Revue littéraire - Encrypted Machine Learning

Thomas

September 2017

1 Introduction

1.1 Notation

1.1.1 Clefs

Clef publique: k_p

Clef privée/secrète: k_s

1.1.2 Algorithmes

Encryptage: $Enc(k_p, .)$

Décryptage: $Dec(k_s, .)$

1.1.3 Messages

Soit m le message non crypté, tel que $m \in M$, où M représente l'espace des messages.

Soit c le message crypté, tel que $c \in C$, où C représente l'espace des messages cryptés.

1.2 Définition

On définit un processus de cryptage homomorphique pour l'opérateur $\circ \in \mathcal{F}_M$, si :
$\exists \diamond \in \mathcal{F}_C$, $Dec(k_s, Enc(k_p, m_1) \diamond Enc(k_p, m_2)) = m_1 \circ m_2$, $\forall m_1, m_2 \in M$

Par ailleurs on parle de transformation totalement homomorphique si la définition précédente est vraie pour les opérateurs $+$ et $\times$.

1.3 Limitations

On ne s'intéresse pas aux cas où les espaces M sont uniquement binaires ,i.e $M = \{0, 1\}^T$, pour des raisons de performances. On doit toujours pouvoir avoir le choix d'espace d'entiers ,i.e $\in Z$.

Il faut également être attentif à la taille des messages une fois cryptés c qui peuvent facilement être très volumineux. *Exemple avec un message de 1Mo qui peut passer à 16.4Go après cryptage par FHE en raison du nombre de coefficients des deux polynômes (8 192) représentés par des entiers en 128 bits.*

Enfin cette méthode présente des coûts de calculs élevés, avec des polynômes de degrés très élevés et des coefficients polynomiaux pouvant dépasser le seuil de ce qui est calculable par un cycle d'horloge d'un CPU d'ordinateur.

Il n'est pour le moment pas possible de réaliser des divisions cryptées au sein des espaces "entiers" de message. Dès lors il n'est pas non plus possible d'utiliser les opérateurs de comparaison binaires habituels $(=, <, >)$. Seuls les additions et multiplication sont donc réalisables.

Une limitation sur l'injection de bruit au sein du processus d'encryptage, en plus des contraintes précédentes, restreint cette méthode de cryptage à des polynômes d'entiers de degrés modérés tandis ce que la vitesse d'évaluation demeurera relativement lente comparée à des données non cryptées.

1.4 Représentation des données

1.4.1 Quantification des valeurs réelles

Puisque la plupart des méthodes de FHE se basent sur un espace d'entiers, il est nécessaire de réaliser quelques approximations lorsque l'on manipule des valeurs réelles. Deux méthodes sont alors possibles :

Une première méthode consiste à approximer le réel par un rationnel (un rapport de deux entiers) puis d'annuler le dénominateur en multipliant tout le jeu de donnés par un entier pré-déterminé et finalement arrondir à l'entier le plus proche les résultats obtenus.

Une deuxième méthode, plus directe, propose de fixer un niveau de précision ϕ représentant le nombre de décimales à garder. Puis de multiplier les données par 10^{ϕ} et d'arrondir à l'entier le plus proche.

1.4.2 Quantification vers des données catégorielles ou ordinales

Les méthodes précédentes transformaient les valeurs réelles en entiers, au détriment des performances puisque cela impliquait un nombre important de multiplications. Or il est également possible de de transformer des données continues en des données discrètes, ce qui permettra alors l'utilisation de certains opérateurs de comparaisons, notamment des tests d'égalités.

2 Random Forest dans le cadre d'un FHE

La nature même des algorithmes de random forests implique un recours important aux opérations de comparaisons afin de pouvoir évaluer la qualité du split à chaque niveau. Or ces opérations sont à proscrire dans un cadre FHE. Néanmoins il existe une implémention adaptée de RF (ou ERF) permettant de les utiliser avec des données cryptées aussi bien lors de la phase de prédiction que celle d'apprentissage.

2.1 Représentation des données

Pour pouvoir utiliser cet algorithme les données doivent au préalable être encodées de la manière suivante, afin notamment de pouvoir plus facilement comparer les données.

Soit X une matrice telle que les éléments x_{ij} représentent le $j^{ème}$ prédicteur de la $i^{ème}$ observation. On considère alors une partition du support de la variable j tel que : $\mathcal{K}_j = \{K_1^j, ..., K_m^j\}$ où $x_{ij} \in \bigcup_{k=1}^{m} K_k^i, \forall i, j$ et $K_j^i \bigcap K_k^i = \emptyset, \forall i \neq k$.

Puis x_{ij} est transformé en un indicateur $\tilde{x}_{ijk} \in \{0, 1\}$ $\forall k$ où pour chaque variable j, $\tilde{x}_{ijk} = 1 \Leftrightarrow x_{ij} \in K_k^j$ et $\tilde{x}_{ijl} = 0$ $\forall l \neq k$.

On obtient alors une manière de tester l'égalité de données cryptées puisque ces comparaisons ne deviennent plus que de simples produits scalaires :
$\sum_{\forall k} \tilde{x}_{i1k}\tilde{x}_{i2k} = 1 \Leftrightarrow$ les observations i_1 et i_2 ont la même valeur discrétisée sur la variable j. (1)
$\sum_{k \in K} \tilde{x}_{ijk} = 1 \Leftrightarrow$ l'observation i a une valeur discrétisée sur K. (2)

2.2 Structure de l'algorithme

1. Les variables prédictrices sont choisies à chaque niveau de manière uniforme et aléatoire au sein d'un sous-ensemble de l'échantillon complet de variables. De plus les "splits" sont également choisis uniformément et aléatoirement. Ensuite grâce à (2) et respectivement à (1) on peut déterminer dans quelle branche chacune des variables doit se trouver et combien d'observations se trouvent dans chacune des feuilles de l'arbre.

2. L'étape précédente est répétée indépendamment pour chaque arbre de la forêt en utilisant un échantillon aléatoire de prédicteurs pour chaque arbre. Chaque observation vaut un vote par arbre selon la feuille et la classe auquel elle appartient.

3. Pour la prédiction la même procédure que l'étape précédente peut être employée pour créer un indicateur qui choisit les votes appropriés pour chaque arbre et chaque classe.

Le modèle retourne alors une prédiction encryptée telle que :

$$\hat{v}_c^* \leftarrow CRF(\tilde{x}, \tilde{y}, \tilde{x}^*)$$

Cela représente un nombre de votes pour chaque classe c au sein de l'espace des messages $y_i \in \mathcal{C}$ grâce aux données cryptées de test $\{\tilde{x}, \tilde{y}\}$ et la valeur à analyser $\tilde{x}^*$.
L'utilisateur peut alors décrypter le résultat grâce à sa clef privée :

$$\hat{z}_c^* = Dec(k_s, \hat{v}_c^*)$$

et ainsi obtenir une probabilité empirique sur la prédiction telle que :

$$\hat{p}_c^* = \frac{\hat{z}_c^*}{\sum_{c=1}^{|\mathcal{C}|} \hat{z}_c^*}$$

2.3 Les différences avec les RF classiques

Dans les algorithmes de RF classiques chaque arbre compte un unique "vote" par prédiction, et ce sans tenir compte du nombre de jeu de test présent dans une feuille. Alors qu'ici en raison des contraintes dues au cryptage, l'algorithme compte le nombre total jeu de test de chaque catégorie présent dans la feuille de la prédiction, et ce sommé sur chaque arbre de la forêt.
La difficulté ici pour faire en sorte que cet algorithme respecte la méthode conventionnelle réside dans le fait que convertir le nombre de jeu de test dans une catégorie, au vote pour la classe la plus probable pour chaque arbre n'est pas possible dans le cadre du FHE sans décryptage des donnés. Pour pallier ce problème une approximation stochastique et asympotiquement consistante permet d'obtenir les votes pour chaque arbre, en se basant sur le fait que l'ajustement requis peut être approximé via un processus de Bernoulli approprié, et ce totalement crypté.

Une autre différence ici concerne la calibration des arbres. En effet, il n'y en a pas, ce qui rend les phases d'apprentissages et de tests particulièrement importantes afin de déterminer convenablement les critères pertinents de la classification. La réponse des tests doit donc être visible, c'est la seule étape qui doit être réalisée sans cryptage.

Un avantage des CRF constitue leur capacité à incrémenter leur apprentissage au fur et à mesure que de nouvelles données sont disponibles : d'autant plus que l'ajout de nouvelles données ne nécessite pas de recalculer l'ensemble des arbres puisqu'il ne figure pas d'étapes d'optimisation dans cet algorithme : ajouter des données à traiter à cette méthode s'effectue donc en temps linéaire ($O(n)$).
Par ailleurs il est également possible de paralléliser ce processus notamment en morcelant les données d'une part, puis en formant les arbres à partir de ces blocs de données, et ainsi reformer la forêt complète en ajoutant les arbres réaliser.

2.4 Choix de paramètres

Il est à noter que le nombre d'arbres semblent plus important dans bons nombre de cas que la taille individuelle des arbres. Dès lors des arbres d'une profondeur de 1 semblent être de bons candidats pour les modèles additifs. Une profondeur plus élevée semble seulement être bénéfique pour les données non-linéaires.
La possibilité d'inclure dans ce modèle une fraction stochastique sans biais et cryptée afin de réduire l'effet de shrinkage subit par les "petites" feuilles des arbres semble pouvoir améliorer grandement la performance des CRF, sans pour autant avoir d'impact négatif lorsqu'il n'est pas bénéfique. Le garder par défaut semble donc être une bonne pratique.

3 Classification naïve bayésienne

3.1 Modèle semi-paramétrique

Les classifieurs naïfs de Bayes permettent de résoudre des problèmes de classification en utilisant une approche générative en modélisant les distributions des prédicteurs.
Néanmoins certaines distributions (comme celle Gaussienne) ne peuvent être directement implémentés dans le cadre de FHE. Pour la Gaussienne par exemple, l'utilisation de divisons et d'exponentielles sur des valeurs continues semble rédhibitoire. Cependant une approche semi-paramétrique (qui permet d'éviter de paramétriser les distributions des prédicteurs) permet de contourner le problème :
Le modèle de classification semi-paramétrique Bayésien (SNB) se présente sous la forme d'une fonction logistique pour la frontière de décision de chaque prédicteur, et où chaque régression logistique implique au plus deux paramètres.

3.2 Régression logistique

Il est proposé une approximation de cette méthode via la première itération de l'algorithe de l'IRLS (pondération itérative des moindres carrés).
Bien que cette méthode soit employée dans le cadre précédent du SNB, elle peut également être utilisée seule comme une méthode de classification compatible dans le cadre FHE.

4 Résultats

4.1 Performance de classification

Pour déterminer la performance des différentes méthodes présentées, chacune d'elles sont comparées à d'autres méthodes classiques (non applicables au FHE) et ce, sur plusieurs jeu de données -non cryptés- de difficulté de classification variable.
Le critère pour déterminer la qualité de la classification est l'aire sous la courbe ROC.

Sur les jeux de données relativement simples, les résultats obtenus sont sans surprise assez semblable pour chaque algorithme (hormis le SNB non appairés qui semble parfois sous-performer).
Avec les datasets plus difficiles on remarque que le RF classique (non applicable en FHE) présente les résultats les plus convaincants, même si les méthodes proposés dans le cadre FHE fournit des résultats légèrement supérieurs à la moyenne des méthodes testées.

4.2 Performance et temps de calcul

Les versions **non cryptées** des algorithmes présentés et compatibles dans le cadre du FHE sont capables de gérer de grandes bases de données car les opérations qu'ils entraînent (addition et multiplication) réclament très peu de temps de calcul pour des CPU modernes. A noter que la plupart des algorithmes présentés voient leur complexités augmenter linéairement avec la taille des données (hormis le CRF).

Table 1: Performance et temps de calcul des différents algorithmes

	SNB-paired	CRF, L=1	CRF, L=2	CRF, L=3
Fitt (s)	18.0	12.5	45.2	347
Prediction (s)	7.8	15.1	48.3	353
Memory /enc. value	154 Ko	128 Ko	528 Ko	1 672 Ko

Les tests ont été réalisés sur des machines de 36 coeurs virtuels d'un intel Xeon E5-2666 V3 @ 2.9GHz - voir table 1

References

[1] L. J. M. Aslett, P. M. Esperan c, C. C. Holmes, *Encrypted statistical machine learning: new privacy preserving methods.* , 2015.