CSE6243: Advanced Machine Learning **Fall 2023**

Lecture 25: Review

Lecturer: Bo Dai *Scribes: Amish Saini, Ninaad Lakshman*

Note: *LaTeX template courtesy of UC Berkeley EECS Department.*

25.1 Module I: Background Knowledge

25.1.1 Convex Optimization

Convex optimization exists as a subset of optimization problems with the following definition.

We can say a function f is convex if and only if the domain is convex set Ω and for $\lambda \in [0, 1]$ and $x, y \in \Omega$, we have:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \tag{25.1}$$

Let's also look at an inequality for convex functions: Jensen's Inequality.

Let's consider a convex function f and some $x \in$ domain of f. We can say that:

$$f(\mathbb{E}_p(x)[x]) \leq \mathbb{E}_p(x)[f(x)] \tag{25.2}$$

This inequality proves to be very important for many machine learning use cases.

25.1.2 Stochastic Gradient Descent

Stochastic gradient descent is a type of gradient descent (trying to minimize some objective function) which essentially adds randomness to our algorithm by sampling datapoints.

Say $\tilde{\nabla} f(x)$ is our stochastic gradient:

$$\tilde{\nabla} f(x) = \sum_{i=1}^{k} \tilde{f}_i(x) \; \forall k << n \tag{25.3}$$

Our stochastic gradient descent would be:

$$x_{t+1} = x_t - y\tilde{\nabla} f(x_t) \tag{25.4}$$

25.1.3 Convex Conjugate

Convex conjugates hold properties that are heavily used in many machine learning settings including generative adversarial networks and elbow functions.

A convex conjugate function is defined as follows:

$$f^*(y) = \sup_{x \in \Omega}(x^T y - f(x)) \tag{25.5}$$

if f is convex:

$$f(x) = \sup_{y}(x^T y - f^*(y)) \tag{25.6}$$

25.1.4 Sampling

25.1.4.1 Basic

Consider some function f and some probability distribution p:

$$\mathbb{E}_{p(x)}[f(x)] \tag{25.7}$$

Approximating this expectation by essentially computing the average across all the sampled points is one approach. We can sample these points by essentially pull from the $p(x)$ distribution.

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i), \{x_i\}_{i=1}^{n} \sim p(x) \tag{25.8}$$

25.1.4.2 Inverse Transformation Sampling

This is a sampling method that starts at the standard uniform distribution. First, we must consider function $F(z)$, which we define as a cumulative distribution function. We also have some probability distribution $p(x)$ which will be our target.

$$F(z) = \int_{-\infty}^{z} p(x)dx = P(X \le z) \tag{25.9}$$

The sampling is a 2-step process:

1. Get some random variable $u \in U[0, 1]$
2. Compute the inverse of the cumulative distribution function $x = F^-1(u)$

25.1.4.3 Acceptance-Rejection Sampling

This is another type of sampling involving accepting or rejecting based on specific criteria. It is used in machine learning use cases like speculative decoding. We start with some distribution $q(x)$:

The sampling procedure is as follows:

1. Choose an M where $Mq(x) \ge p(x)$ for every x
2. Sample $y \sim q(y)$
3. Sample $u \sim U[0, 1]$. Accept if $u \le \frac{p(y)}{M*q(y)}$. Otherwise reject.

25.1.4.4 Importance Sampling

Importance sampling uses the same idea in that we try to generate a sample through some distribution $q(x)$ to arrive at $p(x)$.

It hinges on the idea that we really only need to estimate this:

$$\int p(x)f(x)dx = \int \frac{p(x)}{q(x)}f(x)q(x)dx \tag{25.10}$$

Given a sample $x \sim q(x)$, we can say that the expectation of the distribution $p(x)$ can be approximated as:

$$\mathbb{E}_p(x)[f(x)] \approx \frac{1}{n}\sum_{i=1}^{n}\frac{p(x_i)}{q(x_i)}f(x_i) \tag{25.11}$$

25.1.5 Markov Chain Monte Carlo Methods

MCMC relies on detailed balance:

$$P(y)T(x|y) = P(x)T(y|x) \tag{25.12}$$

25.1.5.1 Metropolis Hasting

We can also sample using Monte Carlo techniques with a random walk. The acceptance rate is as follows:

$$A(x,y) = \min(\frac{p(y)q(x|y)}{p(x)q(y|x)}) = \frac{p(y)}{p(x)} \tag{25.13}$$

25.1.5.2 Gibbs

Gibbs sampling also leverages a random walk to sample a probability distribution:

$$q(y|x) = p(x_i|x_{-i}) \tag{25.14}$$

$$x_{-i} = \{x_0, x_1, ..., x_{i-1}, x_{i+1}, ..., x_n\} \tag{25.15}$$

In Gibbs sampling, the acceptance rate is:

$$A(x,y) = \min(\frac{p(x_i)p(x_{-i}x_i}{p(x_{-i})p(x_i|x_{-i})}) = 1 \tag{25.16}$$

25.1.5.3 Langevin Dynamics: Metropolis Adjusted Langevin Algorithm

We can also leverage Langevin dynamics to help explore the random walk space more efficiently. This is the update formula once we have added Langevin dynamics:

$$x_{new} = x + \eta\nabla\log p(x) + \sqrt{\eta}\epsilon \tag{25.17}$$

25.2 Module II: Deep Generative Models

25.2.1 Exponential Family: Energy-Based Model

The exponential family is a class of probability distributions that includes many commonly used distributions, such as the normal distribution, exponential distribution, Poisson distribution, and Bernoulli distribution. Distributions in the exponential family have a specific form that can be expressed in terms of a few key components.

$$\begin{aligned} p(x) &= h(x)\exp\left(y^T T(x) - A(y)\right) \\ A(y) &= \log \int h(x)\exp\left(y^T T(x)\right) dx \end{aligned} \tag{25.18}$$

where:

x is the random variable,
$h(x)$ is the base measure,
$T(x)$ is the sufficient statistic,
$A(y)$ is the log partition function.

Recall the definition of the Energy-Based Model. Energy-based models provide a unified framework for machine learning, encompassing various tasks and models, including those based on the exponential family, by expressing learning as energy minimization flexibly and coherently.

$$p_\theta(x) = \exp\left(f_\theta(x) - A(\eta)\right) \tag{25.19}$$

which satisfies $p_\theta(x) \geq 1$ and $\int p_\theta(x)dx = 1$.

From this equation, we can take a look at a specific case:

Restricted Boltzmann Machine (RBM):

A Restricted Boltzmann Machine (RBM) is a generative neural network used for unsupervised learning, modeling probability distributions over input data. RBMs have visible and hidden layers with no connections within the same layer, enabling efficient training via contrastive divergence. Characterized by an energy-based formulation, RBMs adjust parameters during training to minimize energy differences between observed data and model-generated samples, finding applications in collaborative filtering, dimensionality reduction, and feature learning.

$$p(x,h) \propto \exp\left(x^\top W h + \alpha^\top x + \beta^\top h\right) \tag{25.20}$$

where h are the hidden units.

Contrastive Divergence: A training algorithm for energy-based models like RBMs, Contrastive Divergence approximates gradients through MCMC sampling. It minimizes the divergence between model and data distributions by iteratively refining samples.

Score Matching: A method for training energy-based models, Score Matching minimizes the discrepancy between observed and model-predicted scores without explicit probability density estimation. It's valuable in scenarios with challenging partition function computation, enabling efficient parameter learning.

25.2.2 Autoregressive Model (ARM)

The Autoregressive Model (ARM) defines the probability distribution of a sequence as the product of conditional probabilities for each element conditioned on its predecessors:

$$P(x) = \prod_{i=1}^{T} P\left(x_i \mid x < i\right) \tag{25.21}$$

This modeling approach allows for efficient probability estimation, and in the context of Maximum Likelihood Estimation (MLE), the tractability of the gradient makes optimization straightforward. Unlike complex sampling methods, the true gradient can be directly employed, simplifying the training process and enhancing the efficiency of learning in autoregressive models.

25.2.3 Variational Autoencoder (VAE)

In a Variational Autoencoder (VAE), the model defines the data distribution as an integral over latent variables, acknowledging the existence of a latent variable z:

$$p(x) = \int p(x \mid z) p(z)\, dz \tag{25.22}$$

Here, z serves as a latent variable capturing hidden representations. However, Maximum Likelihood Estimation (MLE) for this model is intractable due to the involved integral. To address this, VAEs employ the Elbo (Evidence Lower Bound) function derived from Jensens inequality, leveraging the concept of complex conjugates. This technique allows for practical optimization and efficient learning in VAEs.

25.2.4 Diffusion Model

The Diffusion Model represents the data distribution as an integral over latent variables, characterized by the following equation:

$$p(x) = \int p\left(x \mid z_1\right) \prod_{i=2}^{d} p\left(z_{i+1} \mid z_i\right)\, d\left(z_i\right) \tag{25.23}$$

In this model, the latent variables z_i evolve sequentially, capturing the data distribution through a series of conditional distributions. This integral formulation defines the complex relationship between the observed data and latent variables within the diffusion model, offering a flexible framework for probabilistic modeling.

25.2.5 Normalizing Flow and GANs

Normalizing Flow:

$$\begin{aligned} &\varepsilon \sim p_0(\varepsilon) \quad \text{sample noise} \\ &x = f(\varepsilon) \quad \text{sample gen function over noise} \end{aligned} \tag{25.24}$$

GAN:

$$P(x) = P_0 \left(F^{-1}(x) \cdot \left| \det \frac{\partial f^{-1}}{\partial x} \right| \right) \tag{25.25}$$

Normalizing Flow incorporates sample noise, while for Generative Adversarial Networks (GANs), the invertibility of f determines the use of Maximum Likelihood Estimation (MLE) or gradient estimation. The critical aspect for generative models, including GANs, is the accurate estimation of gradients, and practitioners should assess the feasibility of gradient approximation.

25.3 Module III: Differentiable Programming

25.3.1 2 Perspectives: Learning Objectives

Learning to Fit Distributions: In this perspective, the focus is on training models to accurately represent and fit complex probability distributions.

Learning to Sample: All generative models are essentially creating generators and samplers, emphasizing the importance of efficient algorithms for generating representative samples. Inspired learning extends to algorithm design, such as creating samplers, implementing greedy search, dynamic programming, and convex optimization solvers. These algorithms take inputs and produce results, with parameters now becoming adjustable via constructing loss, determining gradients, and employing optimization methods like stochastic gradient descent (SGD). The central theme revolves around efficiently calculating gradients for improved learning.

25.4 Module IV: Reinforcement Learning

- Difficulty is no longer how to calculate the gradient but rather handling distribution shift.
- **No Free Lunch Theorem**: No algorithm demonstrates superiority in all situations; there is always some case where your algorithm performs best.
- Markov Decision Process (MDP): A common way to model Reinforcement Learning (RL), abstracting the sequential decision-making process into this model.
- Bellman Optimality Recursion: An integral concept in RL, differentiating between policy evaluation and the best policy, as well as planning versus learning.

25.4.1 De-Facto Model: Markov Decision Process (MDP)

- Widely accepted in the community.
- **Bellman Recursion**:

$$Q^{\pi}(s,a) = R(s,a) + \gamma \mathbb{E}_{P(s'|s,a),\pi(a',s')}[Q^{\pi}(s',a')] \tag{25.26}$$

  The idea is to be given a policy and generate the Q function from it.

$$Q^{*}(s,a) = R(s,a) + \gamma \mathbb{E}_{P(s'|s,a)}[\max_{a'} Q^{*}(s',a')] \tag{25.27}$$

25.4.2 Two Fundamental RL Challenges

- **Policy Evaluation vs. Optimization:** The challenge of assessing the performance of a given policy versus finding the optimal policy that maximizes expected cumulative reward. For Random Forests (RF), this could involve evaluating different decision policies generated by the forest and optimizing decision-making criteria within each tree. Additionally, the optimization challenge in RF may involve enhancing the decision-making criteria within each tree to contribute to an improved overall policy.
- **Planning vs. Learning:** Balancing the process of deliberating the best course of action (planning) before interacting with the environment and adapting strategies based on observed outcomes (learning). In RF, planning might involve simulating decision paths using the ensemble, while learning could mean adapting decision criteria within each tree based on observed performance. The challenges in RF may involve refining decision policies within each tree and coordinating their efforts to collectively contribute to effective policy evaluation and optimization.

25.4.3 Additional RL Challenges

- **Model-Based (MB) vs. Model-Free (MF):** The challenge of balancing between algorithms that maintain an internal representation of the environment (MB) for planning and those that directly learn a policy or value function without an explicit model (MF).
- **On-Policy vs. Off-Policy:** Balancing between learning from the current policy (on-policy) and learning from a different policy or historical data (off-policy) for more sample-efficient learning.
- **Tabular vs. Function Approximation:** Adapting algorithms to handle the trade-offs between explicitly representing state or action spaces in a table (tabular) and using parameterized functions (function approximation) for handling complex and large state or action spaces.
- **Value (V) vs. Policy (π):** Coordinating the learning of both value functions (V) representing expected cumulative future rewards and policies (π) representing action strategies. Each has its own set of challenges and trade-offs.

Addressing these challenges in RL involves tailoring algorithms to the problem characteristics, considering the trade-offs between different dimensions, and recognizing that there is no one-size-fits-all solution (No Free Lunch Theorem).

25.4.4 Actor-Critic

Actor-Critic is a powerful RL paradigm that combines the strengths of policy-based (actor) and value-based (critic) methods. The actor determines optimal actions, while the critic evaluates these actions' values. This collaboration enhances learning stability by allowing the actor to adapt its policy based on valuable insights from the critic, resulting in more efficient decision-making in dynamic environments.

25.4.5 Avoiding Distribution Shift

We should always sample from the distribution to determine data to avoid distribution shift. This is essentially boils down to:

$$D_t \leftarrow \pi_t \tag{25.28}$$