \Sigma_k) p(z_i = k \mid \phi_k)$$

- If we knew the latent variable $z_i$ for each observation $x_i$, then this is fairly straightforward ($1_A$ is the indicator function):

$$\mu_k = \frac{\sum_{i=1}^{n} 1_{\{z_i=k\}} x_i}{\sum_{i=1}^{n} 1_{\{z_i=k\}}}, \Sigma_k = \frac{\sum_{i=1}^{n} 1_{\{z_i=k\}} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{n} 1_{\{z_i=k\}}}, \phi_k = \frac{1}{n} \sum_{i=1}^{n} 1_{\{z_i=k\}}$$

# Expectation-maximisation algorithm for GMM

- E-step: compute the posterior probability over $z$ for the current model (how much do we think each Gaussian generates each observation)
- M-step: for the current latent classes, change the parameters of each Gaussian to maximise the probability that it would generate the data it is currently responsible for

# Basic idea: Jensen's inequality

- Problem: $l(\theta) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} p(x_i, z_i = k \mid \theta)$ is hard to optimise (because of the sum inside the log)
- Instead, introduce a new distribution $Q_i$ over $z_i$ with probabilities $q_{ik}$, so

$$l(\theta) = \sum_{i=1}^{n} \log \left( \sum_{k=1}^{K} q_{ik} \frac{p(x_i, z_i = k \mid \theta)}{q_{ik}} \right)$$

- Jensen's inequality (for convex functions like $\log(x)$) gives a lower bound:

$$\sum_{i=1}^{n} \log \left( \sum_{k=1}^{K} q_{ik} \frac{p(x_i, z_i = k \mid \theta)}{q_{ik}} \right) \geq \sum_{i=1}^{n} \sum_{k=1}^{K} q_{ik} \log \left( \frac{p(x_i, z_i = k \mid \theta)}{q_{ik}} \right)$$

# Basic idea: Jensen's inequality to EM

- Jensen's inequality (for convex functions like $\log(x)$) gives a lower bound:

$$\sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} q_{ik} \frac{p(x_i, z_i = k \mid \theta)}{q_{ik}}\right) \geq \sum_{i=1}^{n} \sum_{k=1}^{K} q_{ik} \log\left(\frac{p(x_i, z_i = k \mid \theta)}{q_{ik}}\right)$$

- Let's try to keep the lower-bound tight at the current estimate of $\theta$, which we denote $\theta^{(t)}$

- If we choose $q_{ik} = p(z_i = k \mid x_i, \theta^{(t)})$ i.e. the posterior distribution of $z_i$ given $x_i$ and the current parameters, then $p(x_i, z_i = k \mid \theta^{(t)})/q_{ik} = p(x_i \mid \theta^{(t)})$ doesn't depend on the latent variables! And Jensen's inequality holds with equality.

# Expectation-maximisation algorithm

- So the two steps of EM are:

Can view this as the responsibility of cluster $k$ towards $x_i$

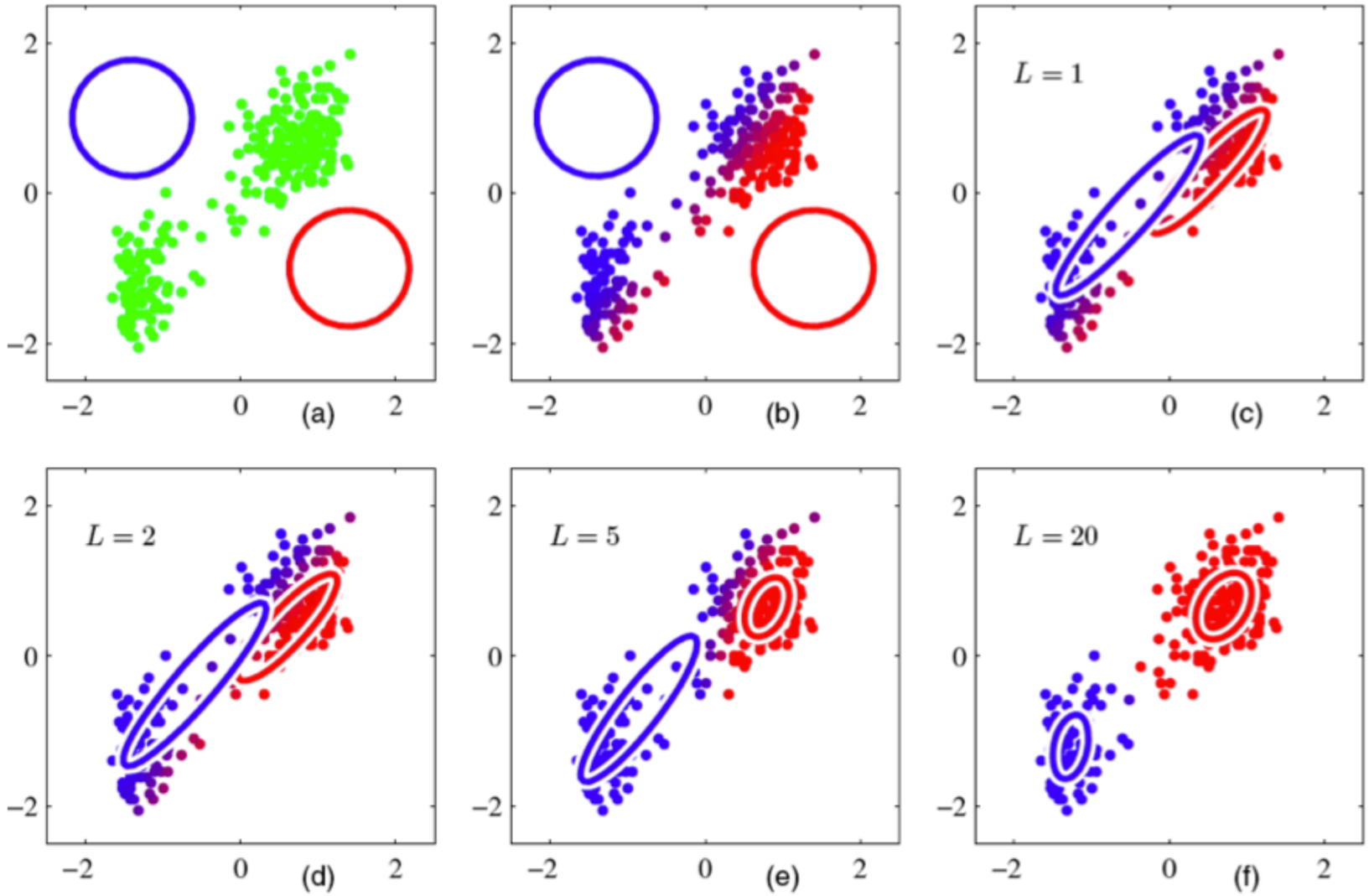
1. E-step: $q_{ik}^{(t)} = p(z_i = k \mid x_i, \theta^{(t)}) = \dfrac{\phi_k^{(t)} f(x_i \mid \mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_{j=1}^{K} \phi_j^{(t)} f(x_i \mid \mu_j^{(t)}, \Sigma_j^{(t)})}$

2. M-step: $\theta^{(t+1)} = \arg\max_\theta \sum_{i=1}^{n} \sum_{k=1}^{K} q_{ik}^{(t)} \log \left( \dfrac{p(x_i, z_i = k \mid \theta)}{q_{ik}^{(t)}} \right)$

This has a similar solution to earlier, just weighed differently

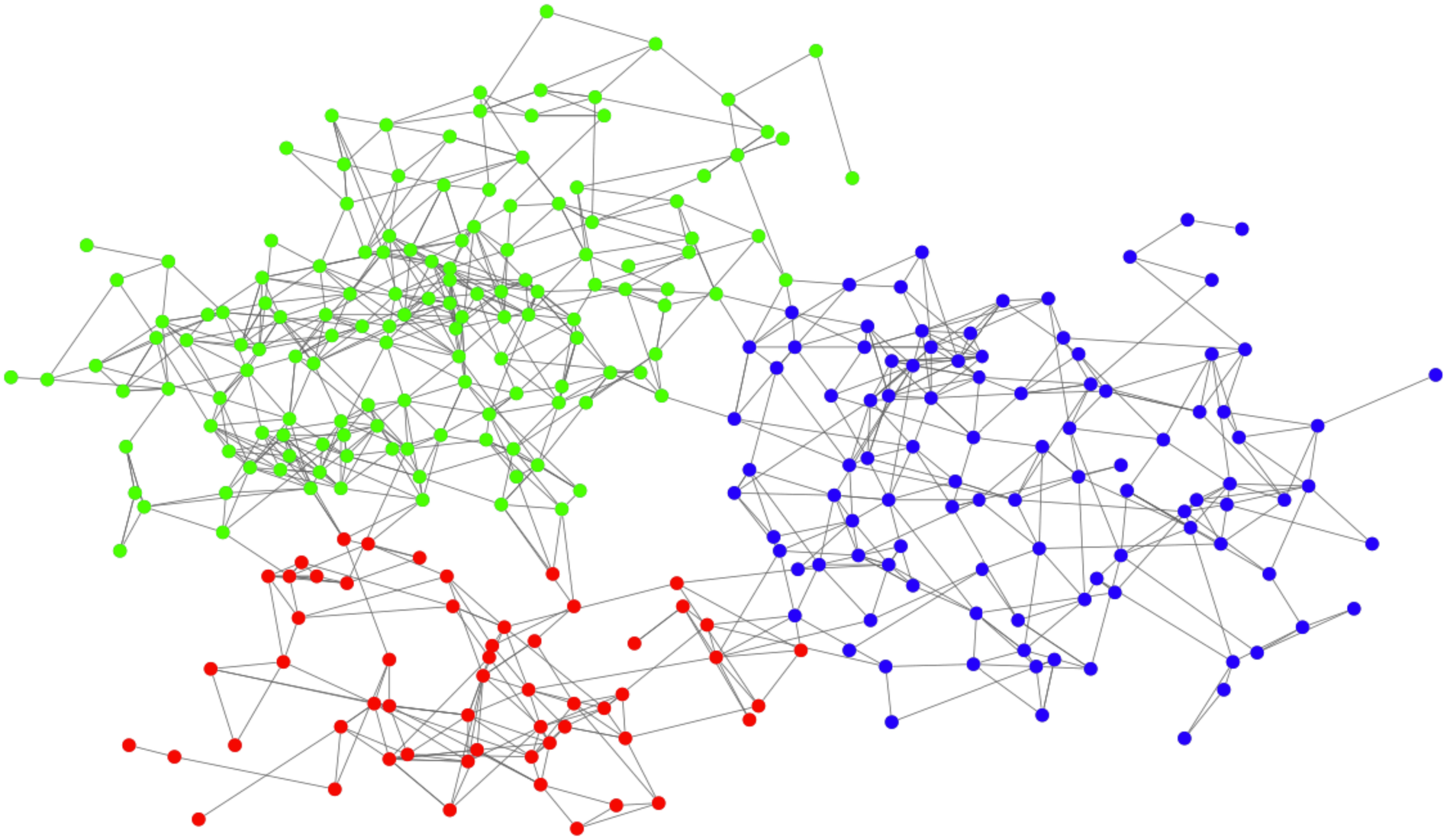
- We can show that $l(\theta^{(t+1)}) \geq l(\theta^{(t)})$, i.e. EM will monotonically improve the log-likelihood.

# Expectation-maximisation algorithm



# Questions?

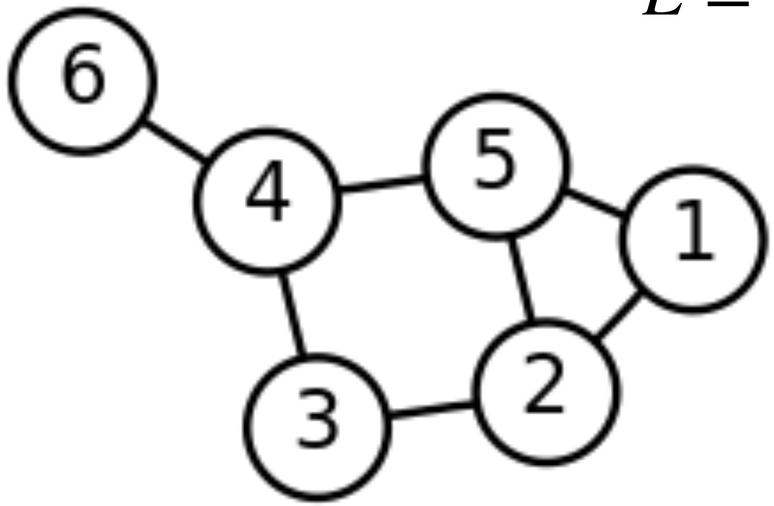
# Spectral clustering



# Spectral clustering

- Now think about clustering using graph networks (using eigenvectors)
- We denote the (undirected, unweighted) graph as an ordered pair $G = (V, E)$
- Observation are represented as vertices $V$, connected to other vertices by edges $E \subseteq \{\{x, y\} \mid x, y \in V,\ x \neq y\}$

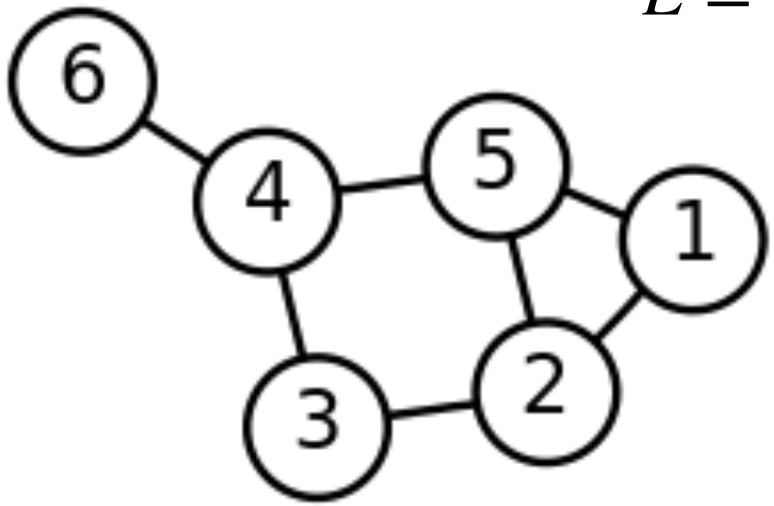
$$E = \{\{1,\ 2\}, \{1,\ 5\}, \{2,\ 5\}, \{2,\ 3\}, \{3,\ 4\}, \{4,\ 5\}, \{4,\ 6\}\}$$




$$V = \{1,\ 2,\ 3,\ 4,\ 5,\ 6\}$$

# Diagonal matrix

- The diagonal matrix $D$ of a graph $G = (V, E)$ is $D_{ij} = \begin{cases} \deg(i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$
- The degree of a vertex is the number of edges into/out of the vertex




$$E = \{\{1, 2\}, \{1, 5\}, \{2, 5\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{4, 6\}\}$$

$$V = \{1, 2, 3, 4, 5, 6\}$$

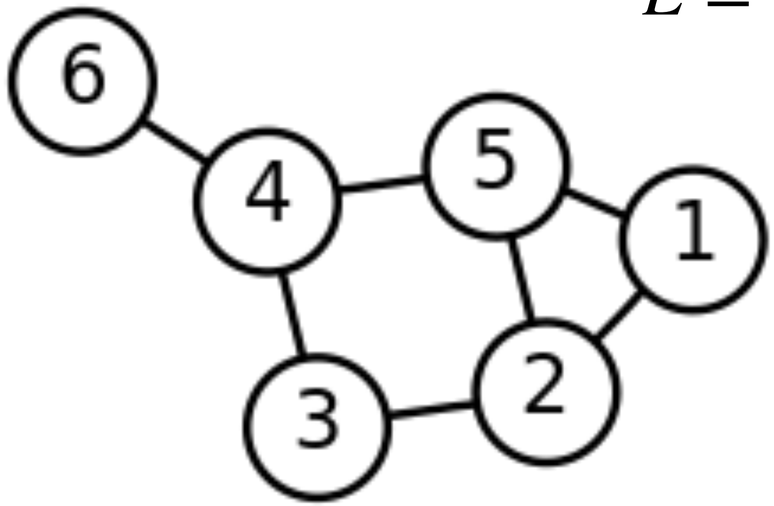
$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

# Adjacency matrix

Can do a similar thing for weighted graphs

- The adjacency matrix $A$ of a graph $G = (V, E)$ is $A_{ij} = \begin{cases} 1 & \text{if } \{i, j\} \in E \\ 0 & \text{otherwise} \end{cases}$
- For an (undirected) graph with $m$ vertices, $A$ is a (symmetric) $m \times m$ matrix
- The sum of each row/column $i$ is the degree of the corresponding vertex $i$

$$E = \{\{1, 2\}, \{1, 5\}, \{2, 5\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{4, 6\}\}$$




$$V = \{1, 2, 3, 4, 5, 6\}$$

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

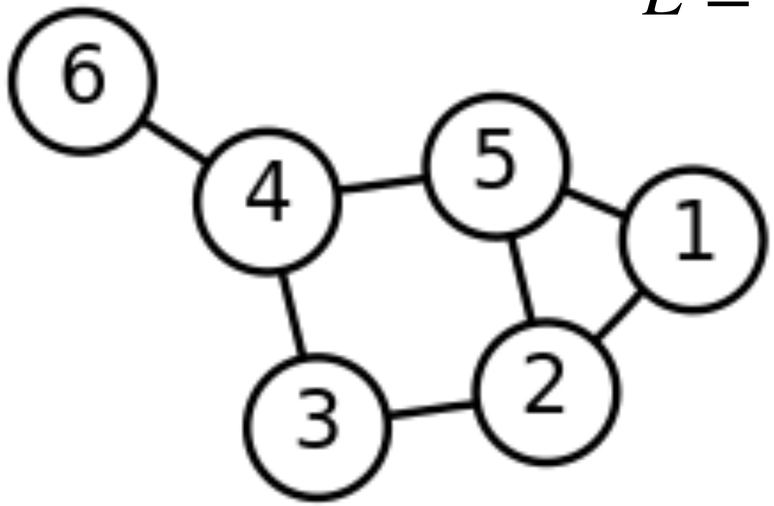
# Laplacian matrix

- The Laplacian matrix $L$ of a graph $G = (V, E)$ is $L = D - A$

Why?

- $L$ is positive semi-definite, so eigenvalues $\lambda_n \geq \lambda_{n-1} \geq \ldots \geq \lambda_2 \geq \lambda_1 = 0$
- Generally use the normalised Laplacian matrix (all diagonal entries equal to 1) instead, which is $L^{\text{norm}} = I - D^{-1/2} A D^{-1/2}$

$$E = \{\{1, 2\}, \{1, 5\}, \{2, 5\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{4, 6\}\}$$




$$V = \{1, 2, 3, 4, 5, 6\}$$

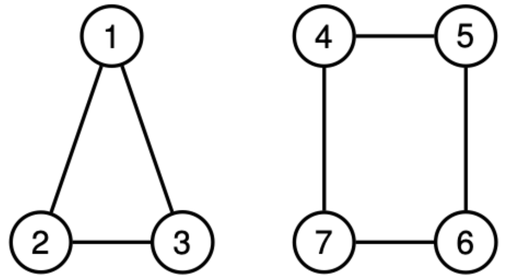
$$L = \begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

# Spectral clustering: basic algorithm

1. Calculate the normalised Laplacian $L^{\text{norm}} = I - D^{-1/2}AD^{-1/2}$
2. Calculate the eigenvalues and eigenvectors of $L^{\text{norm}}$
3. Form a matrix $V$ of $K$ eigenvectors corresponding to the $K$ smallest non-zero eigenvalues
4. $V$ is an $n \times K$ matrix, the $i^{\text{th}}$ row defines features of the network graph node $i$
5. Cluster the graph nodes based on these features, using e.g. k-means

- Why? Spectral properties of the graph contains information about clustering

# Spectral clustering: regular graphs




$\lambda_1 = \lambda_2 = 0$

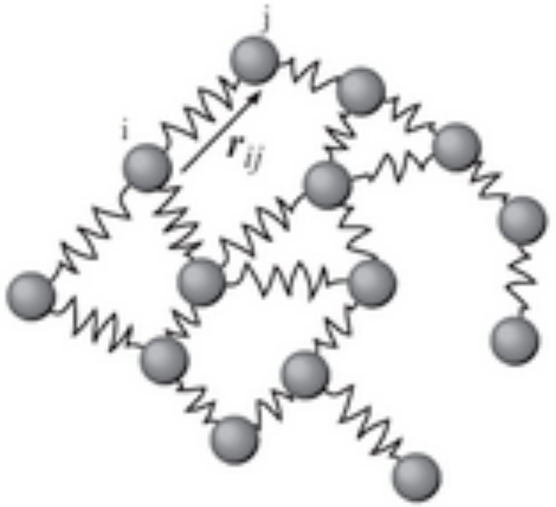
$v_1^T = (1, 1, 1, 0, 0, 0, 0)$

$v_2^T = (0, 0, 0, 1, 1, 1, 1)$

- In a regular graph, each vertex has the same degree ($d$-regular)
- E.g. vertices are connected to their k-nearest neighbours
- Then the normalised Laplacian is $L = I - (1/d)A$
- The Laplacian also defines a quadratic form $\forall x,\ x^T L x = 1/d \sum_{\{i,j\} \in E} (x_i - x_j)^2$
- If $\lambda$ and $v$ are an eigenvalue/eigenvector of $A$, then $Lv = v - (1/d)Av$ and $1 - \lambda/d$ and $v$ are an eigenvalue/eigenvector of $L$
- In this case, the eigenvectors are orthonormal, $\lambda_n \leq 2$ and $\lambda_n = 2$ if and only if there is a bipartite connected component (colour each node, all edges connect to both colours)

# Spectral clustering and mass-spring systems

- This is related to a problem from physics
  - Partitioning a mass-spring system, edge weights on graph are spring stiffness
  - Transversal vibration modes of a mass-spring system is the same as the eigenvalue problem for the Laplacian
  - Related to a generalisation of Hooke's law to higher-dimensions



# Questions?

- Feel free to email me at te269@cam.ac.uk

# Next time

- Clustering
  - Graph-based clustering
  - Density-based clustering
  - Outlier detection