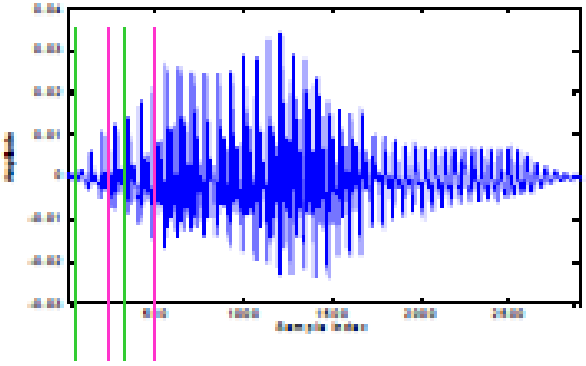
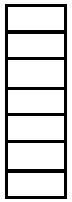
Audio Feature Extraction



Feature Extraction

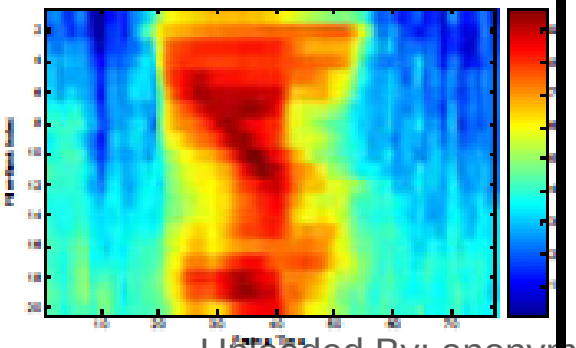


Feature vector o_n

Frame-time index n

o_1 o_2 . . . o_n $o_{nFrames}$

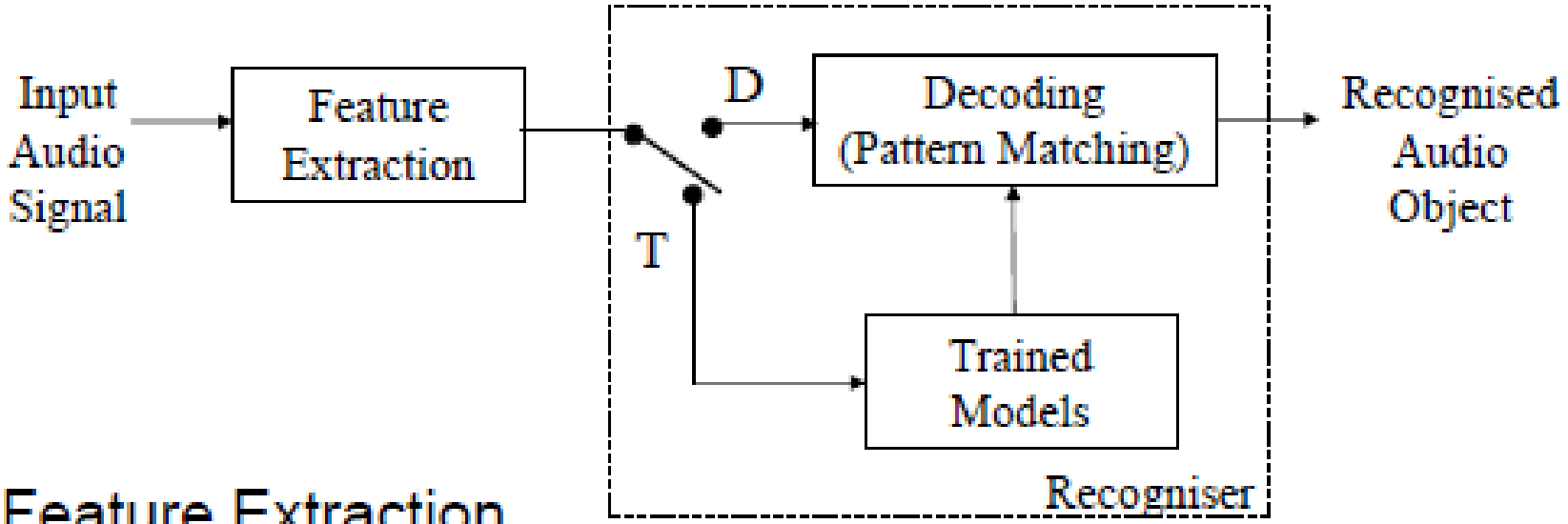
Dim



Output:

- sequence of feature vectors o_n, i.e., a matrix of size: *Dim x nFrames*
- similar to spectrogram

General Scheme of AAR



Feature Extraction

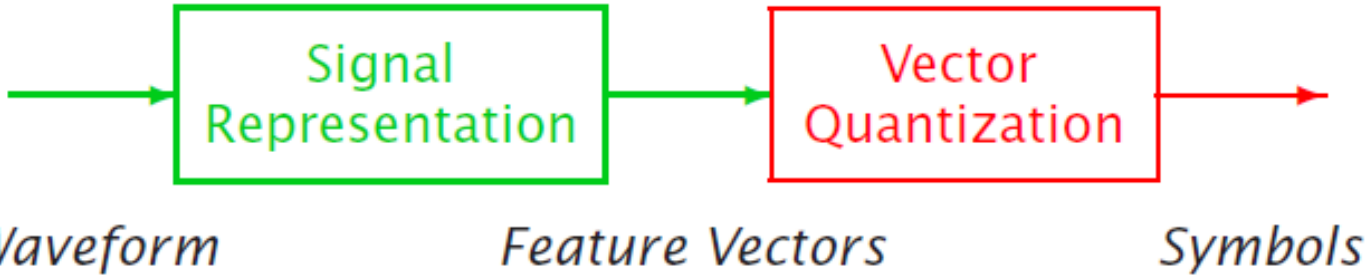
Pattern Modelling/Recognition

- Training stage – using training data set, create a model of each class (object) to be recognized
- Recognition stage – perform a pattern matching – given an unknown piece of audio, assign it to the class of the model which fits best (or find the model which is most likely to have produced that audio)

Pattern Recognition

- Typically we have:
  - A set of **classes** $C_1, \ldots, C_K$, each class characterised by a model
  - A sequence of feature vectors $Y = y_1, \ldots, y_T$
  - The classifier computes probability $P(Y|C_k)$ that the class C_k is the correct explanation of Y
- Classes in our case
  - Speech, Music, (other)
- What model we should use to describe each class?

Acoustic Modeling



- Signal representation produces feature vector sequence
- Multi-dimensional sequence can be processed by:
  - Methods that directly model continuous space
  - *Quantizing* and modelling of discrete symbols
- Main advantages and disadvantages of quantization:
  - Reduced storage and computation costs
  - Potential loss of information due to quantization

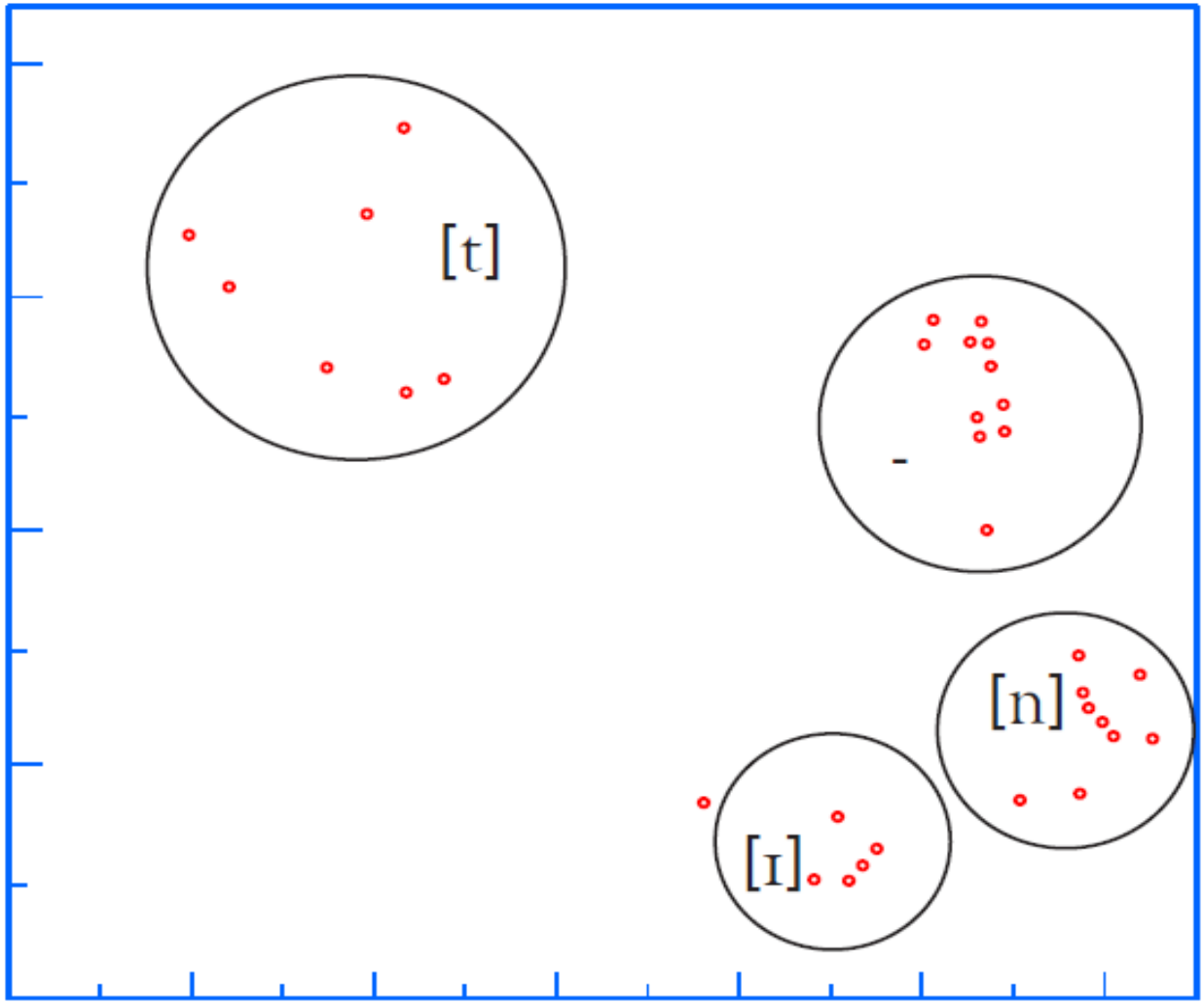
Vector Quantization (VQ)

- Used in signal compression, speech and image coding
- More efficient information transmission than scalar quantization (can achieve less that 1 bit/parameter)
- Used for discrete acoustic modelling since early 1980s
- Based on standard clustering algorithms:
  - Individual cluster centroids are called codewords
  - Set of cluster centroids is called a codebook
  - Basic VQ is K-means clustering
  - Binary VQ is a form of top-down clustering (used for efficient quantization)

VQ and clustering

- Clustering is an example of unsupervised learning
  - Number and form of classes $\{C_i\}$ unknown
  - Available data samples $\{\boldsymbol{x}_i\}$ are unlabeled
  - Useful for discovery of data structure before classification or tuning or adaptation of classifiers
- Results strongly depend on the clustering algorithm

Acoustic Modeling Example



Clustering Issues

- What defines a cluster?
  - Is there a prototype representing each cluster?
- What defines membership in a cluster?
  - What is the distance metric, $d(\boldsymbol{x}, \boldsymbol{y})$?
- How many clusters are there?
  - Is the number of clusters picked before clustering?
- How well do the clusters represent unseen data?
  - How is a new data point assigned to a cluster?

K-means clustering

- Used to group data into K clusters, $\{C_1, \ldots, C_K\}$
- Each cluster is represented by mean of assigned data
- Iterative algorithm converges to a local optimum:
  - Select K initial cluster means, $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$
  - Iterate until stopping criterion is satisfied:
    1. Assign each data sample to the closest cluster

$$\boldsymbol{x} \epsilon C_i, \quad d(\boldsymbol{x}, \boldsymbol{\mu}_i) \leq d(\boldsymbol{x}, \boldsymbol{\mu}_j), \quad \forall i \neq j$$

    2. Update K means from assigned samples

$$\boldsymbol{\mu}_i = E(\boldsymbol{x}), \quad \boldsymbol{x} \epsilon C_i, \quad 1 \leq i \leq K$$

- Nearest neighbor quantizer used for unseen data

K-means example: K=3

- Random selection of 3 data samples for initial means
- Euclidean distance metric between means and samples

K-means Prosperities

- Usually used with a Euclidean distance metric

$$d(\boldsymbol{x}, \boldsymbol{\mu}_i) = \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2 = (\boldsymbol{x} - \boldsymbol{\mu}_i)^t(\boldsymbol{x} - \boldsymbol{\mu}_i)$$

- The total distortion, $\mathcal{D}$, is the sum of squared error

$$\mathcal{D} = \sum_{i=1}^{K} \sum_{\boldsymbol{x} \epsilon C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$$

- $\mathcal{D}$ decreases between n^{th} and $n + 1^{st}$ iteration

$$\mathcal{D}(n+1) \leq \mathcal{D}(n)$$

- Also known as Isodata, or generalized Lloyd algorithm
- Similarities with Expectation-Maximization (EM) algorithm for learning parameters from unlabeled data

K-means clustering: Initialization

- K-means converges to a local optimum
  - Global optimum is not guaranteed
  - Initial choices can influence final result

$K = 3$

- Initial K-means can be chosen randomly
  - Clustering can be repeated multiple times
- Hierarchical strategies often used to seed clusters
  - Top-down (divisive) (e.g., binary VQ)
  - Bottom-up (agglomerative)

K-means clustering: Stopping Criterion

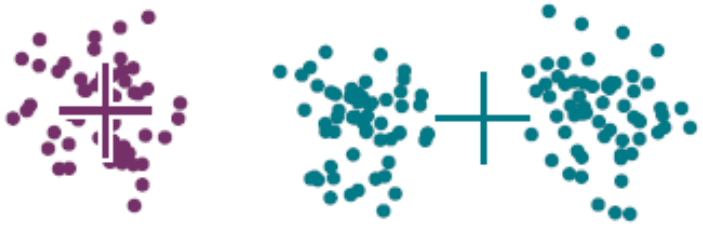
Many criterion can be used to terminate K-means :

- No changes in sample assignments
- Maximum number of iterations exceeded
- Change in total distortion, $\mathcal{D}$, falls below a threshold

$$1 - \frac{\mathcal{D}(n+1)}{\mathcal{D}(n)} < T$$

K-means Issues: Number of clusters

- In general, the number of clusters is unknown



$K = 2$

$K = 4$

- Dependent on clustering criterion, space, computation or distortion requirements, or on recognition metric

Clustering Issues: Distance metric

Distance metrics strongly influence cluster shapes:

- Normalized dot-product: $\frac{\boldsymbol{x}^t \boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}$
- Euclidean: $\|\boldsymbol{x}-\boldsymbol{\mu}_i\|^2 = (\boldsymbol{x}-\boldsymbol{\mu}_i)^t(\boldsymbol{x}-\boldsymbol{\mu}_i)$
- Weighted Euclidean: $(\boldsymbol{x}-\boldsymbol{\mu}_i)^t \boldsymbol{W}(\boldsymbol{x}-\boldsymbol{\mu}_i)$ (e.g., $\boldsymbol{W}=\Sigma^{-1}$)
- Minimum distance (chain): $min \quad d(\boldsymbol{x}, \boldsymbol{x}_i), \quad \boldsymbol{x}_i \epsilon C_i$
- Representation specific ...

Clustering Issues: Training and Testing Data

- Training data performance can be arbitrarily good e.g.,

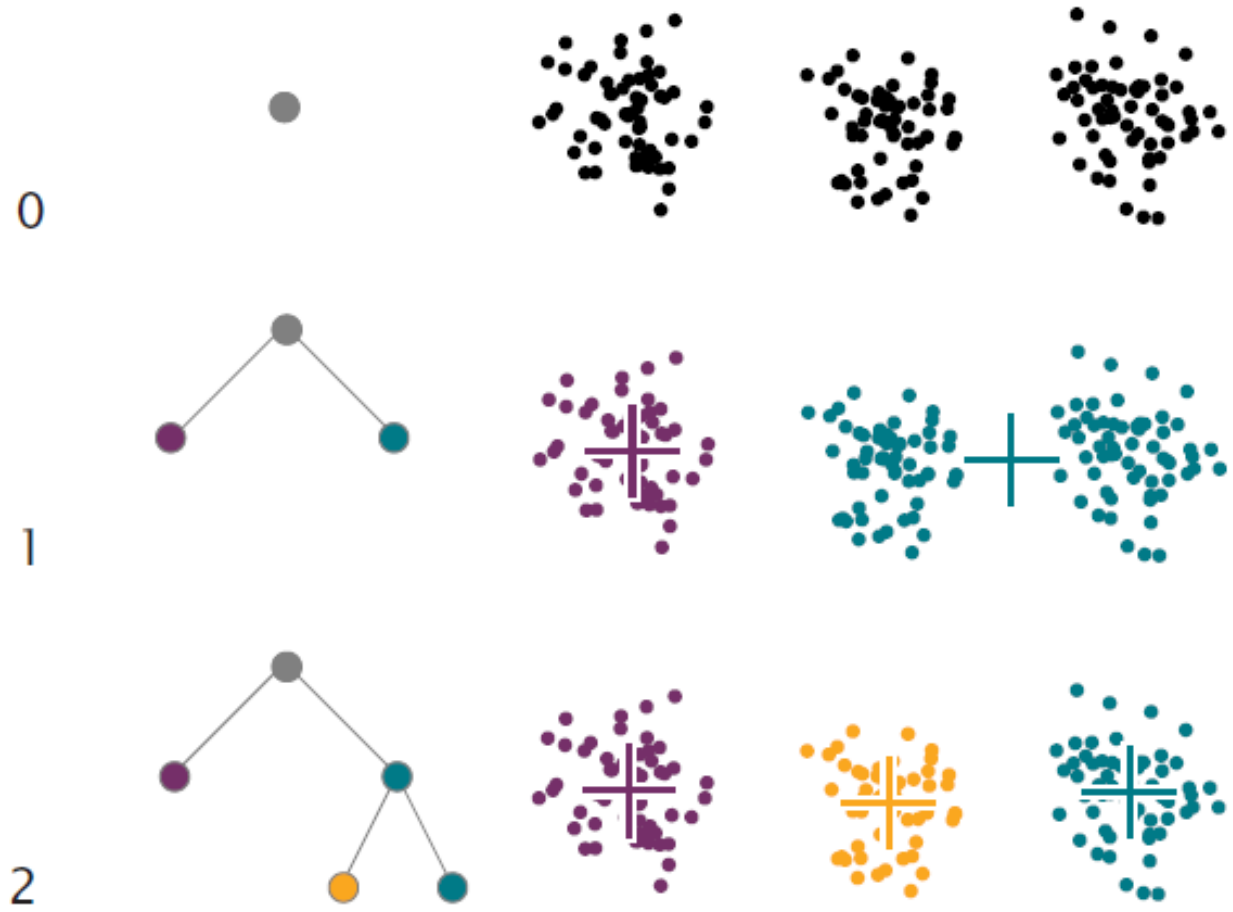
$$\lim_{K \to \infty} \mathcal{D}_K = 0$$

- Independent test data needed to measure performance
  - Performance can be measured by distortion, $\mathcal{D}$, or some more relevant speech recognition metric
  - Robust training will degrade minimally during testing
  - Good training data closely matches test conditions
- Development data are often used for refinements, since through iterative testing they can implicitly become a form of training data

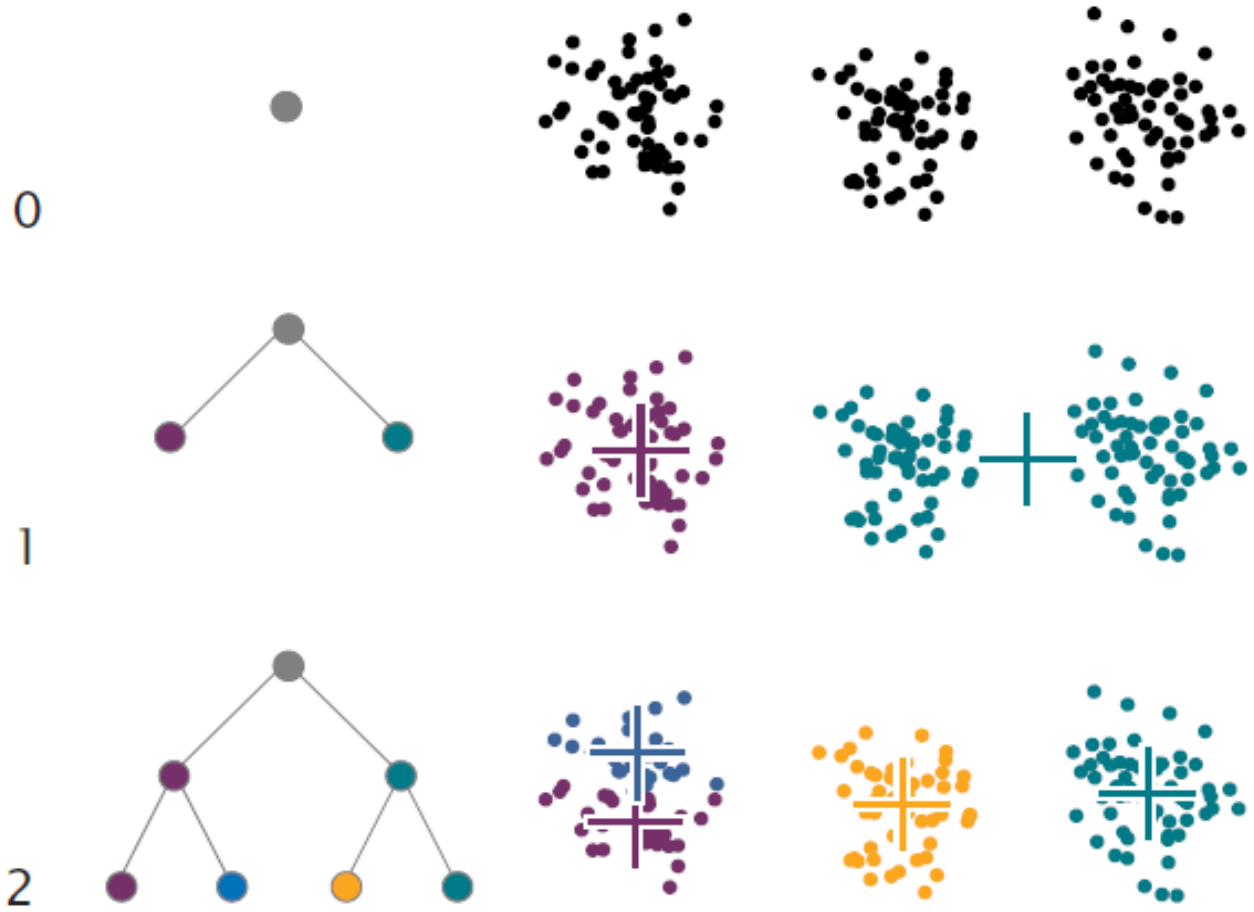
Hierarchical Clustering

- Clusters data into a hierarchical class structure
- Top-down (divisive) or bottom-up (agglomerative)
- Top-down (divisive)
  - Creates hierarchy by successively splitting clusters into smaller groups
  - On each iteration, one or more of the existing clusters are split apart to form new clusters
  - The process repeats until a stopping criterion is met
  - Divisive techniques can incorporate pruning and merging heuristics which can improve the final result

Example of Non-Uniform Divisive clustering



Example of Uniform Divisive clustering



Divisive clustering Example: Binary VQ

- Often used to create $M = 2^B$ size codebook (B bit codebook, codebook size M)
- Uniform binary divisive clustering used
- On each iteration each cluster is divided in two

$$\boldsymbol{\mu}_i^+ = \boldsymbol{\mu}_i(1+\epsilon)$$

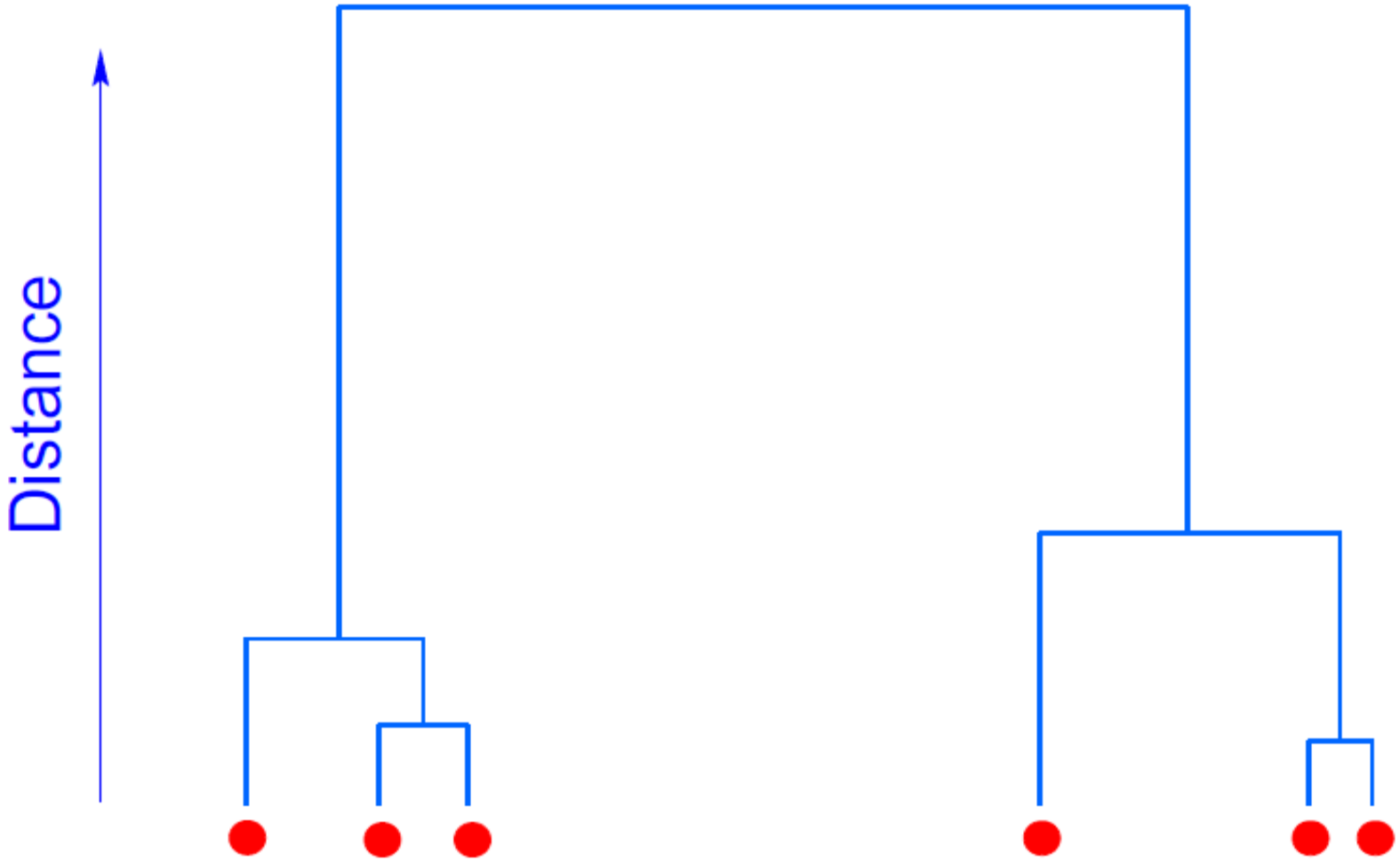
$$\boldsymbol{\mu}_i^- = \boldsymbol{\mu}_i(1-\epsilon)$$

- K-means used to determine cluster centroids
- Also known as LBG (Linde, Buzo, Gray) algorithm
- A more efficient version does K-means only within each binary split, and retains tree for efficient lookup

Agglomerative Clustering

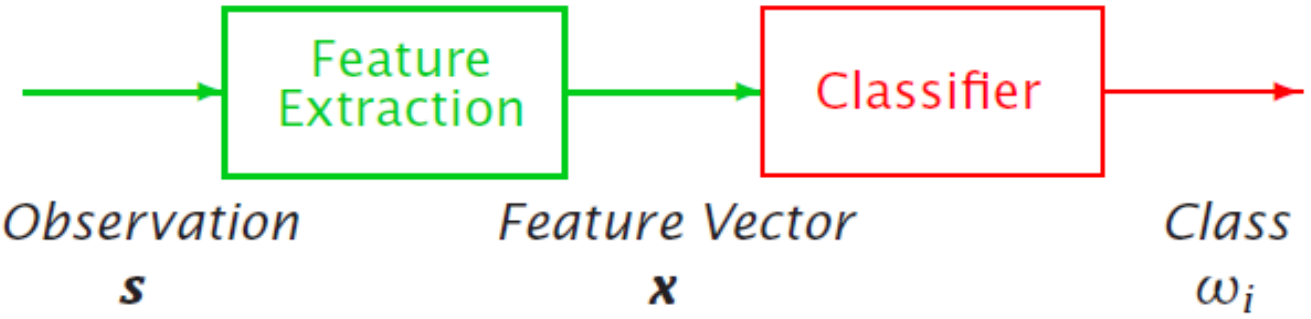
- Structures N samples or seed clusters into a hierarchy
- On each iteration, the two most similar clusters are merged together to form a new cluster
- After $N - 1$ iterations, the hierarchy is complete
- Structure displayed in the form of a dendrogram
- By keeping track of the similarity score when new clusters are created, the dendrogram can often yield insights into the natural grouping of the data

Dendrogram Example: (One dimension)



Pattern Classification

Goal: To classify objects (or patterns) into categories (or classes)



Types of Problems:

1. *Supervised:* Classes are known beforehand, and data samples of each class are available
2. *Unsupervised:* Classes (and/or number of classes) are not known beforehand, and must be inferred from data

Probability Basics

- Discrete probability mass function (PMF): $P(\omega_i)$

$$\sum_i P(\omega_i) = 1$$

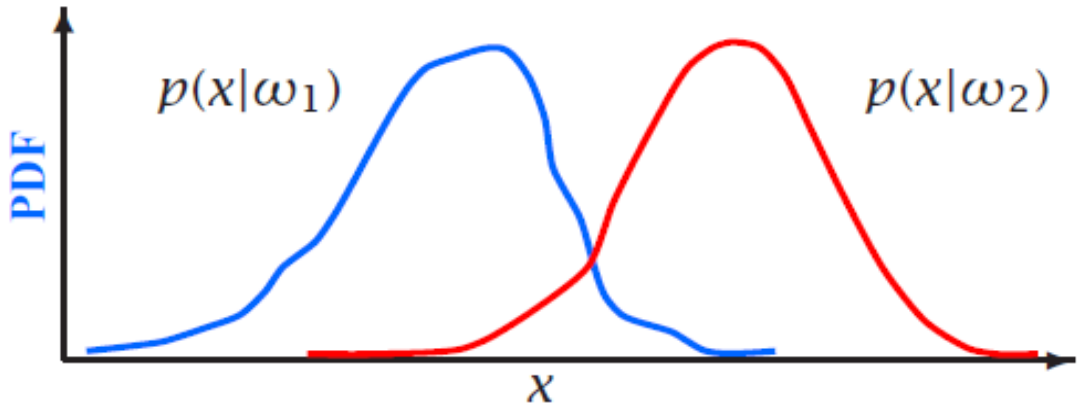
- Continuous probability density function (PDF): $p(x)$

$$\int p(x)dx = 1$$

- Expected value: $E(x)$

$$E(x) = \int xp(x)dx$$

Bayes Theorem



Define:

$\{\omega_i\}$	a set of M mutually exclusive classes
$P(\omega_i)$	a priori probability for class ω_i
$p(\boldsymbol{x}\|\omega_i)$	PDF for feature vector $\boldsymbol{x}$ in class ω_i
$P(\omega_i\|\boldsymbol{x})$	a posteriori probability of ω_i given $\boldsymbol{x}$

From Bayes Rule: $$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})}$$

where $$p(\boldsymbol{x}) = \sum_{i=1}^{M} p(\boldsymbol{x}|\omega_i)P(\omega_i)$$

Bayes decision Theory

- The probability of making an error given $\boldsymbol{x}$ is:

$$P(error|\boldsymbol{x}) = 1 - P(\omega_i|\boldsymbol{x}) \quad \text{if decide class } \omega_i$$

- To minimize $P(error|\boldsymbol{x})$ (and $P(error)$):

$$\text{Choose } \omega_i \text{ if } \quad P(\omega_i|\boldsymbol{x}) > P(\omega_j|\boldsymbol{x}) \quad \forall j \neq i$$

- For a two class problem this decision rule means:

$$\text{Choose } \omega_1 \text{ if } \frac{p(\boldsymbol{x}|\omega_1)P(\omega_1)}{p(\boldsymbol{x})} > \frac{p(\boldsymbol{x}|\omega_2)P(\omega_2)}{p(\boldsymbol{x})}; \text{ else } \omega_2$$

- This rule can be expressed as a likelihood ratio:

$$\text{Choose } \omega_1 \text{ if } \frac{p(\boldsymbol{x}|\omega_1)}{p(\boldsymbol{x}|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)}; \text{ else choose } \omega_2$$

Classification Problem

The problem of classification is to identify the correct class, $\hat{C}$ corresponding to given data, X from a given a set of classes, $\mathcal{C} = \{C_1, C_2, \dots, C_W\}$.

The most obvious way to make a decision in a statistical framework is

$$\hat{C} = \arg\max_{C_i} P(C_i|X)$$

i.e., Select the class with the highest probability given the data.

How do we get $P(C_i|X)$?

Classification Problem

How do we get $P(C_i|X)$?

Bayes Theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

Therefore,

$$\hat{C} = \arg\max_{C_i} P(C_i|X) = \arg\max_{C_i} P(X|C_i)P(C_i)$$

Likelihoods Priors

$P(C_i)$ - Determined by the problem at hand and does not depend on the data being classified

$P(X|C_i)$ - Estimated for the data that is being classified. **What is it?**

Classification Problem

What is $P(X|C_i)$?

It is the probability distribution of data corresponding to class C_i and is usually estimated from lots and lots of data which is known to belong to class C_i. This data is referred to as training data and is required for all $C_i \in \mathcal{C}$.

How do we estimate $P(X|C_i)$?
How do we represent it?

Gaussian Mixture Model (GMM) – One way of doing so.

Discriminant functions

- Alternative formulation of Bayes decision rule
- Define a discriminant function, $g_i(\boldsymbol{x})$, for each class ω_i

$$\text{Choose } \omega_i \text{ if } g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \qquad \forall j \neq i$$

- Functions yielding identical classification results:

$$\begin{aligned} g_i(\boldsymbol{x}) &= P(\omega_i|\boldsymbol{x}) \\ &= p(\boldsymbol{x}|\omega_i)P(\omega_i) \\ &= \log p(\boldsymbol{x}|\omega_i) + \log P(\omega_i) \end{aligned}$$

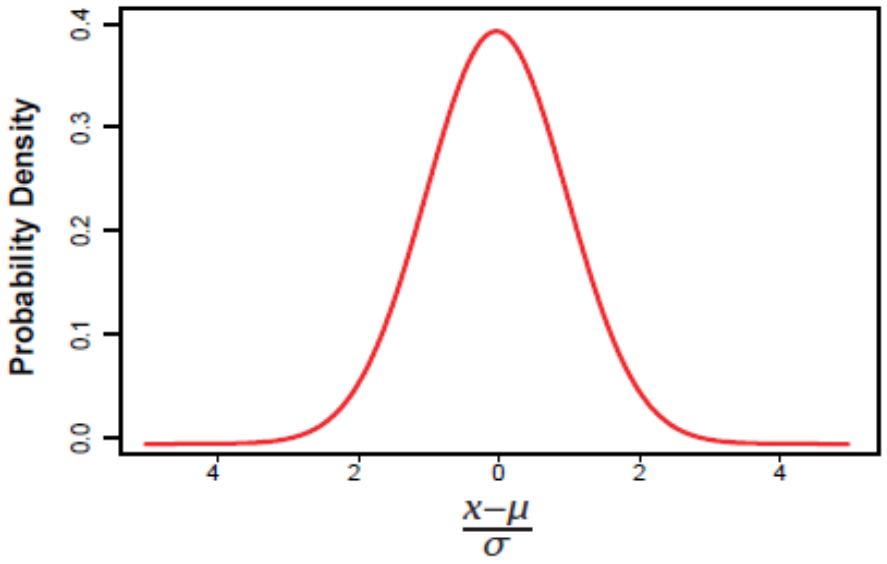
- Choice of function impacts computation costs
- Discriminant functions partition feature space into decision regions, separated by decision boundaries

Parametric Classifiers

- Gaussian distributions
- Maximum likelihood (ML) parameter estimation
- Multivariate Gaussians
- Gaussian classifiers

Gaussian Distribution

- Gaussian PDF's are reasonable when a feature vector can be viewed as perturbation around a reference



- Simple estimation procedures for model parameters
- Classification often reduced to simple distance metrics
- Gaussian distributions also called *Normal*

Gaussian Distribution: One Dimension

- One-dimensional Gaussian PDF's can be expressed as:

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

- The PDF is centered around the mean

$$\mu = E(x) = \int x p(x) dx$$

- The *spread* of the PDF is determined by the variance

$$\sigma^2 = E((x-\mu)^2) = \int (x-\mu)^2 p(x) dx$$

Maximum Likelihood Parameter Estimation

- Maximum likelihood parameter estimation determines an estimate $\hat{\theta}$ for parameter θ by maximizing the likelihood $L(\theta)$ of observing data $\mathcal{X} = \{x_1, \ldots, x_n\}$

$$\hat{\theta} = \arg\max_{\theta} \quad L(\theta)$$

- Assuming independent, identically distributed data

$$L(\theta) = p(\mathcal{X}|\theta) \; = p(x_1, \ldots, x_n|\theta) \; = \prod_{i=1}^{n} p(x_i|\theta)$$

- ML solutions can often be obtained via the derivative

$$\frac{\partial}{\partial \theta} L(\theta) \; = 0$$

- For Gaussian distributions $\log L(\theta)$ is easier to solve

Gaussian ML Estimation: One Dimension

- The maximum likelihood estimate for μ is given by:

$$L(\mu) = \prod_{i=1}^{n} p(x_i|\mu) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

$$\log L(\mu) = -\frac{1}{2\sigma^2}\sum_i (x_i-\mu)^2 - n\log\sqrt{2\pi}\sigma$$

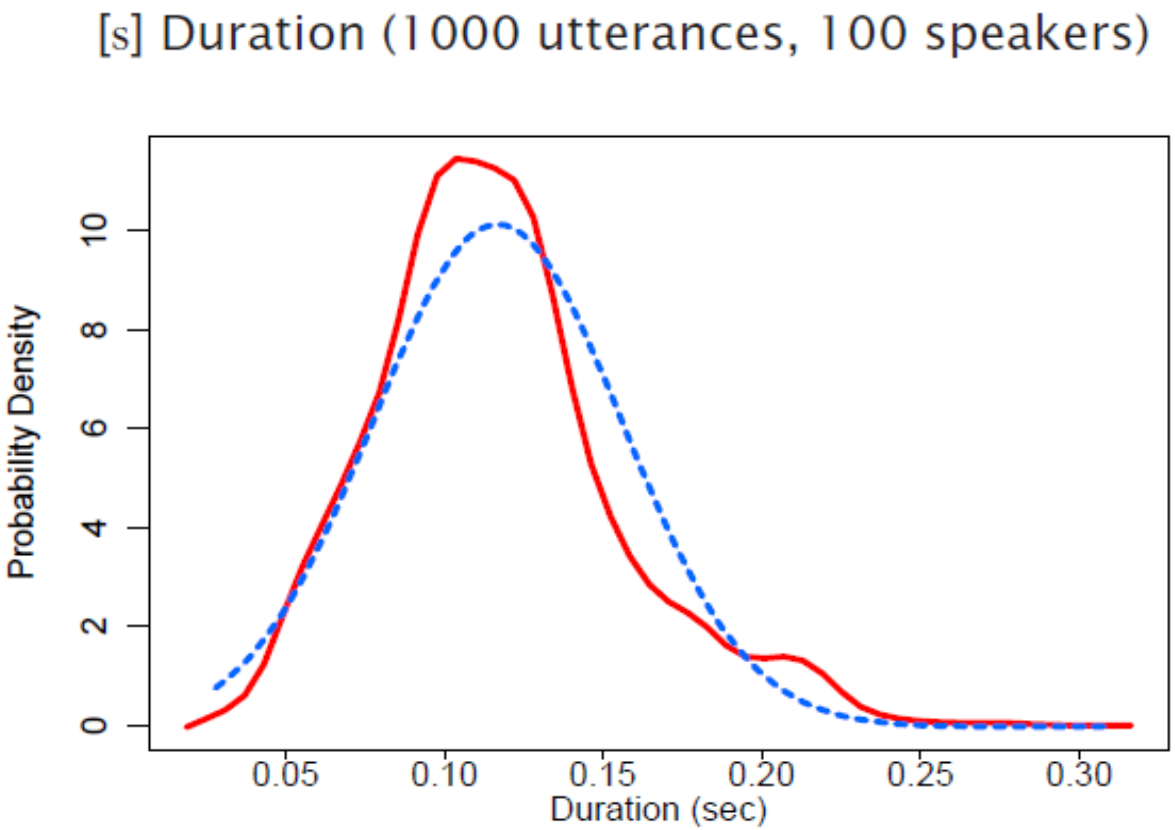
$$\frac{\partial \log L(\mu)}{\partial \mu} = \frac{1}{\sigma^2}\sum_i (x_i-\mu) = 0$$

$$\hat{\mu} = \frac{1}{n}\sum_i x_i$$

- The maximum likelihood estimate for σ is given by:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_i (x_i-\hat{\mu})^2$$

Gaussian ML Estimation: One Dimension



$(\hat{\mu} \approx 120 \text{ ms}, \hat{\sigma} \approx 40 \text{ ms})$

Gaussian Distributions: Multiple Dimension

- A multi-dimensional Gaussian PDF can be expressed as:

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \sim N(\boldsymbol{\mu}, \Sigma)$$

- d is the number of dimensions
- $\boldsymbol{x} = \{x_1, \ldots, x_d\}$ is the input vector
- $\boldsymbol{\mu} = E(\boldsymbol{x}) = \{\mu_1, \ldots, \mu_d\}$ is the mean vector
- $\boldsymbol{\Sigma} = E((\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^t)$ is the covariance matrix with elements σ_{ij}, inverse $\boldsymbol{\Sigma}^{-1}$, and determinant $|\boldsymbol{\Sigma}|$
- $\sigma_{ij} = \sigma_{ji} = E((x_i - \mu_i)(x_j - \mu_j)) = E(x_i x_j) - \mu_i \mu_j$

Gaussian Distributions: Multi-Dimensional Properities

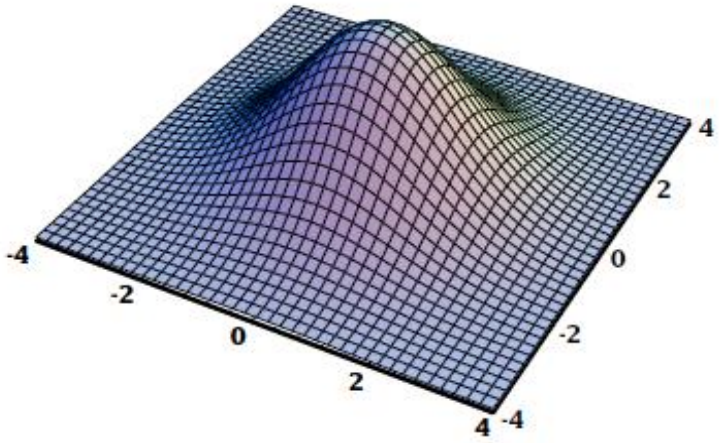
- If the i^{th} and j^{th} dimensions are statistically or linearly independent then $E(x_i x_j) = E(x_i)E(x_j)$ and $\sigma_{ij} = 0$
- If all dimensions are statistically or linearly independent, then $\sigma_{ij} = 0 \quad \forall i \neq j$ and Σ has non-zero elements only on the diagonal
- If the underlying density is Gaussian and Σ is a diagonal matrix, then the dimensions are statistically independent and

$$p(\boldsymbol{x}) = \prod_{i=1}^{d} p(x_i) \qquad p(x_i) \sim N(\mu_i, \sigma_{ii}) \qquad \sigma_{ii} = \sigma_i^2$$

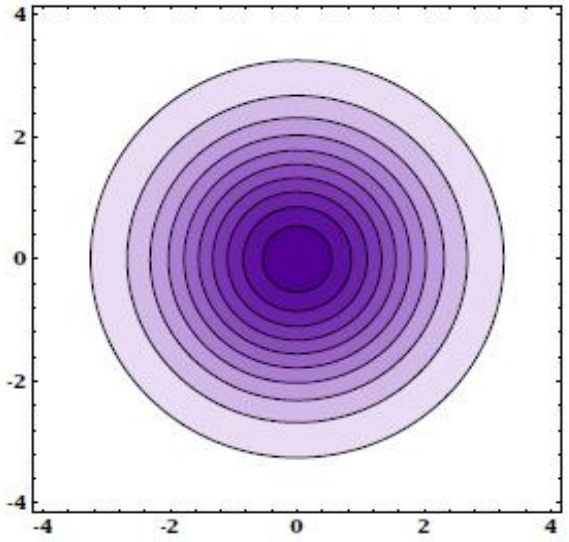
Diagonal Covariance Matrix: $\Sigma = \sigma^2 I$

$$\Sigma = \begin{vmatrix} 2 & 0 \\ 0 & 2 \end{vmatrix}$$

3-Dimensional PDF



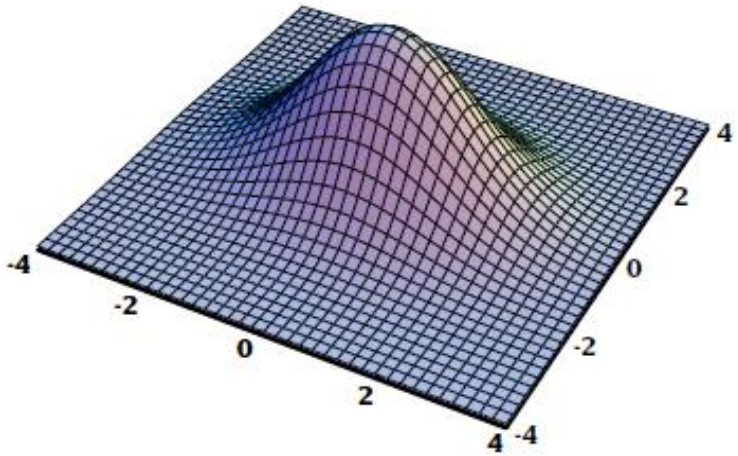
PDF Contour



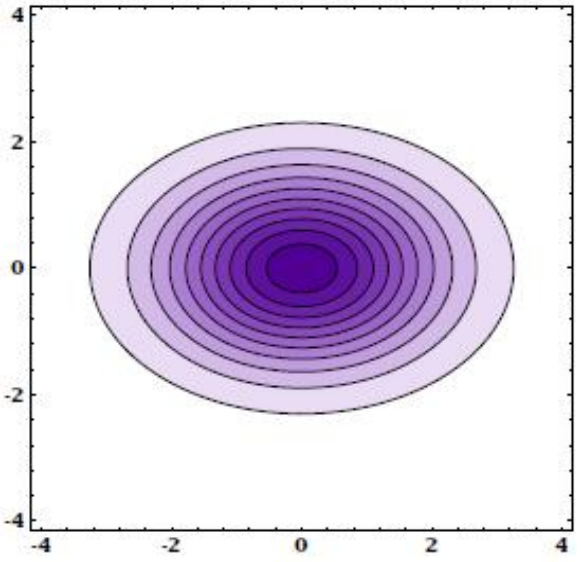
Diagonal Covariance Matrix: $\sigma_{ij} = 0 \quad \forall i \neq j$

$$\Sigma = \begin{vmatrix} 2 & 0 \\ 0 & 1 \end{vmatrix}$$

3-Dimensional PDF



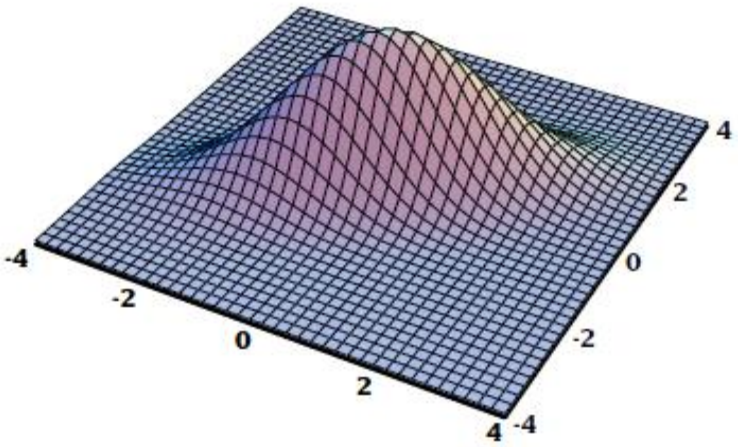
PDF Contour



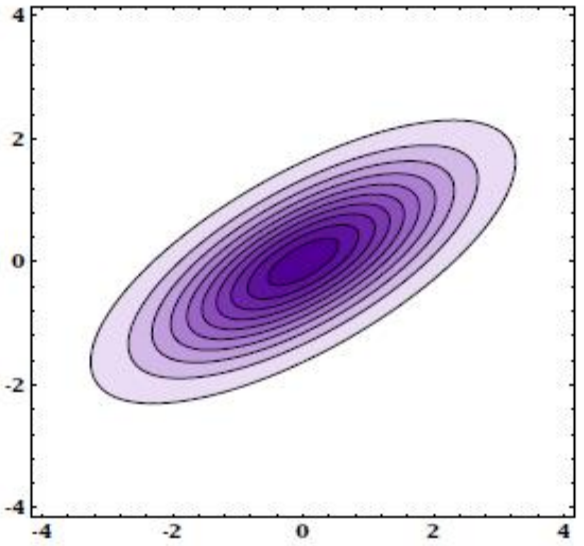
General Covariance Matrix: $\sigma_{ij} \neq 0$

$$\Sigma = \begin{vmatrix} 2 & 1 \\ 1 & 1 \end{vmatrix}$$

3-Dimensional PDF



PDF Contour



Multivariate ML Estimation

- The ML estimates for parameters $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_l\}$ are determined by maximizing the joint likelihood $L(\boldsymbol{\theta})$ of a set of i.i.d. data $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

$$L(\boldsymbol{\theta}) = p(\mathcal{X}|\boldsymbol{\theta}) = p(\boldsymbol{x}_1, \cdot \cdot \cdot, \boldsymbol{x}_n|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(\boldsymbol{x}_i|\boldsymbol{\theta})$$

- To find $\hat{\boldsymbol{\theta}}$ we solve $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \boldsymbol{0}$, or $\nabla_{\boldsymbol{\theta}} \log L(\boldsymbol{\theta}) = \boldsymbol{0}$

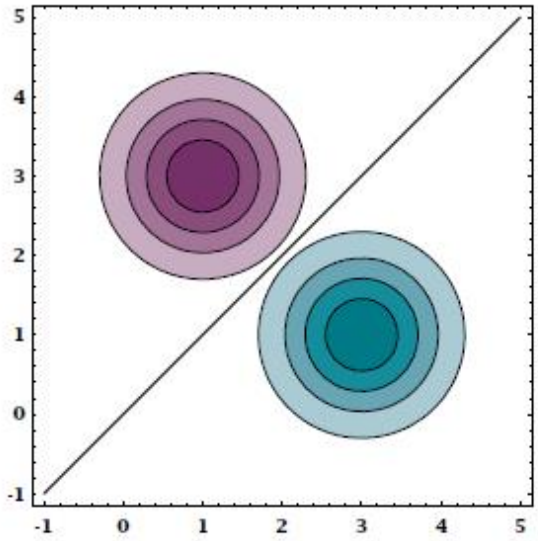
$$\nabla_{\boldsymbol{\theta}} = \{\frac{\partial}{\partial \theta_1}, \cdot \cdot \cdot, \frac{\partial}{\partial \theta_l}\}$$

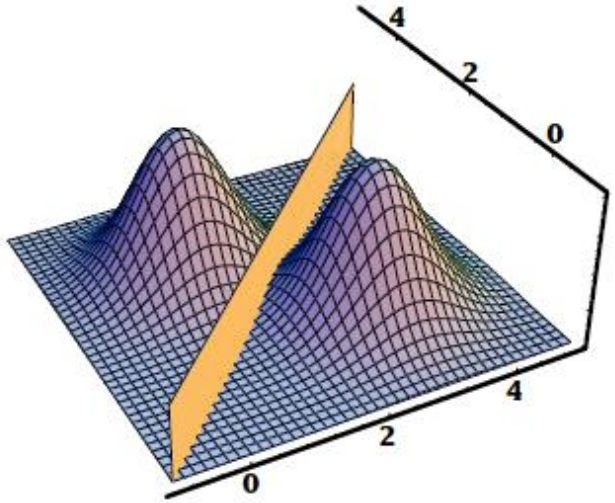
- The ML estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are:

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_i \boldsymbol{x}_i \qquad \hat{\Sigma} = \frac{1}{n} \sum_i (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})^t$$

Multivariate Gaussian Classifier

For distributions with a common covariance structure the decision regions are hyper-planes.





Gaussian Mixture Model (GMM)

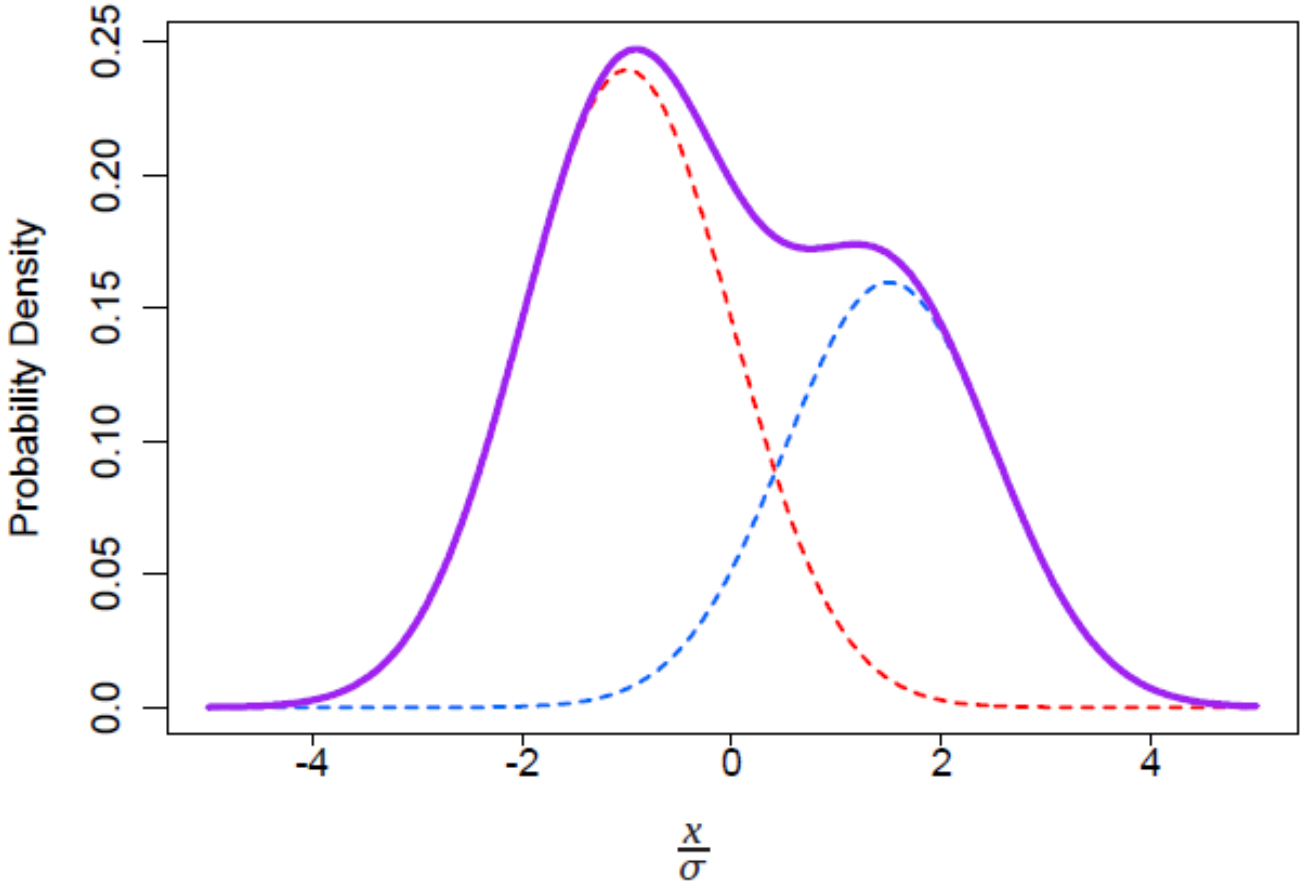
- PDF is composed of a mixture of m component densities $\{\omega_1, \ldots, \omega_m\}$:

$$p(\boldsymbol{x}) = \sum_{j=1}^{m} p(\boldsymbol{x}|\omega_j)P(\omega_j)$$

- Component PDF parameters and mixture weights $P(\omega_j)$ are typically unknown, making parameter estimation a form of unsupervised learning
- Gaussian mixtures assume Normal components:

$$p(\boldsymbol{x}|\omega_k) \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

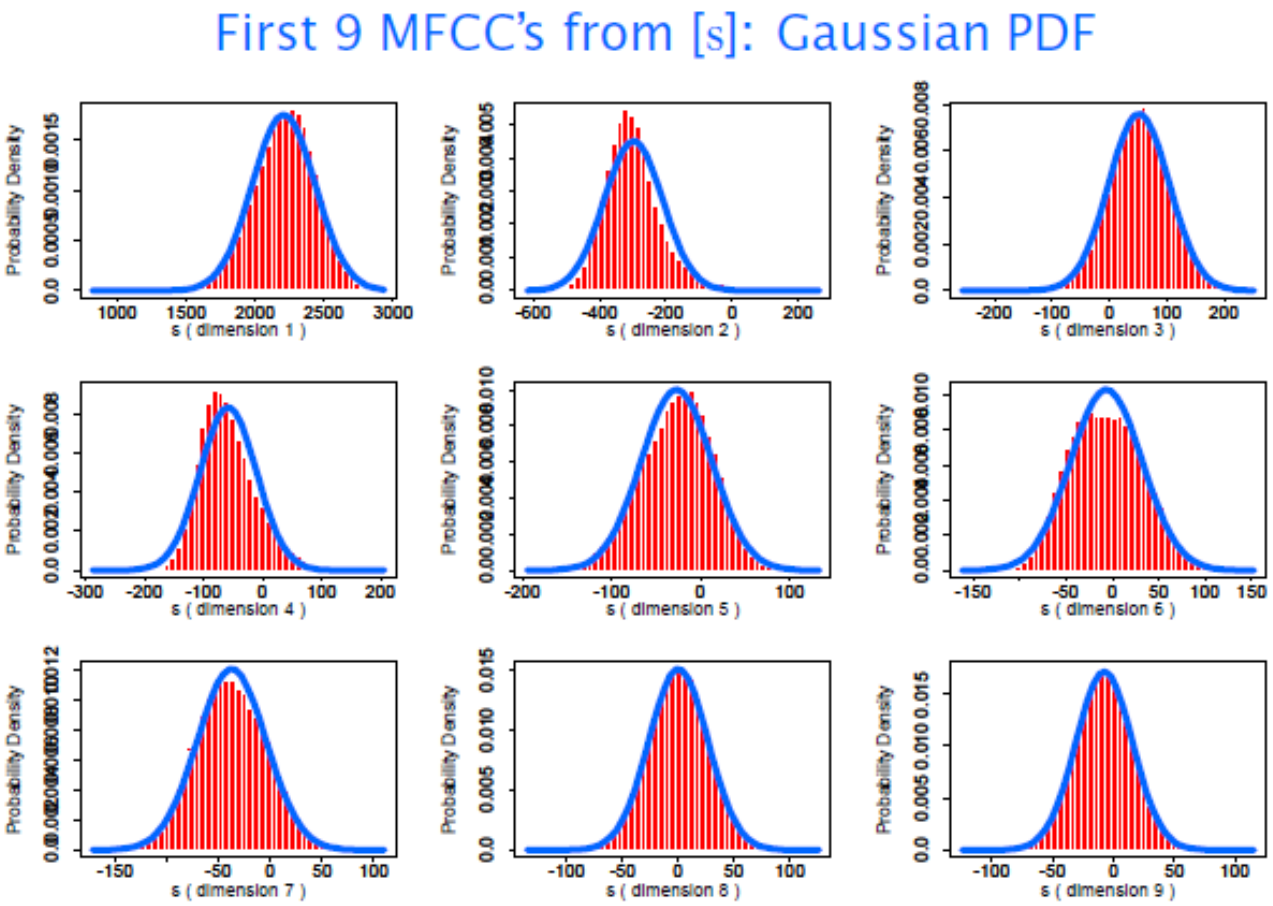
Gaussian Mixture Example: One Dimension



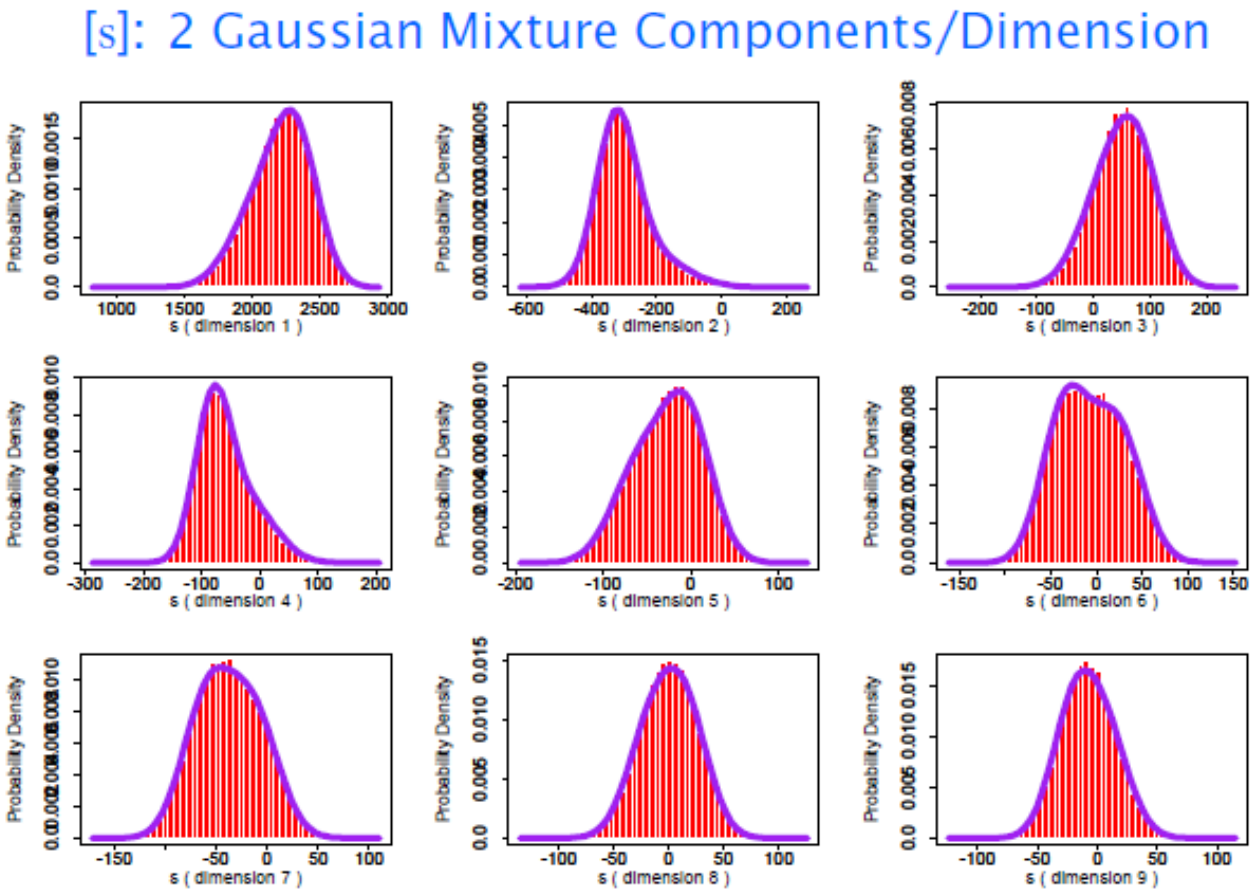
$$p(x) = 0.6p_1(x) + 0.4p_2(x)$$

$$p_1(x) \sim N(-\sigma, \sigma^2) \qquad p_2(x) \sim N(1.5\sigma, \sigma^2)$$

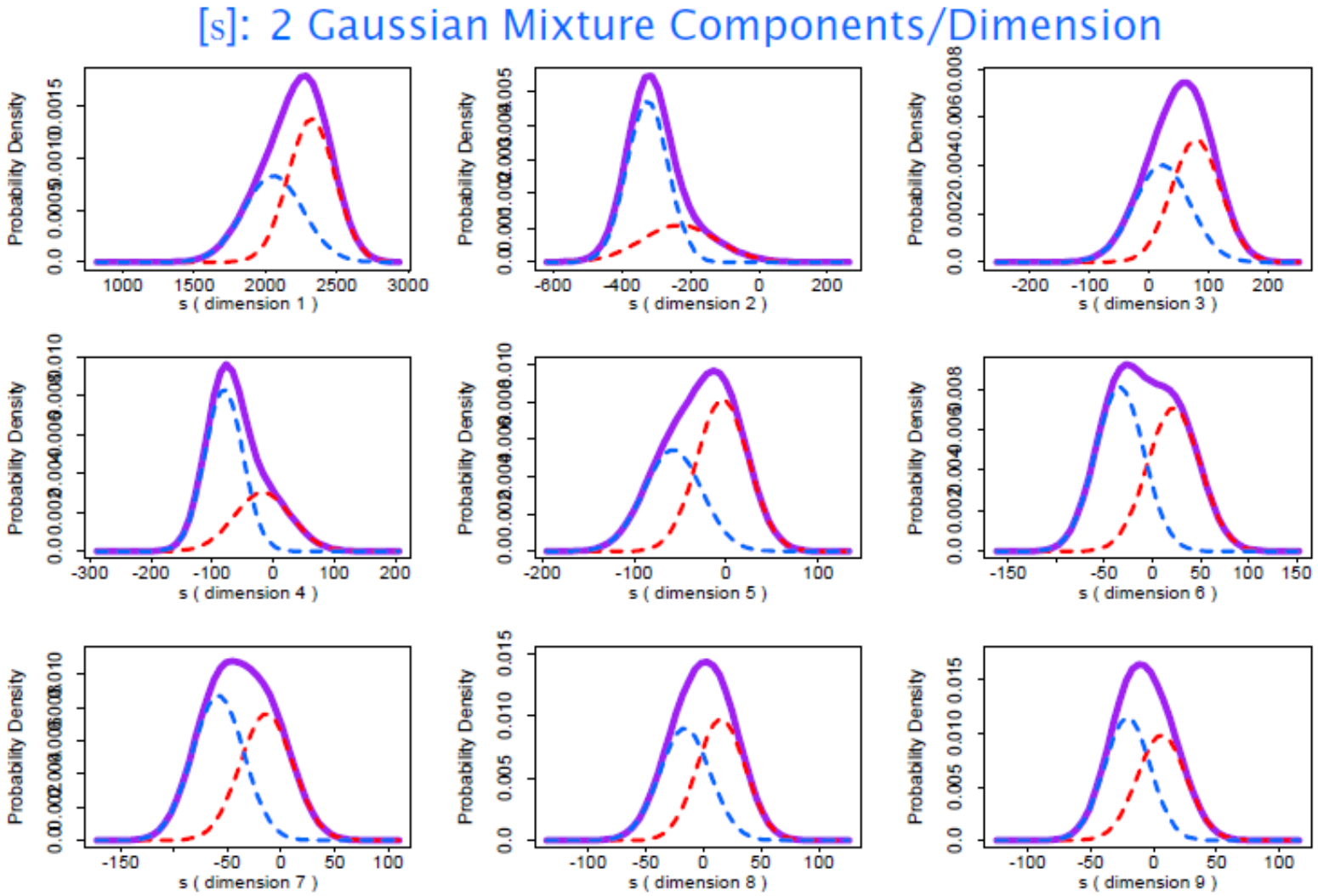
Gaussian Example



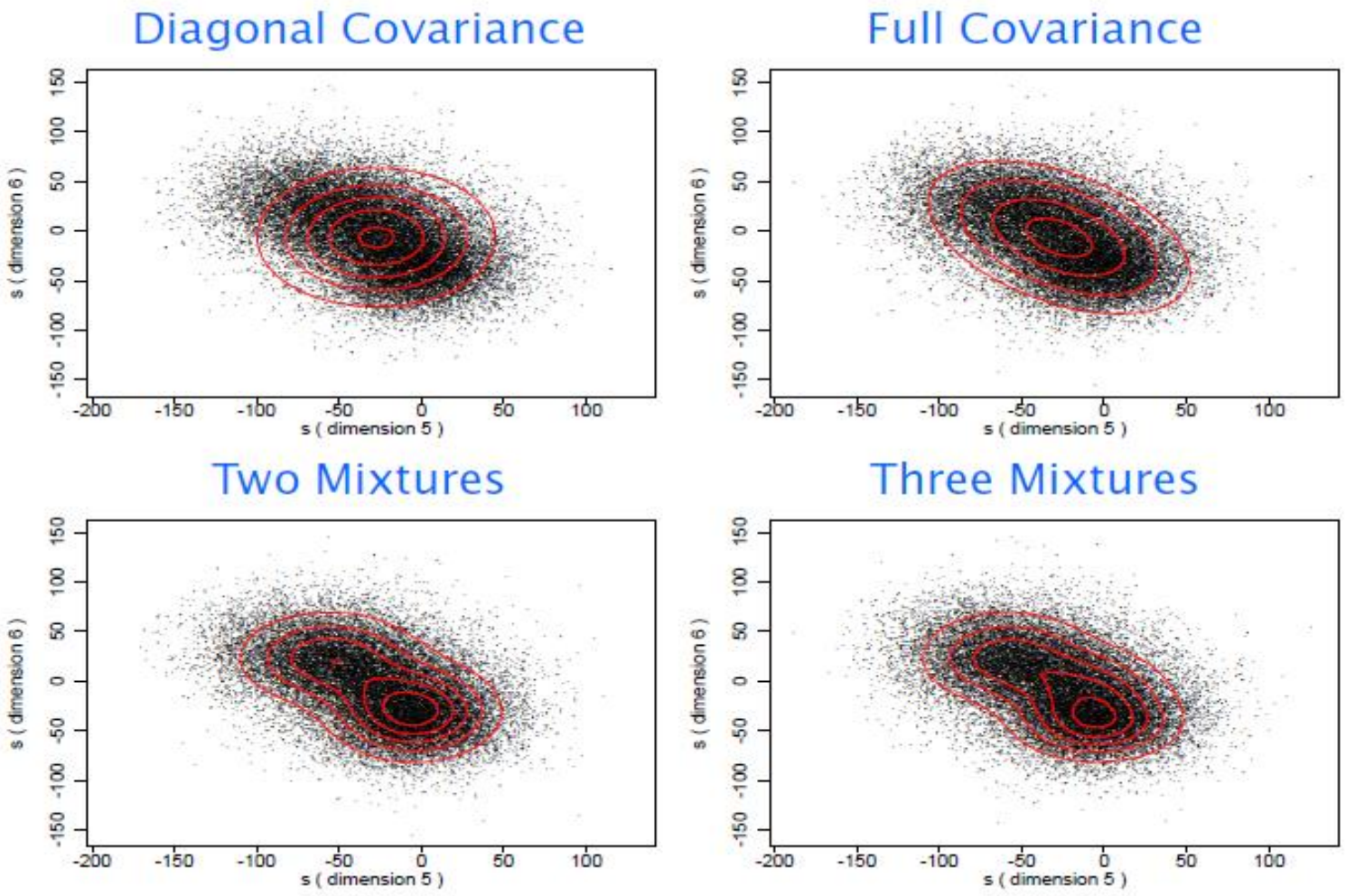
Independent Mixtures



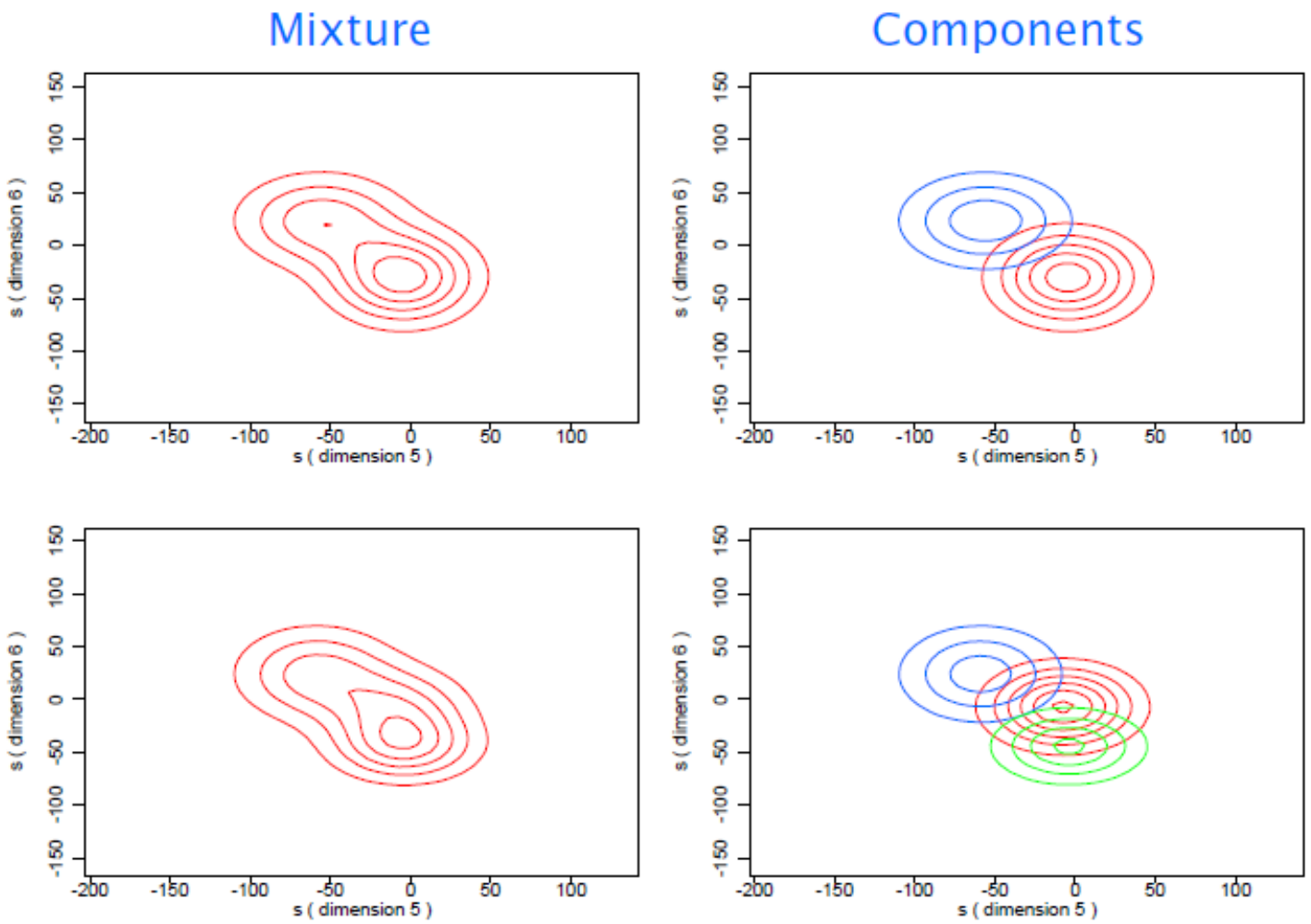
GMM Components



Two Dimensional GMM



Two Dimensional Components



EM Algorithm

- Introduction to the EM algorithm
  - The Expectation Maximization algorithm begins with an initial parameter set
  - Iteratively updates the parameter set such that the likelihood increases with each iteration
  - Converges to a local optimum
    - As with clustering algorithms, convergence to the global optimum can not generally be achieved

EM Algorithm

- Introduction to the EM algorithm
  - Two steps:
  - Expectation step:
    - Evaluate the expectation of the complete data log-likelihood conditional on the training data and the current value of the parameters $\theta^{(n)}$, denoted $Q(\theta, \theta^{(n)})$
  - Maximisation step:
    - Find an updated parameter set $\theta^{(n+1)}$ that maximises $Q(\theta, \theta^{(n)})$

EM Algorithm

- EM algorithm for GMMs – general case
  - E-step:
    - Calculate the probability that training data $\mathbf{x}_i$ belongs to mixture component m

$$P(m \mid x_i, \theta^{(n)}) = E(\mathbf{x}_i \text{ belongs to mixture } m \mid \mathbf{x}_i, \Theta^{(n)})$$

$$= \frac{w_m^{(n)} P(\mathbf{x}_i \mid \boldsymbol{\mu}_m^{(n)}, \mathbf{C}_m^{(n)})}{\sum_{m=1}^{M} w_m^{(n)} P(\mathbf{x}_i \mid \boldsymbol{\mu}_m^{(n)}, \mathbf{C}_m^{(n)})}$$

$$C_m = \Sigma_m$$

$$= \frac{w_m^{(n)} \frac{1}{(2\pi)^{K/2} \left|\mathbf{C}_m^{(n)}\right|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_m^{(n)})^T (\mathbf{C}_m^{(n)})^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_m^{(n)}) \right)}{\sum_{m=1}^{M} w_m^{(n)} \frac{1}{(2\pi)^{K/2} \left|\mathbf{C}_m^{(n)}\right|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_m^{(n)})^T (\mathbf{C}_m^{(n)})^{-1} (\mathbf{x}_i - \boldsymbol{\mu}^{(n)}) \right)}$$

EM Algorithm

- EM algorithm for GMMs – general case
  - M-step:
    - Update of each parameter is weighted by $\omega_{im} = P(G_m \mid x_i, \theta^{(n)})$

$$w_m^{(n+1)} = \frac{1}{N}\sum_{i=1}^{N}\omega_{im}$$

$$\boldsymbol{\mu}_m^{(n+1)} = \frac{\sum_{i=1}^{N}\omega_{im}\mathbf{x}_i}{\sum_{i=1}^{N}\omega_{im}}$$

$$\mathbf{C}_m^{(n+1)} = \frac{\sum_{i=1}^{N}\omega_{im}(\mathbf{x}_i - \boldsymbol{\mu}_m^{(n)})(\mathbf{x}_i - \boldsymbol{\mu}_m^{(n)})^T}{\sum_{i=1}^{N}\omega_{im}}$$

EM Algorithm

- The EM algorithm for GMMs: Notes
  - Clearly an initial set of parameters

    $\{w_m^{(0)}, \mu_m^{(0)}, \sigma_m^{(0)} \mid m = 1,\ldots,M\}$

    need to be provided
    - Means: Typically use K-means or other clustering initialisation techniques
    - Variances: Either set all to 1, or calculate based on variance of training data assigned to each cluster
    - Weights: Start with equal weights, or calculate based on cluster membership of training data

EM Algorithm

- The EM algorithm for GMMs: Notes
  - Stopping criterion
    - Calculate likelihood at each iteration, and stop when relative change is small
    - Fixed number of iterations

Example: 4 Samples, 2 Components

1. Data: $\mathcal{X} = \{x_1, x_2, x_3, x_4\} = \{2, 1, -1, -2\}$
2. Init: $p(x|\omega_1) \sim N(1, 1) \quad p(x|\omega_2) \sim N(-1, 1) \quad P(\omega_i) = 0.5$
3. Estimate:

	x_1	x_2	x_3	x_4
$P(\omega_1\|x)$	0.98	0.88	0.12	0.02
$P(\omega_2\|x)$	0.02	0.12	0.88	0.98

4. Recompute mixture parameters (only shown for ω_1):

$$\hat{P}(\omega_1) = \frac{.98+.88+.12+.02}{4} = 0.5$$

$$\hat{\mu}_1 = \frac{.98(2)+.88(1)+.12(-1)+.02(-2)}{.98+.88+.12+.02} = 1.34$$

$$\hat{\sigma}_1^2 = \frac{.98(2-1.34)^2+.88(1-1.34)^2+.12(-1-1.34)^2+.02(-2-1.34)^2}{.98+.88+.12+.02} = 0.70$$

5. Repeat steps 3,4 until convergence

Notes on Implementation

- Full vs diagonal covariances
  - Diagonal covariances are convenient
    - Avoid the need for matrix inversion
  - Better suited for small training data sets
    - By increasing the number of mixtures, similar performance to full matrix can be achieved
    - Total number of parameters approximately same
- Threshold variances
  - Variances can $\rightarrow 0$
    - Set a minimum value: 'variance flooring'

Some Notes on Modeling

- Underfitting / Overfitting of GMMs
  - Essentially two design choices in modelling using GMMs:
    - Dimension of feature vector (K)
    - Number of mixtures (M)
  - Total number of parameters =
    - $M + M \times K + M \times K^2$ (full covariance matrix)
    - $M + M \times K + M \times K$ (diagonal covariance matrix)
  - Trade off total parameters against amount of training data

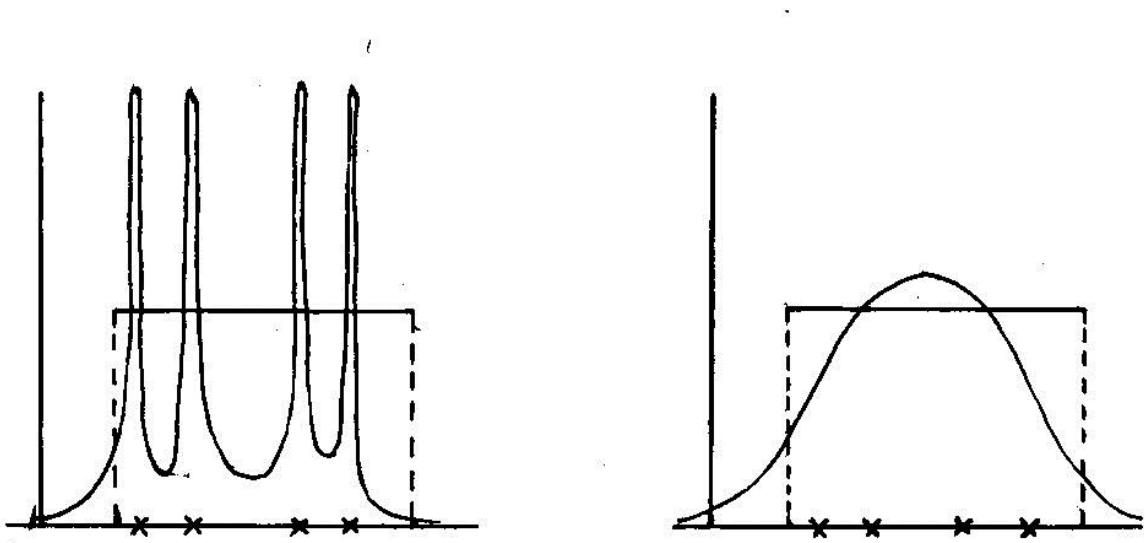
Some Notes on Modeling

- Underfitting / Overfitting of GMMs
  - Huge amounts of training data
    - Small number of parameters may not capture all available information
  - Small amounts of training data
    - Too many parameters cause GMMs to be biased towards training data
    - GMM will not generalise well to other data
    - Known as overfitting
    - One of the fundamental issues of any classifier design

Under Training (Over Fitting)

- A major practical problem in maximum likelihood parameter estimation is **under training**
- Suppose a class *w* gives rise to measurements uniformly distributed over the interval [0,1].
- Unfortunately we don't know this and try to model the distribution using a Gaussian mixture PDF.
- First, we obtain a training set of *S* samples $x_1,\ldots,x_S$
- Suppose $S=4$

Under training (continued)



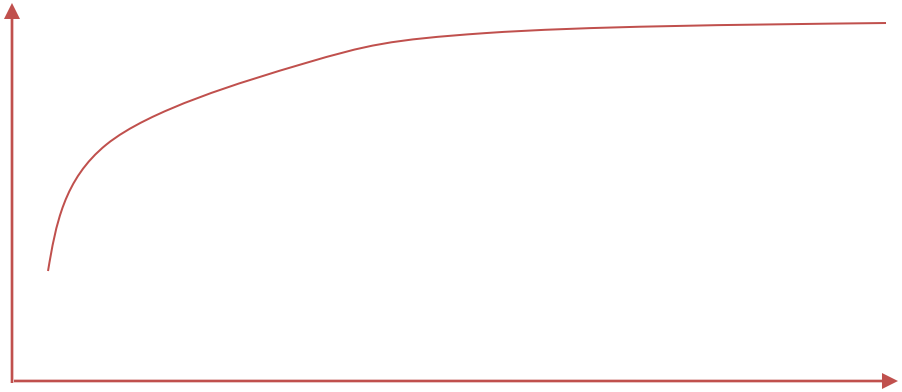
- 4 component PDF gives best fit to training data, but will not generalise to unseen test data
- 1 component PDF performs **worse** on training data, but is a better model!

Under training

- Given a finite training set X, and a ML estimate M of the parameters of a model, $p(X|M)$ will increase, in general, as the number of parameters in M increases
- As number of parameters increases, model begins to characterise detail in the training set which is **not** present in unseen data. The model begins to "remember the training set"
- As number of parameters increases, performance on test data will improve at first, but will then start to degrade as the number of parameters increases and the model focuses on specific detail in the training set

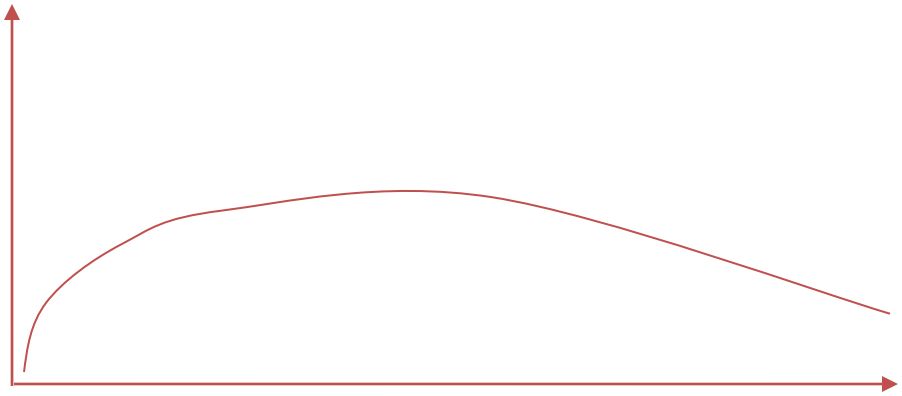
Under training

Performance on **training** data



Number of model parameters

Performance on **test** data



Number of model parameters

Experimental method

- Available data is divided into 3 sets:
  - the **training set**, the **evaluation set** and the **test set**
- For each number of parameters, the ML estimate of the parameters is made using the **training** set
- Classification experiments are run on the **evaluation** set, and the number of parameters which gives best performance is chosen for the final system
- This system is evaluated using the **test** set

GMMs: Summary

- Gaussian Mixture Models (GMMs) are a convenient and flexible approach to defining arbitrary (non Gaussian) PDFs to model data
- To calculate the 'probability' of a single vector y_t we add together the contributions $p_m(y_t)$ from each of the Gaussian components

$$p(y_t) = \sum_{m=1}^{M} w_m p_m(y_t)$$

- To calculate the probability of a sequence $Y=y_1,\ldots,y_T$ we multiply together the probabilities of the individual vectors in the sequence

$$\log p(Y) = \log\left(\prod_{t=1}^{T} p(y_t)\right) = \sum_{t=1}^{T} \log\left(\sum_{m=1}^{M} w_m p_m(y_t)\right)$$

- The parameters of the GMM are estimated automatically from data

Pattern Recognition using GMMs

- A set of **classes** $C_1, \ldots, C_K$, each class modelled by GMM
- A sequence of feature vectors $Y = y_1, \ldots, y_T$
- The sound corresponding to the sequence of feature vectors Y should be assigned to the class C_k which maximises $P(C_k|Y)$
- Bayes theorem

$$P(C_k \mid Y) = \frac{P(Y \mid C_k)P(C_k)}{P(Y)}$$

- $$C^* = \arg\max_{C_k} P(C_k \mid Y) \propto \arg\max_{C_k} P(Y \mid C_k)P(C_k)$$