Convolutional Codes

Viterbi Decoder

Isha Chaudhary

September 2020

Outline

Soft ML Decoding of Convolutional Codes

- We attempt to find the minimum *Euclidean Distance* codeword for a received vector, $\underline{r}$.
- Considering a BPSK modulated codeword, over the AWGN channel, we will denote
  1. $\underline{u} = [u_0, u_1, ..., u_{k-1}]$ as the input message
  2. $\underline{c} = [c_0, c_1, ..., c_{k+\mu-1}]$, where $c_i = [v_i^{(1)} v_i^{(2)}]$, a half design rate, zero-terminated codeword.
  3. This is then BPSK modulated to give $\underline{s} = [s_0, s_1, ..., s_{k+\mu-1}]$, where s_i are BPSK modulated 2 bit vectors.
  4. $\underline{z}$ is added as AWGN noise to $\underline{s}$ to give the received vector $\underline{r} = r_0 + r_1 + ... + r_{k+\mu-1}$
- The *Viterbi Algorithm* is used to efficiently Soft ML decode convolutional codewords.

Trellis outlook

- We need to find

$$\hat{u} = \underset{\underline{u} \in \{0,1\}^k}{\operatorname{argmin}} \|\underline{r} - \underline{s}\|^2$$

- $$\|\underline{r} - \underline{s}\|^2 = \sum_{i=0}^{k+\mu-1} (r_i^{(1)} - s_i^{(1)}) + (r_i^{(2)} - s_i^{(2)})$$

- To perform this minimization, we would otherwise need 2^k computations, for each of the codewords.
- Using the Trellis, the number of computations become exponential in μ and linear in k. We perform the computation and minimization, step-by-step along the trellis.
- As the number of states μ is generally lesser than the number of message bits, k, the trellis outlook reduces the complexity of the decoder significantly.

Terminology

- **Path Metric**: It is a measure for each path along the trellis, which denotes a codeword (BPSK: $\underline{s}$).

$$Path\ metric\ =\ \|\underline{r}-\underline{s}\|^2$$

- **Branch Metric**: *For branch at the l^{th} stage (for l^{th} received bit) of trellis, from state i to state j*

$$Branch\ metric(BM)\ =\ (r_l^{(1)}-s_{ij}^{(1)})^2+(r_l^{(2)}-s_{ij}^{(2)})^2$$

  if we are assuming only two states for the encoder and where s_{ij} are the BPSK modulated codeword bits, corresponding to the transition from state i to state j.

- $$\sum_{Branch \in Path} (Branch\ Metric) = Path\ Metric$$

- **State Metric ($SM^{(l)}(i)$)** is the minimum metric path from state 0 at stage 0 to state i in stage l.

Viterbi Algorithm

Viterbi Algorithm is a Dynamic Programming based approach to find the minimum distance codeword, for the received vector or in other words, to find the minimum Path Metric path for a received codeword.

- We minimize the Path metric by keeping track of only that path at state i, at the I^{th} stage, which has the minimum metric till then. (Greedy Approach)
- If the final path was to pass through i, it would contain the minimum (partial) weight path from 0 to i.
- *This works only because we start and end at the all-zero state.*

Pseudocode

Viterbi Algorithm

1. //For the case when there are only 2 encoder states
2. num^l is the number of states at stage l
3. List min-SM-path(l, list(num^l)) = {}

   //empty initialization of the list min-SM for all the states in each of the stages.
4. for l in range(2, number-of-stages):
5. for j in range(num^l):
6. Let i_1, i_2 be the states in layer l-1 from which there are branches into the state j of stage l.
7. if $(SM^{(l-1)}(i_1) + BM(i_1 -> j) > SM^{(l-1)}(i_2) + BM(i_2 -> j))$:
8. min-SM-path[l][j] = [min-SM-path[l-1][i_2] j]
9. else:
10. min-SM-path[l][j] = [min-SM-path[l-1][i_1] j]

Analysis of the Algorithm

- For any received vector, we can label all the branch metrics in $O(2^{\mu}(k+\mu))$ time.
- **Time Complexity**: $O(2^{\mu}(k+\mu))$
- **Space Complexity**: $O(2^{\mu}(k+\mu))$
- $k2^{\mu} <<< 2^{k}$ in the general case.

Important Implementational details

- **Windowing**: At one stage, we keep track of around 4 paths only. With around 50 stages, we can approximately find the minimum distance path on the trellis.
- **Overflow management**: If the SNR is low, then the branch metric can blow up. To manage such a situation, we subtract the minimum of the path lengths and keep the residues for further comparisons, to prevent overflow.

Translation to Hard ML Decoding

Instead of Euclidean Distance, we use the hamming distance in the branch metric calculation.
Otherwise, the decoding is exactly the same algorithm, with the same complexity.