

Automatic Speech Recognition handout (2)

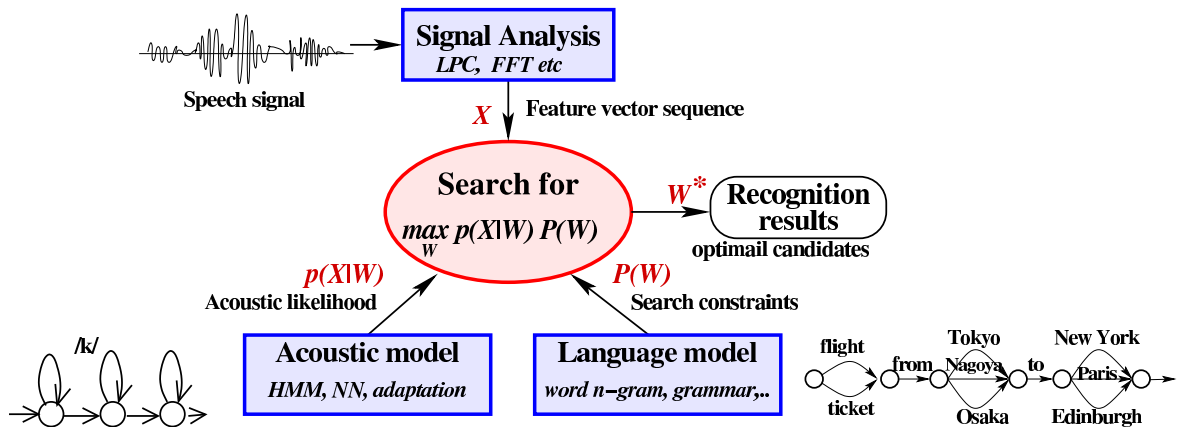
Jan - Mar 2012

Revision : 1.1

— Statistical pattern recognition —

Hiroshi Shimodaira (h.shimodaira@ed.ac.uk)

Speech Recognition recap



(after Sagayama, "Speech Translation Telephony", 1994)

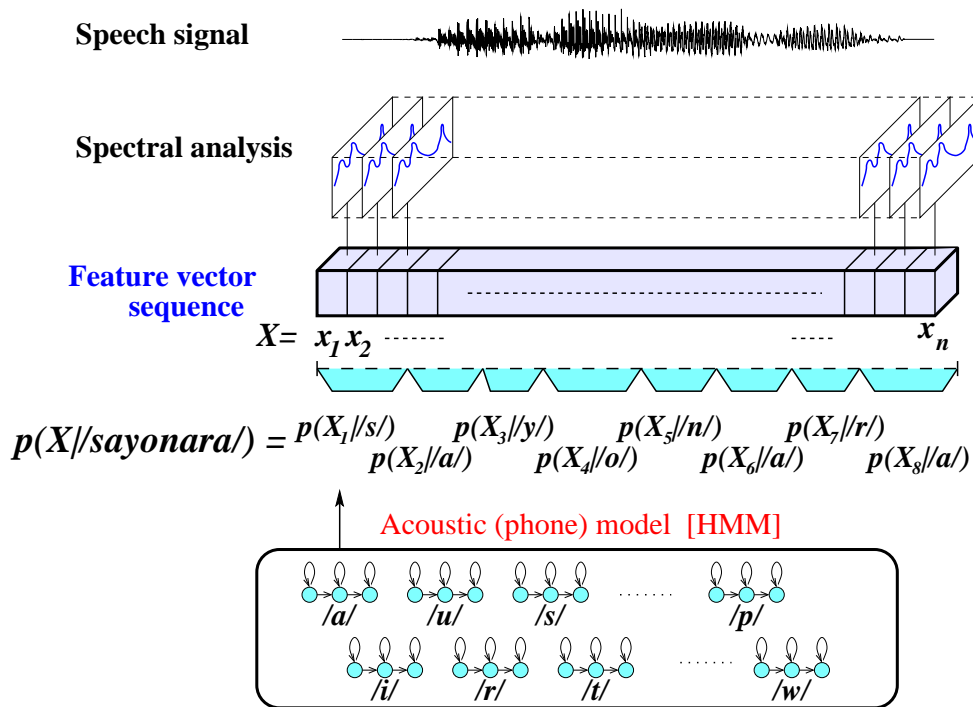
Bayes decision rule:

$$\begin{aligned}
 W^* &= \arg \max_W P(W|X) = \arg \max_W \frac{p(X|W)P(W)}{p(X)} \\
 &= \arg \max_W p(X|W)P(W)
 \end{aligned}$$

$X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T) :$ feature vectors observed

$W = (w_1, w_2, \dots, w_L) :$ hypothesised words

Speech Recognition recap (cont. 2)



How to calculate $p(X|/s/)$?

The **conditional probability** that we observe a feature sequence X for phone $/s/$:

$$p(X|/s/) = p(\mathbf{x}_1, \dots, \mathbf{x}_{T_1}|/s/), \quad \mathbf{x}_i = (x_{1i}, \dots, x_{di})^t \in \mathcal{R}^d$$

We know

- HMM can be employed to calculate this. (**Viterbi** algorithm, **Forward / Backward** algorithm)
- HMM should be trained beforehand with a set of **training samples**. (**Baum-Welch** algorithm (**EM** algorithm), Viterbi training)

To investigate these algorithms in detail, let's start with the simplest case: the length of the sequence is one ($T = 1$), and the dimensionality of x is one ($d = 1$).

$$p(X|/s/) \longrightarrow p(x|/s/)$$

How to calculate $p(x|s)$?

$p(x|s)$: **conditional probability**
(conditional probability density function (**pdf**) of x)

- We learnt that a **Gaussian / normal distribution** function can be employed to approximate the pdf by:

$$P(x|s) = \frac{1}{\sqrt{2\pi}\sigma_{s/}} e^{-\frac{(x-\mu_{s/})^2}{2\sigma_{s/}^2}}$$

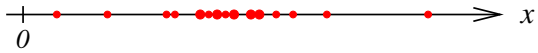
- The function has only two parameters, $\mu_{s/}$ and $\sigma_{s/}$
- Given a set of training samples $\{x_1, \dots, x_N\}$, we estimate $\mu_{s/}$ and $\sigma_{s/}$ by

$$\hat{\mu}_{s/} = \frac{1}{N} \sum_{i=1}^N x_i, \quad \hat{\sigma}_{s/}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu_{s/})^2$$

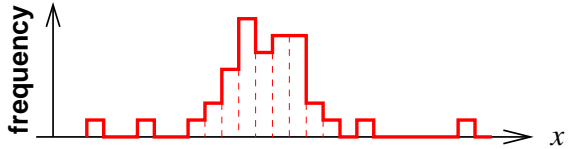
how reliable are these estimates?

how similar $p(x|\hat{\mu}_{s/}, \hat{\sigma}_{s/})$ is to the true one?

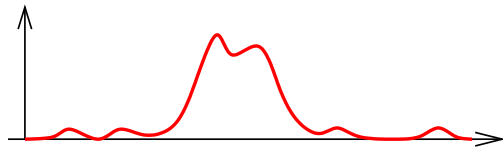
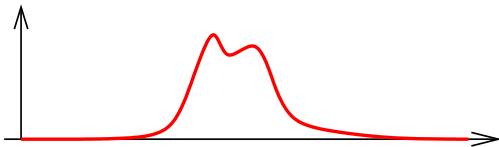
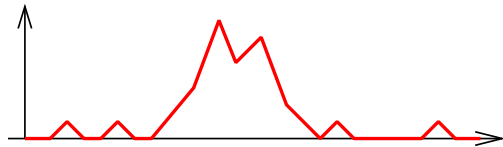
How to estimate $p(x|/s/)$?



(a) observation

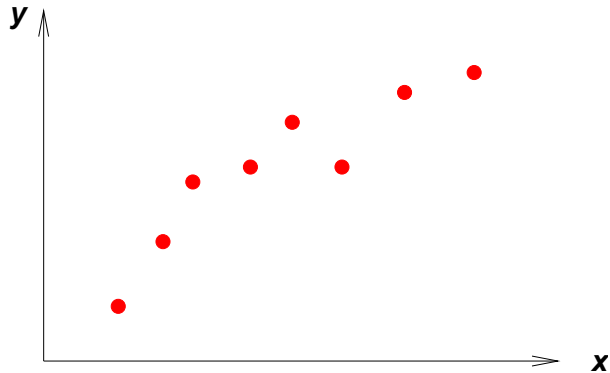


(b) histogram

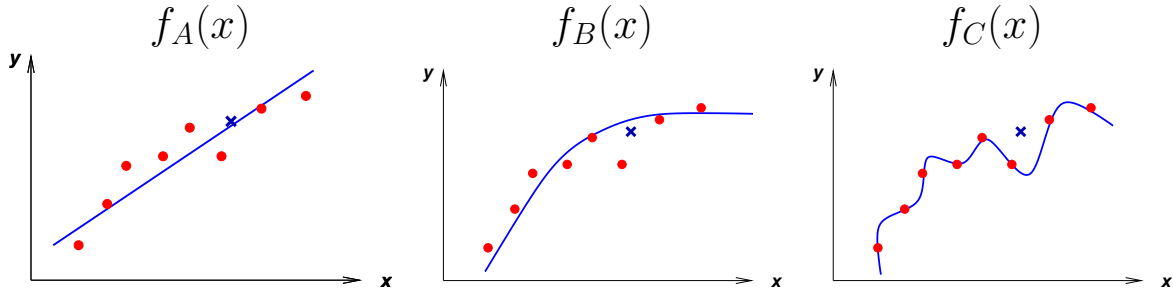


How to estimate $p(x|s)$? (cont. 2)

How does $y = f(x)$ look like?



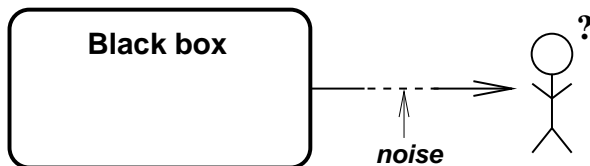
How to estimate $p(x|/s/)$? (cont. 3)



	$f_A(x)$	$f_B(x)$	$f_C(x)$
# of parameters	small	medium	large
Error on training samples	moderate	small	0 (zero)
Error on other samples	moderate?	moderate/large?	huge?

Technical terms: robustness, generalisation, over-fitting

How to estimate $p(x|s)$? (cont. 4)



- What we've observed are just samples from an unobservable system. Even worse, the samples might be distorted.
- With limited amount of samples, it is hard to estimate / identify the system completely, because more than one solution can exist and we do not know which is the best one.
- To find the best one, we need more knowledge about the system.
- Estimating the system from observation is regarded as an “**inverse problem**” or “**ill-posed problem**”.
- To solve an inverse problem, we usually define an **optimisation problem** with some constraints (assumptions, models).

Estimating pdf

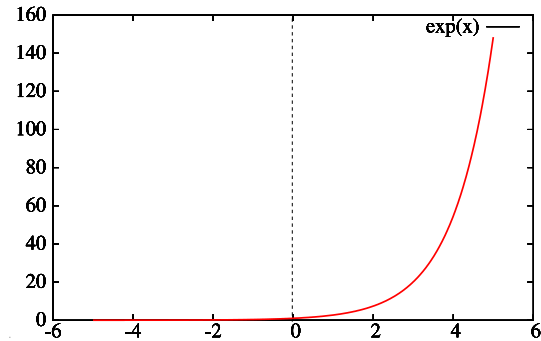
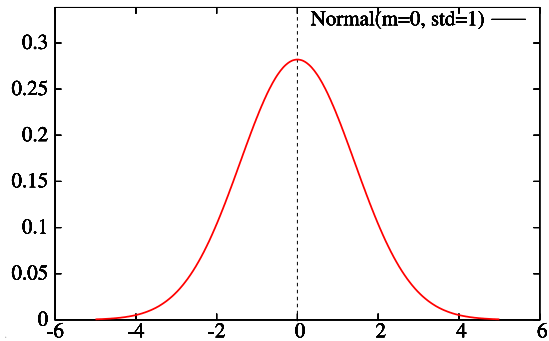
- **Non-parametric approach**
 - **histogram**
 - **kernel method, Parzen window**
 - **neural network**
- **Parametric approach**
 - **Gaussian / normal distribution**
 - **Gaussian mixture model**
 - **etc.**

Gaussian Distribution: 1-dim case

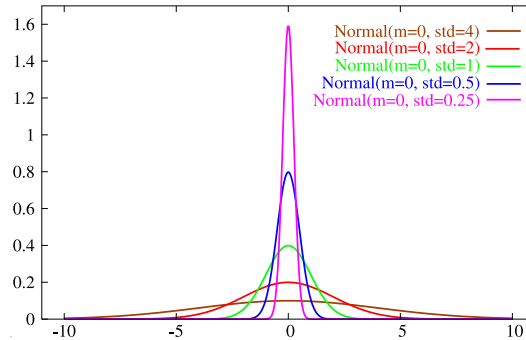
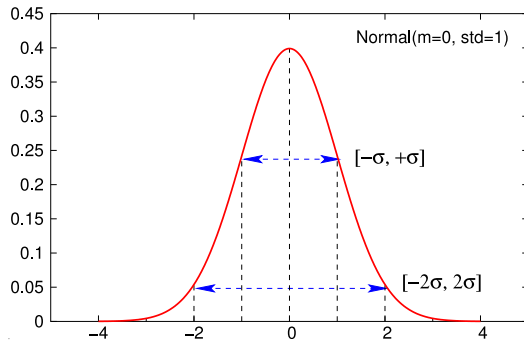
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$y = e^x$$

$$e = 2.7183\dots$$



Gaussian Distribution: 1-dim case (cont. 2)



$$P(-\sigma < x < \sigma) \sim 0.683$$

$$P(-2\sigma < x < 2\sigma) \sim 0.954$$

$$P(-3\sigma < x < 3\sigma) \sim 0.997$$

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

Gaussian Distribution: 1-dim case(cont. 3)

$$P(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \stackrel{\Delta}{=} \mathcal{N}(x; \mu, \sigma)$$

Given samples $\{x_1, \dots, x_N\} = \{x_n\}_{n=1}^N$,

we know how to estimate (infer) the parameters μ, σ :

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \hat{\mu})^2 = \frac{1}{N} \sum_{n=1}^N x_n^2 - \left(\frac{1}{N} \sum_{n=1}^N x_n \right)^2$$

However,

- **Why we can say this is a right inference?**
- **Are there any other inferences?**

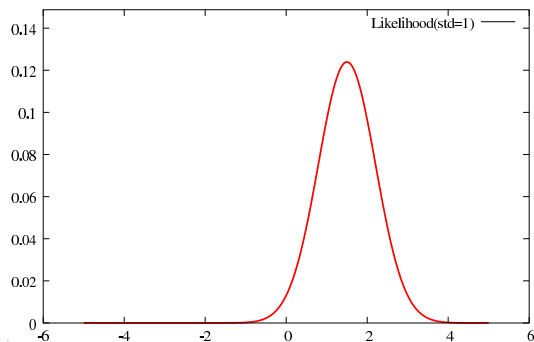
Maximum likelihood estimation

$$p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

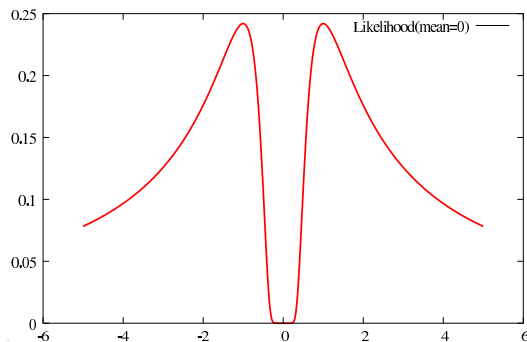
Interpretation of $p(x|\mu, \sigma)$

- **If μ, σ are fixed** $\rightarrow p(x|\mu, \sigma)$ is a function of x .
 $p(x|\mu, \sigma)$: a **probability density function**.
- **If x is fixed** $\rightarrow p(x|\mu, \sigma)$ is a function of μ, σ .
 $p(x|\mu, \sigma)$: a **'likelihood'** function denoted as $L(\mu, \sigma; x)$

Maximum likelihood estimation (cont. 2)



likelihood as a function of μ



likelihood as a function of σ

$$L(\mu, \sigma; x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

ML estimation (MLE)

Assume $X = \{x_n\}_{n=1}^N$ are chosen independently (i.i.d.)¹.

The probability of observing X is

$$p(x_1, \dots, x_N | \mu, \sigma) = p(x_1 | \mu, \sigma) \cdots p(x_N | \mu, \sigma) = \prod_{n=1}^N p(x_n | \mu, \sigma)$$
$$\triangleq L(\mu, \sigma | X)$$

Find parameters μ, σ that maximise the likelihood:

$$\max_{\mu, \sigma} L(\mu, \sigma | X)$$

¹i.i.d: independent identically-distributed, i.e. data points that are drawn independently from the same distribution.

Maximum likelihood estimation(cont. 4)

(Solution) Take the derivative of $\log L(\mu, \sigma|X)$ and set it to zero.

$$\begin{aligned}\ell(\mu, \sigma|X) &= \log L(\mu, \sigma|X) = \log \prod_{n=1}^N p(x_n|\mu, \sigma) = \sum_{n=1}^N \log p(x_n|\mu, \sigma) \\ &= -N \log(\sqrt{2\pi}\sigma) - \sum_{n=1}^N \frac{(x_n - \mu)^2}{2\sigma^2}\end{aligned}$$

$$\frac{\partial \ell}{\partial \mu} = -2 \sum_{n=1}^N \frac{x_n - \mu}{2\sigma^2} = 0 \quad \Rightarrow \quad \mu = \frac{1}{N} \sum_{n=1}^N x_n$$

$$\begin{aligned}\frac{\partial \ell}{\partial \sigma} &= -N \frac{1}{\sqrt{2\pi}\sigma} + \sum_{n=1}^N \frac{(x_n - \mu)^2}{\sigma^3} = 0 \\ &\Rightarrow \quad \sigma^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu)^2\end{aligned}$$

Limitation of a Gaussian pdf

- Gaussian distribution is assumed
- does not fit more complicated distributions (bimodal, etc.)

Solutions

- Transform features
(coordinate **non-linear** transformation)
- Use other distribution functions
 - Poisson dist.
 - exponential dist.
 - gamma dist.
 - log-normal dist.
- Use a mixture of Gaussian distribution functions
- Use neural networks, kernel approaches

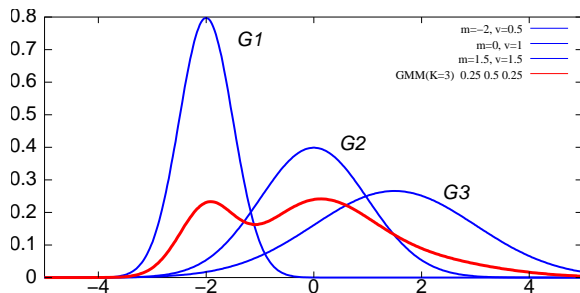
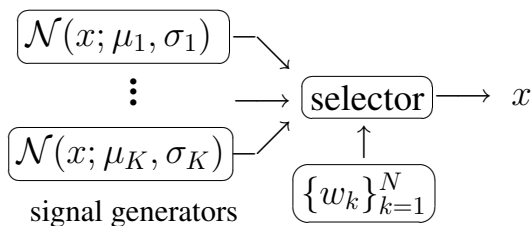
Gaussian Mixture Model (GMM)

Idea Approximate a “true” distribution by more than one Gaussian distribution.

$$p(x|\Lambda) = \sum_{k=1}^K w_k p(x|\mu_k, \sigma_k) = \sum_{k=1}^K w_k \mathcal{N}(x; \mu_k, \sigma_k)$$

where $\Lambda = \{\lambda_k\}_{k=1}^K = \{\mu_k, \sigma_k\}_{k=1}^K \cdots$ a set of model parameters
 $\sum_{k=1}^K w_k = 1$

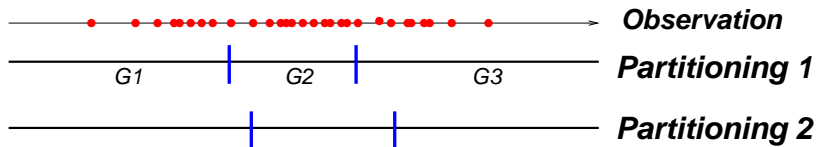
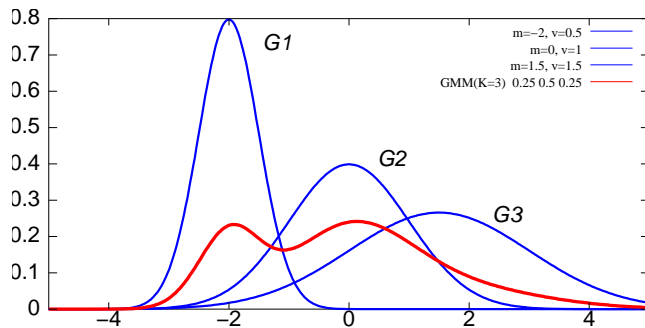
View as a generative-model



Training GMM

Given a set of training samples x_1, x_2, \dots, x_N ,
assuming a mixture of K Gaussians,
we don't know from which Gaussian model each x_i came.

⇒ ML training can not be applied directly !



Training GMM_(cont. 2)

Solution Assume each x_n has a label y_n showing from which model the data came.

i.e. y_n takes an index of Gaussian model $1, \dots, K$.

observations	x_1	x_2	\dots	x_N
labels (unseen)	y_1	y_2	\dots	y_N

Then we would be able to use ML estimation recursively to find the optimal partitioning.

Approach A (hard k -means clustering) \dots assume each x_n belongs to an exact Gaussian y_n .

Approach B (soft k -means clustering) \dots assume each x_n belongs to Gaussian k with probability $P(y_n = k)$.
(y_n is treated as a random variable)

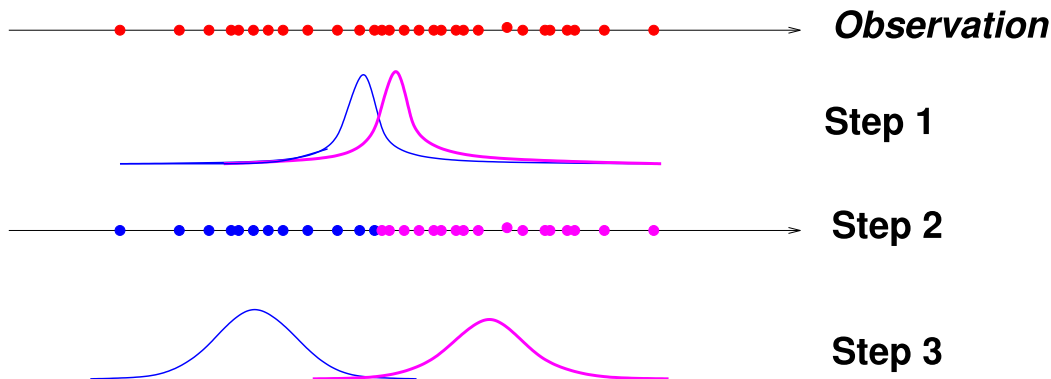
Training GMM by k -means clustering (hard version)

Step 1: Initialisation. Set model parameters Λ arbitrarily.

Step 2: Clustering: $y_n = \arg \max_k p(x_n | \lambda_k)$ for $k = 1, \dots, N$

Step 3: Re-estimation: estimate model parameters Λ .

Step 4: Iteration: Repeat steps 2 and 3 until a predefined convergence criterion is satisfied. (estimate $\{w_k\}$ when exits.)



Training GMM by k -means clustering (soft version)

Problem of the hard k means clustering: non guarantee of ML.

The **EM algorithm** enables us to maximise the likelihood

$$L(\Lambda|X) = \prod_{n=1}^N p(x_n|\Lambda)$$

by iteratively maximising Q function with respect to $\Lambda^{(t+1)}$:

$$Q(\Lambda^{(t)}, \Lambda^{(t+1)}) = \sum_Y P(Y|X, \Lambda^{(t)}) \log p(X, Y|\Lambda^{(t+1)})$$

This is the expectation of log-likelihood from **complete data** (X, Y) .
To maximise Q , we take the derivative of Q with respect to each parameter and set it to zero.

The Expectation-Maximisation Algorithm

The EM algorithm [Dempster, Laird, and Rubin, 1977]

Step 1: Initialisation: Set $t = 0$, and choose an initial estimate $\Lambda^{(0)}$.

Step 2: E-Step: Compute $Q(\Lambda^{(t)}, \Lambda)$ based on current parameter $\Lambda^{(t)}$.

Step 3: M-Step: Compute $\Lambda^* = \arg \max_{\Lambda} Q(\Lambda^{(t)}, \Lambda)$ to maximise Q .

Step 4: Iteration: Set $\Lambda^{(t+1)} = \Lambda^*$ and $t \leftarrow t + 1$, repeat from Step 2 until convergence.

- (Note)**
- The algorithm is a generalisation of MLE, when we have incomplete data (i.e. some data are unobservable (hidden)).
 - The algorithm gives just a general idea, and actual implementation differs depending on the problems.

EM algorithm for GMM

$$\begin{aligned} Q(\Lambda^{(t)}, \Lambda^{(t+1)}) &= \sum_{n=1}^N Q_i(\Lambda^{(t)}, \Lambda^{(t+1)}) = \sum_{n=1}^N \sum_{y_n} P(y_n|x_n, \Lambda^{(t)}) \log p(x_n, y_n|\Lambda^{(t+1)}) \\ &= \sum_{n=1}^N \sum_{y_n} \frac{p(x_n, y_n|\Lambda^{(t)})}{p(x_n|\Lambda^{(t)})} \log p(x_n, y_n|\Lambda^{(t+1)}) \\ &= \sum_{k=1}^K \gamma_k \log w_k^{(t+1)} + \sum_{k=1}^K Q_\pi(\Lambda^{(t)}, \lambda_k^{(t+1)}) \end{aligned}$$

where

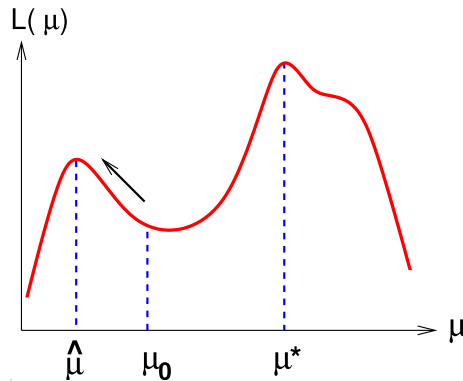
$$\begin{aligned} \gamma_k &= \sum_{n=1}^N \gamma_k^n, \quad \gamma_k^n = \frac{w_k^{(t)} p(x_n|\lambda_k^{(t)})}{p(x_n|\Lambda^{(t)})} \\ Q_\pi(\Lambda^{(t)}, \lambda_k^{(t+1)}) &= \sum_{n=1}^N \gamma_k^n \log p(x_n|\lambda_k^{(t+1)}) \\ w_k^{(t+1)} &= \gamma_k / \sum_{k=1}^K \gamma_k = \gamma_k / N \\ m_k^{(t+1)} &= \sum_{n=1}^N \gamma_k^n x_n / \sum_{n=1}^N \gamma_k^n, \quad \sigma_k^{(t+1)} = \sum_{n=1}^N \gamma_k^n (x_n - m_k^{(t)})^2 / \sum_{n=1}^N \gamma_k^n \end{aligned}$$

Limitation of ML-trained GMM

- **Local optimum problem** (the EM algorithm does not guarantee the global optimum)
- **No mechanism for determining K** (# of mixtures).

Initial value problem of GMM

- The EM iteration converges at a local maximum, which might be different from the global maximum.
- The performance of the algorithm highly relies on how appropriate initial value the algorithm starts from.
- As we increase the number of Gaussian mixture components, the chance we suffer the problem increases.



Initial value problem of GMM_(cont. 2)

Techniques to avoid the local maximum problem

Method-A: Use k -means clustering (hard) to find initial partitions of data, and calculate initial parameters of GMM.

Method-B: Increase the mixture number k of GMM gradually. (start from $k = 1$, and then $k \leftarrow k * 2$.)

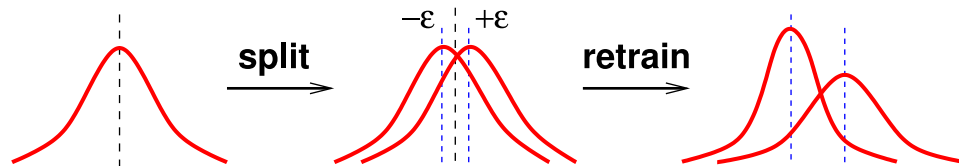


Fig. Mixture splitting

Remaining problems with GMM

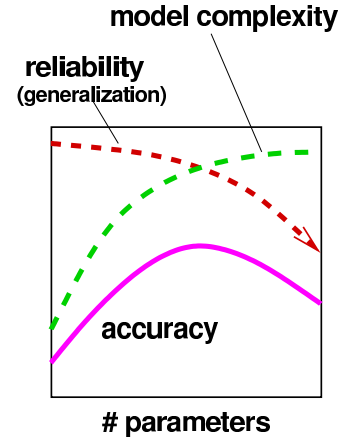
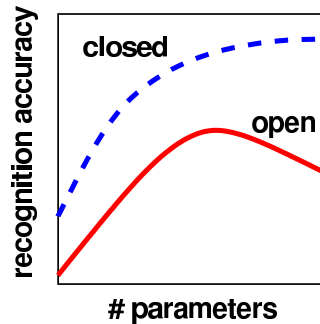
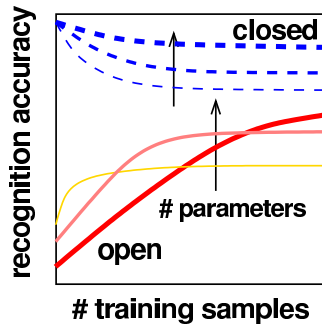
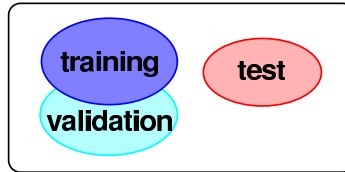
- Avoiding the local maximum problem is not enough
- There is another problem: **“over-fitting problem”**
 - the likelihood on training data increases, as k (# of mixtures) increases.
 - The ML training does not tell us what k is optimal.

Solutions

- use model selection technique such as
 - **AIC** (Akaike’s information criterion)
 - **MDL** (minimum description length)
- **“Occam’s razor”**:
“Entia non sunt multiplicanda praeter necessitatem”
(accepts the simplest explanation that fits the data)
- use **validation data**

Remaining problems with GMM (cont. 2)

Data set



Maximum a posteriori (MAP) estimation

X : a set of training data $\{x_n\}_{n=1}^N$

Λ : a set of parameters to estimate

ML estim. $\Lambda^* = \arg \max_{\Lambda} p(X|\Lambda)$

MAP estim. $\Lambda^* = \arg \max_{\Lambda} p(\Lambda|X) = \arg \max_{\Lambda} p(X|\Lambda)p(\Lambda)$

$p(\Lambda)$: a priori distribution over Λ

$p(\Lambda|X)$: a posteriors distribution after observing X

$$p(\Lambda|X) = \frac{p(X|\Lambda)p(\Lambda)}{p(X)} = \frac{p(X|\Lambda)p(\Lambda)}{\int p(X|\Lambda')p(\Lambda')d\Lambda'}$$

- The object function takes the reliability of parameters, i.e. the a priori distribution, into account
- Regarded as a penalised (regularised) version of MLE
- How to define/calculate the **prior** $p(\Lambda)$?
⇒ several approaches have been proposed

Bayesian estimation

p.d.f. estimation based on

- **point estimation**

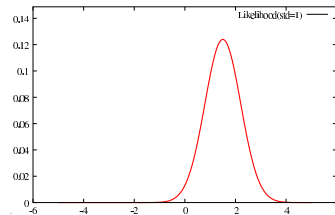
$$\hat{p}(\mathbf{x}|X) = p(\mathbf{x}|X, \Lambda^*), \quad \Lambda^* = \begin{cases} \max_{\Lambda} p(X|\Lambda), & \text{MLE} \\ \max_{\Lambda} p(X|\Lambda)p(\Lambda), & \text{MAP} \end{cases}$$

assuming parameter Λ is fixed but unknown.

- **interval estimation (Bayesian estimation)**

$$\hat{p}(\mathbf{x}|X) = \int p(\mathbf{x}|X, \Lambda)p(\Lambda|X)d\Lambda$$

where $p(\Lambda|X) = \frac{p(X|\Lambda)p(\Lambda)}{\int p(X|\Lambda)p(\Lambda)d\Lambda}$



assuming the parameters Λ are random variables with a prior distribution $p(\Lambda)$

Multivariate Gaussian pdf

One variable

data: $\mathbf{x}_n = (x_n)$

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

Two variables

$$\mathbf{x}_n = (x_{1n}, x_{2n})^t = \begin{pmatrix} x_{1n} \\ x_{2n} \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}$$

$$\mu_1 = \frac{1}{N} \sum_{i=1}^N x_{1i}, \quad \mu_2 = \frac{1}{N} \sum_{i=1}^N x_{2i}$$

$$\sigma_{11} = \frac{1}{N} \sum_{i=1}^N (x_{1i} - \mu_1)^2, \quad \sigma_{22} = \frac{1}{N} \sum_{i=1}^N (x_{2i} - \mu_2)^2$$

$$\sigma_{12} = \frac{1}{N} \sum_{i=1}^N (x_{1i} - \mu_1)(x_{2i} - \mu_2) = \sigma_{21}$$

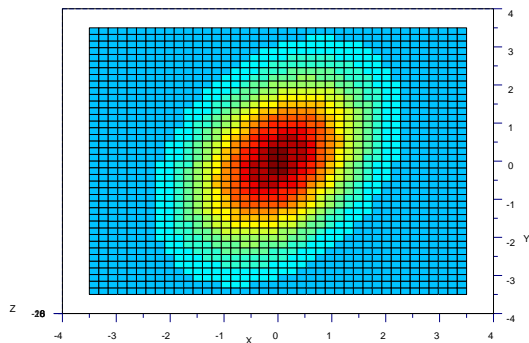
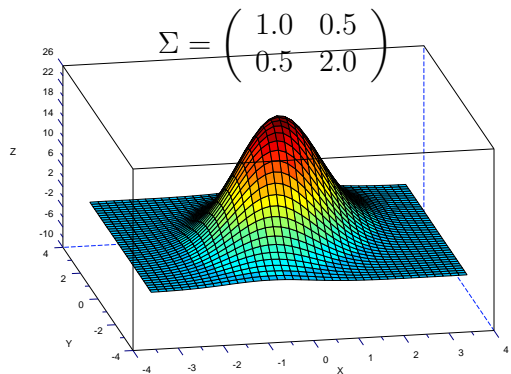
Multivariate Gaussian pdf (cont. 2)

mean vector: $\boldsymbol{\mu} = (\mu_1, \mu_2)^t$

covariance matrix: $\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}$

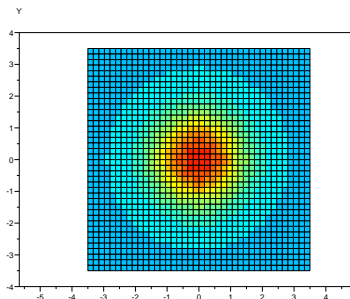
$$p(\mathbf{x} | \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

where d denotes dimensions (here $d = 2$).

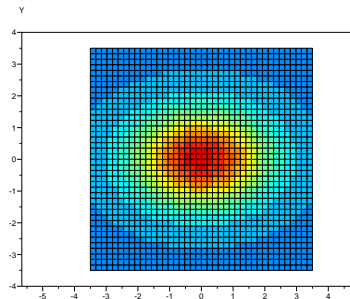


Multivariate Gaussian pdf (cont. 3)

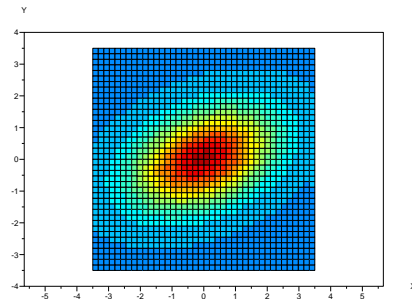
$$\Sigma = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$



$$\Sigma = \begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$



$$\Sigma = \begin{pmatrix} 2.0 & 0.5 \\ 0.5 & 1.0 \end{pmatrix}$$

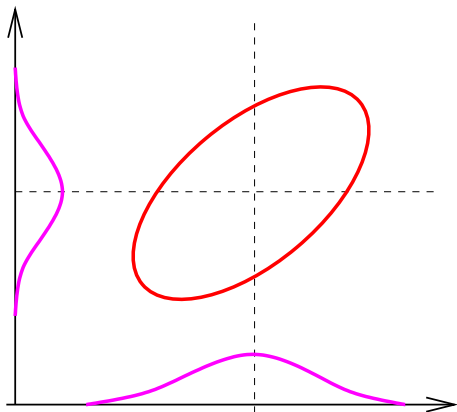


x_1 and x_2 are called **uncorrelated** if $\sigma_{12} = 0$.

Gaussian pdf with diagonal covariances

**Gaussian pdf
with a full-covariance**

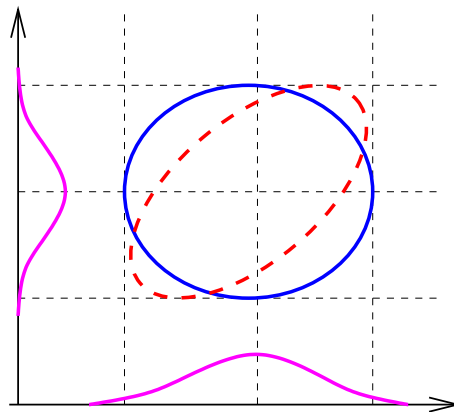
$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}$$



**Gaussian pdf
with diagonal-covariances**

$$\Sigma = \begin{pmatrix} \sigma_{11} & 0 \\ 0 & \sigma_{22} \end{pmatrix}$$

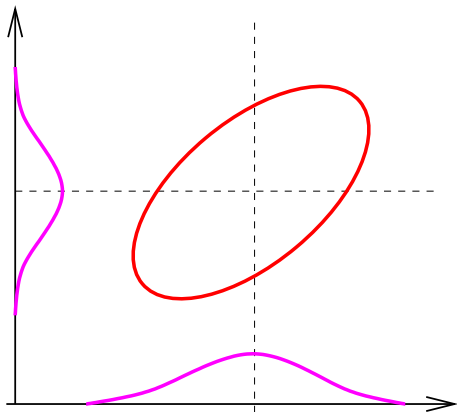
$$p(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = p(x_1|\mu_1, \sigma_{11}) p(x_2|\mu_2, \sigma_{22})$$



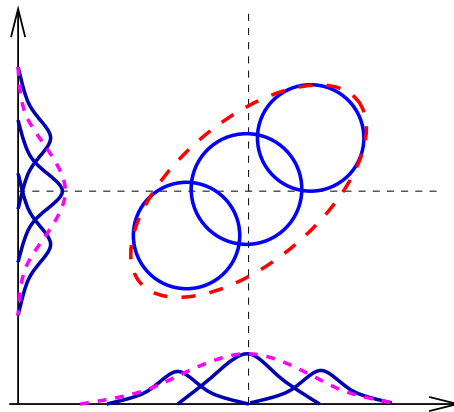
Gaussian pdf with diagonal covariances_(cont. 2)

GMM with diag. covs. can model dependencies (correlations) !

**Gaussian pdf
with full-covariance**



**GMM pdf
with diagonal-covariance**



Gaussian pdf with diagonal covariances(cont. 3)

Question why we normally use GMM with diagonal covariances rather than the one with full covariances?

What has not been discussed yet

$$W^* = \arg \max_W P(W|X) = \arg \max_W p(X|W)P(W)$$

$X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)$: feature vectors observed

$W = (w_1, w_2, \dots, w_L)$: hypothesised words

■ Assumptions so far

■ $T = 1$: X consists of a single frame

■ $L = 1$ and $W = /s/$ (single phoneme $/s/$)

■ Calculation of $P(X|W)$ and $P(W)$ under those conditions:

■ X is a sequence of feature vectors

■ W is a sequence of words/phonemes (L is unknown)

■ boundaries of words/phonemes (+ silence) are unknown

Basic idea of estimating error rate

1. Prepare two subsets of the dataset:

training set to train the recogniser

test(validation) set to evaluate recognition performance

* these two subsets should be mutually exclusive (no overlap between them)

2. Train the recogniser using the training set.

3. Carry out a recognition experiment of the recogniser on the test set, and calculate the recognition error:

$$R_{\text{emp}} = \frac{\text{\# of misrecognised data}}{\text{\# of input data}} \quad \dots \quad \left(\begin{array}{l} \text{test set error, or} \\ \text{empirical error} \end{array} \right)$$

4. Repeat the recognition experiment with changing training and test sets to estimate the expectation of error:

generalisation error: $R = E[R_{\text{emp}}]$

http://en.wikipedia.org/wiki/Generalization_error

k -fold Cross-Validation (CV)

Step 1 split the dataset randomly into k disjoint sets of equal size.

Step 2 train and test the recogniser k times: each time with choosing one set for validation and the remaining $k - 1$ sets for training.

Step 3 Estimate the performance by taking the mean of the k errors.

k-fold CV	subsets
2-fold CV	Set2-0, Set2-1
5-fold CV	Set5-0, Set5-1, Set5-2, Set5-3, Set5-4

Similar techniques: leave-one-out, held-out, jackknife, bootstrap

References:

http://en.wikipedia.org/wiki/Cross_validation

<http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-12.html>

References

■ Books:

- Christopher M. Bishop, “Pattern Recognition and Machine Learning”, ISBN 0387310738, Springer-Verlag, 2006.

■ Web pages:

Normal (Gaussian) Distribution

- http://en.wikipedia.org/wiki/Normal_distribution

Maximum Likelihood Estimation (MLE)

- http://en.wikipedia.org/wiki/Maximum_likelihood

GMM

- http://en.wikipedia.org/wiki/Mixture_model

EM algorithm

- http://en.wikipedia.org/wiki/Expectation-maximization_algorithm