ECE 361: Lecture 19: Maximum-Likelihood Sequence Estimation

In Lecture 18, we considered minimum mean-square-error (MMSE) equalizers that work strictly better than matched filters and zero-forcing and decision-feedback equalizers at all SNR. All these equalizers concentrate on reducing the probability of error for each bit in the receiver output. However, as noted several times earlier in these Lectures, what we really want to do is minimize the probability that the receiver chooses the wrong *sequence* of bits as its output, that is, we want to minimize the sequence or word error probability, not the bit error probability The *maximum-likelihood* receiver achieves the minimum word error probability, but generally is far too complicated to implement in almost all cases of interest. If an n-bit sequence is transmitted, then there are 2^n likelihoods to be computed before the largest likelihood can be determined, and the computational task is overwhelming except in the uninteresting case of small n. Interestingly enough, for transmission over band-limited channels, there exists a clever re-formulation of the computational task that allows the receiver to determine the largest likelihood without computing all 2^n likelihoods. This is achieved by computing likelihoods iteratively as each symbol is received, and discarding all those (partial) likelihoods that are so small that we can be sure that they never going to grow large enough to be the maximum likelihood when the processing is complete. Such *pruning* reduces the computational burden, and makes such *maximum-likelihood sequence estimators* feasible. Indeed, the limiting factor is not n, the number of bits transmitted, but the length of the channel response $h[\cdot]$. The complexity of the maximum-likelihood sequence estimator grows exponentially with the channel response length, and at best linearly with n. How all this works is the subject of this Lecture.

19.1. Likelihoods and Log Likelihoods

Recall that in our model, the data being transmitted is a sequence $\{\mathbb{X}[m]\}$ of independent random variables taking on equally likely values $\pm\sqrt{\mathcal{E}_T}$, and the channel output can be expressed as a sequence $\{\mathbb{Y}[m]\}$ where

$$\mathbb{Y}[m] = \sum_{j=-L}^{M} h[j]\mathbb{X}[m-j] + \mathbb{N}[m] = h[0]\mathbb{X}[m] + \sum_{j=-L}^{M}{}' h[j]\mathbb{X}[m-j] + \mathbb{N}[m] \tag{19.1}$$

where $\mathbb{N}[m]$ is a $\mathcal{N}(0,\sigma^2)$ noise variable independent of all other $\mathbb{N}[m']$ and $\sum'$ in (19.1) means that the $j = 0$ term is excluded from the sum. For convenience in exposition, let us assume that $L = 0$ and $M = 2$, and $h[0] = 1$, so that $|h[1]|, |h[2]| < 1$. Assume also that the transmission consists of n bits $\mathbb{X}[i]$, $0 \le i \le n$ and is preceded (and followed) by two time slots during which nothing is transmitted, that is, $\mathbb{X}[-2] = \mathbb{X}[-1] = \mathbb{X}[n] = \mathbb{X}[n+1] = 0$. This is the model of Lecture 17.1 and thus the channel outputs are

$$\begin{aligned}
\mathbb{Y}[0] &= \mathbb{X}[0] && && && + \mathbb{N}[0]\\
\mathbb{Y}[1] &= \mathbb{X}[1] && + h[1]\mathbb{X}[0] && && + \mathbb{N}[1]\\
\mathbb{Y}[2] &= \mathbb{X}[2] && + h[1]\mathbb{X}[1] && + h[2]\mathbb{X}[1] && + \mathbb{N}[2]\\
&\vdots && \vdots && \vdots && \vdots\\
\mathbb{Y}[m] &= \mathbb{X}[m] && + h[1]\mathbb{X}[m-1] && + h[2]\mathbb{X}[m-2] && + \mathbb{N}[m]\\
&\vdots && \vdots && \vdots && \vdots\\
\mathbb{Y}[n-1] &= \mathbb{X}[n-1] && + h[1]\mathbb{X}[n-2] && + h[2]\mathbb{X}[n-3] && + \mathbb{N}[n-1]\\
\mathbb{Y}[n] &= && h[1]\mathbb{X}[n-1] && + h[2]\mathbb{X}[n-1] && + \mathbb{N}[n]\\
\mathbb{Y}[n+1] &= && && h[2]\mathbb{X}[n-1] && + \mathbb{N}[n+1]
\end{aligned}$$

Let $\sqrt{\mathcal{E}_T}\mathbf{y} = \sqrt{\mathcal{E}_T}(y[0], y[1], \ldots, y[n+1])$ denote the received values, that is, the observation. The likelihoods of this observation are the 2^n conditional pdfs of $\underline{\mathbb{Y}} = (\mathbb{Y}[0], \mathbb{Y}[1], \ldots, \mathbb{Y}[n+1])$ conditioned on the 2^n different

transmitted vectors $\underline{\mathbb{X}} = (\mathbb{X}[0], \mathbb{X}[1], \ldots, \mathbb{X}[n-1])$. Conditioned on $\underline{\mathbb{X}} = \sqrt{\mathcal{E}_T}\mathbf{x}$ where $\mathbf{x}$ is a n-vector of ± 1 entries, the likelihood is

$$\begin{aligned}\Lambda_{\mathbf{x}} &= f_{\underline{\mathbb{Y}}}(\sqrt{\mathcal{E}_T}\mathbf{y} \mid \underline{\mathbb{X}} = \sqrt{\mathcal{E}_T}\mathbf{x}) = \prod_{m=0}^{n+1} f_{\mathbb{Y}[m]}(\sqrt{\mathcal{E}_T}y[m] \mid \underline{\mathbb{X}} = \sqrt{\mathcal{E}_T}\mathbf{x}) \\ &= \prod_{m=0}^{n+1} f_{\mathbb{Y}[m]}\big(\sqrt{\mathcal{E}_T}y[m] \mid (\mathbb{X}[m], \mathbb{X}[m-1], \mathbb{X}[m-2]) = \sqrt{\mathcal{E}_T}(x[m], x[m-1], x[m-2])\big) \qquad (19.2)\end{aligned}$$

since $\mathbb{Y}[m]$ depends only on three transmitted bits, not the whole vector. The pdfs in (19.2) are all Gaussian pdfs with common variance σ^2 and $\mathbb{Y}[m]$ having conditional mean $\sqrt{\mathcal{E}_T}(x[m] + h[1]x[m-1] + h[2]x[m-2])$. Thus, as in Lecture 10, we can compute *negative log likelihoods* which are proportional to squared distances and work with those instead. We have

$$-\ln \Lambda_{\mathbf{x}} = \sum_{m=0}^{n+1} \ln(\sigma\sqrt{2\pi}) + \frac{\mathcal{E}_T}{2\sigma^2}\big(y[m] - (x[m] + h[1]x[m-1] + h[2]x[m-2])\big)^2 \qquad (19.3)$$

Since we are only going to compare likelihoods (or log likelihoods), we can ignore the constant terms as well as the factor $\mathcal{E}_T/2\sigma^2 = \frac{1}{2}\mathsf{SNR}$ and compute

$$D^2_{\mathbf{x}} = \sum_{m=0}^{n+1} \big(h[2]x[m-2] + h[1]x[m-1] + x[m] - y[m]\big)^2 \qquad (19.4)$$

where $y[m]$ is the normalized observation (a real number) while $x[m]$, $x[m-1]$ and $x[m-2]$ have value ± 1. The maximum likelihood estimate of the transmitted sequence of bits is that $\mathbf{x}$ for which (19.4) has the *smallest* value. All this converts the calculations from products as in (19.2) into sums as in (19.4) which reduces the computational burden somewhat. But the significant savings are obtained by doing an iterative calculation of the quantities $D^2_{\mathbf{x}}$ using a concept called the *trellis diagram*.

19.2. Iterative Calculations and Trellis Diagrams

We have to calculate $D^2_{\mathbf{x}}$ as given in (19.4) for 2^n different choices of $\mathbf{x}$. We can compute these numbers iteratively via partial sums of (19.4). Since all the $\mathbf{x}$'s either begin with $x[0] = +1$ or $x[0] = -1$, and $x[-1] = x[-2] = 0$, we first compute

$$\begin{aligned}D^2_+ &= \big(h[2]x[-2] + h[1]x[-1] + x[0] - y[0]\big)^2 = \big(+1 - y[0]\big)^2 \\ D^2_- &= \big(h[2]x[-2] + h[1]x[-1] + x[0] - y[0]\big)^2 = \big(-1 - y[0]\big)^2\end{aligned}$$

Next, we compute *two* terms from D^2_+ by adding $\big(h[2]x[-1] + h[1]x[0] + x[1] - y[1]\big)^2 = \big(h[1] + x[1] - y[1]\big)^2$:

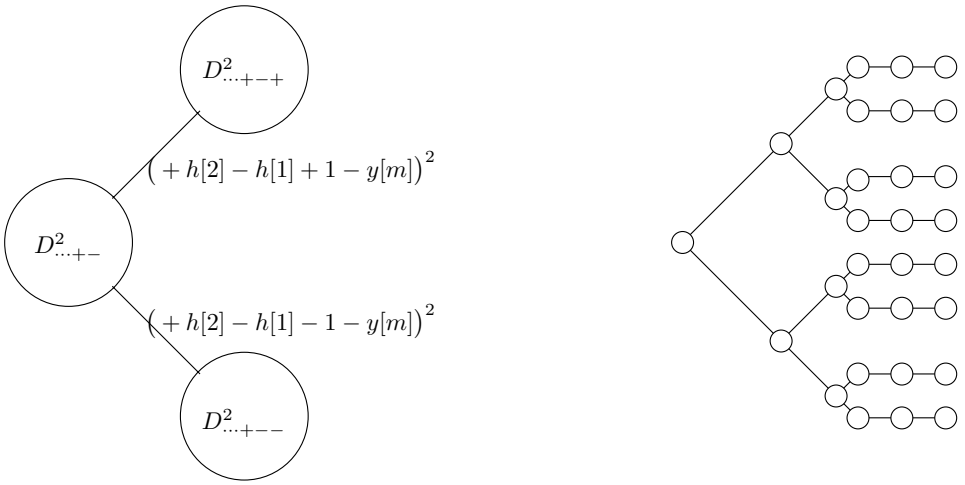
$$\begin{aligned}D^2_{++} &= D^2_+ + \big(h[1] + x[1] - y[1]\big)^2 &&= D^2_+ + \big(h[1] + 1 - y[1]\big)^2 \\ D^2_{+-} &= D^2_+ + \big(h[1] + x[1] - y[1]\big)^2 &&= D^2_+ + \big(h[1] - 1 - y[1])\big)^2\end{aligned}$$

and *two* terms from D^2_- by adding $\big(h[2]x[-1] + h[1]x[0] + x[1] - y[1]\big)^2 = \big(-h[1] + x[1] - y[1]\big)^2$:

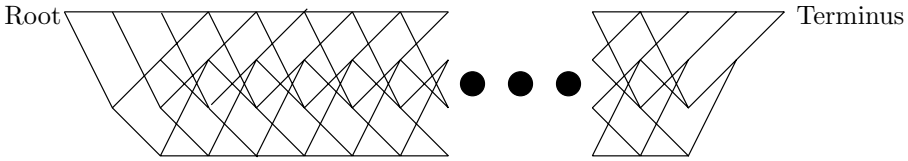
$$\begin{aligned}D^2_{-+} &= D^2_- + \big(-h[1] + x[1] - y[1]\big)^2 &&= D^2_- + \big(-h[1] + 1 - y[1]\big)^2 \\ D^2_{--} &= D^2_- + \big(-h[1] + x[1] - y[1]\big)^2 &&= D^2_- + \big(-h[1] - 1 - y[1]\big)^2\end{aligned}$$

Next, we compute two terms from each of D^2_{++}, D^2_{+-}, D^2_{-+}, and D^2_{--} by adding the quantity $\big(h[2]x[0] + h[1]x[1] + x[2] - y[2]\big)^2$ to each to get eight partial sums $D^2_{+++}, D^2_{++-}, \ldots, D^2_{---}$, and so on. More generally,

from each partial sum $D_{\mathbf{x}'}$ (where $\mathbf{x}'$ is a sequence of m $\pm$ symbols,) we calculate two new partial sums $D_{\mathbf{x}'+}$ and $D_{\mathbf{x}'-}$ by adding $\big(\pm h[2] \pm h[1] + 1 - y[m]\big)^2$ and $\big(\pm h[2] \pm h[1] - 1 - y[m]\big)^2$ respectively where the sign of $h[2]$ is the same as the *next-to-rightmost* symbol in $\mathbf{x}'$ and the sign of $h[1]$ is the same as the *rightmost* symbol in $\mathbf{x}'$. The left-hand figure shown below illustrates how $D^2_{\cdots+-}$ is used to compute $D^2_{\cdots+-+}$ and $D^2_{\cdots+--}$ by adding $\big(+h[2]-h[1]+1-y[m]\big)^2$ and $\big(+h[2]-h[1]-1-y[m]\big)^2$ respectively. The right-hand figure illustrates that we can think of the computation each $D^2_{\mathbf{x}}$ as the traversal of a path from the root to a leaf of a binary tree. Each branch of the tree is labelled with the squared distance to be added as shown in the left-hand figure but not the right-hand figure (due to lack of space). As we traverse the tree from root to leaf, choosing the upper branch for each + and the lower branch for each −, we add up the labels of the branches being traversed. Note that there is no bifurcation at the last two vertices before the leaf corresponding to the $m = n$ and $m = n + 1$ terms in (19.4).

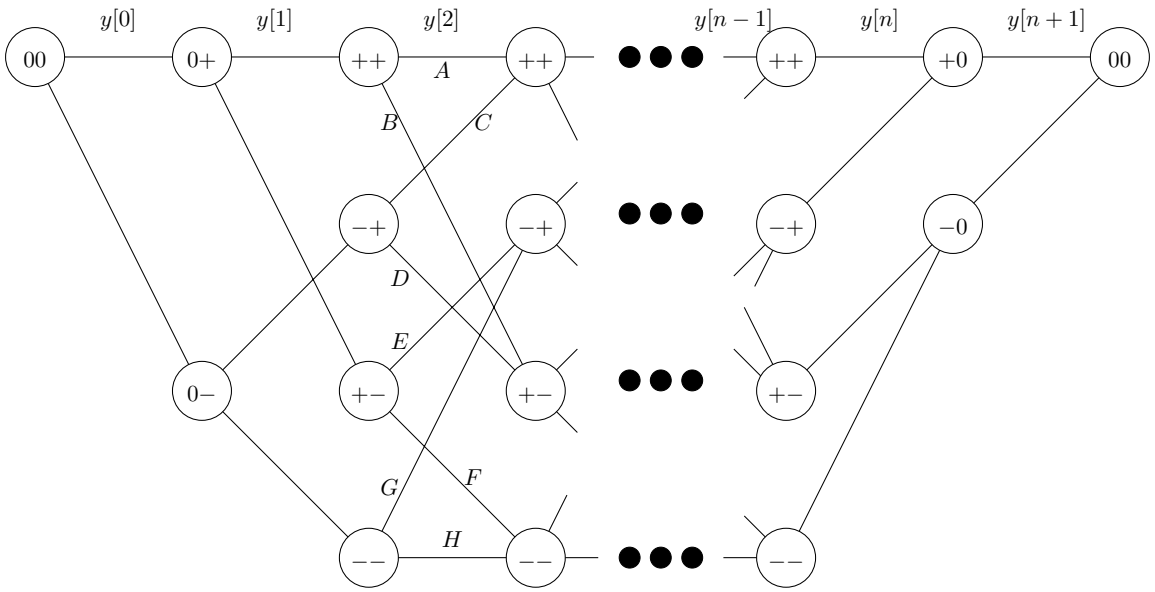


For reasons of space, the figure on the right is for the trivial case $n = 3$ bits being transmitted, but in practice the tree being traversed is huge with there being $2^m \gg 1$ nodes at depth m with two branches emanating from each node. On the other hand, these 2^{m+1} branches have at most 8 different labels $\big(\pm h[2] \pm h[1] \pm 1 - y[m]\big)^2$ among them! Thus, it is possible to describe the computations of the $D^2_{\mathbf{x}}$'s as traversing the edges for a different graph called a *trellis*[1] which looks as shown below.



There is one root node and one terminus node in the trellis. Two branches emanate from the root and from each succeeding node, just as in the tree above. But as we progress into the trellis, two branches meet at a node too. In fact, apart from a initial section of two stages and a terminal section of two stages,[2] the trellis has a very regular structure. There are 4 nodes at each stage[3] and there are 2^n different paths through the trellis (always moving from left to right) from root node to terminus, and if we traverse the different paths adding up the branch labels as we go along, we get the different $D^2_{\mathbf{x}}$'s. More details about the trellis for our example problem are illustrated on the next page in which each node is labelled with the *previous* two data symbols $x[m-2]$ and $x[m-1]$ since these determine the labels of the branches emanating from the node. For lack of space, the branch labels are not shown on the figure but are listed separately.

[1] because of its resemblance to a garden structure used to support vines.
[2] More generally M stages if the channel impulse response has duration $M + 1$ symbols
[3] More generally, there are 2^M nodes at each stage if the channel impulse response has duration $M + 1$ symbols



At the m-th stage in the central portion of the trellis,

- the node labelled $++$ has two branches emanating from it with labels

$$A = \big(h[2] + h[1] + 1 - y[m]\big)^2, \quad \text{and} \quad B = \big(h[2] + h[1] - 1 - y[m]\big)^2.$$

- the node labelled $-+$ has two branches emanating from it with labels

$$C = \big(-h[2] + h[1] + 1 - y[m]\big)^2, \quad \text{and} \quad D = \big(-h[2] + h[1] - 1 - y[m]\big)^2.$$

- the node labelled $+-$ has two branches emanating from it with labels

$$E = \big(h[2] - h[1] + 1 - y[m]\big)^2, \quad \text{and} \quad F = \big(h[2] - h[1] - 1 - y[m]\big)^2.$$

- the node labelled $--$ has two branches emanating from it with labels

$$G = \big(-h[2] - h[1] + 1 - y[m]\big)^2, \quad \text{and} \quad H = \big(-h[2] - h[1] - 1 - y[m]\big)^2.$$

In fact, these formulas apply to the branches in the initial and terminal stages also as long as we use the node labels 0 (not just ± 1) as multipliers for $h[2]$ and $h[1]$ also.

Regardless of whether we formulate the task of computing the 2^n likelihoods (or squared distances) in terms of a binary tree or a trellis, there is a prohibitively large amount of computation that is required. Furthermore, after all the computational effort, the results are essentially discarded since all we are interested in is the identity of the sequence with the argest likelihood. The value of the largest likelihood itself is of no importance: we only need the assurance that it is indeed the largest and the sequence that has this largest likelihood. The *Viterbi algorithm*, which we study next, is a special case of a more general technique called *dynamic programming*, and reduces the computational task to very reasonable proportions. The key idea is that *all* the 2^n paths from the root to the terminus pass through one of the 2^m nodes at stage m, and we can eliminate many of these paths from further consideration.

19.3. Maximum-likelihood Sequence Estimation

Suppose that we have computed the partial sums D^2_{++}, D^2_{+-}, D^2_{-+} which we can do by proceeding along paths from the root node 00 to the first set of nodes marked $++$, $-+$, $+-$, and $--$ in the trellis diagram

above, adding up the labels of the branches we encounter. All these calculations can be assumed to have been done in parallel. Now we extend these partial sums, and in particular compute $D^2_{+++} = D^2_{++} + A$ and $D^2_{-++} = D^2_{-+} + C$. These correspond to paths

$$00 \longrightarrow 0+ \longrightarrow ++ \longrightarrow ++ \quad \text{and} \quad 00 \longrightarrow 0- \longrightarrow -+ \longrightarrow ++$$

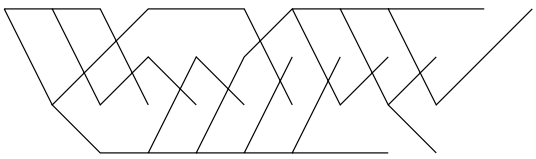
and suppose that $D^2_{+++} > D^2_{-++}$. If we extend these partial paths all the way to the bitter end to compute $D^2_{+++\mathbf{z}}$ and $D^2_{-++\mathbf{z}}$ where $\mathbf{z}$ is a sequence of $n-3$ $\pm$ symbols, adding the squared distances that we encounter, then obviously $D^2_{+++\mathbf{z}} > D^2_{-++\mathbf{z}}$. Thus, *no* sequence beginning $+++$ and having a tail $\mathbf{z}$ can be the maximum-likelihood sequence because the sequence beginning $-++$ and having the same tail $\mathbf{z}$ will always have the larger likelihood. Therefore, we can just ignore all paths with prefix $+++$ and not carry out the remainder of the calculations for the likelihoods of these paths. Similarly, we can look at the partially computed likelihoods (or squared distances) for the paths

$$00 \longrightarrow 0+ \longrightarrow +- \longrightarrow -+ \quad \text{and} \quad 00 \longrightarrow 0- \longrightarrow -- \longrightarrow -+$$

corresponding to prefixes $+-+$ and $--+$ and eliminate all paths with one of these prefixes from future consideration. The same considerations apply to partial paths with prefixes $++-$ and $-+-$ ending in a node on the third row in the trellis diagram above, and $+--$ and $---$ ending in a node on the fourth row in the trellis diagram above.

Put another way, whenever two partial paths merge at a node in the trellis diagram, it is only necessary to keep the better path and extend it further; the poorer path can be discarded from further consideration.[4] From another viewpoint, the Viterbi algorithm progresses through the trellis, finding the best paths from the root to all the four (more generally 2^M) nodes at each stage. When it reaches the terminal node, it has found the maximum-likelihood path through the trellis. The circuits that implement the Viterbi algorithm are called add-compare-select (ACS) circuits because that is, in essence, the algorithm: we extend two paths by adding in the labels, we do a comparison of the results of the additions, and we then select one of the paths to keep, and discard the other. It should be obvious that the complexity of the implementation increases exponentially with the length of the channel impulse response but only linearly with the number of bits processed. Thus, maximum-likelihood sequence estimation, which is the very best that we can do, can be used only in cases where the channel impulse response is of relatively short duration. When the channel impulse response is very long, suboptimum methods such as those studied in previous Lectures provide more feasible alternatives.

In closing, let us look at an example – without numbers – showing how the Viterbi algorithm progresses through the trellis.



Notice how, long before the algorithm has reached the terminus, many paths share the same initial segment. Once such a consensus has been reached, it is possible to output the initial segment from the receiver. Of course, in many cases, especially those where the SNR is low, two different initial segments may remain in contention till the very end with the final choice.

[4] If both paths have the same likelihood, it does not matter which one is kept and which one discarded.