Revisiting the BCJR Algorithm

Keith Kumm, Tucson AZ, Summer 2016

1 Introduction

Bahl, Cocke, Jelenick and Raviv (BCJR) wrote their 1974 IEEE correspondence[1] paper compactly. The development of their creative *a posteriori probablity* (APP) algorithm unfolds flawlessly up to a boundary condition anomaly, easily resolved, but a portent.

The APP algorithm itself is satisfactorily completed. They apply it to convolutional and block decoding, only convolutional considered here. A maximum *a posteriori* (MAP) decoding rule is then described, with issues unvisited by BCJR. We are left with the enticing algorithm, but no simple way, at least in 1974, to build a BCJR decoder. The algorithm languished until 1992 or so when turbo coding appeared.

This revisit reconstructs the algorithm using a more conventional definition of encoder state. The APP algorithm development is rearranged for a possibly better flow among dependencies, expanding the block diagram and adding view of the encoded block for clarity, correcting an error and an ambiguity, and adding discussion. This in no way diminishes BCJR's achievement, the algorithm itself.

The BCJR algorithm contrasts with Viterbi's earlier maximum likelihood (ML) algorithm and his insightful follow-up paper on a forward-and-backward version, the subject of a separate review[2]. The soft output Viterbi algorithm (SOVA) is the most relevant comparison to BCJR, as both are probability estimators.

Both the BCJR and Viterbi algorithms are based on Bayes rule, but on opposite sides of the rule.

MAP and ML differ fundamentally, as discussed elsewhere[2]. Thus, so do BCJR and Viterbi decoding, including Viterbi's 1998 revival paper approximating MAP.

BCJR start their paper with a nod to the Viterbi algorithm, but emphasize its non-optimality for code symbols. The competitive tension in this challenge to ML decoding is palpable. They go on to declare an optimum (MAP) decoding method for linear codes. This turns out to be a bit premature from an implementation standpoint, and they acknowledge as much in terms of the complexity to realize a decoder. Whether there is a "good" way to decode, consistent with their compare-with-½ decoding rule, may depend on the application. Still, the algorithm stands.

---

[1] Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate by Bahl, Cocke, Jelenick and Raviv, March 1974, *IEEE Transactions on Information Theory*

[2] Comparing the BCJR and Viterbi Algorithms, K. Kumm, Summer 2016

2 The Algorithm

BCJR's algorithm is a procedure to determine a set of individual, absolute-valued symbol APPs (probabilities) consistent with an entire received set of encoded data. It is *locally* and *globally* accurate for *each* received code symbol. BCJR sketch out a way to make the algorithm apply accurately to longer sequences, like codewords, possibly seeking competitiveness in this sense with the Viterbi algorithm.

The BCJR algorithm is *inherently* block data-oriented, which is simultaneously a weakness and a strength. It poses an obstacle for use in continuous, non-blocked data. But it also matches it up perfectly for use with blocked data.

The essence of the BCJR algorithm is the rule for assigning APPs to all trellis states and transitions between states at each time-depth in the trellis. This is not yet a decoding rule, let alone a MAP decoding rule. A way to convert these APPs into MAP decisions is described up to a point.

The BCJR algorithm can be applied to algebraic codes, convolutional codes, turbo codes and other linear codes, such as low density parity check codes. Making the actual information decisions varies, sometimes allowing immediate decisions and sometimes requiring belief propagation and iteration for reliable decisions.

To improve on the algorithm's statement, explicit notation is used for variables in hypothesis, a symmetric notation for precedence of states is introduced, an error is corrected for a required extension of received *Y* by introducing an initialization event *Z*, a notation error regarding precedence of states is overcome using the symmetric notation, and the order of presentation is slightly rearranged. Some explanatory expressions in the paper are simply dropped where possible.

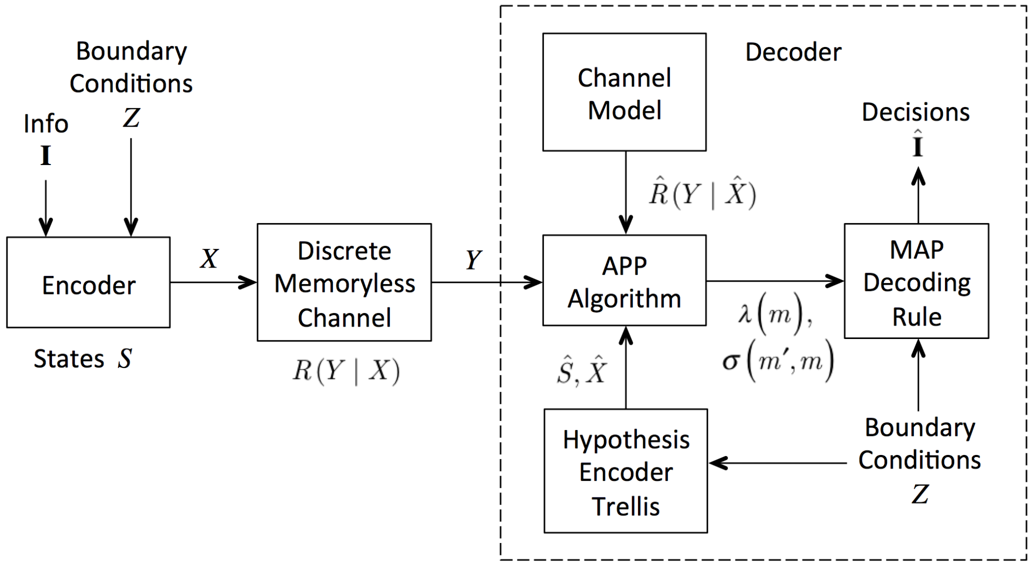
The use of *Z* made explicit throughout can become fatiguing, so major expressions are reproduced in a later section stripped of *Z* to facilitate following the logic along and comparing to the paper. Expressions will not always compare! There are two typographical corrections, a change in the range of time and other changes having nothing to do with *Z*. Stripped expressions are valid away from block boundaries.

Here then, with changes and additions, but using mostly the same notation, is the BCJR algorithm.

Objective: For a code of rate k_0 / n_0 having a trellis representation, find the probabilities $\lambda_t\left(m\right)$, for state *m* and $\sigma_t\left(m',m\right)$ for all supported state transitions from *m′* to *m*, in the forward direction through the trellis, and a complete, received encoded sequence $Y \equiv Y_0^{\tau+1}$, at every trellis node time $t = 0, 1, \ldots, \tau, \tau+1$, i.e. for all states and trellis depths denoted by time, given known boundary conditions of the starting and ending states of the trellis.

Reference Diagram:

The diagram indicates the elements of the decoder and also displays the difference between actual information, states, encoded symbols and channel transition function, versus their hypotheses (shown with carats) in the decoder.



Algorithm:

For a regular trellis of code rate k_0/n_0, and information symbol $\mathbf{I}$ of uniform pdf, the *a priori* probabilities of sending encoder states joined by a transition at t are

$$p_t(m \mid m') = \Pr\{\hat{S}_t = m \mid \hat{S}_{t-1} = m'\} = 2^{-k_0}$$

$$q_t(X_t = X \mid m', m) = \Pr\{X_t = X \mid S_{t-1} = m', S_t = m\} \in \{0, 1\}$$

for the transition probability of state m' to m, and the probability of code symbol X being selected by encoder from an alphabet after the transition.

In the decoder, over the set M of all *hypothesized* encoder states, the joint event $\{\hat{S}_t = m; Y; Z\}$ has *a posteriori* probability

$$\lambda_t(m) \equiv \Pr\{\hat{S}_t = m; Y; Z\} = \Pr\{\hat{S}_t = m; Y_0^t; Z\} \cdot \Pr\{Y_{t+1}^{\tau+1}; Z \mid \hat{S}_t = m\}$$ for

$Z \Leftrightarrow \{Y_0 = X_0; Y_{\tau+1} = X_{\tau+1}\}$, where encoded symbols $\{X_0, X_{\tau+1}\}$ result from known boundary conditions, by inputting to the encoder convenient assigned information by the sending encoder in a convenient assignment to $\left\{\mathbf{I}_0, \mathbf{I}_{\tau+1}\right\} = \left\{0\,,\,0\right\}$ (if binary) or $\left\{\mathbf{0}\,,\,\mathbf{0}\right\}$ (in general) at $t = 0$ and from $t = \tau - \nu$ to $t = \tau + 1$.

For encoder generator **G** of constraint length *ν*, shown for $\nu \geq 5$ for clarity, but extensible to $\nu \geq 2$ without loss of generality, the decoder's *hypothesis* at each time for the encoded sequence $\hat{X}$ and encoder state $\hat{\mathbf{S}}$, given the *actual* *X* and **S**, is

$$\hat{X}_0 = X_0 = \left[\mathbf{I}_0\ \mathbf{S}_{-1}\right]\mathbf{G},\ \ \mathbf{I}_0 = \mathbf{0},\ \ \mathbf{S}_{-1} \equiv \left[\mathbf{0}_{-1}\ \mathbf{0}_{-2}\ \ldots\ \mathbf{0}_{-(\nu-2)}\ \mathbf{0}_{-(\nu-1)}\right]$$

$$\hat{X}_1 = \left[\hat{\mathbf{I}}_1\ \mathbf{S}_0\right]\mathbf{G},\ \ \mathbf{S}_0 = \left[\mathbf{0}_0\ \mathbf{0}_{-1}\ \ldots\ \mathbf{0}_{-(\nu-3)}\ \mathbf{0}_{-(\nu-2)}\right]$$

$$\hat{X}_2 = \left[\hat{\mathbf{I}}_2\ \mathbf{S}_1\right]\mathbf{G},\ \ \hat{\mathbf{S}}_1 = \left[\hat{\mathbf{I}}_1\ \mathbf{0}_0\ \mathbf{0}_{-1}\ldots\ \mathbf{0}_{-(\nu-3)}\right]$$

…

$$\hat{X}_{\tau-(\nu-1)} = \left[\hat{\mathbf{I}}_{\tau-(\nu-1)}\ \hat{\mathbf{S}}_{\tau-\nu}\right]\mathbf{G},\ \ \hat{\mathbf{S}}_{\tau-\nu} = \left[\hat{\mathbf{I}}_{\tau-\nu}\ \hat{\mathbf{I}}_{\tau-(\nu+1)}\ \ldots\ \hat{\mathbf{I}}_{\tau-(\nu-(3-\nu))}\ \hat{\mathbf{I}}_{\tau-(\nu-(2-\nu))}\right]$$

$$\hat{X}_{\tau-(\nu-2)} = \left[\mathbf{I}_{\tau-(\nu-2)}\ \hat{\mathbf{S}}_{\tau-(\nu-1)}\right]\mathbf{G},\ \ \mathbf{I}_{\tau-(\nu-2)} = \mathbf{0},\ \ \hat{\mathbf{S}}_{\tau-(\nu-1)} = \left[\hat{\mathbf{I}}_{\tau-(\nu-1)}\ \hat{\mathbf{I}}_{\tau-\nu}\ \ldots\ \hat{\mathbf{I}}_{\tau-(\nu-(4-\nu))}\ \hat{\mathbf{I}}_{\tau-(\nu-(3-\nu))}\right]$$

$$\hat{X}_{\tau-(\nu-3)} = \left[\mathbf{I}_{\tau-(\nu-3)}\ \hat{\mathbf{S}}_{\tau-(\nu-2)}\right]\mathbf{G},\ \ \mathbf{I}_{\tau-(\nu-3)} = \mathbf{0},\ \ \hat{\mathbf{S}}_{\tau-(\nu-2)} = \left[\mathbf{0}_{\tau-(\nu-2)}\ \hat{\mathbf{I}}_{\tau-(\nu-1)}\ \ldots\ \hat{\mathbf{I}}_{\tau-(\nu-(5-\nu))}\ \hat{\mathbf{I}}_{\tau-(\nu-(4-\nu))}\right]$$

…

$$\hat{X}_{\tau} = \left[\mathbf{I}_{\tau}\ \hat{\mathbf{S}}_{\tau-1}\right]\mathbf{G},\ \ \mathbf{I}_{\tau} = \mathbf{0},\ \ \hat{\mathbf{S}}_{\tau-1} = \left[\mathbf{0}_{\tau-1}\ \mathbf{0}_{\tau-2}\ \ldots\ \mathbf{0}_{\tau-(\nu-2)}\ \hat{\mathbf{I}}_{\tau-(\nu-1)}\right]$$

$$\hat{X}_{\tau+1} = X_{\tau+1} = \left[\mathbf{I}_{\tau+1}\ \mathbf{S}_{\tau}\right]\mathbf{G},\ \ \mathbf{I}_{\tau+1} = \mathbf{0},\ \ \mathbf{S}_{\tau} = \left[\mathbf{0}_{\tau}\ \mathbf{0}_{\tau-1}\ \ldots\ \mathbf{0}_{\tau-(\nu-3)}\ \mathbf{0}_{\tau-(\nu-2)}\right]$$

Let any two { previous, next } states = $\left\{\overset{\frown}{m}\,,\,\breve{m}\right\}$ in a relevant direction, forward or backward through the trellis, generalize the precedence pairs $\left\{m'\,,\,m\right\}$, $\left\{m\,,\,{}^{\backprime}m\right\}$ expressing forward to *m* and backward to *m*, respectively

The channel transition probability function *R* at received code Y_t symbol, independent of direction through the trellis is

$$\hat{R}\left(Y_t \mid \hat{X}\,;Z\right) = \hat{R}\left(Y_t \mid \hat{X}_t = \hat{X}\,;Z\right) = \Pr\left\{Y_t \mid \hat{X}_t = \hat{X}\,;Z\right\}$$

and over a code symbol (actual, hypothesis) = $\left(y_t, \hat{x}_t\right)$, n_0 code bits in length

$$\hat{R}\left(Y_t\,;Z \mid \hat{X}_t\right) = \prod_{j=1}^{n_0} \hat{r}\left(y_t^j\,;z^j \mid \hat{x}_t^j\right)$$

For hypothesis encoder **G** output $\hat{X}_t$ in the forward direction, the transition probability between two states *jointly* with received Y_t is, by definitions

$$\gamma_t\left(\overset{\frown}{m}, \breve{m}\right) \equiv \Pr\left\{\hat{S}_t = \breve{m}; Y_t \mid \hat{S}_{t-1} = \overset{\frown}{m}; Z\right\}$$

$$= \sum_{\hat{X}} \left[\begin{array}{l} \Pr\left\{\hat{S}_t = \breve{m} \mid \hat{S}_{t-1} = \overset{\frown}{m}; Z\right\} \cdot \Pr\left\{\hat{X}_t = \hat{X} \mid \hat{S}_{t-1} = \overset{\frown}{m}, \hat{S}_t = \breve{m}; Z\right\} \\ \qquad \cdot \Pr\left\{Y_t \mid \hat{X}_t = \hat{X}; \hat{S}_{t-1} = \overset{\frown}{m}\,; \hat{S}_t = \breve{m}\,; Z\right\} \end{array} \right],$$

$$= \sum_{\hat{X}} p_t\left(\breve{m} \mid \overset{\frown}{m}; Z\right) \cdot q_t\left(\hat{X}_t = \hat{X} \mid \overset{\frown}{m}, \breve{m}; Z\right) \cdot \hat{R}\left(Y_t \mid \hat{X}; Z\right)$$

$$\gamma_t\left(\breve{m}, \overset{\frown}{m}\right) = \gamma_t\left(\overset{\frown}{m}, \breve{m}\right).$$

The probabilities of two complementary, mutually exclusive sub-sequences divide the total probability $\lambda_t\left(m\right)$ of the complete sequence Y, at each state m at t, as

$$\alpha_t\left(m\right) \equiv \Pr\left\{\hat{S}_t = m; Y_0^t; Z\right\} = \alpha_t\left(\breve{m}\right) = \sum_{\overset{\frown}{m}} \alpha_{t-1}\left(\overset{\frown}{m}\right)\gamma_t\left(\overset{\frown}{m}, \breve{m}\right)$$

$$\beta_t\left(m\right) \equiv \Pr\left\{Y_{t+1}^{\tau+1}; Z \mid \hat{S}_t = m\right\} = \beta_t\left(\breve{m}\right) = \sum_{\overset{\frown}{m}} \beta_{t+1}\left(\overset{\frown}{m}\right)\gamma_{t+1}\left(\breve{m}, \overset{\frown}{m}\right).$$

By three definitions

$$\lambda_t\left(m\right) = \alpha_t\left(m\right) \cdot \beta_t\left(m\right) .$$

Next

$$\begin{array}{l} \left\{\hat{S}_{t-1} = \overset{\frown}{m}; Y_0^{t-1}; Z\right\} \bigcap \left\{\hat{S}_t = \breve{m}; Y_t\,; Z \mid \hat{S}_{t-1} = \overset{\frown}{m}\right\} \bigcap \left\{Y_{t+1}^{\tau+1}; Z \mid \hat{S}_t = \breve{m}; Z\right\} \\ \quad \Leftrightarrow \left\{\hat{S}_{t-1} = \overset{\frown}{m}; \hat{S}_t = \breve{m}; Y\,; Z\right\} \end{array},$$

so by definitions

$$\sigma_t\left(m',m\right) \equiv \Pr\left\{\hat{S}_{t-1} = \overset{\frown}{m}\,;\ \hat{S}_t = \breve{m}\,;\ Y\,;\ Z\right\} = \alpha_{t-1}\left(\overset{\frown}{m}\right)\cdot\gamma_t\left(\overset{\frown}{m},\breve{m}\right)\cdot\beta_t\left(\breve{m}\right)\,.$$

$\alpha_{t-1}\left(\overset{\frown}{m}\right),\ t = 1,2,\ \ldots,\ \tau$ and $\beta_t\left(\breve{m}\right),\ t = \tau, \tau-1,\ \ldots,\ 1$ are computable recursively given boundary conditions consistent with known states driven by Z

$\alpha_0\left(0\right) = 1\,;\ \alpha_0\left(m\right) = 0\ ,\ m \neq 0$ at the start of Y,

$\beta_{\tau+1}\left(0\right) = 1\,;\ \beta_{\tau+1}\left(m\right) = 0\ ,\ m \neq 0$ at the end of Y.

At each trellis depth, or *t*, the two joint probabilities $\lambda_t\left(m\right)$, $\sigma_t\left(m',m\right)$, for all states, are evaluated *promptly*, as soon as $\alpha_{t-1}\left(\overset{\frown}{m}\right), \alpha_t\left(\overset{\frown}{m}\right), \beta_t\left(\breve{m}\right)$ and $\gamma_t\left(\overset{\frown}{m},\breve{m}\right)$ are determined, to produce their individual APP state probabilities and stored, then if desired, used to make an information decision for *t* using an additional procedure.

3 Comments on Expressions

Use of $t = 0\ ,1\ ,\ \ldots\ ,\tau\ ,\tau+1$ for the total range of times (depths) in the trellis, an implication in the paper made clear near the end of it, is made tangible at the outset. That implication is that sent information may only be arbitrary (useful) for $t = 1\ ,\ \ldots\ ,\tau-\left(\nu-1\right)$ with the boundaries at $t = 0$ and $t = \tau-\nu\ ,\ \ldots\ ,\ \tau+1$ constrained to known values (e.g., 0). The "padding" is not symmetric around the block! This boundary value padding allows the recursion for $\beta_t\left(m\right)$ and the assignment of $\alpha_0\left(0\right)$ without "breaking" the general expressions.

The information below is binary except where it is recognized explicitly that it can be non-binary, as a vector **I**. Bits are the baseline for these comments.

The definition of state here is the "memory" of the encoder, from output symbol to symbol. In this conventional way, current state plus next information = next state. This is visualized as the first $\nu-1$ stages of a shift register of length *ν*, with the current information bit (reflected by the current output *X*) already shifted in and part of the state. The last information bit in the register is not part of the state, even though it is reflected by *X*. This is consistent with the paper's trellis figure and markings for "State," as it must be, but as a consequence it extends the trellis at the beginning and at the end by one symbol time in one transition line connecting two *m* = 0 states (at each end).

The meaning of a supported state transition from *m′* to *m* is that the encoder, for any generator **G** (encompassing the polynomial multipliers in conventional convolutional codes), makes possible two consecutive states (m', m) at times (t_{n-1}, t_n) where $m(1:\nu-1) = \left[\mathbf{I}_m \; m'(1:\nu-2)\right]$, for all $\mathbf{I}_m$. State pairs impossible for the encoder do not support state transition.

Regarding the sequences (actual, hypothesized) for code symbols X, $\hat{X}$, and states $\mathbf{S}$, $\hat{\mathbf{S}}$, with known data substituted where possible,

$$\begin{array}{l} X_0 \\ \left[X_1 \; X_2 \cdots X_{\tau-(\nu-1)}\right] \\ \left[X_{\tau-(\nu-2)} \cdots X_\tau \; X_{\tau+1}\right] \end{array}, \quad \begin{array}{l} X_0 \\ \left[\hat{X}_1 \; \hat{X}_2 \cdots \hat{X}_{\tau-(\nu-1)}\right] \\ \left[\hat{X}_{\tau-(\nu-2)} \cdots \hat{X}_\tau \; X_{\tau+1}\right] \end{array} \text{ and } \begin{array}{l} \mathbf{S}_{-1} \\ \left[\mathbf{S}_0 \; \mathbf{S}_1 \cdots \mathbf{S}_{\tau-\nu}\right] \\ \left[\mathbf{S}_{\tau-(\nu-1)} \cdots \mathbf{S}_{\tau-1} \; \mathbf{S}_\tau\right] \end{array}, \quad \begin{array}{l} \mathbf{S}_{-1} \\ \left[\mathbf{S}_0 \; \hat{\mathbf{S}}_1 \cdots \hat{\mathbf{S}}_{\tau-\nu}\right] \\ \left[\hat{\mathbf{S}}_{\tau-(\nu-1)} \cdots \hat{\mathbf{S}}_{\tau-1} \; \mathbf{S}_\tau\right] \end{array}$$

correspond to information sequences I, $\hat{\mathbf{I}}$

$$\begin{array}{l} \mathbf{I}_0 \\ \left[\mathbf{I}_1 \; \mathbf{I}_2 \cdots \mathbf{I}_{\tau-(\nu-1)}\right] \\ \left[\mathbf{I}_{\tau-(\nu-2)} \cdots \mathbf{I}_\tau \; \mathbf{I}_{\tau+1}\right] \end{array}, \quad \begin{array}{l} \mathbf{I}_0 \\ \left[\hat{\mathbf{I}}_1 \; \hat{\mathbf{I}}_2 \cdots \hat{\mathbf{I}}_{\tau-(\nu-1)}\right] \\ \left[\mathbf{I}_{\tau-(\nu-2)} \cdots \mathbf{I}_\tau \; \mathbf{I}_{\tau+1}\right] \end{array}$$

The information $\left\{\mathbf{I}_0 \left[\mathbf{I}_{\tau-(\nu-2)} \cdots \mathbf{I}_\tau \; \mathbf{I}_{\tau+1}\right]\right\}$ of lengths $\left\{1, 1-\left(-(\nu-2)\right)\right\} = \left\{1, \nu-1\right\}$ is predetermined (known, e.g. as zeros), so only the information $\left[\mathbf{I}_1 \; \mathbf{I}_2 \cdots \mathbf{I}_{\tau-(\nu-1)}\right]$ of length $1-\left(\tau-(\nu-1)\right) = \tau-\nu$ is useful. The information is, for binary codes, of one bit each (per vector shown). For non-binary codes, it is the vectors $\mathbf{I}_n$.

Regarding the *joint* state transition, received symbol Y_t probability γ_t, the development of this expression is one of the harder ones to see. Consider first the two events

$$\left\{\hat{X}_t = \hat{X} \,;\, \hat{S}_{t-1} = \widehat{m} \,;\, \hat{S}_t = \breve{m} \,;\, Z\right\}, \left\{\hat{X}_t = \hat{X} \,;\, Z\right\}.$$

The only reason these two events have a common result in the sum over X expression below is the "sorting filter" function imposed by the product

$$p_t\left(\breve{m} \mid \widehat{m};\, Z\right) \cdot q_t\left(\hat{X}_t = \hat{X} \mid \widehat{m},\, \breve{m};\, Z\right).$$

The reason has nothing to do a “unique” $\hat{X}$. $\hat{R}\left(Y_t \mid \hat{X};Z\right)$ is not unique to just this state transition! It appears many times at each trellis depth for transitions in large constraint length codes. The “filter” is just the combined effect of the *p* and *q* functions, allowing the contraction of dropping dependence on $\hat{S}_{t-1} = \overset{\frown}{m}$; $\hat{S}_t = \breve{m}$.

$$\gamma_t\left(\overset{\frown}{m},\breve{m}\right) \equiv \Pr\left\{\hat{S}_t = \breve{m}; Y_t \mid \hat{S}_{t-1} = \overset{\frown}{m}; Z\right\}$$

$$= \sum_{\hat{X}} \left[\begin{array}{c} \Pr\left\{\hat{S}_t = \breve{m} \mid \hat{S}_{t-1} = \overset{\frown}{m}; Z\right\} \cdot \Pr\left\{\hat{X}_t = \hat{X} \mid \hat{S}_{t-1} = \overset{\frown}{m}, \hat{S}_t = \breve{m}; Z\right\} \\ \cdot \Pr\left\{Y_t \mid \hat{X}_t = \hat{X}; \hat{S}_{t-1} = \overset{\frown}{m} ; \hat{S}_t = \breve{m} ; Z\right\} \end{array}\right]$$

$$= \sum_{\hat{X}} p_t\left(\breve{m} \mid \overset{\frown}{m}; Z\right) \cdot q_t\left(\hat{X}_t = \hat{X} \mid \overset{\frown}{m}, \breve{m}; Z\right) \cdot \hat{R}\left(Y_t \mid \hat{X}; Z\right)$$

where $\hat{R}\left(Y_t \mid \hat{X};Z\right)$ is conditional, not joint, a typographical error in the paper.

The middle sum above is required for equivalence to $\Pr\left\{\hat{S}_t = \breve{m}; Y_t \mid \hat{S}_{t-1} = \overset{\frown}{m}; Z\right\}$. This is because of a conditional probability identity on which the expression is based. This can be checked in a compact way, with shortened nomenclature

$\mathrm{A} : \hat{S}_{t-1} = \overset{\frown}{m}, \quad \mathrm{B} : \hat{S}_t = \breve{m}, \quad \mathrm{X} : \hat{X}_t = \hat{X}, \quad \mathrm{Y} : Y_t \quad \mathrm{P} : \Pr$, and *Z* dropped for clarity.

The expression for $\gamma_t\left(\overset{\frown}{m},\breve{m}\right)$ now translates to

$\mathrm{P}\left\{\mathrm{B},\mathrm{Y} \mid \mathrm{A}\right\} = \sum_{\mathrm{X}} \left[\mathrm{P}\left\{\mathrm{B} \mid \mathrm{A}\right\} \cdot \mathrm{P}\left\{\mathrm{X} \mid \mathrm{A},\mathrm{B}\right\} \cdot \mathrm{P}\left\{\mathrm{Y} \mid \mathrm{X},\mathrm{A},\mathrm{B}\right\}\right]$, to be verified.

Since all terms in the expression are conditioned on A, and because A does not appear in any lefthand argument of a probability, the entire expression can be contracted to conditional on A, now just implied, and stripped from each term.

$$\mathrm{P}\left\{\mathrm{B},\mathrm{Y}\right\} = \sum_{\mathrm{X}} \left[\mathrm{P}\left\{\mathrm{B}\right\} \cdot \mathrm{P}\left\{\mathrm{X} \mid \mathrm{B}\right\} \cdot \mathrm{P}\left\{\mathrm{Y} \mid \mathrm{X},\mathrm{B}\right\}\right] \qquad \left(\forall \mid \mathrm{A}\right)$$

$$\mathrm{P}\left\{\mathrm{B},\mathrm{Y}\right\} = \sum_{\mathrm{X}} \left[\mathrm{P}\left\{\mathrm{Y} \mid \mathrm{X},\mathrm{B}\right\} \cdot \mathrm{P}\left\{\mathrm{X} \mid \mathrm{B}\right\} \cdot \mathrm{P}\left\{\mathrm{B}\right\}\right]$$

$$\mathrm{P}\left\{\mathrm{B},\mathrm{Y}\right\} = \sum_{\mathrm{X}} \left[\mathrm{P}\left\{\mathrm{Y} \mid \mathrm{X},\mathrm{B}\right\} \cdot \mathrm{P}\left\{\mathrm{X},\mathrm{B}\right\}\right] = \sum_{\mathrm{X}} \mathrm{P}\left\{\mathrm{Y},\mathrm{X},\mathrm{B}\right\} = \mathrm{P}\left\{\mathrm{Y},\mathrm{B}\right\} \quad \left(\forall \mid \mathrm{A}\right)$$

$\mathrm{P}\left\{\mathrm{B},\mathrm{Y} \mid \mathrm{A}\right\} = \mathrm{P}\left\{\mathrm{B},\mathrm{Y} \mid \mathrm{A}\right\}$, QED.

Regarding the joint transition probabilities $\gamma_t\left(\breve{m}, \overset{\frown}{m}\right) = \gamma_t\left(\overset{\frown}{m}, \breve{m}\right)$, the transitions are symmetric, hence the probabilities identical, for fixed times $t-1$ and t. This is because none of the three probabilities $\left\{p_t, q_t, R\right\}$ depend on direction. If the computations of γ_t in one direction are saved, they can be reused in the opposite direction. And to write

$$\begin{aligned}\gamma_t\left(\breve{m}, \overset{\frown}{m}\right) &= \Pr\left\{\hat{S}_{t-1} = \breve{m}; Y_t; Z \mid \hat{S}_t = \overset{\frown}{m}\right\} \\ &= \sum_{\hat{X}} p_t\left(\overset{\frown}{m} \mid \breve{m}\right) \cdot q_t\left(\hat{X} \mid \breve{m}, \overset{\frown}{m}\right) \cdot \hat{R}\left(Y_t \mid \hat{X}\right)\end{aligned}$$

is unnecessary unless values going forward are not saved for going backward.

Regarding the recursions

$$\alpha_t\left(m\right) \equiv \Pr\left\{\hat{S}_t = m; Y_0^t; Z\right\} = \alpha_t\left(\breve{m}\right) = \sum_{\overset{\frown}{m}} \alpha_{t-1}\left(\overset{\frown}{m}\right) \gamma_t\left(\overset{\frown}{m}, \breve{m}\right)$$

$$\beta_t\left(m\right) \equiv \Pr\left\{Y_{t+1}^{\tau+1}; Z \mid \hat{S}_t = m\right\} = \beta_t\left(\breve{m}\right) = \sum_{\overset{\frown}{m}} \beta_{t+1}\left(\overset{\frown}{m}\right) \gamma_{t+1}\left(\breve{m}, \overset{\frown}{m}\right),$$

they have the same meaning as

$$\alpha_t\left(m\right) = \sum_{m'} \alpha_{t-1}\left(m'\right) \gamma_t\left(m', m\right), \quad \beta_t\left(m\right) = \sum_{`m} \beta_{t+1}\left(`m\right) \gamma_{t+1}\left(m, `m\right)$$

but avoid the paper's confusion of reusing m' for two different purposes, once in $\alpha_t\left(m\right)$ as a preceding variable, and a second time in $\beta_t\left(m\right)$ as a dummy variable.

The odd fact that $\alpha_t\left(m\right)$ is a purely joint probability, while $\beta_t\left(m\right)$ is a conditional probability is explained by the disjoint-ness of events $\left\{\hat{S}_t = m\right\}$ and $\left\{Y_{t+1}^{\tau+1}\right\}$. The $\beta_t\left(m\right)$ has a "target of $\left\{\hat{S}_t = m\right\}$, while $\alpha_t\left(m\right)$ is concurrent with Y_0^t. It may not be intuitive. But it works, perfectly.

Regarding the equivalence of events

$$\begin{aligned}&\left\{\hat{S}_{t-1} = \overset{\frown}{m}; Y_0^{t-1}; Z\right\} \bigcap \left\{\hat{S}_t = \breve{m}; Y_t; Z \mid \hat{S}_{t-1} = \overset{\frown}{m}\right\} \bigcap \left\{Y_{t+1}^{\tau+1}; Z \mid \hat{S}_t = \breve{m}; Z\right\} \\ &\quad \Leftrightarrow \left\{\hat{S}_{t-1} = \overset{\frown}{m}; \hat{S}_t = \breve{m}; Y; Z\right\}\end{aligned}$$

This can be shown in two steps, ignoring Z for clarity

$$\left\{\hat{S}_{t-1}=\widehat{m};Y_0^{t-1}\right\}\bigcap\left\{\hat{S}_t=\breve{m};Y_t\mid\hat{S}_{t-1}=\widehat{m}\right\}\Leftrightarrow\left\{\hat{S}_{t-1}=\widehat{m};\hat{S}_t=\breve{m};Y_0^t\right\}$$

$$\left\{\hat{S}_{t-1}=\widehat{m};\hat{S}_t=\breve{m};Y_0^t\right\}\bigcap\left\{Y_{t+1}^{\tau+1}\mid\hat{S}_t=\breve{m}\right\}\Leftrightarrow\left\{\hat{S}_{t-1}=\widehat{m};\hat{S}_t=\breve{m};Y_0^{\tau+1}\right\}.$$

This also corrects a minor typographical error in the paper, where "|" for condition did not show in $\Pr\left\{Y_{t+1}^{\tau+1};Z\mid\hat{S}_t=\breve{m}\right\}$.

Ultimately, this equivalence works due to the three separate events in the event intersections being mutually exclusive.

However, every aspect of the algorithm must potentially deal with the boundary conditions Z, and hence placing it explicitly into expressions *where it applies* forces saliency and avoids questions about the "impossible" consequences for t taking on values 0 and $\tau+1$, as always occurs. Using Z, the impossibilities vanish. The consequence is that information may be arbitrary for all of $t = 1, \ldots, \tau$, since $\alpha_{t-1}\left(\widehat{m}\right)$ and $\beta_{t+1}\left(\breve{m}\right)$ are boundary conditions at $t = 1$ and $t=\tau$ based on *known* information thereafter, and so do not require further recursion, just assignment.

BCJR didn't obtain this result, requiring the explicit $Y_0^{\tau+1}$, and that can be seen as a minor error.

4 Z-Free Expressions

Below are the main expressions of the algorithm, in order, without explanations. Where they depend on Z, they are stripped of it, marked (Z), indented and become inapplicable at boundaries. If familiar with the definition of all terms, the entire algorithm can be "browsed" here.

$$p_t\left(m\mid m'\right)=\Pr\left\{\hat{S}_t=m\mid\hat{S}_{t-1}=m'\right\}=2^{-k_0}$$

$$q_t\left(X_t=X\mid m',m\right)=\Pr\left\{X_t=X\mid S_{t-1}=m',S_t=m\right\}\in\left\{0,1\right\}$$

(Z) $$\lambda_t\left(m\right)\equiv\Pr\left\{\hat{S}_t=m;Y\right\}=\Pr\left\{\hat{S}_t=m;Y_0^t\right\}\cdot\Pr\left\{Y_{t+1}^{\tau+1}\mid\hat{S}_t=m\right\}$$

(Z) $$\hat{R}\left(Y_t\mid\hat{X}\right)=\hat{R}\left(Y_t\mid\hat{X}_t=\hat{X}\right)=\Pr\left\{Y_t\mid\hat{X}_t=\hat{X}\right\}$$

(Z) $$\hat{R}\left(Y_t\mid\hat{X}_t\right)=\prod_{j=1}^{n_0}\hat{r}\left(y_t^j\mid\hat{x}_t^j\right)$$

(Z) $$\left\{\hat{X}_t=\hat{X};\hat{S}_{t-1}=\widehat{m};\hat{S}_t=\breve{m}\right\}\Leftrightarrow\left\{\hat{X}_t=\hat{X}\right\}$$

$$\gamma_t\left(\overset{\frown}{m}, \breve{m}\right) \equiv \Pr\left\{\hat{S}_t = \breve{m}; Y_t \mid \hat{S}_{t-1} = \overset{\frown}{m}\right\}$$

$$= \sum_{\hat{X}} \left[\begin{array}{c} \Pr\left\{\hat{S}_t = \breve{m} \mid \hat{S}_{t-1} = \overset{\frown}{m}\right\} \cdot \Pr\left\{\hat{X}_t = \hat{X} \mid \hat{S}_{t-1} = \overset{\frown}{m}, \hat{S}_t = \breve{m}\right\} \\ \cdot \Pr\left\{Y_t \mid \hat{X}_t = \hat{X}; \hat{S}_{t-1} = \overset{\frown}{m}\ ; \hat{S}_t = \breve{m}\right\} \end{array} \right]$$

(*Z*)

$$= \sum_{\hat{X}} p_t\left(\breve{m} \mid \overset{\frown}{m}\right) \cdot q_t\left(\hat{X}_t = \hat{X} \mid \overset{\frown}{m}, \breve{m}\right) \cdot \hat{R}\left(Y_t \mid \hat{X}\right)$$

$$\gamma_t\left(\breve{m}, \overset{\frown}{m}\right) = \gamma_t\left(\overset{\frown}{m}, \breve{m}\right)$$

(*Z*)
$$\alpha_t\left(m\right) \equiv \Pr\left\{\hat{S}_t = m; Y_0^t\right\} = \alpha_t\left(\breve{m}\right) = \sum_{\overset{\frown}{m}} \alpha_{t-1}\left(\overset{\frown}{m}\right) \gamma_t\left(\overset{\frown}{m}, \breve{m}\right)$$

(*Z*)
$$\beta_t\left(m\right) \equiv \Pr\left\{Y_{t+1}^{\tau+1} \mid \hat{S}_t = m\right\} = \beta_t\left(\breve{m}\right) = \sum_{\overset{\frown}{m}} \beta_{t+1}\left(\overset{\frown}{m}\right) \gamma_{t+1}\left(\breve{m}, \overset{\frown}{m}\right)$$

$$\lambda_t\left(m\right) = \alpha_t\left(m\right) \cdot \beta_t\left(m\right)$$

(*Z*)
$$\left\{\hat{S}_{t-1} = \overset{\frown}{m}; Y_0^{t-1}\right\} \bigcup \left\{\hat{S}_t = \breve{m}; Y_t \mid \hat{S}_{t-1} = \overset{\frown}{m}\right\} \bigcup \left\{Y_{t+1}^{\tau+1} \mid \hat{S}_t = \breve{m}\right\}$$
$$\Leftrightarrow \left\{\hat{S}_{t-1} = \overset{\frown}{m}; \hat{S}_t = \breve{m}; Y\right\}$$

(*Z*)
$$\sigma_t\left(m', m\right) \equiv \Pr\left\{\hat{S}_{t-1} = \overset{\frown}{m}; \hat{S}_t = \breve{m}; Y\right\} = \alpha_{t-1}\left(\overset{\frown}{m}\right) \cdot \gamma_t\left(\overset{\frown}{m}, \breve{m}\right) \cdot \beta_t\left(\breve{m}\right)$$

$$\alpha_0\left(0\right) = 1\ ;\ \ \alpha_0\left(m\right) = 0\ ,\ m \neq 0$$

$$\beta_{\tau+1}\left(0\right) = 1\ ;\ \ \beta_{\tau+1}\left(m\right) = 0\ ,\ m \neq 0$$

5 Observations on the Algorithm

It is ironic that the objective of the algorithm and its formal statement, being the two probabilities $\lambda_t\left(m\right)$ and $\sigma_t\left(m', m\right)$, involve only a forward state transition from *m′* to *m*, and not a backward state transition from $\grave{m}$ to *m*, as the essense of the algorithm is a *combined* forward and backward computation. The two directions are collapsed into one direction, consistently by way of definitions, as

$$\sigma_t\left(m', m\right) = \alpha_{t-1}\left(\overset{\frown}{m}\right) \cdot \gamma_t\left(\overset{\frown}{m}, \breve{m}\right) \cdot \beta_t\left(\breve{m}\right)$$

where the definition of $\sigma_t\left(m', m\right)$ in the algorithm is now an assignment.

It's for the above reasons that the terms $\{m', m\}$ are retained at the beginning and end, and the term $\{\hat{m}, \breve{m}\}$ for intermediate computations to avoid confusion between the *m′* terms in the $\alpha_t(\hat{m})$ and $\beta_t(\breve{m})$ recursive expressions, where the paper had confusingly used *m′* as both a preceding variable and a dummy variable of reverse meaning of order.

The values of $\gamma_t(\hat{m}, \breve{m})$, need to be computed in but one direction for the $\alpha_t(\hat{m})$, retained in a memory, then applied again, without recomputing, in the opposite direction for the $\beta_t(\breve{m})$. BCJR suggest it may be more efficient to compute γ_t in each direction, perhaps because fast memory was so costly in 1974. Today, it may be better to store and retrieve the γ_t unless data rates are so high that dedicated hardware is used to perform the algorithm, in which case, why not reuse it unless it is in continuous use, e.g. pipelining from block to block? It follows from this and the availability of each successive $\alpha_{t-1}(\hat{m}), \alpha_t(\hat{m}), \beta_t(\breve{m})$ and $\gamma_t(\hat{m}, \breve{m})$ tuple on the backward return from the $\beta_{\tau+1}(0) = 1$ boundary condition, that the final result, whether it be the state transition soft decision or an information decision, can be output at each trellis depth, without waiting for the backward calculation to complete to $\beta_1(\breve{m})$.

5 Decoding

To make decisions, the BCJR paper, once finished with the APP algorithm, proposes, for convolution codes, a comparative test of the total probability of a putative "test" decision, 0 in their case, with ½ at each "time-depth" in the trellis. This is analogous to the likelihood ratio test in Viterbi ML decoding. So it could be the case (here, but suggestively) that BCJR decoding using this comparative test has no decision reliability advantage over a Viterbi forward-backward algorithm! Such foretelling might even be hinted by BCJR when discussing relative decoding performance of the two algorithms, which they suggest may be about equal for codewords.

[Nevertheless, the BCJR state and transition APPs at each code symbol remain optimal, an advantage to BCJR if it can be efficiently exploited, as it would later be in decoding procedures involving belief propagation, iteration, very fast processors and large memories.]

The suggestion that BCJR decoding, with its "½ test," implementable or not, may not be MAP rests on ½ being the correct, call it best, comparison threshold. If ½ is the best threshold at that time, and independent of time as it doesn't change, then the scheme retains MAP status for decisions. However, if ½ is somehow not best for all times, then decoding by the BCJR algorithm plus this decision scheme is no longer MAP. How might this happen? It won't happen because of a channel being worse than average over short intervals, because this doesn't change channel *symmetry*. It might happen for non-scrambled information with anomalously long runs of 0s or 1s, but this requires further investigation.

Regardless, here follows the decision scheme proposed by BCJR. For the arbitrary choice of putative information bit 0 (versus 1) at time *t*, the decision criterion is

$$\Pr\left\{\hat{i}_t^{(j)} = 0; Z \mid Y; Z\right\} > 1/2\,,\ \ \hat{i}_t^{(j)} = 0\,;\ \ \text{else},\ \hat{i}_t^{(j)} = 1$$

where $\hat{i}_t^{(j)}$ is the decision, "looking back" from 0 up to ν−1 code symbols (and the information underlying them) earlier, from *t*. [Presuming the known event *Z* allows eliminating all *Z*s above, the criterion development will look the same as in the paper. However, *Z* plays a crucial role in the decoding scheme.]

This looking back is reminiscent of trace back in Viterbi decoding. Of course, the look-back "reach" in this case may be merely by one symbol, radically different from Viterbi trace back. This also raises a separate issue, not addressed by the paper, about *multiple* potential decisions about the same code symbol as the trellis is traversed by the scheme. What if the same time-depth, which appears in *ν*–1 adjacent depths, can end up containing different decisions from different look backs at near times? Does this scheme need to be elaborated with some new sub-rule, like taking the oldest state bit *j* at a depth, or perhaps by majority rule over multiple depths? These questions are left unanswered.

The conditional probability for $\hat{i}_t^{(j)}$ can be expressed as two joint probabilities.

$$\Pr\left\{\hat{i}_t^{(j)} = 0; Z \mid Y; Z\right\} = \Pr\left\{\hat{i}_t^{(j)} = 0\ ; Y; Z\right\} / \Pr\left\{Y; Z\right\}$$

Z ensures that events $\left\{\hat{S}_\tau = m; Y; Z\right\} = \left\{\hat{S}_\tau = 0; Y; Z\right\} = \left\{Y; Z\right\}$, so by definition $\lambda_\tau\left(0\right) = \Pr\left\{\hat{S}_\tau = 0; Y; Z\right\} = \Pr\left\{Y; Z\right\}$ as $\Pr\left\{\hat{S}_\tau = 0\right\} = 1$ at this depth.

The probability of event $\left\{\hat{i}_t^{(j)} = 0; Y; Z\right\}$ on the other hand can occur at any *t*, and expresses the total probability of event $\left\{\hat{i}_t^{(j)} = 0; Y; Z\right\} = \left\{\hat{i}_t^{(j)} = 0; Y\right\}$ anywhere other than at the boundaries.

The decision scheme must also obtain $\Pr\left\{\hat{i}_t^{(j)} = 0\ ; Y\ ; Z\right\}$. BCJR observe that

$$\Pr\left\{\hat{i}_t^{(j)} = 0 ; Y ; Z\right\} = \sum_{s_t^{(j)} \in A_t^{(j)}} \lambda_t(m) ; \;\; A_t^{(j)} \in \left\{\hat{S}_t : \hat{s}_t^{(j)} = 0\right\},$$
$$j = (1, \ldots, k_0) = (1, \ldots, \nu - 1)$$

and $\hat{s}_t^{(j)} = \hat{i}_t^{(j)}$ because the contents of a state are just information symbols (bits). This is attractive since all the $\lambda_t(m)$ are already known. It is just a regular procedure to pick the applicable states m and sum them.

Putting the two together, noting critical difference between t in general and τ in particular

$$\Pr\left\{\hat{i}_t^{(j)} = 0 ; Z \mid Y ; Z\right\} = \frac{1}{\lambda_\tau(0)} \sum_{s_t^{(j)} \in A_t^{(j)}} \lambda_t(m) \quad .$$

The ½ test is applied and a decision is made. It seems almost trivial until the question is asked, how is the value of $\lambda_\tau(0)$ obtained? BCJR don't suggest how to do it, but there is an obvious approach.

It turns out that $\lambda_\tau(0)$ is problematic, depending on the application. Consider how it might be obtained as

$$\lambda_\tau(0) = \alpha_\tau(0) \cdot \beta_\tau(0) = \alpha_\tau(0) = \Pr\left\{\hat{S}_\tau = 0 ; Y_0^\tau ; Z\right\} = \sum_{m'} \alpha_{\tau-1}(m') \gamma_\tau(m', 0) .$$

Rewriting for the boundary conditions affecting time τ

$\lambda_\tau(0) = \sum_B \alpha_{\tau-1}(m') \gamma_\tau(m', 0)$, $B \Leftrightarrow m' : s_{\tau-1}^{k_0(\nu-1)} \in A_{\tau-1}^{k_0(\nu-1)}$, A as previously defined.

All the terms in the sum are available as previous computations. The $\alpha_{\tau-1}(m')$ and $\gamma_\tau(m', 0)$ are already stored for all states m′ , and possibly also for transitions (m', 0), the subject of a previous observation. $\lambda_\tau(0)$ is readily computable, and need be done only once according to the decoding scheme. That's the good part. The problematic part, simply put, is that one short segment of the channel and trellis, embodied in $\alpha_{\tau-1}(m')$ and literally present in $\gamma_\tau(m', 0)$ is being used to normalize probabilities across the rest of an arbitrarily long block of received symbols. On average, across many blocks, the normalization factor $\lambda_\tau(0)$ will approach its expected value, but not for one block alone. In a sense, this is a "turbo"-like effect on the decoder's original estimate of the channel embodied in

$\hat{R}\left(Y_t ; Z \mid \hat{X}_t\right)$. This suggests that “BCJR decoding” of ordinary convolutional codes should be applied to many, relatively short blocks, instead of fewer, relatively long blocks, running an average over individual samples of $\lambda_\tau(0)$.

This finally raises a question about APP estimation in general. The BCJR, and presumably any APP algorithm, always depend on knowledge about the channel. The channel model in the reference diagram is not the channel! It is a guess, a specified operating point, a calculation based on observation of known data, or perhaps a convergence to an acceptable output error rate. There are many ways to populate the model with numbers. This is true for any convolutional decoder. It is not true for hard decision algebraic block decoders seeking the minimum distance between a received word and all codewords. The reliance on some estimate of channel SNR, particularly E_{cb}/N_0 for code bits, or worse, analog SNRs like C/N_0 which is notoriously difficult to measure, is troubling news for *absolute* probability algorithms like BCJR. Good implementations become system dependent for this reason. But because of this, there remains a demand for communication engineers, long after the algorithms. That’s the good news.