Classification à l'aide d'un estimateur de complexité algorithmique

Marion REVOLLE, François CAYRE, Nicolas LE BIHAN
GIPSA-Lab|DIS|CICS
CNRS|UGA|Grenoble-INP
11 Rue des Mathématiques, 38400 Saint Martin d'Hères, France
prenom.nom@gipsa-lab.grenoble-inp.fr

Résumé – Dans le cas où les données ne se prêtent pas aisément à la modélisation probabiliste, on peut préférer le pendant algorithmique de la théorie de l'information, fondée sur la complexité de Kolmogorov [5]. Dans cette contribution, nous montrons comment estimer une semi-distance algorithmique entre objets binaires quelconques, qui soit à la fois plus précise et moins limitée que l'état de l'art.

Abstract – In the case where the data is not easily modeled by probabilistic tools, one could prefer using algorithmic information theory, based on Kolmogorov complexity [5]. In this paper, we show how to estimate an algorithmic semi-distance between arbitrary binary objects. This estimate is both more precise and incurs less limitations than state-of-the-art proposals.

1 Introduction

La complexité de Kolmogorov n'étant pas calculable sur une machine de Turing, on l'approche généralement avec la sortie d'un compresseur classique [4] ou la complexité de Lempel-Ziv [6]. Nous proposons ici une nouvelle approximation de la complexité de Kolmogorov, toujours fondée sur l'agorithme de Lempel et Ziv. Nous illustrons sa nette supériorité sur l'état de l'art à l'aide d'exemples de classification de données d'ADN, d'œuvres musicales et d'œuvres littéraires intégrales du corpus français.

2 SALZA

Dans [4] ou [9], les auteurs ont adaptés de manière *ad hoc* la notion de complexité relative en s'inspirant des résultats connus en théorie de l'information sur l'entropie conditionnelle et l'information mutuelle. Ici, nous proposons une approche intrinsèquement conditionnelle, appelée SALZA.

Par la suite, on considère deux chaînes de caractères x et y de longueur respective n_x et n_y et prenant leurs valeurs dans les alphabets $\mathcal{A}_x$ et $\mathcal{A}_y$, de taille finie.

2.1 Complexité conditionnelle

La complexité de Kolmogorov conditionnelle [5] de x connaissant y, notée $K(x|y)$, est la taille du plus petit programme capable de générer la chaîne x avec la connaissance *a priori* de la chaîne y. Classiquement, il existe deux approches pour estimer une complexité conditionnelle :

— dans [4], on utilise un compresseur sur la concaténation des chaînes x et y (et il est possible d'utiliser x pour l'expliquer, ce qui n'est pas toujours souhaitable) ;

— dans [10], les deux chaînes sont séparées par construction et seule y contribue à expliquer x.

Ici, nous définissons $\mathcal{R}$ la chaîne à l'aide de laquelle nous voulons expliquer x. Cette chaîne peut prendre différentes valeurs, en fonction de quatre scénarios bien particuliers, exprimés par rapport au symbole courant à encoder :

— $x|y : \mathcal{R}$ correspond au passé de y ; $\mathcal{A}_\mathcal{R} = \mathcal{A}_y$.

— $x|^+y : \mathcal{R}$ correspond à l'intégralité de y ; $\mathcal{A}_\mathcal{R} = \mathcal{A}_y$.

— $x_\text{-}|y$: $\mathcal{R}$ comprend le passé de x plus le passé de y ; $\mathcal{A}_\mathcal{R} = \mathcal{A}_x \cup \mathcal{A}_y$.

— $x_\text{-}|^+y : \mathcal{R}$ correspond à l'intégralité de y plus le passé de x ; $\mathcal{A}_\mathcal{R} = \mathcal{A}_x \cup \mathcal{A}_y$.

On note le cas général de conditionnement $x \wr y$, pouvant prendre ses valeurs parmi les quatre cas précités.

2.2 Décomposition

On note $x[i]$ le i$^{\text{ème}}$ élément de x et $x[i..i+l]$, avec $l > 1$, une sous-chaîne de x de longueur l. SALZA se base sur les deux opérations élémentaires "copier" et "insérer" pour coder la chaîne x à partir du symbole $x[i]$:

— "copier" : s'il existe une sous-chaîne dans $\mathcal{R}$ identique à $x[i..i+l]$, ce symbole, une *référence*, est noté $Z(p \to l)$, où p est la position dans $\mathcal{R}$ du premier octet à recopier et l la longueur à recopier ; l est ici toujours le plus long possible ;

— "insérer" : s'il n'existe aucune sous-chaîne de longueur $l \geq 2$, ce symbole, un *littéral*, est noté $L(x[i])$.

SALZA décompose donc la chaîne x en une séquence de symboles L et Z. $\mathcal{L}_{x\wr y}$ est l'ensemble des longueurs des symboles produits par SALZA (la longueur d'un littéral vaut 1).

Il est à noter que dans le cas $x|x$, la décomposition obtenue est la même que celle réalisée par LZ77 [6] appliqué sur x avec un buffer de taille semi-infinie.

Une fois la décomposition obtenue (succession de *littéraux*

L et de *références* Z), la mesure de complexité de Lempel et Ziv [6], notée $c(x)$ par la suite, est le nombre de symboles (L et Z) obtenus lors de la décomposition. Il est possible d'étendre cette mesure aux différents cas de complexité conditionnelle vus ci-dessus (Sec. 2.1).

$$c(x \wr y) = |\mathcal{L}_{x\wr y}| \tag{1}$$

avec $|\mathcal{L}_{x\wr y}|$ le cardinal de l'ensemble $\mathcal{L}_{x\wr y}$. Cette mesure, uniquement fondée sur le nombre de symboles de la décomposition, ne capture pas toute l'information, et nous proposons de prendre également en compte la longueur des symboles dans une nouvelle mesure de complexité (cf. Sec. 2.3).

2.3 Longueur significative

La prise en compte des longueurs des symboles permet de moduler l'importance que l'on souhaite leur donner. Un résultat connu [6] est que $c(x)$ est bornée :

$$c(x) < \frac{n_x}{\log_{|\mathcal{A}_x|}(n_x)}. \tag{2}$$

Ainsi, si x est une très longue chaîne aléatoire, alors, pour la décomposition de $x|x$, les longueurs stockées dans $\mathcal{L}_{x|x}$ sont de longueur moyenne $\log_{|\mathcal{A}_x|}(n_x)$. Donc dans le cas général $x \wr y$, on définit un seuil de longueur l_0, donné par :

$$l_0 = \log_{|\mathcal{A}_\mathcal{R}|}(|\mathcal{R}|). \tag{3}$$

Si une longueur est très supérieure à l_0, alors il est possible de faire une grand recopie dans $\mathcal{R}$: x est en partie bien expliquée grâce à $\mathcal{R}$. Cette longueur est alors dite significative. À l'inverse, si la longueur est inférieure à l_0, alors rien ne ressemble à cette partie de x dans $\mathcal{R}$: x s'explique difficilement grâce à $\mathcal{R}$. Cette longueur est donc non significative.

Pour moduler à quel point la longueur est significative, nous proposons ici la fonction sigmoïde centrée en l_0, $f(l) = \frac{1}{1+e^{-l+l_0}}$. Ainsi, si la longueur est significative, $l >> l_0$, alors $f(l) = 1$; à l'inverse si la longueur est non significative, $l << l_0$, alors $f(l) = 0$; et si la longueur est proche du seuil l_0, alors $f(l) \approx 0.5$. Le choix de de la fonction sigmoïde est justifié dans [7].

On définit alors la proportion significative du signal, notée $p(x \wr y)$, Cette proportion est la somme des longueurs des symboles pondérées par la fonction sigmoïde centrée en l_0 ,

$$p(x \wr y) = \sum_{\mathcal{L}_{x\wr y}} (l-1) f(l) \tag{4}$$

et traduit quelle part de x est bien expliquée par y.

2.4 Complexité SALZA : S

Une fois la chaîne x décomposée en symboles L et Z selon l'information conditionnelle $x \wr y$, il est possible d'introduire un complexité reposant sur $c(x \wr y)$ (Eq. 1) et $p(x \wr y)$ (Eq. 4).

Nous définissons donc notre complexité comme suit :

$$S(x \wr y) = \frac{c(x \wr y) - 1}{n_x} \left(1 - \frac{p(x \wr y) + 1}{n_x}\right) \tag{5}$$

La complexité S est normalisée pour être indépendante des longueurs des chaînes et comprise entre 0 et 1, où 0 est atteint quand $x = y$. On peut facilement vérifier que S($x|^+x$) = 0 et que S($x|^+y$) = 1 quand les alphabets sont disjoints, la preuve est dans [7].

Il est aussi possible de calculer la complexité simple de la chaîne x en utilisant l'opérateur conditionnel $x|x$:

$$S(x|x) = S(x), \tag{6}$$

cette complexité s'avérant plus précise que celle obtenue par un compresseur [4] du fait de la prise en compte des longueurs des symboles.

2.5 Exemples

Afin d'illustrer le comportement de SALZA, considérons les trois chaînes suivantes : $x = ABCABCCBCABC$, $y = ABCABCABCABC$ et $z = MNOMNOMNOMNO$.

Comme expliqué dans la Sec. 2.2, les différentes décompositions vues dans la Sec. 2.1 sont :

- $x|x : L(A)L(B)L(C)Z(0 \to 3)L(C)Z(1 \to 5)$;
- $y|^+x : Z(0 \to 6)Z(0 \to 6)$;
- $z|^+x : L(M)L(N)L(O)L(M)L(N)L(O)...$;
- $z\_|^+x : L(M)L(N)L(O)Z(0 \to 9)$.

Pour les trois premiers cas, la longueur significative vaut : $l_0 = \log_3(12) = 2.26$, et pour le dernier cas, $l_0 = \log_6(12) = 1.39$. Les différentes valeurs de complexité pour chacune des décompositions sont :

- $S(x) = \frac{6-1}{12}\left(1 - \frac{2f(3)+4f(5)+1}{12}\right) = 0,20$;
- $S(y|^+x) = \frac{2-1}{12}\left(1 - \frac{5f(6)+5f(6)+1}{12}\right) = 0,0086$;
- $S(z|^+x) = \frac{12-1}{12}\left(1 - \frac{0+1}{12}\right) = 0,84$;
- $S(z\_|^+x) = \frac{4-1}{12}\left(1 - \frac{8f(9)+1}{12}\right) = 0,063$.

La complexité simple de x est assez faible car x s'explique en grande partie par lui-même : 7 éléments sur 12 peuvent être expliqués en faisant un "copier" à partir du passé. La complexité de $y|^+x$ est très faible car y s'explique bien avec la connaissance de x : aucun littéral n'est émis. Par contre, la complexité de $z|^+x$ est très élevée car les deux alphabets sont totalement disjoints. Cependant, si l'on rajoute la connaissance du passé de z, qui est une chaîne à faible complexité, alors la complexité de $z\_|^+x$ redevient faible.

3 Application à la classification

La classification requiert la définition d'une (semi-)distance entre deux objets numériques afin de construire le graphe associé à la matrice de (semi-)distances.

3.1 La semi-distance *NSD*

Dans [1], la distance entre deux chaînes x et y est définie par :

$$ID(x,y) = max\{K(x|y), K(y|x)\}, \tag{7}$$

c'est le plus petit programme capable de générer x sachant y et y sachant x.

En pratique, il faut utiliser deux approximations :

— Approcher $K(x)$ par la taille du fichier généré par la compression de x ;

— Avec un compresseur, il n'est pas possible de calculer une complexité conditionnelle, il faut donc considérer $K(x|y) \approx K(xy) - K(y)$, où xy est la concaténation de x et y.

La version calculable et normalisée de la ID devient alors la *NCD* (*Normalized Compression Distance*) [4] :

$$NCD(x,y) = \frac{C(xy) - min\{C(x), C(y)\}}{max\{C(x), C(y)\}}. \quad (8)$$

En injectant SALZA dans l'Eq. 7, nous obtenons une semi-distance normalisée appelée *NSD* (*Normalized SALZA semi-Distance*) :

$$NSD(x,y) = max\{S(x|^{*}y), S(y|^{*}x)\}. \quad (9)$$

Il est facile de montrer [7] les propriétés de symétrie et de non-négativité et l'identité des indiscernables (non respectées par la *NCD*). Cependant, il est possible de trouver des exemples qui violent l'inégalité triangulaire. Cette perte entre ID et *NSD* est due au fait que les seules opérations prises en compte dans SALZA sont "copier" et "insérer", il n'est donc pas possible de détecter par exemple une permutation comme dans la complexité de Kolmogorov.

3.2 Résultats

En calculant une matrice de semi-distances entre plusieurs fichiers, il est possible de créer un arbre phylogénique ([8]) à des fins de classification.

Dans un premier temps, nous nous plaçons dans les conditions de la *NCD* en utilisant `gzip` comme compresseur, ce qui implique que les signaux ne doivent pas excéder la taille de 16 KiB [3].

La Fig. 1 contient les arbres phylogéniques obtenus avec la *NCD* et la *NSD* pour des fichiers d'ADN et de musiques (format MIDI) : notre méthode se révèle beaucoup plus discriminante. Il est donc plus facile de faire la différence entre deux données très proches et donc la classification est facilitée.

Sur les échantillons d'ADN de mammifères terrestres, on retrouve bien dans la Fig. 1 (ligne supérieure) l'arbre phylogénique obtenu classiquement ([2]), mais les branches sont plus aérées avec la *NSD* qu'avec la *NCD*/`gzip`.

Concernant l'exemple de la musique, les fichiers MIDI comparés sont : les *Sonates* 1 (KV279), 2 (KV 280) et 3 (KV 281) de Mozart, le *Clavier Tempéré* en Ut (BWV846), en Sol (BWV854) et en La (BWV860) de Jean-Sébastien Bach et les quatre parties de la *Suite Bergamesque* de Debussy. Les œuvres d'un même compositeur ont entre elles une distance plus faible qu'avec un morceau d'un autre compositeur. La *NSD* permet une classification par compositeur à la fois plus marquée et plus fiable qu'avec la *NCD*/`gzip` (*e.g.*, la *Sonate* 1 de Mozart se retrouve sur la même branche que les œuvres de Bach avec la *NCD*), voir la ligne inférieure de la Fig. 1.

En plus d'être beaucoup plus discriminante, la *NSD* permet de s'affranchir de la limite de 16 KiB. Dans un second temps, nous avons donc classifié des œuvres de la littérature classique française (Fig. 2). Comme les livres sont en général d'une taille bien supérieure à 16 KiB, la *NCD*/`gzip` ne parvient pas à classifier correctement les œuvres, alors que la *NSD*, quant à elle, regroupe la plupart des œuvres par auteur. De plus, les auteurs du XIX^{e} s. (Maupassant, Zola, Stendhal et Dumas) et les auteurs du XVI^{e} s. (Montaigne et Rabelais) sont regroupés sur les mêmes branches. De même, le genre *conte* est classifié à part : *Candide* se retrouve aux côtés des contes de Perrault et d'Aulnoy.

La *NSD* est ainsi une semi-distance plus précise que celles fondée sur un compresseur classique, et qui s'affranchit des limitations induites par l'utilisation de ces derniers.

Références

[1] Charles H. Bennett, Peter Gacs, Ming Li, Paul M.B. Vitániy, and Wojciech H. Zurek. Information Distance. *IEEE Transactions on Information Theory*, 44 :1407–1423, Jul. 1998.

[2] Ying Cao, Axel Janke, Peter J. Waddell, Michael Westerman, Osamu Takenaka, Shigenori Murata, Norihiro Okada, Svante Pääbo, and Masami Hasegawa. Conflict Among Individual Mitochondrial Proteins in Resolving the Phylogeny of Eutherian Orders. *Journal of Molecular Evolution*, 47(3) :307–322, 1998.

[3] Manuel Cebrián, Manuel Alfonseca, and Alfonso Ortega. Common Pitfalls Using the Normalized Compression Distance : What to Watch Out for in a Compressor. *Communications in Information and Systems*, 5 :367–384, 2005.

[4] Rudi Cilibrasi and Paul M.B. Vitányi. Clustering by Compression. *IEEE Transactions on Information Theory*, 51 :1523–1545, Apr. 2005.

[5] Andrei N Kolmogorov. Three Approaches to the Quantitative Definition of Information. *Problems of Information Transmission*, 1(1) :1–7, 1965.

[6] Abraham Lempel and Jacob Ziv. On the Complexity of Finite Sequences. *IEEE Transactions on Information Theory*, 22 :75–81, January 1976.

[7] Marion Revolle, François Cayre, and Nicolas Le Bihan. SALZA : Soft Algorithmic Complexity Estimates for Clustering and Causality Inference. *DRAFT*, 2016.

[8] Naruya Saitou and Masatoshi Nei. The Neighbor-Joining Method : A New Method for Reconstructing Phylogenic Trees. *Molecular Biology and Evolution*, 4 :406–425, July 1987.

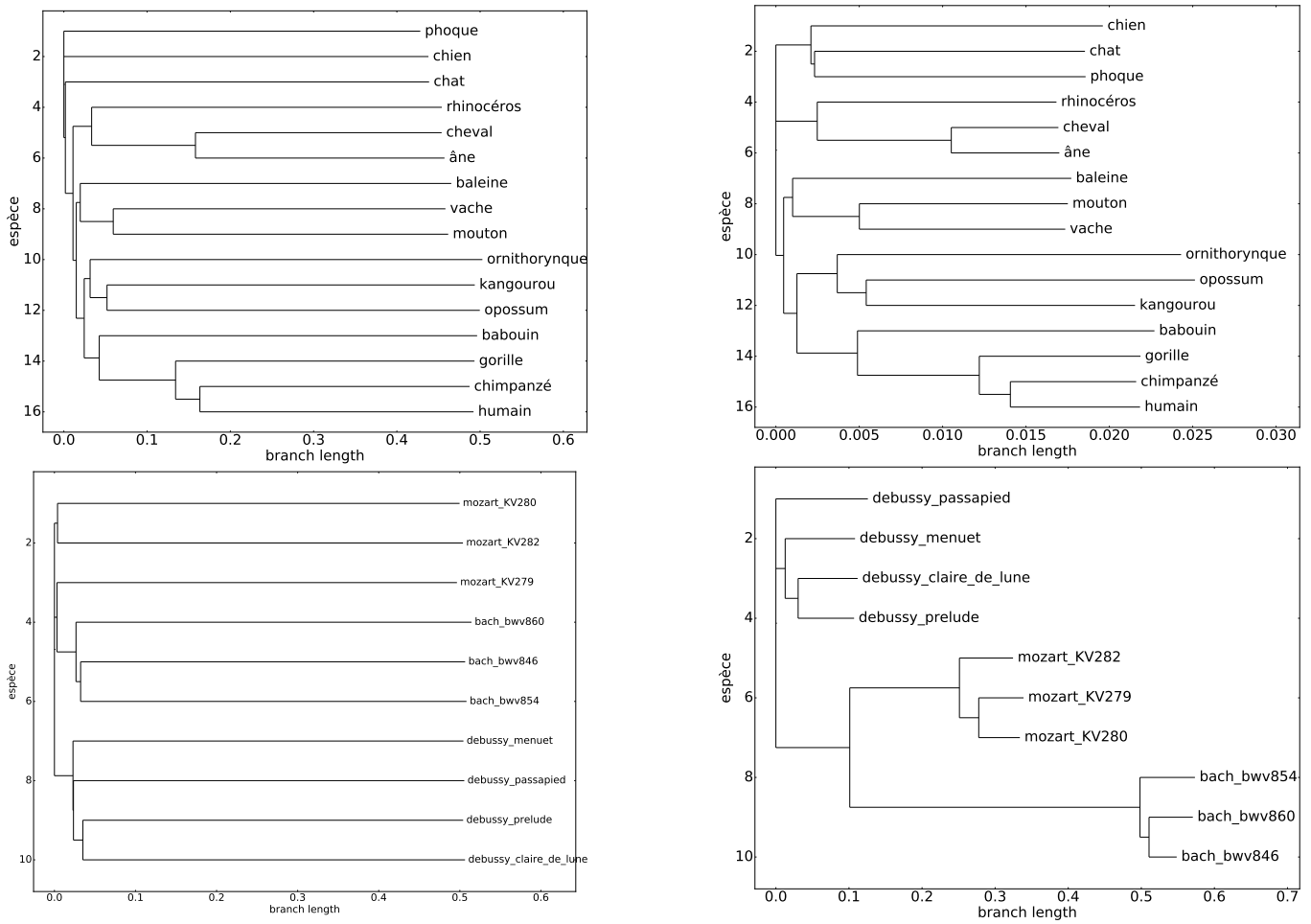

FIGURE 1 – Arbre phylogénique d'ADN de mammifères sur la première ligne et de musique sur la deuxième ligne réalisé grâce à la *NCD*/`gzip` sur la première colonne et grâce à la *NSD* sur la deuxième colonne. La taille des branches correspond à la similitude entre les objets.

[9] Miguel Angel Veganzones, Mihai Datcu, and Manuel Grana. Dictionary based Hyperspectral Image Retrieval. In *ICPRAM (1)*, pages 426–432, 2012.

[10] Jacob Ziv and Neri Merhav. A Measure of Relative Entropy Between Individual Sequences with Application to Universal Classification. *IEEE Transactions on Information Theory*, 39 :1270–1279, Jul. 1993.

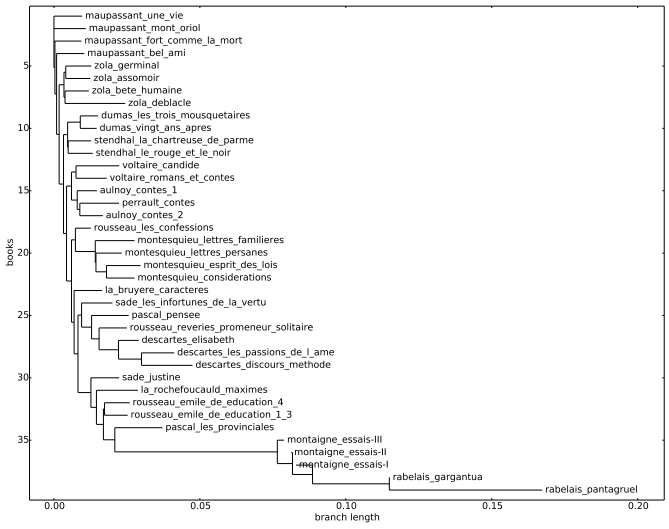

FIGURE 2 – Arbre phylogénique de la littérature française réalisé grâce à la *NSD*, chaque élément est composé du nom de l'auteur suivi du nom de l'œuvre.