Mémoires associatives pour observations floues

Vincent GRIPON[1],Xiaoran JIANG[1] *

[1]Lab-STICC
Télécom Bretagne, Technopôle Brest-Iroise, 29238 Brest, France
vincent.gripon@ens-cachan.org, xiaoran.jiang@telecom-bretagne.eu

Résumé – Les mémoires associatives sont des structures de données permettant de stocker des messages puis de les retrouver à partir d'une fraction de leur contenu. Elles sont utilisées dans de nombreux domaines allant des caches de processeurs [1] aux moteurs de bases de données [2], en passant par les systèmes de détection d'intrusions [3]. Cependant, les techniques existantes ne sont pas adaptées à des entrées floues, dont aucune donnée n'est connue avec précision. Ces données pourraient pourtant ouvrir la voie à des utilisations des mémoires à des plus hauts niveaux, en fouille de données par exemple. Nous montrons comment modifier des mémoires associatives récemment introduites pour traiter ce problème de reconnaissance de messages flous. Nous donnons également les probabilités d'erreur théoriques, ainsi que des illustrations de performance. Lorsque les messages stockés sont uniformément distribués, nous montrons que les mémoires associatives introduites sont capables de stocker puis retrouver des messages floutés avec des taux d'erreur faibles tout en ayant une occupation mémoire presque optimale et une complexité algorithmique réduite par rapport aux systèmes existants.

Abstract – Associative memories are datastructures that enable storing messages and then retrieving them from a fraction of their content. They have been used in numerous applications, ranging from CPU caches [1] to database engines [2] and intrusion detection systems [3]. However, existing techniques are not adapted to blurred inputs, of which none of the data are known precisely. These inputs may arise in some applications though, data mining for example. We demonstrate how to modify recently introduced associative memories in order to deal with the problem of the recognition of blurred messages. We also assess theoretical error probabilities, as well as performance evaluation. As long as the stored messages are uniformly distributed, we show that the introduced associative memories are able to store and then retrieve blurred messages with low error rates, while keeping a near-optimal memory usage and reduced computational complexity compared to existing systems.

1 Introduction

Un exemple simple de mémoire associative est un moteur de recherche pour grille de mots croisés. De tels moteurs proposent, étant donné une fraction des lettres données dans un mot recherché, de retrouver une complétion possible parmi ceux de la langue française.

En pratique, tout le savoir faire du spécialiste des mémoires associatives réside dans le choix de modèles et de paramètres offrant un bon compromis entre complexité algorithmique, taux d'erreur et occupation mémoire. Par exemple, les électroniciens utilisent souvent des *Content Adressable Memories* [4] (*CAMs*), lesquelles comparent simultanément une entrée donnée avec toutes les informations apprises. De tels dispositifs énergivores ne passent pas à l'échelle.

En neuroscience, le modèle le plus réputé est celui de Hopfield [5]. Le réseau de Hopfield, contrairement aux *CAMs*, s'appuie sur la corrélation des messages appris pour les retrouver à l'aide d'un algorithme itératif. Cependant, les performances du modèle sont limitées, et à taux d'erreur fixé leur efficacité mémoire tend vers zéro lorsque le nombre de messages stockés tend vers l'infini. De nombreux autres modèles ont été proposés depuis les travaux de Hopfield, sans pour autant permettre de lever le verrou de l'efficacité mémoire.

Récemment, une nouvelle architecture de mémoire associative a été proposée [6, 7], dont l'efficacité mémoire reste asymptotiquement à un facteur constant de l'optimal pour un taux d'erreur donné et la complexité algorithmique est plus faible que celle du modèle de Hopfield. Elle s'appuie sur une représentation robuste de l'information à l'aide de cliques [1] dans un graphe.

Nous sommes intéressés dans ce document à l'extension de ces mémoires au cas d'entrées floues, c'est à dire imprécises. Pour reprendre l'exemple des mots croisés sus-cité, une entrée floue consisterait en une liste de lettres possibles pour chaque position dans le mot. En base de données, par exemple, ces extensions permettraient de retrouver des informations à partir de critères approximatifs, *e.g.* trouver un utilisateur qui s'est connecté dans les trois dernières minutes et qui a visité l'une des pages web dans une catégorie donnée.

Nous montrons comment ces nouvelles mémoires associatives peuvent s'adapter à ces cas d'utilisation. En particulier, nous montrons qu'à taux d'erreur donné, et pour certains autres paramètres fixés, l'efficacité mémoire reste à un facteur de l'optimal. Les modifications produites étant sans conséquence sur les capacités préalables du modèle, il devient tout aussi bien

*Ce travail a été financé par le projet Européen *ERC Advanced grant* NeuCod.

1. Une clique dans un graphe est un sous-ensemble de nœuds du graphe entièrement interconnectés.

adapté aux entrées floues qu'à celles partiellement effacées.

2 Présentation du modèle

Soit $\mathcal{A}$ un alphabet fini de taille ℓ, que nous identifierons à $[|1;\ell|]$, l'ensemble des entiers compris entre 1 et ℓ, par commodité. Soit c un entier positif non nul et $\mathcal{M}$ un ensemble de M mots de taille c sur $\mathcal{A}$. Afin de proposer une mémoire associative représentant $\mathcal{M}$, nous utilisons un graphe binaire $\mathcal{G} =< \mathcal{V}; \leftrightarrow >$ où $\mathcal{V}$ est l'ensemble des sommets et $\leftrightarrow$ la relation binaire des arêtes entre sommets dans $\mathcal{V}$.

Plus précisément, considérons le graphe multi-parti tel que $\mathcal{V}$ est partitionné en c sous-ensembles de taille ℓ et il n'existe aucune arête entre sommets dans une même partie. Nous indexons alors de 1 à c les différentes parties, et de 1 à ℓ les sommets dans chaque partie. De cette façon, un sommet dans le graphe est déterminé uniquement à l'aide d'un couple (c_0, ℓ_0) où c_0 désigne la partie à laquelle il appartient et ℓ_0 est son index dans cette partie. Par abus de notation, nous confondrons à présent un sommet avec le couple correspondant.

Nous utilisons alors un transport simple depuis les mots de taille c sur $\mathcal{A}$ vers les sous-ensembles de sommets dans le graphe binaire. Ce transport est le suivant :

$$f : \left\{ \begin{array}{rcl} \mathcal{A}^c & \rightarrow & 2^{\mathcal{V}} \\ w = w_1 w_2 \dots w_c & \mapsto & \{(1, w_1); (2, w_2); \dots; (c, w_c)\} \end{array} \right. . \quad (1)$$

Ce transport est parfois nommé "codage disjonctif complet" dans une certaine littérature.

Ce transport est trivialement injectif. Nous définissons donc et étendons son inverse g en introduisant un caractère $\perp \notin \mathcal{A}$, de sorte que :

$$g : \left\{ \begin{array}{rcl} 2^{\mathcal{V}} & \rightarrow & \mathcal{A}^c \\ V & \mapsto & w \text{ tel que} \\ & & w_i = \left\{ \begin{array}{l} j \text{ si } V \cap \{(i,k), 1 \leq k \leq \ell\} = \{(i,j)\} \\ \perp \text{ sinon} \end{array} \right. \end{array} \right. \quad (2)$$

Ainsi, il est équivalent de stocker $\mathcal{M}$ et de stocker $f(\mathcal{M})$ en termes informationnels. Il faut néanmoins ajouter le coup du calcul correspondant, que nous négligeons par la suite.

Afin de stocker $f(\mathcal{M})$, nous modifions les connections du graphe $\mathcal{G}$. Initialement vide, nous procédons élément par élément de la façon suivante : étant donné un message w, nous ajoutons aux connections du graphe celles reliant tous les sommets dans $f(w)$. Si deux cliques font intervenir des connections identiques, celles-ci ne sont pas modifiées davantage ; le graphe reste donc entièrement binaire. Aussi, il est aisé de vérifier que par construction, le graphe reste multi-parti.

Considérons à présent le graphe $\mathcal{G}$ une fois tous les messages de $\mathcal{M}$ stockés. Nous introduisons à présent la notion de flou.

Nous appelons **relation de flou** une relation binaire F sur les sommets dans une même partie, symmétrique et réflexive, i.e. :

$$\forall v, v', (v, v') \in F \Rightarrow (v', v) \in F \wedge \forall v, (v, v) \in F \ , \quad (3)$$

et nous notons $F_v = \{v' \big| (v, v') \in F\}$. Pour plus de simplicité dans les développements suivants, nous considérons que F est b-régulière : $\forall v \in \mathcal{V}, \#(F_v) = b$. Nous appelons **flouter** un élément w de $\mathcal{M}$ l'opération qui consiste à remplacer tous les symboles v de w par un symbole choisi uniformément au hasard parmi les symboles F_v.

Le problème que nous adressons dans ce document est le suivant :

Problème 1 : Étant donné un élément w de $\mathcal{M}$ flouté, retrouver w.

3 Méthodologie

Pour répondre au problème 1, nous utilisons un algorithme itératif de type passage de messages inspiré de celui présenté dans [8]. Notons $\tilde{w}$ la version floutée correspondant à un message w qu'il faut retrouver. L'algorithme est formellement décrit comme l'algorithme 1.

Algorithme 1 : Algorithme pour retrouver un message w à partir d'une version floutée $\tilde{w}$.

début
- Soit $\tilde{\mathcal{V}}' = \{(i,j) \big| 1 \leq i \leq c \wedge j \in F_{\tilde{w}_i}\}$; $\tilde{\mathcal{V}} = \emptyset$.
- **tant que** $\tilde{\mathcal{V}} \neq \tilde{\mathcal{V}}'$ **faire**
  - $\tilde{\mathcal{V}} \leftarrow \tilde{\mathcal{V}}'$; $\tilde{\mathcal{V}}' \leftarrow \emptyset$.
  - **pour** $c_0 \in [|1; c|]$ **faire**
    - Soit $s_{c_0} = 0$.
    - **pour** ℓ_0, *tel que* $(c_0, \ell_0) \in \tilde{\mathcal{V}}$ **faire**
      - Compter e_{l_0} le nombre de parties distinctes dont au moins un sommet dans $\tilde{\mathcal{V}}$ est connecté avec (c_0, ℓ_0) dans $\mathcal{G}$.
      - $s_{c_0} \leftarrow \max(\{e_{\ell_0}; s_{c_0}\})$
    - **fin**
    - **pour** ℓ_0, *tel que* $(c_0, \ell_0) \in \tilde{\mathcal{V}}$ **faire**
      - **si** $e_{\ell_0} = s_{c_0}$ **alors**
        - Ajouter (c_0, ℓ_0) à $\tilde{\mathcal{V}}'$.
      - **fin**
    - **fin**
  - **fin**
- **fin**
- Rendre $\tilde{\mathcal{V}}$.

fin

Le fonctionnement de l'algorithme est comme suit. Au départ se trouve une fenêtre de candidats possibles $F_{\tilde{w}_i}$ qui contient b sommets dans chaque partie i pour $i \in [|1; c|]$. Parmi eux se trouvent les sommets initiaux $\tilde{w}_i$ et ceux cherchés w_i. L'objectif est donc de réduire la fenêtre de candidats possibles jusqu'à ce qu'il n'en reste plus qu'un dans chaque partie $i : w_i$. Pour ce faire, l'algorithme itère le principe suivant : pour chaque sommet est compté le nombre de parties, sauf celle à laquelle il appartient, qui possèdent au moins un sommet dans la fenêtre restante de candidats qui lui soit connecté. Cette opération a une compléxité en $\theta\left(c^2 b^2\right)$. En effet, dans le pire des cas, atteint lorsqu'il n'y a qu'une seule clique dans le graphe, il faut pour

chaque sommet (cb initialement) aller consulter tous les autres nœuds avant de trouver la seule connection qui nous intéresse. Notons que la complexité est indépendante à la cardinalité de l'alphabet $\#(\mathcal{A})$. Une règle de séléction du type *winner-takes-all* est ensuite appliquée : dans chaque partie i, seul le sommet ou les sommets qui ont le compte le plus élevé sont retenus.

Par définition, l'algorithme est donc tel que l'ensemble des nœuds actifs décroît, au sens de l'inclusion ensembliste, avec les itérations. Il s'agit donc d'une décroissance sur un ensemble bien fondé, nécessairement convergente. Une borne supérieure triviale pour le nombre d'itérations est donc $c(b-1)$. Les simulations semblent indiquer cependant qu'un nombre bien plus restreint d'itérations suffit. Nous n'avons cependant pas pour le moment d'outil analytique pour le montrer. Le point fixe de l'algorithme qui définit l'état de convergence ne correspond pas nécessairement à un message, plusieurs sommets pouvant rester actifs dans une même partie. Dans ce cas, il est toutefois certain que les sommets $f(w)$ correspondant au message cherché w sont inclus dans les sommets restant. Ceci est une conséquence directe du fait que le message cherché, inscrit sous la forme d'une clique dans le graphe, assure à ses nœuds de toujours atteindre le compte maximal. En d'autres termes, un échec survient lorsqu'un sommet n'appartenant pas à $f(w)$ atteint et maintien un compte maximal au fil des itérations.

3.1 Performances

Comme pour beaucoup de systèmes itératifs, il est difficile d'estimer la probabilité d'erreur P_e après utilisation de l'algorithme. Du fait des remarques précédentes, nous proposons d'utiliser la probability de succès de l'algorithme après une seule itération comme borne supérieure de P_e.

Supposons que les messages stockés soient générés aléatoirement, indépendamment et uniformément parmi les messages possibles. Nous faisons l'hypothèse que les connections sont independantes dans le graphe pour pouvoir appliquer la loi binomiale. Il s'agit d'une approximation, soutenue par les simulations, qui peut s'interpréter intuitivement comme le fait qu'avec un grand nombre de messages stockés, deux connections prises au hasard dans le graphe ont peu de chances d'avoir été ajoutées en même temps.

La distribution des connections peut alors être vue comme des variables indépendantes de Bernouilli de paramètre d. La valeur de d pour une connection peut être obtenue par des arguments de lois binomiales. En effet, la probabilité d'avoir une connection entre deux sommets (c_1, ℓ_1) et (c_2, ℓ_2) dans des parties différentes c_1 et c_2 du graphe est par construction la probabilité d'avoir dans l'ensemble des messages stockés un message contenant le symbole ℓ_1 en position c_1 et le symbole ℓ_2 en position c_2 :

$$\begin{aligned} d &= p(\exists w \in \mathcal{M}, w_{c_1} = \ell_1 \wedge w_{c_2} = \ell_2) \\ &= 1 - p(\forall w \in \mathcal{M}, w_{c_1} \neq \ell_1 \vee w_{c_2} \neq \ell_2) \\ &= 1 - p(w_{c_1} \neq \ell_1 \vee w_{c_2} \neq \ell_2)^M \\ &= 1 - (1 - p(w_{c_1} = \ell_1 \wedge w_{c_2} = \ell_2))^M \\ &= 1 - (1 - p(w_{c_1} = \ell_1)p(w_{c_2} = \ell_2))^M \\ &= 1 - (1 - \frac{1}{l^2})^M \ . \end{aligned}$$

Par suite, la probabilité qu'un sommet donné soit connecté à k sommets parmi b dans une autre partie est :

$$p_k = \binom{b}{k} d^k (1-d)^{b-k} \tag{4}$$

En particulier, la probabilité qu'un sommet est connecté à aucun des b sommets est :

$$p_0 = (1-d)^b \tag{5}$$

Revenons sur l'estimation de P_e après une seule itération de l'algorithme 1. L'erreur aura eu lieu si et seulement s'il y a au moins un sommet initialement actif et n'étant pas dans $f(w)$ restant actif après l'itération. Pour l'éviter, il faut donc que tous ces sommets n'atteignent pas le compte maximal : $c-1$. En utilisant à nouveau la loi binomiale, nous obtenons que cet événement a pour probabilité :

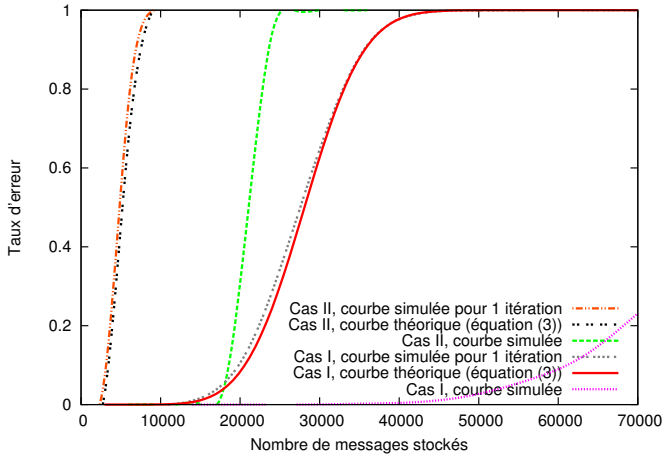
$$\left[1 - \left(1 - (1-d)^b\right)^{c-1}\right]^{c(b-1)} \tag{6}$$

L'évènement d'erreur étant contraire, nous concluons :

$$P_e \approx 1 - \left[1 - \left(1 - (1-d)^b\right)^{c-1}\right]^{c(b-1)} \tag{7}$$

En pratique, il a été montré que les performances peuvent être significativement meilleures lorsque le nombre d'itérations augmente, comme appuyé par la figure 1.

Figure 1 : Évolutions des taux d'erreur, théoriques ou simulés avec une seule itération ou jusqu'à convergence de l'algorithme 1, lorsque l'on cherche à répondre au problème 1 à l'aide de l'algorithme 1 ci-contre et que les messages stockés ont été générés uniformément et aléatoirement. Sur ces exemples, les paramètres du réseau sont les suivants : scénario I : $c = 8$, $\ell = 256$ et $b = 9$ ou scénario II : $c = 12$, $\ell = 256, b = 3$. L'algorithme 1 a donc vocation à retrouver l'unique bonne clique parmi les 9^8 ou respectivement les 3^{12} possibles.



3.2 Efficacité mémoire

Comme nous le soulignions en introduction, une grande avancée obtenue par l'utilisation des modèles de mémoires associatives à cliques est de proposer des efficacités mémoires asymptotiquement à un facteur constant de l'optimal pour une probabilité d'erreur fixée. Ces propriétés restent vraies dans le cas des messages flous, comme nous le montrons ci-après.

Plaçons-nous dans le régime asymptotique suivant : $\ell \to +\infty$, c et b constants, $M = \alpha\ell^2$, où α est une constante de sorte que d tend vers une valeur dans $]0;1[$. L'équation (7) indique que la probabilité d'erreur tend alors vers une constante dans $]0;1[$.

Pour cette probabilité d'erreur constante, l'efficacité mémoire, définie comme le rapport entre la quantité d'informations binaires apprises et la quantité d'informations binaires utilisées (définie par l'inégalité de Kraft [9]) pour spécifier le réseau vaut :

$$= \frac{\alpha\ell^2 c\log_2(\ell)}{\binom{c}{2}\ell^2} = \frac{\alpha\log_2(\ell)}{c-1}\ . \qquad (8)$$

Ainsi, à probabilité d'erreur bornée, l'efficacité mémoire peut-être rendue aussi grande que souhaitée.

4 Conclusion

Nous avons montré comment adapter des mémoires associatives introduites dans [6, 7] au cas d'entrées floues. Les performances obtenues sont comparables à celle des cas d'entrées partiellement effacées, de sorte que le modèle est très bien adapté à cette configuration. De plus, les modifications considérées n'affectent pas les propriétés du modèle initial. Ces extensions s'ajoutent donc aux propriétés déjà existantes de ces mémoires, en faisant des outils polyvalents pour une utilisation dans des applications de bases de données, ou de fouille de données.

Références

[1] Norman P. Jouppi. *Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers*. Actes du 17ème International Symposium on Computer Architecture, 1990

[2] Chyuan Shiun Lin, Diane C. P. Smith et John Miles Smith. *The design of a rotating associative memory for relational database applications*. ACM Transactions on Database Systems, 1976.

[3] Antonis Papadogiannakis, Michalis Polychronakis et Evangelos P. Markatos. *Improving the accuracy of network intrusion detection systems under load using selective packet discarding*. Actes du troisième groupe de travail Européen sur la sécurité des systèmes, 2010.

[4] K. Pagiamtzis et A. Sheikholeslami. *Content-addressable memory (CAM) circuits and architectures : a tutorial and survey*. *IEEE Journal of Solid-State Circuits*, volume 41, numéro 3, 2006.

[5] J. J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities*. Actes de la *National Academy of Sciences*, volume 79, 1982.

[6] V. Gripon et C. Berrou. *A simple and efficient way to store many messages using neural cliques*. Actes de la conférence *IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain*, 2011.

[7] V. Gripon et C. Berrou. *Sparse neural networks with large learning diversity*. *IEEE Transactions on Neural Networks*, volume 22, numéro 7, 2011.

[8] V. Gripon et C. Berrou. *Nearly-optimal associative memories based on distributed constant weight codes*. Actes du groupe de travail *Information Theory and Applications*, 2012.

[9] L. Kraft *A device for quantizing, grouping, and coding amplitude modulated pulses*. Mémoire de Master, Département *Electrical Engineering and Computer Science, MIT*, 1949.