Computer Security Symposium 2013
21 - 23 October 2013

暗号化ベクトルデータベースのための索引構造

川本 淳平 †

† 筑波大学システム情報系
305-0006 茨城県つくば市天王台 1-1-1
junpei.kawamoto@acm.org

あらまし 本論文では，暗号化ベクトルデータベースにおいて類似ベクトルを検索する際の検索結果候補を削減するフィルタリング手法を提案する．提案手法では，局所性検知可能ハッシュと白色化変換を用いる．LSH はベクトル間の類似度を効率的に近似するデータ構造であり，ベクトル空間をいくつかの部分空間に分割する．本論文では LSH を用いて類似度の小さいベクトルを検索対象から外すフィルタを提案している．しかし，検索対象となるベクトルが局所的に存在している場合 LSH の近似精度が下がる問題がある．そこで提案手法では白色化変換を用いてベクトル間の偏りを低減している．その結果，暗号化ベクトルデータベース内のベクトルに偏りがある場合であっても効率的にフィルタリングを行うことができる．

Indexing Methods for Encrypted Vector Databases

Junpei Kawamoto†

†Faculty of Engineering, Information and Systems, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-0006, JAPAN
junpei.kawamoto@acm.org

Abstract We introduce a filtering methodology based on locality sensitive hashing (LSH) and whitening transformation to reduce candidate tuples which encrypted vector databases (EVDBs) must compute similarity between for query processing. LSH is a hashing methodology which is efficient for estimating similarities between two vectors. It hashes a vector space using randomly chosen vectors. We can filter vectors which are less similar to the querying vectors by recording which hashed space each vector belongs to. However, if vectors in EVDBs are found locally, then most vectors are in a same hashed space and so the filter will not work. Since we can treat those cases using whitening transformation to distribute the vectors broadly, our proposal filtering methodology will work effectively on any vector space. We also show that the server's query processing cost is reduced by our filter.

1 はじめに

暗号化ベクトルデータベース は **ベクトルデータベース** をより安全にしたものである．ベクトルデータベースは，キーベクトル $\mathbf{k}$ と対応する値 v の組 $(\mathbf{k}.v)$ を 1 レコードとして保存するデータベースである．そして，問合せとしてクエリベクトルを受け取り，そのクエリベクトルとの類似しているキーベクトルを持つレコードを返す機能を提供している．ベクトルデータベースの例として，画像データベースや文献データベースが考えられる．各画像の特徴ベクトルを

キーベクトルとし画像自身を値とするようなタプルを保存する画像データベースや，各文献の特徴ベクトルをキーベクトルとし文献自身を値とするタプルを保存する文献データベースはベクトルデータベースである．データベース所有者がベクトルデータベースをクラウドサービス等に公開し共同作業のために同僚と共有しようと思ったとすると，ベクトルデータベースのための暗号化が必要となる．一般に，データベース所有者はクラウドサービスを用いることでデータベース管理コストや共有のためのコストを削減できると期待している．しかし，所有者がクラウドサービスを完全には信用しておらずデータベース内の情報をクラウドサービスに対しても秘密にしておきたいという状況は容易に想像できる．共同作業者すなわちデータベース利用者も同様にその問合せを秘密にしておきたい場合もある．サーバに保存するすべてのデータをクラウドサーバに公開する前に暗号化してしまい，問合せも復号することなく処理することの出来る暗号化ベクトルデータベースはこうした状況で有効である．暗号化ベクトルデータベースで用いられる暗号は，暗号化後もキーベクトルと問合せベクトル間の類似度が保存するような特別な機能を持っているものが用いられている．そのため，クラウドサービスは利用者から送られた暗号化問合せベクトルに類似した暗号化キーベクトルを，それらが暗号化されてない時と同様に求めることが出来る．

暗号化ベクトルデータベースに保存されるすべてのキーベクトルは予め暗号化されており，暗号化前のベクトルが持つ構造が暗号化後にも保たれているとは期待できない．これは，暗号化ベクトルデータベースにおいて検索用索引の構築が困難であることを意味する．特に，R-木 [1] などの構造情報を用いた索引の構築は有効ではない．実際，既存の暗号化ベクトルデータベースでは問合せ処理のために **すべて** のデータにアクセスする必要がある．問合せ自身も暗号化されており，同じ問合せであっても暗号化によって異なる問合せへと変換されるため，問合せ結果を保存し再利用することも難しい．こうした理由により，暗号化ベクトルデータベースの問合せ処理は大きな計算コストを必要とする．

本論文では，**局所性検知可能ハッシュ** *(LSH; locality sensitive hashing)* [2] 及び **白色化変換** を用いて暗号化ベクトルデータベースにおいてサーバが問合せ処理のために調べる必要のあるタプル数を削減するフィルタを提案する．LSH はベクトルの類似度を基にしたフィルタに適したデータ構造であり，また様々な用途で利用されている [3, 4]．LSH はベクトル空間をランダムに選んだ **基準ベクトル** を基にいくつかの部分空間へ分割する．これを用いて，問合せベクトルとの類似度が小さい部分空間に含まれるキーベクトルを検索対象から取り除くことで検索効率を向上させるフィルタを構築する．しかし，対象の暗号化ベクトルデータベース内のキーベクトルが局所的に存在している場合，多くのベクトルが単一の部分空間に含まれてしまうような場合では，LSH は効率的にフィルタリング出来ない．そこで，ベクトルを空間内に均等に分散させる白色化変換を用いてこのような状況であっても効率的なフィルタを作成する．

2 基本事項

本論文では，データベース所有者が持つベクトルデータベースを $VDB(\mathbf{Key}, Value)$ と書く．**Key** はベクトルからなるキー属性であり，$Value$ は各キー属性値に関連つけられる値属性である．例えば，画像データベースでは **Key** は画像の特徴ベクトルであり $Value$ は画像そのものとなる．文献データベースでも **Key** は各文献の特徴ベクトルであり $Value$ は文献そのものとなる．データベースの利用者は，キー属性に対してのみ問合せを行うものとし，各問合せはキー属性値のベクトルが問合せのベクトルと閾値 α 以上の類似度を持つようなタプルの検索であるとする．言い換えれば，利用者からの問合せは問合せベクトル $\mathbf{q}$ に対して $\mathrm{sim}(\mathbf{k}, \mathbf{q}) \geq \alpha$ という条件を満足するキーベクトル $\mathbf{k}$ を持つタプルの検索である．簡単のために，本論文では類似度としてコサイン類似度を考え，各ベクトルは正規化されているのもとする．

上記ベクトルデータベースに対応する暗号化

ベクトルデータベースを $EVDB(\mathbf{Key}_e, Value_e)$ と書く．$\mathbf{Key}_e$ はキー属性に対応する暗号化キー属性であり $Value_e$ は暗号化された値属性である．暗号化前のベクトルデータベースにおけるキー属性値 $\mathbf{k} \in \mathbf{Key}$ に対応する暗号化キー属性の値 $\mathbf{k}_e \in \mathbf{Key}_e$ は，キーベクトルに対する暗号化アルゴリズム Enc_k を用いて $\mathbf{k}_e = \mathrm{Enc}_k(\mathbf{k})$ となる．各値属性値 $v \in Value$ を暗号化した $v_e \in Value_e$ は，暗号化アルゴリズム Enc_v を用いて $v_e = \mathrm{Enc}_v(v)$ となる．データベース所有者は，この暗号化ベクトルデータベースをクラウドサービスに配置し平文のベクトルデータベースは秘密に管理する．以降では，暗号化ベクトルデータベースが配置されたクラウドサービスを単純にデータベースサーバと呼ぶ．暗号化ベクトルデータベースでは，利用者からの問合せも暗号化されている．利用者が問合せベクトル $\mathbf{q}$ と閾値 α をデータベースサーバに送ろうとした場合，実際には問合せベクトル用の暗号化アルゴリズム Enc_q を用いた暗号化問合せベクトル $\mathbf{q}_e = \mathrm{Enc}_q(\mathbf{q})$ と閾値 α が送信されることになる．データベースサーバは，条件 $\mathrm{sim}(\mathbf{k}_e, \mathbf{q}_e) \geq \alpha$ を満足するタプル $(\mathbf{k}_e, v_e)$ の集合を計算し問合せ結果として利用者に返信する．この問合せ処理において，データベースサーバは $\mathbf{k}, \mathbf{q}$ 及び v を $\mathbf{k}_e, \mathbf{q}_e$ 及び v_e から計算できないことが保証されている．最後に，問合せ結果としてタプル $(\mathbf{k}_e, v_e)$ の集合を受け取った利用者は，キーベクトル用の復号アルゴリズム Dec_k と値用の復号アルゴリズム Dec_v を用いて，平文のキーベクトルと値をそれぞれ $\mathbf{k} = \mathrm{Dec}_k(\mathbf{k}_e)$ 及び $v = \mathrm{Dec}_v(v_e)$ により取得する．

この一連のプロトコルを実行するために，データベース所有者は Enc_k, Enc_q, Enc_v, Dec_k 及び Dec_v を用意し，Enc_q, Dec_k 及び Dec_v をデータベースの利用者に通知する．なお，データベースサーバにはこれらの暗号化及び復号アルゴリズムは秘密にしておく．

3 LSH フィルタ

本論文では，暗号化ベクトルデータベースにおいて，類似ベクトル検索のためのフィルタを提案する．暗号化ベクトルデータベースでは，ベクトルは暗号化されておりその構造は平文ベクトルのものとは異なる．暗号化ベクトルデータベースにおける既存プロトコルでは，暗号化キーベクトルと暗号化問合せベクトルの類似度のみ暗号化前と同じを保っており，そのため問合せ処理が正しく実行できる．しかし，二つの暗号化キーベクトル間の類似度や二つの暗号化問合せベクトル間の類似度など，その他の演算については暗号化前とは異なった値となる．我々は，暗号化の前後でキーベクトルと問合せベクトルの類似度は変化しないという特徴を用いてフィルタを作成する．局所性検知可能ハッシュ (LSH) は，この目的に適したデータ構造である．

本節では，まず LSH フィルタに用いる LSH と白色化変換という二つの概念を説明する．その後，LSH フィルタの作成方法と暗号化ベクトルデータベースへの適用方法について説明する．

3.1 局所性検知可能ハッシュ

LSH には様々な種類が提案されている [5, 2, 6] が，本論文ではコサイン類似度の計算に適している Charikar の手法 [2] を採用する．この LSH では m 個の異なるハッシュ関数を使用する．それぞれのハッシュ関数 h_i は一つの基準ベクトル $\mathbf{b}_i$ から生成され，

$$\mathrm{h}_i(\mathbf{v}) = \begin{cases} 1; & \mathbf{v} \cdot \mathbf{b}_i \geq 0 \\ 0; & \text{otherwise} \end{cases}$$

と定義される．$\mathbf{v}$ と $\mathbf{b}_i$ は同じ次元のベクトルでなければ成らない．この m 個のハッシュ関数を用いて，ベクトル $\mathbf{v}$ に対する LSH の値 $\mathrm{lsh}(\mathbf{v})$ はそれぞれのハッシュ関数の値からなるタプルとして定義される．すなわち，

$$\mathrm{lsh}(\mathbf{v}) = (\mathrm{h}_1(\mathbf{v}), \mathrm{h}_2(\mathbf{v}), \cdots, \mathrm{h}_m(\mathbf{v})) . \quad (1)$$

二つのベクトル $\mathbf{u}$, $\mathbf{v}$ に対する LSH 値 $\mathrm{lsh}(\mathbf{u})$, $\mathrm{lsh}(\mathbf{v})$ の間には，次のような関係がある．$\Pr[\mathrm{lsh}(\mathbf{u}) = \mathrm{lsh}(\mathbf{v})] \approx 1 - \theta(\mathbf{u}, \mathbf{v})/\pi$. $\Pr[\mathrm{lsh}(\mathbf{u}) = \mathrm{lsh}(\mathbf{v})]$ は $\mathrm{lsh}(\mathbf{u})$, $\mathrm{lsh}(\mathbf{v})$ の中でいくつ同じハッシュ値があるか，すなわち，$\mathrm{h}_i(\mathbf{u}) = \mathrm{h}_i(\mathbf{v})$ を満足する i の個数を表している．また，$\theta(\mathbf{u}, \mathbf{v})$ は二つの

ベクトルのなす角を表す．この近似式は，二つのベクトルのなす角が LSH 値を用いて近似できることを表している．よって，二つのベクトルのコサイン類似度 $\cos(\mathbf{u}, \mathbf{v})$ も，

$$\cos(\mathbf{u}, \mathbf{v}) \approx \cos\left(\pi(1 - \Pr[\mathrm{lsh}(\mathbf{u}) = \mathrm{lsh}(\mathbf{v})])\right) \tag{2}$$

と近似的に求めることができる．

LSH は m 個の基準ベクトルを用いてベクトル空間を 2^m 個の部分空間に分割する．この基準ベクトルは通常ランダムに選択されるため，本論文でもランダムに選択し暗号化キーベクトルや暗号化問合せベクトルの分布は考慮しないものとする．近似式 (2) の正確さは基準ベクトルの個数 m と基準ベクトル $\mathbf{b}_i$ と対象のベクトル $\mathbf{v}$ との関係に影響される．したがって，LSH を用いて暗号化ベクトルデータベースのためのフィルタを構築するためには，LSH によって得られるコサイン類似度の精度が m と暗号化キーベクトル及び暗号化問合せベクトルの分布に影響されることを考慮する必要がある．つまり，暗号化されたベクトルが対象のベクトル空間中に偏って存在していた場合，LSH はそのベクトル空間を効果的な部分空間に分割することができない．そのため，暗号化キーベクトルや暗号化問合せベクトルをベクトル空間中に分散させる必要がある．本論文では白色化変換を用いてこの問題を解決する．

3.2 白色化変換

LSH を用いて効率的に類似ベクトルを発見するためには，ベクトル空間をランダムベクトルを用いて効果的に分割する必要がある．多くのベクトルが一つの部分空間に偏って含まれる場合，LSH はベクトルを選別できない．すべてのベクトルが一つの部分空間に含まれる最悪の場合では，LSH は何の効果もない．このような場合を回避するために，ベクトル間の相関を低減する白色化変換を利用する．

ベクトルの白色化変換に用いる白色化行列を作成するためには，まず，すべてのベクトルから次の共分散行列 Σ を計算する．

$$\Sigma = E\left((\mathbf{v} - \mu)(\mathbf{v} - \mu)^T\right) \tag{3}$$

μ はベクトル $\mathbf{v}$ の平均ベクトルである．次に，この共分散行列 Σ を $\Sigma = \Phi\Lambda\Phi^{-1}$ と分解する．Φ は i 列目が Σ の i 番目の固有ベクトルからなる正方行列であり，Λ は 対応する固有値を並べた対角行列である．白色化行列 W_k は

$$W_k = \Phi\Lambda^{-1/2}. \tag{4}$$

任意のベクトル $\mathbf{v}$ に対して，その白色化されたベクトル $\mathbf{v}_w$ は $\mathbf{v}_w = W_k^T(\mathbf{v} - \mu)$ である．このようにして計算される白色化ベクトルに対する共分散行列は．$E(\mathbf{v}_w\mathbf{v}_w^T) = E(W_k^T(\mathbf{v}-\mu)(\mathbf{v}-\mu)^T W_k) = E(\Lambda^{-1/2}\Phi^T\Sigma\Phi\Lambda^{-1/2}) = I$ であり，白色化されたベクトルは無相関となる．

本論文では，この白色化変換を用いてキーベクトルの空間を偏りの少ない空間へ変換した後，LSH を構築するために部分空間へ分割する．その結果，キーベクトル間に偏りがあるような場合であっても，効率的な LSH フィルタが構成できると期待できる．この白色化変換の効果については 4 節にて評価する．

3.3 暗号化ベクトルデータベースへの適用

既存の暗号化ベクトルデータベースプロトコルは，第 2 節にて紹介した様に，Enc_k，Enc_q，Enc_v，Dec_k 及び Dec_v という五つの暗号化及び復号アルゴリズムを定義している．LSH フィルタでは，これら五つのアルゴリズムは定義済みであるとする．そして，Enc_k，Enc_q 及び Dec_k の三アルゴリズムを拡張してフィルタに関する操作を付け加えた Enc_k^*，Enc_q^* 及び Dec_k^* を定義する．言い換えれば，LSH フィルタは上記のアルゴリズムを定義している既存のプロトコルと共に用いることができる．本節では，先ずこれらの拡張された暗号化及び復号アルゴリズムを説明した後，フィルタについて議論する．

データベースの所有者がベクトルデータベース VDB を管理しているとする．そして，所有者はこのデータベースを暗号化と我々の LSH フィルタを用いてクラウドサービス上に公開しようとしているとする．データベース所有者は，先ずすべてのキーベクトル $\mathbf{k} \in \mathbf{Key}$ を暗号化し，暗号化キーベクトルの平均ベクト

ル μ を計算する．この平均ベクトルを元に，所有者は暗号化キーベクトルの共分散行列を計算する．すなわち，式 (3) を用いて，$\Sigma = E\left((\mathrm{Enc}_k(\mathbf{k}) - \mu)(\mathrm{Enc}_k(\mathbf{k}) - \mu)^T\right)$ となる．次に，この共分散行列 Σ を $\Sigma = \Phi\Lambda\Phi^{-1}$ と分解し，式 (4) にて紹介した白色化行列 W_k を計算する．この白色化行列を用いて，拡張されたキーベクトル用の暗号化アルゴリズム $\mathrm{Enc}_k^*(\mathbf{k})$ を $\mathrm{Enc}_k^*(\mathbf{k}) = W_k^T(\mathrm{Enc}_k(\mathbf{k}) - \mu)$ とする．その他の拡張されたアルゴリズム， すなわち問合せベクトル用の暗号化アルゴリズム Enc_q^* とキーベクトル用の復号アルゴリズム Dec_k^* も白色化行列を用いて $\mathrm{Enc}_q^*(\mathbf{q}) = W_k^{-1}\mathrm{Enc}_q(\mathbf{q})$, $\mathrm{Dec}_k^*(\mathbf{k}_e) = \mathrm{Dec}_k((W_k^T)^{-1}\mathbf{k}_e + \mu)$ と定める．データベース所有者は，暗号化前のベクトルデータベース VDB を上記アルゴリズムを用いて暗号化ベクトルデータベース $EVDB$ へと変換し公開する．併せて，データベースの利用者に Enc_q^*，Dec_k^*，Dec_v 及び μ を通知する．

この拡張されたアルゴリズムを用いた問合せは次のようなる．ある利用者が問合せベクトル $\mathbf{q}$ に閾値 α で類似しているキーベクトルを持つタプルを検索する，つまり $\mathrm{sim}(\mathbf{k}, \mathbf{q}) \geq \alpha$ を満たすキーベクトル $\mathbf{k}$ を検索する場合を考える．類似度としてコサイン類似度を採用しており，各ベクトルは正規化されていると仮定していたので，$\mathbf{k} \cdot \mathbf{q} \geq \alpha$ を考えれば良い．この問合せは暗号化ベクトルデータベース用に変換され，$\mathbf{k}_e^* \cdot \mathbf{q}_e^* \geq \alpha - \mu \cdot \mathrm{Enc}_q(\mathbf{q})$ を満足する暗号化キーベクトル $\mathbf{k}_e^* = \mathrm{Enc}_k^*(\mathbf{k})$ の探索となる．$\mathbf{q}_e^* = \mathrm{Enc}_q^*(\mathbf{q})$ であるから，利用者はこの暗号化問合せベクトル $\mathbf{q}^*$ と閾値 $\alpha^* = \alpha - \mu \cdot \mathrm{Enc}_q(\mathbf{q})$ を問合せとしてサーバへ送信する．このとき，提案暗号化アルゴリズムによる暗号化キーベクトルと暗号化問合せベクトルに対する条件式 $\mathbf{k}_e^* \cdot \mathbf{q}_e^* \geq \alpha - \mu \cdot \mathrm{Enc}_q(\mathbf{q})$ は，本来の条件式 $\mathbf{k} \cdot \mathbf{q} \geq \alpha$ に満足するタプルを取得する．

LSH フィルタを用いた検索機能を提供するために，データベースサーバは m 個のランダムベクトルを作成し，式 (1) で定義されるハッシュ関数を用意する．そしてデータベース所有者が公開したデータベースを次の手順で変換する．先ず，サーバは暗号化ベクトルデータベース内の各暗号化キーベクトル $\mathbf{k}_e^*$ に対してその LSH 値 $\mathrm{lsh}(\mathbf{k}_e^*)$ を計算する．そして，それぞれのタプルにその LSH 値を追加する．そのため，サーバは $EVDB^*(LSH, \mathbf{Key}_e, Value_e)$ というスキーマで所有者が公開したベクトルデータベースを保存する．LSH は LSH 値を保存している属性である．また，S_{lsh} でサーバが保持しているタプルに付けられた LSH 値の集合を表す．サーバはこの集合を簡単に計算できる．

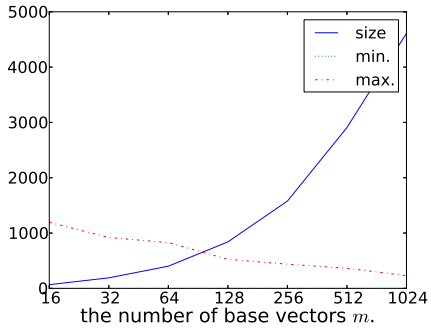
サーバにおける提案フィルタを用いた問合せ処理は次のようになる．ある利用者がデータベースに問合せを送信すると，サーバは暗号化問合せベクトル $\mathbf{q}_e^*$ と閾値 α^* を受け取ることになる．閾値は上で説明したように $\alpha^* = \alpha - \mu \cdot \mathrm{Enc}_q(\mathbf{q})$ となる．次に，サーバはこの問合せベクトルに対する LSH 値 $h_q = \mathrm{lsh}(\mathbf{q}_e^*)$ を計算する．この LSH 値を用いて，サーバは次の条件を満足する S_{lsh} の部分集合 $S_{cand} \subseteq S_{lsh}$ を求める．その条件とは，S_{cand} に含まれるすべての LSH 値 $h \in S_{cand}$ が，

$$\cos\left(\pi(1 - \Pr[h = h_q])\right) \geq \alpha^* \qquad (5)$$

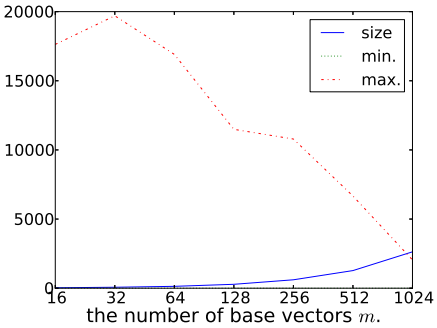
を満たすことである．式 (2) より，暗号化問合せベクトル $\mathbf{q}_e$ と条件 (5) を満足する LSH 値を持つ暗号化キーベクトルとの類似度は，α^* 以上であると近似できる．言い換えれば，S_{cand} に含まれない LSH 値を持つタプルの暗号化キーベクトルは暗号化問合せベクトルとの類似度が閾値未満であると期待される．そのため，サーバは $\mathbf{q}_e$ と LSH 属性の値が S_{cand} に含まれていないタプルの暗号化キーベクトルとの内積を計算せずに済む．これが本提案手法におけるフィルタリング効果である．最後に，サーバは S_{cand} に含まれている LSH 属性値を持つタプルに対し，実際に本来の条件 $\mathbf{k}_e \cdot \mathbf{q}_e \geq \alpha^*$ を満足しているか否かを調べ問合せ結果を作成する．問合せ結果は利用者へと返却される．

4 評価実験

提案フィルタの有効性を検証するため評価実験を行った．ここでは，IPP 法 [7] を適用した暗号化ベクトルデータベースを用いた．この暗

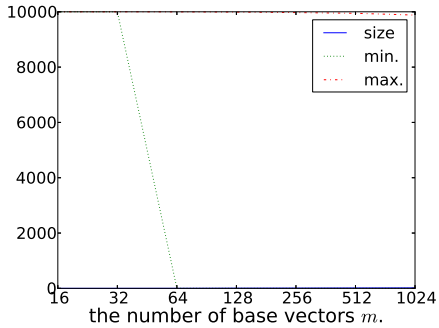


(a) $n = 10000$

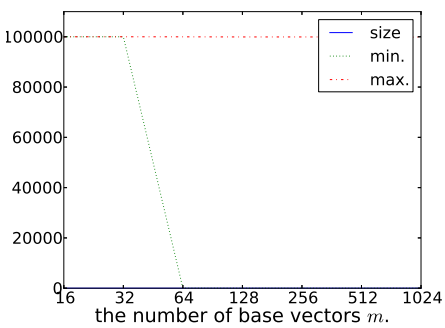


(b) $n = 100000$

図 1: 異なる LSH 値の総数と同じ LSH 値を持つベクトル個数の最大及び最小値.



(a) $n = 10000$



(b) $n = 100000$

図 2: 白色化変換を用いない場合の異なる LSH 値の総数と同じ LSH 値を持つベクトル個数.

号化ベクトルデータベースの総タプル数を n と書く．そして，1) 暗号化キーベクトルの空間が LSH によっていくつの部分空間に分割され各部分空間にはいくつのベクトルが配置されるのか．2) LSH と問合せ結果の再現率との関係，3) LSH と問合せ処理速度との関係について評価した．実験に用いたプログラムはすべて Python 2.7 で実装し，Intel®Core™i7-860 Processor (8M Cache, 2.80 GHz), 8GB RAM で OS が Ubuntu 12.04 LTS である計算機上で実行した．

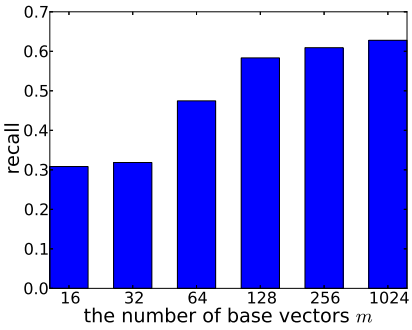
図 1 は，$n = 10000$ タプルと $n = 100000$ タプルからなる暗号化ベクトルデータベースにおける LSH 値を示している．各図の横軸は，LSH における基準ベクトルの個数 m を示している．*size* のグラフはそれぞれの暗号化ベクトルデータベースにおいて，式 (1) で定義される LSH 値が何種類存在したのかを表している．すなわち，LSH によって暗号化キーベクトルの空間が何種類の部分空間に分割されたのかを示している．*min.* と *max.* のグラフは，その部分空間に割り当てられた暗号化キーベクトルの最小数と最大数を示している．*max.* の値が大きい場合，多くの暗号化キーベクトルが一つの部分空間に割り当てられていることを示しており，LSH が効果的にベクトル空間を分割 **できていない** ことを意味する．つまり我々のフィルタも効果的に動作していないことを示している．一方，*min.* の値は，LSH によって識別できるベクトルの最小数を意味している．図 1 では，どちらの場合も最小数は 1 であった．図 1 によると，LSH の基準ベクトル数 m が大きい場合 LSH 値の種類も大きく，LSH はキーベクトルを細かく識別できている．また m が大きければ各部分空間に属するベクトル数の最大値も小さくなっている．このことからも，m を大きくすれば LSH の効果が大きくなることを示している．この傾向は総タプル数 n に関係なかった．

図 2 は，図 1 の場合と同じ条件で白色化変換を利用しない場合についての結果を示したものであり，提案フィルタにおける白色化変換の

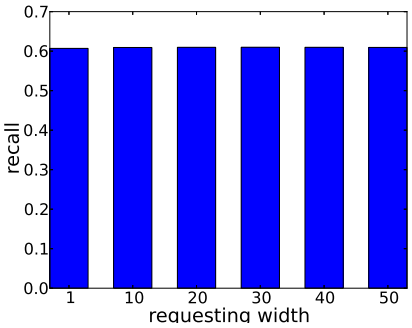
有効性を示したものである．白色化変換を用いない場合，異なる LSH 値の種類（*size*）は小さいく，具体的には基準ベクトルの個数 m とは無関係にほとんどの場合で 1 であった．これは LSH によってキーベクトルの空間が効果的に分割されておらず，LSH がキーベクトルを分類できないことを示している．また，各部分空間に属するベクトルの最大数（*max.*）は，ほとんどの場合で総タプル数と同じ値となっている．白色化変換を用いない場合，ほとんどすべてのキーベクトルが同じ一つの部分空間に属していることを意味する．実際，$m < 64$ の場合では，各部分空間に属するベクトルの最小数（*min.*）も総タプル数 n に等しく，この場合すべてのタプルが単一の部分空間に割り当てられており LSH が機能していないことを意味している．一方，$m \geq 64$ の場合では，各部分空間に属するベクトルの最小数（*min.*）は小さい値を取り LSH はいくつかのベクトルを区別できている．しかし部分空間の総数及び部分空間に割り当てられたベクトル数を考えると，識別能力は不十分である．図 1 と図 2 の比較により，白色化変換は我々の提案フィルタのベクトル識別能力を向上させている．

次に，問合せ結果の再現率について評価した．提案手法では，フィルタによって類似度が小さいベクトルを取り除いた後，通常の問合せ評価を行っている．そのため，問合せに合致しないタプルは取得しない．一方，フィルタが本来は問合せに合致していたタプルを取り除いてしまう場合がある．よって，再現率についてのみ評価を行った．図 3 は LSH における基準ベクトルの個数 m と問合せ範囲の幅を変えて再現率を評価した結果である．図 3(a) は基準ベクトルの個数 m と再現率の関係を示している．図によれば，基準ベクトルの個数が多ければ再現率も大きくなることが分かる．図 3(b) は問合せの幅と再現率の関係を示したものであり，問合せの幅と再現率には関係がみられないことが分かる．なお，$m = 256$ とした．

最後に，サーバにおける問合せ処理時間の評価を行った．図 4 はその結果である．どちらのグラフも，横軸は問合せの幅を示しており，縦軸は問合せ処理に要した時間を対数尺度で記している．なお，このサーバにおける問合せ処理時間には，サーバ・クライアント間の通信時間は含まれていない．総タプル数が $n = 10000$ 及び $n = 100000$ のどちらの場合でも，LSH における基準ベクトルの個数 m が小さい方が問合せ処理時間は少なかった．しかし，図 3 と比較すれば，小さい m の場合は，LSH が本来問合せ結果に含まれるべきタプルを多く取り除いてしまっていることから，問合せ処理時間が短縮されたと考えられる．$n = 10000$ の場合では，図 4(a) によれば，$m > 128$ の場合 LSH フィルタはフィルタを用いない場合より長い時間を要する．これは，暗号化ベクトルデータベース中のタプル数が少ない場合は，フィルタを用いて検索対象を絞り込まなくても，全タプルを調べるコストがそれほど大きくないためである．一方，$n = 100000$ の場合では，$m = 512$ の場合でも，フィルタリングを行わない場合と同等の時間で問合せが処理できている．そして，m が小さくなれば，LSH フィルタを用いた場合の方

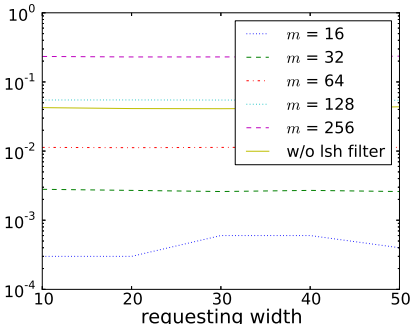


(a) 基準ベクトル数と再現率の比較

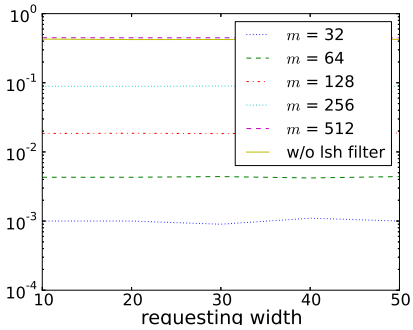


(b) 問合せ幅と再現率の比較 ($m = 256$)

図 3: 平均再現率 ($n = 1000$).



(a) $n = 10000$



(b) $n = 100000$

図 4: 問合せ処理時間 (sec).

が短い時間で問合せを処理できている．縦軸が対数尺度であることを考えると，$m = 256$ の場合であっても，フィルタを用いない場合に比べ処理速度の改善が行えたと言える．

5　まとめと今後の課題

暗号化ベクトルデータベースにおける問合せ処理のためのフィルタリング手法を提案した．提案手法では，局所性検知可能ハッシュ (LSH) と白色化変換を用いて，R 木などの索引が利用できない暗号化ベクトルデータベースにおいてサーバが計算すべきタプル数を削減することができる．第 4 節で議論した評価実験によると，図 1 及び図 2 は LSH が扱うベクトル空間に偏りがある場合でも，白色化変換によって効率よく分割できることを示している．図 4 は，LSH における基準ベクトルの個数 m に依るが適切な m を選択することでサーバにおける問合せ処理時間を削減できていることを示している．

我々のフィルタは式 (2) の近似を用いている．そのため，図 3 に示すように問合せ結果には偽陰性が含まれる．つまり，提案手法は利用者が偽陰性を許容するような暗号化ベクトルデータベースに対して用いることができる．今後の課題としては，問合せ結果の再現率を向上するように提案フィルタを改良することを考えている．

謝辞

本研究は，中島記念国際交流財団の支援により行われました．ここに記して謝意を表します．

参考文献

[1] Wang, J., Wu, S., Gao, H., Li, J., Ooi, B.C.: Indexing Multi-dimensional Data in a Cloud System. In: Proc. of the 30th ACM SIGMOD International Conference on Management of Data, pp. 591–602. ACM Press, Indianapolis, IN, USA (2010)

[2] Charikar, M.S.: Similarity Estimation Techniques from Rounding Algorithms. In: Proc. of the 34th Annual ACM Symposium on Theory of Computing, pp. 380–388. ACM Press, Montreal, Quebec, Canada (2002)

[3] Kirsch, A., Mitzenmacher, M.: Distance-Sensitive Bloom Filters. In: The 18th Workshop on Algorithm Engineering and Experiments. Miami, FL, USA (2006)

[4] Hua, Y., Xiao, B., Veeravalli, B., Feng, D.: Locality-Sensitive Bloom Filter for Approximate Membership Query. IEEE Transactions on Computers **61**(6), 817–830 (2011)

[5] Gionis, A., Indyk, P., Motwani, R.: Similarity Search in High Dimensions via Hashing. In: Proc. of the 25th International Conference on Very Large Data Bases, pp. 518–529. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)

[6] Kulis, B., Grauman, K.: Kernelized Locality-Sensitive Hashing for Scalable Image Search. In: Proc. of the 12th IEEE International Conference on Computer Vision, pp. 2130–2137. IEEE Computer Society, Kyoto, Japan (2009)

[7] Kawamoto, J., Yoshikawa, M.: Private Range Query by Perturbation and Matrix Based Encryption. In: Proc. of the Sixth IEEE International Conference on Digital Information Management, pp. 211–216. IEEE Computer Society, Melbourne, Australia (2011)