複合ソート法による高速な全ペア類似度検索

津田　宏治
産総研生命情報工学研究センター/ JST ERATO

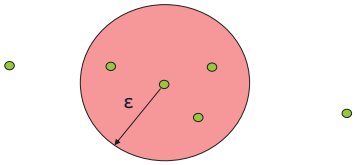
Collaboration with 田部井靖生、清水佳奈、伊東純一、富井健太郎、杉山将、宇野毅明

Do Simple Things in Larger Scale

- 情報爆発 (10^6,10^7,10^8,10^9,...)
  - インターネット上の、画像、音楽、動画、センサー情報
  - 次世代シークエンサー (e.g., Illumina)
  - タンパク質とリガンドのデータベースの成長
- クラスタリング、教師付き分類といった、機械学習の単純なタスクが、計算量超過のため実行できない
- 並列化 (e.g., MapReduce) だけでは解決できない
  - Kコア: 高々K倍の高速化
  - 電力使用量の問題、課金
  - アルゴリズム自体の高速化が、どの場合にも不可欠

全ペア類似度検索

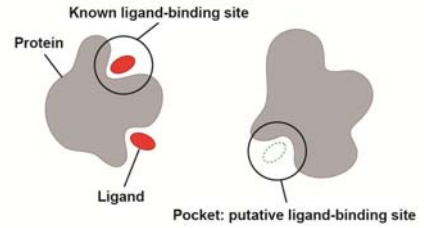
- 近傍グラフ
  - 半教師つき学習、スペクトラルクラスタリングなどで必要
- ε-近傍グラフの作成問題
  - Find all pairs (i,j), i < j, that $\Delta(x_i, x_j) \leq \epsilon$



- 全ペアの距離を計算すると $O(n^2)$

リガンド結合サイトの全体全比較（PDB-wide)

- 類似したリガンド結合サイトを発見して、機能予測
- Minai et al. (2008): 5万既知サイトの解析、クラスタで29日、1コアなら2年
- 本研究：120万サイト(既知と候補)、1コアで4時間程度



アウトライン

- ソーティングによって、高速に全ペア類似度検索を行う方法「複合ソート法」を提案
- ハミング距離に基づく全ペア類似度検索
- SketchSort:コサイン距離に基づく全ペア類似度検索
  - リガンド結合サイト

ハミング距離がd以下のペア発見

```
 1: 1011 1111 0011 1110
 2: 1101 0111 0111 0001
 3: 1100 1000 1101 1100
 4: 0100 0001 0111 1101
 5: 1010 0010 1110 1010
 6: 1111 0011 1001 0111
 7: 0000 0001 0011 1110
 8: 0101 1001 0111 1000
 9: 1101 1000 1101 1110
10: 1001 0011 1001 0111
```

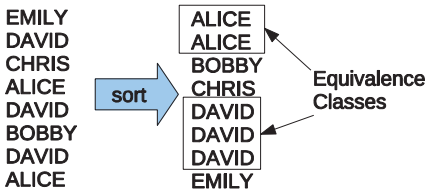
複合ソート法

- 同じ長さｌの文字列がn個与えられている
- ハミング距離ｄ以内のペアの全列挙
- 距離ｄ以内のペアの数をmとする
- 基数ソートを再帰的に繰り返して、全ペアをO(n+m)で列挙可能

- この計算量を達成する方法（文字マスク）は、定数が大きいため実際には低速
- ブロックマスクの導入による高速化

Special Case: 全く同じ文字列ペアの発見 (d=0)

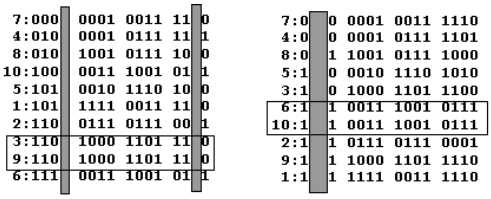
- 基数ソートの後，文字列をEquivalence classに分割: O(n)
- Equivalence class内の全ペアにエッジを張る: O(m)
- 計算量: O(n+m)



複合ソート法 (文字マスク)

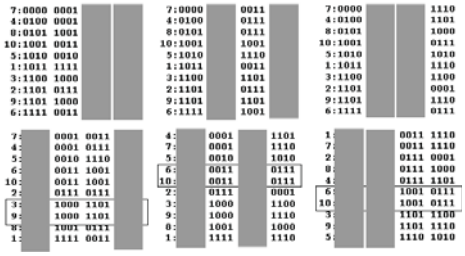
- d個の文字を全通り選んでマスクする

- 基数ソートを $\binom{\ell}{d}$ 回繰り返す
- 計算量はdに対して指数、lに対しても多項式
- しかし、文字列の数nに関しては線形のまま　O(n+m)



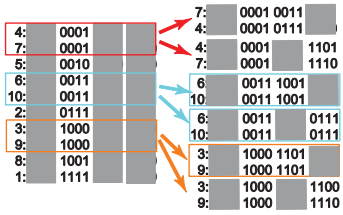
ブロックマスク

- 文字列をk個のブロックに分割する
- d個のブロックを、全通りマスクする
- ソートの回数が劇的に減る
- 近傍でないペアも検出されてしまう
  - 実際に検出されたペアのハミング距離を計算して排除



再帰アルゴリズム

- まず第一のブロックをソートし、Equivalence classを発見する
- 各々のEquivalence classに対して、次のブロックを付け加えて、ソートする。
- k-d個のブロックがつながったら、各Equivalence Classに入っているペアに対して「重複排除」を行う
- 生き残ったペアに関して、ハミング距離を実際に計算



重複排除

- ブロック列に、Lexicographical Orderを導入
- あるブロック列に関して完全一致のペアが見つかっても、ブロック列が「最小」でなければ、出力しない
- 最小性判定

ブロック列中の最右のブロックよりも、左にある空ブロックが完全一致
＝　最小でない

Algorithm 1 Multiple Sorting Method. d: Hamming distance threshold, k: number of blocks.

```
1: function MultipleSortingMethod
2:     I ← {1, ..., n}
3:     B ← φ
4:     Recursion(I, B)
5:     return
6: end function
7: function Recursion(I, B)
8:   if |B| = k − d then
9:     for (i, j) ∈ I × I, i < j do
10:      if s_i^b ≠ s_j^b for all b < max(B), b ∉ B then
11:        if HamDist(s_i, s_j) ≤ d then
12:          Report (i, j) to output file
13:        end if
14:      end if
15:    end for
16:    return
17:  end if
18:  for b in (max(B) + 1)..(k + |B| + 1) do
19:    J ← Sorted indices based on b-th block {s_i^b}_{i∈I}
20:    T ← Intervals of equivalence classes in {s_j^b}_{j∈J}
21:    for each interval (x, y) ∈ T do
22:      Recursion(J[x : y], B ∪ b)
23:    end for
24:  end for
25:  return
26: end function
```

複合ソート法の疑似コード

ブロック数がｋ－ｄに達した際のペアの数え上げ

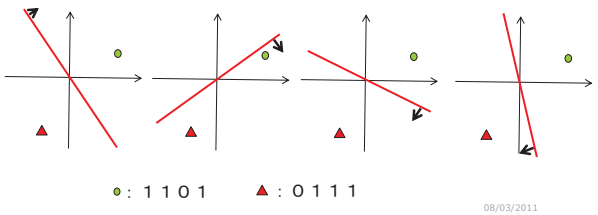
Equivalence Class の再帰的展開

コサイン距離での全ペア類似度検索

- 画像、音声などは、実数値をもつ特徴ベクトルとして表現
- Locality sensitive hashingを用いて、ベクトルを二値の文字列に変換　(スケッチ).
  - 類似したベクトルは、類似した文字列になる
  - 符号付きランダム射影
  - 他の方法を用いれば、ユークリッド距離、Jaccard係数などによる検索も可能
- Missing edge ratio（ペアを逃す確率）を10^-6以下に抑える
- Cover tree (Beygelzimer et al., ICML2006)より10倍以上高速
- 160万個の画像データを用いて実験

ベクトルを、0/1の文字列に射像する

- コサイン距離　$\Delta(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - \frac{\boldsymbol{x}_i^\top \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|}.$
- Locality sensitive hashing
  - D次元のベクトルを長さlの0/1文字列にする
  - ランダム超平面で切断し、片方を0、もう片方を1とする



●：１１０１　▲：０１１１

Cosine LSH

- $R \in \Re^{D \times \ell}$: N(0,1)からサンプルされたランダム行列
- 写像

$$s_{ik} := \mathrm{sign}(\boldsymbol{r}_k^\top \boldsymbol{x}_i), k = 1, \ldots, \ell$$

- 非衝突確率は、角度に比例

$$\Pr\left(s_{ik} \neq s_{jk}\right) = \frac{\theta_{ij}}{\pi}, \quad \forall k,$$

$$\theta_{ij} = \arccos\left(\frac{\boldsymbol{x}_i^\top \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|}\right).$$

- ガウシアンカーネルベースのスケッチを用いれば、ユークリッド距離でも可能 (Raginsky and Lazebnik, NIPS 2009)
  - 近年、多種多様なスケッチが提案されている

SketchSort

- 基本アイデア: ベクトルを文字列にして、複合ソート法適用
- Not good: 長い文字列に、複合ソート法を適用する
- チャンクへの分割
  - 長さ ℓ の文字列プールを Q 個つくる
  - 複合ソート法を各プールに適用する

$$E_q = \{(i,j) \mid HamDist(\boldsymbol{s}_i^q, \boldsymbol{s}_j^q) \leq d, i < j\}.$$

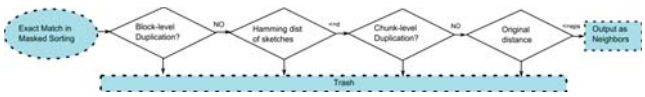
  - 全ての出力セットを併せる

$$E = E_1 \cup \cdots \cup E_Q.$$

  - 中間結果Eの中で、$\Delta(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq \epsilon$ を満たすものを出力

チャンク単位の重複排除

- 異なるチャンクで同じペアが見つかると、重複が発生する
- チャンク q で、ハミング距離 d 以内のペアが見つかった場合には、チャンク 1,..,q-1でハミング距離 d 以内のものがない場合だけ出力
- ３重のチェック体制：できるだけコサイン距離の計算を避ける



SketchSortの疑似コード

```
1: function SketchSort(x_1, ..., x_n)
2:     Use LSH to obtain sketches {s_1i, ..., s_qi}_{i=1}^n from
       data {x_i}_{i=1}^n
3:     I ← {1, ..., n}
4:     for q = 1 : Q do
5:         B ← φ
6:         Recursion(I, B, q)
7:     end for
8:     return
9: end function
10: function Recursion(I, B, q)
11:     if |B| = k − d then
12:         for (i, j) ∈ I × I, i < j do
13:             if s_qi^b ≠ s_qj^b for all b < max(B), b ∉ B then
14:                 if HamDist(s_qi, s_qj) ≤ d then
15:                     if HamDist(s_ri, s_rj) > d for all r < q
                           then
16:                         if Δ(x_i, x_j) ≤ ε then
17:                             Report (i, j) to output file
18:                         end if
19:                     end if
20:                 end if
21:             end if
22:         end for
23:         return
24:     end if
25:     for b in (max(B) + 1)..(k + |B| + 1) do
26:         J ← Sorted indices based on b-th block {s_qi^b}_{i∈I}
27:         T ← Intervals of equivalence classes in {s_qj^b}_{j∈J}
28:         for each interval (x, y) ∈ T do
29:             Recursion(J[x : y], B ∪ b, q)
30:         end for
31:     end for
32:     return
33: end function
```

各チャンクに対する呼び出し

ペアの数え上げ（三重のチェック）

Equivalence Class の再帰的展開

二種類のエラー

- 真にエッジセット E*, SketchSortによる中間結果 E
- False positive: 近傍ではないペアが、1つ以上のチャンクでハミング距離d以内となる事象

$$F_1 = \{(i,j) \mid (i,j) \in E, (i,j) \notin E^*\}.$$

- False negative: 近傍ペアが、全てのチャンクでハミング距離d＋1以上となる現象

$$F_2 = \{(i,j) \mid (i,j) \notin E, (i,j) \in E^*\}.$$

False negative rateの上限: Missing edge ratio

- False negativeの方が致命的
  - False positiveは、コサイン距離計算によって除かれる
- Missing edge ratio (False negative rate) は次のようにバウンドされる

$$E\left[\frac{|F_2|}{|E^*|}\right] \le \left(1 - \sum_{k=0}^{\lfloor d \rfloor} \binom{\ell}{k} p^k (1-p)^{\ell-k}\right)^Q,$$

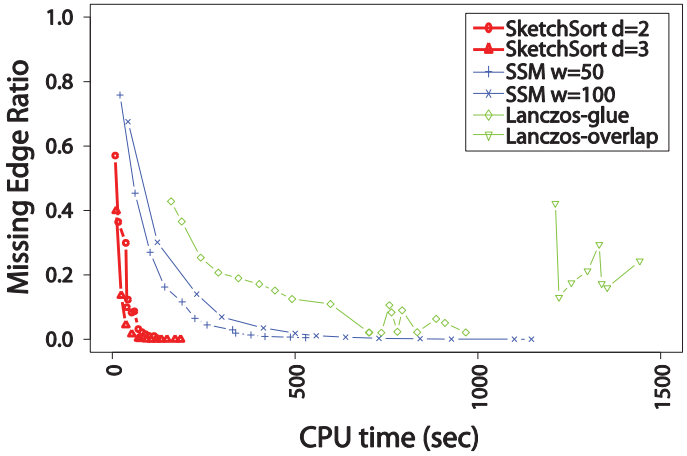
ここで、pは、LSHの非衝突確率の上限である

$$p = \frac{\arccos(1-\epsilon)}{\pi}.$$

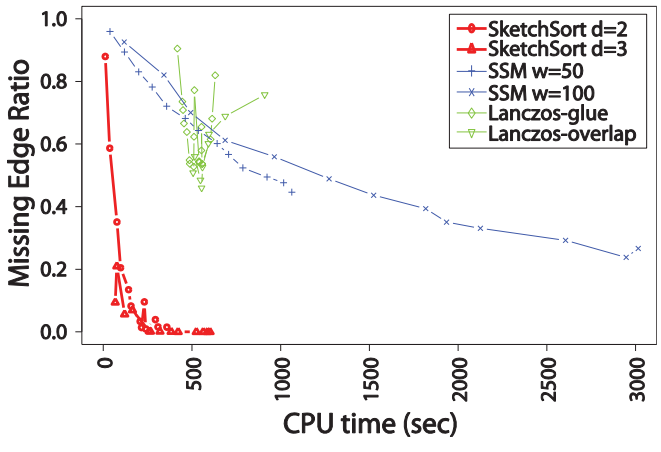
単純ソート、Lanczos Bisectionとの比較実験

- 二つのデータセット
  - MNIST (60,000 points, 748 dims)
  - TinyImage (100,000 points, 960 dims)
    - Missing Edge Ratio計算のため、ダウンサンプリング
- コサイン距離の閾値： 0．15π
- 各チャンクは32ビット
- 複合ソートのハミング距離とブロック数: (2,5), (3,6)
- チャンク数： 2,6,10,...,50
- Lanczos Bisection (JMLR, 2009)とも比較
  - Lanczos法を使って空間を、再帰的に2分割する方法

MNIST, 閾値0.15π

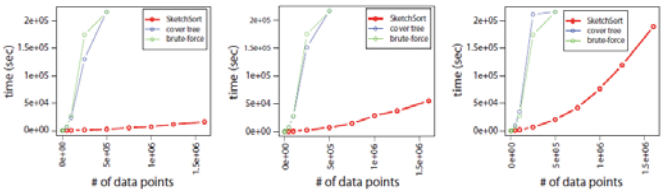


TinyImage, 閾値0.15π



160万画像での実験

- Missing Edge Ratio ＜ 1.0×10^{-6}
- 160万画像の全ペア類似度検索を、4.3時間で処理（0.05π）



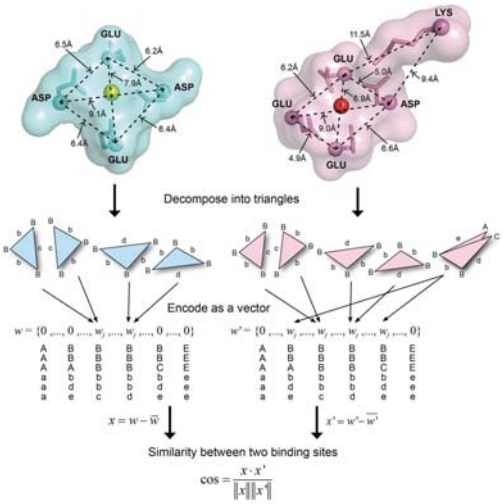
Near duplication detection in up to 1.6 million images at thresholds 0.05π (left), 0.10π (middle) and 0.15π (right)

タンパク質のリガンド結合可能部位の大規模解析

- ＰＤＢに登録されているタンパク質三次元構造
- 1,260,627個のリガンド結合可能部位(ポケット)を抽出
  - その中の約20万個は、実際にリガンドが結合
- SketchSortで類似するペアを列挙

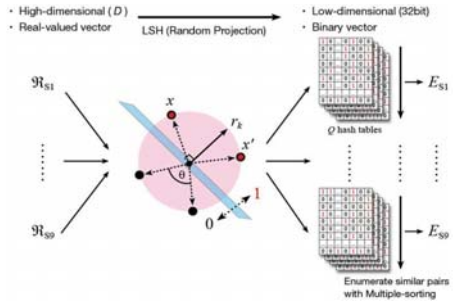
- 通常は、アミノ酸配列の類似性によって、結合部位を発見 (Homolog)
- しかし、配列が違っていても、三次元構造が同じであれば結合する可能性 (Analog)

幾何的特徴抽出

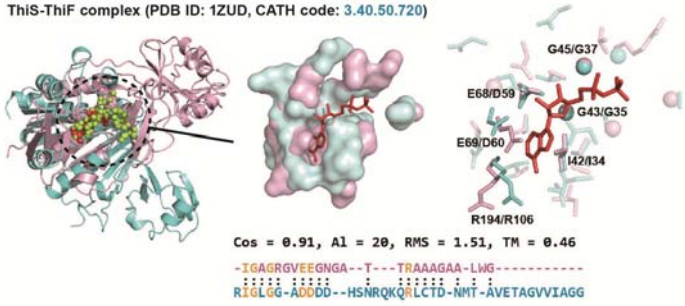


SketchSortの適用

- ８種類の異なった特徴空間を用意
- 各々にSketchSortをかける：全部で３０時間
- コサインの閾値0.85で88,194,290個のペアを抽出



Similar ADP- and putative-binding sites shared between
DR_0571 protein (PDB ID: 3C4N, CATH code: 3.50.50.60) and
ThiS-ThiF complex (PDB ID: 1ZUD, CATH code: 3.40.50.720)



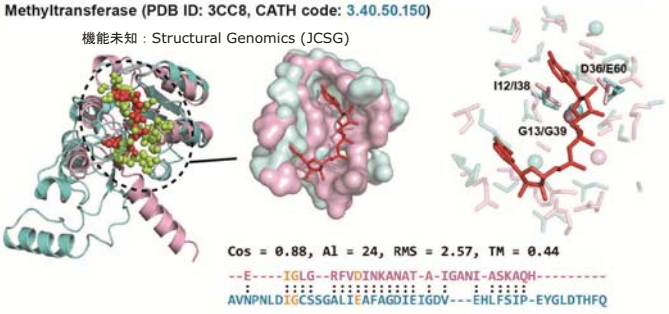
全体の構造にも、配列にも類似性なし

Approved by in-silico modeling in Lehmann et al., Biochemistry, 2006.

ニコチンアミド

Similar NAI- and putative-binding sites shared between
KTN (K+ transport, nucleotide binding) domain (PDB ID: 1LSU, CATH code: 3.40.50.720) and
Methyltransferase (PDB ID: 3CC8, CATH code: 3.40.50.150)

機能未知：Structural Genomics (JCSG)



終わりに

- 高速な全ペア類似度検索法を提案
- 簡単に数千万点のデータが扱える
- 様々な分野における応用が考えられる

- コードはこちら
  - http://code.google.com/p/sketchsort/