自然言語処理における
ハッシングを用いた近傍探索

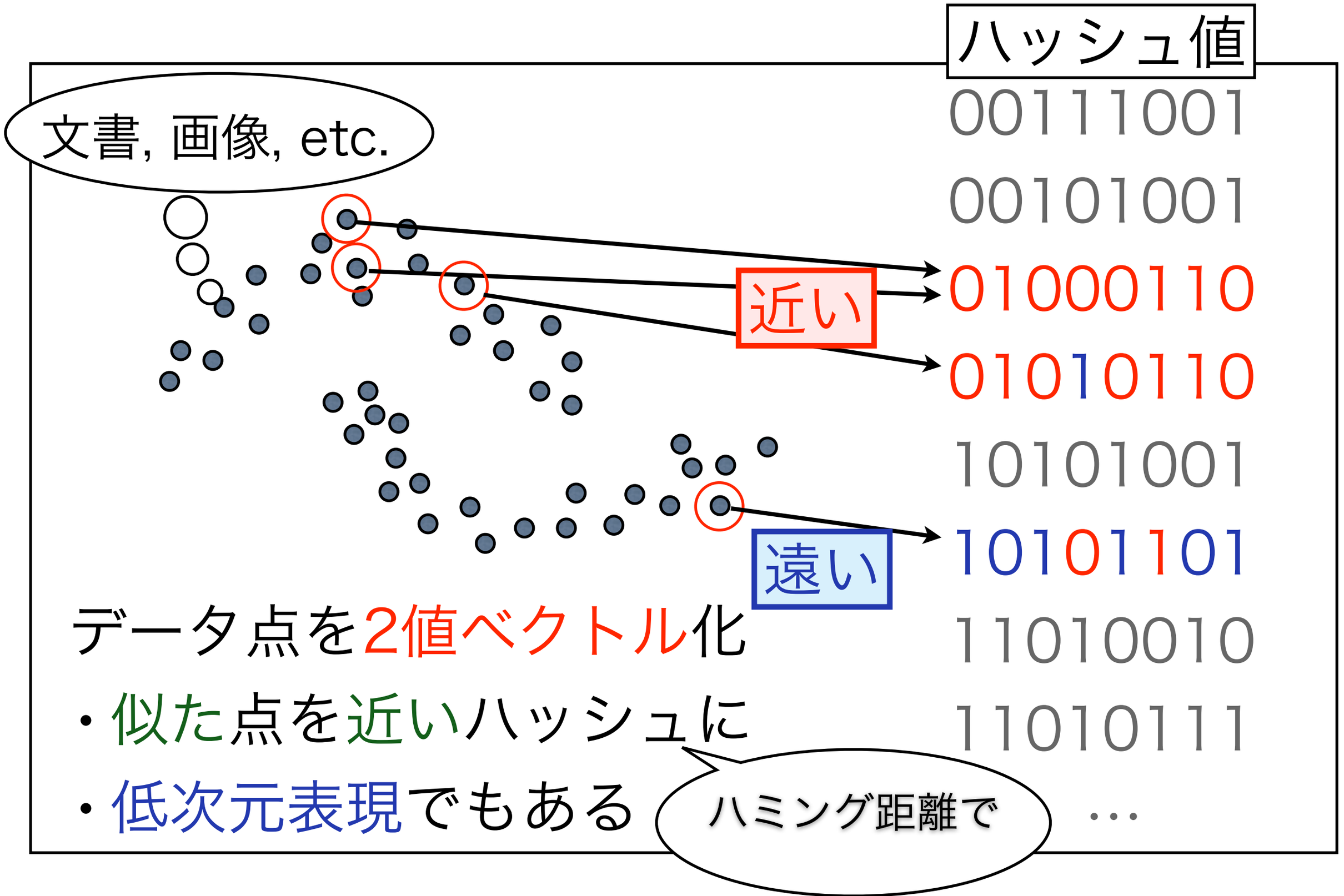
b-Bit Minwise Hashing, Anchor Graph Hashing の比較と Margin AGH の提案

得居誠也†, 佐藤一誠‡, 中川裕志‡

NLP 若手の会
2011/09/22

†東京大学大学院 情報理工学系研究科
‡東京大学 情報基盤センター

近傍探索のためのハッシング



b-Bit Minwise Hash

- 対象: 集合データ (e.g. Bug of Words)
- **Jaccard 係数の高速・省メモリな近似法**
  - cf. MinHash
- ランダム置換に基づく
  - ビット数を増やせば増やすほど精度も上がる
  - cf. LSH (Locality Sensitive Hashing)
- コピー検出など，**ほぼ一致する文書の発見に有効**
  - Jaccard 係数が高いペアに対する推定精度が高い

Ping Li and Arnd Christian König.
b-Bit Minwise Hash. WWW2010

b-Bit Minwise Hash アルゴリズム
(b = 1 の場合, heuristic 版)

前処理 (単語集合 $\Omega = \{0,...,d-1\}$, ビット数 r)

ランダム置換 $\pi_j : \Omega \to \Omega$, $j \in \{1,...,r\}$ を作る

入力 (文書 $S \subseteq \{0, ..., d-1\}$)

出力 (ハッシュベクトル $y \in \{0, 1\}^r$)

For each $j \in \{1,...,r\}$:

$y_j \leftarrow \min \pi_j(S)$ の最下位ビット

MinHash の仕掛け

- Jaccard 係数 (resemblance): $R(S, S') = \frac{|S \cap S'|}{|S \cup S'|}$.
- ランダム置換 π: $\Omega \to \Omega$, 文書 S, $S' \subseteq \Omega$
  - $\min \pi(S) = \min \pi(S')$
    $\Leftrightarrow$ $\min \pi(S \cup S')$ を与える単語 $w \in S \cup S'$ が S, S' の両方に含まれる
  - π はランダムなので, 右辺が真となる確率はランダムに選んだ $w \in S \cup S'$ が $S \cap S'$ に属す確率
  - つまり $P(\min \pi(S) = \min \pi(S')) = R(S, S')$
- $\min \pi(S)$ の下位 b ビットだけからも, 集合サイズの情報を用いて衝突確率の分を間引くことで R を推定できる: b-Bit Minwise Hashing

Anchor Graph Hashing

- 対象: 任意のデータセット
- **Anchor Graph** を用いて，低次元スパースな空間に落としてから線形射影 (PCA)
- 多様体学習の手法
  - cf. Spectral Hashing, Spectral Clustering
  - **少ないビット数で**高性能
  - 「そこそこ似てる」くらいのデータも探索できる

Wei Liu, Jun Wang, Sanjiv Kumar and Shih-Fu Chang.
Hashing with Graphs. ICML2011

[AGH] データの空間

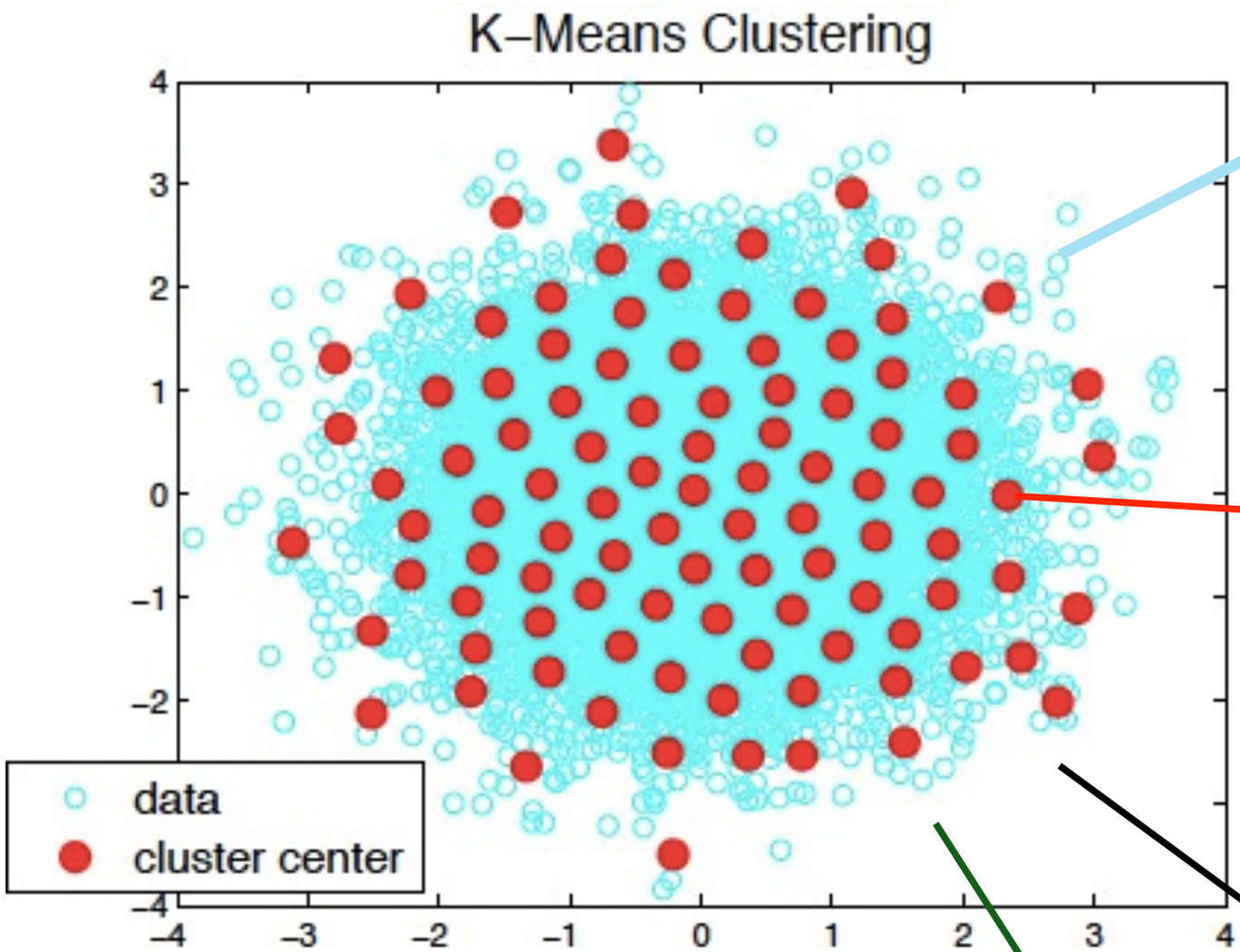


Figure: [Liu+, 2010]

既知のデータ

$\mathbf{x}_i$ $(i \in \{1,\ldots,n\})$

アンカー

$\mathbf{u}_j$ $(j \in \{1,\ldots,m\})$

(K-means 等で選ぶ)

距離関数 $\mathcal{D}$

類似度: **熱核**

$h(\mathbf{x}, \mathbf{x}') = \exp(-\mathcal{D}(\mathbf{x}, \mathbf{x}')^2/t)$

[AGH] データのスパース表現行列

	$\mathbf{u}_1$	$\mathbf{u}_2$	$\mathbf{u}_3$	$\cdots$	$\mathbf{u}_m$
$\mathbf{x}_1$	0.8	0.2	0	$\cdots$	0
$\mathbf{x}_2$	0.3	0	0.7	$\cdots$	0
$\mathbf{x}_n$					

$= Z$

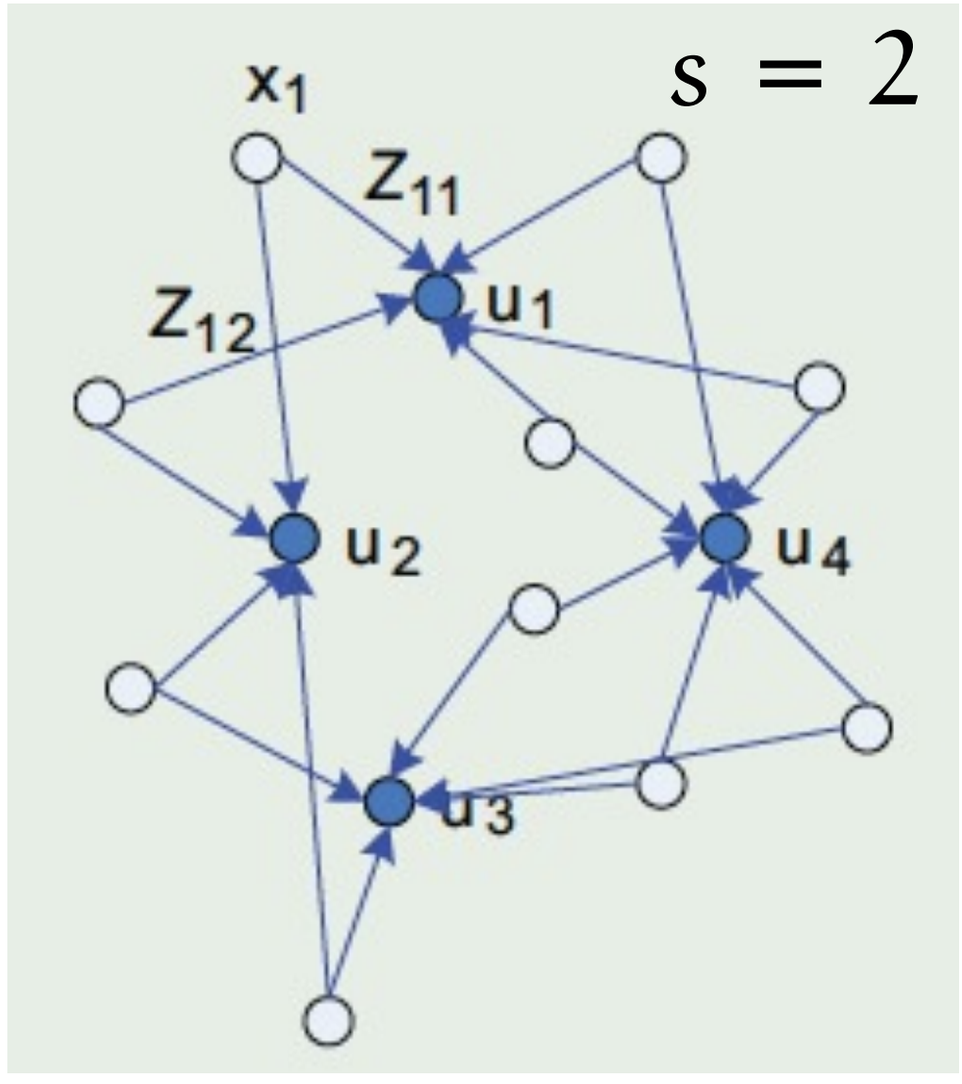


Figure: [Liu+, 2010]

- 行ごとにつくる
- 各要素は $\mathbf{x}_i$ と $\mathbf{u}_j$ の類似度に比例
- 大きい方から s 個以外は 0 に
- 行方向に正規化 → データからアンカーへの遷移確率行列

$\mathbf{x}$ からの遷移確率を $z(\mathbf{x})$, $Z^\top$ の第 i 列を $z_i = z(\mathbf{x}_i)$ とおく

[AGH] アンカーグラフ

- アンカーからデータへも Z の成分に比例する確率での遷移を考え, 2-step の遷移でデータからデータへの遷移とする
- 遷移確率行列は $\Lambda = \mathrm{diag}(\mathbf{1}^\top Z)$ として

$$\hat{A} = Z\Lambda^{-1}Z^\top$$

後半の遷移に関する正規化項

これを類似度行列とする
= **アンカーグラフ**

- $Z\Lambda^{-1/2}$ を計画行列とするカーネル (低次元)

[AGH] ハッシュ値最適化問題

$$\min \frac{1}{2}\sum_{i,j=1}^{n} \|Y_i - Y_j\|^2 \hat{A}_{ij} = \sum_{k=1}^{r} \mathbf{y}_k^\top L \mathbf{y}_k$$

$$\text{s.t. } Y \in \mathbf{R}^{n\times r}, \mathbf{1}^\top Y = 0, Y^\top Y = nI.$$

$$L = \mathrm{diag}(\hat{A}\mathbf{1}) - \hat{A}$$
$$= I - \hat{A}$$

アンカーグラフの
グラフラプラシアン

解 $\mathbf{y}_1, \ldots, \mathbf{y}_r$ は L の固有ベクトル

- 1 つ目は $\mathbf{1}$ なので捨て, 2, ..., r+1 個目を使う
  - $\mathbf{1}$ に直交 → ハッシュ値に偏りがない

★ $\hat{A}$ と L は固有ベクトルが同じ

★ 解は $Z\Lambda^{-1/2}$ を PCA して得られる

$O(s^2n+m^3+srn)$
時間で計算できる

$$Y = ZW$$

PCA の射影行列を
W とおく

[AGH] ハッシュ関数への拡張

$$\min \frac{1}{2}\sum_{k=1}^{r}\int_{\mathcal{M}}\|\nabla y_k\|^2\,dp(x)$$

$$\text{s.t. } y:\mathcal{M}\to\mathbf{R}^r,\ E[y]=0,\ \mathrm{Var}[y]=I.$$

$\mathcal{M}$: データ多様体

(確率密度 $p(x)$)

$$\int_{\mathcal{M}}\|\nabla y\|^2\,dp=\langle y,\mathcal{L}y\rangle_{L^2(\mathcal{M},p)}$$

$\mathcal{L}$: $\mathcal{M}$ 上の Laplace-Beltrami 作用素

解 $y_1, \ldots, y_r$ は $\mathcal{L}$ の固有関数

Nyström 法: $n\to\infty$ で次の関数で近似できる

$$y_k(x)\approx\frac{1}{\sigma_k}\sum_{i=1}^{n}\hat{A}(x,x_i)Y_{ik}$$

$$=w_k^{\top}z(x)$$

[Bengio+, 04]

σ : $\hat{A}$の固有値

$\hat{A}(x,x_i)=z(x)^{\top}\Lambda^{-1}z_i$

$\hat{A}$, Y を展開

AGH でマージンを考える

- AGH の目的関数: 0 付近を特別視しない

  → **「近いのに異符号」「遠いのに同符号」が普通に起きる**

- マージンを取りたい

[Liu+, 2011] では AGH の各ビットに対してこうした境界エラーを緩和するビットを追加: 2-AGH

cf. **MMC** (Max Margin Clustering)

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}[1-|f(x_i)|]_+$$

$$\text{s.t. } f(x) = w^\top x + b, -c \le \sum_i f(x_i) \le c.$$

[Xu+, 2005]

$$[x]_+ = \max\{x, 0\}$$

$w = \nabla f$

- 勾配を小さくしながら f の値にマージン
- AGH でも $\mathcal{M}$ に沿って勾配を小さくしてたので……

AGH でマージンを取る

$$\min \frac{1}{2}\int_{\mathcal{M}} \|\nabla y\|^2 \, dp + C\int_{\mathcal{M}} l(|y|)dp$$
$$\text{s.t. } E[y] = 0.$$

l: ロス関数

Hinge: $l(|y|) = [1 - |y|]_+$

Laplace: $l(|y|) = |1 - |y||$

$\mathcal{L}$ の 2, ..., r+1 番目の固有関数を u_1, ..., u_r とおいて,

$y = \sum_{j=1}^{r} \frac{w_j}{\sqrt{\sigma_j}} u_j$ の形の y だけ考えると, 目的関数は

$$\|\tilde{w}\|^2 + C\int_{\mathcal{M}} l(|y|)dp.$$

σ_j: y の固有値

n を有限として近似すると

※このスライドは
1 bit 分の場合の話

$$\min \ \|\tilde{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} l(|y_i|)$$
$$\text{s.t. } y_i = z_i^\top W \Sigma^{-1/2} \tilde{w}.$$

ほぼMMC

$\Sigma = \text{diag}(\sigma)$

Margin AGH

- r 個の異なるビットをどうやって作る？
  - AGH の解を初期値にして局所解を求める
- 計算が軽い方が嬉しい → **Iterative SVM/SVR + PA**

Iterative SVM/SVR [Zhang+, 2007]	Online Passive Aggressive [Crammer+, 2006]
・現在の y の符号を正解として SVM/SVR を解くことを繰り返す	・Hinge/Laplace-loss で重みの変化を小さく保ってオンライン学習

- 第 r ビットは, 第 r 標準基底 $\boldsymbol{e}_r$ を初期値に, AGH の解 Y_{AGH} の各行を入力として, 予測を正解と思って PA
- 結果を並べた行列を $\tilde{W}$ とおけば, Margin AGH の解は

$$Y = ZW\Sigma^{-1/2}\tilde{W},\ \mathbf{y}(\boldsymbol{x}) = (W\Sigma^{-1/2}\tilde{W})^\top \boldsymbol{z}(\boldsymbol{x})$$

実験設定

- データセット

	データ数	次元	クラス数	種類
news20	15,935	62,061	20	文書
rcv1-multiclass*	518,571	47,236	53	文書
MNIST	60,000	784	10	画像

*元のデータセットの train と test を入れ替えて実験

- 各手法・設定で, クエリーに対する k-NN でのラベル精度や, MAP (Mean Average Precision) を比較
- b-Bit MH は集合サイズ情報を使わない heuristic 版を使用 (集合サイズ情報も使うとより精度が上がるはずです)

http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#news20
http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#rcv1.multiclass
http://yann.lecun.com/exdb/mnist/

ランキング精度 (MAP)

	bit 数	AGH	MAGH (Hinge)	MAGH (Laplace)
news20	32	0.2580	0.2485	**0.2784**
	64	0.2147	0.2042	**0.2400**
rcv1	32	0.3421	0.4009	**0.4068**
	64	0.3095	0.3549	**0.4729**
MNIST	32	0.4487	0.6100	**0.7127**
	64	0.3612	0.4573	**0.6739**

- news20, rcv1 : Hellinger 距離, $t=2$, $s=5$, $m=500$
- MNIST : Euclid 距離, $t=10$, $s=2$, $m=300$
- MAP 値は Precision/Recall 曲線を線形補間で近似して計算
- rcv1 では 10,000 点ランダムサンプルして K-means を実行

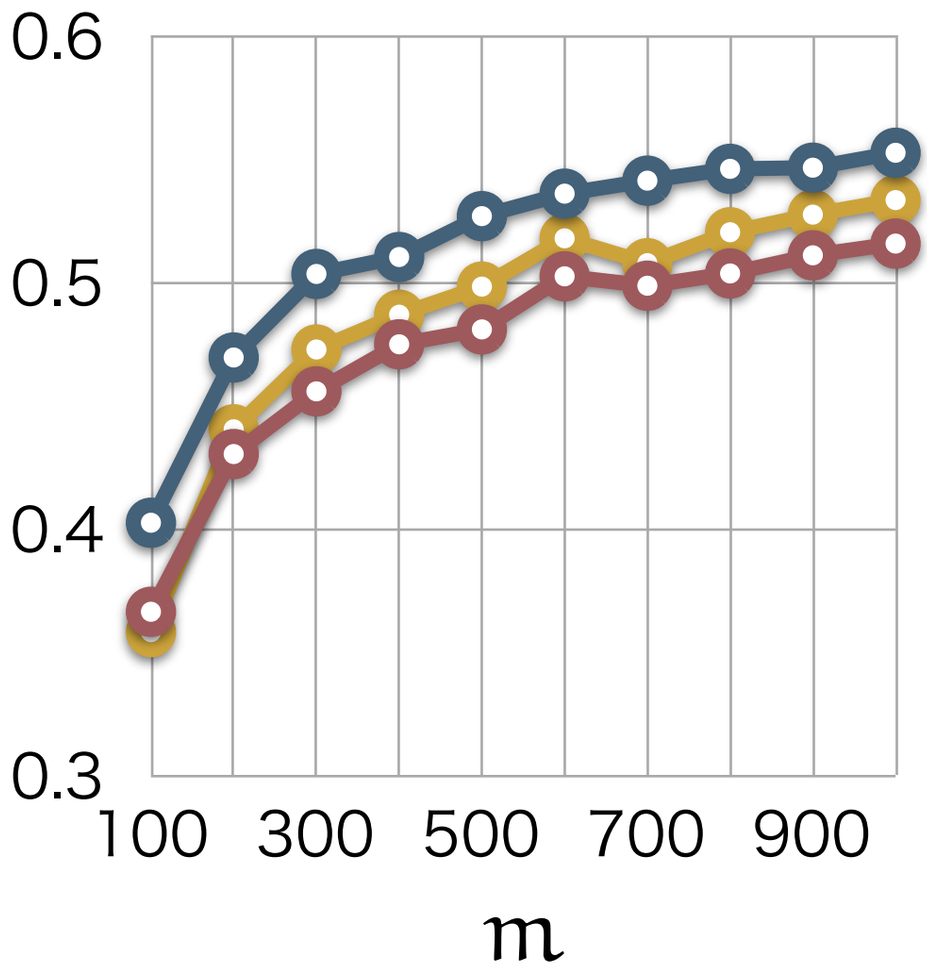
k-NN のラベル精度

	bit 数	news20		rcv1		MNIST
		k=1	k=10	k=1	k=10	k=10
AGH	32	**0.5545**	**0.5272**	**0.7062**	**0.6976**	**0.8785**
MAGH(H)		0.5032	0.4812	0.6707	0.6655	0.8712
MAGH(L)		0.5190	0.4986	0.6744	0.6687	0.8470
AGH	64	**0.5783**	**0.5400**	**0.7712**	**0.7530**	**0.8881**
MAGH(H)		0.5418	0.5036	0.7571	0.7405	0.8872
MAGH(L)		0.5565	0.5195	0.7551	0.7387	0.8431
b-Bit MH		0.2504	0.1097	0.4763	0.4328	-
	256	0.5147	0.2168	0.7191	0.6659	-
	512	0.6279	0.3053	0.7996	0.7509	-

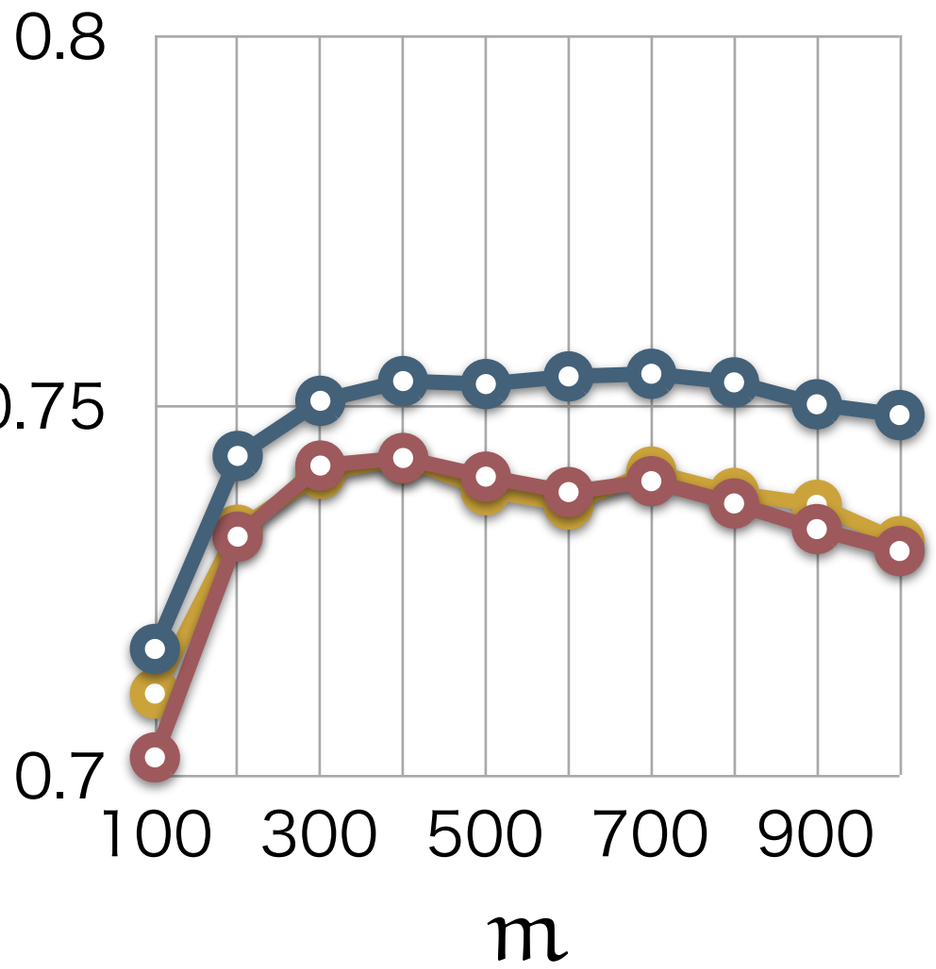
- AGH, MAGH の距離やパラメータは MAP の表と同じ
- 今回の実験では, b-Bit MH で集合サイズ情報を使っていない
  (論文中で heuristic explanation としているもの)

アンカー数 m と 10-NN 精度

- AGH
- MAGH(Hinge)
- MAGH(Laplace)



news20 (32 ビット)

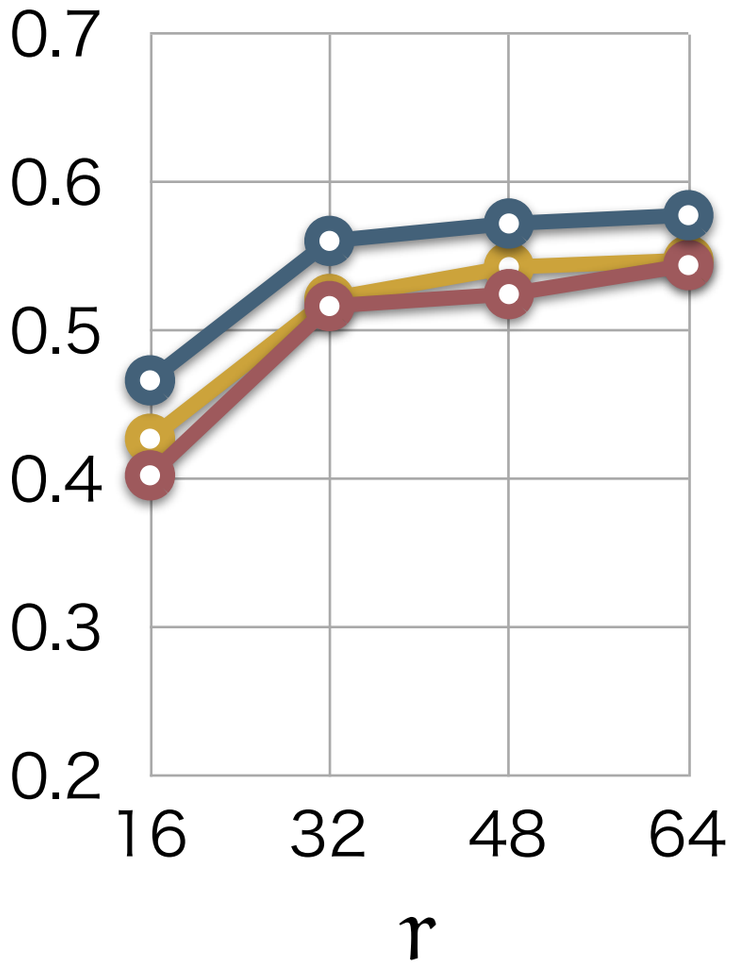


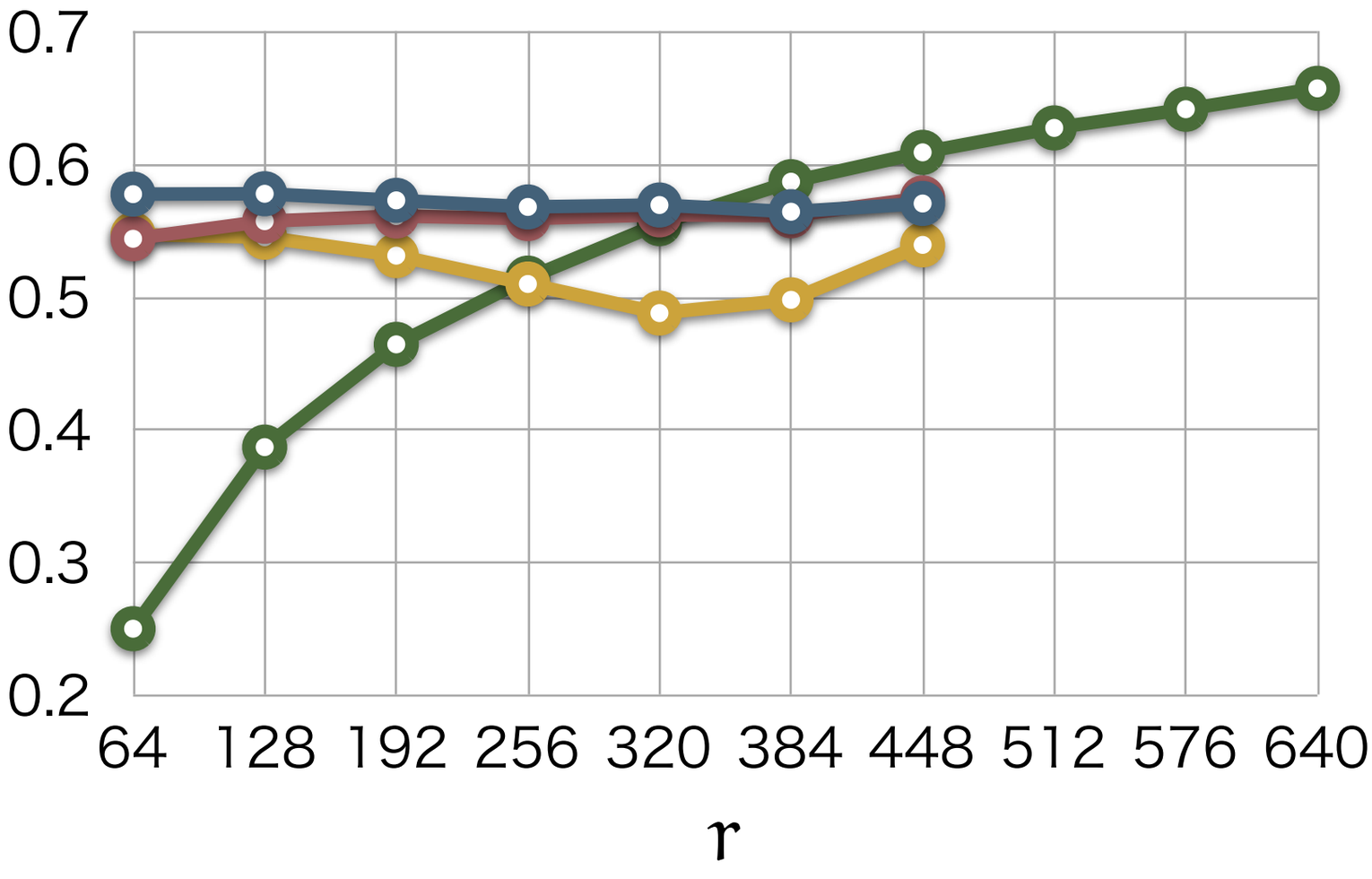
rcv1 (64 ビット)

(その他のパラメータは MAP の表と同じ)

ビット数での 1-NN 精度比較

- AGH
- MAGH(Hinge)
- MAGH(Laplace)
- b-Bit MH





news20. パラメータ設定は MAP の表のものと同じ.

考察

b-Bit Minwise Hashing と AGH の違いは, **対象データの種類**, **ハッシュのサイズ**, **得意な探索範囲**

	AGH	b-Bit MinHash
対象データ	距離関数があって, アンカーが求まるデータ	集合形式のデータ (e.g. Bag of Words)
ハッシュのサイズ	数十ビット (あまり増やしても精度上がらない)	精度を上げたい場合, かなり増やす必要がある
得意な探索範囲	広範囲	狭い範囲

参考文献

Y. Bengio, O. Delalleau, N. Le Roux, and J.-F. Paiement. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation*, 2004.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwarts, and Y. Singer. Online Passive-Aggressive Algorithms. *JMLR 7, 2006*.

P. Li, and A. C. König. b-Bit Minwise Hashing. *Proc. of WWW, 2010*.

W. Liu, J. He, and S.-F. Chang. Large graph construction for scalable semi-supervised learning. *Proc. of ICML*, 2010.

W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. *Proc of ICML*, 2011.

Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. *NIPS 21*, 2009.

L. Xu, J. Neufeld, B.Larson, and D.Schuurmans. Maximum Margin Clustering. *Proc. of NIPS, 2004*.

K. Zhang, I. W. Tsang, and J. T. Kwok. Maximum Margin Clustering Made Practical. *Proc. of ICML, 2007*.