Notions sur les Turbocodes (Mars 2001)

Joël Le Roux

27 mai 2004

Table des matières

1 Introduction

Inventés au début des années 1990 par des enseignants-chercheurs de l'E.N.S.T. Bretagne étudiant l'implémentation d'algorithmes de communication sur des circuits spécialisés (C. Berrou, A. Glavieux and P. Thihimajshima, "Near Shannon limit error-correcting coding and decoding: turbo codes", Proc. Int. Conf. Comm. Geneve 1993, pp 1064-1070; C. Berrou and A. Glavieux, "Near Shannon limit error-correcting coding and decoding: turbo codes", IEEE Trans. Comm., Oct. 1996, pp. 1261-1271), les turbocodes semblent être considérés comme un progrès remarquable par les spécialistes du codage: cette méthode permet de s'approcher de la limite de Shannon pour la correction d'erreurs de transmission tandis que les techniques utilisées jusqu'alors (codes en blocs, codes convolutionnels) présentaient des performances sensiblement inférieures. L'idée originale est d'utiliser deux séquences redondantes, l'une étant calculée à partir du message à transmettre, l'autre étant calculée à partir d'une version permutée de ce message.

Le décodage est probabiliste et utilise en alternance les deux séquences redondantes, l'estimation du message issue du premier décodeur étant utilisée comme information d'entrée par le second et vice-versa.

Un exemple d'utilisation par la NASA pour les échanges de données avec les sondes Galileo et Cassini est présenté sur le site web: "www331.jpl.nasa.gov/ public/ TurboForce.GIF" (en Mars 2000). Ce document indique qu'on peut réduire la probabilité d'erreur à 10^{-5} lorsque le rapport signal à bruit est de l'ordre de 0 dB et même négatif (bruit plus fort que le signal). L'utilisation des turbocodes est envisagée pour la troisième génération de téléphonie mobile, UMTS.

On trouvera sur le site http: //www.turbocodes.fr.st un exemple d'implémentation réalisé par des étudiants de l'ESSI en 2001.

Dans ces notes, il y a deux parties, la première détaille la justification de l'algorithme de décodage de Bahl, Cocke, Jelinek et Raviv et simplifiant la présentation de l'expression de l'algorithme aux dépends de la compacité de l'implémentation. La seconde décrit la manière dont cet algorithme est utilisé dans les turbocodes. J'ai essentiellement utilisé l'article de W. E. Ryan de la New Mexico State University, "A turbo code tutorial" qu'on peut trouver sur différents sites du web

2 L'algorihtme de Bahl, Cocke, Jelinek et Raviv dans un cas simplifié

Cet algorithme (L. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", IEEE trans. Inf. Theory, March 1974, pp. 284-287) calcule la probabilité des valeurs possibles du message émis à l'instant t lorsque les données (éventuellement erronées) ont été reçues. Les simplification apparaissant dans les développements sont dues aux propriétés des sources markoviennes. Il s'agit dans ce cadre de calculer la probabilité qu'un automate soit dans un état donné et la probabilité de transition d'un état au suivant lorsque les mesures des sorties produites par cet automate sont entachées d'erreurs. Pour faciliter la compréhension de l'algorithme, il est utile de posséder les notions de représentations en treillis de l'évolution des états d'un graphe au cours du temps telles qu'elles sont utilisées dans le décodage de Viterbi des codes convolutionnels.

2.1 Notations

Soit un automate à M états.

A l'instant t, l'automate passe de l'état s_{t-1} qui était par exemple m' à l'état s_t, soit m. (L'état initial à $t = 0$ est connu ainsi que l'état final pour $t = T$).

L'entrée de l'automate est u_t qui peut prendre uniquement les valeurs $k = 1$ et $k = -1$. Sous l'effet de l'entrée u_t, l'automate qui était dans l'état s_{t-1} passe à l'état s_t et émet la donnée x_t. Cette donnée est transmise et le récepteur en reçoit une version éventuellement erronée y_t (fig. 1).

On connait les probabilités de transition d'un état au suivant

$$p(s_t = m/s_{t-1} = m'). \tag{1}$$

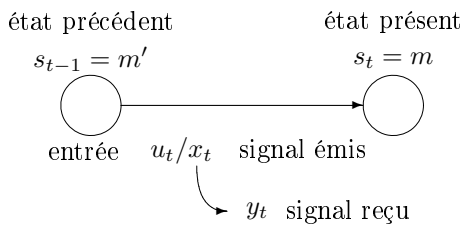

Figure 1: Eléments d'une transition d'un automate dans un modèle de Markov

Dans le cas simplifié des transmissions numériques que nous considérons, nous supposerons que u_t est binaire et que ces probabilités seront *a priori* soit 0, soit $1/2$: quand l'automate est dans l'état m' il ne peut passer que dans deux états m donnés par les contraintes des transitions ; la probabilité de passer dans l'un ou l'autre des deux états possibles est égale à $1/2$. Du fait de la mesure de y_t, ces probabilités *a priori* sont modifiées. Le but de l'algorithme est d'obtenir une évaluation de ces probabilités modifiées par la connaissance des données mesurées $y_1, \ldots, y_T$.

La sortie x_t engendrée lorsque l'automate passe de l'état m' à l'état m est certaine; sa probabilité égale à un (l'entrée u_t est connue).

On émet T données x_t $(t = 1, \ldots, T)$.

Lors de la transmission de x_t, la donnée reçue, y_t est une version de x_t qui peut être erronée suivant une probabilité

$$p(y_t = x_t) = 1 - \varepsilon, \tag{2}$$

$$p(y_t \neq x_t) = \varepsilon. \tag{3}$$

La probabilité d'erreur ε est connue du récepteur.

Le récepteur a mesuré une séquence $[y_1, \ldots, y_T]$. On cherche les probabilités pour que l'automate d'émission se soit trouvé à l'instant t dans l'état m :

$$p\Big(s_t = m/[y_1, \ldots, y_T]\Big) = \frac{p\Big(s_t = m, [y_1, \ldots, y_T]\Big)}{p([y_1, \ldots, y_T])}, \tag{4}$$

que nous écrirons

$$p\Big(s_t = m/[y_1, \ldots, y_T]\Big) = \frac{\lambda_t(m)}{p([y_1, \ldots, y_T])}, \tag{5}$$

avec

$$\boxed{\lambda_t(m) = p\Big(s_t = m, [y_1, \ldots, y_T]\Big).} \tag{6}$$

On calcule aussi la probabilité qu'on ait l'entrée $u_t = k$ à l'instant t lorsque l'état à l'instant $t-1$ était m'.

$$p\Big((s_{t-1} = m', s_t = m)/[y_1, \ldots, y_T]\Big) = \frac{p\Big(s_{t-1} = m', s_t = m, [y_1, \ldots, y_T]\Big)}{p([y_1, \ldots, y_T])}, \tag{7}$$

que nous noterons

$$p\Big((s_{t-1} = m', s_t = m)/[y_1, \ldots, y_T]\Big) = \frac{\sigma_t(m', m)}{p([y_1, \ldots, y_T])}, \tag{8}$$

avec

$$\boxed{\sigma_t(m', m) = p\Big(s_{t-1} = m', s_t = m, [y_1, \ldots, y_T]\Big).} \tag{9}$$

Dans la suite des traitements fondés sur l'analyse de ces probabilités, nous ne tiendrons pas compte des dénominateurs. En effet, lorsque la séquence $[y_1, \ldots, y_T]$ est mesurée, cette probabilité est constante et n'apparait que comme un facteur de normalisation.

La propriété fondamentale des sources markoviennes (l'évolution après l'instant t ne dépend *que* de l'état de l'automate à l'instant t et des entrées futures, en particulier elle ne dépend pas de

la séquence qui a abouti à l'état où se trouve l'automate à l'instant t) permet d'écrire $\lambda_t(m)$ et $\sigma_t(m', m)$ sous la forme de produit de trois probabilités :

$$\boxed{\alpha_t(m) = p(s_t = m, [y_1, \ldots, y_t]),} \tag{10}$$

$$\boxed{\beta_t(m) = p([y_{t+1}, \ldots, y_T]/s_t = m),} \tag{11}$$

$$\boxed{\gamma_t(m', m) = p\Big((s_t = m, y_t)/s_{t-1} = m'\Big).} \tag{12}$$

2.2 Justification des calculs

La justification est fondée sur quelques règles élémentaires : la factorisation d'une probabilité en fonction d'une décomposition judicieuse en différents cas, la règle d'écriture des probabilités conditionnelles et la propriété des modèles de Markov.

2.2.1 Expression de $\lambda_t(m)$

En utilisant la définition des probabilités conditionnelles

$$p(a/b) = \frac{p(a, b)}{p(b)}, \tag{13}$$

on peut écrire, en scindant $[y_1, \ldots, y_T]$ et en utilisant la définition des probabilités conditionnelles

$$\begin{aligned}
\lambda_t(m) &= p\Big(s_t = m, [y_1, \ldots, y_T]\Big), && (14)\\
&= p\Big(s_t = m, [y_1, \ldots, y_t], [y_{t+1}, \ldots, y_T]\Big), && (15)\\
&= p\Big([y_{t+1}, \ldots, y_T]/(s_t = m, [y_1, \ldots, y_t])\Big) p\Big(s_t = m, [y_1, \ldots, y_t]\Big), && (16)\\
&= p\Big([y_{t+1}, \ldots, y_T]/(s_t = m, [y_1, \ldots, y_t])\Big) \alpha_t(m). && (17)
\end{aligned}$$

D'après la propriété des sources markoviennes

$$p\Big([y_{t+1}, \ldots, y_T]/(s_t = m, [y_1, \ldots, y_t])\Big) = p\Big([y_{t+1}, \ldots, y_T]/s_t = m, \Big) = \beta_t(m), \tag{18}$$

et $\lambda_t(m)$ s'écrit

$$\boxed{\lambda_t(m) = \beta_t(m)\alpha_t(m).} \tag{19}$$

2.2.2 Expression de $\sigma_t(m', m)$

On peut aussi écrire, en scindant $[y_1, \ldots, y_T]$ et utilisant ici encore la formule des probabilités conditionnelles

$$\begin{aligned}
&\sigma_t(m', m) = p\Big(s_{t-1} = m', s_t = m, [y_1, \ldots, y_T]\Big), && (20)\\
&\sigma_t(m', m) = p\Big(s_{t-1} = m', [y_1, \ldots, y_{t-1}], s_t = m, [y_t, \ldots, y_T]\Big), && (21)\\
= \;& p(s_{t-1} = m', [y_1, \ldots, y_{t-1}]) p\Big((s_t = m, [y_t, \ldots, y_T])/(s_{t-1} = m', [y_1, \ldots, y_{t-1}])\Big), && (22)\\
= \;& \alpha_{t-1}(m') p\Big((s_t = m, [y_t, \ldots, y_T])/(s_{t-1} = m', [y_1, \ldots, y_{t-1}])\Big). && (23)
\end{aligned}$$

Dans le deuxième facteur de l'éq. (23), la condition

$$s_{t-1} = m', [y_1, \ldots, y_{t-1}], \tag{24}$$

se réduit à la condition

$$s_{t-1} = m', \tag{25}$$

du fait de la propriété des sources markoviennes:

$$\sigma_t(m',m) \quad = \alpha_{t-1}(m')p\Big((s_t=m,[y_t,\ldots,y_T])/s_{t-1}=m'\Big). \tag{26}$$

On peut écrire le deuxième facteur de l'éq. (26) en scindant $[y_t,\ldots,y_T]$ et en utilisant la formule des probabilités conditionnelles

$$p\Big((s_t=m,[y_t,\ldots,y_T])/s_{t-1}=m')\Big) = p\Big((s_t=m,y_t,[y_{t+1},\ldots,y_T])/s_{t-1}=m'\Big), \tag{27}$$

$$= \frac{p\Big(s_{t-1}=m',s_t=m,y_t,[y_{t+1},\ldots,y_T]\Big)}{p(s_{t-1}=m')}, \tag{28}$$

$$= \frac{p\Big(s_{t-1}=m',s_t=m,y_t\Big)}{p(s_{t-1}=m')}p\Big([y_{t+1},\ldots,y_T]/(s_{t-1}=m',s_t=m,y_t)\Big), \tag{29}$$

$$= \gamma_t(m',m)p\Big([y_{t+1},\ldots,y_T]/(s_{t-1}=m',s_t=m,y_t)\Big), \tag{30}$$

ou bien, compte tenu encore une fois de la propriété des sources markoviennes, et en utilisant à noveau la formule des probabilités conditionnelles

$$p\Big((s_t=m,[y_t,\ldots,y_T])/s_{t-1}=m'\Big) \quad = \quad \gamma_t(m',m)p\Big([y_{t+1},\ldots,y_T]/s_t=m\Big), \tag{31}$$

$$= \quad \gamma_t(m',m)\beta_t(m). \tag{32}$$

On en déduit que l'éq. (23) s'écrit

$$\sigma_t(m',m) \quad = \alpha_{t-1}(m')p\Big(s_t=m,[y_t,\ldots,y_T]/s_{t-1}=m'\Big), \tag{33}$$

$$\boxed{\sigma_t(m',m)=\alpha_{t-1}(m')\gamma_t(m',m)\beta_t(m).} \tag{34}$$

La probabilité de transition de l'état m' à l'état m est ainsi le produit de trois facteurs, le premier dépendant du passé, le dernier du futur et le facteur central de la transition à l'instant t.

2.2.3 Calcul récursif de $\alpha_t(m)$

Par définition

$$\alpha_t(m) = p(s_t=m,[y_1,\ldots,y_t]), \tag{35}$$

qu'on peut décomposer en somme de probabilités d'événements indépendants

$$\alpha_t(m) = \sum_{m'} p(s_{t-1}=m',s_t=m,[y_1,\ldots,y_t]). \tag{36}$$

La sommation se fait sur les deux états m' qui permettent d'arriver à l'état m en appliquant l'entrée soit -1 soit 1; elle est justifiée parce que ces deux événements sont disjoints (fig. 2). En

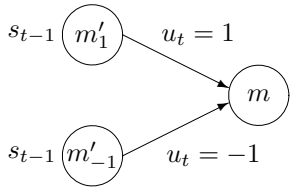

Figure 2: Les états pouvant mener à l'état m

scindant $[y_1, \ldots, y_{t-1}]$ et en utilisant la formule des probabilités conditionnelles

$$\alpha_t(m) = \sum_{m'} p(s_{t-1} = m', [y_1, \ldots, y_{t-1}], s_t = m, y_t); \tag{37}$$

$$= \sum_{m'} p(s_{t-1} = m', [y_1, \ldots, y_{t-1}]) p\Big((s_t = m, y_t)/(s_{t-1}, [y_1, \ldots, y_{t-1}])\Big), \tag{38}$$

qu'on peut simplier du fait de la propriété des sources markoviennes

$$\alpha_t(m) = \sum_{m'} p(s_{t-1} = m', [y_1, \ldots, y_{t-1}]) p\Big((s_t = m, y_t)/s_{t-1} = m'\Big), \tag{39}$$

$$\tag{40}$$

ou encore

$$\boxed{\alpha_t(m) = \sum_{m'} \alpha_{t-1}(m') \gamma_t(m', m).} \tag{41}$$

Initialisations : $\alpha_0(0) = 1$; $m \neq 0$: $\alpha_0(m) = 0$ (L'état initial est connu du récepteur).

2.2.4 Calcul récursif de $\beta_t(m)$

Par définition

$$\beta_t(m) = p([y_{t+1}, \ldots, y_T]/s_t = m), \tag{42}$$

qu'on réécrit en utilisant la formule des probabilités conditionnelles

$$\beta_t(m) = \frac{p(s_t = m, [y_{t+1}, \ldots, y_T])}{p(s_t = m)}. \tag{43}$$

On décompose le numérateur de $\beta_t(m)$ en somme

$$\beta_t(m) = \frac{\sum_{m''} p(s_t = m, s_{t+1} = m'', [y_{t+1}, \ldots, y_T])}{p(s_t = m)}. \tag{44}$$

La sommation se fait sur les deux états m'' qui permettent de remonter à l'état m lorsque l'entrée u_{t+1} vaut soit -1 soit 1 (fig. 3); elle est justifiée parce que ces deux événements sont disjoints.

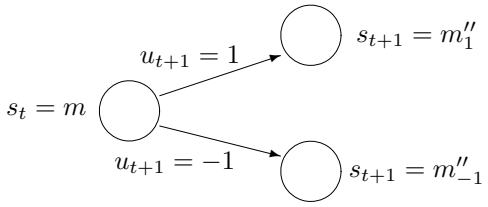

Figure 3: Les états issus de l'état m

En scindant $[y_{t+1}, \ldots, y_T]$ et en utilisant la formule des probabilités conditionnelles

$$\beta_t(m) = \frac{\sum_{m''} p(s_t = m, y_{t+1}, s_{t+1} = m'', [y_{t+2}, \ldots, y_T])}{p(s_t = m)}, \tag{45}$$

$$\beta_t(m) = \frac{\sum_{m''} p(s_t = m, y_{t+1}, s_{t+1} = m'') p\Big([y_{t+2}, \ldots, y_T]/(s_t = m, y_{t+1}, s_{t+1} = m'')\Big)}{p(s_t = m)}. \tag{46}$$

qu'on peut simplier du fait de la propriété des sources markoviennes, puis réécrire en utilisant la formule des probabilités conditionnelles

$$\beta_t(m) = \frac{\sum_{m''} p(s_t = m, y_{t+1}, s_{t+1} = m'')p([y_{t+2}, \ldots, y_T]/s_{t+1} = m'')}{p(s_t = m)}, \tag{47}$$

$$\beta_t(m) = \frac{\sum_{m''} p(s_t = m, y_{t+1}, s_{t+1} = m'')\beta_{t+1}(m'')}{p(s_t = m)}, \tag{48}$$

$$\beta_t(m) = \sum_{m''} \frac{p(s_t = m, y_{t+1}, s_{t+1} = m'')}{p(s_t = m)}\beta_{t+1}(m''), \tag{49}$$

$$\beta_t(m) = \sum_{m''} p\Big((y_{t+1}, s_{t+1} = m'')/s_t = m\Big)\beta_{t+1}(m''), \tag{50}$$

soit

$$\boxed{\beta_t(m) = \sum_{m''} \gamma_{t+1}(m, m'')\beta_{t+1}(m'').} \tag{51}$$

Initialisations : $\beta_T(0) = 1$; $m \neq 0 : \beta_T(m) = 0$ (état final connu).

2.2.5 Interprétation de $\gamma_t(m, m')$ en termes de probabilité d'erreur de transmission

$$\gamma_t(m', m) = p\Big((s_t = m, y_t)/s_{t-1} = m'\Big), \tag{52}$$

est la probabilité d'avoir l'une ou l'autre des entrées u_t et la sortie y_t lorsque l'automate est dans l'état $s_{t-1} = m'$. Cette probabilité peut s'écrire en fonction de de la probabilité de la valeur de l'entrée u_t et de la probabilité d'erreur de transmission ε. Il est utile de bien détailler les différents cas (fig. 4.)

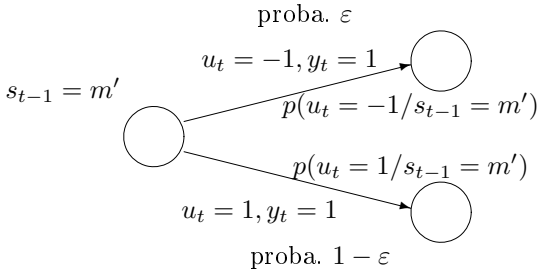

Figure 4: Illustration du calcul de $\gamma_t(m', m)$

Par hypothèse, u_t ne dépend pas de s_{t-1}. Si on suppose que l'entrée qui fait passer de l'état m' à l'état m est $u_t = 1$:

$$\gamma_t(m', m) = p\Big((s_t = m, y_t)/s_{t-1} = m'\Big), \tag{53}$$

qu'on réécrit en interprétant en fonction de l'éntrée u_t et en utilisant la formule des probabilités conditionnelles

$$\gamma_t(m', m) = p(u_t = 1, y_t) = p(u_t = 1)p(y_t/u_t = 1), \tag{54}$$

$$\boxed{\gamma_t(m', m) = p(u_t = 1)p(y_t/u_t = 1) = p(u_t = 1)(1 - \varepsilon),} \tag{55}$$

si $y_t = 1$. De même,

$$\boxed{\gamma_t(m', m) = p(u_t = 1)p(y_t/u_t = 1) = p(u_t = 1)\varepsilon,} \tag{56}$$

si $y_t = -1$.

Le calcul de $\gamma_t(m', m)$ doit être fait pour toutes les transitions possibles et dépend de la valeur *a priori* de $p(u_t = \pm 1)$.

2.2.6 Calcul de $p(u_t = 1, [y_1, \ldots, y_T])$ et de $p(u_t = -1, [y_1, \ldots, y_T])$

La probabilité que u_t soit égale à 1 se déduit de

$$\sigma_t(m', m) = p\Big(s_{t-1} = m', s_t = m, [y_1, \ldots, y_T]\Big). \tag{57}$$

C'est la somme pour toutes les transitions $(m' \to m)$ correspondant à l'application d'une entrée $u_t = 1$

$$\boxed{p(u_t = 1) = \sum_{m'} \sigma_t(m', m).\delta_1(m', m),} \tag{58}$$

où $\delta_1(m', m)$ vaut 1 si il existe une transition $(m' \to m)$ lorsque $u = 1$ et vaut 0 dans tous les autres cas.

De même la probabilité que u_t soit égale à -1 se déduit de

$$\sigma_t(m', m) = p\Big(s_{t-1} = m', s_t = m, [y_1, \ldots, y_T]\Big). \tag{59}$$

C'est la somme pour toutes les transitions $(m' \to m)$ correspondant à une entrée $u_t = -1$

$$\boxed{p(u_t = -1) = \sum_{m'} \sigma_t(m', m).\delta_{-1}(m', m),} \tag{60}$$

où $\delta_{-1}(m', m)$ vaut 1 si il existe une transition $(m' \to m)$ lorsque $u = -1$ et vaut 0 dans tous les autres cas.

Ce sont ces probabilités *a priori*, $p(u_t = 1)$ et $p(u_t = -1)$ qui seront transmises d'un décodeur à l'autre dans le cas des turbocodes.

2.3 Résumé des opérations à effectuer

• Calcul des probabilités de transitions en fonction de la probabilité *a priori* de u_t :

$$\gamma_t(m', m) = p(u_t = 1)p(y_t/u_t = 1) = p(u_t = 1)(1 - \varepsilon), \tag{61}$$

si $y_t = 1$;

$$\gamma_t(m', m) = p(u_t = 1)p(y_t/u_t = 1) = p(u_t = 1)\varepsilon, \tag{62}$$

si $y_t = -1$,

et calculs correspondants pour $u_t = -1$.

• Récurrence (progression en temps croissant) sur α :

Initialisations : $\alpha_0(0) = 1$; $m \neq 0 : \alpha_0(m) = 0$

$$\alpha_t(m) = \sum_{m'} \alpha_{t-1}(m')\gamma_t(m', m). \tag{63}$$

• Récurrence (rétrogression en temps décroissant) sur β :

Initialisations : $\beta_T(0) = 1$; $m \neq 0$: $\beta_T(m) = 0$,

$$\beta_t(m) = \sum_{m''} \gamma_{t+1}(m, m'')\beta_{t+1}(m''). \tag{64}$$

• Calcul de la probabilité des états :

$$\lambda_t(m) = \beta_t(m)\alpha_t(m). \tag{65}$$

• Calcul de la probabilité d'une transition :

$$\sigma_t(m', m) = \alpha_{t-1}(m')\gamma_t(m', m)\beta_t(m). \tag{66}$$

• Réactualisation de la probabité de u_t :

$$p(u_t = 1) = \sum_{m'} \sigma_t(m', m).\delta_1(m', m), \tag{67}$$

où $\delta_1(m', m)$ vaut 1 si il existe une transition $(m' \to m)$ lorsque $u = 1$ et vaut 0 dans tous les autres cas ;

$$p(u_t = -1) = \sum_{m'} \sigma_t(m', m).\delta_{-1}(m', m), \tag{68}$$

où $\delta_{-1}(m', m)$ vaut 1 si il existe une transition $(m' \to m)$ lorsque $u = -1$ et vaut 0 dans tous les autres cas.

Remarques : Cet algorithme a été développé par des chercheurs d'IBM de l'équipe de F. Jelinek, très réputée aussi pour ses travaux en reconnaissance vocale; d'ailleurs cette forme d'algorithme se retrouve dans les méthodes de reconnaissances fondées sur les modèles de Markov cachés (HMM ou Hidden Markov Models) à la base de nombreuses méthodes actuelles pour la reconnaissance vocale: L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition", Proceedings of the IEEE, vol. 77, no2, Feb. 1989, pp. 257-286; L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition, ", IEEE trans. Inform. Theory, vol. IT-21, pp 404-411, 1975. La référence première (de lecture assez ardue) est dans les deux cas celle de L.E. Baum et T. Petrie "Statistical inference for probabilistic functions of finite state markov chains", Ann. Math. Stat. 37 1554-1563, 1966.

3 Application aux turbocodes

Un "turbo" codeur est la combinaison de deux codeurs convolutionnels simples. En entrée on dispose d'un message à émettre u_t et on génère les données redondantes, $x_t^{(A)}$ et $x_t^{(B)}$ avec deux codeurs simples (ayant un petit nombre d'états). On émet aussi l'information brute u_t (le codeur est systématique). L'originalité du codeur est d'effectuer une opération d'entrelacement (permutation) sur les données avant de les traiter par le deuxième codeur si bien que les erreurs qui ne sont pas corrigées par les premier codeur ne seront en général pas identiques aux erreurs non corrigées par le deuxième codeur.

L'opération de décodage est itérative: on utilise l'entrée estimée par le deuxième décodeur pour améliorer le résultat de l'estimation du premier décodeur (au sens du maximum de vraisemblance *a posteriori*); puis le résultat de ce nouveau décodage est utilisée pour améliorer le décodage par le deuxième décodeur, etc... Cette combinaison donne des résultats remarquablement efficaces (proches de la limite donnée par le deuxième théorème de Shannon). Toutefois la quantité de calculs à effectuer est importante. Le décodage est itératif, ce qui peut peut-être expliquer les excellentes performances de cette approche. La figure 7 donne le schéma du décodeur.

3.1 Le codeur

Chacun des deux codeur est un codeur convolutionnel récursif. Ce choix nécessite certaines précautions car il existe des codeurs ne présentant pas de bonnes propriétés pour être utilisés dans les turbocodes. Notons que les données transmises u_t ne sont transmises qu'une fois (elles ne sont pas transmises après entrelacement). Il est possible de réduire la quantité de données transmises par "poinçonnage", de manière à optimiser la redondance et à l'adapter à la probabilité d'erreur caractérisant la ligne de transmission.

3.2 L'entrelacement

On effectue une permutation sur les données émises u_t. Il semble que les meilleures performances soient obtenues par une permutation aléatoire (mais précalculée). On se reportera à l'ouvrage de C. H. Heegard et S. B. Wicker, "Turbo coding", Kluwer 1999 pour une discussion détaillée de l'intérêt des différents types de permutations. Mais on peut envisager d'autres permutations comme le "bit reversal". Pour construire une permutation alétoires sur une séquence de longueur T, on choisit T variables aléatoires équiréparties soit $a(t)$ pour $t = 0, \ldots, T-1$ et on ordonne ces valeurs en

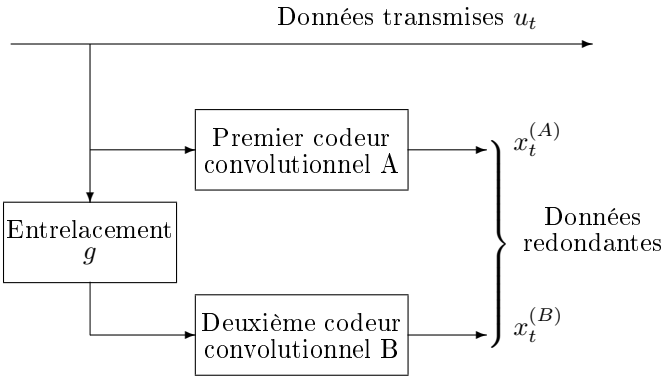

Figure 5: Schéma du codeur des turbocodes : les données redondantes sont générées par un automate similaire à celui des codes convolutionnels; l'originalité consiste à utiliser deux codeurs après avoir appliqué une permutation ou une autre opération inversible sur les données à coder

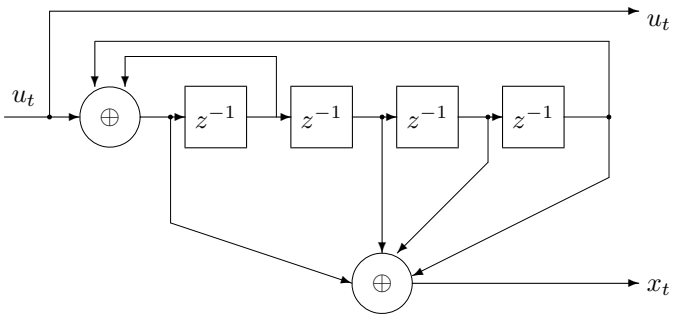

Figure 6: Codeur convolutionnel récursif

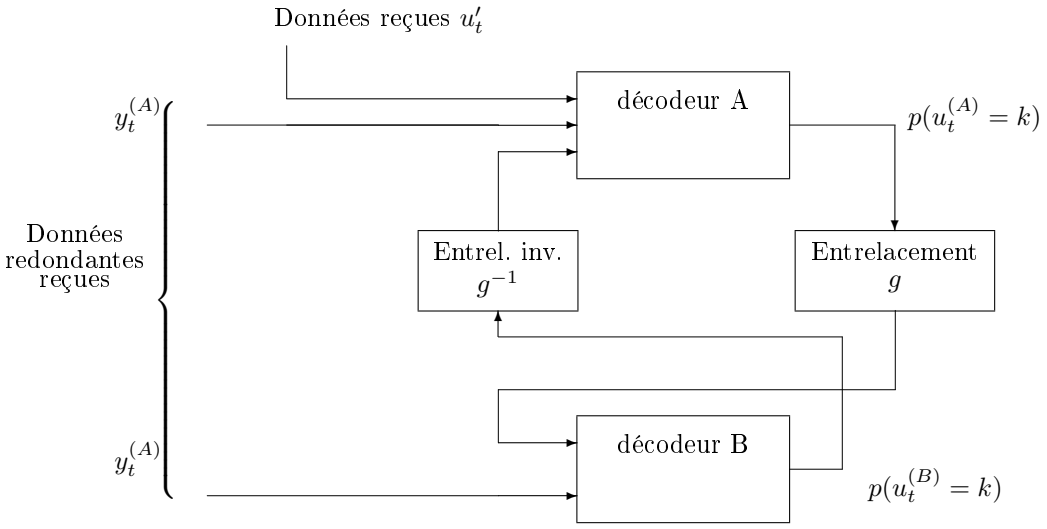

Figure 7: Schéma du turbo décodeur : chacun des deux décodeurs calcule en fonction des données qu'il reçoit la probabilité *a posteriori* qu'une des données ait été émise à l'instant t. Cette nouvelle estimation du message émis est transmise au second décodeur après application de l'opération d'entrelacement ou de l'opération inverse.

s'assurant qu'elles sont toutes différentes les unes des autres. Soit $g'\Big(a(t)\Big)$ la position de $a(t)$ dans cet ordre. Le résultat de la permutation de t sera

$$t' = g(t) = g'\Big(a(t)\Big). \tag{69}$$

On déduit de $g(t)$ la permutation inverse

$$t = g^{-1}(t'). \tag{70}$$

3.3 Le décodeur

Le décodeur doit connaitre la probabilité d'occurence d'une erreur de transmission ε.

Le décodeur se décompose en deux modules identiques qui réalisent la fonction de l'algorithme BCJR du paragraphe précédent. Lors de la première itération, le premier décodeur dispose de deux informations : d'une part la séquence redondante $y_t^{(A)}$ émise par le premier codeur et éventuellement entâchée d'erreurs ; et d'autre part la probabilité que u_t soit égal à $k = \pm 1$. Cette probabilité peut se déduire par exemple du signal reçu u'_t si on suppose comme cela semble courant que le codeur est systématique: Si $u'_t = 1$, on pourra se donner les probabilités

$$p^{(A)}(u_t = 1) = 1 - \varepsilon, \tag{71}$$

$$p^{(A)}(u_t = -1) = \varepsilon. \tag{72}$$

On se donne de même les probabilités lorsque $u'_t = -1$

Connaissant ces probabilités il peut en déduire $\gamma_t(m', m)$ puis $\alpha_t(m)$ et $\beta_t(m)$ et enfin $\sigma_t(m', m)$. De $\sigma_t(m', m)$ il calcule une nouvelle estimation de $p(u_t = k)$, soit $p^{(A+)}(u_t = k)$.

On applique la permutation g à ces probabilités et on obtient la probabilité *a priori* de la séquence d'entrée du deuxième codeur $p^{(B)}(u_t = k)$. Le deuxième codeur qui dispose de la deuxième séquence redondante $y_t^{(A)}$ en déduit $\gamma_t(m', m)$ puis $\alpha_t(m)$ et $\beta_t(m)$ et enfin $\sigma_t(m', m)$. De $\sigma_t(m', m)$ il calcule une nouvelle estimation $p^{(B+)}(u_t = k)$. La permutation g^{-1} est appliquée à cette séquence de probabilités et est transmise au premier codeur qui l'utilise à la place de $p^{(A)}(u_t = k)$.

Le processus est réitéré jusqu'à la convergence de l'algorithme.

3.4 Résumé des opération à effectuer

Le premier codeur (A) dispose de $p^{(A)}(u_t = k)$ et de $y_t = y_t^{(A)}$. Il en déduit

$$\gamma_t(m', m) = p(u_t = 1)p(y_t/u_t = 1) = p(u_t = 1)(1 - \varepsilon), \tag{73}$$

si $y_t = 1$; et

$$\gamma_t(m', m) = p(u_t = 1)p(y_t/u_t = 1) = p(u_t = 1)\varepsilon, \tag{74}$$

si $y_t = -1$.

$$\alpha_t(m) = \sum_{m'} \alpha_{t-1}(m')\gamma_t(m', m), \tag{75}$$

(Initialisations : $\alpha_0(0) = 1$; $m \neq 0$:$\alpha_0(m) = 0$).

$$\beta_t(m) = \sum_{m''} \gamma_{t+1}(m, m'')\beta_{t+1}(m''), \tag{76}$$

(Initialisations : $\beta_T(0) = 1$; $m \neq 0$: $\beta_T(m) = 0$; toutefois l'impossibilité de connaitre en pratique l'état final conduit à choisir un état équiprobable $\beta_T(m) = 1/M$.)

$$\sigma_t(m', m) \quad = \alpha_{t-1}(m')\gamma_t(m', m)\beta_t(m), \tag{77}$$

$$p(u_t = 1) = \sum_{m'} \sigma_t(m', m).\delta_1(m', m), \tag{78}$$

où $\delta_1(m', m)$ vaut 1 si il existe une transition $(m' \rightarrow m)$ lorsque $u = 1$ et vaut 0 dans tous les autres cas.

$$p(u_t = -1) = \sum_{m'} \sigma_t(m', m).\delta_{-1}(m', m), \tag{79}$$

où $\delta_{-1}(m', m)$ vaut 1 si il existe une transition $(m' \rightarrow m)$ lorsque $u = -1$ et vaut 0 dans tous le cas contraire.

En sortie du premier codeur on obtient une nouvelle évaluation de $p^{(A)}(u_t^{(A)} = k)$. On applique la permutation g à chacune de ces probabilités.

$$p^{(B)}(u_t^{(B)} = k) = p^{(A+)}(u_{g(t)}^{(A)} = k). \tag{80}$$

Cette séquence est transmise au second codeur (B) qui effectue les mêmes opérations que A sur les données dont il dispose : $p^{(B)}(u_t = k)$ et $y_t = y_t^{(B)}$; La nouvelle évaluation qui est calculée par B est transformée par g^{-1}

$$p^{(A)}(u_t^{(A)} = k) = p^{(B+)}(u_{g^{-1}(t)}^{(B)} = k). \tag{81}$$

Cette séquence est réitérée jusqu'à obtention d'une convergence.

Comme il s'agit d'une méthode itérative, on peut contrôler la correction apportée aux estimations de probabilité à chaque étape du calcul, soit pour augmenter la vitesse de convergence, soit pour atténuer d'éventuelles oscillations.

Après convergence, on décide de la valeur de u_t en prenant pour $k = \pm 1$ la valeur qui correspond à la valeur la plus probable de u_t.

Remarques : Les versions de C. Berrou et de W. E. Ryan de l'algorithme BCJR incluent des normalisations car les calculs effectués sont des récurrences et des accumulations : ils peuvent donc diverger ou donner des instabilités dans le bouclage. Si on ne souhaite pas compliquer le formalisme des calculs et si on accepte d'augmenter la taille des mémoires, il faut s'assurer après chaque calcul de $\alpha_t(m)$ que

$$\sum_m \alpha_t(m) = 1, \tag{82}$$

et prévoir des normalisation similaires sur les autres variables. C. Berrou et W. E. Ryan évitent de calculer les probabilités $p(u_t = k)$ et ramènent ce calcul à un rapport de vraisemblance

$$V(u_t) = \log \frac{p(u_t = 1)}{p(u_t = -1)}, \tag{83}$$

qui est certainement plus efficace mais qui me parait compliquer la présentation. La présentation de W. E. Ryan est la plus claire de celles que j'ai pu voir sur les turbocodes. Toutefois la fin de sa présentation reste très concentrée. J'espère que dans l'écriture présente, j'ai correctement traduit sa manière de procéder, même si cela se fait au détriment de l'efficacité de l'implémentation... mais mon décodage des articles de Berrou et de Ryan peut très bien être erroné !