These notes are from the Wednesday talks for the Emerging Challenges in Deep Learning workshop at the Simons Institute for the Theory of Computing, which is located at UC Berkeley.

1 Reinforcement Learning via an Optimization Lens

Lihong Li (Google Brain)

The goal of this talk was to argue that Reinforcement Learning (RL) problems should be solved with the methodology of optimization and that there are interesting research areas on the intersection of RL and optimization. It's largely based on this paper.

1.1 Background

The typical problem we solve in RL is a Markov Decision Process (MDP), which is represented by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$, where:

- $\mathcal{S}$ represents the state space of the problem.
- $\mathcal{A}$ represents the set of actions an agent can take.
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ represents the transition probabilities; given an agent in state s that takes action a, $\mathcal{P}(s, a, a')$ gives us the probability of reaching state s'.
- $R : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ gives the reward earned by an agent at a given state when doing a certain action.
- $\delta \in (0, 1)$ represents the discount factor; every round, we "value" the the next rounds rewards less by a factor of δ.

The goal is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$, which gives us an action for each state that will maximize the reward, discounted by δ. Typically, we do so by predicting *value functions* $V : \mathcal{S} \to \mathbb{R}$, which give the expected reward for each state. We can find these values with the *Bellman Equation*, which uses dynamic programming principles to recursively predict the value by examining all possible courses of action (and corresponding rewards) from the given state.

We can also express the Bellman Equation as the *Bellman Operator* $\mathcal{T}$, which is applied to a set of value function for each state to yield the corresponding value function for one additional "step" of computation. The Bellman Operator has some nice mathematical properties (it is monotonic and a δ-contraction), which implies that when applied repeatedly, it should converge to some fixed point, which will be the correct value function (that is, $V^* = \mathcal{T}V^*$, where V^* is the true value function).

1.2 What Goes Wrong

In practice, we approximate the value functions, because explicitly solving the Bellman Equation requires solving a tricky nonlinear optimization problem. However, these often-linear approximations cause the approximated Bellman Operator to lose the δ-contraction property. Empirically, we often see cases where MDPs evaluated with approximate value functions di-

verge and fail to converge to stable values.

1.3 Primal-Dual Formulation

Solving directly for V^* is equivalent to solving the following optimization problem:

$$\min_V \|V - \mathcal{T}V\|^2$$

which is equivalent to solving:

$$\min_V \mathop{\mathbb{E}}_{s \leftarrow \mathcal{S}} \left[\left(V(s) - \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \mathop{\mathbb{E}}_{s'|s,a} [V(s')] \right) \right)^2 \right]$$

However, this formulation is very difficult to solve directly. The space over V is not smooth or continuous, which makes gradient methods impossible. Also, computing a gradient requires double-sampling in order to predict the inner expectation for each state and action, which biases gradients and makes the computation inefficient.

The researchers solve these two issues with *Nestorov smoothing*, which smooths $\mathcal{T}$ by adding an entropy normalization tool while preserving the fixed point, and a *Legendre-Fenchel transform*, which introduces new variables to smooth optimization. With these two modifications, the optimization is framed in a *primal-dual formulation* (equivalently a *minimax* formulation), which involves finding a saddle point in parameter space, rather than a minimum.

The advantage of this transformation is that it can now be optimized with Stochastic Gradient Descent (SGD) rather than by relying on a dynamic programming approach to approximate value functions. Because the field of statistical ML has already intensively researched SGD and optimization, we can more rigorously analyze how we find a fixed point.

1.4 New Kernel Loss

Moving forward, a primal-dual problem is still much harder to solve than a straight-forward minimization problem with gradient methods. They are able to replace the dual variable restricting it to a Reproducing Kernel Hilbert Space (RKHS) and treating it as a constraint. With this constraint, we can remove the maximization component and find a minimum rather than a saddle point when optimizing with gradient methods.

1.5 Conclusion

Reinforcement Learning needs to go beyond tabular cases because convergence conditions for dynamic programming-based RL approaches are very brittle. Li asserts the we can get better convergence properties by shifting to an optimization framework. Algorithms will be more transparent and easier to use in practice. Li will be involved with a workshop at NeurIPS 2019 on "Optimization Foundation for Reinforcement Learning."

Question: How can Deep Q Networks be incorporated?

Answer: It was used in experiments. It translates analogously, since this framework only involves swapping out the objective function; the rest of the network can remain the same.

2 Towards Verified Deep Learning

Sanjit Seshia (UC Berkeley)

Note: Unlike most other researchers here, Seshia is not a member of the ML or theory fields of CS; he works in formal methods, and this talk is about how principles from that field can be applied to deep learning.

2.1 Overview

Machine learning is increasingly applied in safety-critical systems, and there is increased attention given to safety issues with these applications (e.g. adversarial examples and self-driving car crashes). *Formal methods* is a field of CS that consists of computational proof techniques for modeling, design, and analysis, and it is particularly applicable towards adding guarantees to deep learning algorithms. In the past, formal methods has succeeded with circuit design and processor design.

The typical formal methods flow works as follows: given a system, an environment, and a spec, the system determines whether the system applied to the environment satisfies the spec. If yes, it returns a proof. If no, it returns a counterexample to show rejection.

In deep learning, applying this framework immediately exposes several issues:

- It's difficult to model the environment needed for deep learning (particularly with self-driving car applications).
- Testing the system is difficult because neural networks are computationally expensive to train and run.
- There is no given formal specification for perception, the goal of many deep learning tasks.
- When testing, we need to search high-dimensional parameter spaces.

In order to model the context of a NN, it is necessary to create an accurate model of the environment, with which the simulated net interacts. For the rest of this talk, Seshia focuses on deep learning for perception in cyber-physical systems (e.g. self-driving cars). There are two components of the resulting formal specification:

1. *System level* — typical properties of the cyber-physical space (e.g. safety of vehicle)
2. *Component level* — properties of the neural network (e.g. robustness, monotonicity, fairness, . . .)

2.2 Robustness Property

In general, ML researchers define robustness as an optimization problem, where vulnerability to adversarial examples is a quality to minimize; here, we express it as a decision problem in order to deem a model "good enough." Typical representations of robustness refer to

local robustness, which requires slight perturbations of a correctly classified example to be classified the same. Here, Seshia challenges the focus on local robustness and argues that NNs ought to be robust across the feature space of inputs that are semantically equivalent; for example, if the color of a car is changed entirely, it would not be a considered a minor perturbation, but it should still remain classified as a car.

Seshia uses the following flow to test system-level robustness:

- Select an instance from the semantic feature space. For instance, modify the color of a car in a photo when looking at car detection by self-driving vehicles. (Note: this requires having a pre-existing model of the semantic feature space.)
- Use a renderer to construct the corresponding instance and translate it to the neural network's feature space.
- Apply the deep NN to the input.
- Determine whether the output of the NN matches the expected output based on the semantic space.

Question: How do you determine what the semantic feature space should be?

Answer: There's no true rigorous way of doing it. This depends on expert knowledge and a potentially sophisticated model of the environment around the network. It must be derived from empirical examples observed previously.

2.3 System-Level Properties

To verify system-level properties, the tester should specify the end-to-end behavior of the system, without dependence on the NN. From there, we can create safety predicates that must hold all the time. However, due to the continuous and high-dimensional inputs to the NN, it is computationally unfeasible to test all possible results of the NN. Instead, we scale up by (1) abstraction (replacing the DNN with simpler functions) and (2) component reasoning (using logic to determine what correctness results imply at different layers of the net).

For (1), we create two simplified models: an under-approximation (which represents the "best case" ML model that has access to correct labels) and an over-approximation (which under-performs the true model). From these two simplified models, we identify a region of uncertainty, which is then projected to the feature space of the NN. From this smaller region, we can find counterexamples where the safety predicate fails to hold. The search is not exhaustive; it uses gridding to translate the continuous region into a finite search space. The resulting counter-examples could be fed back to the NN to improve accuracy in these realistic cases.

Seshia's lab created SCENIC, a probabilistic programming language that designs and samples from 3D scenes. They used SCENIC to create a classifier for car recognition using badly-parked cars placed in *Grand Theft Auto V*. They also created VerifAI, a set of tools that combines the kinds of formal methods techniques described here to be used for system-level design.

2.4 Questions and Directions for Theory Community

The goal of this intersection of fields is to allow for correct-by-construction DL systems, where practitioners need not worry about robustness or other concerns. The following questions are posed to the theory community:

- Can theory of robustness in NN feature spaces be extended to semantic feature spaces?
- What proerties can we obtain theoretical guarantees for (other than local robustness)?
- Can we impose constraints from formal specifications without sacrificing efficiency?

3 Deep Equilibrium Models: One (Implicit) Layer Is All You Need

Zico Kolter (CMU Bosch)

3.1 Overview

In deep learning, we typically tell the following story: The layers are hierarchical models of input data; simple features are encoded in early layers and higher-level abstractions are represented by later layers. Here, Kolter argues that we can replace many classes of deep NNs with single-layer networks with *implicit layers* without an increase in the number of parameters.[1]

3.2 Implicit Layers in Deep Learning

A traditional *explicit layer* can be expressed as follows: $z_{i+1} = f(z_i)$, where f is a non-linear function. By contrast, an implicit layer involves using root-finding to find z_{i+1} such that $f(z_i, z_{i+1}) = 0$ for another non-linear f.[2] As an example, Kolter and his collaborators trained a model to solve Soduku with an implicit layer that solved partial SAT instances.

To find the optimal parameters for a model with an implicit layer (in place of back-propagation), we can apply the Implicit Function Theorem to compute each node's gradient by using analytically computed gradients as subroutines. In the process, we must use the root-finding algorithm to evaluate the implicit layer.

The ideas expressed here are similar to those discussed in the Neural ODEs paper. A layer which involves solving a system of ODEs can be thought of as an implicit layer, and both frameworks benefit from having lower memory costs than traditional neural networks.

[1] If this sounds too good to be true, it is; an implicit layer isn't really analogous to a traditional feed-forward layer.

[2] This was actually the original formulation of back propagation, where it was framed as an optimization problem on each layer.

3.3 Deep Equilibrium Models

When using implicit layers in a NN, we can abandon the need to think of the notion of depth as a hierarchy, and we can instead focus on finding the fixed point solution a dynamical system.

We can show how *Deep Equilibrium Models* (DEQs) are equivalent to most NNs. Note that most deep NNs can be expressed as a *weight-tied input-injected model* (where the same weights are used for each layer of the network and the network's inputs are also applied to each level). When expressed this way, deep NNs are represented as a repeated application of an operator, which results in a contraction to some equilibrium.[3] The equilibrium can be found with the root-finding approach on the implicit layer, which makes this DEQs essentially identical to infinite-layer weight-tied NNs. Kolter in particular uses Broyden's Method to find the roots of the implicit layers.

A few questions were posed by Kolter and answered:

- *Great! Should I stack DEQ layers?* No! Stacked implicit layers are equivalently represented by a single implicity layer.
- *How do I know equilibrium points exist?* You don't theoretically. But we also don't know theoretically that NNs will converge, and they do. It works empirically, and the convergence of NNs makes this possible.
- *What are the trade-offs?* Runtime still tends to be worse by a factor of 2 to 4. However, memory is far better than with traditional NNs.

3.4 DEQs for Sequential Modeling

Kolter and his collaborators applied DEQs to *sequential modeling*. In sequential modeling, we attempt to predict $y_1, \ldots, y_T$ given $x_1, \ldots, x_T$ with causal relationships (that is, y_i depends exclusively on $x_1, \ldots, x_i$). The results are close to state-of-the-art, and the absolute SOTA is likely only unreachable due to lacking computational resources.

4 The Measure and Mismeasure of Fairness

Sharad Goel (Stanford)

4.1 Overview

The talk is in response to recent work in the math and CS communities to quantify algorithmic fairness and encode those quantifications as algorithmic constraints. In this paper, Goel argues that (1) most of those proposed group-based metrics are actually poor proxies for discrimination and (2) attempts to algorithmically satisfy those criteria can have perverse effects.[4]

[3] Which is similar to the contraction discussed in the RL talk.

[4] This talk reminded me of a paper by Jon Kleinberg that argues that popular fairness metrics are inherently contradictory and that no algorithm can satisfy all formulations of fairness.

The talk focused on the application of the pretrial detention case study, where judges decide which defendants should be released prior to their court case and which should be retained in jail, with the goals of reducing economic and social costs of incarceration and reducing the likelihood of crimes committed while the defendant is free. Historically, the judges' decisions have shown steep racial bias, and in some places, an algorithm called COMPAS was used in place of judicial discretion, with the goal of more objective decisions. However, a ProPublica report argued that COMPAS perpetuated the same racial discrimination.

Here, Goel compares several methods for addressing racial bias and argues that all perform poorly here. We maintain two assumptions while thinking about this problem:

1. We have access to the true label; that is, after deciding whether to detain or release a client, we know whether the client would have committed a crime. This is *not* true in the real world, since there's no real way of confirming criminality of those detained. However, it is a necessary constraint to reason statistically about the problem.
2. Given a set of features, we can predict the true risk without bias; that is, we can estimate the probability of re-offending for any selection of defendant features that we care about.

The most natural and "fair" approach to making a decision based on a risk score is to create a threshold based on a trade-off between concerns of over-incarceration and re-offending. For instance, perhaps we detain anyone who will commit another crime with probability at least $\frac{1}{4}$. Goel argues that drawing different thresholds for different groups is unfair and presents an economic inefficiency; given two different thresholds, moving one to the other could strictly reduce the incarceration rate without impacting the overall re-offending rate.

4.2 Popular Formulations of Fairness

4.2.1 Calibration

A *calibration* requirement (which is traditionally the most popular way of enforcing some notion of fairness) involves mandating that groups should re-offend at equal rates when conditioned on the risk score. However, this bucketing by subgroups can create unequal treatment within the groups based on the distribution of risk scores within a group.

4.2.2 Classification Parity

The *classification parity* framework deems an algorithm fair if it maintains equality across certain metrics, such as the false positive rate (the percentage of people deemed high risk, who did not re-offend). In the past, much higher false positive rates were found among Black defendants, which formed the basis of the ProPublica report.

The false positive rate and other similar scores are *infra-marginal statistics*, which depend both on the selected threshold and the distribution of the risk scores. These statistics can be misleading because different distributions of risk scores by population can lead to different false positive rates even with the same threshold. Moreover, imposing a string requirement

of constant false positive rates can create adverse affects; one way to reduce the false positive rate is to increase the number of arrests of "low-risk" offenders.

4.2.3 Anti-Classification

This part of the talk was largely skipped, but anti-classification is a methodology that involves removing protected features (e.g. race and gender) and possibly correlated features (e.g. postal code) from the model. However, sometimes protected features are needed to readjust for social bias in other features. For example, omitting race while including marijuana arrests could make the model *more* racist, because the arrest of a Black person with marijuana may be less of a risk factor than the same for a white person due to the built-in racism for marijuana arrests. Without including race as a factor to contextualize marijuana arrests, we risk subjecting the model to further bias.

4.3 Bias in Data

Goel identified two sources of bias in data:

1. *Biased labels:* The proxy for re-offending rate is biased and can reinforce incorrect ideas about criminality.
2. *Biased predictors:* The features we use are biased based on how those data were collected in the past.

4.4 Conclusion

So far, mathematical definitions of equity are insufficient for handling algorithmic prejudice. They are unable to handle simple understandings of equality, such as thresholding.

5 Procurement as Policy: Administrative Process for Machine Learning

Deirdre Mulligan (UC Berkeley)

Mulligan's goal as a professor in the School of Information and the Law School is to use legal and technical standards to help advance values like fairness and transparency in the policy-making space. She aims to make government decision-making consistent with democratic norms, and she is concerned with how algorithmic decision-making often loses sight of those norms.

In one case, a potential Supreme Court about gender-based discrimination in sentencing was tossed because no one in government was able to succinctly understand how COMPAS accounted for gender. This lack of algorithmic transparency translates to the impossibility of judicial oversight. In other court cases, local governments were particularly ill-informed in how algorithmic decison-making programs worked. Biased models, metrics, and distributions account for a level of unfairness that governments have been unable to address.

Mulligan contrasted the tasks of *policy-making*, which should be an open process that allows for feedback from citizens, and *procurement*, which is the selection of contractors and can be a more opaque process, so long as no decisions of political significance are made. She argued that those technocratic decisions have become increasingly political and that oversight of government use of data and algorithms requires more openness with the procurement process.

She name-dropped a few agencies that bring in outside technical expertise for technocratic and political reasons: NHTSA, FTC, CA SOS, and USDS. She particularly noted how the USDS is being used as an internal resources for other government agencies to assist with digital needs.

She introduced the notion of *contestable design*, where the values-significant parameters of models are open to the public, where the system is explainable and decisions are apparent to users, where uncertainty is displayed in the form of confidence intervals, and where the agencies responsible are held accountable by constituents.

6 Panel on Ethics

Panelists: Sharad Goel (Stanford), Deirdre Mulligan (UC Berkeley), Emily Witt (Salesforce), Alice Xiang (Partnership on AI), Matus Telgarsky (UIUC)

Question: Should we fear the large-scale corporate control of high-impact algorithms?

Deirdre Mulligan (DM): Yes. This highlights the need for the government to take responsibility for its decisions; it cannot outsource political decisions to companies it contracts. Government already has tools at its disposal to handle issues like this; it just needs to commit.

Emily Watt (EW): It's also necessary for individual workers within corporations to hold themselves accountable.

Alice Xiang (AX): There's still a lot of policy-making that needs to be done to determine how to hold tech companies accountable. Even thought it's not optimal, we have to rely on companies self-policing in the near-term, since Washington moves so slowly.

Sharad Goel (SG): We need to change the frame of reference to think of companies as adversaries. They're orders of magnitude larger than the rest of us, and they are operating on completely different incentive structures that don't care about public welfare.

DM: It's inaccurate to say that larger companies are necessarily less responsible. Bigger companies often pay more attention to fairness and other social concerns because they have the budget to do so.

AX: Agrees with both SG and DM. Potential anti-trust action against tech companies is well-founded, and startups are often the least attentive to fairness and privacy issues. For instance, most startups ignore GDPR.

EW: Even if the companies have different values than the rest of us, the individual workers might not. Workers should be encouraged to enact change from the inside.

Question: Are researchers responsible for the ethics of projects they carry out?

EW: Yes, absolutely. It's essential to ask questions about stakeholders and contestability of algorithmic work, whether it's done in a corporate or academic setting.

DM: Wants to ask EW about the ethics committee that she's a part of at Salesforce.

EW: She's part of the Ethical and Humane Use Committee, which highlights protected variables (like race and gender) in Salesforce models and allows customers to decide whether to include or discount those variables. Surfacing these decisions to customers is particularly helpful, since most of their customers would not have thought critically about what to include in their models otherwise. They want to think more about and discuss robustness.

AX: Working on a study about best way to handle protected classes of data. She's a little concerned with the Salesforce tool that it could imply that including more protected variables implies more biased results, while sometimes the opposite is true.

Question: How much is the judicial system afraid of or resistant to using ML?

SG: There's a lot of interest in the legal sphere of algorithmic decisions, but people know very little. CS academics aren't engaged enough in bridging the knowledge gap.

AX: Wants to distinguish between two types of fears:

1. A fear of transparency: Some judges don't want data science techniques to be used to make justice system more predictable. E.g. in France, people are not allowed to use judicial analytics to predict the outcome of cases.
2. Other fears about the technicalities of algorithmic techniques that result from a lack of understanding.

Question: Should models be publically available? Should code be available on Github?

AX: Open-sourcing code can be a red herring. Code can often be tremendously opaque, and it's inaccessible to non-technical people.

Question: The Simons Institute and the organizers are incredibly lacking in diversity. How can this be addressed? [*Note:* This question was down-voted by many on the online poll. Also, there were numerous occasions in this panel with men in the audience interrupting women on the panel. This community has *a lot* of work to do.]

DM: There needs to be policies at the Simons Institute regarding how organizers are sourced and how issues of gender are handled. This is not a simple issue, and it needs to be dealt with on an institutional level.