メッセージ集約に基づく 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　〒 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　Web データにおける情報抽出・要約にはクラスタリングが有効であることが知られている．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、クラスタを代表するデータポイントを決定し，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．提案手法は複雑な反復計算が不要なデータポイントペアを枝刈りし，それらについてはより簡潔な式でメッセージを求めることで高速化を図る．評価実験により，提案手法が従来の手法と同様のクラスタリング結果を出力しつつ、これまでの既存手法と比べて高速に処理できることを示す．

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

1 はじめに

近年では Web サービスの進歩に伴い，生成されるデータ量が急激に増えている．それに伴い，複雑で大規模なデータを効果的に処理し，サービス内の複雑な現象を把握する必要性が高まっている．クラスタリングは与えられたデータをクラスタと呼ばれるいくつかの部分集合に分割することでデータセット内の隠れたパターンを見つけ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] と呼ばれるクラスタリングアルゴリズムが提案された．この手法は，*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．Affinity Propagation では，データが与えられるとクラスタ数を自動で決定するため，k-means 法 [2] や k-medoid 法 [14] などのようなこれまで提案されてきた手法とは異なり，ユーザが予め出力するクラスタ数を指定する必要がない．この他にも，他のクラスタリングアルゴリズムと比較してクラスタリング精度が良いこと [1,6] や初期値依存がないことなどから，コミュニティ検索 [7,8] やテキストデータ内の主要語句の抽出 [9,10] など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．

$\mathbb{X} = \{x_1, x_2, \cdots, x_N\}$ 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 を選択することは NP-hard となる．この問題に対して 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，データ数を N，反復回数を T とした時，この計算にかかる計算量は $O(N^3T)$ となり，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．

この課題を解決するために Fujiwara らは，Graph-AP [4] と呼ばれる Affinity propagation の高速化手法を提案した．この手法は，反復計算を行う前に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することで反復計算を行うエッジの数を削減し，高速化を図った．枝刈りしたエッジのメッセージは反復計算後に再帰的な計算を行うことなく求めることができる．ゆえに，Graph-AP は従来の Affinity Propagation [1] と同じクラスタリング結果を出力することができる．文献 [4] では，Graph-AP が従来の Affinity Propagation [1] と同じクラスタリング結果を出力しつつ，既存手法 [1,3] よりも高速に処理できることを実験的に示した．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 [5] である．この手法は，反復計算を行う前に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．F-AP についても Graph-AP [4] と同様に従来の Affinity Propagation [1] と同じクラスタリング結果を出力できることが理論的に証明されている．しかし，枝刈りされなかったデータポイントペアのメッセージの中にも複雑な計算が不要なメッセージが含まれており，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．

提案手法の基本的なアイデアは，各 iteration 毎に複雑な計算を行わずに計算できるメッセージを特定することでさらなる高速化を図る．これらの枝刈りしたメッセージは各 iteration における再帰計算後にまとめて計算することができる．本研究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 [5] 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] や既存手法 [4,5] と比較して高速に処理できることを示す（4.2 章）．
- **スケーラビリティ**：提案手法はデータ数の観点から，従来の Affinity Propagation [1] と比較して優れたスケーラビリティを示す（4.3 章）．
- **正確性**：提案手法は高速化のためにメッセージ集約に基づく計算の簡略化を行うが，従来の Affinity Propagaton [1] と同一のクラスタリング結果を出力する事ができる．本稿では定理 31 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，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（4.4 章）．

本研究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，従来の Affinity Propagation [1] と比較して 42.53 倍，state-of-the-art とされる手法 [5] と比較して 5.59 倍の高速化を実現した．また，提案手法は従来の Affinity Propagation [1] と同じクラスタリング結果を出力できることを実験的にも確認した．

本稿の構成は以下の通りである．まず 2 節において Affinity Propagation の概要を説明する．続いて 3 節で提案手法について述べ，4 節で評価実験を示す．そして 5 節で関連研究を述べ，最後に 6 節で結論を述べる．

2 事前知識：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性を表す類似度 $\mathbb{S} = \{s(x_i, x_j) | x_i, x_j \in \mathbb{X}\}$ 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 は類似度の中央値もしくは最小値を設定する [1]．Affinity Propagation では負のユークリッド距離のような距離の公理が成り立つものから jaccard 係数のような距離の公理が成り立たないものまで任意の類似度に対応することができる．また，Affinity Propagation では与えられた $\mathbb{X}$，$\mathbb{S}$ からアルゴリズムが適切なクラスタ数を自動的に決定するため，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．

1 節で述べたように，$\sum_{i=1}^{N} s(x_i, e(x_i))$ を最大化することは NP-hard である．ゆえに Affinity Propagation は，各データポイント間でメッセージを再帰的に伝搬することで近似解を出力する．具体的には，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ことで $\sum_{i=1}^{N} s(x_i, e(x_i))$ を準最大化する exemplar を決定する．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．

定義 21 (responsibility と availability) responsibility $r(x_i, x_j)$ と availability $a(x_i, x_j)$ は以下のように定義される．

$$r(x_i, x_j) = (1-\lambda)\rho(x_i, x_j) + \lambda r(x_i, x_j)$$
$$a(x_i, x_j) = (1-\lambda)\alpha(x_i, x_j) + \lambda a(x_i, x_j)$$

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．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．

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

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

ダンピングファクタ λ は実際には $\lambda = 0.5$ と設定するのが一般的であると言われている [1,4,5]．上記の式から分かるように，responsibility $r(x_i, x_j)$ と availability $a(x_i, x_j)$ は互いに影響を与え合っている．つまり，$r(x_i, x_j)$ は，$a(x_i, x_j)$ を計算するために用いられ，逆も同様である．

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定義 21 が収束するまで再帰的に responsibility と availability を計算する．

定義 22 (各メッセージの初期値) responsibility の初期値 $r_0(x_i, x_j)$ と availability の初期値 $a_0(x_i, x_j)$ は以下のように定義される．

$$r_0(x_i, x_j) = s(x_i, x_j) - \max_{x_k \neq x_j} \{s(x_i, x_k)\}$$
$$a_0(x_i, x_j) = 0$$

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．

Algorithm 1 Affinity Propagation [1]

Input: $\mathbb{S}$
Output: exemplars for each data object in $\mathbb{X}$
1: **repeat**
2: **for each** $(x_i, x_j) \in \mathbb{X}^2$ **do**
3: compute $r(x_i, x_j)$ by Definition 21;
4: **for each** $(x_i, x_j) \in \mathbb{X}^2$ **do**
5: compute $a(x_i, x_j)$ by Definition 21;
6: **until** all $r(x_i, x_j)$ and $a(x_i, x_j)$ are not updated
7: **for each** $x_i \in \mathbb{X}$ **do**
8: get an exemplar $e(x_i)$ by Definition 23;

定義 23 (exemplar の決定) データポイント x_i の exemplar は以下の式によって決定する．

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

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 のアルゴリズムを Algorithm1 に示す．Affinity Propagation ではデータポイントセット $\mathbb{X}$ と各データポイント間の類似度 $\mathbb{S}$ を入力値とし，exemplar を出力する．Affinity Propagation は類似度 $\mathbb{S}$ を元に全てのデータポイント間で responsibility と availability を定義 21 に基づいて計算する．この再帰計算は，全てのメッセージが収束するもしくはユーザが指定する反復回数に達するまで繰り返し行う（1 行目–6 行目）．その後，定義 23 のように，計算した responsibility と availability を用いて exemplar を決定し，クラスタを形成する（7–8 行目）．

Affinity Propagation では，全て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．また，データ数を N とした時，定義 21 より，各データポイント間でメッセージを計算する際に $O(N)$ の計算量を必要とする．ゆえに，反復回数を T とした時，Algorithm1 にかかる計算量は $O(N^3T)$ である．つまり，実行時間はデータ数の 3 乗に比例する．ゆえに，大規模なデータセットについてクラスタリングを行う際には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

本節では，提案手法について説明する．

提案手法の基本的なアイデアは，反復計算を行うデータポイントの数を削減することである．1 節で述べたように既存手法 [4,5] では，メッセージ伝搬が明らかに不要なデータポイントペアの枝刈りを反復計算が行われる前に実行することで高速化を実現している．しかし，そこで枝刈りが行われなかったデータポイントペアの中でも，定義 21 を用いた複雑な計算が不要なペアが存在することが予備実験により明らかになった．そこで本手法では，それらの計算を集約することでさらに反復計算を行うデータポイントペアの数を削減し，高速化を図る．具体的には，以下のアプローチにより高速化を図る．

- **メッセージ集約が可能なデータポイントペアの特定:** 定義 21 より，responsibility を計算するためには availability と類似度の和を最大にするデータポイントを探索する必要がある．また，availability についても自身に伝搬されてくる正の値の responsibility の和を計算する必要がある．この計算を避けるために提案手法では，T を反復回数として，$t-1$ 回目（$1 < t < T$）までに計算された responsibility と availability を用いて t 回目の反復計算において定義 21 を用いた計算をする必要がないデータポイントペアを決定する．
- **メッセージの集約計算:** 定義 21 を用いた計算が必要なデータポイントペアについてのメッセージの計算後，上記で定義したデータポイントペアのメッセージの集約計算を行う．この計算は，上記で定義したデータポイントペアの集合の中で 1 つのペアについてメッセージを計算し，その値を用いたより簡単な式によって他のデータポイントペアのメッセージを計算する．3.2，3.3 節では，この計算方法によって各 iteration における従来の Affinity Propagation [1] と同様のメッセージ値が計算できることを理論的に証明する．また，そこで求めた responsibility と availability をもとに，次の反復計算において定義 21 を用いた計算をする必要がないデータポイントペアを定義する．

提案手法では，メッセージ集約に基づく反復計算の簡略化を行なったとしても，従来の Affinity Propagation [1] と同様のクラスタリング結果を出力することができる．3.3 節で提案手法が正確性を理論的に保証していることを示し．4.4 節で提案手法の正確性を実験的に示す．

3.2 メッセージ集約が可能なデータポイントペアの特定

提案手法では，T を反復回数として，$t-1$ 回目（$1 < t < T$）で計算された responsibility と availability を用いて，t 回目の反復計算において定義 21 を用いた計算をする必要がないデータポイントペアを決定する．本節では，提案手法における再帰計算が不要なデータポイントペアの検出方法について説明する．

はじめに，responsibility の枝刈りについて説明する．まず，t 回目の反復計算における responsibility の計算の際に定義 21 を用いた計算が不要なデータポイントペアの集合 $PR(x_i, t)$ を定義する．提案手法では，これらのデータポイントペアの responsibility の計算を集約する．

定義 31 (集約可能な responsibility の特定) 以下の条件を満たすデータポイントペア (x_i, x_j) は t 回目のデータポイントペアの集合 $PR(x_i, t)$ に含まれる．

$$PR(x_i, t) = \{(x_i, x_j) | x_i \neq x_j \text{ and } x_j \neq x_k \in sum(a, s)\}$$

ここで，$a_m(x_i, x_j)$ は m 回目の反復計算におけるにおける (x_i, x_j) の availability の値であるとすると，

$$sum(a, s) =$$
$$\{x_l | x_l = \max_{x_l}\{a_m(x_i, x_l) + s(x_i, x_l)\}\} \ (m = 0, 1, \cdots, t-1)$$

である．

データポイントペアの集合 $PR(x_i, t)$ は以下の特性を持つ．

補題 31　$PR(x_i,t)$ に含まれるデータポイントペアの responsibility は集約計算が可能である．

証明　ここでは，$PR(x_i,t)$ に含まれるデータポイントペアの responsibility は類似度を用いて計算を集約できることを証明する．本証明では，$PR(x_i,t)$ に含まれる 2 組のデータポイントペア (x_i,x_a)，(x_i,x_b) について考える．

はじめに，

$$r_{t-1}(x_i,x_a)-r_{t-1}(x_i,x_b)=\beta$$
$$\rho_{t-1}(x_i,x_a)-\rho_{t-1}(x_i,x_b)=\beta$$

と仮定する．ここで，$r_{t-1}(x_i,x_j)$ は $t-1$ 回目における responsibility，$\rho_{t-1}(x_i,x_j)$ は $t-1$ 回目における propagating responsibility を表す．すると，t 回目における responsibility $r(x_i,x_a)$ と $r(x_i,x_b)$ の差は，

$$r_t(x_i,x_a)-r_t(x_i,x_b)=\beta$$

となる．つまり，$t-1$ 回目における responsibility の差と t 回目における propagating responsibility の差が等しい場合は t 回目における responsibility も同じ差の値になる．

はじめに $t=1$ の場合について考える．$t=1$ の時，responsibility を計算する式は定義 21 より，

$$r_1(x_i,x_j)=(1-\lambda)\rho_1(x_i,x_j)+\lambda r_0(x_i,x_j)$$

である．ここで，$\rho_1(x_i,x_j)$ は 1 回目の反復計算において計算される propagating responsibility の値であり，$r_0(x_i,x_j)$ は x_i から x_j へ伝搬される responsibility の初期値である．$r_0(x_i,x_j)$ は $s(x_i,x_j)-\max_{x_k\neq x_j}\{s(x_i,x_k)\}$ で求められることから，$x_a,x_b\neq x_k$ の時，

$$r_0(x_i,x_a)-r_0(x_i,x_b)=s(x_i,x_a)-s(x_i,x_b)$$

となることがわかる．

また，$\rho_1(x_i,x_j)$ は定義 21 より，$s(x_i,x_j)-\max_{x_k\neq x_j}\{a(x_i,x_k)+s(x_i,x_k)\}$ で計算することができる．ゆえに，$x_a,x_b\neq x_k$ の時，

$$\rho_1(x_i,x_a)-\rho_1(x_i,x_b)=s(x_i,x_a)-s(x_i,x_b)$$

で計算することができる．

以上より，$t=1$ の時，

$$\begin{aligned}r_0(x_i,x_a)-r_0(x_i,x_b)&=\rho_1(x_i,x_a)-\rho_1(x_i,x_b)\\&=s(x_i,x_a)-s(x_i,x_b)\end{aligned}$$

が成り立つことより，

$$r_1(x_i,x_a)-r_1(x_i,x_b)=s(x_i,x_a)-s(x_i,x_b)$$

つまり，$r_1(x_i,x_a)$ がわかっている場合，$r_1(x_i,x_b)$ は $r_1(x_i,x_a)$ と類似度を用いて簡単に計算することができる．

次に $t\geqq 2$ の場合について考える．$t\geqq 2$ の時，$r_{t-1}(x_i,x_a)-r_{t-1}(x_i,x_b)=s(x_i,x_a)-s(x_i,x_b)$ が成り立っていることは明らかである．また，$\rho_t(x_i,x_j)$ は，$s(x_i,x_j)-\max_{x_k\neq x_j}\{a(x_i,x_k)+s(x_i,x_k)\}$ で計算することができることより，上記と同様に

$$\begin{aligned}r_{t-1}(x_i,x_a)-r_{t-1}(x_i,x_b)&=\rho_t(x_i,x_a)-\rho_t(x_i,x_b)\\&=s(x_i,x_a)-s(x_i,x_b)\end{aligned}$$

となり，

$$r_t(x_i,x_a)-r_t(x_i,x_b)=s(x_i,x_a)-s(x_i,x_b)$$

が成り立つ．

以上より，$t\geqq 1$ で $(x_i,x_a),(x_i,x_b)\in PR(x_i,t)$ であるデータポイントペアの responsibility は availability と類似度の和を最大にするデータポイントを探索することなく計算できる．□

また，$PR(x_i,t)$ の初期セット $PR(x_i,1)$ は以下のように定義される．

定義 32 ($PR(x_i,t)$ の初期セット)　以下の条件を満たすデータポイントペア (x_i,x_j) は 1 回目のデータポイントペアの集合 $PR(x_i,1)$ に含まれる．

$$PR(x_i,1)=\{(x_i,x_j)|x_i\neq x_j \text{ and } x_j\neq x_k\}$$

ここで x_k は，

$$x_k=\max_{x_k}\{s(x_i,x_k)\}$$

である．

データポイントペアの集合 $PR(x_i,1)$ は以下のような特性を持つ．

補題 32　$PR(x_i,1)$ に含まれるデータポイントペアの responsibility は 1 回目の反復計算において集約計算により求めることができる．

証明　補題 31 より明らかである．□

次に，提案手法における availability の枝刈りについて説明する．まず，t 回目の反復計算における availability の計算の際に定義 21 を用いた計算が不要なデータポイントペアの集合 $PA(x_j,t)$ を定義する．提案手法では，これらのデータポイントペアの availability の計算を集約する．

定義 33 (集約可能な availability の特定)　以下の条件を満たすデータポイントペア (x_i,x_j) はデータポイントペアの集合 $PA(x_j,t)$ に含まれる．

$$PA(x_j,t)=$$
$$\{(x_i,x_j)|x_i\neq x_j \text{ and } r_m(x_i,x_j)\leqq 0\}\ (m=0,1,\cdots,t-1)$$

ここで，$r_m(x_i,x_j)$ は m 回目の反復計算におけるにおける (x_i,x_j) の responsibility の値である．

データポイントペアの集合 $PA(x_j,t)$ は以下の特性を持つ．

補題 33　$PA(x_j,t)$ に含まれるデータポイントペアの availability は集約計算が可能である．

証明　ここでは，$PA(x_j,t)$ に含まれるデータポイントペアの availability は定義 21 による計算を必要としないことを証明する．定義 21 より，$x_i \neq x_j$ における propagationg availability $\alpha(x_i,x_j)$ は以下のように求められる．

$$\alpha(x_i,x_j)=\min\{0,r(x_j,x_j)+\sum_{x_k \neq x_i,x_j}\max\{0,r(x_k,x_j)\}\}$$

$t-1$ 回目において，2 つのデータポイントペア $(x_a,x_j),(x_b,x_j) \in PA(x_j,t)$ の responsibility はそれぞれ $r_{t-1}(x_a,x_j) \leqq 0$，$r_{t-1}(x_b,x_j) \leqq 0$ であるため，上記の式より，$r(x_a,x_j)$ と $r(x_b,x_j)$ はいずれも propagationg availability の値に影響を与えないことは明らかである．ゆえに，t 回目の反復計算における propagating availability $\alpha_t(x_a,x_j)$ と $\alpha_t(x_b,x_j)$ は以下のような関係となる．

$$\alpha_t(x_a,x_j)=\alpha_t(x_b,x_j)$$

また，$t-1$ 回目における $(x_a,x_j),(x_b,x_j)$ の availability $a_{t-1}(x_a,x_j)$，$a_{t-1}(x_b,x_j)$ が $a_{t-1}(x_a,x_j)=a_{t-1}(x_b,x_j)$ の時，t 回目の反復計算における availability $a_t(x_a,x_j)$ と $a_t(x_b,x_j)$ は

$$a_t(x_a,x_j)=a_t(x_b,x_j)$$

が成り立つことは明らかである．

以上より，$PA(x_j,t)$ に含まれるデータポイントペアの availability は，1 つのペアについてメッセージを計算し，他のデータポイントペアについては自身に伝搬されてくる正の値の responsibility の和を計算することなく求めることができる．□

また，$PA(x_j,t)$ の初期セット $PA(x_j,1)$ は以下のように定義される．

定義 34 ($PA(x_j,t)$ の初期セット)　以下の条件を満たすデータポイントペア (x_i,x_j) は 1 回目のデータポイントペアの集合 $PA(x_j,1)$ に含まれる．

$$PR(x_i,1)=\{(x_i,x_j)|x_i \neq x_j \text{ and } r_0(x_i,x_j) \leqq 0\}$$

データポイントペアの集合 $PA(x_j,1)$ は以下のような特性を持つ．

補題 34　$PA(x_j,1)$ に含まれるデータポイントペアの availability は 1 回目の反復計算において集約計算により求めることができる．

証明　補題 33 より明らかである．□

3.3 メッセージの集約計算

本節では，3.2 章において定義 21 を用いた計算が不要なデータポイントペアのメッセージの計算方法について説明する．

はじめに，提案手法における枝刈りされたデータポイントペアの responsibility の計算について説明する．データポイントペアの集合 $PR(x_i,t)$ に含まれるデータポイントペアの responsibility は以下の式によって計算できる．

定義 35 (responsibility の集約計算)　t 回目のデータポイントペア集合 $PR(x_i,t)$ に含まれるデータポイントペア (x_i,x_a) の responsibility $r_t(x_i,x_a)$ は以下の式で計算できる．

$$r_t(x_i,x_a)=r_t(x_i,x_b)+s(x_i,x_a)-s(x_i,x_b)$$

ここで，$r_t(x_i,x_a)$，$r_t(x_i,x_b)$ はそれぞれ t 回目の反復計算において $(x_i,x_a),(x_i,x_b) \in PR(x_i,t)$ であるデータポイントペア $(x_i,x_a),(x_i,x_b)$ の responsibility の値である．

定義 35 は以下の特性を持つ．

補題 35　定義 35 によって従来の Affinity Propagation と同様の responsibility の値を計算できる．

証明　補題 31 より明らかである．□

次に，提案手法における枝刈りされたデータポイントペアの availability の計算について説明する．データポイントペアの集合 $PA(x_j,t)$ に含まれるデータポイントペアの availability は以下の式によって計算できる．

定義 36 (availability の集約計算)　t 回目のデータポイントペア集合 $PA(x_j,t)$ に含まれるデータポイントペア (x_a,x_i) の availability $a_t(x_a,x_i)$ は以下の式で計算できる．

$$a_t(x_a,x_i)=a_t(x_b,x_i)$$

ここで，$a_t(x_i,x_a)$，$a_t(x_i,x_b)$ はそれぞれ t 回目の反復計算において $(x_i,x_a),(x_i,x_b) \in PA(x_j,t)$ であるデータポイントペア $(x_i,x_a),(x_i,x_b)$ の availability の値である．

定義 36 は以下の特性を持つ．

補題 36　定義 36 によって従来の Affinity Propagation と同様の availability の値を計算できる．

証明　補題 33 より明らかである．□

最後に，提案手法の正確性について述べる．補題 31，33，35，36 より，提案手法は以下の定理を持つ．

定理 31　提案手法は常に従来の Affinity Propagation [1] と同じ結果を出力する．

証明　定義 23 より，x_i の exemplar の決定には収束した responsibility と availability を用いて $r(x_i,x_j)+a(x_i,x_j)$ を最大化するデータポイント x_j を見つける必要がある．補題 31，33，35，36 より，提案手法による枝刈りは responsibility と availability の値は従来の Affinity Propagation [1] と同じ値を正確に出力することを理論的に証明した．ゆえに提案手法は，$r(x_i,x_j)+a(x_i,x_j)$ の値を正確に得ることができるので，従来の Affinity Propagation と同様の exemplar を出力することができる．□

2 節で述べたように，全ての exemplar が決定すると，クラスタリング結果は一様に形成される．ゆえに定理 31 より，提案手法は従来の Affinity Propagation [1] と同じクラスタリング結果を出力することができる．

Algorithm 2 Proposed Method

Input: $\mathbb{S}$
Output: exemplars for data object in $\mathbb{X}$

1: **for each** $x_i \in \mathbb{X}$ **do**
2: obtain $R = \{\mathbb{X} \times \mathbb{X}\} \backslash P_r$ based on F-AP [5];
3: obtain $A = \{\mathbb{X} \times \mathbb{X}\} \backslash P_a$ based on F-AP [5];
4: **for each** $(x_i, x_j) \in R$ **do**
5: obtain $R(x_i, 1) = \{(x_i, x_j) | (x_i, x_j) \notin PR(x_i, 1)\}$ based on Definition 32;
6: **for each** $(x_i, x_j) \in A$ **do**
7: obtain $A(x_j, 1) = \{(x_i, x_j) | (x_i, x_j) \notin PA(x_j, 1)\}$ based on Definition 34;
8: **repeat**
9: **for each** $x_i \in \mathbb{X}$ **do**
10: **for each** $(x_i, x_j) \in R(x_i, t)$ **do**
11: compute $r(x_i, x_j)$ by Definition 21;
12: **for each** $(x_i, x_j) \in PR(x_i, t)$ **do**
13: compute $r(x_i, x_j)$ based on Definition 35;
14: **if** $r(x_i, x_j)$ is not updated **then**
15: $R = R \backslash \{(x_i, x_j)\}$;
16: **for each** $x_j \in \mathbb{X}$ **do**
17: **for each** $(x_i, x_j) \in A(x_j, t)$ **do**
18: compute $a(x_i, x_j)$ by Definition 21;
19: **for each** $(x_i, x_j) \in PA(x_j, t)$ **do**
20: compute $a(x_i, x_j)$ based on Definition 36;
21: **if** $a(x_i, x_j)$ is not updated **then**
22: $A = A \backslash \{(x_i, x_j)\}$;
23: **for each** $x_i \in R$ **do**
24: obtain $R(x_i, t+1)$ and $PR(x_i, t+1)$ based on Definition 31;
25: **for each** $x_j \in A$ **do**
26: obtain $A(x_j, t+1)$ and $PA(x_j, t+1)$ based on Definition 33;
27: $t = t + 1$;
28: **until** $R, A = \emptyset$
29: **for each** $(x_i, x_j) \in P_r$ **do**
30: get $r(x_i, x_j) = \rho(x_i, x_j)$;
31: **for each** $(x_i, x_j) \in P_a$ **do**
32: get $a(x_i, x_j) = \alpha(x_i, x_j)$;
33: **for each** $x_i \in \mathbb{X}$ **do**
34: get an exemplar $e(x_i)$ by Definition 23;

3.4 アルゴリズム

提案手法のアルゴリズムを Algorithm2 に示す．提案手法では，各データポイント間の類似度 $\mathbb{S}$ を入力値とし，exemplar を出力する．Algorithm2 は大きく 3 つのパートに分かれており，それぞれ初期データポイントペアのセットの取得（1–7 行目），逐次的メッセージ集約を含むメッセージの伝搬（8–28 行目），そして exemplar の検出（29–34 行目）である．

はじめに提案手法では，1 回目の反復計算において定義 21 を用いた計算が不要なデータポイントペアを決定する（1–7 行目）．既存手法である F-AP [5] と同様に提案手法でも各データポイント間の responsibility と availability の上限値と下限値を計算し，反復計算が不要なデータポイントペア P_r，P_a を得ることで枝刈りを実行する（1–3 行目）．そして定義 32，34 に基づいて 1 回目の反復計算において定義 21 を用いた計算が不要なデータポイントペアを定義する（4–7 行目）．

次に，再帰計算を実行する（8–28 行目）．ここでは，定義 21 に基づく計算が必要と判定されたデータポイントペアセット $R(x_i, t)$，$A(x_j, t)$ に対して responsibility と availability を収束するまで再帰的に更新する（10–11，17–18 行目）．次に，データポイントペアセット $PR(x_i, t)$，$PA(x_j, t)$ に対して定義 35，36 を用いて responsibility と availability を計算する（12–13 行目，19–20 行目）．その後，既存手法 [5] と同じように，メッセージが収束したと判定されたデータポイントペアについてはその後の再帰計算を行う対象から取り除く（14–15 行目，21–22 行目）．その後，次の反復計算において定義 21 を用いた計算が不要なデータポイントペアを定義 31，33 に基づいて決定する（23–26 行目）．これにより高速化を実現する．

最後に exemplar を検出する（29–34 行目）．はじめに提案手法では，枝刈りされたデータポイントペア P_r，P_a に対して，既存手法 [4,5] と同様の手法を用いることで反復計算を実行することなく収束後の responsibility と availability の値を計算することができる（29–32 行目）．その後，得られた responsibility と availability を用いて exemplar を決定する（33–34 行目）．

4 評価実験

本節では，実データを用いて提案手法の有用性を実験的に評価する．ここでは以下を示す．

- **高速性**：提案手法は従来の Affinity Propagation [1] や既存手法 [4,5] と比較して高速に処理できることを示す（4.2 節）．
- **スケーラビリティ**：提案手法はデータ数の観点から，従来の Affinity Propagation [1] と比較して優れたスケーラビリティを示す（4.3 章）．
- **正確性**：提案手法は高速化のためにメッセージ集約に基づく計算の簡略化を行うが，従来の Affinity Propagaton [1] と同一のクラスタリング結果を出力する事ができる（4.4 節）．

4.1 実験環境

本実験では，提案手法（ここでは Proposed と記述），従来の Affinity Propagation [1]（ここでは Original-AP と記述），Graph-AP [4]，F-AP [5] を比較した．全ての実験は CPU が Intel(R) Xeon(R) E5-1620 3.5GHz，128GB RAM のサーバ上で実行した．ダンピングファクタ λ は既存手法 [1,4,5] で推奨されている 0.5 に設定し，反復回数は 1,000 回とした．データセットは以下の 3 つのデータセットを用いた．

- **Perfume**：このデータセットは，UCI Machine Learning Repository(`https://archive.ics.uci.edu/ml/index.php`) [11] 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 である．
- **Geo**：このデータセットは，人口統計や公共施設の場所などの自治体オープンデータを公開している CKAN(`https://ckan.open-governmentdata.org/dataset/401005_shinoshisetsu`) [12] にあ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・経度を用いてクラスタリングを行なった．データ数は 1,309 である．
- **Bench**：このデータセットは，Clustering benchmark datasets(`http://cs.uef.fi/sipu/datasets/`) [13] 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 である．

全てのデータセットに対して，類似度 $\mathbb{S}$ は負のユークリッド距離を用いた．また，preference は全ての実験において他の類似度の中央値に設定した．

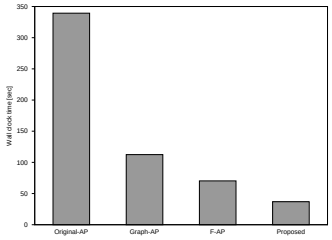

図 1　実行時間 (Perfume)

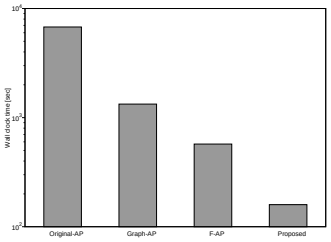

図 2　実行時間 (Geo)

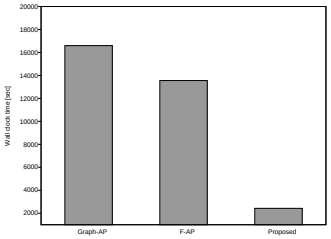

図 3　実行時間 (Bench)

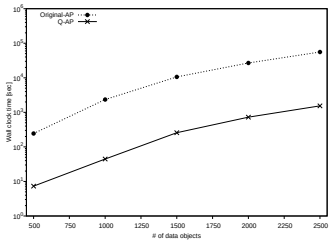

図 4　スケーラビリティ

表 1　正　確　性

	Graph-AP	F-AP	Proposed
Perfume (Recall)	1.00	1.00	1.00
Perfume (Precision)	1.00	1.00	1.00
Geo (Recall)	1.00	1.00	1.00
Geo (Precision)	1.00	1.00	1.00

4.2　高　速　性

3 種類のデータセットに対して各アルゴリズムを実行したときの実行時間を図 1–3 に示す．ここで，Bench データセットにおいて，従来の Affinity Propagation は 24 時間以内に実行が終了しなかったため，途中で実験を打ち切った．

図 1–3 より，提案手法は全てのデータセットに対して比較手法よりも高速に処理できていることがわかる．特に提案手法は，従来の Affinity Propagation や state-of-the-art とされる F-AP と比較してそれぞれ最大で 42.53 倍，5.59 倍の高速化を達成した．提案手法は，Graph-AP や F-AP と同様に反復計算を行うデータポイントペアを削減したことが実行時間の短縮につながっている．また，反復計算の途中でメッセージが収束したデータポイントペアに対しては以降の反復計算を行う対象から取り除いている．さらに，3 節で述べたように，定義 21 を用いた計算が不要なデータポイントペアを集約してメッセージを求めることでさらなる高速化を実現した．

4.3　スケーラビリティ

データ数を変化させることで提案手法のスケーラビリティを評価した．スケーラビリティを評価するために，さらに 5 つのデータセットを生成した．これらのデータセットは全て Bench データセットからランダムにそれぞれ 500，1,000，1,500，2,000，2,500 のデータをサンプリングしたものである．実験結果を図 4 に示す．図 4 より，提案手法は従来の Affinity Propagation と比較してデータ数の観点から優れたスケーラビリティを示していることがわかる．これは，4.2 章で示したように，responsibility と availability のメッセージ計算の際に定義 21 による計算が不要なデータポイントペアについて集約してメッセージを計算しているためである．

4.4　正　確　性

次に，提案手法におけるクラスタリング結果の正確性を実験的に確認する．本実験では精度を評価するため *Precision* と *Recall* [15] を用いる．ここで，Precision とは，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．一方で Recall 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．もしも提案手法，Graph-AP，F-AP で得られた exemplar と従来の Affinity Propagation で得られた exemplar が全て一致しているならば，Precision と Recall はいずれも 1 を示す．実験結果を表 1 に示す．ここで，Bench データセットについては，従来の Affinity Propagation が 24 時間以内に実行が終了せず途中で実行を打ち切ったため，クラスタリング結果が出力されていない．ゆえに結果は省略されている．表 1 より，Perfume データセットと Geo データセットいずれに対しても提案手法は 1 を出力している．定理 31 より，提案手法は精度を落とすことなく無駄な計算を省略できることが理論的に証明されている．以上より，提案手法は従来の 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が行えることを実験的に示した．

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データ分析技術の一つであり，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 [14] がある．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．

2007 年に Frey らによって提案された Affinity Propagation [1] は，ファクタグラフ [16] 上でメッセージ伝搬を行うことでグラフモデルの最適化を行う Belief Propagation [17,18] を応用したクラスタリングアルゴリズムである [19]．Affinity Propagation は k-means 法 [2] や k-medoids 法 [14] とは異なり，ユーザがクラスタ数を予め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 K-AP [20] や MEAP [21] など，多くの応用的な手法が提案されている．

しかし，第 2 節で述べたように，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 2008 年に FSAP [3] と呼ばれる高速化手法を提案した．FSAP は K-最近傍を用いて疎な K 近傍グラフを構築することで反復計算を行うメッセージの数を削減する．また，クラスタリング精度を向上させるために，構築した K 近傍グラフにエッジを追加し，そのグラフ上で再度反復計算を行う．これにより，従来の Affinity Propagation [1] と比較して処理時間の短縮を実現した．しかしこの手法はパラメータ K 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 [4] 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．

Fujiwara らは Graph-AP をベースにさらなる高速化を実現する F-AP [5] 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．しかしこの手法は，枝刈りされなかったデータポイントペアのメッセージの中にも複雑な計算が不要なメッセージが含まれており，無駄な計算を削りきれていない．

6 おわりに

本稿では，大規模データに対する Affinity Propagation の高速化手法を提案した．

提案手法は，既存手法で行われていた複雑な計算を省略できるデータポイントペアを特定し，それらを集約してメッセージを求めることで高速化を図った．評価実験では，提案手法がクラスタリング精度を損なうことなく，既存手法 [1,4,5] よりも高速に処理できることを示した．

今後の課題として，より大規模なデータでの実験やさらなる高速化があげられる．

7 謝　辞

本研究の一部は JST ACT-I ならびに JSPS 科研費 JP18K18057 による支援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] 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.

[4] 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.

[5] 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.

[6] A. Rangrej, S. Kulkarni, and A. V. Tendulkar, “Comparative Study of Clustering Techniques for Short Text Document,” in Proceedings of the 20th international conference companion on World wide web, pp. 111–112, 2011.

[7] Z. Liu, P. Li, and Y. Zheng, and M. Sun, “Community Detection by Affinity Propagationt,” Technical Report, 2008.

[8] C. Jia, Y. Jiang, and J. Yu, “Affinity Propagation on Identifying Communities in Social and Biological Networks,” International Conference on Knowledge Science, Engineering and Management, pp. 597–602, 2010.

[9] Z. Liu, P. Li, and Y. Zheng, and M. Sun, “Clustering to Find Exemplar Terms for Keyphrase Extraction,” in Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1, pp. 257–266, 2009.

[10] H. H. Alrehamy, and C. Walker, “SemCluster: Unsupervised Automatic Keyphrase Extraction Using Affinity Propagation,” UK Workshop on Computational Intelligence, pp. 222–235, 2017.

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

[12] General Affairs Planning Bureau, “Kitakyushu city facilities (common format),” 2018, `https://ckan.open-governmentdata.org/dataset/401005_shinoshisetsu`.

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

[14] K. Leonard, R. Peter J, “Clustering by Means of Medoids,” Elsevier, pp. 405–416, 1987.

[15] C. D. Manning, P. Raghavan, and H. Schütze, “Introduction to Information Retrieval,” Cambridge University Press, 2008.

[16] F. R. Kschischang, B. J. Frey, and H-A. Loeliger, “Factor Graphs and the Sum-Product Algorithm,” IEEE Transactions on information theory, Vol. 47, No. 2, pp. 498–519, 2001.

[17] Y. Weiss, and W. T. Freeman, “On the Optimality of Solutions of the Max-Product Belief-Propagation Algorithm in Arbitrary Graphs,” IEEE Transactions on information theory, Vol. 47, No. 2, pp. 736–744, 2001.

[18] A. Riazanov, y. Maximov, and M. Chertkov, “Belief Propagation Min-Sum Algorithm for Generalized Min-Cost Network Flow,” 2018 Annual American Control Conference (ACC), pp. 6108–6113, 2018.

[19] I. E. Givoni, and B. J. Frey, “A Binary Variable Model for Affinity Propagation,” Neural computation, Vol. 21, No. 6, pp. 1589–1600, 2009.

[20] Z. Xiangliang, W. Wei, N. Kjetil, S. Michele, “K-AP: Generating Specified K Clusters by Efficient Affinity Propagation,” in Proc ICDM, pp. 1187–1192, 2010.

[21] 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.