Error correction capability of Viterbi decoder over the AWGN channel

Michele Pacenti
Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721
mpacenti@email.arizona.edu

***Abstract*—The Viterbi algorithm, which was originally developed in the error correcting field, has been proved to be a very general algorithm, that can be implemented in a vast majority of fields related to Markovian processes. Here, we briefly introduce the algorithm, and show the improvement of performances that it can bring, in respect of a classical BPSK transmission.**

I. Introduction

THE Viterbi decoder [1] performs an optimal maximum likelihood decoding of a random finite-state sequence. It takes into account a graphical structure called *Trellis*, an oriented graph which construction is based on a finite-state machine or an encoding circuit. The Viterbi algorithm represents a milestone in the error correction field: even if it was originally proposed by Andrew J. Viterbi in 1967, it's still widely implemented in a vast majority of applications nowadays, even besides communications, due to its flexibility, good performances and low complexity; for instance, the Viterbi Algorithm (VA) has been used in contactless heart beat detection [2], [3], speech recognition [4], [5] or continuous gravitational-waves detection [6], [7], and in general it can be used in every context in which a Markov chain is involved. A Markov chain or Markov process is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event [8]. In our case we will implement the VA over the Additive White Gaussian Noise (AWGN) channel in order to decode a sequence generated by a finite-state machine that receives a random binary sequence as input. We will show that performing a finite-state encoding coupled with VA decoding leads to better performances in respect of a classical bipolar transmission over AWGN channel.

II. Encoding procedure

The bipolar information source, which can assume values $+1$ and -1, is encoded by the finite-state machine illustrated in Figure 1; the nodes represent the possible states, while the edges represent the passage from a state to another. Every edge is labeled as I/O: if the finite-state machine is currently in a state i and receives an input I, it will give O as output and it will pass to state j connected with i by the corresponding edge; for instance, if the FSM is in state 0 and receives -1 as input, it will output -3 and will remain in state 0; otherwise, if it receives +1 it will give +1 as output and pass to state 1.

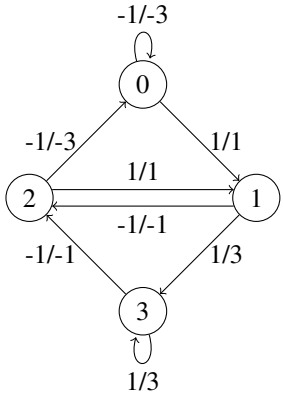

Figure 1: Finite-state machine implemented in this project.

It is evident that this type of encoding introduces correlation between consecutive symbols: infact, if the decoder receives -3, it means that the FSM is either in state 2 or state 0, and has received -1 as input; if it's in state 2, it means that the previous states could be either 1 or 3, and the previous input was -1, while if it was in state 0, the possible previous states could be 2 or 0 itself, and the previous input would be -1 as well. Therefore, we cannot perform a traditional Maximum Likelihood (ML) decoding, because it assumes uncorrelation between symbols; instead, we will adopt the Viterbi decoder.

III. Viterbi decoder

As mentioned in the Introduction, the Viterbi decoder takes into account a graph called *trellis*: the trellis is one-to-one related with a FSM, and represents all the

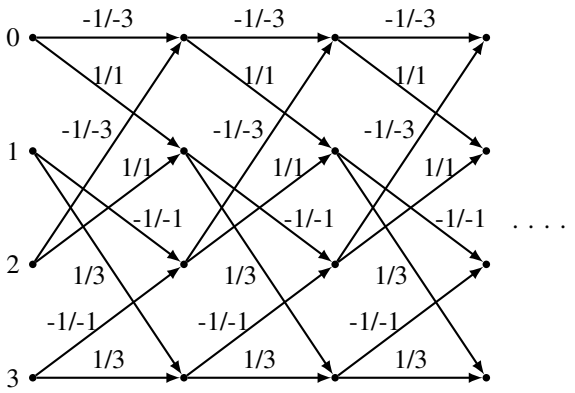

Figure 2: First 3 sections of the trellis representative of the finite-state machine in Figure 1.

possible evolutions over time of states and output. The first three sections of the trellis corresponding to the FSM of Figure 1 are illustrated in Figure 2.

Each section of a trellis represents a time step, where a symbol is received by the decoder. In every section, a set of nodes (which represent states in FSM) is connected to another, identical, set of nodes, following the connections between states in the FSM; you can see a trellis as an *unwrapped* FSM over time. The labels of edges are exactly the same as in the finite-state machine, and the number of sections is equal to the length of the received sequence. A specific received sequence corresponds exactly to one path along the trellis, i.e. to a sequence of edges, connected to each other, one for each section. Therefore, a trellis represents all the possible receivable (and transmittable as well) sequences, and the Viterbi decoder has the objective to find the most likely given the real, noisy, received one.

[h!]

The decoding procedure is depicted hereafter. The decoder takes as input a trellis section $\mathcal{T}$, which is constituted by a set of states $\mathbf{t}$ and a set of edges $\mathbf{e}$; each edge e_i is characterized by its previous state t_{i-1}, the state to which is connected to t_{i+1}, its input μ_i and output ν_i labels. First, path metric $\mathcal{W}$ and path memory $\mathcal{P}$ are initialized. Then, for each state the decoder computes the *branch metrics*: for every edge e_i connected to state t_k (therefore, we have that $t_{i+1} = t_k$) given the received symbol r and the output label ν_i, the branch metric for edge e_i is given by:

$$v_i = (r - \nu_i)^2 \tag{1}$$

(we recall that, on an AWGN channel, $r = s_m + n$ is a Gaussian random variable, which is the sum of a transmitted symbol and a Gaussian continuous noise). Thus, the branch metric is just the euclidean distance between the output symbol of an edge and the received

Algorithm 1 Viterbi Decoder

Input: $\{\mathbf{t}, \mathbf{e}\} \in \mathcal{T}$, $e_i = \{t_{i-1}, t_{i+1}, \nu_i, \mu_i\}$, r
Output: s
Persistent variables: $p_i \in \mathcal{P}$, $w_i \in \mathcal{W}$

```
if Initialization then
    P ← ∅
    w_1 ← 0
    w_i ← ∞ ∀i ∈ [2, n]
    s ← NaN
else
    for t_k ∈ T do
        W ← ∅
        for e_i s.t. t_{i+1} = t_k do
            v_i ← (ν_i − r)^2 ▷ Compute branch metric
            W_i = w_{t_{i−1}} + v_i ▷ Compute path metric
        end for
        w_k ← min_j{W}
        p_k ← [μ_j p_k]
    end for
    if time > latency then
        s ← P(end)
        P_{i+1} ← P_i ∀ i
    else
        s ← NaN
    end if
end if
return s
```

symbol; as in the classical ML detector, this metric has to be minimized in order to obtain a ML decoder. After computing the branch metrics for all the edges connected to t_k, the VA computes the *path metrics* (or *accumulated metrics*) for every path entering the state; this metric is obtained as the sum of the branch metric and the path metric accumulated at the previous state:

$$w_i = w_{t_{i-1}} + \nu_i \tag{2}$$

As every state has several edges entering in it, the VA will have to decide which path to maintain and which one to discard; in particular, the path with minimum path metric is preserved, while all the others are discarded. If two or more paths have the same metric, the decoder will choose randomly. Then, the input label μ of the selected branch will be stored in the path memory, that will act as a FIFO shift register. After several steps, all the *survivor* paths in the path memory will share the same initial values.

This procedure is repeated for every state, in every section, until the decoder stops receiving symbols from the channel, and the pseudocode for the VA is illustrated in Algorithm 1 .

It can be shown that the VA achieves an optimal ML decoding. At a certain time τ, there will be a certain

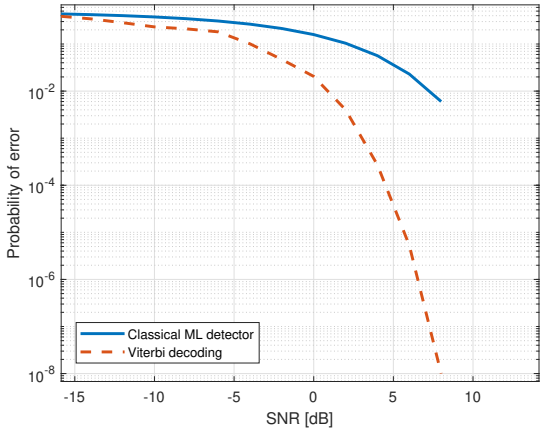

Figure 3: Comparison between theoretical BPSK and FSM encoding with Viterbi decoder.

number of possible paths survived on the trellis, each one with path metric w_i. For each state, the VA keeps only the path with minimum metrics: note that, if two paths that pass through a same state have path metrics $w_i^{(1)} < w_i^{(2)}$, in the next section $\tau+1$ they will both be incremented by the same branch metrics. Hence, we can immediately discard $w_i^{(2)}$, since we are sure that $w_i^{(1)} < w_i^{(2)}$ for every branch coming out from that state, and achieve a path with minimum euclidean distance, which corresponds with the definition of Maximum Likelihood [9]. Moreover, if each section of the trellis has n states, the VA requires just the storage of n paths.

IV. Results discussion

The decoder is implemented in a sequential way, that is one symbol in, one symbol out. It has some initialization time, which needs to accumulate metrics before actually making correct decisions on bits. The advantage of this kind of sequential decoding is that we don't need to store the entire received sequence, but just only one symbol; moreover, the path register has not to be as long as the received sequence, but has to be of an appropriate length in order to have all the paths starting with the same bits. Nonetheless, for this reason, we have to deal with a latency between the received sequence and the decoded one.

As depicted in Figure 3, it's pretty evident that introducing correlation in symbols, and adopting a sequential Viterbi decoding strategy, improves the performances of transmission, even if the the overall framework remains very simple.

V. Conclusion

We have presented the main ideas behind finite-state machine and sequential decoding with Viterbi algorithm, and showed the improvement of performances in respecto of the classical BPSK transmission.

References

[1] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[2] V. Shah, R. Anstotz, I. Obeid, and J. Picone. Adapting an automatic speech recognition system to event classification of electroencephalograms1. In *2018 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–5, 2018.

[3] Tomoaki Ohtsuki and Eriko Mogi. Heartbeat detection with doppler radar based on estimation of average r-r interval using viterbi algorithm. In *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5, 2016.

[4] Wei Liu and Weisheng Han. Improved viterbi algorithm in continuous speech recognition. In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, volume 7, pages V7–207–V7–209, 2010.

[5] F.L. Vargas, R.D.R. Fagundes, and D.B. Junior. A fpga-based viterbi algorithm implementation for speech recognition systems. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 2, pages 1217–1220 vol.2, 2001.

[6] Joe Bayley, Chris Messenger, and Graham Woan. Generalized application of the viterbi algorithm to searches for continuous gravitational-wave signals. *Phys. Rev. D*, 100:023006, Jul 2019.

[7] S. Suvorova, L. Sun, A. Melatos, W. Moran, and R. J. Evans. Hidden markov model tracking of continuous gravitational waves from a neutron star with wandering spin. *Phys. Rev. D*, 93:123009, Jun 2016.

[8] Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[9] John G Proakis, Masoud Salehi, Ning Zhou, and Xiaofeng Li. *Communication Systems Engineering*, volume 2. Prentice Hall New Jersey, 1994.