Maximum Likelihood Sequence Detection (MLSD) and the utilization of the Viterbi Algorithm

Presented to

Dr. Tareq Al-Naffouri

By

Mohamed Samir Mazloum

Omar Diaa Shawky

Abstract

Signaling schemes with memory involve symbols which depend on prior ones when transmitted. Examples of these include NRZI (non-return to zero inverted) code and convolutional codes. Decoding such symbols can be done by various methods, including the MLSD (Maximum Likelihood Sequence Detector) which utilizes the Viterbi Algorithm as a means of reducing the time needed to decode the received symbols.

Introduction

Signaling schemes can be classified as being memoryless or with memory when examining the interdependence between the transmitted symbols. For memoryless signals, the transmitted symbols are independent of one another, and thus the detection of each one takes place alone, using detection schemes such as the MAP (maximum a posteriori) rule or the ML (maximum likelihood) rule (Viterbi algorithm). However, with schemes that include memory, transmitted symbols are interdependent, where each transmitted symbol depends on which symbol was sent before it. In this case, detection of a symbol involves observation of earlier symbols, tracing a path along the possible branches that lead to a certain output in order to find which path minimizes the probability of error. As is the case with the memoryless schemes, detection of the received symbols takes place through ML or MAP. This paper focuses on the application of MLSD for the detection of signaling schemes that utilize memory, and how the Viterbi algorithm is essential for reducing the time involved with applying the ML detector. Two examples of such schemes are discussed; the NRZI (non return to zero inverted) and convolutional codes, and the detection of each using the MLSD and Viterbi are then discussed.

The Viterbi Algorithm

The Viterbi Algorithm is a dynamic programming algorithm that is used to find the most likely sequence of hidden states, which is called the Viterbi path. This algorithm depends on a number of assumptions:

1- The algorithm works on a state machine assumption, meaning that at any point in time, the system in question is being modeled in some state. It also assumes that the number of states, however large, is ultimately finite.

2- Multiple sequences of states (called paths) can lead to a particular state, but only one path is the most likely, and that is called the ‘survivor’ path, which is the only one kept by the Viterbi algorithm. Thus the algorithm keeps only one path per state.
3- Transition from one state to another is accompanied by an incremental metric, which is usually a number. Over a certain path, such metrics are cumulative (usually additive).

Hence, the main objective of the algorithm is that when an event occurs, the algorithm observes that a number of states could follow the most recent one, and so it combines the metrics of possible previous states, and picks the best one. After computing the combinations of incremental metric and state metric, only the best survives and all other paths are discarded (Veeravalli). There are modifications to the basic algorithm which allow for a forward search in addition to the backwards one described here.

An important part of the Viterbi algorithm is path history. In some cases, the search history is complete because the machine has enough memory to keep all the previous states, however, in more realistic limited situations, some programming solutions are used. One example is truncating the history when decoding convolutional codes, in order to keep an acceptable performance level.

The Maximum Likelihood Sequence Detector

In MLSD the transmitted signal sequence is represented in a path in the trellis, where the number of messages in equal to the number of paths throughout the trellis. We assume that the transmitted signal has duration of K symbol intervals, and each path of length K is considered as a message. The Maximum Likelihood Sequence Detector chooses the most likely path according to the received signal $r(t)$ over the K intervals. The Maximum Likelihood detector selects the path that would results in the minimum Euclidean distance between the path and the received signal $r(t)$.

Since

$$\int_0^{KT_s} |r(t) - s(t)|^2 dt = \sum_{K=1}^{K} \int_{(K-1)T_s}^{KT_s} |r(t) - s(t)|^2 dt \quad (5)$$

The optimum decision rule becomes

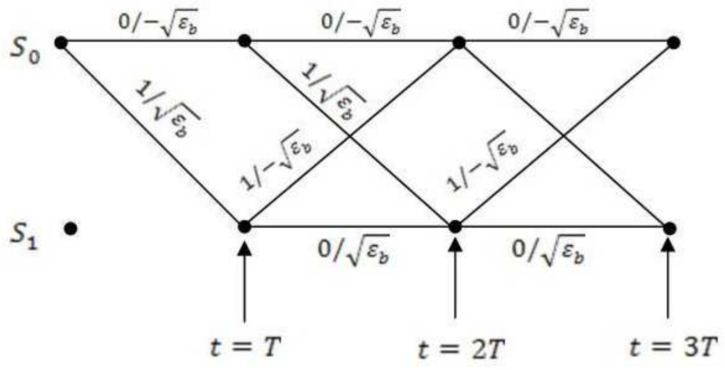
$$\left(\hat{s}^{(1)},\hat{s}^{(2)},\dots,\hat{s}^{(K)}\right) = \arg min_{\left(\hat{s}^{(1)},\hat{s}^{(2)},\dots,\hat{s}^{(K)}\right)\in \Upsilon} \sum_{K=1}^{K} \left\| r^{(K)} - s^{(K)} \right\|^2$$

$$= \arg min_{\left(\hat{s}^{(1)},\hat{s}^{(2)},\dots,\hat{s}^{(K)}\right)\in \Upsilon} \sum_{K=1}^{K} D\left(r^{(K)}, s^{(K)}\right)$$

Where D denotes the Euclidean distance and Υ denotes the trellis.

When searching through the trellis for the most likely sequence we evaluate the Euclidean distance at all the nodes with all possible sequence, then we use the Viterbi sequential trellis search algorithm for performing ML sequence detection to eliminate sequences does not carry the minimum Euclidean distance(Proakis 243).

Example:



We consider the example of a NRZI signal where if the sent signal is different from the received signal then the state is one, and when the sent signal is equal to the received then the state of the NRZI is equal to zero. We consider the NRZI using the Pulse Amplitude Modulation scheme, therefore, $s_1 = -s_2 = \sqrt{\varepsilon_b}$ where $\varepsilon_b$is the energy per bit, and the total number of sequences in this trellis is2^K. However, after using the Viterbi algorithm the number of sequences is going to be reduced.

In this example, we assume that the initial sent signal was s_0 at $t = 0$. At time $t = T$ the signal received is $r_1 = s_1{}^{(m)} + n$ and at $t = 2T$ the signal received is $r_2 = s_2{}^{(m)} + n,$ since the size of the symbols just 1 bit so the trellis reaches steady state after 2 T, also notice in the figure above that when the trellis reaches steady state that two paths entering each of the nodes and two paths leaving each node. At $t = 2T$ there are two paths entering node S_0 (0, 0) and (1, 1) or resulting from the signal points $(-\sqrt{\varepsilon_b}, -\sqrt{\varepsilon_b})$ and $(\sqrt{\varepsilon_b}, -\sqrt{\varepsilon_b})$, at the same time $t = 2T$ there are two paths entering node S_1 (0, 1) and (1, 0) or resulting from the signal points $(-\sqrt{\varepsilon_b}, \sqrt{\varepsilon_b})$ and $(\sqrt{\varepsilon_b}, \sqrt{\varepsilon_b})$. The calculated Euclidean distance for distance for paths entering S_0 at t=2T

$$D_0(0,0) = (r_1 + \sqrt{\varepsilon_b})^2 + (r_2 + \sqrt{\varepsilon_b})^2$$

$$D_0(1,1) = (r_1 - \sqrt{\varepsilon_b})^2 + (r_2 + \sqrt{\varepsilon_b})^2$$

Then the Viterbi algorithm computes the Euclidean, compares the paths, and discards the path having the larger metric. The path having the lower metric and Euclidean distance is saved and called the survivor path.

The calculated Euclidean distance for distance for paths entering S_1 at t=2T

$$D_1(0,1) = (r_1 + \sqrt{\varepsilon_b})^2 + (r_2 - \sqrt{\varepsilon_b})^2$$

$$D_1(1,0) = (r_1 - \sqrt{\varepsilon_b})^2 + (r_2 - \sqrt{\varepsilon_b})^2$$

Also using Viterbi algorithm, we choose the survivor path and then observe the next signal r_3 and calculate its path metric using the previous survivors of the trellis. At t=3T S_0 has two metrics for the paths entering:

$$D_0(0,0,0) = D_0(0,0) + (r_3 + \sqrt{\varepsilon_b})^2$$

$$D_0(0,1,1) = D_1(0,1) + (r_3 + \sqrt{\varepsilon_b})^2$$

The two paths are compared using the Viterbi algorithm, the survival path is used, and the other one discarded, then we compute the two metrics for the paths entering at S_1

$$D_1(0,0,1) = D_0(0,0) + (r_3 - \sqrt{\varepsilon_b})^2$$

$$D_1(0,1,0) = D_1(0,1) + (r_3 - \sqrt{\varepsilon_b})^2$$

The two paths are compared using the Viterbi algorithm, the survival path is used, and the other one discarded. This process is repeated with each new observed signal, this way the number of paths searched in the trellis is reduced by a factor of two at each stage.

Hard Decision Decoding vs. Soft Decision Decoding

The MLSD uses one of two types of decision hard decision or soft decision. Hard decisions where the decoding takes a stream of bits, then compares them with a certain threshold value and determines the results as a certain value. For instance, In Binary signaling the received waveforms are sampled, and the resulting samples are compared to a single threshold value, which would produce one of two results. If the voltage of the sample is higher than the threshold it would be considered a one, if the voltage of the sample is lower than the threshold it would be decoded as a zero regardless how far the sample is from the threshold. Soft decisions are when decoding of the stream of bits results not only the one and zero decision, but also indicates how certain are we of the decision is correct. The results in a 3-bit encoding, this reliability information is represented by 000 to represent a strongest 0, 001 representing a relatively strong 0, 010 representing a relatively weak 0, 011 represent the weakest 0, 100 represent the weakest 1, 101 represents a relatively weak 1, 110 represents a relatively strong 1 while 111 represents the strongest 1. We consider the last two of the three bits as confidence bit or reliability bits. In soft decisions, the three bits are represented by using 2^3 voltage thresholds points, however in hard decisions; the single bit is passed by one threshold point to represent the two different states. The Viterbi can deal with soft decision with almost the same efficiency as hard decisions with a little addition in complexity such as using 3 bits instead of one in the hard decision, however, the decision is usually found out to be much better with the greater reliability of soft decision(Cheetham).

Convolutional codes

Convolutional codes are an example of signaling schemes with memory, where the transmitted symbols depend on earlier symbols. The implementation of this code takes place by utilizing a linear finite-state shift register through which the information sequence is passed. The shift register consists of K(called the constraint length) stages (each of which taking k bits) and for each k bit input, n bits of output are emitted from the encoder. Thus, the code rate for such an encoder is k/n.

In order to better describe convolutional codes, we utilize a number of vectors equivalent to the number of modulo-2 adders that form the outputs of the encoder (n). Each vector indicates which stages of the encoder are connected to the corresponding adder, where an '1' represents a connection, and a '0' indicates none. For example, a vector [100] indicates that out of a 3 stage shift register, only the first stage is connected to the adder.

After realizing the vectors representing the adders within the encoder, the output of the encoder is determined by convolving the input bit sequence with each of the vectors, then interleaving the bits from each convolution process.

$c^{(1)} = u * g_1 \qquad c^{(2)} = u * g_2 \qquad c^{(3)} = u * g_3$

where g_1, g_2, g_3 are the vectors representing the adders.

This process can be mathematically simplified by realizing that convolution in the time domain corresponds to multiplication in the frequency (transform) domain. A common transform used in coding literature is the D transform, where D denotes the unit delay introduced by one memory element in the shift register. This transform has a strong relation to a well known transform, the z transform, where

$$D = Z^{-1}$$

In this case, the vectors [100], [101] and [111] for example, would be represented by

$g_1(D) = 1 \quad g_2(D) = 1 + D \quad g_3(D) = 1 + D + D^2$

Thus, the transforms corresponding to the convolution would be as follows:

$c^{(1)}(D) = u(D)g_1(D)$

$c^{(2)}(D) = u(D)g_2(D)$

$c^{(3)}(D) = u(D)g_3(D)$

And the interleaved output would be:

$$c(D) = c^{(1)}(D^3) + Dc^{(2)}(D^3) + D^2c^{(3)}(D^3)$$

Representation of Convolutional codes:

Convolutional codes are usually represented by three alternative methods; tree diagrams, trellises and state diagrams. The one most relevant to the topic of MLSD and viterbi algorithms is the trellis representation. Utilizing the trellis diagram to represent convolutional codes depends on an important observation; that the constraint length defines the point at which the system would repeat itself. For example, a convolutional encoder with a constraint length of 3 will have its output structure repeating after the third stage. This can be explained as follows; for a convolutional encoder with K =3, the output bit depends only on the input bit and the last two bits (since the bit within the third position would be shifted out of the register, thus having no effect). Therefore, the output mainly depends on the input bit and one out of 4 states for the other two bits (00,01,10,11). Thus the trellis is represented with four nodes (representing the 4 states), and after the second stage, each node has two input paths and 2 output paths.

In general, with a rate k/n constraint length K convolutional encoder, the trellis has $2^{k(K-1)}$ states, and there are 2^k branches entering each node and an equal number of branches exiting.

Decoding Convolutional Codes

Convolutional codes can be decoded using various methods, such as soft vs. hard decision decoding or MAP(maximum a posteriori) vs. ML (maximum likelihood) methods. The optimum decoder in the case of convolutional codes is the MLSE (Maximum Likelihood sequence detector) due to its finite-state machine structure. Thus, the optimum decoding in this case takes place by searching through the trellis corresponding to the convolutional code, and finding the path with the smallest distance metric (most probable sequence), which could be a Hamming distance metric (for hard decision decoding), or Euclidean distance metric (for soft decision decoding).

Decoding takes place as follows; the first node representing an output (after the trellis has stabilized, with an equal number of branches entering and exiting each node) is observed, and the path followed through the trellis from the initial input node to that output node is to be determined. Hence, transmitted bits are denoted by c_{jm} , where j indicates the number of the branch, and m indicates the m-th bit within that branch, whereas the received bits are represented by r_{jm}. If hard decision decoding is applied, then the detector output for each transmitted bit is either is a 0 or a 1, whereas if soft decision decoding is applied, then $r_{jm} = \sqrt{E}\left(2c_{jm} - 1\right) + n_{jm}$ where E is the transmitted signal energy for each code bit and n is the additive noise.

A metric is then defined for each branch(j) within a certain path (i) in the trellis, as the logarithm of the joint probability of the sequence of output bits r_{jm} conditioned on the transmitted sequence c_{jm} :

$$\mu_j^{(i)} = \log p\left(r_j I c_j^{(i)}\right) \qquad\qquad j = 1,2,3,....$$

Thus, the metric for the entire path(i) consisting of B branches becomes:

$$PM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)}$$

Deciding between the various path metrics takes place by choosing the one possessing the larger metric, which corresponds to maximizing the probability of a correct decision (or, minimizing the probability of error for information bits).

For Soft decision decoding, the channel adds white noise to the signal, and thus the demodulated output is represented statistically by the probability density function:

$$p\left(r_{jm} I\ c_{jm}^{(i)}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{[r_{jmc} - \sqrt{\varepsilon}\left(2c_{jm}^{(i)} - 1\right)]^2}{2\sigma^2}$$

Where $\sigma^2 = \frac{1}{2}N_o$ is the variance of the additive Gaussian noise. If the terms common to all branch metrics are ignored, the branch metric for the *jth* branch of the *ith* path is expressed as:

$$\mu_j^{(i)} = \sum_{m=1}^{n} r_{jm}(2c_{jm}^{(i)} - 1)$$

Hence the correlation metric for a path (B) would be:

$$CM^{(B)} = \sum_j \sum_m r_{jm}(2c_{jm}^{(B)} - 1)$$

Where the correlation metric is defined as the negative of the modified distance metric.

General formula for ML decoding of convolutional codes:

For ML decoding, we seek a code sequence within the trellis (τ) that satisfies the following conditions:

$c^{(m)} = max_{c\epsilon\tau} \sum_j \log p(r_j I c_j)$ For a general memoryless channel

$c^{(m)} = min_{c\,\epsilon\tau} \sum_j \left\|r_j - c_j\right\|^2$ For soft decision decoding

$c^{(m)} = min_{c\varepsilon\tau} \sum_j d_H(y_j, c_j)$ For hard decision decoding

Hence, maximum likelihood decoding entails the determination of a path within the trellis that minimizes or maximizes a metric. This is determined through the viterbi algorithm, through which it is observed that after determining which path (entering a particular node) has the larger metric, any nodes that are then added to this path will still keep it larger than the discarded one earlier, so for future nodes, the path which was discarded for the first node will again be discarded, and thus, only the first path (called the survivor) would be considered for the following node. This reduces the number of paths that need to be considered for each stage by a factor of 2, making the decision-making process simpler.

Conclusion

To conclude, despite the apparent complexity of signaling schemes with memory, detection of such symbols takes place using the same methods as memoryless schemes, such as the MAP rule and the ML rule which has been discussed in full. It is important also to realize the importance of the Viterbi algorithm in reducing the time and effort needed during MLSD, eventhough it tends to be taxing in terms of machine memory.

References:

Boyle, Roger. "Viterbi algorithm." School of Computing, University of Leeds. University of Leeds, Web. <http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/viterbi_algorithm/s1_pg2.html>.

Cheetham, Barry . "Hard & soft decisions decoding." University of Manchester. University of Manchester, Web. <http://www.cs.manchester.ac.uk/~barry/mydocs/CS3282/Notes/>.

Proakis, John. Digital Communication. 5th ed. NY: McGraw-Hill, 2008. 242-46. Print.

Veeravalli, V. "Digital Modulation." University of Illinois. University of Illinois, Web. <http://courses.ece.illinois.edu/ece461/handouts/notes5.pdf>.

"Viterbi algorithm." National Institute of standards and technology. 6 Jul 2004. National Institute of standards and technology, Web. <http://www.itl.nist.gov/div897/sqg/dads/HTML/viterbiAlgorithm.html>.

"What is a Parser?." NorKen Technologies. 2008. NorKen Technologies, Web. <http://www.programmar.com/parser.htm>.