ICML 2019 Notes
Long Beach, CA, USA

David Abel*
david_abel@brown.edu

June 2019

Contents

*http://david-abel.github.io

This document contains notes I took during the events I managed to make it to at ICML, in Long Beach, CA, USA. Please feel free to distribute it and shoot me an email at david_abel@brown.edu if you find any typos or other items that need correcting.

1 Conference Highlights

I spent most of my time at the RL sessions this round (and sadly missed all of the keynotes), so most of my reflections (and notes) are concentrated on RL:

1. Lots of great work on off-policy evaluation and off-policy learning (see, for instance, work by Hanna et al. [35], Le et al. [49], Fujimoto et al. [26], Gottesman et al. [32], and talks in Section 6.3). These problem settings are really important, as I (and many others) anticipate RL applications will come along with loads of data from sub-optimal policies.

2. Exploration was a hot topic again, and rightfully so (see work by Mavrin et al. [57], Fatemi et al. [25], Hazan et al. [37], Shani et al. [76]). Along with off-policy evaluation (and a few others), it's one of the foundational problems in RL that we're in a good position to make serious progress on at the moment.

3. Some really nice work continuing to clarify distributional RL [10] (see work by [74, 57, 67]).

4. The AI for climate change workshop on Friday was fantastic and extremely well attended (standing room only for the talks I was there for). I've said this after previous conferences, but: as we all know, there are profoundly important problems, and the tools of ML can be extremely effective in their current form.

5. I really think we need to standardize evaluation in RL. Not that we only need a single method for doing so, or a single domain, but at the moment there is far too much variance in evaluation protocols.

6. Loved the panel at the RL for real life workshop (see Section 6.2.1)

2 Monday June 10th: Tutorials

It begins! I arrived for the second half of the PAC-Bayes tutorial.

2.1 Tutorial: PAC-Bayes Theory (Part II)

The speakers are Benjamin Guedi and John Shawe-Taylor.

Part I Recap: Shawe-Taylor and Williamson [77] carried out PAC [86] analysis of Bayesian estimators (also see Figure 1. Shortly after, McAllester [58] presented the first *PAC-Bayesian* bound:

Theorem 1. *(McAllester [58]) For any prior P, $\delta \in (0, 1]$, we have:*

$$\Pr\left(\forall_{Q \in \mathcal{H}} : R_{out}(Q) \leq R_{in}(Q) + \sqrt{\frac{D_{KL}\left(Q \mid\mid P\right)) + \ln \frac{2\sqrt{m}}{\delta}}{2m}}\right) \geq 1 - \delta, \tag{1}$$

where $\mathcal{H}$ is the hypothesis space, m is the number of samples, R_{out} is the risk of a hypothesis on the test data, $R_{in}(h)$ is the risk of the hypothesis on the training data, P is the prior, and Q is the posterior.

PAC-Bayes: a flexible learning theoretic framework! Tight connections to regression, linear classification & SVMs, transductive learning, uses in RL [24], and more.

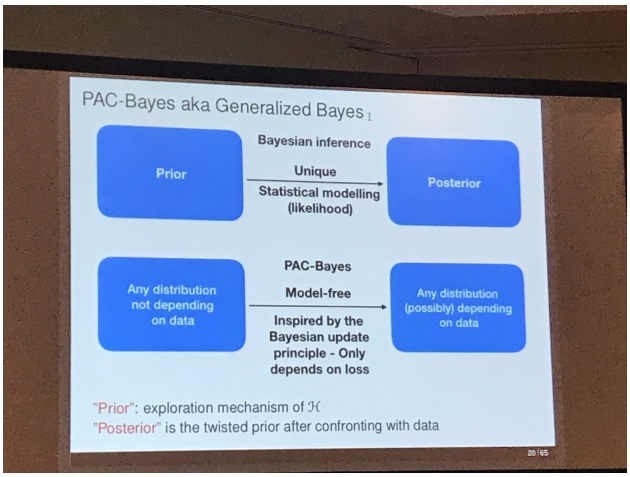


Figure 1: Differences in Bayes and PAC-Bayes

2.1.1 PAC-Bayes Theory

Q: How can PAC-Bayes drive learning?

A: First, recall:

$$R_{out}(Q) \leq R_{in}(Q) + F(Q), \tag{2}$$

Or:

$$\text{Error on unseen data} \leq \text{Error on sample} + \text{complexity term}. \tag{3}$$

This defines a principle strategy for obtaining new algorithms:

$$h \sim Q^* \tag{4}$$
$$Q^* \in \operatorname{arginf}_{Q \ll P} \left\{ R_{in}(Q) + F(Q) \right\}. \tag{5}$$

Presents an optimization problem: can be solved or approximated to find good solutions!

PAC-Bayes interpretation of celebrated algorithms:

- SVM with a sigmoid loss and KL-regularized Adaboost can be reinterpreted as minimizer of PAC-Bayesian bounds [5].
- Also, the minimizer of:
$$\left\{ R_{in}(Q) + \frac{KL}{\lambda} \right\},$$
is the celebrated Gibbs posterior:
$$Q_\lambda(h) \propto \exp(-\lambda R_{in}(h)) P 9h), \quad \forall_{h \in \mathcal{H}}.$$
When $\lambda \to 0$, we get a flat posterior, and as $\lambda \to \infty$, we get Dirac mass on expected risk minimization (ERMs).

Theorem 2.
$$\log \int \exp \phi dP = \sup_{Q \ll P} \left\{ \int \phi dQ - D_{KL}(Q \mid\mid P) \right\}.$$

Proof. First require variational definition of KL-Divergence [18]

$$-D_{\mathrm{KL}}(Q \mid\mid G) = -\int \log \left(\frac{dQ}{dP} \frac{dP}{dG} \right) dQ \tag{6}$$
$$= -D_{\mathrm{KL}}(Q \mid\mid P) + \int \phi dp - \log \int \exp \phi dP. \tag{7}$$

Note that KL> 0, $Q \to -D_{\mathrm{KL}}(Q \mid\mid P)$ reaches its max in $Q = G$. Thus, taking $\phi = -\lambda R_{in}$:

$$Q_\lambda(h) \propto \exp(-\lambda R_{in}(h)) P(h), \quad \forall_{h \in \mathcal{H}}. \qquad \square$$

Q: What about non-i.i.d. data?

A: Sure! Let's drop the i.i.d. and bounded loss assumptions. First, need moments:

Definition 1 (Moment): *The p-th moment of a distribution is given by:*

$$M_p := \int \mathbb{E}\left[|R_{in}(h) - R_{out}(h)|^p \right] dP(h).$$

Also make use of f-divergences, a generalization of the KL-Divergence.

Theorem 3. *Let $\phi_p : x \mapsto x^p$. Fix $p > 1, q = \frac{p}{p-1}$ and $\delta \in (0,1)$. W/ probability at least $1-\delta$, for any distr. Q:*

$$|R_{out}(Q) - R_{in}(Q)| \leq (\frac{M_q}{\delta}^{1/q} (D_{\phi_{p-1}}(Q,P) + 1)^{1/p}.$$

Takeaway: we can bound generalization error using the f-divergence ($D_{\phi_{p-1}}$ and moment (M_q). Proof strategy requires: 1) Jensen's inequality, 2) change of measure, 3) Holder's inequality, and 4) Markov's inequality.

Oracle Bounds; Catoni [13] derived PAC-Bayesian bounds for the Gibbs posterior.

2.1.2 PAC-Bayes and Task Awareness

Note: PAC-Bayesian bounds express a trade-off between empirical accuracy and a measure of complexity.

Q: So, how can we improve the bounds we get? How do we choose the right prior distribution so that we can 1) control complexity, and 2) ensure good performance?

→ So: can we choose a "better' prior? (without looking at the test data itself?)

Main Idea: use part of the data to learn how to choose a prior.

Can use PAC-Bayes in SVMs:

- Assume prior and poster are spherical Gaussians (w/ prior centered at origin, posterior centered at a scaling μ of unit SVM weight vector).
- Implies that KL term in generalization error bound is $\mu^2/2$ (see Theorem 1).
- Can compute stochastic error of posterior distribution behaves like a soft margin, scaling μ trades between margin loss and KL.
- Bound holds for all μ, so choose μ to optimize the bound.

Q: But how do we learn the prior for SVMs?

- Bound depends on distance between prior and posterior
- Better prior means tighter bound
- Idea: learn prior P with part of the data.
- Introduce learnt prior in the bound.
- Computer stochastic error with remaining data: PrPAC.
- Can go a step further: 1) scaling the prior in the chosen direction τ-PrPAC, or 2) Adapt SVM to optimize the new bound: $\eta-$Prior SVM.

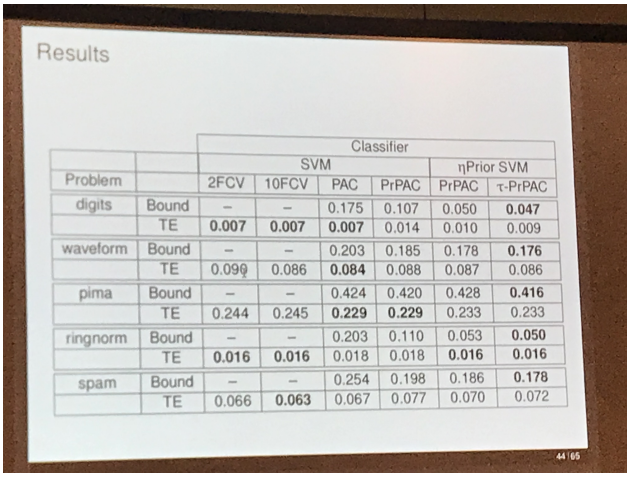


Figure 2: Results from applying different PAC-Bayes prior selection methods to experiments.

Results from the above methods for tightening the bounds: see Figure 2.

Takeaways from results:

1. Bounds are remarkably tight!
2. Model-selection via these new bounds is *as good* as 10-fold cross validation.
3. Best bounds don't necessarily translate into best model selection.

   → We're not *completely* capturing the right thing (but definitely some of the right thing).

Next up: distribution-defined priors:

- Consider P and Q are Gibbs-Boltzmann distributions:

$$P_\gamma(h) = \frac{1}{Z}\exp(-\gamma R_{out}(h) \qquad Q_\gamma(h) = \frac{1}{Z}\exp(-\gamma R_{in}(h)).$$

- These distributions hard to work with since we can't apply it to a single weight vector. From Catoni [13], we can show:

$$D_{\mathrm{KL}}\left(R_{in}(Q_\lambda) \mid\mid R_{out}(Q_\lambda)\right) \tilde{\leq} \frac{1}{m}\left(\gamma/\sqrt{m} + \gamma^2/4m\right),$$

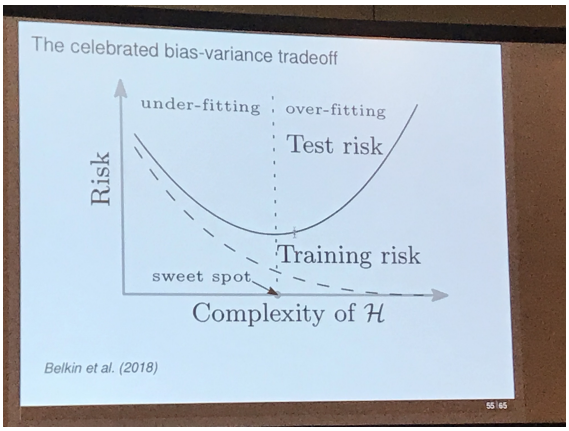
  where $\tilde{\leq}$ ignores log terms on the right hand side (Dave: my (awful) construction for abbreviation, not theirs).

On stability:

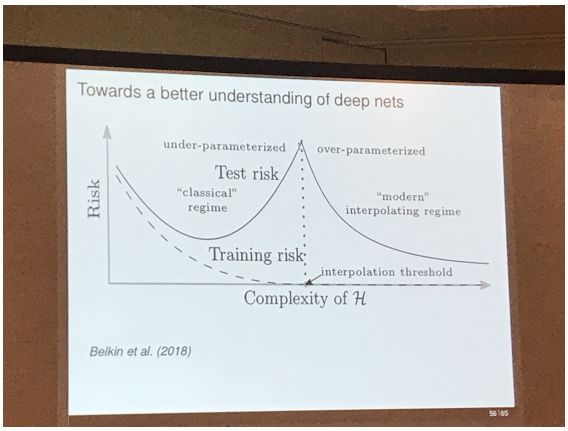
- If A has sensitivity β at sample size m, then generalization error can be bounded [11].
- Q: Algorithm output is highly concentrated, does that imply stronger results?

  → Yes! We can derive (tighter) bounds that depend on KL between a distributionally sensitive prior and a well chosen posterior.

  Open area! Lots of room to explore a new kind of generalization error analysis.



(a) Classical view of Overfitting



(b) New Paradigm?

Figure 3: Classical view of overfitting (left), and a new proposal for why deep nets might be avoiding overfitting (right), from Belkin et al. [9].

A final case study: Can we use any of this to analyze deep neural networks?

Q: Is deep learning breaking the statistical paradigm we know?

→ Neural nets trained on massive data sets achieve *zero training error*, which does not bode well for their performance. Yet! They tend to achieve remarkably low error on test sets, too.

Idea: Perhaps we can use PAC-Bayes to explain this phenomena.

Dziugaite and Roy [22] derived extremely tight deep learning generalization error bounds in this way;

- Based on training to expand the "basin of attraction"
- Hence, not measuring good generalization of *normal training*

Q: How much information is contained in the training set?

A1: Achille and Soatto [1] studied the *amount of information* stored in the weights of deep networks. Overfitting might be related to information being stored in weights that encode training set, as opposed to the data generation distribution.

A2: Information bottleneck criterion [] might control this information, and could lead to a tighter PAC-Bayes bound.

Conclusions:

- PAC-Bayes arises from two fields: 1) statistical learning theory, and 2) Bayesian Learning.
- Generalizes both fields and points to promising directions.

- PAC-Bayes theory can be an inspiration toward new theoretical analysis, but also drive algorithm design (especially when theory has proven difficult).

.....

2.2 Tutorial: Meta-Learning

The speakers are Chelsea Finn and Sergey Levine.

Motivation: Learn from small amounts of data.

→ Recent advancements *thrive* in large diverse data sets in the sense that it allows for broad generalization) (see: BERT, AlexNet)

→ Existing approaches require huge data sets. But, some questions:

1. What if we don't have a large data set?
2. What if we want a general purpose AI system in the world?
3. What if our data has a long tail?

Point: these settings start to break the standard supervised learning setting.

Example: few shot learning with painting, people (the audience) was able to "generalize" to guess the painter of a new painting.

Q: How do we accomplish this?

A: Well, previous experience! We weren't really doing this based on no prior experiences. We have encountered similar questions/tasks/images before.

Q: How might we get a machine to accomplish this task?

A: Well, we might encode structure via: {modeling image formation, geometry, task-specific features, hyperparameter choice}, and so on

Main Point: Can we explicitly learn priors from previous experience that lead to efficient downstream learning?

Outline:

- Problem statement
- Meta-Learning Algorithms
- Meta-Learning Applications
- Meta-Reinforcement Learning

2.2.1 Two Ways to View Meta-Learning

Q: How do we formulate the meta-learning problem?

A1: Mechanistic view! A model reads in an entire data set and makes predictions for new datapoints. Training this network uses a "meta"-dataset which itself consists of many datasets.

A2: Probabilistic view: extract prior info from a set of (meta-training) tasks that allows efficient learning of tasks. Learning a new task uses this prior training set to infer most likely posterior parameters.

→ A1 is more convenient for implementation, A2 is more convenient for understanding.

Definition 2 (Supervised Learning): *Find the parameters ϕ given data D:*

$$\arg\max_{\phi} \Pr(\phi \mid D),$$

where $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Applying Bayes rules, this comes out to:

$$\arg\max_{\phi} \log \Pr(D \mid \phi) + \Pr(\phi).$$

Definition 3 (Meta-Learning): *Find the parameters θ that enables us to quickly solve new tasks, given a bunch of datasets $D_{\text{meta-train}}$:*

$$\arg\max_{\phi} \log \Pr(\phi \mid D, D_{\text{meta-train}}) = \arg\max_{\theta} \log \int_{\Theta} \Pr(\phi \mid D, \theta) \Pr(\theta \mid D_{\text{meta-train}}).$$

That is, we assume θ are the sufficient statistics for ϕ.

Hard to perform this decomposition in general: usually take a Maximum A Posteriori (MAP) approach.

Goal of Meta-Learning: Find an appropriate set of parameters θ, that are maximially probable given the meta-datasets. So:

$$\theta^* = \arg\max_{\theta} \log \Pr(\theta \mid D_{\text{meta-train}}).$$

Notably, $D_{\text{meta-train}}$ might consist of *different* tasks (with related structure).

Example: Want to classify new data sets. First, do meta learning;

$$\theta^* = \arg\max_{\theta} \log \Pr(\theta \mid D_{\text{meta-train}}).$$

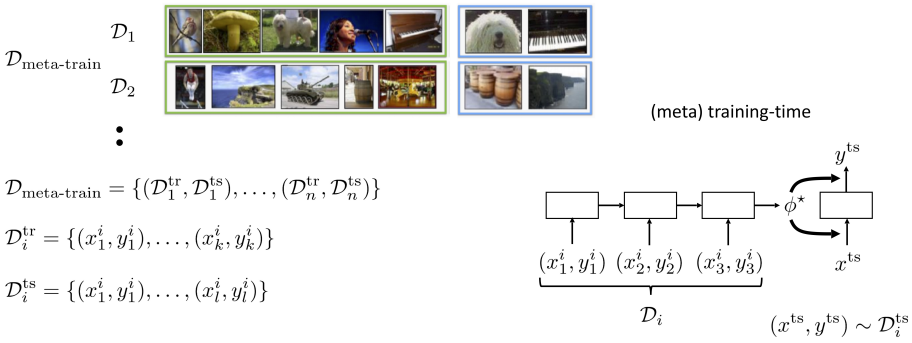


Figure 4: Meta Learning paradigm

Then, adapt and find some nearby parameters, ϕ^*:

$$\phi^* = \arg\max_{\phi} \log \Pr(\phi \mid D, \theta^*).$$

Key Idea: Our training procedure is based on a simple machine learning principle: test and train conditions much match" (see work by Vinyals et al. [88]).

Thus, for our above notion of meta-learning to make sense, we also need to hang on to tests *for each training* task.

Therefore, meta-learning can be written as:

$$\theta^* = \max_{\theta} \sum_{i=1}^{n} \log \Pr(\theta_i \mid D_i^{\text{test}}.$$

Closely related problem settings:

1. Multi-task learning: special case of meta-learning where $\theta = \phi$.
2. Hyperparameter optimization: can also be cast as meta-learning with θ=hyperparameters and ϕ=network weights.

2.2.2 Meta-Learning Algorithms

Before discussing algorithms, let's discuss *evaluation*.

Main Idea: See Lake et al. [48], introduced the Omniglot dataset. Idea: lots of classes, few examples of each class. Consists of 1623 characters from 50 different alphabets, 20 instances of each character.

→ Proposes both few-shot discriminative and few-shot generative problems. Initial few-short learning approaches w/ Bayesian models, non-parameterics. Other such data sets: CIFAR, CUB, Mini-ImageNEt.

Q: How do we evaluate a meta-learning algorithm?

A: Perform usual meta-training, and determine whether the resulting model can still perform quick generalization across different (held-out) tasks.

****General Recipe for Meta-Learning Algorithms:**

1. Choose a form of $\Pr(\phi_i \mid D_i^{\text{train}}, \theta)$.
2. Choose how to optimize θ with respect to max-likelihood objective using $D_{\text{meta-train}}$.

Approach 1: Black-box adaptation. Key idea is to train a *neural net to represent* $\Pr(\phi_i \mid D_i^{\text{train}}), \theta)$.

$\rightarrow$ For instance, might use an RNN to represent f_θ, given a bunch of these meta-training data sets. Then, we can train with standard supervised learning:

$$\max_\theta \sum_{T_i} L(f_\theta(D_i^{\text{train}}, D_i^{\text{test}})).$$

Challenge: Outputting all neural net parameters does not seem scalable $\rightarrow$ But, we don't need to output all parameters of a neural not, just the sufficient statistics.

Q: How can we frame this as an optimization procedure?

Approach 2: Acquire ϕ_i through optimization:

$$\max_{\phi_i} \log \Pr(D_i^{\text{train}} \mid \phi_i) + \log \Pr(\phi_i \mid \theta).$$

Meta-parameters θ serve as a *prior*. What form of prior? One form that has been successful: learn θ from *other tasks*.

Goal: Learn a parameter vector θ that will transfer effectively, in the sense that it makes fine-tuning on new tasks easy/useful. To do so, solve the following problem;

$$\min_\theta \sum_i L(\theta - \alpha \nabla_\theta L(\theta, D_i^{\text{train}}), D_i^{\text{test}}).$$

General algorithm:

1. Sample task T_i.
2. Sample disjoint data sets from D_i
3. Optimize $\phi_i \leftarrow \theta - \alpha \nabla_\theta L(\theta, D_i^{\text{tr}})$
4. Update θ using $\nabla_\theta L(\theta, D_i^{\text{train}})$.

So, we're left with two approaches: Optimization vs. Black-Box Adaptation

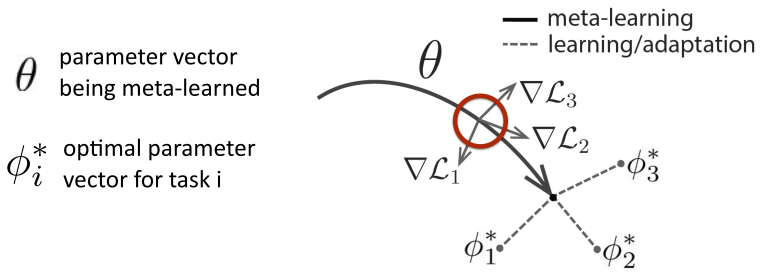


Figure 5: Model-Agnostic Meta-Learning

→ Model-Agnostic Meta-Learning (MAML), in the optimization can, can be viewed as a computation graph with embedded gradient operator (see Figure 5:

$$y^{ts} = f_{\text{MAML}}(D_i^{\text{train}}, x^{\text{test}} \tag{8}$$
$$= f_{\phi_i}(x^{ts}), \tag{9}$$

where $\phi_i = \theta - \alpha\nabla_\theta L(\theta, D_i^{\text{train}})$.

Experiments: Compare MAML with black-box approaches (SMAIL, MetaNetworks). Tend to find higher performance in domain adaption/extrapolated tasks.
→ Task was to shear digits in the Omnioglot data set by some degree and inspect decay on performance. MAML rarely decays in performance.

Q: Does this learned structure come at a cost?

A: Not really! MAML function can approximation any function of D_i^{train}, x^{test}. Assumptions: nonzero α, loss function does not lose information about the label, data points in D_i^{train} are unique.

∴ MAML has benefit of inductive bias without losing expressive power.

Another approach: probabilistic interpretation of optimization-based inference. Find: MAML is a form of *implicit* prior, roughly gradient-descent + early stopping, comes out to an implicit Gaussian prior.

Q; Other forms of priors expressible in meta-learning?

A: Sure! Bayesian linear regression on learned features, or closed-form/convex optimization (as in ridge or logistic regression).

Challenges:

- How do we choose an architecture that is effective for inner gradient step? → But, can do progressive neural search (w/ MAML) to overcome this.
- Second-order meta-optimization can exhibit instabilities. → Lots of solutions available: 1) optimize only inner subset, 2) decouple learning rate, 3) introduce context variables.

Approach 3: Non-parametric methods. In low data-regimes, non-parametric methods are simple and tend to work well.

→ During meta-test time: few shot learning ≡ low data regime. During meta-training, still want to be parametric.

Q: Can we use parametric meta-learners that produce effective non-parametric learners?

A: Yes! Use non-parametric learners by comparing test data with training images.

Key Idea: learn a metric space that leads to more effective comparisons and predictions at test time.

Takeaways: Each approach has some advantages/disadvantages, listed in Figure **??**.

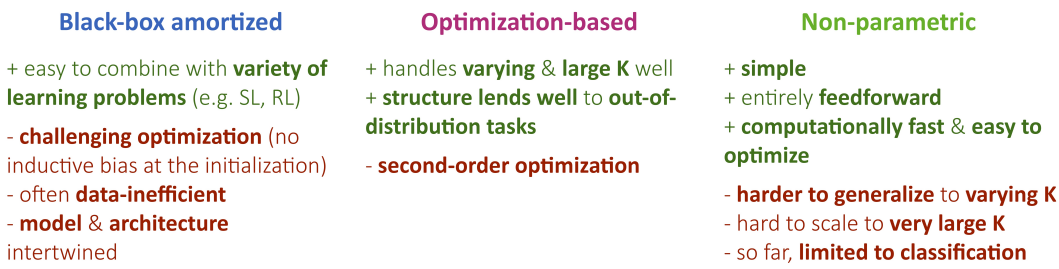


Figure 6: Advantages and disadvantages of different approaches to meta-learning.

Approach 4: Bayesian Meta-Learning.

Assume we have a parameter prior $\Pr(\theta)$, $\Pr(\phi_i)$, can we sample $\phi_i \sim \Pr(\phi_i \mid x_i^{\text{train}}, y_i^{\text{train}})$.

Simple Idea: use neural net to produce a Gaussian distribution over h, with h some subset of relevant weights of the network (like the last layer).

Q: Okay, but what about Bayesian optimization-based meta-learning?

A: Sure! Lots of ways to do this. One idea is to model $\Pr(\phi_i \mid \theta)$ as Gaussian, perform variational inference for training (see Ravi and Beatson [70]). Another approach: do gradient-based inference on last layer only, use SVGD to avoid Gaussian modeling assumption (see work by Liu and Wang [54]).

→ Key Idea; Approximate $\Pr(\phi_i \mid \theta, x_i^{\text{train}}, y_i^{\text{train}})$ with a MAP inference. Very crude, but very convenient!

Further reading; Garnelo et al. [28], Kim et al. [46], Ravi and Beatson [70].
Applications:

- Vision: Few-shot image generation, image-to-image translation, generation of novel viewpoints.

- Imitation Learning/RL: one-shot inverse RL, optimization based inference given demonstrations.
- Language: Adapting to new programs, adapting to new languages, adapting dialogue agents to new personas.

2.2.3 Meta-Reinforcement Learning

Q: Why expect Meta-RL to be useful?

A: Well, major challenges in RL! Almost all related to the sample inefficiency of existing methods. Something TRPO, applied to a real robot, would take on the order of days or weeks for a robot to begin to make any kind of progress (in learning to walk). First, some background;

Definition 4 (Markov Decision Process (MDP)): *An MDP is a four tuple:* $\langle, \mathcal{S}, \mathcal{A}, R, P\rangle$*, with* $\mathcal{S}$ *a set of states,* $\mathcal{A}$ *is an action set,* $P : \mathcal{S} \times \mathcal{A} \to \Pr(\mathcal{S})$ *denotes the transition function, and* $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ *denotes the reward function.*

The goal is for an agent to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes long term expected reward:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\pi_\theta}\left[R(\tau \mid \pi_\theta)\right], \tag{10}$$

where τ is the trajectory taken by π_θ, and $R(\tau \mid \pi_\theta)$ picks out the rewards achieved by π_θ.

"Every RL algorithm in a nutshell": finds π_θ either by: 1) learning a good policy directly, 2) learning a value function, or 3) learning a model and using it to find a good policy.

Meta-Learning so far: Learn θ uch that $\phi_i = f_\theta(D_i^{\text{train}})$ is good for the test D_i^{test}.

So, meta-RL problem is as follows;

Definition 5 (Meta-RL): *The Meta-Learning problem with the RL objective. That is, learn* θ^**:*

$$\theta^* = \arg\max_\theta \sum_{i=1}^{n} \mathbb{E}_{\pi_{\phi_i}}\left[R(\tau \mid \pi_{\phi_i})\right],$$

where $\phi_i = f_\theta(M_i)$

Q: So how do we produce M_i, the meta-training MDPs?
A: One idea is to just choose related tasks (imagine we want a household chore robot, define M_i's to be small and easy related tasks

Now, some meta-RL algorithms: similarly to supervised meta-learning, we also get a meta-RL based on black box approaches (here called "recurrent policies").

Main Question: how do we implement $f_\theta(M_i)$, with M_i an MDP? Well, what should $f_\theta(M_i)$ do?

A: A few things: 1) improve policy with experience from M_i, and 2) Choose how to interact (that is, meta-RL must choose how to explore).

$\rightarrow$ This mostly amounts to just running meta-RL with an RNN.

Q: But how do meta-RL algorithms explore effectively? This basically happens for free in this setting: optimizing reward over all episodes used in training leads to good exploration. Dave: I don't see this, personally. Need to read more!

Next View: Treat meta-RL as an optimization problem. Standard RL formulate this as a policy gradient problem;

$$\theta^* = \arg\max_\theta \underbrace{\mathbb{E}\left[R(\tau)\right]}\, J(\theta),$$

then:

$$\theta^{k+1} \leftarrow \theta_k + \alpha \nabla_{\theta_k} J(\theta)$$

Next approach: meta-RL for partially observable RL! Need a richer environmental model:

> **Definition 6** (Partially Observable MDP (POMDP)): *A POMDP is a six tuple: $\langle S, A, O, P, E, r\rangle$, that is effectively an MDP, but the function E generates observations from O based on the current state. The agent only gets to perceive observations $o \in O$, not state.*

That is, the agent doesn't get to see everything relevant about the world, just some observation o generated based on the state.

Key Idea: Solving POMDPs is very hard! But, it's similar to meta-learning. Look for a policy $\pi_\theta(a \mid s, z)$, where we don't know s, only z. Works as follows: 1) Sample $z \sim \hat{p}(z_t \mid s_{1:t}, a_{1:t}, r_{1:t})$, 2) Act according to $\pi_\theta(a \mid s, z)$, acting as though z was correct.

Three perspectives on Meta-RL:

1. Just RNN it: conceptually simple and easy, but vulnerable to overfitting and challenging to optimize.
2. MAML approach: good extraploating but complex (requires many samples).
3. POMDP approach: simple and effective, but also vulnerable to meta-overfitting.

But: they're not that different! The POMDP approach is really just the RNN approach with an extra hidden variable, and the MAML approach is just a particular choice for the design of f_θ.

2.2.4 Challenges and Frontiers in Meta Learning

Lots of exciting directions forward, but lots of challenges, too.

Challenge 1: Meta-overfitting—meta learning requires a task distribution. Some approaches to meta-learning can *overfit* to these task distributions.

Challenge 2: Task design—often these task distributions have to be chosen by hand, or are not diverse enough to encourage the right kind of behavior. It's hard to pick the task distribution in the right way!

Challenge 3: Understanding which *algorithms* overfit—many difference approaches (black-box, optimization-based, non-parametric), but we don't have a sense of which kinds of algorithms are most vulnerable to meta-overfitting.

Q: What else can we do with meta-learning?

A1: Well, we might automatically propose tasks that generate the appropriate task distributions. $\Longrightarrow$ "Unsupervised meta learning", which refers to algorithms that learn to solve tasks efficiently without using hand-specified task distributions or labels during meta-training.

Challenge 4: Memorization! Some algorithms may end up just memorizing solutions to the relevant test tasks.
Challenge 5: How do we determine which task information should be in the input vs in the data? Broad task distributions can be useful, but makes exploration hard. If their too narrow, not representative enough.

Dave: Had to run for the day!

.....

3 Tuesday June 11th: Main Conference

I arrived today for the best paper award talk.

3.1 Best Paper Talk: Challenging Assumptions in Learning Disentangled Representations

Talk by Francesco Olivier Bachem.

Central Question: Can we learn disentangled representations in a purely unsupervised way?

Context: Representation learning. Consider a picture of a landscape. Goal of representation learning is to learn a function $f : \mathcal{X} \to \Phi$, that translates each item of interest (say, images), into a set of features that capture the important characteristics about the images.

Q: What might we want to have in a representation?

A: lots of things! One idea, though: *disentanglement.*

Definition 7 (Disentanglement): *A single change in a factor should lead to single change in representation.*

Example: Consider geometric shapes placed in a scene, each of a different color/shape/size. If we vary one key aspect (a square's color or size, for instance) of the original image, the representation should change in exactly one way, too.

Focus: Unsupervised learning to disentangle representations (so, critically, don't get to observe the ground truth factors of variation).

Contributions:

1. *Theoretical result:* for arbitrary data, unsupervised learning of disentangled representations is impossible.
2. *Large-Scale Experimental Study:* Can we learn disentangled representations without looking at the labels?

   → Can reconcile and still be effective! The theorem is about *worst case* data. Our data likely has the right structure that we can still learn these disentangled representations.

Experiments: six methods, seven data sets, six metrics, yielding 10,000 trained models and 150,000 scores.

→ Questions to focus on:

Q1 Which method should be used?

Q2 How do we choose the hyperparameters?

Q3 How to select the best model from a set of trained models?

Experimental findings:

1. (Toward answering Q1): Random seed and hyperparameters seem to matter more than the choice of the objective function.
2. (Toward answering Q2): Can't identify obvious trends that can be used as a rule-of-thumb for hyperparameter selection (but, transferring across tasks seems to help). → Transfer of hyperparameters to different data set and metric does not seem to substantially
3. (Toward answering Q3): Unsupervised model selection remains a key challenge!

→ Implication: good runs from bad hyperparameter settings can easily outperform bad runs from good hyperparameter settings.

Key Takeaways:

- Role of inductive biases and supervision should be made explicit
- Concrete practical benefits of disentanglement should be demonstrated
- Sound, reproducible experimental setup with several data sets is crucial.
- Code: https://github.com/google-research/disentanglement_lib/. Also released 10,000 pre-trained models.

.....

3.2 Contributed Talks: Deep RL

Next we have 5 minute contributed talks. They're very short so I suspect I'll have a harder time transcribing.

3.2.1 DQN and Time Discretization [82]

The speaker is Corentin Tallec.

Point: Time discretization actually matters in deep RL!

→ Usual RL algorithms with high frame rate leads to failure. Scalability often limited by algorithms; better hardware, sensors, actuators, can lead to new hyperparameters.

Q: Why is near continuous Q-learning failing?

A: As δ_t (time discretization), $Q^\pi(s,a) \to V^\pi(s)$. As a result, at the limit, Q no longer depends on actions! So we can't perform policy improvement.

Q: Can we solve this?

A: Yes! Can look at all these quantities, form new algorithms (for more, see the paper!).

.....

3.2.2 Nonlinear Distributional Gradient TD Learning [67]

The speaker is Chao Qu.

Background: Distributional RL considers stochastic nature of long term return, $Z(s,a)$.

This Work: Consider a distributional counterpart of Gradient TD Learning. Properties:

- Converges in off policy
- Converges with non-linear function approximation.

$\to$ New algorithm: Distributionl GTD2. Uses temporal distribution difference instead of the temporal difference from GTD2.

Theorem 4. *Under mild assumptions, Distributional GTD2 (D-GTD2) converges in the limit.*

Experiments: Show that D-GTD2 does in fact converge.

.....

3.2.3 Composing Entropic Policies using Divergence Correction [38]

Talk by Jonathan Hunt.

Q: How do people solve complex motor control tasks such as juggling and unicycling at the same time?

A: Well, perhaps we tend to learn each of them independently, and then merging them together.

Note: not option like! We don't do A then B, we do a new $A \circ B$ task.

Problem: Given training tasks T_1 and T_2, we want to solve some merged task $T_b = T_1 + T_2$.

Prior work: generalized policy improvement [8], and compositional optimism [**?**].
Adapt existing ideas to this transfer setting:

1. Successor Features
2. Generalized Policy Improvement

3. Divergence correction

→ New algorithm for composing tasks in continuous action spaces.

.....

3.2.4 TibGM: A Graphical Model Approach for RL [2]

The speaker is Tameem Adel.

Genesis: Graphical modeling approach for 1) increasing transferability generalization, and 2) adopt precise objectives and clear interpretations.

→ Do this by proposing an information theoretic objective aiming at maximizing "local" reward and a more global measure (related to transfer).

Contributions:

- A graphical model based on
- Prove a correspondence between new objective and typical reward maximization objective.
- An information theoretic pre-training procedure focusing on exploration.
- State of the art results on 16 benchmark tasks.

.....

3.2.5 Multi-Agent Adversarial IRL [93]

The talk is by Lantao Yu.

Motivation: Performance of RL agent relies on quality of reward function.

→ But, it can be hard to design the right reward function!

One solution: learn a reward function from expert demonstrations (as in imitation learning).

Q: But why should we care about reward learning?

A: Well, some advantages: 1) scientific inquiry (understanding animal behavior), 2) reward function is the most succinct, robust, and transferable description of a task, 3) can be helpful if we want to re-optimize policies in new environments.

→ These properties are even more desirable in multi-agent setting!

Definition 8 (Single-agent IRL): *Find a reward function that explains expert behavior*

But: intractable! Too many reward functions explain the same behavior.

Generalize this setting to multi-agent using Markov Games:

Definition 9 (Markov Game): *A multi-agent generalization of MDPs [52]*

Solution concept to a Markov Game is a Nash Equilibrium:

Definition 10 (Nash Equilibrium): *When no agent can achieve higher expected reward by changing its own policy.*

Method: introduced Logistic Stochastic Best Response Equilibrium (LSBRE). Optimize the pseudo-likelihood objective.

Experiments: Policy imitation performance. New method achieves state of the art in both cooperative/communicative and competitive tasks.

Summary: New solution concept for Markov games, gives rise to new measures of interest. Propose the first multi-agent MaxEnt IRL framework.

.....

3.2.6 Policy Consolidation for Continual RL [44]

Motivation: Catastrophic forgetting in neural nets! Nets tend to forget information from prior tasks.

→ Even happens in RL during continual learning, since distribution of seen states change over time as exploration/policy changes.

Agents should cope with: discrete and continuous changes to data distribution!

Contribution: Policy Consolidation agent. A bunch of agents are trained (via PPO) and connected via a KL distillation loss.

→ Final policy stored is the result of distilling all other policies trained. Ensures it doesn't deviate too much from prior performance.

Experiments: Alternating task to explore effect of forgetting. Find their algorithm outperforms all other variants of PPO.

Future Work: 1) Look at how to prioritize important memories during training, and 2) Adapt for off-policy learning.

3.2.7 Off-Policy Evaluation Deep RL w/o Exploration [26]

Consider: same off-policy algorithm (DDPG) used on same dataset with two differen approaches (orange and blue).

Agents: 1) Orange—interacts with the environment (standard rl loop), 2) Blue—just uses data from environment but doesn't actually interact! Yet, their performance is different

Q: Why would these be any different?

A: Extrapolation error: $Q(s, a,) \leftarrow r = \gamma Q(s', a')$, where (s, a, r, s') come from the dataset. Basically, you might have a bad target $Q(s', a')$.

Definition 11 (Extrapolation error): *Attempting to evaluate π without sufficient acccess to the (s, a) pairs π visits.*

Solution: batch-constrainted RL: only choose π so that it selects (s, a) pairs that are in the dataset and maximizes performance.

→ New algorithm: Batch-constrained Deep Q Learning (BCQ).

1. Imitate dataset via generative model
2. $\pi(s) = \arg\max_a Q(s, a)$
3. Throw in some extra magic

Finds that BCQ greatly outperforms some existing methods (DDPG), and is quite stable.

.....

3.2.8 Random Expert Distillation [90]

Talk by Ruohan Wang

Definition 12 (Imitation Learning): *Policy learning from a limited set of expert demonstrations*

Useful because: intuitive and efficient skill transfer, and can capture style/preferences of individual demonstrators.

→ Recent framework for IRL: Generative Adversarial Imitation Learning (GAIL).

But, optimization challenges; 1) training instability, and 2) sample inefficiency.

Main Contribution: Random Expert Distillation (RED). Framework fir imitation learning using the estimated support of the expert policy as reward.

Optimize a new trajectory based loss based on a new reward function, and prove it approximates the right thing in the limit.

Experiment 1: In MuJoCo, find high training stability and good sample efficiency compared to other approaches like GAIL.

Experiment 2: Autonomous driving tasks with human actions as training data. Agent learns to follow the preferences of the trainer (speed, lane preference, and so on).

.....

3.2.9 Revisiting the Softmax Bellman Operator [79]

The speaker is Zhao Song.

Recall that the Bellman Operator is a contraction.

→ Mellowmax operator [7] is also a contraction:

$$max_{a'} Q(s', a') \to \sum_{a'} \frac{\exp \tau Q}{\sum_b \exp \tau Q(s', b)}$$

Q: Is softmax really as bad as sour milk? (credit to Ron Parr).

Idea: combing the max function in the target network of DDQN with softmax.

→ Gives rise to the SQDN, achieves higher scores than DQN on most Atari games.

Thus: analysis of softmax is warranted!

Theorem 5. *Analysis showing: good performance, guaranteed convergence, smaller error, reduction bound, and overestimation error monotically increases w.r.t. τ (inverse temperature).*

.....

3.3 Contributed Talks: RL Theory

More RL!

3.3.1 Distributional RL for Efficient Exploration [57]

Talk by Hengshuai Yao.

Point: Lots of sources of uncertainty in RL—estimation (in finite data regime), but also other factors like environmental stochasticity, opponents' play in game, and so on.

→ Exploration strategy based on uncertainty: *optimism under uncertainty*. In multi-armed bandits, choose arm according to:

$$a = \arg\max_k \hat{\mu}_k + c_k,$$

where $\hat{\mu}_k$ is estimated mean payoff, and c_k is some measure of uncertainty in this estimate (UCB like).

But! Using naive exploration bonus doesn't always work: favors actions with high intrinsic uncertainty forever.

New Idea: To use optimism in face of uncertainty, perform decaying schedule of the weight of uncertainty. This ensures that as more evidence is gathered, the agent starts to exploit more.

.....

3.3.2 Optimistic Policy Optimization via Importance Sampling [62]

Talk by Matteo Papini.

Problem: Policy Optimization.

- Parameter space $\Theta \subseteq \mathbb{R}^d$
- Parapmetric policy for each $\theta \in \Theta$
- Each policy induces a distr. over trajectories, with $R(\tau)$ the return of the trajectory.
- Goal is to find he parameters that maximize the expected return:

$$\max_{\theta \in \Theta} \mathbb{E}[R(p_\tau \mid \theta)].$$

Challenge, though: exploration here is hard!

Q: What if we think about this as a multi-armed bandit?
Then:

- Arms: parameters θ
- Payoff: expected return $J(\theta)$
- Continuous Multi-Armed Bandit, effectively.
- Take advantage of arm correlation through trajectory distributions.

- Use Important Sampling to update return of a candidate policy.

Main result: new algorithm "OPTIMIST", enjoys sublinear regret:

$$Regret(T) - \tilde{O}(\sqrt{dT}).$$

Code: https://github.com/wolfLo/optimist

.....

3.3.3 Neural Logic RL [41]

The talk is by Shan Luo.

Two main challenges of deep RL: 1) How can we generalize learned policies from one task to another?, and 2) How can we interpret these learned policies?

Approach: use background knowledge to learn concepts and relations, like `grandfather(x,y)`. To do so, use *differentiable inductive logic programming*.
→ Idea: learn logic rules with policy gradient (a new architecture, DILP, trained with REINFORCE).

Experiments: For many settings, achieve high reward compared to MLP agent (baseline).

.....

3.3.4 Learning to Collaborate in MDPs [68]

The talk is by Goran Radanovic.

Motivation: Human-AI collaboration—consider a helper AI assisting a person in solving some task.

→ The agents might share a common goal, but view the task differently in some way.

Formal model: two-agent MDP, where agents have *commitments*.

→ Goal is to design a learning algorithm (for the first, non-human AI) that achieves sublinear regret.

Challenge: from the perspective of A1, the world looks like a non-stationary MDP due to the presence of A2 (the person).

Main Contribution: "Experts with Double Recency Bias", a new algorithm for this setting, based on the recency bias.

Theorem 6. *(Main Result): The regret of this algorithm decays as* $O\left(T^{\frac{1}{4}}\right)$.

.....

3.3.5 Predictor-Corrector Policy Optimization [15]

Talk by Ching-An Cheng.

Problem: Episodic policy optimization. So, agent is trying to optimize some policy $\pi(a \mid s)$, that achieves high return.

→ One goal: sample efficiency. We should spend time on planning/thinking before any real interactions.

Q: Why should we use models?

A: Can summarize past experience, can be more sample-efficient, and can optimize a policy efficiently without real world interactions.

Q: Why don't use models?

A: Well, models are always inexact! Weaknesses of model can be exploited in policy optimization.

Q: Can we reconcile these two camps?

A: Sure! This paper: PicColo (see talk title). A meta algorithm based on the idea that: "should not fully trust a model but leverage only the correct part"

→ How can this be achieved?

- Frame policy optimization as predictable online learning (POLL
- Design a reduction based algorithm for POL to reuse known algorithms.
- When translated back this gives a meta-algorithm for policy optimization.

Online learning: consider a learner and an opponent, with the learning chooses a decision $\pi_n \in \Pi$ every so often. Opponent chooses a loss function to minimize performance, rinse and repeat.

→ Common performance measure is *regret*. **Idea:** Define policy optimization as an online learning process.

→ Algorithmically: can try typical no-regret algorithms (mirror descent), but not optimal! Want to learn faster. So, view predictability as the ability to predict future gradients. Introduce the following model:

Definition 13 (Predictive model): *A function that estimates the gradient of future loss*

$$\Phi_n(\pi) \approx \nabla \ell_n(\pi)$$

Now, develop an algorithm based on this predictability model.

→ Want a reduction from predictability to online learning. This is PicColo!

Suppose we have a predictable learning problem, idea is to turn it into an adversarial one. So:

$$\ell_n(\cdot) = \hat{\ell}_n(\cdot) + \Delta_n(\cdot),$$

which is a combination of prediction loss (the first term) and error (the second loss). Dave: I didn't catch what the second source of error is
PicColO: two steps—1) Prediction step ($\pi_n = \hat{\pi}_n - \eta_n \hat{g}_n$, and 2) correction step—$\hat{\pi}_{n+1} = \eta_n(g_n - \hat{g}_n$, with g the return.

Experiments: Compare PiColo with a variety of other algorithms on MuJoCo tasks.

Summary:

- PicColO can change a model-free algorithm to be faster but without the bias.
- Predictive model can be viewed as a unified interface for injecting prior
- As PicCoLO is degiend for general predictable online learning, we expect applications to other problems and domains.

.....

3.3.6 Learning a Prior over Intent via Meta IRL [91]

Talk by Kelvin Xu.

Motivation: we often assume we have a well specified reward function!

Q: How can an agent infer rewards from one or a few demonstrations?

→ This work! Use demonstrations from previous tasks to induce tasks priors that can be used in new tasks.

Main idea: use prior task information to accelerate inverse RL.

→ Builds on MAML (see meta-learning tutorial!).

Experiment: Sprite world environment, and 2) First person navigation task. In both cases, learn task priors from meta-training task, use to learn quickly in test tasks.

→ Results: performs very well even only given a small number of demonstrations.

.....

3.3.7 DeepMDP: Learning Late Space Models for RL [30]

Talk by Carles Gelada.

Goal: Find simple representations for RL.

Approach: learn a latent space model $\bar{M} = \langle \bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{R}, \bar{T} \rangle$. Based on two losses;

$$\bar{R}(s,a) \approx R(s,a), \forall_{s,a} \tag{11}$$

$$\bar{T}(s' \mid s,a) \approx T(s' \mid s,a), \forall_{s,a,s'}. \tag{12}$$

Using these losses, ensure that ϕ is a good representation in the sense that it only throws out bad/useless policies.

Experiments 1: Donut world. Image of a circle. Embedding function ends up finding a similar representation.

Experiments 2: Atari w/ C51, find improvement over baselines.

.....

3.3.8 Importance Sampling Policy Evaluation [35]

Talk by Josiah Hanna.

Note: lots of recent empirical RL success!

But: for RL to be successful, we must ask: "How can RL agents get the most from small amounts of experience?".

Core Contribution: Study important sampling for the RL sub-problem of policy evaluation. More specifically: replace the denominator of importance sampling with an estimate, and prove that this is justified empirically and theoretically.

Typical importance sampling in RL:

$$OIS(\pi, D) = \frac{1}{m} \sum_{i=1}^{n} \Pi_{t=0}^{L} \frac{\pi(a_t \mid s_t)}{\pi_D(a_t \mid s_t} \sum_{t=0}^{L} \gamma^t R_t$$

They replace the denominator with an MLE *estimate* of the behavioral policy's performance:

$$New - IS(\pi, D) = \frac{1}{m} \sum_{i=1}^{n} \Pi_{t=0}^{L} \frac{\pi(a_t \mid s_t)}{\hat{\pi}_D(a_t \mid s_t} \sum_{t=0}^{L} \gamma^t R_t$$

Paper provides theory and experiments showing this is a good thing to do.

.....

3.3.9 Learning from a Learner [40]

Talk by Alexis Jacq.

Goal: Want to learn an optimal behaviour by watching others learning.

Assume: learning is optimizing a regularized objective:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_t \gamma^t (r(s_t, a_t) + \alpha H(\pi(\cdot \mid s_t)) \right].$$

The value of a state action couple is given by the fixed point of the regularized bellman equation. Moreover, the softmax is an improvement of the policy (see Haarnoja et al. [33]).

Experiments: on MujoCo, find improvement.

.....

3.3.10 Separating Value Functions Across Time-Scales [72]

Talk by Joshua Romoff.

Multi-step returns:

$$G_t^k = \sum_{i=0}^{k-1} \gamma^i r_{t+1} + \gamma^k V(s_{t+k}).$$

Can choose k to trade-off between bias-variance. Or, the λ returns:

$$G_t^\lambda := (1-\lambda) \sum_{k=1}^{\infty} \lambda^{k-1} G_t^k.$$

Many tasks are *discounted:* when $\gamma \to 1$, training V_γ is difficult.

Q: Why use discount factor?

A: Well, it makes the problem simpler. Only ever thinking about the next few rewards, so agents can learn to make decisions more easily.

→ But, add a lot of bias! Sometimes we don't really care about early rewards more.

Take a step back: what would the ideal RL agent do? It would learn to do well quickly!

Ideally: want to 1) learn with a discount factor that is large but, 2) built on top of methods that are computationally and sample efficient.

Solution: define a sequence of λs:

$$\Delta; = \{\gamma_0, \gamma_1, \ldots, \gamma_Z\},$$

where $\gamma_i \leq \gamma_{i+1}, \forall_i$.

Then, form a sequence of Bellman Equations using this sequence of γs:

$$W_0 : r_t + \gamma_0 W_0(s_{t+1}) \tag{13}$$

$$W_{i>0} : (\gamma_i - \gamma_{i-1}) V_{\gamma_{i-1}}(s_{t+1}) + \gamma_i W_i(s_{t+1}). \tag{14}$$

Algorithm: TD(Δ), equivalent to standard $TD(\lambda)$.

Analysis: Under some conditions, equivalent to standard TD(λ).
$\rightarrow$ So, why would you do this new, sequenced thing?
$\rightarrow$ Well, equivalent when using linear function approximation. And: 1) Same learning rates for each W, and 2) Same k-step, γ, and λ for each W.
$\rightarrow$ (But of course, don't have to set these parameters the same way).

Can extend TD(Δ) to Deep RL: 1) Share network multiple outputs, 2) Train Ws as described, 3) Use the same of Ws instead of Vs

Experiments: Tested in Atari, contrast TD(Δ) with PPO and PPO+ (PPO but with additional parameters).

$\rightarrow$ Further explore what $TD(\Delta)$ learns by plotting the value function.

Things they didn't try (but we should!):

- Adding *more* Ws.
- Q-learning extension (everything they tried was on policy).
- Model architecture for the value functions: could explore using fewer parameters, and so on.
- Distributed variant.

Summary:

1. Decompose the value function based off of γ
2. Train with Bellman-like equations
3. Improves sample efficiency

.....

3.3.11 Learning Action Representations in RL [14]

Talk by Yash Chandak.

Problem: consider RL problems with *thousands* of possible actions! (applications might be a medical treatment, advertisement, or portfolio management).

→ Might capture this by learning options [81], but usually leads to learning *too many options*.

Q: How can we learn good action representations, but not learn too many?

Key Insights:

1. Actions are not independent discrete quantities
2. There is a low dimensional structure underlying their behavior.
3. This structure can be learned independent of the reward.
4. Thus, instead of raw actions, agent can act in this new space of bejavior, behavior can be generalized to similar actions.

New Algorithm: 1) Supervised learning of action representations, and 2) Learn internal policy of these representations using policy gradient.

Experiments: Applied algorithm to a variety of domains with rich action spaces (recommendation in photoshop, for instance), find consistently good performance.

.....

3.3.12 Bayesian Counterfactual Risk Minimization [55]

Talk by Ben London.

Problem: Learning from Logged Data to recommend music.

Challenges:

1. Feedback is bandit feedback, so only learn from chosen actions
2. Data is biased based on chosen logging policy.

→ One way to counteract this bias is to use inverse propensity scoring:

$$\arg\min_{\pi} \frac{1}{n} \sum_{i=1}^{n} -r_i \frac{\pi(a_i \mid x_i)}{p_i}.$$

Can also use *variance regularization*, giving rise to the Counterfactual Risk Minimization (CRM) principle;

$$\arg\min_{\pi} \frac{1}{n} \sum_{i=1}^{n} -r_i \frac{\pi(a_i \mid x_i)}{p_i} + \lambda \sqrt{Var(\pi, S)}$$

→ Motivated by PAC analysis, ensures low generalization error.

This Work: Bayesian view of CRM (motivated by PAC-Bayes analysis). Yields a nice generalization error bound.

→ Application to mixed logits.

3.3.13 Per-Decision Option Counting [36]

Talk by Anna Haratunyan.

Motivation: Agents that reason over long temporal horizons. But: *horizon depends on choice of* γ.

Q: Perhaps we can do this with options [81]?

A: But, doesn't really handle different horizons.

Main contribution: generalize the options framework to allow for per-option discounting.

→ That is, add a per-decision option discount, (γ_r, γ_p).

In this framework, they identify a bias-variance trade-off captured by these new discounts.

3.3.14 Problem Dependent Regret Bounds in RL [94]

Talk by Andrea Zanette.

Focus: Exploration in episodic, tabular RL.

→ State of the art regret bounds:

- No intelligent exploration $\tilde{O}(T)$
- Efficient Expliration; $\tilde{O}(H\sqrt{SAT})$
- Lower Bound: $\tilde{O}(\sqrt{HSAT})$
- Below the Lower Bound: problem dependent!

Main result (requires variance of value, and scaling of reward)

Theorem 7. *With high probability, can achieve regret of:*

$$\tilde{O}(\sqrt{\mathbb{Q}^* SAT}),$$

where $\mathbb{Q}^*$ *is a problem dependent term.*

Answers an open conjecture from Jiang and Agarwal 2018 (COLT): any algorithm must suffer H regret in goal-based MDPs.

→ Also explore effect of stochasticity on regret.

3.3.15 A Theory of Regularized MDPs [29]

Talk by Matthieu Geist.

Motivation: Many deep RL algorithms make use of regularization, but no general theory of how to regularize in RL.

→ This work: generalizes regularization in RL in two ways:

1. Larger class of regularizers
2. General modified policy iteration scheme.

LEt $\Omega : \Delta_A \to \mathbb{R}$ be a strongly convex function. The convex conjugate is a smoothed maximum:

$$\forall_{q_s} \in \mathbb{R}^A, \Omega^*(q_s) = \max_{\pi_s \in \Delta_A} (\pi_s, q_s) - \Omega(\pi_s).$$

Thus, we can regularize the Bellman Equation:

$$T_{\pi,\Omega}[V] = T_\pi[V] - \Omega(V). \tag{15}$$

The regularized Bellman Operators satisfy the same properties as the original ones: 1) $T_{\pi,\Omega}$ is affine, 2) Monotonicity, distributitivity, and γ-contraction.

→ Introduce a new algorithmic scheme, regularized policy improvement, and prove the following result:

Theorem 8. *After k iterations of reg-MPI, the loss is bounded.*

→ Can also use this theory to characterize existing approaches to regularization in RL (like TRPO, DPP, and so on).

Summary:

- Bridges some gaps between dynamic programming and optimization
- Introduced temporal consistency equations, as with entropy
- Can generalized existing approaches to regularization, such as regularized policy gradient.

.....

3.3.16 Discovering Options for Exploration by Minimizing Cover Time [43]

Dave: Now, our paper! The talk is by Yuu Jinnai.

Goal: Choose options that are effective for exploration.

Contribution:

1. Introduce an objective function for exploration called *cover time* (see below).

2. Algorithm for option discovery that minimizes an upper bound on the cover time.

   → Computes the *Fiedler* vector based on the graph induced by the graph of the MDP, uses it to minimize the cover time.

Definition 14 (Cover Time): *The expected number of steps needed to visit every state.*

Theorem 9. *The upper bound on the cover time is improved;*

$$\mathbb{E}[C(G')] \leq \frac{n^2 \ln n}{\lambda_2 C(G')}$$

Performed empirical comparison contrasting learning performance with different kinds of options.

.....

3.3.17 Policy Certificates: Towards Accountable RL [20]

Talk by Christoph Dann.

Key contribution: new algorithm for episodic tabular MDPs with:

Theorem 10. *PAC Bound;*

$$\tilde{O}\left(\frac{SAH^2}{\varepsilon^2}\right),$$

and a matching regret bound.

Motivation, though: *accountability*, not necessarily sample efficiency.

Q: How good will my treatment be? Is it the best possible?
→ What kinds of methods can answer these kinds of questions?

Main Idea: Introduce *policy certificates* to add accountability.

Definition 15 (Policy Certificate): *Confidence interval around optimal and algorithm's performance*

→ Natural extension of model-based optimistic algorithms.

Challenge: Q^π is random, so it can be hard to compute confidence intervals for it. But! From optimism, we know $Q^\pi \to Q^*$ at a known rate. So, we can bound this quantity.

Two main benefits:

1. More accountable algorithms through accurate policy certificates.
2. Better exploration bonuses yield minimiax-optimal PAC and regret bounds.

.....

3.3.18 Action Robust RL [83]

Talk by Chen Tessler.

Goal: Achieve robustness in RL/MDPs. Consider cases of abrupt disruption, highly stochastic problems, and so on.

→ Thus, study robust MDPs where;

$$\pi_\alpha(\pi, \pi') = \begin{cases} \pi & w.p. 1-\alpha \\ \pi' & w.p. \alpha \end{cases}$$

Introduce an algorithm as a a player in a two player, adversarial game. Guaranteed to converge to nash.

→ Propose a Deep RL variant based on this robust MDP algorithm, experiment on MuJoCo.

.....

3.3.19 The Value Function Polytope [19]

Talk by Robert Dadashi.

Central Question: Can we characterize the geometry of the space of possible value functions for a given MDP?

Why ask this?

- Relationship between policy space and value function space
- Better understand the dynamics of existing algorithms
- New formalism of representation learning in RL.

Main Result: what is the geometry of the space of value functions of a given MDP?

Theorem 11. *The ensemble of value functions are (possibly non-convex) value polytopes.*

Building blocks: 1) the line theorem (value functions of mixtures of similar policies describe a line in value function space), and 2) boundary theorem (Dave: missed explanation).

Ongoing future work: new representation learning schemes for RL, new actor-critic algorithms.

Dave: And that's a wrap for Tuesday!

4 Wednesday June 12th: Main Conference

The day (afternoon, really) begins with contributed talks on multitask/lifelong learning.

4.1 Contributed Talks: Multitask and Lifelong Learning

Most talks will again be five minutes, with a few being 20.

4.1.1 Domain Agnostic Learning with Disentangled Representations [64]

Talk by Xingchao Peng.

Idea: Carry out supervised learning when the domain shifts.
→ Well motivated by applications (lots of cases where training data differs from true task).

> **Definition 16** (Domain Adaptation): *Train on some source domain, $P_S(X_S, Y_S)$, with lots of labeled data, then test of target $P_T(X_T, Y_T)$.*

Example: Performing image recognition on canonical, full color images, then translating to recognition on *sketches*, some of which might be black and white.

Lots of related work [75, 27, 85].

New Approach: Deep Adversarial Disentangled Autoencoder (DADA).

- Class disentanglement: disentangle to class-irrelevant and domain-invariant features
- Domain disentanglement: Disentangle to domain-specific and domain invariant features.

Example: class-invariance vs. domain invariance.
→ Domain invariance; given two images, one from a real car, one of a painting of a car → pass both into some neural net yields some features. In principle these features should be *domain-invariant* since they're found in both of these car renditions.

→ Class invariance: take a look at the backgrounds of the cars, this would yield the *class invariant* features, since they identify different classes.

Class disentanglement: train class identifier with the following loss:

$$L_{ce} = \mathbb{E}_{x,y}\left[\sum_{k=1}^{K} \mathbb{1}\{k = y\} \log C(f_D)\right].$$

Create a similar loss (and piece of the architecture) for domain disentanglement;

$$L_{vae} = ||\hat{f}_G - f_G||^2 + D_{\mathrm{KL}}\left(q(z \mid f_g) \mid\mid p(z)\right).$$

Experiments: Three benchmarks: 1) 5 digit datasets, 2) Office domains, 3) DomainNet, each of which contains a variety of domains and categories.

→ Finding: new model (DAD) improves performance over SOTA by 6% on average across these datasets.

.....

4.1.2 Composing Value Functions in RL [87]

Talk by Steve James.

> **Central Question:** Can we blence value functions together from different tasks to solve interesting combinations of the tasks without further learning?

In general, consider skills Q_1 and Q_2. Typically, we see $Q_1 \oplus Q_2 = ☹$.

This work: considers entropy regularized RL:

$$r_{ent}(r, s) = r(s, a) - rD_{\text{KL}}\left(\pi_s \mid\mid \bar{\pi}_s\right).$$

With entropy regularized RL we show that: $Q_1 \oplus Q_2 = \smile$

Idea: OR task composition: can optimally compose $Q(\square)$ and $Q(\circ)$ to solve the task of collecting $\square$ OR $\circ$.

Corollary: in the limit, prov that $Q(\circ OR \square) = \max\{Q(\circ), Q(\square)\}$.

Experiment: Agents should pick up some object, train in tasks where they have to pick up just one type, then test in case where either is OK.

Summary:

1. Can do zero shot composition to provably find solution to OR tasks.
2. Works well in experiments!

.....

4.1.3 CAVIA: Fast Context Adaptation via Meta Learning [95]

Talk by Luisa Zintgraf.

Idea: Meta-learning for fast adaptation—learn how to map x to y on new tasks, fast and with little data.

→ Earl approach: MAML (see meta-learning tutorial).

New Algorithm: CAVIA: Fast Context Adaptation via Meta Learning— 1) less prone to over-fitting and 2) interpretable.

→ Many tasks and currrent benchmarks only require task identification. many parameters and few data points can lead to *overfitting*.

Experiment 1: sine curve experiments. Task is defined by learning amplitude and phase. MAML require 1500 parameters, whereas CAVIA requires only 2 context parameters.

→ Context parameters are interpretable, and can be reused across tasks.

Experiment 2: mini-imagenet experimnents. As the network becomes more complex, MAML requires 30000 parameters, hwereas cAVIA only requires a few hundred.

.....

4.1.4 Gradient Based Meta-Learning [45]

Talk by Mikhail Khodak.

Q1: What kinds of task-relationships can meta-learning algorithms exploit?

Q2: Are we restrictinv outsrleves by using such simple methods ?

Q3: How does meta-learning relate to classical multi-task methods?

This Work: Address these questions in the convex case. Answers:

1. Better avg. performance per-task if optimal task-parameters are close together
2. GBML is the best we can do without stronger task-similarity assumptions
3. NAtural connection between GBML to regularized multi-task learning

Main strategy: connection to online convex optimization.
→ Consider what a standard gradient based method does. Pick first initialization $\phi_1 \in \Phi$, then for task $t = 1 \dots T$;

1. Run m steps of SGD
2. Dave: Missed it.

Can then import guarantees from online convex optimization, specifically regret guarantees that encode distance from initialization.

Main result:

Theorem 12. *GRBML achieve average regret of:*

$$O\left(D + \frac{\log T}{T}\sqrt{m}\right),$$

where T is the number of tasks, D is the radius, Dave: didn't catch m, presumably num samples.

.....

4.1.5 Towards Understanding Knowledge Distillation [65]

Talk by Mary Phuong.

Idea: Knowledge distillation. A teacher (trained neural net), comes up with/represents a function $f_\theta : \mathcal{X} \to \mathcal{Y}$. Then, would like to compress this function into a smaller/simpler network (a student!).

Q: How effective is distillation? Make this concrete by studying a quantity called the *transfer risk*, measuring how bad the distillation process can be.

Setting: linear distillation: 1) linear teacher, 2) student is a deep neural net.

Results:

1. Can compute exactly what the student will learn! Teacher is $f_t(x) = w^\top x$, then student will be either exactly the teacher (depending on amount of data).
2. Can also bound the transfer risk:

$$n_data \geq d \implies risk = 0 \tag{16}$$

$$n_data < d \implies risk \leq \left(\frac{\log n_{data}}{n - data}\right)^\kappa, \tag{17}$$

$$\tag{18}$$

   with $\kappa \in [0, infty)$ the easiness of the distr.

.....

4.1.6 Transferable Adversarial Training [53]

Talk by Hong Liu.

Typical Assumption: training data and test data are drawn from same distribution.

→ What if we change this?

∴ This work: how do we generalize a learner across different distributions P and Q.

Goal: bound the target error (on distr Q) with error from train error (on distr P).

→ existing theory based on domain adaptation.

Prior work (adversarial feature adaptation):

1. Minimize source risk
2. Minimize discrepancy term: learn a few feature representation where discrepancy is minimized

3. Define as a two player game: domain discriminator tries to discriminate the source and target domain, while feature extractor tries to confuse it.

But, a prerequisite for doing domain adaptation: adaptability quantified by λ. If λ is large, we can never expect to adapt a learner trained on source domain.

Q: How can we amend this?

A: One way is to fix feature representations to prevent adaptability from getting worse.

But: now we need to figure out how to adapt to the target domain.

New Model: Transferable ADversarial Training (TAT).

- Main Idea: instead of feature adaptation associated source and target domain with transferable examples
- Generate transferable examples to bridge the gap across the two domains.
  $\rightarrow$ Concretely: train a classifier and a domain discriminator.
- Overall optimization problem:
  1. four loss terms, w/ fixed feature representations.
  2. No need of feature adaption (light weight computation)
  3. Order of magnitude faster than adversarial feature adaptation.

Analysis: consider the rotating two moon problem. Target domain is rotated 30° from source domain.

Experiments: Domain adaptation benchmarks (Office031, Image-CLEF, Office-home, ViSDA).

$\rightarrow$ Finding: achieve comparable performance to SOTA.

Code: https://github.com/thuml/Transferable-Adversarial-Training

.....

4.2 Contributed Talks: RL Theory

Now, more RL theory.

4.2.1 Provably Efficient Imitation Learning from Observation Alone [80]

Talk by Wen Sun.

Prior works can achieve sample complexity of:

$$\text{poly}(Horizon, |A|, |S|, 1/(1-\gamma)).$$

This setting: Imitation learning from observations

- Given: trajectories of observations
- Learning from observations
- No interactive expert, no expert action, no reset, and no cost signals

NE Algorithm: Forward Adversarial Imitation Learning (FAIL), model-free approcah.

$\rightarrow$ Idea: learn a sequence of policies, $\pi_1, \ldots, \pi_{T-1}$. To learn each policy, treat it as a two player game.

$\rightarrow$ To solve the game, treat it as a min-max game, sove it as a minimzing integral probability metric (IPM):

$$\max_{f \in \mathcal{F}} \mathbb{E}_{x \sim P}\left[f(x)\right] - \mathbb{E}_{x \sim Q}\left[f(x)\right],$$

with $\mathcal{F}$ a set of discriminators. π_0 together with the dynamic model define one agent (the generator), and $\pi_1^\star$ as the expert distribution. Then, want:

$$\max_{\pi_0 \in \Pi} \max_{f \in \mathcal{F}} f(\pi_1^\star - f(\pi_0).$$

Solving the above yields the next policy, π_1, and rinse and repeat this process.

Q: What is the sample complexity of this approach?

A: Well, if discriminator is very strong, we'll overfit. But if it's too weak, it will be unable to distinguish well.

$\rightarrow$ So, introduce the "inherent bellman error"

Definition 17 (Inherent Bellman Error): *The inherent bellman error of a function class $\mathcal{F}$ is given by:*

$$\Gamma^* f(x) \triangleq \mathbb{E}_{a \sim \pi^*, x \sim P(\cdot|s,a)}\left[f(x')\right]$$

Result of FAIL:

Theorem 13. *Under the realiazability assumption, so $\pi^* \in \Pi$ and $V^* \in \mathcal{F}$, then to learn a near optimal policy we need:*

$$\textit{poly}\left(T, A, 1/\varepsilon, SC(\Pi), SC(\mathcal{F})\right),$$

with SC the statistical complexity of the function family.

But, the above forces us to suffer this inherent bellman error.

Q: Is this inherent bellman error avoidable in the imitation learning from observation setting?

A: Yes! But in model-based RL.

- Start with a realizable model class, so $P \in \mathcal{P}$.
- Then there exists an algorithm that takes $\mathcal{P}, F$ as input, outputs an ε optimal policy in the above sample bound.

Experiments: Implement this imitation learning algorithm on simulated results (fetch reach and reacher), perform quite well (with limited parameter tuning).

Takeaways:

- With observations alone from experts, we can learn optimal policies
- Near-optimal guarantees
- Supervised learning type sample complexity
- Out-of-box performance is pretty good.

.....

4.2.2 Dead Ends and Secure Exploration [25]

Talk by Mehdi Fatemi.

Q: What is a dead end?

→ A terminal state is *is undesired* if it prevents achieving maximum return.

Definition 18 (Dead End State): *A state s_d is dead end if all trajectories starting from s_d reach an undesired terminal state with probability 1 in some finite (possibly random) number of steps.*

Q: Why should we care about dead ends?

A: Most algorithms only converge under the assumption of infinite (s, a) visitation $\forall_{s,a}$.

→ New idea:

Definition 19 (Policy Security): *A policy is secure if for any $\lambda \in [0, 1]$:*

$$\sum_{s'} T(s, a, s') \geq 1 - \lambda \implies \eta(s, a) \leq \lambda.$$

Solution: Make a new MDP called the "exploration" MDP similar to the original MDP but with some subtle changes. Result:

Theorem 14. *Let η be any policy s.t. $\eta(s,a) \leq 1 + Q^*(s,a)$, where $Q^*(s,a) \neq -1$ for at least one action.*

Then, η is secure.

(Also run experiments).

.....

4.2.3 Statistics and Samples in Distributional RL [74]

Talk by Mark Rowland.

Recall: Distributional RL aims to learn full return distributions:

$$Z^\pi(s,a) = \sum_{i=0}^{\infty} \gamma^t R_t.$$

Main Contribution: New approach—learn functions of the return distributions directltly like moments, tail probabilities, expectations, and so on.

New framework! Allows progress in new areas:

- *Theory:* fundamentally, what properties can we learn about return distributions from dynamic programming
- *Algorithms:* framework for approximate learning of statistics of the return distributions.

Application: "expectiles". New deep RL agent "Expectile-Regression DQN" (ER-DQN) with improved mean performance relative to QR-DQN (quantile regression DQN).

.....

4.2.4 Hessian Aided Policy Gradient [78]

Talk by Zebang Shen.

Idea: Formulate policy optimization as optimization of a trajectory.

→ Old approach: use REINFORCE/SGD to slowly improve policies via polict gradients.

New approach: "oblivious" policy optimization. Can improve sample efficiency of existing policy gradients in this "obvivious case".

But: in the "oblivious" case, the policy gradient algorithm becomes biased. So, correct

Summary: First provable method that reduces sample complexity to achieve an ε optimal from $O(1/\varepsilon^4)$ to $O(1/\varepsilon^3)$ via policy optimization.

.....

4.2.5 Maximum Entropy Exploration [37]

Talk by Elad Hazan.

Setting: Agent is in an MDP without a reward signal. How can it explore?

→ Existing approaches; task-agnostic exploration, curiosity, and exploration bonuses.

Main Q: Can we solve exploration without rewards efficiently?

The setting:

- Every π induces a state distribution
- Given a policy class Π and a concave functional H, action on the state distr., can we find:

$$\max_{\pi \in \Pi} H(d_\pi),$$

  with d_π the stationary distr. on states induced by π.

Proposition 1. *$H(d_\pi)$ is not concave in π.*

New Algorithm: Max Entropy algorithm. Take the concept of a uniform mixture of policies $C = (\pi_1, \ldots, \pi_k)$.

1. Density estimator
2.

Theorem 15. *In finite (small) iterations, algorithm guarantees $H(d_{mix}) \leq \max_{\pi \in \Pi} H(d_\pi) - \varepsilon$ (effectively solves this problem).*

.....

4.2.6 Combining Multiple Models for Off-Policy Evaluation [32]

Talk by Omer Gottesman.

Definition 20 (Off-policy Evaluation): *Assume we have a batch of data collected by some policy π we don't control, and use this data to evaluate some other policy $\widehat{\pi}$.*

Two general approaches:

1. Model-based: Use the data to learn an environmental model, then use this to evaluate the new policy.
2. Importance-sampling: reweight returns/state visitations based on policy differences.
   → Problem: huge variance! (even if other nice statistical properties).

Q: If we had multiple models with different strengths, could we combine them to get better estimates?

A: Yes, this underlies main idea of the work!

Intuitive example: two regions of an MDP— 1) top, well modeled, and 2) bottom, poorly modeled. Suppose we have models that move into just the top/bottom areas: we know the trajectories in the top will be better! So, stick with the model that does will in this region

→ Idea: might want to trade-off short term vs. long term accuracy. Bound on model-errors:

$$|g_T - \hat{g}_T| \leq L - t \sum_{t=0}^{T} \gamma^t \sum_{t'=0}^{T} L_t^{t'} \varepsilon_t (t - t' - 1) + \sum_{t=0}^{T} \gamma_r^{\varepsilon}(t),$$

where ε_r and ε_t are error bounds, g_T is return (and $\hat{g}_T$ is estimate), and L_t is a Lipschitz constant Dave: (on transitions, I think?)

Idea: use MCTS to minimize return error bound over entire trajectories.

Two kinds of models:

1. Nonparametric models: predicting the dynamics for a given state-action pair based on similarity to neighbors.

   Strength: Can be very accurate in regions of state space where data is abundant

2. Parametric models: any parametric regression model or hand coded model incorporating domain knowledge.

   Strength: Tend to generalize better to situations different from ones observed in data.

→ Nice thing about these two kinds of models: they have different strengths! So, let's combine these two.

Experiments: Medical simulators (simulating growth of cancer cells and HIV). Compare, for different behavioral policies, off-policy evaluation.

Summary:

- provide a general framework for combining multiple models to improve off-policy evaluation
- Improvements via individual models, error estimation or combining multiple models.

.....

4.2.7 Sample-Optimal Parametric Q-Learning Using Linear Features [92]

Talk by Lin F. Yang.

Consider: curse of dimenstionality! Optimal sample complexity is:

$$\tilde{\Theta}\left((1-\gamma)^{-3}|\mathcal{S}||\mathcal{A}|\right)$$

→ Too many states and actions. So, how can we optimally reduce dimensionality of the game?

A: Exploit structure!

Approach: Feature-based MDP. decompose transition model:

$$P(s' \mid s, a) = \sum_{k \in [K]} \phi_k(s,a)^T \psi_k(s'),$$

which decomposes MDP into some number of factors, where ϕ is *known* and ψ is unknown.

Toy Example: Stock prices! In some financial models we can decompose a stock price into a linear combination of a set of representative stocks. If this is the case, we can solve the RL problem parametrically using a Q that models this linear relationship.

→ Idea: Generative Model we are able to sample from (any s, a). Represent Q function with parameters $w \in \mathbb{R}^k$, so:

$$Q_w = r(s,a)\gamma\phi(s,a)^t w$$

Then, learning with modified Q-Learning can achieve sample complexity of:

$$\tilde{O}\left(\frac{K}{\varepsilon(1-\gamma)^7}\right).$$

Q: Is this optimal?

A: No, can show that under the "anchor condition", this collapses to:

$$\tilde{\Theta}\left(\frac{K}{\varepsilon^2(1-\gamma)^3}\right).$$

.....

4.2.8 Transfer of Samples in Policy Search [84]

Talk by Andre Tirinzoni.

Policy Search: Effective RL technique for continuous control tasks.

→ High sample complexity remains a major limitation. Samples available from several sources are discarded.

Formally: given some source tasks (MDPs with different models), reuse data collected from these problems in a *new*, related task.

Contribution 1: A new importance sampler technique that allows for a more effective gradient estimator.

→ Nice properties of the estimator

1. Unbiased and bounded weights.
2. Easily combined w/ other variance reduction techniques.
3. Effective sample size ≡ transferable knowledge (adaptive batch size.
4. Provably robust to negative transfer.

Problem! P unknown in general so importance weights can't be computed.

→ Solution: online minimization of upper bound.

Empirical results: good performance with both known and unknown models (cartpole, minigolg). Very effective sample reuse from different policies but same environment.

.....

4.2.9 Exploration Conscious RL Revisited

Q: Why exploration conscious?

A: To learn a good policy an RL agent must explore!

→ New Objective: find the optimal policy knowing exploration will occur under a particular *strategy*.

Main Approach: consider a fixed exploration scheme (ε greedy, etc.). Then: 1) choose greedy action, 2) draw exploration action, 3) act, 4) receive r, s'.

→ Use information about the exploration conscious problem.

Two Methods:

1. *Expected:* Update Q of action you actually took, but expect the agent might explore in the next state. Thus, bootstrap using this method (but, calculating this expectation can be hard).
2. *Surrogate:* (Main Contribution): add exploration directly into the environment.

Conclusion: Exploration conscious RL and specifically, surrogate approach, can easily help improve a variety of RL algorithms.

.....

4.2.10 Kernel Based RL in Robust MDPs [51]

Talk by Shiau Hong Lim.

Definition 21 (Robust MDP): *Extends MDP by considering model mismatch and parameter uncertainty.*

For ustate aggregation, performance bound on $||V_R^\pi - V^*||$ improved via robust policies:

$$O(\frac{1}{(1-\gamma)^2} \rightarrow O(\frac{1}{(1-\gamma)}$$

Contributions;

1. Robust performance bound improvement by extending to kernal averager setting.
2. Formulation of a practical kernel-based robust algorithm, with empirical results on benchmark tasks.

Theorem 16. *Extend value loss bounds of state aggregation to count-based setting.*

$\rightarrow$ Practical algorithm: use kernel averager to approximate MDP model, then solve approximate model with approximate robust Bellman operator.

Conclusion: 1) Performance guarantees for robus kernel-based RL, and 2) Show significant empirical benefits from this method.
Dave: And that's a wrap for Wednesday!

5 Thursday June 13th: Main Conference

The final day of the regular conference!

5.1 Contributed Talks: RL

We begin with the final RL session.

5.1.1 Batch Policy learning under Constraints [49]

Talk by Hoang M. Le.

Motivation:

1. *Change RL Objective:* Consider usual RL objective:

$$\min_x C(\pi) = \mathbb{E}\left[\sum cost(s,a)\right]$$

   But, in reality, not always feasible. → Hard to define a single cost function!

   ∴ Specifying additional constraints seems naturak.

2. *Sample Efficient Batch RL:* But, if we change the cost objective, we need to solve a new problem that can make RL even harder.

   → In batch RL, tend to think about having some existing data set (from some prior policy, π_D, that may be sub-optimal).

   Q1: Can we use this abundant source of data to learn a better, new policy?

   Q2: What if we want to change our constraints? Can we still learn a new policy under these new constraints?

Notation:

- Data set of n tuples of data, $D = \{(s,a,r,s'), \ldots,$ generated by π_D.
- Goal is to find some π such that $\min_\pi C(\pi)$, subject to some constraint $G(\pi) \leq 0$.

Examples: 1) Counterfactual and safe policy learning (constraint specifies to avoid certain states), or 2) Multi-criteria value-based constraints (driving, minimize travel time subject to constraint that we stay in a particular lane).

Problem: Lagrangian:

$$L(\pi, \lambda) = C(\pi) + \lambda^\top G(\pi),$$

where the primal is:

$$\min_\pi \max_{\lambda \geq 0} L(\pi, \lambda)$$

and the dual is:

$$\max_{\lambda \geq 0} \min_\pi L(\pi, \lambda).$$

Thus: extend policy class to allow randomized policies to handle non-convex costs.
Approach: Reductions to supervised learning and online learning.

Algorithm Sketch: Repeat the following:

1. $\pi \leftarrow$ Best-response(λ), where this usually comes from Fitted Q-Iteration [23].
2. L_{max} = evaluate (dual) fixing π
3. L_{min} = evaluate (primal) fixing λ
4. If $L_{max} - L_{min} \leq \omega$, STOP.
5. Otherwise, $\lambda \leftarrow$ Online-algorithm(all prior π).

To handle the evaluation steps, rely on off-policy evaluation. That is: given D, want to estimate $\hat{C}(\pi) \approx C(\pi)$.

$\rightarrow$ Propose a new model-free direct method for off-policy evaluation called Fitted Q Evaluation (FQE). Sketch:

1. Choose a function class. Then, for k iterations:
2. Solve for Q ; $(s, a) \mapsto y = c = Q_{prev}(s', \pi(s'))$
3. $Q_{prev} \leftarrow Q$.

Theorem 17. *End-to-end guarantee for FQE: for* $n = poly(1/\varepsilon, \log 1/\delta, \log K, \ldots)$, *with* $\Pr 1 - \delta$*:*

$$|C(\pi) - \hat{C}(\pi)| \leq O(\omega + \sqrt{\beta}\varepsilon),$$

with ω *a chosen stopping condition for FQE.*

Experiments: A top-down driving task with the goal of minimizing travel time, while satisfying constraints like smooth driving and staying in the center of a lane. Train on a π_D that *ignores* these constraints.

$\rightarrow$ Finding: output policy from the algorithm can satisfy the new constraints, while keeping performance high (travel time).

.....

5.1.2 Quantifying Generalization in RL [17]

Talk by Karl Cobbe.

Note: Lots of deep RL algorithms use same tasks for training and testing.

$\rightarrow$ So, this work introduces new environments designed to study generalization explicitly in deep RL. Better benchmarks $\implies$ Better research, better algorithms.

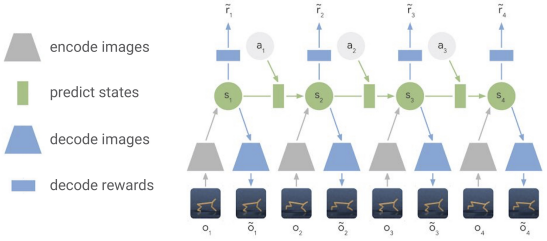


Figure 7: PLaNET latent dynamics model.

Maing Finding: Lots of deep RL algorithms do in fact overfit on this environment (which is explicitly supposed force agents to generalixe well).

Coin run domain:

1. 2D platformer, agent has to collect coins
2. Level generation conditioned on difficulty
3. Can construct training sets of arbitrary size.

Benchmark Results: Run a variety of architectures on tasks from the coin run domain (with a split train/test set).

→ Finding 1: Overfitting is in fact a significant problem
→ Finding 2: deeper architectures tend to generalize better (3 vs. 15 conv layers).
→ Finding 3: Explore different effects of regularization schemes like L_2, dropbout, batch norm, and data augmentation. They are all similarly effective at regularization.

Code: https://github.com/openai/coinrun

.....

5.1.3 Learning Latent Dynamics for Planning from Pixels [34]

Talk by Danijar Hafner.

Problem: Planning has been really effective when the world model is known (as in AlphaGo). But, it has proven ineffective with complex or unknown transitions.

→ Consider visual control tasks: MuJoCo but with images as input.

This Work: PLaNet: 1) Recipe for scalade model-based RL, 2) Efficient planning in latent space with large batch size, 3) Significantly more sample efficient than model-free methods.

Experiments: Compare to model-free agents on visual variant of MuJoCo tasks, achieves competitive performance (to model-free methods) but with far fewer samples.

Thus: a scalable method for model-based RL in image space.

Code: https://danijar.com/planet

.....

5.1.4 Projections for Approximate Policy Iteration [3]

Talk by Riad Akrour.

Consider: Entropy regularization in RL is widespread with actor-critic methods (see: SAC, NAC, PPO, A3C, REPS, ACER, ACKTR, TRPO).

Soft constraint:

$$max\pi_J(\pi) + \alpha H(\pi).$$

where J is the objective (return) and H is entropy. Conversely, the hard constraint:

$$\max_{\pi} J(\pi) \quad \text{s.t.} \quad H(\pi) \geq \beta$$

→ Harder to optimize (the hard constraint), easier to tune.
Contributions: 1) Project hard constraint of Shannon entropy of Gaussian or soft max policies, and 2) Perform projections of hard constraint, yields an optimization scheme for deep RL.

Experiment: Compare different actor-critic/policy-optimization algorithms with entropy bonus vs. their projection-based method.
→ Entropy bonus is ineffective early on, but with a strict entropy constraint, yields better exploration and better policies. Simple scheme, can be incorporated into any RL algorithm.

.....

5.1.5 Learning Structured Decision Problems with Unawareness [39]

Talk by Craig Innes.

Focus: "Unawareness" in decision making.

→ Example: Farming. What might an agent learn? Dynamics between protein levels in grain, when and which grains to fertilize, and so on.

Q: But what if, part way through the problem, we discover new state variables and actions that are critical to performing optimally?

A: If we don't explicitly model this kind of unawareness, agents might not be able to solve many kinds of problems.

Contributions: Agent learns an interpretable model of a decision problem incrementally via evidence from domain trials and expert advice.

→ Approach: if agent's performance in last k trials is below threshold β of true policy π_+, say: "At time t you should have done a' rather than a_t." Agent can then learn lots from this feedback: 1) new action, 2) a' has more value than a_t, and so on.

Experiments: A suite of randomly generated decision problems.

→ Finding 1: New agent is able to learn the true optimal policy, despite starting unaware of state variables and actions. → Finding 2: Varying experts tolerance of when to intervene has a profound impact on agent's learning capacity.

.....

5.1.6 Calibrated Model-Based Deep RL [56]

Talk by Ali Malik.

Consider: Uncertainty is rampant in the real world. To make effective RL systems for the real world, we have to tackle this uncertainty head on.

→ Thus, important to model uncertainty accurately.

Q: What type of uncertainty is important to model in RL?

A: Turn to statistics! Consider meteorologists, for instance: adopt "proper scoring rules", which are losses that hold when the predicted distribution matches the true distribution rules.

Definition 22 (Proper Scoring Rules): *A measurement for how well a model captures uncertainty based on two criteria:*

1. **Sharpness:** *Predictive distributions should be focused (have low variance)*
2. **Calibration:** *Uncertainty should be empirically accurate (about p% of the time I should be right, if I predict something as having a propensity of p).*

CLaim: Calibration in particular is really important for model-based RL.

Q: Why?

A1: For planning! Calibrated uncertainties lead o better estimates of expectation.

Theorem 18. *The value of π for an MDP under true dynamics is equal to the value of the policy under a calibrated dynamics model $\widehat{T}$.*

A2: For exploration! Many exploration/exploitation algorithms use upper confidence bounds (UCB) to guide choices such as LinUCB:

$$\arg\max_{a \in \mathcal{A}} \left(x\theta_a = \alpha \sqrt{x \hat{\Sigma}_a x} \right).$$

Calibration naturally improves UCBs, resulting in better exploration.

Q: How do we make sure our systems have calibrated uncertainty? (note that deep networks' uncertainty is often uncalibrated)

A: Turn to method from prior work: *recalibration* from Kuleshov et al. [47].

New Algorithm:

- Explore: collect observations using current model $\widehat{T}$.
- Learn Model: retrain transition model using new observations.
- Learn Calibration: learn recalibrator R on held-out subset of observations.
- Recalibrate: recalibrate the transition model using the recalibrator.

Experiment 1: Contextual bandits to investigate effect of calibration on exploration.

→ Finding: calibration significantly improves exploration in contextual bandits.

Experiment 2: Explore effect of calibration in MuJoCo continuous control tasks. Add calibration to algorithm from Chua et al. [16] (SOTA on MuJoCo).

→ Finding: Adding calibration can improve sample complexity of continuous control tasks.

Experiment 3: Inventory planning—calibrate a Bayesian DenseNet.

.....

5.1.7 RL in Configurable Continuous Environments [59]

Talk by Alberto Maria Metelli.

Recall: Traditional RL assumes the environment is some *fixed* MDP.

→ But, there are real world scenarios where we can exercise partial control on aspects of the environment.

Example: Consider car racing. Offline, we can modify aspects of the car to make racing more effective.

Definition 23 (Configurable MDP): *An agent seeks policy parameters θ with the environment configuration ω that optimizes performance, where ω effects aspects of the transition model.*

Assumptions (with prior state of the art): finite state-action spaces, full knowledge of the environment dynamics.

Algorithm: Relative Entropy Model Policy Search (REMPS): two phases—1) Optimization (find new stationary distr d' in a trust region), and 2) Projection (find a policy π'_θ and configuration distr. p'_ω that induces the right stationary distr).

Experiments: 1) Chain domain, 2) Cartpole.

Code: https://github.comn/albertometelli/remps.

.....

5.1.8 Target-Based Temporal-Difference Learning [50]

Talk by Niao He.

Recall: Using a target network is pervasive in using DQN-like algorithms. Idea is to use a separate network to bootstrap in the Q update instead of bootstrapping.

→ But! We don't know much about why a target network is effective.

Q: Can we understand the convergence of TD-learning when target variables are used?
A: Yes! Three variants studied: 1) averaged (A-TD), 2) double (D-TD), 3) periodic (P-TD), TD-learning.

Classical TD-Learning:

$$\theta_{t+1} \leftarrow \theta_t - \alpha_t \nabla_\theta L(\theta; \theta'_t; e), \tag{19}$$

followed by a target variable update: $\theta'_{t+1} = \theta_{t+1}$ every so often.

Main results:

Theorem 19. *1) Target extended TD learning (A-TD and D-TD) converges asymptotically to the correct solution, and 2) Sample analysis of convergence rate of these approaches.*

Experiments: Contrast performance of A-TD, D-TD, and P-TD in simulations.

.....

5.1.9 Linearized Control: Stable Algorithms and Complexity Guarantees [73]

Talk by Vincent Roulet.

Problem: non-linear control problem where the system is described by some non-linear dynamics.

Q1: Does ILQR (classical algorithm used to solve these) converge? Can it be accelerated?
Q2: How do we characterize complexities for nonlinear control?

Contributions:

1. ILQR is Gauss-Newton: Regularized ILQR gets convergence to a stationary point
2. Potential acceleration by extrapolation steps: accelerated ILQR akin to Catalyst acceleration.
3. Oracle complexity analysis

Code: https://github.com/vroulet/ilqc.
Dave: Now, our other options paper.

.....

Finding Options that Minimize Planning Time [42]
Talk by Yuu Jinnai.

Contribution: The problem of finding an optimal set of options that minimize planning time is NP-Hard.

Q: Why options?
A: Well, options can help in RL and planning.

Q: Okay, but how do we get *good* options? More generally: what constitues a good option?

A: planning!

Contribution:

1. Formally define the problem of finding an optimal set of options for planning through the Value Iteration algorithm.
2. Show this problem is NP-Hard.
3. Also prove that this problem is hard to approximate: *hard to approximate*. Lower bound on approximation ratio is: $2^{\log^{1-\varepsilon}(n)}$.
4. Identified approximation algorithm for computing optimal options under certain conditions.
5. Experimental evaluation of these options.

Main Takeaway: option discovery is useful for planning if and only if we have structures, priors, or assumptions.

.....

5.2 Contributed Talks: Deep Learning Theory

Next up, Deep learning theory.

5.2.1 Why do Larger Models Generalize Better? [12]

Talk by Alon Brutzkus.

Goal: Understand the gneralization myster of large networks.

→ Empirical observation: large and small models reach zero training error, but large models *generalize better.*

Two Major Challenges:

1. Proving a generalization gap (need something beyond sample complexity)
2. Analyzing convergence of gradient descen on neural networks.

Approach: Analyzed a simplified learning task which 1) empirically exhibits same, 2) shares features with problems in practice.

Main Results:

- Study a high dimensional variant of the XOR problem (XOR "detection" problem0
- Prove that overparameerization improves generalization under certain assumptions

Learning problem: two dimensional filters of ground0truth convnet.
→ Q: What happens when you try to learn this function with small vs. large number of channels?

Empirical finding: large network at convergence finds exactly the ground truth function, whereas the small network (which can in principle find the ground truth function) instead finds a function that has 0 training error, but not 0 test error.

→ This work is about explaining the above.

Definition 24 (XOR Detection Problem): *Four 2d binary patterns, with input* $x = (x_1, \ldots, x_d) \in \{-1, 1\}^{2d}$.
XOR detector:

$$f^*(x) = \begin{cases} 1 & \exists x_i s.t. x_i = p_1 \ OR \ x_i = p_3 \\ -1 & otherwise \end{cases}$$

Consider a network: three layer CNN (Conv→MaxPool→Fully connected), with 2 dimensional filters:

$$\sum_{i=1}^{n} \left[\max_j \left\{ \sigma(w^i x_j) \right\} - \max_j \left\{ \sigma(u^i x_j) \right\} \right].$$

Use: slightly modified hinge loss, SGD.

Definition 25 (Diverse Training Set): *A training set $\mathcal{S}$ is said to be diverse if it consists of both positive and negative patterns, where each positive/negative point contains all patterns (for positive, p_1, p_2, p_3, p_4, for negative, p_2, p_4).*

Training;

- Assume a diverse training set $\mathcal{S}$.
- For $k = 2$ (num channels in the network), there are multiple training error solutions (global minima). $\rightarrow$ Thus, it suffices to detect at least one of these.

Questions:

1. For $k = 2$ does SGD converge to a global minima?
2. If so, which one?
3. What happens when $k > 2$?

Theory says: for a small network, SGD converges to a global minimum which does not recover f^*, whereas a large network can (with high probability). $\rightarrow$ (also translate the above to a PAC guarantee).

Experiments: XOR Detection. Study the effect of network size, convergence rate, and exploration effect.

$\rightarrow$ Finding: large network, for different training sizes, outperforms small network.

.....

5.2.2 On the Spectral Bias of Neural Nets [69]

Talk by Nasim Rahaman.

Q: Why do massive neural nets generalize when they can learn random labels? (See work by Arpit et al. [6]).

A (this work): Neural networks learn simpler functions first (gradually build up more complex functions over the course of training).

Q: But how do quantify simplicity?

A: Use the Fourier spectrum! Nets learn functions at lower frequencies first.

Q: Why should I care?

A: Well, consider label noise. We usually think of i.i.d. noise (white noise). But, white noise is a special case of noise with a flat spectrum (form is constant in expectation).

→ For instance, high frequency noise leads to a big bump in validation score, while low frequency noise does not receive this same benefit.

.....

5.2.3 Recursive Sketches for Modular Deep Learning [31]

Talk by Joshua R. Wang

Problem: Object recognition. Train a neural net that learns to identify objects in an image.

→ Idea: twist on this task: output a succinct representation/embedding of the objects in the picutre.

Takeaway: Can use this model to solve the original object recognition problem.

> **Definition 26** (Modular Network): *A modular network consists of modules, which are independent neural net components that can communicate via output to other modules' inputs.*

→ Use a modular network to output sketches contained each object detected by the network.

Provable Properties: 1) attribute recovery, 2) sketch to sketch similarity, 3) summary statistics, 4) graceful erasure.

.....

5.2.4 Zero-Shot Knowledge Distillation in Deep Networks [60]

Talk by Gaurav Kumar Nayak.

Central Q: Can we do zero-shot knowledge distillation without training data?

A: Yes! (this work).

Idea: pseudo data synthesis. Construct class impression (CI) by creating pseudo images via the technique introduced by Reddy Mopuri et al. [71].

→ But: CI are limited because 1) generated samples aren't diverse, 2) student doesn't generalize well.

This Work: Extend class impressions with "data impressions" that overcome these issues (not class specific).

→ Experiment with MNIST and CIFAR-10, find data impressions enhance learning effectiveness.

Summary: Zero-Shot KD approach based on *data impressions*.

.....

5.2.5 Convergence Theory for Deep Learning via Over-Parameterziation [4]

Consider: training a deep net with L layers, given n training points.

→ Suppose the network is overparameterized (so the number of params is a polynomial of the training data).

Main result 1:

Theorem 20. *If data is not degenerate, an overparameterized net with SGD will find a global minima in:*

$$T = \frac{poly(n, L)}{\delta} \log \frac{1}{\varepsilon}.$$

Key message: for sufficiently large neighborhood of random intitializations, the neighborhood is *almost-convex*. If the objective is smooth, then the objective value can be decreased via the right kind of gradient step.

Main result 2:

Theorem 21. *If number of neurons is sufficiently large $m \geq poly(n, L)$, for a sufficiently large neighborhood of initial;y, neural networks behave like a Neural Tangent Kernel (NTK). That is:*

$$\textit{Over-parameterized deep networks} = NTK$$

and therefore:

$$\textit{networks essentially convex and smooth} \implies \textit{training is easy.}$$

.....

Dave: Heading out for lunch

.....

5.3 Best Paper Award: Rates of Convergence for Sparse Gaussian Process Regression

Talk by David Burt.

This Paper: Comp. Complexity of getting good approximation of

Background: Sparse Gaussian Process (GP) Regression.

- Advantages: can explicitly represent uncertainty as a function of size of data set.
- Disadvantages: High cost in computational complexity: kernel method, requires storage of kernel matrix $O(N^2)$ space, and need to invert this matrix, so requires $O(N^3)$ time.

Pseudo-Observation Approximations: approximate based on $M \ll N$ inducting points.
$\rightarrow$ Time of $O(NM^2 + M^3)$, with memory $O(NM)$ (which is much less!).

This work: *variational* sparse Gaussian inference. Advantages:

- We can simultaneously learn hyperparameters by maximizing ELBO.
- Approximation of posterior remains uncertain in regions without much data

Central Question: How many inducing points (M) do we really need for this to work?

Starting point for bounds: for a fixed data set:

$$D_{\mathrm{KL}}\left(Q \mid\mid \widehat{P}\right) \leq \text{Upper Bound} - \text{ELBO}.$$

Upper and lower bounds collapse on the KL as data set size increases. In particular, *a posteriori* bound leads to:

$$D_{\mathrm{KL}}\left(Q \mid\mid \widehat{P}\right) \leq \frac{\text{quality of lowering matrix approx}}{2\sigma_n^2}\left(1 + \frac{||y||^2}{\sigma_n^2}\right).$$

To prove *a priori* bounds, need: 1) Efficient procedure for choosing inducing points, and 2) Assumptions about distribution of training data.

$\rightarrow$ Approximation kernel matrices:

$$K_{ff} \approx K_{uf}^{\top} K_{uu}^{-1} K_{uf}$$

Can bound the deviation of the above approximation, but requires: 1) knowledge of eigenvalue decay of K_{ff}, and 2) additional error due to initialization scheme

Goal: bound the KL divergence between Q and $\widehat{P}$. Now they have: 10 removed dependence of bound on inducing point locations, and 2) Make intuitive assumptions about distributions of training data.

Main result:

Theorem 22. *Using the k-DPP initialization and assuming $x_i \sim p(x)$:*

$$D_{KL}\left(Q \mid\mid \widehat{P}\right) \leq \frac{NM + \textit{Dave : sometermImissed}}{2\sigma_n^2}\left(1 + \frac{||y||^2}{\sigma_n^2}\right).$$

Takeaways:

- Sparse approximations to GP regression converge quickly
- Smooth priors, dense input data $\implies$ very sparse approximations possible.

Dave: Meetings the rest of the day

6 Friday June 14th: Workshops

Workshops today!

6.1 Workshop: AI for Climate Change

First, John Platt (a follow up to his wonderful talk a few years back!)

6.1.1 John Platt on What ML can do to help Climate Change

Main Point: Energy research is not the same as fighting climate change.

→ Climate crisis: we are running out of our carbon budget! Without intervention, our best models project

Turns out we can't stop energy infrastructure (trillions invested). Best we can do is slowly stop using them: might achieve a 3% reduction (in energy use), or if we suppose some *miraculous* rate, maybe 10%.

→ Assuming *rapid decarbonization* (of 10%), in order to hit our goal (as defined by the Paris agreement), we have to start **now**!

→ Okay, but what if it's not radical? More conservative: 3.3%.
Measure: Final temperature rise (°).
What needs to Happen?:

- Decarbonize rapidly and quickly to avoid ¿2°.
- Zero carbon energy technologies must be shovel ready

  → Renewables are ready now, relatively scalable.
- Post 2040 zero carbon technologies still useful. →Backstop to avoid absolute worst climate change

  → Scenario of plentiful energy for everyone still achievable and desirable.

Idea: Suppose we want to deploy renewable energy tech (solar, wind, and so on).

→ Problem: Demand and supply fluctuate dramatically, and thus there will likely be gaps between when we have energy and need energy (consider a cloudy day!).

Code/work available exploring cost of renewables called DOSCOE [66]. Code: `https://bit.ly/DOSCOE`.

Q: Can ML help this problem?

- Most valuable zero-carbon sources are dispatchable (like hydropower)
- Can dispatch demand as well (turn on when carbon intensity of supply is low)

- Can ML/AI demand response? → Google reported 40% less energy spent on data center cooling via ML control.

Two Caveats:

1. Many sources contribute to climate crisis. Breakdown from John's "pie-chart of sadness": Industry: 21%, Electricity and Heat production 25%, Transportation: 14%, Buildings 6%, Agriculture/Forest and other Land use: 24%.

2. Require "carbon pressure": the free market doesn't encourage these practices!

    - Currently no incentive for demand response
    - Increased efficiency lowers price of electricity and fossil fuel
    - Less energy use → more income + economic growth → more energy.
    - Efficiency makes people better off, even with Jevon's paradox (save 1J of energy, cause >1J of energy consumption).

Idea: ML for material science? 1) New ways of making concrete?, 2) Fixing nitrogen?, 3) Better batteries?, 4) Neutron-resistant materials?

→ Other ideas: detect methane leaks (w/ ML)? Telepresence to reduce transportation (notably a very high bar)? Optimize freight?

Other things ML can do to help:

1. Flood forecasting:

    - Floods are bad now: 9.8 billion annual damage, affects 250 people per year
    - Will get worse at higher temperature!
    - Goal: forecast flood effects using public data.
      → Problems: data usually too noisy, floods usually in flat areas.
    - One solution: use ML to derive surface maps, can be used to forecast more accurately

2. Moonshot: push CO2 out of atmosphere in second half of the century.

    - Bio-energy with carbon capture and sequestration (already assumed in IPCC pathways).
    - Increase carbon in soil via plants (soil contains $> 3\times$ carbon of atmosphere)
    - Free air capture (David Keith claims \$100 per ton cost).
    - Can ML/AI help any of these?

Summary:

1. Climate and energy is a huge problem

2. Major takeaway: multiple time scales 1) ML/AI can help long term technologies (post 2040) 2) But, need immediate and radical carbonization now.

3. Many sources/sinks of greenhouse gases. But, no single silver bullet, must push on many fronts at once.

4. No purely technological solution.

Call to Action:

1. Find an ML application (get inspiration from recent paper)

2. Collaborate with domain experts, and 3) Start work and don't settle!

3. Don't be satisfied with saving 10,000 tons of CO_2. → This a huge problem: aim for megatons!
   → Also: massively hard problem. Most things don't work.

.....

6.1.2 Jack Kelly: Why It's Hard to Mitigate Climate Change, and How to Do Better

Goal: Help us reduce emissions as rapidly as possible.

Central Question: Why is it hard to reduce carbon emissions?

Visual: industry on the left, a "mountain of pain" in the middle", researchers and start ups on the right. What is this mountain, and how can we mitigate it?

Jack recently founded Open Climate Fix, a non-profit based on two hunches:

1. Open science, share data, help researchers and companies be successful in their endeavors on fighting climate change,

2. Develop better forecasting tools for helping predict the amount of available solar energy.
   → Developing an open platform of solar panels in the world. With better forecasts, less spinning reserves.

Challenges:

1. *Incentives:* Wrong incentives in the world—companies optimized to maximize profit, scientists optimize h-index, neither of which well align with climate challenges.

2. *Data:* Lots of challenges here:

   (a) 1) organizations reluctant to share (companies believe it has commercial value, data the new oil, etc),

   (b) Size

   (c) Under diagnosed systems

   (d) Quality: often extremely noisy, missing labels, and so on.

   (e) Access: sometimes in different/unusual physical forms, no standardization.

(f) No perfect simulators.

But, some solutions: 1) Budget lots of time for data engineering! 2) Ask questions early, 3) Find people who can help, 4) Be flexible, and 5) Use simple models.

3. *Intellectual Property:* Most things are behind closed doors.

   → But, some solutions: 1) Open source everything, 2) Talk to commercialization teams (universities/labs).

4. *Industry is unlikely to read your paper:* You need to sit down with people in industry and work on their problems. Help industry use your algorithm! Build a production service, consult, and so on.

5. *Demonstrating Performance:* No standard data sets or metrics or benchmarks.

Takeaways:

1. Spend as much time as possible with people in the industry you're targeting.
2. Be lean, build a minimum viable product, rapidly test your ideas. Fail, and fail fast, then move on to the next thing.

Audience Question: Can you mention why it's so useful to spend time with industry folks? Any specifics?

Jack A: Sure! Great point. Consider working with forecasting clouds. One thing we'll need to do is estimate the 3d structure of clouds. It's hard but with conversations with the meteorologists we can estimate volumetric info from rain fall and numerical weather predictions. Super valuable talking to industry folks, couldn't imagine doing this online.

Audience Question: Do you think a carbon tax could be better communicated to put it in motion?

Jack: Great question! Maybe we need to change the term (people don't like the term "tax"). Figuring out a better way to communicate such things is really important.

.....

6.1.3 Andrew Ng: Tackling Climate Change with AI through Collaboration

Backdrop: We have already seen massive impacts from climate change (see: more fires, flooding, and other disasters).

→ Many of our children will be alive in 2100. The decisions we make today will directly shape the world they live in.

Lots to do: 1) Mitigation (energy, industry, buildings, forest), 2) Adaptation (disaster prevention, societal/ecological) 3) Alteration (sequestration), 4) Climate Science.

Project One: Methane prediction (second most important greenhouse gas).

- Methane comes 50/50 from human and natural sources.
- Biggest source of methane source are *wetlands*.
- Unlike CO2: Limited number of sensors around the world to detect methane.

  → Andrew and colleagues visited a methane sensor: every few weeks, one of the researchers has to visit the sensor to collect data.
- Ng et al. now have access to 37 methane sensors around the world.

  Idea: Use these data to then predict methane emissions elsewhere in the world! Using data from these sensors, can now predict methane all over the world.

Project Two: Wind turbine detection

- The world is moving toward renewables
- But, a problem: with the way solar and wind farms are rolled out, we don't often know where they are.

  → To facilitate growth and integration of renewables, we need to *locate* renewables.
- Can improve USGS database of renewable location by: 1) automatic curation, 2) updating frequently.
- To locate wind turbines:
  - Data consisting of 100k images of locations of wind turbines.
  - Run detection on 1.8M images
  - Baseline model: DenseNet-121
  - Weakly supervised localization: GradCAM.
- *Estimating Wind Energy:* With these models, have a good estimate of location of wind turbines *and* a good estimate of wind in the US (see: http://hint.fm/wind/).

  → Combined with their work, DeepWind, can then predict energy output of wind turbines in the USA.

Other Projects:

- Machine learning for Modeling Atmospheric Convection [61]
- Spatiotemporal Global Climate Model Tracking
- Mitigation: optimizing renewables. DeepMind predicts wind power output 36 hours ahead of schedule.
- GasNet: using deep learning to automatically detect methane leaks from real-time videos at natural gas power plants [89].
- Economic Impact: better understanding of how climate change will impact economically different countries [21].

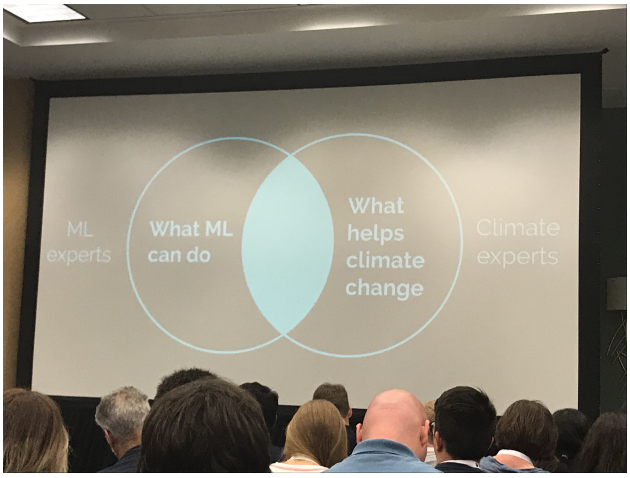


Figure 8: Call for collaboration across ML and climate science.

Echoing Jack's point: we need to collaborate with climate scientists, ecologists, meteorologists, and so on! See Figure 8.
Concluding thought: In eras of technological disruption, collaboration matters. Work together! Share data! This problem is bigger than ourselves.

.....

6.2 Workshop: RL for Real Life

I arrived just in time for the panel.

6.2.1 Panel Discussion

The panelists are Craig Boutilier (CB), Emma Brunskill (EB), Chelsea Finn (CF), Mohammad Ghavamzadeh (MG), John Langford (JL), David Silver (DS), and Peter Stone (PS). I'll abbreviate each by their initials.

Moderated by Alborz Geramifard (AG).

Q: What application areas are most promising for RL and why?

JL: Bread and butter baseline is personalization—of news stories, of advice, layouts, many other things.

CB: For me, interested in agents that can act on behalf of users, lots of problems though (multi-agent, preference elicitation). Recommender systems are promising but also challenging. We won't see wins short term, but recommender systems long term over long periods of time should serve as a really valuable test bed and a way to stress RL algorithms.

PS: We need a definition of promising to answer the question. Excited about promise of RL in robotics and manipulation, like my colleagues, the general class of user engagement through nudges/influencing people's behaviors in helpful ways (insidious ways, too, though). Help people rehab or take medicine; well fit for RL. We do have to think about what settings in the real world are safe/well fit.

CF: The best fits for RL are not really the problem the community focuses on. MuJoCo for instance has served its purpose for a time, but we need to scale now to partial observability, transfer, continuous state/action. Robotics fits that well though.

JL: Moving beyond bread and butter that are deployed right now, I think system optimization is a natural area for RL application. Most of these systems have a good recording over time, so lots of data (and don't have to wrestle with uncertainty).

EB: Personalization is a raw term, lots of things are changing in the world. Prior applications (when I was in grad school) was robotics and games. Opportunities now for personalization, given the world right now, is both a huge responsibility *and* exciting opportunity.

DS: Follow up to something John said in his talk—if we get the foundations right, there will be applications everywhere. If we get an algorithm that just works, we should expect to see RL in all kinds of places. Who knows what methods/approaches are out there? Let's continue to push the foundations. We should of course keep pushing and try to deploy it too, but I think we can expect to be surprised too. Looking a little farther in the future, one open area where there's huge potential for getting RL right is personalized health care. Can transform many people's lives.

MG: Lots of OR applications everywhere—think about system optimization, huge opportunity for RL. What's happening with Amazon storage facilities. One more thing for recommendation systems. Recommender systems are everywhere and we have tons of data. Consequences of our actions are clear, need to consider safety and robustness. Although consequences of actions in recommender systems aren't catastrophic, it's still important we get it right.

.....

AG Q: Great segue into next question. How should we explore in these problems?

DS: Part of every framework, even in simulation. We always have to handle explorations (agents die, we don't receive more reward/data). If it has opportunity to come back and learn more, then we can still learn from the *lack of experience* to get more. That's nice in recommender systems too since we get "parallel lifetimes", if a stream of experience stops we can still learn from it. It's signal that someone stopped using the system

CB: Yeah except for people that die in certain trajectories. Maybe not in recommender systems but in broad multi-user systems. In multi-user systems you get to distribute these decisions/explore. Can learn from individual users in a distributed way to guarantee safety (no individual action that can be so bad). In clinical setting, in recommender setting, can distribute exploration to not do

damage to any individual.

EB: I think this is a really interesting question in a lot of settings—it's a reason to use policy gradients or imitation learning or methods with monotonic improvement. Some work shows we can be optimal too (be greedy w.r.t. information we have so far can still do well). These kinds of approaches help us avoid the major burdens of exploration. Lots of tools and directions to make progress.

PS: To me it's a fascinating question of meta-learning. When we do have a big population of users, should we pick a small subset and burn them and use our exploration budget, or should we explore a bit across all users (or other strategies)? Meta-objective: keep population happy with what you're doing, so: how should we achieve this?

CB: Quick response. If we have a model of cost of exploration this all falls out of the wash.

JL: This is a place where theory is really important. If we're going to optimize only for efficiency, you'll be driven away from ε-greedy, and if we want to think about safety, we can use bounds to guide decision making in a safe way.

MG: Ways to make exploration more conservative. How conservative can we be? Well that depends on individual task/product/setting. If we know information about users/domain, we can set the exploration appropriately. Another important question: how do we fill the gap between simulation and the real world? Really hard. But having those simulators will help us explore in the world.

DS: Even in the real world, there are times when users leave, and we can actually continue to learn from these settings. If we do it right, it's all there in the framework: how should we maximize reward? Sometimes we take actions that keep people engaged or not, and that's ok.

EB: One thing we've been thinking about is how to keep exploration safe. Idea: policy certificates. Can promise explicitly how a particular method will explore/behave. We can have tools that can allow us to expose when we're going to explore and when we won't.

.....

AG-Q: What are general principles that are key for bringing RL to the real world?

JL: First principle is that the framing of the problem is more important than anything else. Second one: if I gave you a long laundry list of things that are more important in real world than simulated worlds.

CF: One principle important to safety is to *identify* cases where it's safe to explore. Maybe you buy a small robot first and then transfer what it learns (like it's okay to fall down as a kid to learn from it). Identify situations where it's safe to explore, then can maybe be equipped to explore in new situations.

PS: Biggest lesson—simulators are never perfect! Super appealing because they're safe and we can get lots of data. We should use simulators, but we're never going to get them perfect. Efforts to make simulators perfect are misguided we should make new methods.

CF: Completely agreed! If you could make a simulator that's perfect you've already solved the problem. Learning in the real world is going to be required.

CB: One other principle I'd like to raise—reward function and objective function design. That first reward function you specify is never the right one! We walk away and say "this is the best policy", but it's clearly not based on the behavior, we got the reward function wrong. Human-in-the loop methods are really important for correcting/designing the right objectives/reward functions.

EB: I want to echo that—my lab does a lot of work in education and health care. Coupling this work (and these choices of reward functions) with domain experts is massively important. The first known application of RL to education was in 1999 to understand how students do well on problems. Example: give students problems until student gets it right, then give the student the same one over and over again (exploit!).

DS: Number one principle in my own work—scalability. We should only try to strive for methods that scale well with resources (three: 1) compute, 2) experience, 3) memory). If we can achieve that level of scalability, we just need to scale up the resources. If we design something where we put a ceiling into the method itself (stop exploring, stop learning, other constraints), that ceiling will hurt us. Goal should always be scalability. If you look at the success of deep learning, I claim it's successful because it scales with all three things (compute, experience, memory). It does better and better with these resources. So we should find the RL methods that scale.

MG: First principle is to make sure RL is the right solution for your problem. Second, make sure we have the data available for RL. Think about supervised learned, if half your data is bad, you can still do well. Same situation for RL we can't do well. Reiterate what Craig said: reward functions are super important. Part of the reward function might be company profit (as opposed to say, customer happiness).

JL: I want to mention I agree with David. The process of educating people on the realities of RL is always fun.

PS: The design principle of our platform at Cogitai is a two part hypothesis: 1) RL is ready for the real world (that is, we, in this room, are ready to solve problems in the real world), and 2) the real world is ready for RL. This second piece is what John mentioned. Lots of problems in the world we know RL is ready, but it's a big challenge for us to work with folks from outside RL/ML to help them identify which problems are RL problems and help them use RL tools to solve their problems. The first bucket is easy (us find RL problems) while the second is hard (convince others). Important point from David: we're going to be surprised! It wasn't the people that made the smart-phone that built the apps. We build the technology and tools that enable people outside this room to use these ideas. That's the goal. Get it robust enough that we can be surprised.

CB: Widespread acceptance of RL is ultimately an important goal. We could have been having this same conversation N years ago about ML in industry. Not very long ago just getting ML to be accepted in large scale production systems, but here we are.

.....

Audience-Q: (paraphrased by PS): can we compare approaches to supervised learning and RL on real problems?

PS: Well, I don't think it's well posed to do this test. There are RL problems and supervised problems. These are different.

JL: Consider recommendation. Enormous number of actions. To apply RL to an enormous number of things often won't work. If you have a supervised signal source, that doesn't suffer from same sample complexity issue. Could apply RL on top to get the personalization which you can't really get from SL. There is no representational limit to RL—the same representations that can work for SL can work for RL. We've done reductions between these and this works.

CB: You can treat RL problems as supervised learning problems in recommender systems. Make a prediction, store it. The differences is whether we're being myopic vs. looking at the long term. The question is, if you learn representations about how users act in long term vs. short term, do you get some benefit? I don't know.

DS: A comment about A-B testing. You raised it as if it were outside RL, but you can think about it as a bad RL algorithm. Do this one thing and stick to it, do this other thing, then compare how these do after the trial. That's part of RL but we can also do better like contextual bandits.

.....

Audience-Q: On horizons/reward design. Algorithms might exploit human weaknesses more heavily in the future, a lot of the behaviors of ML algorithms are probably due to their myopic nature. Being able to look ahead for multiple time steps might allow for better reward functions. Thoughts? Second, when we use RL for problems with humans in the loop, what is the longest horizon we can actually use.

JL: Right now, typically, horizon is relatively short (hours, minutes, days). As that increases data is harder to come by. If we succeed in RL there's no reason it can't be lifetimes.

EB: Depends on what level of granularity you're using: hierarchical can help here! Both of those views are definitely possible. The technology we're developing is dual use and an amplifier. Will be constrained by the people and not the techniques themselves.

CB: Totally agree with with John and Emma. I still don't think we have the technology in our hands to do multi-scale planning, but we'll get there. With respect to doing right by the user.

One important thing: we really don't have good methods for understanding user preferences and utilities, and how their behaviours reveal these. Real limiting factor to applying RL over long horizons is a developing techniques where we can know quickly what a users' utility function is. RL is also about planning but we're far from being able to do that.
CF: On reward design and David's comment about scalability—we should be thinking about algorithms that scale, but if we're in the real world, reward function supervision isn't a scalable source of feedback. In those settings is important to consider how we should frame the problem so it's scalable. Shouldn't be afraid to change the problem statement. For example, if you want a scalable algorithm, design a setting and algorithm that can scale with the resources involved. Don't really feel constrained by the classical RL statement if your problem doesn't match that.

.....

Audience-Q: First, when is a policy safe to deploy?

PS: Tempting but wrong: Deploy when optimal! Will definitely depend on domain. Testing should be empirical—can we drive better than people?

EB: Can learn a lot by looking at verification (in RL, in deep learning). For these examples, sometimes people can't be in the loop to verify. Can use these to avoid worst-case scenarios (see: making aircrafts avoid collisions). Some safety guarantees.

JL: Practical answer: frame your problem so no actions are particularly bad/catastrophic. Then do RL on top of that.
PS: RL has components that are Art and components that are Science. We've been trying to make it 100% science, but we need to accept that parts of it are art.

.....

Lihong Li on closing remarks:

- Goal of workshop: Bring together researchers and practitioners from industry and academia interested in addressing practical and/or theoretical issues in applying RL to real life scenarios.
- RL in real life: games, robotics, transportation, resource allocation.
- Challenges: partial observability, multi-agent systems, stochasticity, large action set, simulators, evaluation, generalization.
- Post workshop discussion via slack (on the workshop website: https://sites.google.com/view/RL4RealLife).

.....

Next up is the afternoon session of a closely related workshop.

6.3 Workshop: Real World Sequential Decision Making

The workshop begins with an invited talk by Emma Brunskill.

6.3.1 Emma Brunskill on Efficient RL When Data is Costly

Questions: What's the best way to teach a student coding? How should this patient be treated for cancer? Will job training increase future income?

→ We'd like to answer these question in an evidence based way! See: interventional science.

Problem: The way we do interventional science at the moment is flawed. Example: run a bootcamp. See which intervention leads to higher graduation rates. One idea: dynamically adapt how we assign students to conditions. $\implies$ use RL!

→ Can use RL to understand the science of intervention.

Historical test beds: video games and robots. Now, though: new kinds of test beds.

Recent changes: massive compute, massive sensors, and ability to dynamically interact with people. $\therefore$ opportunities to apply RL have emerged! Lots of negatives to this, but many opportunities too.

Goal: Learn to make good decisions faster (in a statistically efficient way), to benefit more people.

Q: How can we create AI that help students?

A: Project—RL assistance leads to 1.3× faster learning, longer learning, and more learning.

Key Techniques:

1. Exploration!
2. Counterfactual/Batch Off Policy RL (focus today).

   → Tension between beautiful math and impacting society. But! This area allows for both.

   → Growing interest in Causal Inference and ML. See: *The Book of Why* by Pearl and Mackenzie [63].

Q: How can we learn from historical sequential data?

A: It's really hard! Many challenges: data generation process is fundamentally different. That is:

$$\text{Different Policies} \implies \text{Different Actions} \implies \text{Different States}$$

That is: decisions *compound*.

→ Moreover: we don't even know the behavioral policy.

Classic challenge: Batch Off Policy Evaluation. We're given a dataset D of trajectories gathered by some policy π_D, and we would like to compute the value of some new policy π'.

→ One spin: batch off policy *optimization*. Just want to know how to *act*, given data from prior policy.

Example: when should we treat patients? Really, how can we come up with a *personalized* policy for when to intervene (take a ventilator off, for instance).

→ One specific kind of intervention policy: vast majority of decisions are "nothing", then suddenly we have to intervene. Let's call such a decision the `when-to-treat-policy`.

Q: When should we start HIV treatment? When should we stop some kind of treatment?

→ These cases are well captured by parameterized policies conditioned on relevant contexts.

Two solutions: Importance weighting (also called "inverse propensity weighting"). Goal is to estimate expected value of new `when-to-treat-policy`.

→ Advantage: simple and consistent → But: uses only trajectories that match data, not very robust, high variance.

New Idea: Advantage decomposition: τ_π is a stopping time w.r.t. some filtration `when-to-start-treatment` according to policy π. Then:

$$\mu_{now}(S_{1:t}) = \mathbb{E}\left[\text{Return}\right] \tag{20}$$
$$\mu_{next}(S_{1:t}) = \mathbb{E}\left[\text{Return}\right] \tag{21}$$

If we can determine which $\Delta_\pi := V^\pi - V^{\pi_0}$, where π_0 is the `never-treat` policy, we can determine the advantage of a particular treatment.

→ Specifically: want to estimate Δ_π from data, even *without* π.

Yields the **Advantage Double Robust** (ADR) Estimator for off policy evaluation in the when to treat context. Yields regret bounds, too. Further differences visualized in Figure 9.
Experiments: Keeping a health metric above 0, where state evolves over time in a stochastic way (with brownian motion). Use treatments to try to help outcomes.

→ Given observational data from clinic, want to find way to choose effective treatments.

Another important question: how can we estimate V^*? (perhaps, specifically to help human-in-the-loop)

Q: Can we identify with $< O(d)$ samples if there exists a linear threshold policy with expected value over a threshold b.

A: For some distributions over contexts, yes! This is before we could return any policy.

.....

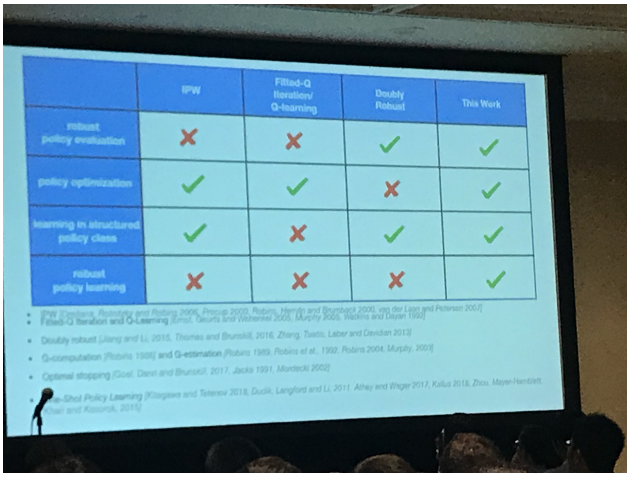


Figure 9: Comparison of different off-policy evaluation types.

6.3.2 Miro Dudik: Doubly Robust Off-Policy Evaluation via Shrinkage

Problem Setting: Contextual bandits. Comes up when learning from interaction data.

> **Definition 27** (Contextual Bandit): *An interaction protocol where a user comes to a service, the service presents objects to the user, and the user makes some decision based on the objects given.*
> *That is, we follow: 1) Observe context x, 2) Choose action a, and 3) Receive reward r, then sample a new context i.i.d.*

→ Lots of applications of contextual bandits.

Running Example: Think about news recommendation as a canonical instance of a contextual bandit. Repeatedly the algorithm perceives a context (user data), suggests some article to the user, and observes some data indicating the quality of the suggested article (click).

→ Challenging because reward model typically trained on prior data, $(x_1, a_1, r_1), \ldots, (x_n, a_n, r_n)$ (often for different policy). Moreover, policy by be randomized.
Fundamental task: off-policy evaluation. What would have been the click rate if we had followed some other policy?
Standard Approach: importance weighted estimates. Reweight prior experiences according to:

$$\frac{\pi_{new}(a_i \mid x_i)}{\pi_{old}(a_i \mid x_i)}.$$

→ Unbiased, but high variance.

Key Question: Can we take advantage of the reward estimator $\hat{R}$ to improve over these methods?

Two tricks:

1. Doubly robust estimate: use direct estimate as a baseline, apply importance weighting to estimate the *departure from the baseline*.

   $\rightarrow$ Still unbiased!

2. Weight shrinkage (see justification below). Idea is to clip estimator with a parameter λ:

$$\hat{w}(x,a) = \begin{cases} w(x,a) & txifw(x,a) < \lambda \\ \lambda & \text{if} w(x,a) \geq \lambda \end{cases}$$

   Allows for λ to control, explicitly, the bias-variance trade-off, while ensuring weights are smaller.

Definition 28 (Doubly Robust Estimate): *The doubly robust estimator is as follows:*

$$\hat{V}_{direct} + \frac{1}{n} w(x,a)(r - \hat{r}(x,a))^2.$$

Critically: doubly robust estimator is asymptotically optimal, but in finite samples, other estimators tend to be more accurate.

$\rightarrow$ Critically:

$$\text{Mean Squared Error} = \text{Bias}^2 + \text{Variance}.$$

But: IPS and DR are unbiased, so error is entirely variance. Their variances are:

$$IPS : \frac{1}{n}\mathbb{E}[w^2(x,a)r^2] + O(1/n) \tag{22}$$

$$DR : \frac{1}{n}\mathbb{E}[w^2(x,a)(r - \hat{r}(x,a))^2] + O(1/n) \tag{23}$$

$$\tag{24}$$

Even if $\hat{r}(x,a) = \mathbb{E}[r \mid x,a]$, so $\hat{V}_{\text{direct}}$ is unbiased, we suffer to due large weights if rewards are inherently noisy. Thus: weight shrinkage (see above).

Q: Many ways to shrink weights? Which one should I use?

A1: Pessimistic shrinkage! Only assume

$$|r - \hat{r}(x,a)| \leq 1$$

This induces bias of $\mathbb{E}[(\hat{w}(x,a) - w(x,a))(r - \hat{r}(x,a))]$.

New variance $\leq \mathbb{E}[\hat{w}(x,a)]$.

$\therefore$ Can come up with the right kind of weight shrinkage method.

A2: Optimistic shrinkage! Assume $\hat{r}$ is fitted by optimizing weighted square error with a weighting function $z(x, a)$:

$$\frac{1}{n}\sum_{i=1}^{n} z(x_i, a_i)(r_i - \hat{r}(x, a))^2$$

Then we can bound the bias and the variance again (using similar pareto-optimizing-trick). New estimator based on λ:

$$\hat{w}(x, a) = \frac{\lambda}{w^2(x, a) + \lambda} w(x, a).$$

As $\lambda \to \infty$, leading coefficient becomes 1, recover DR estimator, as $\lambda \to 0$, recover the Dave: Missed it.

Experiment 1: Do we need both optimistic and pessimistic shrinkage? How often across 108 conditions is each better? (for three different choices of reward predictors).

→ Finding: under all three reward predictors, pessimistic shrinkage tends to be better than optimistic is better (but there are conditions when optimistic is better, or they tie).
Experiment 2: Do we need all these reward predictors?

→ Finding: Again, some cases where each predictor is superior (when using the direct estimator). → When using DR almost always a tie across the different estimators. → Moreover, when using DR with shrinkage, one of the reward estimators ($z = w^2$) suddenly becomes dominant. So: if using shrinkage, might need to rethink how you're using reward estimators.

Summary:

- For best finite-sample performance of DR: → Important to consider both reward predictor and weight shrinkage → Different reward predictors applicable in different settings → Model selection matters and needs to be studied more
- Next steps: structured actions, continuous actions.

.....

Dave: And that's a wrap! Sadly I have to miss the workshops tomorrow (and the rest of today's session). Flying out of Long Beach shortly.

Edits

Many thanks to the following individuals for sending along helpful edits/catching typos:

- Zeno Zantner for catching typos.
- Brandon Amos for catching typos.

References

[1] Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2897–2905, 2018.

[2] Tameem Adel and Adrian Weller. Tibgm: A transferable and information-based graphical model approach for reinforcement learning. In *International Conference on Machine Learning*, pages 71–81, 2019.

[3] Riad Akrour, Joni Pajarinen, Jan Peters, and Gerhard Neumann. Projections for approximate policy iteration algorithms. In *International Conference on Machine Learning*, pages 181–190, 2019.

[4] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

[5] Amiran Ambroladze, Emilio Parrado-Hernández, and John S Shawe-taylor. Tighter pac-bayes bounds. In *Advances in neural information processing systems*, pages 9–16, 2007.

[6] Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 233–242. JMLR. org, 2017.

[7] Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 243–252. JMLR. org, 2017.

[8] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[9] Mikhail Belkin, Daniel J Hsu, and Partha Mitra. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In *Advances in Neural Information Processing Systems*, pages 2300–2311, 2018.

[10] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

[11] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[12] Alon Brutzkus and Amir Globerson. Why do larger models generalize better? a theoretical perspective via the xor problem. In *International Conference on Machine Learning*, pages 822–830, 2019.

[13] Olivier Catoni. Pac-bayesian supervised classification: the thermodynamics of statistical learning. *arXiv preprint arXiv:0712.0248*, 2007.

[14] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip S Thomas. Learning action representations for reinforcement learning. *arXiv preprint arXiv:1902.00183*, 2019.

[15] Ching-An Cheng, Xinyan Yan, Nathan Ratliff, and Byron Boots. Predictor-corrector policy optimization. *arXiv preprint arXiv:1810.06509*, 2018.

[16] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

[17] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

[18] Imre Csiszár. I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, pages 146–158, 1975.

[19] Robert Dadashi, Adrien Ali Taïga, Nicolas Le Roux, Dale Schuurmans, and Marc G Bellemare. The value function polytope in reinforcement learning. *arXiv preprint arXiv:1901.11524*, 2019.

[20] Christoph Dann, Lihong Li, Wei Wei, and Emma Brunskill. Policy certificates: Towards accountable reinforcement learning. *arXiv preprint arXiv:1811.03056*, 2018.

[21] Noah S Diffenbaugh and Marshall Burke. Global warming has increased global economic inequality. *Proceedings of the National Academy of Sciences*, 116(20):9808–9813, 2019.

[22] Gintare Karolina Dziugaite and Daniel M Roy. Entropy-sgd optimizes the prior of a pac-bayes bound: Generalization properties of entropy-sgd and data-dependent priors. *arXiv preprint arXiv:1712.09376*, 2017.

[23] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[24] Mahdi M Fard and Joelle Pineau. Pac-bayesian model selection for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1624–1632, 2010.

[25] Mehdi Fatemi, Shikhar Sharma, Harm Van Seijen, and Samira Ebrahimi Kahou. Dead-ends and secure exploration in reinforcement learning. In *International Conference on Machine Learning*, pages 1873–1881, 2019.

[26] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[27] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[28] Marta Garnelo, Dan Rosenbaum, Chris J Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J Rezende, and SM Eslami. Conditional neural processes. *arXiv preprint arXiv:1807.01613*, 2018.

[29] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. *arXiv preprint arXiv:1901.11275*, 2019.

[30] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. *arXiv preprint arXiv:1906.02736*, 2019.

[31] Badih Ghazi, Rina Panigrahy, and Joshua R Wang. Recursive sketches for modular deep learning. *arXiv preprint arXiv:1905.12730*, 2019.

[32] Omer Gottesman, Yao Liu, Scott Sussex, Emma Brunskill, and Finale Doshi-Velez. Combining parametric and nonparametric models for off-policy evaluation. *arXiv preprint arXiv:1905.05787*, 2019.

[33] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[34] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

[35] Josiah Hanna, Scott Niekum, and Peter Stone. Importance sampling policy evaluation with an estimated behavior policy. *arXiv preprint arXiv:1806.01347*, 2018.

[36] Anna Harutyunyan, Peter Vrancx, Philippe Hamel, Ann Nowe, and Doina Precup. Per-decision option discounting. In *International Conference on Machine Learning*, pages 2644–2652, 2019.

[37] Elad Hazan, Sham M Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. *arXiv preprint arXiv:1812.02690*, 2018.

[38] Jonathan J Hunt, Andre Barreto, Timothy P Lillicrap, and Nicolas Heess. Composing entropic policies using divergence correction. 2018.

[39] Craig Innes and Alex Lascarides. Learning structured decision problems with unawareness. In *International Conference on Machine Learning*, pages 2941–2950, 2019.

[40] Alexis Jacq, Matthieu Geist, Ana Paiva, and Olivier Pietquin. Learning from a learner. In *International Conference on Machine Learning*, pages 2990–2999, 2019.

[41] Zhengyao Jiang and Shan Luo. Neural logic reinforcement learning. *arXiv preprint arXiv:1904.10729*, 2019.

[42] Yuu Jinnai, David Abel, Michael Littman, and George Konidaris. Finding options that minimize planning time. In *International Conference on Machine Learning*, 2019.

[43] Yuu Jinnai, Jee Won Park, David Abel, and George Konidaris. Discovering options for exploration by minimizing cover time. In *International Conference on Machine Learning*, 2019.

[44] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. *arXiv preprint arXiv:1902.00255*, 2019.

[45] Mikhail Khodak, Maria Florina-Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. *arXiv preprint arXiv:1906.02717*, 2019.

[46] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *arXiv preprint arXiv:1806.03836*, 2018.

[47] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018.

[48] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[49] Hoang M Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. *arXiv preprint arXiv:1903.08738*, 2019.

[50] Donghwan Lee and Niao He. Target-based temporal difference learning. *arXiv preprint arXiv:1904.10945*, 2019.

[51] Shiau Hong Lim and Arnaud Autef. Kernel-based reinforcement learning in robust markov decision processes. In *International Conference on Machine Learning*, pages 3973–3981, 2019.

[52] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[53] Hong Liu, Mingsheng Long, Jianmin Wang, and Michael Jordan. Transferable adversarial training: A general approach to adapting deep classifiers. In *International Conference on Machine Learning*, pages 4013–4022, 2019.

[54] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2378–2386, 2016.

[55] Ben London and Ted Sandler. Bayesian counterfactual risk minimization. *arXiv preprint arXiv:1806.11500*, 2018.

[56] Ali Malik, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon. Calibrated model-based deep reinforcement learning. In *International Conference on Machine Learning*, pages 4314–4323, 2019.

[57] Borislav Mavrin, Shangtong Zhang, Hengshuai Yao, Linglong Kong, Kaiwen Wu, and Yaoliang Yu. Distributional reinforcement learning for efficient exploration. *arXiv preprint arXiv:1905.06125*, 2019.

[58] David A McAllester. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

[59] Alberto Maria Metelli, Emanuele Ghelfi, and Marcello Restelli. Reinforcement learning in configurable continuous environments. In *International Conference on Machine Learning*, pages 4546–4555, 2019.

[60] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, R Venkatesh Babu, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. *arXiv preprint arXiv:1905.08114*, 2019.

[61] Paul A O'Gorman and John G Dwyer. Using machine learning to parameterize moist convection: Potential for modeling of climate, climate change, and extreme events. *Journal of Advances in Modeling Earth Systems*, 10(10):2548–2563, 2018.

[62] Matteo Papini, Alberto Maria Metelli, Lorenzo Lupo, and Marcello Restelli. Optimistic policy optimization via multiple importance sampling. In *International Conference on Machine Learning*, pages 4989–4999, 2019.

[63] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[64] Xingchao Peng, Zijun Huang, Ximeng Sun, and Kate Saenko. Domain agnostic learning with disentangled representations. *arXiv preprint arXiv:1904.12347*, 2019.

[65] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *International Conference on Machine Learning*, pages 5142–5151, 2019.

[66] John Platt, J Pritchard, and Drew Bryant. Analyzing energy technologies and policies using doscoe. 2017.

[67] Chao Qu, Shie Mannor, and Huan Xu. Nonlinear distributional gradient temporal-difference learning. *arXiv preprint arXiv:1805.07732*, 2018.

[68] Goran Radanovic, Rati Devidze, David Parkes, and Adish Singla. Learning to collaborate in markov decision processes. *arXiv preprint arXiv:1901.08029*, 2019.

[69] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. 2018.

[70] Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. 2018.

[71] Konda Reddy Mopuri, Phani Krishna Uppala, and R Venkatesh Babu. Ask, acquire, and attack: Data-free uap generation using class impressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[72] Joshua Romoff, Peter Henderson, Ahmed Touati, Yann Ollivier, Emma Brunskill, and Joelle Pineau. Separating value functions across time-scales. *arXiv preprint arXiv:1902.01883*, 2019.

[73] Vincent Roulet, Dmitriy Drusvyatskiy, Siddhartha Srinivasa, and Zaid Harchaoui. Iterative linearized control: Stable algorithms and complexity guarantees. In *International Conference on Machine Learning*, pages 5518–5527, 2019.

[74] Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G Bellemare, and Will Dabney. Statistics and samples in distributional reinforcement learning. *arXiv preprint arXiv:1902.08102*, 2019.

[75] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.

[76] Lior Shani, Yonathan Efroni, and Shie Mannor. Exploration conscious reinforcement learning revisited. In *International Conference on Machine Learning*, pages 5680–5689, 2019.

[77] John Shawe-Taylor and Robert C Williamson. A pac analysis of a bayesian estimator. In *Annual Workshop on Computational Learning Theory: Proceedings of the tenth annual conference on Computational learning theory*, volume 6, pages 2–9, 1997.

[78] Zebang Shen, Alejandro Ribeiro, Hamed Hassani, Hui Qian, and Chao Mi. Hessian aided policy gradient. In *International Conference on Machine Learning*, pages 5729–5738, 2019.

[79] Zhao Song, Ron Parr, and Lawrence Carin. Revisiting the softmax bellman operator: New benefits and new perspective. In *International Conference on Machine Learning*, pages 5916–5925, 2019.

[80] Wen Sun, Anirudh Vemula, Byron Boots, and J Andrew Bagnell. Provably efficient imitation learning from observation alone. *arXiv preprint arXiv:1905.10948*, 2019.

[81] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

[82] Corentin Tallec, Léonard Blier, and Yann Ollivier. Making deep q-learning methods robust to time discretization. *arXiv preprint arXiv:1901.09732*, 2019.

[83] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. *arXiv preprint arXiv:1901.09184*, 2019.

[84] Andrea Tirinzoni, Mattia Salvini, and Marcello Restelli. Transfer of samples in policy search via multiple importance sampling. In *International Conference on Machine Learning*, pages 6264–6274, 2019.

[85] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[86] Leslie G Valiant. A theory of the learnable. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445. ACM, 1984.

[87] Benjamin Van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. In *International Conference on Machine Learning*, pages 6401–6409, 2019.

[88] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[89] Jingfan Wang, Lyne P Tchapmi, Arvind P Ravikumara, Mike McGuire, Clay S Bell, Daniel Zimmerle, Silvio Savarese, and Adam R Brandt. Machine vision for natural gas methane emissions detection using an infrared camera. *arXiv preprint arXiv:1904.08500*, 2019.

[90] Ruohan Wang, Carlo Ciliberto, Pierluigi Amadori, and Yiannis Demiris. Random expert distillation: Imitation learning via expert policy support estimation. *arXiv preprint arXiv:1905.06750*, 2019.

[91] Kelvin Xu, Ellis Ratner, Anca Dragan, Sergey Levine, and Chelsea Finn. Learning a prior over intent via meta-inverse reinforcement learning. *arXiv preprint arXiv:1805.12573*, 2018.

[92] Lin Yang and Mengdi Wang. Sample-optimal parametric q-learning using linearly additive features. In *International Conference on Machine Learning*, pages 6995–7004, 2019.

[93] Lantao Yu, Jiaming Song, and Stefano Ermon. Multi-agent adversarial inverse reinforcement learning. In *International Conference on Machine Learning*, pages 7194–7201, 2019.

[94] Andrea Zanette and Emma Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. *arXiv preprint arXiv:1901.00210*, 2019.

[95] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702, 2019.