**ディープラーニングのための確率的プログラミング**

ダスティン・トラン

学位取得のための要件の一部を満たすために提出した。

実行委員会傘下の哲学博士

大学院総合文化研究科の

コロンビア大学

2020

© 2020

ダスティン・トラン

All Rights Reserved

**抄録**

ディープラーニングのための確率的プログラミング

ダスティン・トラン

我々は、確率的モデリングと深層学習の交差点にあるシステムのための進歩を統合した、*深層確率プログラミングの*アイデアを提案する。このようなシステムは、そうでなければ不可能であった新しい確率的モデルや推論アルゴリズムの開発を可能にします：数十億のパラメータへの前例のないスケール、分散・混合精度環境、AIアクセラレータ、大規模・高次元データセットのモデリングのためのニューラルアーキテクチャとの統合、柔軟な推論やモデル批判のための確率的プログラムの自動微分や任意の操作のための計算グラフの使用など。

深層確率計画法を説明した後、新しい変分推論アルゴリズムや深層確率モデルへの応用について議論する。まず、我々は、複雑な事後分布に適合するようにその形状を適応させるベイズ的ノンパラメトリック変分群である変分ガウス過程(vgp)を紹介する。vgpは、潜在入力を生成し、ランダムな非線形マッピングを通して歪ませることで、近似的な事後サンプルを生成します。ランダムマッピング上の分布は、推論中に学習され、変換された出力が真の事後分布の複雑さの変化に適応できるようにします。第二に、我々は階層的暗黙モデル(HIMS)を導入する。HIMSは、暗黙の密度の考え方と階層的ベイズモデリングを組み合わせたもので、それによって、豊富な隠れた構造を持つデータのシミュレータを介してモデルを定義することができる。

**目次**

テーブル一覧 **.** vi

VII

xi

1

1

3

1.1.2 確率モデルの推論 . 3

4

4

6

6

1.1.7 モデル批判 . 9

1.2 確率論的システムの歴史 . 11

第2章: 深層確率計画法 **.** 13

13

14

2.2.1 サンプル. 15

2.2.2 例.可変長のベイジアン・リカレント・ニューラル・ネットワーク . 16

16

17

18

2.3.2 推論のクラス . 19

2.3.3 推論の構成 . 21

2.3.4 データのサブサンプリング . 21

22

23

2.4.2 GPU加速ハミルトニアンモンテカルロ法. 24

25

第3章： 単純、分散、高速化された確率プログラミング **.** 26

26

27

3.2.1確率的 プログラム、変分プログラムなど . 27

3.2.2 例.TPU を使用したモデル-パラレル VAE . 29

29

3.2.4 トレーシングアプリケーション . 32

3.3 例。低レベル関数の学習 . 32

3.3.1 例.TPU を用いたデータ並列画像変換器 . 33

3.3.2 例. 34

3.3.3 サンプル. 35

3.3.4 例.勾配降下による変量推論による学習 . 35

36

3.4.1 高品質画像の生成 . 36

38

38

40

40

41

41

42

43

4.2.4 万能近似定理 . 45

46

46

47

48

49

50

4.4.1 Binarized MNIST . 50

51

52

53

53

54

5.3尤 度自由変量推論 . 57

58

58

5.3.3 KL目的語の確率的勾配 . 59

5.3.4 アルゴリズム . 61

62

65

参考文献 **.** 81

82

A.1 モデル例 . 82

A.2 分類のためのベイジアンニューラルネットワーク . 82

82

A. 4ガウス行列因数分解n . 83

83

A.6 推論の例. 84

A.7 完全な例 . 86

86

86

89

89

B.2 文法変分自動エンコーダー . 90

92

93

95

C.1変分 ガウス過程の特殊な場合 . 95

96

C. 3変量目的語 . 97

99

C.5変 分データのサイズをスケーリングする . 100

101

D.1ノイズ対潜伏変数. 101

102

D.3 KLの独自性 . 102

D.4 Hingeの損失 . 104

D.5 MAPを用いたベイジアンGANとMLEを用いたGANを比較する . 104

105

# テーブル一覧

2.12 値化されたMNIST上の確率的復号器のための推論手法エドワード

確率的プログラミング言語(pl)は、便利な研究プラットフォームである。

多くのアルゴリズムの開発と実験を簡単に行うことができます。 23

2.2 大規模ロジスティック回帰のためのハミルトニアンモンテカルロ法(hmc)のベンチマーク.エド

ward (GPU) は他のシステムに比べて大幅に高速です。さらに、エドワードには

オーバーヘッド：手書きのTensorFlowと同じくらいの速さです。 . 24

3.1 ベイズロジスティック回帰におけるNo-U-Turn Samplerのリープフロッグステップあたりの時間エド

ward2 (GPU) は PyMC3 (CPU) に比べて 37 倍の高速化を実現。

のEdwardでは、手書きのTensorFlowよりも無視できるほどのオーバーヘッドが発生します。また、Edward2 は手書きの TensorFlow よりも無視できるほどのオーバーヘッドが発生します。

コード . 37

4.1 二値化MNISTの負の予測対数尤度。これまでの最良の結果は

1]（Burdaら、2016a）、[2]（Salimansら、2015）、[3]（Rezende and Mohamed.

2015）、[4]（Raikoら、2014）、[5]（Murray and Salakhutdinov、2009）、[6]（Gregor

ら、2014年）、[7]（Gregorら、2015年）。 . 51

4.2 スケッチの負の予測対数尤度、何百ものエポックで学習された

すべての18,000のトレーニング例 52

5.1 ベイジア ンenerativeadversarialnetwork(gan)とベイジアンの 分類精度

ニューラルネットワークは、小規模から中規模のデータセットにまたがっています。ベイジアンギャンはコム

ベイズ・ニューラル・ネットの相手に例えれば、それ以上の性能を発揮することができます。. . . . . . 63

# フィギュア一覧

|  |  |
| --- | --- |
| 1.1 Boxのループ。 .  1. 2生成されたデータセットに複製された検定統計量の分布と検定値 | 2 |
| 観察されたデータセットに適用される統計量. .  2.1 Beta-Bernoulliプログラム**（左）と**その計算グラフ**（右）**。**xの**取得 | 11 |
| グラフから50要素の2進ベクトルを生成します。 .  2.2 28×28 ピクセル画像のデータセットに対する変分自動エンコーダー。(**左)**グラフィカルモデル(推論モデルは点線**)； (右)**確率的プログラム(2層) | 14 |
| ニューラルネットワーク。 . | 15 |

2.3 ベイジアンリカレントニューラルネットワーク(rnn)。(**左)**グラフィカルモデル**; (右)**確率的モデル



|  |  |
| --- | --- |
| 2.4確率的 な制御フローを持つ確率的プログラムの計算グラフ. .. .  2.5 階層モデル。(**左)**グラフィカルモデル、**(右)**確率プログラム。*D*次元データ{*xn*}∈*RN×D*上のガウシアンの混合物である。*K個の*潜在 | 17 |
| クラスターとは、*β*∈*RK×Dを*意味します。 . | 18 |
| 2.6 **(左)** 変分推論.**(右)** モンテカルロ法. .  2.7 生成的敵対者ネットワーク。(**左)**グラフィカル・モデル、**(右)**確率的プログラム。モデル(生成器)は，パラメータ化された関数(判別器)を用いて | 19 |
| トレーニングを行うことができます。 | 20 |
| 2.8 推論アルゴリズムを組み合わせて変分的EMを実行する. .  2.9 階層モデルによるデータ・サブサンプリング。我々は，完全モデルの部分グラフを定義し，*N*ではなく*Mの*サイズのプレートを形成する． | 21 |
| *N/M*. | 22 |
| 2.10 ベイズロジスティック回帰のためのエドワードプログラム | 24 |

|  |  |
| --- | --- |
| 3.1 ベータベルヌーイプログラム。イガーモードでは、model()は50要素のバイナリベクトルを生成します。グラフモードでは、model()はTensorFlowで評価されるopを返します。 |  |
| セッションに参加することができます。 .  3.2 イーガーモードで利用可能な変動プログラム(Ranganath et al., 2016a)。Pythonの制御フローは生成プロセスに適用可能です：コインフリップが与えられると、プログラムは2つのニューラルネットのうちの1つから生成します。それらの出力は、異なる形状を持つことができます( | 28 |
| 構造)を使用しています。  3.3 分散自己回帰フロー。**(右)** デフォルトの長さは8で，それぞれが4つの独立したフローを持つ．各フローは、自己回帰的順序を尊重したレイヤーを介して入力を変換する。各コアは2つのフローを計算し、ローカルに接続されている。仮想トポロジーは、物理的なテンソル処理ユニット（TPU）のトポロジーと一致している。 | 28 |
| は正確であり，16x16 tpusの場合はデータ並列化のために複製されます． .  3.4 tpusを用いたモデル-並列変分自動エンコーダー(vae)は，8ビットの潜数から16ビットの音声を生成する．プリファレンスとデコーダは，分散自己回帰フローに従って計算を分割します．エンコーダは圧縮機に応じて計算を分割することができる。 | 30 |
| 我々はスペースのためにそれを省略します。  3.5 最小限のトレースの実装 trace はコンテキストを定義します。 | 30 |
| エドワードのランダム変数を登録します。 .  3.6 Aプログラムの実行。有向非周期グラフ であり、様々な操作のためにトレースされています。 | 31 |
| 対数確率の蓄積や条件付独立性の発見など。. . . .  3.7モデルプログラムを入力として 受け取り，その対数ジョイントを返す 高次関数 | 31 |
| 密度関数を使用しています。 .  3.8 モデルプログラムを入力とし、その因果関係のあるプログラムを返す高次関数。介入は条件付けとは異なります。 | 31 |
| サンプリングされた値が、分布  3.9 tpusを用いたデータ並列画像変換器（Parmar et al.これは、自己注意を伴う画像のバッチの対数確率を計算するニューラル自己回帰モデルである。我々の軽量設計により、モデルを対数確率関数として表現し、訓練することが可能である。埋め込み関数と自己注意関数は，Tensor2Tensor (Vaswani et al. | 31 |
| 2018).  3.10 No-U-Turn Samplerのコアロジック(Hoffman and Gelman, 2014)。このアルゴリズムには | 33 |
| データに依存した非尾部再帰. . | 34 |

|  |  |
| --- | --- |
| 3.11 学習では，モデルプログラム**（左）**と変分プログラム（**右）**，またはモデルプログラムとデータテンソル**（下）**のような2つの実行トレースをマッチングさせることがよくあります．赤色の矢印は、先行変数と変量変数を並べたものです。青色の矢印は、観測された |  |
| 変数とデータ；データから変量変数へのエッジは償却を表します。  3.12 前提条件付き勾配降下による変分推論。Edward2は，確率的プログラムを書き，任意のTensorFlow計算を実行することで | 34 |
| を学習します。  3.13 Learning-to-learn.学習のための最適な前提条件を以下の方法で求める（図3.12）． | 36 |
| 前提条件器に関して学習アルゴリズム全体を微分することができる。 .. | 36 |
| 3.14 64x64 ImageNet のベクトル量子化された VAE. . | 37 |
| 3.15 256x256 CelebA-HQのイメージトランス. .  4.1 **(a)** 変分ガウス過程のグラフィカルモデル.vgpは，潜在入力*ξ*のランダムな非線形写像を評価し，写像によってパラメータ化された平均場標本を描画することによって，潜在変数**z**の標本を生成する．これらの潜在変数は，生成モデル**(b)の**事後分布に従うことを目的とする． | 37 |
| データ **x を**条件としています。 .  4.2 推論中の領域マッピングのシーケンス、変分潜在変数空間Rから事後潜在変数空間Q、データ空間Pへの変分潜在変数空間Rから事後潜在変数空間Q、データ空間Pへの変分潜在変数空間のマッピングを行う。 | 44 |
| 後空間での推論と変分空間での補助推論. ..  4.3 vgpを用いたDeep Recurrent Attentive Writer(draw)からの生成画像(上)と、元の変分自動エンコーダーを用いたdraw(下)。vgpはテクスチャを学習します。 | 46 |
| とシャープネス、より複雑な形状をスケッチすることができます。  5.1 （**左**）局所変数 **z** と大域変数 *β の*階層モデル（**右**）．  階層**的暗黙モデル**。これは階層モデルで、**xが**決定論的な | 52 |
| (四角で示される)ノイズ(三角形で示される)の関数.. . . . . . .  5.2 **(上)** 最初の2つのパラメータの限界事後処理。(bot. **左)** 許容誤差を超えるABC法.(bot. **右)** 最初のパラメータの限界事後処理を大規模な | 55 |
| データセット。私たちの推論は、より正確な結果を実現し、大規模データへのスケールアップを実現します。 . | 62 |
| A.1 分類のためのベイジアンニューラルネットワーク. | 83 |
| A.2 Latent Dirichlet allocation (Blei et al., 2003)。 . | 83 |
| A. 3Gaussian行列因数分解. . | 83 |
| A.4 Dirichletプロセス混合モデル。 | 84 |

|  |  |
| --- | --- |
| A.5 バッチ訓練を用いた vae (Kingma and Welling, 2014b) のための完全なスクリプト。それは |  |
| 1000回の更新ごとにMNISTの桁数を生成する．  A.6 バイナリデータのための指数族埋め込み(Rudolph et al., 2016)。ここでは、条件付き対数尤度の総和を最大化するために、最大事後処理(マップ)が使用される | 87 |
| と対数優先である。  D.1 データ点*xn* ∈R10のための2層の深層暗黙モデル。このアーキテクチャでは、確率的層と決定論的層が交互に存在する。確率層を定義するには、単に | 88 |
| ノイズをニューラルネット層の入力に連結することで、ノイズを除去することができます。  D.2 分類のためのベイズガンで、**X** ∈RN×500を入力として取り、ラベルのベクトル **y** ∈{-1*,*1}*Nを*生成する。ではなく、ニューラルネットワークが直接生成する。 | 102 |
| 確率分布のパラメータ化. .  D.3 **(左)** *q* のステップごとの比の差。*y* 軸の分散が小さいほど安定している。興味深いことに、*q が*事後に収束するほど、比率推定器はより正確で安定している。**(中)** *q が*ランダムな初期化時に固定されている。多くのステップを踏んでも比推定量は改善されない。(**右)** *r のステップ数に対する比*の差、*q は事後的に*固定されている。比推定器 | 103 |
| はランダムな初期化から数ステップだけで高精度になります。 | 106 |

# 謝辞

この作品を支えてくださった多くの方々に感謝したいと思います。

まず、私の指導教員であるDavid Blei教授に感謝したいと思います。これほど研究のビジョンが重なり、考え方が明確な指導教官はいなかったでしょう。デイブは、私たちが一緒に行った多くの仕事の中で、大学の学部や学外、年功序列、異なるが似たような研究関心を持つ何十人もの共同研究者と一緒に仕事をする機会を与えてくれました。また、デイブが尊敬の念を持ち、思いやりがあり、人を惹きつけてやまないことにも感心しています。

私が博士課程に移行するにあたり、私が形成された数年間にお世話になったEdo Airoldi教授とFinale Doshi-Velez教授に心から感謝します。特に、私の最初の研究発表の場で、快く経験を積ませてくれた研究室の仲間であるパノス・トゥーリスに感謝します。私の初期の研究プロジェクトは、彼らのサポートなしでは成功することはできませんでした。

コロンビアでは、研究室の仲間に感謝しています。Rajesh Ranganath, Alp Kucukelbir, Adji Dieng.

Maja Rudolph、Jaan Altosaar、Dawen Liang、Stephan Mandt、James McInerney、Fran Ruiz、Christian Naesseth。特にラジェシュには、私のアイデアの提案、頻繁な深夜の研究議論、締め切りに向けた必死のラッシュを乗り切ってくれたことに感謝しています。また、より大きなプロジェクトに取り組み、リサーチのアイデアを反復し、より優れたライターとコーダーになるための手助けをしてくれた Alp に感謝したいと思います。哲学ホールでの頻繁な昼食を楽しみました。ラジェッシュやアルプとは何度も旅を共にしましたが、彼らとアイデアを議論するのはいつも楽しいものでした。

また、統計的応用について私を指導してくれたAndrew Gelman氏、スタンチームに私を招待してくれたAndrew Gelman氏、そして多くの会話をしてくれたAndrew Gelman氏に感謝したいと思います。Aki Vehtari、Bob Carpenter、Daniel Lee は、私がどのようにアプリケーションやソフトウェアにアプローチするかに影響を与えてくれました。特に Alp には、Stan の変分推論で一緒に仕事をしたときに招待してくれたことに感謝しています。また、Adobe Research（後に Google）で Edward の研究をするために私を受け入れてくれた Matt Hoffman にも感謝したいと思います。

最後になりましたが、私の両親であるヴァンとマイ、そして二人の兄弟であるダニエルとダルトンに感謝し、この作品を捧げたいと思います。私の両親は、アメリカで難民として大きな困難を乗り越えてきましたが、子供たちに教育を受けさせ、成功させるために献身的に努力してくれた両親を尊敬しています。

# 第1章：序章、背景、歴史

確率論的モデル化は、確率論の基礎を用いて経験的情報を分析するための強力なアプローチである（Tukey, 1962; Newell and Simon, 1976; Box, 1976）。確率論的モデルは、機械学習（Murphy, 2012; Goodfellow et al. , 2016）や統計学（Friedman et al. , 2001; Gelman et al. , 2013）の重要な要素であり、計算生物学（Friedman et al. , 2000）、計算神経科学（Dayan and Abbott, 2001）、認知科学（Tenenenbaum et al. , 2011）、情報理論（MacKay, 2003）、自然言語処理（Manning and Schütze, 1999）などの分野での応用を特徴としている。

本論文では、確率的モデリングと深層学習の交差点にあるシステムの設計と実装のための進歩を統合した「*深層確率的プログラミング」という概念を*提案する。このようなシステムは、そうでなければ不可能であった新しい確率的モデルや推論アルゴリズムの開発を可能にします：数十億のパラメータへの前例のないスケール、分散・混合精度環境、AIアクセラレータの実現、大規模かつ高次元のデータセットをモデリングするためのニューラルアーキテクチャとの統合と柔軟性、柔軟な推論アルゴリズムやモデル批判戦略のための確率的プログラムの自動微分と任意の操作のための計算グラフの使用などが挙げられます。

以下では、機械学習に対する確率的アプローチの背景と、その研究と最終的な展開のための確率的システムの背景にある歴史について説明する。第2章と第3章では、深層確率的プログラミングシステムの設計について述べる。第4章と第5章では、新しいモデルクラスと同様に、新しい確率的推論戦略への応用について述べる。

## 1.1確率 的機械学習

機械学習や統計学におけるデータ分析のプロセスは、科学的手法のそれを反映しています。

すなわち、迅速な反復を可能にする交換可能なコンポーネントであるコア・ビルディング・ブロックがあります。

モデル

推論

データ

批判する

**図1.1.**ボックスのループ。

互いに積み重なっていきます。これらの構成要素は、サイクルが時間の経過とともに展開されることで、個々の改善につながるサイクルを形成する。これを形式化するために、我々はBoxのループ（Box, 1976; Blei, 2014）として知られている統計学と機械学習の哲学に従う。興味のある現象があるとする。

1. 確率モデルを構築します。モデルは、確率の言語を使用して、現象に関する仮説を形式化します。
2. 与えられたモデルと収集したデータをもとに、現象についての推論を行う。このデータは、実験を設計して実行することで得られる場合もあれば、既に存在するデータ（例えば、過去の実験や、インターネット上のテキストや画像などのコンテンツ）を収集することで得られる場合もあります。
3. モデルのデータへの適合性、つまり仮説がどれだけ実証的に現象を反映しているかを批判してください。修正と繰り返し。

例として、子供がコインを10回ひっくり返したとします。

[0, 1, 0, 0, 0, 0, 0, 0, 0, 1]*,*

ここで 0 は尾、1 は頭を表します。彼女は，コインが頭に着地する確率に興味を持っています．これを分析するために，彼女はまず「モデル」を構築します：コインの裏返しが独立していて，同じ確率で頭に着地すると仮定します．第二に，彼女はその現象について理由をつけます：彼女はデータを与えられたモデルの隠れた構造（未知の確率値）を推測します．最後に、モデルを批判します：彼女は自分のモデルがコインフリップの実世界の現象を捉えているかどうかを分析します。もしそうでなければ，モデルを修正して繰り返します．

次に、確率的モデル、推論、批判の3つの構成要素について、より具体的に説明する。

### 1.1.1 確率モデル

確率モデルは，自然現象からのオブザベーションがどのようにして生じるかを主張する．モデルは，データに対応する観測変数**xと，x**から生成する隠れ構造を提供する潜在変数**zの**合同*分布p*(**x***,***z**)である．

尤*度 p*(**x** | **z**) は、任意のデータ **x が**どのように潜在変数 **z** に依存するかを記述する確率分布である。*尤*度は、データ生成プロセスを仮定しており、データ **x は z** で記述された特定の隠れたパターンを条件に尤度から引き出されると仮定される。これは，隠れ構造の生成過程を仮定している．

ニューラルネットワークは、高次元分布の尤度をパラメータ化することを可能にする関数の一般的なクラスであり、画像分類などの知覚タスクでうまく機能することが経験的に証明されている(Krishevsky et al. , 2012)。本論文の目的のために、我々はニューラルネットワークの背景を提供しない。その特定のアーキテクチャは本論文の中心ではないので、Goodfellow et al.

2016年）を調査対象とした。

### 1.1.2 確率モデルの推論

収集したデータ**xを**分析するためにモデル*p*(**x***,***z**)をどのように使えばいいのか？言い換えれば、どのような隠れた構造**zが**データを説明しているのか？我々はモデルを使ってこの隠れた構造を推論しようとする。推論の1つの方法は、ベイズの法則を利用して、*事後の*

*p*(**x***,***z**) *p*(**z** | **x**) = R *p*(**x***,***z**)**dz**

事後処理は、観測されたデータ**x**に条件付けされた潜在変数**zの**分布で、データの隠れた表現の確率的記述である。

多くのベイズ主義者が実践している帰納主義の観点からは，事後処理は，潜在変数についての我々の更新された仮説であり，現象についての我々の新しい主観的な信念を表す．Box, Rubin, Gelmanなどの統計学者が実践しているように、仮説的帰納主義の観点からは、事後処理は、単にデータに適合したモデルであり、したがって、批判され、最終的に修正されるべき、偽造可能な仮説である(Box, 1982; Gelman and Shalizi, 2013)。

後処理を*推論する*これで事後処理が何を表しているかわかりましたどうやって計算するのでしょうか？これは確率論的推論における中心的な計算課題です。事後処理は、分母の積分である正規化定数のために計算が困難です。これはしばしば高次元の積分であり、解析的な（閉形の）解を欠いています。したがって，事後処理を計算するということは，事後処理を近似*することを*意味する．

### 1.1.3. 3 変量推論

変分推論とは、事後推論を最適化に見立てたアルゴリズムの包括的な用語です。

Hinton and van Camp, 1993; Waterhouseら, 1996; Jordanら, 1999a）。中心となる考え方は、2つのステップを含む。

1. 潜在変数上の分布*q*(**z** ; *λ*)の系列を仮定する．
2. *q*(**z** ; *λ*)をそのパラメータ*λ*にわたって最適化することで事後処理に一致させます。

この戦略は、事後*p*(**z** | **x**)を計算する問題を最適化問題に変換します。

損失関数を最小にする

*λ*∗ = argminloss(*p*(**z** | **x***),q*(**z** ; *λ))。*

*λ*

最適化された分布 *q*(**z** ; *λ*∗) を事後分布 *p*(**z** | **x**) の代理として使用します。

### 1.1.4 最大事後推定値

変分推論の1つの形式は、最大事後推定（MAP）として知られています。これは，モードを事後分布の点推定として使用する．

zMAP = *argmaxp*(**z** | **x**) = *argmaxlogp*(**z** | **x***).*

すなわち，仮定された変分布の系列は，単に確率を持つデルタ分布

1を点で指定します。実際には、数値的なアンダーフローの問題を避けるために、密度の対数を用いて作業を行います (Murphy, 2012)。

MAP推定値は、モデルの下での隠れパターン**zの**最も可能性の高い構成である。しかし、この最適化問題を直接解くことはできません。これを回避するために、我々はベイズの法則を使用して接合密度を最適化します。

**zMAP** = *argmaxlogp*(**z** | **x**) = *argmaxlogp*(**x***,***z***).*

これは、次の理由で有効です。

*logp*(**z** | **x**) = *logp*(**x***,***z**) - *logp*(**x**) = *logp*(**x***,***z**) - **zの**項で定数*。*

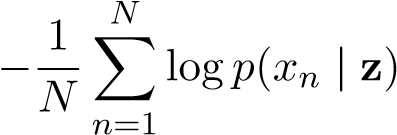
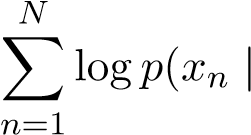
MAP推定には、最尤推定の一般的なシナリオが含まれています。

の場合。

zMAP = *argmaxp*(**x***,***z**) = *argmaxp*(**x** | **z***),*  **zz**

ここでは、事前の *p*(**z**) が平坦で、**z が**サポートするすべての値に一様な確率を置く。不均一な優先順位を置くことは、推定を正規化し、尤度の最大化から離れた値にペナルティを与えると考えることができ、オーバーフィットを引き起こす可能性があります。例えば、**z**での正規先行またはラプラス先行は、リッジ回帰としても知られる*`*2 ペナルティに対応し、LASSOとしても知られる*`*1 ペナルティに対応します。

最尤度はクロス・エントロピー最小化としても知られています。データ集合 **x** = {*xn*} の場合。

**zMAP** = *argmaxlogp*(**x** | **z**) = **argmaxz**) = argmin*.*

最後の式は、*N個の*データ点の集合を用いて、真のデータ分布と*p*(**x** | **z**)の間のクロスエントロピーを近似したものと考えることができます。

*勾配降下*。潜在変数**z**のMAP推定値を見つけるために、対数結合密度∇**z** *logp*(**x***,***z**)の勾配を使用し、(局所的な)オプティマまでそれをたどる。

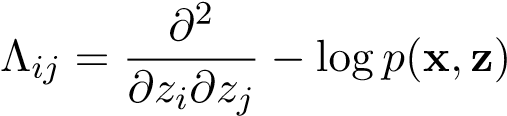
### 1.1.5 ラプラス近似

最大事後推定(MAP)は、そのモードを単純に捉えることで、点質量(デルタ関数)を持つ事後の*p*(**z** | **x**)を近似します。MAPは高速で効率的なので魅力的です。どのようにMAPを使って、より良い事後処理近似を構築することができるでしょうか？

ラプラス近似 (Laplace, 1986) は，MAP推定値を改善する1つの方法である．アイデアは、MAP推定値を中心とした正規分布で事後処理を近似することです。

*p*(**z** | **x**)≒Normal(**z** ; zMAP*,*Λ-1*).*

これには精度行列Λの計算が必要です。テイラー展開から派生したラプラス近似は、MAP推定時の負の対数結合密度のヘシアンを使用します。これは成分ごとに次のように定義されます

*.*

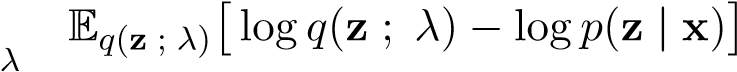
平坦なプリオール（MAPを最尤に下げる）の場合、精度行列は観測されたフィッシャー情報（Fisher, 1925）として知られています。エドワードはTensorFlowの自動微分を利用して、これを分散させています。

### 1.1.6 KL(*qkp*)最小化

MAP推定とラプラス近似は単純ですが、真の事後分布を近似する際に局所的でガウス的な仮定をします。変分推論のもう一つの一般的な形式は、*q*(**z** ; *λ*)から*p*(**z** | **x**)へのKullback-Leibler発散を最小化するものです。

*λ*∗ = argminKL(*q*(**z** ; *λ*) k *p*(**z** | **x**))

*λ*

= argmin *.*

KL ダイバージェンスは、2つの確率分布間の類似度の非対称的な情報理論的尺度である (Hinton and van Camp, 1993; Waterhouse et al , 1996; Jordan et al , 1999a)。

*エビデンス下限*。上記の最適化問題は、直接事後の *p*(**z** | **x**) に依存するため、難解です。これに対処するために、次のような特性を考えてみましょう。



ここで左辺は限界尤度*p*(**x**)=R *p*(**x***,***z**)**dzの**対数であり，モデルエビデンスとしても知られている。

エビデンスは変分パラメータ*λ*に対して一定なので、代わりに*エビデンス下限*(elbo)を最大化することでKL(*qkp*)を最小化することができます。

ELBO*.*

エルボでは、*p*(**x***,***z**)と*q*(**z** ; *λ*)の両方が処理可能である。最適化問題は

*λ*∗ = argmaxELBO(*λ)。*

*λ*

その名の通り、エルボはエビデンスの下限であり、最適化することでデータを観測する確率を最大化しようとします。エルボを最大化すると何ができるのか？エルボを分割すると

下取り

ELBO(*λ*) = *Eq*(**z** ; *λ*)[*logp*(**x***,***z**)] - *Eq*(**z** ; *λ*)[*logq*(**z** ; *λ)]。*

ここで、第1項はエネルギーを表し、第2項（マイナス符号を含む）は*q*のエントロピーを表します。エネルギーは*qが*モデルが高い確率を置く場所に確率の塊を集中させるように促します。エントロピーは、q*が確率の塊*を拡散させて、1つの場所に集中しないようにすることを促します。

勾配に基づく最適化のための勾配を得るためには、スコア関数勾配と再パラメータ化勾配の2つの一般的な戦略があります。

*スコア関数の勾配*。勾配降下は、エルボのような複雑な目的物を最適化するための標準的なアプローチです。考え方は、その勾配を計算することです。

∇*λ* ELBO *.*

を呼び出して、勾配に比例したパラメータの現在のセットを更新します。スコア関数勾配推定器は対数の特性を利用して勾配を

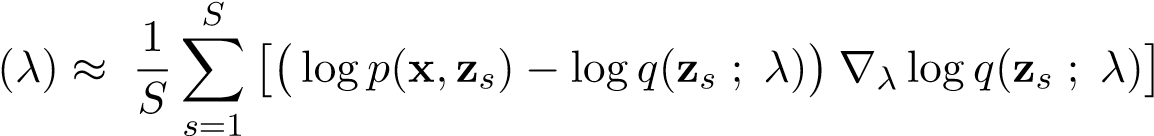
∇*λ* ELBO *.*

エルボの勾配は、変分モデル *q*(**z** ; *λ*) 上の期待値である; それが必要とする唯一の新しい成分は、*スコア関数* ∇*λ logq*(**z** ; *λ*)である (Paisley et al. , 2012b; Ranganath et al. , 2014) (Paisley et al. , 2012b; Ranganath et al. , 2014)。

モンテカルロ積分を用いて、エルボとその勾配の両方のノイズのない推定値を得ることができます。基本的な手順は以下の通りです。

1. *S の*サンプルを描画します。
2. を用いて期待値の引数を評価し
3. 評価された量の経験的平均を計算する．

勾配のモンテカルロ推定値は次のようになります。

∇*λ* ELBO *.*

これは、エルボの実際のグラデーションを偏りなく推定したものです。

*再パラメータ化勾配*。モデルが微分可能な潜在変数を持つ場合，最適化空間をよりよく横断するために，モデルからの勾配情報を活用することが一般的に有利である．これを行うためのアプローチの1つが、リパラメタライゼーション・グラジエントである(Kingma and Welling, 2014a; Rezende et al. , 2014)。

いくつかの変分布 *q*(**z** ; *λ*) は，有用な再パラメータ化を認めるものがあります．例えば，正規分布 **z** ∼ Normal(*μ,*Σ) を **z**として再パラメータ化することができます．一般的には，次のように書きます

*,* **z**

決定論的関数 **z**(-; *λ*) は、代わりに変分パラメータをカプセル化しており、このプロセスに従うことは、元の分布から直接 **z を**引くことと等価です。

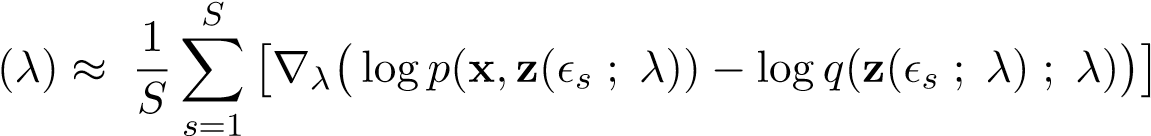
再パラメータ化勾配は、このプロパティを利用して勾配を

∇*λ* ELBO *.*

エルボの勾配は基底分布に対する期待値であり、勾配は内部式に直接適用することができます。モンテカルロ積分を用いて、エルボとその勾配の両方のノイズのない推定値を得ることができます。基本的な手順は以下の通りです。

1. *S の*サンプルを描画します。
2. を用いて期待値の引数を評価し
3. 評価された量の経験的平均を計算する．

勾配のモンテカルロ推定値は次のようになります。

∇*λ* ELBO *.*

これは，エルボの実際の勾配の偏りのない推定値である．経験的に, スコア関数の勾配よりも低い分散を示し, 大規模な問題集合においてより速い収束をもたらします (Tran et al. , 2016b).

### 1.1.7 モデル批判

私たちは、モデルが真かどうかを検証することはできない。実際には、「すべてのモデルは間違っている」(Box, 1976)。しかし，モデルがどこで間違っているかを明らかにしようとすることはできる．モデル批判は、モデルを近似値として正当化したり、モデルを修正するための良い方向性を示したりするのに役立ちます。

モデル批判は典型的には事後予測分布を分析する。

Z *p*(xnew | **x**) = *p*(xnew | **z**)*p*(**z** | **x**)**dz***.*

モデルの事後予測は，過去のオブザベーションを与えられた新しいデータを生成するために使用することができ，また，過去のオブザベーションを与えられた新しいデータで予測を行うことができる．それは，事後分布に従って，潜在変数のすべての集合にわたって平均化された新しいデータの尤度を計算することによって形成される．

スコア*リングルール*。スコアリング・ルールは、訓練されたモデルを評価するためのスカラー値のメトリックである(Winkler, 1994; Gneiting and Raftery, 2007)。例えば、データ中の各オブザベーションのラベルを予測し、それらの真のラベルと比較することで、分類のためのモデルを評価することができます。形式的には、イベント*x*の空間上の2つの分布*p*と*qが*与えられる。

Z

*S*(*p,q*) = *q*(*x*)*S*(*p,x*)*dx.*

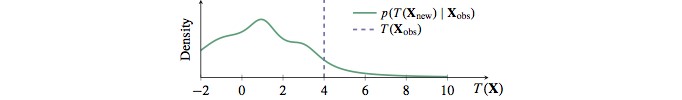
ここで、*S*(*p,x*)は、対数スコアリングルール(*logp*(*x*))のような*x*上の密度*pの*実数関数である。訓練で得られたデータを用いてモデルを批判することはよくあることです。機械学習では、ベンチマークデータセットは訓練とテストの分割を含む。

事後*予測検査*。事後予測検査（PPC）は，モデルから生成されたデータが，真の分布から生成されたデータからどの程度乖離しているかを分析する．この度合を定量化するために数値的に使用することも、この度合を可視化するためにグラフィカルに使用することもできる。PPC は，単一の値ではなく分布を提供するスコアリング規則の確率的な一般化と考えることができる(Box, 1980; Rubin, 1984; Meng, 1994; Gelman et al. , 1996)．

最も単純なPPCは，*T*(xnew) = max(xnew)のように，事後予測から生成された複製されたデータセットに検定統計量を適用することで動作する．*T*(xnew)を多くのデータ複製上の新しいデータセットに適用すると、分布が生成されます。この分布を実データの検定統計量と比較します。

*T*(**x**)。

図1.2では、*T*(**x**) はこの参照分布の低確率領域にあります：モデルが真であれば、検定統計量を観測する確率は非常に低いです。これは、モデルがこのチェックに従ってデータにうまくフィットしていないことを示しています。



**図1.2:** 生成されたデータセットに複製された検定統計量の分布と、観測されたデータセットに適用された検定統計量。

より一般的には，検定統計量は，不一致関数として知られるモデルの潜在変数*T*(**x***,***z**)の関数である．不一致関数の例は，スコアリング・ルールに使用されるメトリクスである．スコアリング・ルールをPPCの特殊なケースとして解釈することができます：それは，参照分布を考慮せずに，単に実データ上で*T*(**x***,***z**)を計算しているだけです．参照分布は、全体の分布を参照して、点についての確率的な記述を行うことを可能にします。

PPCは、モデルを修正するための優れたツールであり、データへの適合性を検討する際に、現在のモデルを単純化したり、拡張したりします。これらの手法は、適合度検定などの古典的な仮説検定に触発されています；これらの手法は、大規模標本評価のフリークエント主義の観点からモデルを批判しています。

PPCは、仮説検定、モデル比較、モデル選択、モデル平均化などのタスクにも適用できます。PPCはベイズ仮説検定の一形態として適用することができますが、仮説検定は一般的に推奨されないことに注意してください。

## 1.2 確率論的システムの歴史

確率論的ソフトウェアシステムの例として、私たちは2つの初期のスレッドを指しています。1つ目は人工知能におけるものである。エキスパートシステムは人間の専門知識から設計されたものであり、それにより、既存の知識に応じたより大きな推論ステップを可能にした(Buchanan et al. , 1969; Minsky, 1975)。コネクショニストモデルでは、設計者は経験から学ぶニューロンのような処理ユニットに焦点を当てた。

第二のスレッドとして、統計解析の問題に対する効率的な計算に広く関心が集まっていた初期の統計計算の研究を紹介します。S言語は、ベル研究所のJohn Chambersとその同僚によって開発されました (Becker and Chambers, 1984; Chambers and Hastie, 1992)。これはR言語の前身である(Ihaka and Gentleman, 1996)。統計モデルのベイズ分析に焦点を当てたBUGS (Spiegelhalter et al. , 1995)のようなよりターゲットを絞った環境は、 確率的プログラミングという新たな分野を立ち上げるのに役立ちました。

我々は、これらの初期の研究を確率的システムの中で発展させたいと考えています。現代のアプリケーションでは、その設計と実装において新たな課題が生じます。我々は、2つの課題に焦点を当てています。第一に、統計学と機械学習は、確率的モデルとその推論の方法論において大きな進歩を遂げてきた（例：Hoffmanら（2013）、Ranganathら（2014）、Rezendeら（2014））。ソフトウェアシステムが高速な実験を可能にするためには、これらの進歩を捉えることができる豊富な抽象化が必要である：それは、確率的モデルの幅広いクラスと、それらの効率的な推論のための幅広いクラスのアルゴリズムの両方を包含しなければならない。第二に、研究者は複雑な確率モデルを、前例のない規模の大規模データで使用することにますます意欲的になっています(Bengio et al. , 2013; Ghahramani, 2015; Lake et al. , 2016)。したがって、我々は、分散訓練をサポートし、（複数の）GPUのようなハードウェアの統合をサポートする効率的なコンピューティング環境を必要としています。

確率的プログラミングのコアテーマは自動推論であり、BUGSなどのシステムで使用されています。

(Spiegelhalter et al. , 1995)のような最近の研究から、Stan (Carpenter et al. , 1995)のようなより最近の研究まで、ギブスサンプラーを用いたものがある。

2015）にNo-U-Turn Sampler（Hoffman and Gelman, 2014）、自動微分変分推論（Kucukelbir et al. , 2017）を加えたものである。本論文では、アルゴリズム自体の研究のための柔軟で構成可能な推論という対極のテーマを効果的に検討する。このユースケースは、様々な領域で機能し、最終的にはよりインテリジェントなシステムを目指して、より優れたモデルやアルゴリズムを考案することが課題となっている機械学習研究とよく合致している。自動推論というテーマは、応用科学者や実務家という別の対象者にも有用です。これらの目的は、より構成可能で柔軟性のあるシステムを開発する上で、ここで述べたことの上に構築されたより高レベルの抽象化として十分に適合するでしょう。

# 第2章：深層確率計画法

## 2.1 イントロダクション

ディープニューラルネットワークの性質は構成的です。ユーザーは、テスト（前方伝播）や推論（勾配ベースの最適化、後方伝播と自動微分）をどのように行うかを気にすることなく、創造的な方法でレイヤーを接続することができます。

この章では、確率的プログラミングのための構成表現を設計する。確率的プログラミングでは、ユーザは生成的確率モデルをプログラムとして指定し、それらのモデルを推論手順に「コンパイル」することができる。確率的モデルもまた、本質的には構成的であり、多くの研究により、ランダム変数の構成を介したリッチな確率的プログラムが可能になっている(Goodman et al. , 2012; Ghahramani, 2015; Lake et al. , 2016)。

しかし、推論に類似した構成性を考慮した研究は少ない。むしろ、多くの既存の確率的プログラミング言語は、推論エンジンをモデルから抽象化されたブラックボックスとして扱っている。これらの言語では、モデルの表現を再利用する確率的推論を捉えることができない。これは、最近の変分推論（Kingma and Welling, 2014b; Rezende and Mohamed, 2015; Tran et al. , 2016b）、生成的敵対的ネットワーク（Goodfellow et al. , 2014）、さらにはより古典的な推論（Dayan et al. , 1995; Gutmann and Hyvärinen, 2010）における重要なアイデアである。

我々は、ランダム変数と推論のための2つの構成表現に基づいて構築されたチューリング完全確率プログラミング言語であるEdward1を提案する。推論をモデリングと同等の一級市民として扱うことで、確率的プログラミングが従来のディープラーニングと同等の柔軟性と計算効率を持つことを示す。柔軟性については、点推定から変分推論、MCMCに至るまで、様々な構成可能な推論手法を用いて、同じモデルを簡単に適合させる方法を示している。効率性については、以下のような手法を統合する方法を示す。

[org/iclr](http://edwardlib.org/iclr2017)2011Tran et al.[7.](http://edwardlib.org/iclr2017)を参照してください。APIの詳細については、Tran et al.この論文のコンパニオンウェブページは [http://edwardlib](http://edwardlib.org/iclr2017) にあります。

Edward を TensorFlow のような既存の計算グラフフレームワークに組み込んだ (Abadi et al. , 2016)．TensorFlowのようなフレームワークは、分散訓練、並列化、ベクトル化、GPUサポートなどの計算上の利点を"無料で"提供する。例えば、我々はベンチマークタスクで、Edwardのハミルトニアンモンテカルロ計算が既存のソフトウェアよりも何倍も高速であることを示している。さらに、Edwardは実行時のオーバーヘッドを発生させず、手書きのTensorFlowと同等の速度を実現しています。

## 2.2 確率モデルのための構成表現

我々はまず、確率的モデルのための構成表現を開発する。我々は2つの基準を求めている。(a)ノードがデータ上の操作を表し、エッジがそれらの間で通信されるデータを表す効率的なフレームワークである計算グラフ（Culler, 1986）との統合、および(b)グラフの下での表現の不変性、つまり推論中に表現を再利用できることである。

エドワードは，ランダム変数を重要な構成表現として定義しています．これはクラスオブジェクトであり、例えば対数密度を計算したり、標本化したりするメソッドを持っています。さらに、各ランダム変数 **x は**テンソル（多次元配列） **x**∗ に関連付けられており、1 つのサンプル **x**∗ ∼ *p*(**x**) を表します。

この連想は、ランダム変数をテンソル上の計算グラフに埋め込みます。

デザインがシンプルなので、計算グラフのフレームワークで確率的プログラムを開発するのが簡単になります。重要なことは、すべての計算がグラフ上で表現されていることです。これにより、ディープ・ニューラル・ネットワーク、多様な数学演算のセット、同じフレームワーク上に構築されたサードパーティ・ライブラリのような複雑な決定論的構造を持つランダム変数を構成することが可能になります。この設計はまた、複雑な確率的構造を捉えるためのランダム変数の構成を可能にします。

例として、ベータ・ベルヌーイモデル、*p*(**x***,θ*) = Beta( Bernoulli(*xn* |*θ*), ここで、*θは*50個のデータ点**x** ∈ {0*,*1}50で共有される潜在確率です。ランダム変数 x は 50 次元であり、ランダムテンソル *θ*∗ でパラメータ化されています。オブジェクト x を取得すると，グラフが実行されます：生成過程をシミュレートし，50 個の要素からなる 2 進ベクトルを出力します．

シータ = ベータ(a=1.0, b=1.0)

*θ*

*θ*

∗

**x**

**x**

x = ベルヌーイ(p=tf. ones(50) \* シータ)∗)

**図2.1.**ベータベルヌーイプログラム**（左）と**その計算グラフ**（右）**。グラフから **x を**取得すると 50 個の要素からなる 2 進ベクトルが生成されます。

*# 確率論的モデル*

z = 正規(mu=tf. zeros([N, d]), sigma=tf. ones([N, d]) h = 密(256, activation='relu')(z)

**z**

*n*

**x**

*n*

*θ*

*φ*

*N*

x = ベルヌーイ(logits=Dense(28 \* 28, activation=None)(h))

*# ♪ 変分モデル*

qx = tf.プレースホルダ(tf. float32, [N, 28 \* 28]) qh = Dense(256, activation='relu')(qx) qz = Normal(mu=Dense(d, activation=None)(qh), sigma=Dense(d, activation='softplus')(qh))

**図2.2.**28 × 28 ピクセル画像のデータセットに対する変分自動エンコーダー。(**左)**グラフィカルモデル、推論モデルは点線、**(右)**2層ニューラルネットワークを用いた確率的プログラム。

すべての計算はランダム変数上に記号的に登録され，その実行にわたって登録されるのではない．記号的表現は，大規模なモデルでは不合理なメモリ消費につながるモデル全体の再定義を必要としない (Tristan et al. , 2014)．さらに、それは、任意のコードを実行する前に、グラフ内の決定論的操作と確率論的操作の両方を単純化することを可能にする(S Ścibior et al. , 2015; Zinkov and Shan, 2016)。

計算グラフでは、確率的プログラム内にミュー タブルな状態を構築することも当然のことです。計算グラフの典型的な使用法として、そのような状態はモデルパラメータを定義することができます；TensorFlowでは、これはtf.Variable.Variableによって与えられます。もう一つの使用例は、識別モデル *p*(**y**|**x**) を構築するためのもので、ここで **x は**訓練データまたはテストデータとして入力される特徴量です。このプログラムはデータから独立して書くことができ、グラフ中の**xの**ための変異可能な状態(tf.placeholder)を使用します。学習とテストの間、プレースホルダに適切な値を与えます。

付録A.1では，分類のためのベイジアンニューラルネットワーク(A.2)，潜在ディリクレ配分(A.3)，ガウス行列因数分解(A.4)の例を示す．以下に他の例を示す．

### 2.2.1 例。可変オートエンコーダー

図3.4は，エドワードに vae (Kingma and Welling, 2014b; Rezende et al. , 2014) を実装したものである．これは，データを対象とした確率モデルと，前者の事後処理を近似するように設計された変分モデルから構成されている．ここでは、確率モデルと変分モデルの両方を構築するためにランダム変数を使用します。

n*個の*データ点*xn*∈{0*,*1}28-28があり、それぞれ*d個の*潜在変数*zn*∈*Rdを*持つ。このプログラムでは

Keras（Chollet, 2015）を用いてニューラルネットワークを定義する。確率モデルは、256個の隠れユニット（およびReLU活性化）を持つ2層のニューラルネットワークによってパラメータ化され、28×28ピクセルの画像を生成する。変分モデルは、256個の隠れユニットを持つ2層の推論ネットワークによってパラメータ化され、正規事後近似のパラメータを出力する。

確率的プログラムは簡潔です。分布の仮定やニューラルネットのアーキテクチャなど、vaのコア要素はすべて拡張可能である。モデルの構成性を利用して、より複雑なモデルに組み込むことができる(Gregor et al. , 2015; Rezende et al. , 2016)し、他の学習タスクにも適用できる(Kingma et al. , 2014)。推論の構成性(セクション5.3で議論する)では、表現的な変分近似(Rezende and Mohamed, 2015; Tran et al. , 2016b; Kingma et al. , 2016)や代替目的(Ranganath et al. , 2016a; Li and Turner, 2016; Dieng et al. , 2016)のような、より複雑なアルゴリズムに埋め込むことができる。

### 2.2.2 例.可変長のベイジアン・リカレント・ニューラル・ネットワーク

また、制御フロー演算でランダム変数を構成することも可能です。例として、図 2.3 に可変長のベイジアンリカレントニューラルネットワーク（rnn）を実装します。データは入力｛x1*,...,xT*｝と出力｛*y*1*,...,yT*｝のシーケンスで、時間ステップごとに*xt*∈*RD*,*yt*∈Rの長さ*T*とします。*ｔ*＝１*,....,Ｔ*に対して、ｒｎｎは、更新を適用する。

*ht* = tanh(*Whht*-1 + *Wxxt* + *bh)。*

ここで、前の隠れた状態は*ht*-1∈*RH*である。各隠れた状態を出力の尤度 *yt* ∼ Normal(*Wyht* + *by,*1) に与え、全てのパラメータ{*Wh* ∈RH}に対して標準的な法線優先順位を置く。

.私たちの実装は動的です。

を固定長の rnn から取得し、計算をパッドしたり展開したりします。

### 2.2.3確率 的な制御フローとモデルの並列化

また、制御フロー自体にランダム変数を配置することも可能であり、確率論的なプログラムを確率制御フローで実現することができる。確率制御フローは、動的な条件依存関係を定義し、文献では偶発依存関係または実存依存関係として知られています(Mansinghka et al. , 2014; Wu et al. , 2016)。図2.4を参照してください。ここで，**xは**与えられた実行のために**a**に依存しているかもしれないし，していないかもしれない．付録A.5では，ディリクレ過程混合モデルを実装するために確率的制御フローを用いる．確率的なテンソル

**def** rnn\_cell(hprev, xt): **return** tf. tanh(tf. dot(hprev, Wh) + tf. dot(xt, Wx) + bh)

Wh = 法線(mu=tf. ゼロ([H, H]), sigma=tf. 物([H, H])

Wx = 法線(mu=tf. ゼロ([D, H]), シグマ=tf. 物([D, H])Wy = 正規(mu=tf. ゼロ([H, 1]), sigma=tf. ones([H, 1]) bh = 正規(mu=tf. ゼロ(H), sigma=tf. ones(H))

------······ by = 正規分布(mu=tf. ゼロ(1), シグマ=tf. ゼロ(1))

**x**

*t*

**b**

*h*

**W**

*x*

**W**

*h*

**W**

*y*

**b**

*y*

**h**

*t*

**y**

*t*

x = tf. placeholder(tf. float32, [None, D]) h = tf. scan(rnn\_cell, x, initializer=tf. zeros(H)) y = Normal(mu=tf. matmul(h, Wy) + by, sigma=1.0)

**図2.3:** Bayesian rnn.(**左)**グラフィカルモデル、**(右)**確率的プログラム。このプログラムは不特定多数の時間ステップを持ち、シンボリックなforループ(tf.scan)を使用しています。

∗ tf.while\_loop(...)

∗

**p**

**p**

∗

**a**

**a**

**x**

**x**

**図2**.**4：確率的**な制御フローを持つ確率的プログラムの計算グラフ。

確率的な制御フローでは、グラフ構造を使用するアルゴリズムでは、条件依存性の関係が実行トレースによって変化するため、困難が生じます。しかし、計算グラフは、静的な条件依存構造(**p**)と動的な依存構造(**a**)を区別するためのエレガントな方法を提供します。我々は、GPUとバッチ・トレーニングを用いて、静的構造に対してモデル並列化（モデルの構成要素間での並列計算）を実行することができます。動的構造を扱うために、より汎用的な計算を用いることができる。

## 2.3 推論のための構成表現

我々は、計算グラフ上の豊かな確率的プログラムを構築するための表現として、ランダム変数を説明した。ここでは、推論のための構成表現について述べる。我々は２つの基準を求めている。(a) 多くのクラスの推論をサポートし、推論された事後処理の形式はアルゴリズムに依存する。

我々のアプローチを説明するために、単純な階層モデルを実行例として使用します。図2.5は，データ**x**，局所変数**z**，大域変数*βの*合同分布*p*(**x***,***z***,β*)を示しています．

N = 10000 *# データ点の数* D = 2 *# データ次元*

*β*

**z**

*n*

**x**

*n*

*N*

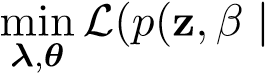
K = 5 *# クラスター数*

beta = 正規(mu=tf. ゼロス([K, D]), sigma=tf. ones([K, D]) z = カテゴリ(logits=tf. ゼロス([N, K]) x = 正規(mu=tf. gather(beta, z), sigma=tf. ones([N, D]) )

**図2.5:** 階層モデル.(**左)**グラフィカルモデル、**(右)確率的**プログラム。*D*次元データ{*xn*}∈*RN×D*上のガウシアンの混合物である。*K個の潜在*クラスター平均*β*∈*RK×Dが*ある。

### 2.3.1 確率的グラフ最適化としての推論

推論の目的は、データ xtrain *が*与えられたときの事後分布 *p*(**z***,β* | xtrain; *θ*) を計算することです。2 これを次の最適化問題として形式化します。

xtrain; *θ)、q*(**z***,β*;*λ))。* (2.1)

ここで、*q*(**z***,β*; *λ*)は事後処理*p*(**z***,β* |xtrain; *θ*)への近似であり、Lは*p*と*qに関する*損失関数である。

近似*q*、損失L、およびパラメータ{*θ,λ*}を更新するルールの選択は、推論アルゴリズムによって指定されます。(注意: *qは*点やサンプルの集合などのノンパラメトリックなものであってもよい)。

エドワードでは、この問題を次のように書きます。

推論 = ed.推論({beta: qbeta, z: qz}, data={x: x\_train})

推論は2つの入力を持つ抽象クラスです。1つ目は潜在確率変数βとzの集合で、それぞれ「事後変数」qbetaとqzに関連付けられています。2つ目は観測されたランダム変数xの集合であり、それらの実現値x\_trainに関連付けられています。

推論は、式2.1の最適化を定義して解くという考え方です。パラメータを調整します。

2例えば，xのシグマ引数をtf.exp(tf.Variable(0.0))\*tf.ones([N, D])に置き換えることができます．これは0で初期化されたモデルパラメータを定義します．

qbeta = 正規( T = 10000 *#サンプル数* mu=tf.変数(tf. zeros([K, D])), qbeta = Empirical(

sigma=tf. exp(tf.Variable(tf. zeros([K, D]))))params=tf.変数(tf. zeros([T, K, D])))

qz = カテゴリー( qz = Empirical(

logits=tf.変数(tf. zeros([N, K])))) params=tf.変数(tf. zeros([T, N])))

推論 = ed.VariationalInference( 推論 = ed.モンテカルロ(

{beta: qbeta, z: qz}, data={x: x\_train }){beta: qbeta, z: qz}, data={x: x\_train})

**図2.6: (左)** 変分推論.**(右)** モンテカルロ法.

qbetaとqz（および任意のモデルパラメータ）の分布が事後分布に近いことを確認するために、クラスメソッドを利用することができます。推論を細かく制御するためにクラスメソッドが用意されています。inference.initialize()を呼び出すと、{*θ,λ*}を更新するための計算グラフが作成されます。inference.update()を呼び出すと、{*θ,λ*}を更新するための計算が一度だけ実行され、収束するまでループでメソッドを呼び出します。重要なことは、エドワードの言語では効率性は失われていないということです：計算グラフは、特定のモデルのために手書きで書かれたのと同じです。これは，実行時間が同じであることを意味します．

エドワードのキーコンセプトは、明確な"モデル"や"推論"のブロックが存在しないということです。モデルは単にランダム変数の集合であり、推論はその集合内のパラメータを別の対象に修正する方法である。この還元主義は、大きな柔軟性を提供します。例えば、モデルの一部のみを推論したり（例えば、レイヤワイズ学習（Hinton et al. , 2006））、複数のモデルで使用される部分を推論したり（例えば、マルチタスク学習）、事後処理を新しいモデルにプラグインしたり（例えば、ベイズ更新）することができる。

### 2.3.2 推論のクラス

推論の設計は非常に一般的です。以下では、多くのアルゴリズムを表現するためのサブクラスを記述する。変分推論、モンテカルロ推論、生成的敵対的ネットワークなどである。

変分推論は，近似分布のファミリーを仮定し，そのファミリーの中で最も事後に近いメンバーを求める (Jordan et al. , 1999a)．エドワードでは，グラフの中に変分系列を構築します；図2.6（左）を参照してください．我々の実行例では，ファミリは，パラメータとして変異可能な変数

*λ* = {*π,μ,σ*}ここで、*q*(*β*;*μ,σ)* = 正規(*β*;*μ,σ*)、*q*(**z**;*π*) = カテゴリ的(**z**;*π*)である。

特定の分散アルゴリズムは、VariationalInferenceクラスを継承しています。それぞれが、損失関数や勾配などの独自のメソッドを定義しています。例えば、我々は、PointMassランダム変数の近似ファミリー（qbetaとqz）を用いてマップ推定を表現しています。

**def** generative\_network(eps): h = Dense(256, activation='relu')(eps) **return** Dense(28 \* 28, activation=None)(h)

**def** discriminative\_network(x).

*n*

*θ*

**x**

*n*

*N*

h = Dense(28 \* 28, activation='relu')(x) **return** Dense(h, activation=None)(1)

*# 確率論的モデル*

eps = Normal(mu=tf. zeros([N, d]), sigma=tf. ones([N, d]) x = generative\_network(eps)

推論 = ed.GANInference(data={x: x\_train}, discriminator=discriminative\_network)

**図2.7:** 生成的敵対者ネットワーク.(**左)**グラフィカル・モデル，(**右)**確率的プログラム．モデル（生成器）は学習のためにパラメータ化された関数（判別器）を使用する。

点に集中した質量。MAPはVariationalInferenceを継承し、損失関数として負の対数結合密度を定義し、TensorFlow内部の既存のオプティマイザを使用する。セクション2.4.1では、ブラックボックス変分推論のための複数の勾配推定器を用いて実験を行う(Ranganath et al. , 2014)。各推定器は同じ損失（発散KL(*q kp*)に比例する目的）と異なる更新規則（確率的勾配）を実装している。

モンテカルロ法は，標本を用いて事後処理を近似する（Robert and Casella, 1999）．モンテカルロ推論は，近似系列が経験的分布である場合の推論であり， .パラメータは .図2.6（右）を参照してください。

モンテカルロ法のアルゴリズムは、経験的近似において、1つのサンプル*β*(*t),***z**(*t*)を一度に更新することによって進行する。特定のmcサンプラーは、ハミルトニアンモンテカルロ(Neal, 2011)のような勾配を利用したり、逐次モンテカルロ(Doucet et al. , 2001)のようなグラフ構造を利用したりすることができる。

また、エドワードは、生成的敵対者ネットワーク（gans）のような非ベイズ的な手法もサポートしている（Goodfellowら , 2014）。図2.7を参照のこと。このモデルでは、*N個の*データ点上のランダムノイズεを仮定しています。このランダムノイズは、実値データxを出力するニューラルネットワークであるgenerative\_network関数に供給されます。我々はGANInferenceを構築し、それを実行することで、2つのニューラルネットワーク関数内のパラメータを最適化する。このアプローチは、GANの多くの進歩（例えば、Dentonら（2015）; Liら（2015））に拡張します。

qbeta = PointMass(params=tf.変数(tf. zeros([K, D]))) qz = カテゴリー(logits=tf.変数(tf. zeros([N, K]))

inference\_e = ed.VariationalInference({z: qz}, data={x: x\_train, beta: qbeta}) inference\_m = ed.MAP({beta: qbeta}, data={x: x\_train, z: qz}) ... **for** \_ **in** range(10000): inference\_e. update() inference\_m. update()

**図2.8:** 変分的EMを実行するための推論アルゴリズムの組み合わせ

最後に、そうでなければ面倒な代数的操作を必要とするアルゴリズムを設計することができます。計算グラフのノード上の記号的代数を用いて、ランダム変数間の共役関係を明らかにすることができます。ユーザーは、変数を統合して、古典的なギブス（Gelfand and Smith, 1990）、平均場更新（Bishop, 2006）、および厳密推論を自動的に導出することができる。これらのアルゴリズムは現在Edwardで開発されています。

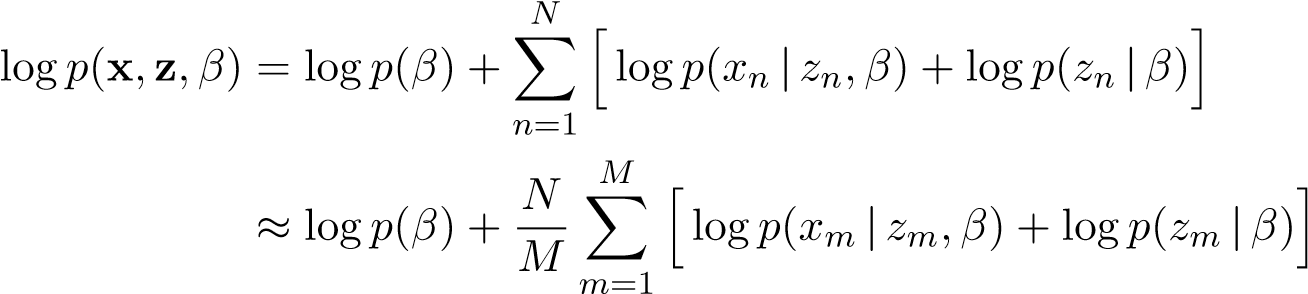
### 2.3.3 推論の構成

エドワードの設計の核心は、推論が別々の推論プログラムの集合として記述できることです。

以下では、局所変数に対する（近似的な）E-ステップと大域変数に対するM-ステップを用いた変分的EMを実証する。我々は、2つのアルゴリズムをインスタンス化し、それぞれが他のアルゴリズムからの推論を条件とし、それぞれの更新を交互に行う (Neal and Hinton, 1993), これは、指数族の厳密EM、対照的発散 (Hinton, 2002)、擬似限界法 (Andrieu and Roberts, 2009)、変分推論内のギブスサンプリング (Wang and Blei, 2012; Hoffman and Blei, 2015) など、他の多くのケースにも拡張される。また、局所的な推論問題の集合を解くメッセージパッシングアルゴリズムを書くこともできる(Koller and Friedman, 2009)。例えば、古典的なメッセージパッシングは、厳密な局所推論を使用し、期待伝播は局所的にKullback-Leibler発散、KL(*pkq*)を最小化する(Minka, 2001)。

### 2.3.4 データのサブサンプリング

確率的最適化（Bottou, 2010）は大規模データへの推論をスケーリングし、確率的勾配ランジュバン力学（Welling and Teh, 2011）や確率的変分推論（Hoffman et al. , 2013）などのアルゴリズムの鍵となります。このアイデアは，不偏な方法でモデルの対数結合密度を安価に推定することである．各ステップでは，サイズ*Mの*データ集合{*xm*}をサブサンプリングし，局所変数に関して密度をスケーリングします．

*.*

確率的最適化をサポートするために、我々は完全モデルの部分グラフのみを表現する。これは，不合理なメモリ消費を引き起こす可能性のある完全モデルの再定義を防ぐ(Tristan et al. , 2014)．初期化の際には，引数を適切にスケーリングするために辞書を渡します．図2.9を参照してください。

β = 正規(mu=tf. zeros([K, D]), sigma=tf. ones([K, D]) z = カテゴリカル(logits=tf. zeros([M, K]) x = 正規(mu=tf. gather(β, z), sigma=tf. ones([M, D]) qbeta = 正規(mu=tf.variable(tf. zeros([K, D])), sigma=tf. nn. softplus(tf.Variable(tf. zeros([K, D]))))) qz = カテゴリー(logits=tf.変数(tf. zeros([M, D]))

*β*

**z**

*m*

**x**

*m*

*M*

推論 = ed.VariationalInference({beta: qbeta, z: qz}, data={x: x\_batch}) inference. initialize(scale={x: float(N)/M, z: float(N)/M})

**図2.9.**階層モデルによるデータのサブサンプリング。我々は、フルモデルの部分グラフを定義し、*Nで*はなく*Mの*サイズのプレートを形成します。

概念的には、 scale 引数は、そのランダム変数 *n/m を何*度も見たかのように、各ランダム変数のプレートに対するスケーリングを表しています。例として、付録A.6は、エドワードで確率的変分推論を実装する方法を示しています。このアプローチは、ストリーミングデータ(Doucet et al. , 2000; Broderick et al. , 2013; McInerney et al. , 2015)、動的なバッチサイズ、および

サブグラフ上での作業がすぐには適用されない(Binder et al. , 1997; Johnson and Willsky, 2014; Foti et al. , 2014)。

## 2.4 実験

このセクションでは、エドワードの2つの主な利点である柔軟性と効率性について説明する。前者については、同じモデル上で異なる推論アルゴリズムを容易に比較できることを示す。後者については、計算グラフを利用することで、どのようにして大幅な高速化を容易に得ることができるかを示す。

|  |  |
| --- | --- |
| 推論方法 | 負の対数尤度 |
| vae (Kingma and Welling, 2014b) | ≤ 88.2 |
| 無解析ＫＬ | ≤ 89.4 |
| 解析的エントロピーを持つベ | ≤ 88.1 |
| スコア関数勾配付きベエ | ≤ 87.9 |
| 流れの正規化（Rezende and Mohamed, 2015 | ≤ 85.8 |
| 階層的変分モデル (Ranganath et al. , 2016b) | ≤ 85.4 |
| 重要度重み付きオートエンコーダー(*K* = 50) (Burda et al. , 2016b) | ≤ 86.3 |
| iwae目的のhvm (*K* = 5) | ≤ 85.2 |
| レニイ・ダイバージェンス(*α*=-1) (Li and Turner, 2016) | ≤ 140.5 |

**表2.1.**2値化されたMNIST上の確率的デコーダの推論手法。確率的プログラミング言語Edward (pl)は便利な研究プラットフォームであり、多くのアルゴリズムの開発と実験を容易に行うことができます。

### 2.4.1 変量推論 の最近の手法

我々は、複雑な推論アルゴリズムを実験するためのEdwardの柔軟性を実証する。図3.4のvaeのセットアップと、2値化されたMNISTデータセット(Salakhutdinov and Murray, 2008)を考えます。データポイントごとに*d* = 50の潜在変数を使用し、ADAMを用いて最適化する。我々は、異なる手法を用いて、vaeセットアップの異なる構成要素を調査する。訓練後、真値の下限であるホールドアウト対数尤度を評価します。

表 4.1 に結果を示す．最初の手法は図3.4のvaeを用いる。次の3つの手法は，同じ vae を使用するが，異なる勾配推定量を適用する：解析的 KL を用いない再パラメータ化勾配，解析的エントロピーを用いた再パラメータ化勾配，およびスコア関数勾配 (Paisley et al. , 2012a; Ranganath et al. , 2014)である．これは典型的には同じオプティマを導くが、異なる収束率である。スコア関数勾配は最も遅かった。分析的エントロピーを持つ勾配は、収束に困難をもたらした：我々は、エントロピーが最適値に近づくにつれて、エントロピーの確率的な推定値に切り替えた。我々はまた、正規化フローの先行を持つ階層的変分モデル(hvms) (Ranganath et al. , 2016b)を使用した；それは潜在変数空間の正規化フロー(Rezende and Mohamed, 2015)と同様の結果をもたらし、重要度加重自動エンコーダー(iwaes) (Burda et al. , 2016b)よりも優れた結果をもたらした。

また、iwae目的語を用いたhvmsや、デコーダ上の画素強度値データを用いた ganベースの最適化、デコーダ上のRényi発散などの新しい組み合わせについても研究を行っている。

|  |  |
| --- | --- |
| 確率的プログラミングシステム | ランタイム |
| 手書きNumPy（1CPU | 534 |
| PyMC3 (12 CPU) (Salvatier et al. , 2015) | 30.0 |
| **エドワード（12CPU** | **8.2** |
| 手書きのTensorFlow（GPU | 5.0 |
| **エドワード（GPU** | **4.9** |

**表2.2:** 大規模ロジスティック回帰のためのhmcベンチマーク．Edward(GPU)は他のシステムと比較して有意に高速です。また，Edwardはオーバーヘッドがなく，手書きのTensorFlowと同等の速度を実現しています．

Rényi発散は対数尤度を直接最適化していないので、パフォーマンスは良くありません。重要なポイントは、Edwardが便利な研究プラットフォームであることです：これらはすべて、与えられたスクリプトを簡単に修正することができます。

### 2.4.2 GPU加速ハミルトニアンモンテカルロ法

*# モデル*

x = tf.変数(x\_data, trainable=False) beta = 正規(mu=tf. ゼロ(D), sigma=tf. ワンズ(D)) y = ベルヌーイ(logits=tf. ドット(x, beta))

*β*

**y**

*n*

**x**

*n*

*N*

*# 推論*

qbeta = Empirical(params=tf.変数(tf. zeros([T, D])) ） 推論 = ed.HMC({beta: qbeta}, data={y: y\_data}) inference.run(step\_size=0.5 / N, n\_steps=10)

**図2.10:** hmcを用いたベイズロジスティック回帰のためのエドワードプログラム.

我々は、最新のハードウェアである12コアIntel i7-5930K CPU（3.50GHz）とNVIDIA Titan X (Maxwell) GPUを用いて、ハミルトニアンモンテカルロ法（mc; Neal, 2011）の反復回数を固定したランタイムのベンチマークを行った。Covertypeデータセット(*N* = 581012, *D* = 54; 回答は2値化)にEdwardとPyMC3を用いてロジスティック回帰を適用しました(Salvatier et al. , 2015)．繰り返し100回のhmcを実行し，1回の繰り返しにつき10回のリープフロッグ更新を行い，ステップサイズは0*.*5*/N，*精度は単精度としました．図2.10にEdwardのプログラムを示します．

表2.2はランタイムを示しています。Edward (GPU)はPyMC3 (12 CPU)に比べて劇的に6倍のスピードアップを実現しています。これは，計算グラフの上にplを構築することの価値を示しています．この高速化は、モデルの対数尤度を計算する際の高速な行列の乗算に起因します; GPUはこの計算を効率的に並列化できます。我々は、ディープ・ニューラル・ネットワークのように、ボトルネックが行列の乗算でもあるモデルでも、同様の高速化が期待できます。

高速化には様々な理由があります。PyMC3については、Edwardの高速化は、EdwardのTensorFlowと比較して、PyMC3のTheanoバックエンドの結果ではないことに注意してください。むしろ、PyMC3 はすべての計算に Theano を使っていないので、NumPy との通信オーバーヘッドが発生しています。(PyMC3 は GPU を使用した場合、実際には遅くなりました。) Edward のデザインを Theano に移植しても、同様のスピードアップが得られると予測しています。

これらの高速化に加えて、Edwardは実行時のオーバーヘッドがないことを強調している。2.3.1節に続いて、これは推論のための計算グラフが実際にはEdwardと手書きのコードで同じであるからです。

## 2.5 ディスカッション

我々は、確率的モデルと推論のための構成表現を持つチューリング完全計画法であるEdwardを説明した。エドワードは、従来のディープラーニングと同等の柔軟性と計算効率を持つように、確率的プログラミングの範囲を拡大する。柔軟性については、エドワードが多様な構成可能な推論手法を使用し、変分的推論と生成的逆襲ネットワークの最近の進歩を取り入れ、推論アルゴリズムを細かく制御する方法を示した。効率性については、エドワードが計算グラフを活用して、高速で並列化可能な計算を実現し、大規模データにスケールアップし、手書きのコードに比べて実行時のオーバーヘッドを発生させない方法を示した。

他の言語設計と同様に、エドワードは、研究のための柔軟性とスピードを追求するために、トレードオフを行っています。例えば、複雑な制御フローと再帰を伴うプログラムをより容易にすることが、エドワードの未解決の課題である。表現は可能ですが、その柔軟な推論戦略をどのようにして可能にするかは未知数です。さらに、動的な計算グラフフレームワークにエドワードの設計をどのように拡張するかも未解決であり、その方がプログラミングパラダイムの柔軟性は高まるが、性能は犠牲になるかもしれない。確率的プログラミングの次の重要なステップは、Edwardが提供する柔軟性と効率性を維持しつつ、動的計算グラフを活用することである。次の章では、このような進歩について議論する。

# 第3章：単純、分散、高速化された確率プログラミング

## 3.1 イントロダクション

ディープラーニングにおける多くの開発は、モデルと計算の間の境界線を曖昧にしていると解釈することができます。中には、単にモデルを訓練するだけではなく、一般的なプログラム合成を行うことを目的とした「微分可能なプログラミング」の新しいパラダイムを宣言するところまで行ったものもあります1。1 この見解では、注意(Bahdanau et al. , 2015)とゲーティング(Hochreiter and Schmidhuber, 1997)はブール論理を記述し、スキップ接続(He et al. , 2016)と条件付き計算(Bengio et al. , 2015; Graves, 2016)は制御フローを記述し、外部メモリ(Giles et al. , 1990; Graves et al. , 2014)は関数の内部スコープ外の要素にアクセスします。学習アルゴリズムもまた、ますます動的になっている：例えば、学習するための学習(Hochreiter et al. , 2001)、ニューラルアーキテクチャ探索(Zoph and Le, 2017)、層内最適化(Amos and Kolter, 2017)などである。

微分可能なプログラミング・パラダイムは、モデラーが計算コストを明示的に考慮することを奨励します：モデルの統計的特性（「モデルは真のデータ分布をどれだけうまく捉えているか」）だけでなく、計算コスト、メモリコスト、帯域幅コスト（「どれだけ効率的に学習して予測を行うことができるか」）も考慮しなければなりません。この哲学により、研究者は、現代のハードウェアが可能にするもののギリギリのところで動作するディープラーニングシステムを設計することができます。

対照的に、確率的プログラミング・コミュニティは、モデルと計算の間に厳しい線引きをする傾向がありました：第1に、確率的モデルをプログラムとして指定します。

"推論クエリ」を用いて、与えられたデータを用いてモデルを自動的に訓練する(Spiegelhalter et al. , 1995; Pfeffer, 2007; Carpenter et al. , 2016)。この設計選択は、数十億パラメータのモデルを訓練するためには、モデル計算をアクセラレータ間で分割し、通信をスケジューリングする必要があります(Shazeer et al. , 2017)が、真に大規模なスケールで確率モデルを実装することを困難にします。のような最近の進歩は、以下のようなものです。

status/9488119250386698241 Recentadvocatesofthis[)tr](https://twitter.com/tdietterich/status/948811925038669824) endandYannincludeLeCunTom ([https://www.facebook.com/yann.lecun/posts/Dietterich(https://twitter.com/tdietterich/)](https://twitter.com/tdietterich/status/948811925038669824)

[10155003011462143](https://www.facebook.com/yann.lecun/posts/10155003011462143)).プログラミング言語の分野では古典的な考え方である(Baydin et al. , 2015)。

エドワード(Tran et al. , 2017)は、深層学習における推論手順のより細かい制御を可能にした。

(Mansinghka et al. , 2014; Bingham et al. , 2018)も参照のこと)。しかし、それらはいずれも推論を閉じたシステムとして扱う：このため、任意の計算を用いて、また生産プラットフォームなどのより広範な機械学習エコシステムを用いて構成することは困難である(Baylor et al. , 2017)。

この論文では、ディープラーニングのエコシステムに確率的プログラミングを埋め込むためのシンプルなアプローチを説明する。この軽量なアプローチは、柔軟なモデリングのための低レベルのモダリティを提供するもので、ディープラーニングの学習者は確率的プリミティブを用いた柔軟なプロトタイピングから恩恵を受け、確率的モデラーは身近な数値エコシステムとの緊密な統合から恩恵を受けることができる。

**貢献。**我々は、確率的プログラミングの核心を、単一の抽象化-ランダム変数-に絞り込んでいる。既存の言語とは異なり、学習のための抽象化は存在しない: アルゴリズムは、例えば、モデルを入力（別の関数）として取り、テンソルを返す関数であるかもしれない。

この低レベル設計には、2つの重要な意味があります。第一に、それは研究の柔軟性を可能にする：研究者は訓練とテストのためにモデル計算を自由に操作できる。第二に、テンソル処理ユニット(tpus)のようなアクセラレータを使用して、より大きなモデルを実現することができる(Jouppi et al. , 2017)：tpusは、物理的なネットワークトポロジーに計算とメモリを分散させるために、特殊な処理を必要とする。

我々は、3つのアプリケーションを例示する：tpusを用いたモデル-並列分散自動エンコーダー(vae) (Kingma and Welling, 2014a)、tpusを用いたデータ-並列自己回帰モデル(Image Transformer (Parmar et al. , 2018)、およびマルチGPUのNo-U-Turn Sampler(nuts) (Hoffman and Gelman, 2014)。64x64 ImageNet上の最先端のvaeと256x256 CelebA-HQ上のImage Transformerの両方について、我々のアプローチは、1から256までのtpuv2チップで最適な線形高速化を達成しています。ナッツでは、PyMC3に対して37倍の高速化が見られます(Salvatier et al. , 2016)。

## 3.2乱数 があれば大丈夫

我々はEdward2の確率的プログラムの概要を説明する。これらのプログラムに必要な抽象度は、ランダム変数という1つの抽象度だけです。

次に、トレースを使って柔軟な低レベルの操作を行う方法を説明します。

### 3.2.1確率 的プログラム、変分プログラム、その他多数

#### import neural\_net\_negative, neural\_net\_positive

defp =modeled.ベータ(): 1., 1., 名前"p" defepsvariational= ed.Normal((x):0., 1., sample\_shape=2) x = ed.ベルヌーイ(probssample\_shape=p,=) )=50, ifelsereturneps[: 0]neural\_net\_positive(eps[> 0. 1], x) **return**  xname="x") **戻り値** neural\_net\_negative(eps[1], x)

**図3.1:** ベータベルヌーイプログラム。ea-**[[1]](#footnote-1)図3. :** 変分プログラム(Ranganath et al. , , gerモードでは、model()はバイナリベクトル2016a)を生成し、ea-**図3.で**利用可能。50要素のPython制御。グラフモードでは、model()の再フローは、生成プロセスに適用可能です：与えられたターンaは、TensorFlowコインフリップで評価されるようにopを、プログラムは、2つのセッションのいずれかから生成します。その出力は異なる形状を持つことができます。

と構造）。

Edward2は、計算可能な確率分布をPythonの関数（プログラム）として再定義します。通常，この関数は生成プロセスを実行し，サンプルを返します．2 プログラムへの入力

スコープされた任意のPython変数-representの値の分布条件を表します。

プログラムのランダムな選択肢を指定するために、Edward（Tran et al.

2016a)は、Zhusuan (Shi et al. , 2017)とProbtorch (Probtorch Developers, 2017)によって同様に構築されてきました。ランダム変数は、log\_probやsampleのようなメソッドを提供します。

TensorFlow の分布（Dillon et al. , 2017）。さらに、エドワードランダム変数は、TensorFlow演算の計算グラフを補強する：各ランダム変数**xは、**グラフ内のサンプリングされたテンソル**x**∗〜*p*(**x**)に関連付けられている。

図3.1に例を示します。 ベータベルヌーイモデル、*p*(**x***,***p**) = ベータ(**p** **ベルヌー**イ(*xn* |**p**)) ここで、**pは**50個のデータ点**x**∈{0*,*1}50の間で共有される潜在確率です。ランダム変数xは50次元で、テンソル**p**∗〜*p*(**p**)でパラメータ化されています。TensorFlowの一部として、Edward2は2つの実行モードをサポートしています。ここでは、model()が生成プロセスを呼び出し、50要素のバイナリベクトルを返します。ここでは、model() は遅延された TensorFlow ベクトルを返します。

重要なことは，下流での使用にかかわらず，すべての分布は確率的プログラムとして書かれていることです．図3.2は、暗黙の変量プログラム、すなわち、サンプリングを認めるが、取り扱い可能な密度を持たないかもしれない変量分布を示しています。一般に、分散プログラム(Ranganath et al. , 2016a)、提案プログラム(Cusumano-Towner and Mansinghka, 2018)、敵対的訓練における識別器(Goodfellow et al. , 2014)は計算可能な確率分布である。これらの確率プログラムを操作するメカニズムがあれば、強力な推論パラダイムをサポートするために追加の抽象化を導入する必要はない。以下では、モデル-並列vaeを用いて、この柔軟性を実証する。

### 3.2.2 例.TPUを用いたモデル-パラレルVAE

図3.4は、デコーダ、先行詞、エンコーダからなるモデル-並列可変型自動エンコーダ(vae)を実装したもので、デコーダは16ビットの音声を生成します。デコーダは16ビットの音声（[0*,*216 - 1]の*T*値を[0*,*1]に正規化したシーケンス）を生成します。この先行処理は，*T*/2ステップの粗い8ビット分解能を表す潜像を仮定したものであり，同様のアーキテクチャで学習可能である．エンコーダは各サンプルを粗い解像度に圧縮する；それは圧縮関数によってパラメータ化される．

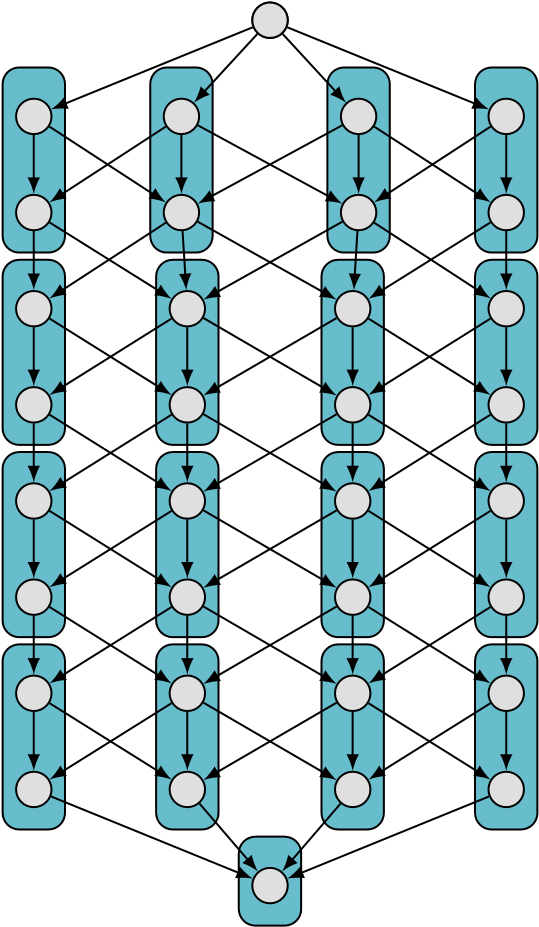
tpuクラスターはコアをトロイダルネットワークに配置し、例えば512個のコアを16x16x2のトーラス相互接続として配置します。クラスタを利用するために、プリファとデコーダは分散自己回帰フローを適用します (図 3.3)。これらは、2つの方法で仮想4x4トポロジにまたがって計算を分割します。"2つのフローがそれぞれ異なるコアに属している場合の「クロスフロー」と、4つの独立したフローが自己回帰的順序を尊重してレイヤーを適用している場合の「インサイドフロー」です（スペースの都合上、フロー内での分割のためのコードは省略します）。エンコーダは圧縮機を介して計算を分割しますが，スペースの都合上，これも省略しています．

確率的プログラムは簡潔です。自己回帰流やマルチスケール潜在変数などの最近の進歩を取り入れており、16x16のTPUV2チップ（512コア）で、モデルを4.1TBのメモリに分割して最大1016FLOPSまで利用できる、これまでにないアーキテクチャを実現しています。vae-distribution、アーキテクチャ、計算配置のすべての要素が拡張可能です。トレーニングには、典型的なTensorFlowのOPSを使用します。

### 3.2.3 トレーシング

我々は確率的プログラムを任意のPython関数として定義した。柔軟な学習を可能にするために、自動微分（Maclaurin et al. , 2015）と同様に、確率的プログラミング（例：Mansinghka et al. , 2014; Tolpin et al. , 2016; Ritchie et al. , 2016; Ge et al. , 2018; Bingham et al. , 2018）で使用されている古典的な手法であるトレーシングを適用します。トレーサーは言語のプリミティブ操作のサブセットをラップし、それらの操作が実行される直前にトレーサーが制御を傍受できるようにする。

**import SplitAutoregressiveFlow**, **masked\_network** tfb = tf. contrib. distributions. bijectors

**クラス DistributedAutoregressiveFlow**(tfb. Bijector).

**def** \_\_init\_\_(flow\_size=[4]\*8): self.flow = [] **for** num\_splits **in** flow\_size: flow = SplitAutoregressiveFlow(masked\_network, num\_splits) self.flow.append(flow)

自己.フロー.append(SplitAutoregressiveFlow(masked\_network, 1)) super(DistributedAutoregressiveFlow, self).\_\_init\_\_()

**def** \_forward(self, x).

l, flow **in** enumerate(self.flow).

**で、**tf. device(tf. contrib. tpu. core(l/2)): x = flow. forward(x)

**return** x **def** \_inverse\_and\_log\_det\_jacobian(self, y): ldj = 0. **for** l, flow **in** enumerate(self. flows[::-1]).

tf. device(tf. contrib. tpu. core(l//2)): y, new\_ldj = flow. inverse\_and\_log\_det\_jacobian(y) ldj += new\_ldj

**戻り値** y, ldj

**図3.3:** 分散自己回帰フロー。**(右)** デフォルトの長さは8で，それぞれ4つの独立したフローを持つ．各フローは、自己回帰的な順序を尊重したレイヤーを介して入力を変換する。各コアは2つのフローを計算し、ローカルに接続されている。仮想トポロジーは物理的なTPUトポロジーと一致しており、4×4コアの場合は正確であり、16×16コアの場合はデータ並列化のために複製されている。

**import upsample**, **compressor def** prior().

*""""8ビットの潜伏体への均一ノイズ、[u1,...,u(T/2)] -> [z1,...,z(T/2)] """* dist = ed.Independent(ed.Uniform(low=tf.zeros([batch\_size, T/2]))

edを**返します**。TransformedDistribution(dist, DistributedAutoregressiveFlow(flow\_size)) **def** decoder(z).

*""""均一ノイズ＋潜伏～16ビットオーディオ、[u1,...,uT], [z1,...,z(T/2)] -> [x1,...,xT]" "d*ist = ed.Independent(ed.Uniform(low=tf. zeros([batch\_size, T])) dist = ed.TransformedDistribution(dist, tfb.Affine(shift=upsample(z))) **return** ed.TransformedDistribution(dist, DistributedAutoregressiveFlow(flow\_size)) **def** encoder(x).

*""""16ビットオーディオから8ビットラテント、[x1,...,xT] -> [z1,...,z(T/2)] """loc*, log\_scale = tf. split(compressor(x), 2, axis=-1) **return ed.**Normal(loc=loc, scale=tf. exp(log\_scale))

**図3.4.**8 ビットの潜在値から 16 ビットの音声を生成する tpus を用いたモデル-並列 vae。プリファレンスとデコーダは，分散自己回帰フローに従って計算を分割します．エンコーダは圧縮機に応じて計算を分割することができますが，ここではスペースの都合上省略します．

STACK = [**ラムダ** f, \*a, \*\*k: f(\*a, \*\*k)]

@contextmanager **def** trace(tracer).append(tracer) **yield** STACK. pop()

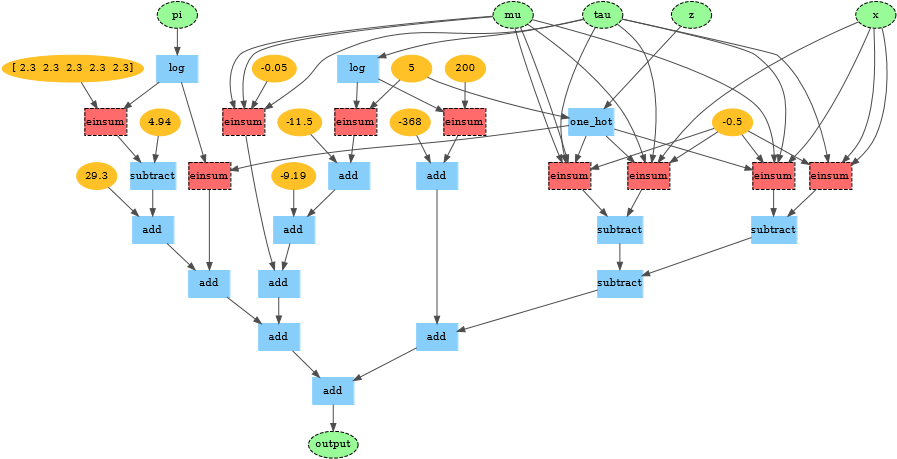
**def** traceable(f)。

**def** fwrapped(\*a, \*\*k).

STACK[-1](f, \*a, \*\*k)

巻き**戻し**

**図 3.5: トレースの**最小実装 trace はコンテキストを定義し，その間に実行されたトレース可能な操作はすべて tracer の呼び出しに置き換えられます．



**図3.6.**プログラムの実行。有向非周期グラフであり、対数確率の蓄積や条件付き独立性の発見など、様々な操作のためにトレースされています。

**def** make\_log\_joint\_fn(model).

**def** log\_joint\_fn(\*\*model\_kwargs).

**def** tracer(rv\_call, \*args, \*\*kwargs): name = kwargs.

**def** mutilate(model, \*\*do\_kwargs).

**def** mutilated\_model(\*args, \*\*kwargs): **def** tracer(rv\_call, \*args, \*\*kwargs).

kwargs[rv = rv\_call("value"\*args,] = model\_kwargs\*\*kwargs) . get(name)nameifreturnname= kwargsindo\_kwargs[name]do\_kwargs:. get("name")

log\_**probsreturn** rv. append(tf. sum(rv. log\_prob(rv))) withreturntrace(tracer):rv\_call(\*args, \*\*kwargs)

log\_**probswith** trace(tracer):= [] log\_**probswith** trace(tracer) returnreturnmutilated\_modelmodel(\*args, \*\*kwargs)

returnreturnmodel(log\_joint\_fnsum\*\*model\_kwargs)(log\_probs) **図3.8:**modelA高次関数を入力としてプログラムし、それを返します。

モデル**図3.7:**プログラムを入力とし、そのlog-jointA高次関数を返します 。介入は条件付けとは異なり、密度関数を変更しません 。

図 3.5 はコアとなる実装を示しています: 10 行のコードです。 [[2]](#footnote-2)trace はコンテキストマネージャで、入力時にはスタックに呼び出し可能なトレーサをプッシュし、終了時にはスタックからトレーサをポップします。Edward2 はランダム変数を登録します: 例えば、Normal = traceable(edward1.Normal)。

また，トレースの実装は数値バックエンドに依存しません．付録B.1では図3.5を適用してSciPyの上にEdward2を実装しています．

### 3.2.4 トレーシングアプリケーション

トレースは確率的プログラミングのための一般的なツールです。しかし、他の言語では、トレースは主に推論の「メタプログラミング」手続きを可能にするための実装の詳細として機能します。我々のアプローチでは、トレースを柔軟な計算のためのユーザレベルの技術として推進している。2つの例を紹介しますが、どちらもユーザがトレーシングにアクセスしないと実装が困難です。

図3.7に make\_log\_joint ファクトリ関数を示します。これは，モデルプログラムを入力として受け取り，トレースを挟んでその結合密度関数を返します．我々は、入力にランダム変数の値を設定し、副作用としてログ確率を蓄積するトレーサを使用して実装します。3.3.3節では、make\_log\_jointを変分推論アルゴリズムに適用しています。

図3.8は，因果介入(Pearl, 2003)を例示しています：これは，名前で索引付けされたランダム変数を別のランダム変数に設定することで，プログラムを"変異させる"ものです．この効果は、子孫以外の子孫はそのままにして、子孫に伝搬されることに注意してください。他の確率論的操作は、より自然に評価の"pull"モデルに従います。平均場の変分推論では、単一の因子に対応するエネルギー項を評価する必要があります。

## 3.3 例。低レベル関数の学習

確率的プログラムとその計算を低レベルのトレース関数で操作する方法について述べた。既存の確率プログラムとは異なり、学習のための抽象化がされていない。以下に例を示す．

**import get\_channel\_embeddings**, **add\_positional\_embedding\_nd**, **local\_attention\_1d def** image\_transformer(input, hparams): x = get\_channel\_embeddings(3, inputs, hparams. hidden\_size) x = tf. reshape(x, [-1, 32\*32\*3, hparams. hidden\_size])

x = tf. pad(x, [[0, 0], [1, 0], [0, 0]])[:, :-1*, :] # ピクセルを右に移動します。*

1. = nn. dropout(x, keep\_prob=0.7) **for** \_ **in** range(hparams. num\_layers).
2. = レイヤーを使用しています。7) + x, begin\_norm\_axis=-1) y = tf. layers. dense(x, hparams. filter\_size, activation=tf. nn. relu) y = tf. layers. dense(y, hparams. hidden\_size, activation=None)

layer\_norm(tf. nn. dropout(y, keep\_prob=0.7) + x, begin\_norm\_axis=-1)

logits = tf. layers. dense(x, 256, activation=None) **return** ed.カテゴリカル(logits=logits).log\_prob(input)

loss = -tf. reduce\_sum(image\_transformer(input, hparams))# *入力は形状を持ちます [batch,32,32,3]* train\_op = tf. contrib. tpu.CrossShardOptimizer(tf. train.AdamOptimizer()). minimize(loss)

**図3.9:** tpusを用いたデータ並列画像変換器(Parmar et al. , 2018)。tpus は自己注意を伴う画像のバッチの対数確率を計算するニューラル自己回帰モデルである。我々の軽量設計は、モデルを対数確率関数として表現し、学習することを可能にする。これは、プログラムを生成過程として表現する一般的な方法よりも効率的である。埋め込み関数と自己注意関数は，環境中で仮定されており，Tensor2Tensor (Vaswani et al. , 2018)で利用可能である．

### 3.3.1 例.TPUを用いたデータ並列画像変換器

これまでのところ、すべてのモデルの統一的な表現、典型的には生成的なプロセスとしての表現に焦点を当ててきました。しかし、これはモデルによっては実際には非効率的であることがある。我々の軽量なアプローチは、学習に必要なシグネチャを持たないため、代替的なモデル表現を可能にします。 [[3]](#footnote-3)

例えば、図3.9はImage Transformer (Parmar et al. , 2018)を対数確率関数として表したものです。Image Transformer は画像生成のための最先端の自己回帰モデルであり、右シフトされた画像のバッチによってパラメータ化されたカテゴリ分布、埋め込み、自己注意層とフィードフォワード層の交互のシーケンス、および出力層から構成されています。この関数は，画像に関するlog\_probを計算し，ピクセル次元にわたって並列化します．log-probabilityとは異なり、サンプリングでは、自己回帰性をシリアルにプログラミングする必要があり、非効率的で実装が困難です。5 log-probability表現では、tpusとのデータ並列化は

**def** nuts(... ): samples = [] **for** \_ **in** range(num\_samples): state = set\_up\_trajectory(... ) depth = 0 **while** no\_u\_turn(state): state = extend\_trajectory(depth, state) depth += 1

samples.append(ステート)

**返却**サンプル

**def** extend\_trajectory(depth, state): **if** depth == 0.

state = one\_leapfrog\_step(state)

**else**: state = extend\_trajectory(depth-1, state) **if** no\_u\_turn(state): state = extend\_trajectory(depth-1, state)

**復帰**状態

**図 3.10:** No-U-Turn Sampler (Hoffman and Gelman, 2014) のコアロジック。このアルゴリズムは、データに依存した非尾部再帰を持つ。

**図3.11:** 学習では，モデルプログラム**（左）**と変分プログラム（**右）**，またはモデルプログラムとデータテンソル**（下）**のような2つの実行トレースをマッチングさせることがよくあります．赤色の矢印は、先行変数と変量変数の位置を合わせます。青い矢印は観測された変数とデータの位置を合わせています。

は、オプティマイザをクロスシャーディングすることで即時に実行することもできます。train op は TF Estimator でラップしたり、コア間のトレーニングを集約するために手動の tpu ops で適用したりすることができます。

### 3.3.2 例。ノーUターンサンプラー

図3.10は、ハミルトニアンモンテであるNo-U-Turn Sampler (nuts)のコアロジックを示しています。

リープフロッグ積分の際にパス長のハイパーパラメータを適応的に選択するカルロアルゴリズム。

その実装は、Hoffman と Gelman の擬似コードに従ったノンテール再帰を使用しています。

(2014, Alg 6); CPUとGPUの両方に対応しています。完全な実装についてはソースコードを参照してください; 付録B.2はまた、データに依存するwhileループを使用して文法vae(Kusner et al. , 2017)を実装しています。

nutsを統合する能力は、イーガーモードとの相互運用性を必要とします: nutsは、TensorFlow opsで再帰をネイティブに実装することが難しいため、Pythonの制御フローを必要とします(NutsはEdward 1などでは利用できません)。(nuts は Edward 1 などでは利用できません。) しかし、イーガー実行にはトレードオフがあります(我々のアプローチに特有のものではありません)。例えば、グラフモードよりも無視できないほどのオーバーヘッドが発生します。我々の軽量設計は両方のモードをサポートしているので、ユーザはどちらかを選択することができます。

密度からサンプリングへの変換は効率的に行うことができます。この例では、サンプリングは最高でも直列計算が可能であり、性能の最適化は必要ありません。

### 3.3.3 例。確率的プログラムの整列

学習アルゴリズムは、多くの場合、複数の確率的プログラムを操作する必要があります。例えば、変分推論アルゴリズムは、モデルプログラムと変分プログラムの2つのプログラムを入力として受け取り、最適化のための損失関数を計算します。このためには、2つのプログラムの中でどの変数が互いに参照するかを指定する必要があります。

アライメント (図 3.11) を適用します (これは、キーと値のペアの辞書であり、それぞれが 1 つの文字列

(ランダム変数の名前)を別のプログラム(他のプログラムのランダム変数)に合わせることができます。この辞書は、各プログラムの仕様に依存せず、ランダム変数がどのように整列されるかについて柔軟性を提供する。例えば、これにより、先行と変分のトポロジカルな順序付けが逆になるラダーベイズ(Sønderby et al. , 2016)や、先行と変分のパラメータが共有されるVampPriors(Tomczak and Welling, 2018)などが可能になります。

図3.12は、固定前提条件器を用いた勾配降下による変分推論を示しています。make\_log\_joint\_fn (図3.7) を適用し、モデルが名前 'x' のランダム変数を適用すると仮定しています。

(3.2.2節のvaeのような)。これは，エドワード1のアライメントを動的プログラムに拡張したものである(Tran et al. , 2017)：構築時に静的グラフのノードをアライメントするのではなく，実行時に実行トレースのノードをアライメントする．また，Metropolis-Hastingsにおけるモデルプログラムや提案プログラムのアラインメント，敵対者訓練におけるモデルプログラムや識別プログラム，さらには入出力パイプラインにおけるモデルプログラムやデータインフィード関数（「プログラム」）のアラインメントにも応用可能である．

### 3.3.4 例。勾配降下による変量推論による学習

軽量設計は、学習アルゴリズムを柔軟に指定できるだけでなく、柔軟な構成が可能であるという利点があります：ここでは、学習によるネストされた推論を実証します。図3.12は勾配降下を用いた変分推論を行っていることを思い出してください。図3.13は，その勾配降下アルゴリズムの出力に勾配降下を適用している．これにより，最適な前置詞が得られる (Andrychowicz et al. , 2016)．これは，学習アルゴリズムが数値演算の単なる合成であり，その合成は完全に微分可能であるために可能である．この微分可能性は，推論オブジェクトを操作するEdwardでは不可能である．[[4]](#footnote-4)付録B.3も参照のこと。

これは、変分推論の中でマルコフ連鎖モンテカルロを説明するものである。

**import model**, **variational**, **align**, **x** grad\_fnoptimizer= tfe= tf. gradients\_function(train).AdamOptimizer(0.1) defdeftrainloss\_fn(precond(x):). 範囲**内の**foroptimizer\_. apply\_gradients(grad\_fn())(100).

qzlog\_joint\_fn= variational(x) **図 3.13:** 学習による学習。kwargs = {align[rv= make\_log\_joint\_fn(model).name]: rv 訓練のための最適な前処理(図3.12)を見つけます。

rv **in** toposort(qz)}**の場合** 学習アルゴリズム全体を差別化することで

エネルギー = log\_joint\_fn(x=x, \*\*kwargs) entropy = sum([tf. reduce\_sum(rv. entropy()) **for** rv **in** toposort(qz)]) **return** -energy - entropy.

grad\_fn = tfe. implicit\_gradients(loss\_fn) optimizer = tf. train.AdamOptimizer(0.1) **for** \_ **in** range(500): grads = tf. tensordot(precond, grad\_fn(x), [[1], [0]] ) optimizer. apply\_gradients(grads)

**戻り値** loss\_fn(x)

**図3.12:** 前提条件付き勾配降下を用いた変分推論。Edward2は確率的プログラムの記述と学習のための任意のTensorFlow計算を提供している。

## 3.4 実験

我々は、ディープラーニングエコシステムに確率的プログラミングを埋め込むための軽量なアプローチを導入した。ここでは、このようなアプローチが、マルチTPU vaesや自己回帰モデル、マルチGPUナッツのための最新のハードウェアを利用する場合に特に有利であることを示す。CPU実験では6コアのIntel E5-1650 v4を使用し、GPU実験では1〜8個のNVIDIA Tesla V100 GPUを使用し、TPU実験では様々なトポロジーの配置の下で第2世代チップを使用しています。TPUv2チップは2つのコアで構成されており，それぞれのコアは16/32ビット精度の混合で約22テラフロップス（32ビット精度のNVIDIA Tesla P100 GPUの約2倍のフロップス）を特徴としています．すべての分散実験では，データ並列化のためにオプティマイザをクロスシャードしています：各シャード（コア）はバッチサイズが1です． すべての数値は5回の実行で平均化されています．

### 3.4.1 高品質画像の生成

64x64 ImageNet (Oord et al. , 2016)での非オートレグレッシブ生成と256x256 CelebA-HQ (Karras et al. , 2018)でのオートレグレッシブ生成について、最先端に近い結果（"bits/dim"）を得たモデルを評価する。1秒間に処理された例（データポイント）の数の壁時計時間を評価する。

TPUのスピードアップ、傾き=7.49 TPUのスピードアップ、傾き=1.40

50

100

150

200

250

300

350

400

事例・事例

500

1000

1500

2000

事例・事例

011664128256 011664128256

# TPU v2 チップ # TPU v2 チップ

**図3.14.** ベクトル量子化された VAE **図3.15:** 256x256のイメージトランス

64x64 ImageNet. CelebA-HQ.

|  |  |
| --- | --- |
| システム | ランタイム(ms) |
| PyMC3 (CPU) | 74.8 |
| 手書きTF(CPU) | 66.2 |
| エドワード2（CPU | 68.4 |
| 手書きTF(1GPU) | **9.5** |
| **エドワード2（1 GPU** | **9.7** |
| **エドワード2（8GPU** | **2.3** |

**表3.1:** ベイズロジスティック回帰におけるNo-U-Turn Samplerのリープフロッグステップあたりの時間エドワード2

(GPU)はPyMC3 (CPU)に比べて37倍の高速化を達成していますが、Edwardではダイナミズムは利用できません。また、Edward2は、手書きのTensorFlowコードよりも無視できるほどのオーバーヘッドが発生します。

64x64 ImageNetでは，ソフトEMで訓練されたベクトル量子化変分自動エンコーダーを使用する (Roy et al. , 2018)．これは64x64x3ピクセルの画像を8x8x8x10のテンソルにエンコードし、コードブックサイズは256で、各コードベクトルは512次元である。事前には、局所的な1次元自己アテンションの6層を有する画像変換器(Parmar et al. , 2018)を使用する。エンコーダは、カーネルサイズ５およびストライド２を有する４つの畳み込み層、２つの残差層、および密層を適用する。復号器は、密な層の逆、2つの残留層、4つの転置された畳み込み層を適用する。

256x256のCelebA-HQについては、メモリ内のモデルを適合させるために、比較的小さなImage Transformer（Parmar et al. , 2018）を使用する。これは、ブロック長が256、隠蔽サイズが128、注意キー/値チャネルが64、フィードフォワード層が256のローカル1D自己注意の5つの層を適用する。

図3.14と図3.15は、両方のモデルにおいて、Edward2がtpuv2のチップ数が1から256までの間で最適な線形スケーリングを達成していることを示しています。また、実験では、バッチサイズを大きくすることで学習が劇的に速くなることがわかりました。

### 3.4.2 ノーUターンサンプラー

我々は、No-U-Turn Sampler (nuts, (Hoffman and Gelman, 2014))を用いて、アクセラレータ上での動的アルゴリズムの威力を説明している。

PyMC3 (Salvatier et al. , 2016)で実装されたナッツを用いたベイズロジスティック回帰を、我々のイーガーモードTensorFlow実装と比較する。モデルの対数関節密度は

"手書き"のTensorFlowコードとEdward2の確率的プログラムによるものである。Covertypeデータセット（581,012個のデータポイント、54個の特徴、結果は2値化されています）を使用しています。適応的なサンプリングは、ナッツの反復が飛躍的に異なる数の飛躍的ステップを取ることになるかもしれないので、我々は、5つの完全なナッツの軌道（これらの実験では、通常、合計で約1000の飛躍的ステップに相当します）で平均した飛躍的ステップあたりの平均時間を報告します。

表 3.1 に示すように，Edward2(GPU)はマルチスレッド CPU の PyMC3 に比べて最大 37 倍の高速化を実現している． [[5]](#footnote-5) また、Edward2 は原則としてトレース機構のためにイーガーモードではオーバーヘッドが発生しますが、手書きの TensorFlow コードとの速度差はごくわずかです（実行間の変動よりも小さい）。このことは、pl形式主義のパワーが無視できるほどのオーバーヘッドを伴うことを示している。

## 3.5 ディスカッション

確率的プログラミングをディープラーニングエコシステムに埋め込むためのシンプルで低レベルなアプローチを記述しました。64x64のImageNet上の最先端のvaeと256x256のCelebA-HQ上のImage Transformerの両方について、1から256までのtpuv2チップで最適な線形高速化を達成しました。ナッツについては、他のシステムと比較して最大100倍のスピードアップを実現しています。

現在の研究として、生成モデルやベイジアンニューラルネットワークの基礎研究の舞台として、この設計を推し進めている(例えば、(Tran and Blei, 2018; Wen et al. , 2018; Hafner et al. , 2018)。我々は次の章でいくつかの例を記述する。さらに、我々の実験では、大規模な高速化を示すためにデータの並列化に依存した。最近の研究では、モデル並列化と超高解像度画像のような大規模な入力に対する並列化の両方のためのニューラルネットワークの分散プログラミングが改善されている(Shazeer et al. , , 2018)。この研究と組み合わせることで、1兆個以上のパラメータと4K以上の解像度(5000万次元)を持つ巨大な確率論的モデルの限界を押し広げることを期待しています。

# 第4章 変分推論の応用

第2章と第3章では、深層確率プログラミングシステムの設計について述べた。この章と次の章では、エドワードの研究が明示的に可能になった新しい変分推論アルゴリズムと確率モデルの開発について掘り下げていきます。

## 4.1 イントロダクション

変分推論は，近似事後推論のための強力なツールである．この考え方は，潜在変数上の分布のファミリーを仮定して，そのファミリーのメンバーの中で最も事後に近いものを見つけることである．もともと1990年代に開発された変分推論は (Hinton and van Camp, 1993; Waterhouse et al. , 1996; Jordan et al. , 1999a)、大規模データセットのためのスケーラブルな最適化の開発 (Hoffman et al. , 2013)、多くのモデルを容易に適合させるための汎用的な戦略の導出 (Ranganath et al. , 2014)、柔軟なパラメトリック近似ファミリーとしてのニューラルネットワークの適用 (Kingma and Welling, 2014b; Rezende et al. , 2014) などで、新たな関心を集めてきた。この研究は、複雑な事後分布の推論を必要とする深層ベイズモデル(Neal, 1990; Ranganath et al. , 2015a)での計算に特に成功している(Hinton et al. , 2006)。

古典的な変分推論では、通常、各潜在変数が独立しており、それ自身の変分分布によって支配される平均場ファミリーを使用します。便利な反面、強い独立性は、データの深い表現の学習を制限します。新しい研究では、潜在変数間の依存性を可能にする、より豊かなファミリを目指しています。依存性を導入する1つの方法は，潜在変数のモデルとして変分系列自体を考慮することです (Lawrence, 2000; Ranganath et al. , 2015b)．これらの変分*モデルは，*平均場の"尤度"を保持するが，変分潜在変数を介した依存性を導入するベイズ階層に自然に拡張される．

この章では、強力な新しい変分モデルである変分ガウス過程(vgp)を開発します。

vgpはベイズ的ノンパラメトリック変分モデルであり、その複雑さは効率的に成長し、*任意の*分布に向かって成長し、手元の推論問題に適応します。この章では、3つの主要な貢献を紹介します。

1. ある条件の下では、vgpは任意の連続的な事後分布を捉えることができます。
2. 我々は、vgpを用いた変分推論のための効率的な確率的最適化アルゴリズムを導出した。このアルゴリズムは、幅広いクラスのモデルで利用可能である。vgpを用いた推論は、ブラックボックス型の変分法である(Ranganath et al. , 2014)。
3. 我々は、教師なし学習の標準的なベンチマーク上でvgpを研究し、深層潜在ガウスモデル(Rezende et al. , 2014)と潜在注意モデルDRAW(Gregor et al. , 2015)の推論に適用した。両方のモデルについて、これまでで最高の結果を報告する。

## 4. 2. 2変量ガウス過程

変分モデルは、潜在変数を変分系列に導入し、事後近似のための豊富な構成を提供する (Ranganath et al. , 2015b)。ここでは、ガウス過程に基づくベイズノンパラメトリック変分モデルである変分ガウス過程(vgp)を紹介します。ガウス過程 (gp) は、複雑さを変えながら下流の分布を捉えることができる潜在変数のクラスを提供します。

最初に変分モデルとガウス過程をレビューします。次に、vgpの力学を概説し、それが普遍的な近似器であることを証明します。

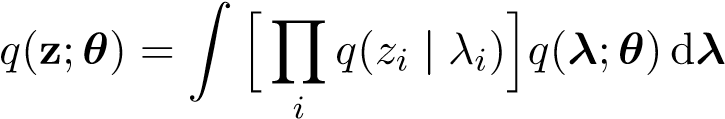
### 4.2.1 変量モデル

*λ*でパラメータ化された分布*q*(**z**; *λ*)のファミリーについて、*変分*推論は発散KL(*q*(**z**; *λ*)*kp*(**z**|**x**))を最小化しようとする。これは、エルボを最大化することと等価である (Wainwright and Jordan, 2008)。エルボは、データの期待対数尤度と変分分布と先行分布との間のKL発散の和として書くことができる。

Ｌ＝*Ｅｑ*（**ｚ**；*λ*）［*ｌｏｇｐ*（**ｘ**｜**ｚ**）］-ＫＬ（*ｑ*（**ｚ**；*λ*）*ｋｐ*（**ｚ**））*．* (4.1)

伝統的に、変分推論では、その密度を分析的に表す形式を持つ、扱いやすい分布の系列を考慮します。一般的な仕様は、完全因数分解された分布*Qi q*(*zi*; *λi*)で、平均場族としても知られています。平均場族は効率的な計算につながりますが、近似の表現力を制限します。

分布の変分系列は，潜在変数 **z の**モデルとして解釈でき，新しい潜在変数を導入することによって，より豊かにすることができる．階層的変分モデルは，平均場パラメータ *q*(*λ*; *θ*) と因数分解された"尤度" *Qi q*(*zi* | *λi*) の変分先行によって指定された分布を考慮する．これは，変分モデルを指定する．

*,* (4.2)

階層的な変分モデルは，古典的な変分族よりも豊かです-表現力は，事前の*q*(*λ*)の複雑さによって決定されます．多くの表現力のある変分法近似は、この構成の下で見ることができます (Saul and Jordan, 1996; Rezende and Mohamed, 2015; Tran et al. , 2015)。

### 4.2.2 ガウス過程

ここでは、ガウス過程(gp)をレビューします（Rasmussen and Williams, 2006）。*m*個*の*ソース・ターゲット・ペアのデータ・セットを考えてみましょう。

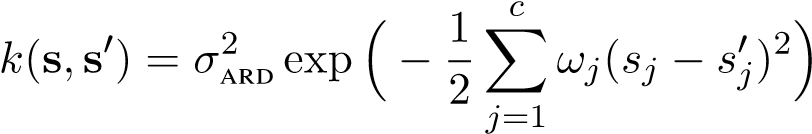
多次元ターゲット *tn*∈*Rd*.全てのソース・ターゲットペアを対象とした関数 *tn* = *f*(*sn*)を学習することを目的としている．*Rc* → *Rd* は未知数である．関数 *f を f* = (*f*1*,....,fd*) とし、各 *fi* : *Rc* → R. gp 回帰により、*f の*関数形を推定する。

*d*

*p*(*f*) = YGP(*fi*; **0***,Kss)。*

i=1

ここで、*Kssは*入力**s***,***s**0のペアに対して評価される共分散関数*k*(**s***,***s**0)を表す。本稿では，自動関連性決定(ard)カーネル

*,* (4.3)

パラメータ *θ* = (*σ*ard2 *,ω*1*,...,ωc*).重み *ωj は，*各次元の重要度を調整する．推論中にゼロにすることができ，自動次元削減につながる．

データDが与えられると、gpの条件付き分布は、入出力ペア間を補間するマッピング上の分布を形成します。

*d*

*ｐ*（*ｆ*｜Ｄ）＝ＹＧＰ（*ｆｉ*；**Ｋ***ξ***ｓＫ**-*ｓ*１ｔ*ｉ，***Ｋ***ξ*-**Ｋ***ξ***ｓＫ**-*ｓ*１**Ｋ**＞ξｓ*）である。* (4.4)

i=1

ここで，**K***ξs は，*入力*ξ*およびすべてのデータ入力 *sn に対する*共分散関数 *k*(*ξ,***s**) を表し，*ti は i 番目の*出力次元を表す．

### 4.2.3 変分ガウス過程

ベイズ的ノンパラメトリック変分モデルである変分ガウス過程(vgp)について説明する。vgpは、潜在入力を生成し、それらをランダムな非線形マッピングでワープし、ワープした入力を平均場分布のパラメータとして使用することで**zを**生成します。ランダムなマッピングは、変分パラメータである"変分データ"に条件付きで描画されます。vgpによって、平均場からのサンプルが任意の複雑な事後分布に従うことが可能になることを示す。

vgpは、事後潜在変数**z**に対して以下のような生成過程を指定しています。

1. 潜在入力*ξ*∈*Rc*：*ξ* ∼ N(**0***,***I)**を描画する*。*
2. 非線形写像*fを*描く .*を*条件とした *Rc* → *Rd の非線形写像を描く。*3.近似事後標本 **z を**描く*．*

図4.1にvgpのグラフィカルモデルを示す。ここでは、変分分布のパラメータである入出力ペアからなる変分データを表しています。マージナル化

*θ* D = {(**s***,***t**)}

*z*

*i*

*f*

*i*

*ξ*

*d*

*z*

*i*

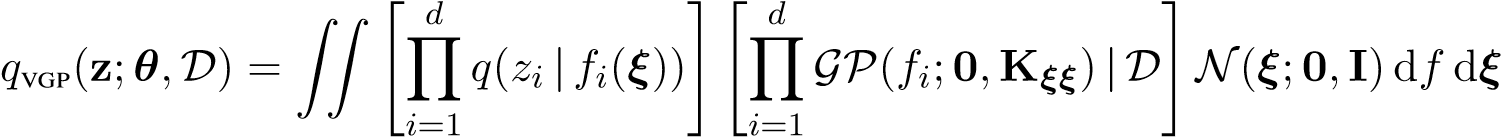
**x**

*d*

変分モデル **(b)** 生成モデル

**図4.1.(a)** 変分ガウス過程のグラフィカル・モデル.vgpは，潜在入力*ξ*のランダムな非線形写像を評価し，写像によってパラメータ化された平均場標本を描画することによって，潜在変数**z**の標本を生成する．これらの潜在変数は，データ**x**に条件付けされた生成モデル**(b)の**事後分布に従うことを目的としている．

すべての潜在入力と非線形マッピングの上で、vgpは

*.* (4.5)

vgpはカーネルハイパーパラメータ*θ*と変分データによってパラメータ化されています。

変分モデルとして、vgpは平均場分布の無限のアンサンブルを形成します。平均場分布は、上記の積分の第1項で与えられます。それは固定関数*f*(*Dot*)と入力*ξに条件付きで*あり、*d*出力*fi*(*ξ*) = *λiは*平均場のパラメータである。vgpは，階層的変分モデル（式4.2）の一形態である（Ranganathら , 2015b）．これは，平均場パラメータの上に連続的なベイズノンパラメトリック優先順位を置く．

平均場とは異なり，vgpは潜在変数間の相関を捉えることができる．その理由は，同じ潜在入力*ξ*で*d個の*独立したgpの描画を評価するからである．これは，それらの出力，平均場パラメータ間の相関を誘導し，したがって，潜在変数間の相関も誘導する．さらに、vgpは柔軟性がある。gpから引き出される複雑な非線形写像は，それが複雑な離散的および連続的な事後を捉えることを可能にする．

我々は、vgpが変分データを必要とすることを強調します。典型的なgp回帰とは異なり、潜在変数**z**の非線形マッピング上の分布を学習するために利用できる観測データはありません。したがって、「データ」は、式4.4の*f*の条件分布に現れる変分パラメータです。これらのパラメータは、特定の入力-入力ペアでのランダムな非線形マッピングを固定します。vgpを最適化するとき、学習された変分データは、事後に近い潜在変数の分布を見つけることを可能にします。

### 4.2.4 万能近似定理

複雑な事後分布を表現するためのvgpの能力を理解するために、ガウス過程の役割を分析します。簡単のために、潜在変数**zが**実値であり、vgpはgpからの関数の出力を事後標本として扱うとします。*ξ* ∼ N(**0***,***I**)を引き、**z** = *f*∗(*ξ*)を計算すると、結果として得られる**z**の分布が事後分布*となる*ような変換である最適関数*f*∗を考えてみましょう。

潜在入力の次元*ξが潜在*変数の数に等しい場合，*f*∗の明示的な構成が存在する．*P*-1を逆事後CDF，Φを標準正規CDFとする．コピュラ文献（Nelsen, 2006）で一般的な手法を用いて，最適な関数は

*ｆ*＊（*ξ*）＝*Ｐ*-１（Φ（*ξ*１*）,...，*Φ（*ξｄ*））*。*

この関数を使って標本 **z を**生成することを想像してみよう．潜在入力 *ξ* ∼ N(**0***,***I**) に対して，標準正規CDFΦは，確率積分変換を適用する：それは，その出力 *ui* = Φ(*ξi*) が [0*,*1]上で一様に分布するような*ξi を*つぶす．逆事後CDFは，その後，一様乱数変数*P*-1(*u*1*,...,ud*) = **z を**事後に従うように変換する．この関数は，厳密な事後標本を生成する．

vgpでは、ランダム関数は変分データの値を補間し、KLダイバージェンスが最小になるように最適化される。したがって、推論の際には、gpの分布はこの最適な関数の周りに集中するように学習される。このような観点から、次のような結果が得られる。

**定理 1** (普遍近似**).***q*(**z**; *θ,*D) を*変分ガウス過程とする。有限の潜在変数と連続分位関数（逆CDF）を持つ事後分布 p*(**z**|**x**) を*考えてみましょう。次のようなパラメータ列* (*θk,Dk*) が存在する。

lim KL(*q*(**z**; *θk,Dk*)*kp*(**z**|**x**)) = 0.

*k*→∞

証明は付録C.2を参照してください。定理1は、厳密に正の密度を持つ任意の事後分布は、vgpで表現できることを示しています。したがって、vgpは事後分布を学習するための柔軟なモデルです。

R

補助的

推論

Q

変分的

推論

P

**図4.2：変分**潜在変数空間Rから事後潜在変数空間Q、データ空間Pへの推論中の領域マッピングのシーケンス。

## 4.3 ブラックボックス推論

我々は、幅広いクラスの生成モデルを対象としたブラックボックス推論のアルゴリズムを導出する。

### 4.3.1 変量目的

元のエルボ（式4.1）は、対数密度logqvgp(**z**)（式4.5）のため、解析的に難解です。これに対処するために、オートエンコーダーに触発された扱いやすい変分目的語を提示します (Kingma and Welling, 2014b)。

モデル証拠*logp*(**x**)の扱いやすい下界は、期待される

エルボからのKL発散項。

*logp*(**x**) ≥ Eqvgp[*logp*(**x**|**z**)] - KL(qvgp(**z)***kp*(**z**))- EqvgphKL(*q*(*ξ,f* |**z**)*kr*(*ξ,f* |**z**))i*.*

ここで*r*(*ξ,f* |**z**)は補助モデルである(次のサブセクションで*rを*記述する)。この目的の様々なバージョンが文献で検討されており(Agakov and Barber, 2004)，最近ではSalimansら(2015)やRanganathら(2015b)によって再検討されている．変分モデルを学習するために，KL(*qkp*)を最小化した後潜変数空間で変分推論を行い，そのために変分潜在変数空間で補助推論を行い，KL(*qkr*)を最小化して補助モデルを学習する．図4.2を参照のこと。

これまでのアプローチとは異なり、自動エンコーダーに接続するために、この変分的な目的語を書き換えます。

Le(*θ,φ*) = Eqvgp[*logp*(**x** | **z**)] - EqvgphKL(*q*(**z**|*f*(*ξ*))*kp*(**z**))i

(4.6)

- EqvgphKL(*q*(*f* |*ξ*; *θ*)*kr*(*f* |*ξ,***z**; *φ*)+ *logq*(*ξ*) - *logr*(*ξ* |**z**)i*.*

ここで，KL 発散は，現在では，トラクタブルな分布上で取られる (付録C.3参照)．自動エンコーダーの用語では，我々は2つの項で正則化された負の再構成誤差の期待値を最大化する：変分モデルと元のモデルの先行値との間の期待発散，および補助モデルと変分モデルの先行値との間の期待発散である．これは、単に変分自動エンコーダー境界の入れ子のインスタンス化である(Kingma and Welling, 2014b)：推論モデルと先行モデルの間の発散は、事後空間と変分空間の両方で正則化される。この解釈は、以前に提案された変分モデルの境界を正当化するものであり、後述するように、確率的最適化の際のより低い分散勾配を可能にします。

### 4.3.2 自動符号化変分モデル

推論ネットワークは、ヘルムホルツ機械(Hinton and Zemel, 1994)、深層ボルツマン機械(Salakhutdinov and Larochelle, 2010)、および変分自動エンコーダ(Kingma and Welling, 2014b; Rezende et al. , 2014)で使用されているような分布の近似の柔軟なパラメータ化を提供する。これは，局所的な変分パラメータをニューラルネットワークからの大域的なパラメータで置き換える．潜在変数*zn*（データ点*xn*に対応する）について、推論ネットワークは*xnを*入力として、その局所変分パラメータ*λnを*出力として受け取るニューラルネットワークを指定する。これは、グローバルパラメータのセットを定義するだけで、推論を償却する。

vgpを自動エンコードするために、変分モデルと補助モデルの両方をパラメータ化する推論ネットワークを指定します。

*xn* 7→*q*(*zn* |*xn*; *θn)。 xn,zn* 7→*r*(*ξn,fn* |*xn,zn*;*φn)。*

形式的には、これらのマッピングの出力は、それぞれパラメータ *θn* と*φn* です。これらのマッピングが、変分モデル*q*と補助モデル*r*の（グローバルな）パラメータ化であることを強調するために、出力を上記の分布と書きます。補助モデル *r は，*局所的な変分パラメータを持つ完全因数分解ガウスとして指定される 

### 4.3.3 確率的最適化

ここで、*θは*新たにvgpのカーネルハイパーパラメータと推論ネットワークのパラメータの両方を表し、*φは*補助モデルの推論ネットワークのパラメータを表す。ブラックボックス法に従って、勾配を期待値として記述し、変分モデルからサンプリングし、ノイジーな勾配を評価する確率近似(Robbins and Monro, 1951)を適用する。

最初に、確率的な勾配の分散を分析的に、任意の困難な期待値を導出することで、確率的な勾配の分散を減らします。

*q*(**z**|*f*(*ξ*))と*p*(**z**)の間のKL発散は，伝統的な変分自動エンコーダーの分散を減らすために一般的に用いられる：それは，深層潜在ガウスモデル(Rezende et al. , 2014)や深層リカレント注意書き(Gregor et al. , 2015)のような深層生成モデルに対して分析的である．*r*(*f* |*ξ,***z**)と*q*(*f* |*ξ*)の間のKL発散は，分布がともにガウス分布であるため分析的である．差 *logq*(*ξ*) - *logr*(*ξ* |**z**) は, 単にガウスの対数密度の差である.詳細は，付録 C.3 を参照のこと．

ブラックボックス勾配を導出するために，我々は最初にvgpを再パラメータ化し，サンプルのノイズ生成をその生成過程のパラメータから分離することができる(Kingma and Welling, 2014b; Rezende et al. , 2014)．gpは，簡単に再パラメータ化を可能にする：潜在入力*ξ* ∼ N(**0***,***I**)に対して，変換

**ｆ**（*ξ*；*θ*）＝**Ｌ***ξ*＋**Ｋ***ξ***ｓＫ**-*ｓｓ* [[6]](#footnote-6)*ｔｉは*、位置スケール変換であり、ここで、**ＬＬ** .

これはgpからのランダムな写像で*ξを*評価するのと等価です。平均場 *q*(**z**|*f*(*ξ*))もまた再パラメータ化可能であり, **z**(; **f**)がその出力**z** ∼ *q*(**z**|*f*(*ξ*))の関数であるような ∼ *w と*します.この2レベルの再パラメータ化は, 4.2.3節で概説した**zの**生成過程と等価である.

**アルゴリズム 1:** 変分ガウス過程を用いたブラックボックス推論

**入力.**モデル *p*(**x***,***z**), 平均場族 *Qi q*(*zi* |*fi*(*ξ*)).

**出力**。変動パラメータと補助パラメータ(*θ,φ*)。

(*θ,φ*)をランダムに初期化する。

**乍ら**

未収束

**遣る**

ノイズサンプルの描画

*ξ*

∼N

(

**0**

*,*

**I**

)

,

∼

*w*

.

変分サンプルのパラメータ化

**z**

=

**z**

(

;

*f*

(

*ξ*

))

,

*f*

(

*ξ*

)=

**f**

(

*ξ*

;

*θ*

)

.

更新情報

(

*θ*

*,*

*φ*

)

確率勾配付き

∇

*θ*

e

L

,

∇

*φ*

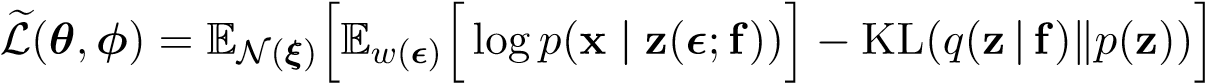
e

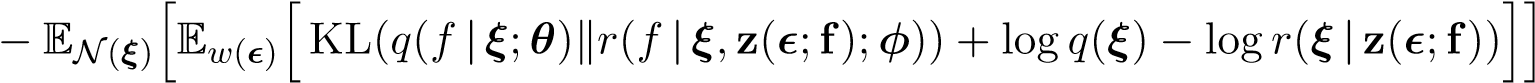
L

.

**打ち切り**

ここでは、変分目的を次のように書き換えます。

 (4.7)

*.*

式4.7は，勾配が期待値の中を移動し，入れ子になった再パラメータ化の上をバックプロパゲーションすることを可能にする。このようにして, 解析的KL項と再パラメータ化の両方によって低い分散を示す非偏りのない確率的勾配をとることができる.この勾配は，最初のKLが解析的に困難な場合を含めて，付録C.4で導出されている．

アルゴリズム 1 で手法の概要を説明する．大規模データについては、**x の**サブサンプリングを適用する (Hoffman et al. , 2013)。モデル対数尤度の勾配については、Stan や Theano (Carpenter et al. , 2015; Bergstra et al. , 2010) のような便利な微分ツールを採用する。非微分可能な潜在変数**z**、または効率的な再パラメータ化のない平均場分布については、内部期待度の勾配を取るために、Ranganathら(2014)からのブラックボックス勾配推定器を適用する。

### 4.3.4 計算 と記憶の複雑さ

ここで、*d は潜在*変数の数、*m は変*分データのサイズ、*L は*ニューラルネットワークの層数であり、*H は*平均隠れ層サイズである。特に、本アルゴリズムは、潜在変数数が線形であるため、他の変分推論法と競合する。変分パラメータと補助パラメータの数は、O(*c* + *LH*)の複雑さを持つ；この複雑さは、カーネルハイパーパラメータとニューラルネットワークパラメータの格納に由来する。

多くの gp 文献とは異なり、我々は、スケーラブルな計算のために誘導変数を使用するなどの低ランクの制約を必要としません (Quiñonero-Candela and Rasmussen, 2005)。変分データは同様の目的を果たしますが、誘導変数は（固定された）カーネル行列のランクを下げます。実際には必要ないことがわかりましたが，変分データのサイズのスケーリングについては，付録 C.5 を参照してください．

## 4.4 実験

深層学習における変分推論の標準的なベンチマークに従って、画像の生成モデルを学習する。特に、ニューラルネットワークアーキテクチャに従うガウスランダム変数の層状階層である深層潜在ガウスモデル(dlgm) (Rezende et al. , 2014)と、最近提案されたドロー(Gregor et al. , 2015)は、リカレントアーキテクチャと変分的自動エンコーダーのシーケンスを用いて複雑な画像を反復的に構築する潜在的注意モデルである(Kingma and Welling, 2014b)。

学習率については， RMSProp (Tieleman and Hinton, 2012) のバージョンを適用する．我々は，すべての実験で変分データのサイズを500に固定し，潜在入力次元を潜在変数の数と同じに設定する．

### 4.4.12 値化されたMNIST

二値化MNISTデータセット(Salakhutdinov and Murray, 2008)は、二値化された結果を持つ28×28ピクセルの画像から構成されている。dlgmの学習には、100個の確率変数と50個の確率変数からなる2つの確率層を適用し、各確率層の間にはtanh非線形性を用いた100単位の決定論的層を配置した。確率層には平均場ガウス分布とベルヌーイ尤度を適用する。確率的層が1層の場合と2層の場合のdlgmを学習するためにvgpを訓練する。

draw (Gregor et al. , 2015) では、我々はもともと

|  |  |  |
| --- | --- | --- |
| モデル | -ロッグ*p*(**x**) | ≤ |
| DLGM＋VAE[1] |  | 86.76 |
| DLGM＋HVI(閏8段階) [2] | 85.51 | 88.30 |
| DLGM + NF (*k* = 80) [3] |  | 85.10 |
| EoNADE-5 2hl(128オーダー) [4] | 84.68 |  |
| DBN 2hl [5] | 84.55 |  |
| DARN 1hl [6]  DLGM 2hl + IWAE (*k* = 50) [1] |  | 82.90 |
| ドロー[7] 80.97 | | |

**表4.1.**negativepredictivelog-likelihoodforbinarizedMNIST.Previousbestresultsは[1](Burda et al. , 2016a), [2] (Salimans et al. , 2015), [3] (Rezende and Mohamed, 2015), [4] (Raiko et al. , 2014), [5] (Murray and Salakhutdinov, 2009), [6] (Gregor et al. , 2014), [7] (Gregor et al. , 2015)であった。

は、パラメータの上に複雑な変量的優先順位を置くため、 vgpを用いて各時間ステップで潜在サンプルを生成します。エンコーディング・リカレント・ニューラル・ネットワークは、現在、（変分モデルに使用される）変分データと（補助モデルに使用される）平均場ガウシアン・パラメータを出力しています。Gregorら(2015)と同じアーキテクチャのハイパーパラメータを使用しています。

学習後、テストセットの対数尤度を評価しますが、これは真の値に対する下限値です。以下を参照してください。

表4.1は、様々な方法の*logp*(**x**)の近似値と下限値の両方を報告しています。vgpは、drawを使用した対数尤度で既知の最も高い結果を達成し、元々の最高値-80.97に対して**-79.88**の値を報告しています。vgpはまた、dlgmを用いた非構造利用モデルのクラスの中で、既知の最高の結果を達成し、値は-81.32で、Burdaら(2016a)によって報告された前回の最高値-82.90と比較しています。

### 4.4.2 スケッチ

|  |  |  |
| --- | --- | --- |
| モデル | エポック | ≦ -*logp*(**x**) |
| 描き出す | 100 | 526.8 |
|  | 200 | 479.1 |
|  | 300 | 464.5 |
| ドロー＋ＶＧＰ | 100  200 | **460.1** |

vgpが表現を学習するために複雑であることを示すために、我々はSketchデータセット（Eitz et al.このデータセットは、250のオブジェクトカテゴリに均等に分散した20,000の人間のスケッチから構成されています。これを18,000個の学習例と2,000個のテスト例に分割した。描画のアーキテクチャは、2x2の読み込みウィンドウ、5x5の書き込み注意ウィンドウ、64のグリンプスを持つように固定しています**。表4.2:** 負の予測対数尤度自動エンコーダー(下)。vgp はテクスチャを学習し、Sketch では数百エポックのシャープネスを学習し、より複雑な形状のスケッチができるようになりました。



対数尤度。推論には、元の自動エンコーダー版と vgp を用いた拡張版を使用します。

表 4.2 を参照してください。vgp を用いた描画では、多くのコンピュータビジョンタスクで最先端の成功を収めているオリジナルバージョンよりも有意に優れた下界を達成しています。(ここで紹介する結果が出るまでは、このデータセットではオリジナルのdrawの結果が最高のパフォーマンスとして報告されていました)。さらに、vgpを用いて推論したモデルは、オリジナルバージョンよりも複雑な画像を生成することができ、性能が向上するだけでなく、より高い視覚的忠実度を維持しています。

## 4.5 ディスカッション

我々は、複雑な事後分布と一致するようにその形状を適応させる変分モデルである変分ガウス過程(vgp)を提示する。vgpは、作業可能な分布からサンプルを抽出し、作業可能な分布から平均場パラメータへの変換に対するベイズ的ノンパラメトリックな事前処理を仮定します。vgpは、すべての連続写像の空間から変換を学習します。これは普遍的な近似器であり、最適化によって良い事後近似を見つけます。

# 第5章：深層確率モデルへの応用

## 5.1 イントロダクション

コイントスのモデルを考えてみよう。確率論的モデルでは、通常、潜在確率を仮定し、各トスがこの確率が与えられたベルヌーイの結果であると仮定します(Murphy, 2012; Gelman et al. , 2013)。コイントスの集合を観察した後、ベイズ分析では、確率に関する推論を記述することができます。

しかし、コイントスの結果は、その初期条件（例えば、インパルスとフリップの角度）によって完全に決定されることが物理学の法則からわかっている（Keller, 1986; Diaconis et al.したがって、コイントスのランダム性は、潜在確率ではなく、ノイズの多い初期パラメータに由来する。この代替モデルは物理システムを組み込み、生成過程をより良く捉えている。さらに、このモデルはシミュレーターとしても知られている暗*黙的な*ものである：生成過程からデータをサンプリングすることはできますが、その密度を計算することはできないかもしれません（Diggle and Gratton, 1984; Hartig et al.

コイントスは単純ですが、複雑な暗黙のモデルの構成要素として機能します。これらのモデルは、実世界の物理システムの法則や理論を捉えたものであり、集団遺伝学（Pritchard et al.

残念ながら、ガンを含む暗黙のモデルは、特定のドメイン以外での成功は限られています。これには2つの理由があります。第一に、より一般的なアプリケーションのために、プリオール、階層、シーケンスなどの豊富な潜在構造を露出させた暗黙モデルをどのように設計するかが不明である。第二に、既存の暗黙モデルの潜在構造を推論する手法は、高次元や大規模なデータセットに十分にスケーリングできないことである。本論文では、新しいクラスの暗黙モデルを設計し、正確でスケーラブルな推論のための新しいアルゴリズムを開発する。

モデル化については、5.2節では、サンプルを生成するプロセスのみを仮定するベイズ階層モデルのクラスである階層*的暗黙モデルについて*述べています。このクラスは、古典的な文献にあるシミュレータとガンで採用されているシミュレータの両方を網羅しています。例えば、我々はベイズ的なガンを指定し、そのパラメータに優先順位を置きます。ベイズの観点から、ガンは不確実性を定量化し、データ効率を向上させることができます。また、離散データにも適用することができます。この設定は、従来のガンのための推定アルゴリズムでは不可能です(Kusner and Hernández-Lobato, 2016)。

推論については、第5.3節では、*尤度自由分散推論(*lfvi*)を*開発しており、これは分散推論と密度比推定を組み合わせたものである(Sugiyama et al. , 2012; Mohamed and Lakshminarayanan, 2016)。変分推論は、潜在変数上の分布のファミリーを仮定し、事後に最も近いメンバーを見つけるために最適化する(Jordan et al. , 1999b)。従来のアプローチでは、尤度ベースのモデルを必要とし、高速な計算のために単純な近似族を使用して粗い近似を使用していました。 lfviは変分推論を暗黙のモデルに拡張し、暗黙の変分族を使用して正確な変分近似を可能にします。さらに、これまでの暗黙モデルのためのベイズ法とは異なり、確率的最適化により、LFVIは何百万ものデータポイントにスケールします。

本研究は、多様な応用が可能である。まず、古典的な問題を近似的な

ベイズ計算(ABC)の文献では，モデルが生態系をシミュレートしている(Beaumont, 2010)．従来の手法では不可能であった10万個の時系列を解析する。第二に、我々は、重みに対する優先順位を持つベイジアンギャンを分析する。ベイジアンギャンは、いくつかの分類タスクにおいて、既知の尤度を持つ対応するベイジアンニューラルネットワークよりも優れた性能を示す。第三に、我々は、リカレント・ニューラル・ネットワークの隠れユニットにノイズを注入することが、どのようにしてディープな

柔軟なシーケンス生成のための暗黙のモデル

## 5.2 階層的暗黙モデル

階層モデルは、事例間の統計的強度を共有する上で重要な役割を果たす(Gelman and Hill, 2006)。階層ベイズモデルの広いクラスでは，隠れた*β βの*共同分布

**x**

*n*

**z**

*n*

*N*

**x**

*n*

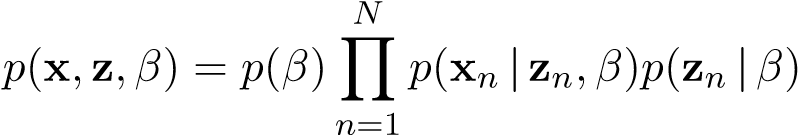
**z**

*n*

*n*

*N*

**図5.1**: (**左**)局所変数**z**と大域変数*βの階層モデル* (**右**)階層**的暗黙モデル**.これは階層モデルであり, **x は**決定論的関数(四角で示す)のノイズ(三角で示す)であり, 観測された変数は

*,* (5.1)

ここで *xn は*オブザベーション，*zn は*そのオブザベーションに関連する潜在変数（ローカル変数），*β は*オブザベーション間で共有される潜在変数（グローバル変数）である．図5.1（左）を参照してください。

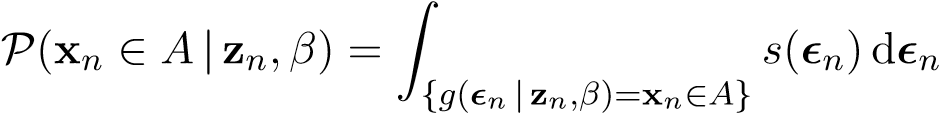
階層モデルでは、局所変数は、混合モデルでのクラスタリング、トピックモデルでの混合メンバーシップ(Blei et al. , 2003)、確率的行列因数分解(Salakhutdinov and Mnih, 2008)での因子に使用することができる。グローバル変数は、階層回帰（Gelman and Hill, 2006）、トピックモデル（Blei et al. , 2003）、およびベイズノンパラメトリック（Teh and Jordan, 2010）のために、データポイント間の情報をプールするために使用できる。

階層モデルは、一般的に扱いやすい尤度 *p*(*xn* |*zn,β*) を使用する。しかし、シミュレータベースのモデル(Hartig et al. , 2011)や生成的敵対的ネットワーク(Goodfellow et al. , 2014)のような多くの尤度は、真のデータ生成プロセスへの高い忠実度を認めており、作業可能な尤度を認めていない。この限界を克服するために、我々は*階層的暗黙*モデル*(*Hims*)を*開発する。

階層的暗黙モデルは、式5.1と同じ共同因数分解を持っていますが、尤度からサンプリングできると仮定しています。*p*(*xn* |*zn,β*)を明示的に定義するのではなく、ランダムなノイズ*n* ∼ *s*(-)を取り込み、*zn*と*βが*与えられた*xnを*出力する関数*gを*定義します。

**x**

与えられた*zn*と*βの***x***n* ∈*A*の誘導的な暗黙の尤度は

*.*

この積分は一般的に難解です。積分する集合を見つけるのが難しく、任意のノイズ分布*s*(-)や関数*g*に対しては、積分自体が高価になることがあります。

図5.1 (右図) は, hims のグラフィカルモデルを示している.ノイズ(*n*)は三角形, 決定論的計算(*xn*)は四角である.ここでは2つの例を示します.

**例。物理シミュレータ。**シミュレータは、初期条件が与えられると、データを生成する確率的なプロセスを記述します。例えば、個体群生態学では、Lotka-Volterraモデルは、確率的微分方程式を用いて、時間経過に伴う捕食者-被捕食者の個体群をシミュレートします(Wilkinson, 2011)。獲物と捕食者の個体群x1*,x*2∈R+に対して、1つのプロセスは

ディーエックスワン

*,*10*)。*

ｄｔｄｘ２

*,*10*)。*

ｄ*ｔ*

ここでは、ガウスノイズが各時間ステップごとに追加されます。シミュレータは、x1*,x*2の初期母集団サイズを与えられた*T*回の時間ステップで実行されます。Lotka-Volterraモデルは理論に基づいていますが、難解な尤度を特徴としています。我々は第5.4節でこれを研究する。

**例.ベイズ的生成的逆襲ネットワーク。** 生成的逆襲ネットワーク(gans)は、暗黙のモデルとパラメータ推定のための方法を定義する(Goodfellow et al. , 2014)。これらのネットワークは画像生成において優れた性能を発揮することが知られている(Radford et al. , 2016)。形式的には， ganの暗黙モデルは次のようになります．

**x**(5.2)

ここで、*gは*パラメータ*θ*を持つニューラルネットワークであり、*sは*標準的な正規値または一様値である。ニューラルネットワーク*gは*一般的には反転可能ではなく、これは尤度を困難なものにします。

ギャンのパラメータ*θは、*生成されたデータと実データの発散最小化によって推定される。我々は、パラメータ*θ*に優先順位をつけることで、ベイズ分析が可能な状態にします。ベイジアンギャンは、パラメータの不確実性のモデリングを可能にし、標準的なニューラルネットワークの不確実性とデータ効率を改善することが示されているベイジアンニューラルネットワークに触発されたものである(MacKay, 1992; Neal, 1994)。我々は5.4節でベイジアン・ギャンを研究する。

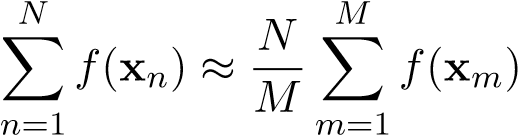
付録Bは、確率的プログラミング言語Edward(Tran et al. , 2016a)における実装例を提供する。

## 5.3尤 度自由変量推論

我々は、潜在変数モデルの豊富なクラスである階層的暗黙モデルを説明した。与えられたデータが与えられたとき、我々はモデルの事後的な *p*(**z***,β* |**x**) = *p*(**x***,***z***,β)/p*(**x**) を計算することを目的としている。正規化定数*p*(**x**)が一般的に難解であるため，これは困難である．暗黙モデルでは、尤度関数がないことが、難解さの追加要因となる。

我々は変分推論を用いる (Jordan et al. , 1999b)。これは、近似家族*q*∈Qを仮定し、*p*(**z***,β* |**x**)に最も近いメンバーを見つけるように最適化する。近さを測定する変分目的語には多くの選択肢がある(Ranganath et al. , 2016a; Li and Turner, 2016; Dieng et al. , 2016)。目的語を選択するために、我々は暗黙モデルのための変分的推論アルゴリズムのためのデザイダータをレイアウトする。

1. *スケーラビリティ*。機械学習は，大規模データにスケールするために確率的最適化に依存している(Bottou, 2010)．変分目的語は，標準的な手法を用いた偏りのないサブサンプリングを認めるべきである．

*,*

ここで，完全なデータに対するいくつかの計算*f*(-)は，データのミニバッチ{*xm*}で近似されます．

1. 暗黙*の局所近似*。暗黙モデルは柔軟な密度を指定しますが、これは非常に複雑な事後分布を引き起こします。したがって、データ点ごとの近似*q*(*zn* |*xn,β*)のための豊富な近似ファミリーが欲しいと思います。これは、変分目的は、*zn* ∼ *q*(*zn* |*xn,β*)をサンプリングすることができることだけを必要とし、その密度を評価しないことを意味します。

我々の目的を満たす変分目的は、古典的な kl ダイバージェンスの最小化に基づいています。(驚くべきことに、付録Cでは、広いクラスの中で kl が*唯一*可能な目的であることを詳細に説明しています)。

### 5.3.1 KL変動目的語

古典的な変分推論では、変分近似*q*から事後処理への発散量klを最小化する。これは、エルボを最大化することと等価である。

L = *Eq*(*β,***z**|**x**)[*logp*(**x***,***z***,β*) - *logq*(*β,***z**|**x***)]。* (5.3)

*qを*事後処理と同じように因数分解してみましょう。

*N*

*q*(β*,***z**|**x**) = *q*(*β*) Y *q*(*zn* |*xn,β*)。

n=1

ここで *q*(*zn* |*xn,β*) は難解な密度であり、推論中のデータ **x は**一定であるので、大域的な *q*(*β*) の条件付けは不要です。*p*と*q*の因数分解を代入すると、次のようになります。

*N*

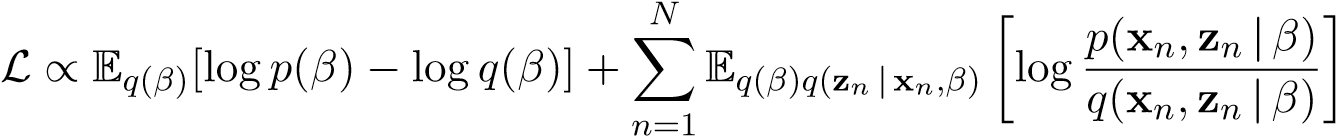
Ｌ＝*Ｅｑ*（*β*）［*ｌｏｇｐ*（*β）*-*ｌｏｇｑ*（*β）*］＋*ＸＥｑ*（*β*）*ｑ*（*ｚｎ*｜*ｘｎ，β*）［*ｌｏｇｐ*（*ｘｎ，ｚｎ*｜*β）*-*ｌｏｇｑ*（*ｚｎ*｜*ｘｎ，β*）］*。*

n=1

この目的は困難である：局所密度*p*(*xn,zn* |*β*)と*q*(*zn* |*xn,β*)の両方が困難である。これを解決するために，我々は比推定を考える．

### 5.3.2 KL目的語の比率推定

*q*(*xn*) をオブザベーション **x** での経験分布とし、"変分ジョイント" *q*(*xn,zn* |*β*) = *q*(*xn*)*q*(*zn* |*xn,β*) でそれを使用することを考えてみましょう。ここで、上記のエルボから経験的な対数*logq*(*xn*) を引きます。エルボは次のようになります。

*.* (5.4)

(ここで比例の記号は加法定数までの平等を意味する) したがって，エルボは2つの難解な密度の比の関数である．この比率の推定値を形成できれば，エルボの最適化を進めることができる．

我々は比推定のための技術を適用する(Sugiyama et al. , 2012)。これはガンにおける重要な考え方であり(Mohamed and Lakshminarayanan, 2016; Uehara et al. , 2016)、統計学や物理学においても同様の考え方が再浮上している(Gutmann et al. , 2014; Cranmer et al. , 2015)。特に、我々はクラス確率推定を使用します：*p*(-)または*q*(-)からのサンプルが与えられた場合、それが*p*(-)に属する確率を推定することを目的とします。ここで、*rは*サンプルの入力を受けて実値を出力するパラメータ化された関数（例えば、ニューラルネットワーク）であり、*σは確率を出力する*ロジスティック関数である。

適切なスコアリングルールとして知られる損失関数を最小化することによって*r*(-;*θ*)を訓練する(Gneiting and Raftery.

2007).例えば、実験では対数損失を利用しています。

Dlog = *Ep*(*xn,zn* |*β*)[-log*σ*(*r*(*xn,zn,β*;*θ*))]+*Eq*(*xn,zn* |*β*)[-log(1-*σ*(*r*(*xn,zn,β*;*θ*)))]と*する。*(5.5)

*σ*(*r*(-; *θ*))が*p*(-)からのサンプルであれば1を返し、*q*(-)からのサンプルであれば0を返す場合、損失はゼロです。(我々はヒンジ損失についても実験を行います; セクション5.4を参照してください) *r*(-; *θ*)が十分に表現力がある場合、損失を最小化すると最適な関数が得られます(Mohamed and Lakshminarayanan, 2016)。

*r*∗(*xn,zn,β*) = *logp*(*xn,zn* |*β*) - *logq*(*xn,zn* |*β)。*

式 5.5 を最小化するので、式 5.4 の対数比の代理として *r*(-; *θ*) を使用します。r *は対数比*を推定していることに注意してください。

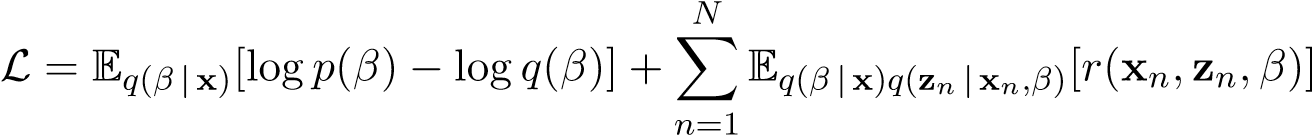
*θ*に対するDlogの勾配は

*Ep*(*xn,zn* |*β*)[∇*θ* log*σ*(*r*(*xn,zn,β*;*θ*)] + *Eq*(*xn,zn* |*β*)[∇*θ* log(1 - *σ*(*r*(*xn,zn,β*;*θ))]。* (5.6)

モンテカルロ法を用いて偏りのない勾配を計算する。

### 5.3.3 KL目的語の確率的勾配

エルボを最適化するために、比率推定器を使用します。

*.* (5.7)

これですべての項が扱いやすくなりました。変分族 *q を*最適化するために勾配を計算することができます。以下では，prisor *p*(*β),p*(*zn* |*β*) が微分可能であることを仮定しています．(次節で離散大域変数を扱う方法について説明します)

我々は、再パラメータ化可能な変分近似(Kingma and Welling, 2014b; Rezende et al. , 2014)に注目する。それらはランダムノイズの微分可能な*変換Tを*介したサンプリングを可能にします。式5.7により、我々は大域近似*q*(*β*; *λ*)がトラクタブルな密度を認めることを要求する。再パラメータ化を行うと, その標本は

*β* = Tglobal(*δ*global; *λ)、δ*global ∼ *s(-)。*

変換Tglobal(-; *λ*)とノイズ*s*(-)の選択に対して.例えば、*s*(-) = N(0*,*1) とし、Tglobal(*δ*global) = µ + *σδ*global とすると、正規分布 N(*μ,σ*2) が誘導されます。

同様に、ローカル変数 *zn* についても、次のように指定します。

*zn* = Tlocal(*δn,xn,β*;*φ),δn* ∼ *s(-).*

大域近似とは異なり、局所的な変分密度*q*(*zn* |*xn*; *φ*)は扱いやすいものである必要はない。比推定器はこの要件を緩和する。これにより、データだけでなく近似事後処理についても暗黙のモデルを活用することができる。実際には、推論ネットワークを用いて計算量を償却し、データ毎の点近似事後処理でパラメータ*φを*共有する。

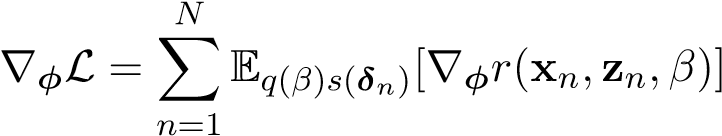
この近似ファミリーの下でのグローバルパラメータ*λ*に対する勾配は

*N*

∇*λ*L = *Es*(*δ*global)[∇*λ*(*logp*(*β*)-*logq*(*β*)]+ *XEs*(*δ*global)*sn*(*δn*)[∇*λr*(*xn,zn,β*)]となります*。* (5.8)

n=1

勾配は局所サンプリング*zn* = Tlocal(*δn,xn,β*; *φ*)と大域再パラメータ化*β* = Tglobal(*δ*global; *λ*)を経てバックプロパゲーションする.モンテカルロ法を用いて偏りのない勾配を計算する。局所パラメータ*φ*に対する勾配は

*.* (5.9)

アル**ゴリズム 2:** 無尤度変分推論 (LFVI)

**入力 :**モデル *xn,zn* ∼ *p*(-|*β*), *p*(*β*)

変分近似 *zn* ∼ *q*(-|*xn,β*;*φ*), *q*(*β* |**x**;*λ*), 比推定器 *r*(-;*θ*) **出力** 変分パラメータ *λ*,*φ* Initialize**.** *θ*, *λ*, *φを*ランダムに初期化する。

∇*θ*D（式5.6）、∇*λ*L（式5.8）、∇*φ*L（式5.9）の偏りのない推定値を計算する。

確率的勾配降下を用いて*θ*，*λ*，*φを*更新する。

ここで、勾配はTlocalを通ってバックプロパゲーションします。 [[7]](#footnote-7)

### 5.3.4 アルゴリズム

アルゴリズム2は、手順の概要を示しています。lfvi はブラックボックスであり、データと局所変数をシミュレートし、グローバル変数の密度を計算できるモデルに適用されます。

このアルゴリズムは、Edward (Tran et al. , 2016a)に掲載されている。

lfvi はスケーラブルである: ミニバッチを用いて全データセットに対する勾配を偏りなく推定できる (Hoffman et al. , 2013)。このアルゴリズムは，連続または離散データのモデルも扱うことができる．微分可能な大域変数と再パラメータ化可能な大域近似の要件は，スコア関数勾配を用いて緩和することができる (Ranganath et al. , 2014)．

グローバルパラメータ*βの*点推定値は、多くのアプリケーションで十分である(Goodfellow et al. , 2014; Rezende et al. , 2014)。アルゴリズム2は点推定値を見つけることができます：パラメータ*β*上に点質量近似*qを*配置します。これは勾配を単純化し、変分EMに対応します。

10 101 VI VI ログヒンジ

拒否

エービーシー

エムシーエムシー

エービーシー

ＳＭＣ

エービーシー

六

ログ

六

ヒンジ

−

5

*.*

5

−

5

*.*

0

−

4

*.*

5

−

4

*.*

0

−

3

*.*

5

−

3

*.*

0

−

2

*.*

5

ログ

*β*

1

真の値

0

−

5

0

5

10

15

ログプロビティヨントルーパラメタ

リジュ・エービーシー

MCMC-ABC

SMC-ABC

拒否

エービーシー

エムシーエムシー

エービーシー

ＳＭＣ

エービーシー

六

ログ

六

ヒンジ

−

2

*.*

0

−

1

*.*

5

−

1

*.*

0

−

0

*.*

5

0

*.*

0

0

*.*

5

1

*.*

0

1

*.*

5

ログ

*β*

2

真の値

−

5

*.*

5

−

5

*.*

0

−

4

*.*

5

−

4

*.*

0

−

3

*.*

5

−

3

*.*

0

−

2

*.*

5

ログ

*β*

1

真の値

**図5.2: (上)** 最初の2つのパラメータの限界事後処理.(**bot. 左)** 許容誤差を超えるABC法.(bot. **右)** 大規模データセットにおける最初のパラメータの限界事後処理。我々の推論では，より正確な結果が得られ，大規模データへの拡張が可能である．

## 5.4 実験

新しいモデルと推論を開発した。実験では、生態学における捕食者-被捕食者間の個体群のための大規模物理シミュレータ、教師付き分類のためのベイズガン、シンボル生成のための深層暗黙モデルの3つの応用を研究している。また、付録Fでは、おもちゃの実験を解析して比推定器の安定性に対処するための実践的なアドバイスを提供している。

標準法線からパラメータを初期化し、ADAMを用いて勾配降下を適用します。

**Lotka-Volterra Predator-Prey シミュレータ。**我々は5.2節のLotka-Volterraシミュレータを解析し、Papamakarios and Murray (2016)と同じセットアップとハイパーパラメータに従います。そのグローバル変数*βは、*捕食者-被捕食者-被捕食者の個体群のシミュレーションにおける変化率を支配します。それらを推論するために、我々は平均場法線近似（同じサポート上にあるように再パラメータ化）を仮定し、比率推定問題のために対数損失とヒンジ損失の両方でアルゴリズム2を実行します; 付録Dは、ヒンジ損失の詳細を説明します。我々は、棄却ABC、MCMC-ABC、およびSMC-ABC (Marin et al. , 2012)と比較する。MCMC-ABCは球面ガウス提案を用いており、SMC-ABCは減衰εスケジュールを用いて手動で調整されています；すべてのABC手法は、許容誤差のような最高性能のハイパーパラメータを用いるように調整されています。

図5.2は2つのデータセットでの結果を示している。上の図と左下の図では、データを解析しています。

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | テストセットエラー |  |
| モデル＋推論 | カニ | ピマコーバータイプ | エムエヌアイエスティー |
| ベイズGAN + VI | 0.03 | **0. 2320.154** | **0.0136** |
| ベイズGAN + MAP | 0.12 | 0. 2400.185 | 0.0283 |
| ベイズ型NN+VI | **0.02** | 0. 2420.164 | 0.0311 |
| ベイジアンNN + MAP | 0.05 | 0. 3200.188 | 0.0623 |

**表5.1:** 小規模から中規模のデータセットにおけるベイジアンギャンとベイジアンニューラルネットワークの分類精度。ベイジアンギャンはベイジアンニューラルネットと同等かそれ以上の性能を達成しています。左下の図は、abc法の許容誤差に対する真のパラメータの負の対数確率を計算したもので、許容誤差が小さいほど精度は高くなりますが、実行時間は遅くなります。上の図は、abc法の最小許容誤差を用いた2つのパラメータの限界事後分布を比較したものです。棄却ABC、MCMC-ABC、SMC-ABCはすべて95%信頼区間内に真のパラメータを含んでいますが、我々の手法よりも信頼性が低いです。さらに、これらの手法はモデルから100*,*000回のシミュレーションを必要とし、棄却ABCとMCMC-ABCではそれぞれ0*.*004%と2*.*990%の許容率であった。

右下の図は、前の図で分析した単一の時系列と同じ大きさの10万*個の*時系列からなるデータを分析したものです。これは従来の手法では不可能なサイズです。また、我々の手法では、事後処理が真実に近いところに集中していることがわかります。また、比推定に対数損失を用いた場合とヒンジ損失を用いた場合の精度の差はほとんどありませんでした。

**ベイズ的生成的敵対的ネットワーク。**5.2節で述べたベイズガンを分析する。

ベイズニューラルネットワークのユースケース(Blundell et al. , 2015; Hernández-Lobato et al. , 2016)を模倣して、我々は小・中規模データの分類にベイズガンを適用する。ganは、特徴*xn*∈*RDを*入力として取り、プロセスを介してラベル*yn*∈{1*,...,K*}を生成する条件*p*(*yn* |*xn*)を定義します。

*, n* ∼ N(0*,*1*).* (5.10)

ここで、*g*(-|*θ*)は、ReLU活性化、バッチ正規化、重みとバイアス*θ*でパラメータ化された2層多層知覚である。 我々は、正規のプリアウト、*θ* ∼ N(0*,*1)を配置する。

変分モデルの2つの選択肢を分析します：1つは*q*(*θ* |**x**)の平均場正規近似、もう1つは点群近似（最大事後処理に相当）です。我々は、式5.10と同じ生成プロセスを使用しますが、ニューラルネットにノイズを与えるのではなく、カテゴリカル分布から描画するベイズ型ニューラルネットワークと比較します。我々は、平均場正規近似と最大a posterioriを使用して個別に適合させます。表5.1は、ベイジアンギャンが一般的にベイジアンニューラルネットの対応するものよりも優れていることを示しています。

ベイジアンガンは、分類ラベルの生成など離散データを分析できることに注意してください。離散データのための伝統的なガンは未解決の課題である(Kusner and Hernández-Lobato, 2016)。付録Eでは、点推定を用いたベイジアンガンを典型的なガンと比較しています。ベイジアンガンは、これらの小・中規模データセットを分析するために、パラメータの不確実性を活用することもできます。

ベイズガンの問題点の1つは、非常に大規模なニューラルネットワークでは動作しないことである。1つのアプローチは、比率推定器をグローバル・パラメータの関数ではないものにすることである。変分EMによってモデルパラメータを最適化する代わりに、変分目的の代わりに比目的をバックプロパゲ ートすることでモデルパラメータを訓練することができる。代替案は、はるかに低次元である入力として隠れユニットを使用することである(Tran and Blei, 2017, Appendix C)。

**隠れたユニットにノイズを注入するこの**セクションでは、隠れたユニットにランダム性を注入するだけで、階層的な暗黙モデルを構築する方法を示します。我々は、シーケンス **x** = (x1*,...,xT* ) をリカレント・ニューラル・ネットワークでモデル化します。*t* = 1*,...,T の*場合。

*,*

*, t,x* ∼ N(0*,*1*).*

ここで、*gz*および*gxは*、ReLU活性化および層の正規化を伴う1層多層知覚である。

我々は、すべての重みとバイアスに標準的な正規のプリアスを置く。図5.3aを参照してください。

射影されたノイ*セット,zがGZ*の出力と直線的に結合した場合、誘導分布*p*(*zt* |*xt*-1*,zt*-1)は、その出力によってパラメータ化されたガウス分布である。これは確率的なrnnを定義し(Bayer and Osendorfer, 2014; Fraccaro et al. , 2016)、決定論的な接続を一般化したものである。非線形な組み合わせでは、暗黙密度はより柔軟性があり（難解である）、これまでの推論手法は適用できない。我々の方法では、変分推論を行い、暗黙の密度として*qを*指定する。

------······ -xx/+x\*xx/+xx\*\*\*xx/\*//x+xx\*+xx++

**x**

*t*

−

1

**x**

*t*

**x**

*t*

+1

**z**

*t*

−

1

**z**

*t*

**z**

*t*

+1

/+x\*x+x\*x/x/x+x+ /x+\*x+x\*x/x+x-x+ x/x\*x/x\*x+x+x+xx+x+x/x\*x\*x+x/x+

**(b) 暗黙**モデルから生成された記号。良いサンプルは変数*xの*間に算術演算子を配置する。 **(a)**シーケンスのための深い暗黙モデルからルールに従うように学習した暗黙モデル。これは、各隠れた状態にノイズが注入されたいくつかの倍数演算子までの文脈自由文法である。隠れた状態が繰り返される。

状態は暗黙の潜在変数になりました。出力を生成する場合も同様です。

確率モデルの暗黙のプリオールと同じアーキテクチャです。

我々は、Kusner and Hernández-Lobato (2016)と同じセットアップとハイパーパラメータに従い、文脈自由文法に従った単純な一変数の算術シーケンスを生成する。

*S* → *xkS* + *SkS* - *SkS* ∗ Sk*S/S.*

ここで、k は文法の可能な生成物を分割します。入力を連結し、変分EMを用いて大域変数（モデルパラメータ）を点推定します。図 5.3b は，最大 15 個のシンボルを持つシーケンスで学習した推定モデルのサンプルを示しています．これにより、文脈自由文法にほぼ沿ったシーケンスが得られています。

## 5.5 ディスカッション

我々は、階層的な暗黙モデルと尤度フリーの変分推論のクラスを開発し、暗黙密度の考え方を階層的ベイズモデリングと近似事後推論と融合させた。これにより、ニューラルサンプラー、物理シミュレータ、およびそれらの組み合わせと豊かで解釈可能な潜在構造を適用する能力を備えたベイズ分析が拡張される。

比推定を用いたより安定した推論は未解決の課題である。これは、暗黙モデルの大規模な実世界応用を解析する場合に特に重要である。ゲノムのための最近の研究は、有望な解決策を提供しています(Tran and Blei, 2017)。

**第5章 書誌**

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G.

このような研究は、機械学習のための大規模機械学習システムの開発を目的としたものである。TensorFlow: 大規模機械学習のためのシステム. *arXiv.org*.

また、そのような場合には、そのような計算方法ではなく、そのような計算方法を用いて計算を行うことができます。補助変分法。ニューラル*情報処理*, 561-566ページ.シュプリンガー。

Amos, B. and Kolter, J. Z. (2017).OptNet.ニューラルネットワークにおける層としての微分可能な最適化．

*機械学習に関する国際会議*にて.

また、このような計算機を用いた効率的なモンテカルロ計算を実現するためには、計算機を用いて計算を行う必要があります。疑似限界アプローチによる効率的なモンテカルロ計算.*統計学の歴史*, 697-725 ページ.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. (2016).勾配降下による勾配降下による学習．ニュー*ラル情報処理システム』*にて。

Anelli, G. Antchev, G. Aspell, P. Avati, V. Bagliesi, M. Berardi, V. Berretti, M. Boccone, V. Bottigli, U. Bozzo, M. et al. (2008).CERN大型ハドロン衝突型加速器でのトーテム実験。

計*装学会誌*、3(08):S08007。

Bahdanau, D., Cho, K., and Bengio, Y. (2015).整列と翻訳の共同学習によるニューラル機械翻訳.In *International Conference on Learning Representations*.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2015).機械学習における自動微分：調査. *arXiv preprint arXiv:1502.05767*.

Bayer, J. and Osendorfer, C. (2014). 確率的リカレントネットワークの学習. *arXiv プレプリント arXiv:1411.7610*.

ベイラー、D、ブレック、E、チェン、H.-T.、フィーデル、N.、フー、C.Y.、ハーク、Z.、ヘイカル、S.、イスピール、M.、ジェイン、V.、コック、L.、他（2017）。TFX.TensorFlowベースの生産規模機械学習プラットフォーム。In *Knowledge Discovery and Data Mining*.

ボーモント，M.A. (2010).進化と生態学における近似ベイズ計算.生態*学・進化学・系統学の年次レビュー*, 41(379-406):1.

ベッカー、R.A.とチェンバース、J.M. (1984).*S: データ分析とグラフィックスのためのインタラクティブな環境*。CRCプレス。

このような研究は、ニューラルネットワークにおける条件付き計算を可能にするために必要不可欠である。より高速なモデルのためのニューラルネットワークにおける条件付き計算. *arXiv preprint arXiv:1511.06297*.

ベンジオ、Y. 、クールヴィル、A. 、ヴィンセント、P. (2013).表現学習.レビューと新しい

パースペクティブ。*IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798-1828.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde.

Farley, D., and Bengio, Y. (2010).Theano: CPUとGPUの数学式コンパイラ.*Python for Scientific Computing Conference (SciPy*).

確率的ネットワークにおける空間効率の良い推論のための研究は、そのような研究者の間でも大きな話題となっている。動的確率ネットワークにおける空間効率の良い推論．このように、本研究では、*人工知能に関する国際合同会議を開催して*います。

ビンガム、E、チェン、J. P.、ジャンコウィアク、M.、オバーマイヤー、F.、プラダン、N.、カラレツォス、T.、シン、R.、スゼルリップ、P.、ホーズフォール、P.、およびグッドマン、N. D. (2018)。Pyro: Deep Universal Probabilistic Programming. *arXiv preprint arXiv:1810.09538*.

ビショップ、C. (2006).*パターン認識と機械学習*.シュプリンガー.

このような場合には、「混合物を用いて信念ネットワークの事後分布を近似する」ことが重要であると考えられます。混合物を用いた信念ネットワークにおける事後分布の近似。このように、本研究では、このような問題を解決*するための研究を行っています*。

ブレイ、D.M. (2014).ビルド、計算、批評、リピート：潜在変数モデルを用いたデータ解析.

*統計学とその応用に関する年鑑*。

潜在ディリクレの割り当て.*機械学習研究*、3:993-1022.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015).ニューラルネットワークにおける重みの不確実性．*機械学習国際会議*にて。

ボットー，L. (2010).確率的勾配降下を用いた大規模機械学習.*COMPSTAT'2010 の講演論文集*, 177-186 ページ.シュプリンガー.

Box, G. E. (1980).科学的モデリングとロバストネスにおけるサンプリングとベイズの推論．*王立統計学会誌.シリーズ A (一般*), 383-430 ページ.

ボックス、G.E. (1982).統計学におけるエキュメニズムのための謝罪。技術報告書、DTICドキュメント。

ボックス、G.E.P. (1976). 科学と統計. *アメリカ統計学会誌*, 71(356):791-799.

このような場合には、「ストリーミング変分ベイズ」を用いて、「ストリーミング変分ベイズ」と「ストリーミング変分ベイズ」の両方を行うことができます。ストリーミング変分ベイズ。ニュー*ラル情報処理システム*。

Buchanan, B., Sutherland, G., and Feigenbaum, E. A. (1969).*ヒューリスティックDENDRAL: 有機化学における説明的仮説生成プログラム*。アメリカのエルゼビア社。

Burda, Y., Grosse, R., and Salakhutdinov, R. (2016a).重要度重み付きオートエンコーダー．*表現の学習に関する国際会議で*．

Burda, Y., Grosse, R. and Salakhutdinov, R. (2016b).重要度重み付きオートエンコーダー．*表現の学習に関する国際会議で*．

カーペンター、B.ゲルマン、A.ホフマン、M.D.、リー、D.グッドリッチ、B.ベタンコート、M.ブルベーカー、M.

Guo, J., Li, P. and Riddell, A. (2016).スタン：確率的プログラミング言語．*Journal of Statistical Software*.

Carpenter, B., Hoffman, M. D., Brubaker, M., Lee, D., Li, P. and Betancourt, M. (2015).スタン数学ライブラリ.C++での逆モード自動微分. *arXiv.org*.

Chambers, J. M. and Hastie, T. J. (1992).*統計的モデルの研究』S.* チャップマン＆ホール，ロンドン．

Chollet, F. (2015).Keras. https://github.com/fchollet/keras[。](https://github.com/fchollet/keras)

尤度比の近似を行うためには、以下のようなことが必要である。校正された識別分類器を用いた尤度比の近似. *arXiv preprint arXiv:1506.02169*.

Culler, D. E. (1986).データフローアーキテクチャ。技術報告書、DTICドキュメント。

点プロセス強度推定のための高速ガウスプロセス法.このような研究は、研究*者の研究成果を反映したもの*である。ACM.

Cusumano-Towner, M. F. and Mansinghka, V. K. (2018).確率的プログラムをプロポーザルとして利用する．

*POPLワークショップ*にて。

ダヤン、P.とアボット、L. F. (2001).*理論的神経科学*、第10巻。ケンブリッジ、マサチューセッツ州。MIT

プレス。

ダヤン、P. ヒントン、G. E. ニール、R. M. およびゼメル、R. S. (1995)。ヘルムホルツマシン.*神経計算*, 7(5):889-904.

デントン、E. L.、チンタラ、S.、ファーガス、R.ら（2015）．ラプラシアンピラミッドを用いた敵対的ネットワークを用いたディープジェネレーティブ画像モデル.ニューラル*情報処理システム*において．

また、このような研究では、研究者の研究者は、研究者の研究者の研究者の研究者の研究者の研究者の研究者の研究者の研究者の研究者の研究者である。コイントスにおける動的バイアス.*SIAM*, 49(2):211-235.

*このように*、本研究では、このような問題を解決するためには、「近似推論のための*χ*-発散」が必要であると考えられる。*arXiv preprint arXiv:1611.00328*にて。

暗黙の統計モデルのための推論のモンテカルロ法.*英国王立統計学会誌.このよう*に，本研究では，このような研究者の研究活動を支援することを目的*としています。*

Dillon, J., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M. and Saurous, R. (2017).TensorFlowの分布。

Donahue, J., Krähenbühl, P., and Darrell, T. (2017).敵対的特徴学習．*表現の学習に関する国際会議*にて．

また、このような研究では、研究者は、研究者としての経験を生かして、研究者としての能力を最大限に発揮できるように、研究を進めています。逐次モンテカルロ法への入門。

*逐次モンテカルロ法の実際*, 3-14ページ.シュプリンガー。

このような場合には、そのような計算機を用いた計算を行う必要があります。ベイズフィルタリングのための逐次モンテカルロサンプリング法について.*統計とコンピューティング*, 10(3):197-208.

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015).最大平均不一致最適化を介した生成的ニューラルネットワークの学習．*人工知能の不確実性』*で．

Eitz, M., Hays, J., and Alexa, M. (2012).人間はどのようにオブジェクトをスケッチするのか？*ACM Trans.Graph.(Proc.*

*SIGGRAPH）*，31(4):44:1-44:10．

フィッシャー，R.A. (1925).統計的推定の理論．*ケンブリッジ哲学協会の数理論文集*, 22(5).

Foti, N., Xu, J., Laird, D., and Fox, E. (2014).隠れマルコフモデルの確率的変分推論.*隠れたマルコフモデルのための確率的変分推論*.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. (2016).確率層を持つ逐次ニューラルモデル。ニューラル*情報処理システム』*にて。

フリードマン、J. 、ハスティ、T. 、およびティブシラニ、R. (2001).*統計的学習の要素*、第1巻。

統計学のシュプリンガーシリーズ シュプリンガー、ベルリン。

フリードマン、N.リニアル、M.ナハマン、I.およびPe'er,D. (2000).発現データの解析にベイジアンネットワークを用いる。*計算生物学*、7(3-4):601-620.

Ge, H., Xu, K., Scibior, A., Ghahramani, Z., et al. (2018).確率的プログラミングのためのチューリング言語.*人工知能と統計学*における．

このような状況下では，標本を作成する際には，標本を作成した後に，標本を作成した後に，標本を作成した後に，標本を作成した後に，標本を作成した後に，標本を作成した後に，標本を作成した後に，標本を作成することができる。限界密度を計算するためのサンプリングに基づくアプローチ．

*アメリカ統計学会誌*、85(410):398-409。

予測されたデータは、そのデータを用いて、そのデータがどのように分析されているかを示しています。*ベイズデータ分析*.統計科学の教科書シリーズ.CRC Press, Boca Raton, FL, 第 3 版。

議論の対象となっているのは，「データ分析」であり，「データ分析」ではなく，「データ分析」である。回*帰と多階層/階層モデルを用いたデータ分析*。

ケンブリッジ大学出版局。

このような場合には、モデルの適合性を評価するためには、モデルの適合性を評価することが重要であると考えられる。実現された不一致によるモデルの適合性の事後予測評価.統計学*の進歩*, 733-760 ページ.

ゲルマン、A.とシャリジ、C.R. (2013).ベイジアン統計学の哲学と実践.*数理統計心理学の英国ジャーナル*、66（1）:8-38。

Ghahramani, Z. (2015). 確率的機械学習と人工知能. *Nature*, 521(7553):452-459.

孫 恭子, G.Z., Chen, H.H., Lee, Y.C., and Chen, D. (1990).高次リカレントネットワークと文法推論．ニューラル*情報処理システム*。

Gneiting, T. and Raftery, A. E. (2007).厳密に適切な採点ルール、予測、推定。

*アメリカ統計学会誌*、102(477):359-378.

Goodfellow, I., Bengio, Y., and Courville, A. (2016).ディープラーニング．MIT Pressのために準備中の本．

このような状況下では，研究者は，研究者としての立場に立った研究を行うことが重要であると考えています。生成的な敵対者ネット.ニューラル*情報処理システム*.

グッドフェロー, I. J. (2014).生成モデルを推定するための区別可能基準について.*ICLRワークショップ*にて。

このような研究は，研究者の間では，「研究者のための言語」と呼ばれています。チャーチ：生成的モデルのための言語.*人工知能における不確実性*。

Graves, A. (2016). リカレントニューラルネットワークの適応的計算時間. *arXiv preprint arXiv:1603.08983*.

Graves, A., Wayne, G., and Danihelka, I. (2014). ニューラルチューリングマシン. *arXiv preprint arxiv:1410.5401*.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. (2015).DRAW: 画像生成のためのリカレントニューラルネットワーク.In *International Conference on Machine Learning*.

Gregor, K., Danihelka, I., Mnih, A., Blundell, C. and Wierstra, D. (2014).深層自己回帰ネットワーク.*機械学習に関する国際会議*にて。

Gutmann, M. and Hyvärinen, A. (2010).ノイズ-コントラスト推定。正規化されていない統計モデルのための新しい推定原理．*人工知能と統計学*。

分類を介した難治性生成モデルの統計的推論. *arXiv preprint arXiv:1407.4981*.

Hafner, D., Tran, D., Irpan, A., Lillicrap, T., and Davidson, J. (2018).ノイズの対比的なプライヤーを用いたディープニューラルネットワークにおける信頼性の高い不確かさの推定. *arXiv preprint*.

確率論的シミュレーションモデルの統計的推論-理論と応用-．確率的シミュレーションモデルのための統計的推論-理論と応用.また、そのためには、そのような研究を*行うことが重要であると考えている*。

He, K., Zhang, X., Ren, S., and Sun, J. (2016).画像認識のための深層残差学習．*コンピュータビジョンとパターン認識*において．

Hernández-Lobato, J. M., Li, Y., Rowland, M., Hernández-Lobato, D., Bui, T. and Turner, R. E. (2016).ブラックボックス*α発*散最小化．*機械学習国際会議*にて．

Hinton, G. E. (2002).対照的発散を最小化することによる専門家の訓練産物.*神経計算*, 14(8):1771-1800.

Hinton, G. E. Osindero, S. and Teh, Y. W. (2006).ディープビリーフネットのための高速学習アルゴリズム.

*神経計算*、18:1527-1554。

このように、ニューラルネットワークの記述長を最小化することで、ニューラルネットワークをシンプルに保つことができます。重みの記述長を最小化することによってニューラルネットワークをシンプルに保つ。*学習理論会議*、5-13ページ、ニューヨーク、ニューヨーク、アメリカ。ACM.

Hinton, G. E. and Zemel, R. S. (1994).このように、本研究では、このような研究を行うためには、研究者は、研究者の研究成果に基づいた研究を行う必要があると考えています。このように、このような研究は、神経*情報処理システムの進歩、*3-3ページ。

Hochreiter, S.とSchmidhuber, J. (1997).長時間短期記憶.*神経計算*, 9(8):1735-1780.

勾配降下を利用した学習は、学習者の学習意欲を向上させるためには、学習者の学習意欲を高める必要がある。勾配降下を用いた学習．

*人工ニューラルネットワーク国際会議*, 87-94ページ.

Hoffman, M. and Blei, D. (2015).確率的構造化変分推論．*人工知能と統計学*では、361-369ページ。

Hoffman, M. D. (2017).マルコフ連鎖モンテカルロ法を用いた深層潜伏ガウスモデルの学習．*機械学習に関する国際会議で*．

確率的変分推論.*機械学習研究ジャーナル*, 14(1):1303-1347.

このような場合には、このような計算機を用いた計算を行うことで、計算機の性能を向上させることができる。ノーUターンサンプラー：ハミルトニアンモンテカルロにおける適応的なパス長設定.*機械学習研究*、15(1):1593-1623.

ホップフィールド，J. J. J. (1982).集団的な計算能力を持つ神経ネットワークと物理システム.また、このような*物理システムでは、神経回路網と物理システムとの間には、それぞれの物理システムに応じて、様々な計算能力があることが知られている。*

伊高，R.とジェントルマン，R. (1996).R: データ解析とグラフィックスのための言語.*計算とグラフ統計*, 5(3):299-314.

ジョンソン、M.とウィルスキー、A.S. (2014).ベイズ時系列モデルの確率的変分推論.*機械学習に関する国際会議*にて。

このような状況下では，このような問題を解決するためには，研究者の努力が必要であると考えられます。グラフィカルモデルのための変分法入門.*機械学習*, 1-51 ページ.

このような研究は、その研究者の研究者が、研究者の研究者としての自覚を持っているかどうか、また、研究者の研究者としての自覚を持っているかどうかにかかっていると考えられます。グラフィカルモデルのための変分法の紹介。*機械学習*。

Jouppi, N. P., Young, C. Patil, N. Patterson, D. Agrawal, G. Bajwa, R. Bates, S. Bhatia, S. Boden, N. Borchers, A. et al. (2017).テンソル処理装置のデータセンター内性能解析。*第44回コンピュータアーキテクチャ国際シンポジウムの講演論文集に掲載され*ています。

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018).品質、安定性、バリエーションの向上のためのガンの漸進的成長。*表現の学習に関する国際会議で*．

ケラー，J. B. (1986).頭の確率．*アメリカ数学月報*, 93(3):191-197.

Kingma, D. and Welling, M. (2014a).変分ベイズの自動符号化．*表現の学習に関する国際会議*にて.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014).深層生成モデルを用いた半教師付き学習.ニュー*ラル情報処理システム*.

Kingma, D.P., Salimans, T., and Welling, M. (2016).逆自己回帰フローを用いた変分推論の改善．ニュー*ラル情報処理システム』*にて。

Kingma, D. P. and Welling, M. (2014b).変分ベイズの自動符号化．*表現の学習に関する国際会議*にて．

確率論的グラフモデル：原理と技術。確率的*グラフィカルモデル: 原理と技術*.マサチューセッツ工科大学出版局.

このように、画像の分類には、画像の種類に応じた分類の仕方があります。深い畳み込みニューラルネットワークを用いた画像ネット分類.このように、本研究では、このような問題を解決*するための研究を行っています*。

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. (2017).自動微分変分推論.*Journal of Machine Learning Research*, 18(14):1-45.

ｋｕｓｎｅｒ，Ｍ.Ｊ.およびＨｅｒｎａｎｄｅｓ-Ｌｏｂａｔｏ，Ｊ.Ｍ. （２０１６）．Gumbel-Softmax分布を持つ離散要素の配列に対するGAN.*NIPSワークショップ*にて。

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. (2017).文法変分自動エンコーダー．*機械学習国際会議*にて．

このような研究は、研究者が研究者の立場に立って、研究者の立場に立って研究を行うことを目的としている。人のように学び、人のように考える機械を構築する。

ラプラス，P.S. (1986). 事象の原因の確率についての回想録. *統計科学*, 1(3):364-378.

ローレンス，N. (2000).*確率モデルにおける変量推論*．博士論文。

ローレンス，N. (2005).ガウス過程潜在変数モデルを用いた確率的非線形主成分分析．*機械学習研究ジャーナル*, 6:1783-1816.

Li, Y., Swersky, K., and Zemel, R. (2015).生成モーメントマッチングネットワーク.*機械学習国際会議*にて．

Li, Y. and Turner, R. E. (2016).Rényi発散を用いた変量推論．ニュー*ラル情報処理システム*において．

マッケイ、D.J. (2003).*情報理論、推論、学習アルゴリズム*.ケンブリッジ大学出版局。

MacKay, D. J. C. (1992).*適応モデルのためのベイズ法*。カリフォルニア工科大学の博士論文。

マクラウリン、D、デュヴノー、D、ジョンソン、M、およびアダムス、R.P. (2015).Autograd.ネイティブPythonの逆モード分化。

マニング、C.D.およびシュッツェ、H. (1999).*統計的自然言語処理の基礎*、第999巻。MITプレス。

このような研究は、研究者が研究者の立場に立って研究を進めていくために必要なものであり、そのためには、研究者が研究者の立場に立って研究を進めていく必要がある。このような研究では、研究者は、研究者の研究成果に基づいて、研究者の研究成果に基づいて、研究者の研究成果に基づいた研究を行うことができます。

近似ベイズ計算法.*統計とコンピューティング*, 22(6):1167-1180.

McInerney, J., Ranganath, R., and Blei, D. M. (2015).ストリーム上の母集団事後とベイズ推論．ニュー*ラル情報処理システム*において．

孟 暁-L. (1994).事後予測p値．*統計学概論*、1142-1160 ページ。

民家, T. P. (2001).近似ベイジアン推論のための期待伝播.このように，本研究では，このような問題点を解決するためには，研究者*の努力が必要であると考えている。*モーガン・カウフマン・パブリッシャーズ・インク.

ミンスキー、M. (1975).知識を表現するためのフレームワーク。

Mohamed, S. and Lakshminarayanan, B. (2016).暗黙的生成モデルにおける学習. *arXiv preprint arXiv:1610.03483*.

確率論的観点からの機械学習の研究には、機械学習と確率論的観点からの研究が必要であると考えられる。*機械学習：確率論的視点*.MITプレス.

このような研究では，高次元の潜在変数モデルの下での確率の評価が重要な課題となっています。高次元潜在変数モデルの下での確率の評価．このように、本研究では、高次元潜在変数モデルを用いて確率を評価することを目的とした研究を*行って*いる。

確率論的フィードフォワードネットワークの学習確率的フィードフォワードネットワークの学習．技術報告書。

ニール，R.M. (1994).*ニューラルネットワークのためのベイズ学習*.トロント大学の博士論文。

Neal, R. M. (2011).ハミルトニアンダイナミクスを用いたMCMC.*マルコフ連鎖モンテカルロのハンドブック*.

Neal, R.M. and Hinton, G.E. (1993).インクリメンタルや他のバリエーションを正当化するemアルゴリズムの新しい見方．*グラフィカルモデルの学習*, 355-368ページ.

Nelsen, R. B. (2006).*コピュラス入門 (Springer Series in Statistics*).スプリンガー・ベルラーグ・ニューヨーク, Inc.

Newell, A. and Simon, H. A. (1976).経験的探究としてのコンピュータ科学.記号と検索.

*ACMの通信*、19(3):113-126。

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016).ピクセルリカレントニューラルネットワーク. *arXiv preprint arXiv:1601.06759*.

オズボーン、M. （2010）。*逐次予測、最適化と直交のためのベイズガウスプロセス*。博士論文、オックスフォード大学ニューカレッジ。

確率論的探索を用いた変分ベイズ推論は，確率論的探索を用いた変分ベイズ推論であり，確率論的探索を用いた変分ベイズ推論である。確率的探索を用いた変分ベイズ推論.

*機械学習に関する国際会議*にて.

確率論的探索を用いた変分ベイズ推論は、確率論的探索を用いた変分ベイズ推論である。確率的探索を用いた変分ベイジアン推論.このように、本研究では、*機械学習に関する国際会議を開催している*。

Papamakarios, G. and Murray, I. (2016).ベイズ条件付き密度推定を用いたシミュレーションモデルの高速-フリー推論．ニュー*ラル情報処理システム*において．

Papamakarios, G., Murray, I., and Pavlakou, T. (2017).密度推定のためのマスクド自己回帰フロー．ニュー*ラル情報処理システムの進歩*、2335-2344ページ。

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. (2018).画像トランスフォーマー.*機械学習国際会議*にて。

パール、J. (2003).因果性：モデル、推論、推論.*エコノメトリック理論*, 19(675-685):46.

Pfeffer, A. (2007).IBALの設計と実装: 汎用確率論的言語.

*統計的関係学習入門、*399ページ。

プリチャード、J.K.、セイエルスタッド、M.T.、ペレス＝レザウン、A.、フェルドマン、M.W. (1999).ヒトY染色体の個体数増加：Y染色体マイクロサテライトの研究。*分子生物学と進化*、16(12):1791-1798。

Probtorch Developers (2017).Probtorch. https://github.com/probtorch/probtorch[。](https://github.com/probtorch/probtorch)

疎な近似ガウス過程回帰の統一的な見方としては，「疎な近似ガウス過程回帰」と「疎な近似ガウス過程回帰」が挙げられる。疎な近似ガウス過程回帰の統一見解.*機械学習研究*, 6:1939-1959.

Radford, A., Metz, L., and Chintala, S.(2016).深い畳み込み型生成的敵対的ネットワークを用いた教師なし表現学習.*表現の学習に関する国際会議*にて．

Raiko, T., Li, Y., Cho, K., and Bengio, Y. (2014).反復的ニューラル自己回帰分布推定器 nade-k.ニューラル*情報処理システムの進歩*, 325-333 ページ.

Ranganath, R., Altosaar, J., Tran, D., and Blei, D. M. (2016a).オペレータの変分推論．ニュー*ラル情報処理システム』*にて。

Ranganath, R., Gerrish, S., and Blei, D. M. (2014).ブラックボックス変量推論．*人工知能と統計学』*にて。

Ranganath, R., Tang, L., Charlin, L., and Blei, D. M. (2015a).ディープな指数族．*人工知能と統計学』*にて。

Ranganath, R., Tran, D., and Blei, D. M. (2015b).階層的変分モデル. *arXiv preprint arXiv:1511.02386*.

Ranganath, R., Tran, D., and Blei, D. M. (2016b).階層的変分モデル．*機械学習国際会議*にて．

このように，機械学習のためのガウス過程の研究は，機械学習のためのガウス過程の研究のために行われています。*機械学習のためのガウス過程*.MIT Press.

リーズンデ、D.J.とモハメド、S. (2015).正規化フローを用いた変量推論.*機械学習に関する国際会議で*．

Rezende, D. J., Mohamed, S., Danihelka, I., Gregor, K., and Wierstra, D. (2016).ディープな生成モデルにおけるワンショット一般化.*機械学習国際会議*にて．

このような研究では、「深層生成モデルにおける確率的バックプロパグと近似推論」が重要な課題となっています。ディープ生成モデルにおける確率的バックプロパゲーションと近似推論.議論は、*機械学習に関する国際会議で行われ*ています。

Ritchie, D., Horsfall, P., and Goodman, N. D. (2016).確率的プログラムのための深い償却推論. *arXiv preprint arXiv:1610.05735*.

Robbins, H. and Monro, S. (1951).確率的近似法．*確率的近似法*

*統計学*です。

Robert, C.P. and Casella, G. (1999).*モンテカルロ統計学的手法*．シュプリンガー．

Roy, A., Vaswani, A., Neelakantan, A., and Parmar, N. (2018).ベクトル量子化オートエンコーダーの理論と実験. *arXiv preprint arXiv:1805.11063*.

確率論的には，統計学者のためのベイズ的に正当化された関連性のある頻度計算が必要である。ベイズ的に正当で関連性のある度数計算は、応用統計学者のためのものである。*統計学概論*, 12(4):1151-1172.

ルドルフ、M. R. Ruiz、F. J. R. Mandt、S. and Blei、D. M. (2016).指数的ファミリーエンベッディング．

*神経情報処理システム*において。

Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. (1988).*並列分散処理*、第1巻。IEEE。

このような研究は，研究者の間では「学習の効率化」が重要な課題となっています．深層ボルツマン機械の効率的な学習.このような研究を行うためには，*人工知能と統計学に関する国際会議を開催する必要があります*。

このように，本研究では，ベイズ的な確率的行列因数分解のためのマルコフ連鎖モンテカルロ法を用いたベイズ確率的行列因数分解の研究を行っている。マルコフ連鎖モンテカルロ法を用いたベイズ確率的行列因数分解．このように、本研究では、*機械学習に関する国際会議、*880-887 ページを参照してください。ACM.

このような研究では、研究者は、研究成果を発表するために必要な情報を得るために、研究成果を発表することが重要である。ディープビリーフネットワークの定量的分析について.*機械学習に関する国際会議*にて。

Salimans, T., Kingma, D., and Welling, M. (2015).マルコフ連鎖モンテカルロと変分推論．ギャップを埋める．*機械学習国際会議*にて．

確率論的なプログラミングのためには、Pythonを用いた確率論的なプログラミングが必要である。PyMCを用いたPythonでの確率的プログラミング. *arXiv preprint arXiv:1507.08050*.

Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. (2016).PyMC3を用いたPythonでの確率的プログラミング．*PeerJ Computer Science*, 2:e55.

このような研究は、研究者の間では「研究者のための研究」と呼ばれており、研究者の間では「研究者のための研究」と呼ばれています。難易度の高いネットワークでの作業可能な部分構造の利用。ニュー*ラル情報処理システム*。

Ścibior, A., Ghahramani, Z., and Gordon, A. D. (2015).モナドを用いた実用的な確率的プログラミング.*第8回ACM SIGPLANシンポジウム*、165-176ページ、ニューヨーク、米国ニューヨーク。ＡＣＭ

プレス。

Shazeer, N., Cheng, Y., Parmar, N., Tran, D., Vaswani, A., Koanantakool, P., Hawkins, P., Lee, H., Hong, M., Young, C., Sepassi, R., and Hechtman, B. (2018).Mesh-TensorFlow: スーパーコンピュータのためのディープラーニング.ニュー*ラル情報処理システム*において。

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017).法外に巨大なニューラルネットワーク。疎ゲート化されたmixture-of-experts層. *arXiv preprint arXiv:1701.06538*.

Shi, J., Chen, J., Zhu, J., Sun, S., Luo, Y., Gu, Y., and Zhou, Y. (2017).Zhusuan.ベイジアンディープラーニングのためのライブラリ. *arXiv preprint arXiv:1709.05870*.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016).ラダー変分自動エンコーダー.ニュー*ラル情報処理システム*において。

Spiegelhalter, D. J., Thomas, A. Best, N. G. and Gilks, W. R. (1995).BUGS.ギブスサンプリングを用いたベイズ推論, バージョン 0.50.*MRC生物統計学ユニット、ケンブリッジ*。

杉山真理子・鈴木俊哉・金森俊哉（2012）．ブレグマン発散下での密度比マッチング：統一的な密度比推定の枠組み.*統計研究所紀要*

....

このように，本稿では，このような問題を解決するために必要となる，階層型ベイズのノンパラメトリックモデルの開発とその応用例を紹介します。階層型ベイズノンパラメトリックモデルの応用.

*ベイズノンパラメトリック*、１．

テネンバウム、J.B.、ケンプ、C.、グリフィス、T.L.、およびグッドマン、N.D.（2011）。どのように心を成長させる。

統計・構造・抽象化. *科学*, 331(6022):1279-1285.

Tieleman, T. and Hinton, G. (2012).講義6.5-RmsProp.勾配を最近の大きさの連続平均で割る.COURSERA: 機械学習のためのニューラルネットワーク.

確率論的主成分分析とは何か？確率的主成分分析.*王立統計学会誌.シリーズ B (統計的方法論*), 61(3):611-622.

Tolpin, D., van de Meent, J.-W., Yang, H., and Wood, F. (2016).確率的プログラミング言語Anglicanの設計と実装.第*28回機能的プログラミング言語の実装と応用に関するシンポジウム講演論文集*、6ページ。

Tomczak, J. M. and Welling, M. (2018).ヴァムプリオを用いたヴェー．*人工知能と統計学*において．

遺伝学的研究のための暗黙の因果関係モデルの開発。ゲノムワイドな関連研究のための暗黙の因果モデル. *arXiv preprint arXiv:1710.10742*.

Tran, D. and Blei, D. M. (2018).ゲノムワイドな関連研究のための暗黙の因果モデル.*表現の学習に関する国際会議*にて。

Tran, D., Blei, D. M. and Airoldi, E. M. (2015).コピュラ変量推論．ニュー*ラル情報処理システム*において．

確率論的計画法は、確率論的計画法と比較して、確率論的計画法と比較して、確率論的計画法と比較して、確率論的計画法と比較して、確率論的計画法と比較して、確率論的計画法と比較して、確率論的計画法と比較して、確率論的計画法と比較して、確率論的計画法の効果が高いことを示す。深い確率的プログラミング.*表現の学習に関する国際会議*にて．

Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D., and Blei, D. M. (2016a).エドワード。確率的モデリング、推論、批判のためのライブラリ。

Tran, D., Ranganath, R., and Blei, D. M. (2016b).変分ガウス過程．*表現の学習に関する国際会議で*．

Tristan, J.-B., Huang, D., Tassarotti, J., Pocock, A. C., Green, S., and Steele, G. L. (2014).Augur.

データ並列確率モデリング.*神経情報処理システム*において。

Tukey, J. W. (1962).データ分析の未来.*数理統計学概論*, 33(1):1-67.

上原幹雄・佐藤一朗・鈴木幹雄・中山一輝・松尾康夫（2016）．密度比推定の観点から見た生成的敵対ネット. *arXiv preprint arXiv:1610.02920*.

また，このような手法を用いることにより，ノンパラメトリックなガウス過程法の情報化率を向上させることが可能となる。ノンパラメトリックガウスプロセス法の情報率.*機械学習研究論文集*, 12:2095-2119.

Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Kaiser, L., Kalchbrenner, N., Parmar, N., Sepassi, R., Shazeer, N. and Uszkoreit, J. (2018).ニューラル機械翻訳のためのTensor2tensor.*CoRR*, abs/1803.07416.

このような場合には、「このようなモデルを使っている」ということになりますが、「このようなモデルを使っている」ということにはなりません。グラフィカルモデル、指数族、変分推論。*機械学習の基礎と動向*, 1(1-2):1-305.

このような場合には、そのような問題を解決するためには、以下のようなことが必要である。ベイジアンノンパラメトリックモデルのための切り捨てなしのオンライン変分推論.また、このような研究は、研究者*の研究成果の一つであり、研究者の研究成果の一つである。*

また，このような研究では，専門家がどのようにして専門家を分析しているのかを知ることができます。専門家の混合物のためのベイズ法．

*神経情報処理システムの進歩、*351-357ページ。

確率論的勾配ランジュバン力学を用いたベイズ学習は，確率論的勾配ランジュバン力学を用いたベイズ学習とは異なります。確率的勾配ランジュバン力学によるベイズ学習.*機械学習国際会議*.

Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. (2018).フリップアウト。ミニバッチ上での効率的な疑似非依存重み摂動.*表現の学習に関する国際会議で*．ウィルキンソン、D.J. (2011).*システム生物学のための確率論的モデリング*．CRCプレス。

Winkler, R. L. (1994).確率の評価．非対称スコアリングルール.*経営科学*、40(11):1395-1405。

Wu, Y., Li, L., Russell, S., and Bodik, R. (2016).Swift.確率的プログラミング言語のためのコンパイルされた推論. *arXiv preprint arXiv:1606.09242*.

ジンコフ、R.とシャン、C.-c.(2016).プログラム変換としての推論アルゴリズムの構成. *arXiv preprint arXiv:1603.01882*.

Zoph, B. and Le, Q. V. (2017).強化学習を用いたニューラルアーキテクチャ探索．*表現の学習に関する国際会議で*．

# 付録A: 深層確率計画法

## A.1 モデル例

http://edwardlib.org には、モデル、推論方法、完全なスクリプトを含む多くの例[が](http://edwardlib.org/)あります。以下では、いくつかのモデルの例を説明します。付録A.6には推論の例(確率的変分推論)を、付録A.7には完全なスクリプトを説明します。

この論文のすべての例は包括的なものであり、インポート文と固定値のみを除いています。この論文のコンパニオンページ（http://edwardlib.org/iclr2017[）では、](http://edwardlib.org/iclr2017)実行可能なコードを含む機械可読形式の例を参照してください。

## A.2 分類のためのベイジアンニューラルネットワーク

ベイズ型ニューラルネットワークとは、重みに事前分布を持つニューラルネットワークのことです。

二値ラベル *yn* ∈ {0*,*1} を持つ観測 (*xn,yn*) の尤度を次のように定義する

*p*(*yn* | **W**0*,***b**0*,***W**1*,***b**1 ; *xn*) = ベルヌーイ(*yn* |NN(*xn* ; **W**0*,***b**0*,***W**1*,***b**1))。

ここでNNは，重みとバイアスが固有変数**W**0*,***b**0*,***W**1*,***b**1に設定された2層の神経回路網である．重みとバイアスの事前分布を標準正規分布と定義する．図A.1を参照のこと．*N個の*データ点、*D個の*特徴量、および*H個の*隠れた単位がある。

## A.3Lat ent Dirichlet Allocation

図A.2を参照してください。このプログラムは説明のために書かれていることに注意してください。我々は，実際にはベクトル化を推奨します：スカラーのランダム変数をリストのリストに格納する代わりに，次元数の多い少数のランダム変数を表現することを好むべきです．

**W**

0

**b**

0

**W**

1

**b**

1

*y*

*n*

*x*

*n*

*N*

W\_0 = 正規(mu=tf. ゼロ([D, H]), シグマ=tf. 物([D, H])W\_1 = 正規(mu=tf. ゼロ([H, 1]), sigma=tf. ones([H, 1])) b\_0 = 正規(mu=tf. ゼロ(H), sigma=tf. ones(H)) b\_1 = 正規(mu=tf. ゼロ(1), sigma=tf. ones(1)) x = tf. プレースホルダ(tf. float32, [N, D])

1. = ベルヌーイ(logits=tf. matmul(tf. nn. tanh(tf. matmul(x, W\_0) + b\_0), W\_1) + b\_1) **図A.1:** 分類のためのベイズ・ニューラル・ネットワーク

D = 4 *#文書数*

N = [11502, 213, 1523, 1351*] # words per doc*

K = 10 *# トピック数*

V = 100000 *#語彙数*

*φ*

*k*

*θ*

*d*

*z*

*ディーエヌ*

*w*

*ディーエヌ*

*N*

*D*

*K*

シータ = ディリクレ(α=tf.ゼロス([D, K]) + 0.1)

phi = Dirichlet(α=tf. zeros([K, V]) + 0.05)

1. = [[0] \* N]\* D w = [[0] \* N]d が range(D) **の場合は** \* D.

**for** n **in** range(N[d]): z[d][n] = Categorical(pi=θ[d, :]) w[d][n] = Categorical(pi=phi[z[d][n], :])

**図A.2:** 潜在ディリクレの割り当て (Blei et al. , 2003)。

## A. 4ガウス行列因数分解

図A.3を参照してください。

N = 10

**U**

*m*

**Y**

*ｎ，ｍ*

**V**

*n*

*M*

*N*

M = 10

K = 5 *# 潜行寸法*

1. = 正規分布(mu=tf.ゼロ([M, K]), sigma=tf. ones([M, K])
2. = 正規分布(mu=tf. ゼロ([N, K]), sigma=tf. もの([N, K])

Y = 正規分布(mu=tf. matmul(U, V, transpose\_b=True), sigma=tf. ones([N, M])**図A.3:** ガウス行列の因数分解.

## A.5 ディリクレプロセス混合物モデル

図A.4を参照してください。

ディリクレ過程混合モデルは次のように書かれています。

mu = DirichletProcess(α=0.1, base\_cls=Normal, mu=tf. ゼロ(D), シグマ=tf. 物(D), sample\_n=N) x = Normal(mu=mu, シグマ=tf. 物([N, D])

ここで mu は形状(N, D)を持ちます。DirichletProcessランダム変数は、基本分布Normal(mu, sigma)によって与えられた形状を持つそれぞれのsample\_n=Nの抽選を返します。DirichletProcessランダム変数を定義する重要な要素は、確率的なwhileループです。以下にそれを定義します。基底分布を用いたより複雑なバージョンについては、エドワードのコードベースを参照してください。

**def** dirichlet\_process(alpha): **def** cond(k, beta\_k): flip = Bernoulli(p=beta\_k) **return** tf. equal(flip, tf. constant(1))

**def** body(k, beta\_k): beta\_k = beta\_k \* Beta(a=1.0, b=alpha)

**戻り値** k + 1, beta\_k

k = tf.定数(0) beta\_k = ベータ(a=1.0, b=α)

stick\_num, stick\_beta = tf. while\_loop(cond, body, loop\_vars=[k, beta\_k]) **return** stick\_num.

**図A.4:** ディリクレ過程混合モデル.

## A.6 推論の例。確率的変動推論

サブグラフの設定では、フルモデルのサブグラフで作業しながらデータのサブサンプリングを行います。この設定は、データとモデルがメモリに収まらない場合に必要です。アルゴリズムの計算複雑度（反復あたり）とメモリの複雑度の両方がデータセットのサイズに依存しないという点でスケーラブルです。

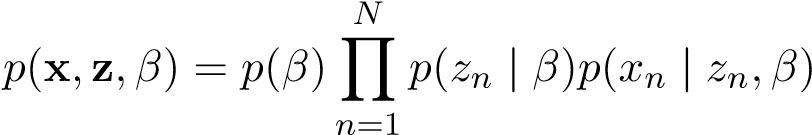
コードには、図2.5で説明した混合モデルの実行例を使用しています。

N = 10000000 *# データセットサイズ*

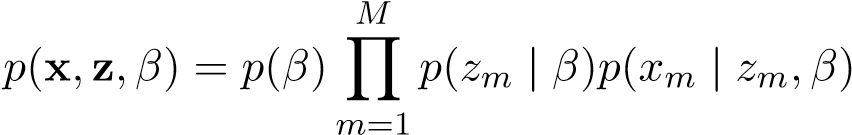
D = 2 *# データ次元*

K = 5 *# クラスター数*

モデルは

*.*

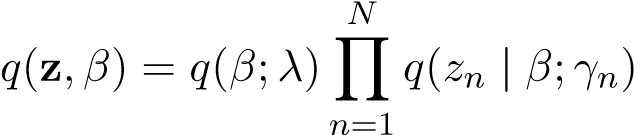
メモリの問題を回避するために、モデルの部分グラフのみで作業を行います。



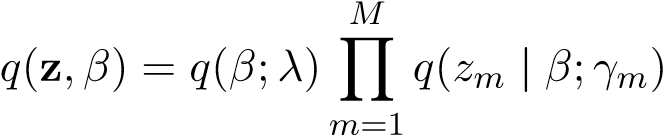
M = 128 *# ミニバッチサイズ*

beta = 正規(mu=tf. ゼロス([K, D]), sigma=tf. ones([K, D]) z = カテゴリ(logits=tf. ゼロス([M, K]) x = 正規(mu=tf. gather(beta, z), sigma=tf. ones([M, D]) )

変分モデルが

*,*

によってパラメータ化されています。ここでも、我々はモデルの部分グラフのみを扱います。

*.*

でパラメータ化された {*λ*,{*γm*}}。重要なことは、{*γm*}については、*Nで*はなく*Mの*パラメータのみがメモリに記憶されていることです。

qbeta = Normal(mu=tf.変数(tf. zeros([K, D])), シグマ=tf. nn. softplus(tf.変数(tf. ゼロス[K, D]))

qz\_variables = tf.変数(tf. zeros([M, K]) qz = Categorical(logits=qz\_variables)

発散尺度 KL(*q kp*)を最小化する変分法 KLqp を使用する (Jordan et al. , 1999a)。**z**の部分集合を与えられた*β*に対する大域推論と、*βを*与えられた**z**の部分集合に対する局所推論の2つのアルゴリズムをインスタンス化する。 また、各ステップでデータを変更できるように、データ用のTensorFlowプレースホルダx\_phを渡す。

inference\_global = ed.KLqp({beta: qbeta}, data={x: x\_ph, z: qz}) inference\_local = ed.KLqp({z: qz}, data={x: x\_ph, beta: qbeta})

アルゴリズムを scale 引数で初期化することで、z と x の計算が適切にスケーリングされるようにします。これにより、確率勾配の不偏な推定が可能になります。 inference\_global. initialize(scale={x: float(N) / M, z: float(N) / M}) inference\_local. initialize(scale={x: float(N) / M, z: float(N) / M})

次のバッチのデータを提供するnext\_batch関数があると仮定して、アルゴリズムを実行します。

qz\_init = tf. initialize\_variables([qz\_variables]) **for** \_ **in** range(1000): x\_batch = next\_batch(size=M) **for** \_ **in** range(10).*# 局所的な推論を行う*

inference\_local.update(feed\_dict={x\_ph: x\_batch})

*# グローバルパラメータを更新する*

inference\_global. update(feed\_dict={x\_ph: x\_batch})

*# 局所因子の再初期化* qz\_init.run()

これは、各バッチごとに新しい局所的な変分因子のセットを用いて推論を行うためです。このデモは、SGLD (確率的勾配ランジュバン力学)のような他の推論アルゴリズムにも簡単に適用できます：qbetaとqzを経験的確率変数に置き換えて、ed.KLqpの代わりにed.SGLDを呼び出します。

データとモデルがメモリ内に収まっていても、高速推論のためにデータのサブサンプリングを実行したい場合は、サブグラフを定義しないことをお勧めします。フルモデルを再定義し、単にプレースホルダでローカル変数をインデックス化することができます。プレースホルダは、一度にどのローカル変数を更新するかを決定するために、実行時に供給されます。(詳細については、ウェブサイトのAPIを参照してください)。

## A.7 完全な例

## A.8 可変オートエンコーダー

図A.5を参照してください。

## A.9単語エンベッディングのための 確率モデル

図A.6を参照してください。この例は、データ・サブサンプリングを使用しています（セクション2.3.4）。プリオールと条件付き尤度は、データのミニバッチについてのみ定義されます。同様に、変分モデルは、与えられたミニバッチで使用される埋め込みのみをモデル化します。TensorFlow 変数には、語彙全体の埋め込みベクトルが含まれています。TensorFlow のプレースホルダは、正しい埋め込みベクトルが与えられたミニバッチの変分パラメータとして使用されることを保証します。

**ed として edward をインポート tf として tensorflow をインポート**

**from edward.models import** Bernoulli, Normal **from scipy.misc import** imsave **from tensorflow.contrib import** slim

**from tensorflow.examples.tutorials.mnist import** input\_data M = 100 *# 学習中のバッチサイズ* d = 2 *# 潜在変数次元*

*# 確率モデル(サブグラフ)*

z = 正規(mu=tf. ゼロ([M, d]), sigma=tf. ones([M, d]) h = 密(256, activation='relu')(z)

x = ベルヌーイ(logits=Dense(28 \* 28, activation=None)(h))

*# 変分モデル(サブグラフ)* x\_ph = tf. placeholder(tf. float32, [M, 28 \* 28]) qh = Dense(256, activation='relu')(x\_ph)(x\_ph) qz = Normal(mu=Dense(d, activation=None)(qh).

シグマ＝デンセ(d,活性化＝'ソフトプラス')(qh)

*# p(x, z) と q(z | x) を x の同じ TensorFlow プレースホルダにバインドします。*

mnist = input\_data. read\_data\_sets("data/mnist", one\_hot=True) data = {x: x\_ph}.

推論 = ed.KLqp({z: qz}, data)

オプティマイザー = tf.RMSPropOptimizer(0.01, epsilon=1.0) inference. initialize(optimizer=optimizer)

tf. initialize\_all\_variables().

n\_epoch = 100 n\_iter\_per\_epoch = 1000 **for** \_ **in** range(n\_epoch).

**for** \_ **in** range(n\_iter\_per\_epoch).

x\_train, \_ = mnist.train.

info\_dict = 推論. update(feed\_dict={x\_ph: x\_train})

*# 画像を生成します。*

imgs = x. value(). eval() **for** m **in** range(M): imsave("img**/%d**.png" % m, imgs[m]. reshape(28, 28))

**図A.5:** バッチ学習を用いた vae (Kingma and Welling, 2014b) のための完全なスクリプト．これは，1000回の更新ごとにMNISTの桁数を生成する．

ベルヌーイ変数 y\_pos と y\_neg は、それぞれ1と0に固定されています。これらの変数は、ある単語が与えられたコンテキストウィンドウの対象となる単語であるか、または負のサンプルとして描画されたかをモデル化します。

正則化がなければ（プリオールを介して）、我々が最適化する目的は負のサンプリングと同じです。

**ed として edward をインポート tf として tensorflow をインポート**

**from edward.models import** Bernoulli, Normal, PointMass

N = 581238 *#総語数*

M = 128 *# 学習中のバッチサイズ* K = 100 *# 因子数* ns = 3 *# 負のサンプル数* cs = 4 *# コンテキストサイズ* L = 50000 # *語彙数 # 事前埋め込みベクトル* p\_rho = Normal(mu=tf.zeros([M, K]), sigma=tf. sqrt(N) \* tf. ones([M, K]) n\_rho = Normal(mu=tf. zeros([M, ns, K]), sigma=tf. sqrt(N) \* tf. ones([M, ns, K])

*# コンテキストベクトルの優先順位*

ctx\_alphas = Normal(mu=tf. ゼロ([M, cs, K]), sigma=tf. sqrt(N)\*tf. ones([M, cs, K]) )

# *条件付き尤度* ctx\_sum = tf. reduce\_sum(ctx\_alphas, [1])

p\_eta = tf. expand\_dims(tf. reduce\_sum(p\_rho \* ctx\_sum, -1),1)

*n\_*eta = tf. reduce\_sum(n\_rho \* tf. tile(tf. expand\_dims(ctx\_sum, 1), [1, ns, 1]), -1) y\_pos = Bernoulli(logits=p\_eta) y\_neg = Bernoulli(logits=n\_eta*) # バッチ学習用プレースホルダ* p\_idx = tf.プレースホルダ(tf. int32, [M, 1]) n\_idx = tf. プレースホルダ(tf. int32, [M, ns]) ctx\_idx = tf. プレースホルダ(tf. int32, [M, cs])

*# 変分パラメータ (埋め込みベクトル)* rho\_params = tf.Variable(tf. random\_normal([L, K])) alpha\_params = tf.変数(tf. random\_normal([L, K])

*# 埋め込みベクトルの変分布*

MAP(

{p\_rho: q\_p\_rho, n\_rho: q\_n\_rho, ctx\_alphas: q\_alpha}, data={y\_pos: tf. ones((M, 1)), y\_neg: tf. zeros((M, ns))})

推論. initialize() tf. initialize\_all\_variables(). run() **for** \_ **in** range(inference. n\_iter).

targets, windows, negatives = next\_batch(M*) # データを生成する関数*

info\_dict = 推論. update(feed\_dict={p\_idx: targets, ctx\_idx: windows, n\_idx: negatives}) inference. print\_progress(info\_dict)

**図A.6:** バイナリデータの指数族埋め込み (Rudolph et al. , 2016).ここでは、条件付き対数尤度と対数優先度の総和を最大化するためにマップが使用されます。

# 付録B: 単純、分散、および高速化された確率計画法

## B.1 Edward2 on SciPy

バックエンドとしてSciPyを適用することで，我々のトレース実装の幅広い適用性を説明します．

実装は scipy.stats のディストリビューションをラップし、各 rvs メソッドをトレーサブルとして登録します。

ネームスコープからのプライベートな変数は、明示的にアンダースコアで前置されます。の Edward2 とは異なり

TensorFlow Distributionsでは、生成処理はPythonクラスではなくrvsを呼び出してPython関数をラップして記録しています。これは scipy.stats の関数 API の結果であり、これは

TensorFlow Distributionsのオブジェクト指向のもの。

**from scipy import** stats

\_globals = globals() **for** \_name **in** sorted(dir(stats))。\_candidate = getattr(stats, \_name)

**if** isinstance(\_candidate, (stats.多変量解析。

stats. rv\_continuous, stats. rv\_discrete, stats. rv\_histogram))。

rvs = traceable(\_candidate.rvs)

\_globals[\_name] = \_candidate **del** \_candidate

以下はSciPyのEdward2線形回帰プログラムです。

**from edward2.scipy import** stats **as** ed *# ここではrvsが装飾されていると仮定しています。*

**def** linear\_regression(features): coeffs = ed. norm. rvs(loc=0.0, scale=0.1, size=features. shape[1], name="coeffs") loc = np. einsum('ij,j->i', features, coeffs)

labels = ed. norm. rvs(loc=loc, scale=1.

log\_joint = ed. make\_log\_joint\_fn(linear\_regression)

feature = np. random. normal(size=[3, 2]) coeffs = np. random. normal(size=[2]) labels = np. random. normal(size=[3]) out = log\_joint(features, coeffs=coeffs, labels=labels) 詳細はソースコードへのリンクを参照してください。

## B.2 文法変分自動エンコーダー

以下に，文法 vae (Kusner et al. , 2017) を実装する．確率的エンコーダーとデコーダーで構成されている。確率的文脈自由文法をニューラルネットワーク、潜在コード、離散構造の表現を学習するためのエンコーダーで拡張したものである。デコーダのlogitsは形状[batch\_size, max\_timesteps, num\_production\_rules]の3次元である。

エンコーダーは文字列を入力として受け取り、parse\_to\_one\_hotを適用します。これは前処理ステップで、文字列を解析して解析ツリーにし、ツリーから生産ルールを抽出し、各生産ルールをワンショットベクトルに変換します。

デコーダは入力として潜在コードを受け取り、それを生成された文字列を表す生産規則のシーケンスにマッピングします。RNN を適用し、その結果が文法内の有効な生産規則のシーケンスとなるようにマスキングステップを行います。その後、生成規則は文字列に変換されます。

**import parse\_to\_one\_hot**

**クラス ProbabilisticGrammarVariational**(tf. keras. Model).

*""""確率的文法のための償却変分後処理"""*

**def** \_\_init\_\_(self, latent\_size).

*""""確率的文法の変分事後処理を構築します""""* super(ProbabilisticGrammarVariational, self).確率的文法の変分的事後処理を構築します。シーケンシャル([)

tf. keras. layers.Conv1D(64, 3, padding="SAME"), tf. keras.バッチ正規化(), tf. keras.活性化(tf. nn. elu), tf. keras. layers.Conv1D(128, 3, padding="SAME"), tf. keras.バッチ正規化(), tf. keras.活性化(tf. nn. elu), tf. keras. layers.ドロップアウト(0.1), tf. keras.GlobalAveragePooling1D().

tf. keras. layers.Dense(latent\_size \* 2, activation=None), ])

**def** call(self, inputs).

*"""" "確率的なエンコーディングを返すためにモデルを前方に実行する"""* net = tf. cast(parse\_to\_one\_hot(input), dtype=tf. float32) net = self.multivariateNormalDiag( loc=net[... , :self. latent\_size].

scale\_diag=tf. nn. softplus(net[... , self. latent\_size:]), name="latent\_code\_posterior")

**class ProbabilisticGrammar**(tf. keras. Model).

*""""文法に従う生成物の上の深い生成モデル"""*

*このよう*な場合には、このようにして、「確率論的な文法を構築する」ということになる。"確率的*な文法を構築する""""* super(ProbabilisticGrammar, self).このような場合には、このようにして、このようにして、このようにして、このようにして、このようにして、このようにする。これは、このような場合には、そのようなデータは、そのようなデータを用いてもよいということを意味する。これは、このような場合には、そのようなデータを使用していることを意味する。

**def** call(self, inputs).

*""""* 一連*の生産物を生成するためにモデルを前方に実行します。*

latent\_code = ed.MultivariateNormalDiag(loc=tf. zeros(self. latent\_size).

sample\_shape=1.

name="latent\_code")

state = self. lstm. zero\_state(1, dtype=tf. float32)

t = 0

制作物 = []

stack = [self. grammar. start\_symbol] **while** stack: symbol = stack. pop()

net, state = self. lstm(latent\_code, state)

logits = self. output\_layer(net) + self. grammar. mask(symbol) production = ed.OneHotCategorical(logits=logits, name="production\_"+ スト(t)\_, rhs = self. grammar.production\_rules[tf.argmax(production, axis=1)] **for** symbol **in** rhs.

**もし** symbol が self.**であれば、**文法.

生産物をappend(生産物)

t += 1

**return** tf. stack(productions, axis=1)

詳細はソースコードへのリンクを参照してください。

## B.3 変分推論の中のマルコフ連鎖モンテカルロ法

我々は、構成可能性の別のレベル、すなわち確率的プログラム内での推論を実証する。すなわち、我々はMCMCを適用して、変分推論のための柔軟な分布のファミリーを構築する(Salimans et al. , 2015; Hoffman, 2017)。のナッツ(nuts)で指定された遷移カーネルの連鎖を適用する。

節と図3.12のtrainで指定された変分推論アルゴリズム。

**import nuts**, **train** IMAGE\_SHAPE = (32, 32, 3, 256)

**def** model()。

*"""" 32x32x32x3 8ビット画像の生成モデル""""* decoder\_net = tf. keras.Sequential([ tf. keras. layers.Dense(512, activation=tf. nn. relu), tf. keras. layers.Dense(np. prod(IMAGE\_SHAPE), activation=None), tf. keras. layers.Reshape(IMAGE\_SHAPE).

])

z = ed.Normal(loc=tf. zeros([FLAGS. batch\_size, FLAGS. latent\_size]), scale=tf. ones([FLAGS. batch\_size, FLAGS. latent\_size]), name="z")

x = ed.Categorical(logits=decoder\_net(z), name="x") **return** x

**def** variational(x).

*""""32x32x3の8ビット画像を与えられた変分モデル""""* encoder\_net = tf. keras.シーケンシャル([ tf. keras.レイヤーの再シェイプ(np. prod(IMAGE\_SHAPE)), tf. keras.Dense(512, activation=tf. nn. relu), tf. keras. layers.Dense(FLAGS. latent\_size \* 2, activation=None).

])

net = encoder\_net(x)

qz = ed.Normal(loc=net[... , :FLAGS. latent\_size], scale=tf. nn. softplus(net[... , FLAGS. latent\_size:]), name="qz")

**for** \_ **in** range(FLAGS. mcmc\_iterations).

qz = nuts(current\_state=qz, target\_log\_prob\_fn=**lambda** z: ed. make\_log\_joint(model)(x=x, z=z)) **return** qz

align\_fn = **lambda** name: {'z': 'qz'}. get(name) loss = train(0.1*) # uses model, variational, align\_fn, x in scope.*

## B.4 ノーUターンサンプラー

ベイズロジスティック回帰のためのEdward2プログラムをナッツを用いて実装しています。

**import build\_dataset**

**def** logistic\_regression(features).

*""""ラベルに与えられた特徴量のベイズロジスティック回帰"*

coeffs = ed.MultivariateNormalDiag(loc=tf. zeros(features. shape[1]), name="coeffs") labels = ed. Bernoulli(logits=tf. tensordot(features.Bernoulli(logits=tf. tensordot(features, coeffs, [[1], [0]]) ) **return** labels

**def** make\_target\_log\_prob\_fn()。

*""""log-joint関数を用いて、データに固定された目標密度を作成します"""* log\_joint\_fn = ed. make\_log\_joint\_fn(model) **def** target\_log\_prob\_fn(coeffs): **return** log\_joint\_fn(features=features, coeffs=coeffs, labels=labels)

**戻り値** target\_log\_prob\_fn

features, labels = build\_dataset()

coeffs = tf. random\_normal(feature. shape[1*]) # 初期状態の*サンプル = ed. nuts(current\_state=coeffs, target\_log\_prob\_fn=make\_target\_log\_prob\_fn())

詳細はソースコードへのリンクを参照してください。

# 付録C: 変分推論の応用

## C.1変分 ガウス過程の特殊な場合

vgpの生成過程を様々な方法で制限することで、よく知られているモデルを復元することができます。これにより、vgpの複雑さの背後にある直観的な理解が得られます。最近提案された多くのモデルも、vgpの特殊なケースとみなすことができることを示します。

**例1.***平均場分布の混合物は、カーネルのない*vgpで*ある。*

平均場分布の離散混合物 (Bishop et al. , 1998; Lawrence, 2000) は，潜在変数間の依存性を持つ古典的に研究された変分モデルである．変分データの入力間を補間する写像の代わりに，vgpが潜在入力の*ξ*に対して単に最近接を実行すると仮定すると，最も近い変分入力*sn*に結び付けられた出力*tn*を選択する．これは，変分出力の平均場パラメータの1つをサンプリングする出力の多項分布を誘導する．この[[8]](#footnote-8)ように，入力間を補間するgp先行値を用いると，vgpは，最近傍関数のカーネル密度平滑化とみなすことができる．

**例2.変分***因子分析は、線形カーネルを持つ*vgpで*あり、変分データがない。変分*空間での因子分析(Tipping and Bishop, 1999)を考えてみましょう。 [[9]](#footnote-9)

*ξ* ∼ N(**0***,***I**)である*。 zi* ∼ N(**w**> ξ*,I)。*

潜在入力をマージナル化すると、 **z** の線形依存性が誘導され、 *q*(**z**; **w**) = N(**z**; **0***,***ww**> ) となる。

二重解釈を考える

*ξ* ∼ N(**0***,***I***),fi* ∼ GP(0*,k(-,-)),k*(**s***,***s**0) = **s**> s0. *zi* = *fi*(*ξ*)である*。*

で *q*(**z**|*ξ*) = N(**z**; **0***,ξ*> ) とする．因子分析における**w**の最尤推定値は，gp定式化における*ξ*の最大事後推定値である．より一般的には，非線形カーネルの使用は，**z** において非線形依存性を誘導する．カーネルのハイパーパラメタの集合 *θ を*学習することは，このようにして **z** の潜伏埋め込みで最も変動を捉える集合を学習する (Lawrence, 2005)．

## C.2定理1の 証明

**定理 1.***q*(**z**; *θ,*D) を変分*ガウス過程とする。有限の潜在変数と連続分位関数（逆CDF）を持つ事後分布 p*(**z**|**x**) を*考えてみましょう。次のようなパラメータ列* (*θk,Dk*) が存在する。

lim KL(*q*(**z**; *θk,Dk*)*kp*(**z**|**x**)) = 0.

*k*→∞

*証明。*平均場分布を縮退デルタ分布で与えよう

*q*(*zi* |*fi*) = *δfi*(*zi)。*

潜在入力の大きさを潜在変数の数 *c* = *d に相当するもの*とし、*σ*ard2 を固定する。= 1、*ωj* = 1とする。さらに，簡単のために，*ξはd*次元超立方体上に一様に描かれていると仮定する。そこで，4.2.4節で説明したように，逆事後累積分布関数を*P*-1とすると，次のような最適な*f（f*∗）が得られる。

KL(*q*(**z**;*θ*)*kp*(**z**|**x**)) = 0

でござる

*ｆ*＊（*ξ*）＝*Ｐ*-１（*ξ*１*,...,ξｄ*）である*。*

*j* = 0 から 2*k の*点 *j*/2*k* の集合を *Ok と定義*し、*Sk を Ok の d* 次元積と定義する。*Sk* の各要素 *si* について，ペア (*si,f*∗(*si*)) を含む集合を *Dk と*する．このランダム写像は、ガウス過程のノイズフリー予測特性（Rasmussen and Williams, 2006）により、すべての *si* ∈*Sk* について *fk*(*si*) = *f*∗(*si*) を満たす。そして、連続性により、*k* → ∞となると、*fk は f*∗ に収束します。

分布の分位関数が連続的である大まかな条件は、その分布がリーベスグ量に関して正の密度を持っている場合です。

変分データの有限サイズに対する収束率は，ランダム共変量の下でのgpsの事後収縮率によって調べることができます(Van Der Vaart and Van Zanten, 2011)．この理論が変分法の設定に適用できるようにするためには、事後分位の強い連続性条件とMatern共分散関数を使用するという追加の仮定だけが必要です。

## C. 3変量目的語

式4.6で示されたモデルエビデンス*logp*(**x**)の扱いやすい下界を導出する。これを行うために、まず、期待されるKL項を用いてエルボにペナルティを与える。

*logp*(**x**)≧L = Eqvgp[*logp*(**x**|**z**)] - KL(qvgp(**z)***kp*(**z**))

≧ Eqvgp[*logp*(**x**|**z**)] - KL(qvgp(**z**)*kp*(**z**))- EqvgphKL(*q*(*ξ,f* |**z**)*kr*(*ξ,f* |**z**))i*.*

以下のように、すべての用語を期待値に組み合わせることができます。

Le = *Eq*(**z***,ξ,f*)*hlogp*(**x**|**z**) - *logq*(**z**) + *logp*(**z**) - *logq*(*ξ,f* |**z**) + *logr*(*ξ,f* |**z**)i

= *Eq*(**z***,ξ,f*)*hlogp*(**x**|**z**) - *logq*(**z**|*f*(*ξ*))+ ｍｍ）＋*ｌｏｇｐ*（**ｚ**）-*ｌｏｇｑ*（*ξ，ｆ*）＋*ｌｏｇｒ*（*ξ，ｆ*｜**ｚ**）ｉ*．*

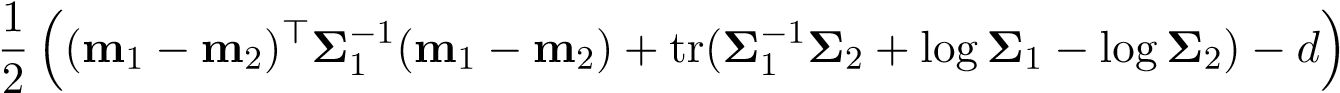
ここで我々は積則 *q*(**z**)*q*(*ξ,f* |**z**) = *q*(**z**|*f*(*ξ*))*q*(*ξ,f*)を適用する.KL発散として項を再結合し,パラメータ(*θ,φ*)で記述することで, 4.3節の自動符号化変分目的語を回復する.

Le(*θ,φ*) = Eqvgp[*logp*(**x** | **z**)] - EqvgphKL(*q*(**z**|*f*(*ξ*))*kp*(**z**))i

- EqvgphKL(*q*(*f* |*ξ*; *θ*)*kr*(*f* |*ξ,***z**; *φ*)+ *logq*(*ξ*) - *logr*(*ξ* |**z**)i*.*

平均場 *q*(**z**|*f*(*ξ*)) とモデル先行値 *p*(**z**) の間の KL 発散は，ある一般的なモデルでは解析的に扱いやすい．例えば，深層潜在ガウスモデル (Rezende et al. , 2014) とdraw (Gregor et al. , 2015)では，平均場分布とモデル先行値の両方がガウス分布であり，分析的KL項を導く：次元*d*のガウスランダム変数に対して．

KL(N(**x**;**m**1*,***Σ**1)kN(**x**;m2*,***Σ**2)) = = 0.5%である。

*.*

一般に、KLが難解な場合は、KL項と再構成項を組み合わせて、分散目的量を最大にする

Le(*θ,φ*) = Eqvgp[*logp*(**x***,***z**) - *logq*(**z**|*f*(*ξ*)]

(C.1)

hi - Eqvgp KL(*q*(*f* |*ξ*; *θ*)*kr*(*f* |*ξ,***z**; *φ*)+ *logq*(*ξ*) - *logr*(*ξ* |**z***) .*

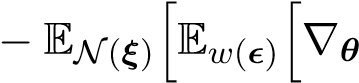
これは、最適化の際に確率的な勾配の中でわずかに高い分散を経験すると予想されます。

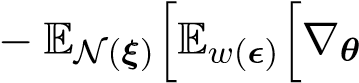
ここで2番目の項を検討する．我々は補助モデルを完全因数分解ガウスであるように指定することを思い出してください, *r*(*ξ,f* |**z**) = N((*ξ,f*(*ξ*))> |z; **m**,S*),* ここで m **∈R**c+d*, S* **∈**Rc+d*.*さらに，変分プリオール*q*(*ξ*)と*q*(*f* |*ξ*)はともにガウス分布であることが定義されている．したがって、ガウス分布のランダム変数間のKL発散でもあります。同様に、*logq*(*ξ*)-*logr*(*ξ* |**z**)は、単にガウスの対数密度の差です。2番目の式は計算が簡単で、勾配を逆伝播させることができます。

## C.4変分 目的語の勾配

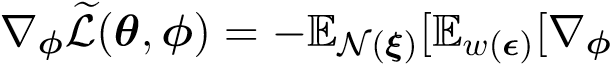
我々は変分目的語の勾配を導出する (式 4.7)。これは，バックプロパゲーションを用いることで簡単に実現できます．



ii KL(*q*(**z**|**f**(*ξ*;*θ*))*kp*(**z**))

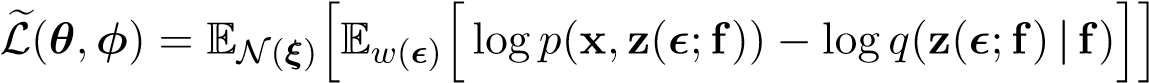
二

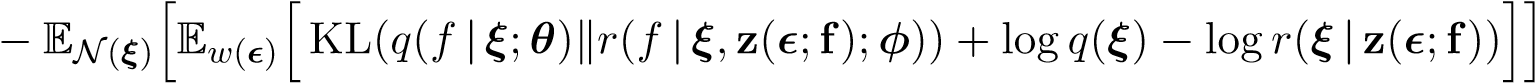
KL(*q*(*f* |*ξ*;*θ*)*kr*(*f* |*ξ,***z**;*φ*)) *,*

KL(*q*(*f* |*ξ*;*θ*)*kr*(*f* |*ξ,***z**;*φ*))- ∇*φ logr*(*ξ*｜**z**;*φ*)]である*。]*

ここで, KL項は付録C.3から解析的に記述されており, その計算グラフを通して同様に勾配が伝搬されると仮定します.実際には, 期待値に注意するだけでよく, 上に書いた関数の勾配は自動微分ツールを用いて処理される.

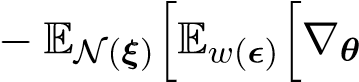
我々はまた，式C.1の一般的な変分境界の勾配を導出し，最初のKL項（*q*と*p*の先行値との間の発散を測定する）が必ずしもトラクショナルではないことを仮定している．第4.3.3節で説明した再パラメータ化に続いて, この変分目的は次のように書き換えることができる.



*.*

ネストされた再パラメータ化をバックプロパゲーションすることで勾配を計算します。



ii KL(*q*(*f* |*ξ*; *θ*)*kr*(*f* |*ξ,***z**; *φ*)

KL(*q*(*f* |*ξ*;*θ*)*kr*(*f* |*ξ,***z**;*φ*))- ∇*φ logr*(*ξ*｜**z**;*φ*)]である*。]*

## C.5 変量データのサイズのスケーリング

大規模なサイズの分散データが必要な場合、例えば、*m* × *m* の行列の反転による立方的な複雑さが計算中のボトルネックになる場合、さらにスケールアップすることができます。分散入力をグリッド上に固定することを考えてみましょう。静止カーネルの場合、これにより、高速な *m* × *m* 行列の反転のために Toeplitz 構造を利用することができます。特に，Toeplitz 行列を循環行列に埋め込み，共役勾配を高速フーリエ変換と組み合わせて適用することで，O(*mlogm*)の計算とO(*m)*のストレージで逆行列ベクトル積を計算することができます (Cunningham et al. , 2008)．プロダクトカーネルについては、さらにKronecker構造を利用して、O(*Pm*1+1*/P* )演算とO(*Pm*2*/P* )ストレージで高速な*m×mの*行列反転を可能にすることができ、ここで*P > 1は*カーネルプロダクトの数です(Osborne, 2010)。ard カーネルは，特に *O*(*cm*1+1*/c*) の複雑さをもたらします．

# 付録D.確率モデルでの応用

## D.1 ノイズと潜在変数の関係

Himsは、各データ点について2つのランダム性の源を持っています：潜在変数*zn*とノイズ*n*；これらのランダム性の源は、*xnを*生成するために変換されます。ベイズ分析は，潜在変数の事後を推論する．自然な疑問は，ノイズの事後も推論すべきかどうかということである．

事後処理の形状、そして最終的にそれが意味のあるものであるかどうかは、ノイズの次元性と変換によって決定される。例えば，局所的な潜在変数を持たない gan モデルを考えてみましょう．

**x**.条件は点群であり、完全に*n*によって決定されます。*g*(-; *θ*) が注入的な場合、事後処理も点群です。

*,*

これは、ランダム性の注入関数（ノイズと潜在変数の両方）については、「事後」は決定論的な隠れ表現として分析する価値があるかもしれないことを意味します（Donahue et al.

点質量事後処理は、非線形最小二乗法を用いて求めることができます。非線形最小二乗法は、反復アルゴリズム

ˆ *,*

のステップサイズ列*ρt*に対して.*f の*勾配がゼロの場合、更新は動かなくなることに注意してください。しかし，*f の*注入的性質により，繰り返しの正しさをチェックすることができます（単純に 

## D.2 エドワードの暗黙のモデル例

我々は、Edward (Tran et al. , 2016a)における例示的な実装を介して暗黙のモデルを実証する。

図D.1は2層の深層暗黙モデルを実装している。これは、ニューラルネットワークを定義するためにtf.layersを使用します：tf.layers.dense(x, 256)は、256個の隠れユニットと入力*xを*持つ完全に接続された層を適用します；重みとバイアスパラメータは、ユーザから抽象化されます。プログラムは、暗黙の潜在変数**z***n,*1*,zn,*2の2つの層を用いて、暗黙の尤度を用いて、*N個の*データ点*xn*∈*R*10を生成します。

図D.2は、分類のためのベイズガンを実装しています。これは手動で2層のニューラルネットワークを定義しており、各データインデックスについて、入力としてノイズ*n*∈Rと結合された特徴*xn*∈R500を取ります。出力は、最後の層の符号で与えられたラベル *yn* ∈ {-1*,*1} です。我々は、すべての重みとバイアスの上に標準的な正規の優先順位を置きます。プレースホルダ **X** ∈RN×500 を与えながらこのプログラムを実行すると、ラベル **y** ∈ {-1*,*1}*N の*ベクトルが生成されます。

*# ランダムノイズは Normal(0, 1)*

eps2 = 正規(tf. ゼロ([N, d]), tf. もの([N, d]) eps1 = 正規(tf. ゼロ([N, d]), tf. もの([N, d]) eps0 = 正規(tf. ゼロ([N, d]), tf. もの([N, d]))

# z2 = tf. layers. dense(eps2, 128, activation=tf. nn. relu) h2 = tf. layers. dense(z2, 128, activation=tf. nn. relu) h2 = tf. layers.

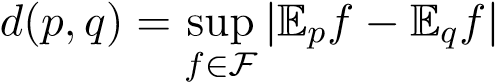
dense(tf. concat([eps1, h2], 1), 128, activation=tf. nn. relu) h1 = tf. layers. dense(z1, 128, activation=tf. nn. relu) x = tf. layers. dense(tf. concat([eps0, h1], 1), 10, activation=None)

**図D.1:** データ点*xn*∈R10のための2層の深層暗黙モデル。このアーキテクチャでは、確率的層と決定論的層が交互に存在する。確率的な層を定義するために、我々は単に

ニューラルネット層の入力に連結します。

## D.3 KLの独自性

積分確率メトリックは、2つの分布*p*と*qの*間の距離を測定します。

*.*

積分確率メトリックは、生成モデルにおけるパラメータ推定に使用されてきた(Dziugaite et al. , 2015)と、牽引可能な密度を持つモデルにおける変分推論に使用されてきた(Ranganath et al. , 2016b)。

**edward.modelsからtfとしてtensorflow**をイン**ポートします。**

*# 重みとバイアスはNormal(0, 1)の先行値を持つ*

W1 = Normal(tf. zeros([501, 256]), tf. ones([501, 256])W2 = 正規(tf. ゼロス([256, 1]), tf. オネエ([256, 1]) b1 = 正規(tf. ゼロス(256), tf. オネエ(256)) b2 = 正規(tf. ゼロス(1), tf. オネエ(1))

*# ニューラルネットワークへの入力を設定する* X = tf. placeholder(tf. float32, [N, 500]) eps = Normal(tf. zeros([N, 1]), tf. ones([N, 1]) )

*# y = neural\_network([x, eps])* input = tf. concat([X, eps], 1) h1 = tf. nn. relu(tf. matmul(input, W1) + b1) h2 = tf. matmul(h1, W2) + b2.

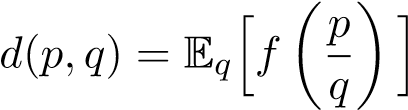
y = tf. reshape(tf. sign(h2), [-1])*# take sign, then flatten*

**図D.2:** 分類のためのベイズガン、入力として**X**∈RN×500を取り、ラベルのベクトル**y**∈{-1*,*1}*Nを*生成する。ニューラルネットワークは確率分布をパラメータ化するのではなく、直接データを生成する。

局所的な潜在変数のみを持つモデルとは対照的に，事後を推論するためには，それと変分近似との間の積分確率メトリックが必要です．直接的なアプローチは，事後処理からのサンプリングが困難であるため，失敗する．

間接的アプローチは、Steinの方法に基づいて事後期待値ゼロを持つ十分に広いクラスの関数を構築することを必要とする(Ranganath et al. , 2016b)。これらの構築には尤度関数とその勾配が必要です。尤度を回避するには、ノンパラメトリック密度推定の形式を必要とします。比推定とは異なり、高次元に十分にスケールする解はありません。

発散の別のクラスとして、*f*発散は

*.*

積分確率メトリックとは異なり、*fの*発散は当然のことながら比推定に寄与し、暗黙の*p*と暗黙の*qを*可能にします。階層モデルでデータをサブサンプルするためには、f*が定数f*(*ab*) = *f*(*a*)+*f*(*b*)を満たす必要があります。連続関数の場合、これは対数関数の定義的性質です。このことは、*q*から*p*へのKL発散が、我々の求めているサブサンプリング手法が可能な唯一の*f*発散であることを示唆しています。

## D.4 Hingeの損失

4節の対数損失と同様に*r*(*xi,zi,β*; *θ*)は実数を出力するとします。ヒンジ損失は

Dhinge = *Ep*(*xn,zn* |*β*)[max(0*,*1 - *r*(*xn,zn,β*;*θ*)]+.

*Eq*(*xn,zn* |*β*)[max(0*,*1 + *r*(*xn,zn,β*;*θ)]。*

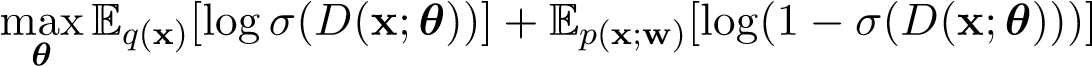
この損失関数は偏りのない勾配に従うことで最小化されます。勾配は対数損失の場合と同様に計算します。最適な*r*∗は対数比です。

## D.5 MAPを用いたベイジアンGANとMLEを用いたGANの比較

セクション4では、ベイジアン・ギャンを用いたMAP推定は離散データの解析を可能にしますが、ギャン（最尤目的（Goodfellow, 2014）であっても）は不可能であることを主張しました。これは驚くべき結果です。MAPの事前分布が平坦であると仮定すると、両者は最終的には同じ目的を最適化しています。

下の2つを比較してみます。

ギャンについては，判別器がロジット確率を出力すると仮定して，[0*,*1]上ではなく，制約を受けないようにします．

*.*

彼らは生成問題を使用しています

**minw** *Ep*(**x**; **w**)[-exp(*D*(**x***)].*

再パラメータ化勾配を用いた生成問題を解くには、モデルから生成されたデータ、**x** ∼ *p*(**x**; **w**)をバックプロパゲーションする必要があります。これは離散的な**xで**は不可能です。さらに、指数化はこの目的語を数値的に不安定にし、実際には使用できません。

これをMLE（MAPと平坦な先行値）を持つベイズ・ガンと対比する．これは，点質量分散近似 *q*(**w**0) = I[**w**0 = **w**]を適用する．これはエルボを最大化する．

*N*

*maxEq*(**w**)[*logp*(**w**) - *logq*(**w**)] + *Xr*(*xn,***w***).*

**ｗ***ｎ*＝１

第1項は、平坦な事前分布*p*(**w**)∝1と点質量近似の場合はゼロであり、問題は以下のようになります。

*N maxXr*(*xn,***w***).*

**w**

n=1

これは離散 **x に対して解くことができます**: それは **w** に対して *r*(**x***,***w**) を通る勾配をバックプロパゲーションすることだけを必要としますが、これはすべて微分可能です。さらに、この目的は数値的に不安定な指数化を必要としません。

結局のところ、違いは比率推定器の役割にあります。ベイズ型ガンでは、比推定問題を使用することを思い出してください。

Ｄｌｏｇ＝*Ｅｐ*（**ｘ**；**ｗ**）［-ｌｏｇ*σ*（*ｒ*（**ｘ***，***ｗ**；*θ*））］＋Ｅｐ（ｘ，ｗ）［-ｌｏｇ*σ（ｒ（***ｘ***，ｗ*；*θ*）］＋Ｄｌｏｇ

*Eq*(**x**)[-log(1 -*σ*(*r*(**x***,***w**;*θ))]。*

最適な比推定量は、対数比 *r*∗(**x***,***w**) = *logp*(**x**|**w**) - *logq*(**x**) である。これを **w に対して最適化**すると、対数尤度 *logp*(**x**|**w**) を最適化することになります。MLE を持つガンに対する最適な判別器は同じ比率、*D*∗(**x**) = *logp*(**x**; **w**) - *logq*(**x**) を持ちますが、これは **w** に関する定数関数です。別の方法としては、重要度サンプリングを使用する方法がありますが、結果は前者の目的になります（Goodfellow, 2014）。

## D.6比率推定器の安定性

暗黙モデルでは、標準的なKL変分推論との違いは、比推定問題にある。したがって、比推定器の精度を評価したい。これを確認するためには， 扱いやすい尤度を持つモデルの下での真の比と比較することができる．

我々はベイズ線形回帰を適用する。これは、我々の分析に活用する、扱いやすい事後処理を特徴としている。

0

50

100

150

200

250

のトレーニング反復

*q*

(

*β*

;

*λ*

)

1200

1000

800

600

400

200

0

E

s

t

i

m

a

t

e

o

f

l

o

g

*q*

(

*x*

)

真の比率と推定比率の比較

0

50

100

150

200

250

のトレーニング反復

*r*

(

*ｘ，β*

;

*θ*

)

2500

2000

1500

1000

500

0

E

s

t

i

m

a

t

e

o

f

l

o

g

*q*

(

*x*

)

真の比率と推定比率の比較

0

50

100

150

200

250

のトレーニング反復

*r*

(

*ｘ，β*

;

*θ*

)

120

100

80

60

40

20

0

E

s

t

i

m

a

t

e

o

f

l

o

g

*q*

(

*x*

)

真の比率と推定比率の比較

**図 D.3: (左)** *q* のステップに対する比の差。*y* 軸の分散が小さいほど安定していることを意味する。興味深いことに，*q が*事後に収束するほど，比率推定器はより正確で安定している．**(中)** *q が*ランダムな初期化時に固定されている。多くのステップを踏んでも比推定量は改善されない。(**右)** *r* のステップ数に対する比の差、q *は事後的に*固定されている。比推定器はランダム初期化から数ステップで高精度になる。

模擬データ点{*yn*∈R2*,xn*∈R}を50個使用する。最適な（対数）比は

*r*∗(**x***,β*) = *logp*(**x**|*β*) - *logq*(**x***)。*

対数尤度*logp*(**x**|*β*)-*usr*∗(**x***,β*)は定数である経験分布*Pn logq*(*xn*)と等しくない。したがって、比推定量*rが*正確であれば、*logp*(**x**|*β*)との差は、*β*の値にわたって分散の小さい定数でなければならない。

図D.3を参照のこと。上のグラフは、変分近似 *q*(*β*) の更新に対する *logq*(**x**) の推定値を表示している；各推定値は現在の *q*(*β*) のサンプルを使用している。各推定値は現在のq(β)からのサンプルを使用しています。*qが*正確に事後に収束するので、比推定量*rの方*がより正確です。これは我々の直感と一致します：モデルから生成されたデータが真のデータに近い場合、比の推定はより安定しています。

図D.3の別の仮説は、レシオ推定器は単に訓練中に情報を蓄積しただけだというものである。これは真実ではないことが判明した。左側では、*qが*ランダムな初期化で固定されており、*logq*(**x**)の推定値が*r*の更新にわたって表示されている。対照的に、右は、*qが*事後的に固定されている場合の同じ手順を示しています。

訓練のための実用的な洞察がいくつかある。第一に、*qを*更新する前に*rを*複数回更新することは（少なくとも初期の反復計算では）役に立ちません。さらに、指定されたモデルがデータとあまり一致しない場合、すべての反復にわたって訓練することは困難です。

変分近似が改善されるほど比推定がより正確になるという性質は、*q*(*xn*)が経験分布に設定されているからです（式4のエルボから任意の密度q(xn)を差し引くことができることに注意してください）。(式4のエルボから任意の密度*q*(*xn*)を差し引くことができることに注意)。尤度フリーの変分推論は、観測されたデータを*p*(*xn* |*β*)の下で可能性の高いものにする*q*(*β*)、すなわち、*q*(*β*)からサンプリングされた値で*p*(*xn* |*β)*が経験分布に近づくq(*β)を*見つけます。*q*(*xn*)を経験分布とすると、*q*(*β*)が向上するにつれて、比推定問題はより簡単に解くことができなくなる（したがって、より正確になる）ことを意味します。

これは、暗黙のグローバル変数と暗黙のグローバル変分近似を可能にするために、なぜ我々が比に*p*(*β* |**x**)の推論を含まないのかという動機にも注意してください。つまり，推定では，事前分布と事後分布の間のサンプルを比較する必要があり，グローバル変数では，それらが重なることはほとんどありません．

1. 密度で表すこともできます。 [↑](#footnote-ref-1)
2. 計算を操作するために、ユーザ制御によるトレーシング（上記の*トレーサー*関数）を必要とする。これは現在のところ TensorFlow Eager や Autograd (Maclaurin et al. , 2015)では利用できないものであり，これが我々の実装の動機となった． [↑](#footnote-ref-2)
3. *p*(*x*)∝exp{*f*(*x*)}がありますが、ここでは閉形ではサンプリングすら利用できません。

   make\_log\_joint5原理的には、サンプリングの観点からどのようなモデルでも再定義することができますが、この例のように実際には必ずしも効率的に行われるわけではありません。対照的に、逆プログラムmake\_log\_jointは、その密度を得るためのものです。しかし。 [↑](#footnote-ref-3)
4. エドワードとは異なり、エドワード2は学習アルゴリズム上の分布を指定することもできます。 [↑](#footnote-ref-4)
5. GPU の Theano と CPU の NumPy との間で頻繁に通信を行います。 [↑](#footnote-ref-5)
6. *rの*推論ネットワークに*xn*と*znの*両方を入力とさせていることにも注意してください。 [↑](#footnote-ref-6)
7. 比*rは*間接的に*φ*に依存するが、その勾配w.r.t.*φは*消滅する。これは、スコア関数の同一性と積則を介して導出される（例えば、Ranganathら（2014年、付録）参照）。 [↑](#footnote-ref-7)
8. ，変分的な入出力ペア{(*sn,tn*)}が与えられると，最近傍関数は*f(ξ) = tj*のように定義され，**その**ようなk*ξ* - sjk *< kξ -* skkがすべてのkについて存在する． [↑](#footnote-ref-8)
9. 簡単にするために、ここではvgpの基礎となる平均場分布の議論を避けます。 [↑](#footnote-ref-9)