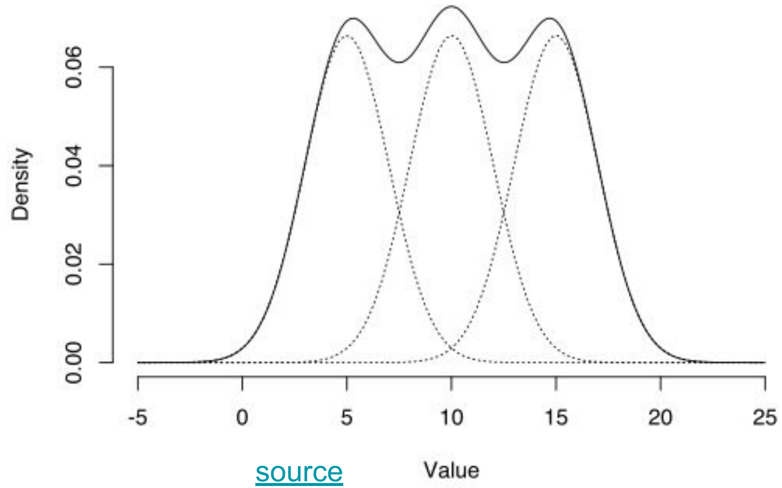
Early Generative Models: Gaussian Mixture Models

Arin Ghazarian
Chapman University

Mixture Models

- A probabilistic model for representing the presence of subpopulations within an overall population (without requiring that an observed data set should identify the sub-population to which an individual observation belongs)
- A mixture distribution representing the probability distribution of observations in the overall population.



source

Examples

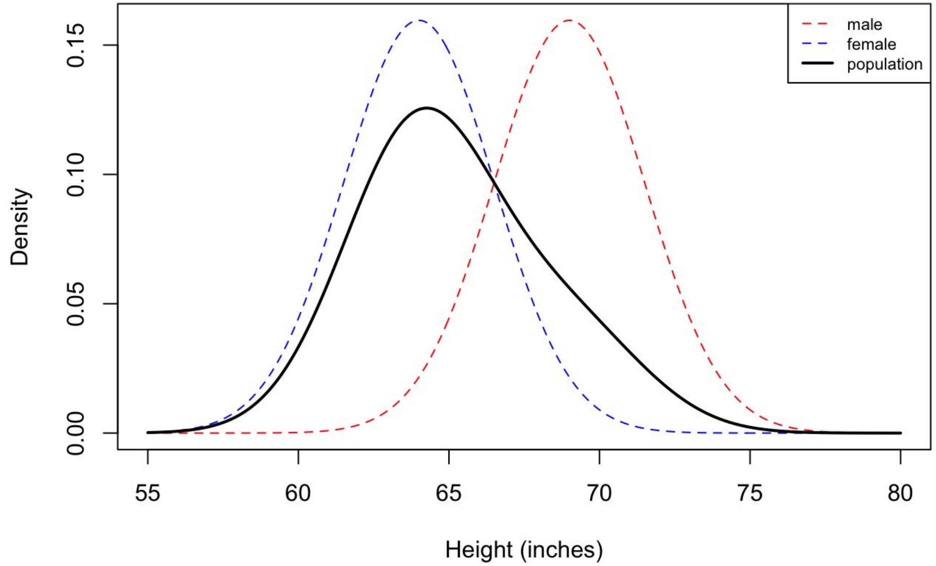
- Financial returns: often behave differently in normal situations and during crisis times
- Housing Prices: a mixture model with K different components, each distributed as a normal distribution with unknown mean and variance, with each component specifying a particular combination of house type/neighborhood.

Gaussian Mixture Model

- A probabilistic model for representing normally distributed subpopulations within an overall population.
- Unsupervised learning
- Very similar to K-means

GMM Example: Human Height

- Human height is typically modeled as a normal distribution for each gender
- Unimodal because of the high level of overlap between the two densities
- Not symmetric, and therefore not normally distributed



source

Gaussian Mixture Models

- Each component is Gaussian
- For a Gaussian mixture model with K components, the k-th component has a mean of μ_k and variance of σ_k
- The mixture component weights are defined as ϕ_k for component C_k, with the constraint that $\sum \phi_i = 1$
- Φ_is follow a multinomial distribution

$$p(x) = \sum_{i=1}^{K} \phi_i \mathcal{N}(x \mid \mu_i, \sigma_i)$$

$$\mathcal{N}(x \mid \mu_i, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$$

$$\sum_{i=1}^{K} \phi_i = 1$$

Gaussian Mixture Models: Multivariate

$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i)$$

$$\mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_i)^\mathrm{T} {\Sigma_i}^{-1} (\vec{x} - \vec{\mu}_i)\right)$$

$$\sum_{i=1}^{K} \phi_i = 1$$

GMM: Universal Approximator

- A probabilistic view of clustering
- Each cluster belongs to a different Gaussian.
- Uses latent variables (the cluster number)
- General approach, can replace Gaussian with other distributions (continuous or discrete)
- GMM is a powerful model
- GMMs are universal approximators of densities (using enough number of Gaussians)
- Optimization is done using the EM algorithm
- GMM is a density estimator

Latent Variable

- Variables which are always unobserved are called latent variables, or sometimes hidden variables
- The identity of the component used to generate the data point represent a latent variable or unobservable data (Z)

Marginal Likelihood of the Observed Data

$$p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$$

$$p(\mathbf{x}) = \sum_{z} p(\mathbf{x}, z) = \sum_{z} p(\mathbf{x}|z)p(z)$$

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}, z = k)$$

$$= \sum_{k=1}^{K} \underbrace{p(z = k)}_{\pi_k} \underbrace{p(\mathbf{x}|z = k)}_{\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}$$

Marginal Likelihood of the Observed Data

- The continuous form will be:

$$L(\boldsymbol{\theta}; \mathbf{X}) = p(\mathbf{X} \mid \boldsymbol{\theta}) = \int p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})\, d\mathbf{Z} = \int p(\mathbf{X} \mid \mathbf{Z}, \boldsymbol{\theta}) p(\mathbf{Z} \mid \boldsymbol{\theta})\, d\mathbf{Z}$$

Maximizing the Marginal Likelihood of the Observed Data

- We need to apply MLE to find the parameters which maximize the log likelihood of the observed data

$$
\begin{aligned}
\ell(\boldsymbol{\pi}, \mu, \Sigma) = \ln p(\mathbf{X}|\pi, \mu, \Sigma) &= \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\pi, \mu, \Sigma) \\
&= \sum_{n=1}^{N} \ln \sum_{z^{(n)}=1}^{K} p(\mathbf{x}^{(n)}|\, z^{(n)}; \mu, \Sigma) p(z^{(n)}|\, \boldsymbol{\pi})
\end{aligned}
$$

- How can we optimize this?
    - Often intractable since Z is unobserved and the distribution of Z is unknown before attaining μ ,Σ
    - No analytic closed form solution
    - Numerical methods also are difficult and slow

Maximizing the Marginal Likelihood of the Observed Data

- If we knew Z^n, it would be easy to optimize

$$\ell(\boldsymbol{\pi}, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(x^{(n)}, z^{(n)} | \pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)} | \, z^{(n)}; \mu, \Sigma) + \ln p(z^{(n)} | \, \boldsymbol{\pi})$$

$$\mu_k = \frac{\sum_{n=1}^{N} 1_{[z^{(n)}=k]} \, \mathbf{x}^{(n)}}{\sum_{n=1}^{N} 1_{[z^{(n)}=k]}}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} 1_{[z^{(n)}=k]} \, (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T}{\sum_{n=1}^{N} 1_{[z^{(n)}=k]}}$$

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} 1_{[z^{(n)}=k]}$$

Expectation maximization (EM)

- The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying these two steps until convergence:
  - **E-step:** Compute the posterior probability over z given our current model (how probably each Gaussian generates each datapoint)
  - **M-step:** Assuming that the data really was generated this way, update the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for
- Expectation maximization technique is commonly used to estimate the mixture model's parameters

Expectation maximization (EM)

- The identity of the component used to generate the data point represent a latent variable or unobservable data.
- E-step estimates the expected value for each latent variable
- M-step helps in optimizing them significantly using the Maximum Likelihood Estimation (MLE).
- This process is repeated until a good set of latent values, and a maximum likelihood is estimated

EM vs. K-means

- EM for mixtures of Gaussians is just like a soft version of K-means, with fixed priors and covariance
  - K-means assigns 0s and 1s vs probabilities
- Instead of hard assignments in the E-step, we do soft assignments based on the softmax of the squared Mahalanobis distance from each point to each cluster.
- Each center moved by weighted means of the data, with weights given by soft assignments
  - In K-means, weights are 0 or 1

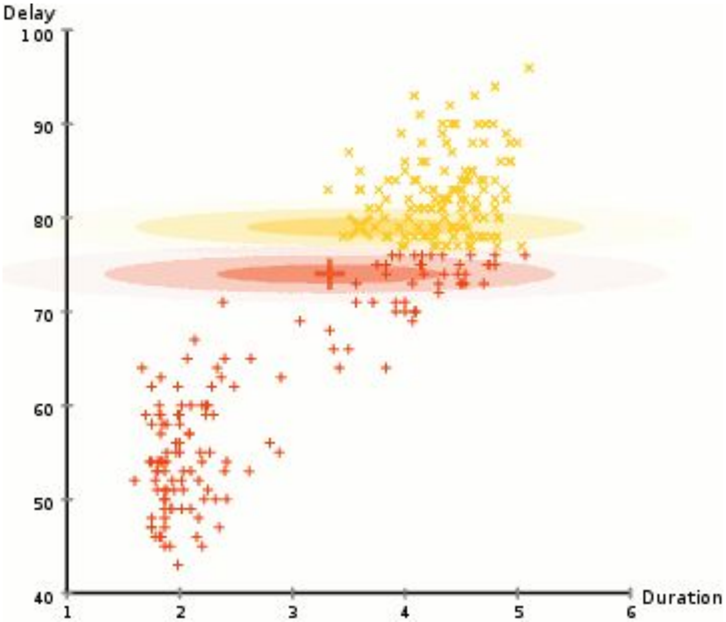
Demo: Analysing Old Faithful Data Using GMMs

- Old Faithful Geyser Data: Waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park,
- A data frame with 272 observations on 2 variables:
    - Eruptions: eruptions duration in mins
    - Waiting: time to next eruption

source

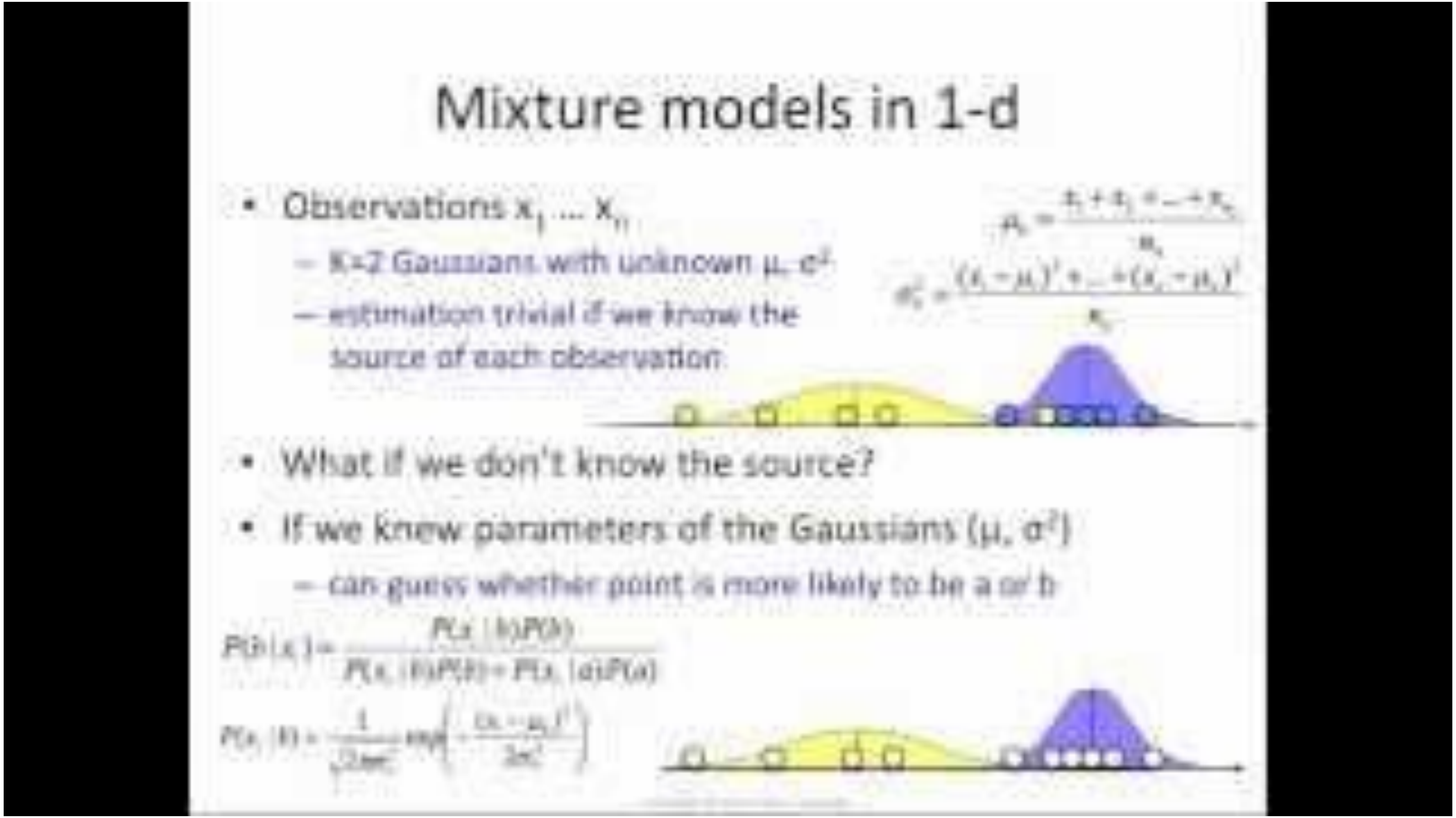
EM: Clustering of Old Faithful Data

- Starting with a random assignment and unit spheres (note that the axes have different scales), EM quickly adapts to the dataset.
- X axis is the duration of the geyser eruption, Y axis is the delay from the previous eruption.



Demo: Analysing Old Faithful Data Using GMMs Notebook

- http://localhost:8888/notebooks/jupyter-notebooks/chapman-generative-AI/GMM-old-faithful-Geyser.ipynb
- GMM in Python from Scratch: Old Faithful Data



https://www.youtube.com/watch?v=REypj2sy_5U

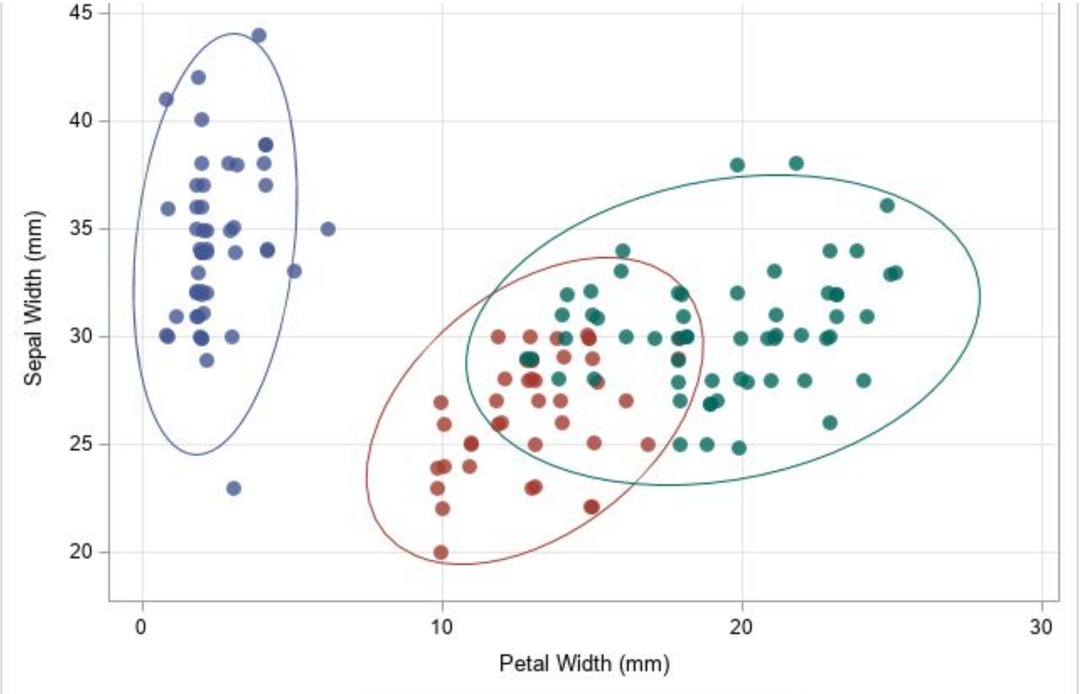
GMMs as Generative Model

GMMS as Generative Models

- Earlier works in Generativ (1950s)
- Appropriate for scenarios when we have
  - low-dimensional sequential data
  - Short-term dependencies

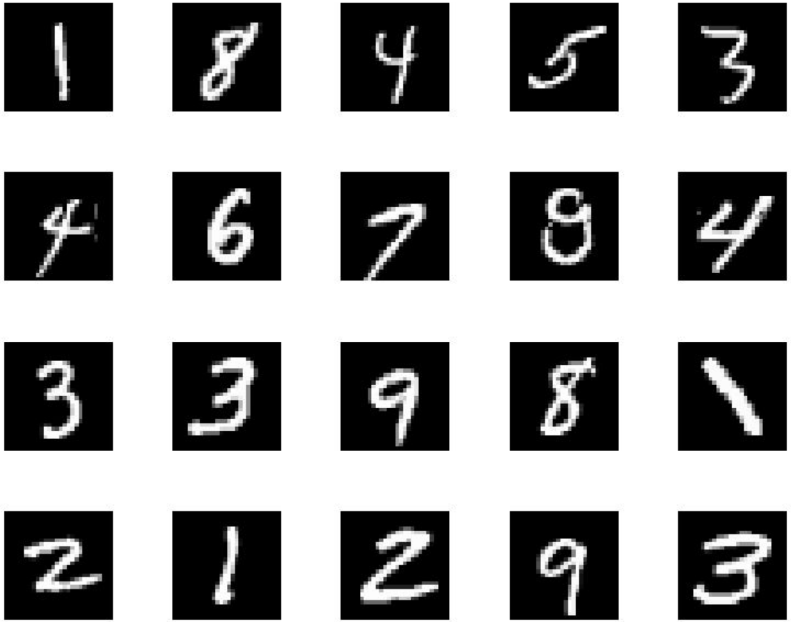
GMM as a Generative Model

- Each data point is sampled from a generative process:
  - Choose component i with probability P(y=i)
  - Generate a random sample according to $\sim N(\mu_i, \Sigma_i)$



Generative Models for Classification: Handwritten Digit Recognition

- Input: N×N black-and-white image that is known to be a scan of a handwritten digit between 0 and 9,
- We don't know which digit is written.
- We can create a mixture model with $K=10$ different components, where each component is a vector of size N^2 of Bernoulli distributions (one per pixel).
- We can train this using expectation-maximization and the output will give us the clusters (Digits)
- The same model could then be used as a classifier, by computing the probability of the new image for each possible component (digit) and returning the digit which had the highest probability



Demo: HandWritten Digit Generation

http://localhost:8888/notebooks/jupyter-notebooks/chapman-generative-AI/GMM-handwritten-digits-generation.ipynb

EM: Handwritten Digit Recognition Demo

- Expectation Maximization for the recognition of handwritten digits in the MNIST dataset