TOR A. MYRVOLL, STEFAN WERNER AND MAGNE H. JOHNSEN

ESTIMATION, DETECTION AND CLASSIFICATION THEORY

1
Estimation Theory

Introduction

This handout aims to give a short and simple overview of what every young electrical engineer should know about the theory, practice and art of estimation. Mathematical rigor will be eschewed whenever possible, and the focus will instead be on learning the most important properties of practical parameter estimation methods. The celebrated practice of revisiting the same toy problems again and again will still be a valuable in understanding key aspects of the theory, but more realistic examples are also provided to provide motivation and hands on experience.

A common task encountered in engineering is the estimation of some parameter of interest based on a set of noisy measurements. A simple example follows:

Example 1. DC voltage in noise *We want to measure the DC voltage between two points in an electric circuit. Our measurement process is subject to noise, so the actual measurement x that we get is*

$$x = A + w$$

where A is DC voltage, the parameter we are interested in, and w is Gaussian noise with zero mean and variance σ^2. Depending on the realization of the noise, the measurement x may be close or not to the parameter of interest, A. We say that $\hat{A} = x$, is an estimator of A.

Given a set of measurements, $\{x[0], x[1], \ldots, x[N-1]\}$ – how can we use these extra measurements to get a better estimate compared to just using one measurement? We propose a new estimator

$$\hat{A} = \frac{1}{N} \sum_{n=0}^{N-1} x[n].$$

There are several question we can ask in relation to these two estimators. Is the second estimator really better than the first? How well is it theoretically possible to estimate A given N independent measurements? Is there a unique estimator that always yields the best estimate?

In the following sections we will provide answers to these questions. We will start by defining the Minimum Variance Unbiased (MVU) estimator, and state its properties. Then the Cramer-Rao Lower Bound (CRLB) is introduced and its relationship with the MVU estimator is explored. This material will give a theoretical basis for the material that follows.

Linear models and the Best Linear Unbiased Estimator (BLUE) is presented next. These are models and estimators of great practical importance. Even when the underlying problem is not strictly linear, a linear approximation may be sufficiently good in many situations.

The Maximum Likelihood Estimator (MLE) is by far the most popular estimator, mostly due to its many good and even optimal

properties. We will define it and describe its properties, as well as present practical methods to determine the MLE.

Finally, we will present Bayesian estimators. The Bayesian approach differs from the classical approach in that the parameter of interest is no longer a fixed, deterministic value, but rather a realization of a random variable which in turn have a distribution. This allow prior assumptions about the parameter to modeled explicitly.

The Minimum Variance Unbiased Estimator

In this section we will define the Minimum Variance Unbiased (MVU) estimator, and discuss its properties. We will also present the Cramer-Rao Lower Bound (CRLB) and describe its relationship to the MVU.

Estimates are Random Variables

An essential fact about estimators is that the estimate they produce are *random variables*. This means that every time you use an estimator on a new set of data to produce an estimate of a parameter, the result will be different. This is an obvious consequence of the input to your estimator, the measurements or data, are random variables.

Since estimates are random variables they can be described in terms of distributions functions. Let us revisit Example 1 and write down the distribution functions for the two estimators suggested there.

Example 2. DC voltage in noise, continued *We want to measure the DC voltage between two points in an electric circuit. Our first estimator is based on a single measurement x*

$$\hat{A}_1 = x$$

where

$$x = A + w.$$

Since w is Gaussian noise with zero mean and variance σ^2, we can easily see that x, and hence $\hat{A}$, must have a Gaussian distribution with mean A and variance σ^2, that is

$$\hat{A}_1 \sim \mathcal{N}(A, \sigma^2)$$

Depending on the realization of the noise, the measurement x may be close or not to the parameter of interest, A. We say that $\hat{A} = x$, is an estimator of A.

Our second estimator is based on N independent measurements,

$$\hat{A}_N = \frac{1}{N} \sum_{n=0}^{N-1} x[n].$$

We can show that this estimator yields an estimate that is a Gaussian random variable as well, also with mean value A, but with variance σ^2/N,

$$\hat{A}_N \sim \mathcal{N}\left(A, \frac{\sigma^2}{N}\right)$$

The Unbiased Estimator

Armed with the knowledge that estimates are random variables, we can start to explore measures of goodness and optimality of

estimators. One obvious measure of goodness would be that the estimator should provide correct answers *on average*. We can formalize this using the expectation operator. If $\hat{\theta}$, is an estimator [1] of the parameter θ, then it is an *Unbiased Estimator* if

$$E\left[\hat{\theta}\right] = \theta.$$

[1] An estimator $\hat{\theta}$ uses data to produce an estimate, so in general we should write $\hat{\theta}(x[0], x[1], \ldots, x[N-1])$, that is as a proper function of the data. This is both cumbersome and verbose, so we hope using $\hat{\theta}$ will be sufficiently clear based on the context.

By this definition both the estimators from Example 2 are unbiased, and so it is clear that unbiasedness in itself is not enough to distinguish between two very different estimators. A more subtle issue is that an estimator $\hat{\theta}$ may not be unbiased for all θ.

Example 3. DC voltage in noise, unbiased estimators *We suggest a third estimator for our DC voltage in noise example:*

$$\hat{A} = \frac{1}{2N}\sum_{n=0}^{N-1} x[n]$$

We can show that the distribution of the estimate is Gaussian with mean $A/2$ and variance $\sigma^2/4$. In other words, the expected value of the estimate is $E[\hat{A}] = A/2$. This estimator is clearly unbiased for $A = 0$, but biased for all other values of A. Although this example is a bit contrived, it demonstrates that estimators can be unbiased for some parameter values and biased otherwise.

The Minimum Variance Criterion

From the previous subsection we saw that unbiasedness is not suited to be the sole optimality criterion used to evaluate and compare estimators. Another natural choice for optimality is the *mean square error* (MSE) between the estimate and the true parameter. We define the mean square error as

$$\text{mse}(\hat{\theta}) = E\left[(\hat{\theta} - \theta)^2\right]. \tag{1.1}$$

The MSE will penalize estimators that makes large errors in favor of estimators that on average makes smaller errors, including biases. We rewrite Equation 1.1 to show how the total MSE is composed of the estimator variance and the bias:

$$\begin{aligned}
\text{mse}(\hat{\theta}) &= E\left[(\hat{\theta} - \theta)^2\right] \\
&= E\left[(\hat{\theta} - E[\hat{\theta}] + E[\hat{\theta}] - \theta)^2\right] \\
&= E\left[\left((\hat{\theta} - E[\hat{\theta}]) + (E[\hat{\theta}] - \theta)\right)^2\right] \\
&= E\left[(\hat{\theta} - E[\hat{\theta}])^2\right] + 2E\left[(\hat{\theta} - E[\hat{\theta}])(E[\hat{\theta}] - \theta)\right] + E\left[(E[\hat{\theta}] - \theta)^2\right]
\end{aligned}$$

The first term is the variance of the estimator, $\text{var}(\hat{\theta})$. The second term is equal to zero since

$$\begin{aligned}
2E\left[(\hat{\theta} - E[\hat{\theta}])(E[\hat{\theta}] - \theta)\right] &= 2(E[\hat{\theta}] - \theta)E\left[(\hat{\theta} - E[\hat{\theta}])\right] \\
&= 2(E[\hat{\theta}] - \theta)(E(\hat{\theta}) - E[\hat{\theta}])) \\
&= 0
\end{aligned}$$

Third, if we define the bias $b(\theta) = E[\hat{\theta} - \theta]$, the final term is

$$\begin{aligned} E\left[(E[\hat{\theta}] - \theta)^2\right] &= (E[\hat{\theta}] - \theta)^2 \\ &= b^2(\theta) \end{aligned}$$

Finally, when we combine the three terms we get

$$\text{mse}(\hat{\theta}) = \text{var}(\hat{\theta}) + b^2(\theta) \tag{1.2}$$

As mentioned earlier, the MSE depends on the variance of the estimator and its bias. It is tempting to try and make a tradeoff between the variance and bias in an attempt to directly minimize the MSE. It turns out however, that this leads to estimators that are impossible to realize. We illustrate this by an example

Example 4. DC voltage in noise, optimal MSE estimator *We want to find an estimator that strikes the optimal balance between bias and variance and leads to the minimum MSE estimator. We start by scaling our mean estimator by a factor a. This yields*

$$\check{A} = \frac{a}{N}\sum_{n=0}^{N-1} x[n]$$

We can show that this estimator leads to the following MSE for a given scaling factor a:

$$mse(\check{A}) = \frac{a^2\sigma^2}{N} + (a-1)^2A^2$$

To find the a that minimizes the MSE we set the derivative with respect to a to zero and solve the equation. This yields the following solution:

$$a_{min} = \frac{A^2}{A^2 + \sigma^2/N}$$

It is now obvious why this estimator is unrealizable – to find the optimal estimator we need to know the parameter we are to estimate!

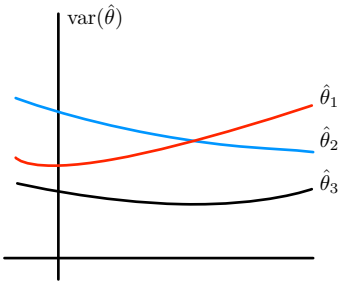


Figure 1.1: An artificial example on how different estimators may behave. We see that θ_1 is performing better than θ_2 for some parameter ranges, but worse in other. The estimator θ_3 is outperforming both, but we cannot say if it is the MVU estimator.

The Minimum Variance Unbiased Estimator

Since the unconstrained minimization of the MSE leads to estimators that may be unrealizable, we will constrain our estimators to be unbiased, minimum MSE. This leads to a class of estimators we will refer to as Minimum Variance Unbiased (MVU) estimators.

In general, for an arbitrary problem, there is no guarantee that we can find the MVU estimator. If it does, it may only be for some range of parameters, while another estimator is optimal for other ranges. A single estimator that is minimum for all parameters θ may not exist at all. As such, it should come as no surprise that in general, there is no straight forward, "mechanical" or "Step 1-2-3" approach to finding the MVU estimator, even when it does exist.

However, for a large class of relevant models and problems, we can identify the MVU estimator. Even more importantly, for every

estimation problem there exists a lower bound on how well a MVU estimator can perform. The bound may not be achievable, that is, there may not exist an estimator that performs as well as the bound. But *if* an estimator achieves the lower bound, we know it is the MVU estimator since it is theoretically impossible to perform better.

MVU estimators are also of great importance when we discuss linear systems later, as well as when we constrain our estimators to be linear. Next we will discuss this lower bound in more detail.

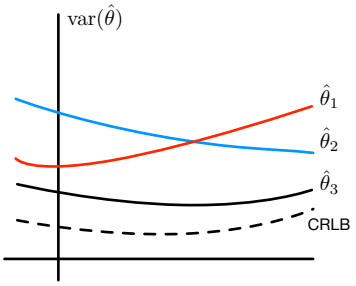


Figure 1.2: An artificial example revisited. We see that θ_3 is performing worse than the CRLB. In other words it is not efficient, but it may still be an MVU estimators if no other estimators outperform it.

The Cramer-Rao Lower Bound

In this section we will show that every MVU estimator $\hat{\theta}$ of some parameter θ will have a variance that is higher or equal to the Cramer-Rao Lower Bound (CRLB). Estimators that achieve the bound are called *efficient*. The bound comes in two forms, both which we will show next.

Theorem 1 (Cramer-Rao Lower Bound. Scalar version)**.** *Assume that our observations* $\mathbf{x}$ *are distributed according to the pdf* $p(\mathbf{x};\theta)$*, and also assume that* $p(\mathbf{x};\theta)$ *satisfies*

$$E\left[\frac{\partial \log p(\mathbf{x};\theta)}{\partial \theta}\right] = 0.$$

where the expectation is taken with respect to $p(\mathbf{x};\theta)$*. Then the variance of any MVU estimator* $\hat{\theta}$ *of the parameter* θ *will be bounded below by*

$$var(\hat{\theta}) \geq \frac{1}{-E\left[\frac{\partial^2 \log p(\mathbf{x};\theta)}{\partial \theta^2}\right]} = \frac{1}{E\left[\left(\frac{\partial \log p(\mathbf{x};\theta)}{\partial \theta}\right)^2\right]}. \tag{1.3}$$

The expectation is taken with respect to $p(\mathbf{x};\theta)$*. The derivative is taken with respect to the* true *parameter* θ*.*

If we make the expectation in Equation 1.3 explicit we get

$$E\left[\frac{\partial^2 \log p(\mathbf{x};\theta)}{\partial \theta^2}\right] = \int \frac{\partial^2 \log p(\mathbf{x};\theta)}{\partial \theta^2} p(\mathbf{x};\theta) d\mathbf{x}.$$

This expression may seem both unintuitive and complicated, but as we shall see in the next example it will become quite manageable for many problems. If the integral proves too much to solve explicitly, there are always numerical approaches.

Before we compute the CRLB for the DC in noise example, we will present a second result on the CRLB, which explicitly yields the MVU estimator if it exists.

Theorem 2 (Existence of an efficient estimator)**.** *Assume that our observations* $\mathbf{x}$ *are distributed according to the pdf* $p(\mathbf{x};\theta)$*. An MVU estimator* $\hat{\theta}$ *is efficient if and only if the following holds,*

$$\frac{\partial \log p(\mathbf{x};\theta)}{\partial \theta} = I(\theta)(g(\mathbf{x}) - \theta), \tag{1.4}$$

for two functions $g()$ *and* $I()$*. The MVU estimator is then* $\hat{\theta} = g(\mathbf{x})$*, and the corresponding variance of the estimator is* $var(\hat{\theta}) = I^{-1}(\theta)$*.*

We are now ready to find the CRLB for the problem of estimating a DC component in additive Gaussian noise. We will also show that the MVU exists and is given by the sample mean.

Example 5. DC voltage in noise, CRLB *As stated in Example 1, our measurements are given as*

$$x = A + w,$$

where w is white, Gaussian noise with zero mean and variance σ^2. This means that x has a Gaussian distribution as well, but with mean A. The pdf of a set of N observations $\mathbf{x} = x[0], x[1], \ldots$ is then given as

$$\begin{aligned} p(\mathbf{x};\theta) &= \prod_{n=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x[n]-A)^2}{\sigma^2}} \\ &= \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{1}{2}\sum_{n=0}^{N-1}\frac{(x[n]-A)^2}{\sigma^2}}. \end{aligned}$$

The CRLB is given by Equation 1.3. We start by computing the second derivative of the log-pdf with respect to the parameter A:

$$\begin{aligned} \frac{\partial^2 \log p(\mathbf{x};A)}{\partial A^2} &= \frac{\partial^2}{\partial A^2} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\sum_{n=0}^{N-1}\frac{(x[n]-A)^2}{\sigma^2}}\right) \\ &= \frac{\partial^2}{\partial A^2}\left(-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}\sum_{n=0}^{N-1}\frac{(x[n]-A)^2}{\sigma^2})\right) \\ &= \frac{\partial}{\partial A}\left(\sum_{n=0}^{N-1}\frac{(x[n]-A)}{\sigma^2})\right) \\ &= \sum_{n=0}^{N-1}\frac{-1}{\sigma^2} = \frac{-N}{\sigma^2}. \end{aligned}$$

Hence, the CRLB is

$$var(\hat{A}) \geq \frac{1}{-E\left[\frac{\partial^2 \log p(\mathbf{x};\theta)}{\partial\theta^2}\right]} = \frac{\sigma^2}{N}$$

We stated in Example 2 that the estimator

$$\hat{A}_N = \frac{1}{N}\sum_{n=0}^{N-1} x[n]$$

has variance $var(\hat{A}) = \frac{\sigma^2}{N}$, which means it is an MVU estimator attaining the CRLB.

We want to confirm this result using Theorem 2:

$$\begin{aligned} \frac{\partial \log p(\mathbf{x};A)}{\partial A} &= \frac{\partial}{\partial A}\log\left(\frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{1}{2}\sum_{n=0}^{N-1}\frac{(x[n]-A)^2}{\sigma^2}}\right) \\ &= \frac{\partial}{\partial A}\left(-\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2}\sum_{n=0}^{N-1}\frac{(x[n]-A)^2}{\sigma^2}\right) \\ &= \sum_{n=0}^{N-1}\frac{(x[n]-A)}{\sigma^2} \\ &= \frac{N}{\sigma^2}\left(\frac{1}{N}\sum_{n=0}^{N-1} x[n] - A\right) \\ &= \frac{N}{\sigma^2}\left(\hat{\theta} - A\right) \end{aligned}$$

We see that Theorem 2 is satisfied with

$$g(\mathbf{x}) = \frac{1}{N}\sum_{n=0}^{N-1} x[n]$$

and

$$I(A) = \frac{N}{\sigma^2}$$

which in turn confirms our earlier conclusions about the optimality of $\hat{\theta}$*. In other words, the sample mean estimator is both MVU and efficient for the DC in noise example.*

The next example will show that there are cases where we can prove that no efficient estimator exists. This does not preclude the existence of an MVU estimator, but for the time being this is goal is outside of our grasp. Note that this is a long example with what looks like a large amount of mathematics, but this is mostly because we take our time and try not to do too much between each equal-sign.

Example 6. CRLB of phase estimator *This example is similar to the previous case of estimating a DC value embedded in noise, but now it is the phase* ϕ *of a sinusoid with known amplitude* A *and frequency* f *that we wish to estimate from noisy observations. Also, we will not attempt to find an actual estimator, but rather to compute the CRLB.*

Our observations are on the form

$$x[n] = A\sin(2\pi f n + \phi) + w[n],$$

which we can rewrite as

$$w[n] = x[n] - A\sin(2\pi f n + \phi).$$

Assuming that the noise has a Gaussian distribution with zero mean and variance σ^2*, the pdf of the observation* $\mathbf{x} = x[0], x[1], \ldots$ *is*

$$\begin{aligned} p(\mathbf{x};\phi) &= \prod_{n=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x[n]-A\sin(2\pi f n+\phi))^2}{\sigma^2}} \\ &= \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{1}{2}\sum_{n=0}^{N-1}\frac{(x[n]-A\sin(2\pi f n+\phi))^2}{\sigma^2}} \end{aligned}$$

We can now start computing the CRLB by find the second derivative of the log*-pdf with respect to the unknown parameter* ϕ*. Note that we will use the following trigonometric relationships when we manipulate our equations,*

$$\begin{aligned} \sin(A \pm B) &= \sin A\ \cos B \pm \cos A\ \sin B \\ \cos(A \pm B) &= \cos A\ \cos B \mp \sin A\ \sin B, \end{aligned}$$

in addition to the well known $\cos^2 A + \sin^2 A = 1$.

We start by computing the first of the derivatives:

$$\begin{aligned}
\frac{\partial^2 \log p(\mathbf{x};\phi)}{\partial \phi^2} &= \frac{\partial^2}{\partial \phi^2} \log\left(\frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{1}{2}\sum_{n=0}^{N-1}\frac{(x[n]-A\sin(2\pi fn+\phi))^2}{\sigma^2}}\right) \\
&= \frac{\partial^2}{\partial \phi^2}\left(-\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2}\sum_{n=0}^{N-1}\frac{(x[n]-A\sin(2\pi fn+\phi))^2}{\sigma^2}\right) \\
&= \frac{\partial}{\partial \phi}\left(\sum_{n=0}^{N-1}\frac{(x[n]-A\sin(2\pi fn+\phi))\cdot A\cos(2\pi fn+\phi)}{\sigma^2}\right) \\
&= \frac{\partial}{\partial \phi}\left(\frac{1}{\sigma^2}\sum_{n=0}^{N-1}(x[n]\cdot A\cos(2\pi fn+\phi) - A^2\sin(2\pi fn+\phi)\cos(2\pi fn+\phi))\right) \\
&= \frac{\partial}{\partial \phi}\left(\frac{A}{\sigma^2}\sum_{n=0}^{N-1}(x[n]\cos(2\pi fn+\phi) - \frac{A}{2}\sin(4\pi fn+2\phi))\right)
\end{aligned}$$

Before we continue computing the second derivative, let us take a moment to investigate the results we've obtained so far. The expression inside the parenthesis is the derivative of $\log p(\mathbf{x};\phi)$*, that is,*

$$\log p(\mathbf{x};\phi) = \frac{A}{\sigma^2}\sum_{n=0}^{N-1}(x[n]\cos(2\pi fn+\phi) - \frac{A}{2}\sin(4\pi fn+2\phi))$$

We see that there is no way to write $\log p(\mathbf{x};\phi)$ *on the form* $I(\phi)(g(\mathbf{x}) - \phi)$ *for this particular problem, and recalling Theorem 2, we conclude that no efficient estimator exists. This does not preclude the existence of an MVU estimator however.*

Continuing with the differentiation we finally obtain,

$$\begin{aligned}
\frac{\partial^2 \log p(\mathbf{x};\phi)}{\partial \phi^2} &= \frac{\partial}{\partial \phi}\left(\frac{A}{\sigma^2}\sum_{n=0}^{N-1}(x[n]\cos(2\pi fn+\phi) - \frac{A}{2}\sin(4\pi fn+2\phi))\right) \\
&= \frac{A}{\sigma^2}\sum_{n=0}^{N-1}(-x[n]\sin(2\pi fn+\phi) - A\cos(4\pi fn+2\phi)) \\
&= \frac{A}{\sigma^2}\sum_{n=0}^{N-1}(-(A\sin(2\pi fn+\phi)+w[n])\sin(2\pi fn+\phi) - A\cos(4\pi fn+2\phi)) \\
&= \frac{A}{\sigma^2}\sum_{n=0}^{N-1}(-A\sin^2(2\pi fn+\phi) - w[n]\sin(2\pi fn+\phi) - A\cos(4\pi fn+2\phi))
\end{aligned}$$

Taking the expectation of the second derivative we get,

$$\begin{aligned}
E\left[\frac{\partial^2 \log p(\mathbf{x};\phi)}{\partial \phi^2}\right] &= E\left[\frac{A}{\sigma^2}\sum_{n=0}^{N-1}(-A\sin^2(2\pi fn+\phi) - w[n]\sin(2\pi fn+\phi) - A\cos(4\pi fn+2\phi))\right] \\
&= \frac{A}{\sigma^2}\sum_{n=0}^{N-1}(-A\sin^2(2\pi fn+\phi) - E[w[n]]\sin(2\pi fn+\phi) - A\cos(4\pi fn+2\phi)) \\
&= -\frac{A^2}{\sigma^2}\sum_{n=0}^{N-1}(\sin^2(2\pi fn+\phi) + \cos(4\pi fn+2\phi)) \\
&= -\frac{A^2}{\sigma^2}\sum_{n=0}^{N-1}\left(\frac{1}{2} - \frac{1}{2}\cos(4\pi fn+2\phi) + \cos(4\pi fn+2\phi)\right) \\
&= -\frac{A^2}{\sigma^2}\sum_{n=0}^{N-1}\left(\frac{1}{2} + \frac{1}{2}\cos(4\pi fn+2\phi)\right)
\end{aligned}$$

After all this work we can finally show that the CLRB for the phase estimation problem is

$$var(\hat{\phi}) \geq \frac{1}{\frac{A^2}{\sigma^2}\sum_{n=0}^{N-1}\left(\frac{1}{2}+\frac{1}{2}\cos(4\pi f n+2\phi)\right)} \tag{1.5}$$

We see that the bound is nowhere as simple and intuitive as for the DC in noise problem we've looked at until now. There is however no guarantee that such a closed form expression even exists for many of the problems we may encounter, and so we should appreciate the fact in this one of the rare cases. Using this expression we can easily compare the performance of a given estimator with the bound.

The Vector Cramer Rao Lower Bound

Until now we have restricted our discussion to scalar estimates. In general however, and for most practical problems, we are interested in estimating multiple parameters. In this case we want to estimate a *vector parameter*, $\Theta = [\theta_1, \theta_2, \ldots, \theta_d]$.

Let $\hat{\Theta}$ be an unbiased estimator of the vector parameter Θ. It can be shown that the variance of the of the ith parameter, $\hat{\theta}_i$, is

$$\text{var}(\hat{\theta}_i) \geq \left[\mathbf{I}^{-1}(\Theta)\right]_{ii},$$

where $[\mathbf{A}]_{ii}$ denotes the ith diagonal component of a matrix $\mathbf{A}$, and $\mathbf{I}(\Theta)$ is the $p \times p$ Fisher matrix. The Fisher matrix is in turn defined by the following,

$$[\mathbf{I}(\Theta)]_{ij} = -E\left[\frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial\theta_i \partial\theta_j}\right].$$

We see that if Θ is a scalar, the above definition becomes the familiar CRLB that was defined in the previous section.

As for the scalar case, we can sometimes determine the MVU estimator when the equality constraints are satisfied. For the vector case we have that $\hat{\Theta} = g(\mathbf{x})$ is an MVU estimator if

$$\nabla_{\hat{\Theta}} \log p(\mathbf{x};\Theta) = I(\Theta)(g(\mathbf{x}) - \Theta). \tag{1.6}$$

In the next example we will again return to the DC signal in noise, but this time the noise variance will be an unknown parameter as well.

Example 7. CRLB for vector parameter – DC voltage in noise with unknown variance *Let $x = A + w$, where A is an unknown DC level and w is Gaussian noise with unknown variance $\xi = \sigma^2$. Let $\mathbf{x} = x[0], x[1], \ldots, x[N-1]$ denote our observations. Our parameter vector of interest is*

$$\Theta = \begin{bmatrix} A \\ \xi \end{bmatrix} \in \mathbb{R}^2.$$

The Fisher information matrix for this problem is

$$\mathbf{I}(\Theta) = \begin{bmatrix} -E\left[\frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial A^2}\right] & -E\left[\frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial A \partial \xi}\right] \\ -E\left[\frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial \xi \partial A}\right] & -E\left[\frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial \xi^2}\right], \end{bmatrix}$$

where

$$\begin{aligned} \log p(\mathbf{x};\Theta) &= \log \prod_{n=0}^{N-1} p(x[n];\Theta) \\ &= -\frac{N}{2}\log 2\pi - \frac{N}{2}\log \xi - \frac{1}{2\xi}\sum_{n=0}^{N-1}(x[n]-A)^2 \end{aligned}$$

Computing the partial derivatives with respect to A and ξ we get,

$$\begin{aligned} \frac{\partial \log p(\mathbf{x};\Theta)}{\partial A} &= \frac{1}{\xi}\sum_{n=0}^{N-1}(x[n]-A) \\ \frac{\partial \log p(\mathbf{x};\Theta)}{\partial \xi} &= -\frac{N}{2\xi} - \frac{1}{2\xi^2}\sum_{n=0}^{N-1}(x[n]-A)^2 \end{aligned}$$

and computing the second partial derivatives finally yields

$$\begin{aligned} \frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial A^2} &= -\frac{N}{\xi} \\ \frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial A \partial \xi} &= -\frac{1}{\xi^2}\sum_{n=0}^{N-1}(x[n]-A) \\ \frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial \xi \partial A} &= -\frac{1}{\xi^2}\sum_{n=0}^{N-1}(x[n]-A) \\ \frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial \xi^2} &= \frac{N}{2\xi^2} - \frac{1}{\xi^3}\sum_{n=0}^{N-1}(x[n]-A)^2. \end{aligned}$$

The expectation values can then be computed

$$\begin{aligned} E\left[\frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial A^2}\right] &= -\frac{N}{\xi} \\ E\left[\frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial A \partial \xi}\right] &= -\frac{1}{\xi^2}\sum_{n=0}^{N-1}E(x[n]-A) = 0 \\ E\left[\frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial \xi \partial A}\right] &= -\frac{1}{\xi^2}\sum_{n=0}^{N-1}E(x[n]-A) = 0 \\ E\left[\frac{\partial^2 \log p(\mathbf{x};\Theta)}{\partial \xi^2}\right] &= \frac{N}{2\xi^2} - \frac{1}{\xi^3}\sum_{n=0}^{N-1}E(x[n]-A)^2 \\ &= \frac{N}{2\xi^2} - \frac{1}{\xi^3}\sum_{n=0}^{N-1}\xi \\ &= \frac{N}{2\xi^2} - \frac{N}{\xi^2} = -\frac{N}{2\xi^2} \end{aligned}$$

which finally yields the Fisher matrix

$$\mathbf{I}(\Theta) = \begin{bmatrix} \frac{N}{\sigma^2} & 0 \\ 0 & \frac{N}{2\sigma^4}. \end{bmatrix}$$

Inverting a diagonal matrix is trivial, and so the CRLBs for the two parameters A and σ^2 are

$$var(\hat{A}) = \frac{\sigma^2}{N}$$
$$var(\hat{\sigma}^2) = \frac{2\sigma^4}{N}$$

The Linear Model

From the previous section it should come as no surprise that finding the MVU estimator is in general a hard task. There is however, a class of problems where the MVU estimator always exists and is readily available to us. This is the class of *linear models*. In addition to tractable estimators, this class is also of great practical interest, as linearity is present, or at least approximately so, for many real world problems.

In this section we will define the linear model and the corresponding MVU estimator. This also prepares the stage for the next section, in which we will handle the case when the model is not strictly linear and define the Best Linear Unbiased Estimator (BLUE), which is of great practical use.

Definition of the Linear Model

Before going into details about the linear model, let us state the definition of a linear mapping:

Definition (Linear mapping)**.** *Let f be a mapping between two spaces X and Y,*

$$f : X \to Y.$$

If for all $a, b \in \mathbb{R}$ and $x, x' \in X$, we have that

$$f(ax + bx') = af(x) + bf(x'),$$

we say that f is a linear mapping.

We are interested in models that are linear in their parameters, $\theta_0, \theta_1, \ldots, \theta_{N-1}$. Some concrete examples of linear models follow:

Example 8. Some linear models *In the models that follow x is our observation, t is time and w is observation noise having a normal distribution, $\mathcal{N}(w; 0, \sigma^2)$.*

- *Constant plus noise: $x = A + w$. Yes, this is the DC voltage in noise that we have been studying for some time.*
- *Linear trend: $x = At + B + w$. Here our observation depends on the time t through the straight line determined by the parameters $\{A, B\}$.*
- *Linear trend plus seasonal variation: $x = At + B\sin(\omega t) + C + w$. This model combines a linear trend with seasonal variation modeled by a sinusoid.*
- *General: $x = Ag_1(t) + Bg_2(t) + Cg_3(t) + w$. Here $\{g_i\}$ are arbitrary functions that cover all the above examples.*

In what follows we are interested in the d-dimensional parameter vector $\Theta = [\theta_0, \ldots, \theta_{d-1}]$, based on a set of N observations $\{x[n]\}$. Our model is the general model[2]

[2] In this definition we have $f[n]$ instead of $f(t)$ as in Example 8. You should think of $f(t)$ a proper function of time t, while $f[n] = f(t_n)$ is its realization corresponding to the nth measured value of x.

$$x[n] = \sum_{d=0}^{M-1} \theta_d f_d[n] + w[n],$$

where the noise is iid., $w[n] \sim \mathcal{N}(w; 0, \sigma^2)$. In matrix form this becomes

$$\mathbf{x} = H\Theta + \mathbf{w}. \tag{1.7}$$

Any model that can be written on the form of Equation 1.7 is a *linear model*. The matrix $H = \{f_d[n]\}$ is the *observation matrix* and is known. The vector $\mathbf{x} = \{x[n]\}$ contains our observations, while the vector $\Theta = \{\theta_d\}$ is the set of unknown parameters we want to estimate.

The Estimator for the Linear Model

Equation 1.7 is a linear set of equations, but it is obviously not directly solvable in terms of Θ since H in general is not invertible with $N \gg d$. This is a well known problem however, and the common solution is to solve

$$\hat{\Theta} = \arg\min_{\Theta} \|\mathbf{x} - H\Theta\|, \tag{1.8}$$

thats is, the estimate is the parameter vector Θ that minimizes the error $\mathbf{e} = \mathbf{x} - H\Theta$. This will in fact yield the MVU estimator, which has the bonus property of being efficient! We will find the estimator twice – first by solving Equation 1.8, and then by using the CRLB.

First we write out the objective function we want to minimize,

$$\begin{aligned}
\hat{\Theta} &= \arg\min_{\Theta} \|\mathbf{x} - H\Theta\| \\
&= \arg\min_{\Theta} (\mathbf{x} - H\Theta)^T (\mathbf{x} - H\Theta) \\
&= \arg\min_{\Theta} (\mathbf{x}^T\mathbf{x} - \mathbf{x}^T H\Theta - \Theta^T H^T \mathbf{x} + \Theta^T H^T H\Theta).
\end{aligned}$$

We then compute the gradient wrt. Θ, set it equal to zero and solve,

$$\begin{aligned}
&\nabla_{\hat{\Theta}} (\mathbf{x}^T\mathbf{x} - \mathbf{x}^T H\hat{\Theta} - \hat{\Theta}^T H^T \mathbf{x} + \hat{\Theta}^T H^T H\hat{\Theta}) \\
&= 0 - H^T\mathbf{x} - H^T\mathbf{x} + 2H^T H\hat{\Theta} \\
&= -2H^T x + 2H^T H\hat{\Theta} \\
&= 0,
\end{aligned}$$

which in the end yields[3]

$$\Rightarrow \hat{\Theta} = \left(H^T H\right)^{-1} H^T \mathbf{x} \tag{1.9}$$

We will now show, using the vector CRLB, that this estimator is in fact both MVU *and* efficient. Starting by rearranging Equation 1.7 as follows

$$\mathbf{w} = \mathbf{x} - H\Theta,$$

it is easy to see that the pdf of $\mathbf{x}$, $p(\mathbf{x}; \Theta)$, can be written

$$p(\mathbf{x}; \Theta) = (2\pi|\Sigma|)^{-N/2} e^{-\frac{1}{2}(\mathbf{x}-H\Theta)^T \Sigma^{-1} (\mathbf{x}-H\Theta)}.$$

[3] Note that this solution is only valid when $H^T H$ is invertible. As long as there are more observations, N, than unknowns, d, and the observation matrix H don't have columns that are linearly dependent, $H^T H$ is invertible – in theory. In practice, $H^T H$ may be *ill-conditioned*, leading to numerical problems.

Computing $\nabla_{\hat{\Theta}} \log p(\mathbf{x};\Theta)$ yields,

$$\begin{aligned}
&\nabla_{\hat{\Theta}} \log p(\mathbf{x};\Theta) \\
=&\nabla_{\hat{\Theta}} \log \left((2\pi|\Sigma|)^{-N/2} e^{-\frac{1}{2}(\mathbf{x}-H\Theta)^T\Sigma^{-1}(\mathbf{x}-H\Theta)} \right) \\
=&\nabla_{\hat{\Theta}} \left(-\frac{N}{2}\log(2\pi|\Sigma|) - \frac{1}{2}(\mathbf{x}-H\Theta)^T\Sigma^{-1}(\mathbf{x}-H\Theta) \right) \\
=&-\frac{1}{2}\nabla_{\hat{\Theta}} \left(\mathbf{x}^T\Sigma^{-1}\mathbf{x} - \mathbf{x}^T\Sigma^{-1}H\Theta - \Theta^T H^T\Sigma^{-1}\mathbf{x} + \Theta^T H^T\Sigma^{-1}H\Theta \right) \\
=&H^T\Sigma^{-1}\mathbf{x} - H^T\Sigma^{-1}H\Theta \\
=&H^T\Sigma^{-1}H \left((H^T\Sigma^{-1}H)^{-1}H^T\Sigma^{-1}\mathbf{x} - \Theta \right).
\end{aligned}$$

We now use the fact that w is iid., which means that $\Sigma = \sigma^2 I$, where I is the identity matrix:

$$\begin{aligned}
&H^T\Sigma^{-1}H \left((H^T\Sigma^{-1}H)^{-1}H^T\Sigma^{-1}\mathbf{x} - \Theta \right) \\
=&\frac{H^TH}{\sigma^2} \left((\sigma^{-2}H^TH)^{-1}\sigma^{-2}H^T\mathbf{x} - \Theta \right) \\
=&\frac{H^TH}{\sigma^2} \left((H^TH)^{-1}H^T\mathbf{x} - \Theta \right) \\
=&I(\Theta)(g(\mathbf{x}) - \Theta)
\end{aligned}$$

From our knowledge of the CRLB we see that

$$\hat{\Theta} = (H^TH)^{-1}H^T\mathbf{x} \tag{1.10}$$

is MVU and efficient, and that the covariance matrix of the estimator is given by

$$C_\Theta = I^{-1}(\Theta) = \sigma^2(H^TH)^{-1} \tag{1.11}$$

Linear Model with Non-White Noise

In many applications the assumption that the noise w is iid. does not hold, and $\Sigma \neq \sigma^2 I$. The estimator in this case is still MVU and efficient, but instead of proving this from scratch using the CRLB as in the last section, we will use a noise whitening argument.

We assume that the noise covariance matrix, Σ, is positive definite.[4] This means that Σ^{-1} is positive definite as well, and that we can write we can write

$$\Sigma^{-1} = S^TS,$$

where S is an invertible matrix that will always exist. Writing our linear model on matrix form and then multiplying both sides by S we have,

$$\begin{aligned}
\mathbf{x} &= H\Theta + \mathbf{w} \\
\Rightarrow S\mathbf{x} &= SH\Theta + S\mathbf{w} \\
\Rightarrow \mathbf{x}' &= H'\Theta + \mathbf{w}'
\end{aligned}$$

We know that a linear transformation of a random vector drawn from a distribution with mean μ and covariance matrix Σ yields a new random vector from a distribution with mean $\mu' = A\mu$ and

[4] A positive definite matrix has the property that $\mathbf{x}^TA\mathbf{x} > 0\ \forall \mathbf{x}$. In principle, Σ may be positive semi-definite, which means that the noise is constrained to a linear subspace of $\mathbb{R}^N$. We will not be addressing this special case.

covariance matrix $\Sigma' = A\Sigma A^T$. This means that the distribution of $\mathbf{w}'$ has zero mean and covariance

$$\begin{aligned}\Sigma' &= S\Sigma S^T \\ &= S(S^TS)^{-1}S^T \\ &= SS^{-1}S^{-1^T}S^T \\ &= I,\end{aligned}$$

which in turn means that the noise of the transformed observations are iid. with variance $\sigma^2 = 1$. Based on the results from the previous section we can now write the estimator, which is MVU and efficient, as

$$\hat{\Theta} = (H'^TH')^TH'^T\mathbf{x}' \tag{1.12}$$
$$= (H^TS^TSH)^{-1}H^TS^TS\mathbf{x} \tag{1.13}$$
$$= (H^T\Sigma^{-1}H)^{-1}H^T\Sigma^{-1}\mathbf{x} \tag{1.14}$$

We now present an example where the unknown DC level is embedded in colored noise.

Example 9. DC voltage in colored noise *Extending our previous model we now have observations*

$$x[n] = A + w[n],\ n = 0, 1, 2, \ldots, N-1$$

where the noise $w[n]$ is correlated with a covariance matrix Σ. We can write our model on matrix form as

$$\mathbf{x} = H\Theta + \mathbf{w},$$

where $\Theta = A$ and $H = [1, 1, \ldots, 1]^T$. Plugging this into our estimator we get

$$\begin{aligned}\hat{A} &= (H^T\Sigma^{-1}H)^{-1}H^T\Sigma^{-1}\mathbf{x} \\ &= \frac{\mathbf{1}^T\Sigma^{-1}\mathbf{x}}{\mathbf{1}^T\Sigma^{-1}\mathbf{1}}.\end{aligned}$$

This estimator has an interesting interpretation when we formulate it in terms of the pre-whitener S,

$$\begin{aligned}\hat{A} &= \frac{\mathbf{1}S^TS\mathbf{x}}{\mathbf{1}S^TS\mathbf{1}} = \frac{(S\mathbf{1})^TS\mathbf{x}}{\mathbf{1}S^HS\mathbf{1}} = \frac{(S\mathbf{1})^T}{\mathbf{1}S^TS\mathbf{1}}\mathbf{x}' \\ &= \sum_{n=0}^{N-1} a_n x'[n],\end{aligned}$$

that is, our estimate is a weighted sum of whitened observations. It is also easy to show that for iid. noise the estimator becomes the sample mean.

The Best Linear Unbiased Estimator

Often one will encounter problems for which there is no MVU estimator to be found, or that the MVU estimator is too complicated to use in practice. A common cause is that the pdf of the data is unknown, which precludes the use of the CRLB. In these cases we need to find an estimator that, while not optimal, at least fulfills some specifications – ie. that is "good enough".

A common estimator estimator in this case is one that is *linear in the observation data*,

$$\hat{\theta} = \sum_{n=0}^{N-1} a_n x[n]. \tag{1.15}$$

From Example 9 we know that this is the MVU estimator for the linear model. For a problem that is not covered by the linear model, this estimator is obviously no longer MVU, but we still want to find the set of weights $\{a_n\}$ so that we minimize the estimator variance.[5] This problem can be solved using only the first two moments of the pdf of the observations. The resulting estimator is the Best Linear Unbiased Estimator (BLUE).

[5] Since the CRLB is not available, there is really no way to compare this variance with an optimal lower bound. In practice one needs to do computer simulations and compare the results to the problem specifications.

Before we derive the BLUE, we should make clear that it may not be a useful estimator at all for some problems.

Example 10. BLUE estimator of variance *We have a set of observations, $x[0], x[1], \ldots, x[N-1]$, drawn iid. from a distribution having zero mean and variance σ^2. We want to estimate the variance using the BLUE estimator, which is*

$$\hat{\sigma}^2 = \sum_{n=0}^{N-1} a_n x[n],$$

for some optimal set of weights $\{a_n\}$. It is easy to show however, that

$$\begin{aligned} E\left[\hat{\sigma}^2\right] &= E\left[\sum_{n=0}^{N-1} a_n x[n]\right] \\ &= \sum_{n=0}^{N-1} a_n E\left[x[n]\right] \\ &= \sum_{n=0}^{N-1} a_n \cdot 0 \\ &= 0, \end{aligned}$$

for all $\{a_n\}$, meaning this estimator will never be unbiased.

In fact, the BLUE estimator can only be guaranteed to be feasible when the expectation of an observation is linear in θ, thats is, $E[x[n]] = s_n\theta$. The set of $\{s_n\}$ is referred to as the *scaled mean*, and is assumed known.

This begs the question whether we are really not just constraining ourselves back to the linear model from the last section. The answer is no, as the observation noise is no longer constrained to be Gaussian, which as we shall see facilitates a much richer set of models.

Deriving the BLUE

To obtain the BLUE we constrain the estimator $\hat{\theta}$ to be unbiased, which in turn constrains the expectation of the observations to be linear in θ. This yields

$$\begin{aligned} E\left[\hat{\theta}\right] &= \theta \\ \Rightarrow \sum_{n=0}^{N-1} a_n E\left[x[n]\right] &= \theta \\ \Rightarrow \sum_{n=0}^{N-1} a_n s_n \theta &= \theta \\ \Rightarrow \sum_{n=0}^{N-1} a_n s_n &= 1 \\ \Rightarrow \mathbf{a}^T \mathbf{s} &= 1 \end{aligned}$$

The variance of the estimator is

$$\begin{aligned} \mathrm{var}(\hat{\theta}) &= E\left[\left(\hat{\theta} - \theta\right)^2\right] \\ &= E\left[\left(\hat{\theta} - E[\hat{\theta}]\right)^2\right] \text{ (use unbiasedness)} \\ &= E\left[\left(\sum_{n=0}^{N-1} a_n x[n] - E\left[\sum_{n=0}^{N-1} a_n x[n]\right]\right)^2\right] \\ &= E\left[\left(\sum_{n=0}^{N-1} a_n x[n] - \sum_{n=0}^{N-1} a_n E\left[x[n]\right]\right)^2\right] \\ &= E\left[\left(\mathbf{a}^T \mathbf{x} - \mathbf{a}^T E[\mathbf{x}]\right)^2\right] \text{ (rewrite sum as inner product)} \\ &= E\left[\left(\mathbf{a}^T (\mathbf{x} - E[\mathbf{x}])\right)^2\right] \\ &= E\left[\mathbf{a}^T \left(\mathbf{x} - E[\mathbf{x}]\right) \left(\mathbf{x} - E[\mathbf{x}]\right)^T \mathbf{a}\right] \text{ (use } (\mathbf{x}^T\mathbf{y})^2 = \mathbf{x}^T\mathbf{y}\mathbf{y}^T\mathbf{x}) \\ &= \mathbf{a}^T E\left[\left(\mathbf{x} - E[\mathbf{x}]\right) \left(\mathbf{x} - E[\mathbf{x}]\right)^T\right] \mathbf{a} \\ &= \mathbf{a}^T C \mathbf{a} \end{aligned}$$

Finding the BLUE now becomes the problem of minimizing $\mathbf{a}^T C \mathbf{a}$ with respect to $\mathbf{a}$, subject to the unbiasedness constraint, $\mathbf{a}^T \mathbf{s} = 1$. The solution is a straight forward optimization problem involving the use of Lagrange multipliers.[6]

The object function we need to minimize can now be written

$$\mathbf{a}_{\min} = \arg\min_{\mathbf{a}} \mathbf{a}^T C \mathbf{a} + \lambda \left(\mathbf{a}^T \mathbf{s} - 1\right).$$

To find the optimal $\mathbf{a}$ we proceed as we always do – find the gradient of the object function wrt. $\mathbf{a}$, set it equal to zero and solve. We simplify our problem a bit by multiplying the quadratic term by $1/2$. This does not affect our problem as the Lagrange term should

[6] Min-/maximizing an object function, say $L(x)$, subject to some equality constraint $c(x) = a$ is as simple as adding the term $\lambda(c(x) - a)$ to the optimization problem. The constant λ is the Lagrange multiplier.

be zero when the constraint is satisfied:

$$\begin{aligned}
&\nabla_{\mathbf{a}} \left(\frac{1}{2} \mathbf{a}^T C \mathbf{a} + \lambda \left(\mathbf{a}^T \mathbf{s} - 1 \right) \right) \\
=& C\mathbf{a} + \lambda \mathbf{s} \\
=& 0 \\
\Rightarrow \mathbf{a} =& -\lambda C^{-1} \mathbf{s}
\end{aligned}$$

We've found $\mathbf{a}$, but we need to find the Lagrange multiplier before we can use the expression. Using the fact that $\mathbf{a}^T \mathbf{s} = 1$, we can multiply both sides of the above solution by $\mathbf{s}^T$, which yields

$$\begin{aligned}
\mathbf{s}^T \mathbf{a} &= -\lambda \mathbf{s}^T C^{-1} \mathbf{s} \\
\Rightarrow 1 &= -\lambda \mathbf{s}^T C^{-1} \mathbf{s} \\
\Rightarrow \lambda &= -\left(\mathbf{s}^T C^{-1} \mathbf{s} \right)
\end{aligned}$$

Finally, replacing λ in the solution for $\mathbf{a}$ yields,

$$\mathbf{a}_{\min} = \frac{C^{-1} \mathbf{s}}{\mathbf{s}^T C^{-1} \mathbf{s}}.$$

Plugging this into our estimator we get

$$\begin{aligned}
\hat{\theta} &= \sum_{n=0}^{N-1} a_n x[n] \\
&= \mathbf{a}^T \mathbf{x} \\
&= \frac{\mathbf{s}^T C^{-1} \mathbf{x}}{\mathbf{s}^T C^{-1} \mathbf{s}}
\end{aligned}$$

We see that the estimator is expressed in terms of the two first moments of the pdf of the observations. Lets illustrate the BLUE in an example.

Example 11. DC voltage in uncorrelated noise *In the example our model is*

$$x = A + w,$$

with the noise being non-Gaussian and uncorrelated, meaning

$$E[w_m w_n] = \begin{cases} 0 & m \neq n \\ \sigma_n^2 & m = n \end{cases}$$

Since the expected value of $x[n]$ is $E[x[n]] = A$, the scaled means s_n are all equal to one, $\mathbf{s} = [1, 1, \ldots]^T$. The estimator $\hat{A}$ then becomes

$$\begin{aligned}
\hat{A} &= \frac{\mathbf{s}^T C^{-1} \mathbf{x}}{\mathbf{s}^T C^{-1} \mathbf{s}} \\
&= \frac{\mathbf{1}^T C^{-1} \mathbf{x}}{\mathbf{1}^T C^{-1} \mathbf{1}} \\
&= \frac{\sum_{n=0}^{N-1} \sigma_n^{-2} x[n]}{\sum_{n=0}^{N-1} \sigma_n^{-2}}.
\end{aligned}$$

We see that observations with large variances are weighted down, emphasizing the "clean" observations.

Vector BLUE

The extension of BLUE from scalar to vector parameters is straight forward. Let Θ be a vector of unknown parameters, θ_i. Then for each unknown parameter we have

$$\hat{\theta}_i = \sum_{n=0}^{N-1} a_{in} x[n]$$

which on matrix form becomes

$$\hat{\Theta} = A\mathbf{x}.$$

The unbiasedness requirements are the same as in the scalar case, that is

$$E[\hat{\Theta}] = AE[\mathbf{x}] = \Theta. \tag{1.16}$$

Again, the expectation of the observation vector must be linear in Θ, which in matrix notation becomes,

$$E[\mathbf{x}] = H\Theta, \tag{1.17}$$

where H is a known matrix. Substituting Equation 1.17 into Equation 1.16, we get

$$AE[\mathbf{x}] = AH\Theta = \Theta \Rightarrow AH = I, \tag{1.18}$$

where I is the identity matrix.[7] This is the equivalent of the constraint $\mathbf{s}^T\mathbf{a} = 1$ in the scalar case.

[7] The matrices A and H are not in general invertible, meaning that $A \neq H^{-1}$. Instead A is a *pseudo-inverse*, also called a Moore-Penrose inverse. The pseudo-inverse is unique and identical to the inverse if H is invertible.

As for the scalar case, the variance of each of the estimators $\hat{\theta}_i$ is

$$\text{var}(\hat{\theta}_i) = \mathbf{a}_i^T C \mathbf{a}_i, \tag{1.19}$$

where C is the covariance matrix of our observation vector $\mathbf{x}$. The vector BLUE estimator is found by minimizing Equation 1.19 subject to the constraint in Equation 1.18 for all i. This yields the estimator

$$\hat{\Theta} = (H^T C^{-1} H)^{-1} H^T C^{-1} \mathbf{x}. \tag{1.20}$$

The covariance of the estimator $\hat{\Theta}$ is

$$C_{\hat{\Theta}} = (H^T C^{-1} H)^{-1} \tag{1.21}$$

Next we give a concrete example on how to do simple localization of a flying object, and we'll see how the known matrices, H and C, are identified naturally during the modeling.

Example 12. 2D localization using timing information

In this examples we will give an example on how to track an object using the time of the reception of radio signals from the object. See Figure 1.3 for a sketch of the problem. Here an object is located somewhere in the $x-y$ plane. At some time T_0 it sends a signal that is received at N different radio receivers at times $\{t_n\}$, where

$$t_n = T_0 + \frac{R_n}{c} + \epsilon_n.$$

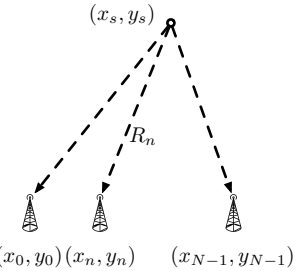


Figure 1.3: A simple example on how BLUE can be used for localization and tracking. An object sends out a signal at time T_0, which in turn is received by N different radios as times $\{t_n\}$. We will use this information to find the coordinates of the object in the $x-y$ plane.

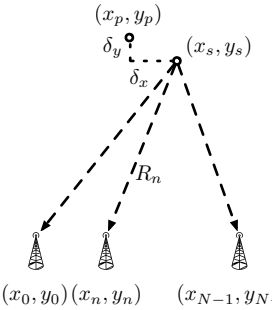


Figure 1.4: We approximate the localization problem by assuming that we know the previous position of the object, (x_p, y_p), and that the new position, (x_s, y_s) is a small distance away.

Here R_n is the distance between the object and receiver n, $c = 3 \times 10^8$ is the speed of light and ϵ_n is an iid. noise term with zero mean and variance σ^2, but otherwise unknown distribution. Simple trigonometry tells us that

$$R_n = \sqrt{(x_s - x_n)^2 + (y_s - y_n)^2},$$

which in principle is all we need to estimate the position, (x_s, y_s). It is however unclear what the optimal estimator is, or if an MVU estimator even exists. Instead of tackling a set of non-linear equations we will instead look at the problem given that we know some previous position that is close to the new one. Assume that the new position is

$$(x_s, y_s) = (x_p + \delta_x, y_p + \delta_y),$$

and that (δ_x, δ_y) are small relative to (x_p, y_p). We can then linearize the distance R_n using a simple Taylor series, yielding the approximation

$$\begin{aligned} R_n \approx & R'_n + \frac{x_p - x_n}{R'_n}\delta_x + \frac{y_p - y_n}{R'_n}\delta_y \\ = & R'_n + \cos(\alpha_n)\delta_x + \sin(\alpha_n)\delta_y \end{aligned}$$

where R'_n is the distance from radio receiver at (x_n, y_n) to the previous object position (x_p, y_p), and α_n is the angle between the x-axis and the straigth line between the receiver and the object. The signal reception time t_n can now be written,

$$\begin{aligned} t_n = & T_0 + \frac{R_n}{c} + \epsilon_n \\ \approx & T_0 + \frac{R'_n}{c} + \frac{\cos(\alpha_n)}{c}\delta_x + \frac{\sin(\alpha_n)}{c}\delta_y + \epsilon_n. \end{aligned}$$

We now have a set of equations for finding the unknown position change, (δ_x, δ_y), which added to the previously known position will yield the new position (x_s, y_s). The problem is now almost *on the form of being linear in the observations – we only need to get rid of the two constant terms, T_0 and R'_n/c. Since R'_n is known, we can just define a new set of equations*

$$\tau_n = t_n - \frac{R'_n}{c} = T_0 + + \frac{\cos(\alpha_n)}{c}\delta_x + \frac{\sin(\alpha_n)}{c}\delta_y + \epsilon_n.$$

The time that the signal was transmitted from the object, T_0, is unknown. To cancel this unknown variable we instead look at the difference in reception time between the radio receivers,

$$\begin{aligned} \xi_n = & \tau_n - \tau_{n-1} \\ = & \frac{\cos(\alpha_n) - \cos(\alpha_{n-1})}{c}\delta_x + \frac{\sin(\alpha_n) - \sin(\alpha_{n-1})}{c}\delta_y + \epsilon_n - \epsilon_{n-1}. \end{aligned}$$

The problem is now on a form that enables us to use the BLUE. Let

$$\Theta = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix},$$

$$H = \begin{bmatrix} cos(\alpha_1) - \cos(\alpha_0) & \sin(\alpha_1) - \sin(\alpha_0) \\ \vdots & \vdots \\ cos(\alpha_{N-1}) - \cos(\alpha_{N-1}) & \sin(\alpha_{N-1}) - \sin(\alpha_{N-1}) \end{bmatrix},$$

and

$$\begin{aligned}\mathbf{w} &= \begin{bmatrix} \epsilon_1 - \epsilon_0 \\ \dots \\ \epsilon_{N-1} - \epsilon_{N-2} \end{bmatrix} \\ &= \begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_{N-1} \end{bmatrix} \\ &= A\epsilon \end{aligned}$$

Since ϵ_n is iid. with variance σ^2, the covariance matrix for the noise vector ϵ is $\sigma^2 I$, which in turn means that the covariance matrix for $\mathbf{w}$ is $\sigma^2 AA^T$. We finally have our BLUE,

$$\hat{\Theta} = \left(H^T \left(AA^T \right)^{-1} H \right)^1 H^T \left(AA^T \right)^{-1} \xi$$

together with the estimator covariance matrix,

$$C_{\hat{\Theta}} = \sigma^2 \left(H^T \left(AA^T \right)^{-1} H \right)^1$$

The Maximum Likelihood Estimator

As mentioned earlier in this handout, the MVU estimator may not be available for a variety of reasons. The Maximum Likelihood Estimator, or MLE, is a commonly used alternative, and may just be *the* most popular estimator in practical use. It owes much of its popularity to an in many cases (almost) turn-the-crank easy of use, and good asymptotic properties.

MLE Definition

The MLE is based on the *likelihood function*. This functions is simply the pdf of our observations $\mathbf{x}$, but a function of the parameter θ instead of the observations,

$$L(\theta|\mathbf{x}) = p(\mathbf{x};\theta). \tag{1.22}$$

The likelihood function measures the likelihood of observing $\mathbf{x}$ for any particular value of θ. The idea behind the MLE is that values of θ that makes our observations more likely are better estimates. We can write this formally as[8]

$$\theta_{\text{MLE}} = \arg\max_{\theta} L(\theta|\mathbf{x}) \tag{1.23}$$

The optimization of Equation 1.23 can in some cases yield closed form solutions, but for more complicated problems one usually resorts to numerical search algorithms.

[8] It is common practice to instead use the *log-likelihood*, $l(\theta|\mathbf{x}) = \log p(\mathbf{x};\theta)$, in the maximization, or even minimizing the negative log-likelihood. The estimator is identical due to the monotone increments of the log-function. The main motivation is that the algebraic manipulations may be significantly simpler, and numerical methods less prone to instabilities.

Example 13. DC Voltage in noise – MLE. *Our observations are once again given as*

$$x = A + w,$$

where $w \sim \mathcal{N}(0,\sigma^2)$. *This means that* $x[n]$ *is distribute according to* $\mathcal{N}(A,\sigma^2)$. *We can write down the log-likelihood function,*

$$\begin{aligned} l(\hat{A}|\mathbf{x}) &= \log p(\mathbf{x};\hat{A}) \\ &= \log \prod_{n=0}^{N-1} p(x[n];\hat{A}) \\ &= \sum_{n=0}^{N-1} \log p(x[n];\hat{A}) \\ &= \sum_{n=0}^{N-1} -\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2}\frac{(x[n]-\hat{A})^2}{\sigma^2} \end{aligned}$$

and set the derivative wrt. A to zero and solve,

$$\begin{aligned}
&\frac{d}{dA} l(\hat{A}|\mathbf{x}) = \sum_{n=0}^{N-1} \frac{x[n] - \hat{A}}{\sigma^2} = 0 \\
\Rightarrow &\sum_{n=0}^{N-1} x[n] = \sum_{n=0}^{N-1} \hat{A} \\
\Rightarrow &\sum_{n=0}^{N-1} x[n] = N\hat{A} \\
&\Rightarrow \hat{A} = \frac{1}{N} \sum_{n=0}^{N-1} x[n]
\end{aligned}$$

We see that in this case the MLE returns the well known MVU estimator.

In the last example the MLE is MVU, which is *not* the case in general. In can be shown however, that the MLE is *asymptotically* MVU, provided the observations are iid. We will address this in the next section.

Asymptotic Properties of the MLE

In this section we will show that the MLE is both *consistent* and *asymptotically efficient*, which in turn mean that as the number of observations increases, the MLE performance will approach the CRLB.

We first define consistency. Let $\{\hat{\theta}_N\}$ be a series of estimators that take N iid. observations $x[0], \ldots, x[N-1]$ and yields an estimate. If for all $\epsilon > 0$ we have that

$$\lim_{N\to\infty} P(|\hat{\theta}_N - \theta| \leq \epsilon) = 1,$$

or alternatively

$$\lim_{N\to\infty} P(|\hat{\theta}_N - \theta| \geq \epsilon) = 0,$$

then $\{\hat{\theta}_N\}$ is a consistent sequence of estimators. The definition may seem complicated, but its meaning is clear – as the number of samples N increases, the probability that the estimate deviates significantly from the true parameter value θ, goes to zero. That is, as $N \to \infty$, $\hat{\theta}_{\text{MLE}}$ becomes unbiased.

Likewise for the asymptotic efficiency, let $\{\hat{\theta}_N\}$ be a series of estimators that take N iid. observations $x[0], \ldots, x[N-1]$ and yields an estimate. Then

$$\sqrt{N}(\hat{\theta}_N - \theta) \xrightarrow{d} \mathcal{N}(0, I^{-1}(\theta)),$$

as $N \to \infty$, and $I^{-1}(\theta)$ is the inverse Fischer information, that is, the CRLB. In other words, as the number of observations increase, the MLE becomes unbiased, and the variance of the estimate converges to the CRLB.

The big question of course becomes – how large does N need to be for all these beneficial properties to befall us. Unfortunately, the distribution of θ_{MLE} is intractable for finite N in almost all cases. The only solution usually available is to rely on computer simulation.

Invariance property of the MLE

A final property of the MLE that is worth a mention is the invariance property. Informally, this property guarantees that if we are after some *function* of the unknown parameter, eg. $\tau(\theta)$, and *not the unknown parameter itself*, then the MLE of $\tau(\theta)$ is $\tau(\hat{\theta})$. Here $\hat{\theta}$ is the MLE of θ. This is different from the MVU estimator case, where transformations of parameters having an MVU estimator in general leads to new parameters having no MVU estimator.

Example 14. Transformed DC level in white Gaussian noise *We have the model*

$$x = A + w,$$

where A is unknown and $w \sim \mathcal{N}(0, \sigma^2)$. We want to find the MLE of $\alpha = e^A$. The pdf parameterized by A is

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

which in turn can be written in terms of α as

$$p(x; \alpha) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x-\log\alpha)^2}{\sigma^2}},$$

since the log*-function is one-to-one. Computing the derivative of the* log *of the above equation wrt. to α, setting it equal to zero and solving gives us*

$$\sum_{n=0}^{N-1} (x[n] - \log\hat{\alpha}) \frac{1}{\hat{\alpha}} = 0$$
$$\Rightarrow \hat{\alpha} = e^{\frac{1}{N}\sum_{n=0}^{N-1}}.$$

But $\frac{1}{N}\sum_{n=0}^{N-1}$ is the MLE of A, so

$$\hat{\alpha} = e^{\hat{A}}$$

The above is a simplified statement of the invariance property as it is only true when the mapping between θ and $\eta = \tau(\theta)$ is one-to-one. That is, for every η there is one and only one θ so that $\eta = \tau(\theta)$. This is is not true for most cases of practical interest however, the simple square of the parameter, $\eta = \theta^2$, being the most obvious example. In this case it is possible that for an MLE $\hat{\theta}$ there can also exist a θ_0 so that $\tau(\hat{\theta}) = \tau(\theta_0)$.

To settle this problem one can define the *induced maximum likelihood*

$$L^\star(\eta|\mathbf{x}) = \max_{\{\theta:\tau(\theta)=\eta\}} L(\theta|\mathbf{x}),$$

which defines the likelihood to be given in term of the parameter θ that maximizes the regular likelihood subject to $\tau(\theta) = \eta$. It can then be shown that (we will not be giving the proof here)

$$L^\star(\tau(\hat{\theta})) = L(\hat{\theta}),$$

where $\hat{\theta}$ is the MLE of the parameter θ, which mean that the invariance property holds in general. We illustrate this using a final example.

Example 15. Transformed DC level in white Gaussian noise. Non-bijection *As in the previous example we have the model*

$$x = A + w,$$

where A is unknown and $w \sim \mathcal{N}(0, \sigma^2)$. We want to find the MLE $\alpha = A^2$. The pdf parameterized by A is

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

but in this case the mapping from A to α is not one-to-one, and $A = \pm\sqrt{\alpha}$. This in turn forces us to consider two pdfs

$$\begin{aligned} p_1(x; \alpha) =& \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x-\sqrt{\alpha})^2}{\sigma^2}} \\ p_2(x; \alpha) =& \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x+\sqrt{\alpha})^2}{\sigma^2}}. \end{aligned}$$

Now the MLE $\hat{\alpha}$ is,

$$\begin{aligned} \hat{\alpha} =& \arg\max_{\alpha \geq 0} \{p_1(\mathbf{x}; \alpha), p_2(\mathbf{x}; \alpha)\} \\ =& \arg\max_{\alpha \geq 0} \{p(\mathbf{x}; -\sqrt{\alpha}), p(\mathbf{x}; \sqrt{\alpha})\} \\ =& \left[\arg\max_{\sqrt{\alpha} \geq 0} \{p(\mathbf{x}; -\sqrt{\alpha}), p(\mathbf{x}; \sqrt{\alpha})\}\right]^2 \\ =& \left[\arg\max_{A} p(\mathbf{x}; A)\right]^2 \\ =& \hat{A}^2, \end{aligned}$$

which means that the invariance property holds in this case as well.

Bayesian estimators

In this section we will present Bayesian estimators, which is based on a branch of statistics called Bayesian statistics. The methods we have discussed until now are referred to as classical estimators, and are based on frequentist statistics. The two branches of statistics have historically both had strong proponents and opponents, but the arguments have usually been of a philosophical character, and not about the inherent correctness of the theories. The short version is that Bayesian statistics lets us incorporate any prior information or plausibility about an estimate into the estimator. The frequentist approach relies on the observed data only. An engineer should always choose the estimator that makes most sense to the problem at hand.

Bayesian statistics

The major difference between Bayesian estimators and the estimators we have discussed until now, is that in the Bayesian framework the parameter we want to estimate is a random variable with its own probability density function. If the parameter we want to estimate is θ, we call its pdf, $p(\theta)$ [9], the *prior*.

The prior is meant to represent our *prior knowledge* about the parameter, which can be viewed as the plausibilities we assign to different parameter values. Let's illustrate this by an example.

[9] Sometimes you will see people writing $p(\theta|\eta)$, which means that the distribution of θ depends on another parameter which we call a *hyper-parameter*. The hyper-parameter η may in turn be described by its *hyper-prior*, $p(\eta)$, and so on.

Example 16. DC level in noise under prior knowledge *Again we assume the model*

$$x = A + w,\ w \sim \mathcal{N}(0;\sigma^2).$$

This time, we have analyzed the circuit, and based on this we are sure that the absolute value of A cannot exceed T. Other than that , A can be any value, so our prior information can be represented by the prior

$$p(A) = \begin{cases} 0 & A < -T \\ \frac{1}{2T} & -T \leq A \geq T \\ 0 & T < A \end{cases} \tag{1.24}$$

Based on this prior, we can suggest the following estimator,

$$\hat{A} = \begin{cases} -T & \bar{x} < -T \\ \bar{x} & -T \leq \bar{x} \geq T \\ T & T < \bar{x} \end{cases},$$

where $\bar{x}$ is the sample mean. As of now, we have no idea regarding the optimality of this estimator.

The real power of Bayesian statistics emerges when we apply Bayes theorem. If x is an observation whose pdf depends on θ, the parameter of interest, we can write

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}. \tag{1.25}$$

Here $p(x|\theta)$ [10] is the pdf of x given the parameter θ, and $p(\theta)$ is the prior. The right hand side denominator warrants further comment. The pdf $p(x)$ can be obtained using marginalization if we know $p(x|\theta)$ and $p(\theta)$,

$$\begin{aligned} p(x) &= \int p(x,\theta)d\theta \\ &= \int p(x|\theta)p(\theta)d\theta, \end{aligned}$$

in other words, $p(x)$ is the average, or expected value, of $p(x|\theta)$ wrt. θ. We can also interpret $p(x)$ as the probability of the data *given the entire model*, not just at a particular parameter value.

The most important part of the above result however, is the *posterior probability* of θ, $p(\theta|x)$. This is the updated information about θ after x has been observed. The prior, $p(\theta)$, is what we know about θ given that nothing has been observed. After seeing the observations, we know more about the parameter, and $p(\theta|x)$ is the representation of this new knowledge. In the next section we will use Bayesian framework to obtain new estimators, starting with the Bayesian mean squared error estimator.

[10] Please note that we now write $p(x|\theta)$ instead of $p(x;\theta)$ to reflect that θ is in fact a random variable, and not a fixed, even if unknown, parameter.

Bayesian MSE estimators

We will start our overview of Bayesian estimators with the Bayesian mean squared error estimator, as this is similar to the minimum variance framework we have been working within until now. Let $\hat{\theta}$ be an estimator of θ. Then the Bayesian mean square error of the estimator is,

$$\begin{aligned} \text{Bmse}(\hat{\theta}) &= E\left[(\theta - \hat{\theta})\right] && (1.26) \\ &= \int\int (\theta - \hat{\theta})^2 p(\mathbf{x},\theta)d\theta d\mathbf{x}. && (1.27) \end{aligned}$$

At first glance this looks very similar to our ill-fated attempt at obtaining an estimator using the MSE criterion. In the non-Bayesian framework this often leads to non-realizable estimators that are dependent on the parameter they are to estimate. This is not a problem in the Bayesian case, as $\text{Bmse}(\hat{\theta})$ doesn't depend on any particular value of θ, but rather the prior information in the form of the distribution of θ. Let us obtain the optimal estimator for this special case, that is, the estimator that minimizes $\text{Bmse}(\hat{\theta})$:

$$\begin{aligned} \hat{\theta} &= \arg\min_{\theta'} E\left[(\theta - \hat{\theta})\right] \\ &= \arg\min_{\theta'} \int\int (\theta - \theta')^2 p(\mathbf{x},\theta)d\theta d\mathbf{x} \\ &= \arg\min_{\theta'} \int\int (\theta - \theta')^2 p(\theta|\mathbf{x})p(\mathbf{x})d\theta d\mathbf{x} \\ &= \arg\min_{\theta'} \int \left[\int (\theta - \theta')^2 p(\theta|\mathbf{x})d\theta\right] p(\mathbf{x})d\mathbf{x}. \end{aligned}$$

The key observation here is that to minimize $\text{Bmse}(\hat{\theta})$, we need to minimize $\int(\theta - \theta')^2 p(\theta|\mathbf{x})d\theta$ for every $\mathbf{x}$. This is done by setting the

derivative wrt. $\hat{\theta}$ to zero,

$$\begin{aligned}&\frac{d}{d\hat{\theta}}\int(\theta-\hat{\theta})^2 p(\theta|\mathbf{x})d\theta\\ =&\int\frac{d}{d\hat{\theta}}(\theta-\hat{\theta})^2 p(\theta|\mathbf{x})d\theta\\ =&-2\int(\theta-\hat{\theta})p(\theta|\mathbf{x})d\theta=0,\end{aligned}$$

which can be solved to yield the Bayesian MSE estimator,

$$\hat{\theta}=\int\theta p(\theta|\mathbf{x})d\theta=E\left[\theta|\mathbf{x}\right]. \tag{1.28}$$

In other words – the Bayesian MSE estimator is the conditional expectation of the parameter θ given our observations $\mathbf{x}$.

Example 17. DC level in Gaussian noise with Gaussian prior *In this example we have,*

$$x=A+w,\ w\sim\mathcal{N}(0,\sigma^2),$$

where $A\sim\mathcal{N}(\eta,\nu^2)$. In other words – our prior belief about the DC level A is that it is most likely equal to or close to η, but we will not exclude the possibility that it can attain any value, although it is increasingly unlikely when further from η.

We have a vector of observations, $\mathbf{x}=[x[0],x[1],\ldots,x[N-1]]^T$. Since both $p(\mathbf{x}|A)$ and $p(A)$ are Gaussian distributions, we can in this special case find a closed form form expression for the posterior, $p(A|\mathbf{x})$. The derivation is long and tedious, and so we give the results here,

$$p(A|\mathbf{x})=\mathcal{N}\left(\frac{\frac{N}{\sigma^2}\bar{x}+\frac{1}{\nu^2}\eta}{\frac{N}{\sigma^2}+\frac{1}{\nu^2}},\frac{1}{\frac{N}{\sigma^2}+\frac{1}{\nu^2}}\right), \tag{1.29}$$

where $\bar{x}$ is the sample mean. The optimal estimator can now be found directly from Equation 1.29,

$$\hat{A}=\frac{\frac{N}{\sigma^2}\bar{x}+\frac{1}{\nu^2}\eta}{\frac{N}{\sigma^2}+\frac{1}{\nu^2}}, \tag{1.30}$$

as can the estimator variance,

$$Bmse(\hat{A})=\frac{1}{\frac{N}{\sigma^2}+\frac{1}{\nu^2}} \tag{1.31}$$

Several interesting observations can be made about this estimator. As $N\to\infty$, we see that the estimator converges to the sample mean, $\hat{A}=\bar{x}$. Likewise, if the variance of the prior, ν^2 goes towards zero, the estimator becomes $\hat{A}=\eta$ for any N. This means that our conviction about the "correct" value of A is so strong that the data be damned. Another interesting thing is what happens if we let the variance of the data, σ^2, become very large. Then the posterior mean goes to η and the variance to ν^2, in other words, the posterior converges to the prior. This means that if the observations are extremely noisy, our knowledge about A is unchanged.

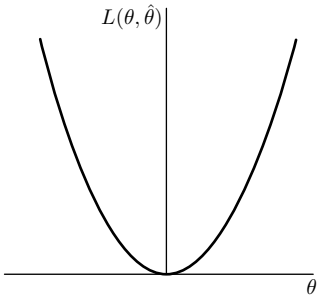


Figure 1.5: Squared error loss function.

General Bayesian estimators

In the previous section we derived the optimal Bayesian estimator for the MSE criterion. In contrast to the classical case, we can find the optimal estimator for any problem, although we may have to resort to numerical integration to obtain an answer.

In this section we will briefly present the theory of general Bayesian estimators. We start by defining a *loss function*, $L(\theta,\hat{\theta})$, that measure the loss incurred by the estimate $\hat{\theta}$ given that the true value is θ. Examples of loss functions are

- Squared error: $L(\theta,\hat{\theta}) = (\theta-\hat{\theta})^2$. This loss function was discussed in detail in the previous section.

- Absolute error: $L(\theta,\hat{\theta}) = |\theta-\hat{\theta}|$. An alternative if one finds that the squared error penalizes large errors too much.

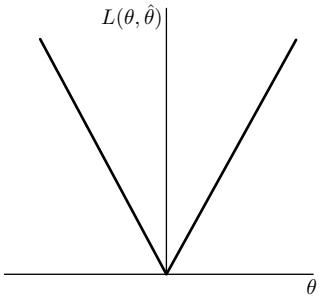


Figure 1.6: Absolute error loss function.

- Posterior mode (Also called the "hit-or-miss" error):

$$L(\theta,\hat{\theta}) = \begin{cases} 0, & \text{for } |\theta-\hat{\theta}| < \delta \\ 1, & \text{for } |\theta-\hat{\theta}| \geq \delta \end{cases}$$

  This loss function accepts any estimate within some margin δ with zero loss, while estimates outside this margin are given a loss of one. This is the basis of the maximum-a-posteriori estimator that we will discuss in the next section.

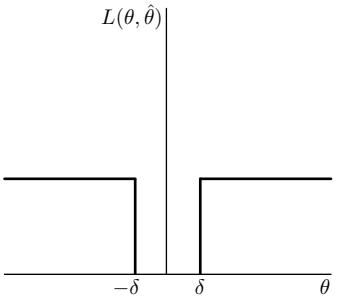


Figure 1.7: Posterior mode loss function.

It should be noted that there are no reasons in general for the loss function to be symmetric.

Having defined the loss function, we can move on to define the *Bayes risk*:

$$\begin{aligned}\mathcal{R} =& E\left[L(\theta,\hat{\theta})\right] \\ =& \int\int L(\theta,\hat{\theta})p(\mathbf{x},\theta)d\mathbf{x}d\theta \\ =& \int\int L(\theta,\hat{\theta})p(\theta|\mathbf{x})p(\mathbf{x})d\mathbf{x}d\theta \\ =& \int\left[\int L(\theta,\hat{\theta})p(\theta|\mathbf{x})d\theta\right]p(\mathbf{x})d\mathbf{x}\end{aligned}$$

Again, to minimize the Bayes risk, we must minimize $\int L(\theta,\hat{\theta})p(\theta|\mathbf{x})d\theta$ with respect to $\hat{\theta}$ for all $\mathbf{x}$ to find the optimal estimator.

The Maximum-a-posteriori estimator

In this section we will develop the maximum-a-posteriori (MAP) estimator from the posterior mode loss function. The MAP is one of the most use estimators, second only to the maximum likelihood. The posterior mode loss function is

$$L(\theta,\hat{\theta}) = \begin{cases} 0, & \text{for } |\theta-\hat{\theta}| < \delta \\ 1, & \text{for } |\theta-\hat{\theta}| \geq \delta \end{cases}$$

Minimizing the Bayes risk means we need to find the $\hat{\theta}$ that minimizes

$$\int L(\theta, \hat{\theta}) p(\theta|\mathbf{x}) d\theta,$$

for all $\mathbf{x}$. We start by plugging in $L(\theta, \hat{\theta})$ in to the integral,

$$\begin{aligned}
&\int_{-\infty}^{\infty} L(\theta, \hat{\theta}) p(\theta|\mathbf{x}) d\theta \\
=&\int_{-\infty}^{\hat{\theta}-\delta} p(\theta|\mathbf{x}) d\theta + \int_{\hat{\theta}+\delta}^{\infty} p(\theta|\mathbf{x}) d\theta \\
=&1 - \int_{\hat{\theta}-\delta}^{\hat{\theta}+\delta} p(\theta|\mathbf{x}) d\theta
\end{aligned}$$

We see that this expression is minimized when $\int_{\hat{\theta}-\delta}^{\hat{\theta}+\delta}$ is maximized. For any given δ we then have

$$\hat{\theta} = \arg\max_{\hat{\theta}'} \int_{\hat{\theta}'-\delta}^{\hat{\theta}'+\delta} p(\theta|\mathbf{x}) d\theta$$

If we let $\delta \to 0$, the estimator becomes

$$\begin{aligned}
\hat{\theta}_{MAP} &= \arg\max_{\theta} p(\theta|\mathbf{x}) \\
&= \arg\max_{\theta} \frac{p(\mathbf{x}|\theta) p(\theta)}{p(\mathbf{x})} \\
&= \arg\max_{\theta} p(\mathbf{x}|\theta) p(\theta)
\end{aligned}$$

We see that the MAP estimator is very similar to the maximum likelihood estimator, but with an added prior term.

Example 18. DC level in Gaussian noise with Gaussian prior. MAP estimate. *We have,*

$$x = A + w, \; w \sim \mathcal{N}(0, \sigma^2),$$

where the prior distribution for the unknown parameter is $A \sim \mathcal{N}(\eta, \nu^2)$.

Given a vector of observations, $\mathbf{x} = [x[0], x[1], \ldots, x[N-1]]^T$*, we want to find the MAP estimate,* $\hat{A}_{MAP}$*,*

$$\begin{aligned}
\hat{A}_{MAP} &= \arg\max_{A} p(\mathbf{x}|A) p(A) \\
&= \arg\max_{A} \left(\prod_{n=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x[n]-A)^2}{\sigma^2}} \right) \frac{1}{\sqrt{2\pi\nu^2}} e^{-\frac{1}{2}\frac{(A-\eta)^2}{\nu^2}}
\end{aligned}$$

To solve this optimization problem we set the derivative wrt. to A to zero and solve. This leads to the following closed form solution:

$$\hat{A}_{MAP} = \frac{\frac{N}{\sigma^2}\bar{x} + \frac{1}{\nu^2}\eta}{\frac{N}{\sigma^2} + \frac{1}{\nu^2}}$$

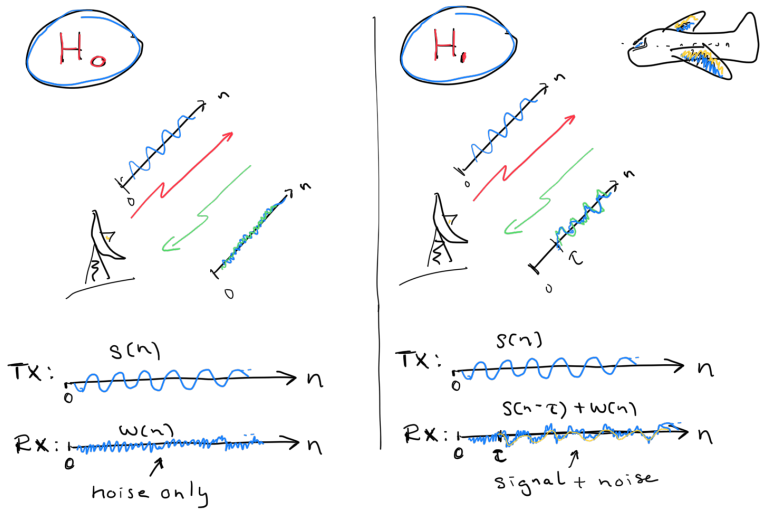
where $\bar{x}$ *is the sample mean. We see that this estimator is equivalent to the Bayesian MSE estimator. This is due to the symmetry of the Gaussian distributions, and not true in general.*

2
Detection Theory

Motivation

This part of the course deals with *signal detection* which, in its simplest form, involves taking a decision whether a signal is present or not in a set of noisy observations. Alternatively, we may consider the related problem of deciding which among two, or more, signals (or models) is present in the observed data set. Comparing to the case of signal (or parameter) estimation, where high-quality estimates are desired, we are here concerned with the case of weak signals buried in noise. That is, the signals to detect generally have low signal-to-noise (SNR) ratio. In particular, we are interested in designing signal processing algorithms or typical detection application areas such as digital communications, radar, sonar, etc. Below we sketch, two examples of the above mentioned signal detection problems: the radar system, and; the coherent communication system.

Figure 2.1: Radar system.



Radar System: Radar systems are concerned with detecting the presence of objects, and based on this knowledge take an appropriate action. A radar system, depicted in Fig. 2, consist of a transceiver, i.e., a transmitter (TX) and a receiver (RX), wherein the transmitted signal is a repeated pulse sequence $s[n]$ that is typically of sinusoidal-type. In case H_0 (left figure) there is no object blocking the transmitted signal path so the signal picked up by RX will be a noise-only sequence $w[n]$. However, in case H_1 (right figure) the signal path of the transmitted sequence $s(n)$ is blocked by an airplane and the signal bounces back and appears at RX after a delay τ. Thus, the signal at RX now consists of a noisy and delayed version of the transmitted signal, i.e., $s(n-\tau)+w(n)$. In

the estimation part of the course we were interested in problems like estimating the delay τ of the reflected signal as it provides information about the distance to the object. We were concerned by acquiring an estimate $\hat{\tau}$ that had all the good properties we could wish for, e.g., minimum variance and unbiasedness. In this part of the course we are more interested in the primal problem of detecting the presence of a signal, so-called *signal detection*. In other words, we ask questions of form: *Based on the collected RX data, decide whether an object is present or not, and provide a confidence measure along with the decision.* In the current radar example we are to decide on one out of two competing outcomes (or hypotheses): 1) the observed RX signal contains only noise $w(n)$, or; 2) the observed RX signal contains a noisy radar pulse $s(n) + w(n)$. Based on the answer we can consider an appropriate action, e.g., what measure to take if we detect an enemy aircraft. Of course our action has to be weighted by the confidence we have in our decision. For example, launching a missile towards an enemy aircraft is costly so before doing so we would like our decisions to be backed up with proper performance measures. Typical measures include the likelihood of the taking a decision that a target is present when it is actually note, and vice versa.

Communication System: Fig. 2 shows signals appearing in a binary phase-shift-keying (PSK) communication system wherein information, '0's or '1's, is transferred from a transmitter (TX) to a receiver (RX) over a noisy wireless channel. Signal '0' is transmitted using sequence $s_0[n] = A \sin 2\pi f n$ while sequence $s_1[n] = A \sin(2\pi f n + \pi)$ is used for transmitting signal '1'. The sequence picked up by RX consists of either $s_0[n]$ or $s_1[n]$ buried in noise $w[n]$. The received signal is then sent to a *detector* which decides whether a '0' or a '1' was sent. Comparing to the radar problem of detecting the presence of a target, we are here to discriminate between two mutually exclusive (known) signals whose amplitudes are small relative to the variance of the noise. In communication systems, the performance of the detector is usually quantified in terms of *probability of error*, i.e., the probability that '0' (or '1') is decided when the actual transmitted signal was '1' (or '0').

Chapter Summary

Detection theory applies statistical hypothesis testing to detecting signals in noise. This course considers three closely related problems: 1) Detection of constant(s) in noise; 2) Detection of deterministic signals (sequences) in noise, and; 3) detection of random signals (sequences) in noise. In all of the above cases we will assume that the probability density functions (PDFs) associated with random quantities are completely known.

We will study the two primary approaches to simple hypothesis testing (defined later), namely, the classic *Neyman-Pearson* (NP) detector and the *Bayes risk* detector. Both approaches rely on the

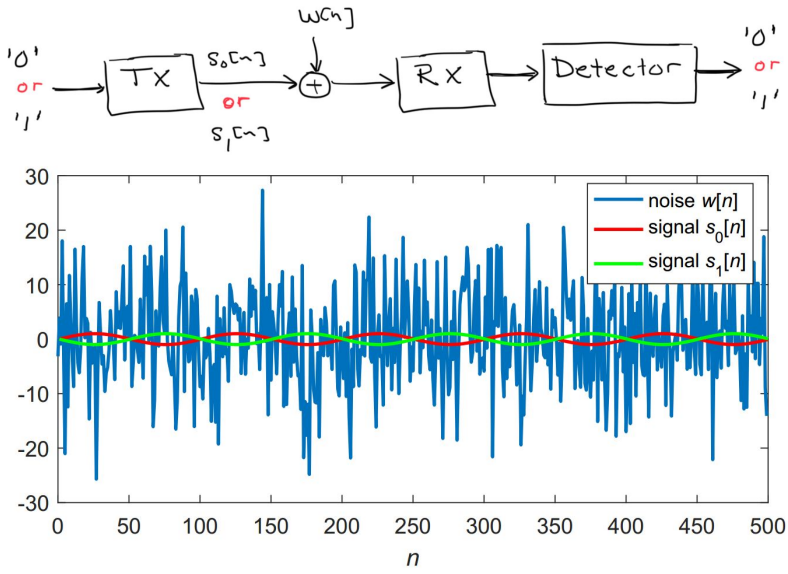


Figure 2.2: Communication signals in noise: System (top) and signals (bottom).

so-called *likelihood ratio test* (LRT), however, are used in different situations. The former approach is commonly employed when one wants to detect the presence of a target in noise (radar or sonar). The latter frequently shows up in digital communication where a receiver needs to decide whether a '0' or a '1' was transmitted. A special case of the latter approach is also known as the *minimum probability of error* (MPE) detector from which the *maximum likelihood* (ML) detector naturally follows. The choice between the two approaches depends on the willingness to incorporate prior knowledge about the probabilities of occurrence of various hypotheses as well as for the need to penalise wrong decisions.

(Binary) Hypothesis Testing

The problem of signal detection can be formulated mathematically as one of binary hypothesis testing. In a binary hypothesis testing problem, we are given data

$$\mathbf{x} = [x(0)\, x(1)\, \cdots\, x(N-1)]^{\mathrm{T}} \tag{2.1}$$

and must decide which of the two possible hypotheses best fit the data. That is, we need to make a decision from which distribution the data may have been sampled.

Let $\mathbf{x} \in \mathbb{R}^N$ be the random vector that contains the N variables (as above) and let the observation space be denoted by Ω, i.e., $\mathbf{x} \in \Omega$. Given the observation $\mathbf{x}$, we test the validity of the following two hypotheses:

$$\begin{aligned} \text{Null Hypothesis} \quad & H_0 : \quad \mathbf{x} \sim \mathcal{P}_0 \\ \text{Alternative} \quad & H_1 : \quad \mathbf{x} \sim \mathcal{P}_1 \end{aligned} \tag{2.2}$$

where $\mathcal{P}_0$ is the probability distribution of $\mathbf{x}$ given that H_0 is true and $\mathcal{P}_1$ is the probability distribution of $\mathbf{x}$ given that H_1 is true. If $\mathcal{P}_0$ and $\mathcal{P}_1$ each represents one single distribution, the above formulation is called a *simple* binary hypothesis testing. An example of simple binary hypothesis testing is the problem of detecting a signal with a constant, say, θ, (though possibly unknown) amplitude embedded in additive noise. Numerous other practical problems, such as quality control of a product and drug testing, can also be formulated as one of simple binary hypothesis testing. In most of these applications, the distributions $\mathcal{P}_0$ and $\mathcal{P}_1$ differ only by the value of a parameter characterizing the underlying distribution. Accordingly, the problem is simply to decide which of the two values this parameter is. Thus the binary hypothesis testing problem formulated above is to choose one of two possible values for the parameter that characterizes the observed random variables' distribution, namely,

$$\begin{aligned} H_0 &: \quad \theta = \theta_0 \\ H_1 &: \quad \theta = \theta_1 \end{aligned} \tag{2.3}$$

where θ is the parameter value, which is most commonly the mean, of the observation and $\theta_0 \neq \theta_1$.

Example 1: DC level in white Gaussian noise (WGN)
We wish to detect a known DC level, $A = 1$, in white noise. For this purpose we collect N noisy measurements $\mathbf{x} = [x(0)\, x(1) \cdots x(N-1)]^{\mathrm{T}}$. Our data then consists of $x(n) = w(n)$ for $n = 0, 1, \ldots, N-1$ under H_0 and $x(n) = A + w(n)$ for $n = 0, 1, \ldots, N-1$ under H_1. Formally we write the detection problem as

$$\begin{aligned} H_0 &: \qquad x(n) = w(n) \;\; n = 0, 1, \ldots, N-1 \\ H_1 &: \quad x(n) = 1 + w(n) \;\; n = 0, 1, \ldots, N-1 \end{aligned} \tag{2.4}$$

where $w(n)$ is WGN with variance σ^2. The above problem can also be cast as a test of the mean of a multivariate Gaussian PDF, reason being that $\mathbf{x} \overset{H_0}{\sim} N(\mathbf{0}, \sigma^2\mathbf{I})$ and $\mathbf{x} \overset{H_1}{\sim} N(\mathbf{1}, \sigma^2\mathbf{I})$, where '$\overset{H_i}{\sim}$' means 'under hypothesis H_i is distributed according to.'

Notation: The PDF under hypothesis H_i, $i \in \{0, 1\}$, is denoted by $p_i(\mathbf{x})$ or $p(\mathbf{x}; H_i)$.

Example 2: On-off keyed (OOK) communication
At times it is convenient to assign *prior* probabilities, $\pi_0 = P(H_0)$ and $\pi_1 = P(H_1)$ to the occurrences of H_0 and H_1. A prominent example arises in an on-off keyed (OOK) digital communication system where we transmit a '0' by sending a pulse with $A = 0$ and a '1' by sending a pulse with amplitude $A = 1$. Considering transmission takes place over an additive white Gaussian channel with variance σ^2, our received data then consists of $x(0) = w(0)$ under H_0 and $x(0) = 1 + w(0)$ under H_1. Given the the observation

$x(0)$, we wish to test if $A = 0$ or $A = 1$, or

$$\begin{aligned} H_0 &: \quad A = 0 \\ H_1 &: \quad A = 1 \end{aligned} \tag{2.5}$$

In an actual communication system, we send a continuous stream of 'o's and '1's and expect that either H_0 or H_1 to be true half of the time on average. That is, it makes sense to consider the hypotheses as random events with $P(H_0) = P(H_1) = 0.5$. The notation for the PDFs are $p(x(0)|H_0)$ and $p(x(0)|H_1)$ to be consistent with the standard notation of a conditional PDF.

Notation: The PDF under random hypothesis H_i, $i \in \{0, 1\}$, with *prior* probability $\pi_i = P(H_i)$, is denoted by $p_i(\mathbf{x})$ or $p(\mathbf{x}|H_i)$.

Our discussion here will focus on the *simple* binary hypothesis testing problem formulated in (2.3). We also assume that either the null hypothesis or the alternative is true. Specifically, let $\pi_0 = P(H_0)$ and $\pi_1 = P(H_1)$ be the *a priori* probabilities of H_0 and H_1 being true, respectively. Then,

$$\pi_0 + \pi_1 = 1 \tag{2.6}$$

The objective here is to decide whether H_0 or H_1 is true based on the observation of $\mathbf{x}$. A decision rule, i.e., a detection scheme, $\delta(\mathbf{x})$ essentially partitions Ω into two subspaces, or decision regions, Ω_0 and Ω_1. The subspace Ω_0 consists of all observations that lead to the decision that H_0 is true, while Ω_1 consists of all observations that lead to the decision that H_1 is true. In particular, for an observed data set $\{x(0), x(1), \ldots, x(N-1)\}$ we have

$$\Omega_0 = \{\mathbf{x} : \text{decide } H_0 \text{ or reject } H_1\} \tag{2.7}$$

$$\Omega_1 = \{\mathbf{x} : \text{decide } H_1 \text{ or reject } H_0\} \tag{2.8}$$

where Ω_1 is referred to as the *critical region*. For notational convenience, one may also define $\delta(\mathbf{x})$ as follows:

$$\delta(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in \Omega_0 \\ 1 & \text{if } \mathbf{x} \in \Omega_1 \end{cases} \tag{2.9}$$

The $\delta(\mathbf{x})$ defined in (2.9) is usually referred to as the *critical function* of a decision rule, or simply the decision rule. Before any decision criteria can be derived, the following constraints need be stated first.

$$\text{Prob}\{\delta(\mathbf{x}) = 0|H_0\} + \text{Prob}\{\delta(\mathbf{x}) = 1|H_0\} = 1 \tag{2.10a}$$

$$\text{Prob}\{\delta(\mathbf{x}) = 0|H_1\} + \text{Prob}\{\delta(\mathbf{x}) = 1|H_1\} = 1 \tag{2.10b}$$

The above constraints, (2.10a) and (2.10b), simply result from the assumptions that $\Omega_0 \cup \Omega_1 = \Omega$ and $\Omega_0 \cap \Omega_1 = \emptyset$. In other words, for every observation, an unambiguous decision must be made.

Example 3: DC level in white Gaussian noise (WGN) (cont.)
A reasonable decision rule for the problem outlined in Example 1 is to average the collected samples and compare the value obtained to a threshold λ so we can decide H_1 ($\delta(\mathbf{x}) = 1$) if $\bar{x} = \frac{1}{N}\sum_{n=0}^{N-1} x(n) > \lambda$. We know that $\bar{x}$ is the minimum variance unbiased (MVU) estimator for the a constant buried in a white Gaussian noise sequence, and we shall so expect that the detector performance improves as N increases. In other words, as $N \to \infty$, we have $\bar{x} \to 0$ under H_0 and $\bar{x} \to 1$ under H_1. Determining the function $\delta(\cdot)$ and the threshold λ is the central problem addressed in detection theory.

Example 4: DC level in white Gaussian noise (WGN) (cont.)
Assume we have access to a single sample $x(0)$ from Example 1, and the noise variance is set to $\sigma^2 = 0.5$. The PDFs under each hypothesis, $p_0(x) = p(x; H_0) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$ and $p_1(x) = p(x; H_1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-1)^2}$, are plotted in Fig. 2, together with a decision threshold $\lambda = 0.5$, and regions Ω_0 and Ω_1. We see that if H_1 ($\delta(\mathbf{x}) = 1$) is decided whenever $x(0) > 0.5$, we have to accept that the observed data may still be from H_0. The two shaded areas in Fig. 2, corresponds to the probabilities of accepting H_1 when H_1 is true, i.e., $P(H_1; H_1) = \text{Prob}\{\delta(x(0)) = 1 | H_1\}$, and when it is false, i.e., $P(H_1; H_0)\,\text{Prob}\{\delta(x(0)) = 1 | H_0\}$.

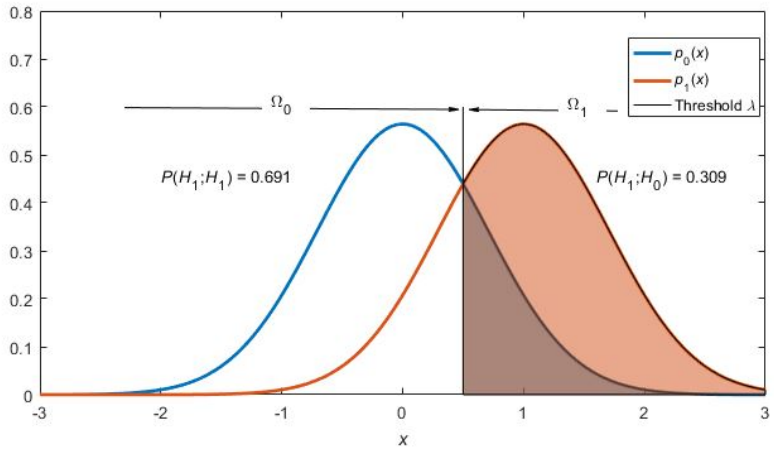


Figure 2.3: The PDFs under each hypothesis and decision threshold $\lambda = 0.5$.

From the simplified example illustrated in Fig. 2, we can conclude that for any decision rule $\delta(\cdot)$, there are four possible outcomes

1. decide H_0 when H_0 is true;
2. decide H_0 when H_1 is true;
3. decide H_1 when H_0 is true;
4. decide H_1 when H_1 is true;

and two types of error can occur

I. Type I error (false-alarm): Decide H_1 when H_0 is true. The probability of Type-I error, also termed *false-alarm rate*, is

$$\alpha = P_{FA} = \int_{\Omega_1} p_0(\mathbf{x}) d\mathbf{x} \tag{2.11}$$

II. Type II error (miss): Decide H_0 when H_1 is true. The probability of Type-II error is

$$P_m = \int_{\Omega_0} p_1(\mathbf{x}) d\mathbf{x} \tag{2.12}$$

Another quantity of concern is the probability of detection, which is often called *power* in signal detection literature, defined by

$$\beta = P_D = 1 - P_m = \int_{\Omega_1} p_1(\mathbf{x}) d\mathbf{x} \tag{2.13}$$

In the above equations, $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$ are, respectively, the PDF of $\mathcal{P}_0$ and $\mathcal{P}_1$. We note that the above calculations require us to evaluate N-dimensional integrals, which in general we be an onerous task. As we shall see later, we can often make use of simplified test statistics and monotone (logarithmic) transformations to simplify our calculations.

Example 5: DC level in white Gaussian noise (WGN) (cont.)
We are given a single sample $x(0)$ in Example 1 together with the following decision rule: Decide H_1 if $x(0) > 0.5$; H_0 otherwise. Calculate the probability of false alarm α and probability of missed detection P_m, and probability of detection β.

$$\alpha = \int_{\Omega_1} p_0(x) dx = \text{Prob}\{x(0) > 0.5; H_0\} = \int_{0.5}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx = Q(0.5) = 0.3085$$

$$\begin{aligned} P_m &= \int_{\Omega_0} p_1(x) dx = \text{Prob}\{x(0) < 0.5; H_1\} = \int_{-\infty}^{0.5} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}[x-1]^2} dx \\ &= \Phi(-0.5) = 1 - Q(-0.5) = Q(0.5) = 0.3085 \end{aligned}$$

$$\begin{aligned} \beta &= \int_{\Omega_1} p_1(x) dx = \text{Prob}\{x(0) > 0.5; H_1\} = \int_{0.5}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}[x-1]^2} dx = Q(-0.5) \\ &= 1 - P_m = 0.6915 \end{aligned}$$

where $Q(x) = \int_x^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}t^2\right) dt$ is the tail distribution function or the complementary cumulative distribution function.

In the remaining sections, we shall look at various approaches to arrive at a decision threshold keeping above error measures in mind. Among the various decision criteria, Bayes and Neyman-Pearson are most popular. Detection schemes may also be classified as parametric and non-parametric detectors. The latter type of detectors does not rely on explicit knowledge of the PDF of the data. In this course we will only consider parametric detectors.

Remark: In a more general framework, one may consider the case for which at least one of the two hypotheses is characterized by a

family of distributions (e.g., parametrized by an unknown constant DC level A). These problems are referred to as *composite* hypothesis testing. A good example of the composite hypothesis testing is that of a radar communication system for which one is testing signal present versus signal absent. If the test is implemented by measuring the amplitude of the received signal embedded in noise whose distribution has zero mean, then it can be formulated as

$$\begin{aligned} H_0 &: \quad \theta = 0 \\ H_1 &: \quad \theta > 0 \end{aligned} \tag{2.14}$$

where θ is the mean value of the observed random variable.

Bayesian Detection

The objective of the Bayes criterion is to minimize the so-called *Bayes risk* which is, again, defined as the expected value of the cost. To derive a Bayes detection rule, the costs of making decisions need to be defined first. Let the cost of deciding i when j is true be denoted by $C_{ij} = (i, j = 0 \text{ or } 1)$. For example,

$$C_{01} = \text{cost of choosing } H_0 \text{ when } H_1 \text{ is true.}$$

$$C_{10} = \text{ cost of choosing } H_1 \text{ when } H_0 \text{ is true.}$$

In addition, assume that the prior probabilities π_0 and π_1 are known. The *Bayes risk*, $\mathcal{R}$, is then evaluated as follows,

$$\begin{aligned} \mathcal{R} \triangleq E\{C\} = \pi_0\{C_{00}\,\text{Prob}\{\delta(\mathbf{x}) = 0|H_0\} + C_{10}\,\text{Prob}\{\delta(\mathbf{x}) = 1|H_0\}\} + \\ + \pi_1\{C_{01}\,\text{Prob}\{\delta(\mathbf{x}) = 0|H_1\} + C_{11}\,\text{Prob}\{\delta(\mathbf{x}) = 1|H_1\}\} \end{aligned} \tag{2.15}$$

Under the assumption that the costs of making correct decisions are less than those of making incorrect decisions, i.e.,

$$C_{01} - C_{11} > 0 \qquad \text{and} \qquad C_{10} - C_{00} > 0,$$

the detector that minimizes the Bayes risk decides H_1 if

$$L(\mathbf{x}) \triangleq \frac{p_1(\mathbf{x})}{p_0(\mathbf{x})} > \frac{\pi_0}{1-\pi_0}\frac{C_{10} - C_{00}}{C_{01} - C_{11}} = \lambda. \tag{2.16}$$

In particular,

$$L(\mathbf{x}) = \frac{p_1(\mathbf{x})}{p_0(\mathbf{x})} \begin{cases} \geq \lambda & \Rightarrow H_1 \\ < \lambda & \Rightarrow H_0 \end{cases} \tag{2.17}$$

A decision rule characterized by the likelihood ratio and a threshold as in (2.17) is referred to as a likelihood ratio test (LRT). The $L(\mathbf{x})$ as defined in (2.17) is called the *test statistic* for the decision rule (or the detection scheme). A Bayes detection scheme is always an LRT.

For a simple binary hypothesis testing problem formulated in (2.3), if the underlying PDF belongs to the exponential family of

distributions, *the likelihood ratio is always a function of the sufficient statistic*. In particular, if

$$p_i(\mathbf{x}) = \{\exp[c(\theta_i)T(\mathbf{x}) + d(\theta_i) + S(\mathbf{x})]\} I_A(x) \quad i = 0, 1$$

Then,

$$L(\mathbf{x}) = \exp[(c(\theta_1) - c(\theta_0))T(\mathbf{x}) + (d(\theta_1) - d(\theta_0))]$$

which clearly shows that $L(\mathbf{x})$ is a function of $T(\mathbf{x})$. In this case, it is always more convenient to consider the *log-likelihood ratio*, namely,

$$\ln L(\mathbf{x}) = (c(\theta_1) - c(\theta_0))T(\mathbf{x}) + (d(\theta_1) - d(\theta_0))$$

To further illustrate this, recall that the sample mean is the sufficient statistic for estimating the mean of a set of observations. We thus expect that, in a simple binary hypothesis testing problem which tests the mean value of the observations, the test statistic is the sample mean (as we shall see in the subsequent sections).

Depending on how the costs and the *a priori* probabilities of the two hypotheses are defined, the Bayes detector can be realized in different ways. For example, if we assume that the costs of making decisions are

$$C_{00} = C_{11} = 0$$

and

$$C_{10} = C_{01} = 1 \ ,$$

then (2.15) is simply the probability of error (P_e). In particular, from (2.15),

$$P_e = \pi_0 \, \text{Prob}\{\delta(\mathbf{x}) = 1 | H_0\} + \pi_1 \, \text{Prob}\{\delta(\mathbf{x}) = 0 | H_1\} \tag{2.18}$$

In this case, the Bayesian detection is the *minimum probability of error* (MPE) detection scheme and it is performed by comparing the likelihood ratio to a threshold which is simply the ratio of two a priori probabilities, namely, $\frac{\pi_0}{\pi_1}$.

A variation of the MPE detection is the so-called MAP (maximum a posteriori) detector that renders the minimum probability of error by choosing the hypothesis with the maximum a posteriori probability. Note that the MPE detection decides H_1 is true if

$$\frac{p_1(\mathbf{x})}{p_0(\mathbf{x})} > \frac{\pi_0}{\pi_1} = \lambda \tag{2.19}$$

Since all terms in the ratios on both sides of the inequality are non-negative (positive for the denominators), the above equation can be written as

$$p_1(\mathbf{x})\pi_1 > p_0(\mathbf{x})\pi_0$$

Normalizing both sides of the inequality by the unconditional PDF, i.e., $p(\mathbf{x})$, and employ Bayes rule would show that the MPE detection decides H_1 is true if

$$\text{Prob}\{H_1 | \mathbf{x}\} > \text{Prob}\{H_0 | \mathbf{x}\} \tag{2.20}$$

The above is just a comparison of the *a posteriori* probabilities, and it is equivalent to (2.19).

Along the line of MPE detection, (2.19), if we further assume that the two hypotheses are equally likely, i.e., $\pi_0 = \pi_1$, the threshold is simply one and the test is just comparing the two conditional likelihood functions $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$. Thus we decide H_0 is true if $p_0(\mathbf{x}) > p_1(\mathbf{x})$, and decide H_1 is true otherwise. The resulting (Bayes) detection scheme is thus termed maximum-likelihood (ML) detector.

Example: On-off keyed (OOK) communication

Consider a communication system that transmit either $s_0(n) = 0$ for a symbol 'o' or $s_1(n) = A$ for a symbol '1'. It is reasonable to assume that the transmitted symbols are equally likely. We now have the detection problem

$$\begin{aligned} H_0 &: \quad x(n) = w(n) \;\; n = 0, 1, \ldots, N-1 \\ H_1 &: \quad x(n) = A + w(n) \;\; n = 0, 1, \ldots, N-1 \end{aligned} \tag{2.21}$$

where $A > 0$ and $w[n]$ are i.i.d. zero-mean Gaussian variables with known variance σ^2. Since both hypotheses are equally likely, i.e., $\pi_0 = \pi_1 = 1/2$, the MPE detection rule is given by (2.19) with $\lambda = 1$. The likelihood ratio is given by

$$\begin{aligned} L(\mathbf{x}) &= \frac{p_1(x(0), x(1), ..., x(N-1))}{p_0(x(0), x(1), ..., x(N-1)} = \prod_{n=1}^{N-1} \frac{p_1(x(n))}{p_0(x(n))} \\ &= \frac{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}(x(n)-A)^2\right]}{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1} x^2(n)\right]}. \end{aligned} \tag{2.22}$$

Inserting this result into (2.19) and taking the logarithm of both sides, we decide H_1 if

$$\ln L(\mathbf{x}) = -\frac{1}{2\sigma^2}\left(-2A\sum_{n=0}^{N-1} x(n) + NA^2\right) > 0$$

or, equivalently if

$$\bar{x} = \frac{1}{N}\sum_{n=0}^{N-1} x(n) > \frac{A}{2}$$

To determine P_e we need to evaluate (2.18), i.e.,

$$\begin{aligned} P_e &= \pi_0 \,\text{Prob}\{\delta(\mathbf{x}) = 1 | H_0\} + \pi_1 \,\text{Prob}\{\delta(\mathbf{x}) = 0 | H_1\} \\ &= \frac{1}{2}\left[\text{Prob}\{\bar{x} > A/2 | H_0\} + \text{Prob}\{\bar{x} < A/2 | H_1\}\right] \end{aligned} \tag{2.23}$$

To be able to proceed we note that $\bar{x} \overset{H_0}{\sim} N(0, \sigma^2/N)$ and $\bar{x} \overset{H_1}{\sim} N(A, \sigma^2/N)$,

which gives us the final expression

$$\begin{aligned} P_e &= \frac{1}{2}\int_{A/2}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2/N}} e^{-\frac{1}{2\sigma^2/N}x^2} dx + \frac{1}{2}\int_{-\infty}^{A/2} \frac{1}{\sqrt{2\pi\sigma^2/N}} e^{-\frac{1}{2\sigma^2/N}[x-A]^2} dx \\ &= \frac{1}{2}\left[Q\left(\frac{A/2}{\sqrt{\sigma^2/N}}\right) + \left(1 - Q\left(\frac{A/2 - A}{\sqrt{\sigma^2/N}}\right)\right)\right] \\ &= Q\left(\sqrt{\frac{NA^2}{4\sigma^2}}\right) \end{aligned} \tag{2.24}$$

where $Q(-x) = 1 - Q(x)$ was used. We see that P_e decreases monotonically with NA^2/σ^2. In other words, various ways of decreasing P_e include: 1) increase the number of samples N constituting a symbol, and/or; 2) increase the ratio A^2/σ^2.

Neyman-Pearson Detection

In the Bayes risk detector, the detection rule assumed $\pi_i = P(H_i)$ known in advance. In many applications, e.g., radar and sonar systems, this prior information is not known. In particular, in applications where the hypothesis testing is related to two disjoint alternatives, e.g., signal (or target) is present or not, a different approach is often needed.

Section 1 listed the four possible outcomes related to a binary hypothesis testing problem. To each of these outcomes we can assign a probability

1. $P_R = \text{Prob}\{\text{decide } H_0 \text{ when } H_0 \text{ is true}\} = \text{Prob}\{\delta(\mathbf{x}) = 0; H_0\}$ (rejection);
2. $P_M = \text{Prob}\{\text{decide } H_0 \text{ when } H_1 \text{ is true}\} = \text{Prob}\{\delta(\mathbf{x}) = 0; H_1\}$ (miss);
3. $P_{FA} = \text{Prob}\{\text{decide } H_1 \text{ when } H_0 \text{ is true}\} = \text{Prob}\{\delta(\mathbf{x}) = 1; H_0\}$ (false alarm);
4. $P_D = \text{Prob}\{\text{decide } H_1 \text{ when } H_1 \text{ is true}\} = \text{Prob}\{\delta(\mathbf{x}) = 1; H_1\}$ (detection);

Since decisions are unambiguous we have $P_D = 1 - P_M$ and $P_{FA} = 1 - P_R$; essentially leaving us with two degrees of freedom to evaluate the test. From Fig. 1, we see that P_{FA} and P_D depend on each other through decision region Ω_1. After recomputing the values of P_{FA} and P_D in Example 5 for various choices of threshold λ (or by inspection of Fig. 1), we make the following observation: as λ increases, P_{FA} decreases and P_D decreases. Ideally we would like $P_D = 1$ and $P_{FA} = 0$, but in practice we have to live with a compromise solution.

Basically we are interested in how to choose Ω_1 keeping the above trade-off in mind. This is done by maximizing P_D subject to given maximum allowed P_{FA}. The principle of the Neyman-Pearson criterion is founded on the *Neyman-Pearson Lemma* stated below:

Neyman-Pearson Lemma Consider a simple hypothesis testing problem formulated in (2.3). Let $\delta_{\lambda^*}(x)$ be the critical function of a likelihood ratio test with a threshold λ^* as defined in (2.17). Let α^* and β^* be the false-alarm rate and power, respectively, of the test $\delta_{\lambda^*}(x)$. Let $\delta_\lambda(x)$ be another arbitrary likelihood ratio test with threshold λ and its false-alarm rate and power be, respectively, α and β. If $\alpha \leq \alpha^*$, then $\beta \leq \beta^*$.

The Neyman-Pearson Lemma showed that if it is desired to increase the power of an LRT, one must also accept the consequence of an increased false-alarm rate. As such, the Neyman-Pearson detection criterion is aimed to maximize the power under the constraint that the false-alarm rate be upper bounded by, say, α_0. The Neyman-Pearson detector can be derived by first defining a cost function

$$J = (1 - \beta) + \lambda(\alpha - \alpha_0) \tag{2.25}$$

It can be shown that

$$J = \lambda(1 - \alpha_0) + \int_{\Omega_0} [p_1(\mathbf{x})d\mathbf{x} - \lambda p_0(\mathbf{x})]d\mathbf{x} \,, \tag{2.26}$$

and an LRT will minimize J for any positive λ. In particular,

$$L(\mathbf{x}) \triangleq \frac{p_1(\mathbf{x})}{p_0(\mathbf{x})} \begin{cases} \geq \lambda & \Rightarrow H_1 \\ < \lambda & \Rightarrow H_0 \end{cases}$$

To satisfy the constraint and to maximize the power, choose λ so that $\alpha = \alpha_0$, namely,

$$P_{FA} = \alpha = \int_{L(\mathbf{x})>\lambda} p_0(\mathbf{x})d\mathbf{x} = \alpha_0$$

where the threshold λ is determined by solving the above equation, i.e.,

$$\lambda = P_{FA}^{-1}(\alpha_0)$$

In summary, the optimal detector is still a likelihood ratio test. The Bayes detector and Neyman-Pearson detector differ in how threshold λ is chosen. The Neyman-Pearson detector is known to be the most powerful detector for the problem of detecting a constant signal in noise. The implementation of the Neyman-Pearson does not require explicit knowledge of the prior probabilities and costs of decisions. However, as is the case for Bayes detector, evaluation of the likelihood ratio still requires exact knowledge of the PDF of $\mathbf{x}$ under both hypotheses. Finally, note that finding the optimal threshold λ such that $P_{FA} = \alpha_0$ is in general a very difficult problem. However, we can exploit test statistics and logarithmic transformations to simplify the problem as we shall see next.

Detection of DC level in Gaussian noise

Consider a signal detection problem formulated as follows:

$$\begin{aligned} H_0 &: \quad x(n) = w(n) \;\; n = 0, 1, \ldots, N-1 \\ H_1 &: \quad x(n) = A + w(n) \;\; n = 0, 1, \ldots, N-1 \end{aligned} \tag{2.27}$$

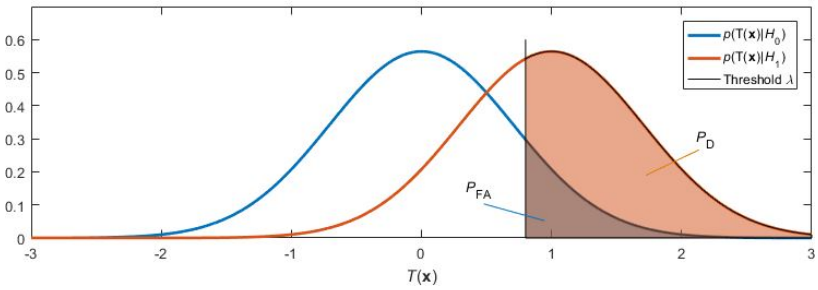


Figure 2.4: Probability density function of the test statistic.

where $A > 0$ is a deterministic constant and $w[n]$ are i.i.d. zero-mean Gaussian variables with known variance σ^2. The NP detector decides H_1 if

$$\frac{p_1(\mathbf{x})}{p_0(\mathbf{x})} = \frac{p(\mathbf{x}; H_1)}{p(\mathbf{x}; H_0)} > \lambda$$

or

$$\frac{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}(x(n)-A)^2\right]}{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}x^2(n)\right]} > \lambda \tag{2.28}$$

After taking the logarithm on both sides and rearranging the terms we have

$$\frac{A}{\sigma^2}\sum_{n=0}^{N-1} x(n) > \ln\lambda + \frac{NA^2}{2\sigma^2} \tag{2.29}$$

Since $A > 0$ we can arrive at the final form

$$\sum_{n=0}^{N-1} x(n) > \frac{\sigma^2}{A}\ln\lambda + \frac{NA}{2} = \lambda' \tag{2.30}$$

We see that the LRT is characterized by comparing a test statistic

$$T(\mathbf{x}) \triangleq \sum_{n=0}^{N-1} x(n) \tag{2.31}$$

with the threshold λ'. If $T(\mathbf{x}) \geq \lambda'$, then H_1 is said to be true, otherwise, H_0 is said to be true. Note that $T(\mathbf{x}) = \sum_{i=0}^{N-1} x_i$ turns out to be the sufficient statistic for estimating A. However, this should not be surprising as detection of a constant signal in noise is a dual problem of estimating the mean of the observations.

Before we proceed to determine the detection performance, we note that under the i.i.d. Gaussian assumption, the test statistic is clearly a Gaussian random variable. In particular we have $\mathrm{E}\{T(\mathbf{x})|H_0\} = 0$, $\mathrm{E}\{T(\mathbf{x})|H_1\} = NA$, and $\mathrm{var}\{T(\mathbf{x})|H_0\} = \mathrm{var}\{T(\mathbf{x})|H_1\} = N\sigma^2$. That is, $T(\mathbf{x}) \overset{H_0}{\sim} N(0, N\sigma^2)$ and $T(\mathbf{x}) \overset{H_1}{\sim} N(NA, N\sigma^2)$. An example PDF of $T(\mathbf{x})$ is plotted in Fig. 2. We can use Fig. 2 to illustrate the Neyman-Pearson Lemma. The shaded area under $p_1(T(\mathbf{x})|H_1)$ is the value of power while the shaded area under $p_0(T(\mathbf{x})|H_0)$ is the false-alarm rate. It is seen that if the threshold is moved to the left, the power increases and so does the false-alarm rate.

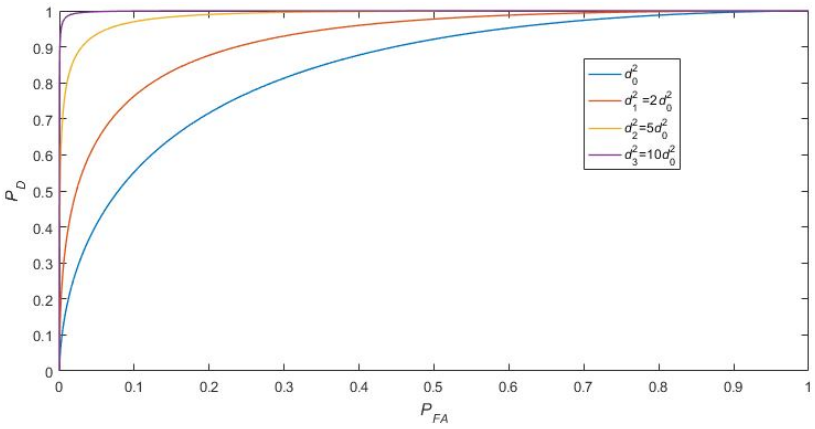


Figure 2.5: ROCs for various SNRs ($d_0 = A^2/\sigma^2$) of the example of DC level in AWGN.

The false-alarm rate and power are given by

$$\alpha = P_{FA} = \text{Prob}\{T(\mathbf{x}) > \lambda'; H_0\} = \int_{\lambda'}^{\infty} \frac{1}{\sqrt{2\pi N\sigma^2}} e^{-\frac{1}{2N\sigma^2}x^2} dx = Q\left(\frac{\lambda'}{\sqrt{N\sigma^2}}\right) \tag{2.32}$$

and

$$\beta = P_D = \text{Prob}\{T(\mathbf{x}) > \lambda'; H_1\} = \int_{\lambda'}^{\infty} \frac{1}{\sqrt{2\pi N\sigma^2}} e^{-\frac{1}{2N\sigma^2}[x-NA]^2} dx = Q\left(\frac{\lambda' - NA}{\sqrt{N\sigma^2}}\right) \tag{2.33}$$

For the Neyman-Pearson detector, the threshold λ' is determined by the constraint on the false-alarm rate, i.e.,

$$\lambda' = \sqrt{N\sigma^2} Q^{-1}(\alpha)$$

Combining (2.32) and (2.33) yields a relation between the false-alarm rate and power, namely,

$$\beta = Q\left(Q^{-1}(\alpha) - \sqrt{\frac{NA^2}{\sigma^2}}\right) \tag{2.34}$$

Remarks:

1. It can be seen from (2.33) that as the sample size N increases, P_{FA} decreases and P_D increases, which corroborates our earlier discussion. In fact, $\lim_{N\to\infty} P_D = 1$

2. Define $d^2 \stackrel{\Delta}{=} N\frac{s^2}{\sigma^2}$, which can be taken as the signal-to-noise ratio (SNR). From (2.34), one can see that $\lim_{d\to\infty} \beta = 1$.

The Neyman-Pearson detector can be further characterized by the receiver-operation-curve (ROC) shown in Fig. 2, which is a plot of (2.34), parameterized by d, the SNR.

It should be noted that all continuous LRT's have ROCs that are above the line of $\alpha = \beta$ and are concave downward. In addition, the slope of the line tangent to a point on a ROC curve is the value of the N-P optimal threshold λ required to achieve the prescribed value of α and β [1].

[1]

Deterministic Signals in Gaussian Noise

In this section, we study the more general case of detecting known signals/sequences in Gaussian noise. We will use the NP criterion whenever the power of the test β is to be maximized subject to a constraint on the false alarm α. Whenever prior information of the hypotheses, in terms of π_0 and π_1, can be exploited, Bayes risk may be employed. As before, both approaches rely on the evaluating the LRT but the decision is made upon using different thresholds. We shall see that the test statistic that is used is linear in the data, which can be seen as a generalization of the result we had for the constant in AWGN, wherein the test statistics was the sample mean.

Signal in white Gaussian noise

We now consider the following detection problem

$$\begin{aligned} H_0 &: \quad x(n) = w(n) \;\; n = 0, 1, \ldots, N-1 \\ H_1 &: \quad x(n) = s(n) + w(n) \;\; n = 0, 1, \ldots, N-1 \end{aligned} \tag{2.35}$$

where signal $s(n)$ is assumed known and $w(n)$ is WGN with variance σ^2. Examples of sequences include $s[n] = A$ discussed earlier and $s[n] = A\cos 2\pi f n$.

The optimal detector is

$$L(\mathbf{x}) \triangleq \frac{p_1(\mathbf{x})}{p_0(\mathbf{x})} \begin{cases} \geq \lambda & \Rightarrow H_1 \\ < \lambda & \Rightarrow H_0 \end{cases}$$

where

$$\begin{aligned} L(\mathbf{x}) &= \frac{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}(x(n)-s(n))^2\right]}{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}x^2(n)\right]} \\ &= \exp\left[-\frac{1}{2\sigma^2}\left\{\sum_{n=0}^{N-1}(x(n)-s(n))^2 - \sum_{n=0}^{N-1}x^2(n)\right\}\right] \end{aligned} \tag{2.36}$$

In terms of the log-likelihood ratio, we have

$$\ln L(\mathbf{x}) \begin{cases} \geq \ln\lambda & \Rightarrow H_1 \\ < \ln\lambda & \Rightarrow H_0 \end{cases}$$

where

$$\ln L(\mathbf{x}) = \frac{1}{\sigma^2}\sum_{n=0}^{N-1} x(n)s(n) - \frac{1}{2\sigma^2}\sum_{n=0}^{N-1} s^2(n) \tag{2.37}$$

where the term $E_s = \sum_{n=0}^{N-1} s^2(n)$ is recognized as the signal energy. Since $s(n)$ is known and not a function of the data, we may rearrange the terms and incorporate E_s and σ^2 into the decision threshold. Consequently, we decide H_1 if

$$T(\mathbf{x}) = \sum_{n=0}^{N-1} x(n)s(n) > \sigma^2\ln\lambda + \frac{1}{2}\sum_{n=0}^{N-1} s^2(n) = \lambda' \tag{2.38}$$

In summary, the detector evaluates a test statistics $T(\mathbf{x}) = \sum_{n=0}^{N-1} x(n)s(n)$ which is a function of the observed data, and compares it to a threshold λ'. In case of an NP detector, this threshold is determined by ensuring the constraint on the false alarm rate. For this purpose, it is fairly straightforward to show that

$$T(\mathbf{x}) \overset{H_0}{\sim} N(0, \sigma^2 E_s)$$

and

$$T(\mathbf{x}) \overset{H_1}{\sim} N(E_s, \sigma^2 E_s)$$

with $E_s = \sum_{n=0}^{N-1} s^2(n)$, and solve for the optimal threshold (see Problem 5).

The false-alarm rate and power are obtained as before, i.e.,

$$\alpha = P_{FA} = \text{Prob}\{T(\mathbf{x}) > \lambda'; H_0\} = \int_{\lambda'}^{\infty} \frac{1}{\sqrt{2\pi E_s \sigma^2}} e^{-\frac{1}{2\sigma^2 E_s} x^2} dx = Q\left(\frac{\lambda'}{\sqrt{\sigma^2 E_s}}\right) \tag{2.39}$$

and

$$\beta = P_D = \text{Prob}\{T(\mathbf{x}) > \lambda'; H_1\} = \int_{\lambda'}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2 E_s}} e^{-\frac{1}{2\sigma^2 E_s}[x - E_s]^2} dx = Q\left(\frac{\lambda' - E_s}{\sqrt{\sigma^2 E_s}}\right) \tag{2.40}$$

For the Neyman-Pearson detector, the threshold λ' is determined by the constraint on the false-alarm rate, i.e.,

$$\lambda' = \sqrt{\sigma^2 E_s} Q^{-1}(\alpha)$$

Combining (2.32) and (2.33) yields a relation between the false-alarm rate and power, namely,

$$\beta = Q\left(Q^{-1}(\alpha) - \sqrt{\frac{E_s}{\sigma^2}}\right) \tag{2.41}$$

Remarks:

1. The test statistic $T(\mathbf{x}) = \sum_{n=0}^{N-1} x(n)s(n)$ is referred to as *correlator* since we weight the data samples according to the actual signal values. Fig. 2(a) depicts the correlator implementation.

2. If we divide both sides of (2.38) with E_s, we can rewrite the decision rule in vector form as

$$\frac{\mathbf{s}^T \mathbf{x}}{\mathbf{s}^T \mathbf{s}} > \frac{\lambda'}{\mathbf{s}^T \mathbf{s}} = \lambda''$$

   with $\mathbf{s} = [s(0)\, s(1) \cdots s(N-1)]^T$ and $\mathbf{x} = [x(0)\, x(1) \cdots x(N-1)]^T$. Knowing that the orthogonal projection of $\mathbf{x}$ onto $\mathbf{s}$ is given by $P_{\mathbf{s}}(\mathbf{x}) = \frac{\mathbf{s}^T \mathbf{x}}{\mathbf{s}^T \mathbf{s}} \mathbf{s}$, we can conclude that LRT test only depends on the projection of the data onto the signal subspace. Other directions do not influence the decision. The projection interpretation of the correlator is shown in Fig. 2(b).

3. The correlator $T(\mathbf{x}) = \sum_{n=0}^{N-1} x(n)s(n)$ may be interpreted as the $(N-1)$th output of an FIR filter with impulse response is $h(n) =$

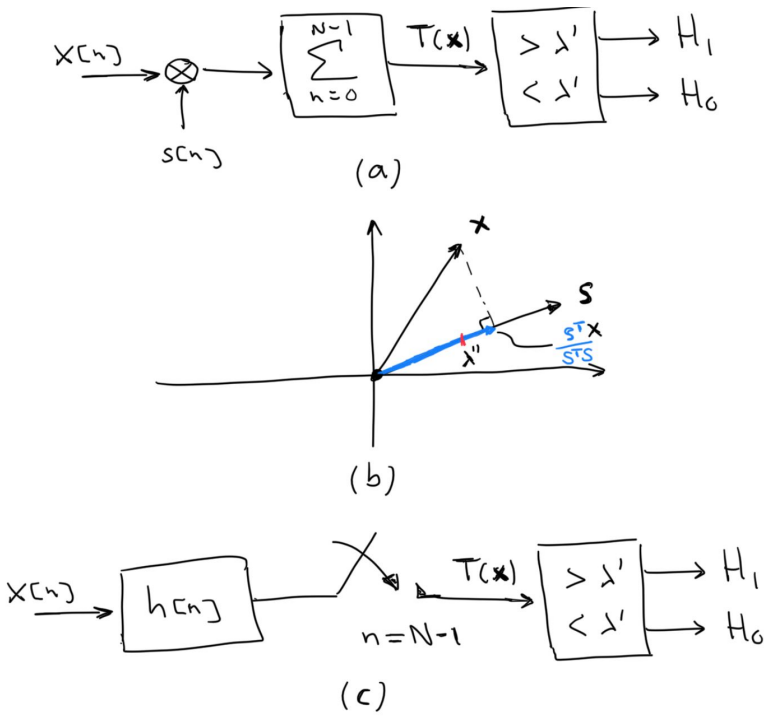


Figure 2.6: Detection of known sequence $s[n]$ in white Gaussian noise: (a) Correlator implementaion, (b) projection implementation, (c) matched filter implementation.

$s(N-1-n)$. Since the output $y(n) = \sum_{k=0}^{n} x(k)h(n-k) = \sum_{k=0}^{n} x(k)s(N-1-(n-k))$, we have $y(N-1) = \sum_{k=0}^{N-1} x(k)s(k)$. This implementation is referred to as the the *matched filter detector* and is depicted in Fig. 2(c).

4. The matched filter *maximizes the SNR at the output of an FIR filter*, where the maximum output SNR is defined given by $\eta_{\max} = \|\mathbf{s}\|^2/\sigma^2 = E_s/\sigma_2$.

5. The detection performance in (2.41) *is only effected by the signal energy* $E_s = \|\mathbf{s}\|^2 = \sum_{n=0}^{N-1} s^2(n)$ *and not by the particular signal shape.*

Multiple signals in white Gaussian noise

We will now consider the slightly more general case where the transmitted signal is non-zero under both hypotheses.

$$\begin{aligned} H_0: &\quad x(n) = s_0(n) + w(n) \quad n = 0, 1, \ldots, N-1 \\ H_1: &\quad x(n) = s_1(n) + w(n) \quad n = 0, 1, \ldots, N-1 \end{aligned} \tag{2.42}$$

where signal $s_0(n)$ and $s_1(n)$ are assumed known and $w(n)$ is WGN with variance σ^2. The detection problem trivially reduces to that of (2.35) whenever $s_0(n) = 0$ for $\forall n$.

The likelihood ratio for this problem is given by

$$\begin{aligned} L(\mathbf{x}) &= \frac{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}}\exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}(x(n)-s_1(n))^2\right]}{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}}\exp\left[-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}(x(n)-s_0(n))^2\right]} \\ &= \exp\left[-\frac{1}{2\sigma^2}\left\{\sum_{n=0}^{N-1}(x(n)-s_1(n))^2-\sum_{n=0}^{N-1}(x(n)-s_0(n))^2\right\}\right] \end{aligned} \tag{2.43}$$

Taking the logarithm of the LRT, and after some rearrangements of terms, we conclude that H_1 is decided whenever

$$T(\mathbf{x}) = \sum_{n=0}^{N-1}(s_1(n)-s_0(n))x(n) - \frac{1}{2}\sum_{n=0}^{N-1}s_1^2(n) + \frac{1}{2}\sum_{n=0}^{N-1}s_0^2(n) > \sigma^2\ln\lambda \tag{2.44}$$

Remarks:

1. If we want to design an MPE detector for the case when transmission of signals $s_0(n)$ and $s_1(n)$ are equally probable, i.e., $\lambda = \pi_1/\pi_0 = 1$, it follows directly from (2.43) that we shall choose the hypothesis H_i that is closest in the distance to the data. In other words we choose the hypothesis whose corresponding $D_i = \sum_{n=0}^{N-1}(x(n)-s_i(n))^2$ is minimum. This receiver is often referred to as the *minimum distance receiver* (or nearest-neighbor detector) and is illustrated in Fig. 2(a).

2. For the above case when $\lambda = \pi_1/\pi_0 = 1$ and $E_{s_1} = E_{s_0} = \frac{1}{2}\sum_{n=0}^{N-1}s^2(n)$. The decision rule in (2.44) reduces to: Decide H_1 when $\sum_{n=0}^{N-1}x(n)s_1(n) > \sum_{n=0}^{N-1}x(n)s_0(n)$. So we may also interpret this detector as the *maximum correlation detector*, see Fig. 2(b).

3. Any detection problem involving two signals in additive (white) Gaussian noise can be reduced to a "signal present versus signal absent" problem. Simply create the new signal $\tilde{x}(n) = x(n) - s_0(n)$. As a results we have $\tilde{x}(n) = w(n)$ under H_0 and $\tilde{x}(n) = [s_1(n)-s_0(n)] + w(n)$ under H_1.

To assess the detector performance for the case of two signal sequences in WGN, we need to determine P_e, i.e.,

$$\begin{aligned} P_e &= \pi_0\,\text{Prob}\{\delta(\mathbf{x})=1|H_0\} + \pi_1\,\text{Prob}\{\delta(\mathbf{x})=0|H_1\} \\ &= \frac{1}{2}\left[\text{Prob}\{T(\mathbf{x})>|H_0\} + \text{Prob}\{T(\mathbf{x})<0|H_1\}\right] \end{aligned} \tag{2.45}$$

with $T(\mathbf{x})$ given in (2.44). It can be shown that $T(\mathbf{x}) \overset{H_0}{\sim} N\left(-\frac{1}{2}\|\mathbf{s}_1-\mathbf{s}_0\|^2, \sigma^2\|\mathbf{s}_1-\mathbf{s}_0\|^2\right)$ and $T(\mathbf{x}) \overset{H_1}{\sim} N\left(\frac{1}{2}\|\mathbf{s}_1-\mathbf{s}_0\|^2, \sigma^2\|\mathbf{s}_1-\mathbf{s}_0\|^2\right)$ so that

$$P_e = Q\left(\frac{1}{2}\sqrt{\frac{\|\mathbf{s}_1-\mathbf{s}_0\|^2}{\sigma^2}}\right) = Q\left(\sqrt{\frac{\bar{E}_s(1-\rho_s)}{2\sigma^2}}\right) \tag{2.46}$$

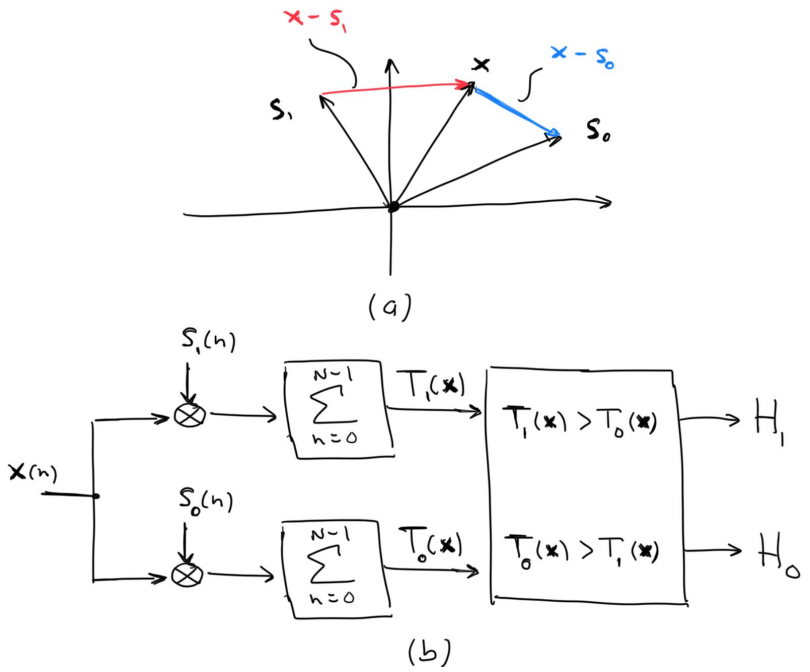


Figure 2.7: Detection of two sequences, $s_0[n]$ and $s_1[n]$, in white Gaussian noise: (a) Minimum distance interpretation, (b) maximum correlation implementation..

where $\bar{E}_s = \frac{1}{2}(E_0 + E_1)$ is the average signal energy and $\rho_s = \frac{\mathbf{s}_1^T \mathbf{s}_0}{\frac{1}{2}(\mathbf{s}_1^T \mathbf{s}_1 + \mathbf{s}_0^T \mathbf{s}_0)}$ is the signal correlation coefficient with $|\rho_s| \leq 1$.

We see that in order to minimize P_e in (2.46), we should look for signal sequences, $s_0(n)$ and $s_1(n)$, that render $\rho_s = -1$. The coherent binary PSK (BPSK) system in Fig. 2 is making use of such *antipodal* sequences. The problem set investigates the detection performance of BPSK signals as well as that of another important class of (orthogonal) signals, namely, the frequency shift keying (FSK) signals.

Signal in colored noise

So far we have assumed that the additive noise is white Gaussian noise, i.e., $\mathbf{w} = [w(0)\, w(1) \cdots w(N-1)]^T$ is distributed according to $N(\mathbf{0}, \sigma^2\mathbf{I})$. We will now relax this assumption and allow for correlation between samples, i.e., $\mathbf{w} \sim N(\mathbf{0}, \mathbf{C})$, with (m,n)th element of matrix $\mathbf{C}$, denoted $\mathbf{C}_{mn}$, given by

$$\mathbf{C}_{mn} = E\{w(m)w(n)\} = \gamma_{ww}(m-n) \tag{2.47}$$

Our detection problem is now

$$\begin{aligned} H_0 &: \quad \mathbf{x} = \mathbf{w} \\ H_1 &: \quad \mathbf{x} = \mathbf{s} + \mathbf{w} \end{aligned} \tag{2.48}$$

where $\mathbf{s} = [s(0)\, s(1) \cdots s(N-1)]^T$ is the vector containing the known signal sequence and $\mathbf{w} \sim N(\mathbf{0}, \mathbf{C})$. The above problem can be cast as a test of the mean of a multivariate Gaussian PDF, reason being that $\mathbf{x} \overset{H_0}{\sim} N(\mathbf{0}, \mathbf{C})$ and $\mathbf{x} \overset{H_1}{\sim} N(\mathbf{s}, \mathbf{C})$.

As usual we start by forming the likelihood ratio and decide H_1 if

$$L(\mathbf{x}) = \frac{p_1(\mathbf{x})}{p_0(\mathbf{x})} = \frac{\frac{1}{(2\pi)^{\frac{N}{2}}\det^{\frac{1}{2}}(\mathbf{C})}\exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{s})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{s})\right]}{\frac{1}{(2\pi)^{\frac{N}{2}}\det^{\frac{1}{2}}(\mathbf{C})}\exp\left[-\frac{1}{2}\mathbf{x}^T\mathbf{C}^{-1}\mathbf{x}\right]} > \lambda \tag{2.49}$$

In terms of the log-likelihood ratio, the left-hand side of (2.49) becomes

$$\begin{aligned}\ln L(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x}-\mathbf{s})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{s}) + \frac{1}{2}\mathbf{x}^T\mathbf{C}^{-1}\mathbf{x} \\ &= -\frac{1}{2}\mathbf{x}^T\mathbf{C}^{-1}\mathbf{x} + \mathbf{x}^T\mathbf{C}^{-1}\mathbf{s} - \frac{1}{2}\mathbf{s}^T\mathbf{C}^{-1}\mathbf{s} + \frac{1}{2}\mathbf{x}^T\mathbf{C}^{-1}\mathbf{x}\end{aligned} \tag{2.50}$$

and we decide H_1 if

$$T(\mathbf{x}) = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{s} > \ln\lambda + \frac{1}{2}\mathbf{s}^T\mathbf{C}^{-1}\mathbf{s} = \lambda' \tag{2.51}$$

The detector in (2.51) is referred to the *generalized matched filter* (GMF). It may be seen as a correlator of the modified signal $\tilde{\mathbf{s}} = \mathbf{C}^{-1}\mathbf{s}$, since

$$T(\mathbf{x}) = \mathbf{x}^T\mathbf{C}^{-1}\mathbf{s} = \mathbf{x}^T\tilde{\mathbf{s}} \tag{2.52}$$

We finally note that whenever $w[n]$ is a WGN sequence, i.e., $\mathbf{C} = \sigma^2\mathbf{I}$, (2.51) reduces to

$$\frac{\mathbf{x}^T\mathbf{s}}{\sigma^2} > \lambda' \tag{2.53}$$

or, equivalently,

$$\mathbf{x}^T\mathbf{s} = \sum_{n=0}^{N-1} x(n)s(n) > \lambda'' \tag{2.54}$$

which is the result we obtained earlier.

The performance of GMF can be obtained by realizing that the test statistic $T(\mathbf{x})$ is Gaussian under each hypothesis, namely, $T(\mathbf{x}) \overset{H_0}{\sim} N(\mathbf{0}, \mathbf{s}^T\mathbf{C}^{-1}\mathbf{s})$ and $T(\mathbf{x}) \overset{H_1}{\sim} N(\mathbf{s}^T\mathbf{C}^{-1}\mathbf{s}, \mathbf{s}^T\mathbf{C}^{-1}\mathbf{s})$. As before, the probability of detection, for a given false-alarm rate, is given by

$$P_D = Q\left(Q^{-1}(P_{FA}) - \sqrt{\mathbf{s}^T\mathbf{C}^{-1}\mathbf{s}}\right) \tag{2.55}$$

with the optimal threshold being

$$\lambda = \sqrt{\mathbf{s}^T\mathbf{C}^{-1}\mathbf{s}} \cdot Q^{-1}(P_{FA}) \tag{2.56}$$

We see that for $\mathbf{C} = \sigma^2\mathbf{I}$, the expressions above reduce to those we obtained for WGN. In contrast to the WGN case, the signal shape $\mathbf{s}$ can affect the detection performance in colored noise. It turns out that the signal vector $\mathbf{s}$ that maximizes P_D is the eigenvector corresponding to the minimum eigenvalue of $\mathbf{C}$. Why?

Random Signals in Gaussian Noise

So far we have assumed that the signal of interest is a deterministic sequence, and the basic detection problem boiled down to detecting a change in the mean of a test statistic. We learned in TTT4120 that in many applications complete knowledge of waveforms is not available and it is better to model the signals as random processes, where first- and second-order statistics are known. Applying the LRT to this scenario will give us either the optimal Neyman-Pearson or MAP detectors.

Energy Detector

We consider the following detection problem

$$\begin{aligned} H_0 &: \quad x(n) = w(n) \;\; n = 0, 1, \ldots, N-1 \\ H_1 &: \quad x(n) = s(n) + w(n) \;\; n = 0, 1, \ldots, N-1 \end{aligned} \tag{2.57}$$

where signal $s(n)$ is *i.i.d.* zero-mean Gaussian random variables with zero mean and variance σ_s, and $w(n)$ is WGN with variance σ^2.

We can now set up the LRT after noticing that $x[n] \overset{H_0}{\sim} N(0, \sigma^2)$ and $x[n] \overset{H_1}{\sim} N(0, \sigma_s^2 + \sigma^2)$. Consequently, we decide H_1 when

$$L(\mathbf{x}) = \frac{\frac{1}{(2\pi[\sigma_s^2+\sigma^2])^{\frac{N}{2}}} \exp\left[-\frac{1}{2[\sigma_s^2+\sigma^2]} \sum_{n=0}^{N-1} x^2(n)\right]}{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} x^2(n)\right]} > \lambda \tag{2.58}$$

or, equivalently, when

$$\ln L(\mathbf{x}) = \frac{N}{2} \ln\left(\frac{\sigma^2}{\sigma_s^2 + \sigma^2}\right) + \frac{1}{2} \frac{\sigma_s^2}{\sigma^2[\sigma_s^2 + \sigma^2]} \sum_{n=0}^{N-1} x^2(n) > \ln \lambda = \lambda' \tag{2.59}$$

After rearranging the terms we can decide H_1 when

$$T(\mathbf{x}) = \sum_{n=0}^{N-1} x^2(n) > \frac{2\sigma^2[\sigma_s^2 + \sigma^2]}{\sigma_s^2} \left[\lambda' - \frac{N}{2} \ln\left(\frac{\sigma^2}{\sigma_s^2 + \sigma^2}\right)\right] = \lambda'' \tag{2.60}$$

The above detector is referred to as the *energy detector*, since it computes the energy of the received sequence and compares it to a threshold λ''. We notice that the test statistic $T(\mathbf{x})$ is a scaled version of the maximum likelihood estimator (MLE) of the variance of $x(n)$. So basically we are trying to detect an increase in variance when going from H_0 to H_1.

Suggested Readings

There is a rich body of literature on the subjects of Statistical Signal Processing and mathematical statistics. Van Trees' [2] is a classic textbook on detection and estimation. This book provides a good basic

treatment of the subject and it is easy to read. Since then, many books have been written. Among others, Poor's [3] and Kay's [4] are most closely related to our treatment of the subject. Poor's book provides a fairly complete coverage of signal detection and estimation. Its presentation is built upon the principles of mathematical statistics and includes some brief discussions of non-parametric and robust detection theory. Kay's books are more oriented towards signal processing applications, though it also includes a good deal of theoretical treatment in statistics. Others like [5] also provide a good coverage on the relevant subjects. In addition, Kassam [6] offers a good understanding of the subject of signal detection in non-Gaussian noise, and Weber [7] offers useful insights into signal design for both coherent and incoherent digital communication systems. If any reader is interested in learning more about mathematical statistics, Bickel and Doksum [8] would be a good reference. It contains in-depth coverage of the subject. For a quick reference to the subject, however, Silvey's book [9] is a concise and very useful one.

It should be noted that studies of statistical signal processing cannot be effective without proper background in probability theory and random processes. Many reference books on that subject , see, e.g., [10], are available.

[3]
[4] ; and
[5]
[6]
[7]
[8]
[9]
[10] ; ; ; and

Appendix: Proof of Bayes Risk

Substituting (2.6), (2.10a) and (2.10b) into (2.15) yields

$$\mathcal{R} = \pi_0 C_{00} \int_{\Omega_0} p_0(x)dx + \pi_0 C_{10} \int_{\Omega_1} p_0(x)dx + \pi_1 C_{01} \int_{\Omega_0} p_1(x)dx + \pi_1 C_{11} \int_{\Omega_1} p_1(x)dx$$

$$= \pi_0 C_{10} + (1-\pi_0) C_{11} + \int_{\Omega_0} \{(1-\pi_0)(C_{01}-C_{11})p_1(x) - \pi_0(C_{10}-C_{00})p_0(x)\}dx$$

Note that the sum of the first two terms is a constant. In general, we can assume that

$$C_{01} - C_{11} > 0 \qquad \text{and} \qquad C_{10} - C_{00} > 0$$

In other words, the costs of making correct decisions are less than those of making incorrect decisions. Let

$$\begin{aligned} I_1(x) &\triangleq (1-\pi_0)(C_{01}-C_{11})p_1(x) \\ I_2(x) &\triangleq \pi_0(C_{10}-C_{00})p_0(x) \end{aligned}$$

It can be seen easily that $I_1(x) \geq 0$ and $I_2(x) \geq 0$; and $\mathcal{R}$ can be rewritten as:

$$\mathcal{R} = \text{constant} + \int_{\Omega_0} [I_1(x) - I_2(x)]dx$$

To minimize $\mathcal{R}$, the observation space Ω need be partitioned such that $x \in \Omega_1$ whenever

$$I_1(x) \geq I_2(x).$$

In other words, decide H_1 if

$$(1-\pi_0)(C_{01}-C_{11})p_1(x) \geq \pi_0(C_{10}-C_{00})p_0(x) \tag{2.61}$$

So, the Bayes decision rule is essentially evaluating the likelihood ratio defined by

$$L(x) \triangleq \frac{p_1(x)}{p_0(x)} \tag{2.62}$$

and comparing it to the threshold

$$\lambda \triangleq \frac{\pi_0}{1-\pi_0} \frac{C_{10}-C_{00}}{C_{01}-C_{11}} \tag{2.63}$$

Problems

1. Generate sequence $\{x[n]\}$ of $N = 10^4$ *i.i.d.* Gaussian random variables with zero mean and unit variance ($\sigma^2 = 1$). Plot 100 samples together with a constant threshold $\lambda = 0.3$. How many samples exceed the threshold? Now consider all sample values: What is the fraction that exceeds the threshold? Compare this value to the theoretical value. Plot histogram. Plot true PDF, etc. (Useful Matlab functions: hist, normpdf, normcdf, etc.)

2. Consider the following binary hypothesis testing problem

$$\begin{aligned} H_0 &: x[0] = w[0] \\ H_1 &: x[0] = w[0] + 1 \end{aligned}$$

   where $w(0)$ is $N(0,1)$. Write the LRT for the NP detector. Compute the optimal threshold to be used in the LRT so that the probability of false alarm is $P_{FA} = 10^{-3}$. What is the power of the test, i.e., what is the probability of detection P_D? How can you increase the detection performance?

3. Verify that $T(\mathbf{x}) = \sum_{n=0}^{N-1} x(n)$ is $N(0, N\sigma^2)$ under H_0 and $N(NA, N\sigma^2)$ under H_1.

4. Consider the following binary hypothesis testing problem

$$\begin{aligned} H_0 &: x[n] \sim N(0, \sigma_0^2) \\ H_1 &: x[n] \sim N(0, \sigma_1^2) \end{aligned}$$

   with $\sigma_1^2 > \sigma_0^2$. For the general case of N samples, derive the LRT and threshold associated with the NP test. For the special case of $N = 1$, illustrate the decision regions Ω_0 and Ω_1.

5. Consider the following binary hypothesis testing problem

$$\begin{aligned} H_0 &: x[n] = w[n] \\ H_1 &: x[n] = w[n] + s[n] \end{aligned} \qquad n = 0, 1, \ldots, N-1$$

   where $\{w[n]\}$ is a sequence of *i.i.d.* Gaussian random variables with zero mean and variance σ^2, while $\{s[n]\}$ is a sequence of deterministic constants. Derive a Neyman-Pearson (N-P) optimum test with a prescribed level α. Simplify your test statistic as much as you can.

6. You are given the task to design a signal that renders the best detection performance in a WGN channel. The following two signals are suggested

$$\begin{aligned} s_0(n) &= 4 \\ s_1(n) &= 4 \cdot (-1)^n \end{aligned} \qquad n = 0, 1, \ldots, N-1$$

   Which signal gives the best detection performance?

7. Consider the following binary hypothesis testing problem

$$\begin{aligned} H_0 &: x[n] = w[n] \\ & \qquad\qquad n = 0, 1, \cdots, N-1 \\ H_1 &: x[n] = w[n] + s \end{aligned}$$

where $\{w[n]\}$ is a sequence of *i.i.d.* random variables with a first-order pdf given by

$$p(w[n]) = \frac{a}{2} e^{-a|w[n]|} \quad a > 0$$

while s is a constant deterministic signal. Derive an N-P optimum test with a prescribed level α. Simplify your test statistic as much as you can. What if s is replaced by s_i in the alternative hypothesis? Specifically, what would be the corresponding N-P optimum test statistic?

8. In coherent binary phase shift keying, information is transmitted using two equally probable sinusoids with one out of two phases: $s_0[n] = A\cos 2\pi f n$ or $s_1(n) = A\cos(2\pi f n + \pi) = -s_0(n)$. The signals propagates over an additive white Gaussian noise channel and at the receiver we are are faced with the following detection problem:

$$\begin{aligned} H_0 &: \quad x(n) = s_0(n) + w(n) \;\; n = 0, 1, \ldots, N-1 \\ H_1 &: \quad x(n) = s_1(n) + w(n) \;\; n = 0, 1, \ldots, N-1 \end{aligned} \tag{2.64}$$

where $w(n)$ is WGN with variance σ^2.

a) Show that $\rho_s = -1$.

b) Show that $\bar{E}_s = E_{s_0} = E_{s_1} \approx A^2 N/2$.

c) Plot P_e versus the energy-to-noise-ratio (ENR), $\bar{E}/\sigma^2$, and provide the required ENR to yield an error rate of 10^{-3}.

9. In coherent frequency shift keying, information is transmitted using two equally probable sinusoids with one out of two frequencies: $s_0[n] = A\cos 2\pi f_0 n$ or $s_1(n) = A\cos 2\pi f_1 n$. The detection problem is the same as in Problem 8 above.

a) Show that $\rho_s \approx 0$ for $|f_0 - f_1| >> \frac{1}{2N}$.

b) Show that $\bar{E}_s = E_{s_0} = E_{s_1} \approx A^2 N/2$.

c) Plot P_e versus the energy-to-noise-ratio (ENR), $\bar{E}/\sigma^2$, and provide the required ENR to yield an error rate of 10^{-3}.

3
Classification

Introduction to classification

The term classification is used for a variety of (related) concepts. A person deciding between ”right” or ”wrong” based on learned ethics and moral is doing a form of classification. The two classes are dependent on the different humans interpretation and moral opininons, and the decision is based on a form of reasoning. In sum we can call this **human intelligence (HI)** based classification. If we try to automate this process, the system has to learn how humans think, so called **artificial intelligence (AI)**.

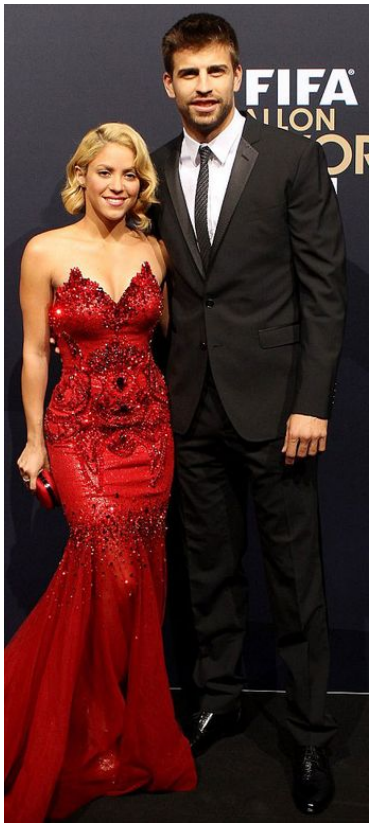
However, in this course we will concentrate on the other end of the classifier alternatives. We will assume that a human is not involved in the classification process. Instead a sensor is measuring some values or signals which can be transformed into appropriate features suited for classification. Further, the classes are logically and scientifically defined, i.e. not suspect to different humans interpretations (like ”right” and ”wrong”). The feature based classification is performed by a program on some digital hardware. However, first the program has to learn (from example features) the behaviour of the different classes. Typically this behaviour is represented by mathematical class models. We often say that this classifier is based on **statistics and machine-learning**. This type of classifiers is much more used than the AI-based. Further, on the same problem they usually outperform the AI-alternative.

Some practical problems have only two classes. However, we can not regard these cases in the same way as we do detection. Assume we set up a camera in the mountains to take pictures of everything moving. However we are only interested in the rare species arctic fox. In a detection mode we want to know if the animal moving is an arctic fox or not; i.e.. there is only a single ’class’ of interest, namely arctic fox. If such a fox is detected the next step is to decide upon the sex, i.e. male or female. We now have two classes, both of equal interest. Thus we have a classification case.

Introducing basic terminology by an example

We humans are quite good to do classification. Let us assume that we are using our eyes to decide on whether a person is a male or female.

Obviously we use several cues/features to help us decide, i.e. height, body shape, hair length, aso. Still it happens that we do some mistakes, i.e. the features overlap (see figure 3) and thus our decision strategy are not error free. However it is clear that the **error rate** decreases as the number of features increases. Thus if we also could use our ears, i.e. the person’s voice; this would obviously help in some cases. Further skin color should be added as an ”eye feature” since the mean height of Asians is lower than for Eurasians and most Africans. On the ”output side” we could introduce **conditions**, i.e. female given Asian. Alternatively we



(a) Typical heights

(b) Less typical heights

Figure 3.1: *Male versus female heights*

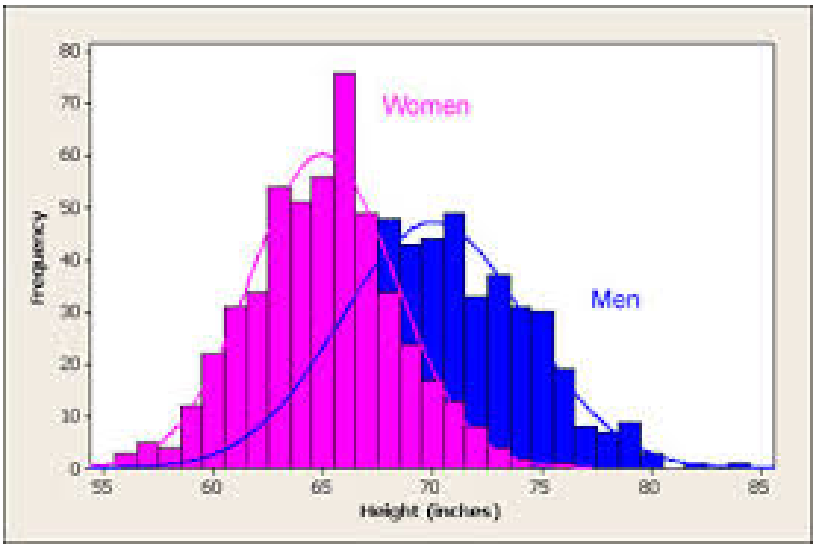


Figure 3.2: *Histogram of heights*

could introduce combinations thorough **subclasses** , i.e ”Asian female”.

How is it that we as humans are able to do the above classification? We are obviously not born with this ability. The answer is of course by observing and learning. In the classification terminology this is called **”supervised learning/training”**. We observe a person and is told by a ”teacher’/supervisor” that this is an Asian female.

However we must also learn to **"generalize"**. Every human is special (except for identical twins...), thus we need to classify a person we never have seen before. It is therefore obvious that we need to learn by observing more than one example from each class. In fact this is often the core of learning by example; you have to see/learn a representative number of each class in order to do a qualified "guess" of class for an unseen person. A golden rule for learning is that ".. the more data the better..".

I some cases we do not have a teacher/supervisor. We just suspect that some data can be organized into a finite number of distinct groups. Even the optimal number of groups is unknown and needs to be estimated as a part of the analysis. This problem type is called **"unsupervised training"** or **"clustering"**. Using the same case as above one unsupervised problem can be to organize people into groups based on skin color. Obviously there is no single solution to this, still we should be able to learn from most of the "suggestions" coming out of the clustering procedure.

Turning back to the features, most of them are continuous valued. Let us use height as an example. Even if this is a continuous variable we usually quantize the value. However we must be sure to use an accurate enough resolution. That is probably the reason that most people is used to applying centimeter, however even coarser quantization (e.g. inches) is sometimes used . Further, when seing a person we of course just take a qualified guess with respect to the height. Thus we **estimate** the input feature height.

Finally the input stimulus from our different senses are sent to the brain for decision. As today we (luckily..?) do know little or nothing about how the brain implements the decision. However this is the core of the HI based classification system. Thus a variety of mathematically based method and algorithms are implemented. For many applications we are able to evaluate the quality of them by comparing them to the HI performance. However for most of the applications we have no human baseline as the features are derived from a set of non-human like sensors, like ultrasound, radar, sonar, hyperspectral camera, accelerator, strain gauge aso. Further for all applications there is a typical maximum tolerable error rate.

A generic classification system.

In many practical problems classifiation is either a mandatory part or even the main goal. Figure 3 shows a complete classification system. An application dependent sensor measures a signal from a physical source. As an example a microphone will convert sound pressure to an electrical signal. The sensor forwards a signal which seldom is directly applicable for classification. The discriminative information is often embedded/hidden in the redundant signal. Thus in a following step features (hereby named x) are extracted, typically in the form of a vector. The features ideally

contains enough discriminative information to yield a satisfactory performance with respect to class error rate. Then the main part concists of deciding which class ω_i $i = 1, \ldots, C$ the system "believes" the signal/feature vector belongs to. This decision can of course be correct or wrong. As an integrated part of the design one must estimate the performance of the classifier. How this is done is explained in a separate subchapter.

There exists a variety of different classifiers. They differ both with respect to method/principle and complexity. In this course we will limit ourselves to a few of the most used and simplest. However, at the end of the lectures we will give a short overview of the state-of-the art. Finally we will stress the following. It is the total system performance which is important. It does not help to use a complex classifier method if the error rate is 30%. In many cases a much better strategy is to search for better sensors and/or feature extaction methods! It is for instance well known that in speech recognition the choice of microphone can result in success or disaster.....

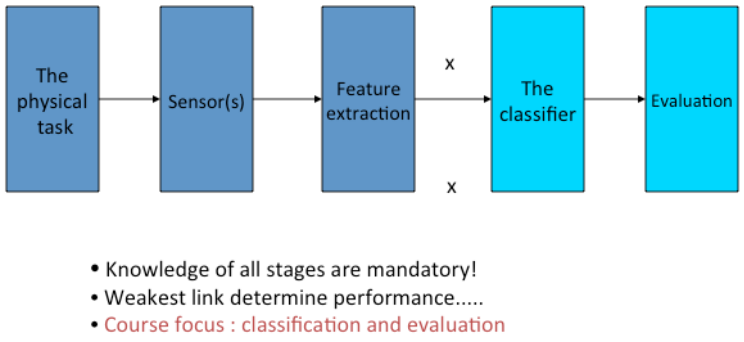


Figure 3.3: *A complete classification system*

In classification the output is a discrete value correponding to one out of C possible classes. **Regression** is a concept related to classification. The difference lies in that the regression outputs can be any continuous valued vector, often with the same dimension as the input feature vector. Thus we can define regression as a kind of mapping method. Models and training methods for classification are often also used for regression. Regression is a central part in a variety of applications, for instance :

- Prediction of future input values
- A signal in noise is input and the same signal with (much) less noise is wanted as output
- Finding connection with pair of values, i.e. respectively the input and the wanted output

In this course we will however focus on classification.

Different classifier types.

We can divide the classifiers into two groups based on static versus dynamic input/outputs. Static means that the the feature input x does not have any temporal information and belongs to a single class. Dynamic means that an input vector sequence (i.e. embedded temporal information) belongs to a single class or to a sequence of classes. An example of the latter is speech recognition where the sound is transformed into a vector sequence which is classified into a sentence (sequence of words).

In this course we will restrict ourselves to static classifiers. However, dynamic based classifiers will be shortly described as a part of the state-of-the-art presentation.

Different problem types of classification.

We can define three main categories, as illustrated in figure 3 for a two-dimensional feature vector $x = [x_1 x_2]^T$ and two classes ω_1 and ω_2. The simplest category is called a **linearly separated** problem. As the name indicates we can in this case separate the two classes without errors by a simple line. If the input dimension is three this decision border is a plane. For higher dimensions we use the term "hyperplane". Note that there in general is an infinite number of lines/planes/hyperplanes that will give error free decision. However, an intuitive and robust choice would be to use the border with the same minimum distance to both classes. If more than two classes exist a corresponding decision border is needed for each pair of classes. A classifier which uses linear decision border is (logical enough) called a **linear classifier**. Such classifers seldom have the best performance, however they usually are simple to design and apply.

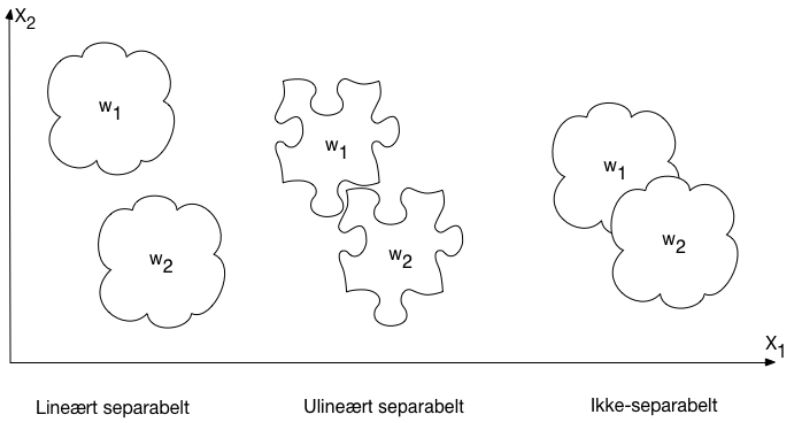


Figure 3.4: *Three different classification types*

For the **nonlinear separated** problem type it is also possible to find decision borders which give error free performance. However, these borders can not be hyperplans. They have to have a nonlin-

ear form, thus the corresponding classifiers are called **nonlinear classifiers**. In the last problem type the classes overlap in the input room. Thus it is not possible to find decision borders that give error free performance. "Unfortunately" most practical problem are of this third type, i.e. so called **nonseparable** problems. Whether one should go for a linear or nonlinear classifier in this case is a balance between complexity and performance (error rate). If the resulting error rate turns up to be too high for practical use, a good strategy is to search for alternative/better features (and even sensors), hoping that this will result in a more separable problem.

We will later see that the classifier performance also is highly dependent on how we design/train them. Central ques here are the **amount of training data** (class labeled features) and also the **representativeness** of the data. .

The optimal classifier.

The theoretically optimal classifier in the sense of minimum error rate has a statistical form. In order to introduce this classifier we have to refresh our basic knowledge of statistics and probabilities. Let us assume that that our features are represented by a continuous valued vector x and that we have $\omega_i,\ i = 1, C$ classes. Only four basic formulaes are needed to describe not only this classifier but any statistically based classifer :

- A priori probablities $P(\omega_i)\ \ i = 1, \ldots, C$ where of course $\sum_i P(\omega_i) = 1$
- Class conditioned densities $p(x/\omega_i)\ i = 1, \ldots, C$ where $\int p(x/\omega_i)dx = 1\ \ i = 1, \ldots, C$
- A posteriori probabilities $P(\omega_i/x)\ \ i = 1, \ldots, C$ hvor $\sum_i P(\omega_i/x) = 1$ for any x
- Joint distributions $p(x, \omega_i)\ i = 1, \ldots, C$ where :
  - $\sum_i p(x, \omega_i) = p(x)$ for any x
  - $\int p(x, \omega_i)dx = P(\omega_i)\ \ i = 1, \ldots, C$
  - $p(x, \omega_i) = p(x/\omega_i)P(\omega_i) = P(\omega_i/x)p(x)\ \ i = 1, \ldots, C$ (named **Bayes law**)

Bayes law can be rewritten as :

$$P(\omega_i/x) = \frac{p(x/\omega_i)P(\omega_i)}{p(x)} \tag{3.1}$$

One can show (not part of syllabus here) that the following decision rule leads to the least possible error rate :

$$x \in \omega_j \quad \Leftrightarrow \quad P(\omega_j/x) = \max_i P(\omega_i/x) \tag{3.2}$$

This decision rule is called the **Bayes Decision Rule (BDR)**. Applying Bayes law the BDR can be written as :

$$x \in \omega_j \quad \Leftrightarrow \quad p(x/\omega_j)P(\omega_j) = \max_i p(x/\omega_i)P(\omega_i) \qquad (3.3)$$

This is due to that the denominator $p(x)$ in equation (3.1) is the same for all classes. A BDR-based classifier is logically enough also called a **Maximum A Posteriori (MAP)** klassifiserer.

However, the cruical point is that the BDR/MAP classifier only is a **theoretically** optimal classifier. The involved densities and probabilities are never known exactly. This is especially critical for the class conditoned densities. Usually one has to choose a parametric form (for instance a Gaussian). And then one will need a set of (training) data to estimate the parameters (for a Gaussian the mean and variance). This is the same kind of challenge as was met in the course Digital Signal Processing where one had to choose a parametric process (i.e. AR[P]) to model physical signals and thereafter estimate the model parameters.

However, choosing a model and estimating model parameters transforms a theoretically optimal classifier to a practical but **sub-optimal** classifier. This again opens up for searching for alternative classifier types (than BDR/MAP) which may show a better performance! In this course we will start with the sub-optimal BDR/MAP classifier, also called a "Plug-in-MAP" classifier.

Many classifiers are not statistically based, i.e. they do neither apply densities nor probabilities. The two largest groups are called respectively discriminant classifiers and template (reference) based classifiers. The linear classifier belongs to the discriminant category.

The three categories of classifiers mentioned above will be described in the following subchapters.

The Plug-in MAP classifier

This classifiser type uses BDR (equation (3.3)) based a chosen parametric density form for which the parameters must be estimated by a set of training data. The most popular parametric form is the Gaussian density

$$p(x/\omega_i) = N(\mu_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_i|}} \exp\left[-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right] \quad i = 1, \ldots, C \tag{3.4}$$

The real (but unknown) density usually have a complex and "noisy" form with multiple peaks. This is especially the case when the input feature dimension $D = dim(x)$ is large. Thus the "bell shaped" Gaussian form is seldom a good approximation. However, for a large number of applications a density in the form of a weighted mixture of Gaussians has shown succesful, i.e. resulted in a good approximation to the real densities. This density model is called a **Gaussian Mixture Model (GMM)** and have the form

$$p(x/\omega_i) = \sum_{k=1}^{M_i} c_{ik} N(\mu_{ik}, \Sigma_{ik}) = \sum_{k=1}^{M_i} \frac{c_{ik}}{\sqrt{(2\pi)^D |\Sigma_{ik}|}} \exp\left[-\frac{1}{2}(x-\mu_{ik})^T \Sigma_{ik}^{-1}(x-\mu_{ik})\right] \quad i = 1, \ldots, C \tag{3.5}$$

where the weights fulfil the constraint $\sum_k c_{ik} = 1$. An important part of the model approximation is to find a good choice of the number of mixtures M_i for each class. ω_i. Note that Gaussian based classifiers are nonlinear as they have quadratic terms (in x) in the exponent.

The priors $P(\omega_i), \;\; i = 1, \ldots, C$ must be estimated as well. However good estimates are usually not as critical as for the densities. The estimates are usually found either by knowledge of the problem, or by doing a simple frequency count in the training set. For many applications even equal priors $P(\omega_i) = 1/C, \;\; i = 1, \ldots, C$ are assumed without any noticable performance degradation.

The linear classifier

In chapter 3.1 it was said that a linear classifier is adequate for a linear separable problem. However, more or less all practical problems are non-separable, thus a linear classifier will usually give a worse performance than a nonlinear classifier. But the performance difference is sometimes acceptable small, and a linear classifier can be preferred due to simplicity.

In a discriminant classifier each class is described by a (so called dicriminant) function $g_i(x)$ and the decision rule is given by

$$x \in \omega_j \quad \Leftrightarrow \quad g_j(x) = \max_i g_i(x) \tag{3.6}$$

Note that this decison rule is quite similar to the BDR. In fact one can define the MAP classifier as a special case of a discriminant classifier. However, the discriminant functions $g_i(x)$ can have a much more general form.

The linear discriminant classifier is defined by the function

$$g_i(x) = w_i^T x + w_{io} \quad i = 1, \ldots, C \tag{3.7}$$

where w_{io} is offset for class ω_i. For $C > 2$ classes this can written in a compact matrix form $g = Wx + w_o$ where g and w_o eare vectors of dimension C (number of classes) and W is a CxD matrix. $C = 2$ is a special case where we can rewrite the decision as $x \in \omega_1 \quad \Leftrightarrow \quad g_1(x) - g_2(x) > 0$. Thus we can define a single function $g(x)$ and a decision rule $x \in \omega_1 \quad \Leftrightarrow \quad g(x) > 0$ and of course $x \in \omega_2$ else.

The template based classifier.

The decision rule is simple for this kind of classifier. The input x is matched towards a set of references (templates) which have the same form as x. The rule finds the reference which is closest (or most "similar") to x and assumes that x belongs to the same class as this reference. This method is called "nearest neighbour - NN". Several variants of this classifier category exist dependent on

- Which decision rule that is used (other than NN)
- The kind of distance or similarity measure that is used.
- How the references are chosen and used

Assume $k = 1, \ldots, M_i$ references ref_{ik} for class ω_i, i.e. a total of $M = \sum_i M_i$ references. The most general reference set is then given by $ref_{ik} = (\mu_{ik}, \Sigma_{ik})$ $k = 1, \ldots, M_i$; for each class $\omega_i \quad i = 1, \ldots, C$. A corresponding distance measure (called the Mahalanobis distance) is defined by

$$d(x, ref_{ik}) = (x - \mu_{ik})^T \Sigma_{ik}^{-1} (x - \mu_{ik}) \tag{3.8}$$

A more advanced decision rule is called KNN. Here one finds the $K > 1$ nearest references. The decision is then based on a majority vote within this K. If several classes end up with the same number, the closest class among them is chosen. .

The parameters in the reference set (as given by equation (3.8)) must be estimated from data in a training/design phase. Especially the covariance matrixes Σ_{ik} requires a lot of data to be well estimate. One therefore often uses simplifications of the Mahalanobis distance, i.e. a) common class matrix $\Sigma_{ik} \rightarrow \Sigma_i$ b) global matrix, i.e. $\Sigma_i \rightarrow \Sigma$ or c) a simple Euclidian distance as follows

$$d(x, ref_{ik}) = (x - \mu_{ik})^T (x - \mu_{ik}) \tag{3.9}$$

i.e. $\Sigma \rightarrow I$ (the unity matrix).

An advantage of the Euclidian case is that the parameters $ref_{ik} = \mu_{ik}$ can be chosen directly from the the training data set. This is discussed later on and also in connection with the subchapter on clustering.

A similarity measure can be used as an alternative to a distance measure, i.e. they are in principle inverse measures. Then the decision rule is to choose the reference which gives the highest similarity score to the feature x. Thus the Plug-in-MAP classifier can be thought of as a tempate based classifier, i.e. equation (3.3) gives the discriminant functions while equations (3.4) and (3.5) give the similarity measures (assuming Gaussian class densities).

Analyzing the features.

In most cases the feature extraction method is given by the choice of the sensor and the knowledge of the application. However, for some cases we will have the opportunity to choose among several types of features. Anyhow, it is always useful to be able to inspect the features with respect to class discrimination and the form of the distribution. This information is important regarding which classifier type to choose. To visualize the features one can use the so called histogram. A histogram shows an approximation to the distribution of a scalar feature. All the training set features are quantized into one of a finite number of values (bins). Then the (relative) numbers of features in the training set in each bin are shown. Note that when the bins increases (i.e. the bin width decreases) the histogram converges towards an estimate of the feature distribution. In figure 3 we show histograms for three different classes for a chosen feature. In figure 3 we show the same for another feature. We easily see that the first feature is good for discriminating class 2 from the others. The same applies to the second feature and class 3.

In all practical cases we will us a feature **vector**. It is custom to analyze/visualize one vector element at a time. However, this method will not detect any correlation between the vector elements. Such a correlation can help to discriminate classes even if the histograms do not show this. This is examplified in figure 3. We clearly see that using both features X_1 and X_2 give a more discriminatory overview than only looking into one direction at a time. Methods exist that extracts the correlation in ways which helps classification, however this is not included into this course.

The histograms are also useful for choosing good alternatives for classifier type and structure. For feature 2 (figure 3) the histogram of class 3 clearly indicates a Gaussian like shape, i.e. a single Gaussian model. However, the histogram for class 1 has two ”peaks”, thus a 2-mix GMM would be a better class model. It is custom to input the whole feature vector to the same structure. Thus a reference based or linear classifier is a potential alternative to a Gaussian based classifier in this case.

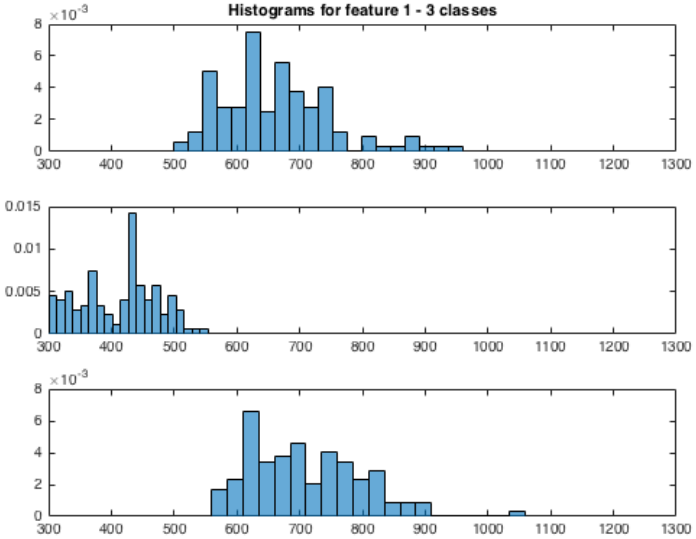


Figure 3.5: *Class histograms for feature 1*

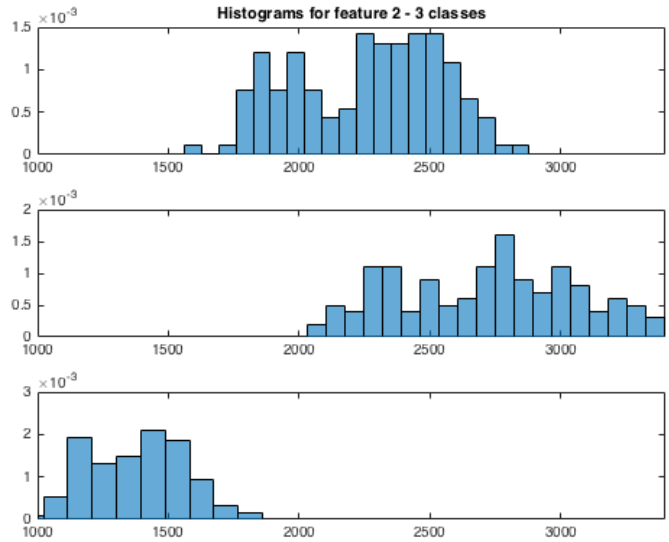


Figure 3.6: *Class histograms for feature 2*

State-of-the-art classifiers

A variety of classifiers exist. Some of them have great similarities to the three we have described. For example the classifier called "Support Vector Machine - SVM" started as a simple linear classifier, however the training/design method was based on complex discriminating principles. Today SVM is only used as a nonlinear classifier due to so called kernel based techniques.

The two largest groups of classifiers used today are respectively the **nonlinear and the context-based classifiers**. In the first group

the neural network based classifiers dominates. They have shown impressive performance, especially for problems where one has no good model. Much of the good performance is due to a more complex structure called a deep neural network (DNN), and a correponding new training technique. However, the potentially success is conditioned on that one has access to a large amount of training data. Thus deep (neural network) learning is often mentioned together with an other buzzword, namely **"Big Data"**. The combination of large data sizes and complex DNNs demands large resources during training with respect to memory and CPU-power. Especially graphical cards have shown powerful for this type of training. When neural networks was introduced late 1980s, the training was called **machine learning**. This is another buzzword today and now includes nearly all kinds of data driven automatic learning strategies.

Context-based classifiers are primarily used for sequence classification where much of the class information is embedded in the temporal evoluation within the sequence. The statistical based Hidden Markov Model (HMM) classifier is probably the most known within this category. Speech recognition and gene classification are examples where temporal evolution is cruical, and HMM is therefore state-of-the-art for this kind of problems.

Design/training and testing/evaluation

The first step in the design phase is to choose the classifier category and the structure/form. The next step is then to estimate the parameters of the chosen structure. For the Plug-in-MAP category the GMM form in equation (3.5) is the most popular structure. The corresponding parameters to be estimated are the mixture weights, the means and the covariance matrixes of the Gaussians. Usually one has to choose the mixture numbers, however searching for good numbers is included in some design techniques.

In nearly all training cases one will need a so called **labeled training set** for the parameter estimation. This set has to be large to achieve robust estimates. If the data set is too small the classifier will perform much worse on unseen data then on the training set, i.e the classifier does not **generalize**. Further the data has to be **representative**, i.e. all variants of input/class pairs should be present in a large enough scale. Further, the data must be unbiased, i.e. using identical sensors, recording conditions and so on.

Let us assume we have a training set of size N. We name the input data $X_N = \{x_1, \ldots, x_N\}$ and the corresponding class labels $T_N = \{t_1, \ldots, t_N\}$ (so called **targets**). This class information can either have the simple scalar form $t_i = \omega_j \leftrightarrow x_i \in \omega_j \; i = 1, \ldots, N$ or a vectorial form $t_i = [0, \ldots, 0, 1, 0, \ldots, 0]^T$, i.e. only element j in the vector is 1 ($t_i(j) = 1 \leftrightarrow x_i \in \omega_j \; i = 1, \ldots, N$). In the latter case the dimension of the target vector is the same as the number of classes C.

Assuming a representative data set one could ask if we can know/guess what size N we need for training. In many cases knowledge of the problem can help us to this. However, there is a rule which always is correct : "the more data the better...".

Most training algorithms are iterative. One applies the training set several times and the algorithm should give better performance (correctness) on the training set for each iteration. However, a so called **validation set** should be used in parallel. This set is not used for estimating parameters, but only evaluated with respect to performance. When the performance for this set stops improving the training should also stop. As both the training and validation set are used for design, a third **test set** must be used for evaluation of the true/unseen performance. Typically, the performance is best for the training set and worst for the test set. The difference between the validation set and the test set should not be large as it tells us how well the classifier generalizes. If the difference is (too) large the reason can be that

- the training/validation sets are too small and/or not representative
- wrong category classifier is chosen
- wrong structure is chosen

Assume a test set of size M. The performance p_M of the test set gives only an **estimate** of the true (but unknown) performance p_t for the trained classifier. This true performance p_t can only be found if the test set is unbiased and is infinitely large. To explain this let us assume we have two different test sets of the same size. It is understandable that we will find some difference in the performances for these two sets. The same applies if we extend this to any finite number of finite sized test sets! Thus we can think of the performance as a statistical variable, i.e an estimate of the true performance. Using statistical theory it is common to give the true performance as a 95% confidence interval. This says that with 95% certainty will the true performance p_t fulfil

$$p_M - \alpha_l \leq p_t \leq p_M + \alpha_u \tag{3.10}$$

The bounds α_l and α_u are dependent on p_M and M. Logically these bounds will shrink towards zero as the test set size M increases. See chapter 3.5.2 for more on this subject.

In the following section we will describe the training process for our three different classifiers.

Plug-in-Map using Maximum Likelihood (ML) based training

The ML algorithm estimates the parameters for a single class at a time. Given a training subset $X_{N_i} = \{x_{i1}, \ldots, x_{iN_i}\}$ where all inputs are independent and all belong to the same class ω_i. We further use the notation $\Lambda_i = \{\mu_i, \Sigma_i\}$ for the unknown parameters we want to estimate. The likelihood for the subset is given by

$$L[X_{N_i}, \Lambda_i] = \prod_{k=1}^{N_i} p(x_{ik}/\Lambda_i) \tag{3.11}$$

A density has known parameters and is a function of the variable x. A likelihood, however, has known values of x. Thus if also the parameters are known the likelihood is a number. However, we can have situations where the likelihood is a function of unknown parameters. This is obviously the case during training since the goal is to estimate the unknown parameters Λ_i of the class.

When the distributions $p(x/\Lambda_i) = p(x/\omega_i)$ are assumed to have an exponential form (for example Gaussian - see equations (3.4) and (3.5) in subchapter 3.3), the logarithm will have a closed analytical form. Thus one can simplify the equation (3.11) by instead using loglikelihood (LL), i.e.

$$LL[X_{N_i}, \Lambda_i] = \log[\prod_{k=1}^{N_i} p(x_{ik}/\Lambda_i)] = \sum_{k=1}^{N_i} \log[p(x_{ik}/\Lambda_i)] \tag{3.12}$$

As mentioned the x-values in equation (3.12) are known while the parameters are unknown. Thus one wishes to find the parameters which maximize this LL. This is a problem which is equivalent to curve fitting, for instance finding the line which has least possible distance sum to a set of points. Maximation of a function is done by taking the gradient and finding the parameters which give the gradient the value zero.

$$\nabla_{\Lambda_i} LL[X_{N_i}, \Lambda_i] = \sum_{k=1}^{N_i} \nabla_{\Lambda_i} \log[p(x_{ik}/\Lambda_i)] = 0 \tag{3.13}$$

ML-training in the single Gauss case.

If we decide on using single Gaussians as class densities we can easily derive optimal closed expressions. These expressions are called **estimators** and when we apply a known set of x-values in the estimator we get a concrete value called an **estimate**. Starting with equation (3.4) we easily get

$$\log[p(x_{ik}/\Lambda_i)] = -\frac{D}{2} log(2\pi) - log|\Sigma_i| - \frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) \tag{3.14}$$

Taking the gradient with respect to μ_i we get

$$\nabla_{\mu_i} \log[p(x_{ik}/\Lambda_i)] = \Sigma_i^{-1}(x-\mu_i) \tag{3.15}$$

Using equation (3.15) into equation (3.13) we easily find that the estimator for the mean is given by equation (3.16). Doing the same exercise for the covariance matrix we end up by equation (3.17). Note that the original covariance estimator uses the true but unknown mean, while we for obvious reasons have to replace it with the estimate.

$$\hat{\mu}_i = \frac{1}{N_i} \sum_{k=1}^{N_i} x_{ik} \tag{3.16}$$

$$\hat{\Sigma}_i = \frac{1}{N_i} \sum_{k=1}^{N_i} (x_{ik} - \hat{\mu}_i)(x_{ik} - \hat{\mu}_i)^T \tag{3.17}$$

The above two estimators are called respectively a sample mean

and a sample covariance estimator. It can be shown that the mean estimator is an MVU-estimator. This is not the case for the covariance estimator, however it can be shown to be asymptotical efficient.

ML-training using GMM densities

If we choose GMM as the class densities (equation (3.5)), we end up with the logarithm of a sum in equation (3.12). This leads to an

insintric gradient based equation. The estimates appears on both sides of the equal sign, i.e. mathematically we will have $\Lambda_i = f(\Lambda_i)$ where the function $f()$ is application dependent. However, there exists an efficient iterative algorithm called **Expectation Maximation - EM** which will find a (suboptimal) solution. The term "Iterative" means that we start with a chosen initial set of estimates $\Lambda_i(0)$ which we put into the function to get a new and better estimate (higher LL), i.e.

$$\Lambda_i(m) = f(\Lambda_i(m-1)) \;\Leftrightarrow\; LL(m) > LL(m-1) \;\; m = 1, 2, \ldots \quad (3.18)$$

The algorithms stops after a finite number of iterations, i.e. when

the improvement in LL gets less than a chosen value.

Later we will present the basics of clustering. We will then relate the EM-algorithm to clustering and show the similarities between the two methods.

MSE based training of a linear classifier.

For $C > 2$ classes the output of the linear classifier was given by the discriminant vector $g = Wx + w_o$ with dimension C. This expression can be rewritten as $g = [W\ w_o][x^T\ 1]^T$. If we then redefine/merge the input and matrix to $[W\ w_0] \rightarrow W$ og $|x^T\ 1]^T \rightarrow x$ we end up with $g = Wx$.

The linear classifier is, in contrast to the the Plug-in-MAP classifier, not statistically based. Thus one can not use an optimization criteria like ML during training. Since most practical problems have a non-separable form, a criteria which works well for this case should be chosen. The most popular variant is the "Minimum Square Error - MSE" . MSE requires target values at the output, i.e. in the vectorial form as given in the introduction to subchapter 3.

$$MSE = \frac{1}{2}\sum_{k=1}^{N}(g_k - t_k)^T(g_k - t_k) \quad (3.19)$$

The total matrix W is trained as one single entity, i.e. the total training set for all classes is used simultaneously. Thus both the training technique and the classifier are named discriminative. This is in contrast to ML-training of the Plug-in-MAP classifier where each class was trained separately.

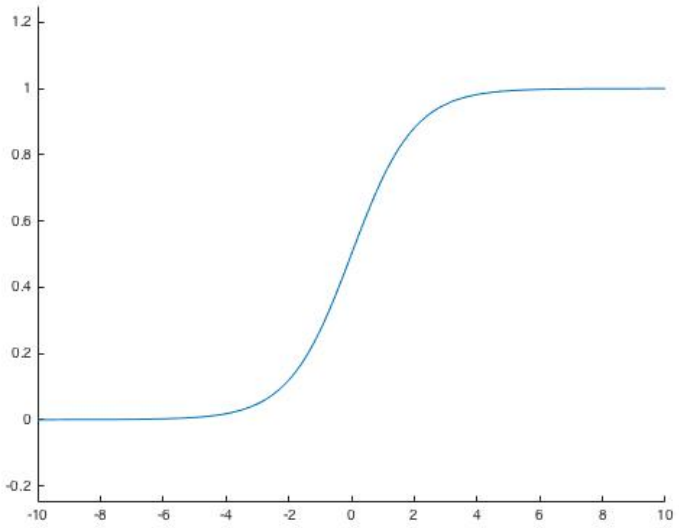
The output vector $g_k = Wx_k$ corresponding to a single input x_k is a continuous valued vector. In order to match this output to the target vector with binary values $\{0, 1\}$ a heavyside function should be ideally be used. However, we shall see that we need a smooth function that has a derivative. A squashing function called a sigmoid (see figure 3) is an acceptable approximation for the Heavyside function

$$g_{ik} = sigmoid(x_{ik}) = \frac{1}{1 + e^{-z_{ik}}} \;\; i = 1, \ldots, C \quad (3.20)$$

Here we have $z_k = Wx_k$ (which was previously called g_k). Further,

the index i points to vector element number i

Figure 3.7: *Sigmoid*



A linear classifier with binary targets thus has the structure given in figure 3

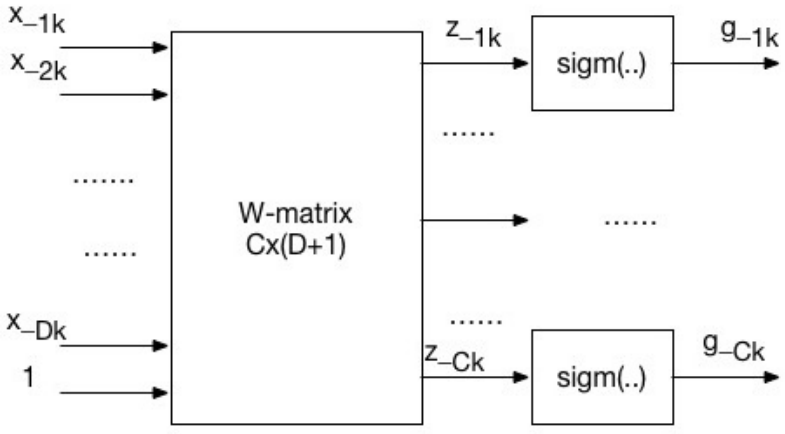


Figure 3.8: *Sigmoide*

As for the GMM-based Plug-in-MAP classifier there exists no explisit solution to equation (3.19). One therefore has to use a gradient based technique, i.e. update the W matrix in the opposite direction of the gradient. Using the chain rule for gradients we get

$$\nabla_W MSE = \sum_{k=1}^{N} \nabla_{g_k} MSE \; \nabla_{z_k} g_k \nabla_W z_k \tag{3.21}$$

It is easily shown that

$$\begin{aligned} \nabla_{g_k} MSE &= g_k - t_k \\ \nabla_{z_k} g &= g_k \circ (1 - g_k) \\ \nabla_W z_k &= x_k^T \end{aligned}$$

where $g_k \circ (1 - g_k)$ means elementwise multiplication

Innserting the above gradients into equation (3.21) we get

$$\nabla_W MSE = \sum_{k=1}^{N} [(g_k - t_k) \circ g_k \circ (1 - g_k)] x_k^T \tag{3.22}$$

In order to reduce the MSE we have to move W in the opposite direction of the gradient, i.e.

$$W(m) = W(m-1) - \alpha \nabla_W MSE \tag{3.23}$$

where m gives the iteration number.

The step factor α is an important constant which must be chosen with care. A too large value will give a corresponding large change in W and results in large fluctuations in the MSE. Too small value will demand unnecessary many iterations in order to even get close to the minimum MSE. There exists however methods which will adapt the step factor during training. However often a "cut and try" technique will lead to satisfactory results.

The equation (3.22) tells us to use the whole training set of size N in each iteration, i.e. a so called "batch" based iteration. Often one prefers to update the matrix W more frequently. Thus one can choose to update based on subsets of size $L << N$. Thus is called a "semibatch" strategy. The extreme variant is when $L = 1$, i.e we update for each input/output value. This last method is called "online" or "realtime" updating. Which to use is typically application dependent. The step factor should increase as L decrease.

Training of a template based classifier using clustering

Training of a template based classifier is the same as finding good references. The simplest approach is not to train at all but instead to use the whole training set as templates. The main objection with this is the amount of processing, i.e. one has to calculate N distances for every new input x. Further it is not possible to find reference specific covariance matrixes Σ_{ik} (see equation (3.8)), i.e. one has to use the simplified distance versions described in subchapter 2.5.

A clearly suboptimal procedure is to use a subset $M << N$ of the references randomly drawn from the total training set. This will reduce the processing amount but will not help with regard to the lack of the reference covariances and thus the possible distance types. The subset is normally chosen such that the relative amount of references per class $M_i \;\; i = 1, C$ reflects the original class amounts N_i (i.e. the prior $P(\omega_i)$) in the total training set.

Drawing references randomly can lead to a reference set that is not representative for one or more classes. Thus a much better strategy is to use a method which guarantees good coverage. This method, called **clustering**, estimates both the necessary number of references, the references themselves and also the reference specific covariance matrixes.

Most clustering algorithms works on one class at a time. There exist some discriminative versions; however in the following we will concentrate of the class-for-class method. Thus we simplify our notation by omiting the class index i, i.e. $N_i \;\rightarrow\; N\,, M_i \;\rightarrow\; M\,, \mu_{ij} \;\rightarrow\; \mu_j$, $\Sigma_{ij} \;\rightarrow\; \Sigma_j$, $\;j = 1, M$. Note that a cluster and a reference means the same in the following!

The iterative procedure for clustering is relatively simple. It is based on successive increasing the number of clusters (i.e. number of class references)

1. Start with a single cluster $M = 1$

2. Use equations (3.16) and (3.17) to calculate $\Lambda_1 = \{\mu_1, \Sigma_1\}$ and the corresponding accumulated distance $D_1 = \sum_{k=1}^{N} d(x_k, \Lambda_1)$ where the distance is defined in subchapter 2.5.

3. Increase number of clusters $M = M + 1$. Split $\Lambda_1 \rightarrow \{\Lambda_1, \Lambda_2\}$ (see below for splitting technique.)

4. Then we iterate to find the best values of the M references. For q =1,... (iteration counter for a given cluster size M)

   - If $q = 1$ set $D_{M_0} = \infty$ (in practice a large number)
   - Classify each training set vector in the class to the closest of the M clusters and calculate the accumulated distance D_{M_q}.
   - If D_{M_q} is "clearly smaller" then the previous $D_{M_{q-1}}$ use the classification labels from above and the equations (3.16) and (3.17) to calculate/update $\Lambda_i \;\; i = 1, \ldots, M$. Increase $q = q + 1$ and loop within step 4.
     Else $D_M = D_{M_{q-1}}$ and go to next step 5

5. If D_M is "clearly smaller" than D_{M-1} increase number of references/cluster $M = M + 1$ by splitting.
   Else the clustering is finished for the specific class

Note that using the equations (3.16) and (3.17) leads to that the corresponding references do not exist as training vectors. However, this is usually not required. If for some reason this is needed, one simply replaces each reference by the closest training vector of the same class.

We did use the term "clearly smaller" twice in the algorithm above. Note that for each iteration q in step 4 the classification usually lead to some change of class labels as the references was modified in the previous iteration. This is equivalent to $D_{M_q} = \beta D_{M_{q-1}}$ where $\beta < 1$. Usually the algorithm converges towards a non-significant change in a few iterations, corresponding to a β close to 1. Usually one terminates when $\beta \approx 0.99$. In the extreme case we can experience no change at all in class labels, thus $D_{M_q} = D_{M_{q-1}}$.

The general form for splitting $M \rightarrow M + 1$ is based on that we first find the cluster Λ_s to split (in step 3 this is given). The most popular choice is to use the cluster which has most vectors classified to it. If reference specific covariance matrixes are used, an alternative is to choose the cluster with largest matrix norm. After choosing cluster Λ_s we define the new cluster, i.e. Λ_{M+1} by slightly modifying the reference $\mu_{M+1} = \mu_s + w$ where w is a uniformly distributed noise vector where the elements obey $\delta < w(j) < \delta \; j = 1, \ldots, dim(x)$ hvor $\delta << 1$. The covariance matrix is just copied $\Sigma_{M+1} = \Sigma_s$.

The above clustering algorithm is well suited for generating references for a template classifier based on a deterministic approach, i.e. without statistical terms like densities aso. However we can

"easily" generalize the algorithm to work in a statistical framework, more precisely to estimate the parameters in a GMM-based Plug-in-MAP classifier. We then need to introduce the so called "soft decision" by using the Posteriori probabilities $P(C_i/x_k) = \gamma_{ik}$. We will not evaluate how to find these values, but obviously they tells us how well the input x_k matches a given cluster C_i $i = 1, \ldots, M$ on a scale from zero to one. Instead of classifying x_k to belong to a specific cluster (see point 4 in the clustering algorithm above), we calculate the probability of the same. Thus any input will have probability larger than zero for belonging to any cluster. This we can use to weigh the contribution of x_k to all clusters; i.e. the equations (3.16) and (3.17) are modified to

$$\hat{\mu}_i = \frac{\sum_{k=1}^{N} \gamma_{ik} x_k}{\sum_{k=1}^{N} \gamma_{ik}} \tag{3.24}$$

$$\hat{\Sigma}_i = \frac{\sum_{k=1}^{N} \gamma_{ik} (x_k - \mu_i)(x_k - \mu_i)^T}{\sum_{k=1}^{N} \gamma_{ik}} \tag{3.25}$$

Note that the sums now will be over all $N = \sum_i N_i$ training vectors x_k. Also note that equations (3.24) and (3.25) become identical to (3.16) and (3.17) if

$$\gamma_{ik} = \begin{cases} 1 & x_k \in C_i \\ 0 & x_k \in C_j \neq C_i \end{cases} \tag{3.26}$$

The equations (3.16) and (3.17) are therefore often called "hard" decision" (or sometimes "winner-takes-all"), i.e. the vector x_k only belongs to the closest cluster

The clustering algorithm described above was based on distances. However, the equations (3.16), (3.17), (3.24) and (3.25) are based on statistics, i.e. maximization of a likelihood (ML). One can show that an ML-based approach for deriving optimal GMMs for a plug-in-MAP classifier leads to exactly the equations (3.24) and (3.25). In this case a reference is replaced by a Gaussian. This clustering variant was developed from a statistical viewpoint and therefore is named the **Expectation-Maximation (EM)** algorithm".

Finally, one should mention the existance of clustering variants which are correlated to the total classification error rate. This type of clustering works simultaneously on the total training set (all classes) and tries to find class references which minimize the error rate on the training set. The most known of these methods are named "The learning vector quantizer - LVQ".

Examplifying using two data sets.

To illustrate clustering and the different classifier strategies given in this chapter we will use two kinds of data examples.

The first is a case were we generate the data as **Gaussian sources/classes** by using the command "randn" in Matlab. We generate 100 2-dimensional samples of each of four Gaussians symmetrically centered around the origo, i.e. with means respectively $[-1\ -1], [+1\ -1], [-1\ +1], [+1\ +1]$. In the clustering example we use all four sources, while we in the classifier cases use only two of the sources.

In the second example we use **real data** from the "classical" Iris flower case. There are three Iris classes (Setosa, Versicolour and Virginica). All three classes have two different flowers (sepal and petal) which differ in length and width, i.e. the features are 4-dimensional vectors. We have 50 samples of each class. For plotting purpose we only use 2 of the 4 dimensions. The two data sets are plotted in figure 3

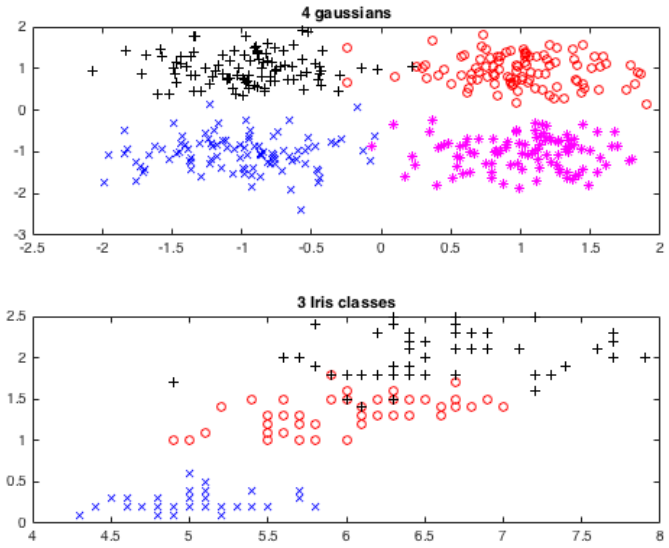


Figure 3.9: *The two data sets used as examples*

Clustering the two example data sets

We start by regarding a data set as unlabeled, i.e. we have no information on which class each sample belongs to. Thus our goal is just to find structures in the data which will lead to a natural division into a number (which also must be found) of clusters. Note that in all clustering cases we us the standard Euclidian distance, i.e. equation (3.9).

For the Gaussian data set the figure 3 shows the cluster centres when increasing the number of clusters from one to five. In this case it is visually easy to see that four clusters match the source data well. In the general case it is not possible to do such a visual inspection, either due to the input dimension or more overlapping data. However, by monitoring the accumulated/total distance, it is possible to decide upon a reasonable number of clusters. The lower right plot in figure 3 has several horisontal parts. They correspond

to when the distance have stopped decreasing, i.e. a stable situation for a given number of clusters. Thus the distance between two consequtive horisontal parts gives the improvement in distance by adding an extra cluster. We see that this improvement stops when we go from four to five clusters.

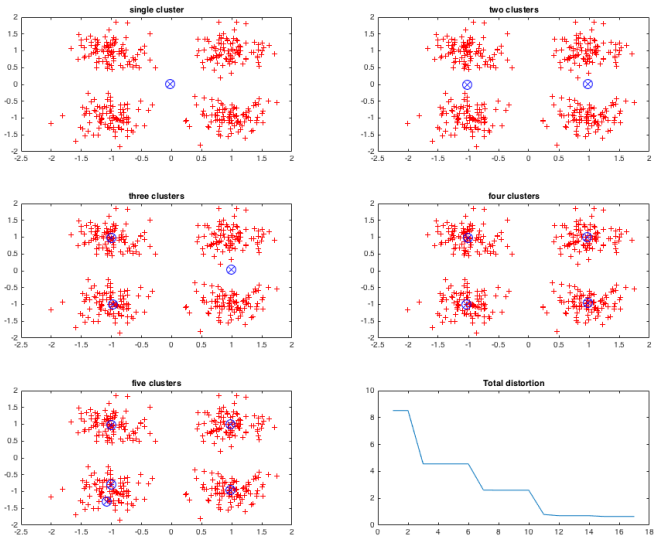


Figure 3.10: *Clustering the Gaussian data*

In the Gaussian case (figures 3 and 3) the clusters are well separated. The degree of overlap (standard deviation) will of course influence the clustering. In figure 3 four different standard deviation are used for the Gaussian sources . The solid lines (Voronoi lines) show the resulting **cluster borders** and a cluster is called a Voronoi cell. We see that the cluster borders deviate from horizontal/vertical lines when the overlap increases. We also see that the cluster centres deviate from the ”true” source centers as a function of the overlap.

Figure 3 shows the same clustering process, but now for the Iris data. This time we see that three clusters are a reasonable choice. This corresponds to to the number of Iris classes, thus indicating that the clustering follows the class labeling to a great extent. Note that we now can not vary the standard deviation as we work with real data!

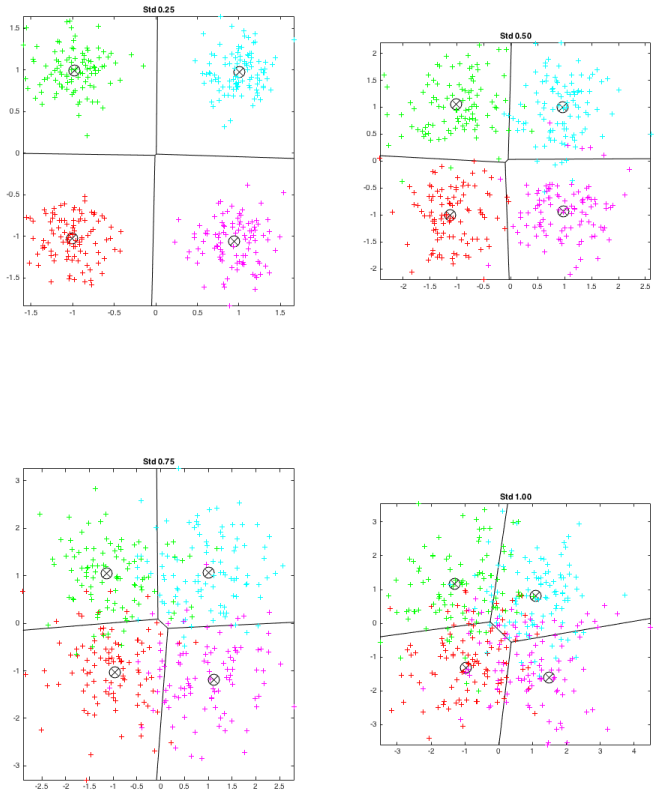


Figure 3.11: *Clustering as a function of degree of data overlap*

Using single Gaussian class models for classifying the data sets
We now use data from two of the four Gaussian sources and try to classify the data by choosing a single Gaussian model for each class (as described in subchapter 3.1.1). Note that this choice gives a match between the data sources and the class models as both are single Gaussians. To simplify we use all data for training. In figure 3 the results are plotted for different standard deviations (variances) for the sources. The ellipses give the contours for equal likelihood while the black curve shows the decision border. Note that when the standard deviation is equal we have a linear separable problem. Thus we ideally could use a linear classifier, however even the Gaussian classifier gives a decision border which approximately is a straight line. However, the border gets more and more curved as the difference in variance increases and the problem becomes nonseparable. Further, when we in addition have that the variance is comparable to the distance between the means, the decision border is an ellipse illustrating that the class with smallest variance is surrounded by the wider class. Else we see, as expected, that the two class models fit the data well. Further, the class with largest

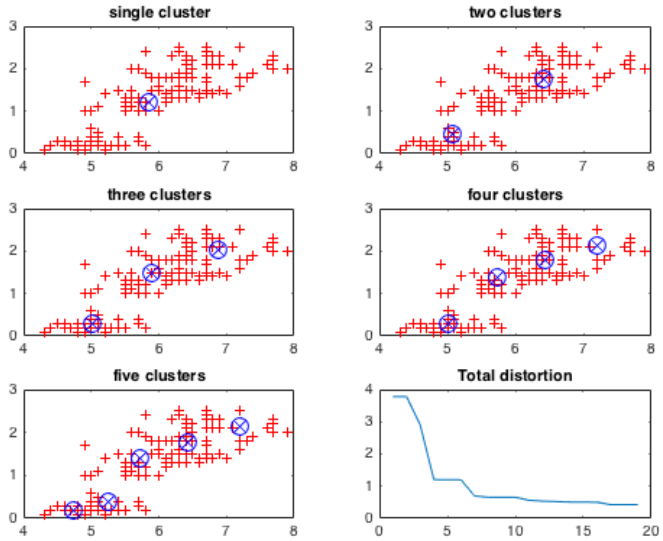


Figure 3.12: *Clustering the Iris data*

variance is responsible for more or less all the errors.

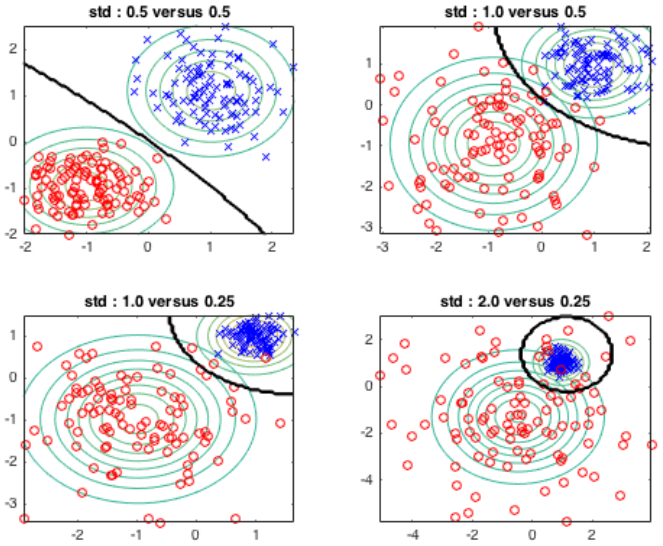


Figure 3.13: *Single Gaussian sources and class models showing likelihood contours and decision border*

We then use the same single Gaussion models for two "lowest" of the three Iris classes using only the two input dimensions as given by figure 3. Note that we of course have no control of the variance in this case, as the data are real measurements. The model contours and the decision border is given by figure 3. We can easily observe from the data samples that this is a linear separable case. However, figure 3 shows that this is not the case if we include the third class.

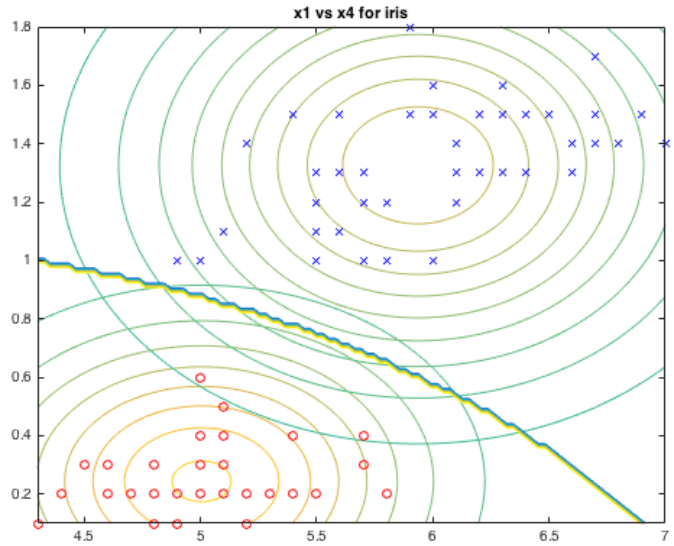


Figure 3.14: *The Iris case for two classes using single Gaussian class models showing likelihood contours and decision border*

The linear classifier (perceptron)

The linear classifier is described in subchapter 3.4.4. In order to visualize the decison borders we choose the Iris data with only two features as shown in figure 3, thus the border will be a line. Further we choose only two classes. We further divide the data set into a training part of 30 samples and a test set of 20 samples for each of the two classes. The border line is shown in figure 3 both for the training set) and the test set. In figure 3 we show the MSEs and the error rates for the training and test set as a function of the number of iterations (gradient updates).

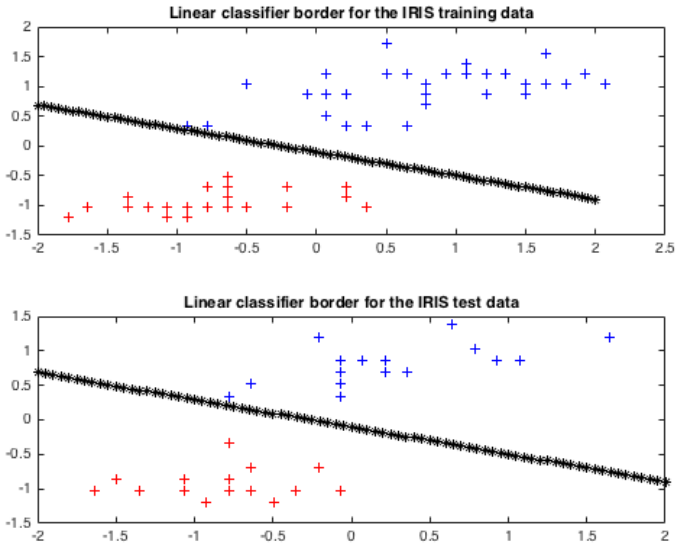


Figure 3.15: *Linear classifier border for the IRIS case*

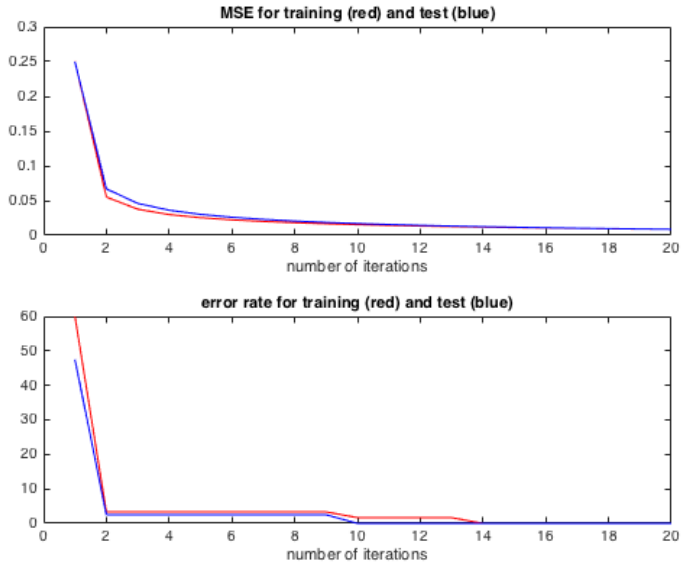


Figure 3.16: *MSE and error rate curves for the linear classifier*

The reference based classifier

In subchapter 3.3 we discussed the two methods for generating references for a template based classifier. The first method was simple; namely using all training samples as references. This method is discussed and illustrated in subchapter 3.4.1 and figures 3, 3 and 3. The second method uses clustering to find a small set of good references. We will focus on this last method in the following. Note the following differences between standard clustering and reference generation. In standard clustering we do not have any class labels for the training samples. The goal is to split the training set into ”natural” clusters. However, in reference based clustering the training samples are labeled, and the goal is to find some representative references for each class. Thus we split the training samples into class subsets and perform standard clustering for **each class** separately. The figures 3 and 3 show the the case where a **single** reference is used for each class. Here, the references in both cases are found by the sample mean, i.e. equation (3.16). Note that we in this single reference case onlyuse the two first steps in the clustering algorithm. The full algorithm is of course used when we want more than one reference per class.

Evaluation of classifiers

The Gaussian source example was described initially in subchapter 3.4. This is an artificially generated problem, thus we know exactly the parameters of the sources and can thus calculate **the true/theoretical minimum error rate (MER)**. Such artifically data are very useful for evaluating classifiers as we know the MER as un upper bound wrt performance. However, we will of course also wish our classifiers to have an error rate close to the MER for prac-

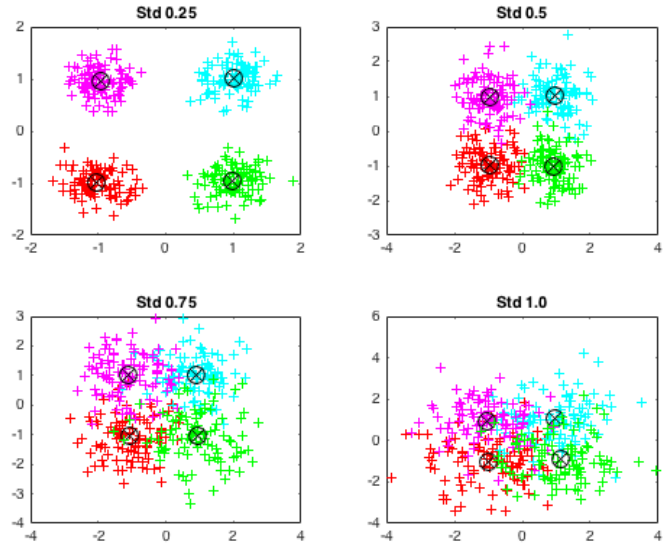


Figure 3.17: *Single references for the four classes of Gaussian data*

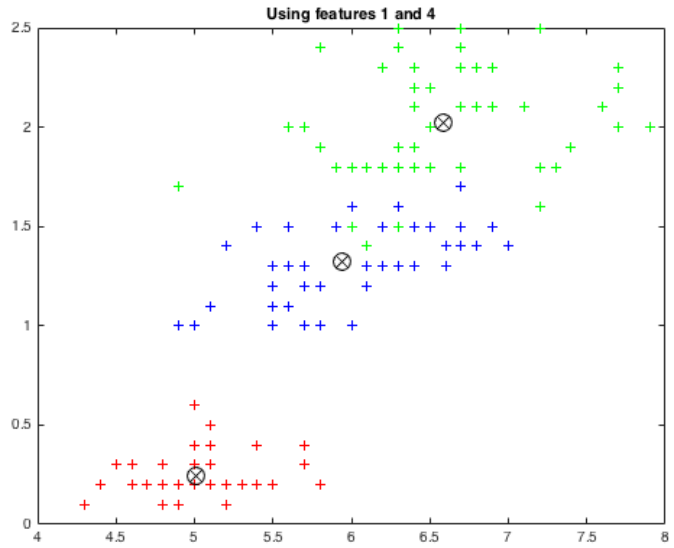


Figure 3.18: *Single references for the three classes of Iris data*

tical problems (real data). Thus we must be able to estimate the true error rate for our chosen classifier. This subchapter deals with how to do this.

For practical cases we have an additional problem. We do not know the true class distribution of the sources/features, thus we can not calculate the MER. However, we still need to calculate the error rate for our chosen classifier for any practical problem! In both cases we will need a large set of N_D samples for training/development and N_T samples for testing. Further we need to have an estimator (formula) for the error rate. A logical choice is

just to count all the errors E in the data set and divide by the data set size N! We will call this the **estimated error rate** in the following.Thus the estimated error rates for respectively the training and test sets are given by $EER_D = E_D/N_D$ and $EER_T = E_T/N_T$. Note that it is EER_T which is the estimated error rate for the classifier.

We will describe the uncertainty/variation in classifier design and performance by using the example of classifier using single Gaussian class models (as described in subchapter 3.1). First assume that we have **two different training sets** of the same size N_D. It is clear that the sample means and covariances (as given by equations (3.16) and (3.17)) will have different values for the two resulting classifiers. Thus the error rate on the (same) test set will therefore differ for the two classifiers. Now let us assume a single training set but **two different test sets** of equal size N_T. In this case we have a single classifier but we will of course get two different estimated error rates as we have two test sets. Thus we see that the bf true (but unknown) error rate depends on the training set (and of course the classifier structure), while the resulting **estimated** error rate depends on the test set.

Then let us analyze the impact of the data set **sizes**. We will use some extreme examples to illustrate this. Let us start by the training set. We assume that we have only a single sample. Thus the estimated mean will be identical to this value, but the estimated variance is impossible to calculate! If we increase the size to two samples we will find estimated values for all parameters. However, choosing two different samples as training set will probably lead to quite different means and variances. From this we can infer that we should wish as many training samples as possible, i.e. ideally should $N_D \rightarrow \infty$. In most cases we will experience an error rate reduction as the training set increases in size. However, we can **not guarantee** that the estimated test error rate decrease consistently with the training set size! This is especially a problem when we choose the "wrong type" of classifier, i.e. in our example the sources are far from Gaussian distributed. Then let us analyse the test set size. Again we start by only one sample. Thus we have only two possible estimated error rates, namely 100% or 0%. If we increase to two samples we additionally have the possibility of 50% error rate. We further easily understand that using two other test samples can lead to a different error rate, ie. a difference in minimum 50%. Again we see that this difference decreases as the test set size increases. Of course the ultimate situation is where $N_T \rightarrow \infty$! Thus we should use a biggest possible test set.

How well do the classifier generalize?

In a practical world the classifier is first trained and tested (by labeled data) and then is set to work on real, unlabeled data. The goal is of course to design a classifier which will give the same error rate on the real data as the test data. However as the real data (of course) is unlabeled we can not calculate/estimate the corresponding error rate! We therefore choose the following strategy : if

the difference in performance between the test set and the training set, $|EER_T - EER_D|$ is "small enough" we can assume the same for the difference between the test set and the real data. If this former difference is small we say that the classifier "generalizes" well. This strategy can also be used to choose between different types of classifiers. A classical mistake wrt generalization is to choose a complex classifier in combination with a small training set. Thus the classifier will learn "too well" the exact training set values (i.e. often error free $EER_D \approx 0$). As a consequence the error rate on the test set will become significantly larger than for the training set. This estimated error rate difference is therefore a good indication of how well the classifier generalizes. An absolute difference on $2-4\%$ indicates a good generalization while more than 10% normally is not acceptable. Note, however that normally we have $EER_D < EER_T$ as the classifier has "learnt" the training data during training/design.

Finally, note that a professional design also requires a third data set called a validation set. The corresponding error rate is used during training as an indication when to stop training. Thus the EER_T is not used during training!

True versus estimated error rate

In the above we use the term "estimated" error rate for a specific test set on a given, trained classifier. We also discussed the difference between this and the true, but unknown, error rate for the classifier. It turns out that it is possible to calculate a "$\pm$ interval" centered on the estimated error rate which with "high" probabilty will include the true error rate. A standard way of reporting the performance is to use a socalled 95% interval, i.e. claiming that with 95% certainty the true error rate will be in this interval. The interval is a function of two parameters, i.e. the value of the estimated error rate and the size of the test set. Figure 3 shows intervals for test set sizes ranging from $N_T = 10$ to $N_T = 1000$ for the (unknown) true error rate p as a function of the estimated error rate $\hat{p}$ (i.e. EER_T). The interval corresponds to the uncertainty/variance of the estimate, and thus shrinks as a function of increasing test set size N_T.

The leave-one-out strategy

In the introduction to this subchapter we discussed the sizes of the training and test sets. We found the simple rule; the bigger the better. However, quite often one has only access to a limited amount of labeled data. Further this data has to be split up into a training and a test part (and ideally also a validation set ...). The Iris case is a typical example of this problem, as it consists of only 50 samples per class. Using most (or nearly all) for training will give the best classifier but the estimated error rate on the test set will have a large uncertainty as seen in figure 3. The "leave-one-out" strategy is a suboptimal technique to circumvent this problem. For the Iris case this method suggests using 49 samples for training and only one sample for test per class. We however has to do this 50 times, i.e. systematically changing the test sample for each time.

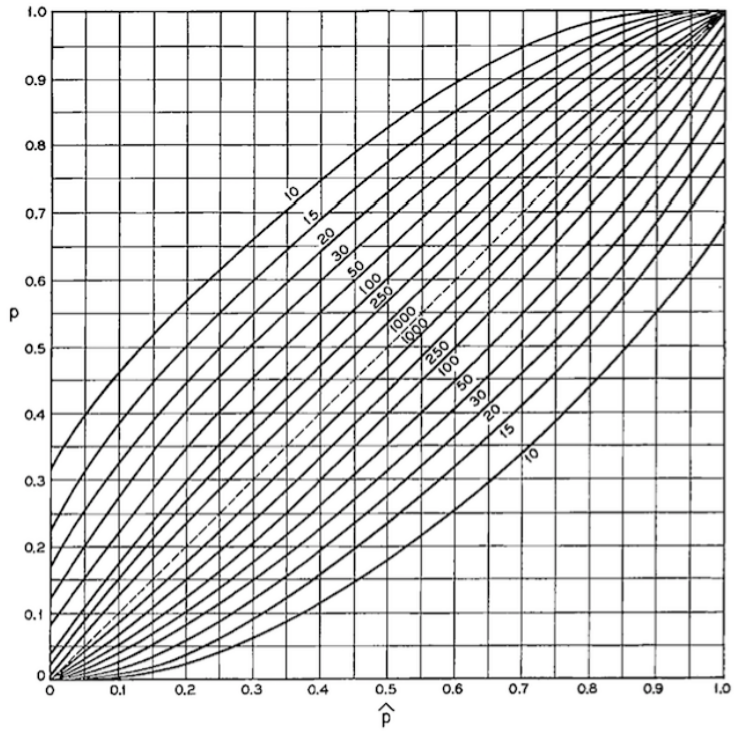


Figure 3.19: 95% *confidence intervals for the error rate as a function of test set size*

The drawback is that the classifier will be slightly different for each time. The estimated error rate is then the mean of the error rate for each of the 50 experiments. This method will give a larger confidence interval than if we had 50 test samples to apply to the *same* classifier. However, the estimated error rate is usually closer to the true error rate than if we for instance used 30 samples for training and 20 for test. Note that one has a huge amount of alternatives for this strategy, i.e one could chose to use all possible combinations of $50 - x$ training and x test samples, where $x = 2, 3, 4, \ldots$.

The concept of significance

Above we discussed the impact of **finite** training and test sets. We argued for that this led to uncertainty both in generalization and performance estimation. Here we will introduce another concept, namely **significance** which is closely related to the error rate confidence interval. Now assume we have two different classifiers which we want to rank using the same test set. We find the corresponding values $EER1_T$ and $EER2_T$. Assume $EER1_T < EER2_T$, should this indicate that the classifier1 is the best? The answer to this is yes only if the difference is significant. What if we had another test set which gave us the opposite result? The probability of this happening is of course highest when the differences are small, i.e. socalled insignificant. Thus we can use Figure 3 in another way. If the difference in the estimated error rate for the same test set is larger than the confidence interval this indicates with 95% certainty that classifier1 is better than classifier2.

The confusion matrix

In the start of this subchapter on evaluation we defined the estimated error rate which is the ”total” error rate. Many times we would like to monitor both the performance for the individual classes and the class confusibility, i.e. which classes is ”closest” in the input room. To do this it is common to use a socalled confusion matrix. In table 3 a confusion matrix is given for the case of four Gaussian classes where class 1 and class 2 is somewhat closer than the other class distances. The test set consists of 50 samples for each of the four classes. Note that the matrix shows both the number of samples which are correctly classified and the errors. The total error rate is $ERR_T = 14/200 = 7\%$ while the error rate for class 2 is $6/50 = 12\%$ Further we see that class 1 and class 2 is the most confusable, which was expected from the placement in the input room. Often the numbers in the matrix is given in percentage (especially when we have large sample numbers). The same results will then have the form as in table 3.

Classified : $\rightarrow$ True/label : $\downarrow$	Class 1	Class 2	Class 3	Class 4
Class 1	47	3	0	0
Class 2	6	44	0	0
Class 3	0	1	48	1
Class 4	1	1	1	47

Table 3.1: Confusion matrix for four Gaussian classes - 50 samples per class

Classified : $\rightarrow$ True/label : $\downarrow$	Class 1	Class 2	Class 3	Class 4
Class 1	94	6	0	0
Class 2	12	88	0	0
Class 3	0	2	96	2
Class 4	2	2	2	94

Table 3.2: Confusion matrix in percentage for four Gaussian classes - 50 samples per class

4
Bibliography

[1] P.J. Bickel and K.A. Doksum (1977), *Mathematical Statistics - Basic Ideas and Selected Topics*. San Francisco: Holden-Day.

[2] P. Billingsley (1979), *Probability and Measure*. New York: John Wiley & Sons.

[3] S.A. Kassam (1988), *Signal Detection in Non-Gaussian Noise*. New York: Springer-Verlag.

[4] S.M. Kay (1993), *Fundamentals of Statistical Signal Processing - Estimation Theory*. Upper Saddle River, New Jersey: Prentice-Hall.

[5] S.M. Kay (1998), *Fundamentals of Statistical Signal Processing - Detection Theory*. Upper Saddle River, New Jersey: Prentice-Hall.

[6] M. G. Kendall and A. Stuart (1977), *The Advanced Theory of Statistics*, Vol. II. New York: Macmillan Publishing Co.

[7] A. Papoulis and S. Unnikrishna Pillai (2002), *Probability, Random Variables and Stochastic Processes*, 4th Edition. New York: McGraw-Hill.

[8] H.V. Poor (1988), *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag.

[9] L.L. Scharf (1990), *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. New York: Addison-Wesley Publishing Co.

[10] S.D. Silvey (1975), *Statistical Inference*. London: Chapman and Hall.

[11] Henry Stark and John W. Woods, (2002) *Probability, Random Processes and Estimation Theory*, 3rd Edition. Upper Saddle River, New Jersey: Prentice-Hall.

[12] H.L. Van Trees (1968), *Detection, Estimation, and Modulation Theory*. New York: John Wiley & Sons, Inc.

[13] C.L. Weber (1987), *Elements of Detection and Signal Design* (a reprint of the McGraw-Hill Edition (1968)). New York: Springer-Verlag.