# TTT4275 Estimation, Detection and Classification Course Summary

**Martin Brandt**

Learning goals:

* Understand the concepts of estimation, detection and classification
* Understand the basic methods within the three topic as a gateway to more advanced techniques.
* Know how to model and solve a variety of real-world problems.
* Understand the importance of data for design.

## CONTENTS

# 1 | ESTIMATION

* *Understand minimum variance unbiased estimators and Cramer-Rao lower bound, linear models and estimators and least squares, maximum likelihood and bayesian estimation.*

## 1.1 | Preliminaries from statistics and linear algebra

**Prior probability:** $P(\omega_i)$

**Posterior probability:** $P(\omega_i|x)$

**Class independent density:** $p(x)$

**Class dependent density:** $p(x|\omega_i)$

**Bayes' Theorem:**

$$P(\omega_i|x) = \frac{P(x|\omega_i)P(\omega_i)}{P(x)} \tag{1}$$

**Mean:**

$$m_x = E\{x\} = \int_{-\infty}^{\infty} xp(x)dx \tag{2}$$

**Covariance matrix:**

$$C_x[m,k] = C_x[k,m] = \gamma_{xx}(m-k) - m_x^2 \tag{3}$$

**Pseudoinverse:**

For a rectangular $m \times n$ matrix $A$ with $B = A^TA$ and $C = AA^T$ the pseudoinverse is defined as:

$$\hat{A}^{-1} = \begin{cases} A^T C^{-1}, & m < n \\ B^{-1}A^T, & m > n \end{cases} \tag{4}$$

**Bias:**

$$b = E\{\hat{\theta}\} - \theta \tag{5}$$

**Variance:**

$$\text{Var}\{\hat{\theta}\} = E\{(\hat{\theta} - E\{\hat{\theta}\})^2\} \tag{6}$$

**Mean square error:**

The optimal estimator has the minimum MSE:

$$\text{MSE} = E\{(\hat{\theta} - \theta)^2\} = \text{Var}\{\hat{\theta}\} + b^2 \tag{7}$$

**Sample mean:**

$$\hat{A}_N = \frac{1}{N}\sum_{n=0}^{N-1} x(n), \qquad E\{\hat{A}_N\} = A, \quad \text{Var}\{\hat{A}_N\} = \frac{\sigma^2}{N} \tag{8}$$

## 1.2 | The Minimum Variance Unbiased Estimator

We choose a suboptimal strategy, by assuming that the estimator is unbiased. This assumption results in the MVU estimator.

## 1.3 | The Cramer-Rao Lower Bound

Assume that $p(x|\theta)$ is given, and that

$$E\{\frac{\partial \log p(x|\theta)}{\partial \theta}\} = 0. \tag{9}$$

Then

$$\text{Var}\{\hat{\theta}\} \leq \frac{-1}{E\{\frac{\partial^2 \log p(x|\theta)}{\partial \theta^2}\}} = \frac{1}{E\{(\frac{\partial \log p(x|\theta)}{\partial \theta})^2\}} \tag{10}$$

is the lower bound of the variance of any MVU estimator $\hat{\theta}$. If equality we have the optimal MVU estimator, called an efficient estimator. Furthermore, the following must be true for an efficient estimator:

$$\frac{\partial \log p(x|\theta)}{\partial \theta} = I(\theta)(g(x) - \theta), \tag{11}$$

where $\hat{\theta} = g(x)$ and $\text{Var}\{\hat{\theta}\} = \frac{1}{I(\theta)}$.

**Fisher information matrix:**

$$I_{ij}(\theta) = -E\{\frac{\partial^2 \log p(x|\Theta)}{\partial \theta_i \partial \theta_j}\} \tag{12}$$

In the CRLB vector case we then get:

$$\text{Var}\{\hat{\theta}_i\} = (I^{-1}(\Theta))_{ij} \tag{13}$$

Furthermore we have

$$\nabla_{\hat{\Theta}} \log p(\mathbf{x}|\Theta) = I(\Theta)(g(\mathbf{x}) - \Theta) \tag{14}$$

for the MVU estimator in the vector case.

## 1.4 | The Linear Model

For linear models the MVU estimator always exists. The linear model in matrix form is: $\mathbf{x} = H\Theta + \mathbf{w}$. The MVU estimator is:

$$\hat{\Theta} = (H^T H)^{-1} H^T \mathbf{x}, \tag{15}$$

which is the pseudoinverse! This MVU estimator is efficient for $p(w) = \mathcal{N}(0, \sigma^2)$, with variance $\text{Var}\{\hat{\theta}\} = (H^T H)^{-1}\sigma^2$. In the vector case we have:

$$\hat{\Theta} = (H^T \Sigma^{-1} H)^{-1} H^T \Sigma^{-1} \mathbf{x}, \tag{16}$$

with covariance $\text{Cov}\{\hat{\Theta}\} = (H^T \Sigma^{-1} H)^{-1}$.

## 1.5 | The Best Linear Unbiased Estimator

Let's assume an estimator that is linear in the observation data:

$$\hat{\Theta} = \sum_{n=0}^{N-1} a_n x(n) = \mathbf{a}^T \mathbf{x} \in \mathbb{R} \tag{17}$$

Furthermore, $E\{x(n)\} = s_n\theta$ and $\text{Cov}\{x\} = C$ is assumed and known. Notice that $p(x|\theta)$ is not known, only the two first moments. Then the BLUE estimator is:

$$\hat{\theta} = \frac{s^T C^{-1} x}{s^T C^{-1} s}, \quad \text{Var}\{\hat{\theta}\} = \frac{1}{s^T C^{-1} s} \tag{18}$$

In the vector parameter case it is easily generalized to:

$$\hat{\Theta} = \left(H^T C^{-1} H\right)^{-1} H^T C^{-1} \mathbf{x}, \quad \text{Var}\{\hat{\Theta}\} = \left(H^T C^{-1} H\right)^{-1} \tag{19}$$

## 1.6 | The Maximum Likelihood Estimator

The likelihood function $L(\theta|x)$ is the density as a function of the parameter, not the measurement. The MLE is then defined as:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} L(\theta|x) \tag{20}$$

The MLE estimator is at least asymptotically efficient. It is usually found by minimizing the log likelihood. It is also invariant, which means that we can transform the parameters and keep the MLE property.

## 1.7 | Bayesian estimators

So far we have assumed that $\theta$ is deterministic. Now we will regard $\theta$ as a stocastic variable with its own prior distribution $p(\theta)$:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{21}$$

We start with the BMSE:

$$\text{BMSE}(\hat{\theta}) = E\{(\theta - \hat{\theta})^2\} = \iint (\theta - \hat{\theta})^2 p(x|\theta) d\theta dx \tag{22}$$

By minimizing the BMSE we get the optimal Bayesian MSE estimator:

$$\hat{\theta} = \int \theta p(\theta|x) d\theta = E\{\theta|x\} \tag{23}$$

So the optimal BMSE estimator is the conditional expectation of the parameter $\theta$ given the measurements $x$.
Now, this integral might be a bitch to evaluate, so a simpler suboptimal strategy is the Maximum-a-posteriori estimator (MAP):

$$\hat{\theta} = \arg\max_{\theta} p(\theta|x) = \arg\max_{\theta} p(x|\theta)p(\theta) \tag{24}$$

Note that MAP and BMSE are identical when the posterior is symmetric.

# 2 | DETECTION

* *Understand statistical decision theory, (Neyman-Pearson, ROC, Bayes risk), detection of deterministic signals and detection of random signals.*

## 2.1 | Hypothesis Testing

The binary hypothesis testing problem is:

$$\begin{aligned} \mathcal{H}_0 : &\quad \theta = \theta_0 \\ \mathcal{H}_1 : &\quad \theta = \theta_1 \end{aligned} \tag{25}$$

where $\theta$ is the parameter value.

Let $\Omega_0$ be the subspace of the observation space $\Omega$ where we decide $\mathcal{H}_0$, and $\Omega_1$ where we decide $\mathcal{H}_1$. $\Omega_1$ is called the critical region, and the decision rule (critical function) $\delta$ is:

$$\delta(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \Omega_1 \\ 0, & \mathbf{x} \in \Omega_0 \end{cases} \tag{26}$$

For a binary hypothesis we have 4 possible cases with the following probabilities:

* Detection: $P(x \in \Omega_1 | \mathcal{H}_1) = P_D$
* Rejection: $P(x \in \Omega_0 | \mathcal{H}_0) = P_R$
* Miss: $P(x \in \Omega_0 | \mathcal{H}_1) = P_M = \int_{\Omega_0} p_1(\mathbf{x}) d\mathbf{x}$
* False alarm: $P(x \in \Omega_1 | \mathcal{H}_0) = P_{FA} = \int_{\Omega_1} p_0(\mathbf{x}) d\mathbf{x}$

Notice that $P_D + P_M = 1$ and $P_R + P_{FA} = 1$.

The probability of detection, often called power, is defined by:

$$\beta = P_D = 1 - P_M = \int_{\Omega_1} p_1(\mathbf{x}) d\mathbf{x} \tag{27}$$

## 2.2 | Likelihood Ratio Test

Let the test statistic for the decision rule be:

$$L(\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{H}_1)}{p(\mathbf{x}|\mathcal{H}_0)} \lessgtr \lambda \tag{28}$$

Usually we have parametric models such that $p(x|\mathcal{H}_i) = p(x|\theta_i)$. Often $\theta_i$ has to be estimated (generalized LRT). Also, it is usually more convenient to consider the log-likelihood ratio.

## 2.3 | Bayesian Detection

We define the cost $C_{ij}$ for choosing $\Omega_i$ given $\mathcal{H}_j$ (so $C_{01}$ and $C_{10}$ are the error costs). Furthermore we assume that the priors $P(H_i) = P_i = \Pi_i$ are known (usually $P_1 << P_0$).
The objective of the Bayes criterion is to minimize the **Bayes risk** i.e. the expected value of the cost:

$$\mathcal{R} = E\{C\} = \mathcal{R} = \sum_{i=0}^{1}\sum_{j=0}^{1} C_{ij} P\left(\mathcal{H}_i|\mathcal{H}_j\right) P\left(\mathcal{H}_j\right) \tag{29}$$

The optimal LRT threshold that minimizes the Bayesian risk decides $\mathcal{H}_1$ if:

$$L(\mathbf{x}) = \frac{p_1(\mathbf{x})}{p_0(\mathbf{x})} > \frac{\Pi_0}{\Pi_1}\frac{C_{10} - C_{00}}{C_{01} - C_{11}} = \lambda \tag{30}$$

We usually let $C_{00} = C_{11} = 0$. In addition for $C_{01} = C_{10}$ we simply get $L(x) = \frac{\Pi_0}{\Pi_1}$, which is the **minimum probability of error (MPE)**.
By manipulating this expression we get the **MAP detector**:

$$L_{\text{MAP}} = \frac{P(\mathcal{H}_1|\mathbf{x})}{P(\mathcal{H}_0|\mathbf{x})} \lessgtr \lambda \tag{31}$$

By further assuming that $\Pi_0 = \Pi_1$ we get that $\lambda = 1$, which is the **maximum-likelihood (ML) detector**.

## 2.4 | Neyman-Pearson Detection

In many cases like signal detection the prior is not known, which was assumed in the Bayes risk detector. We will then base our LRT test statistic on $P_{FA}$. We want to choose $\Omega_1$ such that the compromise between a low $P_{FA}$ and high $P_D$ is good. This is done by maximizing $P_D$ subject to a maximum allowed $P_{FA}$.

**Neyman-Pearson Lemma:** *Let $\delta_{\lambda^*}$ be the critical function of a LRT with threshold $\lambda^*$. Let $\alpha^*$ and $\beta^*$ be the false-alarm rate and power respectively. Let $\delta_\lambda$ be another arbitrary LRT with $\lambda, \alpha, \beta$. If $\alpha \leq \alpha^*$, then $\beta \leq \beta^*$.*

This lemma illustrates that if we want to increase the power of the LRT, we must also accept an increased false-alarm rate.
To maximize the power given an upper bound $\alpha_0$ on the false-alarm rate, it can be shown that the following detector is optimal:

$$\lambda_{NP} = P_{FA}^{-1}(\alpha_0) \tag{32}$$

The receiver-operation-curve (ROC) plot $P_D$ as a function of $P_{FA}$ and help to visualize the trade-offs we do when designing a NP detector. All ROCs are above the $\alpha = \beta$ line and are concave downward.

## 2.5 | Detection of Signals in Gaussian Noise

### 2.5.1 | Constant in Gaussian noise

Consider the detection of a constant signal in Gaussian noise:

$$\begin{aligned} \mathcal{H}_0: &\quad \mathbf{x} = \mathbf{w} \\ \mathcal{H}_1: &\quad \mathbf{x} = \mathbf{A} + \mathbf{w} \end{aligned} \tag{33}$$

By deriving the log-likelihood ratio we get that:

$$\mathrm{LL}(\mathbf{x}) = \sum_{n=0}^{N-1} \frac{Ax(n)}{\sigma^2} - \frac{NA^2}{2\sigma^2} \lessgtr \log\lambda \tag{34}$$

By recasting the problem to a single variable $z$ we get

$$z = T(\mathbf{x}) = \bar{\mathbf{x}} \lessgtr \frac{\sigma^2}{AN}\log\lambda + \frac{A}{2} \tag{35}$$

### 2.5.2 | Signal in Gaussian Noise

Now consider a general deterministic signal in Gaussian noise:

$$\begin{aligned} \mathcal{H}_0: &\quad \mathbf{x} = \mathbf{w} \\ \mathcal{H}_1: &\quad \mathbf{x} = \mathbf{s} + \mathbf{w} \end{aligned} \tag{36}$$

$$\mathrm{LL}(\mathbf{x}) = \sum_{n=0}^{N-1} \frac{s(n)x(n)}{\sigma^2} - \sum_{n=0}^{N-1} \frac{s(n)^2}{2\sigma^2} \lessgtr \log\lambda \tag{37}$$

After rearranging we find that we decide $\mathcal{H}_1$ if

$$z = T(\mathbf{x}) = 2\sum_{n=0}^{N-1} x(n)s(n) > 2\sigma^2\log\lambda + \sum_{n=0}^{N-1} s^2(n) \tag{38}$$

where $T(x)$ is a correlator and we find that $\mathrm{Var}\{z\} = \sigma^2 E_s$.

### 2.5.3 | Signal in Colored Noise

Let $w$ be colored with covariance elements $C_m n = E\{w(m)w(n)\} = \gamma_{ww}(m-n)$. The hypothesis is the same as above. We find that we will decide $\mathcal{H}_1$ when

$$z = T(\mathbf{x}) = 2\mathbf{x}^T C^{-1}\mathbf{s} > 2\log\lambda + \mathbf{s}^T C^{-1}\mathbf{s} \tag{39}$$

which is referred to as the generalized matched filter (GMF).

### 2.5.4 | Multiple Signals in Gaussian Noise

$$\begin{aligned} \mathcal{H}_0: &\quad \mathbf{x} = \mathbf{s}_0 + \mathbf{w} \\ \mathcal{H}_1: &\quad \mathbf{x} = \mathbf{s}_1 + \mathbf{w} \end{aligned} \tag{40}$$

We find that we decide $\mathcal{H}_1$ when

$$z = T(\mathbf{x}) = 2\sum_{n=0}^{N-1} x(n)(s_1(n) - s_0(n)) > 2\sigma^2 \log\lambda + \sum_{n=0}^{N-1} s_1^2(n) - \sum_{n=0}^{N-1} s_0^2(n) \tag{41}$$

### 2.5.5 | Random Signal in Gaussian Noise

Now we assume that the signal is random. We find that

$$\mathrm{LL}(\mathbf{x}) = -\frac{N}{2}\log\frac{\sigma_x^2}{\sigma^2} - \sum_{n=0}^{N-1}\left(\frac{x(n) - 2Ax(n) + A^2}{2\sigma_x^2} + \frac{x(n)^2}{2\sigma^2}\right) \tag{42}$$

where $\sigma_x^2 = \sigma^2 + \sigma_s^2$. Let $\bar{x}_{sm}$ be the sample mean and $\bar{x}_{sp}$ be the sample power. We then decide $\mathcal{H}_1$ when

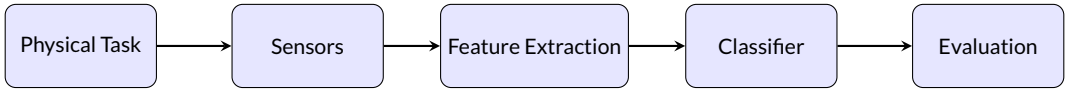
$$z = T(\mathbf{x}) = \sigma_s^2\bar{x}_{sp} + 2A\sigma^2\bar{x}_{sm} > \sigma_x^2\sigma_s^2(\log\frac{\sigma_x^2}{\sigma_s^2} + \frac{2}{N}\log\lambda) \tag{43}$$

# 3 | CLASSIFICATION

* *Understand the theoretical optimal classifier, three basic classifier types, estimation and clustering in classifier design, evaluation of performance and short on state-of-the-art including machine learning.*

## 3.1 | Classification Basics

Classification is the process of mapping a measured signal into one of $C$ classes. We can either do supervised or unsupervised classification. When supervised we only do recognition i.e. the classes are known. When unsupervised we also cluster the data, i.e. we need to find the classes themselves as well.
Consider a general classification system:




A sensor measures a signal from a physical source. The feature we want to use for classification is extracted and the classification is executed. An integrated part of the design should also be a performance estimator for the classifier.

## 3.2 | The Theoretical Optimal Classifier

**Bayes decision rule:** $x \in \omega_k \iff P(\omega_k|x) = \max_i P(\omega_i|x)$
The BDR is optimal with respect to minimum error rate. But these distributions are not known, therefore we have two strategies - approximate them or disregard the BDR and try something else.
If we are to estimate it, a simple Gaussian is too coarse. So we use a weighted sum of Gaussians, which is called the **Gaussian mixed model (GMM):**

$$p(x|\omega_i) = \sum_{k=1}^{K} c_{ik}\, \mathcal{N}(\mu_{ik}, \Sigma_{ik}) \tag{44}$$

The parameters must be estimated from training.

## 3.3 | Training

When training, we will need a **labelled training set** for parameter estimation. It needs to be sufficiently large and representative for the classifier to be able to generalize. For a training set of size $N$ we have the input data $X_N = \{x_1, \ldots, x_N\}$ and the corresponding class labels $T_N = \{t_1, \ldots, t_N\}$. Furthermore a **validation set** is needed for evaluating the performance of the estimator while it is training, and a **test set** is needed to test the true performance. Note that the performance from the test set is only an estimate of the true performance, which can only be found for an infinitely large set! When evaluating classifiers we find the estimated error rate of the classification, denoting $\text{EER}_D$ and $\text{EER}_T$ for the error rate of the training set and testing set respectively. The MER is the theoretical minimum error rate.
One strategy when training is to assume that if the difference between $\text{EER}_D$ and $\text{EER}_T$ is sufficiently small, so is the difference between $\text{EER}_T$ and MER. We could also try to find a confidence interval (usually 95%) for a more formal performance evaluation. The confidence interval is also useful when comparing classifiers. If the difference between $\text{EER1}_T$ and $\text{EER2}_T$ is larger than the confidence interval, we know that the classifier is better with 95% certainty.

**Leave-one-out strategy:** for each sample $i$, train with all other samples and use sample $i$ for testing. Use the mean as $\text{EER}_T$. Useful if data is limited, but downside is that classifier changes slightly each iteration.
**Confusion matrix:** often we want information about each class, not just the overall EER. We then use the confusion matrix, where the elements are defined as $A_{ij}$ = #classifier claims that $x \in \omega_i$ when $x \in \omega_j$. This lets us analyze which classes the classifier confuses the most.

## 3.4 | Maximum Likelihood Training

A common approach is **maximum likelihood training:**

$$LL(\theta) = \log p(x|\theta) = \sum_{k=1}^{N} \log p(x_k|\theta) \tag{45}$$

## 3.5 | The Plug-in MAP classifier

If we have knowledge of $\theta$, i.e. $p(\theta)$, we can use the **Plug-in MAP classifier**. We assume a GMM distribution of $p(x|\omega_i)$. We try to train the Plug-in MAP with ML training. For a single class $\omega_i$ we want to estimate the parameters $\Lambda_i$. The likelihood is:

$$LL(X_{N_i}, \Lambda_i) = \prod_{k=1}^{N_i} p(x_{ik}|\Lambda_i) \tag{46}$$

We maximize the likelihood:

$$\nabla_{\Lambda_i} LL(X_{N_i}, \Lambda_i) = \sum_{k=1}^{N_i} \nabla_{\Lambda_i} \log(p(x_{ik}|\Lambda_i)) = 0 \tag{47}$$

If we assume a single Gaussian we find that the estimators for the mean and covariance is simply the sample mean and sample covariance:

$$\hat{\mu}_i = \frac{1}{N_i} \sum_{k=1}^{N_i} x_{ik} \tag{48a}$$

$$\hat{\Sigma}_i = \frac{1}{N_i} \sum_{k=1}^{N_i} (x_{ik} - \hat{\mu}_i)(x_{ik} - \hat{\mu}_i)^T \tag{48b}$$

In the GMM case we will in general get a dirty ass equation on the form $\Lambda_i = f(\Lambda_i)$ which is difficult to solve, but an effective suboptimal iterative algorithm for solving it is **expectation maximation (EM)**:

$$\Lambda_i(m) = f(\Lambda_i(m-1)) \iff LL(m) > LL(m-1) \tag{49}$$

## 3.6 | The Linear Discriminant Classifier

For approximately linearly separable problems a linear classifier is adequate. The decision borders are hyper-planes, which makes for simple, generalizable classifiers. Each class is described by a discriminant function $g_i(x)$ and the decision rule is given by:

$$x \in \omega_j \iff g_j(x) = \max_i g_i(x) \tag{50}$$

As this point they completely butcher the notation, but I will try to skip some steps to make it more readable. The discriminant function for class *i* is for now:

$$z_i(\mathbf{x}_k) = \begin{bmatrix} \mathbf{w} & w_0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix} = W_i^T \mathbf{x} \tag{51}$$

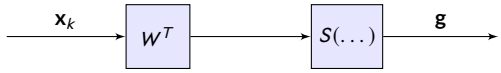
where **w** are the weights to each sample data and $w_0$ is the offset. The vector discriminant function is then:

$$z(\mathbf{x}_k) = \begin{bmatrix} W_1^T & W_2^T & \dots & W_C^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix} = W^T \mathbf{x} \tag{52}$$

Now the final step is to map the discriminant functions to the $[0, 1]$ domain. This is done with the squishy boi aka the sigmoid function:

$$g_{ik}(z_{ik}) = \frac{1}{1 + e^{-z_{ik}}} \tag{53}$$

This creates the basic building block of machine learning, the **perceptron**:




The linear classifier is not like the Plug-in MAP in that it is statistically based, so we can not use ML training.
Good old MSE is used instead:

$$\text{MSE} = \frac{1}{2} \sum_{k=1}^{N} (g_k - t_k)^T (g_k - t_k) \tag{54}$$

Notice that this means we train the entire $W$ i.e. all the classes at once! The gradient is:

$$\nabla_W \text{MSE} = \sum_{k=1}^{N} \nabla_{g_k} \text{MSE} \nabla_{z_k} g_k \nabla_W z_k = \sum_{k=1}^{N} \left[ (g_k - t_k) \circ g_k \circ (1 - g_k) \right] x_k^T \tag{55}$$

With the line search for $W$ being:

$$W(m) = W(m-1) - \alpha \nabla_W \text{MSE} \tag{56}$$

The step factor $\alpha$ must be tuned appropriately.

## 3.7 | The Template Based Classifier and Clustering

### 3.7.1 | The Template Based Classifier

For this type of classifier the decision rule is quite simple, we simply compare the input with a set of references by some distance metric and choose the closest reference. Different distance functions, decision rules and ways to find references produce different results.

**The Mahalanobis distance:**

$$d(x, x_{ik}) = (x - \mu_{ik})^T \Sigma_{ik}^{-1} (x - \mu_{ik}) \tag{57}$$

The Mahalanobis distance can be simplified by just using class specific covariances $\Sigma_i$, or just one covariance $\Sigma$, or even simpler by assuming that it is diagonal. For the simplest case with $\Sigma = I$ we get the euclidean distance.

**Nearest neighbour (NN):** this is the simplest decision rule, just choosing the closest reference, i.e.

$$x \in \omega_j \iff j = \arg\min_i(\min_k d(x, x_{ik})) \tag{58}$$

What this says is just choose the template which has the lowest distance in the vector of the closest distances for each template.

This can be extended to a **K-nearest neighbours (KNN)** search instead, in which one finds the K nearest distances, and a majority vote decides. In case of a draw the closest wins.

### 3.7.2 | Clustering

So the big question is: how do we find the references? The easy solution is to not train at all, and just use the entire training set as references. The issue here is the amount of processing needed to classify new samples. Another kind of shitty idea is to randomly draw references from the total training set.

This brings us to **clustering**, which involves estimating how many $M$ references we should use, what those references are and their covariance matrices.

The iterative clustering procedure is as follows:

1. Start with $M = 1$.
2. Find $\Lambda_1$ and $D_1 = \sum_{k=1}^{N} d(x_k, \Lambda_1)$.
3. Increment M, split $\Lambda_1$.
4. Set $q = 1$, $D_{M_q} = \infty$.
5. Iterate $q$ to find the best references for the cluster size $M$: classify data to closest clusters, find $D_{M_q}$, continue and update $\Lambda_1$ if $D_{M_q}$ is decreasing.
6. When previous step has terminated, if $D_M$ is smaller than $D_{M-1}$ increment $M$ and split references. Otherwise you are done.

The splitting process is done by finding the appropriate cluster to split (either by finding the cluster with the most vectors classified as it or the cluster with the largest covariance matrix norm) and then creating a copy of the cluster and modifying the mean by some weight $w$.

This can be generalized to a statistical framework by estimating the parameters of a GMM-based Plug-in MAP classifier. We introduce a soft decision using the posteriori probabilities $P(C_i|x_k) = \gamma_{ik}$:

$$\begin{aligned}\hat{\mu}_i &= \frac{\sum_{k=1}^{N} \gamma_{ik} x_k}{\sum_{k=1}^{N} \gamma_{ik}} \\ \hat{\Sigma}_i &= \frac{\sum_{k=1}^{N} \gamma_{ik} (x_k - \mu_i)(x_k - \mu_i)^T}{\sum_{k=1}^{N} \gamma_{ik}}\end{aligned} \tag{59a}$$

This is referred as the **Expectation-Maximation (EM)** algorithm.

## 3.8 | State-of-the-art Classifiers

Today nonlinear and context-based classifiers are the two types of classifiers that are mostly used. In the former neural networks dominate. They perform well, especially when no model is known, but you need insane amounts of data. Context-based classifiers e.g. the HMM classifier are used in problems where most of the class information is temporal.