D2 - クラスタリング

嵯峨山 茂樹 ＜sagayama@hil.t.u-tokyo.ac.jp＞
東京大学 工学部 計数工学科
資料所在 http://hil.t.u-tokyo.ac.jp/∼sagayama/applied_acoustics/

- クラスタリング
- k-means アルゴリズム
- ベクトル量子化
- LBG アルゴリズム
- セグメンタル k-means アルゴリズム

クラスタリング[1]

内容

1. クラスタリング
2. k-means アルゴリズム、LBG アルゴリズム
3. ベクトル量子化

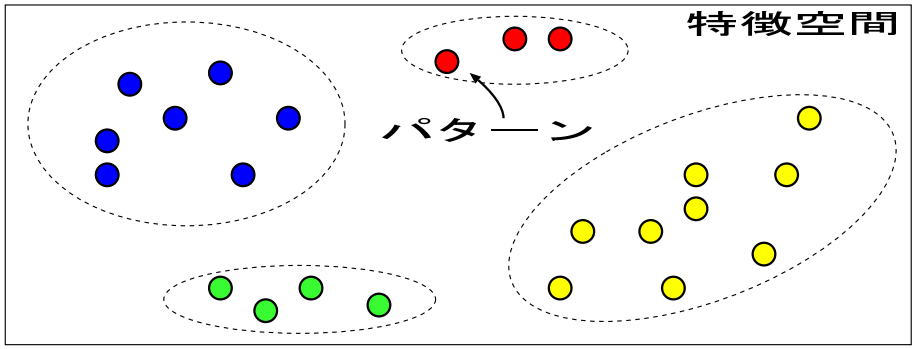
参考文献

- 原島博 監修「画像情報圧縮」オーム社
- 堀内 司朗、有村 一朗 監修「画像圧縮技術のはなし」工業調査会
- 石井 健一郎、上田 修功、前田 英作、村瀬 洋「よくわかるパターン認識」オーム社
- 中川 聖一「パターン情報処理」丸善
- 古井 貞熙「音声情報処理」森北出版
- 谷萩 隆嗣「音声と画像のディジタル信号処理」コロナ社
- 中川 聖一「確率モデルによる音声認識」コロナ社
- 森 健一「パターン認識」コロナ社
- 長尾 真「パターン情報処理」コロナ社
- 鳥脇純一郎「認識工学」コロナ社
- Lawrence Rabiner, Biing-hwang Juang 著, 古井貞熙 監訳「音声認識の基礎」
- 北 研二・中村 哲・永田 昌明「音声言語処理」森北出版
- `http://blitz.ec.t.kanazawa-u.ac.jp/~shinji/study/k-NN/lbg.html` (LBG アルゴリズム)

クラスタリングとベクトル量子化

クラスタリング

- 類似したパターン同士を同じクラスにまとめること
  - 階層的クラスタリング(最近隣法基準)
  - k-平均アルゴリズム
  - C-平均ファジィクラスタリング法

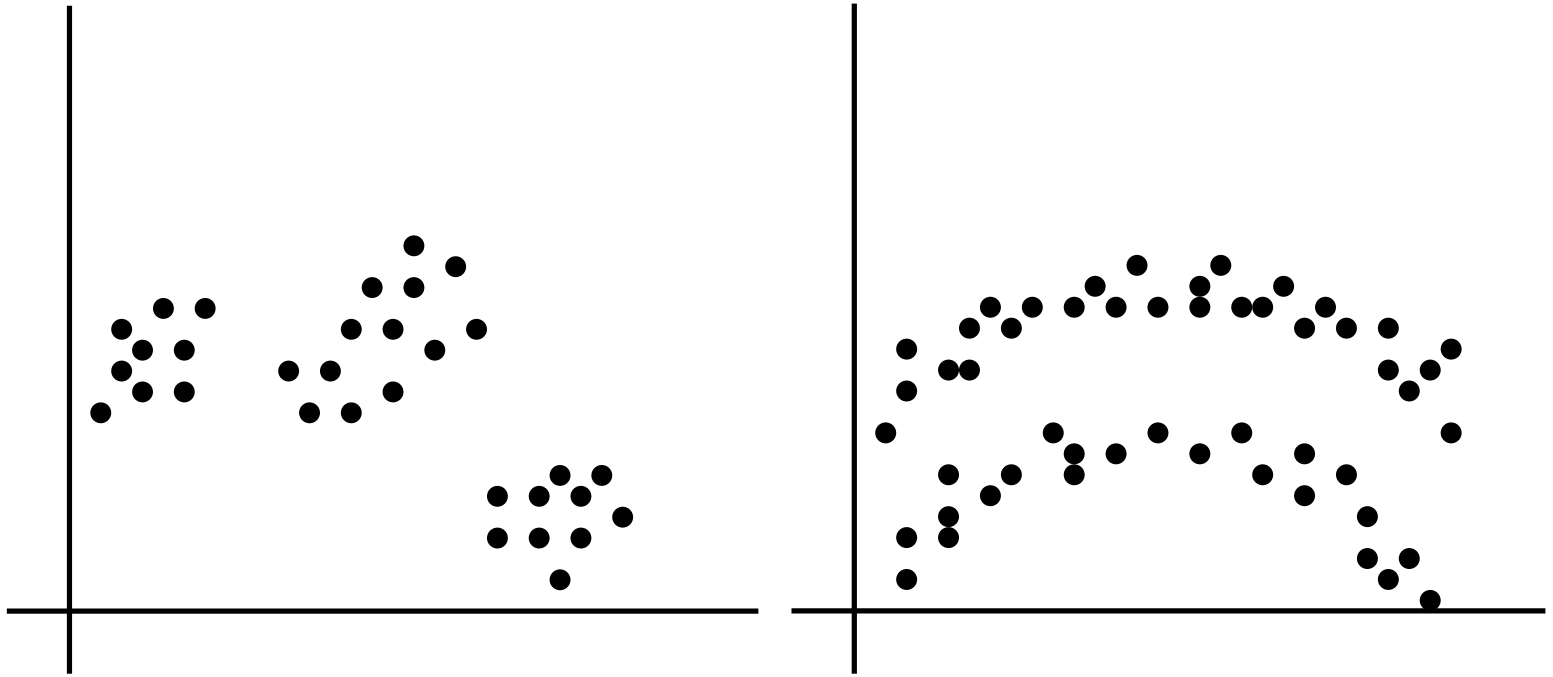


ベクトル量子化

- 観測パターンを有限個の代表的なパターンの１つに置き換えること
  - LBGアルゴリズム

サンプルパターンのグループ

たくさんのサンプルパターン $\boldsymbol{x} = (a_1, a_2, \ldots, a_n)$ という n 次元ベクトルを、n 次元空間にプロットすると、いくつかのグループに分かれる。



クラスタ

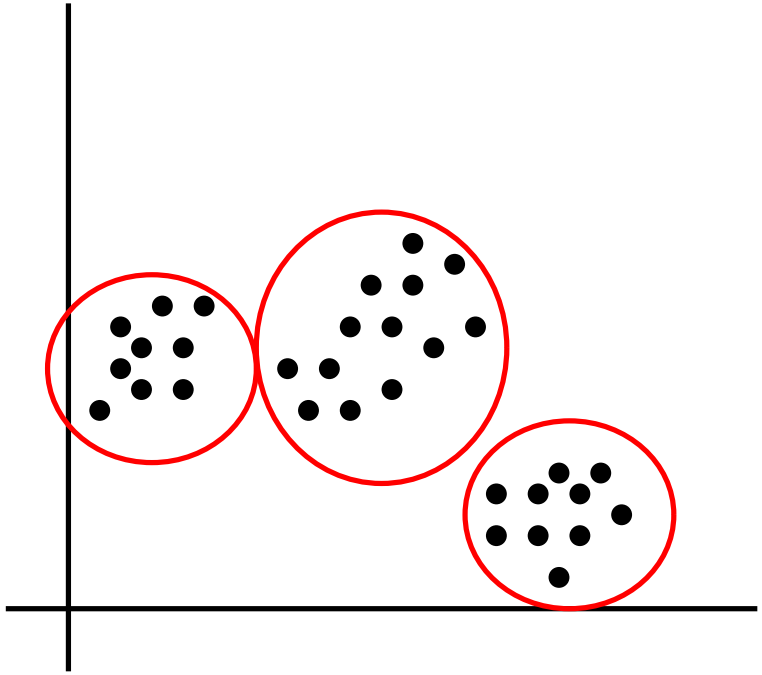


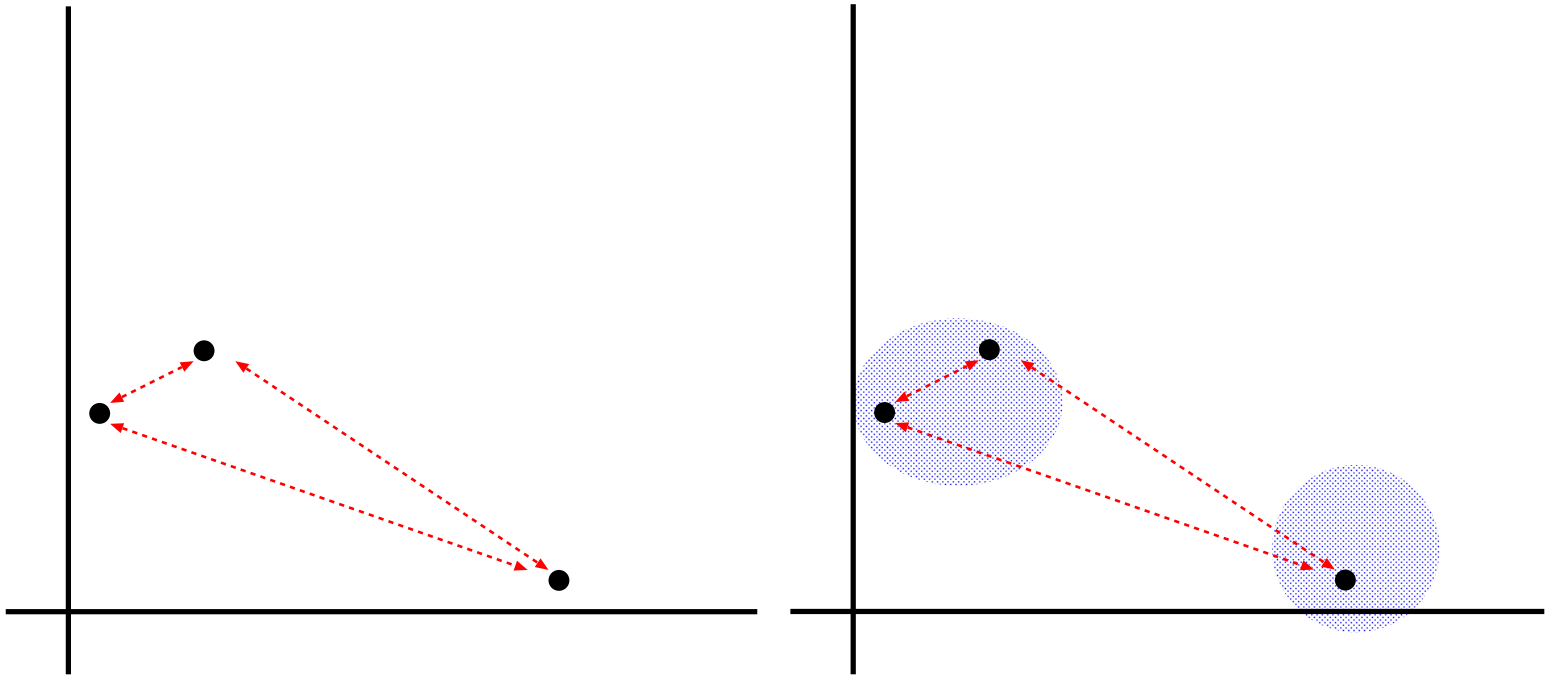
図1. 形成されているクラスタの例

この状態を「クラスタ(cluster)が形成されている」という。
クラスタは一つのクラスに属するデータであると考えられる。
サンプルパターンxがどのクラスに属するか、あらかじめ分かっていないものを分類（教師なし学習）。

クラスタリング

クラスタリング(clustering)

サンプルパターン同士の距離尺度(あるいは類似性、近さの尺度)Dを導入し、Dが小さいサンプルばかりを集めて固まり（クラスタ）を作るという操作。



$$D_1 \ll D_2 < D_3$$

セントロイド (centroid)

Euclid距離の2乗の場合:

- **クラスタ内歪み (ベクトル量子化誤差)：距離の総和**

$$D = \sum_{i=1}^{n} \| \boldsymbol{x}_i - \boldsymbol{m} \|^2 = \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - m_j)^2$$

を最小化するようなセントロイドmは何か？

- **jごとに**

$$\frac{\partial D}{\partial m_j} = \sum_{i=1}^{n} \frac{\partial}{\partial m} (x_{ij} - m_j)^2 = -2 \sum_{i=1}^{n} (x_{ij} - m_j) = -2 \left\{ \sum_{i=1}^{n} x_{ij} - n m_j \right\} = 0$$

とおいて、

$$m_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$$

を得る。すなわち、

$$\boldsymbol{m} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i$$

- **算術平均は、ユークリッド距離に関するセントロイド**

歪み尺度 (評価関数)

- **クラスタリングの目的**
  データをいくつかのクラスタに分けること

- **評価関数**
  総歪み (ベクトル量子化誤差)

$$J = \sum_{j=1}^{N_c} \sum_{x \in C_j} \| x - m_j \|^2$$

平均ベクトル (centroid)

$$m_j = \frac{1}{N_j} \sum_{x \in C_j} x$$

N_c：**クラスタの数**
C_j：j**番目のクラスタに属するサンプル集合**
N_j：C_j**のサンプル数**

クラスタリング手法例1: 単純クラスタリング

手順

- 閾値Tを設定する。
- 任意のサンプルx_1をクラスタの中心z_1とする。
- $x_i(i = 2, \ldots, N)$とz_1の距離Dを計算。
- $D \leq T$ならx_iはz_1を中心とするクラスタに属する。
- $D > T$ならx_iを新たなクラスタ中心z_2とする。

特徴

- 大切なのは、閾値Tの選び方
- Tや$\{x_1, \ldots, x_n\}$の順番を変えたりして何度かやる。

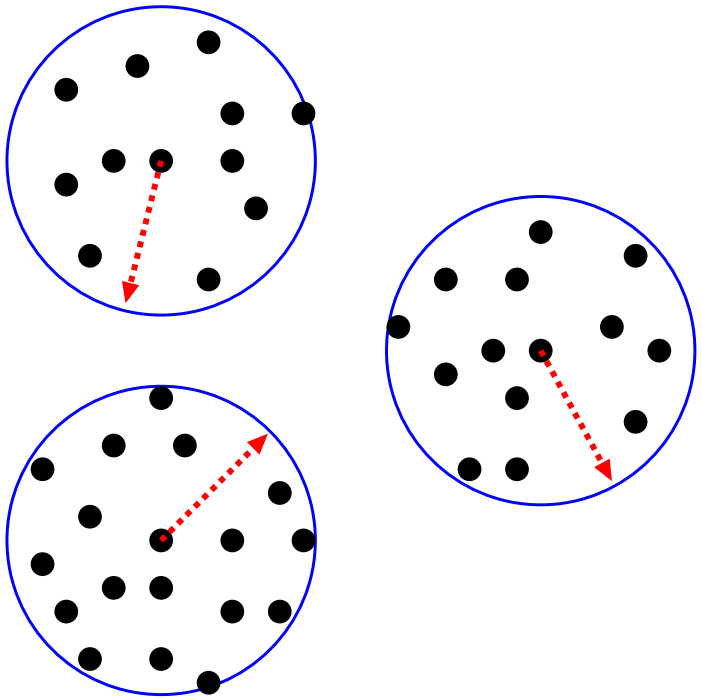


図2. 単純クラスタリング

クラスタリング手法例2: 階層的クラスタリング

手順

- 距離のもっとも近い２つのサンプルを１つのクラスタにまとめる。
- 上記の繰り返しでクラスタの数を減らしていく。

特徴

- クラスタの数を減らせる？
- 欠点
  - 計算量が多い

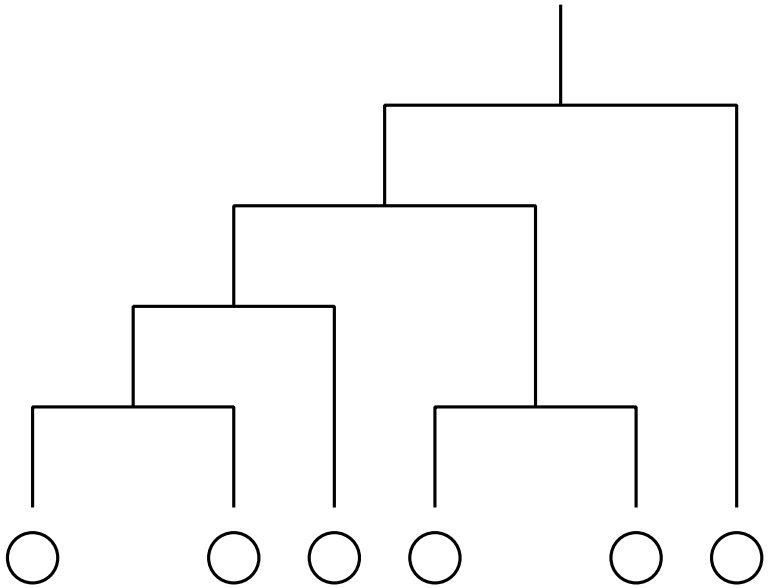


図3. 階層的クラスタリング

階層的クラスタリング

→ クラスタを階層的に統合していく ボトムアップなクラスタリング手法

- 手法の概要
  1. 各サンプルすべてをクラスタとして設定
  2. クラスタを1つずつ統合していく(クラスタ数が1個ずつ減少)
  3. 終了条件を満たすまで2.を続ける

- 手法の特徴
  - クラスタリング後，クラスタ内のサンプル間における階層的な関係を知ることができる
  - クラスタ数を指定する必要がない → サンプルの中にいくつのパターンが内在しているか不明のときに有効

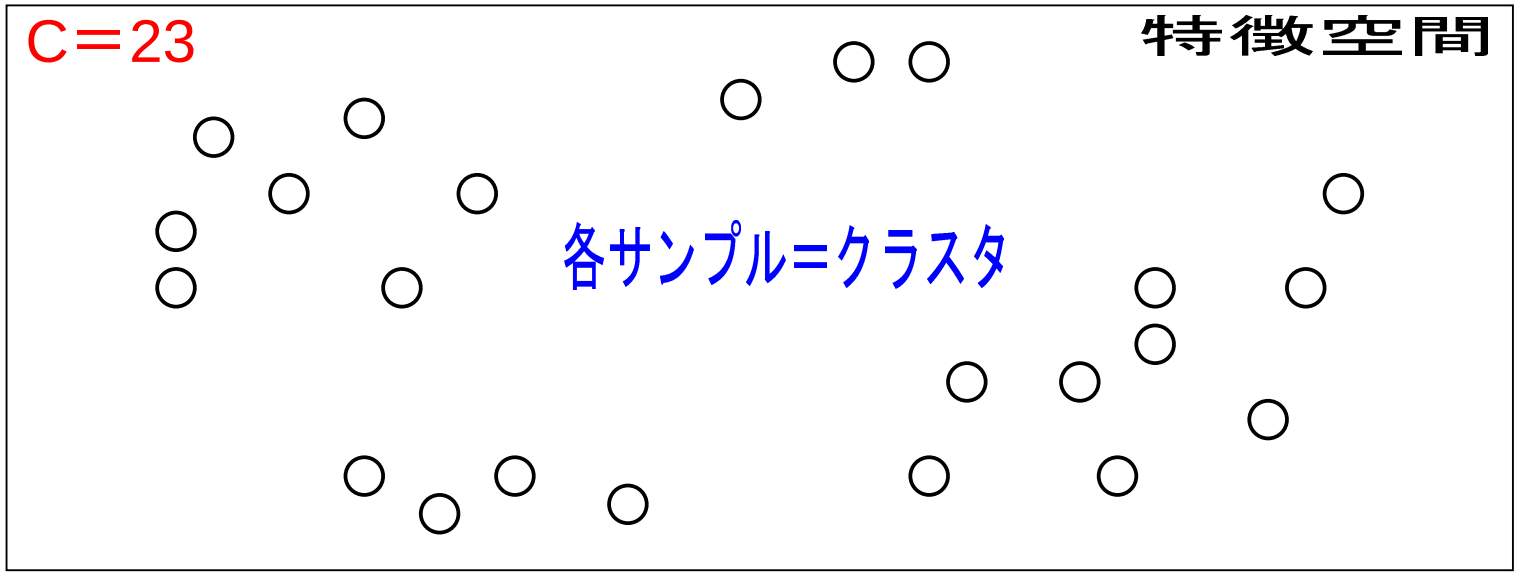
階層的クラスタリング(最近隣法基準)

＜評価基準＞

$$J = \sum_{k=1}^{C} \left\{ \frac{1}{2} \sum_{\{x_i, x_j\} \in C_k} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$

＜統合方法＞クラスタkとクラスタlを統合

$$\{k, l\} = \underset{k,l}{\operatorname{argmin}} \left\{ \min_{x_i \in C_k, x_j \in C_l} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$



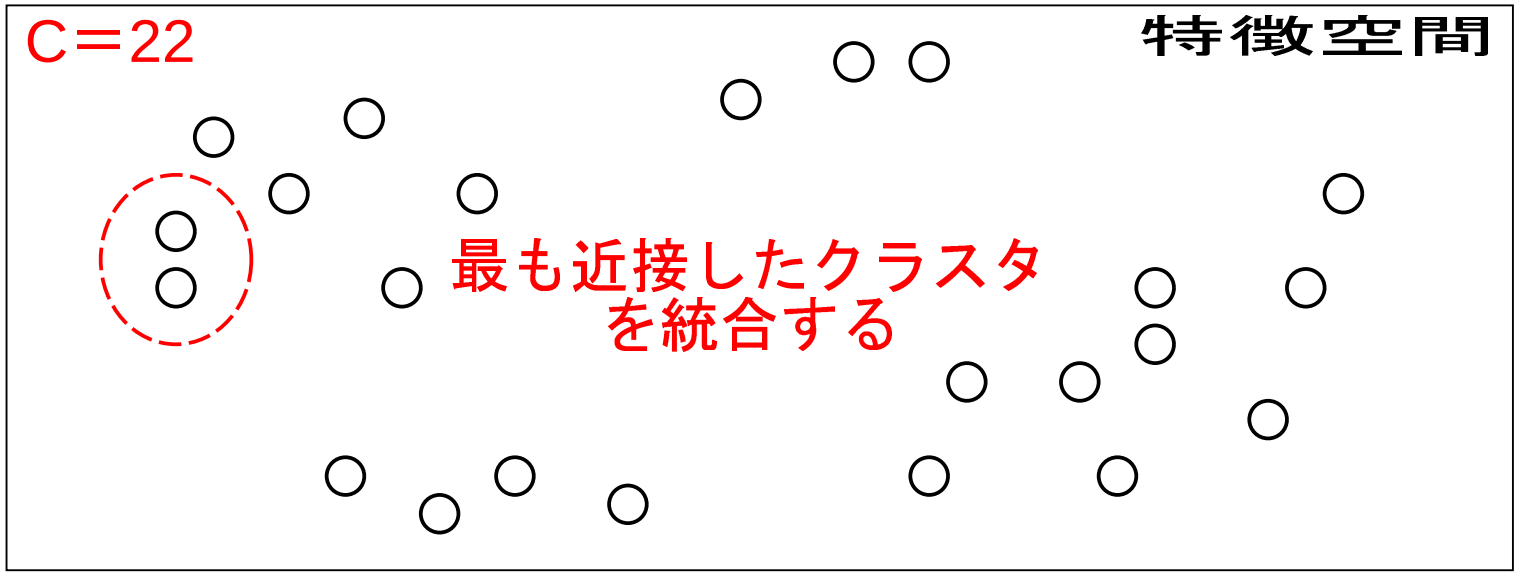
階層的クラスタリング(最近隣法基準)

＜評価基準＞

$$J = \sum_{k=1}^{C} \left\{ \frac{1}{2} \sum_{\{x_i, x_j\} \in C_k} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$

＜統合方法＞クラスタkとクラスタlを統合

$$\{k, l\} = \underset{k,l}{\operatorname{argmin}} \left\{ \min_{x_i \in C_k, x_j \in C_l} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$



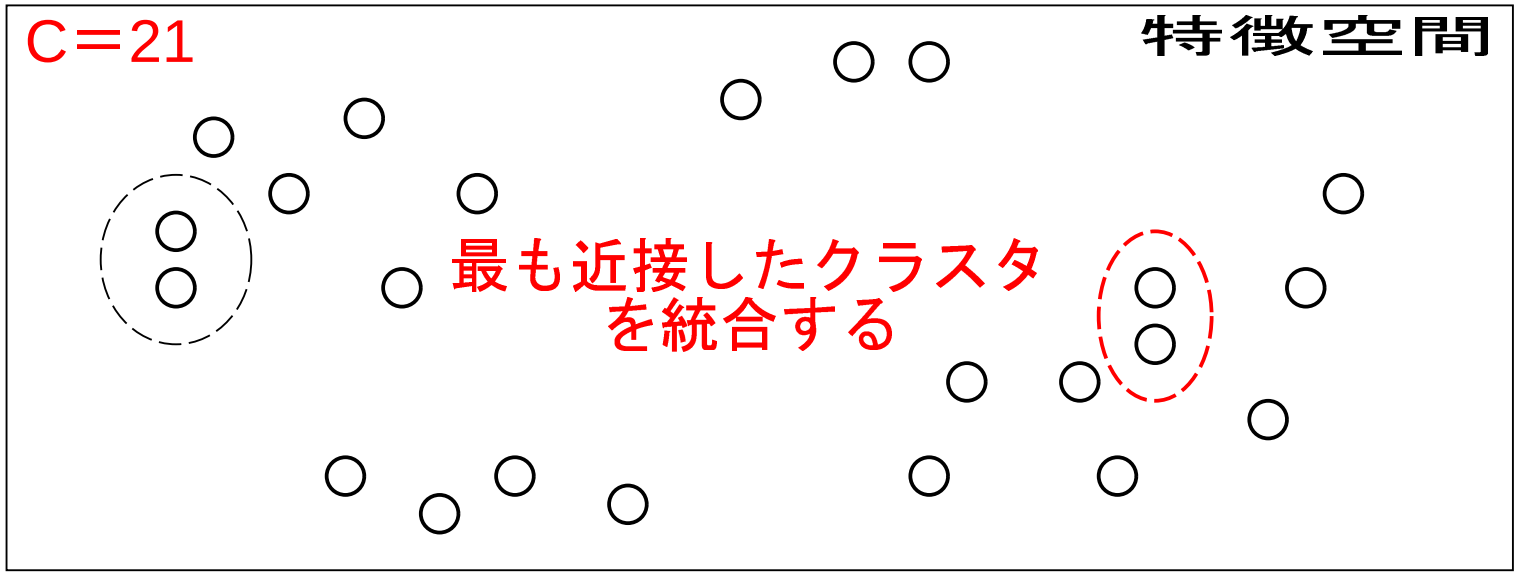
階層的クラスタリング(最近隣法基準)

＜評価基準＞

$$J = \sum_{k=1}^{C} \left\{ \frac{1}{2} \sum_{\{x_i, x_j\} \in C_k} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$

＜統合方法＞クラスタkとクラスタlを統合

$$\{k, l\} = \underset{k,l}{\operatorname{argmin}} \left\{ \min_{x_i \in C_k, x_j \in C_l} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$



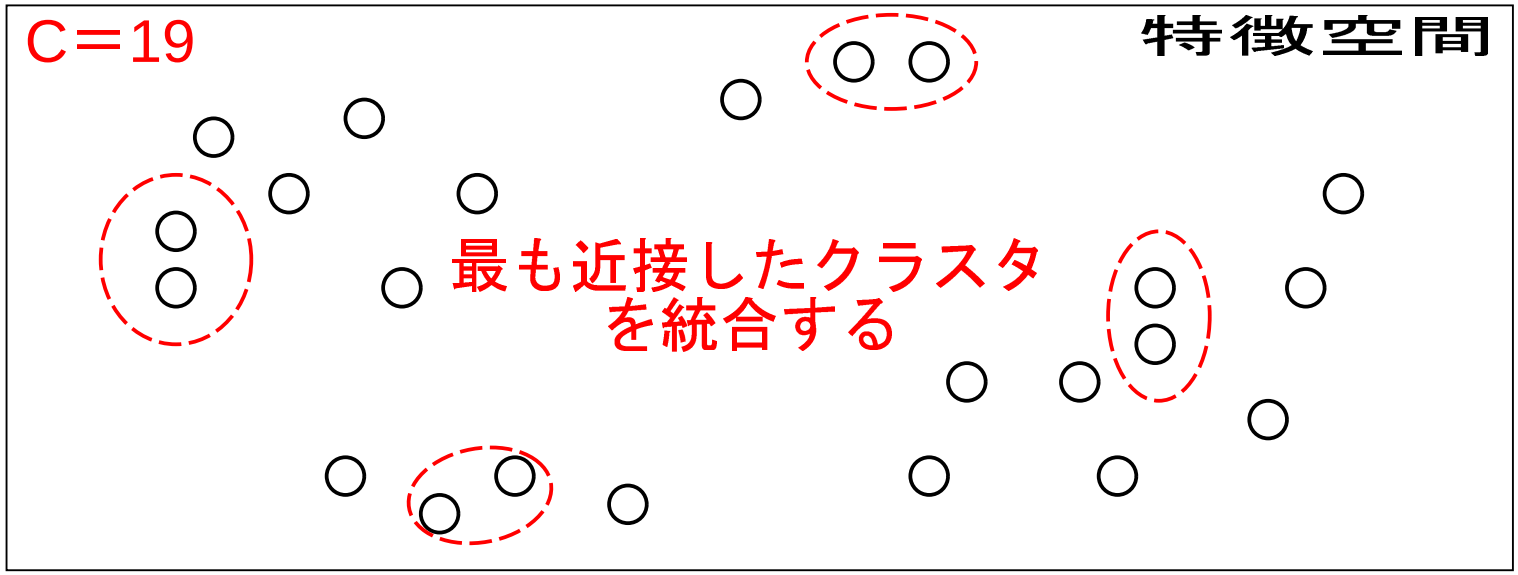
階層的クラスタリング(最近隣法基準)

＜評価基準＞

$$J = \sum_{k=1}^{C} \left\{ \frac{1}{2} \sum_{\{x_i, x_j\} \in C_k} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$

＜統合方法＞クラスタkとクラスタlを統合

$$\{k, l\} = \underset{k,l}{\operatorname{argmin}} \left\{ \min_{x_i \in C_k, x_j \in C_l} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$



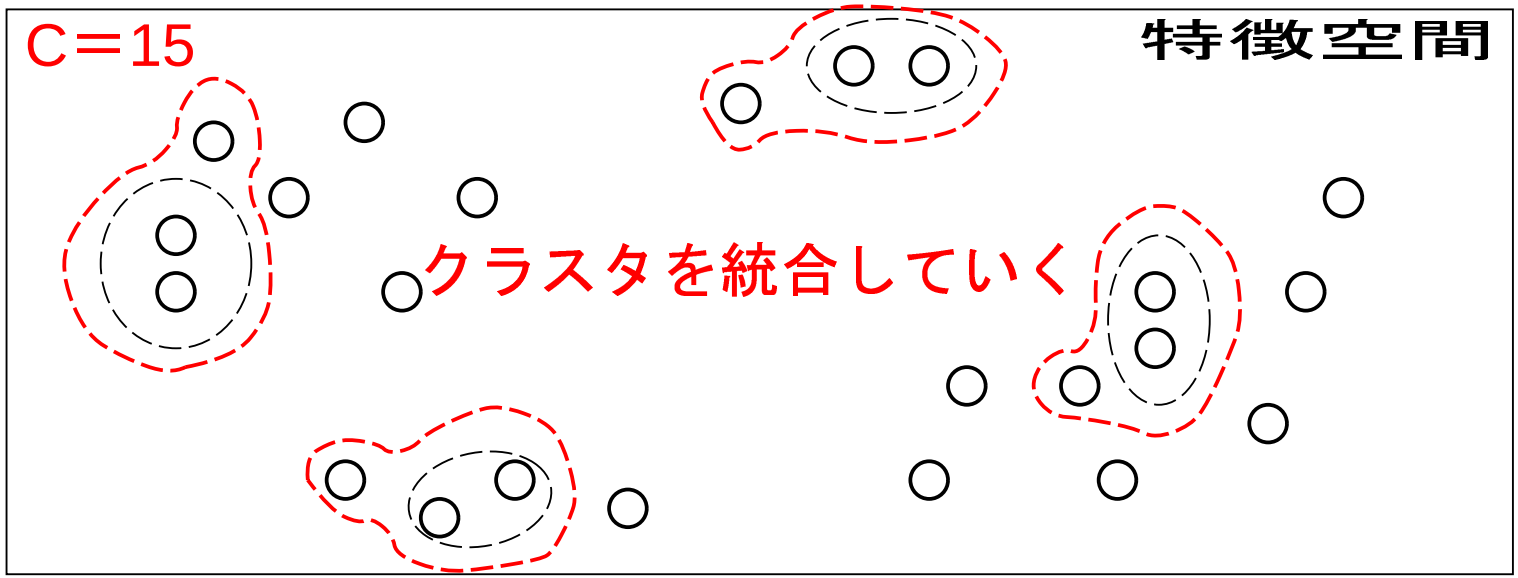
階層的クラスタリング(最近隣法基準)

＜評価基準＞

$$J = \sum_{k=1}^{C} \left\{ \frac{1}{2} \sum_{\{x_i, x_j\} \in C_k} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$

＜統合方法＞クラスタkとクラスタlを統合

$$\{k, l\} = \underset{k,l}{\operatorname{argmin}} \left\{ \min_{x_i \in C_k, x_j \in C_l} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$



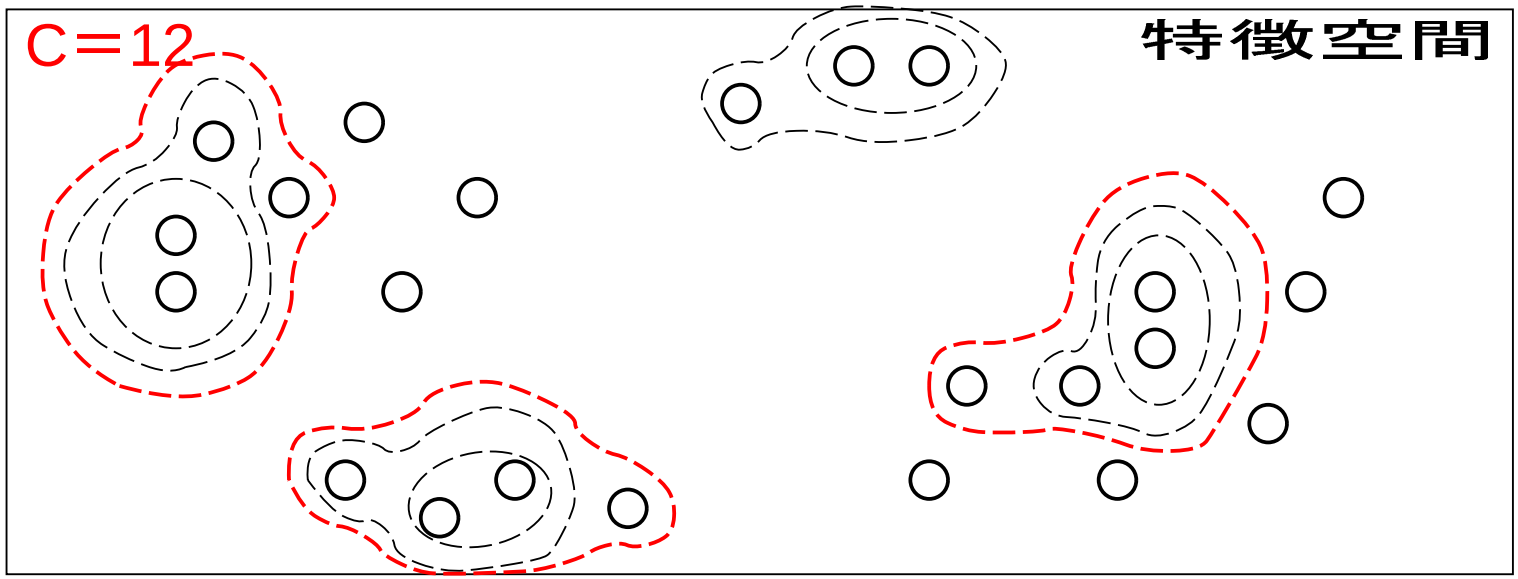
階層的クラスタリング(最近隣法基準)

＜評価基準＞

$$J = \sum_{k=1}^{C} \left\{ \frac{1}{2} \sum_{\{x_i, x_j\} \in C_k} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$

＜統合方法＞クラスタkとクラスタlを統合

$$\{k, l\} = \underset{k,l}{\operatorname{argmin}} \left\{ \min_{x_i \in C_k, x_j \in C_l} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$



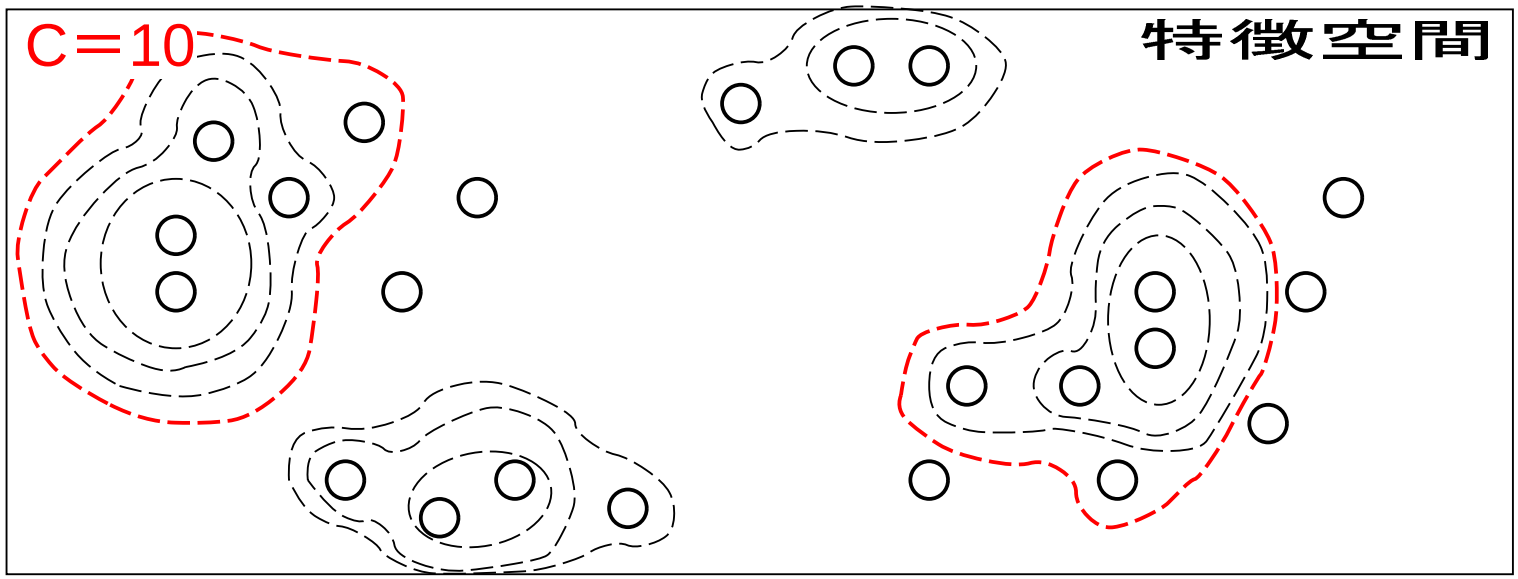
階層的クラスタリング(最近隣法基準)

＜評価基準＞

$$J = \sum_{k=1}^{C} \left\{ \frac{1}{2} \sum_{\{x_i, x_j\} \in C_k} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$

＜統合方法＞クラスタkとクラスタlを統合

$$\{k, l\} = \underset{k,l}{\operatorname{argmin}} \left\{ \min_{x_i \in C_k, x_j \in C_l} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$



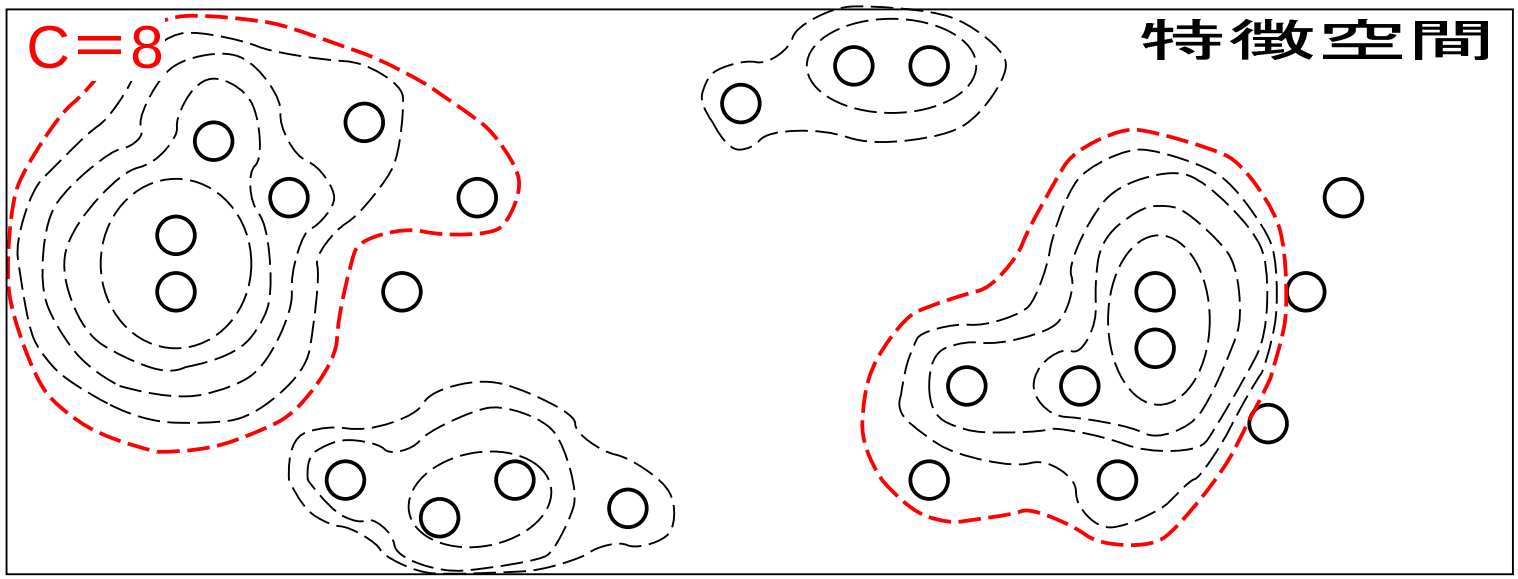
階層的クラスタリング(最近隣法基準)

＜評価基準＞

$$J = \sum_{k=1}^{C} \left\{ \frac{1}{2} \sum_{\{x_i, x_j\} \in C_k} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$

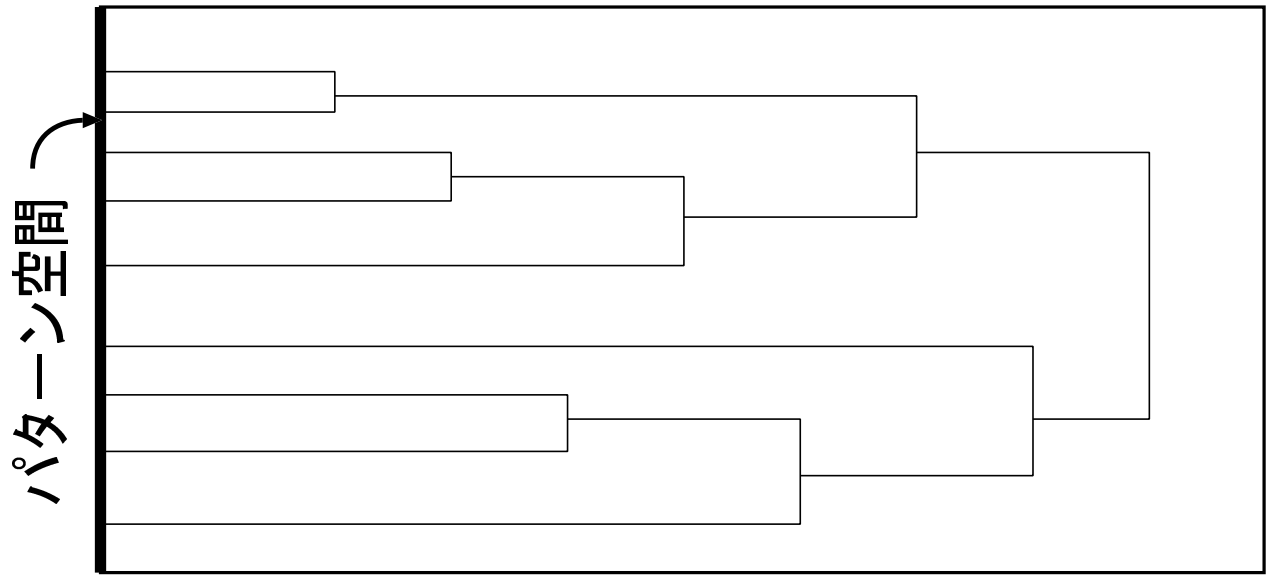
＜統合方法＞クラスタkとクラスタlを統合

$$\{k, l\} = \underset{k,l}{\operatorname{argmin}} \left\{ \min_{x_i \in C_k, x_j \in C_l} |\boldsymbol{x}_i - \boldsymbol{x}_j| \right\}$$



階層的クラスタリング(最近隣法基準)

- デンドログラム
  クラスタリング後，以下のようなサンプル間の階層関係が得られる



- 終了条件は??
  - 評価関数値
  - AIC(クラスタ数が大きいほど大きなペナルティ)
- おまけ
  - トップダウン的に行うこともできる

クラスタリング手法例3: 最大距離アルゴリズム

特徴

- クラスタの中心間の距離を最大にする。

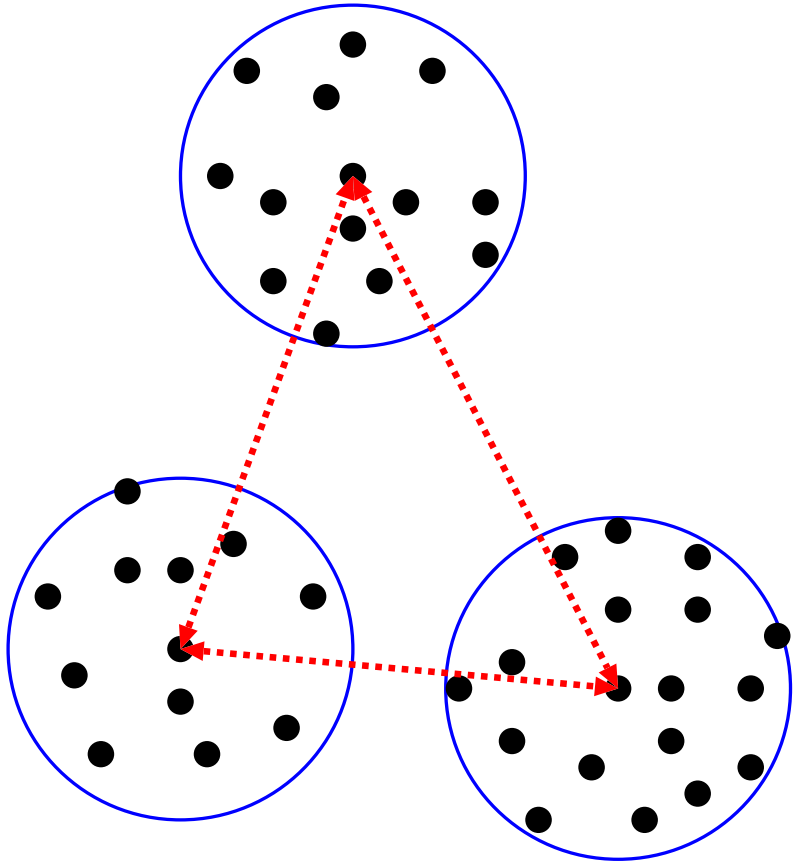


図4. 最大距離アルゴリズム

k-means アルゴリズム: 導入

例： ある村に多数の家が建っているとしよう。ここにn台の郵便ポストを設置したい。その設置法は、次の通り。(道や傾斜など地理的な状況は考えない。)

(1) 各戸は、最も近いポストを使用する。→ 各戸にポスト番号を付与

(2) 各ポストはその使用者の全戸から最も近い地点に設置する。→ ポスト位置を更新

これを適当な初期解か始めて収束するまで繰り返すのが k-means アルゴリズムの概要である。実際には、距離の定義は物理的定義とは限らず、様々な距離が用いられる。

問題：以上を繰り返すとき最小解 (初期値によっては極小解) に収束することを証明せよ。→ 微分などを用いないで極小解に到達できる点に注意。

問題：ポストは、どこかの家に設置する場合でも、上記のアルゴリズムは同様に収束することを証明せよ。

問題：上記のアルゴリズムは、ユークリッド距離の場合には具体的にどのような計算となるか。

2-step の最小化

- 2-step 最小化 (実際は極小)

  (1) 各戸は、最も近いポストを使用する。
  (各戸にポスト番号を付与)
  → 各戸はより近いポストに変更する
  → 総距離は減少(少なくとも非増大)

  (2) 各ポストは、その使用者の全戸から最も近い地点に設置する。
  (ポスト位置を更新)
  → 総距離は減少(少なくとも非増大)

- 収束判定： 上記の変化が起きなくなったら停止

二乗Euclid距離に関するセントロイドの意味

- n 次元ベクトルサンプル値の集合 $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_N\}$ に対して、Euclid 距離二乗和

$$D = \sum_{i=0}^{N-1} |\boldsymbol{x}_i - \boldsymbol{m}|^2$$

を最小にする m (セントロイド) は、

$$\boldsymbol{m} = \frac{1}{N} \sum_{i=0}^{N-1} \boldsymbol{x}_i \qquad \text{(算術平均)}$$

である。

- 導出:

$\frac{dD}{d\boldsymbol{m}^T} = -2 \sum_{i=0}^{N-1} (\boldsymbol{x}_i - \boldsymbol{m}) = \boldsymbol{0}$ と置く (つまり、ベクトル成分ごとに、$\frac{\partial D}{\partial m_j} = -2 \sum_{i=0}^{N-1} (x_{ij} - m_j) = 0, \quad j = 1, \cdots, n$ と置く) ことにより、

$\sum_{i=0}^{N-1} \boldsymbol{x}_i = N\boldsymbol{m}$ すなわち、$\boldsymbol{m} = \frac{1}{N} \sum_{i=0}^{N-1} \boldsymbol{x}_i$ を得る。

k-means アルゴリズム (Lloyd アルゴリズム)

- k 個のクラスタを作るアルゴリズム。(適切な k は未解決問題。)
- 2 ステップの歪み最小化 (実は極小化) の繰り返しアルゴリズム (収束)

  初期解: 適当に (ランダムに) k 個の仮セントロイドを決めて開始。
  Step 1: セントロイドに従って全サンプルをクラスタ再分類 → D 減少
  Step 2: クラスタのサンプルからセントロイドを再計算 → D 減少
  以上を収束するまで繰り返す。

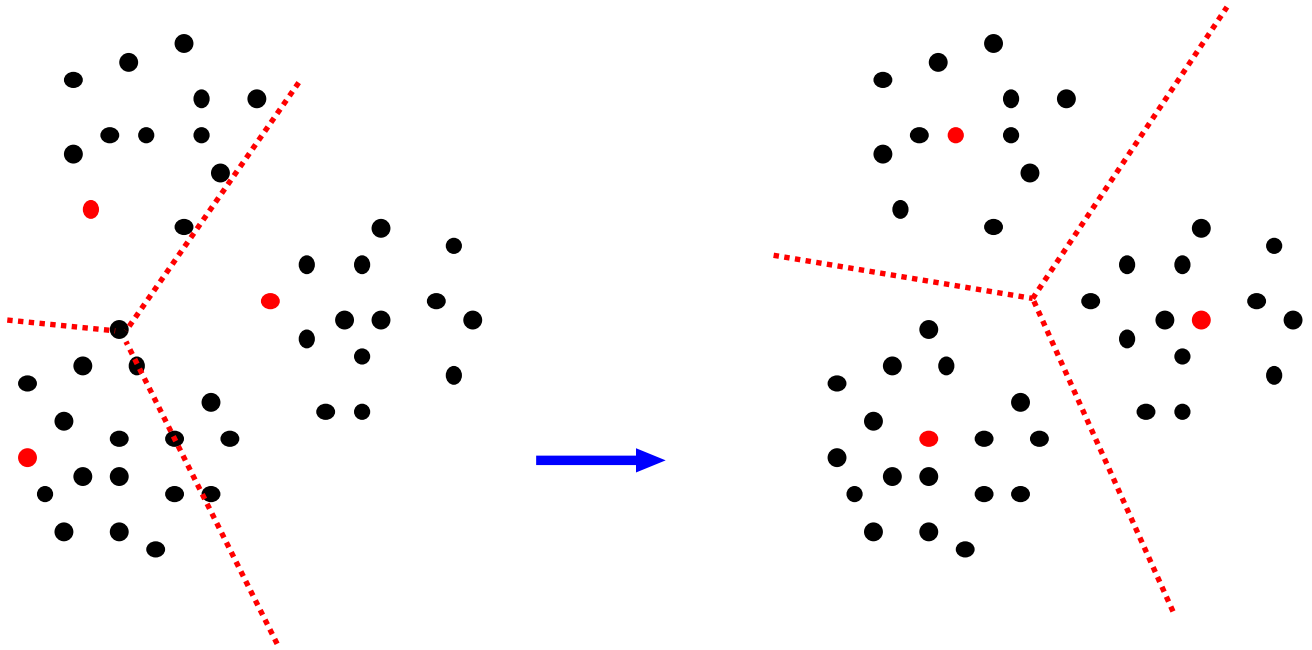


図 5. k-means アルゴリズムの 2 ステップ: (1) cluster 再分類、(2) centroid 再計算 – 境界線 (赤) は Voronoi 図になっている

k-means アルゴリズムの特徴

- クラスタ重心とクラスタ境界を繰返し計算により更新していく手法
- 手法の特徴
  - クラスタ重心の収束性が保証されている
  - 少ない計算量
    - サンプルとセントロイド間の距離(kN 個)を繰り返し計算
    - しかし、サンプル間の距離($N(N-1)/2$個)は不要
  - クラスタ数を指定する必要がある→ サンプルの中にいくつのパターンが内在しているか明らかなときには有効

k-means アルゴリズム

＜評価基準＞

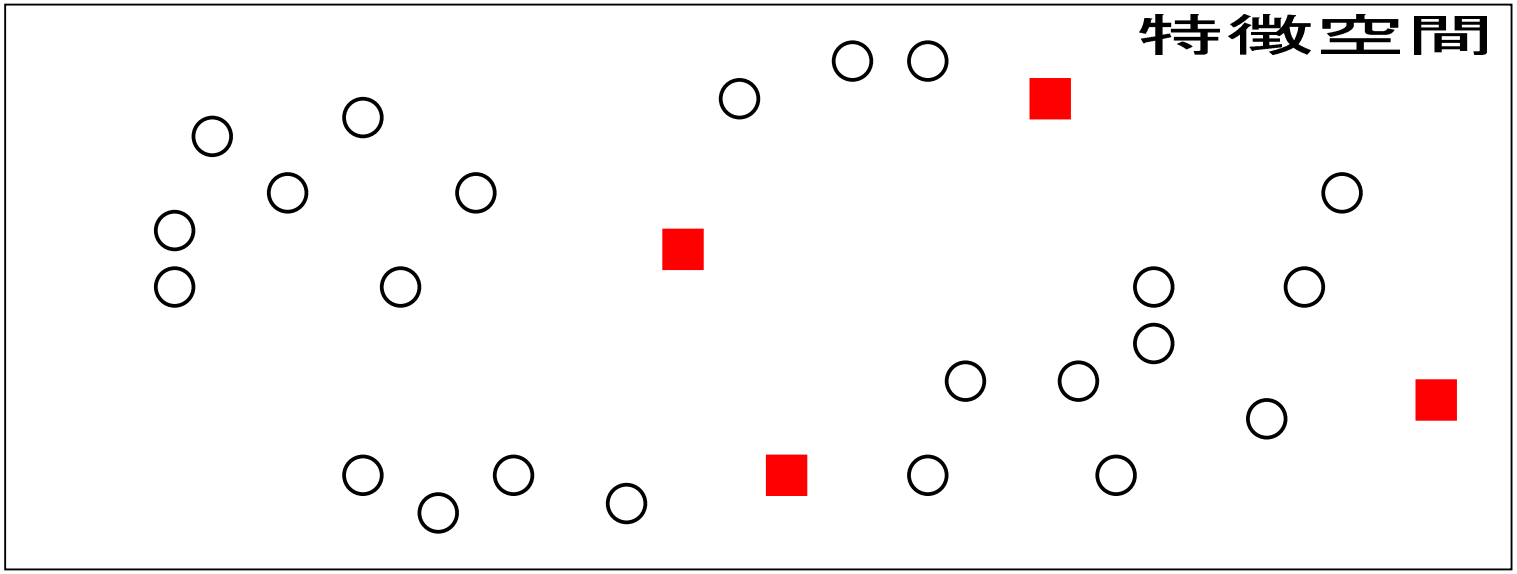
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



k-means アルゴリズム

＜評価基準＞

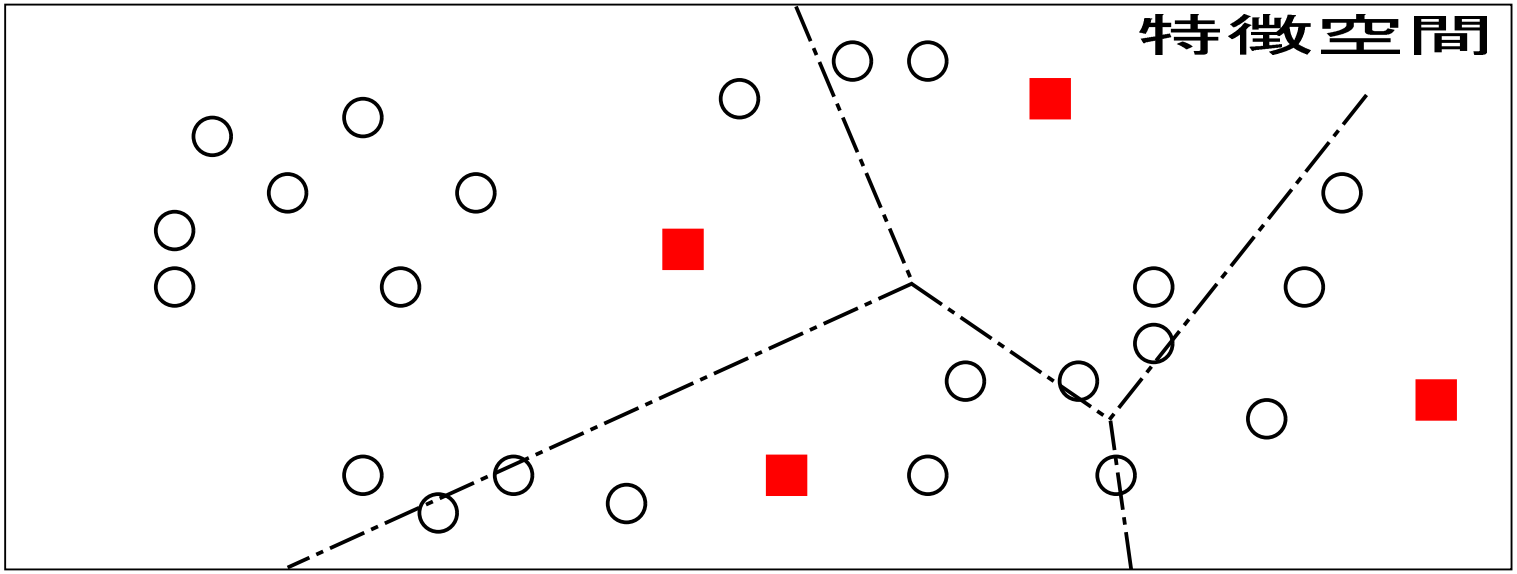
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



k-means アルゴリズム

＜評価基準＞

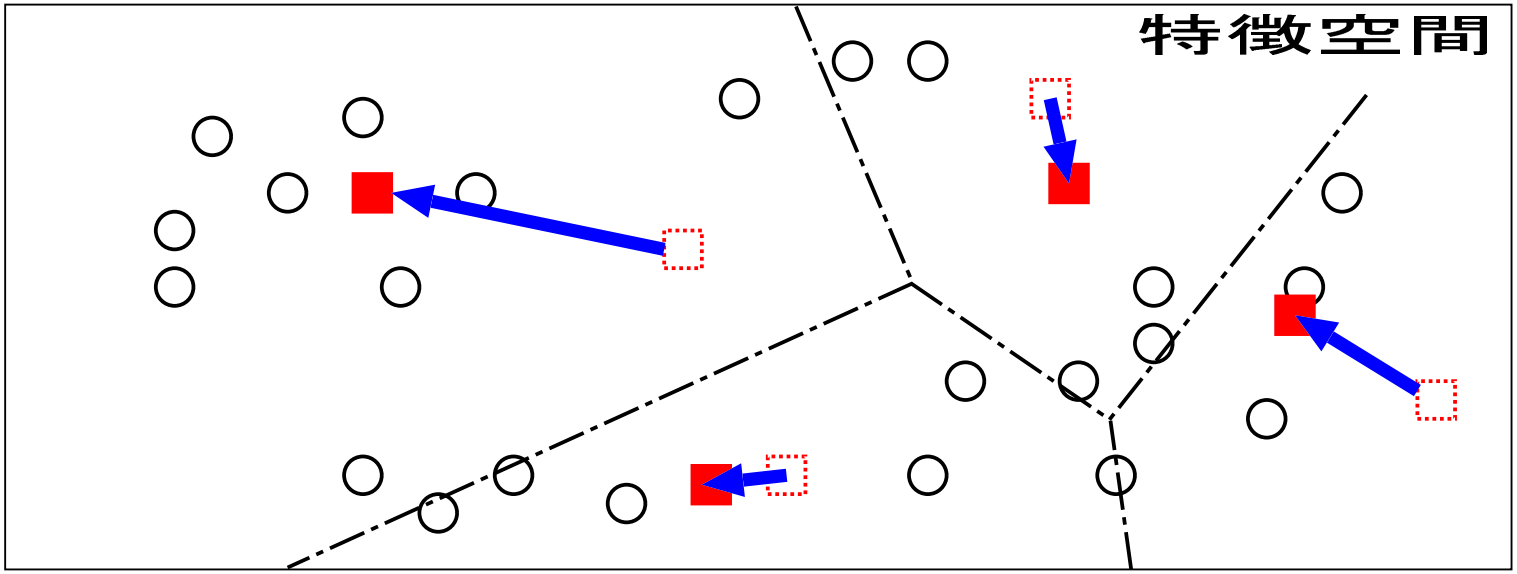
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



k-means アルゴリズム

＜評価基準＞

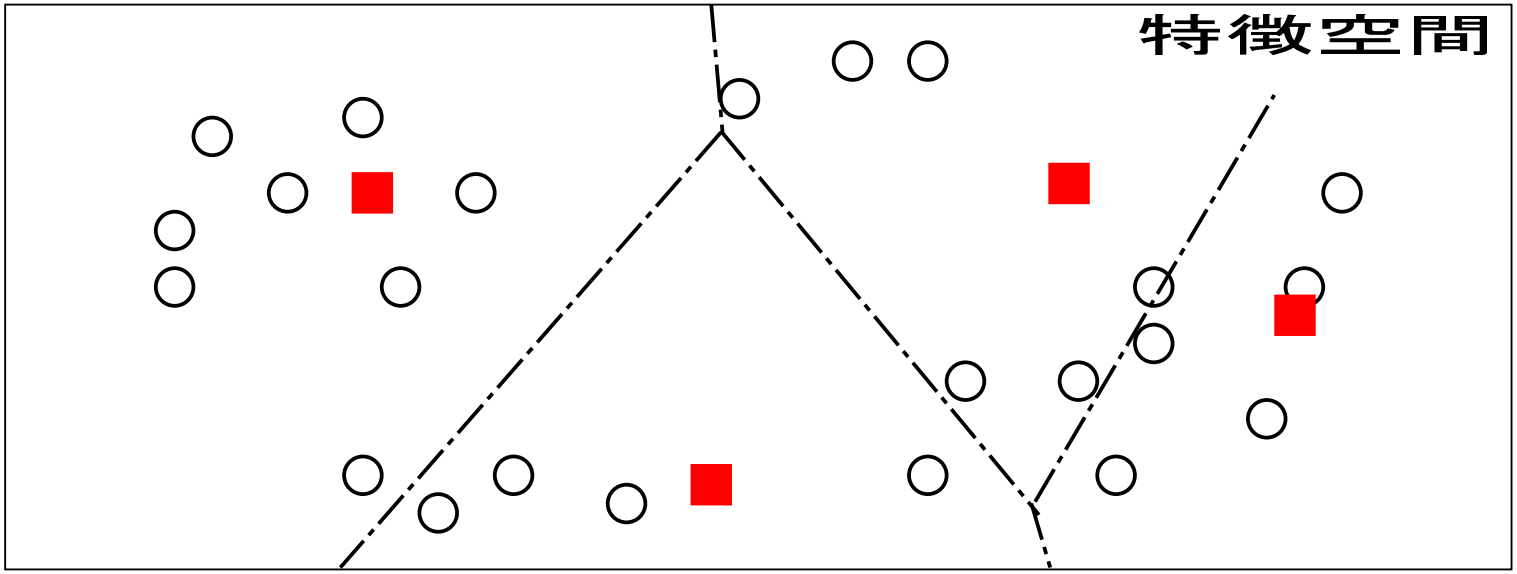
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



k-means アルゴリズム

＜評価基準＞

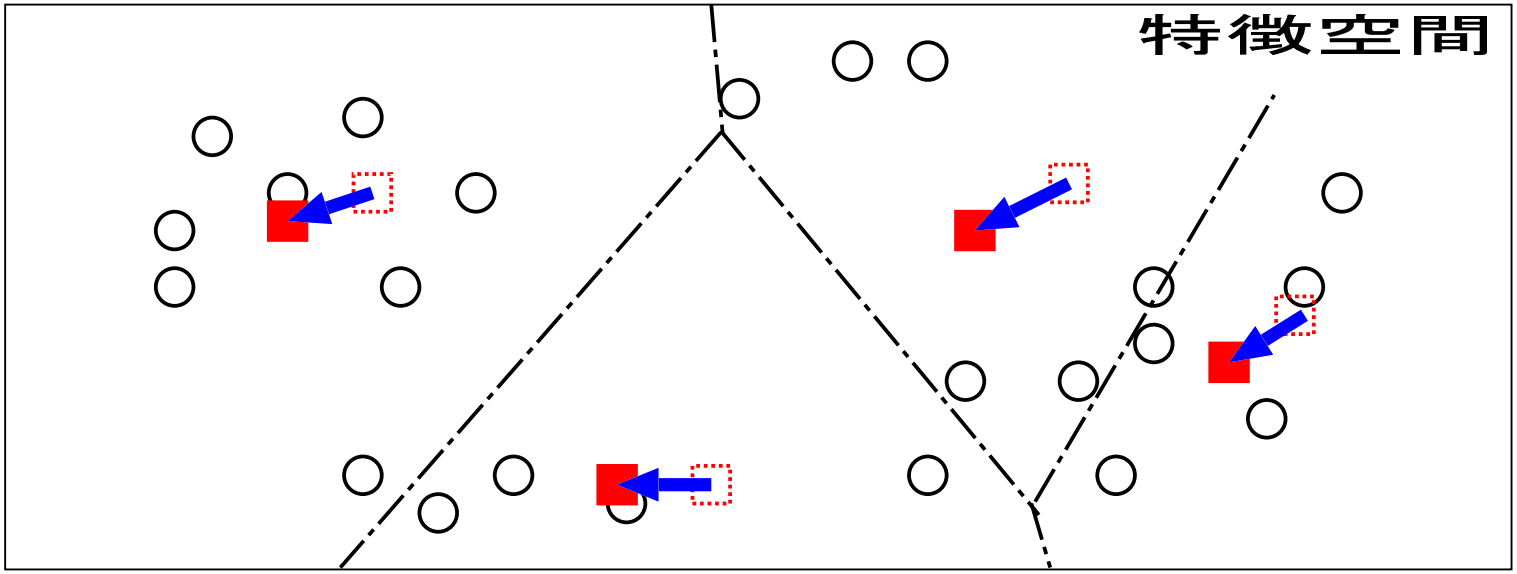
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



k-means アルゴリズム

＜評価基準＞

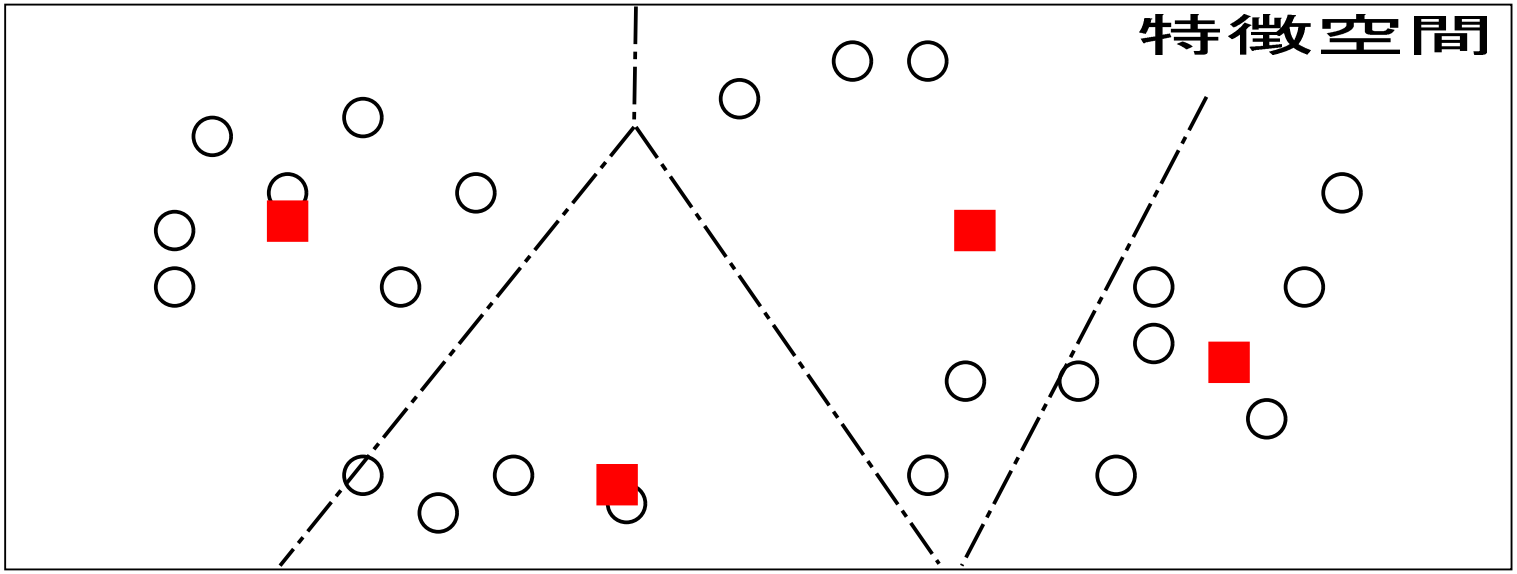
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



k-means アルゴリズム

＜評価基準＞

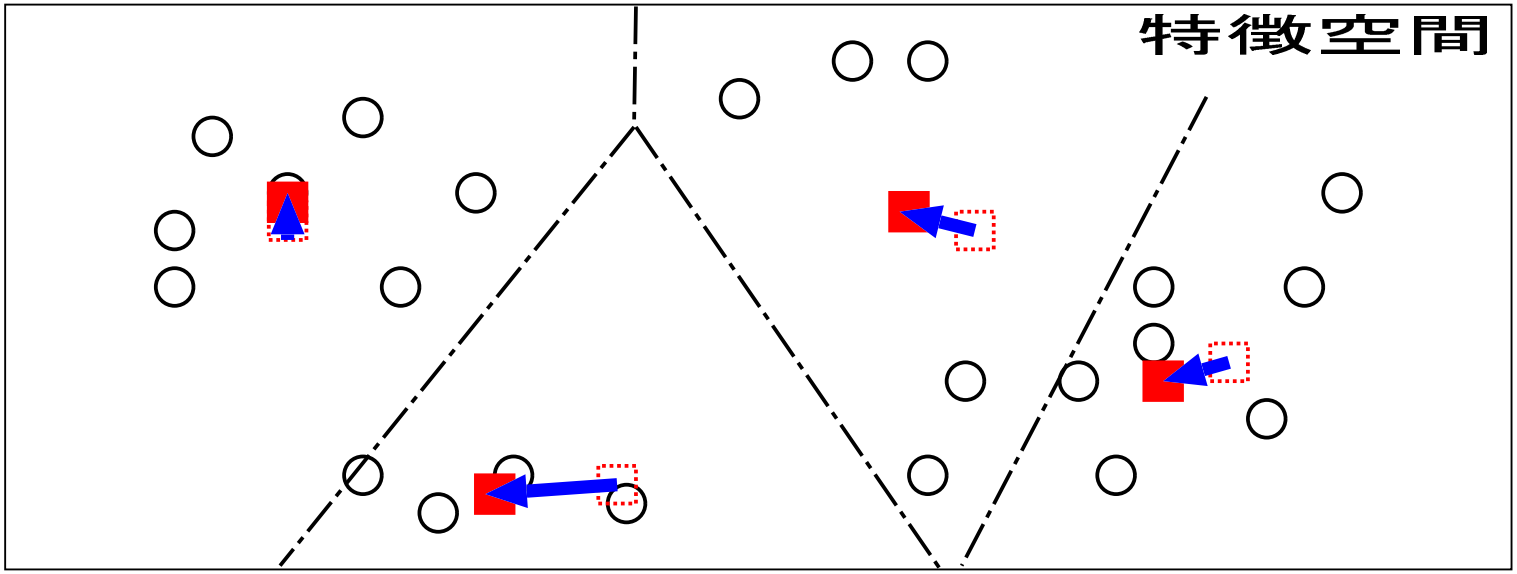
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



k-means アルゴリズム

＜評価基準＞

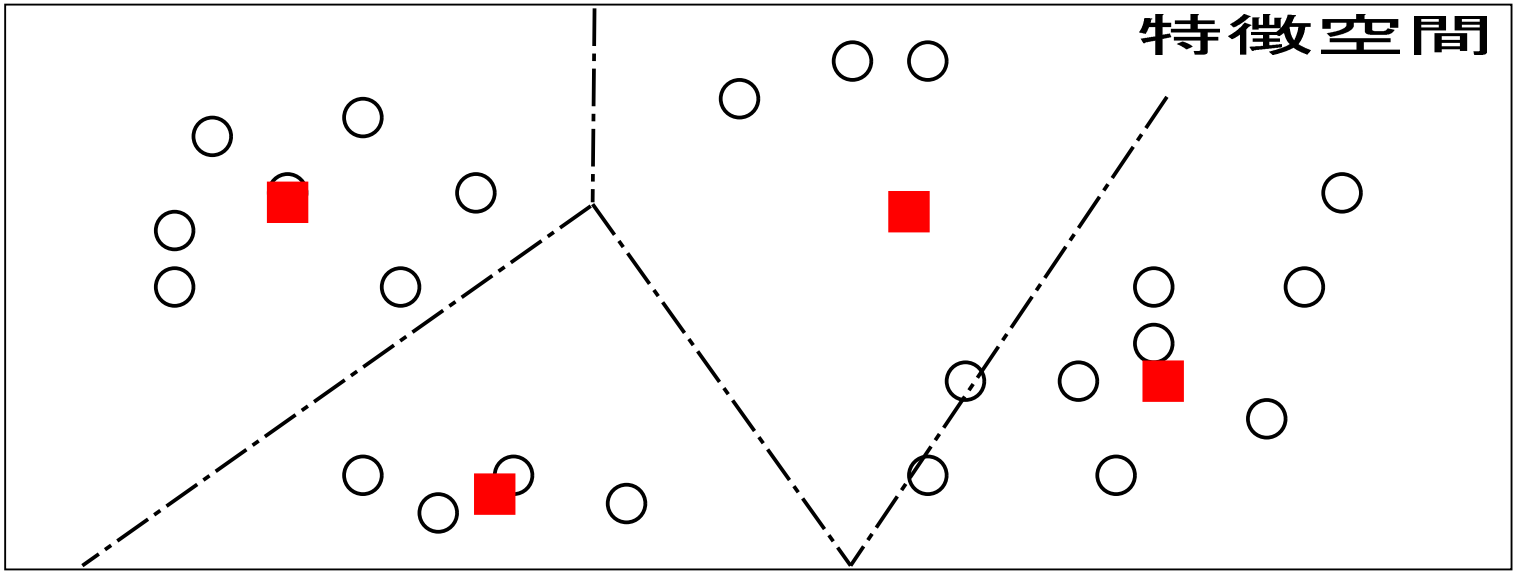
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



k-means アルゴリズム

＜評価基準＞

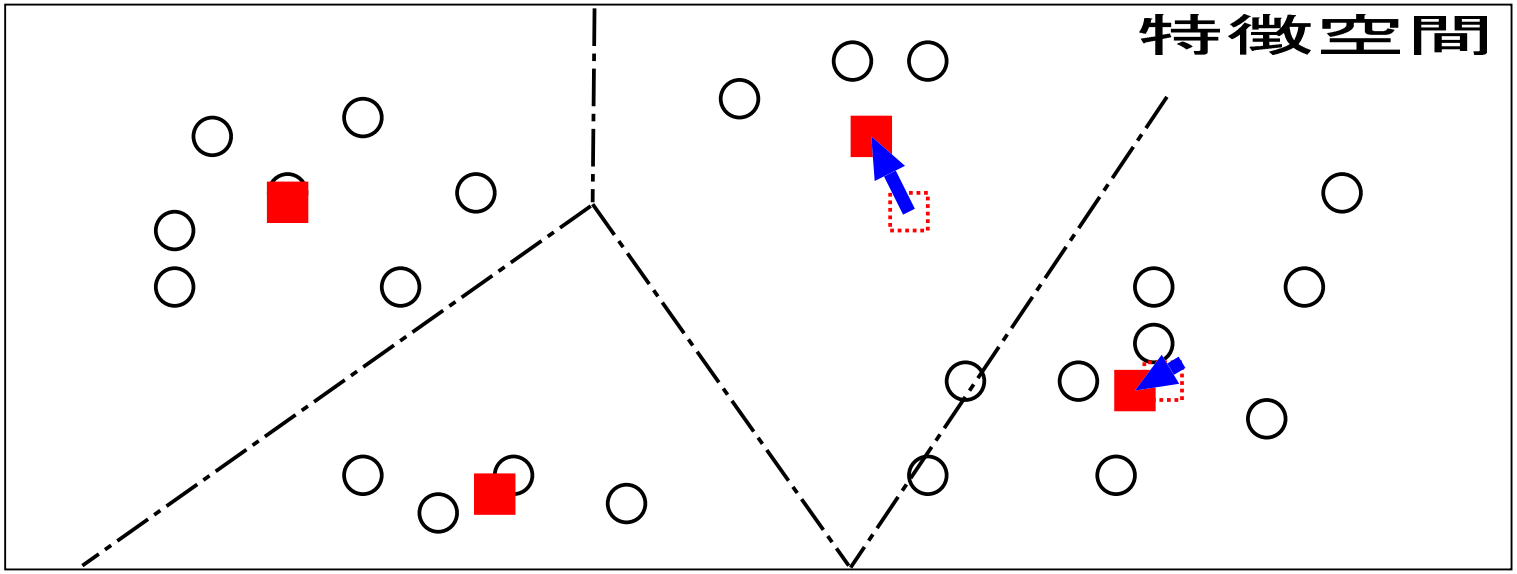
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



k-means アルゴリズム

＜評価基準＞

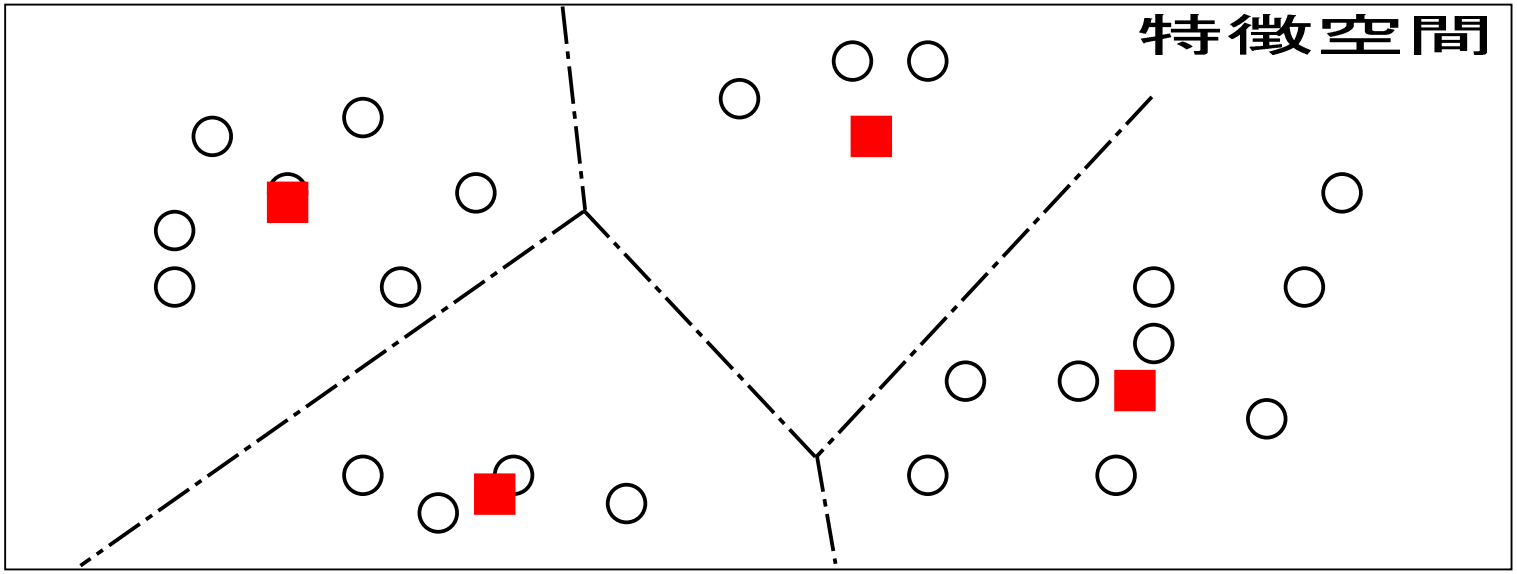
$$J = \sum_{k=1}^{C} \sum_{x_i \in C_k} (\boldsymbol{x}_i - \underbrace{\boldsymbol{\mu}_k}_{\text{クラスタ重心}})^2$$

＜境界の更新＞

各重心間の中点(2次元なら垂直二等分線)を境界とする

＜重心の更新値＞ 評価基準の μ_k についての偏微分が 0 となる μ_k

$$\bar{\mu}_k = \frac{1}{n_k} \sum_{x_i \in C_k} \boldsymbol{x}_i$$



クラスタリングとは

クラスタ決定の「良さ」の基準の最適化

基準の例) $$J = \underbrace{\sum_{i=1}^{C}}_{\text{全クラスタについて}} \underbrace{\sum_{x \in C_i}}_{\text{クラスタ } i \text{ について}} \| \boldsymbol{x} - \underbrace{\boldsymbol{m_i}}_{\text{クラスタ中心}}$$

クラスタ数

◎基準の設け方と最適化方法に応じて，様々なアルゴリズムがある

- 階層的クラスタリング(最近隣法基準)
- k-平均アルゴリズム
- C-平均ファジィクラスタリング法

C-平均ファジィクラスタリング

→ クラスタ重心とクラスタ帰属度を
繰返し計算により更新していく手法

■ 手法の概要

1. クラスタ重心の初期設定
2. クラスタ帰属度を求める
3. クラスタ内のサンプルの重心を求める(クラスタ重心の更新)
4. 収束条件を満たすまで2.と3.を繰り返す

■ 手法の特徴

- 1つのサンプルが複数のクラスタに属することを想定
- クラスタ重心の収束性が保証されている
- クラスタ数を指定する必要がある → サンプルの中にいくつのパターンが内在しているか明らかなときに有効

C-平均ファジィクラスタリング

- **帰属度** p_{ik}
  …サンプル $\boldsymbol{x}_i$ **がクラスタ** C_k **に属する割合**

$$\sum_{k=1}^{C} p_{ik} = 1 \qquad 0 \leq p_{ik} \leq 1$$

- **ファジィネス** P
- **評価基準** J_P

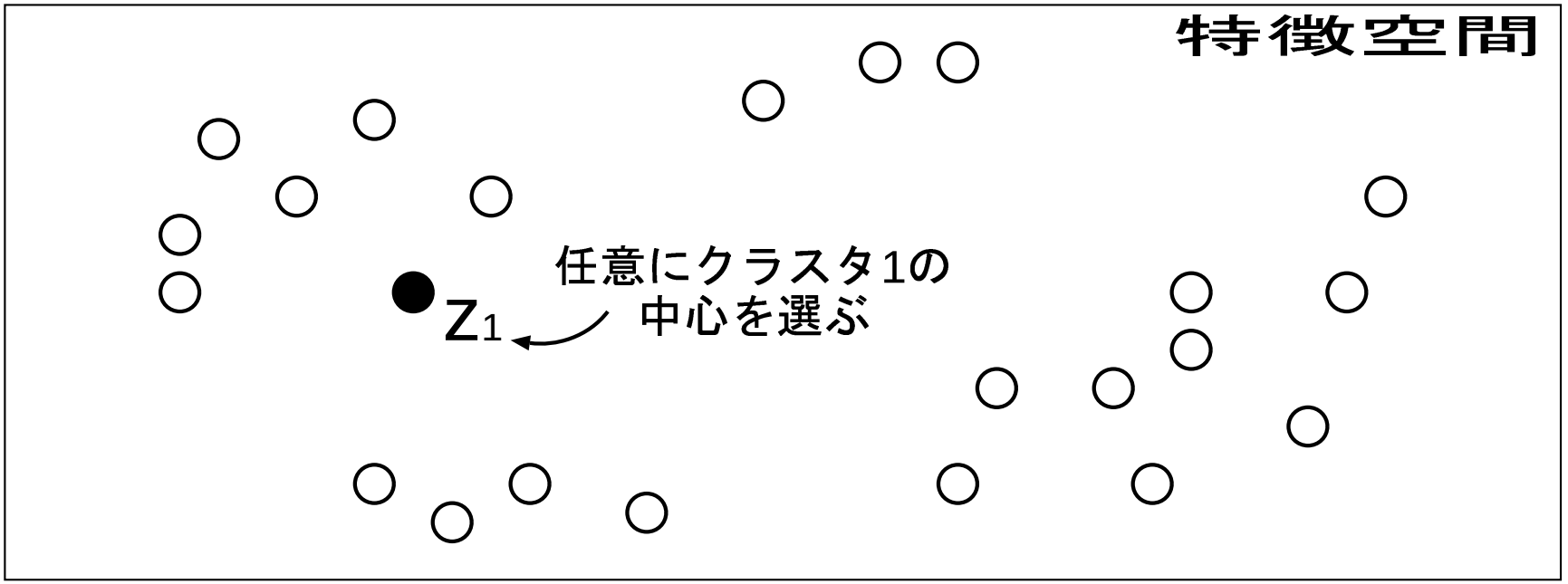
$$J_P = \sum_{i=0}^{n-1} \sum_{k=1}^{C} \underbrace{(p_{ik})^P}_{k\text{-means アルゴリズムと異なるところ}} |\boldsymbol{x}_i - \boldsymbol{\mu}_k|^2$$

- **帰属度とクラスタ重心の更新値**

$$\bar{p}_{ik} = \frac{1}{\sum_{j=1}^{C} \left(\frac{|\boldsymbol{x}_i - \boldsymbol{\mu}_k|}{|\boldsymbol{x}_i - \boldsymbol{\mu}_j|} \right)^{\frac{1}{P-1}}} \qquad \bar{\mu}_k = \frac{\sum_{i=0}^{n-1} (p_{ik})^P \boldsymbol{x}_i}{\sum_{i=0}^{n-1} (p_{ik})^P}$$

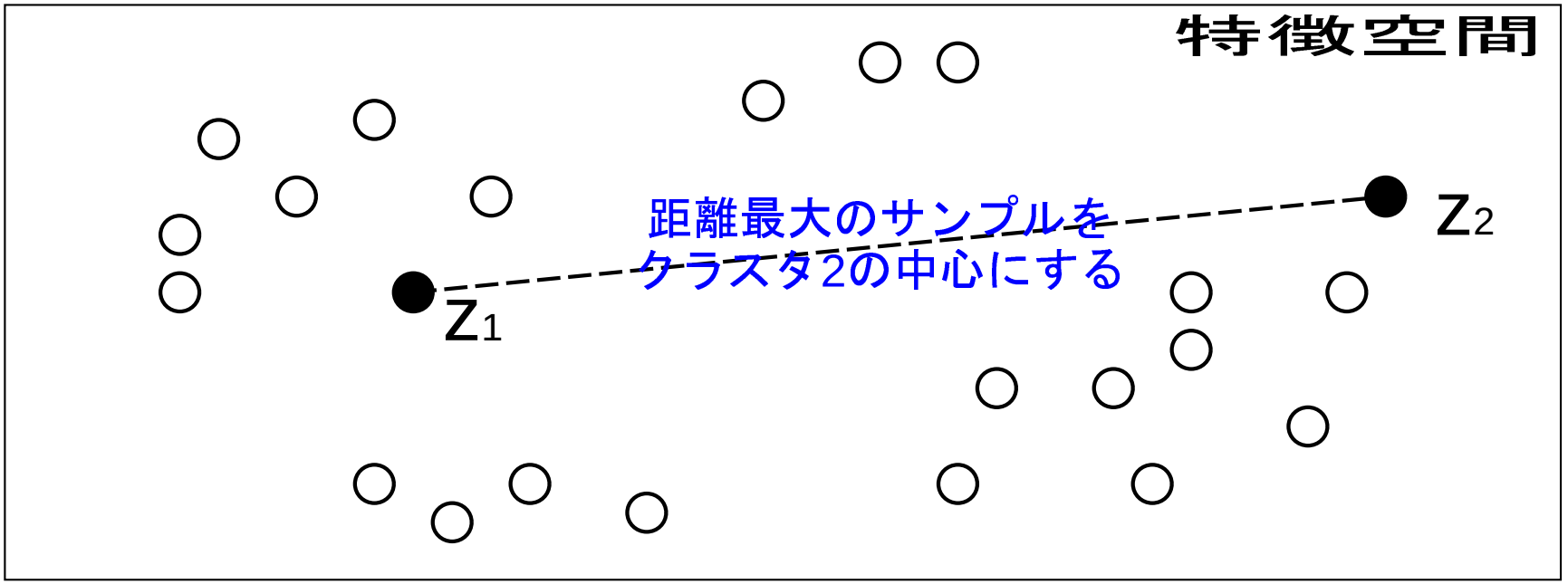
k-meansアルゴリズム&C-平均ファジィクラスタリング

- クラスタ重心の初期設定は??
  - ミニマックス法



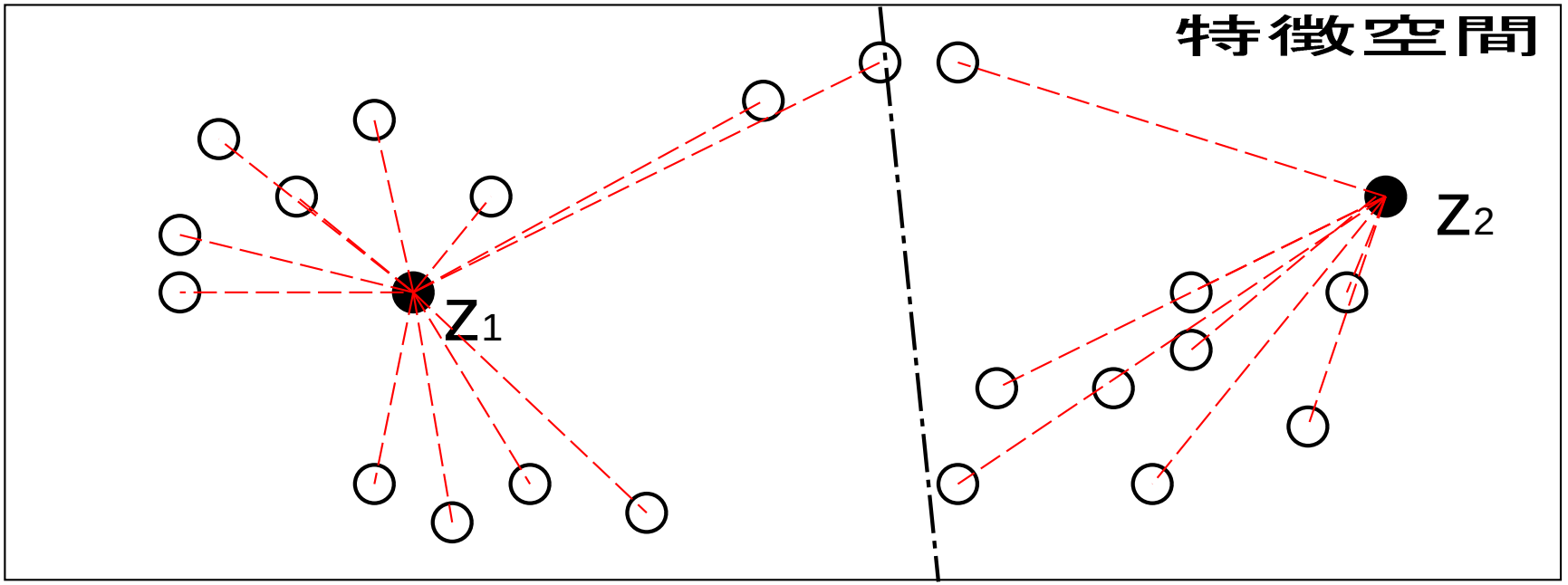
k-meansアルゴリズム&C-平均ファジィクラスタリング

- クラスタ重心の初期設定は??
  - ミニマックス法



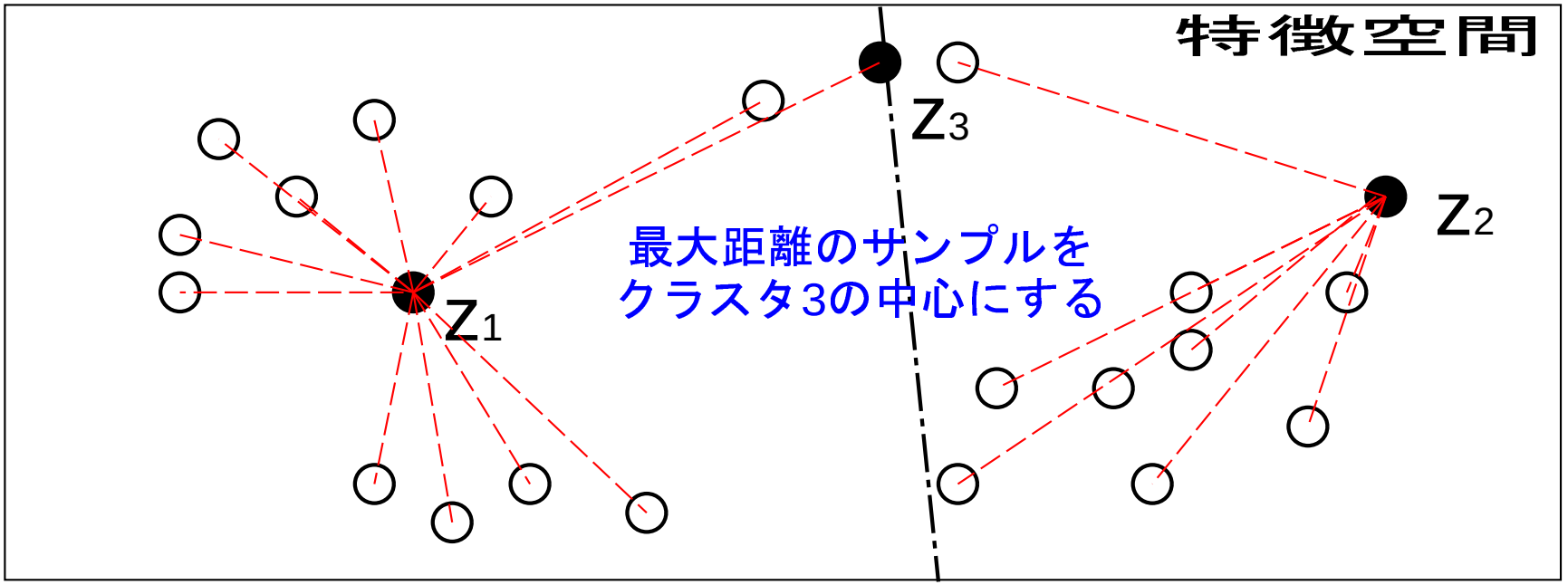
k-means アルゴリズム&C-平均ファジィクラスタリング

- クラスタ重心の初期設定は??
  - ミニマックス法



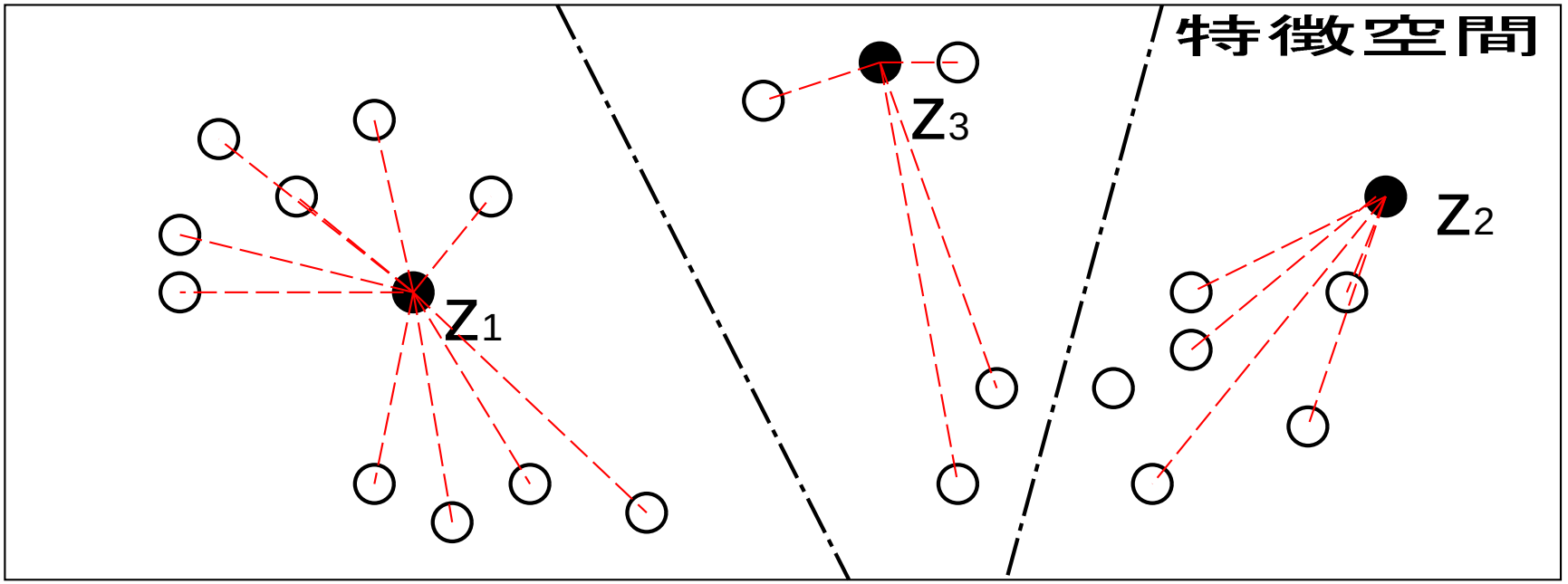
k-means アルゴリズム&C-平均ファジィクラスタリング

- クラスタ重心の初期設定は??
  - ミニマックス法



k-means アルゴリズム&C-平均ファジィクラスタリング

- クラスタ重心の初期設定は??
  - ミニマックス法



k-means アルゴリズム&C-平均ファジィクラスタリング

- クラスタ重心の初期設定は??
  - ミニマックス法