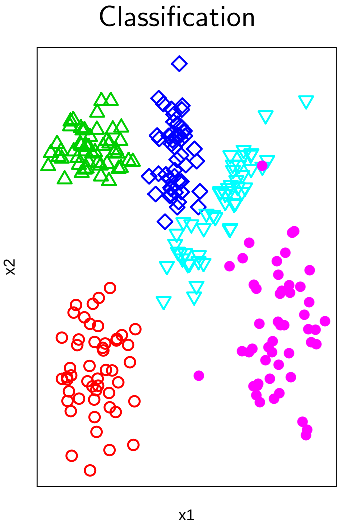
Lecture 4: Probabilistic Learning

DD2431

Giampiero Salvi

Autumn, 2015

Classification with Probability Distributions



$\mathbf{x} \leftarrow$ features

$y \in \{y_1, \dots, y_K\} \leftarrow$ class

$$\hat{k} = \arg\max_k P(y_k|\mathbf{x})$$
$$= \arg\max_k P(y_k)P(\mathbf{x}|y_k)$$

Estimation Theory

in the last lecture we assumed we knew:

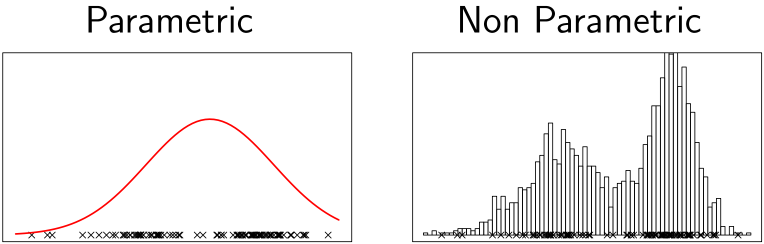
- $P(y) \leftarrow$ *Prior*
- $P(x \mid y) \leftarrow$ *Likelihood*
- $P(x) \leftarrow$ *Evidence*

and we used them to compute the *Posterior* $P(y \mid x)$

How can we obtain this information from observations (data)?

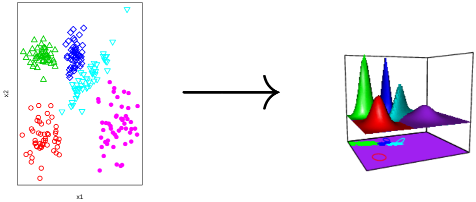
Estimation Theory $\equiv$ Learning

Parametric vs Non-Parametric Estimation



Bayesian non-parametric methods integrate out the parameters.

Assumption # 1: Class Independence

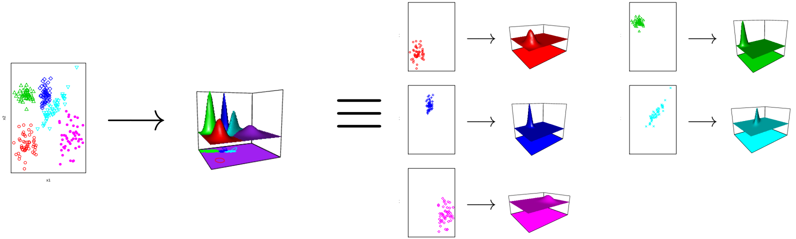


Assumptions:

- samples from class i do not influence estimate for class j, $i \neq j$
- Generative vs discriminative models

Parameter estimation (cont.)

- class independence assumption:



- each distribution is a likelihood in the form $P(\mathbf{x}|\theta_i)$ for class i
- in the following we drop the class index and talk about $P(\mathbf{x}|\theta)$

Assumption #2: i.i.d.

Samples from each class are independent and identically distributed:

$$\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$

The likelihood of the whole data set can be factorized:

$$P(\mathcal{D}|\theta) = P(\mathbf{x}_1, \dots, \mathbf{x}_N|\theta) = \prod_{i=1}^{N} P(\mathbf{x}_i|\theta)$$

And the log-likelihood becomes:

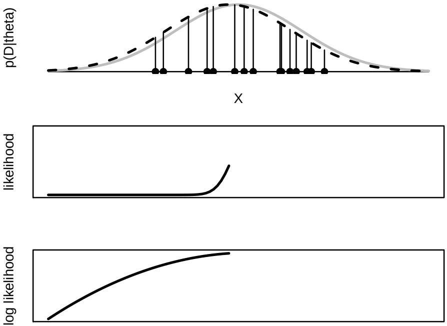
$$\log P(\mathcal{D}|\theta) = \sum_{i=1}^{N} \log P(\mathbf{x}_i|\theta)$$

Three Approaches

- Maximum Likelihood (ML)
- Maximum A Posteriori (MAP)
- Bayesian

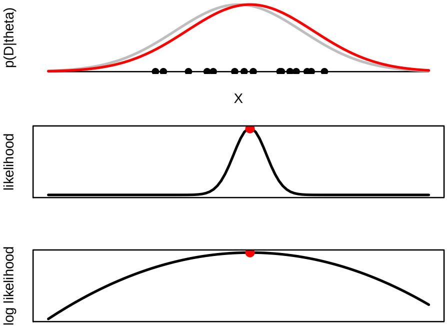
Maximum likelihood estimation: Illustration

Find parameter vector $\hat{\theta}$ that maximizes $P(\mathcal{D}|\theta)$ with $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$



Maximum likelihood estimation: Illustration

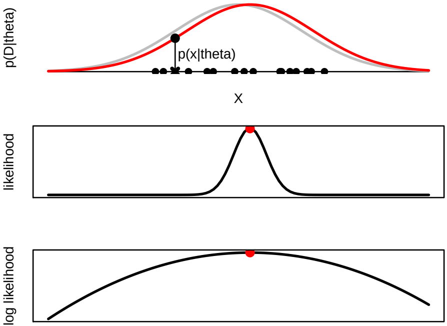
Find parameter vector $\hat{\theta}$ that maximizes $P(\mathcal{D}|\theta)$ with $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$



1. estimate the optimal parameters of the model

Maximum likelihood estimation: Illustration

Find parameter vector $\hat{\theta}$ that maximizes $P(\mathcal{D}|\theta)$ with $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$



1. estimate the optimal parameters of the model
2. evaluate the predictive distribution on new data points

ML estimation of Gaussian mean

$$N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \text{ with } \theta = \{\mu, \sigma^2\}$$

Log-likelihood of data (i.i.d. samples):

$$\log P(\mathcal{D}|\theta) = \sum_{i=1}^{N} \log N(x_i|\mu,\sigma^2) = -N \log\left(\sqrt{2\pi\sigma}\right) - \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{2\sigma^2}$$

$$0 = \frac{d \log P(\mathcal{D}|\theta)}{d\mu} = \sum_{i=1}^{N} \frac{(x_i-\mu)}{\sigma^2} = \frac{\sum_{i=1}^{N} x_i - N\mu}{\sigma^2} \iff$$

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

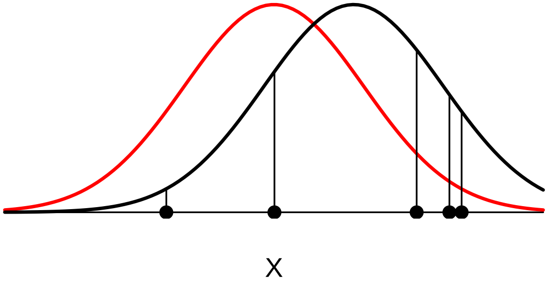
ML estimation of Gaussian parameters

$$\begin{aligned}\hat{\mu} &= \frac{1}{N}\sum_{i=1}^{N} x_i \\ \hat{\sigma}^2 &= \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2\end{aligned}$$

- same result by minimizing the sum of square errors!
- but we make assumptions explicit

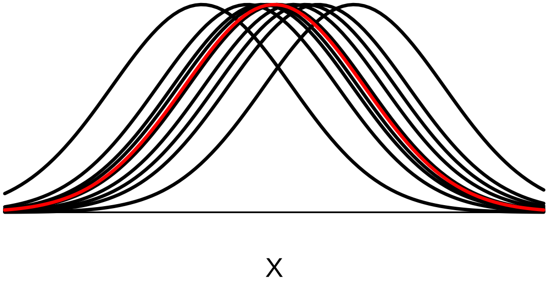
Problem: few data points

10 repetitions with 5 points each



Problem: few data points

10 repetitions with 5 points each



Maximum a Posteriori Estimation

$$\hat{\mu}, \hat{\sigma}^2 = \arg\max_{\mu,\sigma^2} \left[\prod_{i=1}^{N} P(x_i|\mu,\sigma^2) P(\mu,\sigma^2) \right]$$

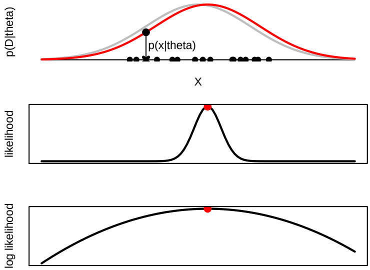
where the prior $P(\mu, \sigma^2)$ needs a nice mathematical form for closed solution

$$\begin{aligned} \hat{\mu}_{\text{MAP}} &= \frac{N}{N+\gamma}\,\hat{\mu}_{\text{ML}} + \frac{\gamma}{N+\gamma}\,\delta \\ \hat{\sigma}^2_{\text{MAP}} &= \frac{N}{N+3+2\alpha}\hat{\sigma}^2_{\text{ML}} + \frac{2\beta+\gamma(\delta+\hat{\mu}_{\text{MAP}})^2}{N+3+2\alpha} \end{aligned}$$

where $\alpha, \beta, \gamma, \delta$ are parameters of the prior distribution

ML, MAP and Point Estimates

- Both ML and MAP produce point estimates of θ
- Assumption: there is a true value for θ
- advantage: once $\hat{\theta}$ is found, everything is known



Bayesian estimation

- Consider θ as a random variable
- characterize θ with the posterior distribution $P(\theta|\mathcal{D})$ given the data

$$\begin{array}{lll} \text{ML:} & \mathcal{D} & \rightarrow & \hat{\theta}_{\text{ML}} \\ \text{MAP:} & \mathcal{D}, P(\theta) & \rightarrow & \hat{\theta}_{\text{MAP}} \\ \text{Bayes:} & \mathcal{D}, P(\theta) & \rightarrow & P(\theta|\mathcal{D}) \end{array}$$

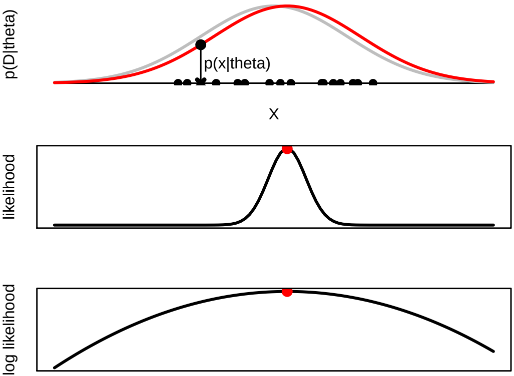
- for new data points, instead of $P(\mathbf{x}_{\text{new}}|\hat{\theta}_{\text{ML}})$ or $P(\mathbf{x}_{\text{new}}|\hat{\theta}_{\text{MAP}})$, compute:

$$P(\mathbf{x}_{\text{new}}|\mathcal{D}) = \int_{\theta\in\Theta} P(\mathbf{x}_{\text{new}}|\theta)P(\theta|\mathcal{D})d\theta$$

Bayesian estimation (cont.)

- we can compute $P(\mathbf{x}|\mathcal{D})$ instead of $P(\mathbf{x}|\hat{\theta})$
- integrate the joint density $P(\mathbf{x}, \theta|\mathcal{D}) = P(\mathbf{x}|\theta)P(\theta|\mathcal{D})$

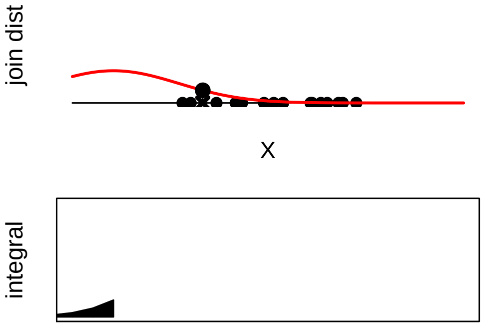
$P(\mathbf{x}|\hat{\theta})$



Bayesian estimation

- we can compute $P(\mathbf{x}|\mathcal{D})$ instead of $P(\mathbf{x}|\hat{\theta})$
- integrate the joint density $P(\mathbf{x}, \theta|\mathcal{D}) = P(\mathbf{x}|\theta)P(\theta|\mathcal{D})$

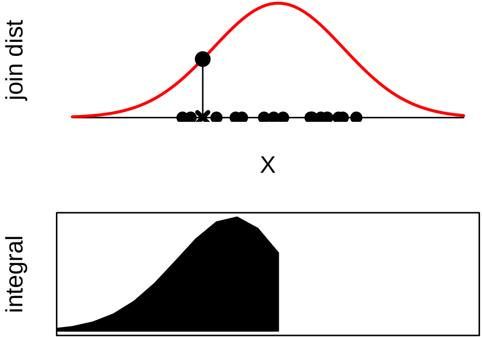
$$P(\mathbf{x}|\mathcal{D}) = \int P(\mathbf{x}|\theta)P(\theta|\mathcal{D})d\theta$$



Bayesian estimation

- we can compute $P(\mathbf{x}|\mathcal{D})$ instead of $P(\mathbf{x}|\hat{\theta})$
- integrate the joint density $P(\mathbf{x}, \theta|\mathcal{D}) = P(\mathbf{x}|\theta)P(\theta|\mathcal{D})$

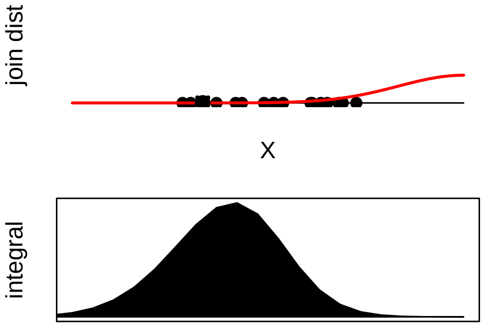
$$P(\mathbf{x}|\mathcal{D}) = \int P(\mathbf{x}|\theta)P(\theta|\mathcal{D})d\theta$$



Bayesian estimation

- we can compute $P(\mathbf{x}|\mathcal{D})$ instead of $P(\mathbf{x}|\hat{\theta})$
- integrate the joint density $P(\mathbf{x}, \theta|\mathcal{D}) = P(\mathbf{x}|\theta)P(\theta|\mathcal{D})$

$$P(\mathbf{x}|\mathcal{D}) = \int P(\mathbf{x}|\theta)P(\theta|\mathcal{D})d\theta$$



Bayesian estimation (cont.)

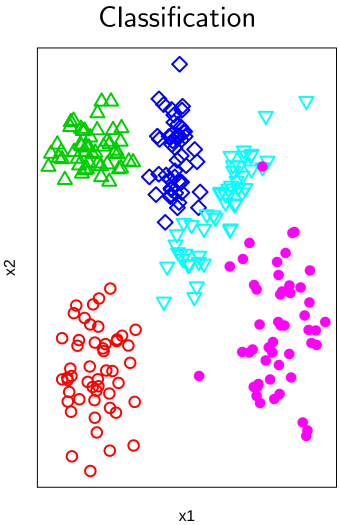
Pros:

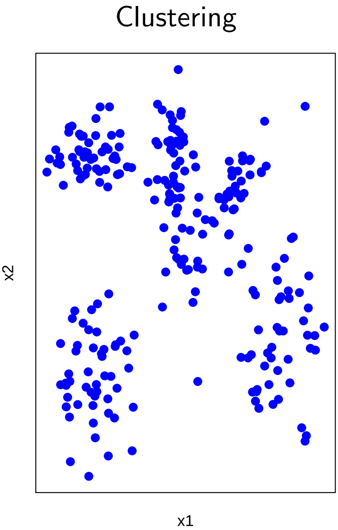
- better use of the data
- makes a priori assumptions explicit
- can be implemented recursively (if conjugate prior)
  - use posterior $P(\theta|\mathcal{D})$ as new prior
- reduce overfitting

Cons:

- definition of noninformative priors can be tricky
- often requires numerical integration
- not widely accepted by traditional statistics (frequentism)

Clustering vs Classification





Fitting complex distributions

We can try to fit a mixture of K distributions:

$$P(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k P(x|\theta_k),$$

$$\text{with } \theta = \{\pi_1, \dots, \pi_k, \theta_1, \dots, \theta_K\}$$

Problem:

We do not know which point has been generated by which component of the mixture

We cannot optimize $P(\mathbf{x}|\theta)$ directly

Expectation Maximization

Fitting model parameters with missing (latent) variables

$$P(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k P(x|\theta_k),$$

$$\text{with } \theta = \{\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_K\}$$

- very general idea (applies to many different probabilistic models)
- augment the data with the missing variables: h_{ik} probability that each data point x_i was generated by each component of the mixture k
- optimize the Likelihood of the complete data:

$$P(\mathbf{x}, \mathbf{h}|\theta)$$

Heuristic Example: K-means

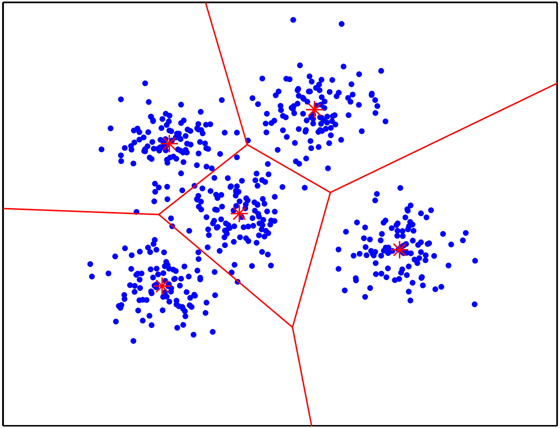
- describes each class with a centroid
- a point belongs to a class if the corresponding centroid is closest (Euclidean distance)
- iterative procedure
- guaranteed to converge
- not guaranteed to find the optimal solution
- used in vector quantization (since the 1950's)

K-means: algorithm

Data: k (number of desired clusters), n data points $\mathbf{x}_i$
Result: k clusters
initialization: assign initial value to k centroids $\mathbf{c}_i$;
repeat
    assign each point $\mathbf{x}_i$ to closest centroid $\mathbf{c}_j$;
    compute new centroids as mean of each group of points;
until *centroids do not change*;
return k clusters;

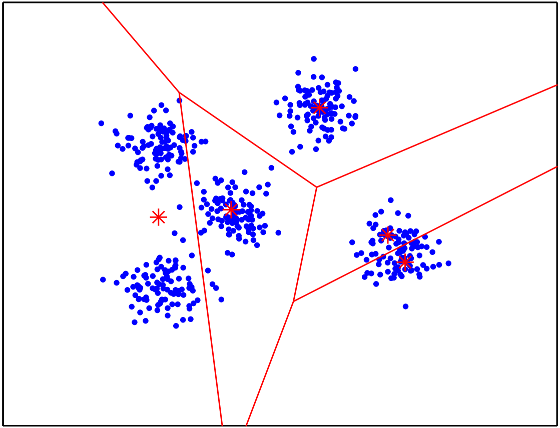
K-means: example

iteration 20, update clusters



K-means: sensitivity to initial conditions

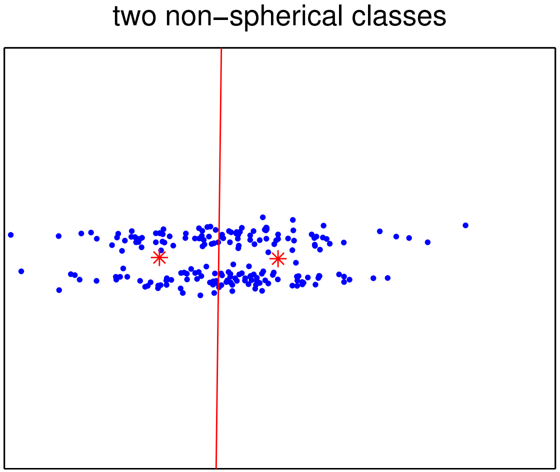
iteration 20, update clusters



K-means: limits of Euclidean distance

- the Euclidean distance is isotropic (same in all directions in $\mathbb{R}^p$)
- this favours spherical clusters
- the size of the clusters is controlled by their distance

K-means: non-spherical classes



Expectation Maximization

Fitting model parameters with missing (latent) variables

$$P(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k P(x|\theta_k),$$

$$\text{with } \theta = \{\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_K\}$$

- very general idea (applies to many different probabilistic models)
- augment the data with the missing variables: h_{ik} probability of assignment of each data point x_i to each component of the mixture k
- optimize the Likelihood of the complete data:

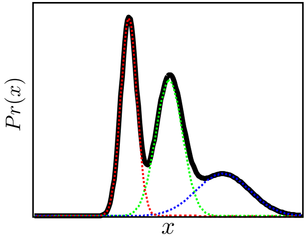
$$P(\mathbf{x}, \mathbf{h}|\theta)$$

Mixture of Gaussians

This distribution is a weight sum of K Gaussian distributions

$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2)$$

where $\pi_1 + \cdots + \pi_K = 1$
and $\pi_k > 0$ $(k = 1, \ldots, K)$.



This model can describe **complex multi-modal** probability distributions by combining simpler distributions.

Mixture of Gaussians

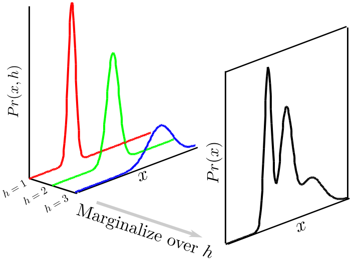
$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2)$$

- Learning the parameters of this model from training data $x_1, \ldots, x_n$ is not trivial - using the usual straightforward maximum likelihood approach.
- Instead learn parameters using the **Expectation-Maximization** (EM) algorithm.

Mixture of Gaussians as a marginalization

We can interpret the Mixture of Gaussians model with the introduction of a discrete hidden/latent variable h and $P(x, h)$:

$$P(x) = \sum_{k=1}^{K} P(x, h = k) = \sum_{k=1}^{K} P(x \mid h = k) P(h = k)$$
$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2)$$



$\leftarrow$ **mixture density**

Figures taken from **Computer Vision: models, learning and inference** by Simon Prince.

EM for two Gaussians

Assume: We know the pdf of x has this form:

$$P(x) = \pi_1 \mathcal{N}(x; \mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(x; \mu_2, \sigma_2^2)$$

where $\pi_1 + \pi_2 = 1$ and $\pi_k > 0$ for components $k = 1, 2$.

Unknown: Values of the parameters (Many!)

$$\Theta = (\pi_1, \mu_1, \sigma_1, \mu_2, \sigma_2).$$

Have: Observed n samples $x_1, \dots, x_n$ drawn from $P(x)$.

Want to: Estimate Θ from $x_1, \dots, x_n$.

How would it be possible to get them all???

EM for two Gaussians

For each sample x_i introduce a *hidden variable* h_i

$$h_i = \begin{cases} 1 & \text{if sample } x_i \text{ was drawn from } \mathcal{N}(x; \mu_1, \sigma_1^2) \\ 2 & \text{if sample } x_i \text{ was drawn from } \mathcal{N}(x; \mu_2, \sigma_2^2) \end{cases}$$

and come up with initial values

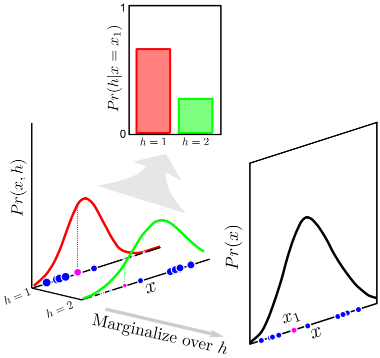
$$\Theta^{(0)} = (\pi_1^{(0)}, \mu_1^{(0)}, \sigma_1^{(0)}, \mu_2^{(0)}, \sigma_2^{(0)})$$

for each of the parameters.

EM is an *iterative algorithm* which updates $\Theta^{(t)}$ using the following two steps...

EM for two Gaussians: E-step

The responsibility of k-th Gaussian for each sample x (indicated by the size of the projected data point)



Look at each sample x along hidden variable h in the E-step

Figure from **Computer Vision: models, learning and inference** by Simon Prince.

EM for two Gaussians: E-step (cont.)

E-step: Compute the *"posterior probability"* that x_i was generated by component k given the current estimate of the parameters $\Theta^{(t)}$. (responsibilities)

for $i = 1, \dots n$

for $k = 1, 2$

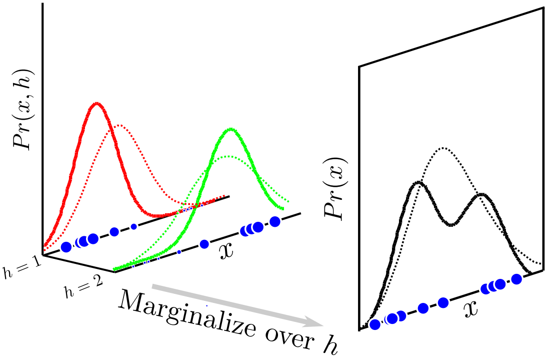
$$\gamma_{ik}^{(t)} = P(h_i = k \,|\, x_i, \Theta^{(t)})$$

$$= \frac{\pi_k^{(t)} \mathcal{N}(x_i; \mu_k^{(t)}, \sigma_k^{(t)})}{\pi_1^{(t)} \mathcal{N}(x_i; \mu_1^{(t)}, \sigma_1^{(t)}) + \pi_2^{(t)} \mathcal{N}(x_i; \mu_2^{(t)}, \sigma_2^{(t)})}$$

Note: $\gamma_{i1}^{(t)} + \gamma_{i2}^{(t)} = 1$ and $\pi_1 + \pi_2 = 1$

EM for two Gaussians: M-step

Fitting the Gaussian model for each of k-th constinuetnt.
Sample x_i contributes according to the responsibility γ_{ik}.



(dashed and solid lines for fit before and after update)

Look along samples x for each h in the M-step

EM for two Gaussians: M-step (cont.)

M-step: Compute the *Maximum Likelihood* of the parameters of the mixture model given out data's membership distribution, the $\gamma_i^{(t)}$'s:

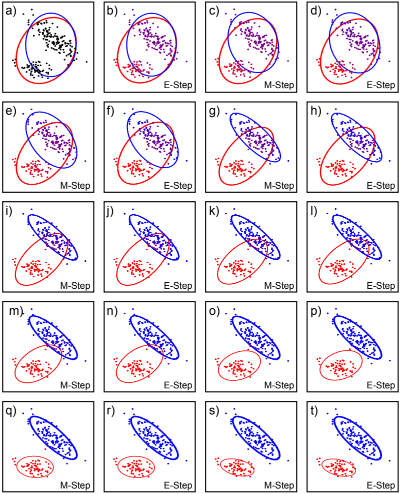
for $k = 1, 2$

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^n \gamma_{ik}^{(t)} x_i}{\sum_{i=1}^n \gamma_{ik}^{(t)}},$$

$$\sigma_k^{(t+1)} = \sqrt{\frac{\sum_{i=1}^n \gamma_{ik}^{(t)} (x_i - \mu_k^{(t+1)})^2}{\sum_{i=1}^n \gamma_{ik}^{(t)}}},$$

$$\pi_k^{(t+1)} = \frac{\sum_{i=1}^n \gamma_{ik}^{(t)}}{n}.$$

EM in practice



EM properties

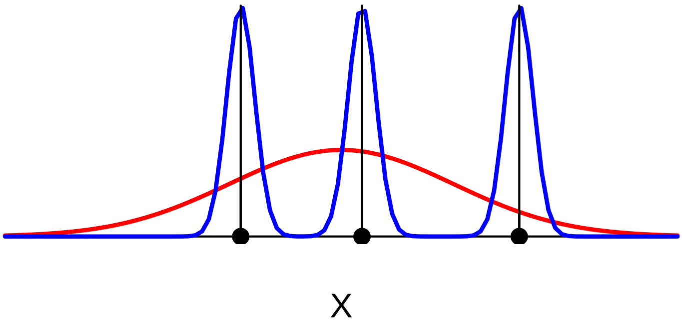
Similar to K-means

- guaranteed to find a local maximum of the complete data likelihood
- somewhat sensitive to initial conditions

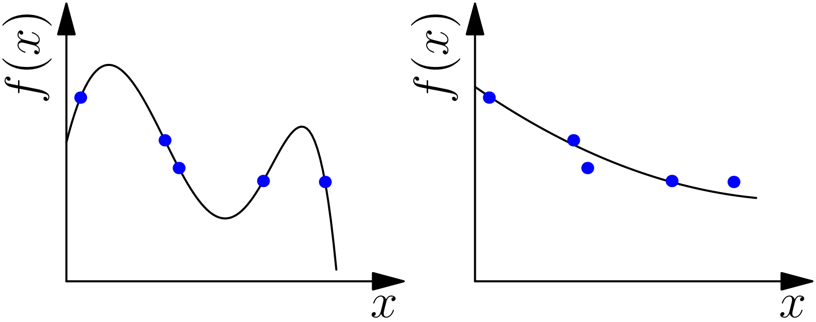
Better than K-means

- Gaussian distributions can model clusters with different shapes
- all data points are smoothly used to update all parameters

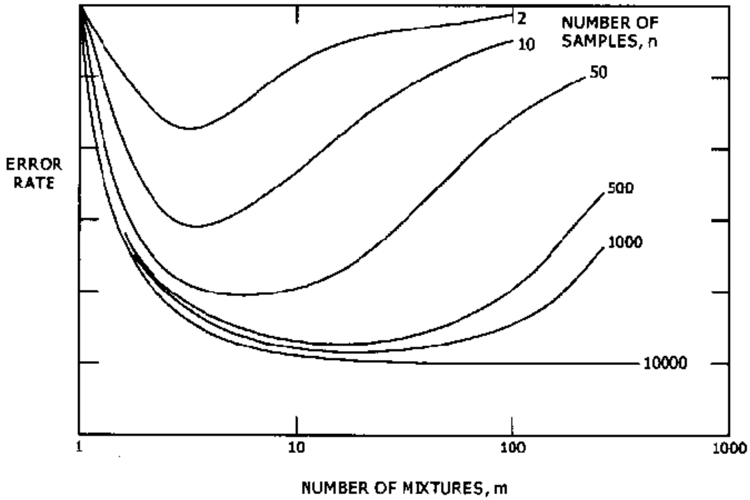
Model Selection and Overfitting



Overfitting



Overfitting: Phoneme Discrimination



Occam's Razor

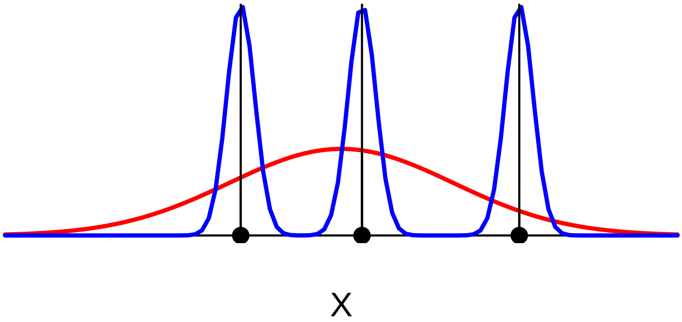
Choose the simplest explanation for the observed data

Important factors:

- number of model parameters
- number of data points
- model fit to the data

Overfitting and Maximum Likelihood

we can make the likelihood arbitrary large by increasing the number of parameters



Occam's Razor and Bayesian Learning

Remember that:

$$P(\mathbf{x}_{\text{new}}|\mathcal{D}) = \int_{\theta\in\Theta} P(\mathbf{x}_{\text{new}}|\theta)P(\theta|\mathcal{D})d\theta$$

Intuition:

More complex models fit the data very well (large $P(\mathcal{D}|\theta)$) but only for small regions of the parameter space Θ.

Summary

1. Fitting Probability Models
   - Maximum Likelihood Methods
   - Maximum A Posteriori Methods
   - Bayesian methods
2. Unsupervised Learning
   - Classification vs Clustering
   - Heuristic Example: K-means
   - Expectation Maximization
3. Model Selection and Occam's Razor

If you are interested in learning more take a look at:

C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer Verlag 2006.