Decoding

There are several different approaches to decoding of convolutional codes. These are grouped in two basic categories.

1.Sequential Decoding
    - Fano algorithm
2. Maximum likely-hood decoding
    - Viterbi decoding

Both of these methods represent 2 different approaches to the same basic idea behind decoding.

The basic idea behind decoding

Assume that 3 bits were sent via a rate ½ code. We receive 6 bits. (Ignore flush bits for now.) These six bits may or may not have errors. We know from the encoding process that these bits map uniquely. So a 3 bit sequence will have a unique 6 bit output. But due to errors, we can receive any and all possible combinations of the 6 bits.

The permutation of 3 input bits results in eight possible input sequences. Each of these has a unique mapping to a six bit output sequence by the code. These form the set of permissible sequences and the decoder's task is to determine which one was sent.

Table 4 – Bit Agreement used as metric to decide between the received sequence and the 8 possible valid code sequences

Input	Valid Code Sequence	Received Sequence	Bit Agreement
000	000000	111100	2
001	000011	111100	0
010	001111	111100	2
011	001100	111100	4
100	**111110**	111100	**5**
101	**111101**	111100	**5**
110	110001	111100	3
111	110010	111100	3

Let's say we received 111100. It is not one of the 8 possible sequences above. How do we decode it? We can do two things

1. We can compare this received sequence to all permissible sequences and pick the one with the smallest Hamming distance(or bit disagreement)

2. We can do a correlation and pick the sequences with the best correlation.

The first procedure is basically what is behind hard decision decoding and the second the soft-decision decoding. The bit agreements, also the dot product between the received sequence and the codeword, show that we still get an ambiguous answer and do not know what was sent.

As the number of bits increase, the number of calculations required to do decoding in this brute force manner increases such that it is no longer practical to do decoding this way. We need to find a more efficient method that does not examine all options and has a way of resolving ambiguity such as here where we have two possible answers. (Shown in bold in Table 4)

If a message of length s bits is received, then the possible number of codewords are 2^s. How can we decode the sequence without checking each and everyone of these 2^s codewords? This is the basic idea behind decoding.

Maximum Likelihood and Viterbi decoding

Viterbi decoding is the best known implementation of the maximum likely-hood decoding. Here we narrow the options systematically at each time tick. The principal used to reduce the choices is this.

1. The errors occur infrequently. The probability of error is small.
2. The probability of two errors in a row is much smaller than a single error, that is the errors are distributed randomly.

The Viterbi decoder examines an entire received sequence of a given length. The decoder computes a metric for each path and makes a decision based on this metric. All paths are followed until two paths converge on one node. Then the path with the higher metric is kept and the one with lower metric is discarded. The paths selected are called the survivors.

For an N bit sequence, total numbers of possible received sequences are 2^N. Of these only 2^{kL} are valid. The Viterbi algorithm applies the maximum-likelihood principles to limit the comparison to 2 to the power of kL surviving paths instead of checking all paths.

The most common metric used is the Hamming distance metric. This is just the dot product between the received codeword and the allowable codeword. Other metrics are also used and we will talk about these later.

Table 5 – Each branch has a Hamming metric depending on what was received and the valid codewords at that state

Bits Received	Valid Codeword 1	Valid Codeword 2	Hamming Metric 1	Hamming Metric 2
00	00	11	2	0
01	10	01	0	2
10	00	11	1	1

These metrics are cumulative so that the path with the largest total metric is the final winner.

But all of this does not make much sense until you see the algorithm in operation.

Let's decode the received sequence 01 11 01 11 01 01 11 using Viterbi decoding.

1. At t = 0, we have received bit 01. The decoder always starts at state 000. From this point it has two paths available, but neither matches the incoming bits. The decoder computes the branch metric for both of these and will continue simultaneously along both of these branches in contrast to the sequential decoding where a choice is made at every decision point. The metric for both branches is equal to 1, which means that one of the two bits was matched with the incoming bits.

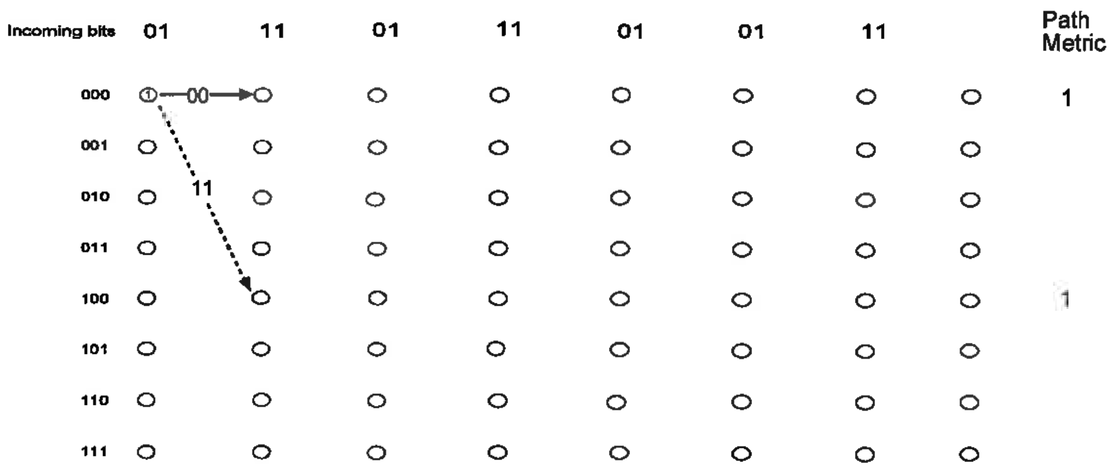

Figure 12a – Viterbi Decoding, Step 1

2. At t = 1, the decoder fans out from these two possible states to four states. The branch metrics for these branches are computed by looking at the agreement with the codeword and the incoming bits which are 11. The new metric is shown on the right of the trellis.

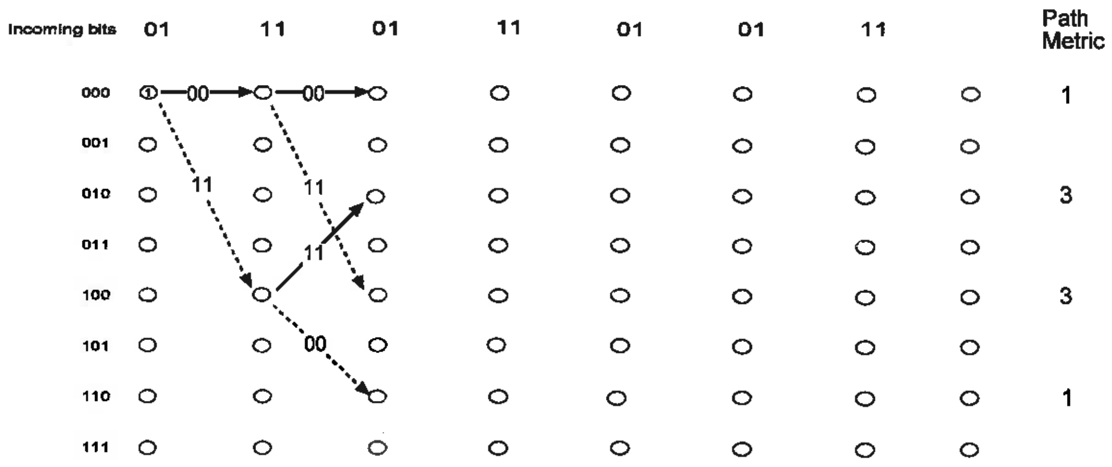

Figure 12b – Viterbi Decoding, Step 2

3. At t = 2, the four states have fanned out to eight to show all possible paths. The path metrics calculated for bits 01 and added to pervious metrics from t = 1.

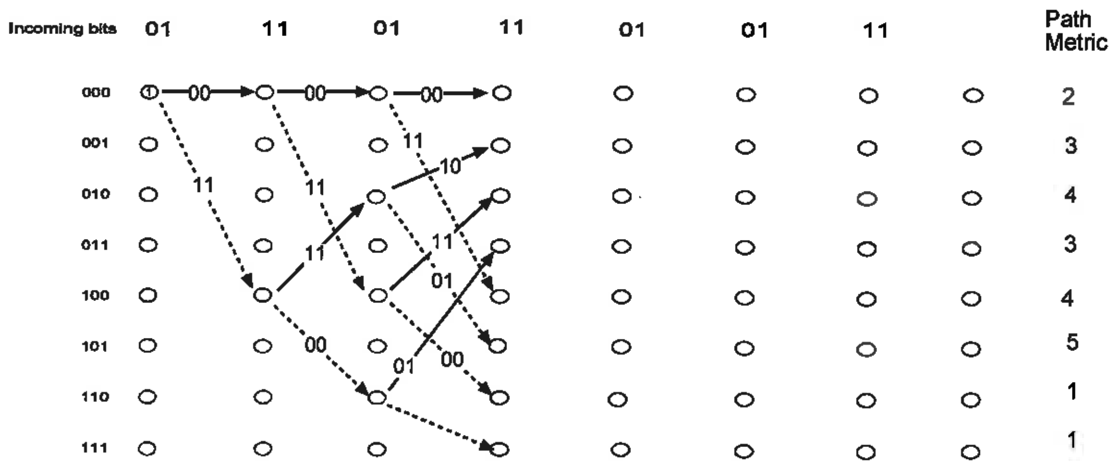

Figure 12c – Viterbi Decoding, Step 3

At t = 4, the trellis is fully populated. Each node has at least one path coming into it. The metrics are as shown in the figure above.

At t = 5, the paths progress forward and now begin to converge on the nodes. Two metrics are given for each of the paths coming into a node. Per the Maximum likelihood principle, at each node we discard the path with the lower metric because it is least likely. This discarding of paths at each node helps to reduce the number of paths that have to be examined and gives the Viterbi method its strength.

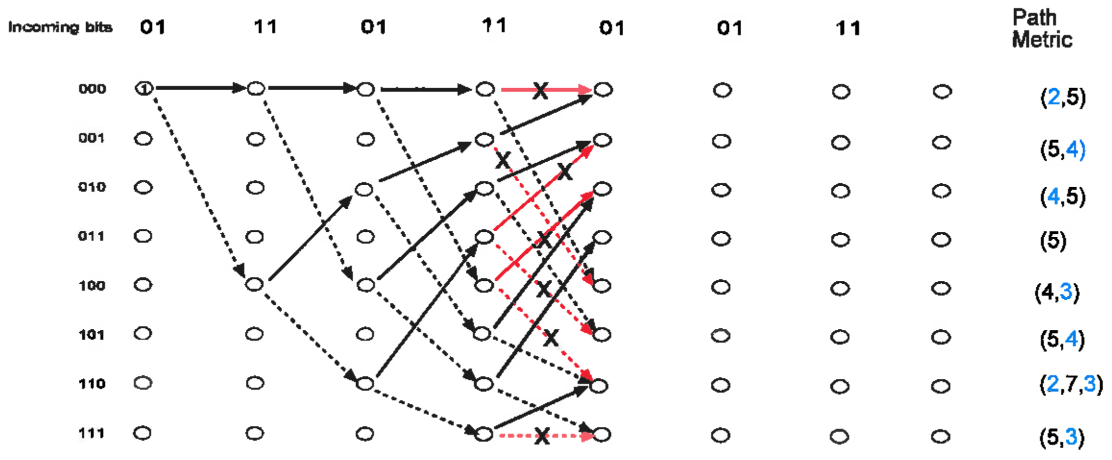

Figure 12d – Viterbi Decoding, Step 4

Now at each node, we have one or more path converging. The metrics for all paths are given on the right. At each node, we keep only the path with the highest metric and discard all others, shown in red. After discarding the paths with the smaller metric, we have the following paths left.
The metric shown is that of the winner path.

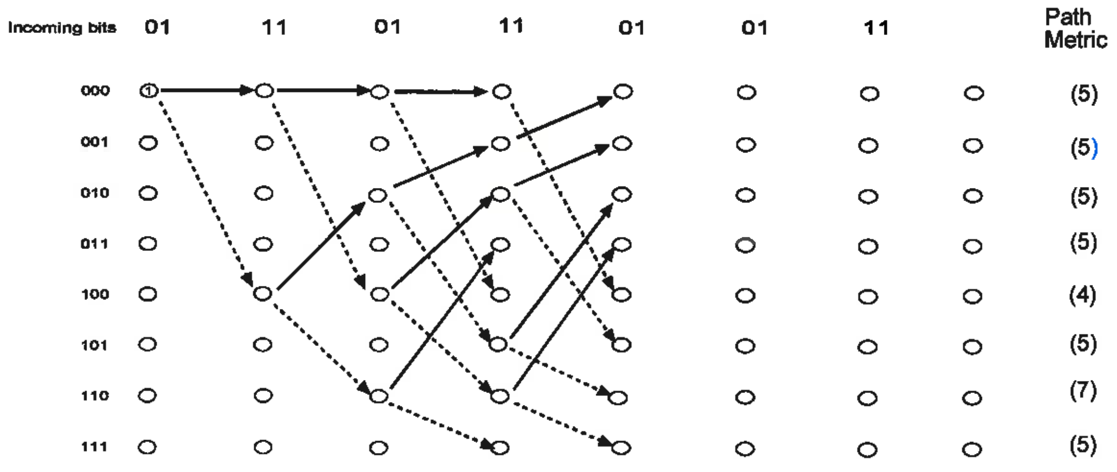

Figure 12e – Viterbi Decoding, Step 4, after discarding

At t = 5, after discarding the paths as shown, we again go forward and compute new metrics. At the next node, again the paths converge and again we discard those with lower metrics.

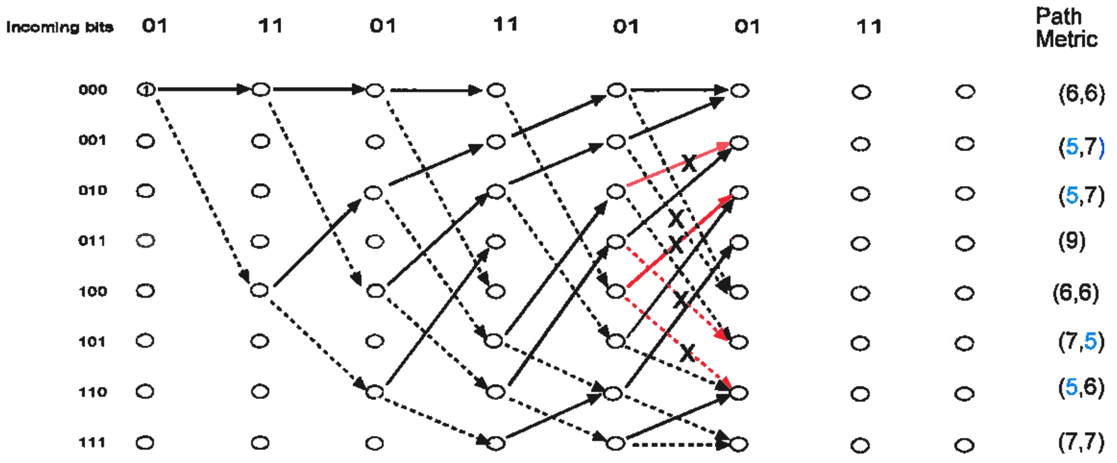

Figure 12f – Viterbi Decoding, Step 5

At t= 6, the received bits are 11. Again the metrics are computed for all paths. We discard all smaller metrics but keep both if they are equal.

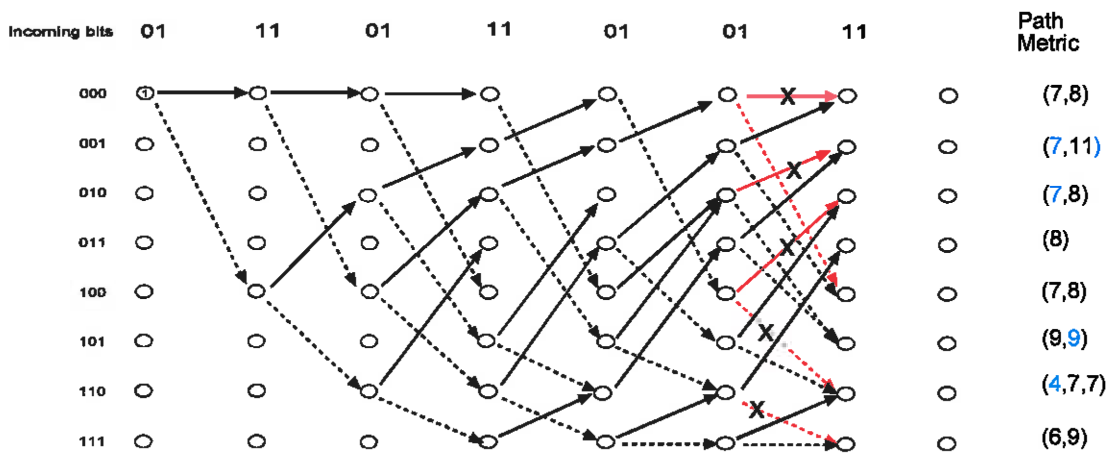

Figure 12g – Viterbi Decoding, Step 6

The 7-step trellis is complete. We now look at the path with the highest metric. We have a winner. The path traced by states 000, 100, 010, 101, 110, 011, 001, 000 and corresponding to bits 1011000 is the decoded sequence.

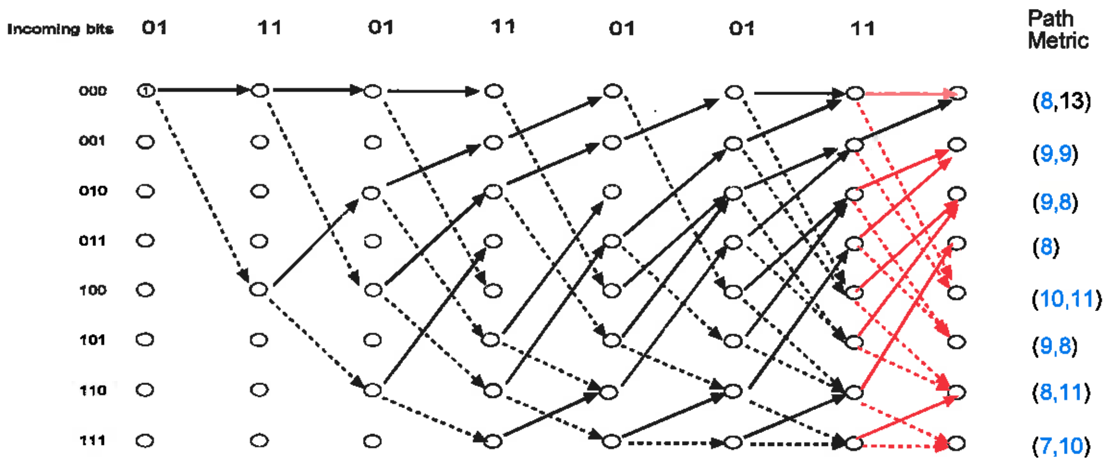

Figure 12h – Viterbi Decoding, Step 7

The length of this trellis was 4bits + m bits. Ideally this should be equal to the length of the message, but by a truncation process, storage requirements can be reduced and decoding need not be delayed until the end of the transmitted sequence. Typical Truncation length for convolutional coders is 128 bits or 5 to 7 times the constraint length.

Viterbi decoding is quite important since it also applies to decoding of block codes. This form of trellis decoding is also used for Trellis-Coded Modulation (TCM).