
Machine Learning

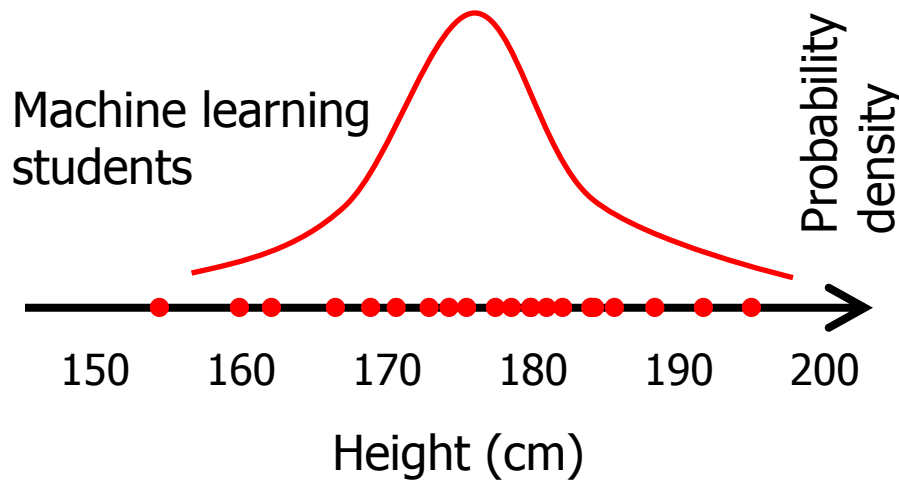
Gaussian Mixture Models

The Generative Model POV

- We think of the data as being generated from some process.
- We assume this process can be modeled statistically as an underlying distribution.
- We often assume a **parametric distribution**, like a Gaussian, because they're easier to represent.
- We infer model parameters from the data.
- Then we can use the model to classify/cluster or even generate data.

Parametric Distribution

- Represent the underlying probability distribution with a parametric density function.



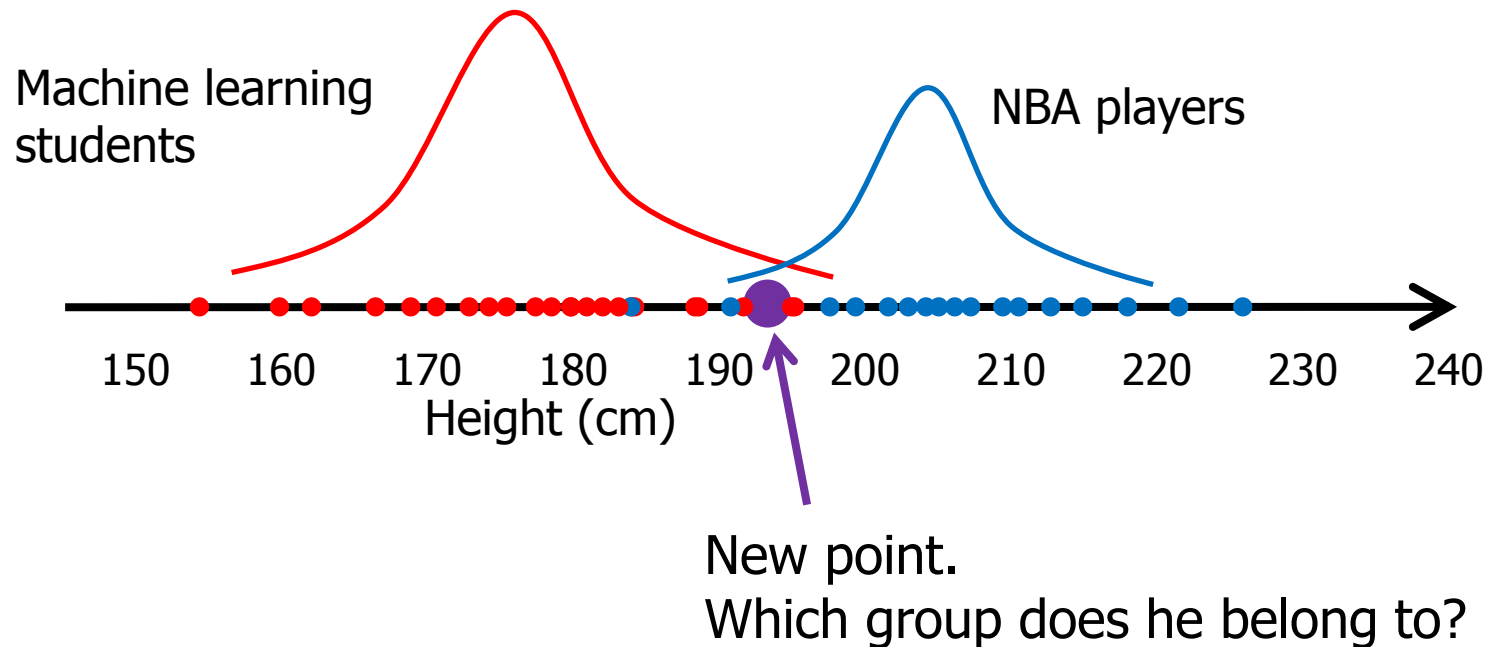
- Gaussian (normal) distribution, two parameters:

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

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

Using Generative Models for Classification

Gaussians whose means and variances were learned from data




Answer: the group (class) that calls the new point most probable.

Maximum Likelihood Estimation

- Our hypothesis space is Gaussian distributions.
- Find parameter(s) θ that make a Gaussian most likely to generate data $X = (x_1, \dots, x_n)$.
- Likelihood function:

$$l(\theta|X) \equiv p(X; \theta) = \prod_{i=1}^N p(x_i; \theta)$$


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



Only if X is i.i.d.

Likelihood Function

$$l(\theta|X) \equiv p(X; \theta) = \prod_{i=1}^N p(x_i; \theta)$$

- In our Gaussian example, x_i is a continuous variable, $p(x_i; \theta)$ is the probability density function (pdf).
 - It is meaningless to talk about probability mass here, as the probability mass at any value of x_i is zero.
- If x_i is a discrete variable (e.g. binary), $p(x_i; \theta)$ should be replaced by the probability mass function $P(x_i; \theta)$.
 - It is meaningless to talk about probability density p here, as the density will be infinite at the value of each data point.

Log-likelihood Function

- Likelihood function

$$l(\theta|X) \equiv p(X; \theta) = \prod_{i=1}^N p(x_i; \theta)$$

- Log-likelihood function

$$L(\theta|X) \equiv \log l(\theta|X) = \sum_{i=1}^N \log p(x_i; \theta)$$

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

Example Gaussian Log-likelihood

- Log-likelihood function

$$L(\theta|X) \equiv \log l(\theta|X) = \sum_{i=1}^N \log p(x_i; \theta)$$

- Recall Gaussian dist. (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 Gaussian would be:

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



a constant term

Maximizing Log-likelihood

- Log-likelihood of Gaussian:

$$L(\mu, \sigma^2 | X) = C - N \log \sigma - \frac{\sum_{i=1}^N (x_i - \mu)^2}{2\sigma^2}$$

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

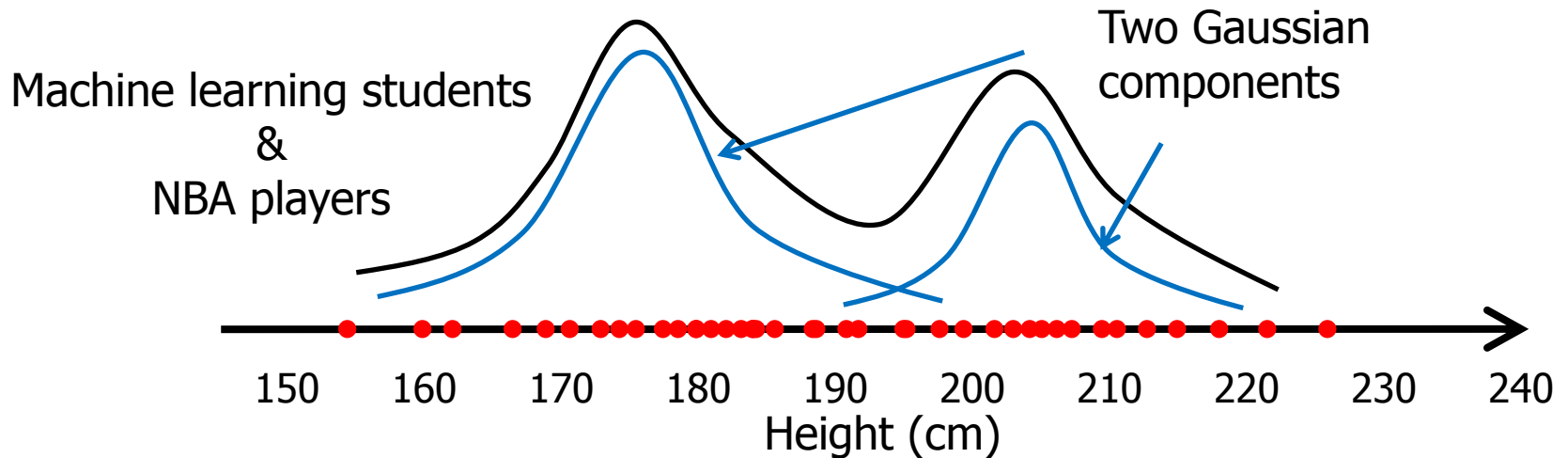
- Then solve... (try it yourself), we get

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i; \quad \sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2$$

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 dist. with a mixture of Gaussians

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

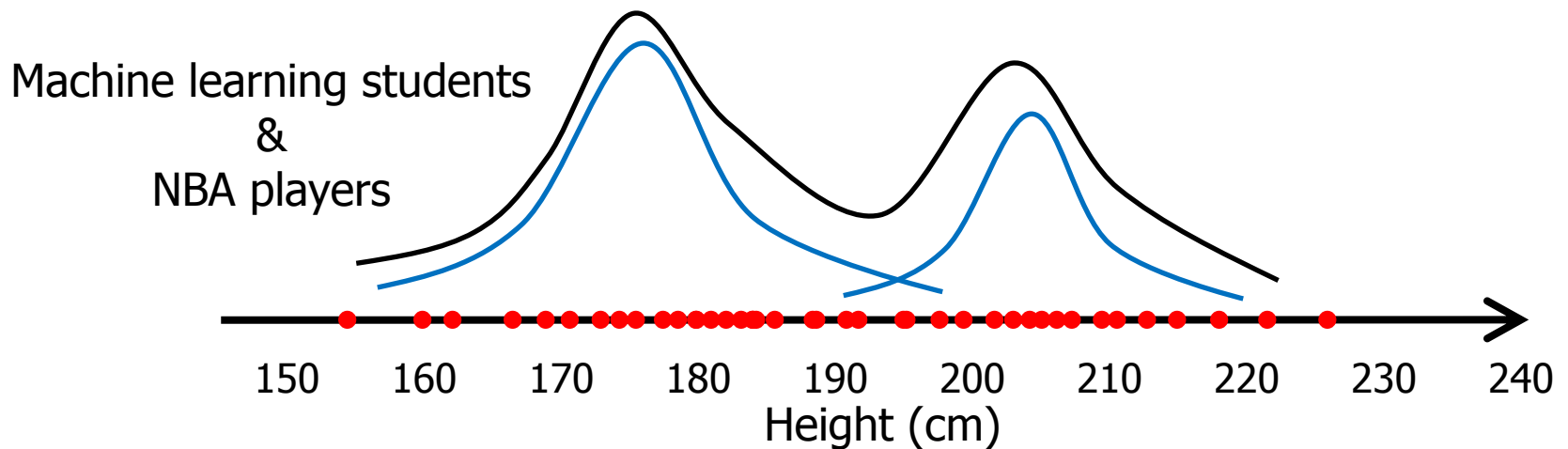
z : a **membership** r.v. indicating which Gaussian that x belongs to.

Weight of j -th Gaussian.
Often notated as w_j

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

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 $X = (x_1, \dots, x_n)$, $x_i \sim p(x)$, log-likelihood is

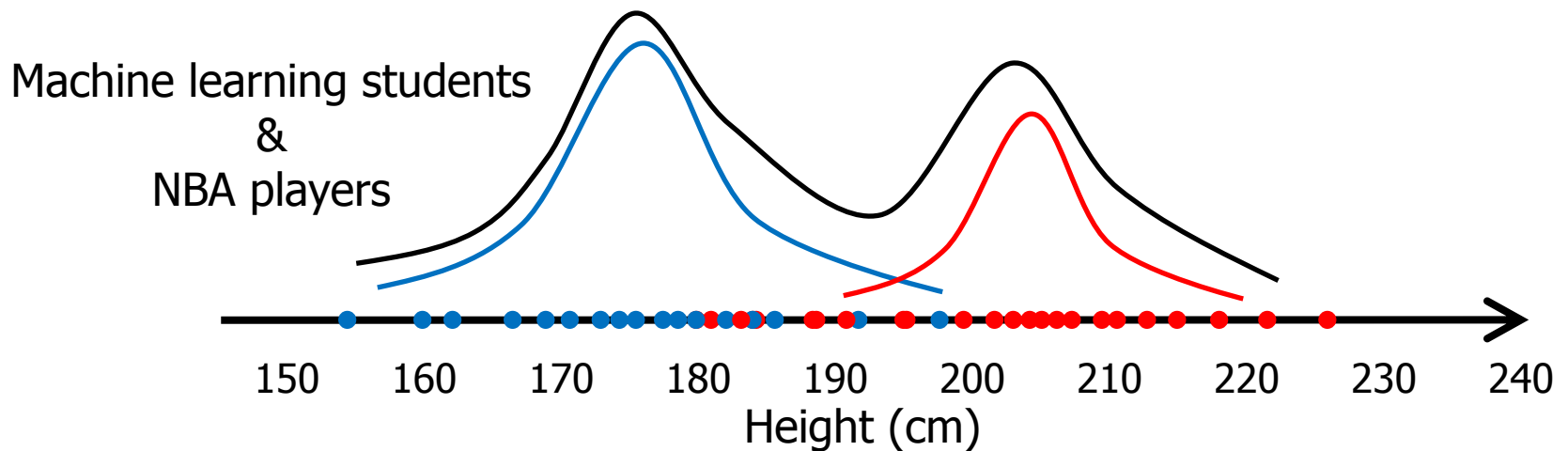
$$\begin{aligned} L(\theta|X) &= \sum_{i=1}^N \log p(x_i) \\ &= \sum_{i=1}^N \log \left\{ \sum_{j=1}^K P(z_i = j) \cdot p(x_i | z_i = j) \right\} \\ &= \sum_{i=1}^N \log \left\{ \sum_{j=1}^K w_j \cdot \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x_i - \mu_j)^2}{2\sigma_j^2}} \right\} \end{aligned}$$

- Try to solve parameters (μ_j, σ_j^2, w_j) by setting their partial derivatives to 0?
- No closed form solution. (Try it yourself)

Why is ML hard 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 as x_i , i.e. we are **uncertain** about which Gaussian x_i comes from.
- z_i is a **latent variable** because we can't observe it.
- Latent variables can also be viewed as **missing data**, data that we didn't observe.

If we know what value z_i takes, ML is easy



- $w_j = \frac{1}{N} \sum_i^N 1\{z_i = j\}$

Indicator function:

- $\mu_j = \frac{\sum_i^N 1\{z_i=j\}x_i}{\sum_i^N 1\{z_i=j\}}$

$$1\{z_i = j\} = \begin{cases} 1, & \text{if } z_i = j; \\ 0, & \text{if } z_i \neq j. \end{cases}$$

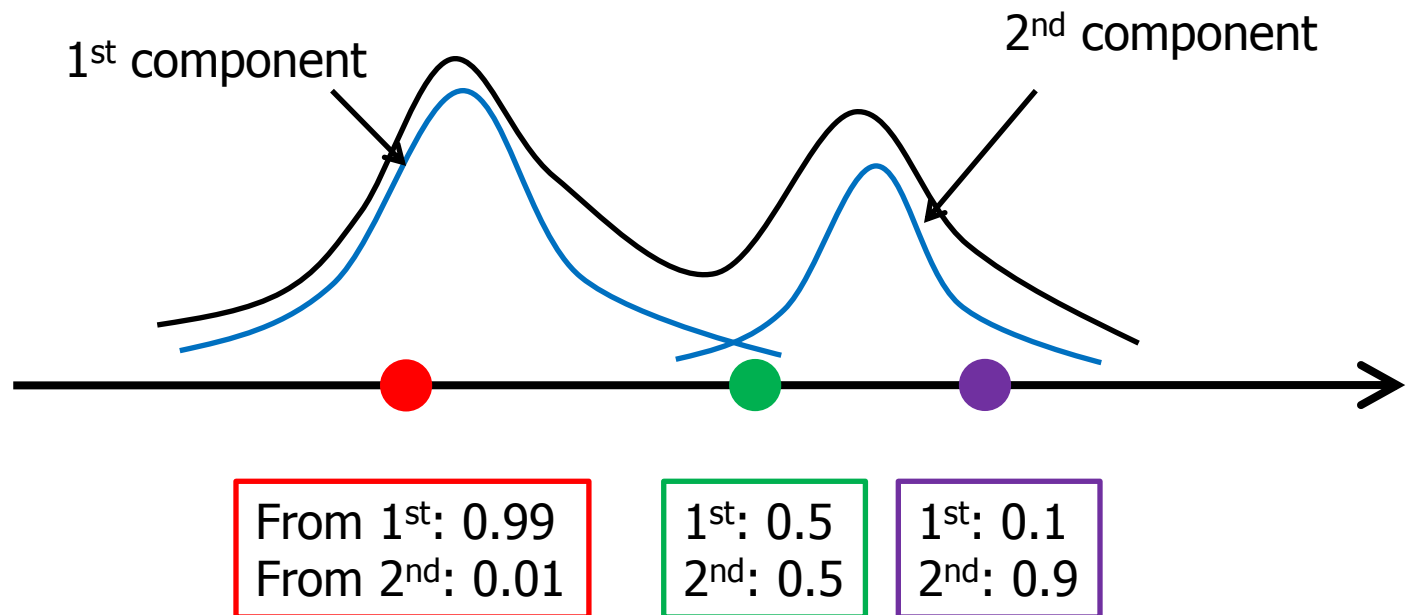
- $\sigma_j^2 = \frac{\sum_i^N 1\{z_i=j\}(x_i - \mu)^2}{\sum_i^N 1\{z_i=j\}}$

N = number of training examples

Illustration of “Soft” Membership

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

$$q_i^{(j)} \equiv P(z_i = j | x_i)$$



Improving our posterior probability

- The “posterior probability” of a Gaussian is the probability that this Gaussian generated the data we observe.
- Let’s find a way to use posterior probabilities to make an algorithm that automatically creates a set of Gaussians 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, EM algorithm.
- EM algorithm updates parameters iteratively.
- In each iteration, the likelihood value increases (at least it doesn't decrease).
- EM algorithm always converges (to some local optimum), i.e. likelihood value and parameters converge.

EM Algorithm Summary

- Initialize parameters.
 w_j, μ_j, σ^2_j for each Gaussian j in our model.
- E step: calculate posterior dist. of latent variables
probability that these Gaussians generated the data
- M step: update parameters.
update w_j, μ_j, σ^2_j for each Gaussian j
- Repeat E and M steps until convergence.
go until parameters don't 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 (w_j, μ_j, σ^2_j) 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 dist. of z_i , using **current parameters**.

$$\begin{aligned} q_i^{(j)} &\equiv P(z_i = j | x_i) = \frac{P(z_i = j, x_i)}{p(x_i)} \\ &= \frac{p(x_i | z_i = j) \boxed{P(z_i = j)}}{\sum_{l=1}^K p(x_i | z_i = l) P(z_i = l)} \end{aligned}$$

Prior dist. of component j 

Bayes rule

- 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 $q_i^{(j)}$ is the “soft” membership of x_i of the j -th Gaussian.

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

Average of their membership

$$\mu_j = \frac{\sum_{i=1}^N q_i^{(j)} x_i}{\sum_{i=1}^N q_i^{(j)}}$$

Weighted average using membership

$$\sigma_j^2 = \frac{\sum_{i=1}^N q_i^{(j)} (x_i - \mu_j)^2}{\sum_{i=1}^N q_i^{(j)}}$$

Weighted variance using membership

- Repeat E step and M step until convergence.
 - Convergence criterion in practice: compare with the previous iteration, the likelihood value doesn't increase much, or the parameters don't change much.

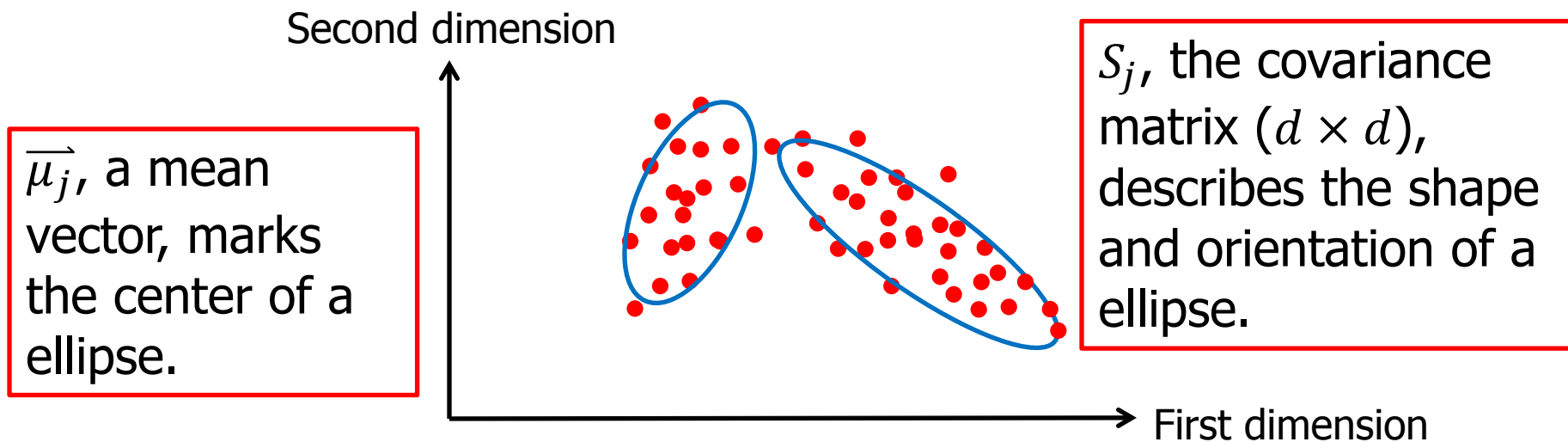
EM Algorithm Summary

- Initialize parameters.
 w_j, μ_j, σ^2_j for each Gaussian j in our model.
- E step: calculate posterior dist. of latent variables
probability that these Gaussians generated the data
- M step: update parameters.
update w_j, μ_j, σ^2_j for each Gaussian j
- Repeat E and M steps until convergence.
go until parameters don't 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



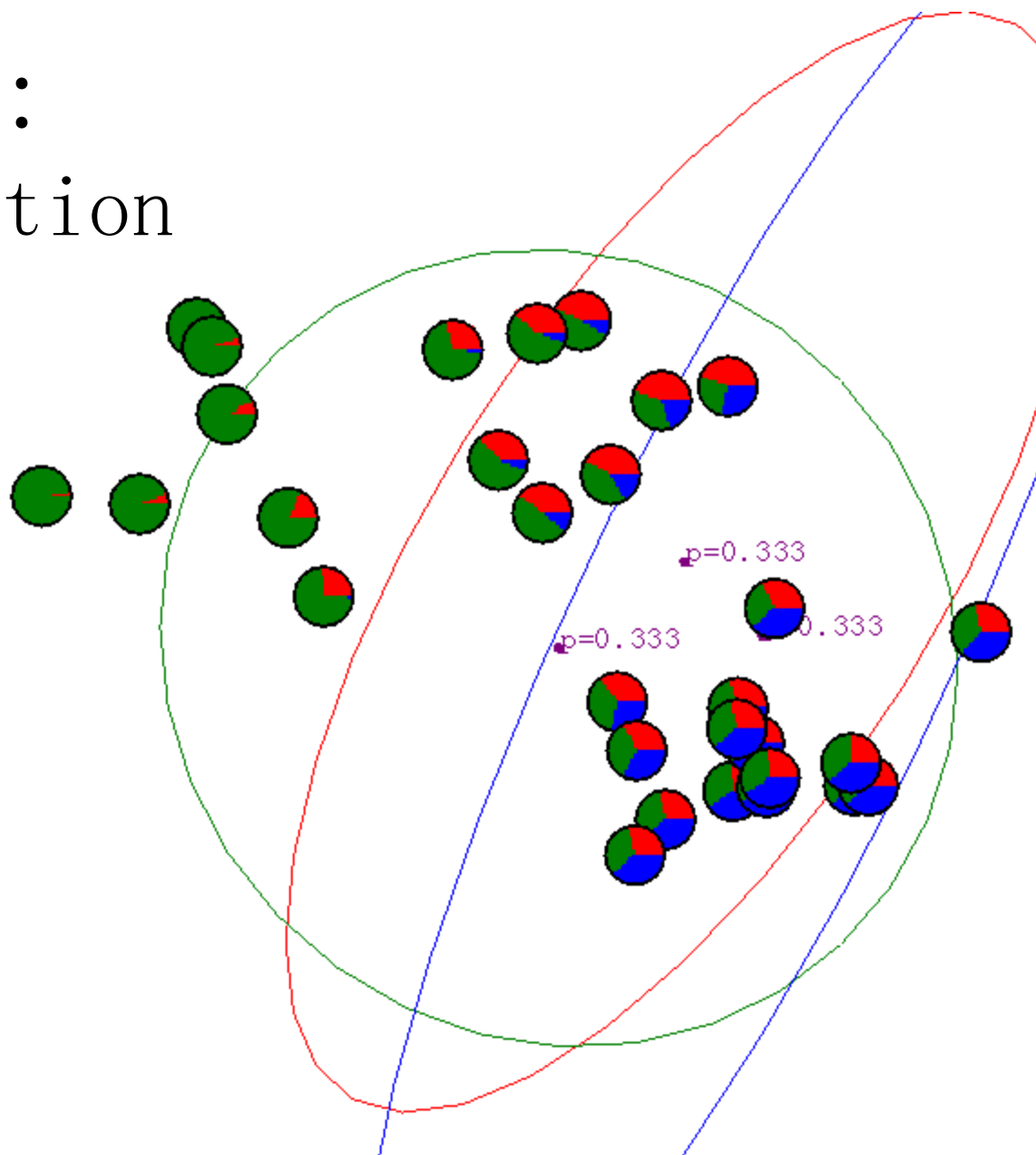
$$p(\vec{x}) = \sum_{j=1}^K w_j \cdot \frac{1}{(2\pi)^{\frac{d}{2}} |S_j|^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2} (\vec{x} - \vec{\mu}_j)^T S_j^{-1} (\vec{x} - \vec{\mu}_j) \right\}$$

d : dimensionality

- Parameters: $(\vec{\mu}_j, S_j, w_j)$ for $j = 1, \dots, K$, with $\sum_{j=1}^K w_j = 1$.

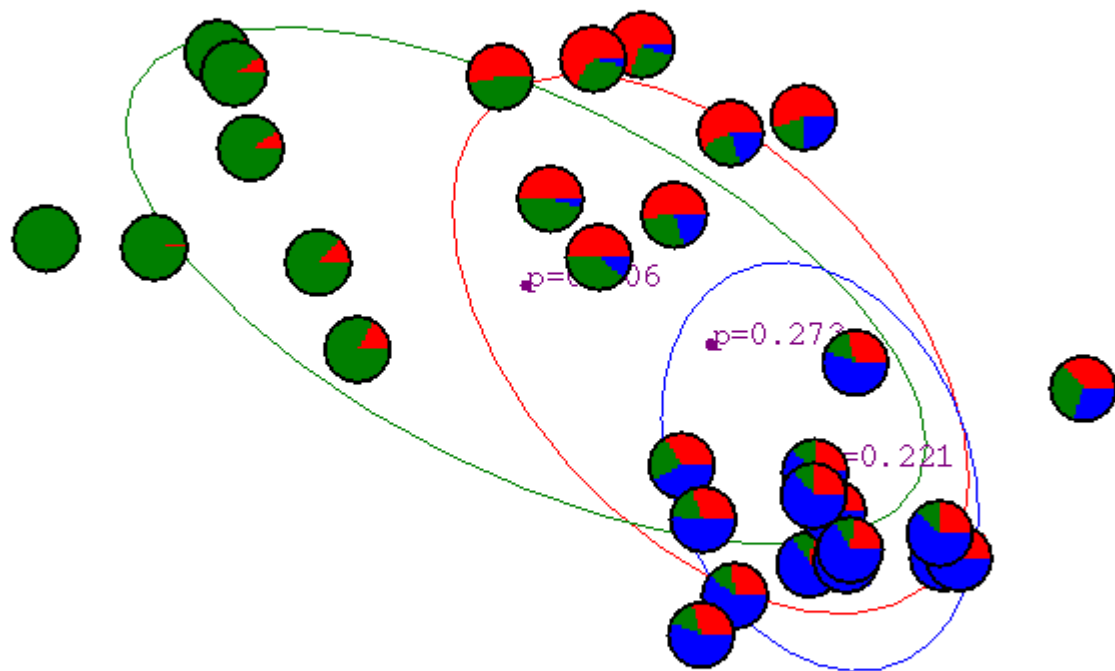
- How many parameters? $\overset{\text{means}}{\downarrow} dK + \overset{\text{covariance's}}{\downarrow} \frac{d(d+1)}{2} K + \overset{\text{weights}}{\downarrow} K$

Example: Initialization



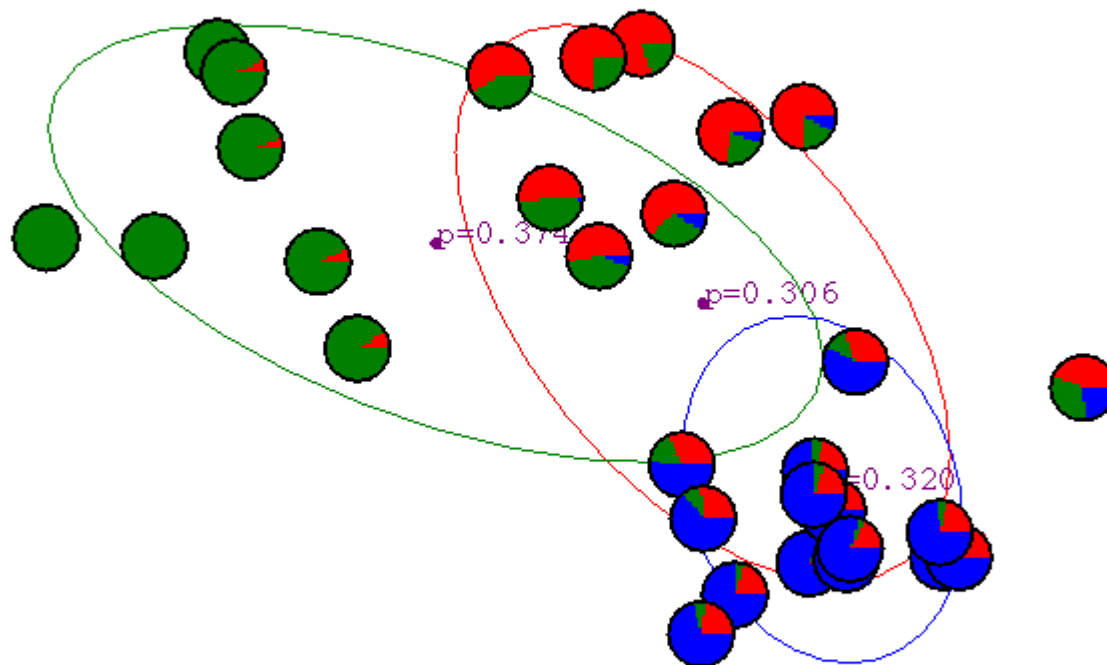
(Illustration from Andrew Moore's tutorial slides on GMM)

After Iteration #1



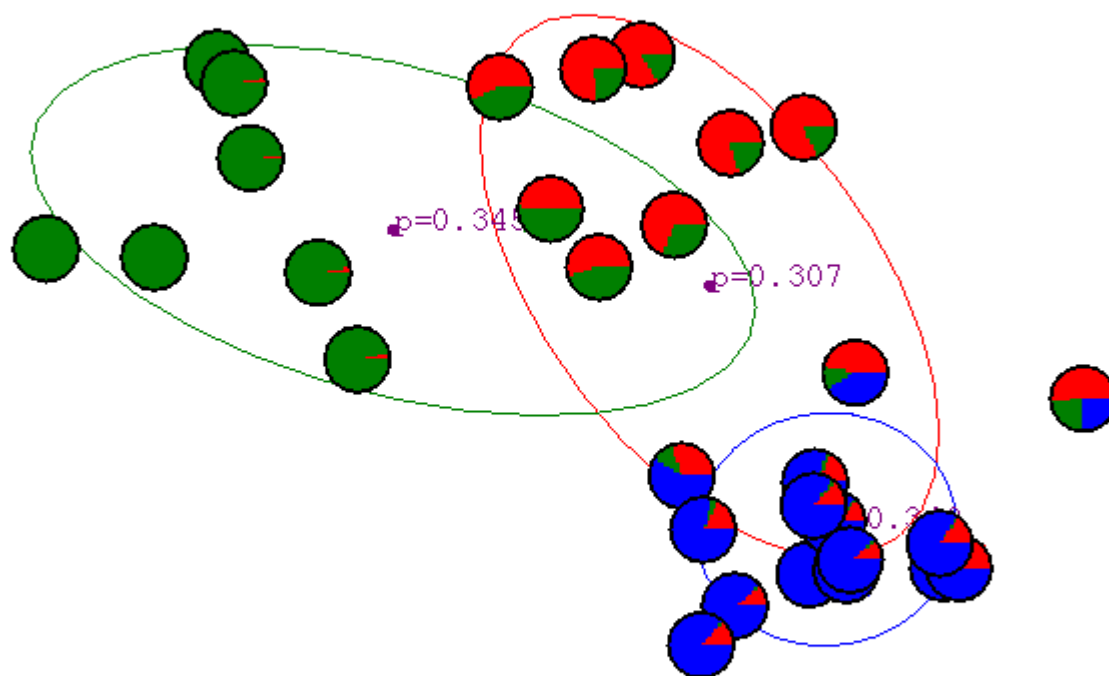
(Illustration from Andrew
Moore's tutorial slides on
GMM)

After Iteration #2



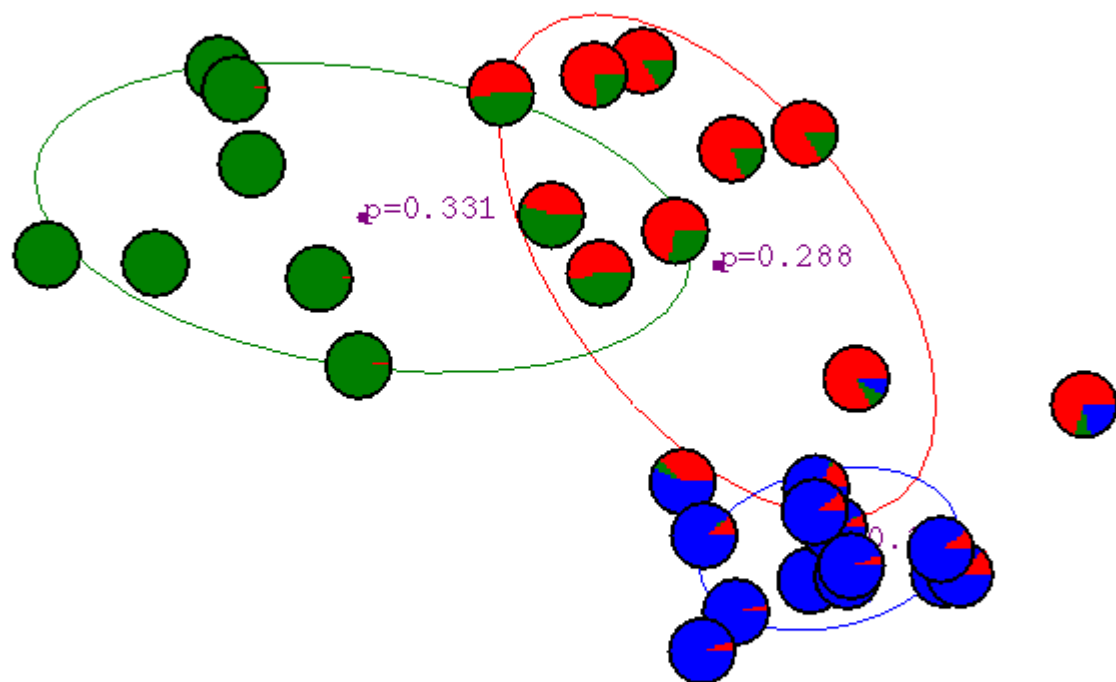
(Illustration from Andrew
Moore's tutorial slides on
GMM)

After Iteration #3



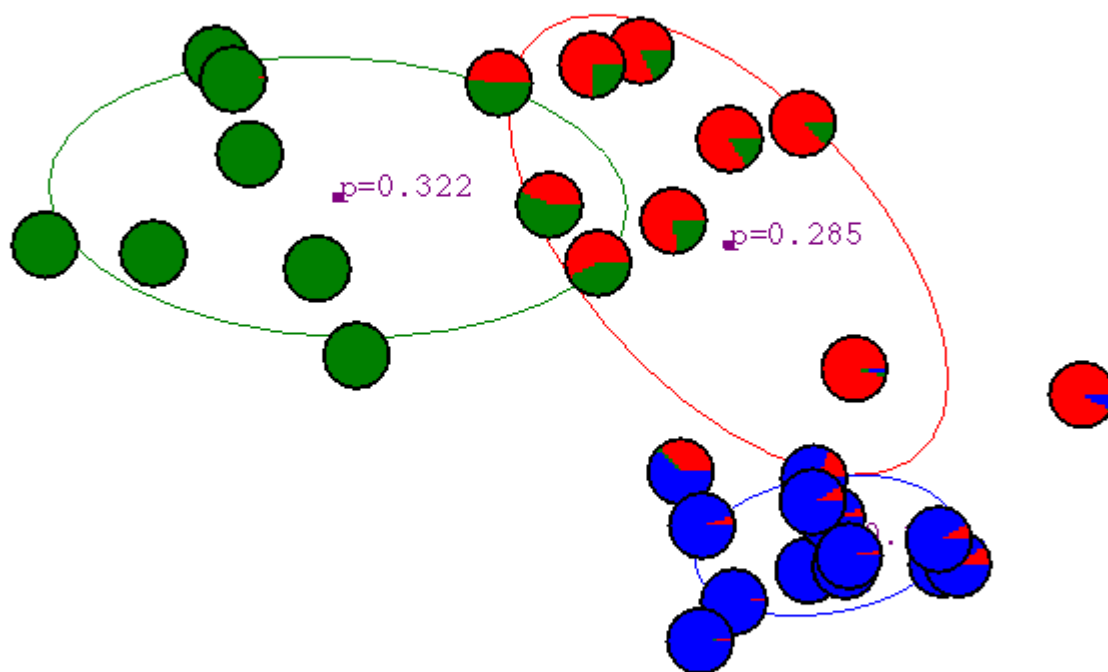
(Illustration from Andrew
Moore's tutorial slides on
GMM)

After Iteration #4



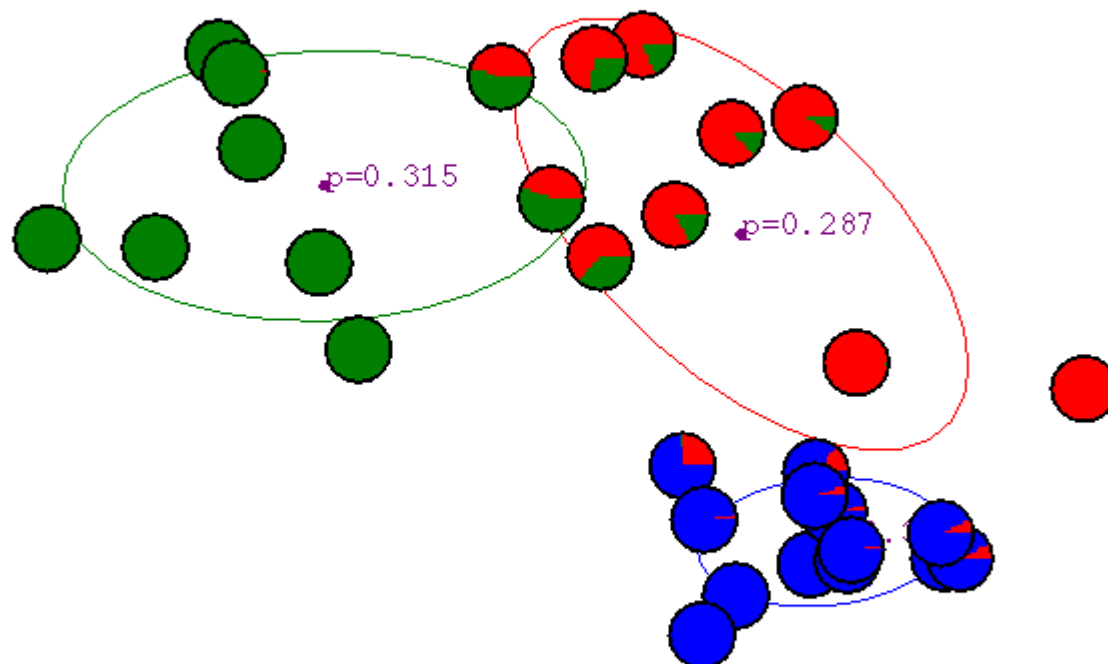
(Illustration from Andrew Moore's tutorial slides on GMM)

After Iteration #5



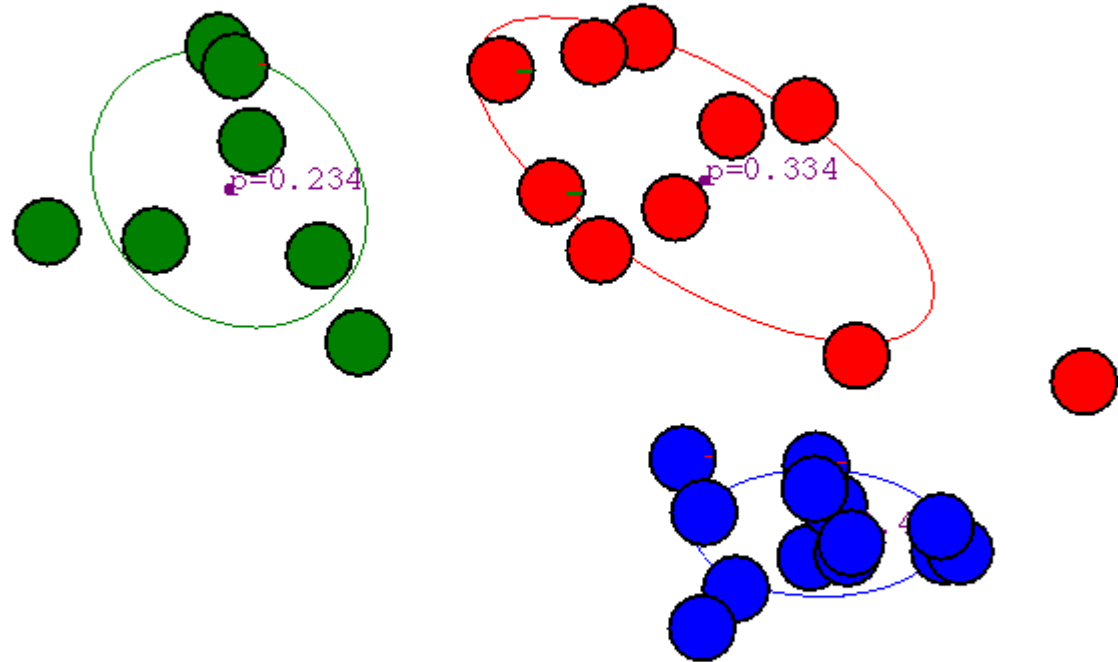
(Illustration from Andrew
Moore's tutorial slides on
GMM)

After Iteration #6



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

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 overfitted.
 - Likelihood 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

- Similarity
 - 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 variances in addition to means.

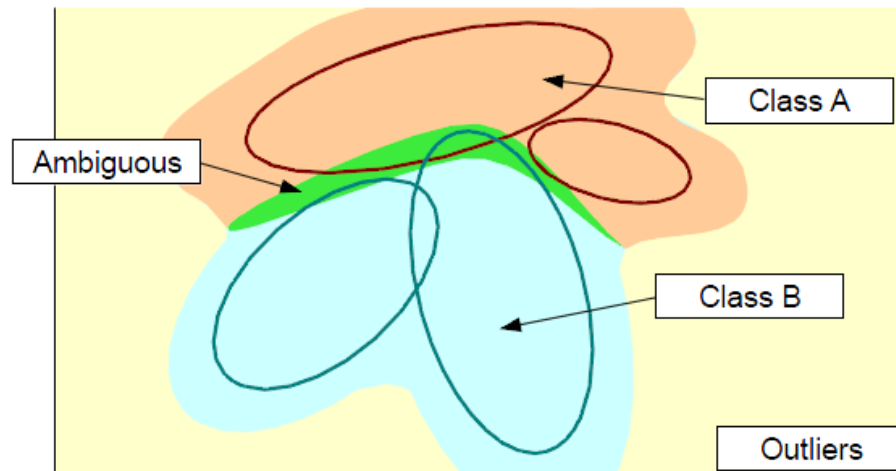
GMM for Classification

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
optimal
classifier

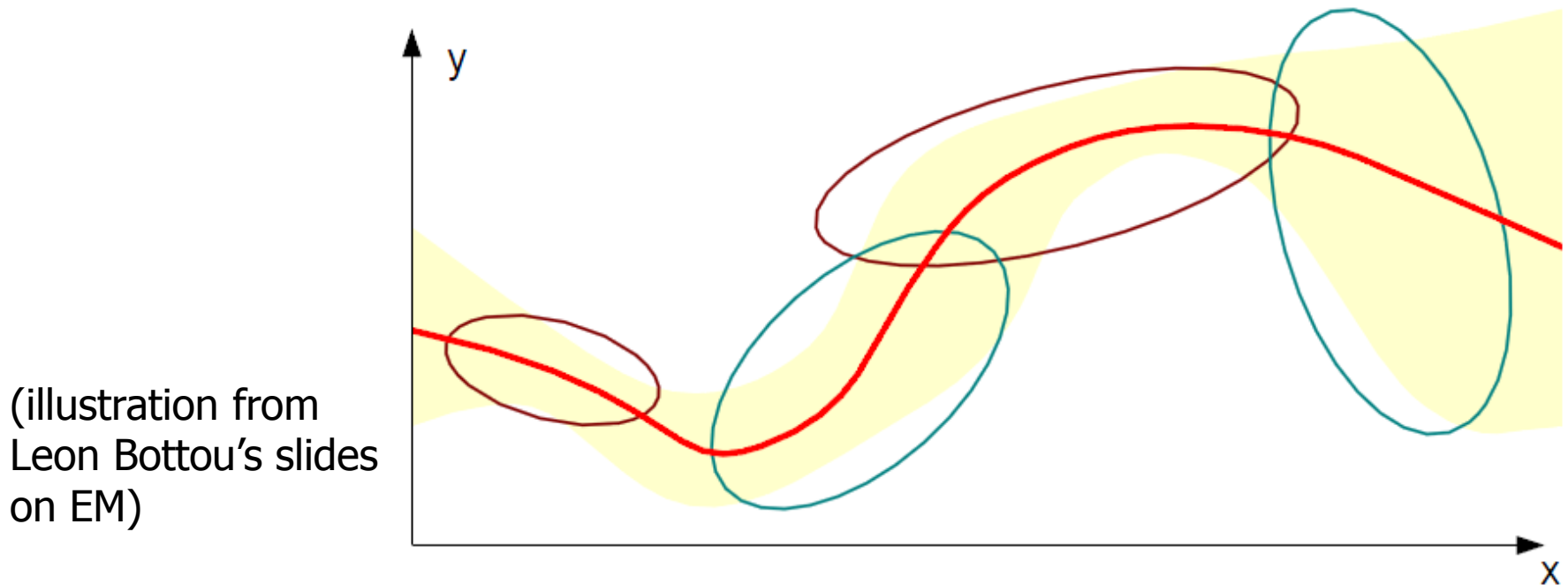
4. Classify x to the class having largest posterior.



(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



You Should Know

- How to do maximum likelihood (ML) estimation?
- GMM models data dist. with a mixture of K Gaussians, with para $(\vec{\mu}_j, S_j, w_j)$, for $j = 1, \dots, K$.
- No closed form solution for ML estimation of GMM parameters.
- How to estimate GMM parameters with EM algorithm?
- How is GMM related to K-means?
- How to use GMM for clustering, classification and regression?