Recherche approximative de plus proches voisins efficace et sûre

Benjamin Mathon, Teddy Furon, Laurent Amsaleg, Julien Bringer

Recherche approximative de plus proches voisins efficace et sûre

Benjamin MATHON[1], Teddy FURON[1], Laurent AMSALEG[2], Julien BRINGER[3]

[1]INRIA Rennes [2]IRISA - CNRS
Campus de Beaulieu, 35042 Rennes, France

[3]MORPHO - SAFRAN Group
11 boulevard Gallieni, 92130 Issy-les-Moulineaux, France

benjamin.mathon@inria.fr, teddy.furon@inria.fr
laurent.amsaleg@irisa.fr, julien.bringer@morpho.com

Thème – 2. [**Communication et codage**] : 2.1 Théorie de l'information ; 4. [**Décision et Interprétation**] : 4.6 - Indexation

Problème traité – Ces travaux portent sur les méthodes de recherche de plus proches voisins à grande échelle. Ces dernières permettent de retourner (selon une métrique donnée) les plus proches éléments d'un signal envoyé par un utilisateur dans une grande base de données détenue par un serveur. Récemment, de nouvelles problématiques ont émergé : la sécurité et le respect de la vie privée. L'utilisateur et le serveur ne veulent pas partager leur données (la requête pour l'utilisateur, la base pour le serveur). De plus, le serveur ne peut avoir une base de données en clair car celle-ci peut-être volée par un utilisateur malhonnête.

Originalité – Nous partons d'une méthode de recherche approximative de plus proches voisins (APPV) efficace qui se base sur des distances entre données quantifiées calculées à l'avance : les PQ-codes [1]. Se plaçant dans le modèle "honnête mais curieux" dans lequel le serveur et l'utilisateur suivent le protocole mais sont capables de déduire de l'information sur les données transitées, nous analysons les menaces en terme de fuite d'information pour cette méthode. Pour empêcher une reconstruction du signal requête par le serveur, nous proposons une version des PQ-codes utilisant deux quantificateurs distincts pour le serveur et l'utilisateur. Contrairement aux techniques existantes qui mettent la sécurité et le respect de la vie privée en haut de la liste des besoins au détriment de la rapidité, notre méthode est modérément sûre mais très efficace.

Résultats – Les avantages de notre méthode sont (i) une perte significative d'information de la requête par le serveur (ii) une base de taille fixée (iii) aucune perte dans la qualité de la recherche.

1 Introduction aux PQ-codes

Le cadre considère un propriétaire possédant une collection de n vecteurs dans $\mathbb{R}^d$: $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$. Ces vecteurs sont alors scindés en M tronçons de longueur ℓ. Nous adoptons $d = M\ell$ et notons $\mathbf{x}_i^{(m)} = (\mathbf{x}_i((m-1)\ell+1), \ldots, \mathbf{x}_i(m\ell))$ le m-ième sous-vecteur de $\mathbf{x}_i$. Pour tout $m \in [M] = [\![1, M]\!]$, le propriétaire lance un K-means sur les sous-vecteurs de $\mathcal{X}^{(m)} = \{\mathbf{x}_i^{(m)}\}_{i\in[n]}$. Ce procédé consiste à générer de façon aléatoire K vecteurs dans $\mathbb{R}^\ell$ et d'appliquer l'algorithme de Lloyd-Max jusqu'à convergence. Nous obtenons en sortie un dictionnaire de K centres $\mathcal{C}^{(m)} = \{\mathbf{c}_i^{(m)}\}_{i\in[K]}$ permettant de définir le m-ième quantificateur $Q^{(m)}(.) : \mathbb{R}^\ell \to [K]$:

$$Q^{(m)}(\mathbf{x}^{(m)}) = \arg\min_{i\in[K]} \|\mathbf{x}^{(m)} - \mathbf{c}_i^{(m)}\|, \quad \forall \mathbf{x}^{(m)} \in \mathbb{R}^\ell, \tag{1}$$

où $\|\cdot\|$ représente la distance euclidienne. Le K-means converge vers un minimum local de la distorsion induite par l'erreur de reconstruction $\sum_{\mathbf{x}\in\mathcal{X}^{(m)}} \|\mathbf{x} - Q^{(m)}(\mathbf{x})\|^2$. Le propriétaire applique alors ce procédé sur un ensemble d'entrainement, sous-espace de $\mathcal{X}^{(m)}$. Nous définissons le quantificateur global $Q(\cdot) : \mathbb{R}^d \to [K]^M$ comme le quantificateur produit $Q^{(1)} \times \ldots \times Q^{(m)}$:

$$Q(\mathbf{x}) = (Q^{(1)}(\mathbf{x}^{(1)}), \ldots, Q^{(M)}(\mathbf{x}^{(M)})), \quad \forall \mathbf{x} \in \mathbb{R}^d, \tag{2}$$

et $Q^{-1}(\cdot) : [K]^M \to \mathbb{R}^d$, l'opérateur qui affecte à une séquence d'indices la concaténation de centres :

$$Q^{-1}((k_1, \ldots, k_M)) = \left(\mathbf{c}_{k_1}^{(1)\top} \ldots \mathbf{c}_{k_M}^{(M)\top}\right)^\top. \tag{3}$$

La base de données $\mathcal{Q} = \{Q(\mathbf{x}_i)\}_{i\in[n]}$ et l'ensemble des M dictionnaires $\mathcal{C} = \{\mathcal{C}^{(m)}\}_{m\in[M]}$ est alors envoyé au serveur. Le rôle du propriétaire est terminé. Le serveur pré-calcule les distances entre centres du même dictionnaire :

$$d_s(i,j,m) = \|\mathbf{c}_i^{(m)} - \mathbf{c}_j^{(m)}\|^2, \forall (i,j,m) \in [K]\times[K]\times[M]. \tag{4}$$

La matrice d_s sera utilisée comme une table de recherche. Lorsque le serveur reçoit une requête $\mathbf{q}$ d'un utilisateur, il calcule en premier lieu le quantifié $Q(\mathbf{q})$. La recherche APPV est ensuite basée sur la distance approchée :

$$\hat{D}(\mathbf{q},\mathbf{x}_i) = \|Q^{-1}(Q(\mathbf{q})) - Q^{-1}(Q(\mathbf{x}_i))\|^2, \tag{5}$$

à la place de la vraie distance $\|\mathbf{q}-\mathbf{x}_i\|^2$. Ce calcul est effectué rapidement grâce à la table de recherche :

$$\hat{D}(\mathbf{q},\mathbf{x}_i) = \sum_{m=1}^{M} d_s(Q^{(m)}(\mathbf{q}^m), Q^{(m)}(\mathbf{x}_i^m), m). \tag{6}$$

Le serveur envoie à l'utilisateur les indices $(i_1,\ldots,i_k)$ correspondant aux k plus petites distances approchées.

2 Analyse des menaces côté serveur

Nos travaux adoptent le modèle "honnête mais curieux" dans lequel le serveur et l'utilisateur suivent le protocole mais sont capables de déduire de l'information sur les données qu'ils reçoivent. Plus précisément, le serveur curieux [1] peut vouloir :

S_1 Reconstruire $\mathbf{x}_i$ à partir de $Q(\mathbf{x}_i)$,

S_2 Créer des groupes parmi les vecteurs de la base à partir de $Q(\mathbf{x}_i)$ (par recherches APPV parmi ces vecteurs),

S_3 Reconstruire la requête $\mathbf{q}$ à partir des informations données par l'utilisateur,

S_4 Détecter les requêtes similaires.

Nous insistons sur le fait que ce travail ne propose pas de méthode qui se prémunit contre toutes ces menaces. Nous présentons dans la section suivante une méthode modérément sûre qui fait le compromis entre l'efficacité de la recherche APPV et la protection contre ces attaques.

3 Notre méthode

L'idée principale est ici de renforcer le scénario S_3 par l'introduction de deux quantificateurs. Le propriétaire génère hors-ligne $\mathcal{C}_S$, un ensemble de M dictionnaires de K_S centres chacun. Cela définit le quantificateur produit $Q_S(\cdot)$ utilisé pour générer la base $\mathcal{Q} = \{Q_S(\mathbf{x}_i)\}_{i=1}^n$ qui sera ensuite envoyée au serveur. Seul le propriétaire connait $\mathcal{C}_S$.

Le propriétaire génère également $\mathcal{C}_U$, un ensemble de M dictionnaires de K_U centres chacun, qui définit le quantificateur $Q_U(\cdot)$. $\mathcal{C}_U$ sera ensuite envoyé à l'utilisateur pour quantifier sa requête $\mathbf{q}$. Le propriétaire calcule les distances :

$$d_{us}(i,j,m) = \|\mathbf{c}_{U,i}^{(m)} - \mathbf{c}_{S,j}^{(m)}\|^2, \forall (i,j,m) \in [K_U]\times[K_S]\times[M], \tag{7}$$

et envoie cette table de recherche au serveur.

En ligne, l'utilisateur récupère $\mathcal{C}_U$, envoie $Q_U(\mathbf{q})$ au serveur qui effectue la recherche APPV grâce à d_{us}. Notons que les deux quantificateurs peuvent ne pas avoir le même nombre de centres par sous-espace. Il est important d'avoir un K_S raisonnable en raison de l'empreinte mémoire de $\mathcal{Q}$ qui est de $nM\log_2 K_S$ bits côté serveur. Une valeur de K_U plus grande améliore la qualité de la recherche APPV tandis que la capacité de la transmission entre l'utilisateur et le serveur, $M\log_2 K_U$, augmente légèrement.

Le serveur ne peut pas reconstruire les vecteurs de la base (menace S_1) car il ne connait pas le dictionnaire $\mathcal{C}_\mathcal{S}$. Idem pour les vecteurs requêtes (menace S_3) car il ne possède pas $\mathcal{C}_U$. Notons que ces affirmations restent correctes tant qu'il n'existe pas de coalition entre le serveur et l'utilisateur et tant que le serveur n'usurpe pas le rôle de l'utilisateur (ces deux cas sortent du modèle "honnête mais curieux"). Le serveur peut détecter les requêtes similaires $\mathbf{q}$ et $\mathbf{q}'$ ($Q_U(\mathbf{q}) \approx Q_U(\mathbf{q}')$, menace S_4). Cependant, il peut difficilement jauger cette différence car il ne possède pas la table des distances entre les centres de $\mathcal{C}_U$.

4 Expériences

Nos tests sont effectués sur la base de descripteurs SIFT locaux *ANN_SIFT1M* [1]. Cette base contient (i) une base de $1,000,000$ vecteurs de dimension $d = 128$, (ii) $100,000$ vecteurs d'entrainement pour générer les K-means, (iii) $10,000$ vecteurs requête et un fichier de confiance qui contient, pour chaque requête, les identifiants de ses plus proches voisins triés par ordre croissant des distances.

1. Ce résumé ne prend pas en compte les menaces côté utilisateur.

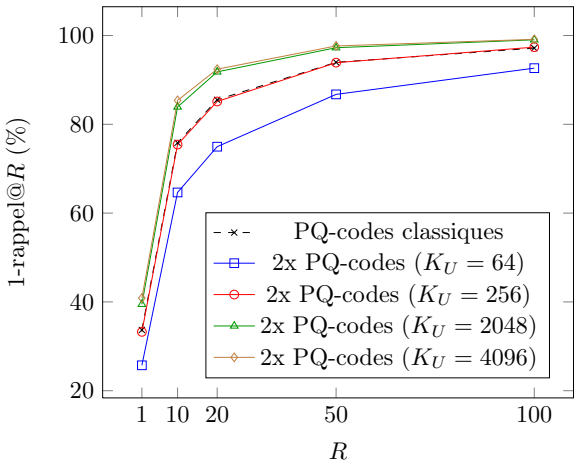

FIGURE 1 – Scores 1-rappel@R pour la version originale et proposée des PQ-codes : $M = 16$, $l = 8$, $K_S = 256$, $N_i = 50$.

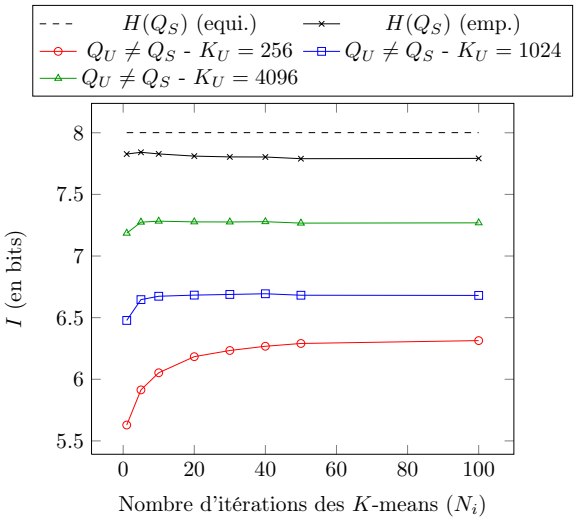

FIGURE 2 – Information mutuelle moyenne empirique entre Q_S and Q_U en fonction du nombre d'itérations du procédé K-means.

4.1 Qualité de la recherche

Les PQ-codes réalisent une recherche APPV, ce qui signifie que les plus proches voisins retournés ne sont pas forcément les bons. Pour mesurer la qualité de cette recherche, le premier rappel au rang R, noté "1-rappel@R" [1] est calculé. Cette quantité représente la probabilité pour que le premier vrai plus proche voisin soit contenu dans les R vecteurs retournés. La figure 1 montre les 1-rappel@R en pourcentages. Côté serveur, les PQ-codes sont calculés avec $M = 16$, $l = 8$, $K_S = 256$ et $N_i = 50$, le nombre d'itérations du procédé K-means. La courbe en tirets montre les performances des PQ-codes classiques. En bref, la recherche retourne le plus proche voisin de manière quasi-certaine pour $R = 100$. Nous augmentons le nombre de centres pour le quantificateur utilisateur (de $K_U = 64$ à 4096) pour une meilleur qualité de recherche lorsque $K_U > K_S$.

4.2 Analyse de la menace S_2

Pour un vecteur donné de la base, le serveur curieux connait $Q_S(\mathbf{x}_i)$ alors qu'il a besoin de $Q_U(\mathbf{x}_i)$ pour calculer la distance approchée entre ce vecteur et les autres entrées de $\mathcal{Q}$ par le biais de d_{us}. C'est pourquoi nous calculons la quantité d'information moyenne qu'il manque au serveur pour pouvoir utiliser d_{us} :

$$I(Q_S; Q_U) = M^{-1} \sum_{m=1}^{M} I(Q_S^{(m)}(\mathbf{X}_i^{(m)}); Q_U^{(m)}(\mathbf{X}_i^{(m)})). \tag{8}$$

La figure 2 montre cette quantité en fonction du nombre d'itérations du procédé K-means, calculée empiriquement sur la base ANN_SIFT1M. La courbe en tirets représente l'entropie de $Q_S^{(m)}(\mathbf{X}^{(m)})$ quand la quantification d'un vecteur est distribuée de façon équiprobable ($\log_2(K_S)$). La courbe en noir (marqueurs en croix) représente l'estimation de cette entropie (calculée empiriquement), qui est plus faible. Ceci s'explique par le procédé K-means, dont le but n'est pas d'assurer l'équiprobabilité de la distribution mais de minimiser l'erreur quadratique moyenne. $I(Q_S; Q_U)$ augmente selon K_U, sans atteindre la valeur de l'entropie. Toutefois, il manque au serveur curieux une quantité d'information de l'ordre de $M.(H(Q_S) - I(Q_S; Q_U))$ bits par entrée de la base pour calculer les distances approchées entre entrées de $\mathcal{Q}$. La fuite d'information augmente avec K_U mais améliore la qualité de la recherche APPV (figure 1). Nous pouvons voir ici le prix à payer pour plus de sécurité. Notons que plus le nombre d'itérations N_i du procédé K-means est élevé, moins on observe de différences entre $\mathcal{C}_U$ et $\mathcal{C}_S$. Un faible nombre d'itération permet alors de rendre le système plus sûr alors que la distorsion induite par l'erreur de reconstruction n'est pas optimale. Toutefois, nous avons constaté que la qualité de la recherche n'est pas affectée tant que $N_i \geq 3$.

Références

[1] H. JÉGOU, M. DOUZE et C. SCHMID : Product quantization for nearest neighbor search. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 33(1):117 –128, jan. 2011.