セル分割によるAffinity Propagation の高速化

著者	松下 朋弘, 塩川 浩昭, 北川 博之
巻	DEIM2018
ページ	F4-5-F4-5
発行年	2018-04-02
URL	http://hdl.handle.net/2241/00157346

セル分割による Affinity Propagation の高速化

松下　朋弘† 　塩川　浩昭†† 　北川　博之††

† 筑波大学システム情報工学研究科コンピュータサイエンス専攻　〒 305–8577 茨城県つくば市天王台 1 丁目 1–1
†† 筑波大学計算科学研究センター　〒 305–8577 茨城県つくば市天王台 1 丁目 1–1
E-mail: †matsushita@kde.cs.tsukuba.ac.jp, ††{shiokawa,kitagawa}@cs.tsukuba.ac.jp

あらまし　Affinity Propagation は，各データポイント間でメッセージと呼ばれる実数値を収束するまで再帰的に計算することで他のクラスタリングアルゴリズムよりも精度の高いクラスタを構築するアルゴリズムである．しかし従来の手法の計算量はデータ数の 2 乗に比例するため，データ数が増えると実行時間が急激に増えるという欠点がある．本稿では，大規模データを対象とした Affintiy Propagation の高速化手法を提案する．提案手法はデータ空間を任意のサイズのセルに分割し，反復計算が不要なエッジをまとめて枝刈りする．評価実験により，提案手法がこれまでの既存手法と比べて高速に処理を行えることを示す．

キーワード　クラスタリング, Affinity Propagation, 高速化

1. はじめに

ソーシャルメディアなどの普及により生成されるデータ量が急激に増えている．それに伴い，データを効果的に処理するデータ分析技術の需要が高まっている．クラスタリングは与えられたデータをいくつかの部分集合に分割するデータ分析手法であり，データ分析の際に最もよく用いられる手法の一つである．

これまで多くのクラスタリングアルゴリズムが提案されてきた中で，2007 年に Frey らによって Affinity Propagation [1] と呼ばれる手法が提案された．この手法は，k-means 法 [2] 等のようなこれまで提案されてきた手法と比較してクラスタリング精度が良いことや初期値依存がないことなどから，画像データのクラスタリング [3] [4] やコミュニティー抽出 [5] といった多くの場面で用いられ，近年幅広い分野から着目されているアルゴリズムである．Affinity Propagation は，exemplar と呼ばれるクラスタの核となるデータポイントをすべてのデータポイントの中から自動的に検出することでクラスタを構築する手法である．$\mathbb{X} = \{x_1, x_2, \cdots, x_N\}$ を与えられたデータポイント，$s(x_i, x_j)$ をデータポイント x_i, x_j 間の類似度，$e(x_i)$ をデータポイント x_i の exemplar とした時，Affinity Propagation では，$\sum_{i=1}^{N} s(x_i, e(x_i))$ を最大化するように exemplar を決定する．しかしながら，exemplar の数は未定であることから，$\sum_{i=1}^{N} s(x_i, e(x_i))$ を最大化する exemplar の選択は NP-hard となる．ゆえに Affinity Propagation では，各データポイント間でメッセージと呼ばれる実数値の値を収束するまで再帰的に送り合うことで $\sum_{i=1}^{N} s(x_i, e(x_i))$ を準最大化する exemplar を自動的に決定する．ところがこの計算は，データ数を N，反復回数を T とした時，計算にかかる計算量は $O(N^2T)$ となり，依然としてデータ数が多くなる場合において膨大な処理時間を必要とするという課題が指摘されている [6].

この課題を解決するために Fujiwara らは，Graph-AP [7] を提案した．この手法は収束後のメッセージの上限値と下限値を推定する事でメッセージの伝搬が不要なエッジを枝刈りし，反復計算後に枝刈りしたエッジのメッセージを計算するアルゴリズムである．しかしこの手法は反復計算中にメッセージが収束したとしても全てのメッセージが収束するまで計算を続けなければならないため無駄な計算が行われている．この問題を解決したアルゴリズムが F-AP [8] である．この手法は収束前のメッセージの上限値と下限値を推定する事でメッセージの伝搬が不要なエッジを枝刈りするだけでなく，反復計算中に収束したと判定されたメッセージに対しては以降の反復計算の対象から取り除く事でさらなる高速化を実現した．しかしこの手法はエッジの枝刈りの際に全てのデータポイントペアを 1 つずつ探索しているため依然として無駄な計算が行われている．

そこで本研究では，大規模なデータを対象とした Affinity Propagation の高速化手法を提案する．提案手法はデータ空間を任意のサイズのセルに分割する事で，反復計算が不要なデータポイントペアを特定し，計算対象から逐次的に枝刈りする．これにより，再帰的なメッセージ伝搬を必要とするデータポイントペアを削減し高速化を図る．

本稿では実データセットと人工データセットを用いて提案手法の有効性の検証を行い，state-of-the-art とされる手法よりも高速に処理できることを示した．本研究の貢献を以下に示す．

- **高速性**：提案手法は従来の Affinity Propagation [1] と比べて 4 倍以上である．また，本稿では 4 節で述べる評価実験を通して，最新の手法よりも提案手法が高速であることを示した．
- **正確性**：提案手法は高速化のためにセルに基づく計算の枝刈りを行うが，従来の Affinity Propagaton と同一のクラスタリング結果を取得する事ができる．本稿では 3 節において提案手法の正確性を理論的に証明するとともに，4 節において実データを用いた評価実験を通して提案手法の正確性を実験的に確認した．

本稿の構成は，まず 2 節において Affinity Propagation の概要を説明し，3 節で提案手法について述べる．4 節で評価実験を示し，5 節で関連研究を述べる．最後に 6 節で結論を述べる．

2. 前提知識：Affinity Propagation

本節では Frey らによって提案された Affinity Propagation [1] について説明する．Affinity Propagation は，与えられたデータから各データポイント間の類似度 $\mathbb{S} = \{s(x_i, x_j) | x_i, x_j \in \mathbb{X}\}$ を計算する．$x_i = x_j$ の場合，$s(x_i, x_i)$ は preference と呼ばれ，一般的に preference は類似度の中央値を設定する．Affinity Propagation では負のユークリッド距離や jaccard 係数など任意の類似度に対応し，ユーザは k-means 法 [2] 等のようにクラスタ数を予め指定する必要がない．

次に Affinity Propagation は，$\sum_{i=1}^{N} s(x_i, e(x_i))$ を最大化する exemplar を決定するために，各データポイント x_i から $\mathbb{X}$ に含まれる全てのデータポイントに対して responsibility というメッセージを送信し，availability というメッセージを受信する．responsibility は $r(x_i, x_j)$ と表記されたとき，データポイント x_i から $x_j \in \mathbb{X}$ に対して送信する実数値であり，x_i が自身の exemplar として x_j を選択した場合にどれだけ $\sum_{i=1}^{N} s(x_i, e(x_i))$ の最大化に貢献するかという情報を表している．これに対して，availability は $a(x_i, x_j)$ と表記されたとき，データポイント x_i が $\mathbb{X}$ に含まれる全てのデータポイントから受信する実数値であり，x_j が x_i の exemplar として選択された場合にどれだけ $\sum_{i=1}^{N} s(x_i, e(x_i))$ の最大化に貢献するかという情報を表している．responsibility と availability はそれぞれ以下の式で定義されている．

$$r(x_i, x_j) = (1-\lambda)\rho(x_i, x_j) + \lambda r(x_i, x_j) \tag{1}$$

$$a(x_i, x_j) = (1-\lambda)\alpha(x_i, x_j) + \lambda a(x_i, x_j) \tag{2}$$

ここで，λ はダンピングファクタであり，0 から 1 までの値をとる．実際には $\lambda = 0.5$ と設定するのが一般的である．上記の式の中で，$\rho(x_i, x_j)$ は propagating responsibility，$\alpha(x_i, x_j)$ は propagating availability と呼ばれ，以下の式により計算する．

$$\rho(x_i, x_j) = \begin{cases} s(x_i, x_j) - \max\limits_{x_k \neq x_j} \{a(x_i, x_k) + s(x_i, x_k)\} & (x_i \neq x_j) \\ s(x_i, x_j) - \max\limits_{x_k \neq x_j} \{s(x_i, x_k)\} & (x_i = x_j) \end{cases} \tag{3}$$

$$\alpha(x_i, x_j) = \begin{cases} \min\{0, r(x_j, x_j) + \sum\limits_{x_k \neq x_i, x_j} \max\{0, r(x_k, x_j)\}\} & (x_i \neq x_j) \\ \sum\limits_{x_k \neq x_i} \max\{0, r(x_k, x_j)\} & (x_i = x_j) \end{cases} \tag{4}$$

反復計算を行う前に responsibility と availability の初期値 $r_0(x_i, x_j)$，$a_0(x_i, x_j)$ はそれぞれ以下のように設定し，全てのデータポイントペアに対して式 (1)・式 (2) が収束するまで再帰的に responsibility と availability を計算する．

$$r_0(x_i, x_j) = s(x_i, x_j) - \max_{x_k \neq x_j} \{s(x_i, x_k)\} \tag{5}$$

$$a_0(x_i, x_j) = 0 \tag{6}$$

すべてのデータポイント間における responsibility と availability が収束した後，データポイント x_i の exemplar は以下の式で決定する．

$$e(x_i) = \arg\max_{x_j} \{r(x_i, x_j) + a(x_i, x_j)\} \tag{7}$$

exemplar が決定した後，exemplar として選択されなかったデータポイントはすべての exemplar との間の類似度を比較し，類似度が最大となる exemplar と同じクラスタに割り当てられる．

Affinity Propagation では，各データポイント間でメッセージの値を反復計算する必要があるため，データ数を N，反復回数を T とすると，計算にかかる計算量は $O(N^2T)$ である．つまり，実行時間はデータ数の 2 乗に比例する．ゆえに，サイズの大きなデータを扱う際に膨大な処理時間を必要とする．

3. 提 案 手 法

3.1 手法の概要

従来のアルゴリズムでは，すべてのデータポイント間で反復計算を行なっている．そのため，扱うデータ数が膨大になると実行にかかる計算量も膨大となる．そこで提案手法では，距離の公理が成り立つ負のユークリッド距離の適用を前提とし，メッセージの伝搬が不要なデータポイントペアを枝刈りすることで計算量の削減を図る．具体的には，まず提案手法ではデータ空間上にセルを展開し，(１）セル間の距離に基づいたメッセージ送信の枝刈り及び（２）1 つセルに含まれるデータポイントに対するメッセージの計算の集約を行う．さらに提案手法では，Fujiwara らが提案したアルゴリズムで用いられた responsibility と availability の上限値と下限値 [8] を適用することで反復計算を行うか否かについても枝刈りを行うことでさらなる高速化を図る．

3.2 セルの展開と格子点間の類似度の計算

複数のデータポイントをまとめて処理するために一辺のサイズが ϵ のセルをデータ空間上に展開し，データポイントが含まれるセルを 1 つのデータポイントとみなしてセル間でメッセージの伝搬が行われるかどうかを判定する．

提案手法では，セル単位でメッセージの伝搬が行われるかを判定するため，セルの各格子点を擬似的なデータポイントとみなし，格子点間でメッセージの伝搬が行われるかを判定する．そこで，各格子点の座標から格子点間の類似度をデータ入力時と同様に計算する．ここで，格子点の preference は実データポイント間の類似度の入力の際に求めた preference の値をそのまま用いる．

3.3 メッセージの上限値と下限値の推定

Fujiwara らが提案した responsibility と availability の上限値と下限値 [8] を以下に示す．

定義 3.1 (responsibility の上限値と下限値) responsibiltiy の上限値 $\overline{r}(x_i, x_j)$ と下限値 $\underline{r}(x_i, x_j)$ は以下の式で計算する．

$\overline{r}(x_i, x_j) =$

$$\begin{cases} \lambda r_0(x_i,x_j)+\overline{\rho}(x_i,x_j) & (r_0(x_i,x_j) > 0, \overline{\rho}(x_i,x_j) > 0) \\ \lambda r_0(x_i,x_j)+(1-\lambda)\overline{\rho}(x_i,x_j) & (r_0(x_i,x_j) > 0, \overline{\rho}(x_i,x_j) \leqq 0) \\ \overline{\rho}(x_i,x_j) & (r_0(x_i,x_j) \leqq 0, \overline{\rho}(x_i,x_j) > 0) \\ (1-\lambda)\overline{\rho}(x_i,x_j) & (r_0(x_i,x_j) \leqq 0, \overline{\rho}(x_i,x_j) \leqq 0) \end{cases} \tag{8}$$

$\underline{r}(x_i, x_j) =$

$$\begin{cases} (1-\lambda)\underline{\rho}(x_i,x_j) & (r_0(x_i,x_j) > 0, \underline{\rho}(x_i,x_j) > 0) \\ \underline{\rho}(x_i,x_j) & (r_0(x_i,x_j) > 0, \underline{\rho}(x_i,x_j) \leqq 0) \\ \lambda r_0(x_i,x_j)+(1-\lambda)\underline{\rho}(x_i,x_j) & (r_0(x_i,x_j) \leqq 0, \underline{\rho}(x_i,x_j) > 0) \\ \lambda r_0(x_i,x_j)+\underline{\rho}(x_i,x_j) & (r_0(x_i,x_j) \leqq 0, \underline{\rho}(x_i,x_j) \leqq 0) \end{cases} \tag{9}$$

ここで，$\overline{\rho}(x_i, x_j)$ と $\underline{\rho}(x_i, x_j)$ はそれぞれ propagating responsibility の上限値，下限値であり，それぞれ以下の式で求められる．

$\overline{\rho}(x_i, x_j) =$

$$\begin{cases} s(x_i,x_j) - s(x_i,x_i) & (x_i \neq x_j) \\ s(x_i,x_j) - \max\limits_{x_k \neq x_j} \{s(x_i,x_k)\} & (x_i = x_j) \end{cases} \tag{10}$$

$\underline{\rho}(x_i, x_j) =$

$$\begin{cases} s(x_i,x_j) - \max\limits_{x_k \neq x_j} \{\overline{a}(x_i,x_k) + s(x_i,x_k)\} & (x_i \neq x_j) \\ s(x_i,x_j) - \max\limits_{x_k \neq x_j} \{s(x_i,x_k)\} & (x_i = x_j) \end{cases} \tag{11}$$

式からもわかるように responsobility の上限値と下限値は類似度のみを用いて計算することができる．ゆえに，反復計算を行う前に推定を行うことができる．

定義 3.2 (availability の上限値と下限値) availability の上限値 $\overline{a}(x_i, x_j)$ と下限値 $\underline{a}(x_i, x_j)$ はそれぞれ以下の式で計算する．

$\overline{a}(x_i, x_j) =$

$$\begin{cases} \overline{\alpha}(x_i,x_j) & (\overline{\alpha}(x_i,x_j) > 0) \\ (1-\lambda)\overline{\alpha}(x_i,x_j) & (\overline{\alpha}(x_i,x_j) \leqq 0) \end{cases} \tag{12}$$

$\underline{a}(x_i, x_j) =$

$$\begin{cases} (1-\lambda)\underline{\alpha}(x_i,x_j) & (\underline{\alpha}(x_i,x_j) > 0) \\ \underline{\alpha}(x_i,x_j) & (\underline{\alpha}(x_i,x_j) \leqq 0) \end{cases} \tag{13}$$

ここで，$\overline{\alpha}(x_i, x_j)$ と $\underline{\alpha}(x_i, x_j)$ はそれぞれ propagating responsibility の上限値，下限値であり，それぞれ以下の式で求められる．

$\overline{\alpha}(x_i, x_j) =$

$$\begin{cases} \min\{0, \overline{r}(x_j,x_j) + \sum\limits_{x_k \neq x_i, x_j} \max\{0, \overline{r}(x_k,x_j)\}\} & (x_i \neq x_j) \\ \sum\limits_{x_k \neq x_i} \max\{0, \overline{r}(x_k,x_j)\} & (x_i = x_j) \end{cases} \tag{14}$$

$\underline{\alpha}(x_i, x_j) =$

$$\begin{cases} \min\{0, s(x_i,x_j) - \max\limits_{x_k \neq x_j} \{s(x_i,x_k)\}\} & (x_i \neq x_j) \\ 0 & (x_i = x_j) \end{cases} \tag{15}$$

availability の上限値と下限値についても responsibility の時と同様に類似度のみを用いて計算することができるため，反復計算を行う前に推定を行うことができる．

3.4 反復計算が不要なポイントペアの枝刈り

提案手法では，メッセージの上限値，下限値を用いて反復計算を行うポイントペアを枝刈りする．枝刈りされたポイントペアのメッセージは反復計算後に再帰的な計算なしで求めることができるため，従来の Affinity Propagation と同じ結果を出力することができる．本稿では例として図 1 のようなデータポイント x_i，x_j とそれぞれを含むセル c，c' について考える．

まずは，responsibility の反復計算が不要なポイントペアの枝刈りを行う条件を示す．

補題 3.1 (responsibility で枝刈りを行う条件) 以下の条件を満たす 2 つのセル c，c' に含まれるすべてのデータポイント同士の responsibility は反復計算を必要としない．

$$x_a \neq x_b \ \ and \ \ s(x_a,x_b) - s(x_a,x_a) \leqq 0 \tag{16}$$

ここで x_a，x_b は，図 2 のように，データポイント x_i，x_j を含むそれぞれのセル c，c' 間の最短距離をとる格子点のペアを指す．

証明 任意のデータポイントペア x_i，x_j に対して Fujiwara ら [8] は，$x_i = x_j$ *or* $\overline{r}(x_i, x_j) > 0$ を満たさない場合，responsibility の反復計算は不要であることを証明した．

式 (8)，(10) より，$x_i \neq x_j$ *and* $\overline{r}(x_i, x_j) \leqq 0$ となるとき，$x_i \neq x_j$ *and* $\overline{\rho}(x_i, x_j) \leqq 0$ となっていることは明らかである．ここで $x_i \neq x_j$ の時，距離の公理が成り立つと仮定していることから $s(x_a, x_b) \geqq s(x_i, x_j)$ であることを利用して，$\overline{\rho}(x_i, x_j) = s(x_i, x_j) - s(x_i, x_i) \leqq s(x_a, x_b) - s(x_i, x_i) = s(x_a, x_b) - s(x_a, x_a) = \overline{\rho}(x_a, x_b)$ となる．よって，$\overline{\rho}(x_a, x_b) \leqq 0$ を満たす時，$\overline{\rho}(x_i, x_j) \leqq 0$ となる． □

続いて，availability の反復計算が不要なポイントペアの枝刈りを行う条件を示す．

補題 3.2 (availability で枝刈りを行う条件) 以下の条件を満たす 2 つのセル c，c' に含まれるすべてのデータポイント同士の availability は反復計算を必要としない．

$$\min\{\overline{a}(x_a,x_b) + s(x_a,x_b)\} < \max\{\underline{a}(x_c,x_d) + s(x_c,x_d)\} \tag{17}$$

ここで x_a，x_b はそれぞれ $x_a \in c$，$x_b \in c'$ となるデータポイントである．また，x_c は $x_c \in c$ となるデータポイントであり，x_d は，あるデータポイント x_i について $\underline{a}(x_i, x'') + s(x_i, x'')$ を 2 番目に大きくするようなデータポイント x'' である．

証明 任意のデータポイントペア x_i，x_j に対して Fujiwara ら [8] は，以下の式を満たさない場合，availability の反復計算は不要であることを証明した．

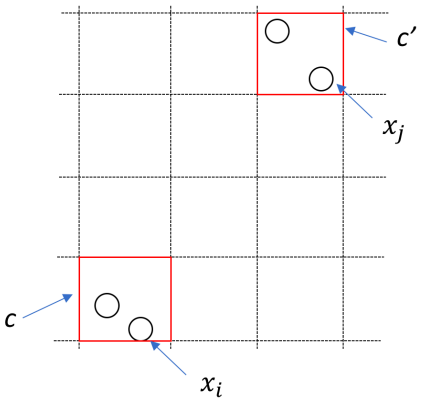

図 1　データポイントとセルの例

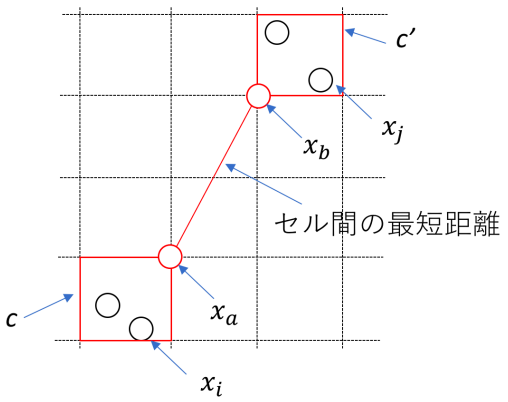

図 2　セル c とセル c' 間の最短距離

$$\overline{a}(x_i, x_j) + s(x_i, x_j) \geqq \underline{a}(x_i, x'') + s(x_i, x'') \tag{18}$$

セル c，c' 間について，式 (18) の左辺の最小値が右辺の最大値よりも小さい場合，セルに含まれるすべてのデータポイント同士は反復計算を必要としないことは明らかである．そこで，左辺の最小値をとるデータポイントペアと右辺の最大値をとるデータポイントペアを考えることで従来の結果を保証しつつセル内に含まれるデータポイントの枝刈りをまとめて行う事ができる．□

提案手法はセルを用いて反復計算が不要なデータポイントペアを特定することにより高速化を図るが，結果として次のような性質を持つ．

定理 3.1 (提案手法の正確性)　提案手法は従来の Affinity Propagation と同一の exemplar を出力する．

証明　式 (7) で示したように，従来の Affinity Propagation では exemplar を特定するために，全てのデータポイントに対して $r(x_i, x_j) + a(x_i, x_j)$ を最大化するデータポイント x_j を特定する必要がある．この時，提案手法は補題 3.1 および補題 3.2 に示したように，exemplar の選出に影響を与えないデータポイントペアに関する計算についてのみ枝刈りを行う．従って、従来の affinity Propagation と同様に $r(x_i, x_j) + a(x_i, x_j)$ を最大化するデータポイント x_j を特定することが可能である．□

3.5　アルゴリズム

提案手法のアルゴリズムを Algorithm 1 に示す．Algorithm 1 は，各データポイント間の類似度 $\mathbb{S}$，セルの一辺の長さ ϵ を受け取り，各データポイントの対する exemplar を出力する．そのために，まずセルの各格子点間の類似度を計算する（1,2 行目）．次に，各データポイント間でで responsibility の上限値 $\overline{r}(x_i, x_j)$ と下限値 $\underline{r}(x_i, x_j)$，availability の上限値 $\overline{a}(x_i, x_j)$ と下限値 $\underline{a}(x_i, x_j)$ を求める（3,4 行目）．補題 3.1 を満たすような 2 つのセルに含まれるすべてのデータポイント同士の responsibility は反復計算が不要である．また，補題 3.2 の条件を満たすような 2 つのセルに含まれるすべてのデータポイント同士の availability は反復計算が不要である（5-7 行目）．その後，上記で枝刈りされなかったポイントペアに対して繰り返し計算を行う（8-11 行目）．そして，既存手法である Graph-AP [7] や F-AP [8] で示されているように収束後のメッセージはそれぞれ $r(x_i, x_j) = \rho(x_i, x_j)$ と $a(x_i, x_j) = \alpha(x_i, x_j)$ であるという特性を用いて枝刈りしたポイントペアの responsibility と availability を計算する（12,13 行目）．これによってすべてのデータポイント間のメッセージが計算されるので最終的に，従来の Affinity Propagation と同じ exemplar を出力できる（14,15 行目）．

Algorithm 1 Proposed Algorithm

Require: $\mathbb{S}$, similarities between each data point pair; ϵ, size of the cell
Ensure: exemplar of each data point

1: **for** each point pair **do**
2:　compute similarities between each point
3: **for** each data point pair **do**
4:　compute $\overline{r}(x_i, x_j)$,$\underline{r}(x_i, x_j)$,$\overline{a}(x_i, x_j)$,$\underline{a}(x_i, x_j)$ from equation (8) - (15)
5: **for** each cell pair including data points **do**
6:　**if** satisfy lemma 3.1 or lemma 3.2 **then**
7:　perform pruning judgment by F-AP algorithm
8: **repeat**
9:　**for** each linked data point pair (x_i, x_j) **do**
10:　compute $r(x_i, x_j)$ and $a(x_i, x_j)$ from equation (1) and (2)
11: **until** all messages updated by iterative computation are converge
12: **for** each unlinked data point pair (x_i, x_j) **do**
13:　compute $r(x_i, x_j) = \rho(x_i, x_j)$ and $a(x_i, x_j) = \alpha(x_i, x_j)$
14: **for** each data point x_i **do**
15:　compute exemplar from equation (7)

4.　評 価 実 験

本節では，実データに対して提案手法，従来の Affinity Propagation [1]，Graph-AP [7], F-AP [8] を実行し，提案手法の高速性とクラスタリング精度を検証する．本節において，Original AP は従来の Affinity Propagation [1]，Proposed は提案手法を表している．本実験では以下のデータセットを使用した．

- **Perfume Data**：このデータセットは，UCI Machine Learning Repository(`https://archive.ics.uci.edu/ml/index.php`) [9] で公開されているものであり，20 種類の香水の匂いで構成されている．データは 28 秒の間，毎秒ハンド

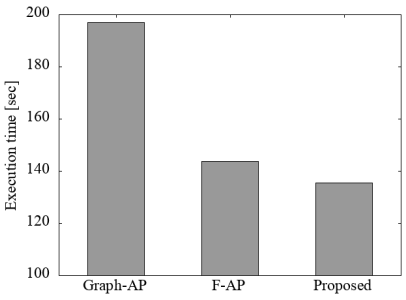

図 3　実行時間 (Perfume Data)

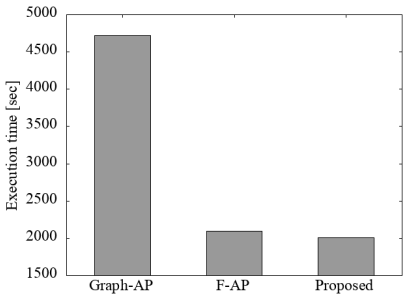

図 4　実行時間 (Geo Data)

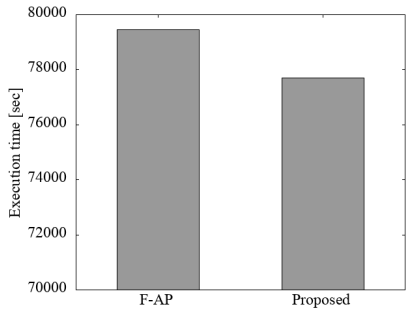

図 5　実行時間 (Synthetic Data)

ヘルド臭気計 (OMX-GR sensor) を用いて得られた．データ数は 560 で次元数は 2 であり，提案手法におけるパラメータ ϵ は 2,500 とした．

- **Geo Data**：このデータセットは，自治体オープンデータ CKAN(`https://ckan.open-governmentdata.org/dataset/401005_shinoshisetsu`) [10] で公開されている北九州市内の市役所，図書館，学校等の施設情報に関するオープンデータである．本実験ではデータの中で，緯度・軽度を用いてクラスタリングを行なった．データ数は 1,309 であり，提案手法におけるパラメータ ϵ は 0.05 とした．
- **Synthetic Data**：このデータセットは，Clustering benchmark datasets(`http://cs.uef.fi/sipu/datasets/`) [11] で公開されている人工データである．データ数は 3,000 で次元数は 2 であり，提案手法におけるパラメータ ϵ は 2,500 とした．

ダンピングファクタ λ は全て 0.5 に設定し，メッセージの反復回数は 1,000 回に固定した．また，各データ間の類似度は全て負のユークリッド距離を用いて計算した．全てのプログラムは C++で実装し，実験は CPU が 3.3GHz Intel Core i7 でメモリが 16GB の Mac OS で行なった．

4.1 高　速　性

3 種類のデータセットに対して，提案手法，Graph-AP，F-AP を実行した時の実行時間を図 3–5 に示す．図中で Original AP の実験結果は示していないが，実行時間は Perfume Data では 441.039[sec]，Geo Data では 9564.676[sec] であった．また，Synthetic Data を用いた実験では，Original AP と Graph-AP は 24 時間で実行が終了しなかったため途中で実験を打ち切った．

実験結果より，提案手法，Graph-AP，F-AP は従来の Affinity Propagation と比較して最大で Graph-AP は 2.2 倍程度，F-AP と提案手法は 4.6 倍程度高速に処理できている事がわかった．これは，従来の Affinity Propagation では全てのデータポイントペアでメッセージの伝播を行うのに対して，提案手法はメッセージの伝搬が不要なエッジを枝刈りしたため処理時間の短縮につながっている．また，提案手法は先行研究である Graph-AP および F-AP と比較して処理時間をそれぞれ最大で 58%，6%短縮できている事がわかる．Graph-AP はメッセージ伝搬の過程において計算不要なデータポイントペアを逐次枝刈りしていく手法である．Graph-AP はメッセージの伝搬の際に枝刈りされなかった全てのエッジで対して 1,000 回の反復計算を実行する必要がある．これに対して，提案手法と F-AP は，計算中に収束したと判定されたデータポイントペアは以降の反復計算を行う対象から取り除いている．加えて，F-AP では全てのデータポイントペアを 1 つずつ探索することで計算不要なデータポイントペアを検出するのに対して，提案手法では，セル分割を活用することで複数のデータポイントをまとめて処理する．図 6 は Synthetic Data に対して F-AP と提案手法を実行した際に計算対象となるデータポイントペア数の比較を示している．図 6 より，提案手法は F-AP と比較して最大で 52%に削減できていることがわかる．

4.2 スケーラビリティ

Synthetic Data において，データサイズを変化させた場合に F-AP と提案手法で実行時間がどのように比例するのかを実験する．本実験では，データ数を 500 から 3000 まで変化させた．実験結果を図 7 に示す．図 7 は各データサイズに対して提案手法の実行時間と F-AP の実行時間の差を表している．図 7 より，データサイズが大きくなればなるほど提案手法と F-AP の実行時間の差が大きくなっていることがわかる．これは，F-AP ではたとえ大規模なデータセットにおいても全てのデータポイント間で枝刈り判定を行なっているのに対して，提案手法ではセル同士で判定を行い，反復計算が不要なデータポイントペアをまとめて枝刈りしているためだと考えられる．図 8 は Synthetic Data においてデータ数を変化させて提案手法と F-AP を実行した際に計算対象となるデータポイントペア数の比較を表している．図 8 より，データサイズが大きくなるほど提案手法と F-AP の枝刈り判定を行う回数の差が開いていることがわかる．

4.3 クラスタリング精度

Perfume Data と Geo Data における提案手法，Graph-AP および F-AP を実行した時のクラスタリング精度を比較する．本実験では Original AP が出力する exemplar を正解とし，これに対する適合率と再現率を評価した．クラスタリング精度を評価するため，[7] と同じように適合率と再現率を用いた．適合率は Original AP で得られた exemplar の中で提案手法，Graph-AP，F-AP で得られた exemplar と一致したデータポイントの割合を表す．一方で再現率は，提案手法，Graph-AP，F-AP で得られた exemplar の中で Original AP で得られた exemplar と一致したデータポイントの割合を表す．実験結果を図 9，10 に表す．実験結果より，いずれのデータセットにおいても全ての手法について適合率と再現率は 1 となる事が確認できた．すなわち，提

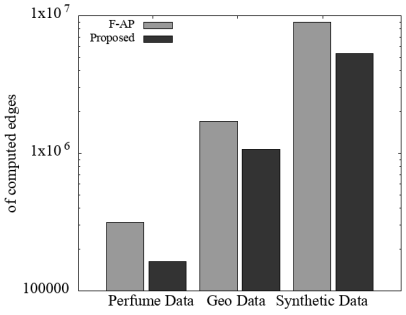

図 6　枝刈り判定を行うエッジ数の比較

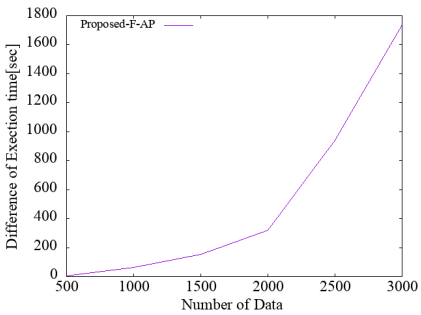

図 7　提案手法と F-AP の実行時間の差

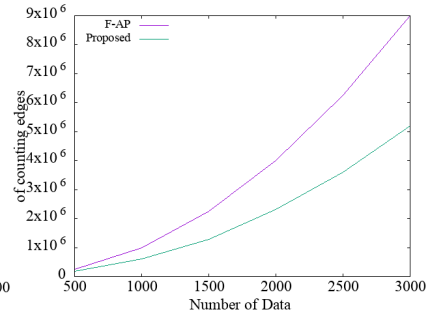

図 8　枝刈り判定を行うエッジ数の比較 (スケーラビリティ)

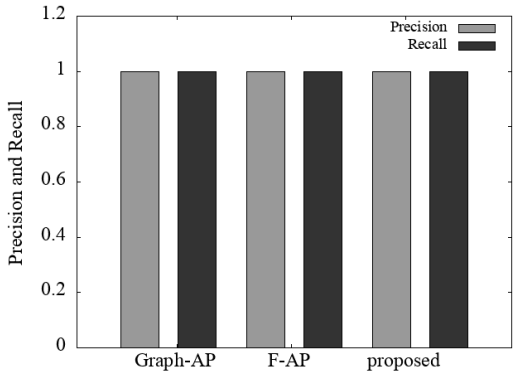

図 9　適合率と再現率 (Perfume Data)

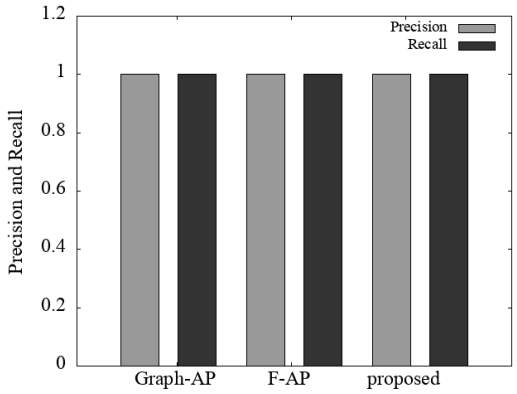

図 10　適合率と再現率 (Geo Data)

案手法および F-AP，Graph-AP は Original AP と同じ結果を出力している事を表している．定理 1 で示したように，提案手法は exemplar を特定する際に必要となる responsibility および availability を損なわず枝刈りを行う事が理論的に保障されている．その結果として，Original AP と比較して正確な exemplar を選択する事ができることを実験的に確認した．以上より，提案手法は従来の Affinity Propagation と同じ結果を保証しつつ，Graph-AP，F-AP より高速な処理が可能なアルゴリズムであることを示した．

5. 関 連 研 究

クラスタリングは与えられたデータの中にある隠れたパターンを見つける方法としてもっとも基本的な技術の一つであり，これまで多くのアルゴリズムが提案されてきた．

k-means 法 [2] はもっとも有名なクラスタリングアルゴリズムの一つである．この手法はクラスタ内の centroid と呼ばれるクラスタの重心を見つけることでユーザが指定した数のクラスタを形成する．また，k-means 法と同じような手法として k-medoids 法 [12] がある．k-medoids 法は k-means 法と同じようなアルゴリズムで medoid と呼ばれるクラスタを代表するデータポイントとそれ以外のクラスタ内のデータポイント間の距離の総和を最小にするように medoid を決定する．しかし，いずれのアルゴリズムもユーザが予めクラスタ数を指定する必要があることやクラスタリング結果が初期状態に依存してしまうという欠点がある．

Affinity Propagation [1] はユーザがクラスタ数を指定する必要がなく，同じデータセットでは常に一定の結果を出力するアルゴリズムとして近年着目され，これまで多くの応用的な手法が提案されている．Renchu らは Affinity Propagation を基にした半教師付きテキストクラスタリングアルゴリズムである SAP [13] を提案した．また，Zhang らは Affinity Propagation のアルゴリズムを基にしてユーザが指定した数の exemplar を特定するアルゴリズムである K-AP [14] を提案した．さらに Chang-Dong らは一つのクラスタ内に複数で exemplar を定義しすることでサブクラスタを形成できるアルゴリズムである MEAP [15] を提案した．しかし，Affinity Propagation はデータ数が大きくなるにつれて実行時間も急激に増えてしまうという欠点がある．そこで Affinity Propagation を高速化した手法がこれまでいくつか提案されてきた．

Jia らは K-最近傍を用いて疎なグラフを構築し，2 段階に分けて反復計算を行うことで Affinity Propagation を高速化する FSAP [6] を提案した．しかしこの手法はパラメータによって結果が異なるため，従来の Affinity Propagation と同一の結果を出力する事ができない可能性がある．

Fujiwara らは，反復計算が不要なエッジを枝刈りすることで Affinity Propagation の高速化を図る Graph-AP [7] を提案した．Graph-AP では，各データポイント間において収束後のメッセージの上限値と下限値を計算し，それらの値を用いた条件式を満たさないようなデータポイント間のメッセージは反復計算を行わないことで無駄な計算を削減できる．枝刈りされたエッジのメッセージの値は反復計算をせずに求めることができるため，従来の Affinity Propagation と同じ結果を出力することができる．しかしこの手法では，反復計算を行うポイントペアの制限の

際にすべてのポイントペアを 1 つずつ探索しているため，無駄な計算が行われている．

Fujiwara らは Graph-AP をベースにさらなる高速化を実現する F-AP [8] を提案した．F-AP では，各データポイント間において収束する前の各 iteration 中のメッセージの値の上限値と下限値を推定することで，exemplar の決定に必要なエッジを限定する．また，各 iteration 中に計算されたメッセージの値が収束しているかどうかの判定を行う．もし収束していると判定された場合は以降の反復計算を行う対象から取り除く．これらのアイデアにより，Graph-AP よりも高速に処理を行うことができる．しかしこの手法についても，依然として反復計算を行うポイントペアの制限の際にすべてのポイントペアを 1 つずつ探索しているため，無駄な計算を削りきれていない．

6. おわりに

本稿では，Affinity Propagation の高速化手法を提案した．提案手法はメッセージの上限値と下限値からメッセージを伝搬する必要のないポイントペアを枝刈りすることで計算量の削減を図る．また，枝刈りの際にデータ空間上に任意のサイズのセルを展開してセルの中に複数含まれるようなデータポイントに対してはまとめて枝刈りを行うことで枝刈りの過程においても処理時間の短縮を図る．評価実験では，提案手法が従来の Affinity Propagation と同じクラスタリング結果を出力しつつ高速な処理を行えることを確認した．今後の課題として，より大規模なデータによる実験や，距離の公理が成り立たない類似度への拡張，パラメータ ϵ の評価が挙げられる．

7. 謝　　辞

本研究は，JSPS 科研費 16H06650 ならびに JST ACT-I の助成を受けたものである．

文　　献

[1] B. J. Frey, D. Dueck, “Clustering by Passing Messages Between Data Points,” Science, Vol. 315, No. 5814, pp. 972–976, 2007.

[2] Jain, Anil K, “Data Clustering: 50 Years beyond K-means,” Pattern recognition letters, Vol. 31, No. 8, pp. 651–666, 2010.

[3] チョ成文, 小田昌宏, 北坂孝幸, 三澤一成, 藤原道隆, 森健策 and others, “3 次元 CT 像からの複数臓器抽出における Affinity Propagation を用いた臓器存在尤度アトラス構築に関する検討,” 電子情報通信学会技術研究報告. MI, 医用画像. Vol.111, No.331, pp.61–66, 2011.

[4] L. Gang, G. Lei, L. Tianming, “Grouping of Brain MR Images via Affinity Propagation,” in Proceedings of IEEE International Symposium on Circuits and Systems, pp. 2425–2428, 2009.

[5] 杉原貴彦, 村田剛志, “Affinity Propagation によるコミュニティ抽出,” 第 25 回人工知能学会全国大会論文集, 3F1-3, 2011.

[6] Y. Jia, J. Wang, C. Zhang, and X.-S. Hua, “Finding Image Exemplars Using Fast Sparse Affinity Propagation,” in Proceedings of the 16th ACM international conference on Multimedia, pp. 639–642, 2008.

[7] Y. Fujiwara, G. Irie, and T. Kitahara, “Fast Algorithm for Affinity Propagation,” in Proceedings of International Joint Conference on Artificial Intelligence, pp. 2238–2243, 2011.

[8] Y. Fujiwara, M. Nakatsuji, H. Shiokawa, Y. Ida, and M. Toyoda, “Adaptive Message Update for Fast Affinity Propagation,” in Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 309–318, 2015.

[9] K Bache, M Lichman, “UCI Machine Learning Repository,” 2013.

[10] 総務企画局, “北九州市　市の施設 (共通フォーマット),” https://ckan.open-governmentdata.org/dataset/401005_shinoshisetsu.

[11] Pasi Fränti et al, “Clustering Datasets,” 2015, http://cs.uef.fi/sipu/datasets/.

[12] K. Leonard, R. Peter J, “Partitioning around Medoids (program pam),” Finding Groups in Data: An Introduction to Cluster Analysis, pp. 68–125, 1990.

[13] G. Renchu, S. Xiaohu, M. Maurizio, Y. Chen, L. Yanchun, “Text Clustering with Seeds Affinity Propagation,” IEEE Transactions on Knowledge and Data Engineering, Vol.23, No.4, pp. 627–637, 2011.

[14] Z. Xiangliang, W. Wei, N. Kjetil, S. Michele, “K-AP: Generating Specified K Clusters by Efficient Affinity Propagation,” in Proceedings of IEEE International Conference on Data Mining, pp. 1187–1192, 2010.

[15] W. Chang-Dong, L. Jian-Huang, S. Ching Y and Z. Jun-Yong, “Multi-Exemplar Affinity Propagation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, Num. 9, pp. 2223–2237, 2013.