Gaussian Mixture Model

Zhiyao Duan
Associate Professor of ECE and CS
University of Rochester

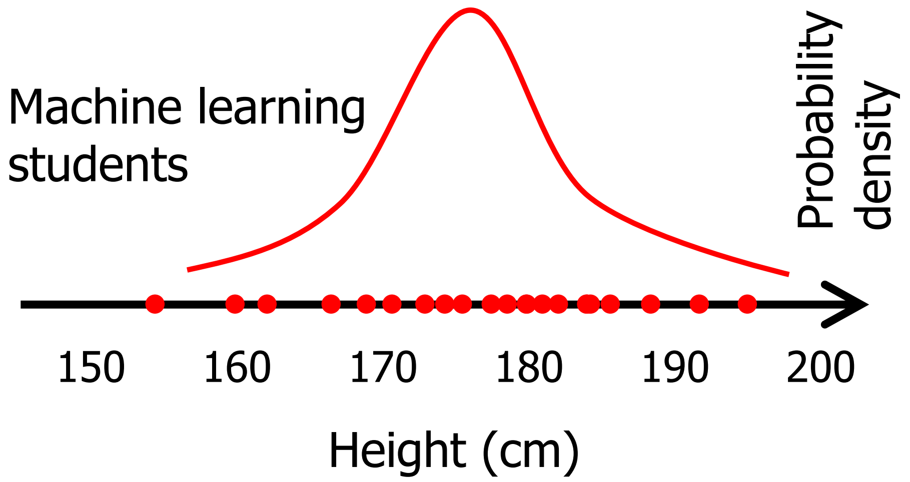
Adapted from slides “Gaussian Mixture Model” by Zhiyao Duan & Bryan Pardo for
EECS 349 – Machine Learning, Northwestern University, Fall 2012.

Generative Model Perspective

- We think of the data as being generated from some process
- We assume that this process is sampling data from an underlying distribution
- This distribution can be a parametric distribution (or called model), e.g., a Gaussian distribution, or a non-parametric distribution. We often prefer parametric distributions as they are easier to represent
- We infer model parameters from data
- Then we can use the model to explain or generate data

Parametric Distribution

- Represent the underlying probability distribution with a parametric probability function



- Gaussian (normal) distribution, two parameters:

$$p(x;\ \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- View each point as generated from $p(x;\ \mu, \sigma^2)$

Maximum Likelihood Estimation

- Our hypothesis space is Gaussian distributions
- Find parameter(s) $\boldsymbol{\theta}$ that make a Gaussian most likely to generate data $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)})$
- Likelihood function:

$$l(\boldsymbol{\theta}) \equiv p(\boldsymbol{X}; \boldsymbol{\theta}) = \prod_{i=1}^{N} p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

$\boldsymbol{\theta} = \{\mu, \sigma^2\}$

Only if $\boldsymbol{X}$ are i.i.d.

Likelihood Function

$$l(\boldsymbol{\theta}) \equiv p(\boldsymbol{X}; \boldsymbol{\theta}) = \prod_{i=1}^{N} p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

- In our Gaussian example, $\boldsymbol{x}^{(i)}$ is a continuous variable, $p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$ is the probability density function (PDF)
  - It is meaningless to talk about probability mass here, as the probability mass at any value of $\boldsymbol{x}^{(i)}$ is zero

- If $\boldsymbol{x}^{(i)}$ is a discrete variable (e.g., binary), $p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$ should be replaced by the probability mass function $P\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$.
  - It is meaningless to talk about probability density here, as the density will be infinite at the value of each data point

Log-likelihood Function

- Likelihood function

$$l(\boldsymbol{\theta}) \equiv p(\boldsymbol{X}; \boldsymbol{\theta}) = \prod_{i=1}^{N} p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

- Log-likelihood function

$$L(\boldsymbol{\theta}) \equiv \log l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

  – Maximizing Log-likelihood ⇔ maximizing likelihood
  – Easier to optimize
  – Prevents underflow!
    - What happens when multiplying 1000 probabilities?

Example Gaussian Log-likelihood

- Log-likelihood function

$$L(\boldsymbol{\theta}) \equiv \log l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

- Recall 1-d Gaussian distribution (probability density function)

$$p(x;\, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- So the log-likelihood of 1-d Gaussian would be:

$$L(\mu, \sigma^2 | X) = \boxed{-\frac{N}{2}\log(2\pi)} - N \log \sigma - \frac{\sum_{i=1}^{N}\left(x^{(i)} - \mu\right)^2}{2\sigma^2}$$

↖ a constant term

Maximizing Log-likelihood

- Log-likelihood of Gaussian:

$$L(\mu, \sigma^2) = C - N \log \sigma - \frac{\sum_{i=1}^{N} \left(x^{(i)} - \mu\right)^2}{2\sigma^2}$$

- Take partial derivatives w.r.t. μ and σ and set them to 0, i.e., let $\frac{\partial L}{\partial \mu} = 0$ and $\frac{\partial L}{\partial \sigma} = 0$.

- Then solve… (try it yourself), we get

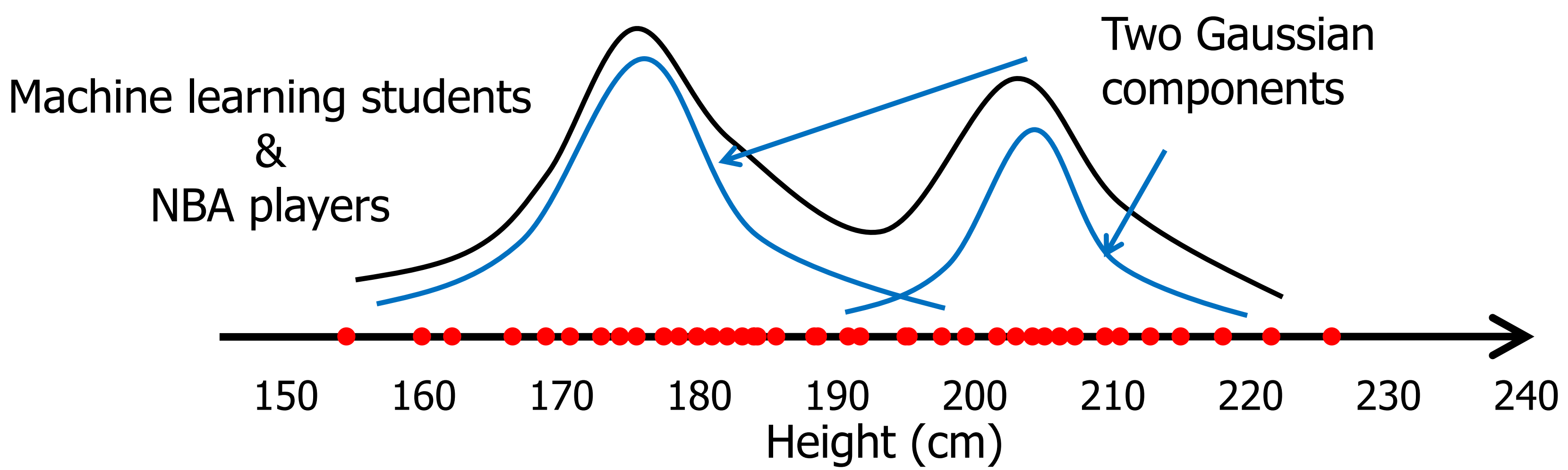
$$\mu = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}; \qquad \sigma^2 = \frac{1}{N} \sum_{i=1}^{N} \left(x^{(i)} - \mu\right)^2$$

What if…

- …the data distribution can't be well represented by a single Gaussian?

- Can we model more complex distributions using multiple Gaussians?

Gaussian Mixture Model (GMM)



- Represent the distribution with a mixture of Gaussians

$$p(x) = \sum_{j=1}^{K} \underbrace{P(z=j)}_{\text{Weight of } j\text{-th Gaussian. Often notated as } w_j} \underbrace{p(x|z=j)}_{\text{The } j\text{-th Gaussian, parameter: } (\mu_j, \sigma_j^2)}$$

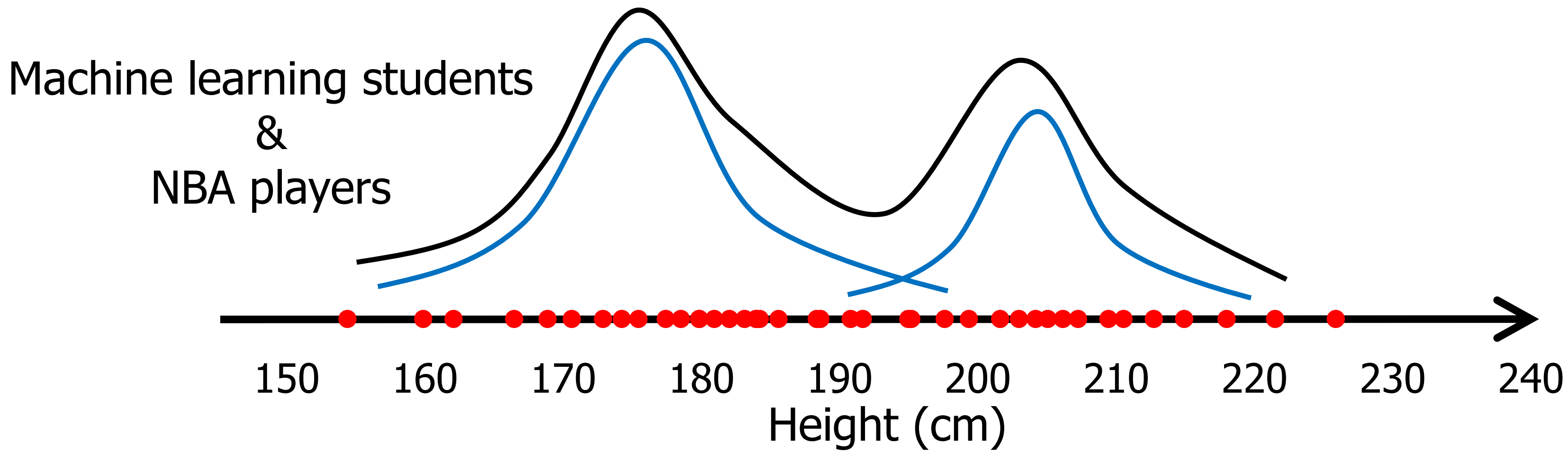
z: a membership r.v. indicating which Gaussian that x belongs to.

Weight of j-th Gaussian. Often notated as w_j

The j-th Gaussian, parameter: (μ_j, σ_j^2)

z is a discrete variable, so we use probability mass P.

Generative Process for GMM



$$p(x) = \sum_{j=1}^{K} P(z = j)p(x|z = j)$$

- 1. Randomly pick a component j, according to $P(z = j)$;
- 2. Generate x according to $p(x|z = j)$.

What are we optimizing?

- GMM distribution:

$$p(x) = \sum_{j=1}^{K} P(z=j)p(x|z=j)$$

$$= \sum_{j=1}^{K} w_j \cdot \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}}$$

- Three parameters per Gaussian in the mixture w_j, μ_j, σ_j^2, where $\sum_{j=1}^{K} w_j = 1$
- Find parameters that maximize data likelihood

Maximum Likelihood Estimation of GMM

- Given $\boldsymbol{X} = \left\{x^{(1)}, \dots, x^{(N)}\right\}$, $x^{(i)} \sim p(x)$, log-likelihood is

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p\left(x^{(i)}\right) = \sum_{i=1}^{N} \log \left\{ \sum_{j=1}^{K} P\left(z^{(i)} = j\right) \cdot p\left(x^{(i)} | z^{(i)} = j\right) \right\}$$

$$= \sum_{i=1}^{N} \log \left\{ \sum_{j=1}^{K} w_j \cdot \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{\left(x^{(i)} - \mu_j\right)^2}{2\sigma_j^2}} \right\}$$

- Try to solve parameters $\left(\mu_j, \sigma_j^2, w_j\right)$ by setting their partial derivatives to 0?
- No closed form solution. (Try it yourself)

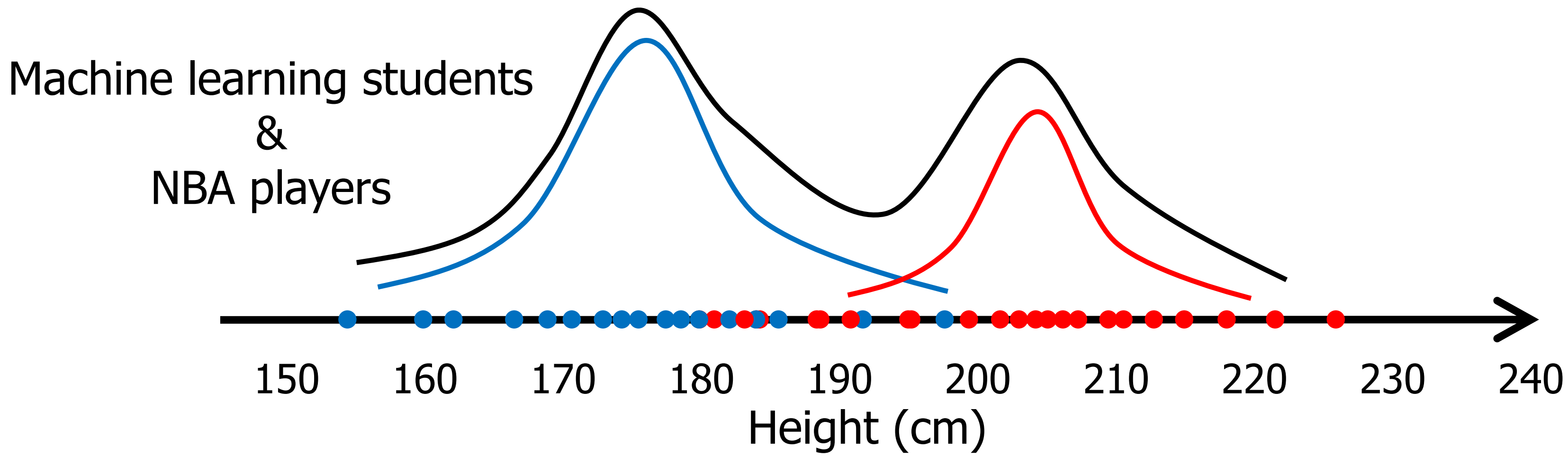
Why is maximum likelihood difficult for GMM?

- Each data point $x^{(i)}$ has a membership random variable $z^{(i)}$, indicating which Gaussian it comes from

- But the value of $z^{(i)}$ cannot be observed, i.e., we are uncertain about which Gaussian $x^{(i)}$ comes from
  - $z^{(i)}$ is a latent variable

- Latent variables can also be viewed as missing data, data that we do not observe

If we know $z^{(i)}$, then maximum likelihood is easy



- $w_j = \frac{1}{N}\sum_i^N 1\{z^{(i)} = j\}$
- $\mu_j = \frac{\sum_i^N 1\{z^{(i)}=j\}x^{(i)}}{\sum_i^N 1\{z^{(i)}=j\}}$
- $\sigma_j^2 = \frac{\sum_i^N 1\{z^{(i)}=j\}\left(x^{(i)}-\mu\right)^2}{\sum_i^N 1\{z^{(i)}=j\}}$

Indicator function:

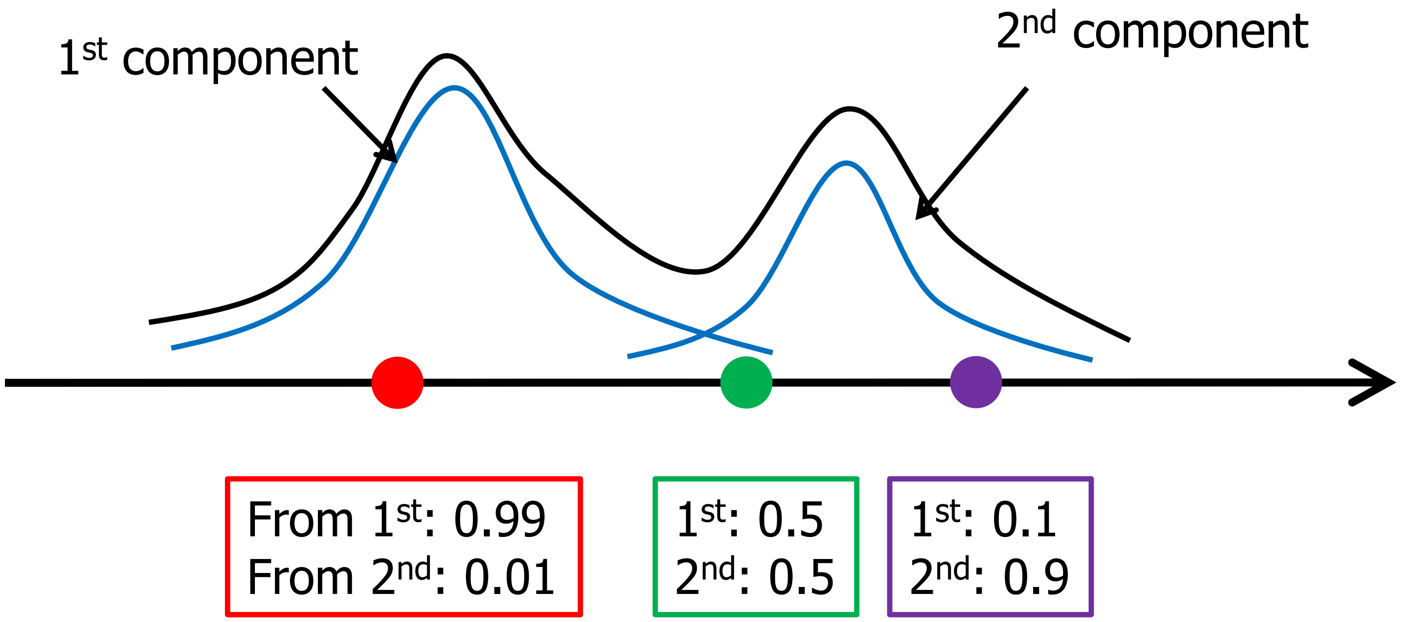
$$1\{z^{(i)} = j\} = \begin{cases} 1, & \text{if } z^{(i)} = j; \\ 0, & \text{if } z^{(i)} \neq j. \end{cases}$$

N = number of training examples

Illustration of "Soft" Membership

- Which component does point i come from?
- The probability that it comes from j:

$$q_j^{(i)} \equiv P\left(z^{(i)} = j | x^{(i)}\right)$$



Improving Our Posterior Probability

- The "posterior probability" of a Gaussian component given a data example is the probability that this data example was generated from this Gaussian component

- Let's find a way to use posterior probabilities to make an algorithm that automatically creates a set of Gaussian components that would have been very likely to generate this data

Expectation Maximization (EM)

- Instead of analytically solving the maximum likelihood parameter estimation problem of GMM, we seek an alternative way, the EM algorithm
- EM algorithm updates parameters iteratively
- In each iteration, the likelihood value increases (at least it does not decrease)
- EM algorithm always converges (to some local optimum)

EM Algorithm Summary

- Initialize parameters

  w_j, μ_j, σ_j^2 for each Gaussian j in our model

- E step: calculate posterior probabilities of latent variables

  probability that these Gaussian components generated the data

- M step: update parameters

  update w_j, μ_j, σ_j^2 for each Gaussian j

- Repeat E and M steps until convergence

  go until parameters do not change much

- It converges to some local optimum

EM for GMM - Initialization

- Start by choosing the number of Gaussian components K

- Also, choose an initialization of parameters of all components $\left(w_j, \mu_j, \sigma_j^2\right)$ for $j = 1, \dots, K$

- Make sure $\sum_{j=1}^{K} w_j = 1$

EM for GMM – Expectation Step

For each $x^{(i)}$, calculate its "soft" membership, i.e., the posterior probability of $z^{(i)}$, using current parameters

$$q_j^{(i)} \equiv P\left(z^{(i)} = j | x^{(i)}\right) = \frac{P\left(z^{(i)} = j, x^{(i)}\right)}{p\left(x^{(i)}\right)}$$

Bayes rule

$$= \frac{p\left(x^{(i)} | z^{(i)} = j\right) P\left(z^{(i)} = j\right)}{\sum_{l=1}^{K} p(x^{(i)} | z^{(i)} = l) P(z^{(i)} = l)}$$

– Note: we are guessing the distribution (i.e., a "soft" membership) of $z^{(i)}$, instead of a "hard" membership

EM for GMM – Maximization step

– M step: update parameters.

Recall $q_j^{(i)}$ is the "soft" membership of $x^{(i)}$ of the j-th Gaussian.

$$w_j = \frac{1}{N}\sum_{i=1}^{N} q_j^{(i)}$$

Average of their membership

$$\mu_j = \frac{\sum_{i=1}^{N} q_j^{(i)} x^{(i)}}{\sum_{i=1}^{N} q_j^{(i)}}$$

Weighted average using membership

$$\sigma_j^2 = \frac{\sum_{i=1}^{N} q_j^{(i)} \left(x^{(i)} - \mu_j\right)^2}{\sum_{i=1}^{N} q_j^{(i)}}$$

Weighted variance using membership

- Repeat E step and M step until convergence
  - Convergence criterion in practice: likelihood value does not increase much or parameters do not change much, compared to the previous iteration

EM Algorithm Summary

- Initialize parameters

  w_j, μ_j, σ_j^2 for each Gaussian j in our model

- E step: calculate posterior probabilities of latent variables

  probability that these Gaussian components generated the data

- M step: update parameters

  update w_j, μ_j, σ_j^2 for each Gaussian j

- Repeat E and M steps until convergence

  go until parameters do not change much

- It converges to some local optimum

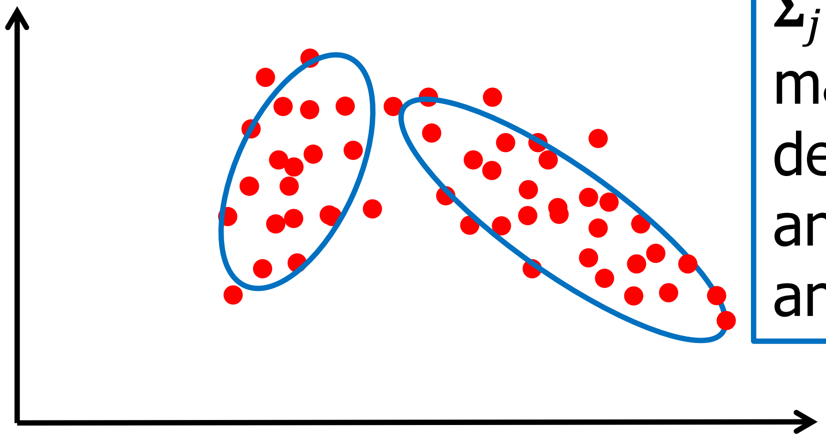
What if…

- …our data isn't just scalars, but each data point has multiple dimensions?

- Can we generalize to multiple dimensions?

Multivariate Gaussian Mixture

$\boldsymbol{\mu}_j$, a mean vector, marks the center of an ellipse



$\boldsymbol{\Sigma}_j$, the covariance matrix ($d \times d$), describes the shape and orientation of an ellipse

$$p(\boldsymbol{x}) = \sum_{j=1}^{K} w_j \cdot \frac{1}{(2\pi)^{\frac{d}{2}} \left|\boldsymbol{\Sigma}_j\right|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_j\right)\right\}$$

d: dimensionality

- Parameters: $\left(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, w_j\right)$ for $j = 1, \ldots, K$, with $\sum_{j=1}^{K} w_j = 1$.
- How many parameters?

$$\underbrace{dK}_{\text{means}} + \underbrace{\frac{d(d+1)}{2}K}_{\text{covariances}} + \underbrace{K}_{\text{weights}}$$

Example: Initialization

(Illustration from Andrew Moore's tutorial slides on GMM)

26

After Iteration #1

(Illustration from Andrew Moore's tutorial slides on GMM)

27

After Iteration #2

(Illustration from Andrew Moore's tutorial slides on GMM)

After Iteration #3

(Illustration from Andrew Moore's tutorial slides on GMM)

After Iteration #4

(Illustration from Andrew Moore's tutorial slides on GMM)

30

After Iteration #5

(Illustration from Andrew Moore's tutorial slides on GMM)

31

After Iteration #6

(Illustration from Andrew Moore's tutorial slides on GMM)

32

After Iteration #20

(Illustration from Andrew Moore's tutorial slides on GMM)

GMM Remarks

- GMM is powerful: any density function can be arbitrarily well approximated by a GMM with enough components

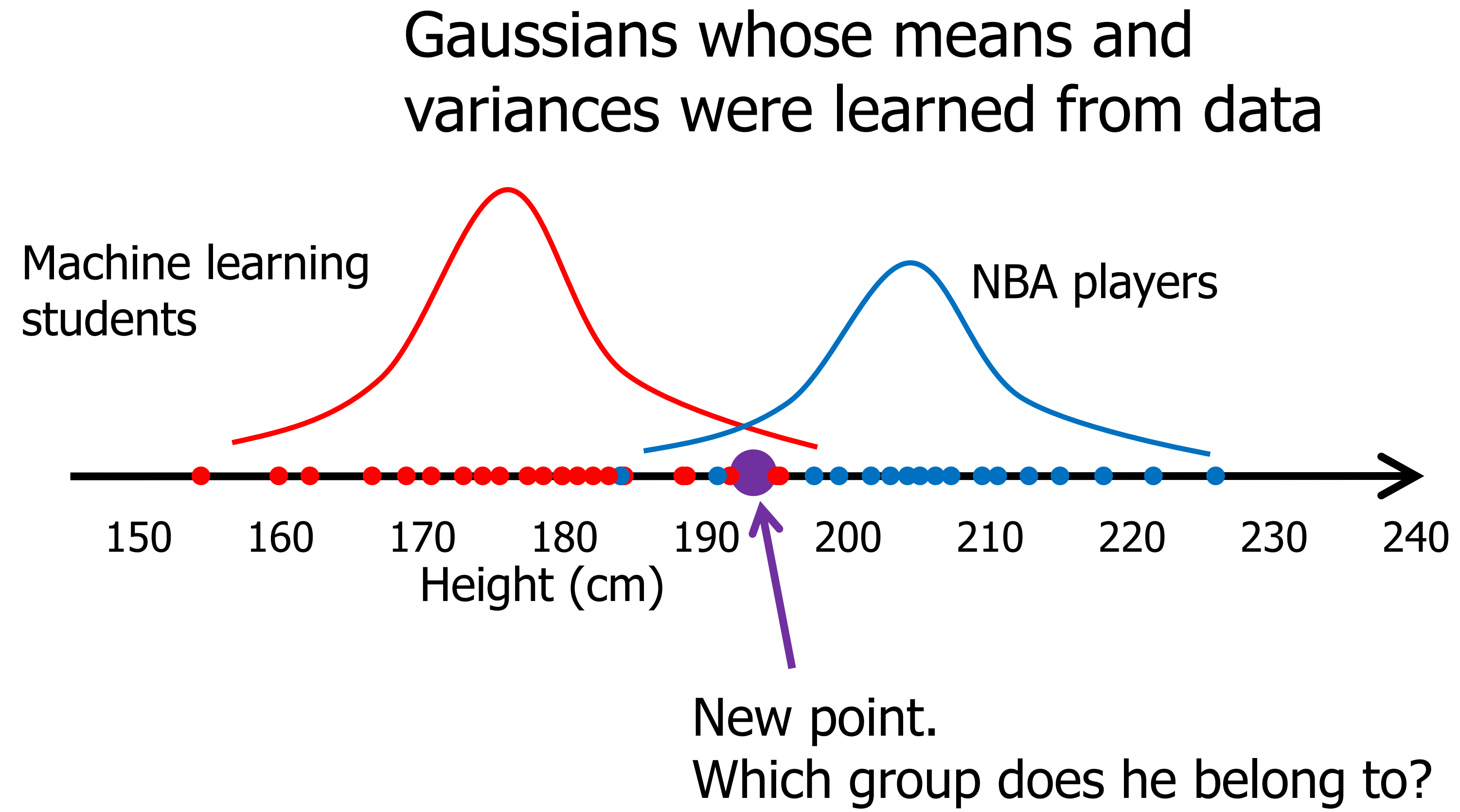
- If the number of components K is too large, data will be overfit
  - Likelihood always increases with K
  - Extreme case: N Gaussians for N data points, with variances $\rightarrow 0$, then likelihood $\rightarrow \infty$

- How to choose K?
  - Use domain knowledge
  - Validate through visualization

GMM is a "soft" version of K-means

- Similarities
  - K needs to be specified
  - Converges to some local optima
  - Initialization matters final results
  - One would want to try different initializations
- Differences
  - GMM assigns "soft" labels to instances
  - GMM considers covariances in addition to means
    - Each cluster is represented as an ellipse instead of a circle

Using Generative Models for Classification



- Bayes Classification answer: The class from which the data point is more likely sampled

GMM for Classification

1. Given $D = \{\langle x^{(i)}, y^{(i)} \rangle\}$, where $y^{(i)} \in \{1, \dots, C\}$
2. Model $p(x|y = l)$ with a GMM, for each l
3. Calculate class posterior probability

$$P(y = l|x) = \frac{p(x|y = l)P(y = l)}{\sum_{k=1}^{C} p(x|y = k)P(y = k)}$$

Bayes classification

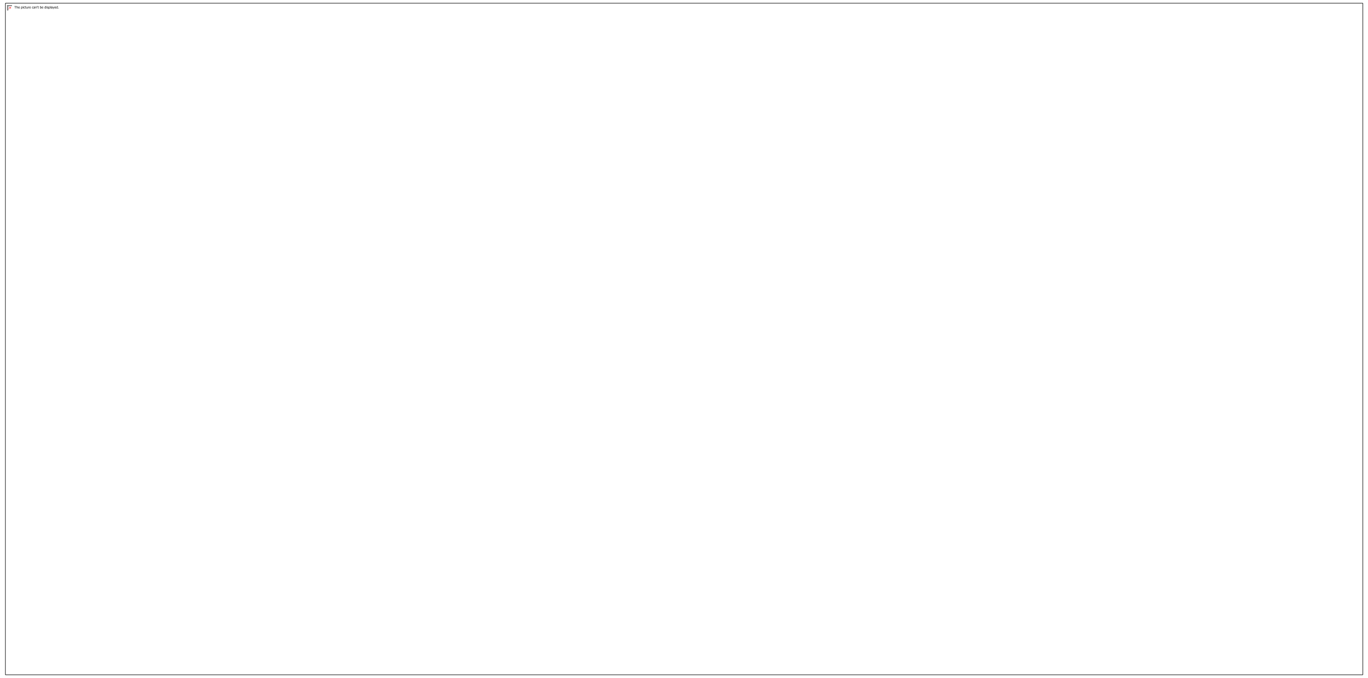
4. Classify x to the class having largest posterior.

(illustration from Leon Bottou's slides on EM)

GMM for Regression

- Given $D = \{\langle x^{(i)}, y^{(i)} \rangle\}$, where $y^{(i)} \in \mathbb{R}$
- Model $p(x, y)$ with a GMM
- Compute $f(x) = \mathbb{E}[y|x]$, conditional expectation

(illustration from Leon Bottou's slides on EM)



Summary

- Maximum Likelihood (ML) estimation of parametric model's parameters
  - Update parameters to increase data likelihood
- GMM models data distribution with a mixture of K Gaussians, with parameters $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, w_j)$, for $j = 1, \dots, K$
  - No closed form solution for ML estimation of GMM parameters, due to latent variables
- How to estimate GMM parameters with EM algorithm?
  - Iterative and greedy algorithm for maximum likelihood estimation with laten variables
- How is GMM related to K-means?
  - Soft version of K-means; models data covariances in addition to means
- How to use GMM for classification and regression?