Deep Learning 輪読会 2017

第18章　分配関数との対峙

2018.01.15

理学系研究科附属
天文学教育研究センター
学部4年 吉村勇紀

構成

18.1 対数尤度

18.2 確率的最尤法

18.3 疑似尤度

18.4 スコアマッチングとレシオマッチング

18.5 雑音除去スコアマッチング

18.6 雑音対照推定 (NCE)

18.7 分配関数の推定

18.1 対数尤度

- 非正規化確率分布の正規化
  - 正規化定数の計算は多くのモデルで一般に困難である

- 対数尤度の勾配
  - 分配関数に対応する項が生じる（負項）
  - 以下負項をMCMCする手法を見る

$$\nabla_{\boldsymbol{\theta}} \log Z$$
$$= \frac{\nabla_{\boldsymbol{\theta}} Z}{Z}$$
$$= \frac{\nabla_{\boldsymbol{\theta}} \sum_{\mathbf{x}} \tilde{p}(\mathbf{x})}{Z}$$
$$= \frac{\sum_{\mathbf{x}} \nabla_{\boldsymbol{\theta}} \tilde{p}(\mathbf{x})}{Z}.$$

$$\frac{\sum_{\mathbf{x}} \nabla_{\boldsymbol{\theta}} \exp\left(\log \tilde{p}(\mathbf{x})\right)}{Z}$$
$$= \frac{\sum_{\mathbf{x}} \exp\left(\log \tilde{p}(\mathbf{x})\right) \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x})}{Z}$$
$$= \frac{\sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x})}{Z}$$
$$= \sum_{\mathbf{x}} p(\mathbf{x}) \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x})$$

18.2 確率的最尤法とコントラスティブ・ダイバージェンス

- 尤度関数最大化に対するMCMCの単純な適用
  - 勾配1ステップ毎に混合を行う

Algorithm 18.1 A naive MCMC algorithm for maximizing the log-likelihood with an intractable partition function using gradient ascent

Set ϵ, the step size, to a small positive number.
Set k, the number of Gibbs steps, high enough to allow burn in. Perhaps 100 to train an RBM on a small image patch.
while not converged **do**
  Sample a minibatch of m examples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ from the training set
  $\mathbf{g} \leftarrow \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}^{(i)}; \boldsymbol{\theta})$.
  Initialize a set of m samples $\{\tilde{\mathbf{x}}^{(1)}, \ldots, \tilde{\mathbf{x}}^{(m)}\}$ to random values (e.g., from a uniform or normal distribution, or possibly a distribution with marginals matched to the model's marginals).
  for $i = 1$ to k **do**
    for $j = 1$ to m **do**
      $\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)})$.
    end for
  end for
  $\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)}; \boldsymbol{\theta})$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \mathbf{g}$.
end while

18.2 確率的最尤法とコントラスティブ・ダイバージェンス

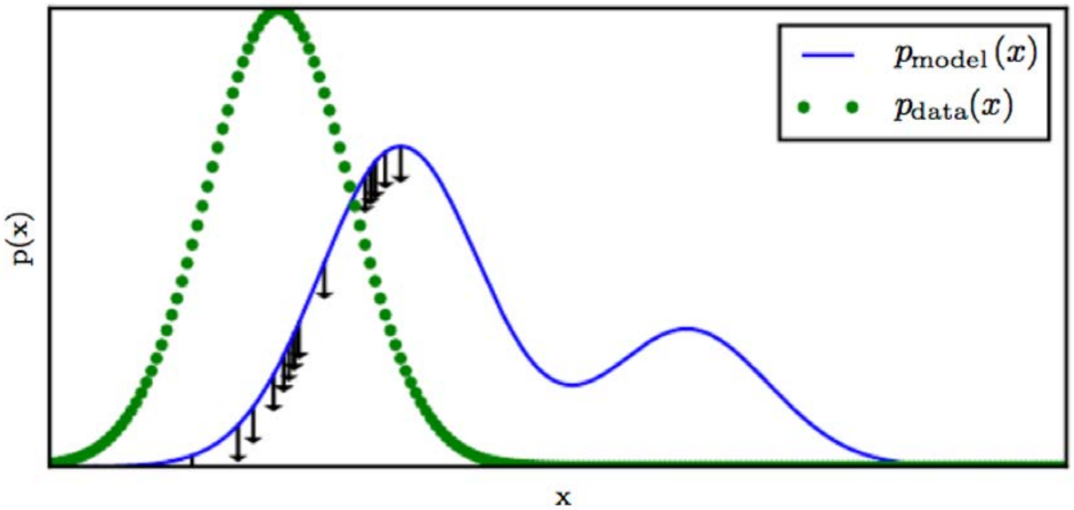
- CDアルゴリズム
  - マルコフ連鎖の初期分布としてデータ分布を用いる

Algorithm 18.2 The contrastive divergence algorithm, using gradient ascent as the optimization procedure

Set ϵ, the step size, to a small positive number.
Set k, the number of Gibbs steps, high enough to allow a Markov chain sampling from $p(\mathbf{x};\boldsymbol{\theta})$ to mix when initialized from p_{data}. Perhaps 1–20 to train an RBM on a small image patch.
while not converged **do**
  Sample a minibatch of m examples $\{\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(m)}\}$ from the training set
  $\mathbf{g} \leftarrow \frac{1}{m}\sum_{i=1}^{m}\nabla_{\boldsymbol{\theta}}\log\tilde{p}(\mathbf{x}^{(i)};\boldsymbol{\theta})$.
  for $i = 1$ to m **do**
    $\tilde{\mathbf{x}}^{(i)} \leftarrow \mathbf{x}^{(i)}$.
  end for
  for $i = 1$ to k **do**
    for $j = 1$ to m **do**
      $\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)})$.
    end for
  end for
  $\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m}\sum_{i=1}^{m}\nabla_{\boldsymbol{\theta}}\log\tilde{p}(\tilde{\mathbf{x}}^{(i)};\boldsymbol{\theta})$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon\mathbf{g}$.
end while

18.2 確率的最尤法とコントラスティブ・ダイバージェンス

- CDアルゴリズムの問題点
  - 偽モードの出現



  - RBMや可視変数ボルツマンマシンでは最尤推定値に収束しない
  - CDの更新方向はいかなる関数の勾配方向にならない

18.2 確率的最尤法とコントラスティブ・ダイバージェンス

- SML(PCD)アルゴリズム
  - マルコフ連鎖の初期分布として前の勾配ステップの分布を用いる

Algorithm 18.3 The stochastic maximum likelihood / persistent contrastive divergence algorithm using gradient ascent as the optimization procedure

Set ϵ, the step size, to a small positive number.
Set k, the number of Gibbs steps, high enough to allow a Markov chain sampling from $p(\mathbf{x}; \boldsymbol{\theta} + \epsilon \mathbf{g})$ to burn in, starting from samples from $p(\mathbf{x}; \boldsymbol{\theta})$. Perhaps 1 for RBM on a small image patch, or 5–50 for a more complicated model like a DBM.

Initialize a set of m samples $\{\tilde{\mathbf{x}}^{(1)}, \dots, \tilde{\mathbf{x}}^{(m)}\}$ to random values (e.g., from a uniform or normal distribution, or possibly a distribution with marginals matched to the model's marginals).
while not converged **do**
  Sample a minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from the training set
  $\mathbf{g} \leftarrow \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}^{(i)}; \boldsymbol{\theta})$.
  for $i = 1$ to k **do**
    for $j = 1$ to m **do**
      $\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)})$.
    end for
  end for
  $\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)}; \boldsymbol{\theta})$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \mathbf{g}$.
end while

18.3 疑似尤度

- 分配関数を計算せずに対数尤度を求める方法
  - 条件付き確率の和で対数尤度を擬似的に表す
  - 条件付き確率は確率の比なので分配関数は打ち消して現れない
  - 疑似尤度 $$\sum_{i=1}^{n} \log p(x_i \mid \boldsymbol{x}_{-i})$$
- 一般化疑似尤度

$$\sum_{i=1}^{m} \log p(\mathbf{x}_{\mathbb{S}^{(i)}} \mid \mathbf{x}_{-\mathbb{S}^{(i)}})$$

  - インデックス集合として一般化
  - 密度推定など完全な同時分布が必要なタスクには向かない
  - 相関がなるべくないようなインデックス集合が取れれば強力

18.4 スコアマッチングとレシオマッチング

- スコアマッチング
  - モデル対数密度の入力微分とデータ対数密度の入力微分の二乗誤差を最小にする

$$L(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{2} ||\nabla_{\boldsymbol{x}} \log p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta}), -\nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})||_2^2,$$

$$J(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} L(\boldsymbol{x}, \boldsymbol{\theta}),$$

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}).$$

  - 分配関数はxの関数ではないので、微分を取ることで落ちる
  - L(x, θ)の最小化は次の期待値の最小化と同じ

$$\tilde{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{n} \left(\frac{\partial^2}{\partial x_j^2} \log p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta}) + \frac{1}{2} \left(\frac{\partial}{\partial x_j} \log p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta}) \right)^2 \right)$$

  - 対数密度の微分、二回微分が必要

18.4 スコアマッチングとレシオマッチング

- レシオマッチング
  - スコアマッチングの離散データへの拡張
  - 次の目的関数の事例平均を最小化する

$$L^{(\mathrm{RM})}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{n} \left(\frac{1}{1 + \frac{p_{\mathrm{model}}(\boldsymbol{x}; \boldsymbol{\theta})}{p_{\mathrm{model}}(f(\boldsymbol{x}), j); \boldsymbol{\theta})}} \right)^2$$

  - 計算量はSMLのn倍
  - 二値データや高次のスパースなデータ（単語など）に適用される

18.5 雑音除去スコアマッチング

- スコアマッチングの正則化
  - 新のデータ分布ではなく、次の分布に置き換える

$$p_{\mathrm{smoothed}}(\boldsymbol{x}) = \int p_{\mathrm{data}}(\boldsymbol{y}) q(\boldsymbol{x} \mid \boldsymbol{y}) d\boldsymbol{y}$$

  - 実際には真のデータ分布ではなく経験分布しか使えないから
  - 一致推定量の漸近的一致性は失われる

18.6 雑音対照推定 (NCE)

- 分配関数も同時に推定する
  - 次の対数尤度からパラメータと分配関数の近似値を同時に推定する

$$\log p_{\text{model}}(\mathbf{x}) = \log \tilde{p}_{\text{model}}(\mathbf{x};\boldsymbol{\theta}) + c,$$

  - 単純な尤度最大化は不適（cが大きくなるだけ）
  - ノイズ分布を導入してスイッチ変数で切り替える

$$p_{\text{joint}}(y=1) = \frac{1}{2},$$

$$p_{\text{joint}}(\mathbf{x} \mid y=1) = p_{\text{model}}(\mathbf{x}),$$

$$p_{\text{joint}}(\mathbf{x} \mid y=0) = p_{\text{noise}}(\mathbf{x}).$$

$$\boldsymbol{\theta}, c = \arg\max_{\boldsymbol{\theta}, c} \mathbb{E}_{\mathbf{x},\mathrm{y}\sim p_{\text{train}}} \log p_{\text{joint}}(y \mid \mathbf{x})$$

18.7 分配関数の推定

- 重点サンプリング

$$
\begin{aligned}
Z_1 &= \int \tilde{p}_1(\mathbf{x})\, d\mathbf{x} \\
&= \int \frac{p_0(\mathbf{x})}{p_0(\mathbf{x})} \tilde{p}_1(\mathbf{x})\, d\mathbf{x} \\
&= Z_0 \int p_0(\mathbf{x}) \frac{\tilde{p}_1(\mathbf{x})}{\tilde{p}_0(\mathbf{x})}\, d\mathbf{x}
\end{aligned}
$$

$$
\hat{Z}_1 = \frac{Z_0}{K} \sum_{k=1}^{K} \frac{\tilde{p}_1(\mathbf{x}^{(k)})}{\tilde{p}_0(\mathbf{x}^{(k)})} \quad \text{s.t. : } \mathbf{x}^{(k)} \sim p_0
$$

  – 一般にp_1は高次元の複雑な分布なので質の悪い推定になってしまう

18.7 分配関数の推定

- 焼きなまし重点サンプリング
  - $D_{\mathrm{KL}}(p_0 \| p_1)$ が大きい時に中間分布を挟んで隔たりを埋める方法
  - 分配関数の比は次のように表せる

$$
\begin{aligned}
\frac{Z_1}{Z_0} &= \frac{Z_1}{Z_0}\frac{Z_{\eta_1}}{Z_{\eta_1}} \cdots \frac{Z_{\eta_{n-1}}}{Z_{\eta_{n-1}}} \\
&= \frac{Z_{\eta_1}}{Z_0}\frac{Z_{\eta_2}}{Z_{\eta_1}} \cdots \frac{Z_{\eta_{n-1}}}{Z_{\eta_{n-2}}}\frac{Z_1}{Z_{\eta_{n-1}}} \\
&= \prod_{j=0}^{n-1} \frac{Z_{\eta_{j+1}}}{Z_{\eta_j}}.
\end{aligned}
$$

  - 中間分布 -> 加重幾何平均をよく用いる

$$p_{\eta_j} \propto p_1^{\eta_j} p_0^{1-\eta_j}$$

  - 中間分布のサンプリングはMCMC

$$p_{\eta_j}(\boldsymbol{x}) = \int p_{\eta_j}(\boldsymbol{x}') T_{\eta_j}(\boldsymbol{x} \mid \boldsymbol{x}')\, d\boldsymbol{x}'$$

18.7 分配関数の推定

- 焼きなまし重点サンプリング
  - 手順
    - for $k = 1 \ldots K$
      - Sample $\boldsymbol{x}_{\eta_1}^{(k)} \sim p_0(\mathbf{x})$
      - Sample $\boldsymbol{x}_{\eta_2}^{(k)} \sim T_{\eta_1}\left(\mathbf{x}_{\eta_2}^{(k)} \mid \boldsymbol{x}_{\eta_1}^{(k)}\right)$
      - $\ldots$
      - Sample $\boldsymbol{x}_{\eta_{n-1}}^{(k)} \sim T_{\eta_{n-2}}(\mathbf{x}_{\eta_{n-1}}^{(k)} \mid \boldsymbol{x}_{\eta_{n-2}}^{(k)})$
      - Sample $\boldsymbol{x}_{\eta_n}^{(k)} \sim T_{\eta_{n-1}}(\mathbf{x}_{\eta_n}^{(k)} \mid \boldsymbol{x}_{\eta_{n-1}}^{(k)})$
    - end
  - 重要度重み

$$w^{(k)} = \frac{\tilde{p}_{\eta_1}(\boldsymbol{x}_{\eta_1}^{(k)})}{\tilde{p}_0(\boldsymbol{x}_{\eta_1}^{(k)})} \frac{\tilde{p}_{\eta_2}(\boldsymbol{x}_{\eta_2}^{(k)})}{\tilde{p}_{\eta_1}(\boldsymbol{x}_{\eta_2}^{(k)})} \cdots \frac{\tilde{p}_1(\boldsymbol{x}_1^{(k)})}{\tilde{p}_{\eta_{n-1}}(\boldsymbol{x}_{\eta_n}^{(k)})}$$

  - 分配関数の近似

$$\frac{Z_1}{Z_0} \approx \frac{1}{K} \sum_{k=1}^{K} w^{(k)}$$

18.7 分配関数の推定

- ブリッジサンプリング
  - １つの中間分布（ブリッジ）で補間する

$$\frac{Z_1}{Z_0} \approx \sum_{k=1}^{K} \frac{\tilde{p}_*(\boldsymbol{x}_0^{(k)})}{\tilde{p}_0(\boldsymbol{x}_0^{(k)})} \Bigg/ \sum_{k=1}^{K} \frac{\tilde{p}_*(\boldsymbol{x}_1^{(k)})}{\tilde{p}_1(\boldsymbol{x}_1^{(k)})}.$$

  - $D_{\mathrm{KL}}(p_0 \| p_1)$ が大きい場合にも適用しうる
  - 最適なブリッジ分布

$$\frac{\tilde{p}_0(\boldsymbol{x})\tilde{p}_1(\boldsymbol{x})}{r\tilde{p}_0(\boldsymbol{x})+\tilde{p}_1(\boldsymbol{x})}, \text{ where } r = Z_1/Z_0$$

  - 粗いrから始めて更新していく
  - AISとブリッジサンプリングを組み合わせた手法も提案されている

参考文献

- Deep Learning
  - Ian Goodfellow, Yoshua Bengio, Aaron Courville
  - 日本語版
    https://www.amazon.co.jp/%E6%B7%B1%E5%B1%A4%E5%AD%A6%E7%BF%92-Ian-Goodfellow/dp/4048930621