UNIVERSITÀ PARTHENOPE
DiST
DIPARTIMENTO DI SCIENZE E TECNOLOGIE

Machine Learning (part II)

Sampling Methods

Angelo Ciaramella

Introduction

- Why sampling?
  - approximate many sums and integrals
    - gradient of the log partition function of an undirected model
  - train a model that can sample from the training distribution

Monte Carlo Sampling

- When a sum or an integral cannot be computed exactly
  - approximate it using Monte Carlo sampling

- Suppose

$$s = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) f(\boldsymbol{x}) = E_p[f(\mathbf{x})]$$

$$s = \int p(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} = E_p[f(\mathbf{x})]$$

Monte Carlo Sampling

- Drawing *n* samples $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$

$$\hat{s}_n = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}^{(i)})$$

$$\mathbb{E}[\hat{s}_n] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[f(\boldsymbol{x}^{(i)})] \overset{\boldsymbol{x}^{(i)}}{=} \frac{1}{n}\sum_{i=1}^{n} s = s$$

Unbiased

$$\lim_{n\to\infty} \hat{s}_n = s$$

for the law of large number $\boldsymbol{x}^{(i)}$ i.i.d

Monte Carlo Sampling

- Variance

$$\mathrm{Var}[f(\mathbf{x}^{(i)})] < \infty$$

$$\mathrm{Var}[\hat{s}_n] = \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}[f(\mathbf{x})]$$
$$= \frac{\mathrm{Var}[f(\mathbf{x})]}{n}.$$

- Central limit theorem
  - converges to a normal distribution

s mean

$\hat{s}_n$ converge

$\frac{\mathrm{Var}[f(\mathbf{x})]}{n}$ variance

Markov Chain Monte Carlo Methods

- Markov chain
  - Updating state x
  - Random state x and transition distribution T(x'|x)
  - T(x'|x) probability that a random update will go to state x' if it starts in state x
- Run infinitely many Markov chains in parallel
  - States drawn from some distribution $q^{(t)}(x)$
  - Goal $q^{(t)}(x)$ converging to p(x)

$$q^{(t+1)}(x') = \sum_{x} q^{(t)}(x) T(x' \mid x)$$

Markov Chain Monte Carlo Methods

- Transition operator

$$A_{i,j} = T(\mathbf{x}' = i \mid \mathbf{x} = j)$$

- over all the different Markov chains run in parallel shifts

"burning in" the Markov chain

$$\boldsymbol{v}^{(t)} = \boldsymbol{A}\boldsymbol{v}^{(t-1)} \qquad \boldsymbol{v}^{(t)} = \boldsymbol{A}^{t}\boldsymbol{v}^{(0)}$$

columns of A (stochastic matrix) represents a probability distribution

$$\boldsymbol{v}^{(t)} = \left(\boldsymbol{V}\operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}\right)^{t}\boldsymbol{v}^{(0)} = \boldsymbol{V}\operatorname{diag}(\boldsymbol{\lambda})^{t}\boldsymbol{V}^{-1}\boldsymbol{v}^{(0)}$$

A is guaranteed to have only one eigenvector with eigenvalue 1

Markov Chain Monte Carlo Methods

- Convergence

$$v' = Av = v$$ Eigenvector equation

- If we have chosen T correctly, then the stationary distribution q will be equal to the distribution p we wish to sample from

Sampling

- Two basic approaches
  - derive T from a given learned p_{model}
  - directly parametrize T and learn it
    - its stationary distribution implicitly defines the p_{model} of interest

- Commonly use of Markov chains
  - draw samples from an energy-based model defining a distribution $p_{model}(x)$
  - we want the q(x) for the Markov chain to be $p_{model}(x)$
    - To obtain the desired q(x), we must choose an appropriate $T(x' \mid x)$

Gibbs sampling

- Special case of the Metropolis-Hastings algorithm

- Markov chain
  - samples from $p_{model}(x)$
  - T (x' | x) is accomplished by selecting one variable x_i and sampling it from p_{model} conditioned on its neighbors in the undirected graph G defining the structure of the energy-based model

Gibbs sampling

- Distribution from which we wish to sample

$$p(\mathbf{z}) = p(z_1, \ldots, z_M)$$

- Each step of the Gibbs sampling procedure
  - replacing the value of one of the variables by a value drawn from the distribution of that variable conditioned on the values of the remaining variables

Gibbs sampling

Gibbs Sampling

1. Initialize $\{z_i : i = 1, \ldots, M\}$
2. For $\tau = 1, \ldots, T$:
   - Sample $z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - Sample $z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.

     $\vdots$
   - Sample $z_j^{(\tau+1)} \sim p(z_j | z_1^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \ldots, z_M^{(\tau)})$.

     $\vdots$
   - Sample $z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$.

$$A(\mathbf{z}^\star, \mathbf{z}) = \frac{p(\mathbf{z}^\star) q_k(\mathbf{z}|\mathbf{z}^\star)}{p(\mathbf{z}) q_k(\mathbf{z}^\star|\mathbf{z})} = \frac{p(z_k^\star|\mathbf{z}_{\setminus k}^\star) p(\mathbf{z}_{\setminus k}^\star) p(z_k|\mathbf{z}_{\setminus k}^\star)}{p(z_k|\mathbf{z}_{\setminus k}) p(\mathbf{z}_{\setminus k}) p(z_k^\star|\mathbf{z}_{\setminus k})} = 1$$

Intractable partition functions

- Valid probability distribution

$$p(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\tilde{p}(\mathbf{x};\boldsymbol{\theta})$$

- Partition function Z

$$\int \tilde{p}(\boldsymbol{x})d\boldsymbol{x}$$

$$\sum_{\boldsymbol{x}} \tilde{p}(\boldsymbol{x}).$$

- This operation is intractable for many interesting models

Intractable partition functions

- Normalized probability distribution

$$p(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\tilde{p}(\mathbf{x};\boldsymbol{\theta})$$

- Techniques used for training and evaluating models
  - Log-Likelihood Gradient
  - Stochastic Maximum Likelihood
  - Markov Chain Monte-Carlo sampling
  - Contrastive Divergence (CD)
  - Pseudolikelihood
  - Score Matching and Ratio Matching
  - Noise-Contrastive Estimation
  - Annealed Importance Sampling
  - Bridge Sampling

Log-Likelihood Gradient

- Gradient of the likelihood

$$\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x};\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta})$$

basis for a variety of Monte Carlo methods for approximately maximizing the likelihood of models with intractable partition functions

$$\nabla_{\boldsymbol{\theta}} \log Z = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x})$$

burning in a set of Markov chains from a random initialization

MCMC

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

Contrastive Divergence

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

Score Matching

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

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

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