Variational Dropout and the Local Reparameterization Trick

Diederik P.Kingma, Tim Salimans and Max Welling

発表者 鈴木雅大

本論文について

- 発表学会不明
  - Submitted on 8 Jun 2015(arXiv)
  - 7/17現在まだ書き終わってないっぽい（結構説明が抜けてたりする）
- 最近よく名前を聞く「変分オートエンコーダー」シリーズの新作
  - 要約すると「Dropout = local reparameterization trickだった！！」っていう論文
  - 当然ながら論文には図がほとんどありません！
  - 抽象的な議論なのでほとんど数式な上、難解
- 今回は元となる確率的勾配変分ベイズ(SGVB)の説明から始めます
  - EMや変分ベイズは大体わかっている前提で話します

※この資料は誤解と偏見であふれています（変なとこあったら訂正お願いします）

目次

- EMアルゴリズムと変分ベイズ
- 確率的勾配変分ベイズ(SGVB)の説明
- Variational Dropout and the Local Reparameterization Trick

目次

- EMアルゴリズムと変分ベイズ
- 確率的勾配変分ベイズ(SGVB)の説明
- Variational Dropout and the Local Reparameterization Trick

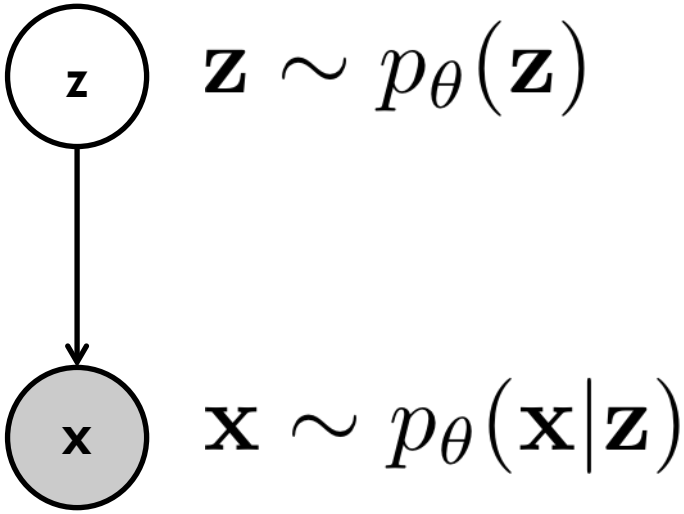
EMアルゴリズムと変分ベイズ

- EMアルゴリズム
  - 尤度 $p(x|\theta) = \int p(x, z|\theta)dz$ を最大化
  - 最尤推定（パラメータ推定）
    - 下界を求めて、q(z)とθについて最大化
- 変分ベイズ
  - 周辺尤度（エビデンス） $p(x) = \int p(x, z)dz$ を最大化
  - ベイズ推定（分布推定）
    - q(z)について平均場近似（因子分解）
    - 下限を求めて、それぞれのq(z)について最大化

目次

- EMアルゴリズムと変分ベイズ
- 確率的勾配変分ベイズ(SGVB)の説明
- Variational Dropout and the Local Reparameterization Trick

問題設定



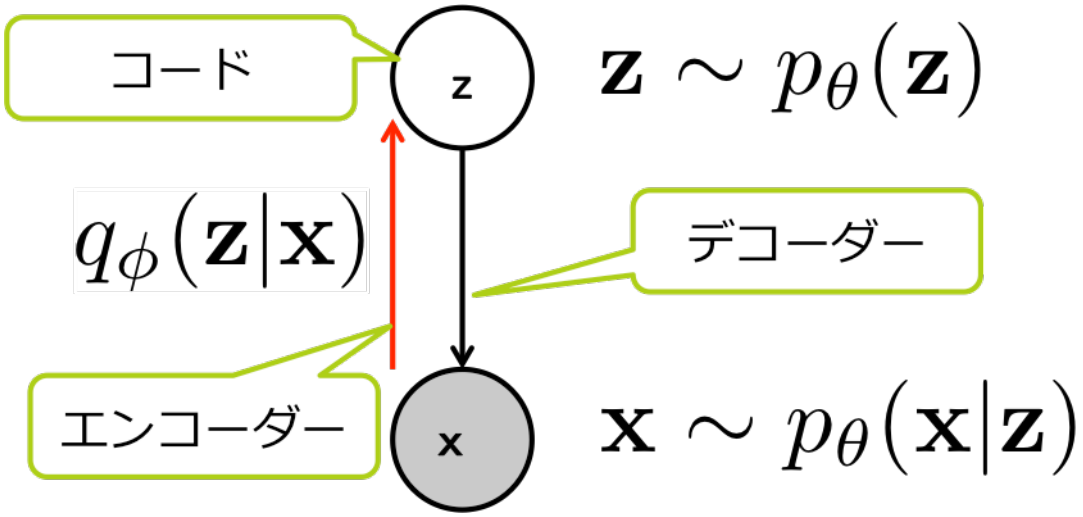
- データセットからこの分布を学習する
  - zは潜在変数、θは分布のパラメータ
  - p(z)とp(x | z)は微分可能
- ただし分布の形は限定しない

問題点

- 周辺分布p(x)が扱いづらい
  - $p(x)=\int p(z)p(z|x)dz$
  - これだけなら、割とよくある問題
- p(z|x)も困難
  - $p(z|x)=p(x|z)p(z)/p(x)$
  - EMアルゴリズムが使えない！
- データがたくさんあるので、サンプリングだと時間がかかる！
  - 解析的に求めたい

→これらを解決する、より一般的なアルゴリズムを作りたい！

「認識モデル」の導入



- q(x | z)という分布を考える
  - p(x | z)は困難なので、別の分布で置いて真の分布に近づける
  - この考え方は変分ベイズでもあるが、因子分解の仮定を置かない
- 求める分布パラメータはφとθ

下限の導出

- 一般的な下限の導出の流れと同じ

$$
\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&= \log \int q_\phi(\mathbf{z}|\mathbf{x}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \quad (\because \text{イェンセンの不等式}) \\
&= \mathcal{L}(\theta, \phi; \mathbf{x})
\end{aligned}
$$

$\mathcal{L}(\theta, \phi; \mathbf{x})$を下限、もしくは変分下限とよぶ

下限の計算

- 下限部分は次のように計算できる

$$
\begin{aligned}
\mathcal{L}(\theta, \phi; \mathbf{x}) &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} + \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z} \\
&= -\int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})} d\mathbf{z} + \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z} \\
&= -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z})) + \underline{E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}
\end{aligned}
$$

期待値部分は解析解を求めることができない！

モンテカルロサンプリング

一般的にサンプリングによって期待値 $E_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})]$ は次のように求まる

1. $q_\phi(\mathbf{z}|\mathbf{x})$ からサンプリングして $\{\mathbf{z}^{(l)}\}_{l=1}^{L}$ を得る.
2. $\mathbf{z}^{(l)}$ を使って, 次のように期待値を求める.

$$E_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] \simeq \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)})$$

reparameterization trick

- サンプリング $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$が微分可能な関数 $g(\epsilon, \mathbf{x})$ から決定論的に求まると考える

reparameterization trick

$$\mathbf{z} = g(\epsilon, \mathbf{x})$$

ただし ϵは $p(\epsilon)$から生成されるノイズ

- よって、期待値は次のように求まる

$$\begin{aligned}
E_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] &= \int q_\phi(\mathbf{z}|\mathbf{x}) f(\mathbf{z}) d\mathbf{z} \\
&= \int p(\epsilon) f(\mathbf{z}) d\epsilon \quad (\because q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} = p(\epsilon) d\epsilon) \\
&= \int p(\epsilon) f(g(\epsilon, \mathbf{x})) d\epsilon \\
&= E_{p(\epsilon)}[f(g(\epsilon, \mathbf{x}))] \simeq \frac{1}{L} \sum_{l=1}^{L} f(g(\epsilon^{(l)}, \mathbf{x}))
\end{aligned}$$

ただし $\epsilon^{(l)} \sim p(\epsilon)$

確率的勾配変分ベイズ(SGVB)

- よって下限の推定量は次のようになる

$$\hat{\mathcal{L}}(\theta, \phi; \mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L} \log p(\mathbf{x}^{(l)}|\mathbf{z}^{(l)}, \theta)$$

$$\text{ただし } \mathbf{z}^{(l)} = g(\epsilon^{(l)}, \mathbf{x}), \epsilon^{(l)} \sim p(\epsilon)$$

この推定量を確率的勾配変分ベイズ(SGVB)推定量と呼ぶ

- 第1項が負のreconstruction error、第2項を正規化項と見なせる

確率的勾配変分ベイズ(SGVB)

- 全データ（データ数N）からランダムに抽出したミニバッチ（データ数M）が与えられたとき、全データの下限はSGVB推定量から次のように求まる

$$\mathcal{L}(\theta, \phi; \mathbf{X}) \simeq \hat{\mathcal{L}}^M(\theta, \phi; \mathbf{X}^M) = \frac{N}{M} \sum_{i=1}^{M} \hat{\mathcal{L}}(\theta, \phi; \mathbf{x}_i)$$

  - ミニバッチ数が多いとき(M=100)、サンプル数Lは1でいいことが実験によりわかっている
  - よってバッチあたりのサンプル数は1回でよい

- この式は微分可能なので、通常の最適化手法（SGDとか）θやφを最大化することができる

確率的勾配変分ベイズ(SGVB)のアルゴリズム

- 全体の流れ：
  1. データセットからM個のミニバッチをランダムに抽出
  2. ノイズをランダムにサンプリング
  3. 勾配 $\nabla_{\theta,\phi}\widetilde{\mathcal{L}}^{M}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{X}^{M},\boldsymbol{\epsilon})$ を求める
  4. 勾配によってθとφを更新（エンコードとデコードを同時に学習できる）
  5. 収束するまで繰り返し

SGVBの例：変分オートエンコーダー

エンコーダーやデコーダーがニューラルネット

- 事前分布　$p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$
- デコーダー
  - ガウス分布

$$\log p(\mathbf{x}|\mathbf{z}) = \log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I})$$
$$\text{where } \boldsymbol{\mu} = \mathbf{W}_4 \mathbf{h} + \mathbf{b}_4$$
$$\log \boldsymbol{\sigma}^2 = \mathbf{W}_5 \mathbf{h} + \mathbf{b}_5$$
$$\mathbf{h} = \tanh(\mathbf{W}_3 \mathbf{z} + \mathbf{b}_3)$$

- エンコーダー
  - ガウス分布（上の式のzの部分をxにする）
  - よってエンコーダのreparamaterization trickは次のようになる

$$\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}, \boldsymbol{\epsilon}^{(l)}) = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)}$$
ただし $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$

SGVBの利点

- ベイズ推定なので、非常にロバスト
- 生成・認識を同時に学習
  - 推論が自在
  - Hintonのヘルムホルツマシンが近い
- 推論が非常に高速
  - MCMC等がいらない。分布からのサンプリング（しかも1回）のみ
- 従来の最適化の方法をそのまま使える
- 既存の生成モデルに適用可能
  - 従来の生成モデルより深い知識を獲得できる
  - 明示的にモデル化できる＋Deep Learningの深さ
  - 時系列モデルへの応用（動的ベイジアンネット：人の注意のモデル化など）

目次

- EMアルゴリズムと変分ベイズ
- 確率的勾配変分ベイズ(SGVB)の説明
- Variational Dropout and the Local Reparameterization Trick

ベイズ的な識別モデル

- データセット$(\mathbf{x}, \mathbf{y})$が与えられたとき、識別モデル$p(\mathbf{y}|\mathbf{x}, \mathbf{w})$のパラメータ$\mathbf{w}$を学習する
  - ベイズ的なアプローチでは、あらかじめ信念として事前分布$p(\mathbf{w})$が与えられ、データによって信念が更新されると考える

$$p(\mathbf{w}|\mathcal{D}) = p(\mathbf{w})p(\mathcal{D}|\mathbf{w})/p(\mathcal{D})$$

  - しかしこの事後分布は扱いづらいので、$q_\phi(\mathbf{w})$という分布を考え、この分布を事後分布に近づけることを考える
  - つまりKLダイバージェンス$D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w}|\mathcal{D}))$を最小化する
  - この計算は変分下限を最大化することで求まる（詳しい話は省略）

$$\mathcal{L}(\phi) = -D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w})) + L_\mathcal{D}(\phi)$$

$$\text{where} \quad L_\mathcal{D}(\phi) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathbb{E}_{q_\phi(\mathbf{w})}\left[\log p(\mathbf{y}|\mathbf{x}, \mathbf{w})\right]$$

識別モデルのSGVB

- SGVBの手法によって、下限の期待値部分（期待対数尤度）は次のように計算できる

$$L_{\mathcal{D}}(\phi) \simeq L_{\mathcal{D}}^{\mathrm{SGVB}}(\phi) = \frac{N}{M} \sum_{m=1}^{M} \log p(\mathbf{y}^m | \mathbf{x}^m, \mathbf{w} = f(\epsilon, \phi))$$

ただし $\epsilon \sim p(\epsilon)$

- よって、これを含めた下限をφについて偏微分することで計算できる

SGDにおける分散の影響

- 確率的勾配降下法（SGD）は勾配の分散が大きすぎると、いくら時間をかけてもよい解にならない
- ここで期待値対数尤度 $L_{\mathcal{D}}^{\mathrm{SGVB}}(\phi) = \frac{N}{M}\sum_{i=1}^{N} s_i L_i(\epsilon, \phi)$ の分散の上限を確認する

ただし $L_i(\epsilon, \phi) = p(\mathbf{y}^i | \mathbf{x}^i, \mathbf{w} = f(\epsilon, \phi))$

$$\mathrm{Var}_{\epsilon,S}\left[L_{\mathcal{D}}^{\mathrm{SGVB}}(\phi)\right] \leq \underbrace{\frac{N}{M}\left(\sum_{i=1}^{N}\frac{L_{\mathcal{D}}(\phi)^2}{N^2} + \mathrm{Var}_{\epsilon}\left[L_i(\epsilon,\phi)\right]\right)}_{\text{ミニバッチ数Mの影響}} + \underbrace{2\sum_{i>j}\mathrm{Cov}_{\epsilon}\left[L_i(\epsilon,\phi), L_j(\epsilon,\phi)\right]}_{\text{データ間の対数尤度の共分散}}$$

- ノイズεは、ミニバッチの各データ毎にサンプリングしているわけではないので、共分散の部分は正になる

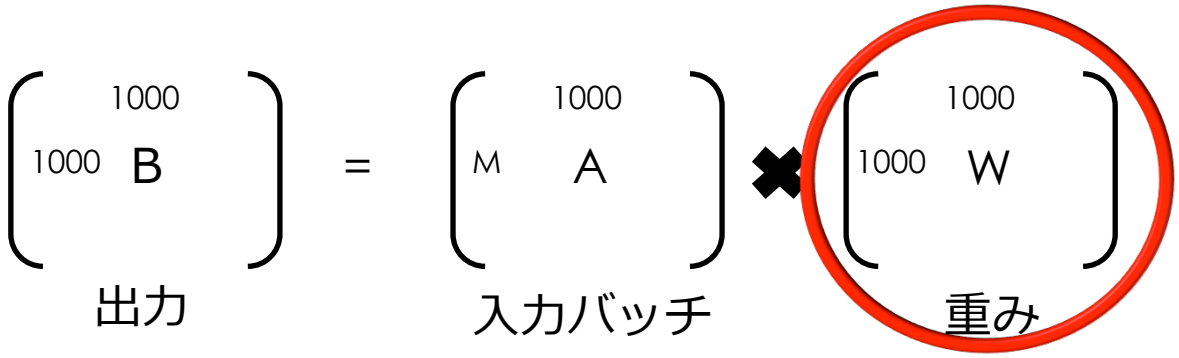
→分散はバッチ数が大きくても共分散部分、すなわちεに影響される！

local reparameteraization trick

- 結局問題は、εを直接サンプリングしていたこと
- 解決策：εをサンプリングするのではなく、各データに依存するf(ε)でサンプリングすればいい
  - そうすれば、データ全体でのグローバルな曖昧さをデータ毎のローカルな曖昧さに落とし込めるので、共分散は0になる！
- このように、グローバルなノイズをローカルなノイズに落とし込むことをlocal reparameterization trickと呼ぶ
- すごく分かり辛いので、論文に載っている例を説明します・・・

local reparameteraization trickの例

- 次のような単純なニューラルネットを考える



- 事前分布 $q_\phi(w_{i,j}) = N(\mu_{i,j}, \sigma^2_{i,j})$ から普通のreparameteraization trickをするとつぎのようになる

$$w_{i,j} = \mu_{i,j} + \sigma_{i,j}\epsilon_{i,j}, \text{ with } \epsilon_{i,j} \sim N(0,1)$$

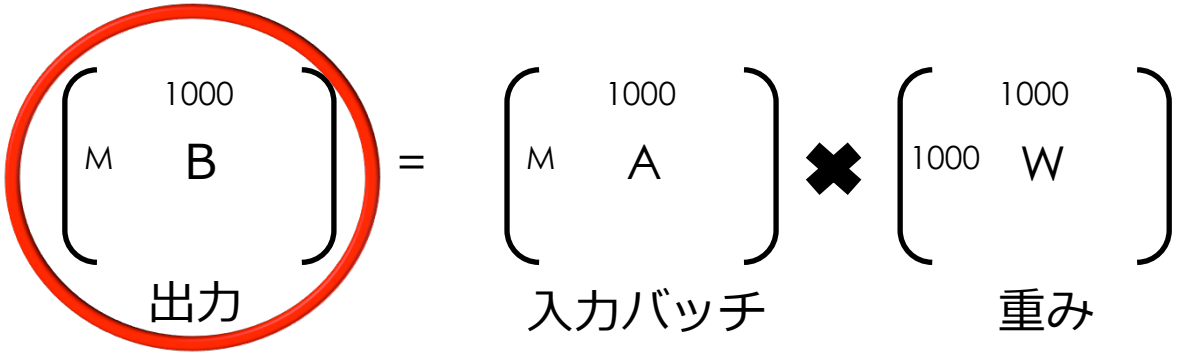
- もし共分散を0にしたかったら、全てのバッチで全ての重みをサンプリングしなければならない

すなわち1000✕1000✕M回！！

  - 実際のネットワークはもっと複雑なので、分散処理も難しい

local reparameteraization trickの例

- 次のような単純なニューラルネットを考える



- 今度は、Bからサンプリングすることを考える

$$q_\phi(b_{m,j}|\mathbf{A}) = N(\gamma_{m,j}, \delta_{m,j})$$

ただし $\gamma_{m,j} = \sum_{i=1}^{1000} a_{m,i}\mu_{i,j}$, $\delta_{m,j} = \sum_{i=1}^{1000} a_{m,i}^2\sigma_{i,j}^2$

- このようなlocalなreparameteraization trickは次のようになる

$$b_{m,j} = \gamma_{m,j} + \sqrt{\delta_{m,j}}\zeta_{m,j}$$

ただし $\zeta_{m,j} \sim N(0,1)$

- この場合、共分散を0にするためのサンプル数はM✖1000回で済む！！

→localであることによって計算が少なくてすみ、分散も小さくなる

変分ドロップアウト

- ドロップアウト：ニューラルネットの汎化性能を上げるテクニック
  - 最適化の際に、次のように各層にノイズを加える

$$\mathbf{B} = (\mathbf{A} \circ \xi)\theta \qquad \text{ただし } \xi_{i,j} \sim p(\xi_{i,j})$$

  - ノイズの分布としてベルヌーイ分布が知られている（ノイズが0または1）
  - また、ガウス分布による方法も同等以上となる

$$\xi_{i,j} \sim N(1, \alpha) \quad \text{ただし} \alpha = p/(1-p)$$

- ドロップアウトを今回の変分アプローチの元で再解釈

⇩

変分ドロップアウト

  - 応用例として、データに適応するようなドロップアウトの割合pを決定できたりする

independent weight noise

- ノイズξがガウス分布N(1,α)から独立に生成されるとすると、bの周辺分布もガウス分布となる

$$q_\phi(b_{m,j}|\mathbf{A}) = N(\gamma_{m,j}, \delta_{m,j})$$

  - Wang and Manning (2013) と同じ
  - ただしこの場合、Bの異なる要素の依存関係を無視している

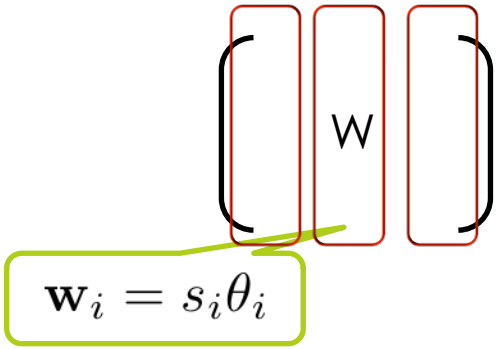
- 先ほどのようにB=AWを考え、Wの事後分布を $q_\phi(w_{i,j}) = \mathcal{N}(\theta_{i,j}, \alpha\theta_{i,j}^2)$ とすると、上の式はlocal reparameterizaiton trickの応用例であることがわかる

correlated weight noise

- 今度は、もともとの定義のように、Bの依存関係を考慮したノイズを考える
  - $\mathbf{b}^m = \mathbf{a}^m \mathbf{W}$の重み $\mathbf{W} = (\mathbf{w}'_1, \mathbf{w}'_2, \ldots, \mathbf{w}'_K)'$ を $\mathbf{w}_i = s_i \theta_i$ と考えてノイズを次のように考える

$$q_\phi(s_i) = N(1, \alpha)$$

  - ノイズは縦ベクトルに対するスケール変数になっている



- このノイズもlocal reparameterizaiton trickと考えることができる

変分ドロップアウトの下限

- これまで考えた事後分布はパラメータθとノイズ項αに分解できる

→dropout posterior

- Dropoutの訓練時はθを期待対数尤度 $\mathbb{E}_{q_\alpha}\left[L_{\mathcal{D}}(\theta)\right]$ のパラメータとする。すると最大化する下限は次のようになる。

$$\mathbb{E}_{q_\alpha}\left[L_{\mathcal{D}}(\theta)\right] - D_{KL}(q_\alpha(\mathbf{w})||p(\mathbf{w}))$$

- KLダイバージェンスの部分については、いろいろ計算すると次のようになる

$$-D_{KL}[q_\phi(w_i)|p(w_i)] \approx \text{constant} + 0.5\log(\alpha) + c_1\alpha + c_2\alpha^2 + c_3\alpha^3$$

$$c_1 = 1.161451241083230, \quad c_2 = -1.502041176441722, \quad c_3 = 0.586299206427007$$

※scale invariant log-uniform priorという事前分布を考えて、かなり長い計算をしていますが省略します

ドロップアウト率の最適化

- 通常、ドロップアウト率αは固定したハイパーパラメータとして扱われる
- 今回の場合、変分ドロップアウトの下限をαについて最大化すれば、簡単に求められる
  - ベイズ推定なので、パラメータに対してロバストだが、それでも効果がある
  - ただし、αは最大値を1とした（ノイズが大きくなることを防ぐため）

実験

- つぎの手法で比較
  - standard binary dropout
  - Gaussian dropout type A (Aにノイズ)
  - Gaussian dropout type B (Bにノイズ)
  - variational dropout type A
  - variational dropout type B
- MNISTで実験
- fully connectedなニューラルネット（隠れ層3）
  - rectified linear units(ReLUs)
  - dropout rate: input layer p=0.2, hidden layers p=0.5
  - early stopping

実験結果（分散）

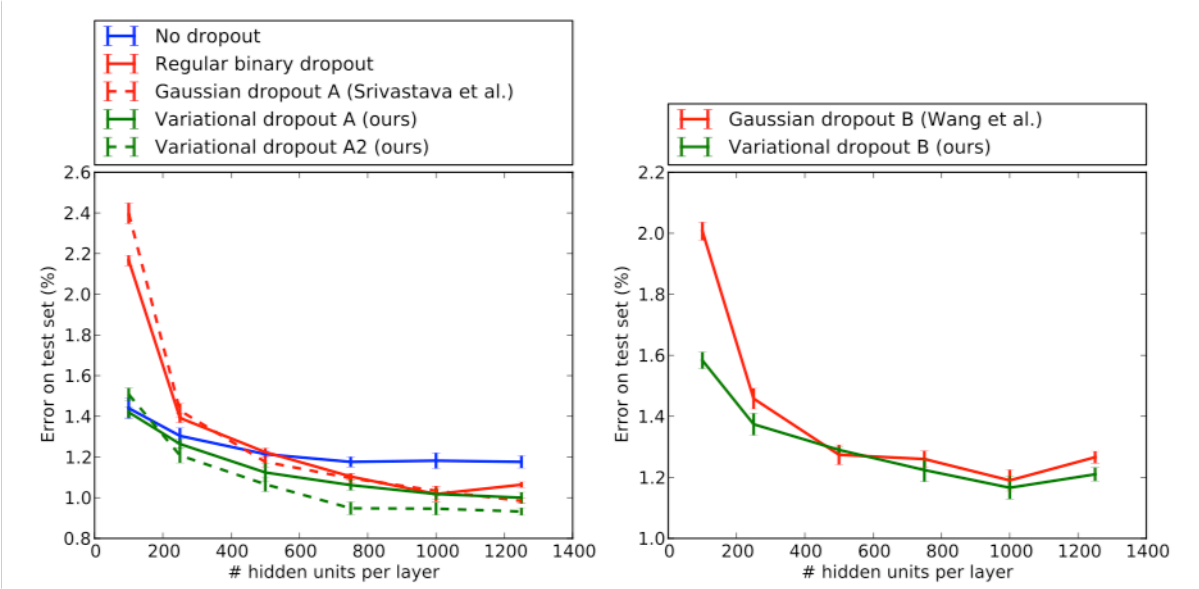
stochastic gradient estimator	top layer 10 epochs	top layer 100 epochs	bottom layer 10 epochs	bottom layer 100 epochs
local reparameterization (ours)	7.8×10^3	1.2×10^3	1.9×10^2	1.1×10^2
separate weight samples (slow)	1.4×10^4	2.6×10^3	4.3×10^2	2.5×10^2
single weight sample (standard)	4.9×10^4	4.3×10^3	8.5×10^2	3.3×10^2
no dropout noise (minimal var.)	2.8×10^3	5.9×10^1	1.3×10^2	9.0×10^0

- variational dropout type Bで学習
- 他のdropoutの結果と比べて分散が抑えられていることがわかる
- ただし、dropoutしない場合に比べるとまだ大きい

実験結果（速度）

- 通常のSGVB（ただしデータごとに全ての重みについてサンプルした場合）とlocal reparameterizationによるSGVBをepochごとの経過時間で比較
  - 通常のSGVB：1635sec
  - 今回のSGVB：7.4sec
- local reparameterizaitonによって200倍以上経過時間が速くなった

実験結果（クラス分類のエラー率）



- 他の手法と比べると同等以上の精度
  - 隠れ層が小さい場合、特に顕著
  - 小さい場合は、
- A2はダウンスケールしたKLダイバージェンスを使用
  - 詳細は不明、書いてない

まとめ

- local reparameterization trickを提案した
  - globalな不確かさをlocalに
  - 計算の複雑さを抑える
  - 簡単に並列化
  - 分散を小さくできる
- ドロップアウトはlocal reparameterization trickの例
  - variational dropout
  - ドロップアウト率を最初に固定するのではなくて、データから推定する