In this chapter we discuss how the transmitted symbols are detected optimally from a noisy received signal (observation).

Based on these results, optimal receiver structures are later developed for the modulation methods presented earlier.

The same theoretical framework is the basis for optimal solutions for the following problems:

1. Symbol detection from a received noisy continuous-time signal, or from the corresponding discrete-time sample sequence.

   We consider first the simplified case of detecting an isolated symbol in case of an AWGN channel model.

   Then symbol sequence detection in case of an ISI channel is considered.

2. Decoding of error correction codes.

3. Parameter estimation, like estimating the signal amplitude, carrier phase, symbol timing, and other synchronization parameters of the receiver.

The Viterbi algorithm is a commonly used efficient implementation algorithm for the optimal sequence detection principle.

Source: Lee&Messerschmitt, Chapter 7.
Terminology and System Model

Here we consider mostly a discrete-time system model.

The term estimation is used when we want to know the value of a continuous variable.

The term detection is used when we want to decide which of possible values from a discrete set of values is the correct one.

The system model includes two aspects:

- **Signal generation model**: Deterministic processing of the signal, like coding, modulation, filtering.
  
  In theoretical considerations, the channel response is assumed to be known through channel estimation, so for example, the ISI produced by multipath channel belongs to the signal generation model.

- **Noise generation model**: Random, usually additive noise/distortion effects.

  We consider in the continuation two main cases:
  
  - Random bit errors produced by a binary symmetric channel (BSC) to a bit sequence.
  - Additive Gaussian noise.

There are two general detection/estimation principles:

- **ML** = maximum likelihood
- **MAP** = maximum a-posteriori.
Detection of an Isolated Real-Valued Symbol Based on a Discrete-Valued Observation

The receiver makes a decision about the transmitted symbol $A$ based on an observation $Y$.

The input is a symbol $A$, randomly chosen from the alphabet $\Omega_A$.

The output $Y$ is a discrete-valued random variable, which is distorted due to noise generation.

The system is described by the conditional probabilities

$$p_{Y|A}(y|\hat{a}).$$

An ML detector chooses the symbol $\hat{a} \in \Omega_A$ which maximises $p_{Y|A}(y|\hat{a})$ when there is the observation $y$ from the output $Y$.

A MAP detector maximises the a-posteriori probability

$$p_{A|Y}(\hat{a}|y).$$

The latter criterion is intuitively better. As will be seen shortly, it also minimizes the error probability.
About ML and MAP Detection

Based on the Bayes rule, the ML and MAP detectors are related through

\[ P_A|Y(\hat{a}|y) = \frac{p_{Y|A}(y|\hat{a})p_A(\hat{a})}{p_Y(y)} \]

In MAP detection, it is sufficient to maximize the numerator, since the denominator is invariant in the maximization. So, to be able to do MAP detection, the symbol probabilities \( p_A(\cdot) \) must be known.

In the important special case that the symbols are equally probable, the MAP detector is equivalent to the ML detector.

The ML detector is easier to implement than the MAP detector, and it is commonly used in practise. Also the symbol probabilities are usually equal. The transmitter may include a special function (scrambling or energy dispersal) to satisfy this requirement. In some cases the symbol probabilities might not be known, but it is sensible to assume them to be equally probable.
About the Optimality of MAP Detection

In general, the probability of correct decision can be written as

\[
\Pr[\text{correct decision}] = \sum_{y \in \Omega_Y} \Pr[\text{correct decision}|Y = y]p_Y(y)
\]

\[
= \sum_{y \in \Omega_Y} p_{A|Y}(\hat{a}|y)p_Y(y)
\]

Since the probabilities \(p_Y(y)\) are always non-negative, this sum is maximized when each of its terms is maximized. And this is exactly what a MAP detector is doing. So we can say:

The MAP detector minimizes the error probability.

As mentioned earlier, in most practical cases the symbol probabilities are equal, and in these cases also the ML detector minimizes the error probability.
An Example about ML and MAP Detection

(a) Equally probable symbols

\[ Y = A + N \]

\[ a \in \{0,1\} \quad p_A(0) = p_A(1) = 0.5 \]

\[ n \in \{0,1\} \quad p_N(0) = p_N(1) = 0.5 \]

\[ A \text{ and } N \text{ are independent} \]

\[ \Rightarrow y \in \{0,1,2\} \]

\[ p_{Y|A}(0|0) = 0.5 \quad p_{Y|A}(0|1) = 0 \]

\[ p_{Y|A}(1|0) = 0.5 \quad p_{Y|A}(1|1) = 0.5 \]

\[ p_{Y|A}(2|0) = 0 \quad p_{Y|A}(2|1) = 0.5 \]

An ML detector makes the decisions as follows:

\[ y = 0 \Rightarrow \hat{a} = 0 \]

\[ y = 1 \Rightarrow \hat{a} = 0 \text{ or } \hat{a} = 1 \text{ (e.g. randomly)} \]

\[ y = 2 \Rightarrow \hat{a} = 1 \]

Since the symbols are equally probable, the MAP detector works in the same way.

(b) Case with different symbol probabilities

\[ p_A(0) = 0.75 \quad p_A(1) = 0.25 \]

An ML detector works as before.

In the cases of \( y = 0 \) and \( y = 2 \) the MAP detector works as above, but in the case of \( y = 1 \) there is a difference. The a-posteriori-probabilities are

\[ p_{A|Y}(0|1) = 0.75 \quad p_{A|Y}(1|1) = 0.25 \]

so the MAP detector chooses now \( \hat{a} = 0 \) when \( y = 1 \).
An Example about ML and MAP Detection (cont.)

The error probabilities can be evaluated as follows:

(a) ML and MAP detection work in the same way:

\[ y = 0 \quad \text{with probability } 0.25; \quad \text{error probability } 0 \]
\[ y = 1 \quad \text{with probability } 0.5; \quad \text{error probability } 0.5 \]
\[ y = 2 \quad \text{with probability } 0.25; \quad \text{error probability } 0 \]

\[ \Rightarrow \text{average error probability } 0.25 \]

(b) In cases of \( y = 0 \) or \( y = 2 \) no errors.

ML:

\[ y = 1 \quad \text{with probability } 0.5; \quad \text{error probability } 0.5 \]
\[ \Rightarrow \text{average error probability } 0.25 \]

MAP:

\[ y = 1 \quad \text{with probability } 0.5; \quad \text{error probability } 0.25 \]
\[ \Rightarrow \text{average error probability } 0.125 \]

So we can see that the MAP detector works clearly better in this situation.
Binary Symmetric Channel, BSC

The BSC model is commonly used in error control coding studies.

Binary symbols are transmitted:

\[ \Omega_A = \{0,1\}, \quad p_A(0) = q, \quad p_A(1) = 1 - q \]

ML detector assuming \( p < 0.5 \):
\[ \hat{a} = y \]
\[ \Pr[\text{error}] = p \]

MAP detector:

Case \( y = 0 \):
\[ p_{A|Y}(0|0) = q(1 - p) \]
\[ p_{A|Y}(1|0) = (1 - q)p \]
\[ \hat{a} = \begin{cases} 0 & \text{if } q > p \\ 1 & \text{if } q < p \end{cases} \]

Case \( y = 1 \):
\[ p_{A|Y}(0|1) = qp \]
\[ p_{A|Y}(1|1) = (1 - q)(1 - p) \]
\[ \hat{a} = \begin{cases} 1 & \text{if } q + p < 1 \\ 0 & \text{if } q + p > 1 \end{cases} \]
Detecting a Single Real-Valued Symbol Based on a Continuous-Time Observation

The observation is now \( Y = A + N \) where \( N \) is an additive continuous-valued noise sample.

The Bayes rule can now be written as:

\[
P_A|Y(\hat{a}|y) = \frac{f_Y|A(y|\hat{a})p_A(\hat{a})}{f_Y(y)}
\]

where \( f_Y|A(\cdot|\cdot) \) and \( f_Y(\cdot) \) are now continuous probability distributions.

ML and MAP detection are defined in the same way as earlier.
Additive Gaussian Noise Case

(a) The ML detector uses the following probability distributions:

\[ f_{y|a}(y|-a) \]
\[ f_{y|a}(y|+a) \]

The threshold is at 0.

(b) The MAP detector uses the following probability distributions:

\[ f_{y|a}(y|-a)p_a(-a) \]
\[ f_{y|a}(y|+a)p_a(+a) \]

The threshold is now at the point where the two probability distributions cross each other.
Another Example with Uniform Noise PDF

\[ \Omega_A = \{-1,1\}, \quad p_A(-1) = 0.25, \quad p_A(+1) = 0.75 \]
\[ Y = A + N \]

\( N \) has uniform distribution:

MAP detector compares the following probability functions:

In the range \(-0.5 < y < 0.5\) it chooses \( \hat{a} = 1 \). The threshold is at 0.5. The error probability is 1/12.

ML detector compares the following probability functions:

In the range \(-0.5 < y < 0.5\) it chooses (e.g. randomly) either of the two symbols. Or it may set a threshold somewhere in this range. Depending on the chosen threshold,

\[ 1/12 < \text{error probability} < 1/3. \]
Detecting a Signal Vector

In the following, we consider detecting a sequence of symbols optimally, taking eventually into account also the intersymbol interference. We consider the symbol sequence as a vector-valued symbol, which can be detected using the ML or MAP principles.

A simple example of vector-valued signal is an I/Q symbol in baseband format, the real and imaginary parts being the elements of the vector.

But in the following, the vector elements are the (real- or complex-valued) symbols of a finite-length symbol sequence. We can denote the signal vector as

$$S = [A_1, A_2, A_3, \ldots, A_M]$$

where $M$ is the number of symbols in the vector.

Any transmission link can be considered as a vector channel using the following model:

![Diagram of a vector channel](image)

The vector dimensionality may be defined by the application in a natural way, but in other cases it can be chosen freely for analysis purposes.
Detecting a Signal Vector (continued)

In case of signal vector detection, the observation vector has the same dimensionality as the signal vector. The noise process is determined by the conditional probability distributions

\[ f_{Y|S}(y|s) \].

The noise components of the observation vector are usually assumed to be independent (in the Gaussian case, this is satisfied if the components are white, i.e., uncorrelated). In this case we can write

\[ f_{Y|S}(y|s) = \prod_{k=1}^{M} f_{Y_k|S_k}(y_k|s_k) \]

ML-detection selects the symbol vector \( \hat{s} \) that maximizes the conditional probability \( f_{Y|S}(y|\hat{s}) \). As earlier, the statistics of the transmitted signals needs not be taken into account.
ML-Detection in Case of Additive Gaussian Noise

Assuming additive Gaussian noise with variance $\sigma^2$ and with uncorrelated samples, we can write the conditional probability as

$$f_{Y|S}(y|\hat{s}) = f_{N|S}(y - \hat{s}|\hat{s}) = f_N(y - \hat{s})$$

$$= \frac{1}{(2\pi)^{M/2} \sigma^M} \exp \left[ -\frac{1}{2\sigma^2} \|y - \hat{s}\|^2 \right]$$

ML-detector chooses the vector $\hat{s}$ that maximizes the above function. This is clearly equivalent to minimizing the Euclidean distance

$$\|y - \hat{s}\|$$

(or equivalently the squared Euclidian distance $\|y - \hat{s}\|^2$)

In the additive Gaussian noise case, ML-detector chooses the signal vector $\hat{s}$ that is closest to the observation vector $y$ using the Euclidean distance measure. This will be the leading principle in the following!

The elements of the signal vector may be also complex-valued. The (squared) Euclidean distance measure to be minimized can be written as

$$\|y - \hat{s}\|^2 = |y_1 - \hat{s}_1|^2 + \cdots + |y_M - \hat{s}_M|^2$$
ML-Detection in Case of BSC

In this case, the elements of the signal vector and the observation vector are binary-valued.

The conditional probabilities in the BSC case are

\[
p_{y|s}(y|s) = \begin{cases} 
   p & \text{if } y_k \neq \hat{s}_k \\
   1 - p & \text{if } y_k = \hat{s}_k 
\end{cases}
\]

**Definition:** The Hamming distance \(d_H(\hat{s}, y)\) of two (equal-length) binary vectors \(\hat{s}\) and \(y\) is equal to the number of elements where the vectors differ from each other.

Now we can write the conditional probability for the binary signal vector of length \(M\) as (independence assumption !)

\[
p_{Y|S}(y|s) = p^{d_H(\hat{s}, y)} (1 - p)^{M - d_H(\hat{s}, y)}.
\]

This is the probability for the case that the differing bits have been changed by that channel, but the others not.

In practice, \(p < 0.5\) and the ML-detector minimises \(d_H(\hat{s}, y)\).

In the BSC case, ML-detector chooses the signal vector \(\hat{s}\) that is closest to the observation vector \(y\) using the Hamming distance measure.
About ML-Detection in the AWGN and BSC Cases

In ML detection, basically the same rule is used in both of the previously discussed cases: The signal vector that is closest to the observation vector is chosen.

The only difference is the used distance measure, either Euclidean or Hamming.

In both cases, the set of possible observation vectors can be divided into decision regions, which include the possible observation vectors that are closest to each of the signal vectors.

The earlier presented rule for detecting PAM/QAM/PSK symbols in case of ISI-free channel is one important case of this general criterion.
An Example of MAP-Detection

Let us consider again the additive Gaussian noise case. A vector MAP-detector maximizes the probability

\[ f_N(y - \hat{s}) p_S(\hat{s}) = \frac{1}{(2\pi)^{M/2} \sigma^M} \exp \left( -\frac{1}{2\sigma^2} \|y - \hat{s}\|^2 \right) p_S(\hat{s}) \]

This is equivalent to minimizing the following function:

\[ \|y - \hat{s}\|^2 - 2\sigma^2 \ln(p_S(\hat{s})) \]

Also in this case, the space of possible observed signal vectors can be divided into decision regions, but the thresholds depend on the noise variance and the signal vector probabilities.

In the optimal case, the noise variance should be estimated. Another possibility is to design the MAP-detector for typical or worst case situation. In any case, the implementation is more complicated than in ML-detection.

It should be noted that the possible observation vectors are usually not equally probable, even if the individual symbols are. Therefore, in (non-trivial) sequence detection, ML-detection may not be such a good approximation for MAP as in the case of isolated symbol detection.

The practical sequence detection techniques are usually based on the ML principle, but also the MAP-based ideas have a role in advanced developments. However, in the continuation we consider only the ML techniques.
Error Probability Evaluation: (i) Two Signal Vectors, Gaussian Noise

Let us consider first the case of two $M$-dimensional signal vectors, $s_i$ and $s_j$.

The noise produces an $M$-dimensional vector $N$, the elements of which are zero-mean Gaussian-distributed random variables, the variance of which is $\sigma^2$.

A sequence detection error occurs if $S = s_i$ is transmitted and $\hat{s} = s_j$ is detected.

To simplify notation, we can assume the special case with $s_i = 0$ and $s_j = q$. Assume now that $s_i = 0$ is transmitted, thus the received vector $y = s_i + N = 0 + N = N$. An error occurs if $\|y - q\|^2 < \|y - 0\|^2$ or if $\|N - q\|^2 < \|N - 0\|^2$. This is equivalent to

$$(N_1 - q_1)^2 + \cdots + (N_M - q_M)^2 < N_1^2 + \cdots + N_M^2$$

or

$$q_1N_1 + \cdots + q_MN_M > \frac{q_1^2 + \cdots + q_M^2}{2}.$$  

In vector form this can be expressed as:

$$q'N > \frac{\|q\|^2}{2} = \frac{d^2}{2}$$

where $d$ is the Euclidean norm of $q$. 
Error Probability Evaluation: (i) Two Signal Vectors, Gaussian Noise (continued)

Above $q^tN$ is a linear combination of independent Gaussian variables, and thus it has the variance

$$\left(q_1^2 + \cdots + q_M^2\right)\sigma^2 = d^2\sigma^2$$

The error probability becomes

$$Q\left[\frac{d^2/2}{d\sigma}\right] = Q\left[\frac{d}{2\sigma}\right]$$

The error probability of binary PSK derived earlier is a special case of this.

This result can be generalized for two arbitrary vectors:

If the observation vector is $Y = s_i + N$, where the elements of the noise vector $N$ are zero-mean Gaussian random variables with variance $\sigma^2$, then the error probability is given by

$$\Pr\left[\|Y - s_j\| < \|Y - s_i\|\right] = Q\left[\frac{d}{2\sigma}\right]$$

where $d$ is the Euclidean distance of the two vectors,

$$d = \|s_i - s_j\|.$$
Error Probability Evaluation:
(ii) Two Signal Vectors, BSC

Now we consider the case of two \( M \)-dimensional binary signal vectors, \( s_i \) and \( s_j \) that have a Hamming distance of \( d \), i.e., they differ in \( d \) bits.

The conditional probabilities are now of the form:

\[
p_{Y|S}(y|\hat{s}) = p^{d_H(\hat{s},y)}(1 - p)^{M - d_H(\hat{s},y)}.
\]

When \( s_i \) is transmitted, ML-detector chooses \( s_j \) if

\[
d_H(s_j, y) \leq d_H(s_i, y)
\]

Further it is assumed that in a boarder case (with same distance to both signal vectors), the detector always makes the wrong decision.

We can also note that errors in those bits where the two signal vectors are the same do not matter.

A sequence detection error occurs if there are more than \( t \) errors in those bits, where \( s_i \) and \( s_j \) differ. Here \( t \) is largest integer for which \( 2t < d \), i.e.,

\[
t = \begin{cases} 
   (d - 1)/2 & \text{if } d \text{ odd} \\
   d/2 - 1 & \text{if } d \text{ even} 
\end{cases}
\]
Error Probability Evaluation:  
(ii) Two Signal Vectors, BSC (continued)

Now the error probability can be written as:

\[ Q(d, p) = \sum_{i=t+1}^{d} \binom{d}{i} p^i (1 - p)^{d-i} \]

Here, this Q-function is based on the binomial probability distribution.

The result is similar to the additive Gaussian noise case, now only the Q-function is based on a different probability distribution.

**Example:**

Let \( s_i = [000000], s_j = [110111] \).

Here the Hamming-distance is 5.

A detection error occurs if there is an error in at least three of the five bits that are different in the two signal vectors. The error probability is

\[
Q(5, p) = \binom{5}{3} p^3 (1 - p)^2 + \binom{5}{4} p^4 (1 - p)^1 + \binom{5}{5} p^5 \\
= 10 p^3 (1 - p)^2 + 5 p^4 (1 - p)^1 + p^5
\]
Error Probability Evaluation:
(iii) Many Signal Vectors

Now there are \( K \) signal vectors, \( \{s_1, \ldots, s_K\} \) in use. We assume that \( s_i \) is transmitted and the observation vector is \( Y \). Let \( E_j \) represent the error case where the observation is closer to \( s_j \) than \( s_i \).

With \( K=3 \), the error probability can be written as

\[
\Pr[\text{error}|s_1 \text{ transmitted}] = \Pr[E_2] + \Pr[E_3] - \Pr[E_2 \cap E_3].
\]

In practise, we can ignore the last term that has only a minor effect. In the general case, we obtain an upper bound using the union bound principle as:

\[
\Pr[\text{error}|s_i \text{ transmitted}] \leq \sum_{j=1}^{K} \Pr[E_j]
\]

On the other hand, the most probable error event, \( E_{\max} \), gives a (rather tight) lower bound:

\[
\Pr[\text{error}|s_i \text{ transmitted}] \geq \Pr[E_{\max}]
\]
The following approximation, which depends on the minimum distance, $d_{\text{min}}$, is commonly used:

$$\Pr[\text{error} | s_i \text{ transmitted}] \approx \Pr[E_{\text{max}}]$$

where

$$\Pr[E_{\text{max}}] = Q(d_{\text{min}} / 2\sigma)$$ for additive Gaussian noise case
$$\Pr[E_{\text{max}}] = Q(d_{\text{min}}, p)$$ for BSC case

The overall average error probability is obtained as a weighted average of the error probabilities of the different signal vectors (weighted by the probabilities of the signal vectors).

Usually, the different signal vectors are equally probable, and the overall error probability is given by

$$\Pr[\text{error}] = \Pr[\text{error} | s_i \text{ transmitted}].$$

So we have obtained lower and upper bounds, and an approximative formula for the error probability in vector detection.

The BSC case and the additive Gaussian noise case can be handled using very similar principles.

The error probability depends essentially on the minimum distance and noise variance!
Discrete-Time Correlator and Matched Filter

Here we apply the previously developed theory for the detection of an isolated symbol in case of a dispersive channel with additive Gaussian noise.

\[ a \delta_k \rightarrow h_k \rightarrow N_k \]

The finite-length impulse response \( h_k, 0 \leq k \leq M \), modeling both the transmit filter and the channel response, causes dispersion to the pulse. The symbol and channel response are here real-valued.

An observation vector \( Y = h \alpha + N \) of dimension \( M+1 \) is obtained by sampling the received signal. In this example case, the sample rate is assumed to be higher than the symbol rate.

ML-detector chooses the symbol \( \hat{\alpha} \) of the used alphabet that minimizes the Euclidean distance

\[
\| y - h \hat{\alpha} \|^2 = \| y \|^2 - 2 \langle y, h \rangle \hat{\alpha} + \| h \|^2 \hat{\alpha}^2
\]

where

\[
\langle y, h \rangle = y^T h = \sum_{k=0}^{M} y_k h_k
\]

This is equivalent to maximizing the following expression:

\[
2 \langle y, h \rangle \hat{\alpha} - \| h \|^2 \hat{\alpha}^2
\]
Discrete-Time Correlator and Matched Filter (cont’d)

The first term can be calculated using a correlator:

$$\langle y, h \rangle = \sum_{k=0}^{M} y_k h_k$$

or, equivalently, using a matched filter:

$$\langle y, h \rangle = \sum_{k=0}^{M} y_k h_k = \left( \sum_{k=0}^{M} y_k h_{n-k} \right)_{n=0}$$

The impulse response of the matched filter is the mirror image of the received pulse shape in case of real signals (or complex conjugate mirror image in case of complex channel model.)

We see here the equivalency of correlation receiver and matched filter receiver as optimum receiver principles. The equivalency is due to the close relation between correlation and convolution operations. These principles will be discussed further later on.
Optimum ML-Receiver for Detecting an Isolated Symbol

Here the matched filter is implemented in causal form, with delay $M$. (In the structure of the previous page, the non-zero taps are with indices $-M, \ldots, -2, -1, 0$.)
Example of an Optimum ML-Receiver for Detecting an Isolated Symbol

$$\Omega_A = \{0, 1\}, \quad h_k = \delta_k + 0.5 \delta_{k-1}$$

$$M = 1, \quad \|h\|^2 = 1.25$$

Receiver structure:

Simplified structure:

The signal vectors are: $$s_1 = [1, 0.5]$$ and $$s_2 = [0, 0]$$. The minimum distance of these determines the error probability:

$$\Pr[error] = Q(d / 2\sigma) = Q(\sqrt{1.25} / 2\sigma)$$

This is for the case of isolated symbol detection. In practice, when detecting a symbol sequence, the ISI makes the performance worse.
About Markov Chains

A Markov chain $\{\Psi_k\}$ is a discrete-time and discrete-valued random process that satisfies the condition:

$$p(\Psi_{k+1}|\Psi_k, \Psi_{k-1}, \ldots) = p(\Psi_{k+1}|\Psi_k)$$

$\Psi_k$ is the state of the Markov chain at time $k$. It depends only on the previous state.

Markov chains are used, e.g., to model finite state machines with random inputs. In the following, they are used as the signal generation model for ISI channels, and later for modeling certain type of coding methods.

A Markov chain is homogenous if the conditional probability $p(\Psi_{k+1}|\Psi_k)$ does not depend on $k$. In this case, the system is in a way stationary or time-invariant. The properties of homogenous Markov chains are determined by the state transition probabilities

$$p(j|i) = p_{\Psi_{k+1}|\Psi_k}(j|i).$$
Shift Register Process as a Markov Chain

**Shift register process:**

Here $X_k$ is a discrete-valued random variable, which is independent of the variables $X_{k-1}, \ldots, X_{k-M}$. The state is defined as:

$$
\Psi_k = \{X_{k-1}, \ldots, X_{k-M}\}.
$$

It can be shown that such a system satisfies the Markov condition. This is a vector-valued Markov chain.
Using Shift Register Process to Model ISI Channels

Let us consider the detection of a symbol vector (or symbol sequence) in a case of a channel that has ISI. The ISI is defined by the symbol-rate channel impulse response $h_k$.

A signal with ISI can be modeled as a homogenous Markov chain, using the shift register process:

The state is defined by the contents of the delay line, $\Psi_k = \{X_{k-1}, \ldots, X_{k-M}\}$, and the output of the signal generation model can be formulated as $S_k = g(\Psi_k, \Psi_{k+1})$, where $g(\cdot, \cdot)$ is a memoryless function. The random variables $X_i$ are independent and identically distributed.

The impulse response $h_k$ is usually finite in length and then

$$S_k = \sum_{i=0}^{M} h_i X_{k-i}$$

As usual, the additive noise is assumed to have independent samples.
**Example of ISI Channel**

We consider a continuous-valued channel with impulse response

\[ h_k = \delta_k + 0.5\delta_{k-1} \]

This has the following Markov chain model (left part):

Here the state is the previous transmitted symbol \( A_{k-1} \). On the right, a corresponding state transition diagram for the system is given. The arcs are labeled with noise-free input/output pairs \((A_k / S_k)\).

In other words, each node of the diagram corresponds to a state of the Markov chain. The noise-free input and output values of the signal generation model are shown for each branch (state transition) of the diagram.
**Trellis Diagram**

Trallis diagram is another type of presentation for Markov chains, here shown for the previous example case:

At each stage of the diagram, the nodes correspond to the nodes of the state diagram, i.e., the all the possible states of the Markov chain.

The whole trellis shows all the possible transitions between states at consecutive symbol intervals, each section of the trellis corresponding to one symbol interval.

A trellis diagram for symbol sequences of length $K$ has initial state $\Psi_0$, $K$ sections of trellis, and a final state $\Psi_K$.

Between the initial state and the final state there are a number of paths, each corresponding to a specific input symbol sequence of length $K$. The branches corresponding to such a path indicate also the output sequence $\{S_k; k = 1, \ldots, K\}$ of the signal generation model. This would be the corresponding noise-free received signal after the ISI channel in question.
Sequence Detector

Continuing the previous continuous-valued, additive Gaussian noise case, the observed sequence at the detector would be $y_k = s_k + n_k$. A sequence detector aims at determining the transmitted symbol sequence based on such an observation.

When doing the detection, the signal generation model is assumed to be completely known; in practical systems the ISI channel must somehow be estimated to be able to utilize the idea.

According to the ML vector detection principle, one should choose the sequence that is closest to the observed sequence according to proper metric, i.e., the Euclidean metric in our example case.

The candidate sequences are the sequences produced by the signal generation model as responses to all the possible symbol sequences (with the used alphabet) of length $K$.

For each possible symbol sequence, there is a unique path through the trellis, and the corresponding ideal sequence, produced by the signal generation model, is determined by the path.

Thus, the ML vector detection problem is equivalent to finding the path through the trellis that is closest to the observed sequence according to the used metric!
Given an observed sequence, a path metric is defined for each path through the trellis. It is the distance between the corresponding ideal sequence and the observed sequence, using the relevant metric.

Each section of the trellis corresponds to one symbol interval, with observed sample $y_k$. The branch metric is defined for each branch $j$ of each section $k$ as

$$
\begin{cases}
|y_k - s^j_k|^2 & \text{additive Gaussian noise case} \\
H(y_k - s^j_k) & \text{BSC case}
\end{cases}
$$

A path metric is obtained by adding up the branch metrics corresponding to the path.

An ML sequence detector could be realized by
- Calculating all the branch metrics.
- Calculating all the path metrics out of the branch metrics.
- Finding out the path with the smallest path metric.
- The symbol sequence corresponding to this path is the optimally detected sequence accoring to the ML vector detection principle.

However, such a scheme would be very complicated to implement for long symbol sequences.
Viterbi Algorithm

Basically, Viterbi algorithm is an application of dynamic programming to solve the previously described path minimization problem efficiently.

The principle is as follows:

1. The trellis is examined section by section, starting from the initial state and ending at the final state.

2. At the beginning of each phase, the shortest paths (surviving paths) from the initial node to the input nodes of the section under examination are known. Then the shortest paths from the initial node to all the output nodes of the section are searched. They are found by calculating the path metrics of all possible combinations of surviving paths and branches of the current section, and choosing the shortest path to each output node.

3. This is continued until we reach the final state. At this point, only one path is surviving, and this corresponds to the detected sequence.

At each stage, there are \( N \) surviving paths, where \( N \) is the number of parallel states in the trellis, i.e., the number of states in the Markov model.

The Viterbi algorithm is based on the following property: If the shortest path goes through node \( x \), then the metric of this path is the sum of the two shortest paths, one from initial node to \( x \) and one from \( x \) to final node.
An Example of Viterbi Algorithm

We continue the previous example with signal generation model

\[ h_k = \delta_k + 0.5\delta_{k-1} \]

and additive Gaussian noise. We assume that the received sequence is \{0.2, 0.6, 0.9, 0.1\}.

The following trellis diagram shows all the branch metrics. Then the surviving paths are shown for each phase of the Viterbi algorithm.
So the Maximum Likelihood Sequence Detection (MLSD) gives as the detected symbol sequence \( \{0, 1, 0, 0\} \), whereas a simple symbol-by-symbol slicer ignoring the ISI would give \( \{0, 1, 1, 0\} \). The knowledge of ISI if present is, of course, useful!
About the Viterbi Algorithm

During the processing of each section of the trellis (except for the first and last sections), a constant amount of computation is carried out. So the computational complexity is proportional to the sequence length $K$.

In the presented form, the Viterbi algorithm gives the detected sequence after all the sections have been processed. Partial detection of the first symbols before the final result often takes place (like in the previous example), but is never guaranteed. This could introduce excessive detection delays when detecting a long sequence.

This can be avoided by defining truncation depth, $d$, such that when processing section $k$, possible parallel surviving paths starting earlier than section $k-d$ are removed by keeping only the shortest of those paths. This means that the maximum detection delay is $d$ symbols.

Choosing the truncation depth large enough, the detection performance is not significantly degraded. In this way, the Viterbi algorithm can be used even for infinitely long symbol sequences.

Viterbi algorithm can be used for all ML and MAP detection problems, where the signal generation can be modelled as a homogenous Markov chain. Important special cases include:

- Detection after a channel with ISI and additive Gaussian noise.
- Decoding of certain error control codes (convolution codes and trellis codes to be discussed later).
Evaluation of Error Probabilities

In case of Viterbi detection/decoding, we can talk about an error event, which means single ‘detour’ of the detected path outside the right path.

The length of an error event is the number of wrong nodes, or number of trellis sections with wrong path –1.

As an example, error events of lengths 1 and 2:

In this context we can talk about detection errors. In sequence detection, a detection error may produce several symbol errors, an error burst, the length of which depends on the length of the error event.

The detection error probability can be evaluated using the earlier formulas for ML sequence detection:

\[ KQ(d_{\text{min}} / 2\sigma) \quad \text{for additive Gaussian noise channels} \]
\[ KQ(d_{\text{min}}, p) \quad \text{for BSC} \]

Here \( K \) is a coefficient that is not very easy to be determined but can often assumed to be 1, and \( d_{\text{min}} \) is the minimum distance, i.e., the shortest distance between the right path and the ‘detour’, over all possible error events. Notice that in this context, the path lengths are calculated using the branch values corresponding to the ideal signal generation model, not using the actual branch metrics used in detection.
**Example:** In the example case considered earlier in Viterbi detection, the shortest error events are:

The length 1 and 2 error events have the following distances, respectively:

\[
\sqrt{1^2 + 0.5^2} = \sqrt{1.25} \quad \text{and} \quad \sqrt{1^2 + 1.5^2 + 0.5^2} = \sqrt{3.5}
\]

So the minimum distance in this example is

\[
d_{\text{min}} = \sqrt{1.25}
\]

This can be verified by checking all the (short) error events.

The minimum distance in this example is, more or less by change, equal to the minimum distance in isolated symbol detection with the same channel, that was discussed earlier. However, with ISI channels, the performance in sequence detection is usually much worse than in isolated symbol detection.

In sequence detection, it is usually not easy to find the minimum distance by inspection. However, a modification of the Viterbi algorithm can be used for that purpose, as described in the Lee&Messerschmitt book.
Evaluation of Error Probabilities *(continued)*

It should be noted that when calculating the detection error probabilities, there is no distinction whether the error produces a single symbol error or long burst error, so relating the detection error probability and BER/SER is not an easy task.

In fact, when the sequence length grows, the sequence detection error probability approaches 1.

Since BER/SER, or the probability of a short block of bits/symbols is what matters, it turns out that ML sequence detection is not the best choice, after all.

There are also methods that minimize the BER or SER in sequence detection, but they tend to be rather complicated and not often used in practise. Furthermore, with reasonably high SNR’s, ML sequence detection gives almost as good performance as the best BER/SER minimizing techniques.