

# IT3105 Artificial Intelligence Programming

## Project 2 – Speech Recognition. Lecture 3.

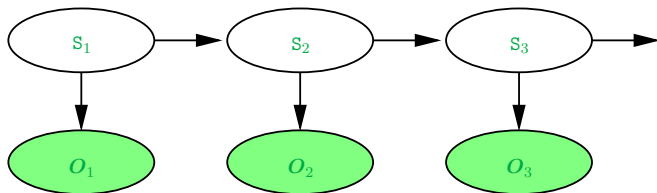
Norwegian University of Science and Technology

Helge Langseth  
IT-VEST 310  
helgel@idi.ntnu.no



- 1 Wrap-up from last time
- 2 Learning BN parameters from complete data
  - The Maximum Likelihood Principle
  - The general case
- 3 Learning parameters from incomplete data
  - Motivation
  - The EM algorithm
  - Example of EM at work: Mixture model
- 4 EM for HMMs – The Baum-Welch algorithm
  - General setup
  - The *really* dirty details
- 5 Conclusions

## An example of a Hidden Markov Model



## Parameters:

- $S_t \in \{1, 2\}$ , with  $\pi = \left[ \frac{1}{2} \ \frac{1}{2} \right]^T$ .
- Dynamic model:  $\mathbf{A} = \begin{bmatrix} .7 & .3 \\ .3 & .7 \end{bmatrix}$ .
- $O_t$  is univariate, with observation model  
 $b_j(o_t) = f(o_t | S_t = s_t) = \mathcal{N}(\mu = s_t, \sigma^2 = 1)$ .

## Focus today:

Learn model parameters that fit the speech signals.

# Rabiner's Forward-algorithm



- For inference we were looking for  $P(\mathbf{o}_{1:T} \mid \text{word})$ , i.e., the total probability for the observation sequence given that the HMM represents the correct.
- Rabiner calculates  $\alpha_t(j) = P(S_t = j, \mathbf{o}_{1:t} \mid \text{word})$ , which is interesting also because  $P(\mathbf{o}_{1:T} \mid \text{word}) = \sum_{j=1}^N \alpha_T(j)$ .
- Define  $\mathbf{B}_{t+1}$  to be a matrix of size  $N \times N$ .  $\mathbf{B}_{t+1} = 0$  except on the diagonal, where element  $(j, j)$  is  $P(\mathbf{o}_{t+1} \mid S_{t+1} = j)$ .

## Scaled version of Rabiner's Forward-algorithm:

- **Initialization:**

- $\mathbf{f}'_1 = \mathbf{B}_1 \boldsymbol{\pi}$ ;  $\ell_1 = \sum_{j=1}^N \mathbf{f}'_1(j)$ ;  $\mathbf{f}_1 \leftarrow \mathbf{f}'_1 / \ell_1$ .

- **Induction:**

Do for  $t = 1, \dots, T - 1$ :

- $\mathbf{f}'_{t+1} = \mathbf{B}_{t+1} \mathbf{A}^T \mathbf{f}_t$ ;  $\ell_{t+1} = \sum_{j=1}^N \mathbf{f}'_{t+1}(j)$ ;  
 $\mathbf{f}_{t+1} \leftarrow \mathbf{f}'_{t+1} / \ell_{t+1}$ .

# Learning probabilities from a database

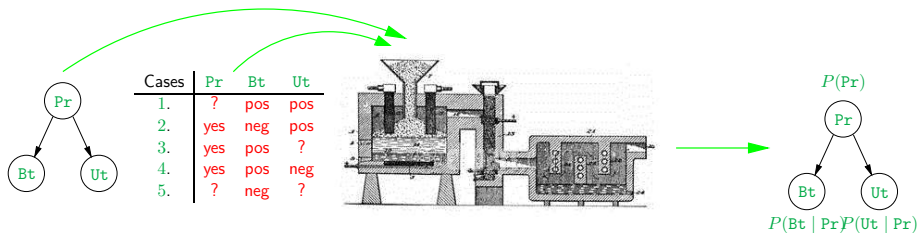


## We have:

- A Bayesian network structure.
- A database of cases over (some of) the variables.

## We want:




- A Bayesian network model (with probabilities) representing the database.



# Complete data: Maximum likelihood estimation



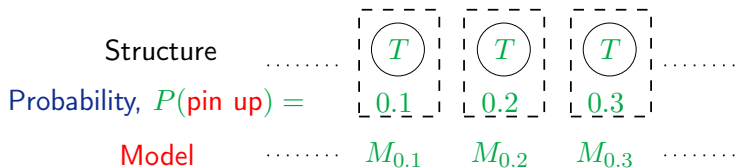
We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we look for the model that best fits the data:

Structure	.....				.....
Probability, $P(\text{pin up}) =$		0.1	0.2	0.3	
Model	.....	$M_{0.1}$	$M_{0.2}$	$M_{0.3}$	.....

# Complete data: Maximum likelihood estimation



We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we look for the model that best fits the data:



We can measure how well a model fits the data using:

$$\begin{aligned}
 P(\mathcal{D}|M_\theta) &= P(\text{pin up}, \text{pin down}, \text{pin down}, \dots, \text{pin up}|M_\theta) \\
 &= P(\text{pin up}|M_\theta) \cdot P(\text{pin down}|M_\theta) \cdot \dots \cdot P(\text{pin up}|M_\theta)
 \end{aligned}$$

## Complete data: Maximum likelihood estimation



We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we look for the model that best fits the data:

Structure	.....	$T$	$T$	$T$	.....
Probability, $P(\text{pin up}) =$		0.1	0.2	0.3	
Model	.....	$M_{0.1}$	$M_{0.2}$	$M_{0.3}$	.....

We select the parameter  $\hat{\theta}$  that maximizes:

$$\begin{aligned} \hat{\theta} &= \arg \max_{\theta} P(\mathcal{D}|M_{\theta}) = \arg \max_{\theta} \prod_{i=1}^{100} P(d_i|M_{\theta}) \\ &= \arg \max_{\theta} \mu \cdot \theta^{80} (1 - \theta)^{20}. \end{aligned}$$

# Complete data: Maximum likelihood estimation



We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we look for the model that best fits the data:

Structure	.....	$T$	$T$	$T$	.....
Probability, $P(\text{pin up}) =$		0.1	0.2	0.3	
Model	.....	$M_{0.1}$	$M_{0.2}$	$M_{0.3}$	.....

By setting:

$$\frac{d}{d\theta} \mu \cdot \theta^{80} (1 - \theta)^{20} = 0$$

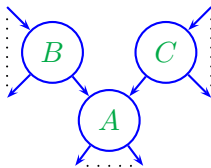
we get the maximum likelihood estimate:

$$\hat{\theta} = 0.8 \equiv \frac{\# \text{pin up}}{\# \text{tosses}}.$$

## Complete data: General maximum likelihood estimation



In general, you get a maximum likelihood estimate for **discrete variables** as the fraction of counts over the total number of counts.



We want  $P(A = a \mid B = b, C = c)$ !

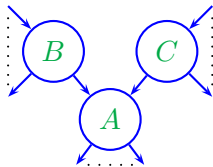
To find the maximum likelihood estimate  $\hat{P}(A = a \mid B = b, C = c)$  we simply calculate:

$$\hat{P}(A = a \mid B = b, C = c) =$$

## Complete data: General maximum likelihood estimation



In general, you get a maximum likelihood estimate for **discrete variables** as the fraction of counts over the total number of counts.



We want  $P(A = a \mid B = b, C = c)$ !

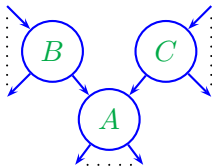
To find the maximum likelihood estimate  $\hat{P}(A = a \mid B = b, C = c)$  we simply calculate:

$$\hat{P}(A = a \mid B = b, C = c) = \frac{\hat{P}(a, b, c)}{\hat{P}(b, c)}$$

## Complete data: General maximum likelihood estimation



In general, you get a maximum likelihood estimate for **discrete variables** as the fraction of counts over the total number of counts.



We want  $P(A = a \mid B = b, C = c)$ !

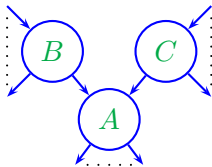
To find the maximum likelihood estimate  $\hat{P}(A = a \mid B = b, C = c)$  we simply calculate:

$$\hat{P}(A = a \mid B = b, C = c) = \frac{\hat{P}(a, b, c)}{\hat{P}(b, c)} = \frac{\left[ \frac{N(A=a, B=b, C=c)}{N} \right]}{\left[ \frac{N(B=b, C=c)}{N} \right]}$$

## Complete data: General maximum likelihood estimation



In general, you get a maximum likelihood estimate for **discrete variables** as the fraction of counts over the total number of counts.



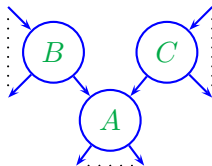
We want  $P(A = a \mid B = b, C = c)$ !

To find the maximum likelihood estimate  $\hat{P}(A = a \mid B = b, C = c)$  we simply calculate:

$$\begin{aligned} \hat{P}(A = a \mid B = b, C = c) &= \frac{\hat{P}(a, b, c)}{\hat{P}(b, c)} = \frac{\left[ \frac{N(A=a, B=b, C=c)}{N} \right]}{\left[ \frac{N(B=b, C=c)}{N} \right]} \\ &= \frac{N(A = a, B = b, C = c)}{N(B = b, C = c)}. \end{aligned}$$

So we have a simple counting problem!

## Complete data: Continuous variables



Now assume that  $A$  is **Gaussian**  
 $B$  &  $C$  are discrete.

- As before we can estimate  $f(a | b, c)$  “locally” (i.e., once per state of  $B$  and  $C$ , and independently of parent distributions).
- Now we need the parameters of the Gaussian distribution,  
 $\mu_{b,c} = \mathbb{E}[A | B = b, C = c]$  and  
 $\Sigma_{b,c} = \text{Cov}(A | B = b, C = c)$ .

$$\hat{\mu}_{b,c} = \sum_{i:\{b_i=b, c_i=c\}} a_i / N(B = b, C = c)$$

$$\hat{\Sigma}_{b,c} = \sum_{i:\{b_i=b, c_i=c\}} (a_i - \hat{\mu}_{b,c})^2 / N(B = b, C = c)$$

# Incomplete data



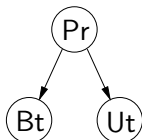
How do we handle cases with missing values:

- Faulty sensor readings.
- Values have been intentionally removed.
- **Some variables may be unobservable.**

## **Goal for the learning:**

Make sure that we get the most out of the data that we have!

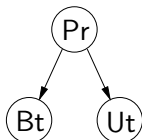
# The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

Estimate the required probability distributions for the network

# The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

If the database was complete we would estimate the required probabilities,  $P(\text{Pr})$ ,  $P(\text{Ut} \mid \text{Pr})$  and  $P(\text{Bt} \mid \text{Pr})$  as:

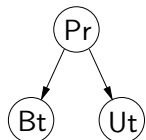
$$P(\text{Pr} = \text{yes}) = \frac{N(\text{Pr} = \text{yes})}{N}$$

$$P(\text{Ut} = \text{yes} \mid \text{Pr} = \text{yes}) = \frac{N(\text{Ut} = \text{yes}, \text{Pr} = \text{yes})}{N(\text{Pr} = \text{yes})}$$

$$P(\text{Bt} = \text{yes} \mid \text{Pr} = \text{no}) = \frac{N(\text{Bt} = \text{yes}, \text{Pr} = \text{no})}{N(\text{Pr} = \text{no})}$$

So estimating probabilities is still basically a counting problem!

# The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

Estimate  $P(\text{Pr})$  from the database above:

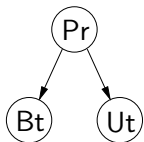
Case 2, 3 and 4 contributes with a value 1 to  $N(\text{Pr} = \text{yes})$ , but what is the contribution from case 1 and 5?

- Case 1 contributes with  $P(\text{Pr} = \text{yes} | \text{Bt} = \text{pos}, \text{Ut} = \text{pos})$ .
- Case 5 contributes with  $P(\text{Pr} = \text{yes} | \text{Bt} = \text{neg})$ .

To find these probabilities we assume some initial distributions,  $P_0(\cdot)$ , have been assigned to the network.

We are basically calculating the expectation for  $N(\text{Pr} = \text{yes})$ , denoted  $\mathbb{E}[N(\text{Pr} = \text{yes})]$

## The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

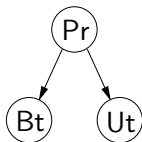
Using  $P_0(\text{Pr}) = (0.5, 0.5)$ ,  $P_0(\text{Bt} \mid \text{Pr} = \text{yes}) = (0.5, 0.5)$  etc., as starting distributions we get:

$$\begin{aligned} \mathbb{E}[N(\text{Pr} = \text{y})] &= P_0(\text{Pr} = \text{y} \mid \text{Bt} = \text{Ut} = \text{pos}) + 1 + 1 + 1 \\ &\quad + P_0(\text{Pr} = \text{y} \mid \text{Bt} = \text{neg}) \\ &= 0.5 + 1 + 1 + 1 + 0.5 = 4 \end{aligned}$$

$$\begin{aligned} \mathbb{E}[N(\text{Pr} = \text{no})] &= P_0(\text{Pr} = \text{no} \mid \text{Bt} = \text{Ut} = \text{pos}) + 0 + 0 + 0 \\ &\quad + P_0(\text{Pr} = \text{no} \mid \text{Bt} = \text{neg}) \\ &= 0.5 + 0 + 0 + 0 + 0.5 = 1 \end{aligned}$$

So we e.g. get  $\hat{P}_1(\text{Pr} = \text{yes}) = \mathbb{E}[N(\text{Pr} = \text{yes})]/N = 0.8$ .

## The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

To estimate  $\hat{P}_1(\text{Ut} = \text{pos} \mid \text{Pr} = \text{yes}) = \mathbb{E}[N(\text{Ut} = \text{pos}, \text{Pr} = \text{yes})] / \mathbb{E}[N(\text{Pr} = \text{yes})]$  we e.g. need:

$$\begin{aligned} \mathbb{E}[N(\text{Ut} = \text{p}, \text{Pr} = \text{y})] &= P_0(\text{Ut} = \text{p}, \text{Pr} = \text{y} \mid \text{Bt} = \text{Ut} = \text{p}) + 1 \\ &+ P_0(\text{Ut} = \text{p}, \text{Pr} = \text{y} \mid \text{Bt} = \text{p}, \text{Pr} = \text{y}) + 0 + P_0(\text{Ut} = \text{p}, \text{Pr} = \text{y} \mid \text{Bt} = \text{n}) \\ &= 0.5 + 1 + 0.5 + 0 + 0.25 = 2.25 \end{aligned}$$

$$\begin{aligned} \mathbb{E}[N(\text{Pr} = \text{yes})] &= P_0(\text{Pr} = \text{yes} \mid \text{Bt} = \text{Ut} = \text{pos}) + 1 + 1 + 1 \\ &+ P_0(\text{Pr} = \text{yes} \mid \text{Bt} = \text{neg}) = 0.5 + 1 + 1 + 1 + 0.5 = 4 \end{aligned}$$

$$\hat{P}_1(\text{Ut} = \text{pos} \mid \text{Pr} = \text{yes}) = \frac{\mathbb{E}[N(\text{Ut} = \text{p}, \text{Pr} = \text{y})]}{\mathbb{E}[N(\text{Pr} = \text{yes})]} = \frac{2.25}{4} = 0.5625$$

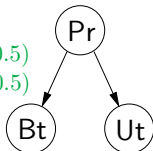
# The EM algorithm



$$P_0(Pr) = (0.5, 0.5)$$

$$P_0(Ut = \text{pos} \mid Pr) = (0.5, 0.5)$$

$$P_0(Bt = \text{pos} \mid Pr) = (0.5, 0.5)$$



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

## The EM algorithm

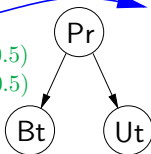


E-step 1

$$P_0(Pr) = (0.5, 0.5)$$

$$P_0(Ut = \text{pos} \mid Pr) = (0.5, 0.5)$$

$$P_0(Bt = \text{pos} \mid Pr) = (0.5, 0.5)$$



$$\mathbb{E}_0[N(Pr)] = (4, 1)$$

$$\mathbb{E}_0[N(Ut = \text{pos}, Pr)] = (2.25, 0.5 + 0 + 0 + 0 + 0.25)$$

$$\mathbb{E}_0[N(Bt = \text{pos}, Pr)] = (0.5 + 0 + 1 + 1 + 0 = 2.5, 0.5 + 0 + 0 + 0 + 0 = 0.5)$$

Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

## The EM algorithm

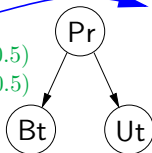


E-step 1

$$P_0(Pr) = (0.5, 0.5)$$

$$P_0(Ut = \text{pos} | Pr) = (0.5, 0.5)$$

$$P_0(Bt = \text{pos} | Pr) = (0.5, 0.5)$$



$$\mathbb{E}_0[N(Pr)] = (4, 1)$$

$$\mathbb{E}_0[N(Ut = \text{pos}, Pr)] = (2.25, 0.5 + 0 + 0 + 0 + 0.25)$$

$$\mathbb{E}_0[N(Bt = \text{pos}, Pr)] = (0.5 + 0 + 1 + 1 + 0 = 2.5, 0.5 + 0 + 0 + 0 + 0 = 0.5)$$

M-step 2

$$P_1(Pr) = \left(\frac{4}{5}, \frac{1}{5}\right)$$

$$P_1(Ut = \text{pos} | Pr) = \left(\frac{2.25}{4}, \frac{0.75}{1}\right)$$

$$P_1(Bt = \text{pos} | Pr) = \left(\frac{2.5}{4}, \frac{0.5}{1}\right)$$

Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

## The EM algorithm

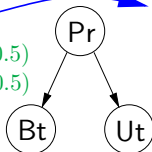


E-step 1

$$P_0(Pr) = (0.5, 0.5)$$

$$P_0(Ut = \text{pos} | Pr) = (0.5, 0.5)$$

$$P_0(Bt = \text{pos} | Pr) = (0.5, 0.5)$$



$$\mathbb{E}_0[N(Pr)] = (4, 1)$$

$$\mathbb{E}_0[N(Ut = \text{pos}, Pr)] = (2.25, 0.5 + 0 + 0 + 0 + 0.25)$$

$$\mathbb{E}_0[N(Bt = \text{pos}, Pr)] = (0.5 + 0 + 1 + 1 + 0 = 2.5, 0.5 + 0 + 0 + 0 + 0 = 0.5)$$

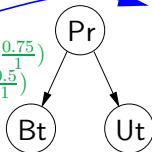
M-step 2

E-step 3

$$P_1(Pr) = \left(\frac{4}{5}, \frac{1}{5}\right)$$

$$P_1(Ut = \text{pos} | Pr) = \left(\frac{2.25}{4}, \frac{0.75}{1}\right)$$

$$P_1(Bt = \text{pos} | Pr) = \left(\frac{2.5}{4}, \frac{0.5}{1}\right)$$



$$\mathbb{E}_1[N(Pr)] = \left( \quad , \quad \right)$$

$$\mathbb{E}_1[N(Ut = \text{pos}, Pr)] = \left( \quad , \quad \right)$$

$$\mathbb{E}_1[N(Bt = \text{pos}, Pr)] = \left( \quad , \quad \right)$$

Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

## The EM algorithm

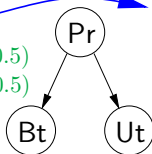


E-step 1

$$P_0(Pr) = (0.5, 0.5)$$

$$P_0(Ut = \text{pos} | Pr) = (0.5, 0.5)$$

$$P_0(Bt = \text{pos} | Pr) = (0.5, 0.5)$$



$$\mathbb{E}_0[N(Pr)] = (4, 1)$$

$$\mathbb{E}_0[N(Ut = \text{pos}, Pr)] = (2.25, 0.5 + 0 + 0 + 0 + 0.25)$$

$$\mathbb{E}_0[N(Bt = \text{pos}, Pr)] = (0.5 + 0 + 1 + 1 + 0 = 2.5, 0.5 + 0 + 0 + 0 + 0 = 0.5)$$

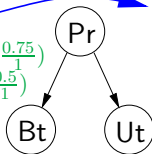
M-step 2

E-step 3

$$P_1(Pr) = \left(\frac{4}{5}, \frac{1}{5}\right)$$

$$P_1(Ut = \text{pos} | Pr) = \left(\frac{2.25}{4}, \frac{0.75}{1}\right)$$

$$P_1(Bt = \text{pos} | Pr) = \left(\frac{2.5}{4}, \frac{0.5}{1}\right)$$



$$\mathbb{E}_1[N(Pr)] = \left( \quad, \quad \right)$$

$$\mathbb{E}_1[N(Ut = \text{pos}, Pr)] = \left( \quad, \quad \right)$$

$$\mathbb{E}_1[N(Bt = \text{pos}, Pr)] = \left( \quad, \quad \right)$$

M-step 4

$$P_2(Pr) = (\cdot, \cdot)$$

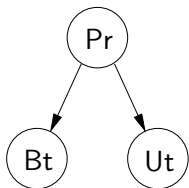
$$P_2(Ut = \text{pos} | Pr) = (\cdot, \cdot)$$

$$P_2(Bt = \text{pos} | Pr) = (\cdot, \cdot)$$

Until convergence

Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

## The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

- Let  $\theta^0 = \{\theta_{ijk}\}$  be start estimates ( $P(X_i = j \mid \text{pa}(X_i) = k) = \theta_{ijk}$ ).
- Repeat until convergence:
  - E-step:** For each  $X_i$  calculate the table of expected counts:

$$\mathbb{E}_{\theta^t}[N(X_i, \text{pa}(X_i) \mid \mathcal{D})] = \sum_{\mathbf{d} \in \mathcal{D}} P(X_i, \text{pa}(X_i) \mid \mathbf{d}, \theta^t).$$

- M-step:** Use the expected counts as if they were “real”:

$$\hat{\theta}_{ijk} = \frac{\mathbb{E}_{\theta^t}[N(X_i = k, \text{pa}(X_i) = j \mid \mathcal{D})]}{\sum_k \mathbb{E}_{\theta^t}[N(X_i = k, \text{pa}(X_i) = j \mid \mathcal{D})]}.$$

# EM for Estimating $k$ Gaussians

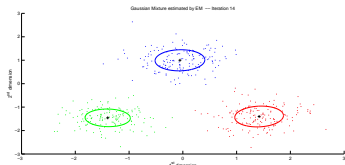
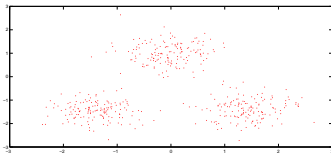


## Given:

- **Data:** Instances in  $\mathbb{R}^2$  generated by a mixture of  $k$  Gaussian distributions with unknown means
- **Model parameters:** Unknown means  $\langle \mu_1, \dots, \mu_k \rangle$  of the  $k$  Gaussians
- **Complicating fact:** Don't know which instance  $x_i$  was generated by which Gaussian

## Determine:

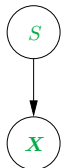
- Maximum likelihood estimates of  $\langle \mu_1, \dots, \mu_k \rangle$



EM for Estimating  $k$  Gaussians (2)

Think of each instance as  $\mathbf{y}_i = \langle \mathbf{x}_i, s_{i1}, \dots, s_{ik} \rangle$ , where

- $s_i = j$  if  $\mathbf{x}_i$  generated by  $j$ th Gaussian
- $\mathbf{x}_i$  observable,  $s_i$  unobservable



Cases	$\mathbf{x}$	$s$
1.	(1.2, 2.0)	(?)
2.	(2.2, 0.1)	(?)
3.	(2.3, 0.1)	(?)
4.	(-0.2, -1.2)	(?)
5.	(6.7, -3.0)	(?)

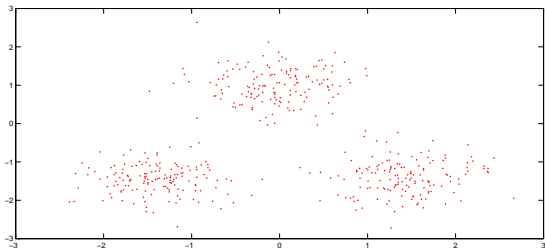
**Idea:**

Use the EM algorithm to learn the Maximum Likelihood parameters. As a side-effect we “fill in” values for  $s_i$ , which are the labels for each observation’s mixture belonging.

# EM for Estimating $k$ Gaussians (3)



Initialize:  $\mu_1, \dots, \mu_k$  picked on random

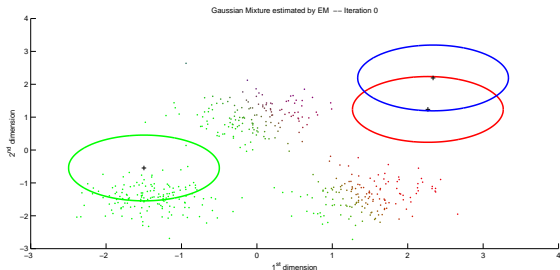


# EM for Estimating $k$ Gaussians (3)



Initialize:  $\mu_1, \dots, \mu_k$  picked at random

Calculate the *responsibilities* each Gaussian takes for each datapoint



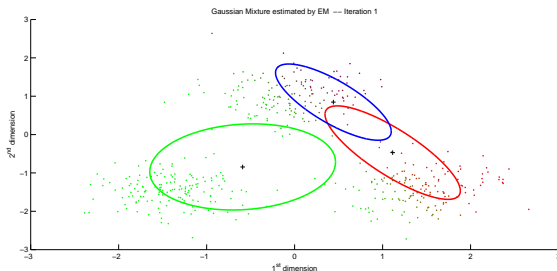
# EM for Estimating $k$ Gaussians (3)



Initialize:  $\mu_1, \dots, \mu_k$  picked at random

Calculate the *responsibilities* each Gaussian takes for each datapoint

Update  $\mu_j$  based on those datapoints Gaussian  $j$  takes responsibility for



# EM for Estimating $k$ Gaussians (3)



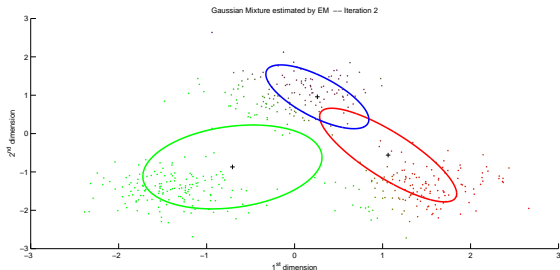
Initialize:  $\mu_1, \dots, \mu_k$  picked at random

Calculate the *responsibilities* each Gaussian takes for each datapoint

Update  $\mu_j$  based on those datapoints Gaussian  $j$  takes responsibility for

Calculate the *responsibilities* each Gaussian takes for each datapoint

And so on ...



EM for Estimating  $k$  Gaussians (3)

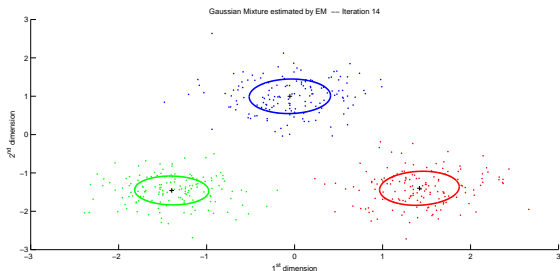
Initialize:  $\mu_1, \dots, \mu_k$  picked at random

Calculate the *responsibilities* each Gaussian takes for each datapoint

Update  $\mu_j$  based on those datapoints Gaussian  $j$  takes responsibility for

Calculate the *responsibilities* each Gaussian takes for each datapoint

And so on ...  
Until convergence



EM for Estimating  $k$  Gaussians (4)

**EM Algorithm:** Pick random initial parameterization  $\langle \mu_1, \dots, \mu_k \rangle$ , then iterate

**E step:** Calculate the expected value  $E[I(S_i = j)]$  of each hidden variable  $s_i$ , assuming the current parameterization  $\langle \mu_1, \dots, \mu_k \rangle$  holds.

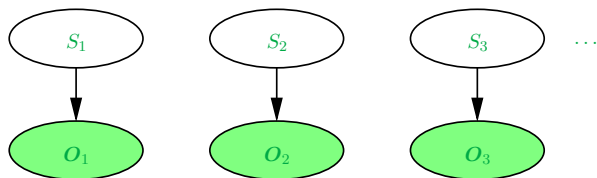
$$E[I(S_i = j)] = \frac{p(\mathbf{x} = \mathbf{x}_i | \mu = \mu_j)}{\sum_{n=1}^k p(\mathbf{x} = \mathbf{x}_i | \mu = \mu_n)}$$

$$p(\mathbf{x} = \mathbf{x}_i | \mu = \mu_j) \propto \exp(-\|\mathbf{x}_i - \mu_j\|^2)$$

**M step:** Calculate a new ML parameterization assuming the value taken on by each hidden variable  $S_i$  is determined by its expected value  $E[S_i]$  calculated above. Also, for  $\langle \mu_1, \dots, \mu_k \rangle$ , we use

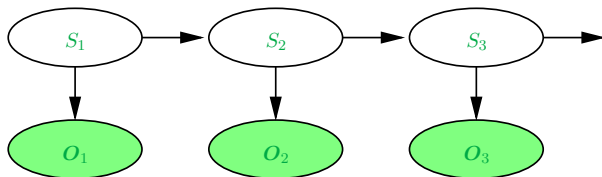
$$\hat{\mu}_j \leftarrow \frac{\sum_{i=1}^m E[I(S_i = j)] \mathbf{x}_i}{\sum_{i=1}^m E[I(S_i = j)]}$$

# What to learn from the $k$ – Gaussians example



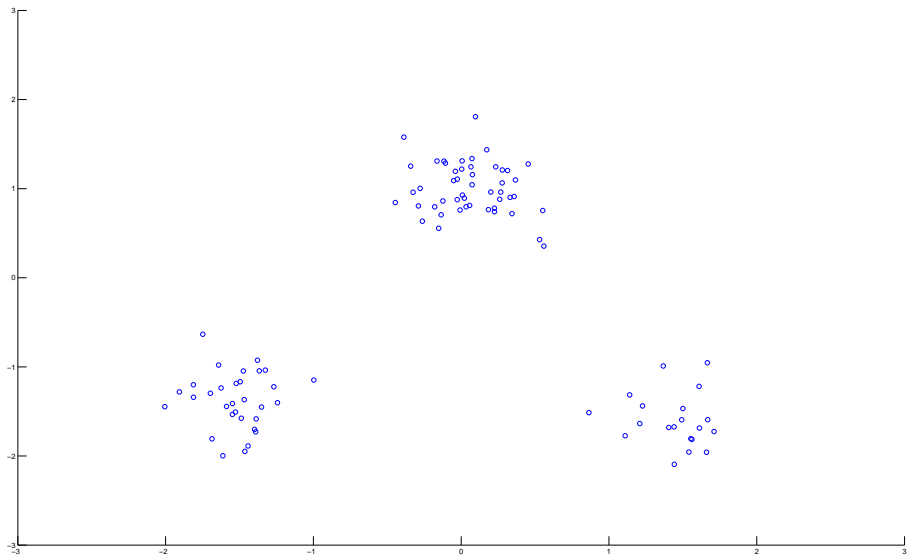
- In the example, the state-variable (called  $S$ ) is unobserved.
- The observations are Gaussians; params depend on parent.
- Learning done by EM:
  - 1 “Fill in” values of unobserved variables.
  - 2 Calculate parameters of distributions given “completed” data.

# What to learn from the $k$ – Gaussians example

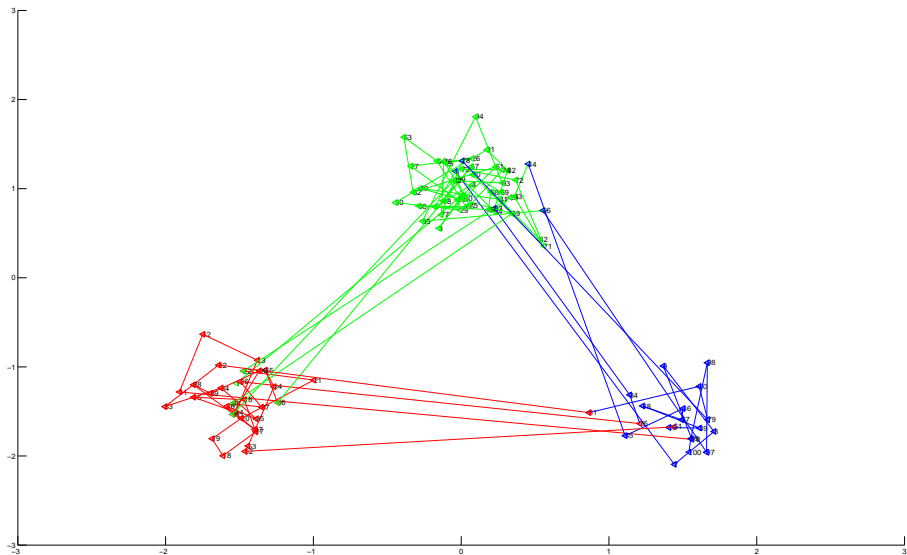


- In the **Hidden Markov models**, the state-variable is unobserved.
- The observations are Gaussians — or MoGs. We focus on Gaussians here; MoGs described by Rabiner.
- Learning in HMMs relates to the “ $k$  – Gaussians” example. However, the **E-step** is a bit more difficult as different  $O_{t-\ell}$  and  $S_t$  interact (as  $S_{t-\ell}, \dots, S_{t-1}$  all are unobserved). Note also that the **M-step** must be extended to estimate  $A$ .

# What it means that the data is dynamic



# What it means that the data is dynamic



# The E-step (1): Rabiner's backwards algorithm



- Accompanying the *Forward* algorithm is a related **Backward algorithm**.
- It calculates  $\beta_t(i) = P(\mathbf{o}_{t+1:T} \mid S_t = i)$ , i.e., the probability of the observation sequence from  $t + 1$  and until the end, given that the system is in state  $i$  at time  $t$ .

## Rabiner's Backward-algorithm:

- 1 Initialization:  $\beta_T = \langle 1 \dots 1 \rangle^T$  (a vector of ones).
- 2 Induction:  $\beta_t = \mathbf{A}\mathbf{B}_{t+1}\beta_{t+1}$ .

# The E-step (2): Rabiner's *scaled* Backward-algorithm



Rabiner's *scaled* Backward-algorithm is like the unscaled backward algorithm, but using the scaling factors from the Forward-phase  $\ell_t$  to generate a scaled message  $r_t$ .

- **Initialisation:**

- $r'_T = [1 \dots 1]^T$ .
- $r_T \leftarrow r'_T / \ell_T$ .

- **Induction:** Do for  $t = T - 1, \dots, 1$ :

- $r'_t = \mathbf{A}\mathbf{B}_{t+1}r_{t+1}$
- $r_t \leftarrow r'_t / \ell_t$ .

# The E-step (3): Calculating the distribution of $S_t$



- If we first run the **Forward** algorithm and then the **Backward** algorithm, this gives us
  - “The forward messages”:  $\alpha_t(j) = P(S_t = j, \mathbf{o}_{1:t})$
  - “The backward messages”:  $\beta_t(j) = P(\mathbf{o}_{t+1:T} | S_t = j)$
- Notice that

$$\alpha_t(j) \cdot \beta_t(j) = P(S_t = j, \mathbf{o}_{1:t}, \mathbf{o}_{t+1:T}) = P(S_t = j, \mathbf{o}_{1:T})$$

- Rabiner defines  $\gamma_t(j) = P(S_t = j | \mathbf{o}_{1:T})$ , and calculates it by

$$\gamma_t(j) = \frac{P(S_t = j, \mathbf{o}_{1:T})}{P(\mathbf{o}_{1:T})} = \frac{\alpha_t(j) \cdot \beta_t(j)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)}$$

- Exactly the same calculations can be done using the scaled versions:

$$\gamma_t(j) = \frac{\mathbf{f}_t(j) \cdot \mathbf{r}_t(j)}{\sum_{i=1}^N \mathbf{f}_t(i) \mathbf{r}_t(i)}$$

# First shot of the M-step



- We have some understanding of the “fill-ins” in terms of  $\gamma_t$ . They hold the same information as the  $z_i$  did in the  $k$  – Gaussians example.
- We can therefore use same intuition as previously for the M-step equations for  $\mu$  and  $\Sigma$  (and also for  $\pi$ ):

$$\hat{\mu}_j \leftarrow \frac{\sum_{t=1}^T \gamma_t(j) \mathbf{o}_t}{\sum_{t=1}^T \gamma_t(j)}$$

$$\hat{\Sigma}_j \leftarrow \frac{\sum_{t=1}^T \gamma_t(j) (\mathbf{o}_t - \hat{\mu}_j)(\mathbf{o}_t - \hat{\mu}_j)^\top}{\sum_{t=1}^T \gamma_t(j)}$$

$$\hat{\pi}_j \leftarrow \gamma_1(j)$$

We still have no information about the **transitions**, i.e., the movements  $s_t \rightarrow s_{t+1}$ . Hence we cannot estimate the transition matrix  $\mathbf{A}$  yet. This is the last piece of the jigsaw.

# The E-step (3): Focus on transitions



- To get an idea of the transitions taking place, Rabiner looks at  $\xi_t(i, j) = P(S_t = i, S_{t+1} = j \mid \mathbf{o}_{1:T})$ .
- Notice that

$$\begin{aligned}
 & \alpha_t(i) \mathbf{A}_{ij} \mathbf{b}_j(\mathbf{o}_{t+1}) \beta_{t+1}(j) \\
 &= P(S_t = i, \mathbf{o}_{1:t}) \cdot P(S_{t+1} = j \mid S_t = i) \cdot \\
 & \quad P(\mathbf{o}_{t+1} \mid S_{t+1} = j) \cdot P(\mathbf{o}_{t+2:T} \mid S_{t+1} = j) \\
 &= P(S_t = i, S_{t+1} = j, \mathbf{o}_{1:T})
 \end{aligned}$$

- Rabiner calculates  $\xi_t(i, j)$  by

$$\xi_t(i, j) = \frac{\alpha_t(i) \mathbf{A}_{ij} \mathbf{b}_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^N \sum_{l=1}^N \alpha_t(k) \mathbf{A}_{kl} \mathbf{b}_l(\mathbf{o}_{t+1}) \beta_{t+1}(l)}$$

# The E-step (4): Using the scaled version



Again, we can get the same results as Rabiner, by just looking at the *scaled* version:

- $\xi_t(i, j)$  can be calculated by

$$\xi_t(i, j) = \frac{\mathbf{f}_t(i) \mathbf{A}_{ij} \mathbf{b}_j(\mathbf{o}_{t+1}) \mathbf{r}_{t+1}(j)}{\sum_{k=1}^N \sum_{l=1}^N \mathbf{f}_t(k) \mathbf{A}_{kl} \mathbf{b}_l(\mathbf{o}_{t+1}) \mathbf{r}_{t+1}(l)}$$

# Rest of the M-step



- As we did when learning in a general Bayes net, we estimate the transition matrix by “counting”:

$$\hat{A}_{ij} \leftarrow \frac{\text{Expected no. transitions from } i \text{ to } j}{\text{Expected no. times leaving } i}$$

- Using the machinery of the previous slides, this gives

$$\hat{A}_{ij} \leftarrow \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}.$$

# What's next?



## Lectures

The “standard lectures” are over; last two weeks are used for to let you ask questions or discuss issues regarding your implementation.

## Making implementation choices

The project description is under-specified! Stuff like how many frequencies to use, how big overlap one should have between widows, no. hidden states, termination of EM, how to choose the observation model, etc. are not covered, and is **for you to experiment with** and discuss with me during the next lectures.

## Finally, a word of warning. . .

Do not postpone everything to the last week! This project is difficult, and will require some time to be solved adequately.