類似したアイテムを探す
(Finding Similar Items)

1. 類似したアイテムを探す

文書間の類似度（Similarity of Documents）

　ウェブやニュース記事の集まりなどの大規模コーパスに含まれる文書の中から，文章として似ている文書を探す

- 　盗用された文書を探す（剽窃）

文書間の類似度（Similarity of Documents）

　ウェブやニュース記事の集まりなどの大規模コーパスに含まれる文書の中から，文章として似ている文書を探す

- 盗用された文書を探す（剽窃）
- ページのミラーリング

検索において，結果の最初のページとほとんど同じページを表示することを防ぐことができれば，サーチエンジンがより良い結果を生成することができる

文書間の類似度（Similarity of Documents）

　ウェブやニュース記事の集まりなどの大規模コーパスに含まれる文書の中から，文章として似ている文書を探す

- 盗用された文書を探す（剽窃）
- ページのミラーリング

  検索において，結果の最初のページとほとんど同じページを表示することを防ぐことができれば，サーチエンジンがより良い結果を生成することができる

- 同じ情報源の記事

  ニュースの集約（news aggregation）への応用など

協調フィルタリング(Collaborative Filtering)

利用者に対して同じ傾向を持つ別の利用者によって参照されたアイテムを推薦する処理

- オンラインでの購入（On-Line Purchases）

  購入した商品の類似度が高い　⇒　顧客が類似している

  高い類似度を持つ顧客によって購入された商品　⇒

  商品が類似している

協調フィルタリング(Collaborative Filtering)

利用者に対して同じ傾向を持つ別の利用者によって参照されたアイテムを推薦する処理

- オンラインでの購入（On-Line Purchases）

購入した商品の類似度が高い　⇒　顧客が類似している

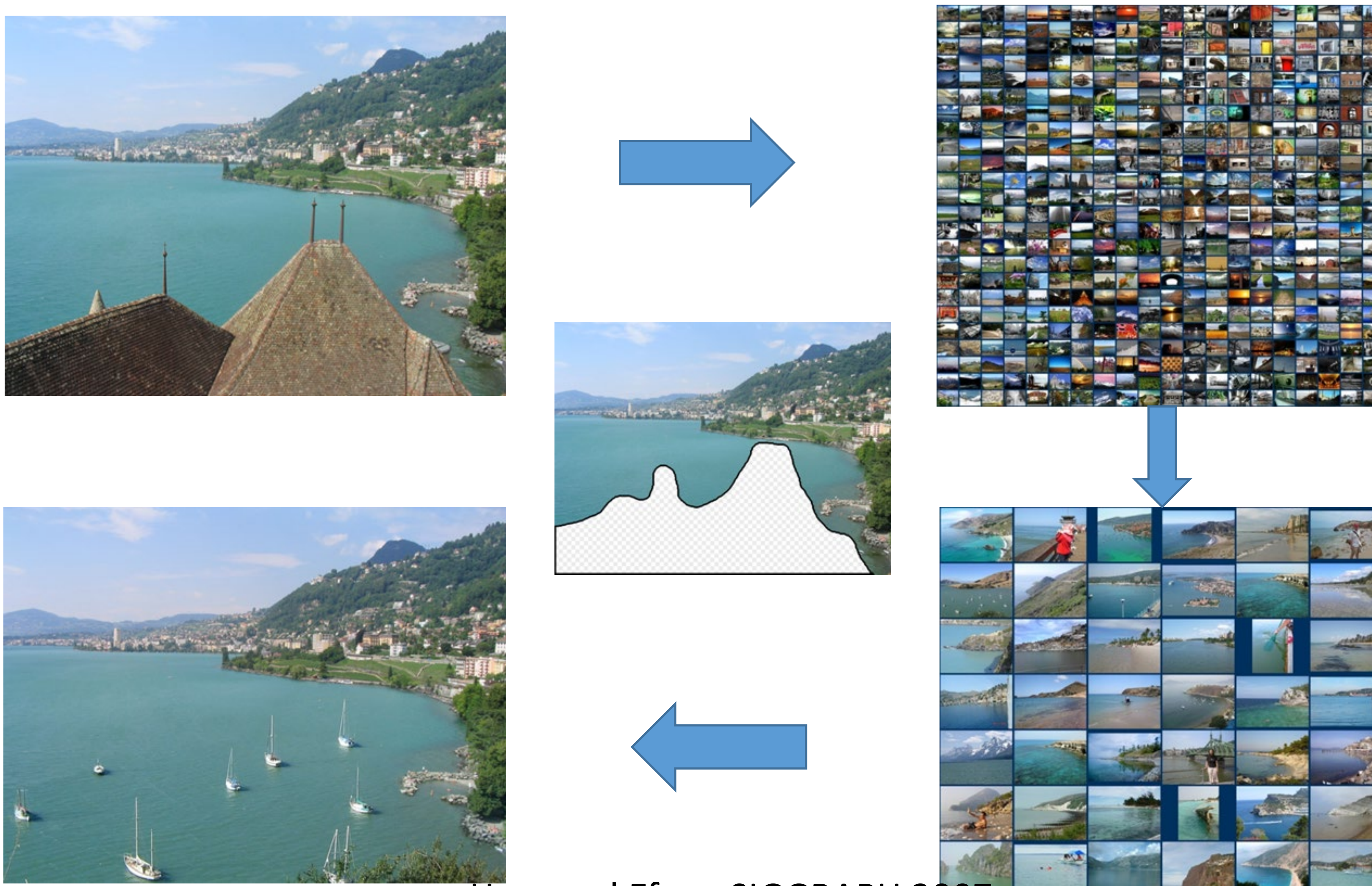
高い類似度を持つ顧客によって購入された商品　⇒　商品が類似している

- 映画の格付け (Movie Ratings)

同じ映画を見て，同じ映画に高い格付けを与えている顧客　⇒　顧客が類似している

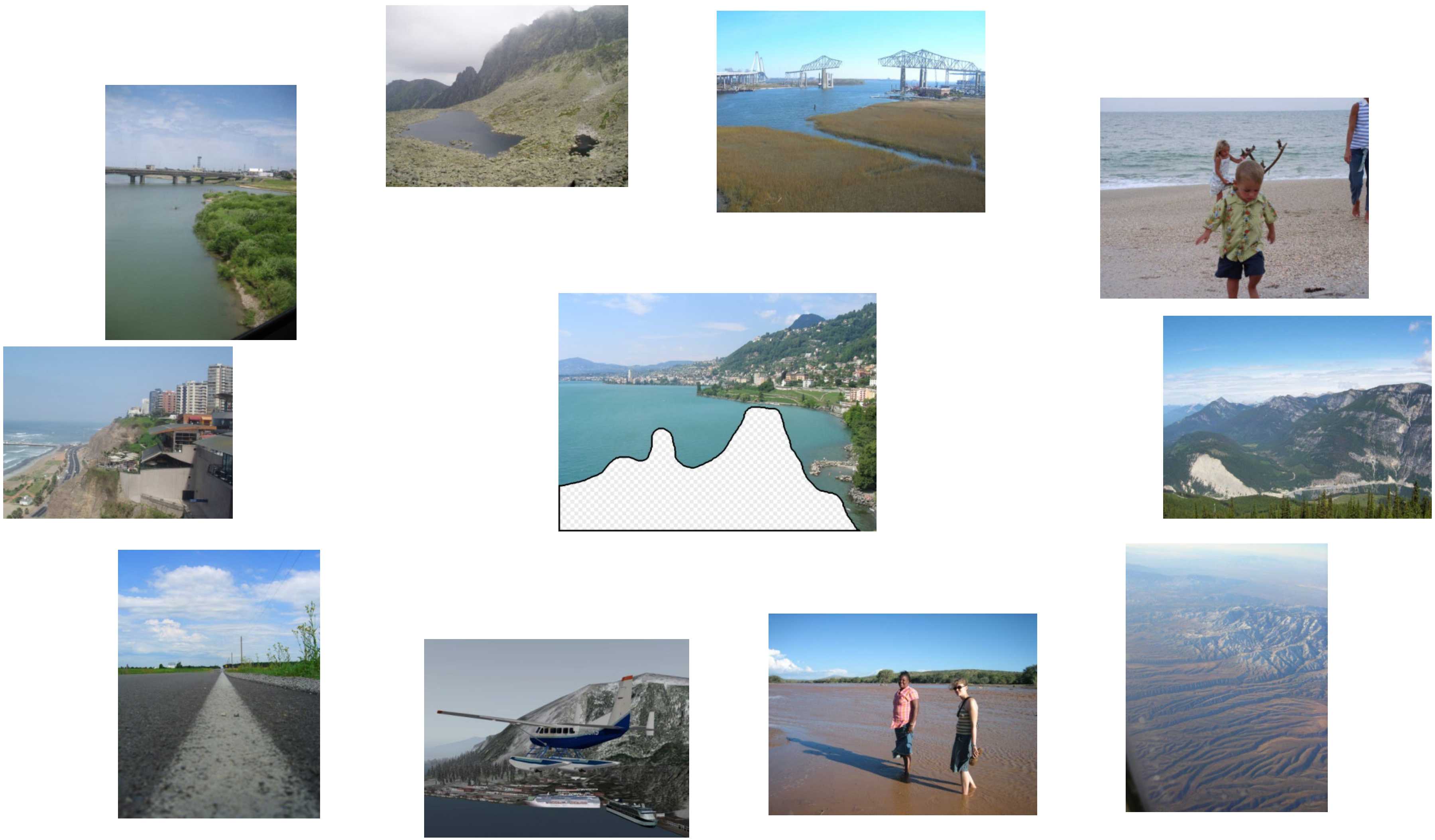
同じ顧客によって見られて，高い格付けが与えられていることが多い映画　⇒　映画が類似している

画像の類似度(Image Similarity)



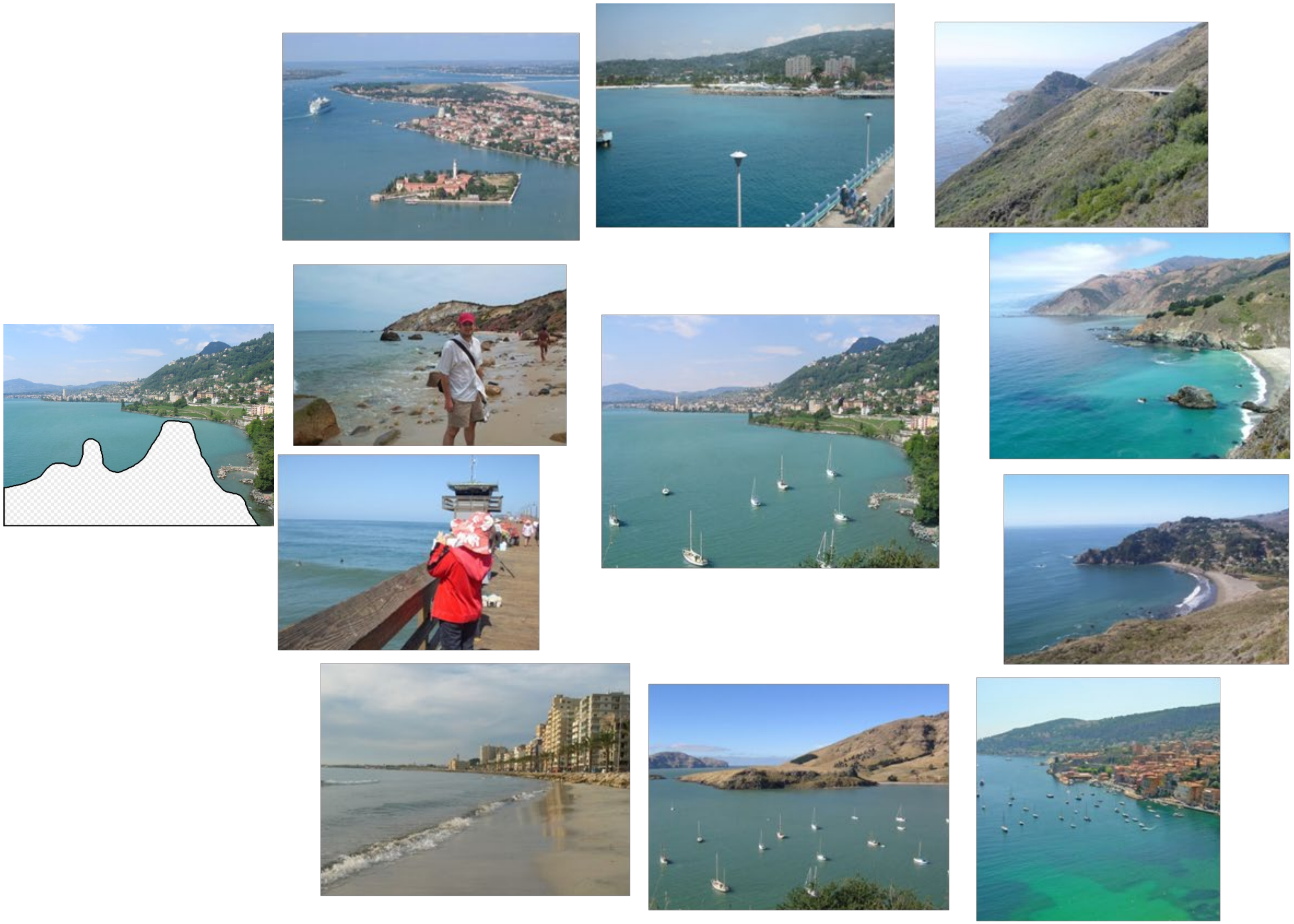
Hays and Efros, SIGGRAPH 2007

画像の類似度(Image Similarity)



10 nearest neighbors from a collection of 20,000 images

画像の類似度(Image Similarity)



10 nearest neighbors from a collection of 2 million images

任意の種類の類似アイテムを探すときに生じる重要な問題

アイテムの対の類似計算が非常に簡単に行われるにしても，検査されるアイテム対の数が莫大

- 例：ウェブやニュース記事の集まりなどの大規模コーパスに含まれる文書の中から，文章として似ている文書を探す
- 文書の数 N = 100万
- 比較される文書の対の数 $N(N-1)/2 \approx$ 5*10^11
- 10^5 secs/day and 10^6 comparisons/sec ⇒ 5日
- N = 1000万 ⇒ 1年以上

Naïve solution $O(N^2)$ ☹

MAGIC: $O(N)$でもできる！

今回紹介する局所性鋭敏型ハッシング(locality-sensitive hashing, LSH)

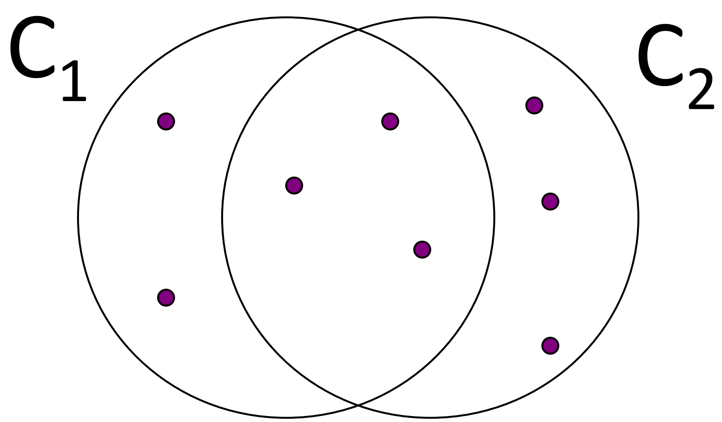
集合間のジャッカール類似度　Jaccard similarity

ジャッカール類似度（*Jaccard similarity*）

和集合の大きさに対する積集合の大きさの比率

$$Sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$

Intersection over Union (IoU)



- 2つの集合の積集合の要素は3つ
- どちらかあるいは両方に現れる要素は8つ
- $Sim(C_1, C_2) = 3/8$

文書のシングリング(Shingling of documents)

- タスク:類似している文書を特定する
- 文書を文書中に現れる短い文字列の集合として表現する
- k シングル(k-shingle, k-gram): 文書内で見出すことができる長さk のすべての部分文字列

文書 ⇒その中に1回以上出現する長さ kのシングル の集合

例: k = 2; 文書D = abcdabd

Set of 2-shingles = {ab, bc, cd, da, bd}

文書のシングリング(Shingling of documents)

- タスク: 類似している文書を特定する
- 文書を文書中に現れる短い文字列の集合として表現する
- k シングル(k-shingle, k-gram): 文書内で見出すことができる長さ k のすべての部分文字列

文書 ⇒その中に1回以上出現する長さ kのシングル の集合

ホワイトスペース(空白，タブ，改行など)をどのように扱うか

ホワイトスペースを除去する

例:　k = 9;

着陸

The plane was ready for touch down.　　touchdown

The quarterback scored a touchdown.　　touchdown

[アメフト]　タッチダウン，得点

文書のシングリング(Shingling of documents)

- タスク：類似している文書を特定する
- 文書を文書中に現れる短い文字列の集合として表現する
- k シングル(k-shingle，k-gram)：文書内で見出すことができる長さ k のすべての部分文字列

文書 ⇒その中に1回以上出現する長さ kのシングル の集合

ホワイトスペース(空白，タブ，改行など)をどのように扱うか

ホワイトスペースを除去しない

例：　k = 9;　　着陸

The plane was ready for touch down.　touch dow, ouch down

The quarterback scored a touchdown.　touchdown

[アメフト]　タッチダウン，得点

集合の行列表現(Matrix Representation of Sets)

特徴行列(characteristic matrix)

- 集合の集まりを特徴行列Mとして可視化できる
- 列は異なる集合（例えば異なる文書）に対応する
- 行は集合の要素（例えば全てのk-シングルの集合）に対応する

	C_1	C_2	C_3	C_4
a	0	1	1	0
b	0	0	1	1
c	1	0	0	0
d	0	1	0	1
e	0	0	0	1
f	1	1	0	0
g	0	0	1	0

集合の行列表現(Matrix Representation of Sets)

特徴行列(characteristic matrix)

- 集合の集まりを特徴行列Mとして可視化できる
- 列は異なる集合（例えば異なる文書）に対応する
- 行は集合の要素（例えば全てのk-シングルの集合）に対応する

	C_1	C_2	C_3	C_4
a	0	1	1	0
b	0	0	1	1
c	1	0	0	0
d	0	1	0	1
e	0	0	0	1
f	1	1	0	0
g	0	0	1	0

- r行目の要素が，列cで表現される集合の要素である場合，行r列cの値M(r, c)が1となる
- それ以外の場合M(r, c)の値は0になる
- 特徴行列はデータを保存するためにはあまり使われない
- 特徴行列は疎(sparse)である

集合の行列表現(Matrix Representation of Sets)

特徴行列(characteristic matrix)

- 集合の集まりを特徴行列Mとして可視化できる
- 列は異なる集合（例えば異なる文書）に対応する
- 行は集合の要素（例えば全てのk-シングルの集合）に対応する

	C_1	C_2	C_3	C_4
a	0	1	1	0
b	0	0	1	1
c	1	0	0	0
d	0	1	0	1
e	0	0	0	1
f	1	1	0	0
g	0	0	1	0

練習:

（１） 集合C_1とC_2に対してジャッカール類似度を計算せよ.

（２） 集合C_2とC_3に対してジャッカール類似度を計算せよ.

集合の行列表現(Matrix Representation of Sets)

特徴行列(characteristic matrix)

- 集合の集まりを特徴行列Mとして可視化できる
- 列は異なる集合（例えば異なる文書）に対応する
- 行は集合の要素（例えば全てのk-シングルの集合）に対応する

C_1	C_2		
0	1		*
1	0		*
1	1	*	*
0	0		
1	1	*	*
0	1		*

（別の例）

$Sim(C_1, C_2) =$
$2/5 = 0.4$

	C_1	C_2	C_3	C_4
a	0	1	1	0
b	0	0	1	1
c	1	0	0	0
d	0	1	0	1
e	0	0	0	1
f	1	1	0	0
g	0	0	1	0

集合の行列表現(Matrix Representation of Sets)

- 列 C_1 と C_2 が与えられた場合，すべての行を以下の通り分類できます:

	C_1	C_2
a	1	1
b	1	0
c	0	1
d	0	0

- *a* = # rows of type *a* , etc.
- $Sim(C_1, C_2) = a/(a + b + c)$.

	C_1	C_2	C_3	C_4
a	0	1	1	0
b	0	0	1	1
c	1	0	0	0
d	0	1	0	1
e	0	0	0	1
f	1	1	0	0
g	0	0	1	0

３－２

類似したアイテムを探す (Finding Similar Items)

２．ミンハッシング(Minhashing)

類似度を保持した集合の要約 シグネチャー(signature)

- 問題: 巨大な集合;すべての集合を主記憶に置いておくことはできない

(主記憶に収まったとしても, 対の数が膨大で各対の類似度を計算することはできない. . .

→ これについての解法は後で紹介する)

類似度を保持した集合の要約 シグネチャー(signature)

- 問題: 巨大な集合;すべての集合を主記憶に置いておくことはできない

- 巨大な集合をそれよりはるかに小さなシグネチャー(signature) で置き換える手法が必要

- シグネチャーに求められる重要な性質

 ＊シグネチャーから集合のジャッカール類似度を推定(近似)できる

 ＊シグネチャーが大きければ大きいほど推定が正確になる

ミンハッシング(Minhashing)

- ミンハッシュ (minhash) :

計算原理 ➔

現実的なミンハッシュの近似値の計算方法

ミンハッシング(Minhashing)

- ミンハッシュ (minhash) :

1. 特徴行列の行の順列(permutation)を選ぶ
2. 任意の列のミンハッシュの値は，並べ替えられた順でその列が値1を持つ最初の行の番号である

	C_1	C_2	C_3	C_4
1	0	1	1	0
2	0	0	1	1
3	1	0	0	0
4	0	1	0	1
5	0	0	0	1
6	1	1	0	0
7	0	0	1	0

行の
並べ替え

	C_1	C_2	C_3	C_4
7	0	0	1	0
6	1	1	0	0
5	0	0	0	1
4	0	1	0	1
3	1	0	0	0
2	0	0	1	1
1	0	1	1	0

ミンハッシング(Minhashing)

- ミンハッシュ (minhash):

1. 特徴行列の行の順列(permutation)を選ぶ
2. 任意の列のミンハッシュの値は，並べ替えられた順でその列が値1を持つ最初の行の番号である

	C_1	C_2	C_3	C_4
7	0	0	1	0
6	1	1	0	0
5	0	0	0	1
4	0	1	0	1
3	1	0	0	0
2	0	0	1	1
1	0	1	1	0

$h(C_1)$	$h(C_2)$	$h(C_3)$	$h(C_4)$
2	2	1	3

ミンハッシュ関数$h(C_i)$:
各集合を一つの値に写像する

ミンハッシング(Minhashing)

- 練習　以下の特徴行列の並べ替えでミンハッシュシグネチャーを計算せよ

	C_1	C_2	C_3	C_4
1	0	1	1	0
2	0	0	1	1
3	1	0	0	0
4	0	1	0	1
5	0	0	0	1
6	1	1	0	0
7	0	0	1	0

行の並べ替え

	C_1	C_2	C_3	C_4
6	1	1	0	0
3	1	0	0	0
1	0	1	1	0
7	0	0	1	0
2	0	0	1	1
5	0	0	0	1
4	0	1	0	1

ミンハッシング(Minhashing)

- 練習　以下の特徴行列の並べ替えでミンハッシュシグネチャーを計算せよ

	C_1	C_2	C_3	C_4
6	1	1	0	0
3	1	0	0	0
1	0	1	1	0
7	0	0	1	0
2	0	0	1	1
5	0	0	0	1
4	0	1	0	1

$h(C_1)$	$h(C_2)$	$h(C_3)$	$h(C_4)$
1	1	3	5

ミンハッシング(Minhashing)

特徴行列M

	C_1	C_2	C_3	C_4
1	0	1	1	0
2	0	0	1	1
3	1	0	0	0
4	0	1	0	1
5	0	0	0	1
6	1	1	0	0
7	0	0	1	0

Minhashing

シグネチャー行列
（Signature Matrix）

$h(C_1)$	$h(C_2)$	$h(C_3)$	$h(C_4)$
2	2	1	3
1	1	3	5

・・・

類似度を保持した集合の要約 シグネチャー(signature)

- 問題:　巨大な集合;すべての集合を主記憶に置いておくことはできない

- 巨大な集合をそれよりはるかに小さなシグネチャー(signature) で置き換える手法が必要

- シグネチャーに求められる重要な性質

  ＊シグネチャーから集合のジャッカール類似度を推定（近似）できる

  ＊シグネチャーが大きければ大きいほど推定が正確になる

ミンハッシングとジャッカール類似度の関連

- 2つの集合において，行のランダムな並べ替えに対するミンハッシュ関数が同じ値を出力する確率は，これらの集合間のジャッカール類似度に等しい

$$P(\ h(C_1) = h(C_2)\ )\ = Sim(C_1, C_2)$$

ミンハッシングとジャッカール類似度の関連

- 2つの集合において，行のランダムな並べ替えに対するミンハッシュ関数が同じ値を出力する確率は，これらの集合間のジャッカール類似度に等しい

$$P(\ h(C_1) = h(C_2)\ ) = Sim(C_1, C_2)$$

- 列 C_1 と C_2 が与えられた場合，すべての行を以下の通り分類できます:

	C_1	C_2
a	1	1
b	1	0
c	0	1
d	0	0

- *a* = # rows of type *a* , etc.
- $Sim(C_1, C_2) = a/(a + b + c)$.

	C_1	C_2	C_3	C_4
a	0	1	1	0
b	0	0	1	1
c	1	0	0	0
d	0	1	0	1
e	0	0	0	1
f	1	1	0	0
g	0	0	1	0

ミンハッシングとジャッカール類似度の関連

- $P(\ h(C_1) = h(C_2)\ )$

＊先頭から処理を行うのであれば，b, c型より先にa型に出会う

確率はa/(a+b+c)

＊もし最初に見つかったd型以外の行がa型の行である場合，

$h(C_1) = h(C_2)$ となる

	C_1	C_2
a	1	1
b	1	0
c	0	1
d	0	0

- *a* = # rows of type *a* , etc.
- $Sim(C_1, C_2) = a/(a+b+c)$.

	C_1	C_2	C_3	C_4
6	1	1	0	0
3	1	0	0	0
1	0	1	1	0
7	0	0	1	0
2	0	0	1	1
5	0	0	0	1
4	0	1	0	1

ミンハッシングとジャッカール類似度の関連

- $P(\ h(C_1) = h(C_2)\ )$

＊もし最初に見つかったd型以外の行がb, c型の行である場合，列の値が1である方の集合が，その行をミンハッシュ値としてとることになる

＊列の値が0である方の集合は，並べ替えられたリストの下方にある値のどれかをとる

$h(C_1) \neq h(C_2)$ となる　確率は(b+c)/(a+b+c)

結論: $P(\ h(C_1) = h(C_2)\ ) = a/(a+b+c)$

$= Sim(C_1, C_2)$

	C_1	C_2
a	1	1
b	1	0
c	0	1
d	0	0

- *a* = # rows of type *a* , etc.
- $Sim(C_1, C_2) = a/(a + b + c)$.

	C_1	C_2	C_3	C_4
6	1	1	0	0
3	1	0	0	0
1	0	1	1	0
7	0	0	1	0
2	0	0	1	1
5	0	0	0	1
4	0	1	0	1

ミンハッシュシグネチャー(Minhash signatures)

- 特徴行列*M*で表現された集合の集まりが与えられる
- 各集合を表現するために，*M*の行のランダムな並べ替えの数nを選ぶ（数百個）
- それらの並べ替えによって定義されるミンハッシュ関数を
  $h_1,\ h_2, \ldots, h_n$ とする
- 集合Cを表現する列から，Cのミンハッシュシグネチャー(minhash signature) $[h_1(C),\ h_2(C), \ldots, h_n(C)]$ を構築する

ミンハッシュシグネチャー(Minhash signatures)

- 特徴行列*M*で表現された集合の集まりが与えられる
- 各集合を表現するために，*M*の行のランダムな並べ替えの数n を選ぶ（数百個）
- それらの並べ替えによって定義されるミンハッシュ関数を

  $h_1,\ h_2, \ldots, h_n$ とする
- 集合*C*を表現する列から，*C*のミンハッシュシグネチャー(minhash signature） $[h_1(C),\ h_2(C), \ldots, h_n(C)]$ を構築する

行列*M* => シグネチャー行列（signature matrix）***SIG***

＊*M*の *i* 番目の列の値を，*i* 番目の列のミンハッシュシグネチャーによって置き換えたものである

＊シグネチャー行列は， *M*と同じ数の列を持つが，行は*n* 個のみである

ミンハッシュシグネチャー(Minhash signatures)

- シグネチャー行列を用いることで，ジャッカール類似度を推定することができる

$$P(\ h(C_1) = h(C_2)\ )\ = a/(a+b+c)$$

シグネチャー行列の任意の行に2つの列が同じ値を持っている確率は，対応する集合のJaccard類似度に等しいである

- **一致している行の数の割合(の期待値) ＝ 対応する集合のJaccard類似度！**
- ミンハッシュの数を増やしたら，大数の法則によりJaccard類似度の推定の誤差が小さくなる

＊シグネチャー行列の各行が独立事象

ミンハッシュシグネチャーSIGの計算

- 問題： 大きな特徴行列（数十億の行～）を明示的に行ごとに並べ替えることは現実的ではない

ミンハッシュシグネチャーSIGの計算

- 問題：　大きな特徴行列（数十億の行～）を明示的に行ごとに並べ替えることは現実的ではない
- 行番号を行の数と同数のバケツに写像するランダムハッシュ関数を適用することによって，ランダムな行の並べ替えをシミュレートすることが可能である

行r	h = 3r + 1 mod 5
0	1
1	4
2	2
3	0
4	3

Random hash function h(r)

$$h_{a,b}(r) = (a \cdot r + b) \bmod k$$

ミンハッシュシグネチャーSIGの計算

- 問題：大きな特徴行列（数十億の行～）を明示的に行ごとに並べ替えることは現実的ではない
- 行番号を行の数と同数のバケツに写像するランダムハッシュ関数を適用することによって，ランダムな行の並べ替えをシミュレートすることが可能である

行r	h = 3r + 1 mod 5
0	1
1	4
2	2
3	0
4	3

Random hash function h(r)

$h_{a,b}(r) = (a\cdot r+b) \bmod k$

- 行のn個のランダムな並べ替えをとる代わりに，行においてランダムに選んだn個のハッシュ関数 $h_1, h_2, \ldots, h_n$ を用いる
- 与えられた順番で各行を考慮し，シグネチャー行列SIGを構築する

SIG(i, c)　シグネチャー行列中 i 番目のハッシュ関数と c 番目の列の要素

ミンハッシュシグネチャー*SIG*の計算

ALGORITHM

SIG(i, c)　シグネチャー行列中 i 番目のハッシュ関数と
c 番目の列の要素

＊最初にすべての i と c について，SIG(i, c)に無限大∞をセットする

for each row r **do**
　for each hash function h_i **do**
　　compute $h_i(r)$;
　for each column c
　　if c has 1 in row r
　　　for each hash function h_i **do**
　　　　if $h_i(r)$ is smaller than $SIG(i, c)$ **then**
　　　　　$SIG(i, c) := h_i(r)$;

ミンハッシュシグネチャー*SIG*の計算

M

行r	C_1	C_2	C_3	C_4	$h_1 = r + 1 \bmod 5$	$h_2 = 3r + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

初期状態

SIG

	C_1	C_2	C_3	C_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

ミンハッシュシグネチャー*SIG*の計算

M

行r	C_1	C_2	C_3	C_4	$h_1 = r + 1 \bmod 5$	$h_2 = 3r + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

$r = 0$

SIG

	C_1	C_2	C_3	C_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

ミンハッシュシグネチャー*SIG*の計算

M

行r	C_1	C_2	C_3	C_4	$h_1 = r + 1 \bmod 5$	$h_2 = 3r + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

	C_1	C_2	C_3	C_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

$r = 1$

	C_1	C_2	C_3	C_4
h_1	1	∞	2	1
h_2	1	∞	4	1

SIG

ミンハッシュシグネチャーSIGの計算

M

行r	C_1	C_2	C_3	C_4	$h_1 = r + 1 \bmod 5$	$h_2 = 3r + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

$r = 2$

	C_1	C_2	C_3	C_4
h_1	1	∞	2	1
h_2	1	∞	4	1

	C_1	C_2	C_3	C_4
h_1	1	3	2	1
h_2	1	2	4	1

SIG

ミンハッシュシグネチャーSIGの計算

M

行r	C_1	C_2	C_3	C_4	$h_1 = r + 1 \bmod 5$	$h_2 = 3r + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

	C_1	C_2	C_3	C_4
h_1	1	3	2	1
h_2	1	2	4	1

$r = 3$

	C_1	C_2	C_3	C_4
h_1	1	3	2	1
h_2	0	2	0	0

SIG

ミンハッシュシグネチャーSIGの計算

M

行r	C_1	C_2	C_3	C_4	$h_1 = r + 1 \bmod 5$	$h_2 = 3r + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

$r = 4$

	C_1	C_2	C_3	C_4
h_1	1	3	2	1
h_2	0	2	0	0

	C_1	C_2	C_3	C_4
h_1	1	3	0	1
h_2	0	2	0	0

SIG

SIGからJaccard類似度の推定

M

行r	C_1	C_2	C_3	C_4
0	1	0	0	1
1	0	0	1	0
2	0	1	0	1
3	1	0	1	1
4	0	0	1	0

$Sim(C_1, C_4) = 2/3$

SIG

	C_1	C_2	C_3	C_4
h_1	1	3	0	1
h_2	0	2	0	0

$Sim(C_1, C_4) = 1.0$

SIGからJaccard類似度の推定

M

行r	C_1	C_2	C_3	C_4
0	1	0	0	1
1	0	0	1	0
2	0	1	0	1
3	1	0	1	1
4	0	0	1	0

$Sim(C_1, C_3) = 1/4$

SIG

	C_1	C_2	C_3	C_4
h_1	1	3	0	1
h_2	0	2	0	0

$Sim(C_1, C_3) = 1/2$

SIGからJaccard類似度の推定

M

行r	C1	C2	C3	C4
0	1	0	0	1
1	0	0	1	0
2	0	1	0	1
3	1	0	1	1
4	0	0	1	0

$Sim(C_1, C_2) = 0$

SIG

	C1	C2	C3	C4
h1	1	3	0	1
h2	0	2	0	0

$Sim(C_1, C_2) = 0$

練習： $Sim(C_2, C_3)$， $Sim(C_3, C_4)$

３－３

類似したアイテムを探す
(Finding Similar Items)

3. 局所性鋭敏型ハッシング
Locality-Sensitive Hashing (LSH)

任意の種類の類似アイテムを探すときに生じる重要な問題

　アイテムの対の類似計算が非常に簡単に行われるにしても，検査されるアイテム対の数が莫大

- 例：ウェブやニュース記事の集まりなどの大規模コーパスに含まれる文書の中から，文章として似ている文書を探す
- 文書の数　N = 100万，シグネチャー　1文書あたり1,000バイト
  全体のデータは1GB程度に収まる
- 比較される文書の対の数　$N(N-1)/2 \approx 5*10^{11}$
- 10^5 secs/day and 10^6 comparisons/sec　⇒　5日
- N = 1000万　⇒　1年以上

Naïve solution $\boldsymbol{O}\left(\boldsymbol{N}^{2}\right)$ ☹

MAGIC: $\boldsymbol{O}(\boldsymbol{N})$でもできる！

　今回紹介する局所性鋭敏型ハッシング(locality-sensitive hashing, LSH)

局所性鋭敏型ハッシング(locality-sensitive hashing, LSH)

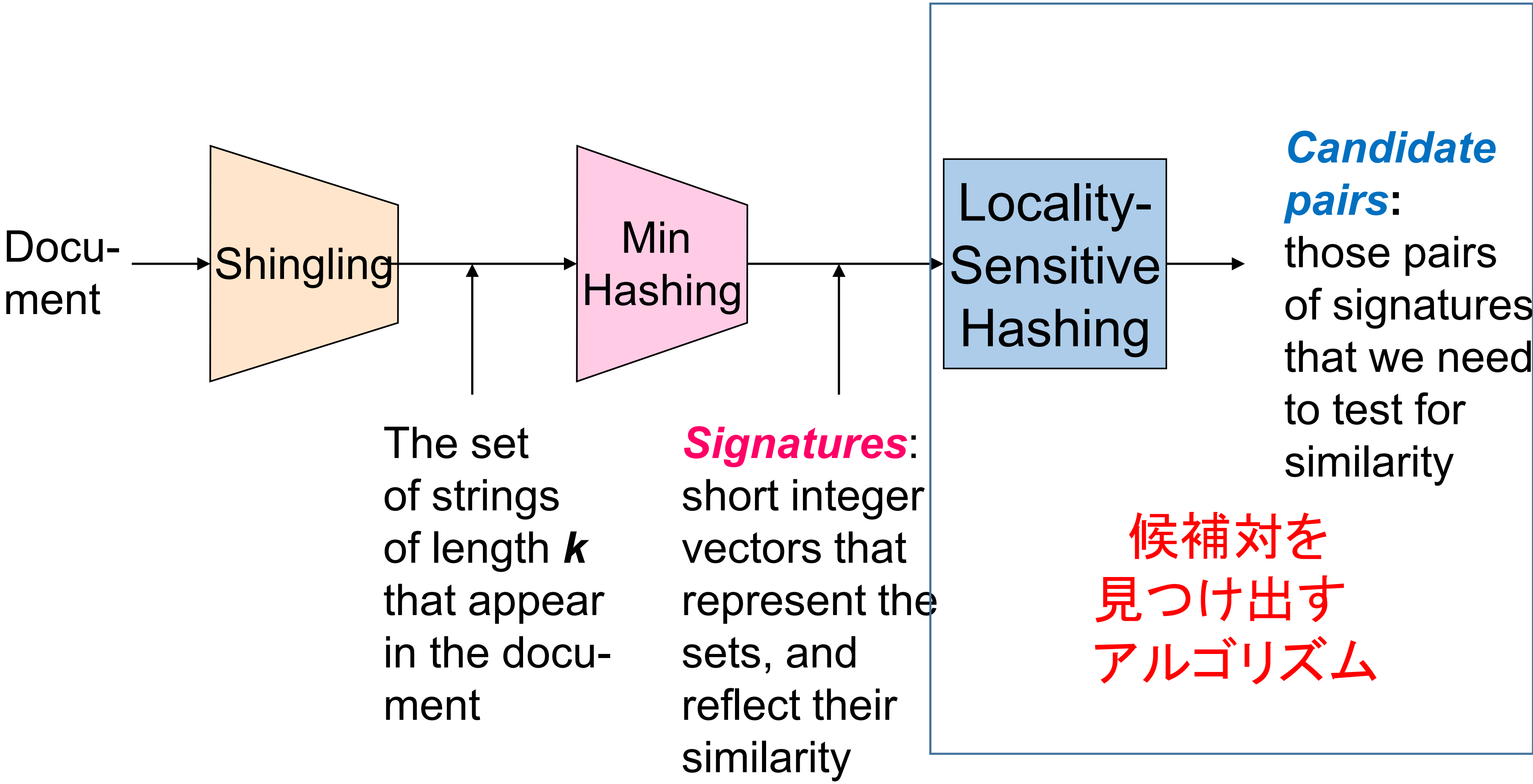
- すべての対の類似度を計算したい場合（処理の短縮が不可能 → 並列処理を用いるしかない）

- ある類似度以上の対だけを探したい場合が多い（例：似ている文書・画像を探す）

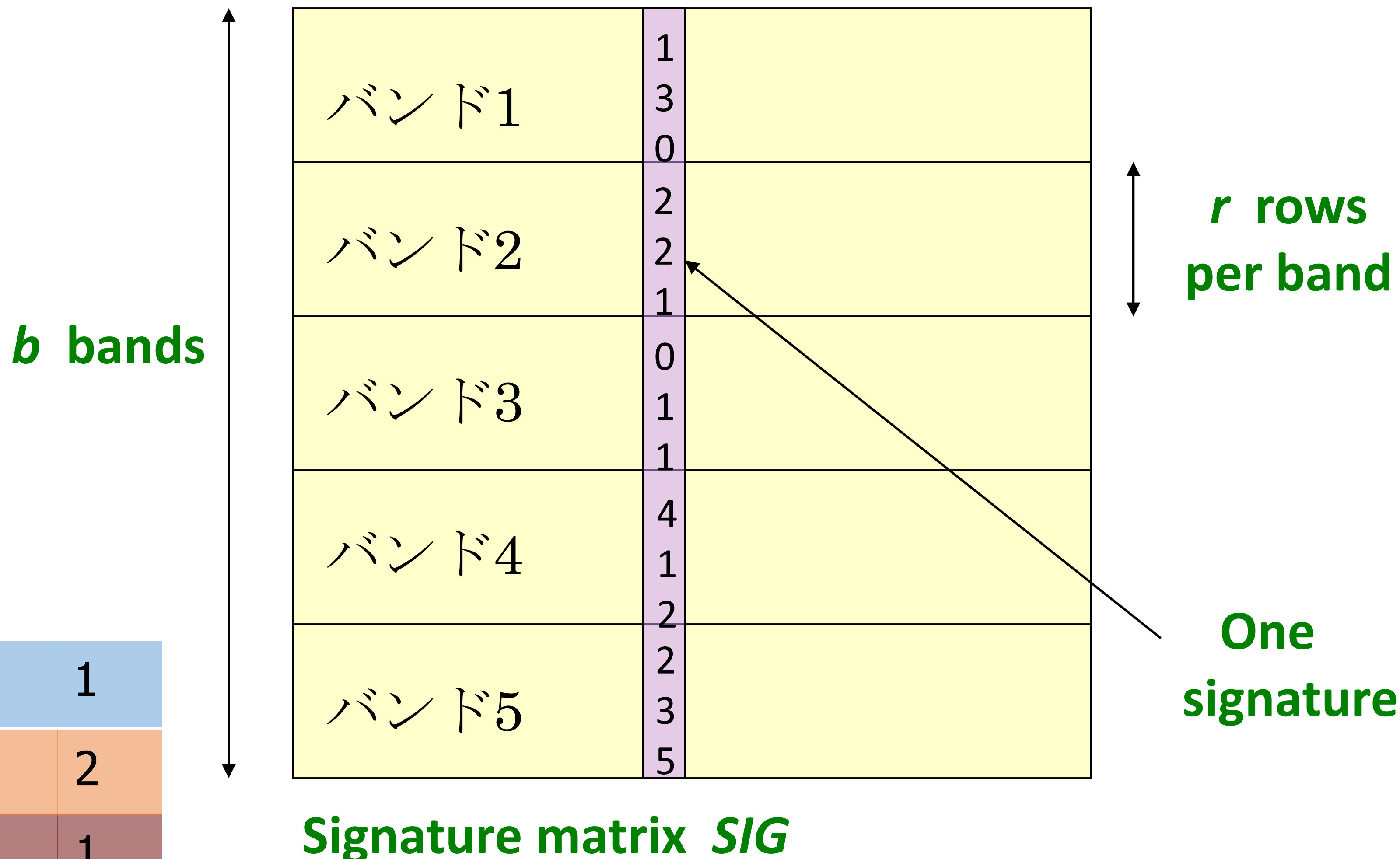
すべての対について調べるのではなく，
類似していそうな対のみに着目すればよい！

局所性鋭敏型ハッシング(locality-sensitive hashing, LSH)

局所性鋭敏型ハッシング(locality-sensitive hashing, LSH)

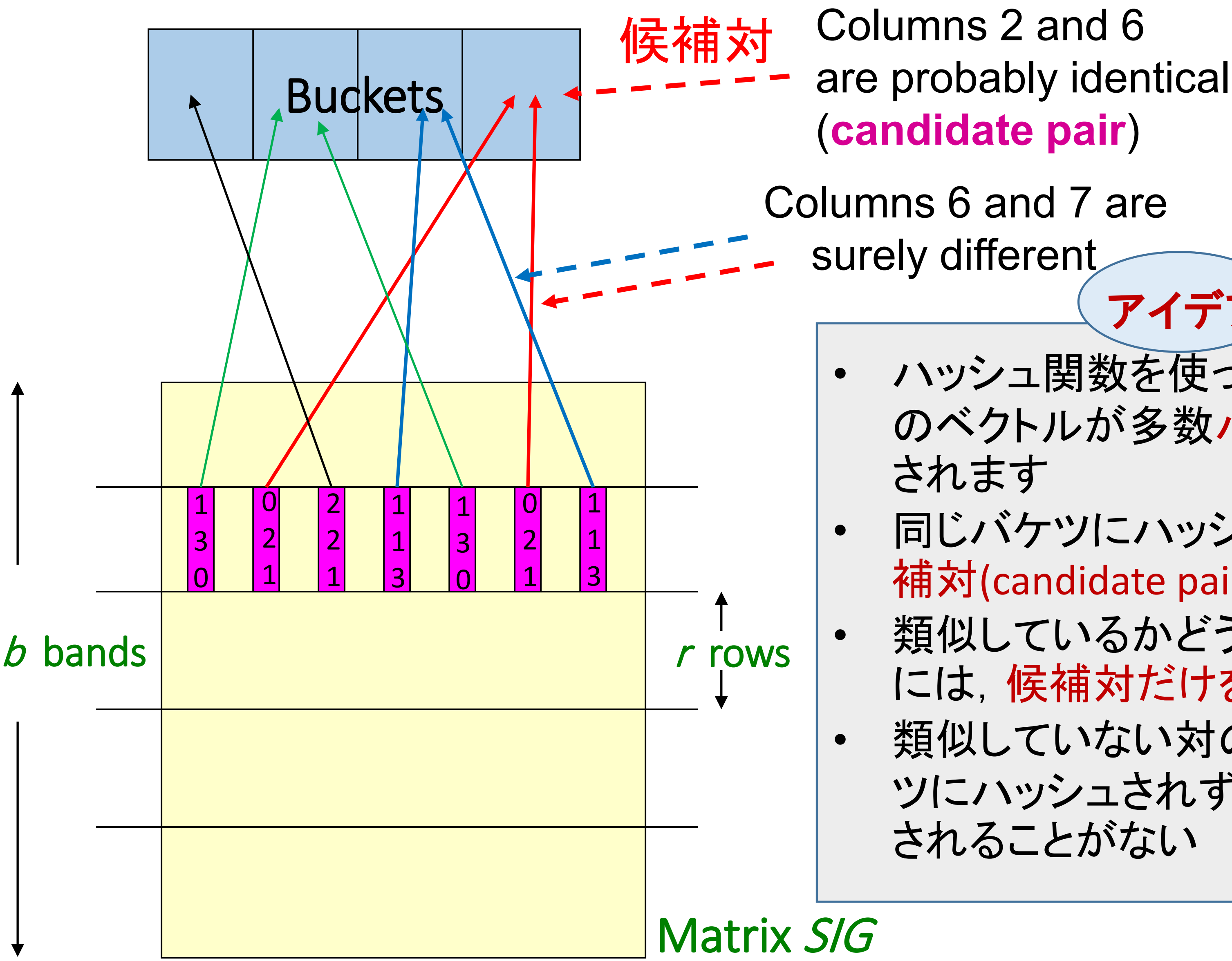


ミンハッシュシグネチャーにおけるLSH



2	1	4	1
1	2	1	2
2	1	2	1

ミンハッシュシグネチャーにおけるLSH



アイデア

- ハッシュ関数を使って，同じバンドのベクトルが多数バケツへハッシュされます
- 同じバケツにハッシュされた対が候補対(candidate pair)
- 類似しているかどうかを調べるためには，候補対だけを考慮する
- 類似していない対の多くが同じバケツにハッシュされず，これ以上処理されることがない

ミンハッシュシグネチャーにおけるLSH（詳細）

- 重要なポイント： 各アイテムを数回（*b* 回，バンド毎）ハッシュする

  類似するアイテムが，異なるアイテムよりも，より確かに（少なくても一回）同じバケツにハッシュされることが期待したい

  ＊あるバンドの中に一致しない2つの列が別のバンドで一致する可能性がある

  ＊2つの列が類似していればいるほど，別のバンドにおいても同一である可能性が高くなる

バンドを作成する戦略は，類似している列を候補対にする可能性を高める

ミンハッシュシグネチャーにおけるLSH（詳細）

- すべてのバンドに対して同じハッシュ関数を使うこともできるが，　それぞれのバンドに対して別のバケツ配列を使うことで，異なるバンドで同じベクトルを持つ列が違うバケツにハッシュされる

ミンハッシュシグネチャーにおけるLSH（詳細）

- すべてのバンドに対して同じハッシュ関数を使うこともできるが，それぞれのバンドに対して別のバケツ配列を使うことで，異なるバンドで同じベクトルを持つ列が違うバケツにハッシュされる
- 偽陽性(false positive)： 類似していないにも関わらず同じバケツにハッシュされる対 ➔ その割合が低いほどよい

ミンハッシュシグネチャーにおけるLSH（詳細）

- すべてのバンドに対して同じハッシュ関数を使うこともできるが，それぞれのバンドに対して別のバケツ配列を使うことで，異なるバンドで同じベクトルを持つ列が違うバケツにハッシュされる
- 偽陽性(false positive)：類似していないにも関わらず同じバケツにハッシュされる対 ➔ その割合が低いほどよい
- 偽陰性(false negative)：真に類似する対の多くが，少なくとも1つのハッシュ関数において同じバケツにハッシュされることが望ましい．そうでないものは，偽陰性(false negative) と呼ばれ，真に類似する対のほんの一部であって欲しい

バンド作成戦略の解析（Analysis of the Banding Technique）

- ある文書対(C_1, C_2)のジャッカール類似度がsであるとする

それらの文書に対するミンハッシュシグネチャーが，”シグネチャー行列のどれか1つの行において一致する確率は sである”

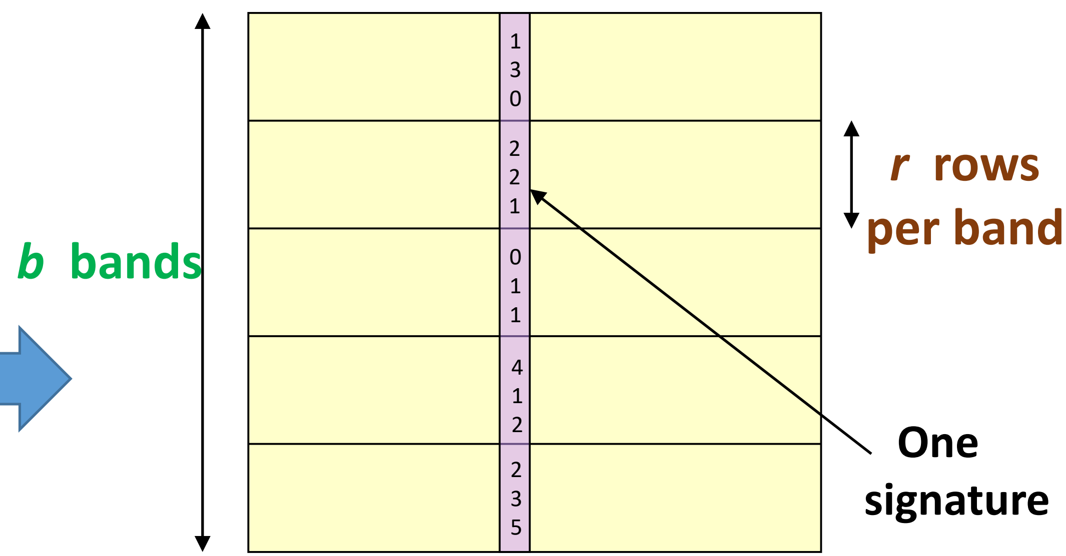
$s = sim(C_1, C_2) = 0.8$

b = 20（バンドの数）

r = 5（各バンドの行の数各）

2	1	4	1
1	1	1	2
2	2	2	1

ミンハッシュシグネチャー行列*SIG*



Signature matrix *SIG*

バンド作成戦略の解析（Analysis of the Banding Technique）

- ある文書対(C_1, C_2)のジャッカール類似度がsであるとする

それらの文書に対するミンハッシュシグネチャーが，”シグネチャー行列のどれか1つの行において一致する確率は s である”

$s = sim(C_1, C_2) = 0.8$　　b = 20（バンドの数)　　r = 5（各バンドの行の数各）

- シグネチャーが特定の1個のバンドのすべての行で一致する確率

$$s^r = (0.8)^5 = 0.328$$

バンド作成戦略の解析（Analysis of the Banding Technique）

- ある文書対(C_1, C_2)のジャッカール類似度がsであるとする

それらの文書に対するミンハッシュシグネチャーが，”シグネチャー行列のどれか1つの行において一致する確率は sである”

s = sim(C_1, C_2) = 0.8　b = 20（バンドの数)　r = 5（各バンドの行の数各）

- シグネチャーが特定の1個のバンドのすべての行で一致する確率

$$s^r = (0.8)^5 = 0.328$$

- シグネチャーが特定の1個のバンドの少なくとも1つの行で一致しない確率

$$1 - s^r = 1 - (0.8)^5 = 0.672$$

バンド作成戦略の解析(Analysis of the Banding Technique)

- ある文書対(C_1, C_2)のジャッカール類似度がsであるとする

それらの文書に対するミンハッシュシグネチャーが, ”シグネチャー行列のどれか1つの行において一致する確率は sである”

$s = sim(C_1, C_2) = 0.8$　　b = 20 (バンドの数)　　r = 5 (各バンドの行の数各)

- シグネチャーが特定の1個のバンドのすべての行で一致する確率

$$s^r = (0.8)^5 = 0.328$$

- シグネチャーが特定の1個のバンドの少なくとも1つの行で一致しない確率

$$1 - s^r = 1 - (0.8)^5 = 0.672$$

- シグネチャーがどのバンドの少なくとも1つの行においても一致しない確率

$$(1 - s^r)^b = (1 - 0.328)^{20} = 0.00035$$

偽陰性率: 0.035%

バンド作成戦略の解析（Analysis of the Banding Technique）

- ある文書対(C_1, C_2)のジャッカール類似度がsであるとする

それらの文書に対するミンハッシュシグネチャーが，”シグネチャー行列のどれか1つの行において一致する確率は s である”

$s = sim(C_1, C_2) = 0.8$　　b = 20（バンドの数）　　r = 5（各バンドの行の数各）

- シグネチャーが特定の1個のバンドのすべての行で一致する確率

$$s^r = (0.8)^5 = 0.328$$

- シグネチャーが特定の1個のバンドの少なくとも1つの行で一致しない確率

$$1 - s^r = 1 - (0.8)^5 = 0.672$$

- シグネチャーがどのバンドの少なくとも1つの行においても一致しない確率

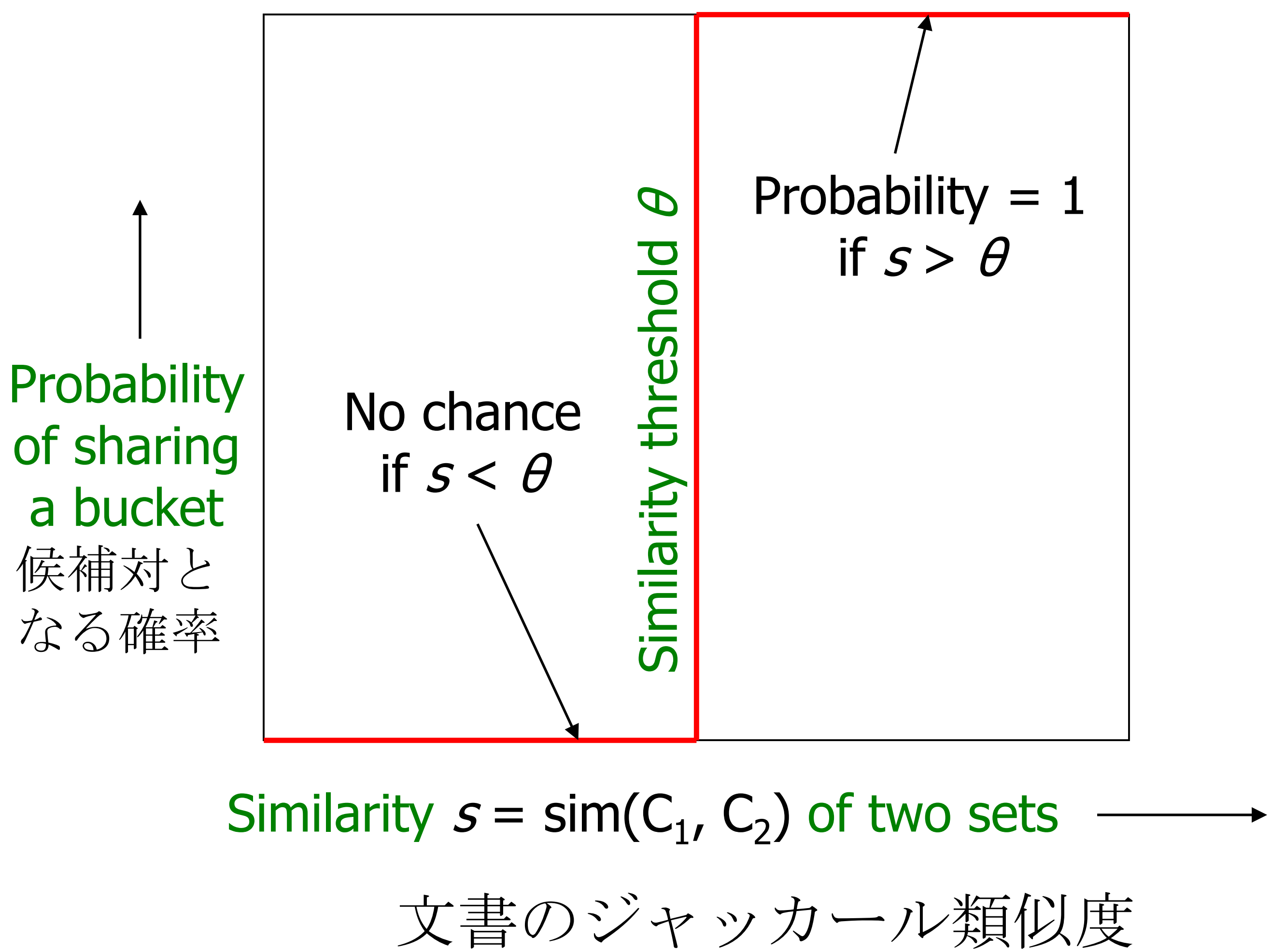
$$(1 - s^r)^b = (1 - 0.328)^{20} = 0.00035$$

- シグネチャーが1つ以上のバンドのすべての行で一致する確率

$$1 - (1 - s^r)^b = 1 - (1 - 0.328)^{20} = 0.99965$$

99.965%　実際に類似している対を候補対にする

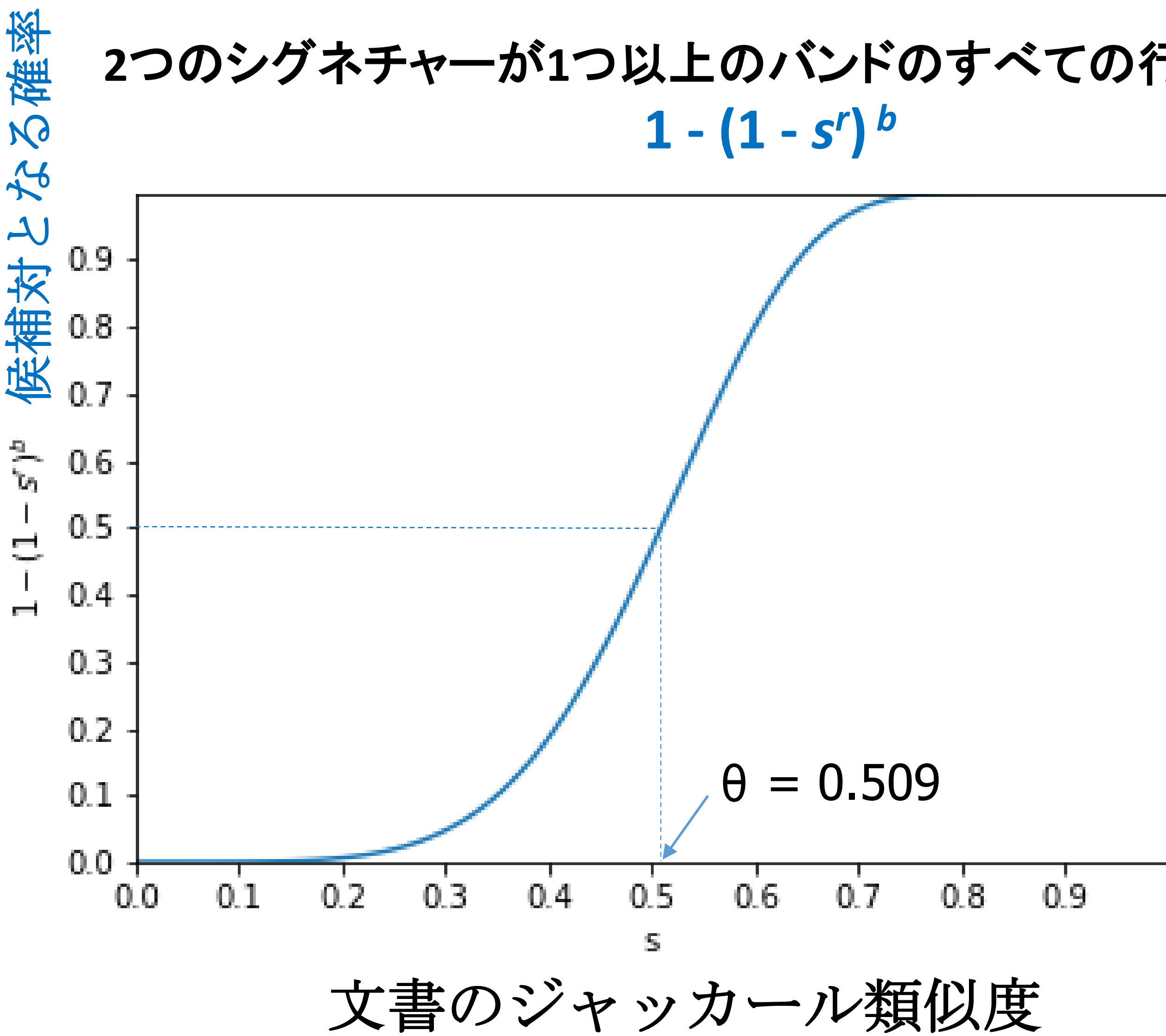
バンド作成戦略の解析（Analysis of the Banding Technique）



バンド作成戦略の解析（Analysis of the Banding Technique）

2つのシグネチャーが1つ以上のバンドのすべての行で一致する確率

$$1 - (1 - s^r)^b$$

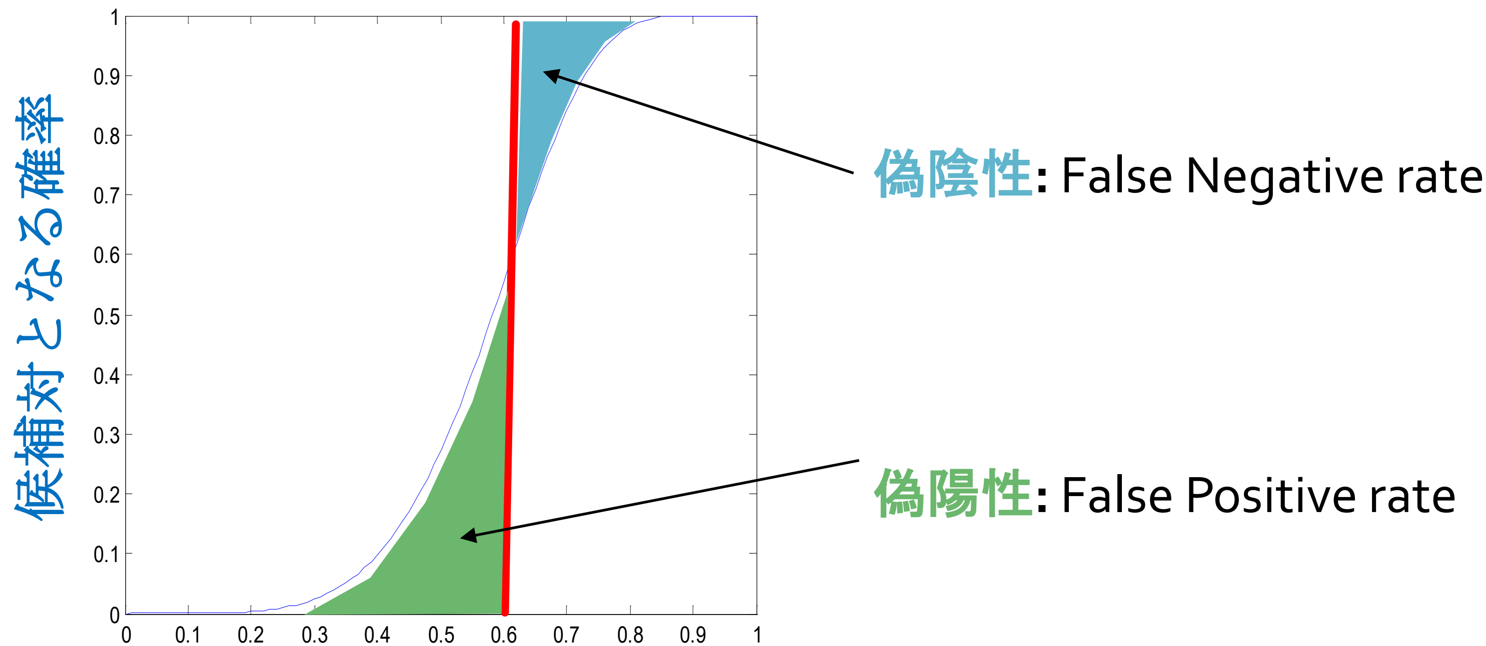


s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

b = 20,　r = 5

バンド作成戦略の解析（Analysis of the Banding Technique）

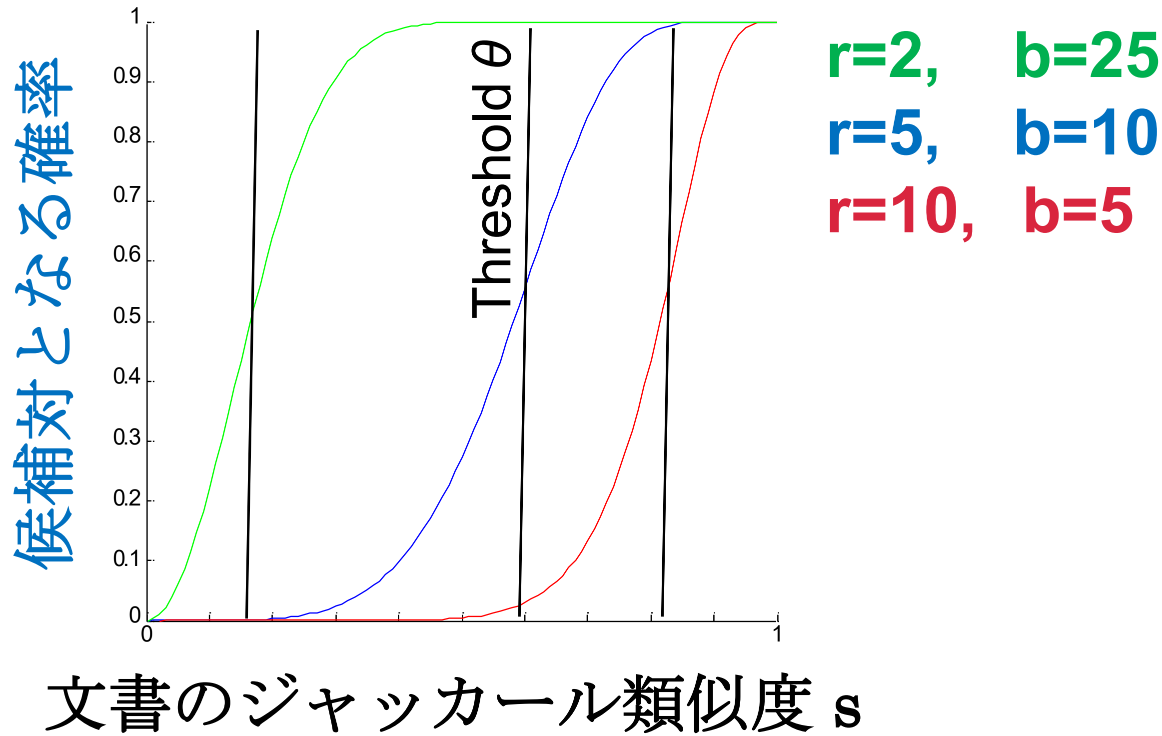
50 minhash-functions (r=5, b=10)



文書のジャッカール類似度 s

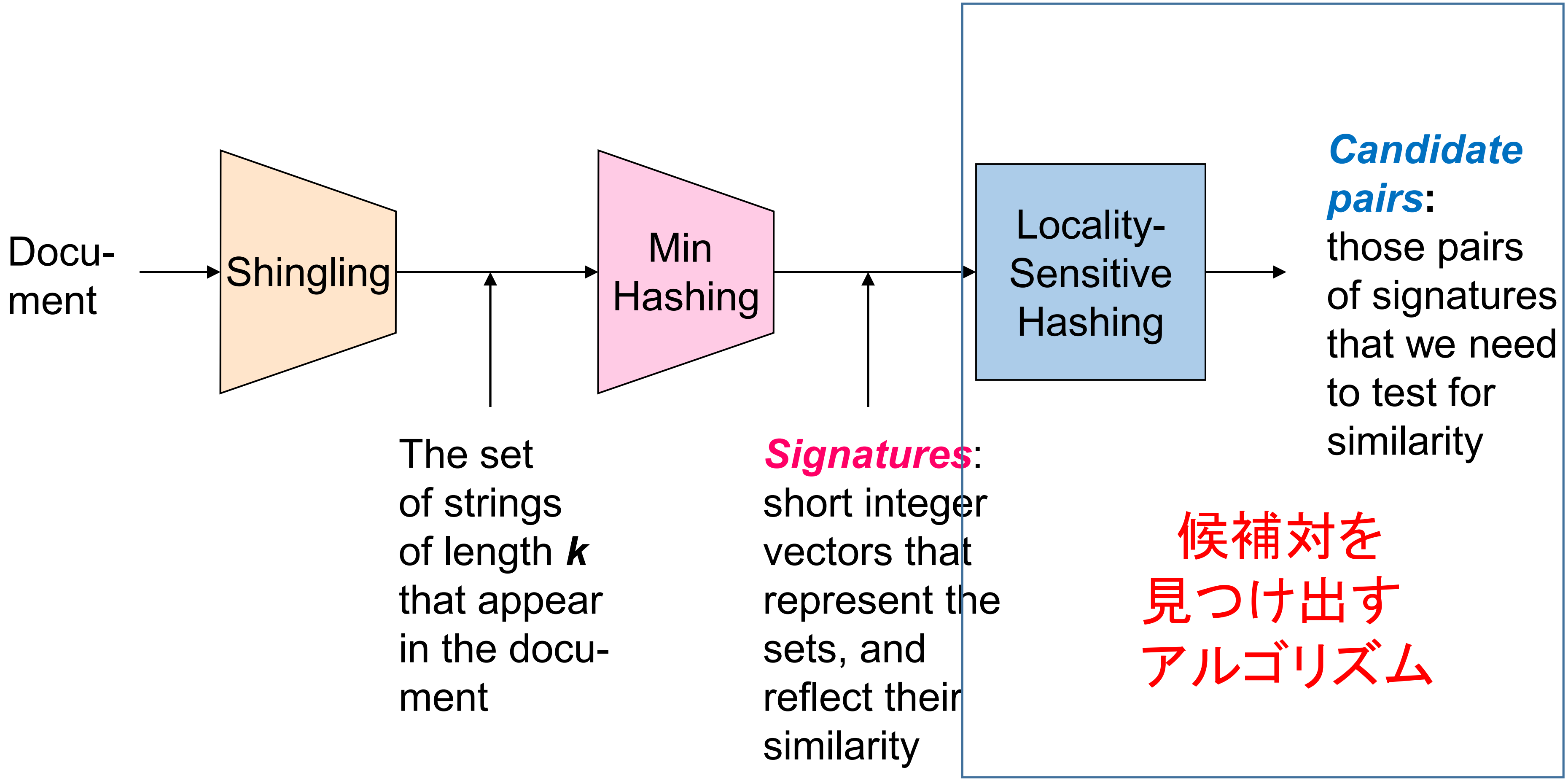
$\theta \sim (1/b)^{1/r}$

バンド作成戦略の解析（Analysis of the Banding Technique）



50 minhash-functions (*r * b = 50*)

局所性鋭敏型ハッシング(locality-sensitive hashing, LSH)



局所性鋭敏型ハッシング(locality-sensitive hashing, LSH)

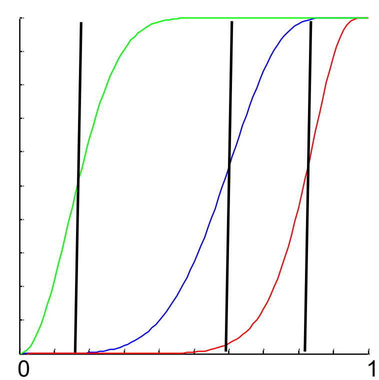
(1)　類似対の定義：類似した文書が，望ましい"類似対"としてみなされるべき度合いを定義する類似対の閾値 t を1つ選択する

　＊バンド b の数と $br = n$(シグネチャーの行数)となるような行の数 r を選択

　　　　S曲線の閾値 θ の近似値は~ $(1/b)^{1/r}$ となる

　＊もし偽陰性を避けることが重要ならば，t よりも低い閾値θを与えるような b と r を選ぶ

　＊もし処理の高速性が重要で，偽陽性を制限したければ，より高い閾値θになるように b と r を設定する



局所性鋭敏型ハッシング(locality-sensitive hashing, LSH)

(1)　類似対の定義：類似した文書が，望ましい“類似対”としてみなされるべき度合いを定義する類似対の閾値 *t* を1つ選択する

　＊バンド*b*の数と$br = n$(シグネチャーの行数)となるような行の数 *r* を選択

　　　　S曲線の閾値 θ の近似値は ~ $(1/b)^{1/r}$ となる

　＊もし偽陰性を避けることが重要ならば，*t* よりも低い閾値θを与えるような *b* と *r* を選ぶ

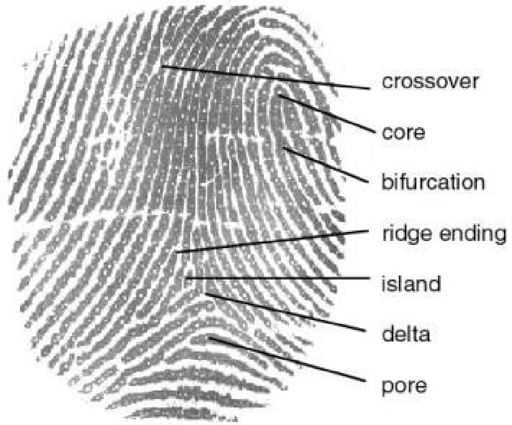
　＊もし処理の高速性が重要で，偽陽性を制限したければ，より高い閾値θになるように *b* と *r* を設定する

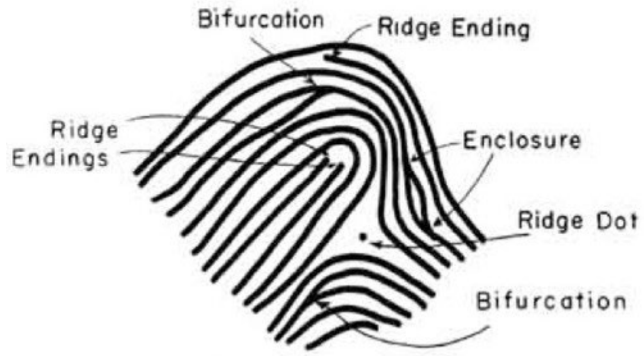
(2)　LSHの手法を適用して，候補対を作成する

(3)　各候補対のシグネチャーを調べ，一致している成分の割合が少なくとも *t* 以上であるかを決定する

(4) （オプション）シグネチャーが十分に類似している場合には，文書そのものを調べることで，たまたま類似したシグネチャーを持っているのではなく，本当に類似しているかどうかをチェックすることもできる

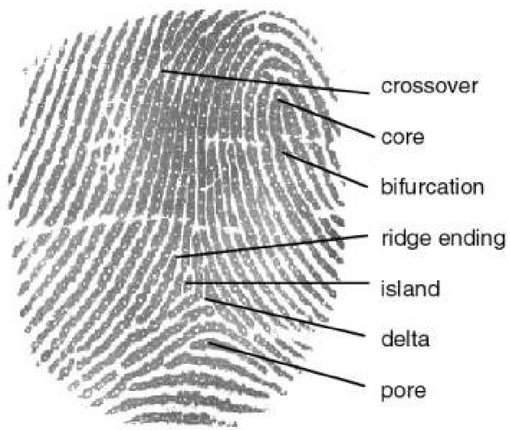
指紋の照合 (Matching Fingerprints)

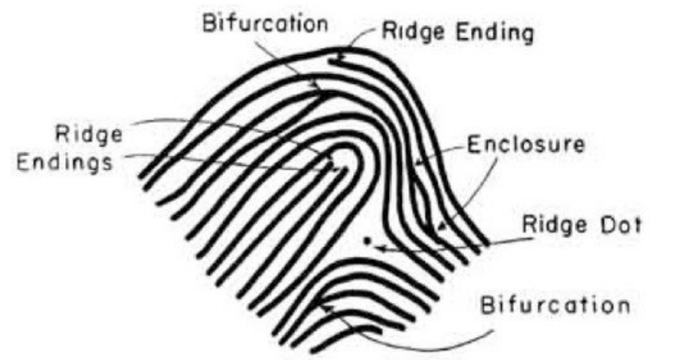




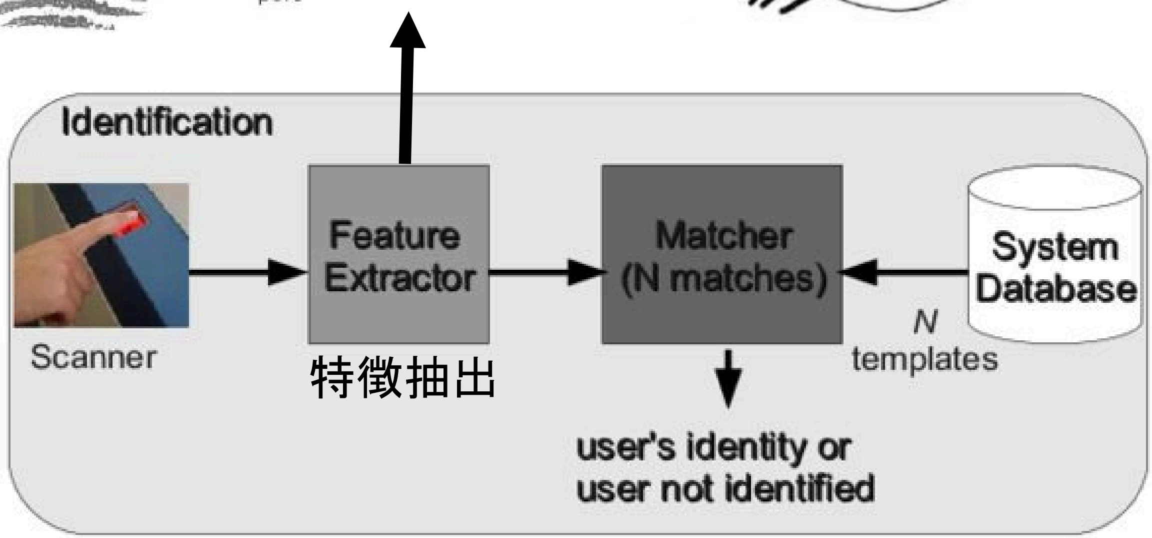
特徴点
(minutia)

指紋の照合 (Matching Fingerprints)



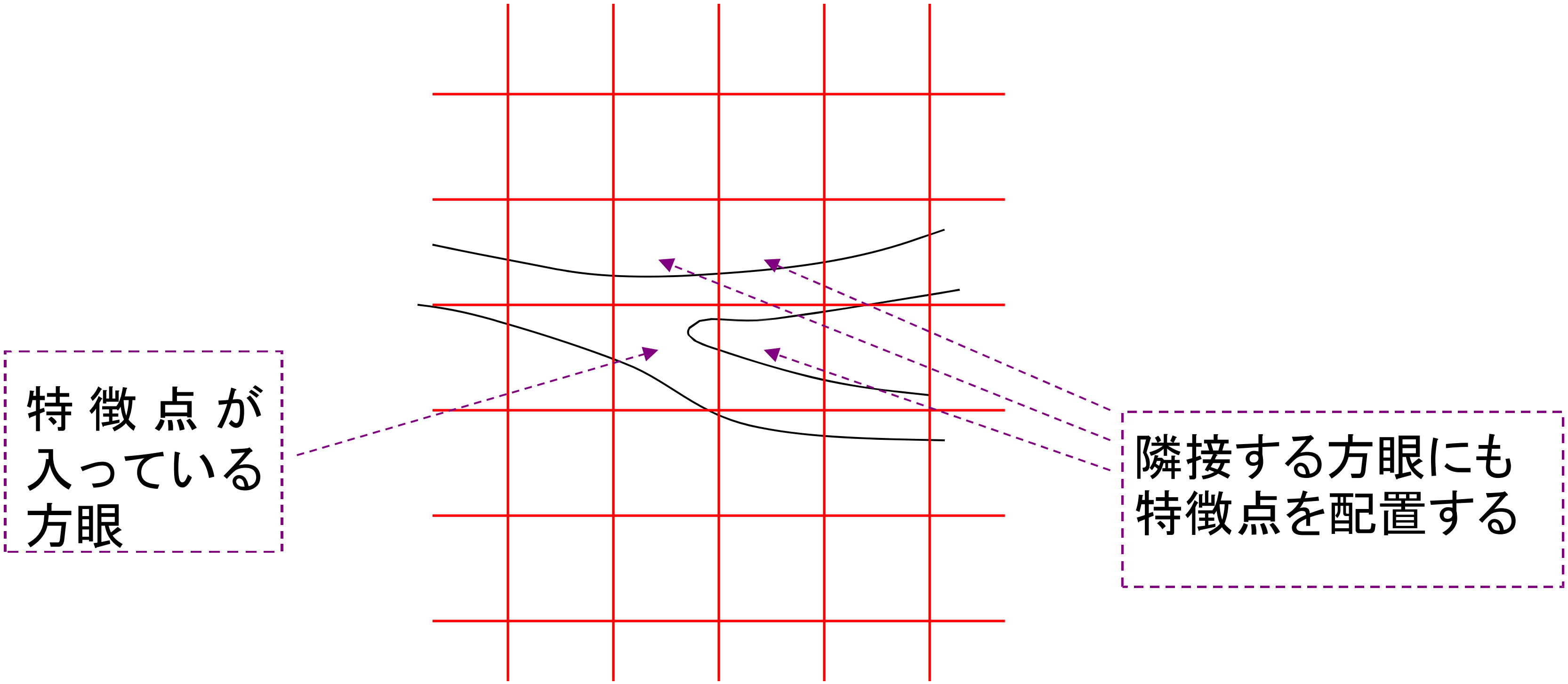


特徴点
(minutia)



指紋照合の流れ

指紋の照合 (Matching Fingerprints)



指紋の照合 (Matching Fingerprints)

- ランダムに選んだ1つの指紋に対して，1つのランダムな方眼で特徴点を見つけられる確率を20%とする

1	0	0	1	0	0	1	0	…	0

指紋の照合 (Matching Fingerprints)

- ランダムに選んだ1つの指紋に対して，1つのランダムな方眼で特徴点を見つけられる確率を20%とする

1	0	0	1	0	0	1	0	…	0

- もし2つの指紋が同じ指のものである場合，片方の指紋で与えられた方眼中に特徴点が含まれるならば，別の指紋の方にも同じものがある確率を80%とする

1	0	0	1	0	0	1	0	…	0
1	0	0	0	0	0	1	0	…	0

指紋の照合 (Matching Fingerprints)

- ランダムに選んだ3つの方眼で定義されるハッシュ関数 f

  ＊f のすべての3つの方眼に特徴点を持った指紋は，
  1つのバケツに送られる（“f のバケツ”） → 候補対となる

  ＊このような異なる関数 f を（例えば）1024個用意して，
  データベースに入っているすべての指紋に対して，
  各関数 f のバケツを事前に計算する

  ＊照合したい新しい指紋があったときに，どのバケツに
  それが属するかを決定し，それらのバケツに含まれる
  すべての指紋と比較する

指紋の照合 (Matching Fingerprints)

照合したい指紋とデータベースに格納された何千万の指紋1つ1つと比較することなしに，適度な確率で合致するものを手に入れるためには，**f 関数がいくつ必要になるか**？

指紋の照合 (Matching Fingerprints)

照合したい指紋とデータベースに格納された何千万の指紋1つ1つと比較することなしに，適度な確率で合致するものを手に入れるためには，ｆ関数がいくつ必要になるか？

（1）**異なる指**の2つの指紋が関数ｆに対応するバケツに含まれる確率

$$(0.2)^6 = 0.000064$$

＊ランダムに選んだ1つの指紋に対して，1つのランダムな方眼で特徴点を持っている確率　→　20%

＊2つの指紋の両方がｆに関連付けられた3つの方眼のそれぞれの中に特徴点を含む場合に限り両者が同じバケツに送られる

＊6つの独立した事象それぞれの発生確率は0.2

指紋の照合 (Matching Fingerprints)

（２） **同じ指**からとられた2つの指紋が f に対応する同じバケツに取り込まれる確率

＊最初の指紋がfに属す3つの方眼に特徴点を持っている確率

$(0.2)^3 = 0.008$

理由：ランダムに選んだ1つの指紋に対して，1つのランダムな方眼で特徴点を持っている確率　→　20%　x 3つの方眼

指紋の照合 (Matching Fingerprints)

（２） **同じ指**からとられた2つの指紋がｆに対応する同じバケツに取り込まれる確率

＊最初の指紋がfに属す3つの方眼に特徴点を持っている確率

$$(0.2)^3 = 0.008$$

＊もし最初の指紋がｆに属す3つの方眼に特徴点を持っていた場合，もう1つの指紋も同様である確率

$$(0.8)^3 = 0.512$$

理由：　もし2つの指紋が同じ指のものである場合，片方の指紋で与えられた方眼中に特徴点が含まれるならば，別の指紋の方にも同じものがある確率　→　80%　＋　3つの独立した事象

指紋の照合 (Matching Fingerprints)

（２） **同じ指**からとられた2つの指紋が f に対応する同じバケツに取り込まれる確率

＊最初の指紋がfに属す3つの方眼に特徴点を持っている確率

$$(0.2)^3 = 0.008$$

(ランダムに選んだ1つの指紋に対して，1つのランダムな方眼で特徴点を持っている確率 → 20% x 3つの方眼)

＊もし最初の指紋が f に属す3つの方眼に特徴点を持っていた場合，もう1つの指紋も同様である確率

$$(0.8)^3 = 0.512$$

＊もし2つの指紋が**同じ指**のものであれば，**両方が f のバケツに送られる確率**

$$(0.2)^3(0.8)^3 = 0.008 \times 0.512 = 0.004096$$

指紋の照合 (Matching Fingerprints)

例： ランダムに選んだ1,024個の f 関数を使う

- 同じ指からの指紋を一緒に少なくとも1つのバケツに置く確率

$$1 - (1 - (0.2)^3(0.8)^3)^{1024} = 1 - (1 - 0.004096)^{1024}$$
$$= 0.985$$

＊ 偽陰性: 1.5%

指紋の照合 (Matching Fingerprints)

例：　ランダムに選んだ1,024個の f 関数を使う

- 同じ指からの指紋を一緒に少なくとも1つのバケツに置く確率

$$1 - (1 - (0.2)^3(0.8)^3)^{1024} = 1 - (1 - 0.004096)^{1024}$$
$$= 0.985$$

＊　偽陰性:　　1.5%

- 違う指の2つの指紋が同じバケツに置かれる確率

$$1 - (1 - (0.2)^6\ )^{1024} = 1 - (1 - 0.000064)^{1024} = 0.063$$

＊　偽陽性:　　6.3%

References

A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions,” Comm. ACM 51:1, pp. 117–122, 2008.

A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing,” Proc. Intl. Conf. on Very Large Databases, pp. 518–529, 1999.

J. Leskovec, A. Rajaraman, J. D. Ullman, “Mining of Massive Datasets” 3ed, Cambridge University Press, 2020.