修士学位論文

線形カーネルによる
最大マージンクラスタリングの
安定化と効率化

平成23年度

茨城大学大学院理工学研究科

情報工学専攻

林　華

概要

最大マージンクラスタリング (MMC:Maximum Margin Clustering) は精度の高いクラスタリング手法であり，近年活発に研究されている．そこでは SVM で利用される最大マージンの考え方を基本とする．データを 2 つのクラスタに分割したときに，その 2 つのクラスを識別する SVM を学習し，そのときのサポートベクトルと分離平面までのマージン (最大マージン) を求める．この最大マージンが最大になるような 2 つのクラスタの分割を求める．n 個のデータに対し，2^n パターンの 2 分割が考えられるので， 2^n 回の SVM を実行し，得られた 2^n 個の最大マージンから最大のものに対する分割パターンを出力すれば良い．しかし，当然，これは現実的に計算不可能である．そのため初期の MMC では，ある種の条件を付け，SVM の最大値問題を半正定値問題に変換することで実行していた．ただし，これでも計算コストは高く，様々な提案がなされている．ここでは Valizadegan の手法 (Generalized MMC) を利用する．ただし， GMMC ではメタパラメータの適切な設定が現実的には困難である．パラメータの値によって，最終的なクラスタリングの精度は大きく変化する．また，Valizadegan の GMMC の手法は他の手法に比べて比較的に高速であるが，それでも計算コストは高い．例えば 200 個のデータに対して 200 秒の計算時間が必要になる．しかもデータの数の増加に伴って，計算時間が指数に増大するため，500 個のデータ程度でも実行できない．そこで，本研究では，データに対して中心化を行い，カーネルを線形カーネルに限定する手法を提案する．実験では本手法と Kmeans 法及び CLUTO との比較を行った．その結果，ほとんどのデータセットにおいて，本手法の方が精度が高かった．また本手法の実行時間は 200 個のデータに対し 0.01 秒程度と高速であり，3,200 個のデータでも 2 分程度で実行できた．

Abstract

Maximum Margin Clustering which shows high performance in a number of applications has been actively being researched in recent years. It works based on an idea named maximum margin which is usually used in Support Vector Machines. When given a dataset in a clustering test, MMC makes SVM find out the maximum margin which is a distance between support vectors and hyperplane, and finally obtains the largest one of whole patterns.When given a dataset sized n, it can be thought that 2^n patterns of clustering results exist, and the same number of SVM necessary be run. Obviously, this is not a computable program in practice as you have known. But we can transfer the optimization problem of SVM to semi-definite programming by adding some conditions to make it be a computable program. Unfortunately, SDP still shows rather high computational cost in practice, and several improvements had been proposed. In this paper, we fix eyes on a new approach named Generalized Maximum Margin Clustering by Valizadegan. However, in Valizadegan's paper, precisions of clustering results would be instability according to the value of the parameters. So how to set parameters to different test data is a difficult problem in practice. Additionally, although the result of Valizadegan's proposal showing comparatively low computational cost compared with other methods, it cost almost 200 seconds to cluster just 200 data. Due to runtime increases exponentially in accordance with the increasing of the number of data, it is not able to cluster a dataset larger than 500. To solve this problem, we proposed a new method that centrally normalized every dataset firstly, then applied them just to the linear kernel simply. Concluding from the result of experiments, this method we proposed showed higher performance compared to both Kmeans method and CLUTO on almost dataset. In our experiment, it just took 0.01 second on clustering a dataset sized of 200, and about 2 minutes on a dataset being 3200 examples.

目 次

第1章　はじめに

1.1　研究の背景と目的

クラスタリングとは，内的結合 (internal cohesion) と外的分離 (external isolation) が達成されるようなクラスタと呼ぶ部分集合に，データの集合を分割することである．クラスタリングはデータマイニングの重要なツールとして利用され，大規模データ処理などの新たな要求が生じている．近年，これらの要求に対処する様々な手法が研究されるようになっている．

クラスタリングの中で，教師ありの二値分類であるサポートベクトルマージン (SVM) が非常に精度の高い手法とされる．しかし，SVM を利用する場合，予め訓練データに正解とされるラベルを付けなければいけない．そこで，新たな手法として，最大マージンクラスタリング (MMC) が近年盛んに研究される．

最大マージンクラスタリング問題は SVM を利用し，可能となるクラスタリング結果のすべてのパターンの中から最もマージンが大きいものを求める．しかし，この方法はデータサンプル数 n に対して 2^n 回の SVM を計算する問題になってしまい，現実ではなかなか応用できない状況である．

そこで，最大マージンクラスタリング問題を半正定値問題に変換し，計算コストを下げる様々な方法が提案されてきた．しかし，これらの手法でも計算コストが依然に高く，また，カーネルの選択またはパラメータの設定により，クラスタリング結果が不安定になる問題が存在する．

本研究では，最大マージンクラスタリングにおいて，計算コストを改善する手法を提案した．本研究において，データの中心化を行い，カーネルを線形カーネルに限定することで，比較的に高い精度で高速に問題を解くことができた．

1.2　本論文の構成

第 2 章に主にサポートベクトルマージンと半正定値問題に関する基本知識を述べる．第 3 章に近年代表的な最大マージンクラスタリングの手法について少し説明し，それらの計算コストの比較を行う．そして第 4 章では本研究における提案を述べる．第 5 章に提案に基づく実験とその結果を説明する．第 6 章に本研究においての問題点，改善点また今後の課題について述べる．

第2章　サポートベクトルマシーン

サポートベクターマシン (SVM; support vector machine)[1, 2, 3] は、分類と回帰問題を主としたデータ解析方法で，広く知られるようになったのは 1990 年代の中頃であり、Vapnik の貢献が高く評価されている．SVM を利用する前に訓練データを作成する必要がある．SVM が訓練データに対し，クラス間のマージンが最大になるようにモデルを学習しておく．そして新たなテストデータに対して，モデルを使い，各サンプルがどのクラスに属するかを予測する．また，一般的に高次元の場合 SVM が有効とされる．次の図 2.1 は SVM のイメージである．

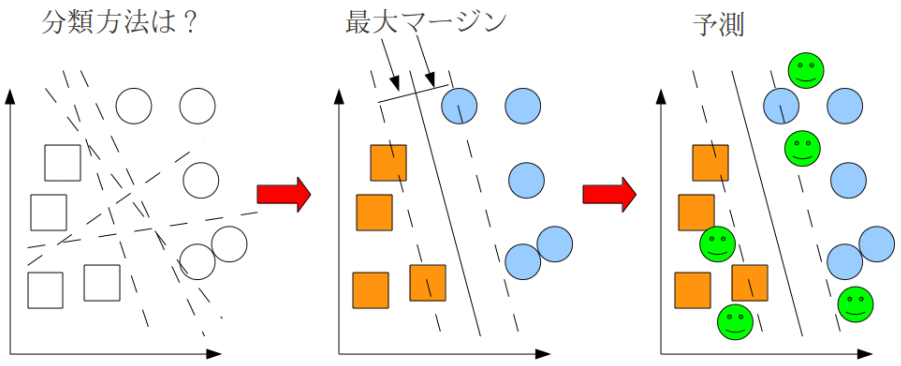

図 2.1: SVM

2.1　学習

学習 (厳密には教師付き学習—supervised learning) を数理的に扱うために，以下のような状況を設定する．学習/トレーニングの対象となるデータ群が与えられたときに，それらの情報から何らかの知見を得て，未知のデータに対する判断を行うことを目的とする．簡単のために，二つのクラスに分けられたデータ群の学習から，未知のデータをどちらかのクラスに分類する作業を考える．(分類クラス数の一般化も可能である．)

トレーニングデータの集合を

$$S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_\ell, y_\ell)\},\ \mathbf{x}_i \in \mathbb{R}^N, y_i \in \{-1, 1\}$$

と表す．すなわち，それぞれのデータは N 個の成分 (入力) とクラス分けのための指標 $\{-1, 1\}$(出力) から成っている．このデータはある (未知の) 分布 $P(\mathbf{x}, y)$ によって生成されていると仮

定する．上で述べたように，学習とはこれらのデータから識別関数 $f : \mathbb{R}^N \to \{-1, 1\}$ を選びトレーニングデータに入っていないデータ $(\mathbf{x}, y)$(通常，検証用データと言う) を正しく $(f(\mathbf{x}) = y)$ 推定することを目的とする．与えられたトレーニングデータに対して正しい答えを出す $(f(\mathbf{x}_i) = y_i, i = 1, ..., \ell)$ 関数は無数に存在するし，それらが検証用データに対して異なった答えを出すこともあり得る．したがって，トレーニングデータを正しく識別するという基準だけでは未知のデータに対して正しい答えを与えるという本来の目的 (汎化能力ー generalization) に沿った関数を選ぶことは出来ない．何らかの基準によって候補となる関数を制限しなくてはならない．

候補とする識別関数の集合を H とする．期待リスク (expected risk)：

$$R(f) = \int \frac{1}{2}|f(\mathbf{x}) - y|P(\mathbf{x}, y)d\mathbf{x}dy$$

を最小にする関数 $f \in H$ を求めることが究極の目的である．しかし，分布 $P(\mathbf{x}, y)$ は未知なので期待リスク $R(f)$ を計算することは不可能である．そこで，我々は経験的リスク (empirical risk)：

$$R_{emp}(f) = \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{1}{2}|f(\mathbf{x}_i) - y_i|$$

を扱わざるを得ない．上で述べたように，単純な $R_{emp}(f)$ の最小化 (経験的リスク最小化の原理-Empirical Risk Minimization Principle) では自由度があり過ぎるので何らかの基準を必要とする．Vapnik 達による統計的学習理論によると，トレーニングデータの数を増やしていったときに Remp(f) の最小化が R(f) の最小化と矛盾しないためには $R_{emp}(f)$ が $R(f)$ に一様収束することが必要十分条件となる．そして，この条件は集合 H の VC(Vapnik and Chervonekis)-次元 h の有界性と同値となる．集合 H の VC-次元とは，この集合の複雑性/多様性 (capacity) を表すもので,「集合に属する関数によってあらゆる可能な分離が出来る最大の点の数」のことである．

VC 次元によって $R(f)$ と $R_{emp}(f)$ の誤差の評価を行うことができる．たとえば，$h < \ell$ ならば，確率 $1 - \delta$ で

$$R(f) \leq R_{emp}(f) + \sqrt{\frac{h(log\frac{2\ell}{h} + 1) - log\frac{\delta}{4}}{\ell}}, \forall f \in H$$

が成立する．したがって，期待リスクを小さくするためには上の不等式の右辺の 2 つの項を小さくすることが考えられる．すなわち，$R_{emp}(f)$ と h/ℓ を小さくすれば良い．一般に，$R_{emp}(f)$ の値は h(関数の自由度) が大きい程小さくなるので，上記右辺の和を最小にするような VC 次元の最適値が存在するであろう．

Vapnik による構造的リスク最小化 (Structural Risk Minimization) の原理は VC 次元 h_n が単調増加するような関数集合の列

$$H_1 \subset H_2 \subset ... \subset H_n \subset ...$$

を考えて，上記不等式の右辺，あるいは簡単化して $R_{emp}(f) + \sqrt{h_n/\ell}$ を最小化するような H_n と $f \in H_n$ を求めようとするものである．厳密にこの操作を実行するのは困難であるが，サポートベクターマシンはある意味でこの原理をなぞっているものと解釈できる．

2.2 超平面による識別

トレーニングデータ集合 S が $\mathbb{R}^N$ の超平面によって $y_i = 1$ のグループと $y_i = -1$ のグループに分離される場合を線形分離可能 (linearly separable) と言う．候補となる超平面を

$$(\mathbf{w} \cdot \mathbf{x}) + b = 0 \tag{2.1}$$

と表す．ここで $\mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}$ で，$(\mathbf{w} \cdot \mathbf{x})$ は $\mathbf{w}$ と $\mathbf{x}$ の内積を表す．そして，識別関数を

$$f(\mathbf{x}) = sgn((\mathbf{w} \cdot \mathbf{x}) + b) \tag{2.2}$$

とする．($sgn(0) = 1$ と約束する．) 一般にトレーニングデータを分離する超平面は一意には決まらないことに注意する．

超平面 $(\mathbf{w}, b)$ に対するサンプル $(\mathbf{x}_i, y_i)$ のマージンは

$$\gamma_i = \frac{y_i((\mathbf{w} \cdot \mathbf{x}_i) + b)}{||\mathbf{w}||} \tag{2.3}$$

と定義される．($||\cdot||$ は 2 ノルムを表す．)$\gamma_i > 0$ ならば $(\mathbf{x}_i, y_i)$ は正しく識別されていることになる．点 $\mathbf{x} \in \mathbb{R}_N$ を超平面へ射影した点を $\mathbf{x} \in \mathbb{R}_N$ とすると，

$$||\mathbf{w}||\,||\mathbf{x} - \hat{\mathbf{x}}|| = |(\mathbf{w} \cdot (\mathbf{x} - \hat{\mathbf{x}})| = |(\mathbf{w} \cdot \mathbf{x} + b) - (\mathbf{w} \cdot \hat{\mathbf{x}} + b)| = |(\mathbf{w} \cdot \mathbf{x} + b)|$$

となるから，$\gamma_i > 0$ ならば γ_i は点 $\mathbf{x}_i$ から超平面 $(\mathbf{w}, b)$ への距離を表す．

上記マージン $\gamma_i, i = 1, ...\ell$ の最小値をサンプルデータ集合 S に対する超平面 $(\mathbf{w}, b)$ のマージンと言う．そして，すべての超平面のマージンの最大値をこのデータ集合 S のマージンと呼ぶ．線形分離可能なデータ集合に対してはマージンは正となる．

2.3 線形分離可能な場合

以下では，線形分離可能なデータ集合を考える．この場合，超平面 $(\mathbf{w}, b)$ が存在して，

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) > 0, i = 1, ..., \ell$$

となるから，超平面を正規化して

$$\min_t \{y_i((\mathbf{w} \cdot \mathbf{x}_i) + b)\} = 1$$

とすることが出来る．この性質をみたす形式を正準形 (canonical form) と言う．正準形式の超平面のマージンは $1/||\mathbf{w}||$ となる．マージンが最大になるような超平面を最適な超平面 (optimal hyperplane) と考えると，それは

$$\begin{array}{lll} \text{最小化} & \frac{1}{2}||\mathbf{w}||^2, & \mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R} \\ \text{条　件} & y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, & i = 1, ..., \ell \end{array} \tag{2.4}$$

の解によって与えられる．

この問題 (2.4) は凸 2 次計画問題であり，問題の構造も特に複雑ではないが，以下で非線形の (曲面による) 識別に拡張するために双対問題を考えると都合が良い．ラグランジュ関数を $L(\mathbf{w}, b, \boldsymbol{\alpha})$ とする．ここで，$\boldsymbol{\alpha} \in \mathbb{R}$ は双対変数である．Wolfe の双対問題は

$$\begin{array}{ll} \text{最大化} & L(\mathbf{w}, b, \boldsymbol{\alpha}), \qquad\qquad\qquad\qquad \mathbf{w} \in \mathbb{R}_N, b \in \mathbb{R}, \boldsymbol{\alpha} \in \mathbb{R} \\ \text{条　件} & \nabla_{\mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \nabla_b L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \boldsymbol{\alpha} \geq 0 \end{array} \tag{2.5}$$

となる．具体的には

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{\ell} \alpha_i (y_i((\mathbf{w} \cdot \mathbf{x}_i) + b - 1)$$

であるから，最適性の必要十分条件である Karush-Kuhn-Tucker(KKT) 条件は

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i = 0, \tag{2.6}$$

$$\nabla_b L(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i y_i = 0, \tag{2.7}$$

$$\alpha_i (y_i((\mathbf{w} \cdot \mathbf{x}_i) + b - 1) = 0, \boldsymbol{\alpha} \geq 0, y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 \geq 0, \ i = 1, ..., \ell \tag{2.8}$$

これらの条件を考慮すると，双対問題は

$$\begin{array}{ll} \text{最大化} & -\frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_i^{\ell} \alpha_i, \quad \boldsymbol{\alpha} \in \mathbb{R}^{\ell} \\ \text{条　件} & \sum_{i=1}^{\ell} \alpha_i y_i = 0, \qquad\qquad\qquad\qquad \boldsymbol{\alpha} \geq 0 \end{array} \tag{2.9}$$

となる．双対問題のヘッセ行列を $-D \in \mathbb{R}^{\ell \times \ell}$, $D_{ij} = y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$ と置くと行列 D は対称非負定値行列となり，上記問題は

$$\begin{array}{ll} \text{最大化} & -\frac{1}{2}(\boldsymbol{\alpha} \cdot D\boldsymbol{\alpha}) + (e \cdot \boldsymbol{\alpha}), \quad \boldsymbol{\alpha} \in \mathbb{R}^{\ell} \\ \text{条　件} & (y \cdot \boldsymbol{\alpha}) = 0, \qquad\qquad\qquad\qquad \boldsymbol{\alpha} \geq 0 \end{array} \tag{2.10}$$

と書ける．ここで，$e = (1, ..., 1)^t \in \mathbb{R}$ である．最適解を $(\mathbf{w}^*, b^*, \boldsymbol{\alpha}^*)$ とすると，双対変数 $\alpha_i^*, i = 1, ..., \ell$ の中で非零のものは不等式制約条件が有効となっているもの，すなわち

$$y_i((\mathbf{w}^* \cdot \mathbf{x}_i) + b) = 1 \tag{2.11}$$

となるものなので，数が少ない場合が多い．(2.6) より，最適解を与える $\mathbf{w}^*$ は非零の α_i^* に対応するサンプルデータ $\mathbf{x}_i$ の 1 次結合によって表される．そのようなデータをサポートベクターと呼ぶ．サポートベクターの集合を $S_V \subset S$ と表わすと，

$$\mathbf{w}^* = \sum_{t \in S_V} \alpha_i^* y_i \mathbf{x}_i \tag{2.12}$$

である．双対変数のほんの一部のみが非零となる (有効制約が少数である) ことはスパース性 (sparseness) と呼ばれている．式 (2.11) より，b^* も任意のサポートベクター $\mathbf{x}_i$ によって

$$b^* = y_i - (\mathbf{w}^* \cdot \mathbf{x}_i) = y_i - \sum_{j \in S_V} \alpha_j^* y_j (\mathbf{x}_j \cdot \mathbf{x}_i) \tag{2.13}$$

と表される．識別関数は

$$f(\mathbf{x}) = sgn\left(\sum_{i\in S_V} \alpha_i^* y_i(\mathbf{x}\cdot\mathbf{x}_i) + b^*\right) \tag{2.14}$$

となる．また，b^* の表式より

$$||\mathbf{w}^*||^2 = \sum_{i\in S_V} \alpha_i^* y_i(\mathbf{x}_i\cdot\mathbf{w}^*) = \sum_{i\in S_V} \alpha_i^*(1-y_i b^*) = \sum_{i\in S_V} \alpha_i^* \tag{2.15}$$

となるので，データ集合 S のマージン γ は

$$\gamma = \frac{1}{||\mathbf{w}^*||} = \left(\sum_{i\in S_V} \alpha_i^*\right)^{-1/2} \tag{2.16}$$

で与えられる．

2.4 線形分離が不可能な場合

次に，線形分離が不可能 (linearly non-separable) な場合，すなわちデータ集合 S のマージンが正にならない場合を考える．この場合は元の問題の条件：

$$\text{条　件}\quad y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq 1,\quad i=1,...,\ell$$

にスラック変数を導入して

$$\text{条　件}\quad y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq 1-\xi_i, \xi_i \geq 0\ i=1,...,\ell$$

と変換して，制約条件がすべてみたされない場合に対応する．このような形式の制約条件を扱うものをソフトマージンによる最適化と言う．スラック変数の値はなるべく小さくしたいので目的関数にペナルティを課すのは最適化の常套手段であろう．ペナルティ項の形の代表的なものを二つあげておく．

2.4.1 1 ノルムによるソフトマージン最適化

解くべき問題を

$$\begin{array}{ll} \text{最小化} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{\ell}\xi_i, \qquad \mathbf{w}\in\mathbb{R}^N, b\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^\ell \\ \text{条　件} & y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq 1-\xi_i,\quad \xi_i\geq 0,\quad i=1,...,\ell \end{array} \tag{2.17}$$

とする．ここで，$C>0$ はペナルティパラメータである．ラグランジュ関数を

$$L(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\alpha_i(y_i((\mathbf{w}\cdot\mathbf{x}_i)+b)-1+\xi_i) - \sum_{i=1}^{\ell}\beta_i\xi_i$$

とする．ここで，$\boldsymbol{\alpha} \in \mathbb{R}^\ell$ と $\boldsymbol{\beta} \in \mathbb{R}^\ell$ は双対変数である．線形分離可能な場合と同様に，KKT 条件は

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbf{w} - \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i = 0, \tag{2.18}$$

$$\nabla_b L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^{\ell} \alpha_i y_i = 0, \tag{2.19}$$

$$\nabla_{\boldsymbol{\xi}} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = Ce - \boldsymbol{\alpha} - \boldsymbol{\beta} = 0 \tag{2.20}$$

$$\alpha_i(y_i((\mathbf{w} \cdot \mathbf{x}_i) + b - 1 + \xi_i) = 0, \boldsymbol{\alpha} \geq 0, y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 + \xi_i \geq 0,\ i = 1, ..., \ell \tag{2.21}$$

$$\beta_i \xi_i = 0, \beta_i \geq 0, \xi_i \geq 0, i = 1, ..., \ell \tag{2.22}$$

となる．(2.20) から得られる $\boldsymbol{\beta} = Ce - \boldsymbol{\alpha}$ を (2.22) に代入すると

$$(C - \alpha_i)\xi_i = 0,\ \alpha_i \leq C, \xi_i \geq 0, i = 1, ..., \ell \tag{2.23}$$

を得る．結局，双対問題から $\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\beta}$ が消去できて

$$\begin{array}{lll} \text{最大化} & -\frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{i=1}^{\ell} \alpha_i, & \boldsymbol{\alpha} \in \mathbb{R}^\ell \\ \text{条　件} & \sum_{i=1}^{\ell} \alpha_i y_i = 0, & 0 \leq \boldsymbol{\alpha} \leq Ce \end{array} \tag{2.24}$$

を得る．分離可能の場合と異なるのは，変数に上限が付加されることである．

- 双対変数 $\alpha_i^* > 0$ に対応するのは，不等式制約条件が有効となっているもの，すなわち

  $$y_i((\mathbf{w}^* \cdot \mathbf{x}_i) + b^*) = 1 - \xi_i^*$$

  であり，さらに $\alpha_i^* < C$ ならば $\xi_i^* = 0$ となる．したがって，$0 < \alpha_i^* < C$ となるデータが正しく識別されたサポートベクターとなる：

  $$0 < \alpha_i^* < C \Rightarrow y_i((\mathbf{w}^* \cdot \mathbf{x}_i) + b^*) = 1$$

- $\alpha_i^* = 0$ の場合は，(2.23) より $\xi_i = 0$ となるので，

  $$\alpha_i^* = 0 \Rightarrow y_i((\mathbf{w}^* \cdot \mathbf{x}_i) + b^*) \geq 1$$

- $\alpha_i^* = C$ の場合は，

  $$y_i((\mathbf{w}^* \cdot \mathbf{x}_i) + b^*) - 1 + \xi_i^* = 0$$

  かつ $\xi_i \geq 0$ となるので，

  $$\alpha_i^* = C \Rightarrow y_i((\mathbf{w}^* \cdot \mathbf{x}_i) + b^*) \leq 1$$

  上記より，正しく識別されたサポートベクター $\mathbf{x}_i$ が存在するとき b^* は

  $$b^* = y_i - (\mathbf{w}^* \cdot \mathbf{x}_i) = y_i - \sum_{j \in S_V} \alpha_j^* y_j (\mathbf{x}_j \cdot \mathbf{x}_i)$$

  と表される．

2.4.2　2 ノルムによるソフトマージン最適化

2 ノルムのペナルティを利用する場合には，問題は

$$\begin{array}{ll} \text{最小化} & \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_{i=1}^{\ell}\xi_i^2, \qquad \mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}^\ell \\ \text{条　件} & y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq 1-\xi_i, \quad i = 1, ..., \ell \end{array} \tag{2.25}$$

となる．ここで，ξ_i に対する非負条件は不必要なので除去されている．何故なら，$\xi_i < 0$ の点では $y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq 1$ の制約条件はみたされていて，$\mathbf{w}, b$ の値を固定したまま $\xi_i = 0$ とした方が目的関数の値はより小さくなる．したがって，$\xi_i < 0$ が最適解として得られることは無いからである．ラグランジュ関数は

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_{i=1}^{\ell}\xi_i^2 - \sum_{i=1}^{\ell}\alpha_i(y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) - 1 + \xi_i)$$

となる．KKT 条件は

$$\nabla_w L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^{\ell}\alpha_i y_i \mathbf{x}_i = 0, \tag{2.26}$$

$$\nabla_b L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \sum_{i=1}^{\ell}\alpha_i y_i = 0, \tag{2.27}$$

$$\nabla_{\boldsymbol{\xi}} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = C\boldsymbol{\xi} - \boldsymbol{\alpha} = 0 \tag{2.28}$$

$$\alpha_i(y_i((\mathbf{w}\cdot\mathbf{x}_i)+b-1+\xi_i) = 0, \boldsymbol{\alpha} \geq 0, y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) - 1 + \xi_i \geq 0, i = 1, ..., \ell \tag{2.29}$$

となる．上式より，双対問題から $\mathbf{w}, b, \boldsymbol{\xi}$ が消去できて

$$\begin{array}{ll} \text{最大化} & -\frac{1}{2}\sum_{i,j=1}^{\ell}\alpha_i\alpha_j y_i y_j((\mathbf{x}_i\cdot\mathbf{x}_j) + \frac{1}{C}\delta_{ij}) + \sum_{i=1}^{\ell}\alpha_i, \quad \boldsymbol{\alpha} \in \mathbb{R}^\ell \\ \text{条　件} & \sum_{i=1}^{\ell}\alpha_i y_i = 0, \qquad\qquad\qquad\qquad\qquad\qquad\quad \boldsymbol{\alpha} \geq 0 \end{array} \tag{2.30}$$

を得る．ここで，δ_{ij} は Kronecker のデルタ記号である．1 ノルムの場合と異なるのは，変数 $\boldsymbol{\alpha}$ に関する上限制約が存在しないことと，ヘッセ行列の対角項に $1/C$ という数が引かれていることである．したがって，この場合，ヘッセ行列は負定値となり，数値計算上はより好条件となる．最適解では，

- $\alpha_i^* = 0$ ならば $\xi_i^* = 0$ となり，

$$\alpha_i^* = 0 \Rightarrow y_i((\mathbf{w}^*\cdot\mathbf{x}_i)+b^*) \geq 1$$

- $\alpha_i^* > 0$ ならば $\xi_i^* = \alpha_i^* > 0$ なので，

$$\alpha_i^* > 0 \Rightarrow y_i((\mathbf{w}^*\cdot\mathbf{x}_i)+b^*) < 1$$

すべての場合に平面による識別が適切とは限らない．そこで，より複雑な識別に対応するために曲面による分離を考える．このような場合に，高次元の空間への非線形変換とその空間でのカーネルトリックと言われる方法が知られている．まず，入力データを

より高次元な空間に (非線形) 射像する．すなわち，b^* は任意のサポートベクター $\mathbf{x}_i$ によって

$$b^* = y_i(1 - \alpha_i^*/C) - (\mathbf{w}^* \cdot \mathbf{x}_i) = y_i(1 - \alpha_i^*/C) - \sum_{j \in S_V} \alpha_j^* y_j (\mathbf{x}_j \cdot \mathbf{x}_i)$$

と計算される．

2.5 曲面による識別

すべての場合に平面による識別が適切とは限らない．そこで，より複雑な識別に対応するために曲面による分離を考える．このような場合に，高次元の空間への非線形変換とその空間でのカーネルトリックと言われる方法が知られている．まず，入力データをより高次元な空間に (非線形) 射像する．すなわち，

$$\mathbf{x} \mapsto \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ...)$$

と対応付けられる入力データ空間 $\mathbf{x} \subset \mathbb{R}^N$ から特徴空間 (feature space)$F = \{\phi(\mathbf{x}) | \mathbf{x} \in \mathbf{X}\}$ への射像を考える．そして，特徴空間において線形分離を考える．その際に，双対問題で表された最適化問題において，特徴空間に対応した量はすべて内積の形式でのみ現れることに注意する．すなわち，$(\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$, $\mathbf{x}, \mathbf{y} \in \mathbf{X}$ の形式である．高次元空間において内積を文字通り計算するのは現実的ではないので，対応する項をカーネル関数で置き換える：

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$$

このとき，識別関数は

$$f(\mathbf{x}) = sgn\left(\sum_{i \in S_V} \alpha_i^* y_i K(\mathbf{x}, \mathbf{x}_i) + b^*\right) \tag{2.31}$$

となる．

1 ノルムソフトマージン最適化問題は

$$\begin{array}{lll} \text{最大化} & -\frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{i=1}^{\ell} \alpha_i, & \boldsymbol{\alpha} \in \mathbb{R}^{\ell} \\ \text{条　件} & \sum_{i=1}^{\ell} \alpha_i y_i = 0, & 0 \leq \boldsymbol{\alpha} \leq Ce \end{array} \tag{2.32}$$

となり，

$$b^* = y_i - \sum_{j \in S_V} \alpha_j^* y_j K(\mathbf{x}_j, \mathbf{x}_i)$$

また，2 ノルムソフトマージン最適化問題は

$$\begin{array}{lll} \text{最大化} & -\frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (K(\mathbf{x}_i \cdot \mathbf{x}_j) + \frac{1}{C}\delta_{ij}) + \sum_{i=1}^{\ell} \alpha_i, & \boldsymbol{\alpha} \in \mathbb{R}^{\ell} \\ \text{条　件} & \sum_{i=1}^{\ell} \alpha_i y_i = 0, & \boldsymbol{\alpha} \geq 0 \end{array} \tag{2.33}$$

となり，

$$b^* = y_i(1 - \alpha_i^*/C) - \sum_{j \in S_V} \alpha_j^* y_j K(\mathbf{x}_j, \mathbf{x}_i)$$

任意に与えた関数 $K(\mathbf{x},\mathbf{y})$ をカーネル関数として使えるわけではない．それは，内積の形で表せる必要があるが，その保証を与えるのが，関数解析で古くから知られている Mercer の定理である．

定理 $(Mercer)$ $\mathbf{x}$ は $\mathbb{R}^N$ の有界閉集合で，関数 $K:\mathbf{X}\times\mathbf{X}\to\mathbb{R}$ は連続かつ対称 $(K(\mathbf{x},\mathbf{y})=K(\mathbf{y},\mathbf{x}))$ とする．このとき，関数 K が，一様収束する級数：

$$K(\mathbf{x},\mathbf{y})=\sum_{j=1}^{\infty}\alpha_j\phi_j(\mathbf{x})\phi_j(\mathbf{y}),\quad \alpha_j>0 \tag{2.34}$$

によって展開可能となる必要十分条件は

$$\int_{\mathbf{X}\times\mathbf{X}}K(\mathbf{x},\mathbf{y})f(\mathbf{x})f(\mathbf{y})d\mathbf{x}d\mathbf{y}\geq 0,\quad \forall f\in L_2(\mathbf{x}) \tag{2.35}$$

この定理の正定符号カーネルの条件 (2.35) は

$$\sum_{i,j=1}^{n}K(\mathbf{x}_i,\mathbf{x}_j)v_iv_j\geq 0,\quad \forall\mathbf{x}_i,\mathbf{x}_j\in\mathbf{x},\quad \forall v_i,v_j\in\mathbb{R} \tag{2.36}$$

と表すことができる．すなわち，行列 $\{K(\mathbf{x}_i,\mathbf{x}_j),i,j=1,...,n\}$ が非負定値という条件である．

Mercer の条件をみたす具体的なカーネル関数の形としてよく使用されているものをあげておく．

- radial basis function(RBF) カーネル　$K(\mathbf{x},\mathbf{y})=\exp(-||\mathbf{x}-\mathbf{y}||/(2\sigma^2))$
- d 次の多項式カーネル　$K(\mathbf{x},\mathbf{y})=(1+\mathbf{x}\cdot\mathbf{y})^d$
- シグモイドカーネル　$K(\mathbf{x},\mathbf{y})=tanh(k(\mathbf{x}\cdot\mathbf{y})-\theta)$

第3章　最大マージンクラスタリング

3.1　最大マージンクラスタリングとは

クラスタリング [5] とは，内的結合 (internal cohesion) と外的分離 (external isolation) が達成されるようなクラスタと呼ぶ部分集合に，データの集合を分割することである．クラスタリングはデータマイニングの重要なツールとして利用され，大規模データ処理などの新たな要求が生じている．近年，これらの要求に対処する様々な手法が研究されるようになっている．

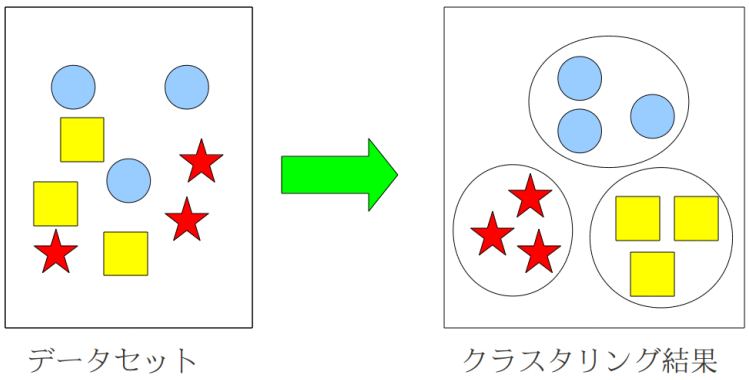

図 3.1: クラスタリング

クラスタリング手法は大きく，最短距離法などの階層的手法 (hierarchical) と，k-means などの分割最適化手法 (partitioning-optimization) に分けられるが，第 2 章のサポートベクトルマシーンが教師あり 2 値分類に基づいた手法で，精度が高く，汎用性が高いクラスタリング手法の一つとして知られている．

代表的なクラスタリング手法

最短距離法または**単連結法**：$D(C_1, C_2) = \min_{x_1 \in C_1, x_2 \in C_2} D(x_1, x_2)$
最長距離法または**完全連結法**：$D(C_1, C_2) = \max_{x_1 \in C_1, x_2 \in C_2} D(x_1, x_2)$
群平均法：$D(C_1, C_2) = \frac{1}{n_1 n_2} \sum_{x_1 \in C_1} \sum_{x_2 \in C_2} D(x_1, x_2)$
ウォード法：$D(C_1, C_2) = E(C_1 \cup C_2) - E(C_1) - E(C_2)$ ただし，$E(C_i) = \sum_{x \in C_i} (D(x, c_i))^2$
K-means **法**：$\min \sum_{i=1}^{k} \sum_{x \in C_i} (D(x, c_i))^2$

しかし，SVM の場合，他のクラスタリング手法に比べて，学習用データが必要となる．学習用データをなしにした上で，SVM を適応させるのは最大マージンクラスタリングである．

データを 2 つのクラスタに分割したときに，その 2 つのクラスを識別する SVM を学習し，そのときのサポートベクトルと分離平面までのマージン (最大マージン) を求める．この最大マージンが最大になるような 2 つのクラスタの分割を求める．n 個のデータに対し，2^n パターンの 2 分割の結果が可能となり，その数に相当する回数の SVM の実行が必要となる．当然，データサイズが大きい場合にはこれは現実的に計算不可能である．しかし，ある種の条件を付け，SVM の最大値問題を半正定値問題に変換することで実行できる．次の節に半正定値問題について説明する．

3.2 半正定値問題

半正定値計画問題 (Semidefinite Programming : SDP)[6, 7] は， 半正定値行列を変数とし，それらに線形制約が課された最適化問題であり，線形計画問題 (LP) や二次錐計画問題など多くのクラスの問題を含んでいる．SDP は，主双対内点法によって多項式時間で最適解を導き出すことができることから， 組合せ最適化， 制御分野， 構造最適化及びデータマイニングなど幅広い分野で活用されている．また，SDPA，CSDP，SeDuMi など，SDP を解く多くのソフトがフリーで公開されている．

半正定値問題の定義を述べる前に，まず，線形行列不等式 ($linear\ matrix\ inequality$)[8] の形が次に定義される．

$$F(\mathbf{u}) := F_0 + u_1F_1 + ... + u_qF_q \preceq 0.$$

ここで，$\mathbf{u}$ は決定変数ベクトルで，$F_0, ..., F_q$ は $p \times p$ の対称行列である．$F(\mathbf{u}) \preceq 0$ というのは，対称行列 F は半負定値であることを意味する．このような式には非線形的な構造になっているのが一般的である．LMI 構造で最大な特徴はその凸性質である．それは，LMI を満たす $\mathbf{u}$ は凸セットであることを示す．

半正定値問題 (Semidefinite Program) は一つの最適値問題であり，一つの線形目的関数と線形不等式行列または一次等式行列という制限条件から成り立つ．SDP の形が次となる．

$$\begin{array}{ll} \text{最小値} & \mathbf{c}^T\mathbf{u}, \quad \mathbf{c}, \mathbf{u} \in \mathbb{R}^q \\ \text{条　件} & F^j(\mathbf{u}) = F_0^j + u_1F_1^j + ... + u_qF_q^j \preceq 0, \\ & A\mathbf{u} = \mathbf{b}, \quad j = 1, ..., L, F_i^j = (F_i^j)^T \in \mathbb{R}^{p\times p} \end{array} \tag{3.1}$$

LMI 条件式の凸性質を考え，半正定値問題は凸最適問題であることが分かる．また，半正定値問題に二つの利点が存在する．まず，半正定値問題形式が独特に見えるが，実は多くの応用問題が適応できる．また，既存の内点法で半正定値問題を解くには理論上と実用上ともよい成果があげている (Vandenberghe,Boyd, 1996).

Schur Complement Lemma という一つ非常に都合のよいレンマが知られ，これにより問題を半正定値問題に適応することができる．

Schur Complement Lemma ： ある対称的な分割行列 X が次のように構成されるとする．

$$X = X^T = \begin{pmatrix} A & S \\ B^T & C \end{pmatrix}$$

ここで，A, C が正方対称行列であり，$det(A) \neq 0$,$S = C - B^T A_{-1} B$ とする．この場合，$S \succeq 0$ の場合のみ，$A \succ 0$ ならば，$X \succeq 0$ になる．

このレンマを用いて非線形凸最適化問題である二次錐計画問題 $(QCQP)$，式 (3.2) を半正定値問題に変換できる証明を次に示す．

$$\begin{array}{ll} \text{最小化} & f_0(\mathbf{u}), \quad \mathbf{u} \in \mathbb{R}^n \\ \text{条　件} & f_i(\mathbf{u}) \leq 0, \quad i = 1, ..., M, \end{array} \tag{3.2}$$

上の式 (3.2) まず次のように変換できる．

$$\begin{array}{ll} \text{最小化} & t \\ \text{条　件} & t - f_0(\mathbf{u}) \geq 0, \quad \mathbf{u} \in \mathbb{R}^n \\ & -f_i(\mathbf{u}) \geq 0, \quad i = 1, ..., M. \end{array} \tag{3.3}$$

式 (3.3) において，$t = f_0(\mathbf{u})$ が最適解となる．ここで，$f_i(\mathbf{u}) = (A_i\mathbf{u} + \mathbf{b}_i)T(A_i\mathbf{u} + \mathbf{b}_i) - \mathbf{c}_i^T\mathbf{u} - d_i$ とすれば，$t - f_0(\mathbf{u}) = (t + c_0^T\mathbf{u} + d_0) - (A_0\mathbf{u} + \mathbf{b}_0)^T I^{-1}(A_0\mathbf{u} + \mathbf{b_0}) \geq 0$ が **Schur Complement Lemma** を満たし，式 (3.3) が次の式 (3.4) に同等になる．

$$\begin{array}{ll} \text{最小化} & t \\ \text{条　件} & \left(\begin{array}{cc} I & A_0\mathbf{u} + \mathbf{b_0} \\ (A_0\mathbf{u} + \mathbf{b_0})^T & \mathbf{c_0}^T\mathbf{u} + d_0 + t \end{array} \right) \succeq 0, \\ & \left(\begin{array}{cc} I & A_i\mathbf{u} + \mathbf{b}_i \\ (A_i\mathbf{u} + \mathbf{b}_i)^T & \mathbf{c}_i^T\mathbf{u} + d_i \end{array} \right) \succeq 0, \quad i = 1, ..., M \end{array} \tag{3.4}$$

このように，$QCQP$ 問題も半正定値問題として扱えることが分かる．また，半正定値問題とその双対問題の関係を次のように定義される．

$$SDP \begin{cases} \text{主問題} & \min & \sum_{i=1}^m c_i x_i \\ & s.\,t. & X = \sum_{i=1}^m F_i x_i - F_0 \\ & & X \succeq O, \quad X \in S \\ \text{双対問題} & \max & F_0 \cdot Y \\ & s.\,t. & F_i \cdot Y = c_i, \quad i = 1, 2, ..., m \\ & & Y \succeq O, \quad Y \in S \end{cases}$$

ここの S は $n \times n$ の対称行列である．

3.3 MMC における代表的な提案

最大マージンクラスタリング問題を半正定値問題に緩和してから解く手法が様々提案されてきたが，その中からいくつか代表的なものを述べる．

まず，Xu,Neufeld らの『**Maximum Margin Clustering**』(2004)[9] についてである．SVM の 1 ノルムソフトマージン最適問題 (2.32) は次のように簡潔に書き換える．

$$\begin{array}{rcl} \gamma^{-2} & = & \min_{\mathbf{w}, b, \boldsymbol{\xi}} ||\mathbf{w}||^2 + C\boldsymbol{\xi}^\top \mathbf{e} \quad s.t.\ y^i(\mathbf{w}^\top \phi(\mathbf{x}^i) + b \geq 1 - \xi_i,\ \forall_{i=1}^N, \boldsymbol{\xi} \geq 0 \\ & = & \max_{\boldsymbol{\alpha}} 2\boldsymbol{\alpha}^\top \mathbf{e} - \langle K \circ \boldsymbol{\alpha}\boldsymbol{\alpha}^\top, \mathbf{y}\mathbf{y}^\top \rangle \quad s.t.\ 0 \leq \boldsymbol{\alpha} \leq C, \boldsymbol{\alpha}^\top y = 0 \end{array} \tag{3.5}$$

ここで，$\Phi = [\phi(x_1), ..., \phi(x_N)]$, $K = \Phi^\top \Phi$, $k_{ij} = \phi(x_i)^\top \phi(x_j)$, $\langle A, B \rangle = \sum_{ij} a_{ij} b_{ij}$ と定義される．

各クラスタのサンプル数がバランスよくとれように，Xu,Neufeld らはクラスタ間のサンプルの差が ℓ 以内であるように制限を入れた．

$$\begin{array}{ll} \min_{y \in \{-1,+1\}^N} & \gamma^{-2}(\mathbf{y}) \quad s.t. \ -\ell \leq \mathbf{e}^\top \mathbf{y} \leq \ell \\ & \gamma^{-2}(\mathbf{y}) \ = \max_{\boldsymbol{\alpha}} 2\boldsymbol{\alpha}^\top \mathbf{e} - \langle K \circ \boldsymbol{\alpha}\boldsymbol{\alpha}^\top, \mathbf{y}\mathbf{y}^\top \rangle \quad s.t. \ 0 \leq \boldsymbol{\alpha} \leq C \end{array} \tag{3.6}$$

しかし，式 (3.6) は凸関数ではない上，効率的に解く方法がない．そこで，直接に $\mathbf{y}$ を求めるのではなく，カーネル行列 $M = \mathbf{y}\mathbf{y}^\top$ を利用して，問題を M を求めるように変換した．

$$\gamma^{-2}(M) = \max_{\boldsymbol{\alpha}} 2\boldsymbol{\alpha}^\top \mathbf{e} - \langle K \circ \boldsymbol{\alpha}\boldsymbol{\alpha}^\top, M \rangle \quad s.t. \ 0 \leq \boldsymbol{\alpha} \leq C \tag{3.7}$$

これで，式 (3.7) が M に関する凸関数になっているが，M と $\mathbf{y}$ との関連づけがまたできていない．つまり，関係 $M = \mathbf{y}\mathbf{y}^\top$ を M に関する関数である (3.7) に反映されなければならない．そこで，Xu,Neufeld らは

$$M \in \{-1, +1\}^{N \times N}, \quad m_{ij} = \begin{cases} 1 & \text{if } y_i = y_j \\ -1 & \text{if } y_i \neq y_j \end{cases}$$

を次のように書き換えた．

$$\begin{array}{l} L_1 : m_{ii} = 1; \ m_{ij} = m_{ji}; \ m_{ik} \geq m_{ij} + m_{jk} - 1; \ \forall_{ijk} \\ L_2 : m_{jk} \geq -m_{ij} - m_{ik} - 1; \ \forall_{ijk} \\ L_3 : -\ell \leq \sum_i m_{ij} \leq \ell; \ \forall_j \end{array}$$

ここに計算コストを下げるため，凸整数問題を更に緩和し，条件 $M \succeq 0$ を入れ，連続空間での最適値問題にする．最終的に MMC が次の式 (3.8) で計算される．また式 (3.9) は式 (3.8) を SDP 問題形式に書き換えたものである．

$$\min_{M \in [-1,+1]^{N \times N}} \max_{\boldsymbol{\alpha}} 2\boldsymbol{\alpha}^\top \mathbf{e} - \langle K \circ \boldsymbol{\alpha}\boldsymbol{\alpha}^\top, M \rangle \quad s.t. \ 0 \leq \boldsymbol{\alpha} \leq C, L_1, L_2, L_3, M \succeq 0 \tag{3.8}$$

$$\begin{array}{ll} \min_{M, \delta, \boldsymbol{\mu}, \boldsymbol{\nu}} \delta & s.t. \quad L_1, L_2, L_3, \boldsymbol{\mu} \geq 0, \boldsymbol{\nu} \geq 0, M \succeq 0, \\ & \qquad \begin{bmatrix} M \circ K & \mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu} \\ (\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu})^\top & \delta - 2C\boldsymbol{\nu}^\top \mathbf{e} \end{bmatrix} \succeq 0 \end{array} \tag{3.9}$$

次に本研究に引用した Valizadegan らの『**Generalized Maximum Margin Clustering and Unsupervised Kernel Learning**』(2007)(略して GMMC)[10] について説明する．

GMMC の場合，SVM の 1 ノルムソフトマージンの式 (2.32) の双対問題に，$\boldsymbol{\alpha}$ の上限 C をなくした形で，次のように使う．

$$\begin{array}{l} \min_{\boldsymbol{\nu}, \mathbf{y}, \lambda} \frac{1}{2} (\mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y})^\top \mathrm{diag}(\mathbf{y}) K^{-1} \mathrm{diag}(\mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y}) \\ s.t. \quad \boldsymbol{\nu} \geq 0, \ \mathbf{y} \in \{-1, +1\}^n \end{array} \tag{3.10}$$

ここで，新たな変量 $\mathbf{z}$ を導入し，式を更に簡単にした．

$$\mathbf{z} = \mathrm{diag}(\mathbf{y})(\mathbf{e} + \boldsymbol{\nu})$$

$\boldsymbol{\nu} \geq 0$ を考慮して，$|z_i| \geq 1$ または $z_i^2 \geq 1,\ i = 1, 2, ..., n$ が成り立つことを上の式から得られる．そして式 (3.10) が次のように変形できる．

$$\begin{array}{c} \min_{\mathbf{z},\lambda} \frac{1}{2}(\mathbf{z} + \lambda\mathbf{e})^\top K^{-1}(\mathbf{z} + \lambda\mathbf{e}) \\ s.t.\ \ z_i^2 \geq 1,\ i = 1, 2, ..., n \end{array} \tag{3.11}$$

しかし，式 (3.11) において，$\mathbf{z}' = \mathbf{z} + \epsilon\mathbf{e}, \lambda' = \lambda - \epsilon$ とすれば，$(\mathbf{z}, \lambda)$ と $(\mathbf{z}', \lambda')$ における最適値が同じになってしまうことが容易に確認できる．そこに，差別するため，目的関数を次のようにする．

$$\begin{array}{cc} \min_{\mathbf{z},\lambda} & \frac{1}{2}(\mathbf{z} + \lambda\mathbf{e})^\top K^{-1}(\mathbf{z} + \lambda\mathbf{e}) + C_e(\mathbf{z}^\top\mathbf{e})^2 \\ s.t. & z_i^2 \geq 1,\ i = 1, 2, ..., n \end{array} \tag{3.12}$$

ここで，$\mathbf{w} = (\mathbf{z}; \lambda), P = (I_n, \mathbf{e})$ とすれば，式 (3.12) が更に次のように表せる．また，式 (3.14) は式 (3.13) のラグランジュ関数である．

$$\begin{array}{cc} \min_{\mathbf{w}\in\mathbf{R}^{n+1}} & \mathbf{w}^\top P^\top K^{-1} P\mathbf{w} + C_e(\mathbf{e}_0^\top\mathbf{w})^2 \\ s.t. & w_i^2 \geq 1,\ i = 1, 2, ..., n \end{array} \tag{3.13}$$

$$\begin{array}{rcl} L(\mathbf{w}, \boldsymbol{\gamma}) & = & \mathbf{w}^\top P^\top K^{-1} P\mathbf{w} + C_e(\mathbf{e}_0^\top\mathbf{w})^2 - \sum_{i=1}^n \gamma_i(\mathbf{w}^\top I_{n+1}^i\mathbf{w} - 1) \\ & = & \mathbf{w}^\top \left(P^\top K^{-1} P + C_e\mathbf{e}_0\mathbf{e}_0^\top - \sum_{i=1}^n \gamma_i I_{n+1}^i\right)\mathbf{w} + \sum_{i=1}^n \gamma_i \end{array} \tag{3.14}$$

これで，式 (3.13) の双対問題が次の (3.15) となる．また，最終結果の $\mathbf{y}$ が $\mathbf{w}$ から得られ，その $\mathbf{w}$ が KKT 条件 $\left(P^\top K^{-1} P + C_e\mathbf{e}_0\mathbf{e}_0^\top - \sum_{i=1}^n \gamma_i I_{n+1}^i\right)\mathbf{w} = \mathbf{0}_{n+1}$ より，$P^\top K^{-1} P + C_e\mathbf{e}_0\mathbf{e}_0^\top - \sum_{i=1}^n \gamma_i I_{n+1}^i$ の最小な固有値と比例するため，間接的に $\mathbf{y}$ が求められる．

$$\begin{array}{cc} \max_{\boldsymbol{\gamma}\in\mathbb{R}^n} & \sum_{i=1}^n \gamma_i,\ \gamma_i \geq 0, i = 1, 2, ..., n \\ s.t. & P^\top K^{-1} P + C_e\mathbf{e}_0\mathbf{e}_0^\top - \sum_{i=1}^n \gamma_i I_{n+1}^i \succeq 0 \end{array} \tag{3.15}$$

最後に Zhao,Wang らの『**Efficient MultiClass Maximum Margin Clustering**』(2009) (略して EMMC)[11, 12] について少し紹介する．

EMMC では今までの SVM，MMC，GMMC などとは違い，双対問題から最適問題を解くのではなく，SVM の主問題から直接に解を求めるのである．まず，SVM の 1 ノルムソフトマージンの多クラスのクラス問題が次のように書ける．

$$\begin{array}{cc} \min_{\mathbf{w}_1,...,\mathbf{w}_n,\boldsymbol{\xi}} & \frac{1}{2}\beta\sum_{p=1}^k ||\mathbf{w}_p||^2 + \frac{1}{n}\sum_{i=1}^n \xi_i \\ s.t & \forall i = 1, ..., n, r = 1, ..., k \\ & \mathbf{w}_{y_i}^\top\mathbf{x}_i + \delta_{y_i,r} - \mathbf{w}_r^\top\mathbf{x}_i \geq 1 - \xi_i \end{array} \tag{3.16}$$

ここで，$\mathbf{y} = (y_1, ..., y_n) \in \{1, ..., k\}^n$ で，$p \in \{1, ..., k\}, \mathbf{w}_p$ はクラス p の重みベクトルで，サンプル $\mathbf{x}$ の識別関数が $y^* = \arg\max_{y\in\{1,...,k\}} \mathbf{w}_y^\top\mathbf{x}$ である．

式 (3.16) から $\xi_i \geq 1-(\mathbf{w}_{y_i}^\top \mathbf{x}_i + \delta_{y_i,r} - \mathbf{w}_r^\top \mathbf{x}_i), \forall r = 1, ..., k$ が得られ，式 (3.16) の目的は $\frac{1}{n}\sum_{i=1}^n \xi_i$ の最小化なので，ξ_i が次のように書ける．

$$\xi_i^{(1)} = \min_{y_i=1,...,k} \max_{r=1,...,k} \{1-(\mathbf{w}_{y_i}^\top \mathbf{x}_i + \delta_{y_i,r} - \mathbf{w}_r^\top \mathbf{x}_i)\}$$

ここで，$\mathbf{x}_i$ とそのラベル y_i との位置関係を次のように設定された．まず，各サンプルが各クラスへの重み付いた順位を $\mathbf{w}_{i_1}^\top \mathbf{x}_i \geq \mathbf{w}_{i_2}^\top \mathbf{x}_i \geq ... \geq \mathbf{w}_{i_k}^\top \mathbf{x}_i$ のように並べたとする．$(i_1, i_2, ..., i_k)$ はラベルの排列 $(1, 2, ..., k)$ である．こうすれば，$y_i \neq i_1$ の場合には，$\max_{r=1,...,k}\{1-(\mathbf{w}_{y_i}^\top \mathbf{x}_i + \delta_{y_i,r} - \mathbf{w}_r^\top \mathbf{x}_i)\} \geq 1$ となり，$y_i = i_1$ の場合，$\max_{r=1,...,k}\{1-(\mathbf{w}_{y_i}^\top \mathbf{x}_i + \delta_{y_i,r} - \mathbf{w}_r^\top \mathbf{x}_i)\} \leq 1$ なので，$y_i^{(1)} = i_1$ となる．つまり，一番重みが高いものが y_i に対応させた．$\xi_i^{(1)}$ も次のようになる．

$$\begin{array}{rcl} \xi_i^{(1)} & = & \max_{r=1,...,k}\{1-(\mathbf{w}_{y_i}^\top \mathbf{x}_i + \delta_{y_i,r} - \mathbf{w}_r^\top \mathbf{x}_i)\} \\ & = & \max\{0, 1-(\mathbf{w}_{i_1}^\top \mathbf{x}_i - \mathbf{w}_{i_2}^\top \mathbf{x}_i)\} \end{array} \tag{3.17}$$

これは各クラスの重み $\mathbf{w}_i$ がみんな同じ最適値 ξ_i を持ち，ラベル $\mathbf{y}$ も同じという意味になる．これで式 (3.16) が次のように変形される．

$$\begin{array}{ll} \min_{\mathbf{w}_1,...,\mathbf{w}_n,\boldsymbol{\xi}} & \frac{1}{2}\beta\sum_{p=1}^k ||\mathbf{w}_p||^2 + \frac{1}{n}\sum_{i=1}^n \xi_i \\ s.t & \forall i = 1, ..., n, r = 1, ..., k \\ & \sum_{p=1}^k \mathbf{w}_p^\top \mathbf{x}_i \prod_{q=1,q\neq p}^k I_{(\mathbf{w}_p^\top \mathbf{x}_i > \mathbf{w}_q^\top \mathbf{x}_i)} \\ & + \prod_{q=1,q\neq r}^k I_{(\mathbf{w}_r^\top \mathbf{x}_i > \mathbf{w}_q^\top \mathbf{x}_i)} - \mathbf{w}_r^\top \mathbf{x}_i \geq 1-\xi_i \end{array} \tag{3.18}$$

更に，$\mathbf{e}_p$ を $k \times 1$ で p 番目が 1 で残りを 0 としたベクトルとする．$\mathbf{e}_0$ は $k \times 1$ のゼロベクトルであり，$\mathbf{e}$ は単位ベクトルである．また，$z_{ip} = \prod_{q=1,q\neq p}^k I_{(\mathbf{w}_p^\top \mathbf{x}_i > \mathbf{w}_q^\top \mathbf{x}_i)}\ \forall i = 1, ..., n; p = 1, ..., k; \mathbf{c} = (\mathbf{c}_1, ..., \mathbf{c}_n)$ は $k \times n$ 行列で，$\boldsymbol{\xi}^* = \frac{1}{n}\sum_{i=1}^n \xi_i^*$ とすれば，式 (3.18) が次のように変形できる．

$$\begin{array}{ll} \min_{\mathbf{w}_1,...,\mathbf{w}_n,\boldsymbol{\xi}\geq 0} & \frac{1}{2}\beta\sum_{p=1}^k ||\mathbf{w}_p||^2 + \xi_i \\ s.t & \forall \mathbf{c}_i \in \{\mathbf{e}_0, \mathbf{e}_1, ..., \mathbf{e}_k\}, i = 1, ..., n \\ & \frac{1}{n}\sum_{i=1}^n \left\{\mathbf{c}_i^\top \mathbf{e}\sum_{p=1}^k \mathbf{w}_p^\top \mathbf{x}_i z_{ip} + \sum_{p=1}^k c_{ip}(z_{ip} - \mathbf{w}_p^\top \mathbf{x}_i)\right\} \\ & \geq \frac{1}{n}\sum_{i=1}^n \mathbf{c}_i^\top \mathbf{e} - \boldsymbol{\xi} \end{array} \tag{3.19}$$

式 (3.19) では変数の数が $2n-1$ までに減らせることができたが，制約条件の数が nk から $(k+1)^n$ まで増えた．そこで，著者らは切断面アルゴリズムを用いて，制約条件の数を減らすことができた．このアルゴリズムではまず制約条件の部分集合を Ω とし，式 (3.19) を解き，最も条件に合わない制限を Ω に入れる．この方法で切断面によって元の MMC 問題の近似解を実行可能領域から得られる (Kelley,1960)．すべての ξ_i が閾値 ϵ より小さくなれば，式 (3.19) が計算終了となる．また，最も条件に合わない制限 $\mathbf{c}_i$ と ξ を次のように定義される．

$$p^* = \arg\max_p(\mathbf{w}_p^\top \mathbf{x}_i;\ r* = \arg\max_{r\neq p^*}(\mathbf{w}_r^\top \mathbf{x}_i;\ i = 1, ..., n$$

$$\mathbf{c}_i = \begin{cases} \mathbf{e}_{r^*} & \text{if } (\mathbf{w}_{p^*}^\top \mathbf{x}_i - \mathbf{w}_{r^*}^\top \mathbf{x}_i) \leq 1 \\ \mathbf{0} & otherwise \end{cases}, i = 1, ..., n \tag{3.20}$$

$$
\begin{aligned}
\xi^* &= \tfrac{1}{n}\sum_{i=1}^{n} \max_{c_i} \left\{ \mathbf{c}_i^\top \mathbf{e} - \mathbf{c}_i^\top \mathbf{e} \sum_{p=1}^{k} \mathbf{w}_p^\top \mathbf{x}_i z_{ip} - \sum_{p=1}^{k} c_{ip}(z_{ip} - \mathbf{w}_p^\top \mathbf{x}_i) \right\} \\
&= \tfrac{1}{n}\sum_{i=1}^{n} \max_{c_i} \left\{ \mathbf{c}_i^\top [\mathbf{e} - \mathbf{w}_{p^*}^\top \mathbf{x}_i \mathbf{e} - \mathbf{z}_i + \mathbf{t}_i] \right\} \\
\mathbf{t}_i &= (\mathbf{w}_1^\top \mathbf{x}_i, ..., \mathbf{w}_k^\top \mathbf{x}_i)^\top
\end{aligned}
\tag{3.21}
$$

最初に紹介した『**Maximum Margin Clustering**』のように，各クラスの要素数がバランスよくとれるように，クラス間の要素数の差が ℓ 以下であるように制限する．これで，切断面多クラスの最大マージンクラスタリングが式 (3.22) に表せる．

$$
\begin{aligned}
\min_{\mathbf{w}_1,...,\mathbf{w}_k,\xi \geq 0} \quad & \tfrac{1}{2}\beta \sum_{p=1}^{k} ||\mathbf{w}_p||^2 + \xi \\
s.t. \quad & \tfrac{1}{n}\sum_{i=1}^{n} \left\{ \mathbf{c}_i^\top \mathbf{e} \sum_{p=1}^{k} \mathbf{w}_p^\top \mathbf{x}_i z_{ip} + \sum_{p=1}^{k} c_{ip}(z_{ip} - \mathbf{w}_p^\top \mathbf{x}_i) \right\} \\
& \geq \tfrac{1}{n}\sum_{i=1}^{n} \mathbf{c}_i^\top \mathbf{e} - \xi, \ \forall [\mathbf{c}_1, ..., \mathbf{c}_n] \in \Omega \\
& -\ell \leq \sum_{i=1}^{n} \mathbf{w}_p^\top \mathbf{x}_i - \sum_{i=1}^{n} \mathbf{w}_q^\top \mathbf{x}_i \leq \ell, \forall p, q = 1, ..., k
\end{aligned}
\tag{3.22}
$$

切断面多クラスの最大マージンクラスタリング (CPM3C) のアルゴリズムは次の表 3.1 となる．

表 3.1: CPM3C アルゴリズム

切断面の多クラスの最大マージンクラスタリング
初期化　$\Omega = \phi$
repeat
1. 制約条件 Ω のもとに問題 (3.22) を解く．
2. 式 (3.20) に従って最も合わない制約条件 $\mathbf{c}$ を計算する．
3.$\Omega = \Omega \cap \{\mathbf{c}\}$.
until $\mathbf{c} \leftarrow (\mathbf{w}_1, ..., \mathbf{w}_k)$ が精度 ϵ に満たす．

しかし，式 (3.22) では目的関数が凸関数ではあるが，その制約条件がそうではなくて，結局求めることが難しい．そこで，制限付凹凸処理 ($CCCP$:Smola et al., 2005) によって，この問題を高速化することができる．

$\frac{1}{n}\sum_{i=1}^{n} \left[\mathbf{c}_i^\top \mathbf{e} \sum_{p=1}^{k} \mathbf{w}_p^\top \mathbf{x}_i z_{ip} + \sum_{p=1}^{k} c_{ip} z_{ip} \right]$ は $(\mathbf{w}_1, ..., \mathbf{w}_k)$ に対する凸関数で，$CCCP$ を利用するには劣勾配 ($subgradient$) を求める必要がある．

$$
\partial_{\mathbf{w}_r} \left\{ \frac{1}{n}\sum_{i=1}^{n} \left[\mathbf{c}_i^\top \mathbf{e} \sum_{p=1}^{k} \mathbf{w}_p^\top \mathbf{x}_i z_{ip} + \sum_{p=1}^{k} c_{ip} z_{ip} \right] \right\} |_{\mathbf{w}=\mathbf{w}^{(t)}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{c}_i^\top \mathbf{e} z_{ip}^{(t)} \mathbf{x}_i \ \forall r = 1, ..., k \tag{3.23}
$$

初期値 $(\mathbf{w}_1^{(0)}, ..., \mathbf{w}_k^{(0)})$ が与えられ，$CCCP$ では $\frac{1}{n}\sum_{i=1}^{n} \left[\mathbf{c}_i^\top \mathbf{e} \sum_{p=1}^{k} \mathbf{w}_p^\top \mathbf{x}_i z_{ip} + \sum_{p=1}^{k} c_{ip} z_{ip} \right]$ を $\frac{1}{n}\sum_{i=1}^{n} \left[\mathbf{c}_i^\top \mathbf{e} \sum_{p=1}^{k} \mathbf{w}_p^\top \mathbf{x}_i z_{ip} + \sum_{p=1}^{k} c_{ip} z_{ip} \right]$ の 1 次テーラー展開に置換して，$(\mathbf{w}_1^{(t)}, ..., \mathbf{w}_k^{(t)})$ から $(\mathbf{w}_1^{(t+1)}, ..., \mathbf{w}_k^{(t+1)})$ を求める．$(\mathbf{w}_1^{(t)}, ..., \mathbf{w}_k^{(t)})$ の 1 次テーラー展開の例が次となる．

$$
\begin{aligned}
&\tfrac{1}{n}\sum_{i=1}^{n} \left\{ \mathbf{c}_i^\top \mathbf{e} \sum_{p=1}^{k} \mathbf{w}_p^\top \mathbf{x}_i z_{ip} + \sum_{p=1}^{k} c_{ip} z_{ip} \right\} + \tfrac{1}{n}\sum_{i=1}^{n} \mathbf{c}_i^\top \mathbf{e} \sum_{p=1}^{k} (\mathbf{w}_p - \mathbf{w}_p^{(t)})^\top \mathbf{x}_i z_{ip}^{(t)} \\
&= \tfrac{1}{n}\sum_{i=1}^{n} \left\{ \mathbf{c}_i^\top \mathbf{e} \sum_{p=1}^{k} \mathbf{w}_p^\top \mathbf{x}_i z_{ip}^{(t)} + \sum_{p=1}^{k} c_{ip} z_{ip}^{(t)} \right\}
\end{aligned}
$$

$\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{c}_i^\top \mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^\top \mathbf{x}_i z_{ip} + \sum_{p=1}^{k} c_{ip} z_{ip}\right]$ の 1 次テーラー展開を用いて，それを式 (3.22) に置換することによって，次の QP($QuadraticProgramming$) 問題が得られる．

$$
\begin{array}{ll}
\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi} & \frac{1}{2}\beta\sum_{p=1}^{k}||\mathbf{w}_p||^2 + \xi \\
s.t. & \forall[\mathbf{c}_1,...,\mathbf{c}_n]\in\Omega,\ \xi\geq 0 \\
 & \frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^\top \mathbf{e} - \xi + \frac{1}{n}\sum_{i=1}^{n}\sum_{p=1}^{k} c_{ip}\mathbf{w}_p^\top \mathbf{x}_i \\
 & -\frac{1}{n}\sum_{i=1} n\left\{\mathbf{c}_i^\top \mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^\top \mathbf{x}_i z_{ip}^{(t)} + \sum_{p=1}^{k} c_{ip} z_{ip}^{(t)}\right\} \leq 0 \\
 & -\ell \leq \sum_{i=1}^{n}\mathbf{w}_p^\top \mathbf{x}_i - \sum_{i=1}^{n}\mathbf{w}_q^\top \mathbf{x}_i \leq \ell, \forall p,q = 1,...,k
\end{array}
\tag{3.24}
$$

$CCCP$ を利用して式 (3.22) を解くプロセスが次の表 3.2 になる．

表 3.2: 効率的な多クラスの最大マージンクラスタリング

CCCP による CPM3C アルゴリズム

初期化　$\mathbf{w}_p = \mathbf{w}_p^0,\ p = 1,...,k$

repeat

1. 問題 (3.24) を解き，最適解 $(\mathbf{w}_1^{t+1},...,\mathbf{w}_k^{t+1})$ を得る．
2. $\mathbf{w}_p = \mathbf{w}_p^{t+1},\ p = 1,...,k$

until 終了条件を満たす．

これで代表的な最大マージンクラスタリング手法 (MMC) を紹介してきた．最初に述べたように，MMC 手法は精度が高いが，従来の $K-means$ 法などに比べ，計算コストが高く，中規模あるいは小規模のデータセットにしか適応できない欠点がある．精度の比較は別にして，三種類の MMC における計算コストの比較結果は次の図 3.2 になる．

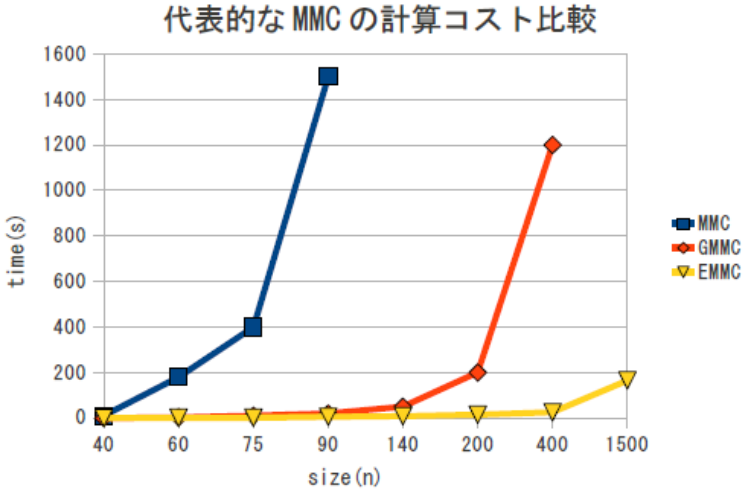

図 3.2: 計算コスト比較

上の図 3.2 から分かるように，MMC が計算コストが高くて，データセットのサイズ (n) が 100 を越えた段階では計算不能になる．それに比較して，GMMC は計算コストある程度改善されたが，それでもデータセットのサイズが 500 を越えると計算不能になる．この中で一番計算コストの問題を改善したのが EMMC である．その限界が上図では示されなかったが，5000 までは計算可能であろう．

第4章　本研究における提案

本研究では紹介した代表的なMMC手法から，汎用性と実行性を考慮した上，Valizadeganらの『**Generalized Maximum Margin Clustering and Unsupervised Kernel Learning**』(2007)(略してGMMC)を元に計算コストを改善する提案をする．

GMMCの元となるMMCではデータをクラスタリングする前に，中心化を行ったが，GMMCではそれを省略した．なぜかと言うと，中心化を行ったとはいえ，必ず分離境界腺が元データを中央から通るという保証がないからである．特に各クラスの要素数のバランスがよくない場合はこの処理が望ましくないである．

中心化の説明をする前にデータセットの構成を次のように定義する．

データセット D が n 個の**要素**(サンプルともいう)から構成し，各要素が一行だけを占める．各要素が m 個の**属性**(次元ともいう)を有する．よって，データセット D は $n \times m$ の行列である．

$$D = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

$$norm1i = \sqrt{a_{i1}^2 + a_{i2}^2 + \cdots + a_{im}^2}$$

$$norm2i = \sqrt{a_{1i}^2 + a_{2i}^2 + \cdots + a_{ni}^2}$$

$$N1 = \begin{bmatrix} \frac{a_{11}}{norm11} & \frac{a_{12}}{norm11} & \cdots & \frac{a_{1m}}{norm11} \\ \frac{a_{21}}{norm12} & \frac{a_{22}}{norm12} & \cdots & \frac{a_{2m}}{norm12} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{norm1n} & \frac{a_{n2}}{norm1n} & \cdots & \frac{a_{nm}}{norm1n} \end{bmatrix} \quad N2 = \begin{bmatrix} \frac{a_{11}}{norm21} & \frac{a_{12}}{norm22} & \cdots & \frac{a_{1m}}{norm2m} \\ \frac{a_{21}}{norm21} & \frac{a_{22}}{norm22} & \cdots & \frac{a_{2m}}{norm2m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{norm21} & \frac{a_{n2}}{norm22} & \cdots & \frac{a_{nm}}{norm2m} \end{bmatrix}$$

ここで注意しなければいけないのは中心化の定義である．中心化実は二通りが存在する．一つはデータセットの各要素に対して中心化を行い，各要素のベクトルの長さが1になるようにし，すべての要素が単位球面に配置するようにする処理である．データセット D は中心化を行った結果が $N1$ となる．この場合は，要素間の関係または距離を角度の大きさによって表す．このパターンの中心化の利点は要素間の関係あるいは距離がより簡単に計れるようになったが，その一方，オリジナルデータと中心化を行った後の要素間の対応した属性の関係が保たないようになってしまう．つまり，オリジナルデータの要素A,Bのs属性に対する関係が中心化した後の要素A,Bのs属性に対する関係が等しくなくなってしまったのである．

各属性間の関係を保つには二通り目の中心化で実現できる．一つ目の要素に対する中心化を行うのに対して，二通り目では属性に対する中心化を行う．データセット D は中心化を行った結果が $N2$ となる．このパターンでは中心化を行った後でも属性間また要素間の関係がオリジナルデータと同じであることが保証される．

提案その一：中心化を行う．GMMC では中心化を行わなかったが，本研究ではオリジナルデータとの同一性を保った上に，中心化を行った．これによって計算上有利のほかに，データ間の関係も一目瞭然になる．

次に，データセット各要素間の関係を決めるカーネル関数について述べる．第 2 章の最後によく使われているカーネル関数 (2.5) について紹介したが，この中で特によく使われているのが RBF カーネル (ガウスカーネルとも呼ぶ) である．

$$\text{RBF カーネル}：K(\mathbf{x},\mathbf{y}) = e^{\frac{-||\mathbf{x}-\mathbf{y}||}{2\sigma^2}}$$

RBF カーネルを用いる場合，時には非常によりパフォーマンスが得られるが，データセットの性質あるいはカーネルの幅 $(2\sigma^2)$ によって，クラスタリング結果の精度が大きく変わる欠点が存在する [10, 13]．ほとんどの論文では，RBF カーネルを用いる場合，カーネルの幅を経験的に決めるのは現状である．次の図 4.1 ではカーネルの幅によって，クラスタリング結果のエラー率が変わることを示してある．

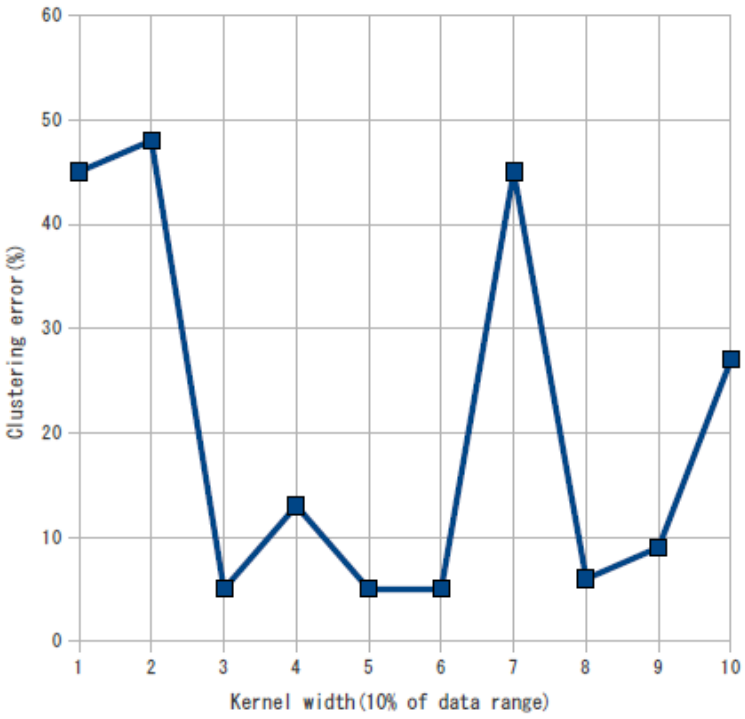

図 4.1: RBF カーネルの幅とクラスタリング結果のエラー率

図 4.1 のように，RBF カーネルによる結果が不安定であることが分かるでしょう．そこで，我々は安定性を求めることで，次のように提案する．

提案その二：線形カーネルを利用する．線形カーネルの利用が安定かつ簡単で，多くの場合では RBF カーネルと同等な精度で結果を生み出せる．

$$\text{線形カーネル：} K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$$

線形カーネルの場合，単なる内積を計算するなので，計算自体が簡単な上，他のパラメータを考慮しなくてよいメリットがある．

以上の二つの提案では最大マージンクラスタリングを安定的に求めることが出きるようになるが，効率性については触れていない．実は，この提案によって計算コストが劇的に抑えられたことがこれからの実験で分かった．これで，線形カーネルによる最大マージンクラスタリングの安定化と効率化が実現できた．これから，本研究の提案を LMMC と呼ぶ．

第5章　実験

本研究では我々が提案した手法を元に，LMMC の精度と計算コストについて実験を行った．

まず実験用データについてだが，本研究では主に機械学習データベース UCI にあるいくつかのデータセットを利用した．データセットのサイズ n が 100 から 4000 までで，属性の数 m が 4 から 10000 までである．

次に実験用プログラミング言語についてだが，データの構造や計算コストなどから総合的に考慮して，実験のメイン計算部分をプログラミング言語 R で作成し，データの作成など細かい操作を要するの場合はプログラミング言語 Python で作成した．

本研究では実験を二段階に分けて行った．第一段階では MMC における各種カーネル関数の精度と安定性について調べた．第二段階では，第一段階の結果を踏まえて最も安定したカーネルを利用し，既存の K-means 法と CLUTO にあるクラスタリング手法との比較を行った．

実験の第一段階は MMC における各種カーネル関数の精度と安定性についての確認である．この段階には，RBF カーネル，2 次多項式カーネル，線形カーネル，マルチカーネル (RBF カーネルと 2 次多項式カーネルを合わせたカーネル) の四つのカーネルを実装した．また，GMMC を元に，実際に解いた問題は次になる．

$$\begin{array}{ll} \max_{\boldsymbol{\gamma}\in\mathbb{R}^n} & \sum_{i=1}^{n}\gamma_i, \\ s.t. & P^\top K^{-1}P + C_e\mathbf{e}_0\mathbf{e}_0^\top - \sum_{i=1}^{n}\gamma_i I_{n+1}^i \succeq 0 \\ & 0 \le \gamma_i \le C_\delta, i = 1, 2, ..., n \end{array} \tag{5.1}$$

実験ではまず，各データセットに対して中心化を行い，各カーネル関数により，それぞれのグラム行列 K を計算しておく．次には式 5.1 を解いてクラスタリングを行うが，その前に 5.1 にあるパラメータに注意して欲しい．既に第 4 章に紹介したように，GMMC の場合，クラスタリングできるデータセットのサイズ n が 500 までである．それは，GMMC の著者 Valizadegan らは式中の C_δ を $100,000$ に設定したので，範囲 $0 \le \gamma_i \le C_\delta$ から単純に γ の値を整数としても，$100,000$ 回の固有値計算 (半正定値の判定) になるので，当然計算コストが高い．しかし，我々の実験には，データの中心化を行うことにより，この制限をなくすことができることが分かった．

第一段階：GMMC パラメータと結果との関係

第一段階ではまず各カーネルの精度について調べた．2 次多項式カーネルとマルチカーネルのクラスタリング結果の精度が宜しくなかったので，後の実験にはこの二種類のカーネルを利用しないことにした．

続いて RBF カーネルの各パラメータと結果精度との関連性について実験を行ったが，この段階で新たな発見があった．Valizadegan らは式 (5.1) におけるパラメータを C_e=10,000; C_δ = 100,000 のように設定したが，我々の実際の実験結果により，クラスタリング結果の精度が C_e にしか関係ないことが分かった．まず，ワインに関する“ wine.data ”というデータセット ($n = 130, m = 13$) についての実験の結果が次のようになる．他のデータセットも同じ結果が出たので，ここでは省略する．

表 5.1: LMMC におけるパラメータと結果との関係

C_e	C_δ	γ^*	正解率 (%)
1	1.5 15 150 1,500 15,000 150,000	0.0001 0.001 0.01 0.1 1 10 100 1000	73.08
10	1.5 15 150 1,500 15,000 150,000	0.0001 0.001 0.01 0.1 1 10 100 1000	56.15
100	1.5 15 150 1,500 15,000 150,000	0.0001 0.001 0.01 0.1 1 10 100 1000	53.08
1,000	1.5 15 150 1,500 15,000 150,000	0.0001 0.001 0.01 0.1 1 10 100 1000	50.0
10,000	1.5 15 150 1,500 15,000 150,000	0.0001 0.001 0.01 0.1 1 10 100 1000	55.38
100,000	1.5 15 150 1,500 15,000 150,000	0.0001 0.001 0.01 0.1 1 10 100 1000	60.0
1,000,000	1.5 15 150 1,500 15,000 150,000	0.0001 0.001 0.01 0.1 1 10 100 1000	63.08
10,000,000	1.5 15 150 1,500 15,000 150,000	0.0001 0.001 0.01 0.1 1 10 100 1000	52.31
100,000,000	1.5 15 150 1,500 15,000 150,000	0.0001 0.001 0.01 0.1 1 10 100 1000	61.54

$\gamma^* = \frac{1}{2\sigma^2}$，RBF カーネルの幅である．

上の表 5.1 から分かるように，実は C_δ と γ はクラスタリングの結果に関係がない．関係あるのは C_e だけである．しかし，単純に C_e の単調増加による結果精度の単調変化が見られないので，最初からどれが一番よいかは分からない．つまり，C_e は大きければ大きいほどがいいということが言えないし，その逆も言えない．C_e を大きく設定すれば，それに関係して C_δ も大きく設定しなければいけないので，計算コストに影響する．その上，RBF カーネルをここに用いる場合は，必ず半正定値問題として解が求まる保証がないことも実験で分かった．

一方，中心化を行ったデータセットに線形カーネルを用いた場合は，表 5.1 と同じ実験を行ったが，パラメータに関係なく，常に一定で比較的高い精度で結果が得られた．同じワインに関する“ wine.data ”というデータセットだが，中心化を行った後に線形カーネルを用いた結果，クラスタリングの正解率は 94.62%である．ただし，中心化を行わない場合の線形カーネルは常に低い精度の結果しか得られない．

第一段階の実験により，RBF カーネルの場合，無駄にすべてのパラメータをいちいち試す必要がないことが分かった．また，パラメータ C_e の値によってクラスタリングの結果が変わる．一方，データセットの中心化＋線形カーネルというモデルで，パラメータに関係なく，安定した最大マージンクラスタリング手法が可能であることも分かった．ここで，一番大事なのは，このモデルでパラメータに制限がないことである．すると，式 (5.1) にあるパラメータを

$C_e = 1, \gamma_i = 0.5$ に設定することができる．そしてこのモデルの場合には次の式が利用できる．

$$\begin{aligned} \max_{\boldsymbol{\gamma} \in \mathbb{R}^n} \quad & \frac{n}{2}, \\ s.t. \quad & P^\top K^{-1} P + \mathbf{e}_0 \mathbf{e}_0^\top - 0.5 \sum_{i=1}^{n} I_{n+1}^i \succeq 0 \\ & i = 1, 2, ..., n \end{aligned} \tag{5.2}$$

更に，$\mathbf{e}_0 \mathbf{e}_0^\top = \mathbf{e_0}$ なので，$\mathbf{e}_0 \mathbf{e}_0^\top - 0.5 \sum_{i=1}^{n} I_{n+1}^i$ は次の $S_{n+1 \times n+1}$ である．

$$S = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0.5 & 0 & \cdots & 0 & 0 \\ 0 & 0.5 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0.5 & 0 \\ 0 & 0 & \cdots & 0 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & \cdots & 0 & 0 \\ 0 & 0.5 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0.5 & 0 \\ 0 & 0 & \cdots & 0 & -0.5 \end{bmatrix}$$

このように，γ_i が既に使わないので，最初の最大値探す問題は一気に半正定値判定問題になる．しかも判定は一回だけで済むなので，固有値判定法を利用する場合には一回の固有値計算になる．GMMC の 100,000 万回の固有値計算に比べ，はるかに早いである．各 MMC の計算コストの比較は次の図 5.1 に示した．

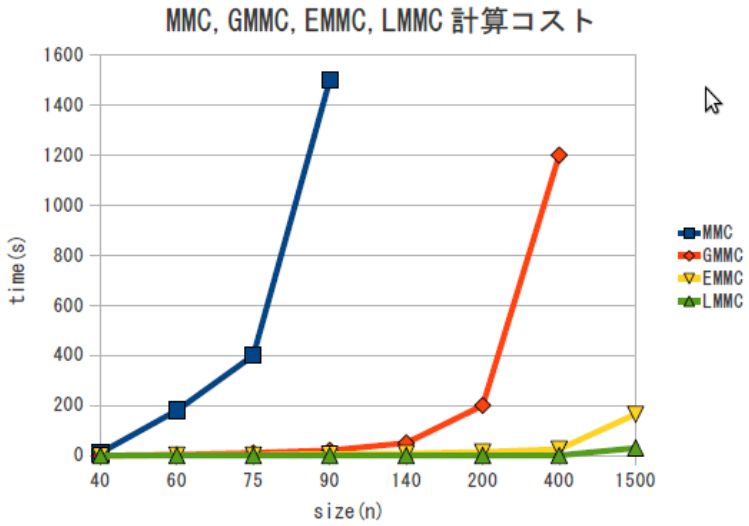

図 5.1: 計算コスト比較

最終的に，問題は次のように変換される．

$$\text{if } (P^\top K^{-1} P + S \succeq 0) = \begin{cases} \text{TURE} & P^\top K^{-1} P + S \text{ の最小固有値ベクトル} \\ \text{FALSE} & \underline{\text{半正定値問題として}}\text{答えがない} \end{cases} \tag{5.3}$$

式 (5.3) にすべてのデータセットに対して半正定値問題として答えがある証明ができないが，手元にあるデータセット皆半正定値問題として解けたので，ほとんどの問題に対応できるでしょう．

第二段階：LMMC と既存のクラスタリング手法の比較

第一段階では線形カーネルを用いた最大マージンクラスタリング (LMMC) を提案したが，これから LMMC，K-means，CLUTO にある vcluster を用いて，いくつかのデータセットについてクラスタリングを行った．

K-means 法についてはプログラミング言語 R にある *kmeans* 関数が標準に存在するので，それを利用した．CLUTO にある vcluster は下記の URL よりダウンロードできる．

実験結果の評価には情報エントロピー (Entropy) と精度 (Purity) を利用した．エントロピーは低ければよい結果であるを示されるが，精度は高いほど結果がよいとされる．実験の結果が次の表 5.2 に示す．

表 5.2: LMMC，K-means，vcluter の比較

データセット	サイズ	時間 (s)	K_e	K_p	vcluster$_e$	vcluster$_p$	LMMC$_e$	LMMC$_p$
iris100	100	0.032	0.5908	0.84	**0.2417**	**0.95**	0.3234	0.94
wine	130	0.017	0.417	0.9154	0.5676	0.8077	**0.2785**	**0.9462**
ionosphere	351	0.274	**0.8112**	**0.7123**	0.9284	0.6125	0.9181	0.641
wdbc	569	1.112	0.5503	0.8541	**0.496**	0.8858	0.4994	**0.8893**
wisconsin	699	2.194	0.2488	0.9585	0.809	0.6509	**0.2204**	**0.9642**
abalone8-9	1257	11.914	0.9514	**0.6285**	0.9466	0.6173	**0.944**	0.6245
kr-vs-kp	3196	188.39	0.9914	0.5544	0.9914	0.5544	**0.9755**	**0.5914**

実験の結果から，LMMC が K-means と CLUTO と同等かそれ以上の精度でクラスタリングできたことが分かる．これで本研究に提案した LMMC の有効性が証明された．これで，中規模のクラスタリングの場合でも，LMMC が一つの選択にもなれるでしょう．

[1]最新の CLUTO が http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download からダウンロードできる．

第6章　考察

6.1　改良点

本研究では最大マージンクラスタリングの最大問題である計算コストの問題をある程度改善できた．第 4 章で紹介したように，これまで様々な最大マージンクラスタリングにおける計算コストの改善案が提案されてきたが，小中規模までのデータセットにしか応用できない欠点が存在する．その上に，実行には多数のパラメータを設定しなければならないので，簡単にはできない問題も存在する．それにより，安定した結果が得られない．本研究では線形カーネルを用いたことにより，簡単かつ安定で，中規模のデータセットまでクラスタリングを行うことができた．

6.2　問題点

まず，精度 (正解率) の問題である．LMMC は GMMC よりはるかに高速に改善されたが，常に高精度にクラスタリングを行うことができない．今の段階では K-means 法より比較的によい結果を得られたが、絶対的ではない．

実際に LMMC で解いた問題は双対問題の式 5.1 であるが，その主問題は次の式 (6.1) である．

$$\begin{array}{cc} \min_{\mathbf{w}\in\mathbb{R}^{n+1}} & \mathbf{w}^\top P^\top K^{-1} P\mathbf{W} + C_e(\mathbf{e}_0^\top \mathbf{w})^2 \\ s.t. & w_i^2 \geq 1,\ i = 1, 2, \ldots, n \end{array} \tag{6.1}$$

しかし，式 (6.1) の制約条件 $w_i^2 \geq 1$ は凸問題ではないので，式 (5.1) の最適値が元問題式 (6.1) の最適値ではない．よって，LMMC では最大マージンクラスタリング問題の最適値の近似値しか出せないので，最良な結果が得られない．

次に，データセットのデータ構造によって，クラスタリングの結果が宜しくない場合がある．LMMC では計算コストを抑えるため，グラム行列 K の逆行列 K^{-1} を $K^{-1} = I - D^{1/2}KD^{1/2}$, $D_{i,i} = \sum_{j=1}^{n} K_{i,j}$ のように定義した．この場合，負の数が存在すると，$D^{1/2}$ が正確に求められなくなることがある．実験結果の表 5.2 にあるデータセット“ ionosphere ”はそうである．電離層に関するデータなで多数の負の数が含まれ，結果に影響が出たと思われる．また，超過疎グラム行列の場合，中心化によって，データ全体よりゼロに近づき，内積を取ると更にゼロが出るので，線形カーネルをベースにする LMMC では不向きだと考えられる．

6.3 今後の課題

まず，LMMC の基本モデルを元に，主問題 6.1 の凸問題への変換方法を考える．これに成功できれば，更なる精度の向上が可能になるでしょう．

次に，できればデータ構造からの影響を弱めることを試みる．例えば，負の数の問題の場合，元の負数を正数に，元の正数を 100 倍になど，データ間の距離を拡大させることも考えられる．同じ考えを超過疎グラムにも適応できるでしょう．

最後に，現段階の LMMC では 2 クラスのクラスタリングしかできないが，今後 3 クラス以上の多クラスへの拡張が課題の一つである．

第7章　おわりに

最大マージンクラスタリングは精度の高いクラスタリング手法の一つである．しかし，計算コストが高く，小規模のデータセットにしか適応できない欠点が存在する．そこで，本研究では中心化の基に，線形カーネルによる最大マージンクラスタリングの安定化と効率化を提案した．実験の結果により，これらの提案が有効であることが示された．特に，MMC に最大の問題となる計算コストの問題が本研究の提案でクリアされることが大きい．

今後として，精度の向上に直面したいくつかの問題を中心に，改善策を考案する．LMMC による実問題への応用が今後の課題である．

参考文献

[1] N.Cristianini,J.Shawe-Talor, *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.

[2] Kristin P.Bennett,Colin Campbell. *Support Vector Machines: Hype or Hallelujah?*.ACM SIGKDD Explorations Newsletter - Special issue on " Scalable data mining algorithms ", Volume 2 Issue 2, Dec. 2000

[3] 山下浩，田中茂．サポートベクターマシンとその応用. 第 13 回日本 OR 学会 RAMP シンポジウム

[4] 福水健次 (統計数理研究所)，松井知子統計数理研究所)，赤穂昭太郎（産業技術総合研究所・脳神経情報研究部門 情報数理研究グループリーダー) . 統計数理研究所　公開講座「カーネル法の最前線—SVM, 非線形データ解析, 構造化データ」

[5] 神嶌敏弘. データマイニング分野のクラスタリング手法 (1) ークラスタリングを使ってみよう !，人工知能学会誌, vol.18, no.1, pp.59-65 (2003)

[6] Gert R.G. Lanckriet,Nello Cristianini,Peter Bartlett,Laurent EI Ghaoui,Michael I. Jordan. *Learning the Kernel Matrix with Semidefinite Programming.*In *Journal of Machine Learning Research* (2004)

[7] 小島政和. 半正定値計画とその応用 第 1 回 半正定値計画問題の基礎

[8] Stephen Boyd,Laurent El Ghaoui,Eric Feron,Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory* Society for Industrial and Applied Mathematics

[9] Linli Xu,James Neufeldy,Bryce Larsony,Dale Schuurmansy. *Maximum Margin Clustering.*In *Advances in Neural Information Processing Systems*, 2004.

[10] Hamed Valizadegan,Rong Jin. *Generalized Maximum Margin Clustering and Unsupervised Kernel Learning.In Advances in Neural Information Processing Systems 19*,pages 1417-1424, 2007.

[11] Bin Zhao,Fei Wang,Changshui Zhang. *Efficient MultiClass Maximum Margin Clustering.*In ICML 2008 25th.

[12] Bin Zhao,Fei Wang,Changshui Zhang. *Efficient Maximum Margin Clustering via Cutting Plane Algorithm*

[13] SEI-HYUNG LEE, KAREN DANIELS. *GAUSSIAN KERNEL WIDTH GENERATOR FOR SUPPORT VECTOR CLUSTERING*

付 録A　ソースコード

A.1　GMMC の各パラメータの関係を調べるソースコード

```
#Generalized Maximum Margin Clustering.

#Input file and output file.
Input <- "./optdigits1-7.data"
Output <- "gmmc-result4"

#The number of the instances of the input file.
M <- 776

#The number that used in training. N could be equal to M.
N <- 776

#The number of the attribute with each instance except the class label.
Ncol <- 64

#Cluster number. You can apply multi-classification on Kmeans and U-SVM but not GMMC right
Class <- 2

#Identity matrix.
I <- diag(N)

#Normalize the data(by columns).
Normalization <- function(data){
   data <- t(data)
   rs <- rowSums(data)
   for (i in 1:N)
      if (rs[i] > 0.0) data[i,] <- data[i,] / rs[i]
   data <- t(data)
   return (data)
}
```

```
#Output the message or result to screen and file.
Printout <- function(obj){
   if (is.character(obj)){
      cat(obj, "\n")
      write(obj, Output, append=T)
   }else if (is.matrix(obj)){
      print(obj); cat("\n")
      write.table(obj, Output, row.name=F, col.name=F, append=T)
   }else{
      print(obj); cat("\n")
      n = length(obj)
      if (n <= 40){
         write.table(matrix(obj,1,n),Output,row.name=F,col.name=F,append=T)
      }else{
         i = 1
         while (i+39 <= n){
            write.table(matrix(obj[i:(i+39)],1,40),Output,row.name=F,col.name=F,append=T)
            i = i+40
         }
         if (i < n)
            write.table(matrix(obj[i:n],1,n-i+1),Output,row.name=F,col.name=F,append=T)
      }
   }
}

#Calculate the accuracy rate of the result.
Accuracyrate <- function(table){
   AR = round(sum(diag(table)) / N, 4)
   if (AR > 0.5)
      return (paste(as.character(AR*100), "%"))
   else
      return (paste(as.character((1-AR)*100), "%"))
}

#Make a pseudo inverse matrix K^(-1) of matrix K.
#K^(-1)=I-sqrt(D)*K*sqrt(D), D is a diagonal matrix Dii=sum(Kij).
InverseMatrix <- function(K){
   D <- matrix(0, N, N)
   for (i in 1:N) D[i,i] <- sqrt(sum(K[i,]))
   Inverse_K <- I - D %*% K %*% D
```

```
   return (Inverse_K)
}

data <- matrix(scan(Input), nrow=M, byrow=T)
data <- data[1:N, ]
#answer <- data[, 1]
#data <- data[, -1]
answer <- data[, Ncol+1]    #It will be used while the answer be the last column.
data <- data[, -(Ncol+1)]
#a <- rep(1,200)            #It will be used while the answer not been written in data.
#b <- rep(2,200)
#answer <- c(a, b)
One <- answer[1]
answer <- ifelse(answer == One, 1, 2)



##################### Generalized Maximum Margin Clustering ############################

#library(MASS)

#We need to normalize the data before using it in test1,2 and 3.
#data <- Normalization(data)


#Linear kernel and K^(-1).
MakeLinearKernel <- function(){
   #We need to normalize the data before using it in linear kernel.
   #data <- Normalization(data)
   t1 <- Sys.time()
   #Printout("Making linear kernel and its inverse matrix...")
   L_K <- data %*% t(data)
   #InverseL_K <- ginv(L_K)
   t2 <- Sys.time()
   #Printout(paste("It took", t2-t1, "seconds.\n"))

   return(L_K)
}

#RFB Kernel and K^(-1), K(i,j)=exp(-γ|xi-xj|^2)=exp(-γ(xi·xi+xj·xj-2xi·xj)).
```

```
MakeRFBKernel <- function(gamma){
   t1 <- Sys.time()
   #Printout("Making Gauss kernel(RBF) and its inverse matrix...")
   Data <- data * data
   G_K <- diag(N)
   for (i in 1:N)
      for (j in 1:N){
         if (i <= j) next
         G_K[i,j] <- exp(-gamma*sum(Data[i,] + Data[j,] - 2*data[i,]*data[j,]))
      }
   G_K <- t(G_K) + G_K - diag(N)
   t2 <- Sys.time()
   #Printout(paste("It took", t2-t1, "seconds.\n"))

   return(G_K)
}

#Polynomial kernel and K^(-1).
#t1 <- Sys.time();
#Printout("Making polynomial kernel and its inverse matrix...")
#P_K <- Gmmcresult_K + 1
#P_K <- P_K * P_K
#InverseP_K <- InverseMatrix(P_K)
#t2 <- Sys.time()
#Printout(paste("It took", t2-t1, "seconds.\n"))

#Multiple kernel and K^(-1).
#Printout("Making multiple kernel and its inverse matrix...")
#M_K <- P_K + G_K
#InverseM_K <- InverseMatrix(M_K)


#Make the constraint matrix.
#t(P)%*%InverseK%*%P + C*e0%*%t(e0) + sum(gamma_i*I).
#C <- 100
#C δ <- 1
P <- cbind(diag(N), 1)
e0<- rbind(matrix(1,N,1), 0)

GMMC <- function(mess, C, C δ , InverseK, g){
```

```
   Maxgamma <- -1
   gamma <- 0.0
   pos <- 0
#   Printout(paste("Making the constraint matrix with",mess,"kernel..."))
   t1 <- Sys.time()
   K_S.T. <- t(P) %*% InverseK %*% P + C * e0 %*% t(e0)
   S.T. <- K_S.T.

   while (gamma < C δ){
      #Make a constraint matrix.
      for (i in 1:N)
         S.T.[i,i] <- K_S.T.[i,i] - gamma

      #Check the matrix whether it is a semidefinite program matrix.
      #It is not necessary to calculate the eigenvactors of the matrix here.
      ev <- eigen(S.T., symmetric=TRUE, only.values=TRUE)
      val <- ev$values
      for (i in 1:(N+1))
         if (val[i] < 0){#(Re(val[i]) < 0 || Im(val[i]) != 0){
            pos <- i
            break
         }
      if (pos == 0){
         Maxgamma <- gamma
         pos <- 0
         #if (g) Printout(paste("gamma =", gamma, "C =", C))
      }
      #test1
      #gamma <- gamma + 0.45

      #test2
      gamma <- gamma + 0.1
   }

   #test3
   #if (Maxgamma<0 || g==0) return()

   t2 <- Sys.time()
   Printout(paste("It took", as.numeric(t2-t1,units="secs"), "seconds in making the
            constraint matrix."))
```

```
   Printout(paste("Max gamma =", Maxgamma))

   #If Maxgamma>0, it means there would be a best constraint matrix. Otherwise no answer.
   for (i in 1:N)
      S.T.[i,i] <- K_S.T.[i,i] - Maxgamma


   #Calculate the labels.
   ev <- eigen(S.T., sym<-T)
   vec <- ev$vectors[, N+1]
   Gmmcresult <- ifelse(vec > 0.0, 1, 2)
   One = Gmmcresult[1]
   Gmmcresult <- ifelse(Gmmcresult == One, 1, 2)
   Gmmcresult <- Gmmcresult[1:N]

   Printout(paste("Clustering result with C=",C,"C δ=",C δ,"gamma=",g,"in",mess,"kernel."))
   #Printout(Gmmcresult)
   tbl = table(Gmmcresult, answer)
   #Printout(tbl)
   Printout(paste("Accuracy rate =", Accuracyrate(tbl), "\n"))
}

Printout(paste("Testing the data set - \"", Input, "\".\n"))
Printout(paste("M =", M, "  N =", N, "  Ncol = ", Ncol, "\n"))

#test 1, and the result will be saved in "./gmmc-result1"
#test 4, the condition are the same as test 1 except nomorlizing the data.
C <- c(1, 10, 100, 1000, 1e+4, 2e+4)
C δ  <- 1.5 * C
gamma <- c(0.0001,0.001,0.01,0.1,1,10,100,1000)

#test 2, and the result will be saved in "./gmmc-result2". test5.
#C <- c(1, 10, 100, 1000, 1e+4, 1e+5, 1e+6, 1e+7, 1e+8)
#C δ<- 1.5   #test2 and test3,gamma increases 0.1 each step
#gamma <- (1:10) / 10

#test 3, and the result will be saved in "./gmmc-result3"
#C <- 1:2e+5
#C δ<- 1.5
#gamma <- 1
```

```
#g <- 1
#cd <- 1.5

L_K <- MakeLinearKernel()
InverseL_K <- InverseMatrix(L_K)

for (c in C){
   for (cd in C δ){
      for (g in gamma){
         G_K = MakeRFBKernel(g)
         InverseG_K <- InverseMatrix(G_K)
         GMMC("Gauss", c, cd, InverseG_K, g)
      }
      GMMC("Linear", c, cd, InverseL_K, g=0)
   }
}
write("\n\n\n############################################################", Output, append=T)
```

A.2 LMMC と K-means 法と vcluster との比較

```
#To compare Kmeans,vcluster and Linear kernel Maximum Margin Clustering.

library(MASS)

#Input file and output file.
Input <- "abalone8-9.data"
Output <- "kmeans-gmmc-result"

#The number of samples in dataset.
N <- 1257

#The number of the attribute with each instance except the class label.
Ncol <- 8

#Cluster number. You can apply multi-classification on Kmeans but not GMMC right now.
Class <- 2
```

```
#Identity matrix.
I <- diag(N)

#Normalize the data(by columns).
Normalization <- function(data){
   d <- abs(data)
   d <- t(d)
   rs <- rowSums(d)
   data <- t(data)
   for (i in 1:Ncol)
      if (rs[i] > 1e-6) data[i,] <- data[i,] / rs[i]
   data <- t(data)

   return (data)
}

#Output the message or result to screen and file.
Printout <- function(obj){
   if (is.character(obj)){
      cat(obj, "\n")
      write(obj, Output, append=T)
   }else if (is.matrix(obj)){
      print(obj); cat("\n")
      write.table(obj, Output, row.name=F, col.name=F, append=T)
   }else{
      print(obj); cat("\n")
      n = length(obj)
      if (n <= 40){
         write.table(matrix(obj,1,n),Output,row.name=F,col.name=F,append=T)
      }else{
         i = 1
         while (i+39 <= n){
            write.table(matrix(obj[i:(i+39)],1,40),Output,row.name=F,col.name=F,append=T)
            i = i+40
         }
         if (i < n)
            write.table(matrix(obj[i:n],1,n-i+1),Output,row.name=F,col.name=F,append=T)
      }
   }
}
```

```
#Calculate the accuracy rate of the result.
Accuracyrate <- function(table){
   AR = round(sum(diag(table)) / N, 4)

   return (paste(as.character(AR*100), "%"))
}

#Calculate the Entropy of the result. ct:cross table
Entropy <- function(ct){
   round(-sum((apply(ct,1,sum) / N) * apply(ct,1,ClassEntropy)), 4)
}

ClassEntropy <- function(pv){
   p1 <- pv / sum(pv)
   p2 <- p1[p1 != 0]
   sum(p2 * log(p2, Class))
}

#Calculate the Purity of the result. ct:cross table
Purity <-function(ct){
   round(sum(apply(ct,1,max)) / N, 4)
}

#Make a pseudo inverse matrix K^(-1) of matrix K.
#K^(-1)<-I-sqrt(D)*K*sqrt(D), D is a diagonal matrix Dii<-sum(Kij).
InverseMatrix <- function(K){
   #Maybe it will use library to calculate K^-1
   #while D cannot be able to be calculated correctly.
   d <- TRUE

   D <- matrix(0, N, N)
   for (i in 1:N){
      s <- sum(K[i,])
      if (s >= 0) D[i,i] <- sqrt(s)
      else{
         d <- FALSE
         break
      }
   }
```

```
   if (d) Inverse_K <- I - D %*% K %*% D
   else Inverse_K <- ginv(K)

   return (Inverse_K)
}

data <- matrix(scan(Input), nrow=N, byrow=T)
data <- data[1:N, ]
#answer <- data[, 1]
#data <- data[, -1]
answer <- data[, Ncol+1]    #It will be used while the answer be the last column.
data <- data[, -(Ncol+1)]
#a <- rep(1,200)            #It will be used while the answer not been written in data.
#b <- rep(2,200)
#answer <- c(a, b)
One <- answer[1]
answer <- ifelse(answer == One, 1, 2)

#Kmeans method. Let the first instance belong to class one, and others two.
km <- kmeans(data,Class)
One <- km$cluster[1]
kmresult <- ifelse(km$cluster == One, 1, 2)

Printout(paste("Testing the data set - \"", Input, "\".\n"))
Printout(paste("N =", N, "  Ncol = ", Ncol, "\n"))
Printout("The clustering result of Kmeans method.")
Printout(kmresult)
Printout("Check the result of Kmeans method.")
tbl <- table(kmresult, answer)
Printout(tbl)
Printout(paste("Accuracy rate =", Accuracyrate(tbl)))
Printout(paste("Entropy =", Entropy(tbl)))
Printout(paste("Purity =", Purity(tbl), "\n"))

Printout("The clustering result by Cluto.")
file1 <- paste("./cluto-2.1.1/MMC-testdata/", Input, ".clustering.2", sep="")
file2 <- paste("./cluto-2.1.1/MMC-testdata/", Input, ".rlabel", sep="")
result1 <- c(matrix(scan(file1), nrow=N, 1))
result2 <- c(matrix(scan(file2), nrow=N, 1))
```

```
One <- result1[1]
result1 <- ifelse(result1 == One, 1, 2)
One <- result2[1]
result2 <- ifelse(result2 == One, 1, 2)
Printout(result1)
tbl <- table(result1, result2)
Printout(tbl)
Printout(paste("Accuracy rate =", Accuracyrate(tbl)))
Printout(paste("Entropy =", Entropy(tbl)))
Printout(paste("Purity =", Purity(tbl), "\n"))

#################### Generalized Maximum Margin Clustering ############################

#We need to normalize the data before using it.
data <- Normalization(data)

#Linear kernel and K^(-1).
L_K <- data %*% t(data)
InverseL_K <- InverseMatrix(L_K)

C <- 1
gamma <- 0.5
t1 <- Sys.time()

#Make the constraint matrix.
P <- cbind(diag(N), 1)
e0<- rbind(matrix(1,N,1), 0)
K_S.T. <- t(P) %*% InverseL_K %*% P + C * e0 %*% t(e0)
S.T. <- K_S.T.
for (i in 1:N)
   S.T.[i,i] <- K_S.T.[i,i] - gamma

#Check the matrix whether it is a semidefinite program matrix.
ev <- eigen(S.T., symmetric=TRUE)
val <- ev$values
pos <- 0
for (i in 1:(N+1))
   if (val[i] < 0){
      pos <- i
      break
```

```
   }

#There is not any answer with convex constraint.
if (pos){
   Printout("There is not any answer with convex constraint. ")
   return()
}

#Calculate the labels.
vec <- ev$vectors[, N+1]
Gmmcresult <- ifelse(vec > 0.0, 1, 2)
One = Gmmcresult[1]
Gmmcresult <- ifelse(Gmmcresult == One, 1, 2)
Gmmcresult <- Gmmcresult[1:N]

t2 <- Sys.time()
Printout(paste("It took", as.numeric(t2-t1,units="secs"), "seconds in GMMC method."))

Printout("The clustering result of GMMC method.")
Printout(Gmmcresult)
Printout("Check the result of GMMC method.")
tbl = table(Gmmcresult, answer)
Printout(tbl)
Printout(paste("Accuracy rate =", Accuracyrate(tbl)))
Printout(paste("Entropy =", Entropy(tbl)))
Printout(paste("Purity =", Purity(tbl), "\n"))

write("\n\n#########################################################", Output, append=T)
```

A.3 vcluter用データの生成

```
#Create a testdata for vcluster from a matrix.

Input <- "./wisconsin.data"
Output1<- "./cluto-2.1.1/MMC-testdata/wisconsin.data"
Output2<- "./cluto-2.1.1/MMC-testdata/wisconsin.data.rlabel"
```

```
N <- 699
M <- 10

data <- matrix(scan(Input),nrow=N, byrow=TRUE)
newdata <- data[, 1:M-1]
#newdata <- data[, 2:M]

write.table(matrix(c(N, M-1), Output1, row.name=F, col.name=F)
write.table(matrix(newdata, N, M-1), Output1, row.name=F, col.name=F)
write.table(matrix(data[,M], N, 1), Output2, row.name=F, col.name=F)
#write.table(matrix(data[,1], N, 1), Output2, row.name=F, col.name=F)
```

A.4 CLUTO データからテストデータの生成

```
#Create a matrix from cluto-database.

n = 5832 + 1

input = "./test_data"
output= "./test_data_matrix"

fin = open(input, "r")
fout= open(output,"a")

for line in fin:
   line = line.split()
   l = len(line)
   label = [0]
   value = [0]
   for i in range(l):
      if i%2==0 :
         label.append(line[i])
         value.append(line[i+1])
   row = ""
   j = 0
   for i in range(1,n):
      if str(i) in label:
```

```
            j = j+1
            row = row + value[j] + ' '
         else:
            row = row + '0 '
      fout.write(row + '\n')

fin.close()
fout.close()
```