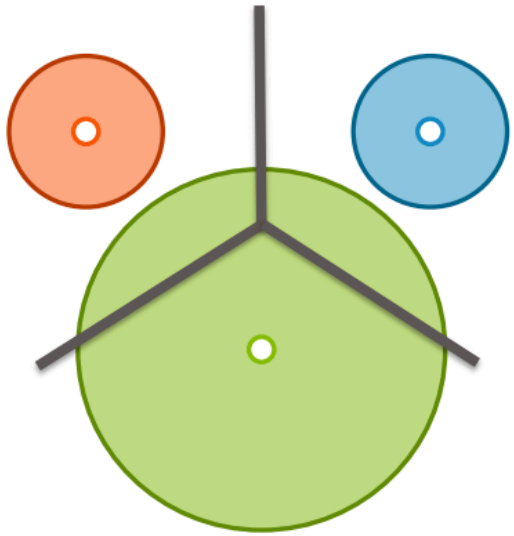
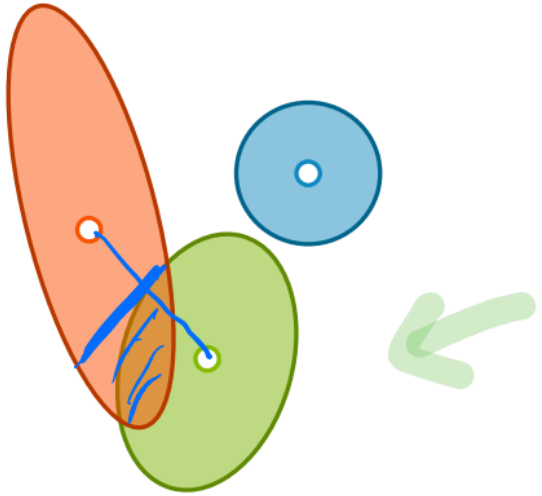
CSE 446
Gaussian Mixture Models (GMM)

Natasha Jaques

- K-means algorithm fails, when the data has:



disparate cluster sizes



different shaped/oriented clusters

- What can we do?

CSE 446

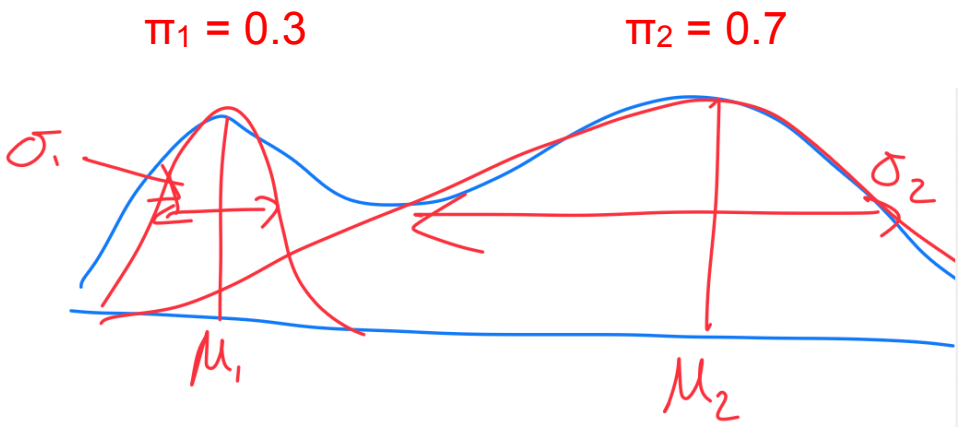
- Supervised learning
  - Linear models
    - Linear regression
    - Ridge regression
    - LASSO regression
    - Logistic regression
  - Non-parametric & non-linear
    - Nearest neighbors
    - Trees & Random forests
    - Boosting
    - Kernel methods
  - Neural networks
    - Backpropagation
    - CNN
    - RNN/LSTM
    - Attention/Transformer

- Unsupervised learning
  - Clustering
    - k-means
    - **Gaussian Mixture Models (GMM)**
  - PCA/SVD

Gaussian Mixture Model

- input: data $\{x_i\}_{i=1}^n$ in $\mathbb{R}^d$
- parameters of a **Gaussian Mixture Model**
  - mixing weights:
    - $\pi_j = \mathbf{P}(\text{cluster membership} = j)$ for $j \in \{1,\ldots,K\}$
  - means:
    - $\mu_j \in \mathbb{R}^d$ for $j \in \{1,\ldots,K\}$
  - covariance matrices:
    - $\mathbf{\Sigma}_j \in \mathbb{R}^{d\times d}$ for $j \in \{1,\ldots,K\}$
- we suppose that the given data has been generated from a GMM, and try to find the best GMM parameters (this naturally will define clustering of the training data)

GMM params are σ, μ, π



Gaussian Mixture Model

- input: data $\{x_i\}_{i=1}^n$ in $\mathbb{R}^d$
- parameters of a **Gaussian Mixture Model**
  - mixing weights:
    - $\pi_j = \mathbf{P}(\text{cluster membership} = j)$ for $j \in \{1,\ldots,K\}$
  - means:
    - $\mu_j \in \mathbb{R}^d$ for $j \in \{1,\ldots,K\}$
  - covariance matrices:
    - $\mathbf{\Sigma}_j \in \mathbb{R}^{d\times d}$ for $j \in \{1,\ldots,K\}$
- we suppose that the given data has been generated from a GMM, and try to find the best GMM parameters (this naturally will define clustering of the training data)

- under the GMM, the i-th sample is drawn as follows
  - first sample a cluster $z_i \in \{1,\ldots,K\}$, from $\pi = [\pi_1, \ldots, \pi_K]$
  - conditioned on this cluster, x_i is sampled from

$$x_i \sim N(\mu_{z_i}, \mathbf{\Sigma}_{z_i})$$

demo: https://lukapopijac.github.io/gaussian-mixture-model/

Maximum likelihood estimation (MLE)

- we can find the best GMM for given data, by MLE
- for simplicity, suppose $d = 1$ and $K = 2$
- Model parameters are $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$
- the probability of observing a sample x_i can be written as

$$\mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2) = \underbrace{\pi_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i;\mu_1,\sigma_1^2)} + \underbrace{\pi_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i;\mu_2,\sigma_2^2)}$$

Maximum likelihood estimation (MLE)

- we can find the best GMM for given data, by MLE
- for simplicity, suppose $d = 1$ and $K = 2$
- Model parameters are $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$
- the probability of observing a sample x_i can be written as

$$\mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2) = \underbrace{\pi_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i;\mu_1,\sigma_1^2)} + \underbrace{\pi_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i;\mu_2,\sigma_2^2)}$$

- MLE tries to find

$$\arg \max_{\pi_1,\pi_2,\mu_1,\mu_2,\mathbf{C}_1,\mathbf{C}_2} \sum_{i=1}^{n} \log \mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2)$$

- however, unlike least squared or logistic regression, this is not a concave function of the parameters (thus hard to find the optimal solution)
- in general, MLE of a mixture model is not convex/concave optimization

Recall lecture 1: fitting a single Gaussian model

- given $\{x_i\}_{i=1}^n \in \mathbb{R}$,fit the best Gaussian model with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$
- using MLE we want to solve

$$\text{maximize}_{\mu,\sigma^2} \; \mathcal{L}(\mu,\sigma^2) = \sum_{i=1}^n \underbrace{\left(-\frac{(x_i-\mu)^2}{2\sigma^2} - \log\left(\sqrt{2\pi\sigma^2}\right)\right)}_{\log N(x_i|\mu,\sigma^2)}$$

- we compute gradient and set it to zero:
  - $$\nabla_\mu \mathcal{L}(\mu,\sigma^2) = \frac{1}{\sigma^2}\sum_{i=1}^n (\mu - x_i)$$

    which is zero for $\mu = \frac{1}{n}\sum_{i=1}^n x_i$

    (which makes sense as it is the empirical mean)

Recall lecture 1: fitting a single Gaussian model

- given $\{x_i\}_{i=1}^n \in \mathbb{R}$,fit the best Gaussian model with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$
- using MLE we want to solve

$$\text{maximize}_{\mu,\sigma^2} \; \mathcal{L}(\mu,\sigma^2) = \sum_{i=1}^n \underbrace{\left(-\frac{(x_i-\mu)^2}{2\sigma^2} - \log\left(\sqrt{2\pi\sigma^2}\right)\right)}_{\log N(x_i|\mu,\sigma^2)}$$

- we compute gradient and set it to zero:
  - $\nabla_\mu \mathcal{L}(\mu,\sigma^2) = \frac{1}{\sigma^2}\sum_{i=1}^n (\mu - x_i)$

    which is zero for $\mu = \frac{1}{n}\sum_{i=1}^n x_i$

    (which makes sense as it is the empirical mean)
  - $\nabla_{\sigma^2} \mathcal{L}(\mu,\sigma^2) = \frac{\sum_{i=1}^n (x_i-\mu)^2}{2(\sigma^2)^2} - \frac{n}{2\sigma^2}$

    which is zero for $\sigma^2 = \frac{1}{n}\sum_{i=1}^n (x_i-\mu)^2$

    (which makes sense as it is the empirical variance)

MLE for GMM

- we want to fit a model by solving

$$\max_{\pi_1,\pi_2,\mu_1,\mu_2,\sigma_1^2,\sigma_2^2} \quad \sum_{i=1}^{n} \log\Bigg(\pi_1 \underbrace{\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i;\mu_1,\sigma_1^2)} + \pi_2 \underbrace{\frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i;\mu_2,\sigma_2^2)} \Bigg)$$

MLE for GMM

- we want to fit a model by solving

$$\max_{\pi_1,\pi_2,\mu_1,\mu_2,\sigma_1^2,\sigma_2^2} \sum_{i=1}^{n} \log\Bigg(\pi_1 \underbrace{\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i;\mu_1,\sigma_1^2)} + \pi_2 \underbrace{\frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i;\mu_2,\sigma_2^2)} \Bigg)$$

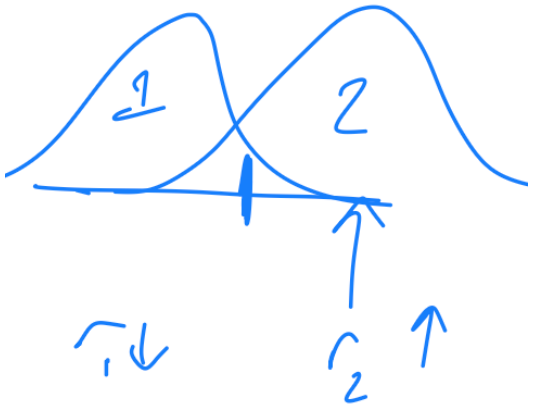
- define $r_i = \mathbf{P}(z_i = 1 \mid x_i) = \dfrac{\mathbf{P}(z_i = 1, x_i)}{\mathbf{P}(z_i = 1, x_i) + \mathbf{P}(z_i = 2, x_i)}$

  # 1 = cluster 1

  $$= \frac{\pi_1 N(x_i; \mu_1, \sigma_1^2)}{\pi_1 N(x_i; \mu_1, \sigma_1^2) + \pi_2 N(x_i; \mu_2, \sigma_2^2)}$$

  # normalization

Estimated soft membership



MLE for GMM

- we want to fit a model by solving

$$\max_{\pi_1,\pi_2,\mu_1,\mu_2,\sigma_1^2,\sigma_2^2} \sum_{i=1}^{n} \log\Bigg(\pi_1 \underbrace{\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i;\mu_1,\sigma_1^2)} + \pi_2 \underbrace{\frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i;\mu_2,\sigma_2^2)} \Bigg)$$

- define $r_i = \mathbf{P}(z_i = 1 \mid x_i) = \dfrac{\mathbf{P}(z_i = 1, x_i)}{\mathbf{P}(z_i = 1, x_i) + \mathbf{P}(z_i = 2, x_i)}$

  # r_i = soft membership of data point i → there are n r_i's

$$= \frac{\pi_1 N(x_i; \mu_1, \sigma_1^2)}{\pi_1 N(x_i; \mu_1, \sigma_1^2) + \pi_2 N(x_i; \mu_2, \sigma_2^2)}$$

normalization over K clusters

- setting the gradient to zero, we get
  - $\pi_1 = \dfrac{N_1}{n}$ where $N_1 = \sum_{i=1}^{n} r_i$, and $\pi_2 = \dfrac{N_2}{n}$ where $N_2 = \sum_{i=1}^{n} (1 - r_i)$
  - $\mu_1 = \dfrac{1}{N_1} \sum_{i=1}^{n} r_i x_i$ and $\mu_2 = \dfrac{1}{N_2} \sum_{i=1}^{n} (1 - r_i) x_i$
  - $\sigma_1^2 = \dfrac{1}{N_1} \sum_{i=1}^{n} r_i (x_i - \mu_1)^2$ and $\sigma_2^2 = \dfrac{1}{N_2} \sum_{i=1}^{n} (1 - r_i)(x_i - \mu_2)^2$
- both LHS and RHS depend on the parameters, and no closed form solution exists
- **note that if we know r_i's it is trivial to compute parameters, and vice versa**

$\mu, \pi, \sigma \in \mathbb{R}^k$

Expectation Maximization (EM) algorithm to approximate the solution of MLE

- EM is a popular method to solve MLE for mixture models
- input: training data $\{x_i\}_{i=1}^n$
- output: $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$ # GMM params

- initialization: randomly initialize the parameters

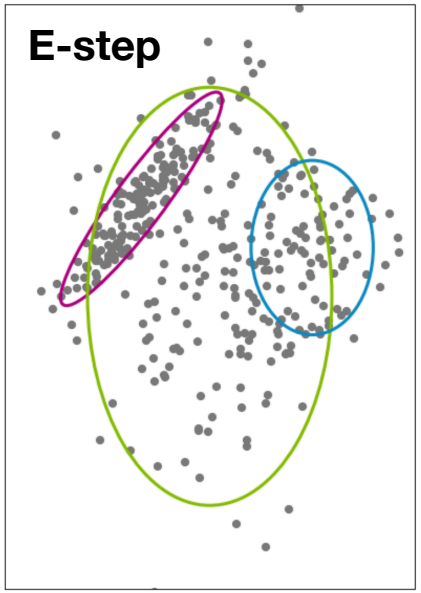
Expectation Maximization (EM) algorithm to approximate the solution of MLE

- EM is a popular method to solve MLE for mixture models
- input: training data $\{x_i\}_{i=1}^n$
- output: $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$

- initialization: randomly initialize the parameters
- repeat

# of each point in each cluster

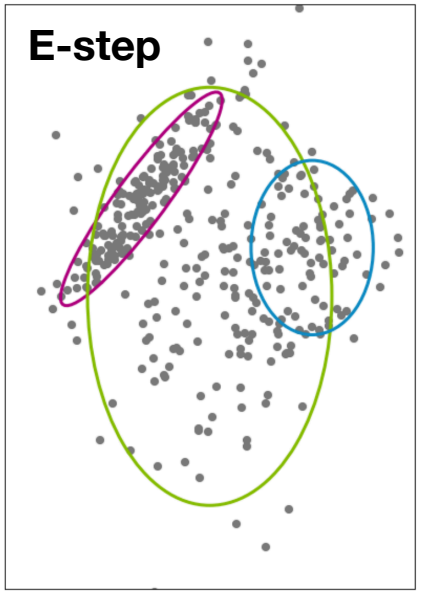
  - **E-step** (Expectation): parameters → soft membership
    - $r_i = \dfrac{\pi_1 N(x_i; \mu_1, \sigma_1^2)}{\pi_1 N(x_i; \mu_1, \sigma_1^2) + \pi_2 N(x_i; \mu_2, \sigma_2^2)}$ for all $i \in \{1,2,\ldots,n\}$
  - **M-step** (Maximization): soft membership → parameters

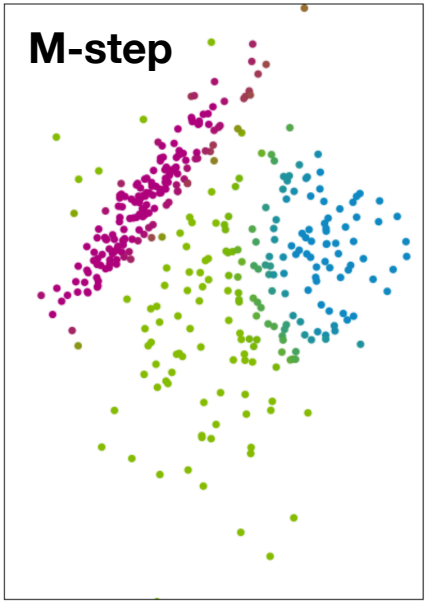


Expectation Maximization (EM) algorithm to approximate the solution of MLE

- EM is a popular method to solve MLE for mixture models
- input: training data $\{x_i\}_{i=1}^n$
- output: $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$

- initialization: randomly initialize the parameters
- repeat
  - **E-step** (Expectation): parameters → soft membership
    - $r_i = \dfrac{\pi_1 N(x_i; \mu_1, \sigma_1^2)}{\pi_1 N(x_i; \mu_1, \sigma_1^2) + \pi_2 N(x_i; \mu_2, \sigma_2^2)}$ for all $i \in \{1,2,\ldots,n\}$
  - **M-step** (Maximization): soft membership → parameters
    - $\pi_1 = \dfrac{N_1}{n}$ where $N_1 = \sum_{i=1}^n r_i$, and $\pi_2 = \dfrac{N_2}{n}$ where $N_2 = \sum_{i=1}^n (1 - r_i)$
    - $\mu_1 = \dfrac{1}{N_1} \sum_{i=1}^n r_i x_i$ and $\mu_2 = \dfrac{1}{N_2} \sum_{i=1}^n (1 - r_i) x_i$
    - $\sigma_1^2 = \dfrac{1}{N_1} \sum_{i=1}^n r_i (x_i - \mu_1)^2$ and $\sigma_2^2 = \dfrac{1}{N_2} \sum_{i=1}^n (1 - r_i)(x_i - \mu_2)^2$





For general number of clusters K and dimension d

- we can derive EM for general case, in an analogous way
- Initialize parameters: $\pi_1, \ldots, \pi_K, \mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K$

- **E-step**:
  - For k=1,…,K

$$r_{i,k} = \frac{\pi_k \, N(x_i \,|\, \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \, N(x_i \,|\, \mu_j, \Sigma_j)}$$

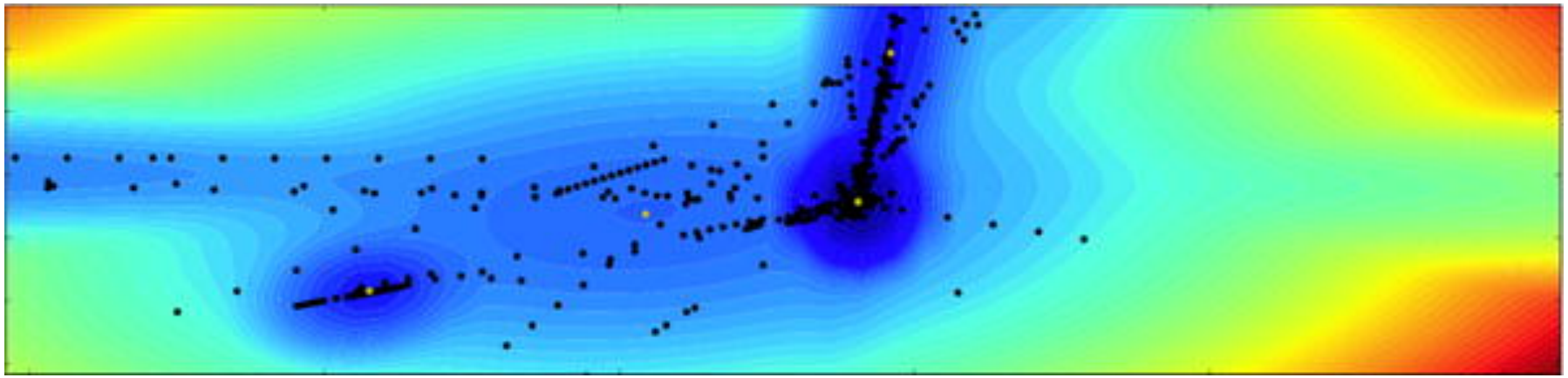
- **M-step**:
  - For k=1,..,K

$$\pi_k = \frac{N_k}{n} \quad \text{where} \quad N_k = \frac{\sum_{i=1}^{n} r_{i,k}}{n}$$

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{n} r_{i,k} x_i \quad \text{and} \quad \Sigma_k = \frac{1}{N_k} \sum_{i=1}^{n} r_{i,k} (x_i - \mu_k)(x_i - \mu_k)^T$$

- **once GMM is learned, clustering is straight forward: cluster according to the $r_{i,k}$'s**

GMM for real data

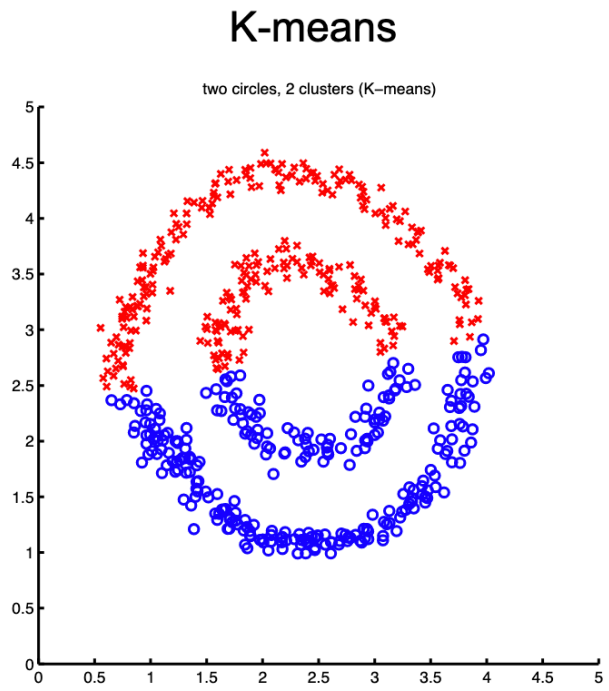


- Real-time location data for one participant (MIT student) for 30 days
- Shading shows learned likelihood under the GMM model
  - Darker blue regions have higher likelihood; they correspond to the student's dorm and their office
- Enables computing the **likelihood of their day**, in terms of location; whether they **deviate from routine**
- This turns out to be a massively useful feature for stress and mood prediction

Personalized Multi-task Learning for Predicting Tomorrow's Mood, Stress, and Health. **Taylor*, Jaques*, et al. 2017**

k-means and GMMs are inherently linear

- It tries to find linear boundaries between centers
- It fails completely on non-linearly clustered datasets such as

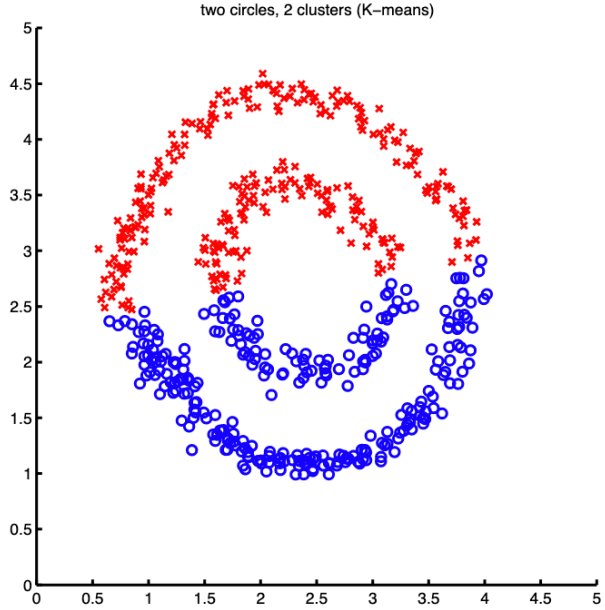


[Shi,Malik,'00],[Ng,Jordan,Weiss,'01]

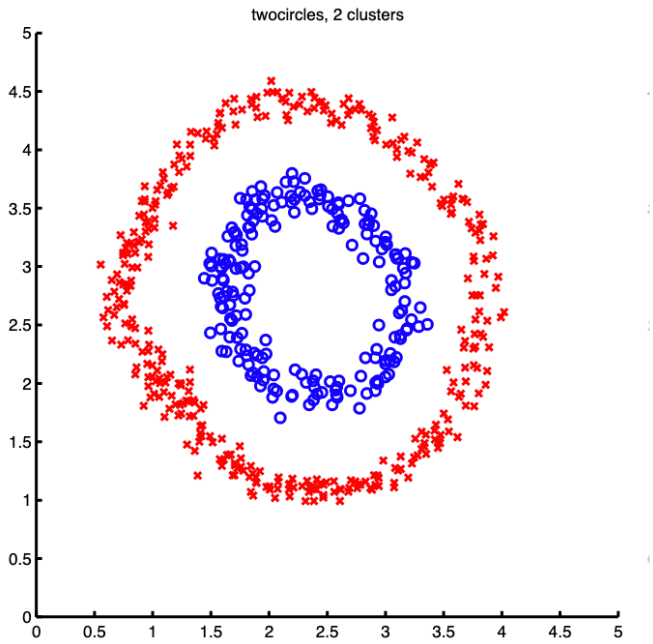
Spectral clustering

- Main idea:
  - Transform the dataset into a graph
  - Use eigenvalues (also called spectrum) and vectors of a graph to cluster

K-means



Spectral clustering



[Shi,Malik,'00],[Ng,Jordan,Weiss,'01]