Revue littéraire - CryptoNet Report

Thomas

September 2017

1 Introduction

L'enjeu est de pouvoir appliquer des réseaux neuronaux sur des bases de données cryptées à partir de la méthode FHE, sans avoir accès à la clef privée et donc sans avoir à les décrypter.
Les CryptoNets ont été testés sur le jeu de données MNIST basée sur la reconnaissance de caractères manuscrits. Obtenant ainsi une précision de 99% et réalisant près de 51000 prédictions par heure sur une seule machine :

$$1\,process/570\,secondes$$

$$1\,process = 8192\,prédictions\,en\,même\,temps$$

$$D'où :\ \frac{3600 * 8192}{570} \approx 51739\,pred/h$$

2 Contraintes d'implémentation

L'encryptage homomorphique ne supportant que les additions et les multiplications, seuls les fonctions polynomiales peuvent être directement calculées.
Par ailleurs en raison de la complexité importante induite par les différentes fonctions d'activation traversées et les multiplications imbriquées que cela implique : il est préférable de restreindre les calculs à des polynômes de degrés faibles.

A ce titre la fonction de somme pondérée peut directement être intégrée puisqu'elle n'utilise que des additions et des multiplications et que ces multiplications sont réalisée entre des poids pré-calculés et des valeurs issues des layers en amont. En revanche, la fonction de *Max pooling* ne peut être simplement implémenté car il ne s'agit pas d'une fonction polynomiale, cependant il est possible d'utiliser la relation suivante afin de pouvoir en déduire une approximation :

$$max(x_1, ...x_n) = lim_{d\to\infty}(\sum_i x_i^d)^{1/d}$$

En prenant d suffisamment petit pour garder des degrés faibles on peut remplacer la fonction précédente par la fonction de moyenne "réduite", à savoir : $\sum_i x_i$, simple à implémenter dans le cadre de cryptage homomorphique.

Donc pour rendre un réseau neuronal compatible avec le cadre de l'encryptage FHE, certaines modifications sont nécessaires, de sorte que les fonctions d'activations s'écrivent sous une forme polynomiale. Par ailleurs ces changements sont préférablement à prendre en compte lors de la phase d'apprentissage.

3 Leveled homomorphic encryption

3.1 Introduction

Alors que le FHE (Fully Homomorphic Encryption) permet d'effectuer autant d'additions et de multiplications sur les données cryptées que l'on souhaite sans perdre leur consistance, ici est utilisée

une variante appelée LVE (Leveled Homomorphic Encryption) qui limite le nombre d'opérations à une valeur pré-définie lors de l'encryptage.
Cette limitation ici n'est pas un problème puisque l'on connaît "à l'avance" le nombre d'étapes que comporte notre réseau neuronal.

3.2 Présentation de la méthode

La méthode d'encryptage transforme les messages issus de l'anneau $R_t^n := Z_t[x]/(x^n + 1)$ vers l'anneau $R_q^n := Z_q[x]/(x^n + 1)$.
La procédure d'encryptage choisie deux polynômes aléatoirement $f', g \in R_q^n$ et définie $f := t.f' + 1$ comme étant la clef privée, alors que la clef publique est définie telle que : $h := t.g.f^{-1}$.
Puisque tous les éléments de R_q^n ne sont pas inversables, ces étapes s'effectuent itérativement jusqu'à obtenir f inversable.

3.3 Cryptage d'un message

Un message $m \in R_t^n$ est crypté selon la règle suivante :

$$c := [\lfloor q/t \rfloor m + e + hs]_q$$

Avec e et s du bruit polynomial dans R_q^n. On utilise la notation $[a]_q$ afin de définir la réduction du coefficient de a modulo q vers l'intervalle de longueur q centré autour de 0.
Le décryptage s'effectue tel que :

$$m := [\lfloor \frac{t}{q} fc \rceil]_t$$

Ici le produit fc est d'abord calculé dans R_q^n, les coefficients sont alors interprétés comme des entiers puis *scalés* par t/q et arrondi à l'entier le plus prêt, avant d'être réduit modulo t.

4 Contraintes pratiques

Il est important de remarquer que la méthode présentée ne fonctionne que si les bruits apparaissant lors de l'encryptage demeurent suffisamment petit, sinon le décryptage pourrait être altéré.
Par ailleurs le niveau de sécurité du système dépend des paramètre n,q et t ainsi que de la quantité de bruit ajouté. Cependant la quantité maximale de bruit que peut supporter un message crypté en gardant sa consistance dépend des paramètres q et t.
Lorsque les messages cryptés sont ajoutés ou multipliés, le bruit qui leur est associé augmente naturellement, et ce particulièrement dans le cas de la multiplication. Il faut donc s'assurer d'avoir une valeur de q suffisamment grande pour contenir cette augmentation, ce qui nécessite donc une augmentation de n pour des raison de sécurité.

La principale limitation de cette méthode dans notre contexte correspond au nombre de multiplication réalisée au sein du réseau neuronal. Il s'agit du *niveau*, s'il correspond à une faible valeur il permet de choisir des valeurs plus petites pour les paramètres ce qui permet d'obtenir un temps de calcul réduit et des messages cryptés moins "gros".

Il est également nécessaire de garder t assez grand afin d'éviter que les coefficients des polynômes des messages non cryptés puissent être réduit modulo t durant les calculs au sein du réseau neuronal et provoquer alors une altération des données. Il faut donc garder un oeil sur la taille des coefficients de ces polynômes afin de pouvoir ajuster le paramètre t en conséquence.

Enfin pour améliorer les performances de notre méthode, on remarquera que les poids issus de la somme des inputs provenant du layer précédent au sein du réseau neuronal n'ont pas besoin d'être cryptés et peuvent donc être multipliés directement avec les données cryptées de manière bien plus efficiente.

Lors de l'encodage, il faut garder à l'esprit qu'un réseau neuronal ne fonctionne qu'avec des nombres réels en entrée, ce qui tranche avec les polynômes générés par la méthode d'encryptage homomorphique. Dès lors plusieurs méthodes sont possibles pour transformer ces données. Attention cependant, encrypter de très grands nombres peut s'avérer délicat (*cf. méthode des restes chinois*).

Il est également possible de paralléliser les calculs liés à l'encryptage, en ayant recours une nouvelle fois au théorème des restes chinois. En effet, le cryptage a recours à l'utilisation de polynômes de degrés élevés (n), mais si les données sont encodées sous la forme d'un scalaire alors seulement un unique coefficient parmi les n sera utilisé. Ce qui a pour conséquence de ralentir "inutilement" les opérations alors que le résultat ne comportera qu'un seul coefficient significatif.
En utilisant le théorème des restes chinois on peut réaliser des opérations SIMD (*Single Instruction Multiple Data*) : En supposant que t soit choisis tel que $x^n + 1 = \Pi(x - \alpha_i)(mod\, t)$ il est possible de montrer que $R_t^n \simeq Z_t^{\times n}$. L'isomorphisme est explicite et calculé facilement ce qui permet d'encoder n valeurs dans un seul polynôme, effectuer les opérations nécessaires sur ce polynôme, puis décoder les n résultats différents.

5 Résultats empiriques obtenus

5.1 Paramétrisation

Afin de pouvoir utiliser le théorème des restes chinois pour encoder de grands nombres, deux valeurs de t ont été utilisées à savoir : $t_1 = 1099511922689$ et $t_2 = 1099512004609$, en remarquant que leur produit est supérieur à 2^{80} ce qui permet d'appliquer le réseau, tout en étant suffisamment petit pour prendre $q = 2^{383} - 2^{33} + 1$ et ainsi avoir :

$$x^{8192} + 1 = \prod_{i=0}^{8191} (x - \alpha_i^{1,2}) \ (mod\, t_{1,2})$$

en garantissant que le bruit ne croît pas excessivement.

5.2 Configuration

Les CryptoNets ont été testés sur le dataset MNIST composé de 60000 images, scindés en un jeu de données d'apprentissage de 50000 images et d'un jeu de test de 10000 images.
Le réseau accepte des batchs de 8192 images (puisqu'on a choisi des polynômes de degrés $n = 8192$ lors de la paramétrisation du processus d'encryptage). Ses performances ne seront donc mesurés que sur les 8192 premières images du jeu de test.
Le réseau a été testé sur un PC équipé d'un intel Xeon E5-1620 @3.5GHz avec 16Go de RAM.

5.3 Résultats

Appliquer le réseau neuronal sur cette machine a pris 570 secondes (à noter qu'il est possible en utilisant la méthode SIMD d'effectuer simultanément 8192 prédictions simultanément et ainsi obtenir jusqu'à 51739 prédictions par heure.
L'encryptage des données s'est déroulé en 122 secondes (pour les 8192 images) auquel on ajoute 0.06 secondes pour l'encodage de chaque instance (soit $\frac{3600 \times 8192}{122 + 0.06 \times 8192} = 48068$ instances par heure).
Le décryptage des données est effectué en 5 secondes en sus des 0.046 secondes pour décoder chaque instance (soit $\frac{3600 \times 8192}{5 + 0.046 \times 8192} = 77236$ prédictions par heure).
Les images analysées sont au format 28×28 pixels, alors que chaque pixel est encrypté sous forme de deux polynômes (pour pouvoir encoder des grandes valeurs via le théorème des restes chinois). Chaque coefficient polynomial (il y en a donc 8192 pour chaque polynôme) nécessite 48 octets et donc la taille de chaque image équivaut à $28 \times 28 \times 8192 \times 2 \times 48$ octets soit 588 Mo. Cependant, nous avons vu qu'il était possible qu'un même message contienne jusqu'à 8192 images, ramenant la taille d'une image à 73.5 Ko.
La réponse du classifieur ne contient que 10 valeurs (pour les 10 chiffres possibles) et donc la taille de la réponse cryptée est de $10 \times 8192 \times 2 \times 48$ octets soit 7.5 Mo. Si les 8192 images ont été encodées ensemble, la taille d'une image correspond alors à 0.94 Ko.

A titre de comparaison, si l'on considère que chaque pixel d'une image est codé selon un entier flottant à double précision (encodé sur 64 bits soit 8 octets, un format peu optimisé pour ces données), la taille d'une image non cryptée est de $28 \times 28 \times 8$ octets, soit 6.27 Ko. C'est environ 12 fois plus petit que la version cryptée.

6 Différences avec la méthode d'Aslett

La méthode d'Aslett qui propose des algorithmes tels des classifieurs bayésiens et des variations de random forests fonctionnant sur des données cryptées, diffère de celle présentée ici en plusieurs points :

- Les algorithmes présentés n'ont pas les mêmes capacités prédictives que les réseaux neuronaux sur certaines tâches.
- Utilisation d'une méthode de codage particulière qui requiert à chaque valeur d'être comparée à un "niveau" afin de faire tourner les algorithmes (en particulier les random forests).
- Les CryptoNets ne peuvent pas être entraînés sur des données cryptées.

References

[1] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, J. Wernsings, *CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy.* , 2016.