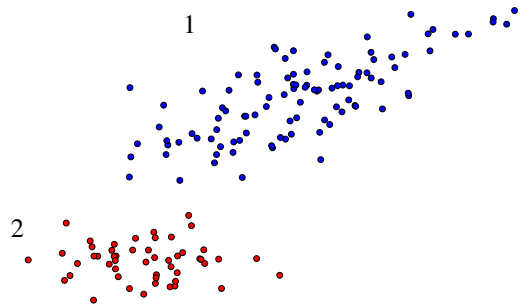
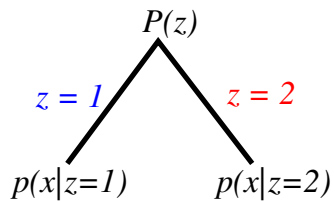


# Machine Learning

Volker Roth

Department of Mathematics & Computer Science  
University of Basel

# Chapter 9: Mixture Models



# Structure and mixtures

- Assume that input examples come in different potentially **unobserved types (groups, clusters, etc.)**.
- Assume that
  - 1 there are  $m$  underlying types  $z = 1, \dots, m$ ;
  - 2 each type  $z$  occurs with probability  $P(z)$ ;
  - 3 examples of type  $z$  distributed according to  $p(\mathbf{x}|z)$ .
- According to this model, each observed  $\mathbf{x}$  comes from a **mixture distribution:**

$$p(\mathbf{x}) = \sum_{j=1}^m \underbrace{P(z=j)}_{\pi_j} p(\mathbf{x}|z=j, \theta_j)$$

- In many practical data analysis problems (such as probabilistic clustering), we want to **estimate** such parametric models from **samples**  $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ . In particular, we are often interested in finding the types that have generated the examples.

# Mixture of Gaussians

A mixture of Gaussians model has the form

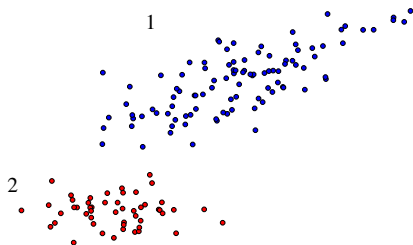
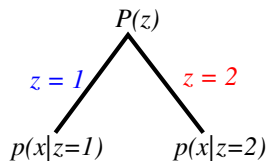
$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^m \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j),$$

where  $\boldsymbol{\theta} = \pi_1, \dots, \pi_m, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_m$  contains all the parameters.  $\{\pi_j\}$  are the **mixing proportions**.



# Mixture densities

- Data generation process:

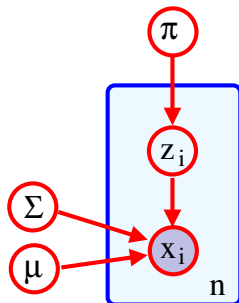


$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^m \pi_j p(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

- Any data point  $\mathbf{x}$  could have been **generated in two ways**.  
↪ the responsible component needs to be **inferred**.

# Mixtures as Latent Variable Models

- In the model  $p(\mathbf{x}|z = j, \boldsymbol{\theta})$  the class indicator variable  $z$  is **latent**. This is an example of a large class of **latent variable models** (LVM).
- Bayesian network (DAG) = graphical representation of the joint distribution of RVs (nodes) as  $P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents}(x_i))$



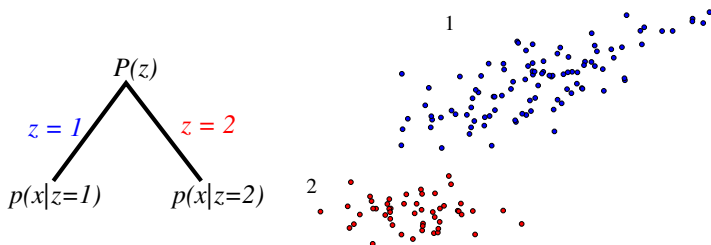
$$\begin{aligned} p(\mathbf{x}_i | \boldsymbol{\theta}) &= \sum_{z_i} p(\mathbf{x}_i, z_i | \boldsymbol{\theta}) \\ &= \sum_{z_i} p(\mathbf{x}_i | \boldsymbol{\mu}, \boldsymbol{\Sigma}, z_i) p(z_i | \boldsymbol{\pi}). \end{aligned}$$

# Mixture densities

- Consider a two component mixture of Gaussians model.

$$p(\mathbf{x}|\boldsymbol{\theta}) = \pi_1 p(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \pi_2 p(\mathbf{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

- If we knew the generating component  $z_i = \{1, 2\}$  for each example  $\mathbf{x}_i$ , then the estimation would be easy.



- In particular, we can estimate each Gaussian independently.

# Mixture density estimation

- Let  $\delta(j|i)$  be an indicator function of whether example  $i$  is labeled  $j$ .  
Then for each  $j = 1, 2$

$$\hat{\pi}_j \leftarrow \frac{\hat{n}_j}{n}, \quad \text{where } \hat{n}_j = \sum_{i=1}^n \delta(j|i)$$

$$\hat{\mu}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^n \delta(j|i) \mathbf{x}_i$$

$$\hat{\Sigma}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^n \delta(j|i) (\mathbf{x}_i - \mu_j)(\mathbf{x}_i - \mu_j)^t$$



# Mixture density estimation

- We don't have such labels... but we can guess what the labels might be based on our current distribution.
- One possible choice: evaluate posterior probability that an observed  $\mathbf{x}$  was generated from first component

$$\begin{aligned} P(z = 1|\mathbf{x}, \boldsymbol{\theta}) &= \frac{P(z = 1) \cdot p(\mathbf{x}|z = 1)}{\sum_{j=1,2} P(z = j) \cdot p(\mathbf{x}|z = j)} \\ &= \frac{\pi_1 p(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)}{\sum_{j=1,2} \pi_j p(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \end{aligned}$$

↪ Information about the component responsible for generating  $\mathbf{x}$ .

- **Soft labels** or **posterior probabilities**

$$\hat{p}(j|i) \leftarrow P(z_i = j|\mathbf{x}_i, \boldsymbol{\theta}),$$

where  $\sum_{j=1,2} \hat{p}(j|i) = 1, \forall i = 1, \dots, n$ .

# The EM algorithm: iteration k

- **E-step:** softly assign examples to mixture components

$$\hat{p}(j|i) \leftarrow P(z_i = j | \mathbf{x}_i, \boldsymbol{\theta}^t), \forall j = 1, 2 \text{ and } i = 1, \dots, n.$$

Note: superscript is time index.

- **M-step:** estimate new mixture parameters  $\boldsymbol{\theta}^{t+1}$  based on the soft assignments (can be done separately for the two Gaussians)

$$\hat{\pi}_j \leftarrow \frac{\hat{n}_j}{n}, \quad \text{where } \hat{n}_j = \sum_{i=1}^n \hat{p}(j|i)$$

$$\hat{\boldsymbol{\mu}}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^n \hat{p}(j|i) \mathbf{x}_i$$

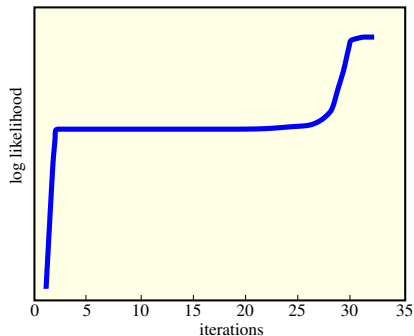
$$\hat{\boldsymbol{\Sigma}}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^n \hat{p}(j|i) (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^t$$

# The EM-algorithm: Convergence

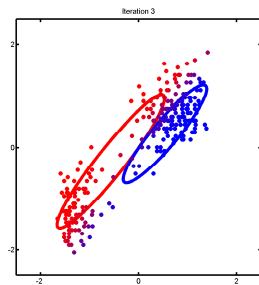
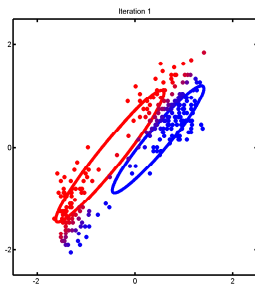
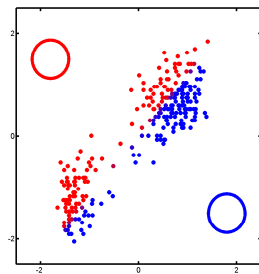
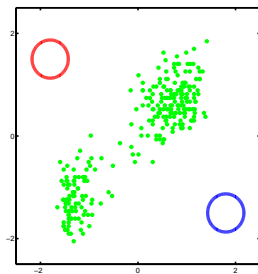
The EM-algorithm **monotonically increases the log-likelihood** of the training data (we will show this later). In other words,

$$l(\theta^0) < l(\theta^1) < l(\theta^2) < \dots \text{ until convergence}$$

$$l(\theta^t) = \sum_{i=1}^n \log p(\mathbf{x}_i | \theta^t).$$



# Mixture density estimation: example



# Mixture density estimation: example

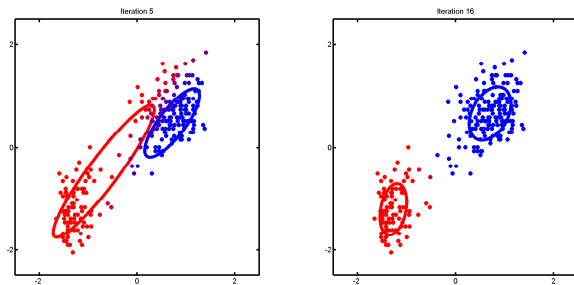
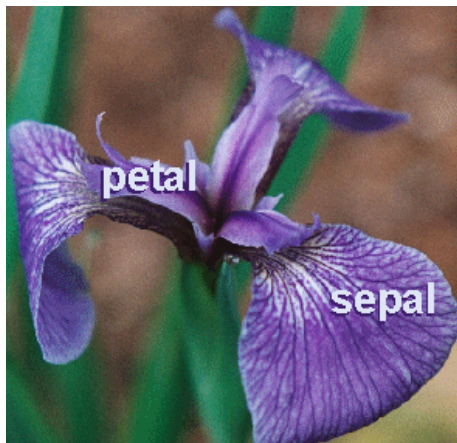


Fig. 11.11 in K. Murphy: Machine Learning. MIT Press 2012.

## EM example: Iris data

- The famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables **sepal length and width** and **petal length and width**, respectively, for 50 flowers from each of **3 species of iris**.
- The species are **Iris setosa, versicolor, and virginica**.





# Bayesian model selection for mixture models

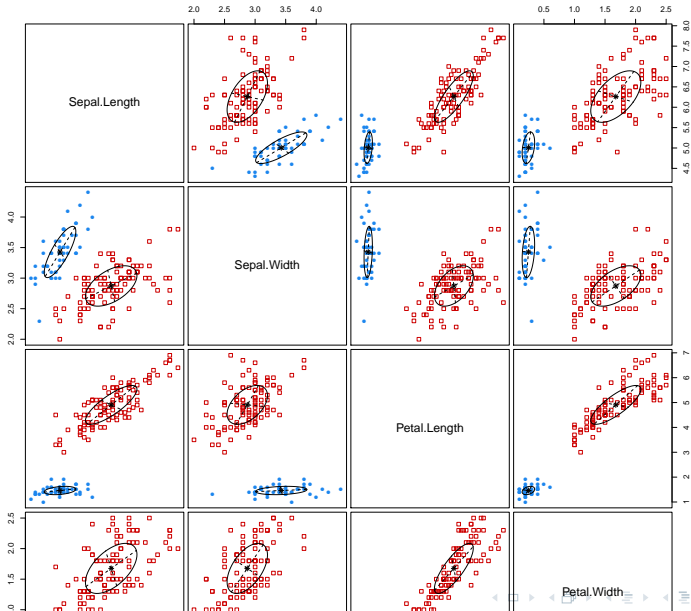
As a simple strategy for selecting the appropriate number of mixture components, we can find  $m$  that minimizes the overall description length (cf. BIC):

$$DL \approx -\log p(\text{data}|\hat{\theta}_m) + \frac{d_m}{n} \log(n)$$

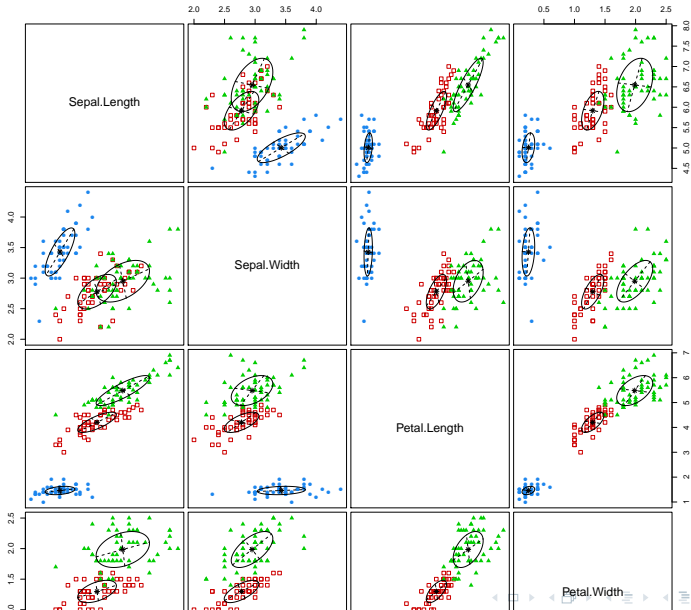
- $n$  is the number of training points,
- $\hat{\theta}_m$  are the maximum likelihood parameters for the  $m$ -component mixture, and
- $d_m$  is the number of parameters in the  $m$ -component mixture.



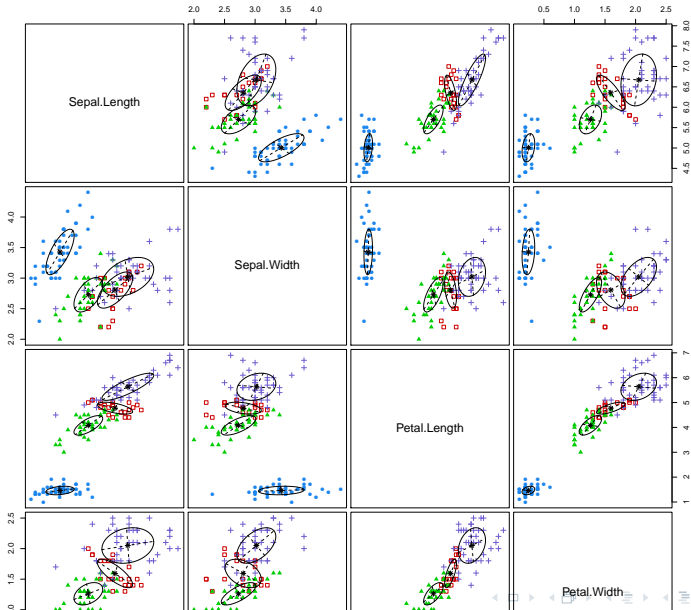
# Model selection example: Iris data, $m = 2$



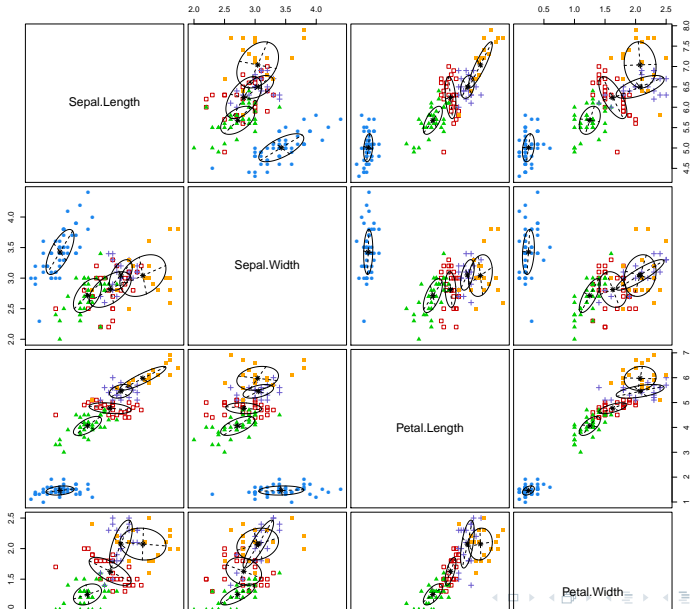
# Model selection example: Iris data, $m = 3$



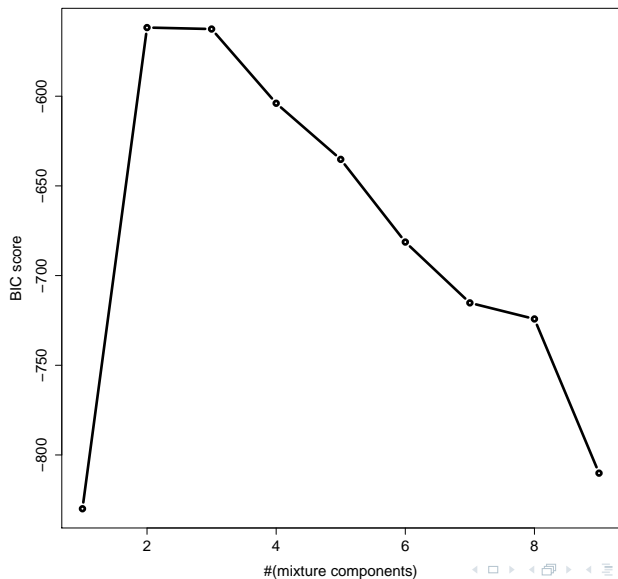
# Model selection example: Iris data, $m = 4$



# Model selection example: Iris data, $m = 5$



## Model selection example: Iris data, BIC



# The EM-algorithm: Convergence

**Step 0:** specify the initial setting of the parameters  $\theta = \theta^0$ .

**E-step:** complete the incomplete data (missing  $z$ ) with the posterior probabilities (“soft labels”)

$$P(z = j | \mathbf{x}_i, \theta^t), \quad j = 1, \dots, m, \quad i = 1, \dots, n.$$

**M-step:** find the new setting of the parameters  $\theta^{t+1}$  by maximizing the log-likelihood of the inferred (or “expected complete”) data

$$\theta^{t+1} = \arg \max_{\theta} \underbrace{\sum_{i=1}^n \sum_{j=1}^m P(z = j | \mathbf{x}_i, \theta^t) \log[p_j p(\mathbf{x}_i | \theta_j)]}_{\text{inferred (= expected complete) log-likelihood } Q(\theta, \theta^t)} .$$

The expected complete log-likelihood  $Q(\theta, \theta^t)$  is called

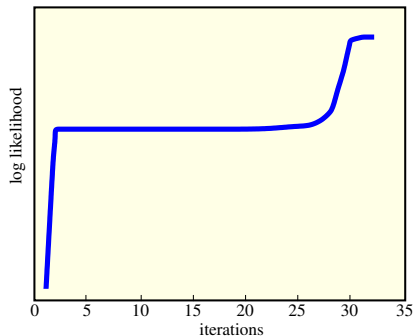
**auxiliary objective.**

# The EM-algorithm: Convergence

The EM-algorithm **monotonically increases the log-likelihood** of the training data. In other words,

$$l(\theta^0) < l(\theta^1) < l(\theta^2) < \dots \text{ until convergence}$$

$$l(\theta^t) = \sum_{i=1}^n \log p(\mathbf{x}_i | \theta^t).$$



# Jensen's inequality

- Convex function: secant line above graph of the function  
     $\rightsquigarrow$  Jensen's inequality for two points.
- Secant line consists of weighted means of the convex function.  
For  $a \in [0, 1]$ :

$$af(x_1) + (1 - a)f(x_2).$$

Graph: convex function of the weighted means:

$$f(ax_1 + (1 - a)x_2).$$

- Thus, Jensen's inequality is

$$f(ax_1 + (1 - a)x_2) \leq af(x_1) + (1 - a)f(x_2).$$

- Probability theory: if  $X$  is a RV and  $\varphi$  a convex function, then

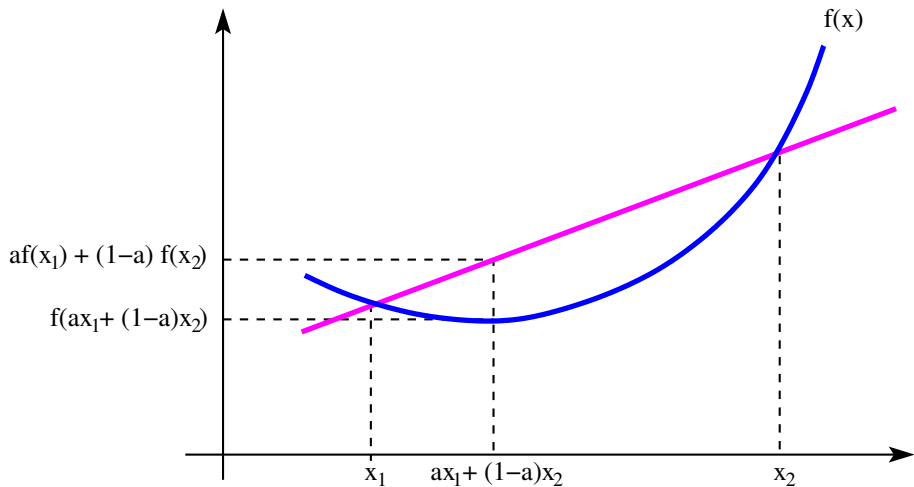
$$\varphi(E[X]) \leq E[\varphi(X)].$$

- $\varphi$  convex  $\rightsquigarrow \psi := -\varphi$  concave:

$$\psi(E[X]) \geq E[\psi(X)]. \quad \text{Example:} \quad \log(E[X]) \geq E[\log(X)].$$



# Jensen's inequality



# Non-negativity of KL divergence

$$\begin{aligned} -\mathbb{KL}(p(x) \| q(x)) &= \int p(x) \log \left( \frac{q(x)}{p(x)} \right) dx \\ \text{(Jensen's inequality)} &\leq \log \left( \int p(x) \frac{q(x)}{p(x)} dx \right) \\ &= \log \left( \int q(x) dx \right) \\ &= \log(1) = 0 \end{aligned}$$

This is also called **Gibbs' inequality**.

# The EM-algorithm: Theoretical basis

Consider distribution  $q(z_i)$  over latent assignment variables.

Log-likelihood:

$$\begin{aligned}l(\boldsymbol{\theta}) &= \sum_{i=1}^n \log p(\mathbf{x}_i | \boldsymbol{\theta}) \\&= \sum_{i=1}^n \log \sum_{z_i} p(\mathbf{x}_i, z_i | \boldsymbol{\theta}) \\&= \sum_{i=1}^n \log \sum_{z_i} q(z_i) \frac{p(\mathbf{x}_i, z_i | \boldsymbol{\theta})}{q(z_i)} \\&= \sum_{i=1}^n \log \mathbb{E}_{q_i} \frac{p(\mathbf{x}_i, z_i | \boldsymbol{\theta})}{q(z_i)} \\(\text{Jensen's inequality}) &\geq \sum_{i=1}^n \sum_{z_i} q(z_i) \log \frac{p(\mathbf{x}_i, z_i | \boldsymbol{\theta})}{q(z_i)} \\&=: Q(\boldsymbol{\theta}, \mathbf{q}).\end{aligned}$$

- Lower bound

$$l(\theta) \geq Q(\theta, q) := \sum_{i=1}^n \sum_{z_i} q(z_i) \log \frac{p(\mathbf{x}_i, z_i | \theta)}{q(z_i)}$$

valid for any positive distribution  $q$ . Which one should we choose?

- Intuition: pick the  $q$  that yields the tightest lower bound.

This will be the **E-step**.

- At time  $t$ , assume we have chosen  $q^t$  based on current parameters  $\theta^t$ .

In the next **M-step** we maximize the

**expected complete log-likelihood:**

$$\theta^{t+1} = \arg \max_{\theta} Q(\theta, \theta^t) = \arg \max_{\theta} \sum_{i=1}^n \mathbb{E}_{q_i^t} \log p(\mathbf{x}_i, z_i | \theta)$$

- Last equation follows from

$$Q(\theta, q) = \underbrace{\sum_{i=1}^n \mathbb{E}_{q_i} \log p(\mathbf{x}_i, z_i | \theta)}_{\text{Expected complete log-}l} + \sum_{i=1}^n \left[ \underbrace{- \sum_{z_i} q(z_i) \log q(z_i)}_{h(q_i), \text{ independent of } \theta} \right].$$

# The E-step

Re-write lower bound as

$$Q(\boldsymbol{\theta}, q) = \sum_i L(\boldsymbol{\theta}, q_i),$$

with

$$\begin{aligned} L(\boldsymbol{\theta}, q_i) &= \sum_{z_i} q(z_i) \log \frac{p(\mathbf{x}_i, z_i | \boldsymbol{\theta})}{q(z_i)} \\ &= \sum_{z_i} q(z_i) \log \frac{p(z_i | \mathbf{x}_i, \boldsymbol{\theta}) p(\mathbf{x}_i | \boldsymbol{\theta})}{q(z_i)} \\ &= \sum_{z_i} q(z_i) \log \frac{p(z_i | \mathbf{x}_i, \boldsymbol{\theta})}{q(z_i)} + \sum_{z_i} q(z_i) \log p(\mathbf{x}_i | \boldsymbol{\theta}) \\ &= \underbrace{-\text{KL}(q(z_i) \| p(z_i | \mathbf{x}_i, \boldsymbol{\theta}))}_{\text{always } \geq 0, \text{ and } = 0, \text{ if } q=p} + \underbrace{\log p(\mathbf{x}_i | \boldsymbol{\theta})}_{\text{independent of } q_i}. \end{aligned}$$

# The E step

For  $q_i^t(z_i) = p(z_i|\mathbf{x}_i, \boldsymbol{\theta}^t)$ , the KL divergence is zero, and  $L(\boldsymbol{\theta}^t, q_i)$  is maximized over all possible distributions  $q_i$ :

$$q_i^t(z_i) = p(z_i|\mathbf{x}_i, \boldsymbol{\theta}^t) = \arg \max_{q_i} L(\boldsymbol{\theta}^t, q_i) \quad (\rightsquigarrow \text{E-step})$$

$$L(\boldsymbol{\theta}^t, q_i^t) = \log p(\mathbf{x}_i|\boldsymbol{\theta}^t)$$

$$Q(\boldsymbol{\theta}^t, \boldsymbol{\theta}^t) = \sum_i \log p(\mathbf{x}_i|\boldsymbol{\theta}^t) = l(\boldsymbol{\theta}^t)$$

$\rightsquigarrow$  lower bound “touches” the log-likelihood

$\rightsquigarrow$  after the E-step, the auxiliary objective equals the log-likelihood

$\rightsquigarrow$  lower bound is tight after the E-step.

# The E step

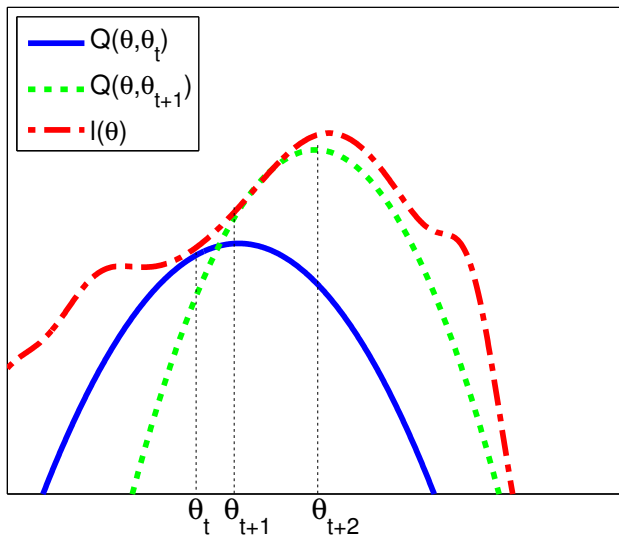


Fig 11.16 in K. Murphy: Machine Learning. MIT Press 2012.

# EM-algorithm: max-max and monotonicity

We can now rewrite the EM-algorithm in terms of two maximization steps involving the auxiliary objective:

**E-step:**  $\mathbf{q}^t = \arg \max_{\mathbf{q}} Q(\boldsymbol{\theta}^t, \mathbf{q})$

**M-step:**  $\boldsymbol{\theta}^{t+1} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)$ .

The monotonic increase of the log-likelihood now follows from

$$\underbrace{l(\boldsymbol{\theta}^{t+1})}_{Q(\boldsymbol{\theta}^{t+1}, \bullet) \text{ is lower bound on } l(\boldsymbol{\theta}^{t+1})} \geq \underbrace{Q(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t)}_{\geq Q(\boldsymbol{\theta}^t, \boldsymbol{\theta}^t)} = l(\boldsymbol{\theta}^t).$$

Second inequality:  $Q(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t) = \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) \geq Q(\boldsymbol{\theta}^t, \boldsymbol{\theta}^t)$ .



# Conditional mixtures

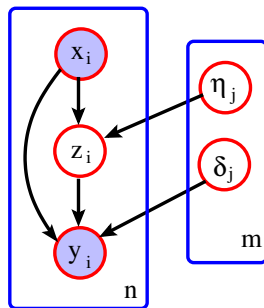
- Some regression or classification problems can be decomposed into easier sub-problems.
- Examples:
  - ▶ style in handwritten character recognition
  - ▶ dialect/accents in speech recognition, etc.
- Each sub-problem could be solved by a specific “expert”.
- The selection of which expert to rely on now depends on the position  $x$  in the input space. **Mixtures of experts models.**

# Experts (regression)

- Suppose we have several regression “experts” generating conditional Gaussian outputs

$$p(y|\mathbf{x}, z = j, \boldsymbol{\theta}) = \mathcal{N}(y|\beta_j^t \mathbf{x}, \sigma_j^2)$$

- $\boldsymbol{\delta}_j = \{\beta_j, \sigma_j^2\}$ : Parameters of  $j$ -th expert.
- Need to find a way of allocating tasks to these experts.
- Parameter vector  $\boldsymbol{\theta}$  contains the means and variances of the  $m$  experts and the additional parameters  $\boldsymbol{\eta}$  of this allocation mechanism:  
 $\boldsymbol{\theta} = \{\boldsymbol{\delta}_j, \boldsymbol{\eta}_j\}_{j=1}^m$ .



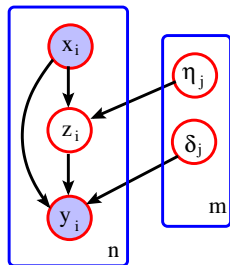
# Joint distribution

From the DAG we conclude:

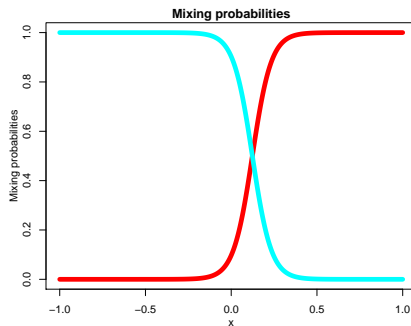
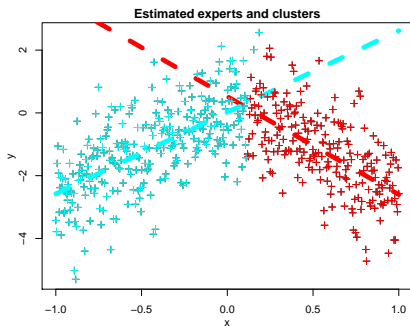
$$\begin{aligned} p(y, z = j | \mathbf{x}) &= p(y | \boldsymbol{\delta}, z = j, \mathbf{x}) P(z = j | \boldsymbol{\eta}, \mathbf{x}) \\ &= p(y | \boldsymbol{\delta}_j, \mathbf{x}) P(z = j | \boldsymbol{\eta}, \mathbf{x}) \\ &= \mathcal{N}(y | \boldsymbol{\beta}_j^t \mathbf{x}, \sigma_j^2) P(z = j | \boldsymbol{\eta}, \mathbf{x}) \end{aligned}$$

Thus, the overall prediction is

$$\begin{aligned} p(y | \mathbf{x}, \boldsymbol{\theta}) &= \sum_j p(y, z = j | \mathbf{x}, \boldsymbol{\eta}, \boldsymbol{\delta}) \\ &= \sum_j P(z = j | \mathbf{x}, \boldsymbol{\eta}) p(y | \mathbf{x}, \boldsymbol{\delta}_j) \\ &= \sum_j P(z = j | \mathbf{x}, \boldsymbol{\eta}) p(y | \mathbf{x}, \boldsymbol{\beta}_j, \sigma_j^2). \end{aligned}$$



# Mixtures of experts



Here we need to switch from one linear regression model to another:  
 $p(y|\mathbf{x}, z = j, \boldsymbol{\theta}) = \mathcal{N}(y|\boldsymbol{\beta}_j^t \mathbf{x}, \sigma_j^2)$ . The switch can be probabilistic  
 $\rightsquigarrow$  probabilistic gating function  $P(z|\mathbf{x}, \boldsymbol{\eta})$  (right).

# Gating network

- A gating network specifies a distribution over  $m$  experts, conditionally on the input  $\mathbf{x}$ .
- Example: when there are just two experts the gating network can be a logistic regression model

$$P(z = 1|\mathbf{x}, \boldsymbol{\eta}) = \sigma(\boldsymbol{\eta}^t \mathbf{x}),$$

where  $\sigma(z) = (1 + e^{-z})^{-1}$  is the logistic function.

- For  $m > 2$ , the gating network can be a softmax model

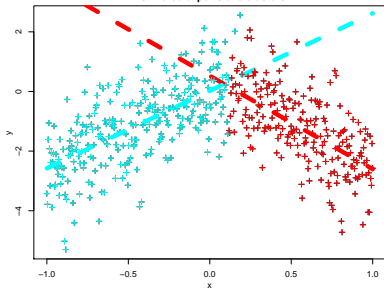
$$P(z = j|\mathbf{x}, \boldsymbol{\eta}) = \frac{\exp(\boldsymbol{\eta}_j^t \mathbf{x})}{\sum_{j'=1}^m \exp(\boldsymbol{\eta}_{j'}^t \mathbf{x})},$$

where  $\boldsymbol{\eta} = \{\boldsymbol{\eta}_1, \dots, \boldsymbol{\eta}_m\}$  are the parameters of the gating network.

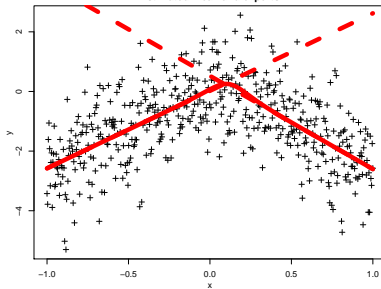
- Overall prediction

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \sum_j p(y, z = j|\mathbf{x}, \boldsymbol{\eta}, \boldsymbol{\delta}) = \sum_j P(z = j|\mathbf{x}, \boldsymbol{\eta})p(y|\mathbf{x}, \boldsymbol{\delta}_j).$$

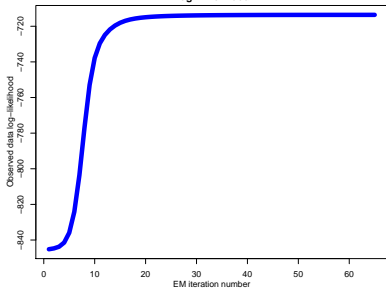
Estimated experts and clusters



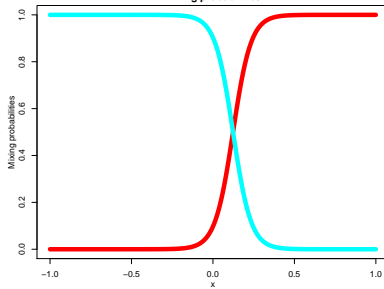
Estimated mean and experts



Log-Likelihood



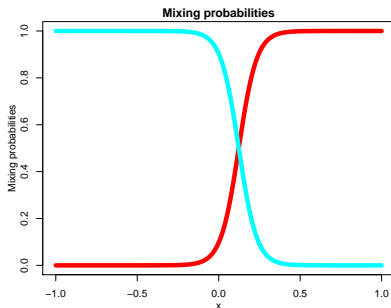
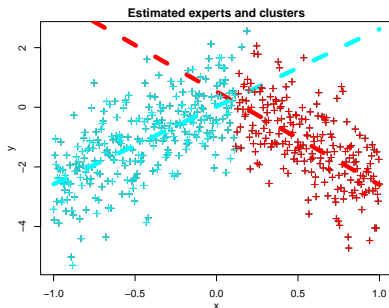
Mixing probabilities



# A mixture of experts model: estimation

“Soft labels”: Conditional probability that  $(\mathbf{x}_i, y_i)$  came from expert  $j$ :

$$\begin{aligned}\hat{P}(j|i) &= P(z = j | \mathbf{x}_i, y_i, \boldsymbol{\theta}) \\ &= \frac{P(z = j | \mathbf{x}_i, \boldsymbol{\eta}^t) p(y_i | \mathbf{x}_i, (\boldsymbol{\beta}_j, \sigma_j^2))}{\sum_{j'=1}^m P(z = j' | \mathbf{x}_i, \boldsymbol{\eta}^t) p(y_i | \mathbf{x}_i, (\boldsymbol{\beta}_{j'}, \sigma_{j'}^2))}\end{aligned}$$



# EM for mixtures of experts

**E-step:** compute soft labels  $\hat{P}(j|i)$

**M-step:** separately re-estimate the experts and the gating network based on these soft assignments:

- 1 For each expert  $j$ : find  $(\hat{\beta}_j, \hat{\sigma}_j^2)$  that maximize

$$\sum_{i=1}^n \hat{P}(j|i) \log p(y_i | \mathbf{x}_i, (\beta_j, \sigma_j^2))$$

↪ linear regression with weighted training set.

- 2 For the gating network: find  $\hat{\eta}$  that maximize

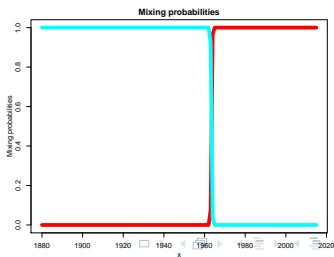
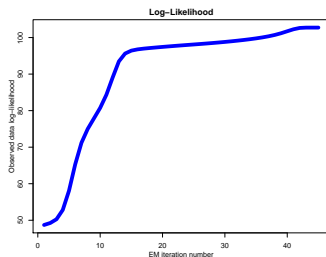
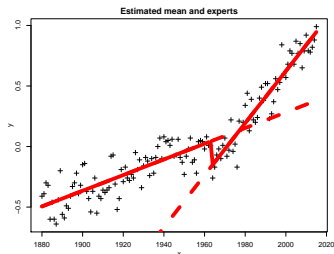
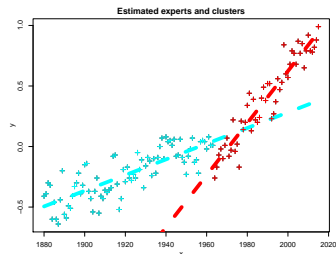
$$\sum_{i=1}^n \sum_{j=1}^m \hat{P}(j|i) \log P(j | \mathbf{x}_i, \eta_j)$$

↪ logistic regression with weighted training set.

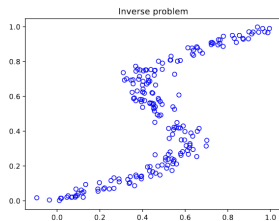


# Real World Example

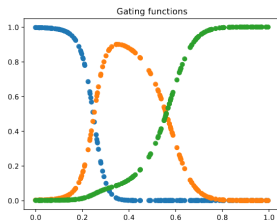
Global annual temperature anomalies (degrees C) computed using data from land meteorological stations, 1880-2015. Anomalies are relative to the 1951-1980 base period means.



# Conditional Mixtures for Inverse Problems



(a)



(b)

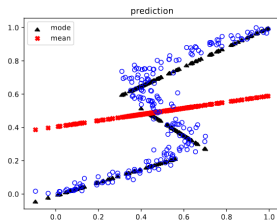
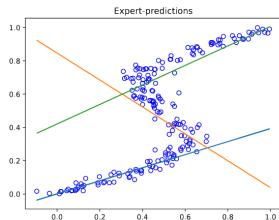


Figure 13.23 in K.Murphy: Probabilistic Machine Learning. MIT Press, 2023. a) data one-to-many function. (b) Responsibilities of each expert. (c) Prediction of each expert. (d) Overall prediction. Mean is red, mode is black.