【徹底解説】変分ベイズをはじめからていねいに

機械学習　2025年2月3日

本記事は機械学習の徹底解説シリーズに含まれます。

機械学習の記事一覧へ →

初学者の分かりやすさを優先するため，多少正確でない表現が混在することがあります。もし致命的な間違いがあればご指摘いただけると助かります。

目次

もっと見る

はじめに

機械学習を勉強したことのある方であれば，変分ベイズ（VB：variational bayes）の難しさには辟易したことがあるでしょう。私自身，学部生時代に意気揚々と機械学習のバイブルと言われている「パターン認識と機械学習（通称PRML）」を手に取って中身をペラペラめくってみたのですが，あまりの難しさから途方に暮れてしまったことを覚えています。

機械学習の登竜門は，変分ベイズ（変分推論）だと私は考えています。また，VAE（変分オートエンコーダ；variational autoencoder）に代表されるように，変分ベイズは最近の深層学習ブームにおいて理論面の立役者となっている側面もあります。一方で，多くの書籍やWeb上の資料では式変形の行間が詰まっていないことがあるため，初学者は必ず変分ベイズで躓くと言っても過言ではありません。

この問題を解決するため，本稿では変分ベイズをはじめからていねいに説明していきます。具体的には「**この解説を読んだだけで変分ベイズの概要と実際の応用方法が理解できる**」状態を目指します。多少記事が長くなってしまいますが，ゆっくり自分のペースで読み進めていけば，必ず変分ベイズを理解できるはずです。

一般に，難しい概念を噛み砕いて説明するときには，ボトムアップ的に必要となる知識を武装していく方法と，トップダウン的に求められる知識に寄り道していく方法があります。本稿では，両者を組み合わせて説明していきます。最初に，変分ベイズの目的をお伝えします。その上で，必要となる知識をボトムアップ的に積み上げていくという方針を採用します。

STEP 1
変分ベイズの目的

変分ベイズがどのような枠組みで何のために用いられるのかを説明します。

STEP 2
各種推定方法の実現

最尤推定・MAP推定・ベイズ推論を実現するための枠組みについて説明します。

STEP 3
混合ガウス分布への適用

混合ガウス分布を例にとってEMアルゴリズムと変分ベイズの使い方を確認します。

STEP 4
実装

pythonを使ってEMアルゴリズムと変分ベイズを実装します。

変分ベイズの目的

本章では，変分ベイズがどのような目的で用いられるのかを説明します。先に結論からお伝えすると，変分ベイズは**確率モデルの潜在変数・パラメータに関する事後分布を近似するため**の手法です。そこで，まず最初に確率モデルと事後分布に関する説明から始めていきます。

確率モデルというのは「現象の裏側に何か適当な分布を仮定する」枠組みのことです。私たちの目的は，ある現象を確率分布を用いて記述することです。そのためには，以下のステップが必要になります。

- ある現象をよく観察して最もよくフィットする既存の確率分布を選択する
- 仮定した確率分布の形状を決定するパラメータを推定する

ある現象に対して既存の分布を仮定するという操作は，観測データの発生方法に対して尤度関数を定めることに相当します。すると，私たちの最終的な目的は分布の形状を決定するパラメータを推定することになります。

> 尤度関数と同時にパラメータの情報を与える事前分布を設定すると，同時分布を定めることに相当します。ここでは，ある現象に対する既存の分布として尤度関数を定めていますが，ベイズ推定を行う場合には既存の分布として同時分布を定めます。

パラメータの推定方法は，パラメータを1つの値に決め切ってしまう点推定と，パラメータ自体にも既存の分布を仮定するベイズ推定に分けられます。さらに，点推定は尤度関数を最大にする最尤推定と事後確率を最大にする最大事後確率（MAP: maximum a posteriori）推定に分けられます。

- 点推定
  - 最尤推定
  - MAP推定

- ベイズ推定

以下では，数式を用いて点推定とベイズ推定を説明していきます。

点推定

あるパラメータθが与えられたときに，観測データXのふるまいを定義する関数$p(X|\theta)$が尤度関数です。最尤推定では，尤度関数$p(X|\theta)$を最大にするθの値を求めます。

$$\hat{\theta}_{\mathrm{ML}} = \arg\max_{\theta} p(X|\theta) \tag{1}$$

条件付き確率の定義より，$p(X|\theta)$は「θが与えられたときのX」の確率を表します。言い換えれば「θが与えられたときにXはどれだけ尤もらしいか」を表しています。最尤推定では，条件付けられたθを変数として読み替えることで，データXに対して最も尤もらしいパラメータθを推定します。

> $p(X|\theta)$を関数と呼ぶことに違和感がある人は鋭い視点をお持ちです。測度論を用いると，確率は可測空間上の確率測度（つまり関数）として定義されます。測度論を用いない場合の確率pの定義は，こちらからご確認ください。

一方で，MAP推定では事後確率$p(\theta|X)$を最大にするθの値を求めます。

$$\hat{\theta}_{\mathrm{MAP}} = \arg\max_{\theta} p(\theta|X) \tag{2}$$

$$= \arg\max_{\theta} \frac{p(X|\theta)p(\theta)}{p(X)} \tag{3}$$

$$= \arg\max_{\theta} p(X|\theta)p(\theta) \tag{4}$$

ただし，変形にはベイズの定理を利用しました。θに関する最大化問題であるため，最終行では$p(X)$を無視しました。

計算を簡単にするため，最尤推定やMAP推定では両辺の対数を取った対数尤度関数や対数事後確率を最大化することが多いです。対数関数は単調増加関数であるため，対数を取る前後の最大最小問題は等価になります。

$$\hat{\theta}_{\mathrm{ML}} = \arg\max_{\theta} \ln p(X|\theta) \tag{5}$$

$$\hat{\theta}_{\mathrm{MAP}} = \arg\max_{\theta} \{\ln p(X|\theta) + \ln p(\theta)\} \tag{6}$$

確率モデルをうまく設定できれば，これらの点推定は解析的に解くことも可能です。例えば，正規分布の形状を決定する母平均の最尤推定量は標本平均に一致し，母分散の最尤推定量は標本分散に一致します。これらはラグランジュの未定乗数法を用いて証明されます。

ベイズ推定

ベイズ推定では，点推定とは異なりパラメータを値ではなく分布として求めることで，データへの過学習を防止したり，表現力を上げたりすることができます。パラメータの分布というのは，ある観測データXが得られたときの事後分布$p(\theta|X)$のことを表しています。

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)} \tag{7}$$

点推定とは異なり，ベイズ推定では$\arg\max$が付いていませんね。$\arg\max$・$\arg\min$は目的関数が最大・最小になるパラメータを値として求めることを意味しています。耳が痛いようですが，ベイズ推定では仮定した確率分布pの形状を決めるパラメータθの事後分布$p(\theta|X)$を求めます。

> 点推定の限界として，対象とする現象が多峰性のふるまいを示しているケースが挙げられます。多峰性とは，複数のピークをもつ性質のことを指します。多峰性の現象に

対して点推定を行うと，1つのピーク以外の情報を完全に捨て去ってしまいます。

各種推定方法の実現

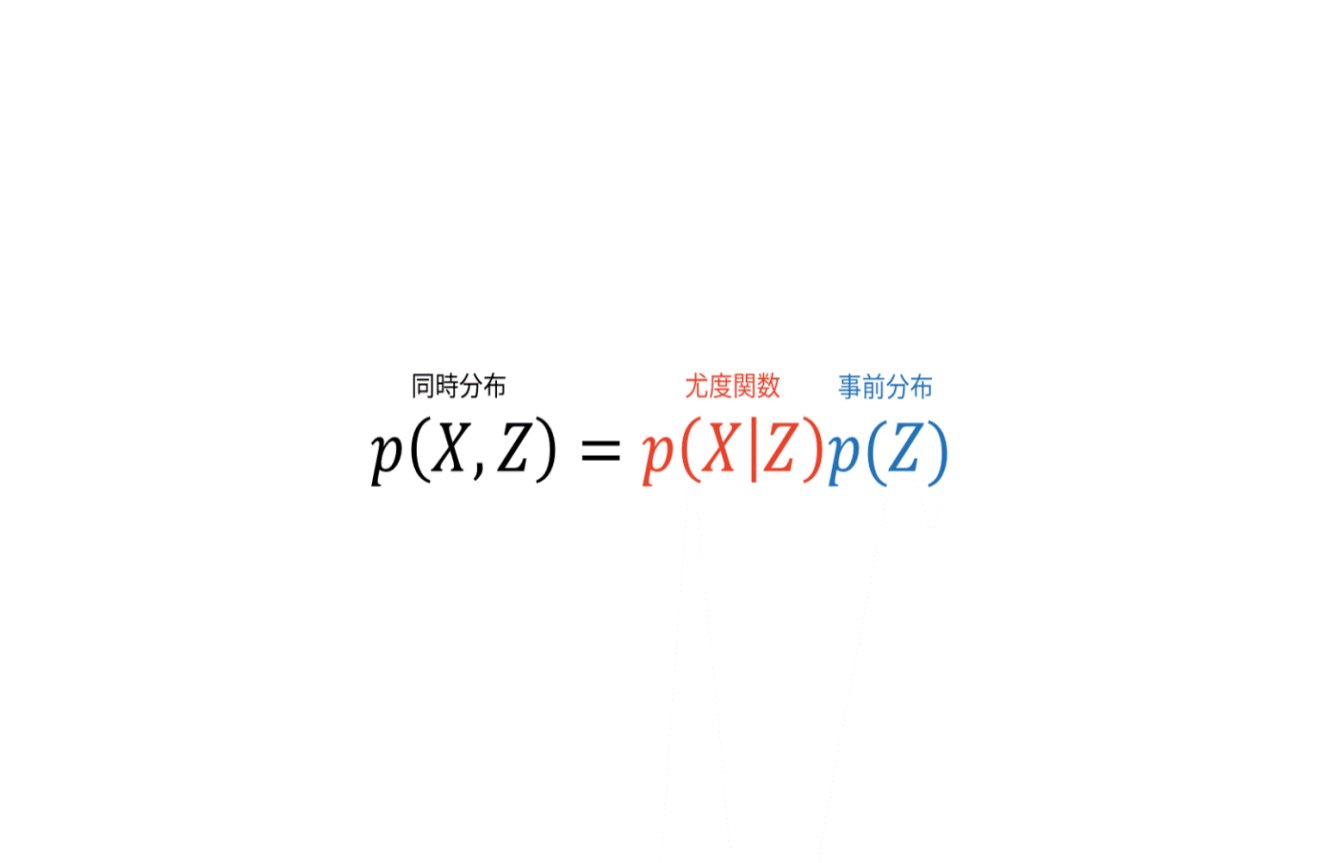
本章では，各種パラメータの推定方法がどのように実現されるのかについて説明します。先に結論からお伝えすると，点推定を実現する方法はEMアルゴリズム，ベイズ推定を実現する方法は変分ベイズがよく利用されます。

- 点推定
  - EMアルゴリズム
- ベイズ推定
  - 変分ベイズ（変分推論）

ちなみに，変分ベイズは文脈によっては「変分推論」とも呼ばれます。ベイズモデリングであることを強調したい場合は「変分ベイズ」と呼ばれることが多いようです。

確率モデルの問題を考えるときには，まず最初に確率モデルに潜在変数Z（尤度関数に姿を現さない変数）を導入します。そうすることで，複雑な分布をより単純な分布を使って表すことが可能になります。潜在変数を導入することは，同時分布$p(X, Z)$を設計することに相当します。ここが全ての始まりです。観測データXと潜在変数Zにどのような依存関係があるのかを同時分布として設定してしまうわけです。

ベイズ推定では，副次的に尤度関数$p(X|Z)$と事前分布$p(Z)$を仮定することになります。



同時分布の設定

尤度関数と事前分布を定めることは，同時分布を定めることだけでなく，事後分布の形を定めることにも相当します。なぜなら，Xが観測データであることから$p(X)$は定数だからです。

$$p(Z|X) = \frac{p(X,Z)}{p(X)} \tag{8}$$

$$\propto p(X,Z) \tag{9}$$

$$= p(X|Z)p(Z) \tag{10}$$

点推定では，事前分布が無情報（定数）だと仮定します。結局，上でお伝えしたように，点推定では同時分布を設計することは尤度関数を設定することに相当します。

$$p(X,Z) = p(X|Z)p(Z) \tag{11}$$

$$= p(X|Z) \cdot \text{const} \tag{12}$$

$$\propto p(X|Z) \tag{13}$$

ベイズ推定では，事前分布には共役事前分布を仮定するケースが多いです。共役事前分布を設定すると，事後分布は事前分布と同じ形になりますので，圧倒的に計算が簡単になります。一方で，全ての尤度関数に対して共役事前分布が存在する訳ではなく，以下の分布対が利用されることが多いです。

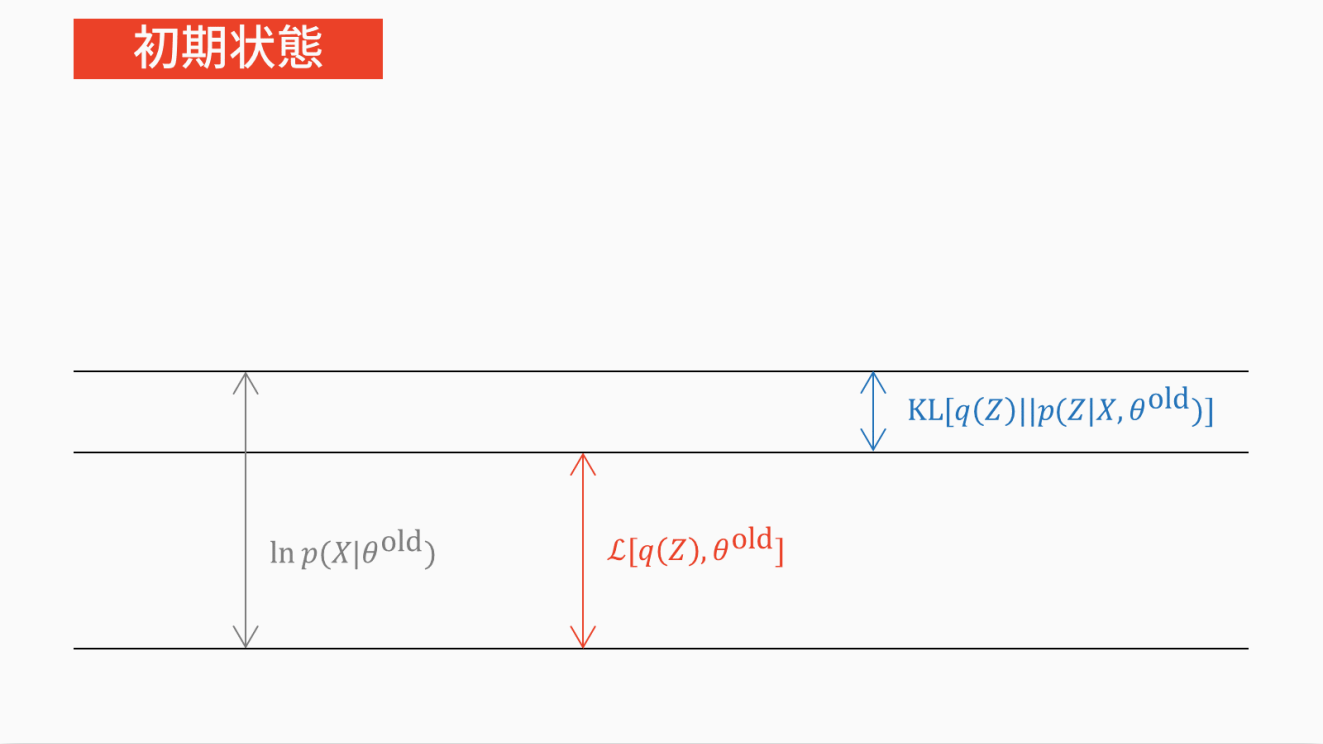
事後分布	尤度関数	共役事前分布
ガウス分布	ガウス分布 (平均)	ガウス分布
ウィシャート分布	ガウス分布 (精度行列)	ウィシャート分布
ディリクレ分布	多項分布	ディリクレ分布
ベータ分布	ベルヌーイ分布	ベータ分布
ベータ分布	二項分布	ベータ分布
ガンマ分布	ポアソン分布	ガンマ分布

代表的な共役事前分布

変分ベイズの本領が発揮されるのは，共役事前分布を定められないとき，もしくは事前分布が部分的な共役性しか持たないときです。今回扱う混合ガウス分布も，部分的な共役性をもつ確率モデルの一例ですので，変分ベイズの本領が発揮されることになります。

EMアルゴリズム

結論からお伝えすると，EMアルゴリズムは以下の流れで計算されます。



EMアルゴリズムのアニメーション

詳しい導出方法は「EMアルゴリズムをはじめからていねいに」をご参照下さい。

EMアルゴリズムをはじめからていねいに →

変分ベイズ

変分ベイズの目的は，確率モデルの潜在変数・パラメータに関する事後分布を求めることでした。EMアルゴリズムでは潜在変数をZ，パラメータをθと区別しましたが，変分ベイズでは両者を一括りにしてZと表しますので，事後分布は$p(Z|X)$と表されます。EMアルゴリズムでは$p(Z|X)$を計算できるという立場を取りましたが，変分ベイズでは$p(Z|X)$を計算できないという立場を取ります。

変分ベイズは，事後分布$p(Z|X)$を別の新しい分布$q(Z)$で近似してしまおうという大胆かつ汎用性の高い手法です。

$$q(Z) \sim p(Z|X) \tag{14}$$

この出発点こそ，変分ベイズが変分法の一種であることを裏付けています。変分法とは，ある関数の最適化問題を解く手法の一つです。今回は「真の事後分布を別の新しい分布$q(Z)$で近似する」という文脈で変分法を用いています。語弊を恐れずに言うならば，変分法は微分法の拡張です。微分法ではN次元ユークリッド空間における臨界点（微分が0になって関数が平らになる部分）を求めるのに対し，変分法では無限次元関数空間における臨界点を求めます。

結論からお伝えすると，変分ベイズでは以下の更新式を利用して潜在変数・パラメータに関する近似事後分布$q^*(Z)$を求めます。

$$\ln q_i^*(Z_i) = E_{j \neq i} \left[\ln p(X, Z) \right] + \text{const} \tag{15}$$

qとZの添え字であるiとjの意味については後ほどお伝えします。

更新式の導出には2通りの方法があります。1つ目は，事後分布とのKLダイバージェンスを最小化するような分布を求める方法です。2つ目は，EMアルゴリズムと同じ枠組みで周辺対数尤度関数を下限とKLダイバージェンスの２つに分解して考える方法です。前者は直感的に理解しやすい方法であり，後者はEMアルゴリズムと数学的な背景を一貫させて理解することができる方法です。それぞれ詳しく見ていきましょう。

KL最小化による近似

私たちの目的は，$p(Z|X)$をよく近似する$q(Z)$を求めることでした。2つの分布間の距離を測るオーソドックスな指標としては，KLダイバージェンスが利用できます。指標としてKLダイバージェンスを用いる理由は，必然性を理論的に説明できるからです。この必然性については次の「EMアルゴリズムの類推」で説明します。

2つの分布間の距離を測る指標としてKLダイバージェンスを採用することを受け入れると，$p(Z|X)$とのKLダイバージェンスが小さくなるような$q(Z)$を求めれば良いことになります。EMアルゴリズムでは，真の事後分布を計算できるという立場を取るため，KLダイバージェンスを0にする近似事後分布を求めることができました。一方で，変分ベイズでは真の事後分布を計算することができないという立場を取るため，近似分布が真の事後分布と厳密に等しくなることはないと仮定します。

しかし，何の制限もない中で$p(Z|X)$とのKLダイバージェンスが小さくなるような$q(Z)$を求めるのは自由度が高すぎて困難です。そこで，変分ベイズでは「平均場近似」と呼ばれる仮定を採用します。

$$q(Z) = \prod_{i=1}^{M} q_i(Z_i) \tag{16}$$

平均場近似は，事後分布$p(Z|X)$をよく近似する$q(Z)$は独立な分布$q_i(Z_i)$の積で表されるという強い仮定を表しています。ただし，必ずしも**全ての**要素が独立だという仮定ではないことに注意してください。あくまでも，どのような要素に分解するかの「グループ分け」が平均場近似の仮定だと認識しておきましょう。

すると，グループ分けした潜在変数・パラメータのインデックスをiとおいたときに，求める$q_i(Z_i)$というのは以下の解になります。ただし，$q_i(Z_i)$のことをq_iと省略して書きます。

$$\arg\min_{q_i} \mathrm{KL}\left[\prod_{i=1}^{M} q_i(Z_i) \middle\| p(Z|X)\right] \tag{17}$$

KLダイバージェンスは非対称ですので，$\mathrm{KL}(p||q)$と$\mathrm{KL}(q||p)$は異なります。前者をForward KL，後者をReverse KLと呼ぶことがあります。KLダイバージェンスの定義より，Forward KLは$\log$関数をpで重みづけしていますので，$p \neq 0$の部分をqで網羅しようとします。その結果，qの分散は大きくなりやすいです。一方で，Reverse KLは$\log$関数をqで重みづけしていますので，qは0ではない部分でpを網羅しようとします。その結果，qの分散は小さくなりやすいです。例えば，真の事後分布が多峰性の場合，Forward KLでは複数のピークをならしたような近似事後分布が得られるのに対し，Reverse KLではある1つのピークに着目した近似事後分布が得られやすいです。

多峰性の場合は近似事後分布に混合ガウス分布を持ち出せばうまくフィッティングできますが，KLダイバージェンスの非対称性を把握しておくことは非常に大切です。

実際に計算していきます。KLダイバージェンスの定義より，以下のように計算することができます。

$$q_i(Z_i) = \arg\min_{q_i} \mathrm{KL}\left[\prod_{i=1}^{M} q_i(Z_i) \middle\| p(Z|X)\right] \tag{18}$$

$$= \arg\min_{q_i} E_q\left[\ln \frac{\prod_{i=1}^{M} q_i(Z_i)}{p(Z|X)}\right] \tag{19}$$

今求めたいのはZ_iに対する近似事後分布ですので，期待値を取る際には$q_i(Z_i)$に対する期待値と$q_{j\neq i}(Z_{j\neq i})$に対する期待値を分けて考えてあげます。$q_{j\neq i}$は「iとは異なるj」と読むと理解しやすいです。

$$\begin{aligned} q_i(Z_i) &= \arg\min_{q_i} E_q\left[\ln \frac{q_i(Z_i)q_{j\neq i}(Z_{j\neq i})}{p(Z|X)}\right] \\ &= \arg\min_{q_i} E_{q_i}\left[E_{q_{j\neq i}}\left[\ln \frac{q_i(Z_i)q_{j\neq i}(Z_{j\neq i})}{p(Z|X)}\right]\right] \\ &= \arg\min_{q_i} \Big\{ E_{q_i}\left[E_{q_{j\neq i}}\left[\ln q_i(Z_i)\right] + E_{q_{j\neq i}}\left[\ln q_{j\neq i}(Z_{j\neq i})\right] - E_{q_{j\neq i}}\left[\ln p(Z|\right. \end{aligned}$$

$q(Z)$をiと$j\neq i$に分けるというアイディアが少し奇抜に感じられるかもしれません。これは後に，KLダイバージェンスを0にする$q(Z)$を見つけるための苦肉の策です。平均場近似は強い仮定であることから「各因子ごとであればKLダイバージェンスを0にする事後分布を計算できるのではないか」というアイディアに着想した結果です。実際に計算を進めていくと，式(32)でKLダイバージェンスを0にする$q_i(Z_i)$を導出できることが分かります。

さて，ここで期待値に関する以下の性質を利用します。

- 1の期待値は1

$$E_X[1] = 1 \tag{23}$$

- 期待値の対象と中身が独立な（定数とみなせる）場合に期待値の外に出せる [参考]

$$E_X[aX + b] = aE_X[X] + b \tag{24}$$

- 確率変数の過不足なく期待値を取った結果はスカラー

$$E_Y[E_X[XY]] = E_Y[\mu_x Y] = \mu_x E_Y[Y] = \mu_x \mu_y \tag{25}$$

最後の項目について少し補足しておきます。期待値の定義は**注目している確率変数に対する周辺化操作**に相当しますので，確率変数に過不足なく期待値を取った結果はスカラーになります。

以上を踏まえると，以下が成り立ちます。

$$E_{q_{j\neq i}}[\ln q_i(Z_i)] = \ln q_i(Z_i) \tag{26}$$

$$E_{q_{j\neq i}}[\ln q_{j\neq i}(Z_{j\neq i})] = \text{const} \tag{27}$$

したがって，先ほどの計算を進めることができます。

$$q_i(Z_i) = \arg\min_{q_i} \left\{ E_{q_i}\left[\ln q_i(Z_i) + \text{const} - E_{q_{j\neq i}}\left[\ln p(Z|X)\right]\right] \right\} \tag{28}$$

$$= \arg\min_{q_i} \left\{ E_{q_i}\left[\ln q_i(Z_i) - E_{q_{j\neq i}}\left[\ln p(Z|X)\right]\right] \right\} \tag{29}$$

$$= \arg\min_{q_i} \left\{ E_{q_i}\left[\ln \frac{q_i(Z_i)}{\exp\left(E_{q_{j\neq i}}\left[\ln p(Z|X)\right]\right)}\right] \right\} \tag{30}$$

$$= \arg\min_{q_i} \left\{ E_{q_i}\left[\ln \frac{q_i(Z_i)}{\exp\left(E_{q_{j\neq i}}\left[\ln p(Z|X)\right]\right)/C} - \ln C\right] \right\} \tag{31}$$

$$= \arg\min_{q_i} \mathrm{KL}\left[q_i(Z_i) \middle\| \frac{\exp\left(E_{q_{j\neq i}}\left[\ln p(Z|X)\right]\right)}{C}\right] \tag{32}$$

KLダイバージェンスを最小にするのは2つの分布が等しいときですので，結局以下が得られます。

$$q_i(Z_i) = \frac{\exp\left(E_{q_{j\neq i}}\left[\ln p(Z|X)\right]\right)}{C} \tag{33}$$

両辺の対数を取ると，冒頭で説明した公式(15)が得られます。

$$\ln q_i(Z_i) = E_{q_{j\neq i}}\left[\ln p(Z|X)\right] + \text{const} \tag{34}$$

$$= E_{q_{j\neq i}}\left[\ln \frac{p(X,Z)}{p(X)}\right] + \text{const} \tag{35}$$

$$= E_{q_{j\neq i}}\left[\ln p(X,Z)\right] + \text{const} \tag{36}$$

「自分以外全ての潜在変数・パラメータで仮定した確率モデルの期待値を取ると近似事後分布の形が得られる」と理解しましょう。冒頭にもお伝えした通り，我々の目標は得られたデータXの背後に潜む事後分布$p(Z|X)$の形を求めることでした。確率モデルの問題では，確率変数同士の依存関係を同時分布$p(X,Z)$としてこちら側で定めてしまうのでした。ゆえに，式(15)の右辺を計算することができます。

EMアルゴリズムの類推

ここまでの方針は，$p(Z|X)$を良く表す近似事後分布$q(Z)$を，KLダイバージェンスという恣意的な指標を用いて測りました。そこで，ここからはKLダイバージェンスが出てくる必然性を説明していきます。

思い出して欲しいのは，EMアルゴリズムにおける下限の導出過程です。下限と対数尤度関数の差は，KLダイバージェンスの形になりましたね。この経験から類推するに，変分ベイズでも目的関数に対してイェンセンの不等式を適用すれば，KLダイバージェンスが出現するはずです。

さて，変分ベイズの目的関数というのは一体何なのでしょうか。EMアルゴリズムの目的は対数尤度関数の最大化でしたので，目的関数が自明でした。しかし，変分ベイズの目的は事後分布pをよく表す近似事後分布qを見つけることでしたので，目的関数は自明ではありません。目的ドリブンで考えると，変分ベイズの目的関数がqとpのKLダイバージェンスになるというのは「距離指標としてKLダイバージェンスを採用する必然性が（今のところ）ないこと」を除いて理にかなっています。

こういうときは，出発点に立ち返ることが大切です。私たちの目的は，ある現象を確率分布を用いて記述することです。そのためには，以下のステップが必要になるのでした。

- ある現象をよく観察して最もよくフィットする既存の確率分布を選択する
- 仮定した確率分布の形状を決定するパラメータを推定する

点推定では「パラメータの値」自体に興味があるため，対数尤度関数をパラメータの関数と読み替えて最大化問題を解きました。その際，潜在変数の出現による和の対数部分が計算困難であるため，イェンセンの不等式を利用して対数尤度関数を下から評価したのでした。

$$\ln p(X|\theta) = \mathcal{L}_{\mathrm{ML}}\left[q(Z), \theta\right] + \mathrm{KL}\left[q(Z)||p(Z|X,\theta)\right] \tag{37}$$

変分ベイズでは，パラメータと潜在変数を同一視しますので，θは$q(Z)$に吸収されます。つまり，式(37)の両辺におけるθを単に消去して，$q(Z)$の中にθを含めてあげればよいのです。

$$\ln p(X) = \mathcal{L}_{\mathrm{VB}}\left[q(Z)\right] + \mathrm{KL}\left[q(Z)\|p(Z|X)\right] \tag{38}$$

なぜパラメータを潜在変数に含めるのかについては，変分ベイズではパラメータの値自体には興味がなく，パラメータの分布に興味があるからです。興味の対象はθではなくqになりますので，θをZに含めてしまってqだけに注目するように仕向けたのです。

すると，変分ベイズの目的関数は式(38)の左辺，すなわち周辺対数尤度関数であることが分かりました。ここで注意するべきなのは，Xは観測データですので周辺対数尤度関数は定数であるということです。この事実は変分ベイズ法の解説でよく誤解されている点ですので，おさえておくと良いと思います。

このことから，周辺対数尤度関数を目的関数に掲げるのは不適切といえます。そこで，式(38)の右辺に注目しましょう。対数周辺尤度関数の下限とKLダイバージェンスから構成されていますね。何を隠そう，第二項目のKLダイバージェンスは前の方針で考えたqとpの距離を測る指標そのものです。したがって，変分ベイズの目的関数はqとpのKLダイバージェンスに帰着します。

先ほど「距離指標としてKLダイバージェンスを採用する必然性が（今のところ）ない」とお伝えしましたが，KLダイバージェンスを用いる必然性は，目的関数の下限をイェンセンの不等式で評価していたことに裏付けられているのです。イェンセンの不等式を用いると言う前提に立つ場合には，KLダイバージェンスを用いる必然性は担保されるということです。

ここまでをまとめます。変分ベイズの目的関数は対数周辺尤度関数ですが，一定値であるため目的関数としては不適切です。そこで，対数周辺尤度関数を構成する2つの項に注目すると，pをよく近似するqを見つけるためにKLダイバージェンスを目的関数に設定すれば良いことが分かりました。

一方で，式(38)の右辺の和が一定値であることに注意すると，KLダイバージェンスを最小化することは下限を最大化することと等価です。したがって，変分ベイズでは**2つの等価な目的関数が存在する**ことになります。ただし，KLダイバージェンスでは最小化問題，下限では最大化問題を考えるという点に注意してください。

最後に，EMアルゴリズムと変分ベイズの比較表を載せておきます。

EMアルゴリズムと変分ベイズの比較

$\mathcal{L}_{\mathrm{VB}}[q(Z)]$のことをELBO（evidence lower bound）や変分下限（variational lower bound: VLB）と呼びます。先ほどお伝えしたように，lower boundは下界ですから，VLBは正確には変分下界と訳されるべきですが，変分ベイズでは専らイェンセンの不等式により下界を導出することから，下界を下限と呼ぶようになったと推察されます。

最尤推定とMAP推定とのつながり

本章では，変分ベイズの最尤推定とMAP推定とのつながりを説明していきましょう。結論からお伝えすると，最尤推定とMAP推定は，変分ベイズの特殊な場合に相当します。このような文脈で，変分ベイズは点推定の上位互換の概念であるといえるでしょう。KLダイバージェンスを目的関数と捉えた場合でも，下限を目的関数と捉えた場合でも，最尤推定とMAP推定とのつながりを美しく説明することができます。

変分ベイズがどのような場合でも必ず点推定よりも優れているという訳ではありません。上位互換というのはあくまでも概念としての包含関係の話であって，パラメータ推定の性能に関する話ではありません。実際には，与えられたデータ量や利用できるリソースに応じて，点推定とベイズ推定のメリットとデメリットを比較しながら臨機応変に使い分ける必要があります。

KL最小化

ベイズ推論ではパラメータの近似事後分布を求めるのでした。点推定ではパラメータの値を求めるのでした。そこで，ベイズ推論における近似事後分布を「一点にしか値を持たない」関数とすれば，ベイズ推論は点推定と等価になります。数学の世界では「一点にしか値を持たない」関数としてディラックのデルタ関数が有名です。

ディラックのデルタ関数

任意の実連続関数$f:\mathbb{R}\to\mathbb{R}$に対し，

$$\int_{-\infty}^{\infty} f(x)\delta(x)dx = f(0) \tag{39}$$

を満たす実数値シュワルツ超関数δをディラックのデルタ関数と呼ぶ。

例えば，ディラックのデルタ関数をaだけ平行移動させると，以下のように$f(a)$が抽出されます。

$$\int_{-\infty}^{\infty} f(x)\delta(x-a)dx = f(a) \tag{40}$$

したがって，パラメータを含む潜在変数の最適解を$\hat{Z}$とおくと，近似事後分布にディラックのデルタ関数$\delta(Z-\hat{Z})$を仮定すればよいことが分かります。

$$\begin{aligned}
&\mathrm{KL}\left[\delta\left(Z-\hat{Z}\right)\middle\|p(Z|X)\right] \\
&=\int \delta\left(Z-\hat{Z}\right)\ln\frac{\delta(Z-\hat{Z})}{p(Z|X)}dZ && (41)\\
&=\int \delta\left(Z-\hat{Z}\right)\ln\delta\left(Z-\hat{Z}\right)dZ-\int \delta\left(Z-\hat{Z}\right)\ln p(Z|X)dZ && (42)\\
&=\ln\delta\left(\hat{Z}-\hat{Z}\right)-\ln p(\hat{Z}|X) && (43)\\
&=\ln\delta(0)-\ln p(\hat{Z}|X) && (44)
\end{aligned}$$

結局，最適解$\hat{Z}$は以下のように表されます。$\ln \delta(0)$は定義されませんが，$\hat{Z}$とは関係ないため無視できる点がポイントです。

$$\arg\min_{\hat{Z}} \mathrm{KL}\left[\delta\left(Z-\hat{Z}\right)\middle\|p(Z|X)\right] = \arg\min_{\hat{Z}} \left\{\ln\delta(0) - \ln p(\hat{Z}|X)\right\} \tag{45}$$

$$= \arg\max_{\hat{Z}} \ln p(\hat{Z}|X) \tag{46}$$

$$= \arg\max_{\hat{Z}} p(\hat{Z}|X) \tag{47}$$

これはMAP推定そのものを表していますよね。美しいです。さらに，MAP推定において事前分布が一様（つまりパラメータに関する情報が得られない）と仮定したケースは，最尤推定に相当するはずです。そこで，式(47)において事前分布として定数$p(\hat{Z}) = C_{\mathrm{ML}}$を仮定してみましょう。

$$\begin{aligned}
\arg\min_{\hat{Z}} \mathrm{KL}\left[\delta\left(Z-\hat{Z}\right)\middle\|p(Z|X)\right] &= \arg\max_{\hat{Z}} \ln p(\hat{Z}|X) \\
&= \arg\max_{\hat{Z}} \ln \frac{p(X|\hat{Z})p(\hat{Z})}{p(X)} \\
&= \arg\max_{\hat{Z}} \left\{\ln p(X|\hat{Z}) + \ln p(\hat{Z}) - \ln p(X\right. \\
&= \arg\max_{\hat{Z}} \left\{\ln p(X|\hat{Z}) + C_{\mathrm{ML}} - \ln p(X)\right\} \\
&= \arg\max_{\hat{Z}} \ln p(X|\hat{Z}) \\
&= \arg\max_{\hat{Z}} p(X|\hat{Z})
\end{aligned}$$

これは最尤推定そのものを表していますよね。美しいです。ここまでの流れをまとめておきましょう。

変分ベイズのKL最小化の文脈において，以下の仮定と推定方法が対応する。

- 近似事後分布にディラックのデルタ関数を仮定
  - MAP推定に相当
- 近似事後分布にディラックのデルタ関数を仮定かつ事前分布に定数を仮定
  - 最尤推定に相当

シュワルツ超関数は単に超関数とも呼ばれていて，関数を拡張した概念です。ディラックのデルタ関数は元々，空間の一点にだけ存在する粒子を表現するために考案された関数です。上の定義の通り，積分して初めて意味をもつことから，一般的な関数とは異なる超関数として定義されています。

下限最大化

下限最大化の文脈においては，KL最小化の文脈よりもシンプルにMAP推定と最尤推定との繋がりを示すことができます。EMアルゴリズムにおける下限$\mathcal{L}_{\mathrm{ML}}$の定義より，変分ベイズにおける下限$\mathcal{L}_{\mathrm{VB}}$は以下のように計算されます。ただし，変分ベイズでは潜在変数を連続値として扱うため，シグマを積分で置き換えています。

$$\mathcal{L}_{\mathrm{VB}}\left[q(Z)\right] = \int_Z q(Z) \ln \frac{p(X,Z)}{q(Z)} dZ \tag{54}$$

$$= \int_Z q(Z) \ln \frac{p(X|Z)p(Z)}{q(Z)} dZ \tag{55}$$

$$= \int_Z q(Z) \ln p(X|Z) dZ - \int_Z q(Z) \ln \frac{q(Z)}{p(Z)} dZ \tag{56}$$

$$= \int_Z q(Z) \ln p(X|Z) dZ - \mathrm{KL}\left[q(Z)||p(Z)\right] \tag{57}$$

変分ベイズでは，この下限を最大化していくのでした。したがって，変分ベイズでは式(57)の第一項目の下限は最大化され，第二項目のKLダイバージェンスは最小化されます。同時に二つの項を考えると大変ですので，それぞれを最大化する場合について考えてみましょう。

第一項目の最大化は対数尤度に対応しますので，第一項目だけを考えると最尤推定に相当することが分かります。実際，第一項目を最大にするqは，対数尤度を最大にする$\hat{Z}$で値をもつディラックのデルタ関数$\delta(Z-\hat{Z})$になります。なぜなら，期待値は分布の平均に相当する概念を表しますが，期待値を最大にするためには分布の最大値を抽出すれば良いからです。この結果は，KL最小化文脈の考察と一致します。

第二項目は負のKLダイバージェンスに対応しますので，第二項目だけを考えると「事後分布$q(Z)$を事前分布$p(Z)$に近づける」ことに相当することが分かります。これは，MAP推定における事前分布の正則化項に対応しており，KL最小化文脈の考察と一致します。

結局，下限を二つの項に分解すると以下のような結果になることが分かりました。用語に数学的な厳密性は担保していません。

$$\begin{aligned} \text{下限} &= q\text{による対数尤度の期待値} + \text{近似分布と事前分布の負の距離} && (58)\\ &= \text{最尤推定項} + \text{正則化項} && (59) \end{aligned}$$

このように，下限を最大化することは，第1項目を大きくする（最尤推定の効果を大きくする）か，第2項目を大きくする（正則化の効果を大きくする）かのバランスを取っていることが分かりました。以上をまとめます。

> 変分ベイズの下限最大化の文脈において，下限は以下のように分解される。
>
> $$\text{下限} = \text{最尤推定項} + \text{正則化項} \qquad (60)$$

混合ガウス分布への適用

本章では，混合ガウス分布（GMM：Gaussian Mixture Model）を題材に，EMアルゴリズムと変分ベイズを応用する方法をお伝えしていきます。

EMアルゴリズム

結論からお伝えすると，EMアルゴリズムのGMMへの適用は以下のようにまとめられます。

EMアルゴリズムのGMMへの適用（GMM-EM）は，以下の4ステップから構成される。

1. 初期値設定
2. 負担率r_{nk}と$\mathcal{Q}$関数の計算（Eステップ）
3. 各パラメータの更新（Mステップ）
4. 収束判定

まずは各パラメータに初期値$\boldsymbol{\mu}_0$，$\boldsymbol{\Sigma}_0$，π_0を与える。次に，以下のEステップとMステップを収束するまで繰り返す。

Eステップ

以下の負担率r_{nk}と$\mathcal{Q}$関数を計算する。

$$r_{nk} = \frac{\pi_k \mathrm{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathrm{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{6}$$

$$\mathcal{Q}(\theta) = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\left[-\frac{1}{2}\left\{\ln|\boldsymbol{\Sigma}_k| - (\mathbf{x}_n - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)\right\}\right] \tag{6}$$

Mステップ

$\mathcal{Q}$関数を最大にするように以下のパラメータを計算する。

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{63}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\mathbf{x}_n}{N_k} \tag{64}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T}{N_k} \tag{65}$$

$$\pi_k = \frac{N_k}{N} \tag{66}$$

詳しい導出方法は「EMアルゴリズムをはじめからていねいに」をご参照下さい。

EMアルゴリズムをはじめからていねいに →

変分ベイズ

本節では，以下の流れで変分ベイズをGMMのパラメータ推論に適用していきます（GMM-VB）。

1. 同時分布の依存関係確認
2. 平均場近似のグループ分けの仮定
3. 更新式適用による変分下限の最大化

1. 同時分布の依存関係確認

本節では，潜在変数とパラメータの依存関係をグラフィカルモデルを通して確認します。GMMのパラメータは，潜在変数も含めると，$\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$です。今回仮定する依存関係は，

一般的によく利用されるものです。以下にグラフィカルモデルを示します。

GMM-VBのグラフィカルモデル

これを数式を用いて表すと，以下のようになります。

$$p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Sigma})p(\mathbf{Z}|\boldsymbol{\pi})p(\boldsymbol{\pi})p(\boldsymbol{\mu}|\boldsymbol{\Sigma})p(\boldsymbol{\Sigma}) \tag{67}$$

今回は，各分布を以下のように定めます。

$$p(\mathbf{X}|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\mathrm{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k^{-1})^{z_{nk}} \tag{68}$$

$$p(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\pi_k^{z_{nk}} \tag{69}$$

$$p(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_0) \tag{70}$$

$$p(\boldsymbol{\mu}|\boldsymbol{\Sigma}) = \prod_{k=1}^{K}\mathrm{N}\left\{\boldsymbol{\mu}_k|\mathbf{m}_0,(\beta_0\boldsymbol{\Sigma}_k)^{-1}\right\} \tag{71}$$

$$p(\boldsymbol{\Sigma}) = \prod_{k=1}^{K}\mathrm{W}(\boldsymbol{\Sigma}_k|\mathbf{W}_0,\nu_0) \tag{72}$$

EMアルゴリズムとの対応を図るために，$\mathbf{Z}$の周辺尤度関数も確認しておきます。

$$\begin{aligned}
p(\mathbf{X}|\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}) &= \sum_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}|\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}) && (73)\\
&= \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Sigma})\cdot p(\mathbf{Z}|\boldsymbol{\pi}) && (74)\\
&= \sum_{\mathbf{Z}}\left[\prod_{n=1}^{N}\prod_{k=1}^{K}\left\{\pi_k\mathrm{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k^{-1})\right\}^{z_{nk}}\right] && (75)\\
&= \prod_{n=1}^{N}\sum_{k=1}^{K}\pi_k\mathrm{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k^{-1}) && (76)
\end{aligned}$$

ただし，最終行は「one-hot-vector$\mathbf{Z}$の全ての候補について足し合わせる」操作を「全てのクラスkについて和をとる」操作に読み替えて変形しました。この結果は，GMM-EMにおける以下の周辺尤度関数と矛盾しません。

$$p(\mathbf{x}_n|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sum_{k=1}^{K}\pi_k\mathrm{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \tag{77}$$

繰り返しにはなりますが，変分ベイズはEMアルゴリズムとは異なりベイズ推定を行いますので，尤度関数に加えて$\boldsymbol{\pi}$，$\boldsymbol{\mu}$，$\boldsymbol{\Sigma}$の事前分布を設定します。

> 分散共分散行列に関するガウス分布の共役事前分布は逆ウィシャート分布になります。一方で，分散共分散行列の逆行列に関するガウス分布の共役事前分布はウィシャート分布になります。今回は，素朴なウィシャート分布を持ち出すために，ガウス分布の分散共分散行列を逆行列で定めます。なお，分散共分散行列の逆行列のことを精度行列（precision matrix）と呼びます。

以下では，簡単に尤度関数と事前分布の根拠を説明しておきます。

EMアルゴリズムと同様に，$\mathbf{Z}$の条件付き分布は$\mathbf{z}_n$と$\boldsymbol{\pi}$の性質から自然に導かれる関係です。$\mathbf{z}_n$はone-hot-vectorですので，$\mathbf{z}_n$の生成確率をπ_kを利用して表現すれば，上のように累乗と総乗で表すことができるのです。データの条件付き確率も同様に，$\mathbf{z}_n$がone-hot-vectorであることを利用して表現できます。

$\boldsymbol{\pi}$の事前分布がディリクレ分布であるのは，$p(\mathbf{Z}|\boldsymbol{\pi})$が多項分布の形（$n=1$のカテゴリカル分布）をしており，その共役事前分布として定めているからです。同様に，多変量正規分布$p(\mathbf{X}|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Sigma})$の$\boldsymbol{\mu}$に関する共役事前分布としてガウス分布，精度行列（分散共分散行列の逆数）に関する共役事前分布としてウィシャート分布を設定しています。

> 多くの書籍やWeb上の資料では「$p(\boldsymbol{\mu},\boldsymbol{\Sigma})$の共役事前分布としてガウスウィシャート分布を仮定する」と説明されています。しかし，本稿では上で示したグラフィカルモデルに基づいた説明を心掛けています。すなわち，まず$\boldsymbol{\Sigma}$が生成されて，その$\boldsymbol{\Sigma}$に依存して$\boldsymbol{\mu}$が生成されるというフローを忠実に再現するために，$\boldsymbol{\mu}$に関する共役事前分布と$\boldsymbol{\Sigma}$に関する共役事前分布を別々に説明しました。

以下では，これらの分布を使って計算を行うために定義を確認しておきましょう。具体的には，ディリクレ分布，多変量正規分布，ウィシャート分布の確率密度関数と期待値を知っておく必要があります。ディリクレ分布とウィシャート分布に関しては，後で$E[\ln x]$を利用することになるため，ここで一緒に確認しておきましょう。

ディリクレ分布の確率密度関数・$E[X]$・$E[\ln x]$は以下のように表されます。

ディリクレ分布

$$f_{\mathbf{x}}(\mathbf{x}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{n} x_i^{\alpha_i - 1} \tag{78}$$

$$E[X_i] = \frac{\alpha_i}{\sum_{i=1}^{n} \alpha_i} \tag{79}$$

$$E[\ln X_i] = \psi(\alpha_i) - \psi\left(\sum_{i=1}^{n} \alpha_i\right) \tag{80}$$

ただし，$B(\cdot)$はベータ関数，$\psi(\cdot)$はディガンマ関数を表す。

$$B(a, b) = \int_0^1 x^{a-1}(1-x)^{b-1} dx \tag{81}$$

$$\psi(a) = \frac{d}{da} \ln \Gamma(a) \tag{82}$$

多変量正規分布の確率密度関数・$E[X]$は以下のように表されます。

多変量正規分布

$$f_{\mathbf{x}}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right\} \tag{83}$$

$$E[\mathbf{X}] = \boldsymbol{\mu} \tag{84}$$

ウィシャート分布の確率密度関数・$E[X]$・$E[\ln x]$は以下のように表されます。

ウィシャート分布

$$\mathrm{W}(\mathbf{\Sigma}|\mathbf{W},\nu) = C(\mathbf{W},\nu)|\Sigma|^{(\nu-D-1)/2}\exp\left(-\frac{1}{2}\mathrm{Tr}\left[\mathbf{W}^{-1}\mathbf{\Sigma}\right]\right) \tag{85}$$

$$E[\mathbf{\Sigma}] = \nu\mathbf{W} \tag{86}$$

$$E[\ln|\mathbf{\Sigma}|] = \sum_{i=1}^{D}\psi\left(\frac{\nu+1-i}{2}\right) + D\ln 2 + \ln|\boldsymbol{W}| \tag{87}$$

ただし，$C(\cdot,\cdot)$は以下で表される定数，$\psi(\cdot)$はディガンマ関数である。

$$C(\mathbf{W},\nu) = |\mathbf{W}|^{-\nu/2}\left\{2^{\nu D/2}\pi^{D(D-1)/4}\prod_{i=1}^{D}\Gamma\left(\frac{\nu+1-i}{2}\right)\right\}^{-1} \tag{88}$$

$$\psi(a) = \frac{d}{da}\ln\Gamma(a) \tag{89}$$

特に，$\mathbf{W}$が対称行列であることに注意されたい。

分布を利用する準備が整いました。これにて「1. 同時分布の依存関係の確認」が完了しました。

2. 平均場近似のグループ分けの仮定

本節では，平均場近似のグループ分けを考えていきます。今回は，潜在変数と他のパラメータが独立であることを仮定しましょう。

$$q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\mathbf{\Sigma}) = q(\mathbf{Z})q(\boldsymbol{\pi},\boldsymbol{\mu},\mathbf{\Sigma}) \tag{90}$$

この仮定は$\mathbf{Z}$と$\{\boldsymbol{\pi},\boldsymbol{\mu},\mathbf{\Sigma}\}$が独立であることを仮定しているだけであり，$\mathbf{Z}$と$\boldsymbol{\pi}$が独立であることは仮定していません。$\mathbf{Z}$と$\boldsymbol{\pi}$が独立であるための必要十分条件は，$\mathbf{Z},\boldsymbol{\pi}$の同時分布がそれぞれの確率分布の積で分離されることであり，$\mathbf{Z},\boldsymbol{\pi}$以外も含まれる同時分布から$\mathbf{Z}$のみが分離されたからといって$\mathbf{Z}$と$\boldsymbol{\pi}$が独立になるとは限りません。

「1. 同時分布の依存関係確認」で仮定したように，$\boldsymbol{\pi}$と$(\boldsymbol{\mu}, \boldsymbol{\Sigma})$は独立ですから，$q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$を$q(\boldsymbol{\pi})$と$q(\boldsymbol{\mu}, \boldsymbol{\Sigma})$に分解してもよいです。しかし，次節で計算して分かる通り，分解せずとも$\boldsymbol{\pi}$と$\boldsymbol{\mu}, \boldsymbol{\Sigma}$が独立である結果が導かれますから，ここではより弱い仮定を採用することにします。これにて「2. 平均場近似のグループ分けの仮定」が完了しました。

3. 更新式適用による変分下限の最大化

本節では，変分ベイズの更新式を用いて変分下限を最大化します。「2. 平均場近似のグループ分けの仮定」では，潜在変数$\mathbf{Z}$の属する分布とパラメータ$\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$の属する分布が別々であると仮定しました。そこで，以下では潜在変数に関する最適化を考えた後に，パラメータに関する最適化を考えます。EMアルゴリズムの類推により，前者の最適化をEステップ，後者の最適化をMステップとみなせます。

> 管理人の考えでは，変分ベイズではEステップとMステップを分ける意味はあまりないと考えています。なぜなら，変分ベイズではEMアルゴリズムとは異なり，パラメータと潜在変数を同一視するからです。一方で，本稿の趣旨はEMアルゴリズムとの比較を通じて変分ベイズをはじめからていねいに理解することです。そのため，今回は変分ベイズにおけるEステップを変分Eステップ，Mステップを変分Mステップと表記することにします。詳しくは末尾の付録で説明します。

最初に結論からお伝えします。

> 変分ベイズのGMMへの適用（GMM-VB）は，以下の4ステップから構成される。
>
> 1. 初期値設定
> 2. 負担率r_{nk}の計算（変分Eステップ）
> 3. 各パラメータの更新（変分Mステップ）
> 4. 収束判定
>
> まずは各パラメータに初期値α_0，β_0，ν_0，$\mathbf{m}_0$，$\mathbf{W}_0$を与える。次に，以下の変分Eステップと変分Mステップを収束するまで繰り返す。

変分Eステップ

負担率r_{nk}を計算する。

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \tag{91}$$

ただし，ρ_{nk}は以下で計算できる。

$$\ln \tilde{\pi}_k = \psi(\alpha_k) - \psi\left(\sum_{k=1}^{K} \alpha_k\right) \tag{92}$$

$$\ln \tilde{\Sigma}_k = \sum_{i=1}^{D} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + D\ln(2) + \ln|\mathbf{W}_k| \tag{93}$$

$$\rho_{nk} = \tilde{\pi}_k \tilde{\mathbf{\Sigma}}_k^{\frac{1}{2}} \exp\left\{-\frac{D}{2\beta_k} - \frac{\nu_k}{2}(\mathbf{x}_n - \mathbf{m}_k)^T \mathbf{W}_k (\mathbf{x}_n - \mathbf{m}_k)\right\} \tag{94}$$

変分Mステップ

パラメータを更新する。

$$\alpha_k = \alpha_0 + N_k \tag{95}$$

$$\beta_k = \beta_0 + N_k \tag{96}$$

$$\nu_k = \nu_0 + N_k \tag{97}$$

$$\mathbf{m}_k = \frac{1}{\beta_k}(\beta_0 \mathbf{m}_0 + N_k \overline{\mathbf{x}}_k) \tag{98}$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\overline{\mathbf{x}}_k - \mathbf{m}_0)(\overline{\mathbf{x}}_k - \mathbf{m}_0)^T \tag{99}$$

ただし，N_k，$\overline{\mathbf{x}}_k$，$\mathbf{S}_k$は変分Eステップで計算した負担率r_{nk}を用いて計算できる。

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{100}$$

$$\overline{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{101}$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \overline{\mathbf{x}}_k)(\mathbf{x}_n - \overline{\mathbf{x}}_k)^T \tag{102}$$

GMM-VBのアニメーション

変分Eステップ

まずは変分Eステップの導出から始めましょう。最適なqである$q^*(\mathbf{Z})$を求めます。変分ベイズの更新式を適用すると，以下のように$q^*(\mathbf{Z})$を計算することができます。

$$\ln q^*(\mathbf{Z}) = E_{\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}}\left[\ln p(\mathbf{X},\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma})\right] + \text{const} \tag{103}$$

$$= E_{\boldsymbol{\pi}}\left[\ln p(\mathbf{Z}|\boldsymbol{\pi})\right] + E_{\boldsymbol{\mu},\boldsymbol{\Sigma}}[\ln p(\mathbf{X}|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Sigma})] + \text{const} \tag{104}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} E[\ln \pi_k] + \frac{1}{2}E[\ln|\boldsymbol{\Sigma}_k|] - \frac{D}{2}\ln(2\pi)$$

$$-\frac{1}{2}E_{\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k}\left[(\mathbf{x}_n-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k(\mathbf{x}_n-\boldsymbol{\mu}_k)\right] + \text{const} \tag{105}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\ln\rho_{nk} + \text{const} \tag{106}$$

ただし，$\ln\rho_{nk}$は以下のように置きました。

$$\ln\rho_{nk} = E[\ln\pi_k] + \frac{1}{2}E[\ln|\boldsymbol{\Sigma}_k|] - \frac{D}{2}\ln(2\pi)$$

$$-\frac{1}{2}E_{\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k}\left[(\mathbf{x}_n-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k(\mathbf{x}_n-\boldsymbol{\mu}_k)\right] \tag{107}$$

一旦，定数項constを無視すると，$q^*(\boldsymbol{Z})$は以下のように求められます。

$$q^*(\mathbf{Z}) \propto \prod_{n=1}^{N}\prod_{k=1}^{K}\rho_{nk}^{z_{nk}} \tag{108}$$

以下では，厳密に$q^*(\boldsymbol{Z})$を定めていきます。すなわち，先ほど無視した定数項constによる正規化定数を求めます。求める正規化定数をAとおくと，

$$\frac{1}{A} = \sum_{\mathbf{Z}} \prod_{n=1}^{N} \prod_{k=1}^{K} \rho_{nk}^{z_{nk}} \tag{109}$$

$$= \left(\sum_{\mathbf{z}_1} \prod_{k=1}^{K} \rho_{1k}^{z_{1k}} \right) \cdot \left(\sum_{\mathbf{z}_2} \prod_{k=1}^{K} \rho_{2k}^{z_{2k}} \right) \cdots \tag{110}$$

$$= \left(\sum_{k=1}^{K} \rho_{1k} \right) \cdot \left(\sum_{k=1}^{K} \rho_{2k} \right) \cdots \tag{111}$$

$$= \prod_{n=1}^{N} \sum_{k=1}^{K} \rho_{nk} \tag{112}$$

と計算することができます。ただし，途中で$\mathbf{z}$がone-hot-vectorであることを利用しました。したがって，$q^*(\mathbf{Z})$は正確に以下のように表されます。

$$q^*(\mathbf{Z}) = \frac{1}{A} \prod_{n=1}^{N} \prod_{k=1}^{K} \rho_{nk}^{z_{nk}} \tag{113}$$

$$= \frac{\prod_{n=1}^{N} \prod_{k=1}^{K} \rho_{nk}^{z_{nk}}}{\prod_{n=1}^{N} \sum_{k=1}^{K} \rho_{nk}} \tag{114}$$

$$= \prod_{n=1}^{N} \frac{\prod_{k=1}^{K} \rho_{nk}^{z_{nk}}}{\sum_{k=1}^{K} \rho_{nk}} \tag{115}$$

$$= \prod_{k=1}^{N} \frac{\prod_{k=1}^{K} \rho_{nk}^{z_{nk}}}{\prod_{k=1}^{K} \left\{ \sum_{j=1}^{K} \rho_{nj} \right\}^{z_{nk}}} \tag{116}$$

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} \left\{ \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \right\}^{z_{nk}} \tag{117}$$

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} r_{nk}^{z_{nk}} \tag{118}$$

ただし，r_{nk}は以下のように置きました。

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \tag{119}$$

このr_{nk}がEMアルゴリズムにおける負担率となります。その証明としては，式(118)の期待値がr_{nk}となることから確認できます。z_{nk}が1のときだけr_{nk}が寄与するからです。

$$E[z_{nk}] = r_{nk} \tag{120}$$

この式は，EMアルゴリズムで定義した負担率と全く同じです。

最後に，式(107)と事前分布の仮定を利用して，r_{nk}を計算するためにρ_{nk}を求めましょう。その前に，D次元ベクトル$\mathbf{x}$，$\boldsymbol{\mu} \sim \mathrm{N}(\mathbf{m}, (\beta\Sigma)^{-1})$，および$\Sigma \sim \mathrm{W}(\Sigma|\mathbf{W}, \nu)$に対し，

$$E_{\boldsymbol{\mu},\Sigma}\left[(\mathbf{x}-\boldsymbol{\mu})^T\Sigma(\mathbf{x}-\boldsymbol{\mu})\right] = \nu(\mathbf{x}-\mathbf{m})^T\mathbf{W}(\mathbf{x}-\mathbf{m}) + \beta^{-1}D \tag{121}$$

が成り立つことを証明します。$\mathrm{N}(\mathbf{m}, \Sigma)$の確率密度関数を$f_{\mathrm{N}}(\mathbf{m}, \Sigma)$とおいて$E_{\boldsymbol{\mu},\Sigma}$の定義から計算すると，

$$
\begin{aligned}
E_{\boldsymbol{\mu},\Sigma}\left[(\mathbf{x}-\boldsymbol{\mu})^T\Sigma(\mathbf{x}-\boldsymbol{\mu})\right] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma(\mathbf{x}-\boldsymbol{\mu})f_{\mathrm{N}}(\mathbf{m},\Sigma)d\boldsymbol{\mu}d\Sigma \\
&= \int_{-\infty}^{\infty}\left\{\int_{-\infty}^{\infty}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma(\mathbf{x}-\boldsymbol{\mu})f_{\mathrm{N}}(\mathbf{m}|\Sigma)d\boldsymbol{\mu}\right\}f_N \\
&= \int_{-\infty}^{\infty}E_{\boldsymbol{\mu}}[(\mathbf{x}-\boldsymbol{\mu})^T\Sigma(\mathbf{x}-\boldsymbol{\mu})]f_N(\Sigma)d\Sigma \\
&= \int_{-\infty}^{\infty}E_{\boldsymbol{\mu}}[\mathrm{Tr}\left\{\Sigma(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\right\}]f_N(\Sigma)d\Sigma \\
&= \int_{-\infty}^{\infty}E_{\boldsymbol{\mu}}[\mathrm{Tr}\left(\Sigma(\mathbf{x}\mathbf{x}^T-2\boldsymbol{\mu}\mathbf{x}^T+\boldsymbol{\mu}\boldsymbol{\mu}^T)\right)]f_N(\Sigma)d \\
&= \int_{-\infty}^{\infty}\mathrm{Tr}\left[E_{\boldsymbol{\mu}}[\left(\Sigma(\mathbf{x}\mathbf{x}^T-2\boldsymbol{\mu}\mathbf{x}^T+\boldsymbol{\mu}\boldsymbol{\mu}^T)\right)]\right]f_N(\Sigma) \\
&= \int_{-\infty}^{\infty}\mathrm{Tr}\left[\Sigma(\mathbf{x}\mathbf{x}^T-2E_{\boldsymbol{\mu}}[\boldsymbol{\mu}]\mathbf{x}^T+E_{\boldsymbol{\mu}}[\boldsymbol{\mu}\boldsymbol{\mu}^T])\right]f_N(\\
&= \int_{-\infty}^{\infty}\mathrm{Tr}\left[\Sigma(\mathbf{x}\mathbf{x}^T-2\mathbf{m}\mathbf{x}^T+\mathbf{m}\mathbf{m}^T+(\beta\Sigma)^{-1})\right]f \\
&= \int_{-\infty}^{\infty}\mathrm{Tr}\left[\Sigma(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T+\Sigma(\beta\Sigma)^{-1}\right]f_N(\Sigma \\
&= \int_{-\infty}^{\infty}\mathrm{Tr}\left[\Sigma(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T+\beta^{-1}I_D\right]f_N(\Sigma)d \\
&= \int_{-\infty}^{\infty}\left\{\mathrm{Tr}\left[\Sigma(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\right]+\mathrm{Tr}\left[\beta^{-1}I_D\right]\right\} \\
&= \int_{-\infty}^{\infty}\left\{(\mathbf{x}-\mathbf{m})^T\Sigma(\mathbf{x}-\mathbf{m})+\beta^{-1}D\right\}f_N(\Sigma)d\Sigma \\
&= E_{\Sigma}\left[\mathrm{Tr}\left[\Sigma(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\right]+\beta^{-1}D\right] \\
&= E_{\Sigma}\left[\mathrm{Tr}\left[\Sigma(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\right]\right]+\beta^{-1}D \\
&= \mathrm{Tr}\left[E_{\Sigma}\left[\Sigma(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\right]\right]+\beta^{-1}D \\
&= \mathrm{Tr}\left[E_{\Sigma}\left[\Sigma\right](\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\right]+\beta^{-1}D \\
&= \mathrm{Tr}\left[\nu\mathbf{W}(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\right]+\beta^{-1}D \\
&= (\mathbf{x}-\mathbf{m})^T\nu\mathbf{W}(\mathbf{x}-\mathbf{m})+\beta^{-1}D \\
&= (\quad\quad)^T\mathbf{W}(\quad\quad)+\beta^{-1}D
\end{aligned}
$$

となり，式(121)が示されました。ただし，$X^TAX = \mathrm{Tr}[AXX^T]$であること，分散共分散行列の定義より

$$(\beta\Sigma)^{-1} = E[(\boldsymbol{\mu} - \mathbf{m})(\boldsymbol{\mu} - \mathbf{m})^T] \tag{141}$$
$$= E[\boldsymbol{\mu}\boldsymbol{\mu}^T] - 2\mathbf{m}E[\boldsymbol{\mu}^T] + \mathbf{m}\mathbf{m}^T \tag{142}$$
$$= E[\boldsymbol{\mu}\boldsymbol{\mu}^T] - 2\mathbf{m}\mathbf{m}^T + \mathbf{m}\mathbf{m}^T \tag{143}$$
$$= E[\boldsymbol{\mu}\boldsymbol{\mu}^T] - \mathbf{m}\mathbf{m}^T \tag{144}$$

となり$E[\boldsymbol{\mu}\boldsymbol{\mu}^T]=\mathbf{m}\mathbf{m}^T+(\beta\Sigma)^{-1}$であること，トレースには線形性があること，ウィシャート分布の期待値が式(86)で表されること，および期待値とトレースは交換可能であることを利用しました。

期待値とトレースの交換

D次元正方行列Aを考える。

$$A = \begin{pmatrix} a_{11} & \cdots & \\ \vdots & \ddots & \\ & & a_{DD} \end{pmatrix} \tag{145}$$

期待値の線形性とトレースの定義により，期待値とトレースが交換可能であることが分かる。

$$E\left[\mathrm{Tr}\left[A\right]\right] = E\left[\sum_{d=1}^{D} a_{dd}\right] \tag{146}$$
$$= \sum_{d=1}^{D} E\left[a_{dd}\right] \tag{147}$$
$$= \mathrm{Tr}\left[\begin{pmatrix} E[a_{11}] & \cdots & \\ \vdots & \ddots & \\ & & E[a_{DD}] \end{pmatrix}\right] \tag{148}$$
$$= \mathrm{Tr}\left[E\left[A\right]\right] \tag{149}$$

式(121)を今回の仮定$\boldsymbol{\mu}\sim\mathrm{N}\{\boldsymbol{\mu}_k|\mathbf{m}_k,(\beta_k\boldsymbol{\Sigma}_k)^{-1}\}$および$\Sigma_k\sim\mathrm{W}(\boldsymbol{\Sigma}_k|\mathbf{W}_k,\nu_k)$に適用すると，

$$E_{\boldsymbol{\mu}_k,\Sigma_k}\left[(\mathbf{x}_n-\boldsymbol{\mu}_k)^T\Sigma_k(\mathbf{x}_n-\boldsymbol{\mu}_k)\right]=\nu_k(\mathbf{x}_n-\mathbf{m}_k)^TW_k(\mathbf{x}_n-\mathbf{m}_k)+\beta_k^{-1}D\quad($$

が得られます。この結果を利用して$\ln\rho_{nk}$を計算すると，

$$\begin{aligned}\ln\rho_{nk}&=E[\ln\pi_k]+\frac{1}{2}E[\ln|\boldsymbol{\Sigma}_k|]-\frac{D}{2}\ln(2\pi)\\&\quad-\frac{1}{2}E_{\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k}\left[(\mathbf{x}_n-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k(\mathbf{x}_n-\boldsymbol{\mu}_k)\right]&(151)\\&=\psi(\alpha_k)-\psi\left(\sum_{k=1}^K\alpha_k\right)\\&\quad+\frac{1}{2}\sum_{i=1}^D\psi\left(\frac{\nu_k+1-i}{2}\right)+D\ln(2)+\ln|\mathbf{W}_k|\\&\quad-\frac{1}{2}\left\{\nu_k(\mathbf{x}_n-\mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_n-\mathbf{m}_k)+\beta_k^{-1}D\right\}&(152)\\&=\ln\tilde{\pi}_k+\ln\tilde{\Sigma}_k-\frac{1}{2}\left\{\nu_k(\mathbf{x}_n-\mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_n-\mathbf{m}_k)+\beta_k^{-1}D\right\}&(153)\end{aligned}$$

が得られます。ただし，ごちゃごちゃした項は以下のように置き換えました。

$$\begin{aligned}\ln\tilde{\pi}_k&=E[\ln\pi_k]&(154)\\&=\psi(\alpha_k)-\psi\left(\sum_{k=1}^K\alpha_k\right)&(155)\\\ln\tilde{\Sigma}_k&=\frac{1}{2}E[\ln|\boldsymbol{\Sigma}_k|]&(156)\\&=\sum_{i=1}^D\psi\left(\frac{\nu_k+1-i}{2}\right)+D\ln(2)+\ln|\mathbf{W}_k|&(157)\end{aligned}$$

変分ベイズのEステップでは，このρ_{nk}を計算してr_{nk}を求めます。そのために，ダイレクトにr_{nk}を更新できるような式を求めておきましょう。式(153)の定義からρ_{nk}を求めます。ρ_{nk}さえ求まれば，あとは正規化するだけで負担率r_{nk}が計算できます。

$$\begin{aligned}\rho_{nk} &= \exp\left[\ln\tilde{\pi}_k + \frac{1}{2}\ln\tilde{\boldsymbol{\Sigma}}_k - \frac{1}{2}\left\{\nu_k(\mathbf{x}_n - \mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_n - \mathbf{m}_k) + \beta_k^{-1}D\right\}\right] \\ &= \tilde{\pi}_k\tilde{\boldsymbol{\Sigma}}_k^{\frac{1}{2}}\exp\left\{-\frac{\nu_k}{2}(\mathbf{x}_n - \mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_n - \mathbf{m}_k) - \frac{D}{2\beta_k}\right\}\end{aligned}$$

以上で，「3. 更新式適用による変分下限の最大化」における変分Eステップの導出が完了しました。

変分Mステップ

続いて，パラメータに関する最適化を行う変分Mステップの導出を行います。Mステップでは，変分ベイズの更新式を用いて潜在変数以外のパラメータに関して最適化を施していきます。変分ベイズの更新式を利用するためには，$\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$の対数同時分布$\ln p(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$を知る必要があります。「1. 同時分布の依存関係確認」の仮定に基づくと，$\ln p(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$は以下のように表されます。

$$\ln p(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \ln p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \ln p(\mathbf{Z}|\boldsymbol{\pi}) + \ln p(\boldsymbol{\pi}) + \ln p(\boldsymbol{\mu}|\boldsymbol{\Sigma}) + \ln p(\boldsymbol{\Sigma}$$

したがって，変分ベイズの更新式を用いると，$\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$に関する近似事後分布は以下のように求められます。

$$\begin{aligned}\ln q^*(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= E_{\mathbf{Z}}\left[\ln p(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\right] && (161) \\ &= \sum_{k=1}^{K}\sum_{n=1}^{N} E[z_{nk}]\ln \mathrm{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}^{-1}) + E_{\mathbf{Z}}\left[\ln p(\mathbf{Z}|\boldsymbol{\pi})\right] \\ &\quad + \ln p(\boldsymbol{\pi}) + \sum_{k=1}^{K}\ln p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \mathrm{const} && (162)\end{aligned}$$

以下では，$\boldsymbol{\pi}$と$\boldsymbol{\mu}, \boldsymbol{\Sigma}$に分けて近似事後分布を導出したいと思います。方針としては，式(162)において，$\boldsymbol{\pi}$に関わる部分と$\boldsymbol{\mu}, \boldsymbol{\Sigma}$に関わる部分を別々に抽出します。ここで$\boldsymbol{\pi}$と$\boldsymbol{\mu}, \boldsymbol{\Sigma}$を別々に考えるのは，「1. 同時分布の依存関係確認」の仮定に基づいています。グラフィカルモデルを見ても分かる通り，$\boldsymbol{\mu}, \boldsymbol{\Sigma}$は依存関係にありますが，$\boldsymbol{\pi}$は孤立しています。

多項分布$p(\mathbf{Z}|\boldsymbol{\pi})$の共役事前分布としてディリクレ分布$p(\boldsymbol{\pi})$を仮定していますので，$\boldsymbol{\pi}$に関する近似事後分布$q^*(\boldsymbol{\pi})$もまたディリクレ分布の形になります。今回は対数近似事後分布を考えていますので，式(162)の$\boldsymbol{\pi}$に関わる部分はディリクレ分布の対数を取った形になるはずです。したがって，ディリクレ分布の対数を取った理想の形と係数比較を行うことで，$\boldsymbol{\pi}$に関する近似事後分布を求めることができます。

$\boldsymbol{\mu}, \boldsymbol{\Sigma}$も同様に導出します。多変量正規分布$p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$の共役事前分布としてガウスウィシャート分布$p(\boldsymbol{\mu}|\boldsymbol{\Sigma})p(\boldsymbol{\Sigma})$を仮定していますので，近似事後分布$q^*(\boldsymbol{\mu}|\boldsymbol{\Sigma})$は多変量ガウス分布，$q^*(\boldsymbol{\Sigma})$はウィシャート分布になるはずです。したがって，多変量正規分布・ウィシャート分布の対数を取った理想の形と係数比較を行うことで，$\boldsymbol{\mu}, \boldsymbol{\Sigma}$に関する近似事後分布を求めることができます。

早速，式(162)において$\boldsymbol{\pi}$に関わる部分を抽出します。

$$\ln q^*(\boldsymbol{\pi}) = \sum_{k=1}^{K}\sum_{n=1}^{N} r_{nk} \ln \pi_k - \ln \left\{ B(\boldsymbol{\alpha}_0) \prod_{k=1}^{K} \pi_k^{\alpha_0 - 1} \right\} + \text{const} \tag{163}$$

$$= \sum_{k=1}^{K}\sum_{n=1}^{N} r_{nk} \ln \pi_k + \sum_{k=1}^{K} (\alpha_0 - 1) \ln \pi_k + \text{const} \tag{164}$$

$$= \sum_{k=1}^{K} \left(\alpha_0 - 1 + \sum_{n=1}^{N} r_{nk} \right) \ln \pi_k + \text{const} \tag{165}$$

ただし，ディリクレ分布のパラメータ$\boldsymbol{\alpha}_0$について，Kクラスの初期値の対称性を考えて全ての要素をα_0としました。ここで，分かりやすさのため，

$$N_k = \sum_{n=1}^{N} r_{nk} = \sum_{n=1}^{N} E[z_{nk}] \tag{166}$$

とおくと，以下のように$\ln q^*(\boldsymbol{\pi})$をキレイな形に変形することができます。

$$\ln q^*(\boldsymbol{\pi}) = \sum_{k=1}^{K} (\alpha_0 - 1 + N_k) \ln \pi_k + \text{const} \tag{167}$$

先ほどもお伝えした通り，この形はディリクレ分布の対数をとったものになっているはずです。

$$\ln q^*(\boldsymbol{\pi}) = \ln \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) = \sum_{k=1}^{K} (\alpha_k - 1 + N_k) \ln \pi_k + \text{const} \tag{168}$$

すると，以下の恒等式が成り立ちます。

$$\sum_{k=1}^{K} (\alpha_0 + N_k - 1) \ln \pi_k = \sum_{k=1}^{K} (\alpha_k - 1) \ln \pi_k \tag{169}$$

係数比較により，α_kに関する更新式が得られます。

$$\alpha_k = \alpha_0 + N_k \tag{170}$$

これにて，$\boldsymbol{\pi}$に関する最適化は終了です。続いて，$\boldsymbol{\mu}, \boldsymbol{\Sigma}$に関する最適化を行います。式(162)において，$\boldsymbol{\mu}, \boldsymbol{\Sigma}$に関わる部分を抽出します。

$$\ln q^*(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} \ln \mathrm{N}(\boldsymbol{\mu}|\mathbf{m}_0, (\beta_0 \boldsymbol{\Sigma}_k)^{-1}) + \sum_{k=1}^{K} \ln \mathrm{W}(\boldsymbol{\Sigma}_k|\mathbf{W}_0, \nu_0)$$

$$+ \sum_{n=1}^{N} \sum_{k=1}^{K} E[z_{nk}] \ln \mathrm{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k^{-1}) + \mathrm{const} \tag{171}$$

$$\propto \sum_{k=1}^{K} \left\{ \frac{1}{2} \ln |\beta_0 \boldsymbol{\Sigma}_k| - \frac{1}{2} (\boldsymbol{\mu}_k - \mathbf{m}_0)^T \beta_0 \boldsymbol{\Sigma}_k (\boldsymbol{\mu}_k - \mathbf{m}_0) \right.$$

$$+ \frac{\nu_0 - D - 1}{2} \ln |\boldsymbol{\Sigma}_k| - \frac{1}{2} \mathrm{Tr}(\mathbf{W}_0^{-1} \boldsymbol{\Sigma}_k)$$

$$\left. + \frac{1}{2} N_k \ln |\boldsymbol{\Sigma}_k| - \frac{1}{2} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k (\mathbf{x}_n - \boldsymbol{\mu}_k) \right\} \tag{172}$$

ここで注意するべきなのは，最後の項で早とちりして$\sum_{n=1}^{N} r_{nk} = N_k$と変形しないことです。なぜなら，$\sum_n$の中身に$\mathbf{x}_n$が含まれているため，$N_k$の定義とは一致しないからです。ここは，引っかかりポイントだと思います。

さて，ここからはグラフィカルモデルに基づいて$\boldsymbol{\mu}$と$\boldsymbol{\Sigma}$の依存関係を明確にしましょう。$\boldsymbol{\mu}$は$\boldsymbol{\Sigma}$に依存していますから，ベイズの定理より$q^*(\boldsymbol{\mu}, \boldsymbol{\Sigma})$は理想的には以下のように分解されます。

$$q^*(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = q^*(\boldsymbol{\mu}|\boldsymbol{\Sigma}) q^*(\boldsymbol{\Sigma}) \tag{173}$$

両辺の対数を取ります。

$$\ln q^*(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \ln q^*(\boldsymbol{\mu}|\boldsymbol{\Sigma}) + \ln q^*(\boldsymbol{\Sigma}) \tag{174}$$

したがって，式(172)において$\boldsymbol{\mu}$に関する項だけを抽出した結果は$\ln q^*(\boldsymbol{\mu}|\boldsymbol{\Sigma})$となることが分かります。

$$\ln q^*(\boldsymbol{\mu}|\boldsymbol{\Sigma})$$

$$= -\frac{1}{2}\sum_{k=1}^{K}\left\{(\boldsymbol{\mu}_k - \mathbf{m}_0)^T\beta_0\boldsymbol{\Sigma}_k(\boldsymbol{\mu}_k - \mathbf{m}_0) + N_k(\mathbf{x}_n - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k)\right\}$$

先ほどもお伝えした通り，この形は多変量正規分布の対数をとったものになっているはずです。

$$\ln q^*(\boldsymbol{\mu}|\boldsymbol{\Sigma}) = \sum_{k=1}^{K}\ln \mathrm{N}(\boldsymbol{\mu}_k|\mathbf{m}_k, \beta_k\boldsymbol{\Sigma}_k) \tag{17}$$

$$= \sum_{k=1}^{K}\left\{-\frac{1}{2}|\boldsymbol{\Sigma}_k| - \frac{\beta_k}{2}(\boldsymbol{\mu}_k - \mathbf{m}_k)^T\boldsymbol{\Sigma}_k(\boldsymbol{\mu}_k - \mathbf{m}_k) + \mathrm{const}\right\} \tag{17}$$

$\boldsymbol{\pi}$のときと同様に，式(172)と式(177)の恒等式から係数比較により更新式を導出していきます。そこで，一旦指数部の$\boldsymbol{\mu}_k^T\boldsymbol{\mu}_k$の項だけに注目して係数比較を行ってみましょう。式(172)において，$\boldsymbol{\mu}_k^T\boldsymbol{\mu}_k$に関する項は以下です。

$$\boldsymbol{\mu}_k^T\beta_0\boldsymbol{\Sigma}_k\mu_k + N_k\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_k\boldsymbol{\mu}_k = \boldsymbol{\mu}_k^T(\beta_0 + N_k)\boldsymbol{\Sigma}_k\boldsymbol{\mu}_k \tag{178}$$

式(177)において，$\boldsymbol{\mu}_k^T\boldsymbol{\mu}_k$に関する項は以下です。

$$\boldsymbol{\mu}_k^T\beta_k\boldsymbol{\Sigma}_k\boldsymbol{\mu}_k \tag{179}$$

したがって，$\boldsymbol{\mu}_k$の二次の項である$\boldsymbol{\mu}_k^T\boldsymbol{\mu}_k$に着目すると，以下の恒等式が成り立ちます。

$$\boldsymbol{\mu}_k^T(\beta_0 + N_k)\boldsymbol{\Sigma}_k\boldsymbol{\mu}_k = \boldsymbol{\mu}_k^T(\beta_k)\boldsymbol{\Sigma}_k\boldsymbol{\mu}_k \tag{180}$$

係数比較により，β_kに関する更新式が得られます。

$$\beta_k = \beta_0 + N_k \tag{181}$$

α_kに関する更新式(170)と対称的な結果が得られて美しいですね。しかし，まだ$\mathbf{m}_k$に関する更新式が得られていません。そこで，指数部の$\boldsymbol{\mu}_k$の一次の項だけに注目して係数比較を行ってみましょう。式(172)において，$\boldsymbol{\mu}_k$に関する項は以下です。

$$\boldsymbol{\mu}_k^T \beta_0 \boldsymbol{\Sigma}_k \boldsymbol{m}_0 + \sum_{n=1}^{N} r_{nk} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}_k \mathbf{x}_n = \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}_k \left(\beta_0 m_0 + N_k \overline{\mathbf{x}}_k \right) \tag{182}$$

式(177)において，$\boldsymbol{\mu}_k$に関する項は以下です。

$$\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}_k \beta_0 \mathbf{m}_k \tag{183}$$

したがって，以下の恒等式が成り立ちます。

$$\beta_k \mathbf{m}_k = \beta_0 \mathbf{m}_0 + N_k \overline{\mathbf{x}}_k \tag{184}$$

係数比較により，$\mathbf{m}_k$に関する更新式が得られます。

$$\mathbf{m}_k = \frac{1}{\beta_k} (\beta_0 \mathbf{m}_0 + N_k \overline{\mathbf{x}}_k) \tag{185}$$

最後に，$\boldsymbol{\Sigma}$に関する更新式を導出していきたいと思います。式(174)を変形すると，$\ln q^*(\boldsymbol{\Sigma})$は以下のように表されます。

$$\ln q^*(\boldsymbol{\Sigma}) = \ln q^*(\boldsymbol{\mu}, \boldsymbol{\Sigma}) - \ln q^*(\boldsymbol{\mu} | \boldsymbol{\Sigma}) \tag{186}$$

式(172)と式(177)から式(186)を計算していきます。

$$\ln q^*(\boldsymbol{\Sigma}) = \sum_{k=1}^{K} \left\{ \frac{1}{2} \ln |\beta_0 \boldsymbol{\Sigma}_k| - \frac{1}{2} (\boldsymbol{\mu}_k - \mathbf{m}_0)^T \beta_0 \boldsymbol{\Sigma}_k (\boldsymbol{\mu}_k - \mathbf{m}_0) \right.$$
$$+ \frac{\nu_0 - D - 1}{2} \ln |\boldsymbol{\Sigma}_k| - \frac{1}{2} \mathrm{Tr}(\mathbf{W}_0^{-1} \boldsymbol{\Sigma}_k)$$
$$+ \frac{1}{2} N_k \ln |\boldsymbol{\Sigma}_k| - \frac{1}{2} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)$$
$$\left. + \frac{1}{2} |\boldsymbol{\Sigma}_k| + \frac{\beta_k}{2} (\boldsymbol{\mu}_k - \mathbf{m}_k)^T \boldsymbol{\Sigma}_k (\boldsymbol{\mu}_k - \mathbf{m}_k) + \mathrm{const} \right\} \tag{187}$$

先ほどもお伝えした通り，この形はウィシャート分布の対数をとったものになっているはずです。

$$\ln q^*(\boldsymbol{\Sigma}) = \sum_{k=1}^{K} \left(\frac{\nu_k - D - 1}{2} \ln |\boldsymbol{\Sigma}_k| - \frac{1}{2} \mathrm{Tr}[\mathbf{W}_k^{-1} \boldsymbol{\Sigma}_k] \right) \tag{188}$$

$\ln |\Sigma_k|$に注目すると，以下の恒等式が成り立ちます。

$$\frac{\nu_0 - D - 1}{2} \ln |\boldsymbol{\Sigma}_k| = \frac{\nu_k - D - 1}{2} \ln |\boldsymbol{\Sigma}_k| \tag{189}$$

係数比較により，ν_kに関する更新式が得られます。

$$\nu_k = \nu_0 + N_k \tag{190}$$

α_kに関する更新式(170)，β_kに関する更新式(181)と対称的な結果が得られて美しいですね。しかし，まだ$\mathbf{W}_k$に関する更新式が得られていないため恒等式を立てます。ここでは，トレースに関する以下の3つの性質を利用します。

$$\mathbf{x}^T A \mathbf{x} = \mathrm{Tr}\left[A \mathbf{x} \mathbf{x}^T \right] \tag{191}$$
$$\mathrm{Tr}\left[A \right] + \mathrm{Tr}\left[B \right] = \mathrm{Tr}\left[A + B \right] \tag{192}$$
$$\mathrm{Tr}\left[A^T \right] = \mathrm{Tr}\left[A \right] \tag{193}$$

上から順番に，トレースと二次形式の関係，トレースの線形性，トレースと転置の関係を表しています。これらの性質を念頭に置くと，式(187)における$\mathbf{\Sigma}_k$に関する項は以下のように整理されます。

$$
\begin{aligned}
&-\frac{1}{2}\sum_{k=1}^{K}\Bigg\{\mathrm{Tr}\left[\beta_0\mathbf{\Sigma}_k(\boldsymbol{\mu}_k-\mathbf{m}_0)(\boldsymbol{\mu}_k-\mathbf{m}_0)^T\right]+\mathrm{Tr}\left[\mathbf{W}_0^{-1}\mathbf{\Sigma}_k\right] \\
&\quad+\mathrm{Tr}\left[\sum_{n=1}^{N}r_{nk}\mathbf{\Sigma}_k(\mathbf{x}_n-\boldsymbol{\mu}_k)(\mathbf{x}_n-\boldsymbol{\mu}_k)^T\right]-\mathrm{Tr}\left[\beta_k\mathbf{\Sigma}_k(\boldsymbol{\mu}_k-\mathbf{m}_k)(\boldsymbol{\mu}_k-\mathbf{m}\right. \\
&=-\frac{1}{2}\sum_{k=1}^{K}\mathrm{Tr}\Bigg[\beta_0\mathbf{\Sigma}_k(\boldsymbol{\mu}_k-\mathbf{m}_0)(\boldsymbol{\mu}_k-\mathbf{m}_0)^T+\mathbf{W}_0^{-1}\mathbf{\Sigma}_k \\
&\quad+\sum_{n=1}^{N}r_{nk}\mathbf{\Sigma}_k(\mathbf{x}_n-\boldsymbol{\mu}_k)(\mathbf{x}_n-\boldsymbol{\mu}_k)^T-\beta_k\mathbf{\Sigma}_k(\boldsymbol{\mu}_k-\mathbf{m}_k)(\boldsymbol{\mu}_k-\mathbf{m}_k)^T\Bigg] \\
&=-\frac{1}{2}\sum_{k=1}^{K}\mathrm{Tr}\Bigg[\beta_0\mathbf{\Sigma}_k(\boldsymbol{\mu}_k-\mathbf{m}_0)(\boldsymbol{\mu}_k-\mathbf{m}_0)^T+\mathbf{\Sigma}_k\mathbf{W}_0^{-1} \\
&\quad+\sum_{n=1}^{N}r_{nk}\mathbf{\Sigma}_k(\mathbf{x}_n-\boldsymbol{\mu}_k)(\mathbf{x}_n-\boldsymbol{\mu}_k)^T-\beta_k\mathbf{\Sigma}_k(\boldsymbol{\mu}_k-\mathbf{m}_k)(\boldsymbol{\mu}_k-\mathbf{m}_k)^T\Bigg]
\end{aligned}
$$

ただし，$\mathbf{W}_0$と$\mathbf{\Sigma}_k$はどちらも対称行列であることを利用しました。一方，式(188)における$\mathbf{\Sigma}_k$に関する項も，トレースの転置に関する性質を利用すると以下のように整理することができます。

$$
-\frac{1}{2}\sum_{k=1}^{K}\mathrm{Tr}\left[\mathbf{W}_k^{-1}\mathbf{\Sigma}_k\right]=-\frac{1}{2}\sum_{k=1}^{K}\mathrm{Tr}\left[\mathbf{\Sigma}_k\mathbf{W}_k^{-1}\right] \tag{197}
$$

したがって，得られる恒等式は以下のようになります。

$$\boldsymbol{\Sigma}_k \mathbf{W}_k^{-1} = \beta_0 \boldsymbol{\Sigma}_k (\boldsymbol{\mu}_k - \mathbf{m}_0)(\boldsymbol{\mu}_k - \mathbf{m}_0)^T + \boldsymbol{\Sigma}_k \mathbf{W}_0^{-1}$$

$$+ \sum_{n=1}^{N} r_{nk} \boldsymbol{\Sigma}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T - \beta_k \boldsymbol{\Sigma}_k (\boldsymbol{\mu}_k - \mathbf{m}_k)(\boldsymbol{\mu}_k - \mathbf{m}_k$$

両辺の左から$\boldsymbol{\Sigma}_k^{-1}$を掛けると，$\mathbf{W}_k$に関する以下の更新式が得られます。

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + \beta_0 (\boldsymbol{\mu}_k - \mathbf{m}_0)(\boldsymbol{\mu}_k - \mathbf{m}_0)^T$$

$$+ \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T - \beta_k (\boldsymbol{\mu}_k - \mathbf{m}_k)(\boldsymbol{\mu}_k - \mathbf{m}_k)^T \quad (199$$

式(199)をキレイに表すために，以下のように新しい変数を導入します。

$$\overline{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{200}$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \overline{\mathbf{x}}_k)(\mathbf{x}_n - \overline{\mathbf{x}}_k)^T \tag{201}$$

これらを利用して，式(199)を整理していきます。ポイントとなるのは，今回は$\ln q^*(\boldsymbol{\Sigma})$を考えていますので，$\boldsymbol{\mu}_k$の項は打ち消されるという点です。そこで，以下では$\boldsymbol{\mu}_k$の項を無視して考えていきます。式(199)を整理する際に必要となる計算を予め行っておきます。

$$\sum_{n=1}^{N} r_{nk}\mathbf{x}_n\mathbf{x}_n^T = \sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \overline{\mathbf{x}}_k)(\mathbf{x}_n - \overline{\mathbf{x}}_k)^T$$

$$+2\sum_{n=1}^{N} r_{nk}\mathbf{x}_n\overline{\mathbf{x}}_k - \sum n = 1^N r_{nk}\overline{\mathbf{x}}_k\overline{\mathbf{x}}_k^T \tag{202}$$

$$= \sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \overline{\mathbf{x}}_k)(\mathbf{x}_n - \overline{\mathbf{x}}_k)^T + 2N_k\overline{\mathbf{x}}_k\overline{\mathbf{x}}_k^T - \overline{\mathbf{x}}_k\overline{\mathbf{x}}_k^T \tag{203}$$

$$= \sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \overline{\mathbf{x}}_k)(\mathbf{x}_n - \overline{\mathbf{x}}_k)^T + N_k\overline{\mathbf{x}}_k\overline{\mathbf{x}}_k^T \tag{204}$$

$$= N_k\mathbf{S}_k + N_k\overline{\mathbf{x}}_k\overline{\mathbf{x}}_k^T \tag{205}$$

式(199)を整理しましょう。β_kの更新式である式(181)と$\mathbf{m}_k$の更新式である式(185)を利用します。先ほどもお伝えしましたが，$\boldsymbol{\mu}_k$を含む項を無視するのがポイントです。

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + \beta_0\mathbf{m}_0\mathbf{m}_0^T + \sum_{n=1}^{N} r_{nk}\mathbf{x}_n\mathbf{x}_n^T - \beta_k\mathbf{m}_k\mathbf{m}_k^T \tag{206}$$

$$= \mathbf{W}_0^{-1} + \beta_0\mathbf{m}_0\mathbf{m}_0^T + (N_k\mathbf{S}_k + N_k\overline{\mathbf{x}}_k\overline{\mathbf{x}}_k^T) - \beta_k\mathbf{m}_k\mathbf{m}_k^T \tag{207}$$

$$= \mathbf{W}_0^{-1} + N_k\mathbf{S}_k + \beta_0\mathbf{m}_0\mathbf{m}_0^T + N_k\overline{\mathbf{x}}_k\overline{\mathbf{x}}_k^T$$

$$-(\beta_0 + N_k)\cdot\frac{1}{(\beta_0 + N_k)^2}(\beta_0\mathbf{m}_0 + N_k\overline{\mathbf{x}}_k)(\beta_0\mathbf{m}_0^T + N_k\overline{\mathbf{x}}_k^T) \tag{208}$$

$$= \mathbf{W}_0^{-1} + N_k\mathbf{S}_k + \frac{1}{\beta_0 + N_k}\Big\{(\beta_0 + N_k)(\beta_0\mathbf{m}_0\mathbf{m}_0^T + N_k\overline{\mathbf{x}}_k\overline{\mathbf{x}}_k^T)$$

$$-(\beta_0\mathbf{m}_0 + N_k\overline{\mathbf{x}}_k)(\beta_0\mathbf{m}_0^T + N_k\overline{\mathbf{x}}_k^T)\Big\} \tag{209}$$

$$= \mathbf{W}_0^{-1} + N_k\mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\overline{\mathbf{x}}_k - \mathbf{m}_0)(\overline{\mathbf{x}}_k - \mathbf{m}_0)^T \tag{210}$$

$\beta_k = \beta_0 + N_k$ を利用すれば，$\boldsymbol{\mu}_k$の項が消えることを計算しても示すことができます。

以上で，潜在変数と全てのパラメータに関する更新式が得られました。改めて，変分ベイズのGMMへの適用の流れをまとめておきます。

変分ベイズのGMMへの適用（GMM-VB）は，以下の4ステップから構成される。

1. 初期値設定
2. 負担率r_{nk}の計算（変分Eステップ）
3. 各パラメータの更新（変分Mステップ）
4. 収束判定

まずは各パラメータに初期値α_0，β_0，ν_0，$\mathbf{m}_0$，$\mathbf{W}_0$を与える。次に，以下の変分Eステップと変分Mステップを収束するまで繰り返す。

変分Eステップ

負担率r_{nk}を計算する。

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \tag{211}$$

ただし，ρ_{nk}は以下で計算できる。

$$\ln \tilde{\pi}_k = \psi(\alpha_k) - \psi\left(\sum_{k=1}^{K} \alpha_k\right) \tag{212}$$

$$\ln \tilde{\Sigma}_k = \sum_{i=1}^{D} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + D\ln(2) + \ln|\mathbf{W}_k| \tag{213}$$

$$\rho_{nk} = \tilde{\pi}_k \tilde{\mathbf{\Sigma}}_k^{\frac{1}{2}} \exp\left\{-\frac{D}{2\beta_k} - \frac{\nu_k}{2}(\mathbf{x}_n - \mathbf{m}_k)^T \mathbf{W}_k (\mathbf{x}_n - \mathbf{m}_k)\right\} \tag{214}$$

変分Mステップ

パラメータを更新する。

$$\alpha_k = \alpha_0 + N_k \tag{215}$$

$$\beta_k = \beta_0 + N_k \tag{216}$$

$$\nu_k = \nu_0 + N_k \tag{217}$$

$$\mathbf{m}_k = \frac{1}{\beta_k}(\beta_0 \mathbf{m}_0 + N_k \overline{\mathbf{x}}_k) \tag{218}$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\overline{\mathbf{x}}_k - \mathbf{m}_0)(\overline{\mathbf{x}}_k - \mathbf{m}_0)^T \tag{219}$$

ただし，N_k，$\overline{\mathbf{x}}_k$，$\mathbf{S}_k$は変分Eステップで計算した負担率r_{nk}を用いて計算できる。

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{220}$$

$$\overline{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{221}$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \overline{\mathbf{x}}_k)(\mathbf{x}_n - \overline{\mathbf{x}}_k)^T \tag{222}$$

GMM-VBのアニメーション

以上で，変分ベイズのGMMへの適用は完了です。初期値の設定や収束判定は，実装の章で確認します。

ここまで読了された賢明な読者の皆さまはお気づきかもしれませんが，変分ベイズの欠点は更新式の導出が煩雑になることです。今回採り上げた混合ガウス分布は比較的単純なケースなのですが，それでも上で確認したように計算がかなり煩雑になってしまいます。このような背景から，ベイズモデリングの実際の応用ではマルコフ連鎖モンテカルロ法（Markov chain Monte Carlo methods: MCMC）などのサンプリング法が用いられることがあります。

実装

本章では，GMMを題材にEMアルゴリズムに基づくクラスタリングと変分ベイズに基づくクラスタリングの実装をお伝えしていきます。具体的には，10000個の３次元データをGMM-EM（EMアルゴリズムによる混合ガウス分布の推論）とGMM-VB（変分ベイズによる混合ガウス分布の推論）を利用してクラスタリングする実装例をお伝えしていきます。要するに，以下のパラメータを仮定します。

$$N = 10000 \tag{223}$$

$$D = 3 \tag{224}$$

実装はGithubで公開しています。

Githubで実装を確認する →

ソースコードのコメントは英語で書いています。これはGithubで公開する際に，外国の方々にも参考にしていただきたいからです。コメント規則であるdocstringはGoogleスタイルを利用しています。

ここからは，以下の形式でメソッド単位で解説を行っていきます。

［メソッド名・その他タイトルなど］

```
# ソースコード
```

［ソースコードの解説］

Github上のコードはDockerを用いて動かせるように整備していますが，下記のコードはGoogle Colaboratoryを用いて気軽に試せるようにしています。ぜひコードをコピペしてお試しください。

データの準備

以下では，GMM-EMとGMM-VBを適用するデータを生成していきます。

```
import sys # 引数の操作
import csv # csvの読み込み
import numpy as np # 数値計算
import matplotlib.pyplot as plt # 可視化
from numpy import linalg as la # 行列計算
from collections import Counter # 頻度カウント
from scipy.special import digamma, logsumexp # 数値計算
from scipy.stats import multivariate_normal # 多次元ガウス分布の確率密度関数の計算
from mpl_toolkits.mplot3d import Axes3D # 可視化
```

標準的なライブラリをインポートします。

```python
# 各クラスターに属するデータ数
# 今回は全てのクラスタでデータ数が同じと仮定する
# 全体のデータ数は N = N1 + N2 + N3 + N4 となる
# 各クラスタのデータ数
N1 = 4000
N2 = 3000
N3 = 2000
N4 = 1000

# 平均
Mu1 = [5, -5, -5]
Mu2 = [-5, 5, 5]
Mu3 = [-5, -5, -5]
Mu4 = [5, 5, 5]

# 共分散
Sigma1 = [[1, 0, -0.25], [0, 1, 0], [-0.25, 0, 1]]
Sigma2 = [[1, 0, 0], [0, 1, -0.25], [0, -0.25, 1]]
Sigma3 = [[1, 0.25, 0], [0.25, 1, 0], [0, 0, 1]]
Sigma4 = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]

# 乱数を生成
X1 = np.random.multivariate_normal(Mu1, Sigma1, N1)
X2 = np.random.multivariate_normal(Mu2, Sigma2, N2)
X3 = np.random.multivariate_normal(Mu3, Sigma3, N3)
X4 = np.random.multivariate_normal(Mu4, Sigma4, N4)
```

データを作成します。今回は4つのガウス分布からデータを生成しましょう。クラスタリングとしては非常に簡単な問題設定になっています。なお，以下で載せている画像は当サイトのカラーリストを利用していますが，今回お伝えする実装は `tab10` のcolormapを利用します。本質的には重要でない部分ですので，気にしなくても大丈夫です。

```python
# 描画準備
fig = plt.figure(figsize=(4, 4), dpi=300)
ax = Axes3D(fig)
fig.add_axes(ax)

# 当サイトのカスタムカラーリスト
cm = plt.get_cmap("tab10")

# メモリを除去
ax.set_xticks([])
ax.set_yticks([])
```

```
ax.set_zticks([])

# 少し回転させて見やすくする
ax.view_init(elev=10, azim=70)

# 描画
ax.plot(X1[:,0], X1[:,1], X1[:,2], "o", ms=0.5, color=cm(0))
ax.plot(X2[:,0], X2[:,1], X2[:,2], "o", ms=0.5, color=cm(1))
ax.plot(X3[:,0], X3[:,1], X3[:,2], "o", ms=0.5, color=cm(2))
ax.plot(X4[:,0], X4[:,1], X4[:,2], "o", ms=0.5, color=cm(3))
plt.show()
```


クラスタリングを行うデータ

可視化して確認していきましょう。

```
# 4つのクラスを結合
X = np.concatenate([X1, X2, X3, X4])

# 描画準備
fig = plt.figure(figsize=(4, 4), dpi=300)
ax = Axes3D(fig)
fig.add_axes(ax)


```

```
# メモリを除去
ax.set_xticks([])
ax.set_yticks([])
ax.set_zticks([])

# 少し回転させて見やすくする
ax.view_init(elev=10, azim=70)

# 描画
ax.plot(X[:,0], X[:,1], X[:,2], "o", ms=0.5, color=cm(0))
plt.show()
```


クラスラベルが分かっていない状況でクラスタリングを行う

今回は，4つのクラスラベルを排除したものをクラスタリング対象としますので，データを結合しましょう。

```
# csvでデータを保存
np.savetxt("data.csv", X, delimiter=",")
```

毎回同じデータに対してクラスタリングを行うため，データをcsvに吐き出しておきます。当サイトのデモで利用しているcsvは以下でダウンロードできます。

csvをダウンロードする →

```
# ご自身の環境におけるcsvへのパス
csv_dir = "[path to data.csv]"

# csvを読み取って(N, D)行列を生成
with open(csv_dir) as f:
    reader = csv.reader(f)
    X = [_ for _ in reader]
    for i in range(len(X)):
      for j in range(len(X[i])):
        X[i][j] = float(X[i][j])

# 後のためにnumpy化しておく
X = np.array(X)
```

当サイトと全く同じデータを利用したい場合は，上記ボタンからcsvをダウンロードしていただき，csvからデータを生成してください。ただし，`[path to data.csv]` にはご自身の環境におけるcsvファイルへのパスを挿入してください。この後から，EMアルゴリズムと変分ベイズで実装が分岐していきます。

EMアルゴリズム

GMMEMクラスの定義

詳しい実装内容は「EMアルゴリズムをはじめからていねいに」をご参照下さい。

EMアルゴリズムをはじめからていねいに →

GMMEMの実行

上で定義したGMMEMクラスのインスタンスを生成して，GMM-EMを実行しましょう。以下では，GMM-VBとの比較を行うためにいくつかの条件で実行してみます。

$K = 4$の場合

```
# モデルをインスタンス化する
model = GMMEM(K=4)
# EMアルゴリズムを実行する
model.execute(X, iter_max=100, thr=0.001)
```

クラスタ数Kは4，最大更新回数は100回としました。収束判定における対数尤度増加幅の閾値は0.001としました。

$K = 4$の場合

しっかりとクラスタリングできていますね。ログを確認すると，更新回数は10回だったことが分かります。

```
Log-likelihood gap: 19.66
Log-likelihood gap: 11.95
Log-likelihood gap: 4.53
Log-likelihood gap: 0.04
Log-likelihood gap: 0.74
Log-likelihood gap: 0.57
Log-likelihood gap: 0.01
Log-likelihood gap: 2.94
Log-likelihood gap: 1.65

```

```
Log-likelihood gap: 0.0
EM algorithm has stopped after 10 iteraions.
```

$K = 8$の場合

```
# モデルをインスタンス化する
model = GMMEM(K=8)
# EMアルゴリズムを実行する
model.execute(X, iter_max=100, thr=0.001)
```

クラスタ数Kは8として，他のパラメータは先ほどと同様の設定で行いました。

$K = 8$の場合

正しくクラスタリングされませんでした。ログを確認すると，更新回数は最大の100回となっていたため，うまく収束できなかったようです。点推定では，4つに分かれているクラスタを$K = 8$の条件下では正しく推定できなかったということです。

```
Log-likelihood gap: 23.37
Log-likelihood gap: 8.06
Log-likelihood gap: 2.36
Log-likelihood gap: 3.5
```

```
Log-likelihood gap: 3.02
~~~~~~~~~~
省略
~~~~~~~~~~
Log-likelihood gap: 0.22
Log-likelihood gap: 0.2
Log-likelihood gap: 0.19
Log-likelihood gap: 0.18
Log-likelihood gap: 0.17
EM algorithm has stopped after 100 iteraions.
```

データの中に実質的な情報が僅かしか含まれていない性質のことを「スパース性」と呼びます。今回の場合は，仮定したクラス数よりも実際に推定するクラス数の方が少なく，EMアルゴリズムがスパース性を考慮できなかったために，クラスラベルを正しく推定することができなかったと考えられます。変分ベイズでは，後に述べる「関連度自動決定」により，スパース性を担保したクラスタリングが可能になります。

変分ベイズ

GMMVBクラスの定義

```
class GMMVB():
```

ここからは，GMMVBクラスを定義していきます。

コンストラクタ

```
    def __init__(self, K):
        """コンストラクタ

        Args:
            K (int): クラスタ数

```

```
        Returns:
            None.

        Note:
            eps (float): オーバーフローとアンダーフローを防ぐための微小量
        """
        self.K = K
        self.eps = np.spacing(1)
```

クラスタ数 `K` と微小量 `eps` を定義します。 `eps` は0除算を防ぐため等に利用します。

パラメータ初期化メソッド

```
    def init_params(self, X):
        """パラメータ初期化メソッド

        Args:
            X (numpy ndarray): (N, D)サイズの入力データ

        Returns:
            None.
        """
        # 入力データ X のサイズは (N, D)
        self.N, self.D = X.shape
        # スカラーのパラメータセット
        self.alpha0 = 0.01
        self.beta0 = 1.0
        self.nu0 = float(self.D)
        # 平均は標準ガウス分布から生成
        self.m0 = np.random.randn(self.D)
        # 分散共分散行列は単位行列
        self.W0 = np.eye(self.D)
        # 負担率は標準正規分布から生成するがEステップですぐ更新するので初期値自体には意
        self.r = np.random.randn(self.N, self.K)
        # 更新対象のパラメータを初期化
        self.alpha = np.ones(self.K) * self.alpha0
        self.beta = np.ones(self.K) * self.beta0
        self.nu = np.ones(self.K) * self.nu0
        self.m = np.random.randn(self.K, self.D)
        self.W = np.tile(self.W0[None,:,:], (self.K, 1, 1))
```

スクロールできます ⟶

変数	サイズ	意味
`N`	`int`	データ数
`D`	`int`	データの次元
`alpha0`	`float`	ディリクレ分布のパラメータ初期値
`beta0`	`float`	ガウス分布の分散共分散行列の係数初期値
`nu0`	`float`	ウィシャート分布のパラメータ初期値
`m0`	`(K, D)`	ガウス分布の平均
`W0`	`(K, D, D)`	ガウス分布の分散共分散行列
`r`	`(N, K)`	負担率
`alpha`	`(K)`	ディリクレ分布のパラメータ
`beta`	`(K)`	ガウス分布の分散共分散行列の係数
`nu`	`(K)`	ウィシャート分布のパラメータ
`m`	`(N, D)`	ガウス分布の平均
`W`	`(K, D, D)`	ウィシャート分布のパラメータ

フィールドで宣言している変数

パラメータの初期化を行います。

ディリクレ分布パラメータα_0の初期値は0.01としました。α_0はπ_kの事前分布のパラメータですので，素朴に考えると$1/K$が適切のように思えますが，変分ベイズではスパース性を担保した推定が行えることから，初期値として小さな値を採用しています。

ガウス分布のパラメータβ_0の初期値は1.0としました。β_0は$\mathbf{\Sigma}_k$のスケールを表していますので，初期値としては等倍である1.0を選択しました。

ウィシャート分布のパラメータν_0の初期値は次元数Dとしました。上では指摘していませんが，ウィシャート分布におけるνは$D-1$よりも大きくなければいけません。ν_0として1.0や0を与えてしまうと，推定が不安定な挙動を示してしまうため注意が必要です。

GMMの確率密度関数計算メソッド

```
    def gmm_pdf(self, X):
        """N個のD次元データに対してGMMの確率密度関数を計算するメソッド

        Args:
            X (numpy ndarray): (N, D)サイズの入力データ

        Returns:
            Probability density function (numpy ndarray): 各クラスタにおけるN個の
        """
        pi = self.alpha / (np.sum(self.alpha, keepdims=True) + np.spacing(1)) #
        return np.array([pi[k] * multivariate_normal.pdf(X, mean=self.m[k], cov
```

GMMの確率密度関数を計算します。EMアルゴリズムでは混合ガウス分布の混合率π_k・平均$\boldsymbol{\mu}_k$・分散共分散行列$\boldsymbol{\Sigma}_k$は更新対象のパラメータでしたが，変分ベイズではπ_k・$\boldsymbol{\mu}_k$・$\boldsymbol{\Sigma}_k$には事前分布を設定しているために，それら自体は更新対象のパラメータではありません。言い換えれば，現時点ではGMMの確率密度関数を計算するためのπ_k・$\boldsymbol{\mu}_k$・$\boldsymbol{\Sigma}_k$が手に入っていないということです。

このような背景から，π_k・$\boldsymbol{\mu}_k$・$\boldsymbol{\Sigma}_k$はそれぞれの事前分布から代表値を抽出してあげる必要があります。代表値として何を用いるのかは自明ではありませんが，ここでは単純に期待値を採用します。式(70)と式(79)より，$\boldsymbol{\pi}$の期待値はディリクレ分布の期待値になります。

$$E[\pi_k] = \frac{\alpha_k}{\sum_{j=1}^{K} \alpha_j} \tag{225}$$

同様に，式(71)と式(84)より，$\boldsymbol{\mu}$の期待値はガウス分布の期待値になります。

$$E[\boldsymbol{\mu}_k] = \mathbf{m}_0 \tag{226}$$

同様に，式(72)と式(86)より，$\boldsymbol{\Sigma}$の期待値はウィシャート分布の期待値になります。

$$E[\boldsymbol{\Sigma}_k] = \nu_k \mathbf{W}_k \tag{227}$$

これらの代表値を用いて，GMMの確率密度関数を計算します。計算自体はEMアルゴリズムと同様ですので，詳細は割愛します。

変分Eステップメソッド

```
    def e_step(self, X):
        """変分Eステップを実行するメソッド

        Args:
            X (numpy ndarray): (N, D)サイズの入力データ

        Returns:
            None.

        Note:
            以下のフィールドが更新される
                self.r (numpy ndarray): (N, K)サイズの負担率
        """
        # rhoを求めるために必要な要素の計算
        log_pi_tilde = (digamma(self.alpha) - digamma(self.alpha.sum()))[None,:
        log_sigma_tilde = (np.sum([digamma((self.nu + 1 - i) / 2) for i in rang
        nu_tile = np.tile(self.nu[None,:], (self.N, 1)) # (N, K)
        res_error = np.tile(X[:,None,None,:], (1, self.K, 1, 1)) - np.tile(self
        quadratic = nu_tile * ((res_error @ np.tile(self.W[None,:,:,:], (self.N
        # 対数領域でrhoを計算
        log_rho = log_pi_tilde + (0.5 * log_sigma_tilde) - (0.5 * self.D / (sel
        # logsumexp関数を利用して対数領域で負担率を計算
        log_r = log_rho - logsumexp(log_rho, axis=1, keepdims=True) # (N, K)
        # 対数領域から元に戻す
        r = np.exp(log_r) # (N, K)
        # np.expでオーバーフローを起こしている可能性があるためnanを置換しておく
        r[np.isnan(r)] = 1.0 / (self.K) # (N, K)
        self.r = r # (N, K)
```

変分Eステップを実行します。具体的には，以下を計算した後に，

$$\ln \tilde{\pi}_k = \psi(\alpha_k) - \psi\left(\sum_{k=1}^{K} \alpha_k\right) \tag{228}$$

$$\ln \tilde{\Sigma}_k = \sum_{i=1}^{D} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + D\ln(2) + \ln|\mathbf{W}_k| \tag{229}$$

$$\rho_{nk} = \tilde{\pi}_k \tilde{\mathbf{\Sigma}}_k^{\frac{1}{2}} \exp\left\{-\frac{D}{2\beta_k} - \frac{\nu_k}{2}(\mathbf{x}_n - \mathbf{m}_k)^T \mathbf{W}_k (\mathbf{x}_n - \mathbf{m}_k)\right\} \tag{230}$$

その結果を用いて以下を計算します。

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \tag{231}$$

EMアルゴリズム同様，オーバーフローを防ぐために対数領域で計算を行います。早速，負担率の対数を取ってみましょう。

$$\ln r_{nk} = \ln \rho_{nk} - \ln \sum_{j=1}^{K} \rho_{nj} \tag{232}$$

EMアルゴリズム同様，$\ln \rho_{nk}$の計算は行列演算を用いて高速化を図ります。

rhoの行列演算

$(1, K)$と(N, K)同士の和算・減算がありますが，ここでは$(1, K)$側を自動で(N, K)側に揃えてくれるブロードキャストという `numpy` の機能を利用します。

改めて式(230)をみると，第二項目に対数領域ではないρ_{nk}が含まれています。ρ_{nk}には指数演算が含まれていますので，ρ_{nk}自体の計算は対数領域で行ったうえで，一番最後に指数演算を施して元に戻してあげた方がベターです。オーバーフローのリスクはなるべく一か所に集中させてあげた方が良いという思想です。

そこで，以下の変形を無理やり行うことで，第二項目でも$\ln \rho_{nk}$を利用して計算を行うことが可能になります。

$$\ln r_{nk} = \ln \rho_{nk} - \ln \sum_{j=1}^{K} \exp\left(\ln \rho_{nj}\right) \tag{233}$$

ここで，第二項目に現れた$\ln \sum \exp$は `scipy.special` の `logsumexp` というモジュールを利用することができます。

`logsumexp` は対数領域で計算を行う場合によく出現する項です。今回のケースのように，オーバーフローを防ぐための施策として頻繁に用いられる汎用的な手法ですので，ここでおさえておくとよいでしょう。

Mステップメソッド

```
    def m_step(self, X):
        """変分Mステップを実行するメソッド

        Args:
            X (numpy ndarray): (N, D)サイズの入力データ

        Returns:
            None.
```

```
        Note:
            以下のフィールドが更新される
                self.alpha (numpy ndarray): (K) サイズのディリクレ分布のパラメータ
                self.beta (numpy ndarray): (K) ガウス分布の分散共分散行列の係数
                self.nu (numpy ndarray): (K) サイズのウィシャート分布のパラメータ
                self.m (numpy ndarray): (K, D) サイズの混合ガウス分布の平均
                self.W (numpy ndarray): (K, D, D) サイズのウィシャート分布のパラメ
        """
        # 各パラメータを求めるために必要な要素の計算
        N_k = np.sum(self.r, 0) # (K)
        r_tile = np.tile(self.r[:,:,None], (1, 1, self.D)).transpose(1, 2, 0) #
        x_bar = np.sum((r_tile * np.tile(X[None,:,:], (self.K, 1, 1)).transpose
        res_error = np.tile(X[None,:,:], (self.K, 1, 1)).transpose(0,2,1) - np.
        S = ((r_tile * res_error) @ res_error.transpose(0,2,1)) / (N_k[:,None,N
        res_error_bar = x_bar - np.tile(self.m0[None,:], (self.K, 1)) # (K, D)
        # 各パラメータを更新
        self.alpha = self.alpha0 + N_k #(K)
        self.beta = self.beta0 + N_k #(K)
        self.nu = self.nu0 + N_k #(K)
        self.m = (np.tile((self.beta0 * self.m0)[None,:], (self.K, 1)) + (N_k[:
        W_inv = la.pinv(self.W0) + (N_k[:,None,None] * S) + (((self.beta0 * N_k
        self.W = la.pinv(W_inv) # (K, D, D)
```

変分Mステップを実行します。変分EステップはEMアルゴリズムと同様に負担率を求めるだけでしたので，負担率の計算方法を除いて特に変わりはありませんでした。一方で，変分Mステップでは更新対象のパラメータがごっそり変わるため実装もかなり変わってきます。具体的には，以下を計算した後に，

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{234}$$

$$\overline{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{235}$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \overline{\mathbf{x}}_k)(\mathbf{x}_n - \overline{\mathbf{x}}_k)^T \tag{236}$$

その結果を用いて以下を計算します。

$$\alpha_k = \alpha_0 + N_k \tag{237}$$

$$\beta_k = \beta_0 + N_k \tag{238}$$

$$\nu_k = \nu_0 + N_k \tag{239}$$

$$\mathbf{m}_k = \frac{1}{\beta_k}(\beta_0 \mathbf{m}_0 + N_k \overline{\mathbf{x}}_k) \tag{240}$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\overline{\mathbf{x}}_k - \mathbf{m}_0)(\overline{\mathbf{x}}_k - \mathbf{m}_0)^T \tag{241}$$

EMアルゴリズム同様，$\mathbf{S}_k$の計算は行列演算を用いて高速化を図ります。

Sの行列演算

同様に，$\mathbf{W}_k^{-1}$の計算も行列演算を用いて高速化を図ります。

W_invの行列演算

注意が必要なのは，フィールドでは `self.W` を逆行列として定めていないという点です。式(241)は逆行列$\mathbf{W}^{-1}$に対する更新式ですので，`self.W` をアップデートする際は逆行列にしてあげる必要があります。

逆行列の計算ですが，`la.inv` を利用すると行列が正則でない場合に逆行列が計算できずにエラーが出てしまいます。そこで，正則でない行列に対しても逆行列（に似た性質の行列）を計算できる `la.pinv` を利用します。また，除算を行う際は0除算を防ぐために微小量 `eps` を分母に加算していることに注意してください。

可視化メソッド

```
    def visualize(self, X):
        """可視化を実行するメソッド

        Args:
            X (numpy ndarray): (N, D)サイズの入力データ

        Returns:
            None.

        Note:
            このメソッドでは plt.show が実行されるが plt.close() は実行されない
        """
        # クラスタリングを実行
        labels = np.argmax(self.r, 1) # (N)
        # 利用するカラーを極力揃えるためクラスタを出現頻度の降順に並び替える
        label_frequency_desc = [l[0] for l in Counter(labels).most_common()]
        # tab10 カラーマップを利用
        cm = plt.get_cmap("tab10")
        # 描画準備
        fig = plt.figure(figsize=(4, 4), dpi=300)
        ax = Axes3D(fig)
        fig.add_axes(ax)
        # メモリを除去
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_zticks([])
        # 少し回転させて見やすくする
        ax.view_init(elev=10, azim=70)
        # 各クラスタごとに可視化を実行する
        for k in range(len(label_frequency_desc)):
```

```
            cluster_indexes = np.where(labels==label_frequency_desc[k])[0]
            ax.plot(X[cluster_indexes, 0], X[cluster_indexes, 1], X[cluster_inc
        plt.show()
```

VBが収束した後に，各クラスごとに色分けした可視化を行い結果を確認します。EMアルゴリズムと全く同じメソッドです。

実行メソッド

```
    def execute(self, X, iter_max, thr):
        """VBを実行するメソッド

        Args:
            X (numpy ndarray): (N, D)サイズの入力データ
            iter_max (int): 最大更新回数
            thr (float): 更新停止の閾値（対数尤度の増加幅）

        Returns:
            None.
        """
        # パラメータ初期化
        self.init_params(X)
        # 各イテレーションの対数尤度を記録するためのリスト
        log_likelihood_list =[]
        # 対数尤度の初期値を計算
        log_likelihood_list.append(np.mean(np.log(np.sum(self.gmm_pdf(X), 1) +
        # 更新開始
        for i in range(iter_max):
            # Eステップの実行
            self.e_step(X)
            # Mステップの実行
            self.m_step(X)
            # 今回のイテレーションの対数尤度を記録する
            log_likelihood_list.append(np.mean(np.log(np.sum(self.gmm_pdf(X), 1
            # 前回の対数尤度からの増加幅を出力する
            print("Log-likelihood gap: " + str(round(np.abs(log_likelihood_list
            # もし収束条件を満たした場合，もしくは最大更新回数に到達した場合は更新停止
            if (np.abs(log_likelihood_list[i] - log_likelihood_list[i+1]) < thr
                print(f"VB has stopped after {i + 1} iteraions.")
                self.visualize(X)
                break
```

今まで定義してきたメソッドを駆使して，GMM-VBを実行します。初期化，変分Eステップ，変分Mステップ，収束判定と繰り返すことで，パラメータを更新していきます。収束判定では，「閾値」もしくは「最大更新回数」のいずれかが引っかかった場合に更新をストップするようにしています。EMアルゴリズム同様，閾値判定に用いる対数尤度は，全てのデータ点の和ではなく平均を利用しています。結局，EMアルゴリズムと全く同じメソッドです。

> EMアルゴリズムの目的関数は対数尤度でしたので，収束条件に対数尤度を設定するのは合理的です。一方で，変分ベイズの目的関数はKLダイバージェンスもしくは下限でしたので，収束条件に対数尤度を設定するのは一見合理的ではありません。しかし，対数尤度が「パラメータが与えられたときの観測データの尤もらしさ」を表すことを踏まえれば，変分ベイズでも収束条件に対数尤度を設定するのは十分合理的です。

GMMVBの実行

上で定義したGMMVBクラスのインスタンスを生成して，GMM-VBを実行しましょう。以下では，GMM-VBとの比較を行うためにいくつかの条件で実行してみます。

$K=4$の場合

```
# モデルをインスタンス化する
model = GMMVB(K=4)
# VBを実行する
model.execute(X, iter_max=100, thr=0.001)
```

クラスタ数Kは4，最大更新回数は100回としました。収束判定における対数尤度の増加幅の閾値は0.001としました。

$K = 4$の場合

しっかりとクラスタリングできていますね。ログを確認すると，更新回数は10回だったことが分かります。

```
Log-likelihood gap: 28.41
Log-likelihood gap: 0.08
Log-likelihood gap: 0.05
Log-likelihood gap: 0.01
Log-likelihood gap: 0.01
Log-likelihood gap: 0.03
Log-likelihood gap: 0.07
Log-likelihood gap: 0.14
Log-likelihood gap: 0.07
Log-likelihood gap: 0.0
VB has stopped after 10 iteraions.
```

$K = 8$の場合

```
# モデルをインスタンス化する
model = GMMVB(K=8)


```

```
# VBを実行する
model.execute(X, iter_max=100, thr=0.001)
```

クラスタ数Kは8として，他のパラメータは先ほどと同様の設定で行いました。

$K = 8$の場合

しっかりとクラスタリングできていますね。ログを確認すると，更新回数は6回だったことが分かります。

```
Log-likelihood gap: 12.22
Log-likelihood gap: 2.33
Log-likelihood gap: 1.33
Log-likelihood gap: 0.08
Log-likelihood gap: 0.01
Log-likelihood gap: 0.0
VB has stopped after 6 iteraions.
```

EMアルゴリズムのときとは異なり，クラスタ数が過剰に設定されている場合でも変分ベイズでは正しくクラスタリングが行われていることが分かりました。このように，変分ベイズがスパースなクラスタ分類を行うはたらきを「関連度自動決定」と呼びます。クラスタの個数を大きめ（$K = 8$など）に設定すれば，勝手に適切なクラスタ数にフィットしてくれるというのです。

周辺尤度最大化するときにデータの傾向にそぐわない基底関数は周辺尤度を減少させる方向に寄与するため，勝手にクラスタ数をフィットしてくれるという訳です。詳しくは[1]の7.2.2章をご覧ください。

おわりに

本稿では，変分ベイズの目的とその実現方法を，点推定との比較を通じてボトムアップ的にお伝えしてきました。冒頭でも述べたように，変分ベイズは機械学習を学ぶ上で高く反り立つ壁です。多くの人が挫折を経験したことは想像に難くありません。本稿が，一人でも多くの人にとって変分ベイズを学ぶ助けとなれることを心から願っております。

参考文献

[1] Christopher M. Bishop, "Pattern Recognition and Machine Learning."

付録

本章では，EMアルゴリズムと変分ベイズのEステップとMステップの対応を考えます。

Zを構成する確率変数を$Z_0, \ldots, Z_K$と表すことにします。EMアルゴリズムでは潜在変数とパラメータは区別して，変分ベイズでは潜在変数にパラメータを含めるのでした。したがって，$Z_0, \ldots, Z_K$の中に，EMアルゴリズムにおけるθが対応していることに注意してください。本稿では，潜在変数とそれ以外を明示的に区別するため，Z_0を潜在変数，それ以外の$Z_1, \ldots, Z_K$をパラメータと設定することにします。

GMMを例にした潜在変数の対応関係

変分ベイズは，言ってしまえば式(15)を計算するだけです。式(15)に基づけば，あるZ_kに対する更新を行うときは，Z_k以外の値を固定します。そのため，変分ベイズはEMアルゴリズムのような交互更新を行う必要があります。$K+1$個のZに対して交互更新を行うため，理論上は$K+1$ステップ必要になります。

EMアルゴリズムと変分ベイズを比較する文脈では，変分Eステップと変分Mステップという用語が登場します。例えば，変分ベイズを混合ガウス分布に適用する場合，一般的には負担率を計算する変分Eステップとパラメータを更新する変分Mステップに分けられます。

変分ベイズにおいては，Mステップは存在しないというのが私の考えです。EMアルゴリズムにおけるMステップは，対数尤度を最大化するパラメータを点推定するステップでしたが，変分ベイズではパラメータを点推定するフェーズは存在しないからです。

多くの書籍やWeb上の資料では変分Eステップ・変分Mステップという用語が利用されているため，本稿では変分ベイズとEMアルゴリズムを対応付けたいと思います。いま，K種類のパラメータ$\boldsymbol{\theta}$と潜在変数Zが存在するとします。すると，EMアルゴリズムは，K種類のパラメータを固定して潜在変数Zに関する最適化を行うEステップと，潜在変数Zを固定してK種類のパラメータを点推定するMステップで構成されます。このような立場に立つと，EMアルゴリズムは**EM…Mアルゴリズム**と捉えることができます（MはK回連続しています）。

同じ立場に立てば，変分ベイズは$k+1$種類の確率変数Zがある状況では**EE…Eアルゴリズム**と捉えることができます（Eは$K+1$回連続しています）。なぜなら，変分ベイズが式(15)を計算するだけだからです。EMアルゴリズムにおけるK回のMステップが，全てEステップに

置き換えられていますね。言ってしまえば，自分以外の期待値を取る操作を収束するまで繰り返すのが変分ベイズです。

EM...MアルゴリズムとEE...Eアルゴリズム

特に「EM」アルゴリズムと比較する場合は，潜在変数Zとそれ以外のパラメータθを別々に考えて，潜在変数以外の確率変数に関する式(15)を計算してZに関するKLを最小化するE_Zステップと，θ以外の確率変数に関する式(15)を計算してθに関するKLを最小化するE_θステップに無理矢理二分割します。

VBを無理矢理二分割

このE_Zステップが巷では変分Eステップと呼ばれており，E_θステップが変分Mステップと呼ばれているようです。しかし，私はこの呼び方が変分ベイズに対して誤解を生んでいる気がしてなりません。EMアルゴリズムとは違って，変分Mステップでは上限を最大化している訳ではないからです。あくまでも，両手法のMステップにおいては，潜在変数以外の確率変数

（EMアルゴリズムで言えばパラメータという値）を最適化しているという対応関係しかありません。

変分Eステップと変分Mステップ

上図においては，先ほど設定したようにZ_0は潜在変数を表しています。Z_0以外のパラメータは，先ほど説明した平均場近似の仮定をおくことで分離されているものとしています。変分Eステップでは，潜在変数Z_0に関して最適なパラメータ（**負担率**と呼ぶ）を求めておきます。

この時点では，実際にZ_0にとって最適なパラメータを探しただけであって，実際に更新はしていません。Z_0は他のパラメータとは異なり潜在変数です。パラメータの更新を行うのは変分Mステップであり，Z_0以外の因子についての最適化を行うことで変分下限を最大化します。このとき，変分Eステップで求めておいた負担率を利用することになるため，EMアルゴリズムと同様の交互更新になります。

しかし，本質的には変分ベイズでは潜在変数もパラメータも確率変数とみなしていますので，潜在変数とパラメータの立場は対等なはずです。変部ベイズにおいて「潜在変数とそれ以外」を区別する必要性が，私はあまりないように感じています。逆に言えば，両者を同一視した概念が変分ベイズだと思っています。

多くの書籍やWeb上の資料では，負担率という言葉だけが一人歩きしているように思えます。負担率の本質はEステップで計算される値であり，Mステップのパラメータ更新で利用される値であるということです。