le prédicteur universel discret de Shannon

Sujet proposé par Philippe Jacquet

`Philippe.Jacquet[at]inria.fr` *Difficulté : moyen (**).*

1 Préambule : le contexte

Tout commence par un jeu mathématique à l'allure de jeu de société et qui a été inventé par Claude Shannon en 1948. Un volontaire se voit assigné la tâche de simuler un générateur aléatoire, par exemple en inventant une succession imaginaire de piles ou faces. Derrière un rideau un opérateur avec un ordinateur essaie de deviner le prochain terme de la succession avant que le volontaire ne l'annonce. En général l'ordinateur devine les termes de la succession avec une statistique supérieure à 50 %. Ce jeu du prédicteur est aussi connu sous le terme de *mind reader machine* puisque la machine donne l'impression de pouvoir lire dans les pensées du cobaye. Bien sûr, il ne s'agit pas de magie, la machine exploite une faille du cerveau humain qui fait de lui un piètre générateur aléatoire. Les prédicteurs ont beaucoup d'applications en contrôle, en automatique, en bioinformatique, dans les éditeurs de texte (par exemple ceux qui sont dans les téléphones portables), dans les jeux de réflexes ou de stratégies sur vidéo.

2 Détail du sujet

2.1 Interface

Interface on-line ou off-line. Lire une séquence de caractères et énoncée par un cobaye (on-line) ou par un fichier texte (off-line). L'ordinateur affiche les caractères prédit ou la séquence de caractères prédits. Prévoir aussi l'affichage des statistiques de succès.

Programmer un prédicteur basé sur cet algorithme. Essayer l'algorithme en ligne avec un cobaye humain. Etendre cet algorithme sur un alphabet plus grand, comme par exemple celui des caractères. Essayer l'algorithme sur un fichier texte important, noter les statistiques de succès.

2.2 L'algorithme de base

Le cobaye énonce la séquence pseudo aléatoire $x_0x_1\cdots x_n\cdots$ non bornée on suppose que $x_i \in \{0,1\}$. La machine énonce la suite binaire $y_0y_1\cdots y_n\cdots$. Le terme y_n est une fonction de $x_0x_1\cdots x_{n-1}$: $y = f(x_0^{n-1})$ où x_i^j désigne la séquence $x_ix_{i+1}\cdots x_j$ quand $i < j$.

La fonction est la suivante : la machine cherche le plus long suffixe de x_0^{n-1} qui soit répété dans x_0^{n-1}. En d'autre terme on cherche le plus grand entier k tel qu'il existe $k-1 \leq j < n-1$ et $\forall i < k$ $x_{n-1-i} = x_{j-i}$. On appelle j un "marqueur" de x_0^{n-1} pour la longueur k, et la plus grande valeur de k la "profondeur" de la séquence x_0^{n-1}. La machine prend pour y_n la valeur de x_{j+1}, c'est à dire la valeur qui suit juste à droite du marqueur. Si il existe plusieurs marqueurs possibles pour

k égal à la profondeur, on prend celui qui est le plus grand, c'est à dire le plus à droite, correspondant à la copie la plus récente du plus long préfixe.

Par exemple, pour $x_0^9 = 0101110110$, la profondeur est 3 et il n'y a qu'un seul marqueur pour 3 situé en position 6. Donc $y_{10} = x_7 = 1$. Il y a deux marqueurs pour la longueur 2 : un en 2, l'autre en 6. L'ensemble des marqueurs pour la longueur 1 est $\{0, 2, 6\}$, correspondant à toutes les positions où les valeurs sont égales à $x_9 = 0$.

Pour $x_0^9 = 0000000000$, la profondeur est 9 et il n'y a qu'un seul marqueur en position 8 et le prédicteur donnera $y_{10} = x_9 = 0$. Pour la longueur 8 il y a deux marqueurs en positions $\{7, 8\}$. Pour la longueur $k < 9$ les marqueurs sont en positions $\{k - 1, \dots, 8\}$.

Pour $x_0^9 = 0000100000$, la profondeur est 4, il y a deux marqueurs en position 3 et 8 et la prédiction sera 0.

Remarque : le cobaye aura peut-être appris par avance la suite antiprédictive : 0100110101110…, cette suite a des propriétés intéressantes au niveau des codes pseudo-aléatoires, mais elle ne sera pas étudiée ici.

2.3 Structure de données

Faire en sorte que la complexité reste proportionnelle à n, c'est-à dire la taille de la séquence qui précède. Pour cela faire on pourra metre à jour de manière dynamique un tableau des marqueurs avec leurs profondeurs respectives.

3 Pour aller plus loin

3.1 Un algorithme plus efficace

L'algorithme précédent est loin d'être efficace dans le cas général. En effet supposons que la séquence du cobaye soit réellement alèatoire et que les x_i sont indépendants et identiquement distribués avec une probabilité p_0 d'avoir 0 et une probabilité p_1 d'avoir 1 ($p_0 + p_1 = 1$). On peut alors s'attendre à ce que la probabilité d'avoir $y_n = 1$ est p_1 et celle d'avoir $y_n = 0$ est p_0, donc que la probabilité que le prédicteur vise juste (c'est à dire $y_n = x_n$) est $(p_0)^2 + (p_1)^2$. Pour $p_0 = p_1 = 1/2$ c'est optimal mais si $p_0 < p_1$ alors $(p_0)^2 + (p_1)^2 < p_1$, c'est à dire le prédicteur qui prédirait systématiquement la valeur 1 est plus efficace que le prédicteur basé sur l'algorithme précédent.

Mathématiquement le prédicteur optimal est atteint quand la machine connait le système aléatoire du cobaye. Le système aléatoire du cobaye peut être autre chose qu'une simple suite i.i.d. (par exemple un processus de Markov sophistiqué) il n'y a pas de raison que le prédicteur optimal donne toujours la même valeur pour y_n indépendamment de la séquence x_0^{n-1}.

Dans ce qui suit nous allons définir un prédicteur universel, c'est à dire un prédicteur qui s'approche asymptotiquement du prédicteur optimal pour une large classe de systèmes aléatoires, et ce bien sûr en ignorant *a priori* le système aléatoire du cobaye.

Nous décrivons l'algorithme dans le cas binaire mais l'extension à n'importe quel alphabet est évidente. L'algorithme optimisé détermine la profondeur k de la séquence x_0^{n-1}. En posant $m = \lceil \frac{k}{2} \rceil$, il identifie les marqueurs pour la longueur m, c'est à dire pour la moitiée de la profondeur. Il y aura donc plus de marqueurs à examiner que dans l'algorithme précédent. L'ordinateur enregistre pour chaque marqueur la valeur du caractère qui le suit. La prédiction y_n est la valeur enregistrée qui est apparue le plus souvent.

Programmer l'algorithme optimisé. Essayez le sur un cobaye humain. Au lieu de prendre $m = \lceil \frac{k}{2} \rceil$, prendre $m = \lceil \alpha k \rceil$ pour $\alpha < 1$. Essayer sur des fichiers textes longs. Prédire plus d'un caractère à la fois. Définir un algorithme pour décider de la longueur de la chaîne de caractère à prédire en fonction de la statistique d'occurence après les marqueurs.

3.2 Un algorithme généralisé

Les algorithmes que nous avons décrit précédemment est du genre tout ou rien c'est à dire que dans la comparaison de deux séquences il s'arrête dés le premier caractère différent. Si x_0^n et z_0^n sont deux séquences on a défini implicitement

$$\begin{aligned} L(x_0^n, z_0^n) &= \delta(x_n, z_n) + \delta(x_n, z_n)\delta(x_{n-1}, y_{n-1}) + \\ &+ \cdots + \delta(x_n, z_n)\delta(x_{n-1}, z_{n-1}) \cdots \delta(x_i, z_i) + \cdots \end{aligned}$$

où $\delta(a, b) = 1$ si $a = b$ et $\delta(a, b) = 0$ autrement. $L(x_0^n, z_0^n)$ donne ainsi la longueur du plus long suffixe commun entre x_0^n et z_0^n. La profondeur est donc $\max_{i<n} L(x_0^i, x_0^n)$.

Noter qu'on peut écrire pareillement

$$\begin{aligned} L(x_0^n, z_0^n) &= \delta(x_n, z_n)\left(1 + \delta(x_{n-1}, z_{n-1}\left(\cdots(1 + \delta(x_0, z_0))\cdots\right)\right) \\ &= \delta(x_n, z_n)(1 + L(x_0^{n-1}, z_0^{n-1}))\ . \end{aligned}$$

L'objectif est maintenant d'assigner des valeurs autres que 0 et 1 à la fonction δ. Par exemple on peut définir la matrice symétrique Δ dont les coefficients (i, j) sont les valeurs de $\delta(i, j)$. On garde $\delta(i, i) = 1$. Par exemple :

$$\Delta = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

De cette manière la métrique de comparaison est plus flexible et permet de prendre en compte et de quantifier la notion d'erreur. Cela peut être particulièrement utile si il s'agit de comparer une suite de caractères qui sont des octets. Par exemple la succession des signaux audio d'un morceau de musique ou un histogramme de températures, ou une succession d'intensités lumineuses dans une image, avec $\delta(x, y) = \exp(-(x-y)^2)$ ou quelque chose de comparable. Plus simplement cela peut permettre de sauter d'éventuelles erreurs de mesures comme dans le séquençage d'un génome, par exemple.

L'algorithme généralisé fonctionne comme dans le prédicteur optimal, sauf que maintenant la profondeurs est un nombre flottant $R(x_0^n)$. On identifie les marqueurs comme les valeurs $i < n$ telles que $L(x_0^i, x_0^n) > \alpha R(x_0^n)$. La machine enregistre les valeurs du caractère qui suit chaque marqueur. Le point délicat est dans le choix de la valeur y_n prédite. Il y a deux stratégies possibles :

- la valeur la plus fréquente parmi les valeurs enregistrées ;
- la valeur qui diffère le moins avec les valeurs enregistrées

La première méthode est en droite ligne des algorithmes précédents. Il donnera une réponse satisfaisante tant que la taille de l'alphabet n'est pas trop grande. Si la taille de l'alphabet est trop grande et que les caractères subissent trop de dispersions, alors il est possible que chaque valeur enregistrée ne soit atteinte qu'une seule fois. L'algorithme reviendrait à choisir une valeur uniformément au hasard, ce qui n'est pas satisfaisant. Dans ce cas la seconde stratégie est meilleure.

Soit M l'ensemble des marqueurs de x_0^n, les valeurs enregistrées par la machines sont donc les valeurs x_{i+1} pour $i \in M$. La valeur prédite peut être celle qui maximise la quantité $\sum_{i \in M} \delta(y, x_{i+1})$.

Noter que si pour toute paire de caractères (a, b) : $\delta(a, b) = 0$ quand $a \neq b$ et $\delta(a, a) = 1$, alors cela revient à prendre la valeur la plus fréquente.

Noter aussi que la valeur qui maximise $\sum_{i \in M} \delta(y, x_{i+1})$ pourrait très bien ne pas être une valeur enregistrée, par exemple si les températures sautent entre les valeurs 18.4 et 18.6, la meilleure valeur sera peut-être 18.5. Mais dans ce cas la recherche de la meilleure valeur peut être trop longue si la matrice Δ est arbitraire.

Chercher les meilleurs compromis et essayer l'algorithme sur des longs fichiers. Garder la complexité linéaire de l'algorithme (le tableau de marqueurs sert encore !).

Les projets info sont dispo à `http://www.enseignement.polytechnique.fr/informatique/IF/`

Références

[1] P. Jacquet W. Szpankowski and I. Apostol : "A Universal Predictor Based on Pattern Matching," *IEEE Trans. Information Theory*, 48, 1462-1472, 2002. (Special issue in memory of Aaron Wyner.) http ://www.cs.purdue.edu/homes/spa/papers/prediction.ps