CS 165: Foundations of Machine Learning **Winter 2019**

Lecture 3: 1/15/2019

Lecturer: Anima Anandkumar *Scribes: Bhairav Chidambaram, Connor Soohoo, Lucien Werner*

3.1 Recap on non-convex optimization

3.1.1 Symmetries

- Symmetries that arise in a function being optimized result inevitably in the introduction of saddle points. First- and second-order optimality criteria are unable to detect these saddle points. Techniques like Gradient Descent and Newton's method are not applicable.
- However, symmetries can be helpful because many local optima might have identical values. In neural networks for example, overparameterization leads to an explosion of critical points. It is not clear how much this affects performance though.
- How to escape saddle points? Two main methods.
  - **Cubic Regularization** - second order optimization that considers 3rd-order terms in the Taylor series approximation. Not well suited for machine learning in practice. [NP06]
  - **SGD with noise** - adds noise to SGD step to escape stationary points with very little computational overhead. This technique is the standard in practice [JGN+17]

3.1.2 Computational Challenges in Optimization

The optimizer is treated as a black box in ML but there might be computational problems when scaling up problem sizes. There are methods to compress gradient-based optimization, for example **Distributed Gradient Computation**. The data are shared between workers and gradient computations are carried out in a distributed manner. To limit overhead of reporting gradient values in each iteration, only the signs of the gradients are sent back to the master. 1st- but not 2nd-order convergence guarantees exist. [BWAA18]

3.1.3 Takeaways on optimization for ML

- Perfect optimizers (i.e., that provide a perfect fit to the training data) can lead to overfitting. In practice, they are not desirable.
- Can't treat optimizer as a black box. Need to consider in conjunction with choice of loss function and regularizer.
- Setting hyperparameters for optimizers is a dark art still. One should be skeptical of crude comparisons between different optimizers for this reason.
- Early stopping is a crucial technique for preventing overfitting to training data. More proof for why a perfect optimizer does not necessarily lead to better learning outcomes. An equivalence exists between early stopping and training to zero loss + regularization. [WYW17]

- Example: **multi-lingual word embeddings**. Suppose we have two different word vectors $u(w)$ and $v(w)$ on some input w. We want to project these two vectors to some common vector space so that the words in u and v that have similar meaning (regardless of language) appear very close together in the resultant common space. Let A and B be projection matrices to project word vectors u and v, respectively, into the common space. The multi-lingual word embedding objective can be written as:

$$\underset{A,B}{\text{argmin}} \sum_{w} ||A * u(w) - B * v(w)||_2^2$$

Observe that the global minimizer for this loss function is to choose A and B to be zero matrices, which yields a net loss of 0. Clearly, we do not want to use this globally optimal solution. We would much rather prefer a different locally optimal solution that provides more useful information. To work around converging to the global minimum, we can initialize A and B away from 0 in the hopes that a training method converges to a locally optimal solution instead. Consequently, the derived local minimizer for a given pair of word embeddings will be heavily dependent on the initial values of A and B, illustrating the importance of parameter tuning.

For more information, check out Facebook AI's explanation of multi-lingual embeddings [SA18].

3.2 Competitive Optimization and Multi-agent Optimization

3.2.1 Generative Adversarial Networks (GANs)

Traditionally, a GAN is composed of a generator G and a discriminator D. The generator generates x by taking as input a noisy latent vector and turning it into data-point x, mimicking sampling from the training distribution. The discriminator takes input x and tries to determine if the input image is real (from training set) or fake (generated). In machine learning, we typically assume both G and D are fully differentiable and can be trained using gradient methods. GANs are most often used for realistic image generation.

Let $D(x)$ be the output (1 for "real", 0 for "fake") of the discriminator on input x. Let p_{data} be the training set distribution, and let p_{latent} be the generator's output distribution. The GAN objective can then be written as

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{x \sim p_{latent}}[\log(1 - D(x))]$$

Given this objective, the goal of an optimization algorithm would be to find a saddle-point, one that is a minimum w.r.t. the parameters of G and a maximum w.r.t. the parameters of D. Unfortunately, it is difficult to measure the absolute performance of a GAN since there is no notion of a "better" saddle-point. In practice, metrics such as Inception Score are used to compare the performance of different GANs but remain controversial [BS18].

For more information, check out the original paper on GANs [Goo14] or this blog post on GANs [Wen17]

3.2.2 Nash Equilibrium

Consider a general m-player game where player i aims to minimize her loss function $L_i(\theta_1, ..., \theta_m)$. Note player i can only influence her own choice and that player i's loss function depends on the choices of *all* players in the game. In other words, each player is assumed to know the equilibrium strategies of all other players in the game. A Nash Equilibrium is defined as a global optimum where no player can improve her loss function in a unilateral fashion. However, computing the global Nash Equilibrium is often not practical for machine learning applications for reasons explained in 3.1.3, so we aim to find local Nash Equilibriums in machine learning contexts. Per usual, we assume that the loss functions for all players are twice differentiable.

3.2.3 Nash Equilibrium in Games vs. ML

Game theory assumes agents are selfish. This is one limitation of Nash equilibrium, which makes the concept ill-suited for cooperative multi-agent settings or settings with centralized decision-making. Moreover, in machine learning we would like to learn strategies that are biased towards higher utility for all agents.

References

[BS18] Shane Barratt and Rishi Sharma. A note on the inception score, 2018.

[BWAA18] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

[Goo14] Ian Goodfellow. Generative adversarial nets. *NIPS*, 2014.

[JGN$^+$17] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*, 2017.

[NP06] Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[SA18] Ves Stoyanov and Necip Fazil Ayan. Under the hood: Multilingual embeddings, 2018.

[Wen17] Lilian Weng. From gan to wgan, 2017.

[WYW17] Yuting Wei, Fanny Yang, and Martin J Wainwright. Early stopping for kernel boosting algorithms: A general analysis with localized complexities. In *Advances in Neural Information Processing Systems*, pages 6065–6075, 2017.