Convolutional Codewords

Keith Kumm, Tucson AZ, Summer 2016

The story of a code is spelled out in codewords. Or, in the case of convolutional codes, so it was before the BCJR algorithm. Still, in a sense of the term, BCJR's algorithm turns an entire block of encoded information and its corresponding received block of length $\tau+2$, i.e. $\boldsymbol{Y}_0^{\tau+1}$ in BCJR's paper, into one long codeword, regardless of the code's minimum distance property. The old yarn still rings true!

The idea of one long codeword suggests possible simplications to, and thus faster decoding using BCJR's method, given the ramifications of what convolutional codewords actually are.

In a linear, binary block code, codewords are definite fixed length vectors $\mathbf{y} = \mathbf{x}\,\mathbf{G}$, where $\mathbf{x}$ is the finite length information vector and $\mathbf{G}$ the code generator matrix.

But what is a *word* in convolutional encoder output? If a convolutional encoded sequence starts and terminates in a known state, it's BCJRs' "maximum" sense of a codeword. But this differs from the historical understanding.

In a linear convolutional code, a "codeword" is a sequence of encoded data that is not fixed in length, but instead of length determined by the information sequence underlying the encoded codeword and the encoder itself. This length is neither arbitrary nor fixed as in block codes. The convention was that a convolutional codeword could be defined by its Hamming distance from another codeword of the same length. If this sounds odd, it is only because of the subtlety of the meaning of a convolutional codeword.

For any code constraint length K, two encoded symbol sequences originating from one (or any) particular encoder state, i.e. from where they depart in their state-to-state paths* through the trellis to where they come back together at one (or any) particular *common* encoder state, are two codewords of the same length.

Code linearity** permits the convenience of using a departure from the all-zeros (**0**) state and (a first) convergence of the two paths taken by the two codewords back at *any* common state. Linearity then allows for an additional trick. If one of the two codewords is taken to actually be the **0** encoder output word (of the same length as the other, later-merging, non-all-zeros codeword), then we consider only a convergence back to the **0** state. The two sequences, the **0** encoder output (due to **0** information) and the non-**0** sequence (due to non-**0** information) are both codewords. The Hamming distance between two such encodings is the codeword distance that constrains how many errors can be corrected in an "error burst" over that length of code symbols. Since we work only with states, then the shortest possible non-zero codeword is the one that leads off with a single non-zero information symbol (in binary codes, 1) and then returns to all-zero (binary 0) information symbols. The shortest codeword length is then the quotient of the

constraint length and the code rate, i.e. K/r_c. But is this also the minimum distance for all pairs of equal-length codewords output from the encoder?

We can look at each criteria, the raw information and **G**, one at a time. First, any short sub-sequence of "random" data (and sufficient "scrambling" assures this) longer than K will eventually repeat, and then the delay to a (first) repetition is the length of a codeword in the linear code sense. It could be a long departure from all zeros, but the Poisson probability of not yet reaching $K-1$ zeros of information falls quickly with length. For these sub-sequences longer than K, codeword length is either only weakly dependent on or even independent of **G.** The set of codewords forms a distribution of lengths, from $K+1$ and upward, culminating in a thin tail.

When the "random" sub-sequence which departs and again converges at the same state (e.g., all zeros) is less than or equal to K in length, both the information and **G** strongly or equally determine codeword length. There will be a distribution of lengths for this set, and the fact of a distribution denies a definite fixed length.

Due to distribution, theorists resorted to a description of convolutional codeword distance that, pun intended, distanced itself from that of block codes, which were already shown inferior in an exponential error rate bounding sense.

Because of the very distribution of codeword lengths, one can say that the length of a codeword is "free," meaning not confined to a constant value.

Given any two codewords, i.e. which depart from each other then converge at the same state as they departed from, we now have a codeword distance. Each such distance is one of the many "free distances" d_{free} distributed by the code. And since any "random" information source will produce sub-sequences fitting both criteria above, we have d_{free} ranging from a minimum value, determined by the shortest information sequence differing from $K-1$ zeros and the length and composition of **G**, up to some maximum value in the tail of the information sequence distribution. One could call it a spectrum of d_{free} lengths.

The minimum free distance, $d_{\text{free,min}}$, is the key to the code's strength. It is nice that $d_{\text{free,min}}$ be as large as possible. The $d_{\text{free,min}}$ is almost the only determinant of strength for memoryless channels, although some codes might be strong in the sense that $d_{\text{free,min}}$ is small but codeword lengths approaching it (from above) are few in number. Absent side information and iteration, i.e. when there is but one calculation available to determine the most probable information sequence corresponding to a corrupted received encoded sequence, a code is only guaranteed to correct errors causing a Hamming distance between the sent and received *encoded* sequences of less than $d_{\text{free,min}}/2$. This is analogous to the $d_{\text{min}}/2$ error correction criteria of an algebraic codeword. A main if not the main difference between convolutional and algebraic code strengths of the same code rate is that convolutional codes can also correct error patterns in codewords for which $d_{\text{free}} > d_{\text{free,min}}$, whereas block codes fail at once for heavier error patterns.

* A *path* is the sequence of convolutional encoder states through a trellis resulting from an input information bit sequence.

** When a code is linear, i.e. when any codeword can be expressed as the sum of two other codewords, then all features of a code can be between *pairs of sums* of codewords. Hence comparisons between any two codewords can be made as if comparing one codeword, as a particular sum of two, to the all zeros codeword.