CEMアルゴリズムを用いたマルウェアのクラスタリング

レー ナン トゥアン アン[1,a)]　大久保 潤[1,b)]

概要：クラスタリング手法のひとつに Expectation-Maximization (EM) アルゴリズムを用いたものがある．混合ガウス分布などを仮定した EM アルゴリズムがよく知られているが，同時期に収集できたデータ同士の類似度が高いという特徴を有するマルウェアデータに対して精度が悪くなる場合があること，年ごとに解析してしまうと全体のデータを有効活用できないこと，などの問題がある．これらの問題を解決するために，条件付き確率を利用した Conditional EM (CEM) アルゴリズムが提案されている．本研究では，時系列マルウェアデータに CEM アルゴリズムと EM アルゴリズムをそれぞれ適用し，クラスタリングの性能を比較する．その結果，特徴ベクトルの生成方法を表現力を高められるように工夫することにより，CEM アルゴリズムのほうが EM アルゴリズムよりも高い分類性能を示す可能性が示唆された．

キーワード：マルウェア，クラスタリング，CEM アルゴリズム，EM アルゴリズム

Application of CEM algorithm to malware clustering

Tuan Anh Le Nang[1,a)]　Jun Ohkubo[1,b)]

Abstract: Expectation-Maximization (EM) algorithm is one of the famous methods for clustering. It is possible to use various probabilistic models in the EM algorithm, and a Gaussian mixture model is widely used. However, some of malware data sets collected in a similar time sometimes show high similarity, and a naive application of the EM algorithm gives sometimes a low accuracy in clustering. In addition, separate analysis for each year cannot use the entire data effectively. In order to solve these problems, the conditional EM (CEM) algorithm has been proposed, in which conditional probabilities are employed. In the present paper, the EM and CEM algorithms are applied to time-series data set of malware, and comparisons of the performance are given. The numerical results indicate that the CEM algorithm shows higher classification performance than the EM algorithm by using adequate feature vectors with high-power of expression.

Keywords: malware, clustering, CEM algorithm, EM algorithm

1. はじめに

BSA グローバルソフトウェア調査 [1] によると，不正ソフトウェアから侵入したマルウェアが引き起こす被害が毎年 3,590 億ドルにも達しており，これからも増加していく傾向が見られている．また，一件のマルウェア攻撃を検出するためには平均で 243 日かかり，解決するには最大で 50 日を要するということが同調査から明らかになっている．さらに，McAfee 社が公開している 2018 年 3 月の脅威レポート [2] によると 2017 年の第 4 四半期だけでも，過去最高の 6,340 万件の新型マルウェアが発見された．つまり，現在毎秒新たな 8 件の脅威に直面している．こういった状況のなかで，マルウェアの検出への注力が求められている．

近年，機械学習が急速に発展し，様々な分野への応用が図られている．マルウェア分野もその一つで，高精度，低誤検知率かつ高速で未知のマルウェアの検出を可能にすることが期待されている．

本研究では，機械学習によるマルウェアのクラスタリングに着目する．機械学習によるクラスタリング手法は様々あるが，EM アルゴリズムが一般的に使用されている．な

[1] 埼玉大学大学院理工学研究科
Graduate School of Science and Engineering, Saitama University
[a)] le.n.t.a.811@ms.saitama-u.ac.jp
[b)] johkubo@mail.saitama-u.ac.jp

お，同時期に発見されるマルウェアは類似度が高いことがわかっている [3]．その理由として，その時期のセキュリティホールを悪用してマルウェアを生成する，または，流行しているマルウェアと似たものを生成するなどといったことが考えられる．そのため，マルウェアデータに EM アルゴリズムを適用する際，いくつかの問題に直面する．まず，マルウェアデータを大量に扱いたい場合，マルウェアデータが時系列データになり，EM アルゴリズムでは同時期のマルウェアの類似度が高いことからファミリーごとではなく時期ごとにクラスタリングする傾向があるという問題である．また，その問題を避けるために，マルウェアデータを一度にすべて扱うのではなく，同時期のデータのみを扱う方法も考えられるが，そうすると全体データの一部しか使えないという無駄が生じてしまう．

この問題点を解決するために，本研究では CEM アルゴリズム [4] を導入することを提案する．CEM アルゴリズムは条件付き確率を用いた EM アルゴリズムの拡張である．CEM アルゴリズムを用いることで条件付き分類をできるようになるため，時系列マルウェアデータに対して，時間を条件にしてクラスタリングすることにより，マルウェアの全体データを有効活用でき，EM アルゴリズムの問題点を解決できると期待される．そのために本研究では，実際に時系列マルウェアデータに CEM アルゴリズムを適用し，EM アルゴリズムと比べてどのような性能を示すか，数値的に検証する．

2. 関連研究

2.1 EM アルゴリズム

EM アルゴリズムは潜在変数を持つモデルの最尤解を求めるための強力な方法である [5]．

以降，$\mathbf{X}$ を観測変数，$\mathbf{Z}$ を潜在変数，$\boldsymbol{\theta}$ を求めたいパラメータとする．最尤法により，パラメータ $\boldsymbol{\theta}$ について対数尤度関数 $\ln p(\mathbf{X}|\boldsymbol{\theta})$ を最大化することが目的になる．この計算は解析的に困難なため，以下のアルゴリズムにより数値的に実装される．

Algorithm 1 EM アルゴリズム

1: $\boldsymbol{\theta}^{\text{old}}$ を初期化する．
2: E ステップ：$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$ を計算する．
3: M ステップ:Q 関数 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ を $\boldsymbol{\theta}$ について最大化する．すなわち $\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ とする．
4: 収束条件を満たしていれば終了する．そうでなければ $\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}}$ として E ステップに戻る．

2.1.1 問題点

EM アルゴリズムは非常に強力であるが，同じ時間に存在するマルウェアは類似度が高く，また，時間経過とともに，少しずつ変化するといった特徴を有するマルウェアデータにおいては，問題が生じてうまく機能しない．**図 1** を用いてこの問題について説明する．なお，実際のマルウェアデータとは異なるが，ここでは図 1 のデータが模擬的にマルウェアデータを表しているものとする．ここで，データは 4 個の塊に集中しているが，横軸 x1 を時間と考えて，異なる 2 年に存在する 2 つのマルウェアファミリーを表しているものとする．縦方向を見ると，同じ年に出た 2 つのマルウェアファミリーのデータ，横方向を見ると，1 つのマルウェアファミリーの異なる 2 年のデータであることがわかる．同時間でのマルウェアは類似度が高いため，縦方向のほうが塊間の距離が近い．そのようなデータを EM アルゴリズムを用いて 2 つにクラスタリングすると**図 2** のような結果が得られる．この結果は距離の近い 2 つの塊に分けられており，直感的にも理解できるものである．ただし，マルウェアファミリーごとにクラスタリングしたいという本来の目的を考えると，これは望ましい結果ではないということがわかる．

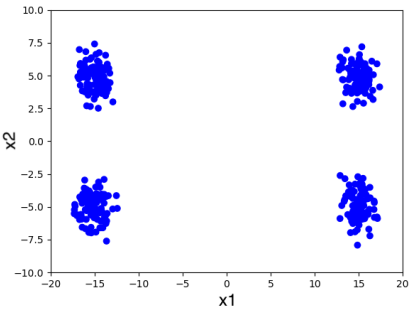

図 1 データセット

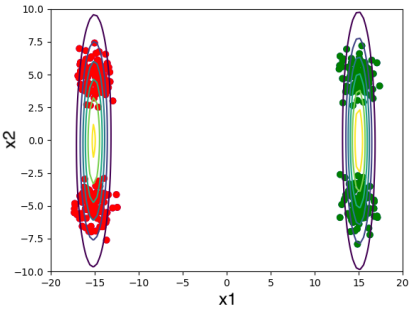

図 2 EM アルゴリズムを用いたクラスタリング

その問題を避けるために，複数年度のデータを取らず，1 年のデータだけを用いる方法も考えられる．しかし，その年度のデータだけを使うと多年度のデータを使えないため，データを有効利用できずに無駄にしてしまうという問題が出てくる．マルウェアは時間経過につれて少しずつ変化していくはずである．その関連性を活かした解析手法があれば，この問題を解決できると期待される．

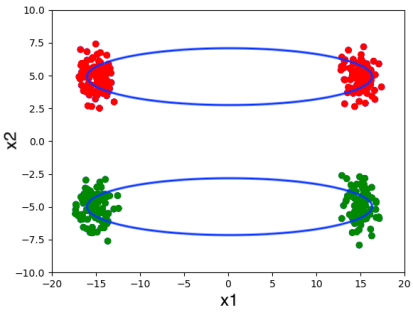

図 3 CEM アルゴリズムを用いたクラスタリング

2.2 CEM アルゴリズム

CEM アルゴリズムは条件付き確率を用いた EM アルゴリズムの拡張である [4]．CEM アルゴリズムを用いることで，条件付き分類をできるようになるため，2.1.1 節で示した EM アルゴリズムの問題点を解決できると期待される．CEM のアルゴリズムは以下のように実装される．

Algorithm 2 CEM アルゴリズム

1: パラメータを初期化し，条件対数尤度 $\ln p(\mathbf{Y}|\mathbf{Z},\boldsymbol{\theta})$ の初期値を計算する．
2: CE ステップ：現在のパラメータ値を使って，h_{nk}, u_{nk}, r_n を計算する．

$$h_{nk} = \ln \frac{p(k,\mathbf{y}_n,\mathbf{z}_n|\boldsymbol{\theta}^{\text{old}})}{\sum_{k=1}^{K} p(k,\mathbf{y}_n,\mathbf{z}_n|\boldsymbol{\theta}^{\text{old}})} \tag{1}$$

$$u_{nk} = \ln p(k,\mathbf{y}_n,\mathbf{z}_n|\boldsymbol{\theta}^{\text{old}}) \tag{2}$$

$$r_n = \frac{1}{\sum_{k=1}^{K} p(k,\mathbf{z}_n|\boldsymbol{\theta}^{\text{old}})} \tag{3}$$

3: M ステップ：式 (A.20)，(A.21)，(A.26)，(A.31) によりパラメータ値を再計算する．(付録 A.1 を参照)
4: 条件対数尤度 $\ln p(\mathbf{Y}|\mathbf{Z},\boldsymbol{\theta})$ を計算し，パラメータ値または対数尤度の収束性を確認し，収束基準を満たしていれば終了する．そうでなければ CE ステップに戻る．

実際に，CEM アルゴリズムを用いて図 1 のデータセットを x1 条件の下で 2 クラスタにクラスタリングした結果が**図 3** である．例えば，x1 = −15 のところで x2 方向に線を引くと 2 つのガウス分布の山ができていることがわかる．

EM アルゴリズムを用いて CEM アルゴリズムの場合と同様の結果を得ようとするならば，x1 軸方向をいくつかの区間に分割し，複数回に分けてクラスタリングしなければならない．なお，得られた複数の結果を統合しても，最終的に妥当な結果が得られるとは限らない．また，統合しなければ，一部のデータしか使えず全体データを有効活用できないということになる．つまり，このような条件をつけてクラスタリングしたい場合には，全体データを有効活用する観点からも CEM アルゴリズムが適していると期待される．実際に CEM アルゴリズムを用いた場合に精度がどのようになるのか不明であるため，本稿ではマルウェアデータに CEM アルゴリズムを適用する方法を提案し，数値実験を通して検証する．

3. 提案手法

本研究では，マルウェアデータを CEM アルゴリズムに適用するために，まずマルウェアの解析ログに含まれる API コール列を抽出し，特徴ベクトルを生成する．API コール列は自然言語における文章として捉えられるため，自然言語処理における特徴ベクトル生成手法である BoW，Bit BoW もしくは Word2Vec を用いる．

また，得られたベクトルに時間情報を追加した後で時間を条件とし CEM アルゴリズムを用いる手法，時間情報が追加されたデータに EM アルゴリズムを用いる手法，そして時間情報が追加されないデータに EM アルゴリズムを用いる手法の 3 つの手法を提案する．

3.1 API コール列と時間情報を抽出する

API コール列はマルウェアの解析結果から API の関数名を実行順に並べたものである．また，時間情報は VirusTotal の検査結果の分析日時とする．

3.2 特徴ベクトルを生成する

以下の 3 つの手法のうち，いずれかを用いて特徴ベクトルを生成する．

3.2.1 BoW

これは Bag of Word の略で，文章をベクトル化するのが目的である．辞書（すべての文章に現れる単語の集合）の単語数がベクトルの次元数となり，その文章に現れる単語の回数によりベクトルを生成する．例えば，辞書を 4 つの単語からなる集合 $\{A, B, C, D\}$ とすると，文章「A B A D」を表すベクトルは $(2, 1, 0, 1)^{\mathrm{T}}$ となる．

これによって，特徴ベクトルを生成し，得られたベクトルを標準化せず主成分分析によって次元圧縮をする．本研究では，70 次元にまで圧縮する．

3.2.2 BitBoW

これは Bit Bag of Word の略で，BoW と似ているが，BoW ではその文章に現れる単語の回数を数えるのに対し，Bit BoW では単語が現れたら 1，そうでなければ 0 という値になる．例えば，上記の例を使って BitBoW で文章「A B A D」をベクトル化すると $(1, 1, 0, 1)^{\mathrm{T}}$ が得られる．

これによって，特徴ベクトルを生成した後，3.2.1 節の BoW と同様に得られたベクトルを標準化せず主成分分析によって次元圧縮をする．本研究では 70 次元にまで圧縮する．

3.2.3 Word2Vec

これは Mikolov ら [6], [7] によって提案された手法である．ニューラルネットワークを用いて，文章の集合であるコーパスを入力として単語の意味や文脈を学習し，単語ごとのベクトルを生成する．Word2Vec には CBoW (Continuous

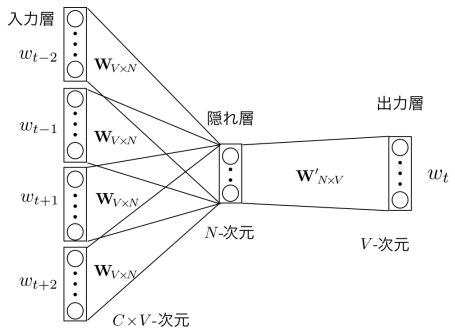

図 4 CBoW モデル

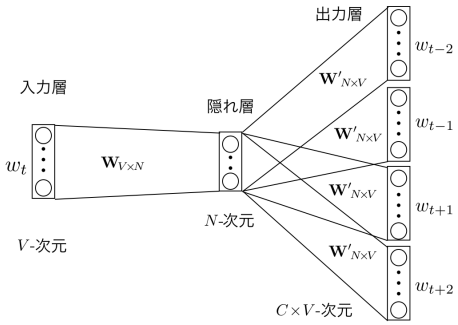

図 5 Skip-gram モデル

Bag of Words) モデルと Skip-gram モデルがある．**図 4** に示すように CBoW では，前後の単語から対象単語を推定する．一方，**図 5** に示すように Skip-gram では単語からその周辺単語を推定する．なお，いくつかの先行研究では，Skip-gram の方が単語の予測精度が高いとわかっている．

本研究では，4.1 節で説明するデータセットに含まれる 2013 年から 2015 年まで出た検体のうち，1 つ以上の API が呼び出される 8639 検体を用いて API コール列を抽出する．その結果，8639 個の API コール列を作成でき，それをコーパスとする．そのコーパスから，次元数を 70 にして，Skip-gram モデルを使用し学習させると API ごとのベクトルが得られる．各検体の特徴ベクトルは，その検体の API コール列に現れる API のベクトルを足し合わせたものとする．

3.3 時間情報を追加する

時間情報を整数に変換し，平均 0 で −1 から 1 まで正規化を行って 3.2 節で得られたベクトルに追加し，次元を 1 つ増やす．4 節の数値実験では，時間情報を追加しないものについても検討する．

3.4 クラスタリングする

本研究では，時間情報を加えたデータについては，時間を条件とし CEM アルゴリズムを用いて初期条件を変えて 10 回クラスタリングする．また，EM アルゴリズムを用いる手法については，時間情報を加えたデータ，または加えていないデータそれぞれについて，初期条件を変えて 10 回クラスタリングする．

4. 数値実験

4.1 データセット

本研究では，MWS が提供する研究用データセット MWS Datasets 2018 の一部として FFRI Dataset 2018（含む FFRI Dataset 2013 - 2017）を用いる [8]．これは，株式会社 FFRI が収集した PE 形式かつ Windows プラットフォーム上で実行可能なマルウェア検体の動的解析ログである．この中から，2013 年から 2015 年までの 100 個以上出現するマルウェアファミリーを選んで，それらのマルウェアファミリーのすべての検体をデータセットとする．ただし，Generic という名前がつけられたファミリーは特定の種類ではなく「その他」として扱われているため，実験対象から除く [9]．その結果として，**表 1** に示すように 7 種類のファミリーの 1368 検体が含まれるデータセットが得られた．

また，本研究で用いる正解データはデータセットに含まれる Kaspersky 社が検体に付けたラベル（名称）を利用する．

表 1 実験に用いるファミリーと検体数

ファミリー名	検体数
Hoax.Win32.ArchSMS	286
Worm.Win32.WBNA	275
Trojan-Spy.Win32.Zbot	243
Trojan.Win32.Yakes	208
Trojan.Win32.Jorik	149
Backdoor.Win32.Androm	104
Trojan-Ransom.Win32.Foreign	103
合計	1368

4.2 評価方法

Bayer ら [10] が提案した精度と再現率の計算方法を用いてクラスタリングの性能を評価する．ここで，サンプル数を N，マルウェアの種類（クラスタ数）を K とする．また，正解クラスタをそれぞれ K 個の集合の $T_1, T_2,..., T_K$，クラスタリングした結果のクラスタをそれぞれ K 個の集合の $C_1, C_2,..., C_K$ とする．精度，再現率と F 値を以下のように定義する．まず，結果のクラスタ C_j に対する精度を

$$P_j = max(|C_j \cap T_1|, |C_j \cap T_2|, ..., |C_j \cap T_K|) \quad (4)$$

とすると，全体の精度は

$$P = \frac{P_1 + P_2 + ... + P_K}{N} \quad (5)$$

となる．また，正解クラスタ T_j に対する再現率を

$$R_j = max(|C_1 \cap T_j|, |C_2 \cap T_j|, ..., |C_K \cap T_j|) \quad (6)$$

とすると，全体の再現率は

$$R = \frac{R_1 + R_2 + ... + R_K}{N} \tag{7}$$

となる．F 値は精度と再現率を使って以下のように計算する．

$$F = \frac{2PR}{P + R} \tag{8}$$

4.3 結果

表 2 は，CEM アルゴリズムと EM アルゴリズムをそれぞれ用いた場合の F 値の平均と標準偏差（括弧内の数字）を表す．表中の太字は，CEM アルゴリズムもしくは EM アルゴリズムで性能がよかったほうを示している．表 2 により，BoW と Word2Vec では CEM アルゴリズムのほうが F 値が高く，特に Word2Vec では最も高い値を示していることがわかる．また，情報量が少ない Bit BoW では EM アルゴリズムの方が F 値が高いことがわかる．そして，時間情報を追加しても EM アルゴリズムではあまり精度が変わらないこともわかる．

CEM アルゴリズムが EM アルゴリズムより良い結果を与えたのは BoW と Word2Vec であることがわかった．Bit BoW と比べると BoW と Word2Vec は特徴ベクトルに多くの情報を含んでいることから，用いる特徴ベクトルが多い情報量を持つ場合には CEM アルゴリズムが適していると考えられる．特徴ベクトルが多くの情報量を含んでいるほうが一般的に分類精度が高いと期待されるが，Bit BoW に対して CEM アルゴリズムを適用した場合には精度が低いことから，CEM アルゴリズムに適していない特徴ベクトルも存在することがうかがえる．このことから，CEM アルゴリズムの利用の際には注意が必要であることもわかる．

また，Word2Vec で特徴ベクトルを生成し，CEM アルゴリズムを適用した場合が最も精度が高かったことから，さらに表現能力が高い特徴ベクトル生成方法を用いることで，より高い精度が期待できる可能性が示唆された．

表 2 CEM と EM を用いたマルウェアクラスタリングの F 値比較

	CEM	EM	EM (時間追加)
BoW	**69.34** (4.70)	52.54 (1.00)	52.78 (0.88)
Bit BoW	43.34 (3.81)	**62.62** (1.97)	61.16 (1.74)
Word2Vec	**71.93** (2.61)	52.80 (0.98)	52.43 (0.73)

5. まとめ

EM アルゴリズムは様々な場面で強力であるが，条件付き分類では全体データを有効活用できないという問題がある．CEM アルゴリズムはこのような問題を，精度を落とさず解決できると期待された．そして本研究では実際に時系列マルウェアデータのクラスタリング問題に対して CEM アルゴリズムと EM アルゴリズムを適用し，性能を比較した．

結果として，表現力が高い特徴ベクトル生成方法であるほど，CEM アルゴリズムは EM アルゴリズムよりもいい性能を示しているということがわかった．

参考文献

[1] BSA: グローバルソフトウェア調査, 入手先 ⟨https://gss.bsa.org/wp-content/uploads/2018/05/2018_BSA_GSS_Report_jp.pdf⟩

[2] McAfee: 脅威レポート 2018 年 3 月, 入手先 ⟨https://www.mcafee.com/enterprise/ja-jp/assets/reports/rp-quarterly-threats-mar-2018.pdf⟩

[3] 柏井 祐樹, 森井 昌克, 井上 大介, 中尾 康二: NONSTOP データを用いたマルウェアの時系列分析, コンピュータセキュリティシンポジウム 2013 論文集, pp. 848 - 853, 2013 年 10 月.

[4] T. Jebara and A. Pentland: “Maximum conditional likelihood via bound maximization and the CEM algorithm.”, Advances in neural information processing systems 11, pp.494-500, 1999.

[5] C.M. ビショップ著, 元田 浩, 栗田 多喜夫, 樋口 知之, 松本 裕治, 村田 昇 共監訳: パターン認識と機械学習 上下, シュプリンガー・ジャパン, 2012 年.

[6] T.Mikolov, K.Chen, G.Corrado and J.Dean: “Efficient Estimation of Word Representations in Vector Space.”, In Proceedings of Workshop at ICLR, arXiv:1301.3781, 2013.

[7] T.Mikolov, K.Chen, G.Corrado and J.Dean: “Distributed Representations of Words and Phrases and their Compositionality.”, In Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS), pp.3111-3119, 2013.

[8] 高田雄太, 他: マルウェア対策のための研究用データセット ～ MWS Datasets 2018 ～, 情報処理学会, Vol.2018-CSEC-82, No.40, 2018 年 7 月.

[9] 佐藤 拓未: Word Vector を用いたマルウェアの 亜種判別法, 早稲田大学修士論文, 2017.

[10] U.Bayer, P.M.Comparetti, C.Hlauschek, C.Kruegel and E.Kirdam: “Scalable, Behavior-Based Malware Clustering.”, In Symposium on Network and Distributed System Security(NDSS), online PDF only, 2009.

付　　録

A.1 CEM アルゴリズムの導出 [4]

A.1.1 問題設定

N 個のデータを $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$，データの次元数を D とする．データ $\mathbf{x}$ を $\mathbf{y}$ と $\mathbf{z}$ に分解する．すなわち，

$$\mathbf{x} = \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} \tag{A.1}$$

とする．ここで，N 個のデータを K 個のガウス分布を持つ混合ガウス分布に $\mathbf{z}$ の条件下でクラスタリングする問題を考える．データ集合は第 n 行を $\mathbf{x}_n^{\mathrm{T}}$ とする $N \times D$ 行列 $\mathbf{X}$ で表される．同様に，$\mathbf{z}$ の次元数を $D_z (0 < D_z < D)$ とすると，条件は $\mathbf{z}_n^{\mathrm{T}}$ を行ベクトルとする $N \times D_z$ 行列 $\mathbf{Z}$ で表される．そして，$\mathbf{X}$ を次のように $\mathbf{Y}$ と $\mathbf{Z}$ で表せるよう

な $\mathbf{Y}$ を導入する．

$$\mathbf{X} = \begin{pmatrix} \mathbf{Y} & \mathbf{Z} \end{pmatrix} \tag{A.2}$$

これにより，$\mathbf{Y}$ の次元数は $D_y = D - D_z (0 < D_y < D)$ となる．また，求めたいパラメータを $\boldsymbol{\theta}$ とする．$\mathbf{y}, \mathbf{z}$ の周辺確率は以下のように計算できる．

$$p(\mathbf{y}, \mathbf{z}|\boldsymbol{\theta}) = \sum_{k=1}^{K} p(k, \mathbf{y}, \mathbf{z}|\boldsymbol{\theta}) \tag{A.3}$$

$p(k, \mathbf{y}, \mathbf{z}|\boldsymbol{\theta})$ はクラスタ k における $\mathbf{y}, \mathbf{z}$ の同時確率で，以下のように計算される．

$$p(k, \mathbf{y}, \mathbf{z}|\boldsymbol{\theta}) = p(k, \mathbf{z}|\boldsymbol{\theta}) p(k, \mathbf{y}|\mathbf{z}, \boldsymbol{\theta}) \tag{A.4}$$

また，$\mathbf{x}$ はガウス分布に従うため，式 (A.2) により $p(\mathbf{z}|\boldsymbol{\theta})$ と $p(\mathbf{y}|\mathbf{z}, \boldsymbol{\theta})$ は以下のような式になる．

$$p(\mathbf{z}|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}, \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}) \tag{A.5}$$

$$\begin{aligned} p(\mathbf{y}|\mathbf{z}, \boldsymbol{\theta}) = \mathcal{N}(&\mathbf{y}, \boldsymbol{\mu}_y + \boldsymbol{\Sigma}_{yz}\boldsymbol{\Sigma}_{zz}^{-1}(\mathbf{z} - \boldsymbol{\mu}_z), \\ &\boldsymbol{\Sigma}_{yy} - \boldsymbol{\Sigma}_{yz}\boldsymbol{\Sigma}_{zz}^{-1}\boldsymbol{\Sigma}_{zy}) \end{aligned} \tag{A.6}$$

ここで $\mathcal{N}(\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ は，確率変数 $\mathbf{z}$ についての平均 $\boldsymbol{\mu}$，共分散行列 $\boldsymbol{\Sigma}$ の確率密度関数を表す．また，$\boldsymbol{\mu}_y, \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{yy}, \boldsymbol{\Sigma}_{yz}, \boldsymbol{\Sigma}_{zy}, \boldsymbol{\Sigma}_{zz}$ は $\mathbf{x}$ に対応するガウス分布の平均 $\boldsymbol{\mu}$ と共分散行列 $\boldsymbol{\Sigma}$ を分解した要素に対応し，以下のように表される．

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_y \\ \boldsymbol{\mu}_z \end{pmatrix} \tag{A.7}$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{yy} & \boldsymbol{\Sigma}_{yz} \\ \boldsymbol{\Sigma}_{zy} & \boldsymbol{\Sigma}_{zz} \end{pmatrix} \tag{A.8}$$

したがって，$p(k, \mathbf{z}|\boldsymbol{\theta})$ と $p(k, \mathbf{y}|\mathbf{z}, \boldsymbol{\theta})$ は以下のように計算できる．

$$p(k, \mathbf{z}|\boldsymbol{\theta}) = \alpha_k \overline{\mathcal{N}}(\mathbf{z}, \boldsymbol{\mu}_z^k, \boldsymbol{\Sigma}_{zz}^k) \tag{A.9}$$

$$\begin{aligned} p(k, \mathbf{y}|\mathbf{z}, \boldsymbol{\theta}) &= \mathcal{N}(\mathbf{y}, \boldsymbol{\mu}_y^k + \boldsymbol{\Sigma}_{yz}^k (\boldsymbol{\Sigma}_{zz}^k)^{-1}(\mathbf{z} - \boldsymbol{\mu}_z^k), \\ &\qquad \boldsymbol{\Sigma}_{yy}^k - \boldsymbol{\Sigma}_{yz}^k (\boldsymbol{\Sigma}_{zz}^k)^{-1} \boldsymbol{\Sigma}_{zy}^k) \\ &= \mathcal{N}(\mathbf{y}, \boldsymbol{\nu}^k + \boldsymbol{\Gamma}^k \mathbf{z}, \boldsymbol{\Omega}^k) \end{aligned} \tag{A.10}$$

ここで，α_k は正規化係数で，$\overline{\mathcal{N}}(\mathbf{z}, \boldsymbol{\mu}_z^k, \boldsymbol{\Sigma}_{zz}^k) = \exp\left(-\frac{1}{2}(\mathbf{z} - \boldsymbol{\mu}_z^k)^{\mathrm{T}} (\boldsymbol{\Sigma}_{zz}^k)^{-1} (\mathbf{z} - \boldsymbol{\mu}_z^k)\right)$ とし，エキスパートパラメータと呼ばれる $(\boldsymbol{\nu}^k, \boldsymbol{\Gamma}^k, \boldsymbol{\Omega}^k)$ を導入した．また，$(\alpha, \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz})$ をゲートパラメータと呼ぶ．

A.1.2 CEM アルゴリズムによる最尤推定

EM アルゴリズムでは対数尤度関数 $\ln p(\mathbf{X}|\boldsymbol{\theta})$ を最大化するのに対して，CEM アルゴリズムでは増分条件対数尤度関数 $\boldsymbol{\Delta}\ell$ を最大化するのが目的である．$\boldsymbol{\Delta}\ell$ は以下のように定義される．

$$\boldsymbol{\Delta}\ell = \ln p(\mathbf{Y}|\mathbf{Z}, \boldsymbol{\theta}) - \ln p(\mathbf{Y}|\mathbf{Z}, \boldsymbol{\theta}^{\mathrm{old}}) \tag{A.11}$$

$\boldsymbol{\Delta}\ell$ を変形すると以下のようになる．

$$\begin{aligned} \boldsymbol{\Delta}\ell &= \ln p(\mathbf{Y}|\mathbf{Z}, \boldsymbol{\theta}) - \ln p(\mathbf{Y}|\mathbf{Z}, \boldsymbol{\theta}^{\mathrm{old}}) \\ &= \ln \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{z}_n, \boldsymbol{\theta}) - \ln \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{z}_n, \boldsymbol{\theta}^{\mathrm{old}}) \\ &= \sum_{n=1}^{N} \left(\ln \frac{p(\mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta})}{p(\mathbf{z}_n|\boldsymbol{\theta})} - \ln \frac{p(\mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})}{p(\mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \right) \\ &= \sum_{n=1}^{N} \left(\ln \frac{p(\mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta})}{p(\mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} - \ln \frac{p(\mathbf{z}_n|\boldsymbol{\theta})}{p(\mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \right) \\ &= \sum_{n=1}^{N} \left(\ln \frac{\sum_{k=1}^{K} p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta})}{\sum_{k=1}^{K} p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \right. \\ &\qquad \left. - \ln \frac{\sum_{k=1}^{K} p(k, \mathbf{z}_n|\boldsymbol{\theta})}{\sum_{k=1}^{K} p(k, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \right) \end{aligned} \tag{A.12}$$

式 (A.12) の右辺の第一項にイェンセンの不等式を適用すると

$$\begin{aligned} &\ln \frac{\sum_{k=1}^{K} p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta})}{\sum_{k=1}^{K} p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \\ &= \ln \sum_{k=1}^{K} \frac{p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta})}{\sum_{k=1}^{K} p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \\ &= \ln \sum_{k=1}^{K} h_{nk} \frac{p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta})}{p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \qquad \text{(A.13)} \\ &\geq \sum_{k=1}^{K} h_{nk} \ln \frac{p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta})}{p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \qquad \text{(A.14)} \end{aligned}$$

が得られる．ここで，

$$h_{nk} = \ln \frac{p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})}{\sum_{k=1}^{K} p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \tag{A.15}$$

とする．また，式 (A.12) の右辺の第二項に $\ln x \leq x - 1$ を適用し，式 (A.14) を利用すると $\boldsymbol{\Delta}\ell$ の下界である関数 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$ を求めることができる．

$$\begin{aligned} \boldsymbol{\Delta}\ell \geq \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}}) = \sum_{n=1}^{N} \left(\sum_{k=1}^{K} h_{nk} \ln \frac{p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta})}{p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \right. \\ \left. - \frac{\sum_{k=1}^{K} p(k, \mathbf{z}_n|\boldsymbol{\theta})}{\sum_{k=1}^{K} p(k, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} + 1 \right) \end{aligned} \tag{A.16}$$

この関数 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$ を書き換えると

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \{ h_{nk} (\ln p(k, \mathbf{y}_n|\mathbf{z}_n, \boldsymbol{\theta}) \\ + \ln p(k, \mathbf{z}_n|\boldsymbol{\theta}) - u_{nk}) - r_n p(k, \mathbf{z}_n|\boldsymbol{\theta}) + M^{-1} \} \end{aligned} \tag{A.17}$$

ここで，

$$u_{nk} = \ln p(k, \mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}}) \tag{A.18}$$

$$r_n = \frac{1}{\sum_{k=1}^{K} p(k, \mathbf{z}_n|\boldsymbol{\theta}^{\mathrm{old}})} \tag{A.19}$$

とする．この関数 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$ は $\boldsymbol{\Delta}\ell$ の下界であるため，これを最大化しパラメータを更新することで，$\boldsymbol{\Delta}\ell$ を徐々に増加させ最大化することができる．また，エキスパートパラメータ $(\boldsymbol{\nu}^k, \boldsymbol{\Gamma}^k, \boldsymbol{\Omega}^k)$ とゲートパラメータ $(\alpha, \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz})$ を独立とみなし，その順番でそれぞれ最適化し更新することを行う．

エキスパートパラメータの最適化では，関数 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$ を $\boldsymbol{\Phi}^k = \{\boldsymbol{\nu}^k, \boldsymbol{\Gamma}^k, \boldsymbol{\Omega}^k\}$ で微分し，0 とおいて解く．

$$\frac{\partial \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})}{\partial \boldsymbol{\Phi}^k} = \sum_{n=1}^{N} h_{nk} \frac{\partial \ln \mathcal{N}(\mathbf{y}_n, \boldsymbol{\nu}^k + \boldsymbol{\Gamma}^k \mathbf{z}_n, \boldsymbol{\Omega}^k)}{\partial \boldsymbol{\Phi}^k} = 0 \tag{A.20}$$

ゲートパラメータの最適化では，それぞれ $\alpha, \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}$ を求める．α_k を求めるには，関数 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$ を α_k で微分し 0 とおいて解けばよい．結果として α_k は以下のようになる．

$$\alpha_k = \frac{\sum_{n=1}^{N} h_{nk}}{\sum_{n=1}^{N} r_n \overline{\mathcal{N}}(\mathbf{z}_n, \boldsymbol{\mu}_z^k, \boldsymbol{\Sigma}_{zz}^k)} \tag{A.21}$$

しかし，$\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{zz}$ を求めるために関数 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$ を微分し，0 とおいて解くことは容易ではない．そのため，新しいアプローチが必要である．

まず，$\boldsymbol{\mu}_z$ を求める方法を述べる．関数 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$ を $\mathcal{Q}_{nk}$ の総和とみなし，以下のように二次関数を下界とすることができる．

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}}) &= \sum_{n=1}^{N} \sum_{k=1}^{K} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})_{nk} \\ &\geq \sum_{n=1}^{N} \sum_{k=1}^{K} b_{nk} - w_{nk} \left\| \boldsymbol{\mu}_z^k - \mathbf{c}_{nk} \right\|^2 \end{aligned} \tag{A.22}$$

ここで，$b_{nk}, w_{nk}, \mathbf{c}_{nk}$ は二次関数のそれぞれ最大値，開き具合，重心となる．白色化変換と呼ばれる，データの要素間の相関をなくすように変換することを行なった後，その二次関数が $\mathcal{Q}_{nk}$ と $\boldsymbol{\mu}_z^k = 0$ で接し，微分値が一致するという事実を用いることで，$w_{nk}, \mathbf{c}_{nk}$ は以下のように計算できる．（b_{nk} は省略する）

$$\mathbf{c}_{nk} = \frac{1}{2w_{nk}} \left(h_{nk} - r_n \alpha_k \mathrm{e}^{-\frac{1}{2}\mathbf{z}_n^{\mathrm{T}} \mathbf{z}_n} \right) \mathbf{z}_n \tag{A.23}$$

$$\begin{aligned} w_{nk} \geq\ & r_n \alpha_k \left(\mathrm{e}^{-\frac{1}{2}(\mathbf{z}_n - \boldsymbol{\mu}_z^k)^{\mathrm{T}} (\mathbf{z}_n - \boldsymbol{\mu}_z^k)} - \mathrm{e}^{-\frac{1}{2}\mathbf{z}_n^{\mathrm{T}} \mathbf{z}_n} \right. \\ & \left. - \mathrm{e}^{-\frac{1}{2}\mathbf{z}_n^{\mathrm{T}} \mathbf{z}_n} \mathbf{z}_n^{\mathrm{T}} \boldsymbol{\mu}_z^k \right) \left(\boldsymbol{\mu}_n^{k\,\mathrm{T}} \boldsymbol{\mu}_n^k \right)^{-1} + \frac{h_{nk}}{2} \end{aligned} \tag{A.24}$$

w_{nk} の最小値である w_{nk}^* を計算すると

$$\begin{aligned} w_{nk}^* &= r_n \alpha_k \max_c \left\{ \mathrm{e}^{-\frac{1}{2}\rho^2} \frac{\mathrm{e}^{-\frac{1}{2}c^2} \mathrm{e}^{c\rho} - c\rho - 1}{c^2} \right\} + \frac{h_{nk}}{2} \\ &= r_n \alpha_k \mathrm{e}^{-\frac{1}{2}\rho^2} f(\rho) + \frac{h_{nk}}{2} \end{aligned} \tag{A.25}$$

となる．ここで，$c^2 = \boldsymbol{\mu}_n^{k\,\mathrm{T}} \boldsymbol{\mu}_n^k, \rho^2 = \mathbf{z}_n^{\mathrm{T}} \mathbf{z}_n$ とする．また，$f(\rho)$ を毎回計算する必要はなく，予め計算しメモリに保持すれば計算時間を大幅に削減できる．$\boldsymbol{\mu}_z^k$ の更新式は以下のようになる．

$$\boldsymbol{\mu}_z^k = \frac{\sum_{n=1}^{N} w_{nk}^* \mathbf{c}_{nk}}{\sum_{n=1}^{N} w_{nk}^*} \tag{A.26}$$

また，これを逆白色化変換する必要がある．

次に，$\boldsymbol{\Sigma}_{zz}$ を求める方法について述べる．$\boldsymbol{\mu}_z$ の最適化では二次関数を下界とするのに対し，ここではガウス分布を下界とする．

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})_{nk} \geq b_{nk} - w_{nk} \mathbf{c}_{nk}^{\mathrm{T}} {\boldsymbol{\Sigma}_{zz}^k}^{-1} \mathbf{c}_{nk} - w_{nk} \ln |\boldsymbol{\Sigma}_{zz}^k| \tag{A.27}$$

$\boldsymbol{\mu}_z$ の最適化と同様に，データを白色化変換し，式 (A.27) の右辺が $\mathcal{Q}_{nk}$ と $\boldsymbol{\Sigma}_{zz}^k = 1$ で接し，微分値が一致することを用いて解くと以下の式が成り立つ．（ここでまた，$\rho^2 = \mathbf{z}_n^{\mathrm{T}} \mathbf{z}_n$ とする）

$$\mathbf{c}_{nk} \mathbf{c}_{nk}^{\mathrm{T}} = \frac{1}{2w_{nk}} \left(h_{nk} - r_n \alpha_k \mathrm{e}^{-\frac{1}{2}\rho^2} \right) \mathbf{z}_n \mathbf{z}_n^{\mathrm{T}} + I \tag{A.28}$$

$$\begin{aligned} w_{nk} \geq\ & r_n \alpha_k \left(\frac{1}{2} \mathrm{e}^{-\frac{1}{2}\rho^2} \rho^2 - \frac{1}{2} \mathrm{e}^{-\frac{1}{2}\rho^2} \mathbf{z}_n^{\mathrm{T}} {\boldsymbol{\Sigma}_{zz}^k}^{-1} \mathbf{z}_n \right. \\ & \left. + \mathrm{e}^{-\frac{1}{2}\rho^2} - \mathrm{e}^{-\frac{1}{2}\mathbf{z}_n^{\mathrm{T}} {\boldsymbol{\Sigma}_{zz}^k}^{-1} \mathbf{z}_n} \right) \left(\mathrm{Tr}(I) - \mathrm{Tr}({\boldsymbol{\Sigma}_{zz}^k}^{-1}) \right. \\ & \left. - \ln |{\boldsymbol{\Sigma}_{zz}^k}^{-1}| \right)^{-1} \end{aligned} \tag{A.29}$$

w_{nk} の最小値である w_{nk}^* を計算すると

$$\begin{aligned} w_{nk}^* &= r_n \alpha_k \max_c \left\{ \left(\frac{1}{2} \mathrm{e}^{-\frac{1}{2}\rho^2} (\mathrm{e}^{-\frac{1}{2}\rho^2 c} + \rho^2 - \rho^2 c + 2) \right. \right. \\ &\quad \left. \left. - \mathrm{e}^{-\frac{1}{2}\rho^2 c} \right) (1 - c - \ln |c|)^{-1} \right\} \\ &= r_n \alpha_k g(\rho) \end{aligned} \tag{A.30}$$

となる．ここで，また $\boldsymbol{\mu}_z$ の最適化と同様に，事前に $g(\rho)$ を計算しメモリに保持することで，計算時間の削減を図る．$\boldsymbol{\Sigma}_{zz}^k$ の更新式は以下のようになる．

$$\boldsymbol{\Sigma}_{zz}^k = \frac{\sum_{n=1}^{N} w_{nk}^* \mathbf{c}_{nk} \mathbf{c}_{nk}^{\mathrm{T}}}{\sum_{n=1}^{N} w_{nk}^*} \tag{A.31}$$

この値を逆白色化変換すると元の $\boldsymbol{\Sigma}_{zz}^k$ を計算できる．