NeurIPS 2019 Notes
Vancouver, BC, Canada

David Abel*
david_abel@brown.edu

December 2019

Contents

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

This document contains notes I took during the events I managed to make it to at NeurIPS 2019 in Vancouver, BC, Canada. 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.

I was in meetings most of the conference this year, so unfortunately I missed a lot more talks than usual. Of the things I missed, I'm most looking forward to catching:

1. Prof. Celeste Kidd's talk on "How to Know": https://slideslive.com/s/celeste-kidd-21913.
2. The ML for Climate Change workshop https://www.climatechange.ai/NeurIPS2019_workshop.html.
3. More from the Optimization Foundations in RL workshop (https://optrl2019.github.io/).

1 Tuesday December 10th: Main Conference

The main conference begins!

1.1 Session: Deep Learning Theory

First up is the "Outstanding New Directions" paper on uniform convergence and generalization error in deep learning.

.....

1.1.1 Outstanding New Directions Paper: Uniform Convergence may be Unable to Explain Generalization in Deep Learning [42]

Talk by Vaishnavh Nagarajan.

Question: Why do over-parameterized networks generalize well?

Previous work: bound generalization bound (test error - train error $\leq$ generalization error), but these bounds all come from *uniform convergence*.

This paper: this direction of using uniform convergence to solve the generalization error puzzle may be limited!

Definition 1 (Uniform convergence bounds): *A bound on the generalization gap based on the representational complexity of the whole hypothesis class:*

$$test\ error - train\ error \leq O\left(\sqrt{\frac{f(\mathcal{H})}{m}}\right) \tag{1}$$

previously proposed solution; refined uniform convergence bounds, leading to lots of new tools; PAC-Bayes, Rademacher Complexity, Covering Numbers, compression, and so on.

But! Each of these bounds:

1. Is too large/grow with parameter count.
2. Or are too small but don't hold on the original network (require change to network).

First Finding: these existing generalization bounds *increase* with training set size empirically.

- Focus has been on parameter count dependence, not training set size.
- We need to worry about training-dataset-size too!

Second Finding: Provable failure of uniform convergence.

- There are situations where any uniform convergence bound, however refined, will fail to explain generalization (it will be vacuous).
- Main proof idea: *tightest uniform convergence*. What is the smallest uniform convergence for a given learner/hypothesis class? Can show even this is vacuous for deep nets.

  → High level: even most refined hypothesis class is too complex.

Setup:

- Given a training set $\mathcal{S}$, algorithm $h_S \in \mathcal{H}$, then w.h.p. $\mathcal{S} \sim D$:

$$\text{test-error}(h_S) - \text{train-error}(h_S) \leq \text{generalization gap}$$

- Now, exclude all irrelevant hypothesis leads to the *most refined hypothesis class]* $\mathcal{H}^*$.
- Proof idea is to show that, in binary (hypersphere) classification, even this most refined class will have vacuous generalization bounds.
- Learning details: 1000-dimensional hypersphere classification, 1-hidden layer ReLu, 100k units, SGD, trained to zero error.

Conclusion: Can uniform convergence provide a complete answer to the generalization puzzle? We believe *it cannot*

Future Work:

- Mathematically characterizes the complexities in the decision boundary of deep networks
- Explore other learning theoretic tools such as stability
- Need to go beyond uniform convergence to explain generalization in deep nets.

.....

1.1.2 On Exact Computation with an Infinitely Wide Neural Net [3]

Talk by Rusong Wang.

Recent work: shows NNs with sufficiently large width can achieve 0 training error via gradient descent [1, 17].

This work: relationship between infinite width network and neural tangent kernel (NTK).

Questions:

1. Can we formally show prediction of NNs is equivalent to NTKs when width is sufficienlty large?
2. How does NTK perform?

First Contribution:

Theorem 1. *When width is sufficiently large (poly in data, depth, and $1/\varepsilon$), the predictor learned by applying GD on a NN is ε close to kernel regression predictor of corresponding NTK.*

Second Contribution:

- Experiments with NTK for CNNs with efficient GPU implementations.
- Compare test accuracy of different architectures/modifications of different width to NTK.
- Main Observations:
  1. Performance of NTK is highly correlated with CNN.
  2. Still a gap between performance of CNN and NTK.

Future Directions;

- Understand design of neural net architectures and common techniques in deep learning such as batch norm, residual layers.
- Combine NTK with other methods.

.....

1.1.3 Generalization Bounds of SGD for Wide and Deep NNs [9]

Talk by Yuan Cao.

Previous observation (see above references): wide nets can achieve zero training error, even when the test error remains large.

Questions:

1. Why do wide neural nets generalize?
2. What functions can be learned by wide nets?

Results:

1. As long as an NN is wide enough, the generalization error can compete with the best function class in the hypothesis class.
2. New generalization bounds based on the NTK.
3. The "classifiability' of underlying data distribution can be measured by some function of the Gram matrix of the NTK (see Jacot et al. [31]).

Proof is based on: 1) Deep ReLu nets are almost linear in parameters in a small neighborhood around initialization, and 2) Loss is Lipschitz continuous and almost convex.

$\rightarrow$ This result is applicable to general loss functions.

Summary:

1. New generalization bounds for wide DNNs that do not increase with network width
2. A random feature model (NTRF) naturally connects over-parameterized DNNs with NTK
3. Quantification of classifiability of data.

.....

1.1.4 Efficient and Accurate Estimation of Lipschitz Constants for DNNs [20]

Talk by Alex Robey.

Main Idea: Efficiently measure Lipschitz constant in NNs.

Definition 2 (Lipschitz Constant): *The smallest L_2 such that:*

$$||f(x) - f(y)||_2 \leq L_2||x - y||_2 \ \forall_{x,y\in\mathbb{R}^2}. \tag{2}$$

$\rightarrow$ Extremely useful in ML for exploiting smoothness/structure in function or data space. Also: a lower Lipschitz constant implies more robustness.

But! It's NP-Hard to compute in general. So how can we estimate it efficiently?

- Simple method: can upper bound lipschitz constant via product or norms, but leads to highly conservative upper bound.
- **This paper;** transform NN into a family of incrementally quadratically constrained linear Nets (IQCLN).
- Use optimization techniques (induce a semi-definite program) to give a tight upper bound to the Lipschitz constant of the original net.

Q: How does this bound compare to existing bounds in the literature?

A: This new estimation is much tighter than existing methods.

Also take a close look at adversarial robustness as it relates to Lipschitz constant.

.....

1.1.5 Regularization Effect of Large Initial Learning Rate [38]

Talk by Colin Wei.

Observation: Large initial learning rate is crucial for generalization.

This Paper: Explain this phenomena.

→ LR schedule changes order of learning patterns, which leads to generalization.

- Small learning rate quickly memorizes hard to fit class signatures (ignores patterns, harms generalization)
- Large initial learning rate learns easy to fit patterns first: only memorizes hard to fit patterns after annealing. Learns to use all patterns, helps generalization!

Experiment on modified CIFAR10:

- Group 1: 20% examples with hard-to-generalize, easy to fit patterns (hard to generalize because of variations in image).
- Group 2: 20% examples with easy-to-generalize but hard-to-fit patterns (colored patches that identify the class).
- Group 3: 60% examples with both patterns.
- Finding:
  - Small LR: memorizes patch, ignores rest of image
  - Large LR: ignores patch, only learns after annealing.
- Also present proof of this trend in linear classification.

.....

1.1.6 Data-Dependent Sample Complexity for Deep NNs [53]

For those that missed the previous talk, this is also by Colin Wei! :)

Q: How do we design principled regularizers for deep models?

A1: Principled approach—prove upper bounds on generalization error, regularize the upper bounds empirically.

But: bottleneck in prior work. Mostly consider the norms of weight matrices, which lead to pessimistic bounds that are exponential in the net depth.

This Work: Data-Dependent Generalization Bounds. Add $g(\text{weights}, \text{training data})$ as an explicit regularizer, which shows up in generalization bounds.

Theorem 2. *(informal)*

$$g(\cdot) = \frac{\textit{jacobian norm of model w.r.t hidden layers} \times \textit{hidden layer norm}}{\textit{margin}\sqrt{\textit{train set size}}}. \tag{3}$$

Roughly measures stability/Lipschitzness of the network around training examples.

Final ingredient to derive the regularizer: penalize squared Jacobian norm in loss. Hidden layer controlled by normalization layers.

Conclusion:

- Tighter bounds by considering data-dependent properties
- Bounds avoid exponential dependence on depth.
- Optimizing this bounds improves empirical performance.

.....

1.2 Test of Time: Dual Averaging Method for Stochastic Learning and Online Optimization [55]

Talk by Lin Xiao.

Acknowledge influence from Nesterov: this work was inspired by/extension of Nesterov [44].

Exciting time to be in ML/AI! It was exciting in 2009 as well. There was lots of excitement about two topics, the conflict of which led to this paper. They were: 1) Stochastic Gradient Descent and 2) Online convex optimization..

SGD in stochastic optimization:

$$\min_w \mathbb{E}_z[f(w,z)], \tag{4}$$

or in empirical risk minimization:

$$\min_w \frac{1}{n}\sum_{i=1}^{n} f(w,z_i), \tag{5}$$

Simply means doing gradient based updates.

$\rightarrow$ Will also need some basic convergence theory:

1. $O(1/\sqrt{t})$ rate if $f(\cdot,z)$ convex and $\alpha_t \sim t/\sqrt{t}$
2. $O(1/t)$ if $f(\cdot,z)$ strongly convex and $\alpha_t \sim 1/t$.

In convex optimization:

- Input is a convex set $\mathcal{S}$

- Then, for each round, simply:
  1. Predict a vector $w_t \in \mathcal{S}$
  2. Receive a convex loss function f_t, suffer loss $f_t(w_t)$
- Goal is to minimize regret:

$$R_T = \sum_{t=1}^{T} f_t(w_t) - \min_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w).$$

- $R_t = O(\sqrt{T})$ for convex, $O(\ln(T))$ for strongly convex.
- See more by Duchi et al. [18].

Another exciting area from 2009: compressed sensing/sparse optimization.

→ Lasso [49]!

$$\min_w \frac{1}{2}||Xw - y||_2^2, \qquad \text{subject to} ||w||_1 \leq \delta. \tag{6}$$

Lots of theory and algorithms from the 2000s (interior point methods, proximal gradient methods).

Proximal gradient method:

- Compositve convex optimization:

$$\min_w f(w) + \psi(w),$$

  with f convex and smooth, ψ convex and simple.
- Subgradient method;

$$w_{t+1} = w_t - \alpha_t(\nabla f(w_t) + \chi_t).$$

- Proximal gradient method:

$$w_{t+1} = \arg\min_w \left\{ f(w_t) + \langle \nabla f(w_t), w - w_t \rangle + \psi(w) + \frac{1}{2\alpha}||w - w_t||^2 \right\},$$

  with a constant α, achieves $O(1/t)$ convergence.
  → Equivalent form: forward-backward splitting.
- Soft-thresholding, where $\psi(w) = \lambda||w||_1$:

$$w_{t+1}^{(} i) = \text{shrink}\left(w_{t+1/2}^{(i)}, \alpha\lambda\right) \tag{7}$$

$$\text{shrink}\,(\omega, \alpha\lambda) - \begin{cases} \omega - \alpha\lambda & \omega > \alpha\lambda \\ 0 & |\omega| \leq \alpha\lambda \\ \omega + \alpha\lambda & \omega < -\alpha\lambda \end{cases} \tag{8}$$

Motivation for this paper: *how do we put together all three of these techniques?*

→ SGD/OCO meets sparse optimization!

Stochastic subgradient method:

$$w_{t+1} = w_t - \alpha_t(g_t + \chi_t)$$

where $g_t = \nabla f(w_t, z_t)$, yields a slow convergence rate of $O(1/\sqrt{t})$.

Q: But what about sparsity?

A: Sure! Extend proximal gradient method in the stochastic setting, where we replace the gradient with a subgradient.

The Algorithm: Regularized Dual Averaging (RDA) makes two changes: 1) Replace the single linear approximation with the average of all linear approximations, and 2) quadratic penalty becomes smaller, allowing for larger step sizes.

$$w_{t+1} = \arg\min_w \left\{ \langle \bar{g}_t, w \rangle + \psi(w) + \frac{\gamma}{\sqrt{t}} \frac{||w - w_0||_2^2}{2} \right\}, \tag{9}$$

where $\bar{g}_t = \frac{1}{t} \sum_{\tau=1}^{t} g_\tau$.

Can replace γ with some parameter β_t to penalize depending on aspects of the learning problem. Thus, this is why this is an extension of Nesterov's DA method: $\phi(w) = 0$ or indicator of convex set.

→ Standard theory here: same regret/convergence rate.

Experiments on MNIST:

- Binary classification on MNIST.
- Explore impact of the regularization/penalty term on performance.
- Compare SGD, proximal methods, RDA.

Further results and extensions:

- Later interpretation of RDA from McMahan [39] that shows some equivalence/relationship to Proximal SGD, FTRL-Proximal.
- Further developments: manifold identication [37], analyze the general and strong convex case.
- Other extensions: accelerated version of RDA, RDA-ADMM, distributed, and so on.
- Adaptive RDA (ADA-RDA) [18]: an adaptive version (variant of AdaGrad) of RDA that is well suited to sparsity.

Q: What about nonconvex optimization?

A: Very active in recent years due to deep learning.

→ Proximal SGD works with ℓ_1 regularization.

Q: Would variants of RDA work for nonconvex optimization?

→ Promising for sparse CNN [33]. Provable convergence with constant step size.

Final reflections:

- Motivation for RDA remains valid today:
  1. we know stochastic gradient online optimization is a mainstay of ML.
  2. But, structured sparsity is essential in scaling, too!
- Additional challenges today: non-convexity!
  → Also more complex and structured models today as well.
- Exciting progress lies ahead.

.....

1.3 Session: Reinforcement Learning

Next up, RL!

1.3.1 Causal Confusion in Imitation Learning [15]

Talk by Pim de Haan.

Imitation learning: powerful mechanism for learning complex behaviors! See: flying drones, grasping fish.

→ Simplest version is behavioral cloning, reduces imitation learning to supervised learning. Just learn to clone the expert exactly. Not perfect, though, since errors from the expert accumulate as we move to states outside the regular state distribution.

Q: Does more information mean better performance?

A: Let's answer via example. Consider learning to drive where an agent learns to drive and sees the first-person view of driving a car. But, also including a "break light" can lead to confusion! This is causal confusion.

That is: consider two systems; 1) predicts expert action from road view and brake indicator, and 2) predicts expert action from only road view. Turns out the 2nd can be better by avoiding confusion!

Control **experiments:**

- Add nuisance information to the state to create a confounded state representation.
- Apply this transformation to Atari, Mountain Car, and MuJoCo—specifically this means drawing (visually) information about the past action onto the screen.

  → Nuisance is roughly a Pong screen with a big number in the bottom left indicating which action was last taken (and so on for other environments).
- To solve causal confusion, we need:
  - Causal graph of expert
  - Neural net predictor of action given causes
- Phase one: model causal graph as binary vector
- Training: 1) randomly sample a causal graph, 2) then mask out nuisance part of the state, 3) feed this into an NN to predict action, 4) NN is fit to a behavioral cloning loss to learn a policy.

Q: How do we infer the right causal graph?

- Expert demonstrations are insufficient, intervention is necessary
- Intervene by executing policies
- Score graphs based on additional info.
- Two Modes of intervention:
  1. Targeted intervention by expert queries.
  2. Targeted intervention by policy execution, given access to some rewards in the environment. Graphs with high rewards become more likely.
- Give upper bounds on performance for both modes, results indicate strong learning performance.

  → Conclude from the data the algorithm infers a good causal graph.

Summary:

- Causal confusion happens often in imitation learning
- More information in the state can hurt performance by being confusing
- Solved in benchmark environments by few targeted interventions.

1.3.2 Imitation Learning from Observations by Minimizing Inverse Dynamics Disagreement

Talk by Chao Yang.

Definition 3 (Imitation learning): *Given an MDP without R and a set of expert demonstrations, find a policy that mimics expert behavior well.*

Q: What if we don't have *action* information in these expert demonstrations? So:

$$D = \{\tau_1, \ldots, \tau_m\} = \{(s_0, \neg a_{0,1}, \ldots)\}.$$

Taking the divergence minimization perspective seems feasible (as in GAIL), but generalize state to observation, yields GAiFO. Without actions, though, need anothher trick. So idea for the new loss function is:

$$\text{IDD} = \text{GAIL} - \text{GAIfO}.$$

Carry out quantitative comparison to other imitation learning approaches in OpenAI gym environments, achieve state of the art.

1.3.3 Learning to Control Self-Assembling Morpholgies [45]

Talk by Deepak Pathel.

Q: How do we train a robot?

A: Lots of options! Demonstrations, batch of data, from scratch/exploration, self-supervision, and so on.

→ But, doing learning from scratch can be challenging for *hardware* reasons. Hardware is designed for learned systems, typically, not to make learning to control easy.

Main Idea; Co-Evolution of control and morphology.

Example: A cylinder and a cone can be combined via a magnet to make more complex objects. Individual items are "dumb" and can't do much, but when they assemble they can create complicated structures. Thus it's about learning to combine (learn to stand up).

Q: How do we learn compositional controllers?

A: Each limb/component is a separate policy network with shared parameters. Treat these policies as Lego blocks. Let them pass messages to each other and accept messages from others.

→ This idea of message passing we call "Dynamic Graph Networks".

Experiments: 1) combine a large number of mini components (in 3d mujoco like environment) via magnets to get them to stand up, 2) also look at locomotion tasks involving climbing stairs and crossing trenches.

1.3.4 A Structured Prediction Approach for Generalization in Cooperative Multi-Agent RL [10]

Talk by Nicolas Carion.

Setting: lots of agents that must coordinate to solve a task. Each agent is given its own task.

Main contribution: Find policies that generalize to new agent-task pairs not seeing during training.

Idea: compute score on previous pairs, using structured inference, use this to infer new policy on new pairs.

Q: How can we increase collaboration across agents?

A: Add a new constraint to the optimization to make it a linear program. The constraint encourages agents to cooperate. Further extend the LP to a quadratic program.

Experiments in Starcraft: strategy learns to focus attack of an army on a handful units, and thereby seems to encourage cooperation. Also generalizes to new and larger armies.

Conclusion: A practical method for scaling MARL by generalizing to large instances from small training scenarios.

1.3.5 Learning Compositional neural Programs with Recursive Tree Search [10]

Talk by Thomas Pierrot.

Natural way to define rewards is *binary*: did the RL agent succeed or not?

→ But, of course, this makes exploration extremely hard, as the agent only gets feedback after a success.

New agent: AlphaNPI that uses modularity, hierarchy, and recursion as structural biases to reduce sample complexity, improve generalization, and increase interpretability.

Assumptions:

- Assumes a hierarchical program specification
- Sparse rewards: 1 when the program execution satisifes the spec, and 0 otherwise.

AlphaNPI is based on the Neural Program Interpreter: a recursive compositional neural network's policy and a value function estimator.

AlphaNPI works as a standard program execution in a computer: execution is saved into a stack, execution is passed back after a subprocedure finishes, and so on.

Experiments: AlphaNPI to learn libraries of programs to achieve sorting and searching tasks, seems to work!

1.3.6 Guided Meta-Policy Search [40]

Talk by Russell Mendonca.

RL has high sample complexity: typical benchmark tasks with SOTA approaches takes on the order of millions of samples required.

Meta-Learning is one possible solution to this problem by making efficent use of previous training tasks.

→ But, most meta-RL methods also requires huge amounts of data for the training phase, even if the adaptation is efficient. Also shortcomings in exploration and credit assignment during the training phase.

Introduce: Guided Meta-Policy Search, that can quickly adapt to solve any task from the distribution of training tasks.

- On each train task, learn a local policy using sample efficient methods.[1]
- Then, using a supervised meta-RL objective, form updated policies. This decouples the meta-RL problem, allowing for the use of off-policy learning.

Experimental evaluation in 1) simulated sawyer robot tasks, and 2) ant Locomotion tasks. Method achieves extremely sample efficient learning on these tasks.

→ Also study Meta-learning from demos.

Code: https://github.com/russellmendonca/GMPS

1.3.7 Using a Logarithmic Mapping to Enable a Lower Discount Factor [50]

Talk by Harm ven Seijen.

Q: How do we measure performance?

A: A few methods:

$$\text{finite horizon} = \sum_{i=1}^{H} R_i, \tag{10}$$

$$\text{infinite horizon} = \sum_{i=1}^{\infty} \gamma^{i-1} R_i, \tag{11}$$

[1] Dave: ?

for $\gamma \in [0, 1)$. But, most γs need to be near one to be useful/meaningful, which makes learning hard! How can we get around this?

Suppose we do linear function approximation with tile coding on a long chain problem

- For high discount factors, performance is okay!
- For anything below a high discount factor, however, performance is dreadful. Even with more data, even with tuning the tile coding, performance is really bad.

Q: But why?

A: One idea is to look at the *action gap*.

Definition 4 (Action Gap): *The action gap of a state is:*

$$\Delta_A(s) := Q^*(s, a^*) - Q^*(s, \tilde{a}), \tag{12}$$

where $\tilde{a}$ is the best non-optimal action.

The action gap is useful because it tells us how small our error should be if we want to be optimal.

$\rightarrow$ Action gap exponentially decreases the further you get away from positive reward (in a long chain problem). Thus, with small discount factors, this gap will decrease much more rapidly.

Q: But why bad behavior for low discount factors?

A: Systematiclly disproved lots of hypotheses, but found hypothesis that seemed promising:
(Possible) Cause of bad behavior for low discount factors: *poor performance of low discount factors is caused by large size differences in the action gap across the state space.*

$\rightarrow$ Gives rise to logarithmic mapping of the discount factor. So: $\tilde{Q} \mapsto \ln(Q)$.

New Algorithm: (special case with $R \in \mathbb{R})_{\geq 0}$ and deterministic T). Logarithmic Q-Learning. Performs Q update where:

$$\tilde{Q}_{t+1}(s_t, a_t) := (1 - \alpha)\tilde{Q}_t(s_t, a_t) + \alpha f(r_t + \gamma \max_{a'} f^{-1}(Q_t(s_{t+1}, a')). \tag{13}$$

Issues in extending to Stochastic Environments. Becuase of Jensen's Inequality ($\mathbb{E}[\ln(X)] \leq \ln(\mathbb{E}[X])$, where X is some random variable, so it seems like previous update rule is applied to stochastic environment, converges to underestimate.

But! Can be fixed. New update involves two step sizes, one for the "regular" Q space, one for the logarithmic space.

Theorem 3. *This variant of logarithmic Q-learning preserves convergence guarantees of Q-Learning.*

Empirical performance is good, too: applied to tabular MDPs and propose logDQN; on average it improves over DQN.

→ Underperforms relative to DQN on a few games where the action gaps are too homogeneous.

1.3.8 Better Exploration with Optimistic Actor Critic [13]

Talk by Kamil Ciosek.

Greedy is bad!

But, policy gradients are greedy. Also: use a lower bound on the critic. If the bootstrap is too large, then the value estimate becomes too large, which is amplified by policy optimization.

→ Conservative update reduces overestimation (as in Double Q).

Note: combination of greediness + lower bound can lead to pessimistic exploration. Variance is small, don't explore enough.

This Paper: Use the bootstrap to make an upper bound to achieve optimistic exploration.

→ UCB style exploration bonus.

Want a policy that: 1) is close to target policy, and 2) maximizes the critic upper bound.

Super easy to implement this form of optimistic exploration: shift the exploration policy in the direction of the upper bound. Empirical results are very promising, improve over Soft Actor-Critic (SAC).

1.3.9 Robust Exploration in Linear Quadratic Reinforcement Learning

Talk by Hakan Hjalmarsson.

Goal: Do exploration that is 1) robust and 2) targeted.

- Robust: does not destabilize the system or cause failure.
- Targeted: provides knowledge that helps complete the task.

Handle the exploration-exploitation tradeoff by formulating policy search as a convex optimization problem which can be solved for the global optimum.

→ Focus on Linear Quadratic Control. Task is to find a state-feedback controller to minimize a quadratic cost.

Key quantity: the empirical covariance $D = \sum_{t=0}^{T} \begin{bmatrix} x_t \\ u_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top$.

Yields a simplified optimization problem. Preserves structure by working with this empirical covariance matrix D.

Empirical findings: performance of this method is better even though absolute uncertainty is larger.

1.3.10 Tight Regret bounds for Model-Based RL with Greedy Policies [19]

Talk by Yonathon Efroni.

One Line Summary: Same minimax bounds to model-based RL with short-term and long-term planning.

$\rightarrow$ Factor of $\mathcal{S}$ less computation.

Previous regret bounds;

- Model-Based RL: minimax regret $O(\sqrt{HSAT})$
- Model-Free RL: Q-Learning regret $O(\sqrt{H^3SAT})$

Q: So, why aren't we using model-based RL?

A1: Lots of space complexity.
A2: High computational complexity (requires MDP solving each episode).

This work; model-based RL with short-term planning is minimax optimal for finite-horizon MDPs.

Open question: if one-step planning is minimax optimal, why do we need to use lookahead policies?

1.3.11 Hindsight Credit Assignment [28]

Talk by Anna Harutyunyan.

Central Q of Credit Assignment: How did past actions influence future outcomes?

Usual Answer/algorithmic strategy: rely on MDP structure and use *time* as main proxy for credit relevance.

Example; Credit Assignmeint in RL (CARL) has to figure out what caused him to be wet during the day. As an RL algorithm, CARL would spend many days getting wet by not bringing an umbrella.

$\rightarrow$ This paper: try to learn credit relevance *explicitly*.

Idea: study $P(a \mid x, f(\tau))$, where a is a past action and $f(\tau)$ is some relevant future outcome (like CARL getting rained on).

Hindsight Credit Assignment is about rewriting our usual value function:

$$Q^{\pi}(x,a) = r(x,a) + \mathbb{E}_{\tau \sim T(x,\pi)} \left[\sum_{k \geq 1} \gamma^k \frac{P(a \mid x, f(\tau))}{f(\tau)} V^{\pi}(x') \right]. \tag{14}$$

Whole new family of algorithms dedicated to learning the hindsight distribution.

1.3.12 Weight Agnostic Neural Networks [24]

Talk by Adam Gaier.

Central Q of this work: To what extent can neural net architectures alone encode solutions to tasks?

→ A different kind of neural architecture search.

- Search for networks that perform without training! Do zero shot learning
- But, add single shared weight value used for all connections.

Four key steps

1. Initialize: create pop of minimal nets.
2. Evaluate: test with range of shared weight values.
3. Rank: by performance/complexity.
4. Vary networks.

Test WANNs on a variety of RL tasks (cart pole, bipedal walker, : find that WANNS perform well with and without training, but perhaps more importantly can do so with significantly smaller neural nets.

For fun: tried on MNIST as well using the same rough scheme. Random weight finds 82% accuracy, tuned gets around 92%.

Point: steps toward a new kind of neural architecture search.
Code and more at weightagnostic.github.io.

2 Wednesday December 11th: Main Conference

Day two!

2.1 Session: Reinforcement Learning

Mostly in the RL sessions today.

2.1.1 A Neurally Plausible Model Learns Successor Representations in Partially Observable Environments [51]

Talk by Eszter Vertes.

Q: How should we represent belief states?

A: Distributed distributional code (DDC), and successor features (SFs)!

This Paper: How to elegantly combine DDC and SFs to form distributional successor features, how to learn them, and connections to neuroscience.

Definition 5 (Successor representation): *The successor representation is given by:*

$$M^{\pi}(s_i, s_j) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k \mathbb{1}\{s_{t+k} = s_j\} \mid s_t = s_i\right].$$

Q: How do we learn distributional successor features?

A: Two problems: 1) update belief states μ, and 2) Learn/compute distributional SFs $M^{\pi}(\mu)$.

To learn the distributional SF: three methods

1. Online learning: TD Learning using DDC believe states
2. Offline learning: TD learning using simulations in latent space
3. Direct computation: using recurrent network dynamics.

Contributions:

- Generalize SFs to POMDPs by exploiting the relationship between DDC belief states and SFs
- Propose learning algorithms that span the space of model-free to model-based algorithms with biological plausibility.
- Highlight further connections to neuroscience data:

– Propose a novel role of hippocampal replay for learning to infer spatial location
– Connects threads in dopamine signals and sensory prediction errors.

.....

2.1.2 DualDICE: Behavior-Agnostic Estimation of Discounted Stationary Distribution Corrections [41]

Talk by Ofir Nachum.

Central Q of the paper: Off-policy estimation ("if I run this policy, what is the average discounted reward I will get?").

→ Off-policy makes it tricky! Only get data from some other policy, may not even know which policy was followed.

Main Ideas:

1. Reduce the off-policy estimation problem to density ratio estimation:

$$d^{\pi}(s,a) := (1-\gamma)\sum_{t=0}$$

   - Using importance weighting trick with a finite dataset, can compute the weighted average of the importance weights.
   - Thus: reduces to the density ratio problem.

2. DualDICE operator is a zero-reward Bellman operator:

$$B_{\pi}v(s,a) := \gamma\mathbb{E}_{s'\sim T, a\sim\pi}[v(s',a)]$$

The DualDICE operator then leads to a new error term that consist of minimizing squared Bellman error while also maximizing initial ν-values. Desirable because objective is based on expectations from things we have access to.

.....

2.1.3 VIREL: A Variational Inference Framework for RL [21]

Talk by Matthew Fellows.

Goal; cast RL as a problem of probaltistic inference.

$But : existingmethodspresentseveraltheoreticalandpracticalbarriers.$

Two classical approaches:

1. Pseudo-likelihood methods, which promote risk-seeking behavior

2. Maximum entropy RL objective (Soft Actor-Critic), but optimal deterministic policies are not learned, and counterexamples show several cases when optimal RL policy can't be recovered (new result in this paper).

Thus, new desiderata for an inference framework:

1. Naturally learns optimal deterministic policies
2. Temperature not a hyperparameter.
3. Function approx explicitly used.
4. Stochastic policies used for learning.
5. Discounting easily incorporated.
6. Optimizes the reverse form of KL divergence.

$\implies$ Main result from the work is to outline the VIREL framework, which they show satisfies these desiderata.

Yields new class of Actor-Critic algorithms based on Expectation-Maximization. Evaluate on MuJoCo environments and find gains in sample efficiency.

.....

2.1.4 Unsupervised Curriculua for Visual Meta RL [30]

Talk by Kyle Hsu.

Objective: move RL from highly specialized learning to more general learning.

$\rightarrow$ Setting is Meta-RL, in which the policy must also learn to identify the task it's in.

Key Component for Meta-RL; Task distribution chosen for training in Meta-RL.

Q: Can we learn useful Meta-RL strategies by learning from tasks that care chosen without supervision?
A: Yes! This paper: meta-pre-training. Do unserupvised pre-training, then train.

Prior work: do task acquisition, then do meta-learning based on the chosen tasks.

This Paper; Do these two things jointly to provide an automatic curriculum.

Criteria for task distribution; 1) Diverse tasks, and 2) Structure in tasks. Need to trade-off between these two! Use mutual information maximization:

$$\max_{\theta,\phi} I(\tau; z),$$

Via an EM like approach: (E-step) updates task distribution, (M-step) do meta-training by updating policy.

Q: What kinds of tasks are discovered by the procedure?

A: Experiments in VizDoom, visualize different generated tasks.

.....

2.1.5 Policy Continuation with Hindsight Inverse Dynamics [47]

Talk by Hao Sun.

Focus: goal-oriented reward sparse tasks.

→ Purely random exploration is clearly impractical due to the sparsity of the reward. But: people learn from failures, not just success! How can we leverage this idea?

Two ideas;

- One approach: hindsight experience replay (HER) [2], which explicitly tries to learn from failures.
- Extrapolating from success! How can we generalize from cases we were successful?

New Idea: equip inverse dynamics with hindsight (HID). But, 1-step HID is not enough, so use multi-step policy continuation to achieve multi-step optimality.

Experiments on simulated robot domain and grid navigation.

.....

2.1.6 Learning Reward Machines for Partially Observable RL [29]

Talk by Rodrigo Icarte.

Main Q: How can we learn a reward machine from experience?

Reward Machines (RMs) are automate based reward functions, extremely useful for specifying complex behaviors.

This paper:

- Show how to learn RMs from experience (new algorithm: LRM).
- Use RMs as memory for partially observable RL
- Extends QRMs to work under partial observability

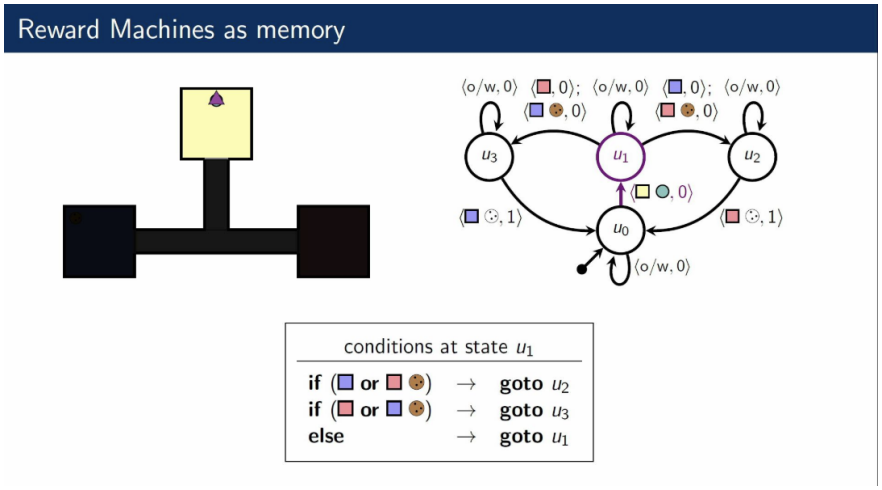


Figure 1: Reward Machines as memory

- Provide theoretical and empirical analysis of LRM.

Introduce RM-DQN, a DQN variant that learns and uses a reward machine.

Overall approach:

- Collect traces of experience
- Do optimization from this experience to propose a candidate RM
- Use the candidate RM to train an RL agent
- Add extra traces from new RL agent if the RM isn't perfect.

.....

2.2 Keynote: Yoshua Bengio on System 1 and System 2 in Deep Learning

State of deep learning: amazing progress this century!

Q: is it enough to grow datasets, model sizes. or computer speed?

Q: Are wwe still far from human-level AI?

Some Answers: Narrow AI, sample efficiency, human-provided labels, robustness, stupid errors. Next step could be completely different from deep learning.

Focus Today: There is a path from deep learning today to system 2!

Inspiration: "Thinking Fast and Slow" by Daniel Kahneman.

Definition 6 (System 1): *Intuitive, fast unconscious non-linguistic, and habitual decision making/inference.*

Note: deep learning is good at this (system 1)!

Definition 7 (System 2): *Slow, logical, sequential, conscious, linguistic, algorithmic,decision making.*

$\Longrightarrow$ Claim: the future of deep learning is extending what works in system 1 to system 2.

Some things that are missing from deep learning right now:

1. Out of distribution generalization and transfer
2. High level cognition: need to move from system 1 to system 2.
   - High level semantic representation.
   - Compositionality.
   - Causality.
3. Agent perspective:
   - Machines that understand the world by building world models
   - Knowledge-seeking

Theme Today: These three areas are intimately connected, and these connections can help us identify a path forward in AI/ML research.

The talk:

1. ML Goal: Handle Changes in Distribution
2. System 2 Basics: Attention and Consciousness
3. Consciousness Prior: Sparse Factor Graph
4. Theoretical framework: meta-learning, localized change hypothesiscausal discovery
5. Compositional DL architectures

.....

2.2.1 ML Goal: Handle Changes in Distribution

Classical ML theory is *for i.i.d. data*. Without this assumption we can't say anything about generalization.

→ Naturally, this assumption is unrealistic.

But: *"Nature does not shuffle the data, we shouldn't"* – Leon Bottou (ICML 2019 Keynote).

Out of distribution generalization and transfer breaks the i.i.d. hypothesis, so what do we do? Can we replace it with something else?

A: Agent learning needs out-of-distribution (OOD) generalization.

Why? Well: Agents face non-stationarities resulting from 1) their actions, 2) actions of other agents, 3) different places, times, sensors, and so on.

Claim: compositionality helps OOD generalization.

→ Different forms of compositionality, each comes ith different exponential advantages. See, for instance:

- Distributed representations
- Composition of layers in deep nets DAVEEITE: Montufar NeurIPS 2014
- Systematic generalization in language, analogies, abstract reasoning.

Q: So, how do we do systematic generalization?

A: Important to dynamically recombing existing concepts, *even when new combinations have 0 probability under training distributions*. Think of a science fiction scenario! We don't live that, but can still imagine it and draw meaningful insights from it.

→ Current methods are not effective for this.

Q: How does what you are proposing contrast with the symbolic AI program?

A: Avoids pitfalls of classical AI rule-based symbol manipulation. That is, we need the following:

- Need efficient large-scale learning
- Need semantic grounding in system 1
- Need distributed representations for generalization
- Need efficient (trained) search for both systems
- Need systems that can properly handle uncertainty in the world.

But, we also want:

- Systematic generalization
- Factorizing knowledge in small exchangeable pieces
- manipulating variables, instances, references, and indirection.

.....

2.2.2 System 2 Basics: Attention and Consciousness

Claim: Key ingredient for consciousness is *attention*.

Q: What is attention?

A: Focus on one or a few elements at a time. Content-based soft attention is convenient because we can backprop to learn where to attend. Lots of benefits:

1. Neural Machine Translation revolution [6].
2. SOTA in NLP, self attention/transformers
3. Memory-extended neural nets
4. Address vanishing gradients
5. Operating on unordered sets.

→ Attention creates a *dynamic* connection between two layers of a neural net.

From Attention to Consciousness:

- "C"-word is not taboo anymore in cognitive neuroscience
- One main theory: "Global Workspace Theory" [5].
  - Bottleneck of conscious processing
  - Selected item is broadcast, stored in short-term memory, conditions perception and action
  - System 2 like sequential processing, conscious reasoning and planning and imagination.

Q: So how can we connect these ideas to machine learning? (can the two communities help each other?)

A: ML can help us formalize, in a mechanistic way, and test specific hypothesized functionalities of consciousness. This could: 1) get the magic out of consciousness, 2) understand evolutionary advantage of consciousness by grounding it to computation and statistics.

Concsciousness closely related to language: we assign consciousness from humans reporting, and moreover, high level representations are deeply connected to language.

→ Thus, a connection between System 1 and System 2: meaning anchored in low-level perception and action.

So need *grounded language learning*, such as BabyAI [12].

.....

2.2.3 Consciousness Prior: Sparse Factor Graph

Q: So what kinds of priors and structures can we use to get our learning algorithms to use these kinds of abilities?

Proposal one: the consciousness prior [7].

- Attention: to form conscious state and thought
- A thought is a low-dimensional object, few selected aspects of the unconscious state
- Need 2 high level states: 1) large unconscious state, and 2) tiny conscious state.
- Part of inference mechanism with respct to joint distribution of high-level variables.

Concretely, the consciousness prior proposes a *sparse factor graph*, where nodes are variables, and edges represent relations between these variables.

Why is this a reasonable hypothesis?

- Property og high-level variables which we manipulate with language; we can predict *some* given very few others.

  For instance: "if I drop the ball, it will fall on the ground"
- Disentangled factors are not marginally independent (ball, hand, etc)
- Prior: sparse factor graph imposes a joint distribution between high-level variables.
- Note: this prior does not work in the space of pixels.

.....

2.2.4 Theoretical framework: meta-learning, localized change hypothesis → causal discovery

Meta-Learning (or "learning to learn"): having multiple time scales of learning.

- Bi-level optimization:
  - Inner Loop of learning that outputs a loss

  - Outer loop: continual learning that optimizes the loss from the inner loop
- Examples: evolution & individual learning, Lifelong learning and adaptation to new environments.

Q: what causes changes in distribution?

Hypothesis to replace i.i.d. assumption: the "localized change hypothesis":

> **Definition 8** (Localized Change Hypothesis): *"changes are consequences of an intervention on a few causes or mechanisms." [46]*

Then: use OOD generalization as a training signal for factorizing knowledge [8].

- A meta-transfer objective for learning to disentangle causal mechanisms
- Learning whether A causes B or vice-versa
- Learning to disentangle (A,B) from (X,Y)
- Exploit changes in distribution and speed of adaptation to guess causal direction.

Newer work: Learning neural causal models from unknown interventions [34].

→ Learning small causal graphs to avoid exponential explosion of number of graphs by parameterizing factorized distribution over graphs.

Note: attacking this problem in a very deep learning friendly way. Can plug into the deep learning tools that work already.

.....

2.2.5 Compositional DL architectures

Recurrent Independent Mechanisms (RIMs) [26]: modularize computation and operate on sets of named and typed objects.

- multiple recurrent sparsely interacting modules, each with dynamics and object input-outputs
- Results: much better OOD generalization!
- Also RIMs as a replacement for LSTM in PPO, find improvements in Atari.

Summary: Hypotheses for Conscious Processing by Agents, Systematic Generalization:

- Sparse factor graph in space of high-level semantic variables
- Semantic variables are causal: agents, intentions, controllable objects
- Shared rules across instance tuples

- Distributional changes from localized causal interventions.

  → Changes are mostly localized *if you represent things in the right space*

- Things that are preserved across changes in distribution are the fundamental stable aspects of the world. Meaning should target these aspects.

Conclusions:

- Time is ripe for ML/AI to explore consciousness
- Could bring new priors to help systematic and OOD generalization
- Could benefit cognitive neuroscience too
- Would allow us to expand deep learning from system 1 to system 2.

3 Thursday December 12th: Main Conference

Today I'm only attending one session and a keynote.

3.1 Session: Game Theory and Fairness

As with the other sessions, there will be one longer talk (20min) followed by 5 minute spotlights.

3.1.1 Optimizing Generalized Rate Metrics with Three Players [43]

Talk by Harikrishna Narasimhan.

Focus: solve constrained learning problems!. So:

$$\min_{\theta} P(\theta) \ s., t. P^j(\theta) \leq \varepsilon, \forall_{j \in \mathbb{N}}.$$

Example: fair hiring. We want to ensure that we minimize our F-measure, while ensuring equal precision (to be unbiased in hiring).

Main Q: How does one design learning algorithms to handle general performance metrics and constraints?

Problem setup:

$$\min_{\theta} \psi(R_1(\theta), \ldots, R_k(\theta)),$$

such that:

$$\psi^j(R_1(\theta), \ldots) \leq 0, \ j = 1, 2, \ldots$$

Where there are K prediction rates:

$$R_k(\theta) = \mathbb{E}_{(X,Y)}[I(Y \neq \text{sign}(f(X)))].$$

Prior methods:

1. Solution A: use surrogates!
    - Relax indicators with continuous surrogates
    - Relaxing constraints with surrogates may make the problem infeasible
    - Surrogates may output values outside the range of ψ
2. Solution B: Cost-weighted minimization oracle (breaks problem into simpler sub problems)

This Paper: General framework which recovers prior methods as special cases. Also use to develop practical algorithms with minimal use of surrogate and tight handling.

Heart of framework:

- Min-Max game formulation:

$$\min_{\theta} \psi^j(R_1(\theta), \ldots) \ldots$$

- Replace these functions with slack variables to decouple rates:

$$\min_{\theta,\chi} \psi(\chi_1, \ldots)$$

- Then induce a Lagrangian min-max problem

Three player viewpoint: 1) λ-player, linear, 2) χ-player, convex, 3) θ-player:

$$\min_{\theta,\chi} \max_{\lambda \geq 0} \psi(\ldots) - \sum_k \lambda_k \chi_k + \sum_k \lambda_k R_k(\theta)$$

Then, surrogate based algorithm:

1. χ-player: best response
2. λ-player: SGD with indicators
3. θ-player: SGD with surrogates: only works with last objective ($R_k(\theta)$).

Prove: near-optimal approach for original learning problem, and near-feasible for constraints. Also prove results related to finding mixed nash and coarse correlated equilibrium.

Big advantage: new algorithms with guarantees, can also handle non-convex functions of prediction rates.

Experiment: try to maximuze the F-measure subject to a constraint on the F-measure of a minority.

Open source library: https://github.com/google-research/tensorflow_constrained_optimization.

3.1.2 Modeling Conceptual Understanding in Image Reference Games

Talk by Rodolfo Corona.

New game: image-reference game played by pairs of agents, where the listener sometimes can't distinguish aspects of the image. So:

- Speaker (agent 1) must described image to listener
- Description must be a single attribute
- But, sometimes the listener can't perceive aspects of the object (colorblind, for instance)

  → Speaker then plays sequences of games with listeners that can't perceive attributes of objects and must *learn* to adapt to different listeners.

Experiments: take a meta-learning approach! Explore in practice games to get to know the listener, before having to adapt.

→ Compare against a random practice game baseline, speaking baseline in contrast. Meta-learning exploration approach works quit well.

→ Performed ablation study and found that it was *crucial* that the speaker explicitly model aspects of the listener.

Strong performance in adapting to new speakers on the CalTech birds dataset (real images!).

3.1.3 This Looks Like That: Deep Learning for Interpretable Image Recognition [11]

Talk by Oscar Li.

Q: Look at a bird! What kind of bird is it? More generally: how do we answer this question?

A: We tend to look for specific features that are indicative of a class.

Proposal: Draw inspiration from how people identify classes to form a new kind of interpretability with richer explanations.

→ New model ProtoPNet: provides a new kind of interpretability with highlighting in an image, but also adds an explanation as to why those pieces where chosen.

Training algorithm:

1. SGD of layers before last layer
2. Projection of prototypes
3. Convex optimization of last layer.

Experiments: Achieve near state of the art performance on benchmark dataset, and carry out analysis of latent space.

3.1.4 Assessing Social and Intersectional Biases in Word Representations [48]

Two main Questions: 1) Do embedding association tests demonstrate social bias on contextual word encodings in the test sentence? 2) Can we develop comprehensive test for gender, race, and intersectional identities?

→ Focus on *contextual* word encoding, since they tend to improve downstream NLP performance, and do not have the problem of obfuscation.

Example Q: How related is concept X with attribute A, and concept Y, with attribute B? As opposed to X with B, and Y with A? (example concepts: names, attributes: stereotypically male/female occupation).

Study two concepts: gender, race, and a bunch of relevant attributes like occupation, likability, and so on.

→ Main contribution: new approach for finding intersectional biases in embeddings.

3.1.5 Paradoxes in Fair Machine Learning [25]

Talk by Anson Kahng.

Q: What is the relationship between fairness in machine learning and fairness in fair division?

Fairness in ML: statistical notions of fairness.
Fairness in FD: axioms of fair division (resource monotinicity).

Setting: classification with cardinality constraints. So, a classification problem with a fixed budget of available resource to distribute, e.g. financial aid.

→ Find a classifier that maximizes efficiency. But, two views on fairness: classification and fair division.

Fair Division Axioms: resource monotonicity (can't make people worse off with more resources), population monotonicity (if someone leaves the pool, it can't hurt).

→ The axioms are roughly in place to preclude paradoxes.
New Q, then: How much does efficiency suffer if we must satisfy these axioms, too?

Definition 9 (Equalized Odds): *A predictor $\hat{Y}$ satisfies equal odds w.r.t protected attribute on Y if $\hat{Y}$ and A are independent conditional on Y*

Results:

1. In the cardinality constrained model, we characterize optimal allocation rule that satisfies equalized odds.
2. Equalized odds and resource monotonicity are achievable with no loss to optimal equalized odds efficiency.
3. Any rule that satisfies equalized odds and population monotonicity can't achieve a constant factor approximation to optimal equalized odds efficiency.

.....

3.2 Keynote: Jeffrey Heer on Agency and Automation

Some of us are weery about the current rhetoric about AI, in part due to hype, but also because it is so focused on *pure autonomy*, rather than a harmonious combination of people with machines.

20 years ago: work by Eric Horvitz on "Principles of Mixed-Initiative User Interfaces". Studies: 1) support uncertainty around user goals, 2) allow efficient direct invocation and termination, and 3) continue to learn by observing user actions.

→ lots of work since then, too!

60 years ago: Bar-Hillel said *"Fully automatic high quality translation is not a reasonable goal, not even for scientific texts. A human translator in order to arrive at high quality output, is often obliged to make intelligent use of extra-linguistic knowledge which sometimes has to be of considerable breadth and depth."*

Long standing design challenge: determine regions of optimality in possible divisions of labor among directed and autonomous actions.

Fundamental Q: what is the appropriate balance between automation and user control?

Many challenges;

- Automated mehtods may be biased or inaccurate
- Consequences of poor models let loose in the world
- Loss of critical engagement and domain expertise
- Ambiguity of human intent, cognitive biases, mistakes
- Often lack a global view.

Proposed Strategy: Shared representations that enhance user interfaces with models of capabilities, actions, and goals to reason about principled human-AI collaboration.

Three examples:

1. Data cleaning and data visualization.
2. Data cleaning and transformation.
3. Natural language translation

3.2.1 Data Cleaning and Data Visualization

Let's think about *topic models* like Latent Dirichilet Allocation (LDA). We don't just care about the output, we would prefer this have some relevant structure such as text-order and other relationships—these can help us digest, search through, and process the findings of the topic model.

→ One thing learned: users of these data/systems are quite savvy! Running the same LDA leads to different results, perhaps contrast the results of different models to gain further insight into the right kinds of representations.

Recent work: mapping machine-learned latent spaces into something visualizable that people can understand. This can let us *align* the learned latent space to something we can make sense of.

Broad Question: What makes a visualization *good*?

→ A quick experiment: how much bigger is the small shape than the larger shape?

- Two circles, one big one large, asked to guess the precise relative size of the two.
  → Really high variance in audience responses!
- Same task but with two bar charts.
  → Really low variance in responses.

But: the two ratios were identical, but people were much more concentrated around the right answer in the latter case.

→ Can turn to psychophysics to rank visual encodings quantitatively; people are good at determining position, length, slope, but not as effective with color, volume, and area.

Common Exploration Pitfalls: 1) Overlook data quality issues, 2) Fixating on specific relationships, 3) Plus many other biases.

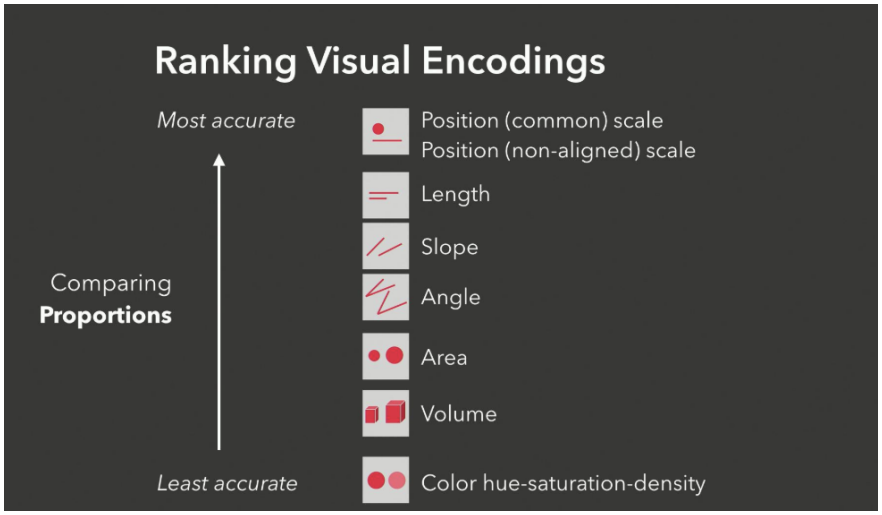


Figure 2: Different accuracy for people in estimating quantities of visual stimuli

Project: Visual exploratory tool for large datasets, use inference in the background to automatically recommend new visuals. Can also be taught how to create these plots by the tool itself.

Key Idea from Demo:

1. Augment manual exploration with visualization recommendations sensitive to the user's current focus.
2. Ultimate goal is to support systematic consideration of the data without exacerbating false discovery.

3. To model a user's search frontier, we optimize for related chart specifications, seeded current focus.

Representative quotes about the tool:

- Related view suggestion accelerates exploration a lot
- I like that it show what fields to include to see a specific graph
- The related views are so good but it's spoiling that I start thinking less. I'm not sure if that's a good thing

3.2.2 Data Cleaning and Transformation

Quote from Anonymous Data Scientist: *"I spend more than half my time integrating, cleansing, and transforming data without doing any actual analysis. Most of the time I'm lucky if I get to do any 'analysis' at all"*
→ Responses; that can't be true! I wish I only spent 50% of time on data cleaning.. (see big data borat on twitter).

New System: DataWrangler (2011). DataWrangler: tool for generating relational datasets based on input preferences of user! Taught the system to automate aspects of the data cleaning process, but also generates a *program* that does the data cleaning. We can extract this program and use it for other thing, too.

→ "Predictive Interaction": has an underlying domain specific language that can be used to "lift" the program to generate visualizations and interactions. These lifted components can be used to inform a *guide-decide* loop for end users.

Also ran user studies with this system:

- Reduces spec time, promotes use of reusable scalable transformations.
- Complementarity: suggestions good for tasks people found hard (extraction patterns, etc). People good in cases where inference is less tractable
- Agency: Users appreciated suggestions in respond to an initiating interaction, but did not always act on practice assistance.

→ Company called Trifacta uses this interaction system and the discovered paradigms to scale the software to production.

3.2.3 Natural Language Translation

Started the talk with 1960s quote from Bar-Hillel—around this time there was a clear vision for interactive translation and interfaces. Even a proposal for an interactive translation workflow.

→ Two big challenges, though: 1) Translation wasn't *good* enough to be a reliable collaborator, and 2) Getting these interface designs right is also challenges.

But! Revisited recently: predictive translation memory (PTM).
Again conducted user studies for interactive machine translation.

- Experiments with professional translators across language pairs (Arabic, French, German to English) and text genres (software, medical, news)
- Results:
  - Post-editing of full translation leads to reduced time and improved quality (over manual translation)
  - Interactive translation with PTM slightly slower than post-editing but yields higher quality translations
  - **Similar concerns around agency:** "I feel less susceptible to be creative", and "distracts from my own original translation process by putting words in my head."

Design Challenge: Determine regions of optimality in possible divisions of labor among directions and autonomous agents.

Future Challenges:

1. *Design process, tools, and monitoring:* how might we support productive prototyping, development, deployment, using machine learning?
2. *Mapping machine-learned representations:* Can ML methods suggest structural task models? can people help test an constrain ML models?
3. *Evaluating Trade-Offs of Agency and Automation:* Must go beyond results of quality and productivity.

   → Locus of control, agency vs passive acceptance? Training and skill acquisition vs. de-skilled labor?

4 Friday December 13th: Workshops

Today I will be at the Meta-Learning workshop!

4.1 Workshop: Meta-Learning

First up is an invited talk.

4.1.1 Erin Grant on Meta-Learning as Hierarchical Modelling

Main Idea: Reformulate meta-learning as hierarchical modeling.

OpenAI released a chart that tracks the compute usage and identifies an inflection point in 2012 where compute went from a linear increase to an exponential increase.

→ Really different from the settings in which people solve problems. People can: 1) make sharp inferences given impoverished data, 2) adapt learning given computational constraints, 3) don't expereince lifetimes of data.

Lots of exciting questions here from both cognitive science perspective and machine learning.

Also: a major bridge between the two fields is *inductive bias*.

- Shape Bias: in category learning, by the age of 2 years, English children exhibit the bias that shapes are generalized more rapidly than other properties like color.

  → Cross linguistic differences attenuate the shape-bias.

- This is an example of a learned bias!

  Q: Can we formalize how such a bias could be acquired?

Yes! Work by Kemp et al. [35]. Model can reproduce shape bias on categorical learning task, both on known categories and novel categories.

→ Example of using ML tools to understand how biases arise in people.

Roadmap:

1. Main lcaim: recast meta-learning as hierarchical modeling
2. Q: Why meta-learning as hierarchical modeling?
3. Cast estimation/inference for parameters in hierarchical probabilistic models as a meta-learning framework for humans and machines
4. Connecting these domains (cog sci and ML) can be mutually informative.

Generic Recipe for Meta-Learning Algorithm:

- Sample a meta-train dataset:

$$D_j \sim \mathscr{D}_{\text{meta-train}}$$

- Learn task-specific parameters:
$$\hat{\phi} \leftarrow g_\theta(D_j)$$
- Task-specific predictions:
$$y_1, \ldots, y_n \leftarrow f_{\hat{\phi}}(D_j)$$
- Finally: do a global hyperparamater update to our main parameters
$$\theta \leftarrow \theta - \nabla_\theta \mathcal{L}(f_{\hat{\phi}}(D_j)).$$

Q: Can we make this procedure probabilistic?

A: Yes! New recipe, same general formula:

- Sample a meta-train dataset:
$$D_j \sim \mathscr{D}_{\text{meta-train}}$$
- *(Main Change)* Task-specific **distribution**:
$$p(\phi_j \mid D_j) \propto \int \prod_i p(y_i \mid \ldots)$$
- Then, updates involve updating:
  1. Task specific predictive distribution
  2. Hyperprior update for general learning.
$$p(\theta) \leftarrow p(\theta \mid D_i)$$

Problem! Probabilistic inference steps are now prohibitively expensive. Can we make this more efficient?

A: Yep: try hierarchical bayes. idea: don't maintain full uncertainty, use approximate empirical bayes to fit a point estimate via maximum likelihood:
$$\arg\max_\theta p(y_1, \ldots y_n \mid x_1, \ldots, x_j; \theta),$$
rather than modeling the full posterior. Also need to do mode estimation. Trying to solve:
$$\arg\max_\theta p(y \mid X, \hat{\phi}) p(\hat{\phi} \mid \theta).$$

Q: What is an appropriate way to represent the task parameters in a new domain? How do we formulate a local parameter prior?

A: Isotropic priors and early stopping might be able to help.

→ One approach that has this flavor is Model-Agnostic Meta Learning (MAML) [22].

Approximate Empirical Bayes: optimize marginal likelihood w.r.t. θ using a point estimate

$$\arg\max_{\theta} p(Y \mid X, \hat{phi}) p(\hat{\phi} \mid \theta),$$

subject to:

$$\hat{\phi} = \arg\max_{\phi} p(y \mid X\phi) p(\phi \mid \theta).$$

Similar to MAML:

$$\arg\min_{\theta} -L(X; \theta - \alpha \nabla_{\theta} L(X; \theta))$$

But: gradient descent not the only way to compute a point estimate for these parameters.

Q: Can this estimate of $\hat{\phi}$ still be a valid point estimate for hierarchical Bayes?

A: Yes! When viewwed as an amortization of the MAP estimation

$$\hat{\phi} = \arg\max_{\phi} p(y \mid X) p(X \mid \phi)$$

We can also go beyond point estimates; full empirical Bayes. Three approaches;

1. Laplace approximation: LLAMA [27]
2. Sampling based estimate: Gibbs sampling, combinations of sampling techniques with variational inference
3. Variational approximation: neural Statistication, Amortized Bayesian meta-learning.

→ Another key way to make progress: is the underlying graphical model (from hierarchical Bayes) useful in the kinds of tasks we care about?

Apply these ideas to *continual learning* in which images are modified over time. Three phases: 1) image blur, 2) night mode, 3) pencil drawing filter.

Meta-test responsibility over time can identify when change points in the data occur.

→ Also investigate catastrophic forgetting as change point detection in this continual learning setting.

Conclusion:

- We can cast meta-learning as hierarchical Bayes, which allows a number of new insights.
- Can unify our understanding of meta-learning from hierarchical bayes
- Can draw inspiration from the probabilistic inference toolbox.
- Hopefully will bring closer links between human and machine learning.

Challenge Question: Bayes-optimal meta-learning is intractable, but is still exists. How do meta-learning produced by our current meta-learning algos compare to the Bayes optimal strategy, both quantitatively in terms of regret of qualitatively?

Erin Grant: Great question! Naturally these optimal forms are so intractable that we don't often focus on them too much. But I do think there is a big gap between optimal behavior and what we are achieving now. This is probably especially striking in RL, where exploration from a Bayes optimal perspective can help us view uncertainty in an optimal way.

4.1.2 Jeff Clune on How Meta-Learning Could Help Us Accomplish our Grandest AI Ambitions

Two main points today:

1. Q: How might we produce capable AI?
   → AI-generating algorithms!
2. Exotic meta-learning approaches could help!

Fruitful to think about the long term objective of the field. Yields some paths:

- Manual Path to AI!
  → Lots of building blocks: convolution, hierarchy, abstraction, (hundreds!)
- Once we identify these building blocks, we will need to combine building blocks into a giant Rube-Goldbergian complex thinking machine.
  → Doesn't fit our scientific culture, doesn't lead to effective debugging and optimization.
  → Might not even be possible.
- **New Path:** Hand designed pipelines are ultimately outperformed by learned solutions: powerful path is *an all in bet on AI-generating algorithms.*

Main Idea: Simple initial conditions that can be bootsrapped repeatedly to yield better and more capable algorithms. Darwinian evolution is one existence proof. Progress comes from three pillars:

1. Meta learn architectures
2. Meta learn algorithms
3. Generate effective learning environments

This talk is on work in these three pillars.

(Pillar 1): Meta-learning architectures, as in architecture search.

- Architectures matter
- Hard to design! So, let's search for them.

- Common approach: train on real data for moderate number of steps.

  → Can be sped up by only doing a few steps of SGD, then use that to inform architecture design.

Lots of work that shows if we carefully select data we train on, we can speed data up a lot.

Q: Can we meta-learn to generate data that enables rapid learning?
A: Generative Teaching Networks—generates synthetic data, which a new network will train on for small number of steps. Then train new network on actual data, differentiate on the whole pipeline including the generator.

→ Empirical Finding: this works on MNIST and CIFAR10. Few-step accuracy is higher on synthetic GTN data than Real data.

GTN conclusion: can produce synthetic data that trains NNs faster than real data, enabling rapid estimates of an architectures performance. SOTA-competitve.

(Pillar 2): Meta-learn algorithms. Two major camps;

1. Meta-learn good weights and apply SGD: MAML ish
2. Meta-learn rNN, which creates its own learning algorithm.

Point: materials matter! Still have to decide the **materials** of the network. RNNs forced to do all lifetime learning with activations: 1) may be unstable, and 2) proposal store information in weights too.

New work: Differentiable Hebbian Learning. Idea:

- Can store info in weights in addition to activations.
- Train Hebbian learning via SGD
- Idea of Hebbian learning; "neurons that fire together, wire together"

$$w_{ij}^{t+1} = w_{ij}^{t} + \eta x - i^t x_j^t.$$

- Yields a recurrent Hebbian network.
  - Inner loop: network update with no SGD
  - Outer loop: differentiate through episode, update trainable parameters via SGD.

→ Empirical finding: near SOTA on omniglot, also applied to maze navigation.

More work on Differentiable Neuromodulated Plasticity.

- Hebbian learning is local: hard optimization problem!

- Better idea: turn learning on some weights, only in certain contexts.

  Example: If I am playing chess and I just wont, then I should increase learning in only chess playing parts of the network.

- Add a new filter to the Hebbian update to gate whether or not you update certain activations.

→ Empirical findings: performed similar experiments as in the Hebbian case.

New work: Learning to Continually Learn: specifically try to solve catastrophic forgetting.

- catastrophic forgetting is the "Achilles heel of machine learning"
- In sequential learning: cannibalize how to solve prior problems once new ones are introduced. We don't forget catastrophically.
- Must solve this problem to continually learn.

→ Many proposed approaches to solve this, but again we should use meta-learning. This work falls more into the MAML camp of meta-learning.

Meta-learning for continual, multi-task learning.

→ New work at NeurIPS this year: Online-aware Meta-Learning (OML) [32]. Idea:

- Meta-learns a representation network
- At inference time, freeze this representation network, and a new task learning network will learn on top of the representation.
- Gets a lot right!

  → Subject to the "tyranny" of SGD during learning: not optimized for continual learning.

New Proposal: allow for control over SGD via neuromodulation. This yields; A Neuromodulated Meta-Learning algorithm or "ANML". Two networks; 1) Neuromodulated network, and 2) Task network.

→ Empirical work on sequential variation of Omniglot, following OML.

ANML conclusions: OML and ANML show the promise of meta-learning a solution to catastrophic forgetting.

(Pillar 3): Generating the right environments.

Q: How do we get the right tasks?

Another variant: Can algorithms generate their own problems while learning to solve them?

→ Inspired by open-ended algorithms that endlessly innovate. Examples: 1) Natural evolution, and 2) Human culture. Both of these generate their own problems while they're trying to solve them.

New approach: POET. Periodically generates new learning environments that are in the "Goldilocks" zone (not too hard, not too easy).

→ Deploy in RL, generates environments and tries to solve them. Empirical finding: learns well on these problems, but also learns to find a better solution on the simple tasks after seeing the harder tasks.

POET conclusions: invents effective curricula, endlessly innovates, captures spirit of open-ended innovation engines, fuels meta-learning

Overall Conclusions:

- Let's take a step back: is the manual path to powerful AI the fastest?
- AI-Generating algorithms is an all in bet on meta-learning
- Three pillars: 1) meta-learn architectures, 2) meta-learn learning algos, 3) generate algorithms.

Challenge Question: When does neuroevolution outperform?

Jeff Clune: Naturally thee optimization algorithm is going to make a big difference. We often confuse evolution the optimization algorithm vs. algorithms inspired by evolution (such as POET). Final answer: if we do actually compare evolution to other optimizers, it doesn't require gradients, more stable, and leads to different behavior; different exploration profile, can escape local optima, care about population rather than individuals.

4.1.3 Spotlight: Meta-Learning Contextual Bandit Exploration

Talk by Arvin.

Q: Can we learn to explore in contextual bandits?

A: Yes! Meta-learning.

Definition 10 (Contextual Bandit): *Interactive learning setting constituting a bandit problem where each step comes with a context feature that describes something relevant about the current bandit.*

→ Allows for study of generalization and exploration.

Solution: At meta-training time, assume access to π^* that knows how to explore/exploit and use it to train.

Q: What happens when we don't have access to π^*?

A: Generalization via meta-features!

→ Compare approach in a variety of contextual bandit problems, find significant improvements in most settings.

Theoretical guarantees: no regret property of "Aggrevate" can be leveraged here too.

4.1.4 Spotlight: ES-MAML: Hessian Free Meta Learning

Talk by Wenbo Gao.

Objective: applying *evolutionary strategies* (ES) to MAML.

Key Q: Can we perform meta-learning in blackbox case?

A: Yes! Through ES methods which perform gradients on Gaussian smoothing of function.

Toy problem: four corner task. Agent gets reward signal if close to the corner of a continuous grid.

→ ES-MAML adaptation targets only 1 or 2 corners while policy gradient (PG) MAML must "circle around" all 4 corners.

→ Further compare ES-MAML vs. PG-MAML, find: 1) ES-MAML is more stable, and 2) allows the use of non-smooth adaptation methods, yields new algorithmic design choices.

4.1.5 Spotlight: Quantile Based Approach for Hyperparameter Transfer Learning

Setting: assume many possible hyperparameter (HP) evaluations, want to find the best for transfer.

→ Difficulties, though: 1) sacles of objectives may vary across tasks, 2) noise may not be Gaussian, and 3) many observations so it's hard to apply approximate Gaussian Processes.

New idea: Gaussian Copula transform:

- If only every y' was Gaussian, we could use GPs!
- So: apply change of variable $\psi = \phi^{-1} \circ F$
- Can prove this transform yields a centered normal distribution.

Idea: do transfer learning by estimating a transfer variable, then learn the parameters of a Gaussian distribution.

Two new hyperaparameter optimization strategies: 1) Thompson sampling, and 2) Gaussian Sampling.

→ Evaluated on several dataset and find large improvements to learning.

4.1.6 Spotlight: Meta-World: Benchmark for Meta-RL

Previously: in multi-task RL, use things like DMLab and Atari games for testing.

→ However: tasks are relatively disjoint.

→ In Meta-RL, mostly use tasks that are very similar to each other.

Goal: To enable the generalization ability of Meta-RL algorithms. We need a large diverse task set, which is what Meta-World is.

- 50 qualitatively different manipulation tasks using a simulated sawyer robot
- Parameter variability
- Unified observation and action space
- Structured, multi-component rewards for all tasks.
- Also include benchmark results for meta-RL algorithms.

Videos and code: http://meta-world.github.io/

4.1.7 Pieter Abbeel: Better Model-based RL through Meta-RL

Talk: Share work on how meta RL can help model based RL (MBRL). At the end: show how we can make meta-training more efficient.

Overview:

1. Learning to RL
2. Domain randomization
3. Better MBRL through Meta RL
4. Speeding up Meta RL training

Part I: Learning to RL

Q: How fast can we learn?

A: People can learn a game in around 15 minutes, while DDQN takes 1500 hours!

Main Q: How can we bridge this gap?

Ideas:

- Humans might play Atari for first time, but have done lots of similar things before.
- So, let's let RL systems learn on previous tasks, then generalize its knowledge to the new tasks.

Formulation: end-to-end optimization problem. Searching for an RL agent that can, on expectation, learn quickly:

$$\max_{\theta} \mathbb{E}_m \mathbb{E}_{\tau_M} \left[\sum_{k=1}^{K} R(\tau_m^k) \mid \text{RLAgent}_\theta \right].$$

The setting is then about finding these agents via meta-training. Generate training environments that enables agents to learn to do well on the test environment.

Q: How do we represent RLAgent_θ?

A: RNN! Since it's a generic computation architecture, it can learn programs/algorithms. Different weights in the RNN means different RL algorithm and prior. Different activations in the RNN means different current policy.

→ Objective: it's the same as a usual RL algorithm but instead of a new state, we get dropped into a new environment. So, pick your favorite RL algorithm to optimize the meta-objective.

Evaluation:

1. Multi-Armed Bandits: simple, focus only on exploration.

   → Can then compare what a meta-RL agent does compared to the asymptotically optimal (Gittins) approach.

2. Navigation in Mazes: agent gets first person view of a 3d environment, looking for a treasure in a maze.

   → Random exploration fails, but after meta-training, meta-RL agent learns an algorithm for systematically exploring the maze to find the treasure.

Part II: Domain Randomization.

Q: Why do we care about simulation?

A: Less dangerous, easy to get labels, faster/more scalable.

→ But! Of course it doesn't match the real world.

Follow up Q: how can we learn useful real world skills in simulation?

A: That's what domain randomization is for.

→ If the model sees enough simulated variation, the real world may look like just the next simulator.

Initial results: quad copter flight can train in simulation with domain randomization $\implies$ copter learns to fly in the real wworld.

→ Q: Can we just train on unrealistic simulators like ImageNet, but randomize a lot?

A: Yes, actually! Evaluate on real world data and the approach works quite well. Used for robotic grasping, too. Real world is able to pick up objects in the real world based on simulated training alone.

Hypothesis: Training on a diverse array of precedurally generated objects can produce comparable performance to training on realistic object meshes.

→ Even harder challenge: in hand robotic manipulation.

- By training on wider range of randomized simulators, can learn to manipulate a block
  → Super challenging because contact forces are highly complicated.
- Goal is to orient a cube with letters on it to a particular angle.
- Robot learns to do manipulate these blocks from simulation alone, also extension to rubix cube.

Part III: Better MBRL through Meta RL

Definition 11 (Model-Based RL): *Model-Based RL uses experience to learn a simulator that we can use to train our behavior.*

Canonical MBRL:

1. Colllect data under current policy
2. Improve learned simulator from all past data
3. Improve policy in learned simulator.

Problem! Standard overfitting in ML rears its head.

- Policy optimization tends to exploit regions where insufficient data is availablle ot train the model, leading to catastrophic failures.
- Model-bias: see Atkeson and Santamaria [4], or Deisenroth and Rasmussen [16].
- Idea: Just learn a better model?
  → Use the two tools we discussed: 1) Domain randomization, and 2) Meta policy optimization.

Canonical MBRL with Meta Policy Optimization (MB-MPO):

1. Collect data under current adaptive policies $\pi\theta_1, \ldots, \pi_{\theta_n}$.
2. Learn ensemble of K simulators from all past data
3. Meta-policy optimization over ensemble:

(a) New meta-policy π_θ

(b) Net adaptive policies $\pi\theta_1, \ldots, \pi_{\theta_n}$.

Experiemnts in Cheetah: MB-MPO compared to PPO. Learns in a much more sample efficient way. Achieves state of the art relative to model-free approaches in Mujoco 9Ant, HalfCheetah, Hopper, PR2, swimmer, Walker2D).

→ Compare wall-clock time as well of robotic grasing/lego block stacking → achieves very quick learning even relative to wall-clock by training **asynchronously** (on the order of 10 minutes).

Part IV: Speeding up Meta RL Training.

We do meta RL in part because we want to get away from the sample complexity of RL.

→ But! Meta training can be just as demanding in terms of sample complexity.

All we want is a fast RL algorithm. We can make the meta training faster by matching the outer policy to a policy we trained on that task already: now it's a supervised loss rather than RL. Requires a policy on a few specific tasks, but can be a lot more efficient than doing full meta RL.

Experiments in pushing and opening doors and locomotion tasks.

Challenge Question: Recently seems to be a push in MBRL. Techniques from representation learning seem promising to learn helpful representations for more efficient planning. Any intuition on how to choose suitable objectives for learning good representations for MBRL?

Pieter Abbeel: Learning better representations is a really important research direction. It's probably large under researched to find better architectures for Meta-RL. In principle, in meta learning memory should play a big role. Smarter architectures could play a big role in improving these results. In MBRL, if we think about where model learning could make a big difference, it's really about reasoning over long horizon. Lots of benchmarks that are sample efficient with MBRL are still short horizon. Big question: how can we get simulators that simulate over long horizons in a reliable way.

4.1.8 Panel Discussion: Erin Grant, Jeff Clune, Pieter Abbeel

Q: How should we design Meta-learning training task distributions?

PA: Big challenge in meta-learning! Recent work on MetaWorld contains 50 robotic maniuplation tasks. Good solution. Still seems to be on the lower side. Most important direction is to, in some unsupervised way, generate the right tasks. Use some very objective but light weight principle to generate the task distribution.

JC: Unsurprisingly I'll agree with Pieter. We ultimately want to automatically generate the tasks. Continue to fan out tasks. Unknown unknown problem: might be things we don't think about to include. For that, I advocate for the POET style that automatically generates environments.

Allow the search algorithm to choose which worlds to learn from. Opens the floodgates to learning a wide distribution of tasks.

EG: Can be turtles all the way down. Some way we have to provide inductive bias to our learning algorithm. What meta-learning does it remove constraints from the learning algorithm to instead apply constraints to the data set. Nevertheless we do have to specify some manner of bias to generate datasets. Needs to be a balance between things the agent can reasonably solve and things it can't. Perhaps instead of thinking about functional priors instead of smoothness/locality, we should be thinking about the kinds of invariances we might an agent to pick up on.

JC: Another thing—what is the curriculum that's going to get you to solve some harder task? The whole curriculum can't be full of hard tasks. One way to solve that is a distribution that winds you up in complexity. Learn the tasks that help you learn.

PA: Want to add one thing to Erin's point. Might not be that easy to generate or design environments. Before humans were ruling Earth, dinosaurs were around. They were in some sense in a very similar environment to us. But, there was something different that led to intelligence.

EG: Could we speculate about what it is about the environment that changed that enabled humans to be intelligent?

PA: Not sure! Maybe the dinosaurs developed their bodies rather than their brains.

Q: Is there a limit to how much we can learn via Meta RL? Will we hit no free lunch?

EG: Reasonable hypothesis that the world has structure. Wouldn't want to be a nihilist in that sense. Maybe it's too complex to characterize in a learning system, but there is some structure that can be picked up on so agents can learn to quickly solve a variety of tasks.

JC: No free lunch doesn't keep me up at night because the space of problems is so vast. We don't care about algorithms that get better at these white-noise TV problems. We can focus our algorithms on ones that are relevant to our world. What does keep me up at night is generating tasks that are grounded in the real world in the relevant sense.

Dave: Need to run to lunch so missing the rest of the panel. Also, my talk is up next! So, no notes.

4.1.9 Raia Hadsell on Scalable Meta-Learning

Note: Learning from i.i.d. data is strange! Think about how we learn? Think about how we learn philosophy, Latin, biology, and so on: we don't randomly sample lessons/textbooks and learn. There's a curriculum!

$\implies$ We learn best by prolonged inspection and consideration of one subject at a time.

Q: How fast should learning be? How quickly should a learning algorithm adapt?

A: Transfer and generalization are central problems in machine learning. Deep learning approaches refuse to cooperate on these problems.

Q: How slow should learning be? A: Learning to optimize behaviour from extensive experience in an environment. Evolutionary optimization, for example. Slow and potentially not robust nut highly optimized.

Need: slow learning *and* fast learning!

Alternate title of this talk: learning as a not-too fast, not–to slow, non-stationary, non-i.i.d. process!

Exploring the potential for scalable meta-learning methods: Learning from memory, learning to compare, and so on.

Paper: Meta-learning with warped gradient descent [23].

- Main insight: Learning a geometry does not depend on the initialization – only on the search space!
- In WarpGrad, learn to precondition gradients over a search space.
- SGD defines an empirical parameter distribution of this process.
- A new efficient few-shot learner, but can also handle complex architectures and problems.
  - Maze navigation task vs. A2C RNN agent.
  - Hypernetwork as warp-layers
  - Trained online with periodic meta updates.
  - Also experiments on continual learning.
    → Main result: network learns to both parition and share capacity.

Next up: Learning from Memory, from upcoming paper "Stabilizing Transformers for RL"

- Transformers are extremely powerful memory architectures: also no vanishing/exploding gradients, arbitrary indexing over time, receptive fields only limited by accelerator memory.
- Recent breakthroughs surpass recurrent models by large margins!
- But, vanilla transformers are hard to optimize! Lots of attempts at using transformers:
  - DMLab30: attempts at memory unsuccessful.
  - Unstable
  - Did not even match basic performance
- Further reported difficulties: "Hypothesize transformers inadequacy stems from the fact that pure attentive lookups cannot easily process sequential information.
- New algorithm/architecture: GTrXL, uses transformers and gating variations in RL.

Results:

- On DMLab 30: 3d, first person view POMDP problems that often require memory to solve puzzles. Lots of variations in tasks.
- Achieves SOTA on this set of tasks, comparing to MERLIN (Memory, RL, and Inference Network) [52]. Lots of complexity in MERLIN: memory-based predictor, read-only policy, huge number of losses/coders/decoders.
- In contrast: new transformer based approach (GTrXL) is quite simple, but still achieves roughly the value of MERLIN.

High level: 1) started with transformer (TrXL), then 2) moved to TrXL-I (adds a layer norm, moved inside the residual connection), then 3) added a gating connection between the residual and output of the previous layer (GTrXL).

→ Tried several kinds of gating variants, from Highway, PixelCNN, to a GRU-type, found that GRU worked the best. Extension ablation studies to understand each component (compare # parameters with mean human normalized score on DMLab30).

Numpad task: a continuous control memory problem.

- A 3d represented set of buttons on a pad; $N \times N$ grid of buttons. Agent has to move a cursor over the pad in the right pattern.
- Continuous action space
- State observation space
- 500 episodes.
- New memory-based approach can work quite well on these hard memory control problems!

Q: How does this relate to Meta-RL?

- A recurrent network can encode the dynamics of current environment and adapt policy without further parameter optimization.
- A transformer learning to use attention over the experience buffer.

Dave: I missed the challenge Q :(

4.1.10 Spotlight: Meta-Learning with Warped Gradient Descent

Background: meta-learning is all about transfer across learning *processes*, rather than structures.

→ Depending on what the learning process is, we get different meta-learning types.

MAML [22]: learn an initialization for good final performance. Learning as a function:

$$\mathcal{F}(\theta_0) = L(\theta - \alpha \nabla L(\theta)).$$

Powerful idea! But, hard to scale: 1) Vanishing/exploding meta-gradients, 2) Computationally costly, 3) Hard to make wwork for more than 10 adaptation steps.

Main Question: Can we come up with a gradient-based meta-learner that is not a function of the initialization?

→ Key insight to get it to work: rather than taking the loss surfaces as a given, *warp* it by learning a projection $\gamma = \Omega(\theta; \phi)$.

→ This is equivalent to learning a gradient "preconditioner": $P(\theta)\nabla\mathcal{L}(\theta)$.

Meta-learning now means:

1. In WarpGrad, learn precondition gradients ove search space
2. SGD defines empirical parameter distribution over this space
3. Meta learn warp parameters to yield steepest descent.

Experimental findings:

- Few-shot adaptation: works quite well!
- Also goes beyond trivial few-shot adaptation to methods that MAML based methods didn't work on.
- Extend to RL on maze navigation tasks based on an A2C rNN agent.

Conclusion:

- WarpGrad is an effective and scalable gradient-based meta-learner
- WarpGrad is a flexible parameterization of an optimizer
- Next steps: what are the effective architectures for warp layers?

4.1.11 Spotlight: Meta-Pix: Few Shot Video Targeting

Talk by Jessica Lee.

Problem: video retargeting. Goal is to translate information from a source video to a new video.

→ Pior work: "everybody dance now" Dave: I've seen this, it is mind blowing! Check out the videos.

Problem with prior work, though: requires a massive dataset of prior dance.

Goal: Generate a video of a person given a few samples of their appearance.

Two approaches under "pose-guided image synthesis":

1. Pose2Im: methods learn a model specific to a person and background (sort of like model-based RL, it seems).

   → Same problem as everybody dance now: data hungry.

2. Pose Transfer: transforms a source image directly into a target image.

   → Prior work: "Dance Dance Generation"

Q: Can we reduce this data requirement?

Approach: MetaPis—learn a general model that can be efficiently and quickly adapted to an appearance via meta-learning. Combine REPTILE (first order method) and Pix2VAE.

Pretraining dataset: 10 dance videos from youtube. Follow roughly the same evaluation protocol as the Dance Dance Generation work.

Baselines: random weights, pretrained weights, and MetaPix.

Qualitative results: videos of people being puppeteered in videos, seems to work quite well!

Also include quantitative evaluation and ablation studies (including some very cool visuals of pretraining and fine tuning process).

Conclusions:

- Meta-learning as a personalization technique.
- Meta-training as a generative model provides insights to what meta-learning captures
- Image synthesis as a meta-learning task.

Code: github.com/imjal/metapix

4.1.12 Brenden Lake on Compositional Generalization in Minds and Machines

Goal: understand *systematic compositionality*: the alhebaric capacity to understand and produce novel combiations from known components.

Example: "This is how you dax" (swirl your hand above your head). Now dax twice! Backwards! while jumping! We can do this.

Central Qs:

1. Do modern AI models show systematic compositionality?
2. How do people make compositional generalizations
3. Can compositional skills be acquired via meta learning?

Q: Do neural language models show systematic compositionality?

→ Sequence task to train a model to follow commands. Ex: Walk, then jump after walk, move to the left, and so on. Given this sequence of commands, needs to parse these commands into a predicate like language.

Zero-shot evaluation: never seen this precise command.

SCAN learning setup:

- 21k commands and corresponding action sequences
- Simplifications: no scope ambiguity, no recursion
- Evaluate Seq2Seq RNNs on SCAN with an extensive hyperparameter search.

SCAN Experiments

1. First, use a random train/test split, where test always consists of new and composed commands.

   → Neural net masters zero-shot generalization with about 8% coverage in training (1,650 of 21,000 commands).

2. Generalizing to longer action sequences.

   → Neural net fails to generalize to novel longer sequences, even though all sub-commands are familiar.

3. Composing a new primitive (like the "dax" experiment): all primitives except one are used in training, new primitive "jump" only seen in a few contexts.

   → Neural net has difficulty incorporating new primitive and shows patchy generalization (fails on all conjuncts of "dax thrice" or "dax", while ironically succeeding on "dax thrice and dax").

Insights from SCAN: 1) Powerful Seq2Seq models can do zero-shot generalization to novel compositional instruction, but still far from systematic generalization, and 20 Current models are missing the ability to abstract systematic rules and variables (Marcus, Pinker, etc).

Ex: "Do X and Y" should be learned as a rule with *variables*, where we can seamlessly plug in new X and Y.

Back to people: how do people solve this? Why are people better compositional learners than neural nets?

→ Let's collect some data about how people learn patterns.

- Built a dataset where people are given instructions (pseudowords like "dax" and "zap") with responses (abstract like red circle).

- Four primitives, based around compositionality.

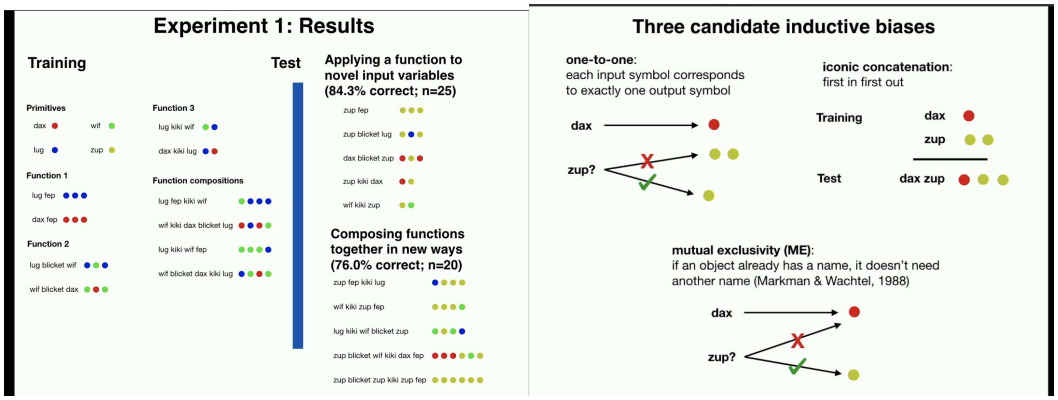
  → No scope ambiguity.

- As suggested above, most neural nets fail miserably on all of the tasks.

Behavioral experiment 1: Design. Told participants to "learn a set of commands and their corresponding outputs". Outputs produced by dragging symbols from a pool of options.

Results ($n = 20 - 25$):

- Overall performance when applying to a *novel input variable* was $\approx 85\%$.
- Achieved 76% accuracy when composing function in new ways



(a) Examples (b) Inductive Biases

Figure 3: Some examples from the dataset (left) and visuals of inductive biases (right).

Summary: find several inductive biases in participants such as → 1) one-to-one: people like these mappings, 2) iconic concatenation(first in first out), and 3) mutual exclusivity: everything has *one* name, not more.

Experiment 2: free-form prompts, examine inductive biases on seq2seq task with NO training examples.

→ More than a source of error, these biases provide critical inductive constraints.

Central Q: Can compositional skills be acquired through meta-learning?

Goals:

- Want a structured neural network that can:
  - Learn new primitives and use the compositionality
  - Encode inductive biases

  - Capture rich starting point people bring to compositional learning tasks

- Do NOT intend models to explain developmental process, resolve which components are learned vs. innate.

New work that tries to capture the above goals: Meta sequence-to-sequence learning [36].

- Learning is distributed over a series of dynamically changing datasets that encourage compositional generalization
- Training episodes contain thousands of support/query set.
- No weight updates at test set.
- Architecture has RNN encoders for the support, passed with attention to a decoder.
  → But, meaning of words is changing between episodes! So can't learn a fixed mapping.

Q: Can a network acquire human-like biases?

A: Yes, it seems! Results: achieved 100% on quickly acquiring new mappings.

→ Also applied to SCAN: evaluated on new episodes with original mappings. Meta seq2seq achieves 99.95% on adding "jump" (the piece that broke earlier models).

Coming back to "dax": both people and the model can learn a new verb and use it compositionally. Model needs to see a lot of combinations of similar pairs, though.

A Step Back:

1. Wanted a framework that can:
   - Learn new primitives and use the compositionality
   - Encode inductive biases
   - Capture rich starting point people bring to compositional learning tasks
2. We now have those three properties!
3. Lots to do though: 1) modeling details of human behavior, 2) acquire human priors, 3) Learn new primitives, 4) Understand how these abilities develop, 5) Large scale compositional word learning.

Started with the question: "what are the computational underpinning of human compositional skills?" Studied:

1. Powerful seq2seq models fail spectacularly when systematic compositional skills are required
2. People can learn new concepts and apply them in new ways
3. Structured, memory-augmented networks can perform few shot composition.

Challenge Question: How do we know what the right inductive biases are?

Brenden Lake: We should look to cognitive development! Early emerging abilities are likely to be most important. We should study computational problems that are easier fr people than for machines. Collect behavioral data and reverse engineer human inductive biasers. I see meta learning as a tool for encoding priors in cognitive modeling or machin learning that are too difficult to specify by hand or easier to specify in data than an explicit prior. In the Bayesian sense, we want a put a prior on our model but it's hard to specify: meta-learning can be an effective way to specify complex priors.

Dave: And that's a wrap! I'm doing the panel now.

5 Saturday December 14th: Workshops

I had meetings the whole day! I caught a few minutes of the OptRL panel

5.1 Panel at OptRL: Doina and Rich

Q: Are you happy with the balance of engineering vs. theory in RL?

Doina: From my point of view we are trying to do good science and want to be building algorithms that we understand. What does it mean to understand algorithms? In the past this has often meant theory: convergence, PAC bounds, etc. But in some cases the theory can't yield understanding: vacuous bounds, hard to analyze cases, mismatch from real world, and so on. So, what we need is a concerted effort to make theory *meaningful*, and to make our experimental analysis more thorough in the sense of trying to understand what is going on, even in large scale experiments. Not necessarily theory but might point out new directions for theory.

Rich: Well, Doina, that's a very bold point of view you have that it's all about science! I have a hard time disagreeing. The engineering and experimental pieces are also really important. Every field has to deal with this clash. There shouldn't be a conflict. Any field, engineering or science, shoudl be like a conveyor belt: at one end you put in big ideas and practice, and at the other end you output a better deeper theoretical understanding. There is a natural drift toward experimental work. The goal is to increase understanding. You have to start with things you don't understand! We need new things on the conveyor belt so we understand more over time.

Doina: From a theoretical point of view, people say RL is very hard; but, that makes it very exciting! It's a good time to do theory in RL. After stalling for some number of years now we have momentum.

Rich: I totally agree. The field is hard because we're working with fundamental questions. I think about it as how intelligence works, how to achieve goals, and so on. The more we get to the core of these key problems, we should expect them to be challenging. Don't be discouraged when you do hard things! The real conclusion is that we have to value incremental progress on hard problems. A small amount of fundamental progress is a big deal.

Dave: Had to run to meetings :(

5.2 Talk at Deep RL: Michael Littman on Assessing the Robustness of Deep RL Algorithms

Background:

- Started off interested in explainable RL: why does DQN choose the moves it does in Atari?
- Ended up wondering if any explanation at all is possible
- **Punchline:** Generalizing Q values is hard![54]

Case study 1: Amidar! DQN can do extremely well on this game.

→ But: can we look at one of the moves and figure out *why* it made this decision?

To answer this:

- An explanation is something that tells us how things would have been if something else happened.
- Tried causal interventions:
  1. Idea one: Prefill one of the boxes! Think Pac-Man with some random pellets missing (couldn't have reached the state).
     → Agent stops moving entirely and starts to be killed.
  2. Next up: get rid of the enemies! Same game (again think Pac man with no ghosts).

$\implies$ Takeaway here: DQN doesn't *really* know how to play the game! All it does is learn a good strategy, but it's very brittle.

Next up: investigate *saliency*. Ask: what makes big changes in action choice or value prediction if blurred out? What does the learned netwwork pay attention to?

→ Memorized movement!

So: maybe we're thinking about learning wrong. Let's go back to **supervised learning**. How do we measure success?

- how well it does on training set
- Interpolation: How well it does on test/validation set(s)
- Extrapolation: How well it does on out of distribution data

Analogues in RL, though:

- Training examples → On policy states
- Interpolation → Off-policy states
- Exrapolation → unreachable states.

Q: How can we test these different kinds of generalization/states in RL?

A: Different interventions! Off-policy: push to different states, force to unreachable states, and so on.

Evaluation Metrics: 1) Value Estimation Error (VEE): how far off is Q error?, and 2) Total Accumulated Reward (TAR).

Findings:

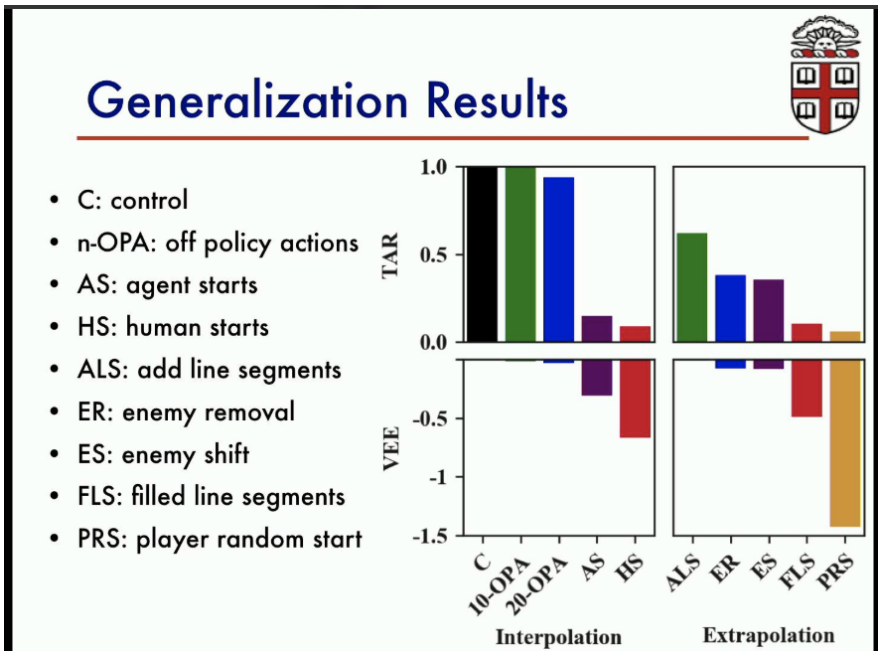


Figure 4: Results on generalization types for RL

1. Full results in Figure 5.
2. VEE and TAR correlate.
3. Novel states not recognized! Learned representation does not find the novel states to be similar to those seen in training (via activation analysis).

Q: So how can we improve generalization?

A: Well, in supervised learning, we do a few things: 1) more data, and 2) less model (regularization).

Q: So how about in RL?

A: A few things: 1) can increase time, 2) Diversify training data, or 3) Regularize (reduce model capacity).

Tried these three changes to the learning procedure and inspected performance.

→ Finding: all three help! But not very much.

Case Study: CoinRun [14]. New set of environments that are designed explicitly for studying generalization in RL.

- Goal is to collect coin at the end of the level
- Agent spawns on the left

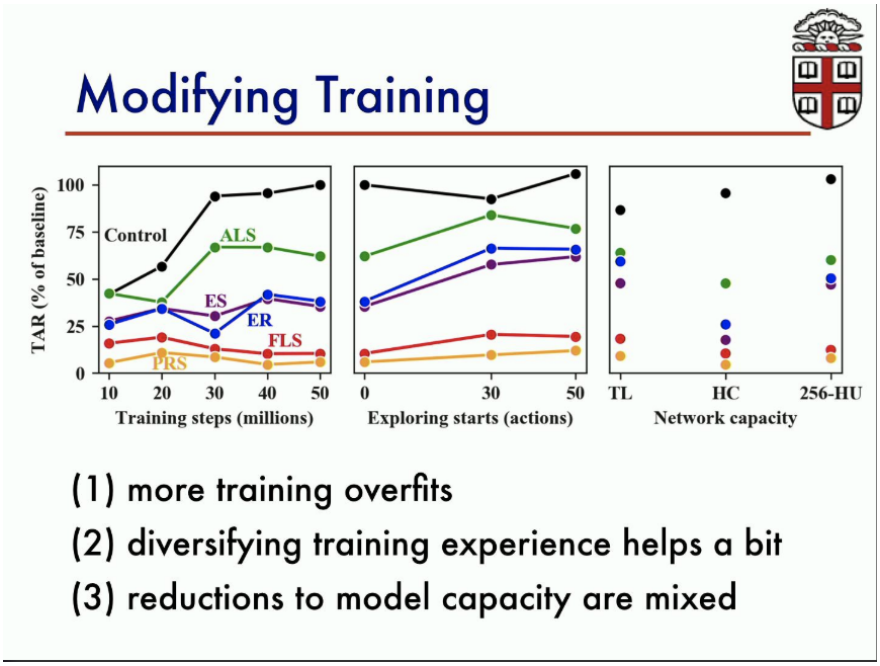


Figure 5: Results for changing three aspects of training procedure

- Contains obstacles, enemies, level ends by death.
- Levels divided into three levels of difficulty.

  → Results from Cobbe et al. [14] show *overfitting* curves, which is exciting to see in RL. See Figure 6.

But, the results on CoinRun were using a policy optimization method rather than DQN.

→ Tried to replicate with DQN, found the opposite result: it didn't work! Contacted the authors and they had found the same thing.

→ Later finding: DQN (and PPO) will generalize well with a lot of levels used (when the hard levels are removed).

Also studied prediction errors;

- High prediction error associated with failure.
- Prediction error lower in training than testing.
- Training performance = testing performance given enough data

Summary:

- Good RL performance is seductive, but we need to look closer
- Analogy between RL and supervised learning is subtle

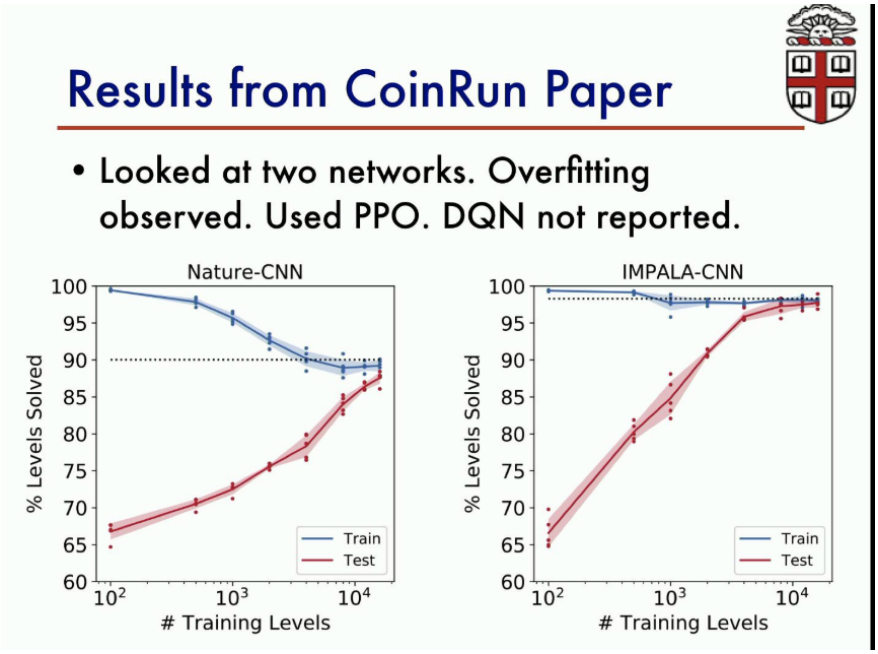


Figure 6: Results on CoinRun domain from Cobbe et al. [14]

- DQN does not generalize well in Amidar, CoinRun, and some weak generalization in CoinRun difficulty 2.
- Found prediction error and internal representation distance good predictors of poor generalization
- Adjusting training volume, model capacity, and exploration help, but only a bit.

.....

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in Neural Information Processing Systems*, pages 6155–6166, 2019.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[3] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, 2019.

[4] Christopher G Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3557–3564. IEEE, 1997.

[5] Bernard J Baars. Global workspace theory of consciousness: toward a cognitive neuroscience of human experience. *Progress in brain research*, 150:45–53, 2005.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Advances in Neural Information Processing Systems*, 2015.

[7] Yoshua Bengio. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017.

[8] Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv preprint arXiv:1901.10912*, 2019.

[9] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[10] Nicolas Carion, Nicolas Usunier, Gabriel Synnaeve, and Alessandro Lazaric. A structured prediction approach for generalization in cooperative multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8128–8138, 2019.

[11] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, pages 8928–8939, 2019.

[12] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. 2018.

[13] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, pages 1785–1796, 2019.

[14] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

[15] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *arXiv preprint arXiv:1905.11979*, 2019.

[16] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[17] Simon S Du and Jason D Lee. On the power of over-parametrization in neural networks with quadratic activation. *arXiv preprint arXiv:1803.01206*, 2018.

[18] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[19] Yonathan Efroni, Nadav Merlis, Mohammad Ghavamzadeh, and Shie Mannor. Tight regret bounds for model-based reinforcement learning with greedy policies. In *Advances in Neural Information Processing Systems*, 2019.

[20] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[21] Matthew Fellows, Anuj Mahajan, Tim GJ Rudner, and Shimon Whiteson. Virel: A variational inference framework for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 7120–7134, 2019.

[22] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[23] Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, 2019.

[24] Adam Gaier and David Ha. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[25] Paul Gölz, Anson Kahng, and Ariel D Procaccia. Paradoxes in fair machine learning. In *Advances in Neural Information Processing Systems*, pages 8340–8350, 2019.

[26] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

[27] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

[28] Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*, pages 12467–12476, 2019.

[29] Rodrigo Toro Icarte, Ethan Waldie, Toryn Klassen, Rick Valenzano, Margarita Castro, and Sheila McIlraith. Learning reward machines for partially observable reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 15497–15508, 2019.

[30] Allan Jabri, Kyle Hsu, Abhishek Gupta, Ben Eysenbach, Sergey Levine, and Chelsea Finn. Unsupervised curricula for visual meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 10519–10530, 2019.

[31] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[32] Khurram Javed and Martha White. Meta-learning representations for continual learning. *arXiv preprint arXiv:1905.12588*, 2019.

[33] Xiaodong Jia, Liang Zhao, Lian Zhang, Juncai He, and Jinchao Xu. irda method for sparse convolutional neural networks. 2018.

[34] Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019.

[35] Charles Kemp, Amy Perfors, and Joshua B Tenenbaum. Learning overhypotheses with hierarchical bayesian models. *Developmental science*, 10(3):307–321, 2007.

[36] Brenden M Lake. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, 2019.

[37] Sangkyun Lee and Stephen J Wright. Manifold identification in dual averaging for regularized stochastic online learning. *Journal of Machine Learning Research*, 13(Jun):1705–1744, 2012.

[38] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pages 11669–11680, 2019.

[39] H Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. 2011.

[40] Russell Mendonca, Abhishek Gupta, Rosen Kralev, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Guided meta-policy search. In *Advances in Neural Information Processing Systems*, 2019.

[41] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, 2019.

[42] Vaishnavh Nagarajan and J Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 11611–11622, 2019.

[43] Harikrishna Narasimhan, Andrew Cotter, and Maya Gupta. Optimizing generalized rate metrics with three players. In *Advances in Neural Information Processing Systems*, pages 10746–10757, 2019.

[44] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[45] Deepak Pathak, Chris Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: a study of generalization via modularity. In *Advances in Neural Information Processing Systems*, 2019.

[46] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012.

[47] Hao Sun, Zhizhong Li, Xiaotong Liu, Bolei Zhou, and Dahua Lin. Policy continuation with hindsight inverse dynamics. In *Advances in Neural Information Processing Systems*, pages 10265–10275, 2019.

[48] Yi Chern Tan and L Elisa Celis. Assessing social and intersectional biases in contextualized word representations. In *Advances in Neural Information Processing Systems*, pages 13209–13220, 2019.

[49] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[50] Harm van Seijen, Mehdi Fatemi, and Arash Tavakoli. Using a logarithmic mapping to enable lower discount factors in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

[51] Eszter Vertes and Maneesh Sahani. A neurally plausible model learns successor representations in partially observable environments. In *Advances in Neural Information Processing Systems*, 2019.

[52] Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

[53] Colin Wei and Tengyu Ma. Data-dependent sample complexity of deep neural networks via lipschitz augmentation. In *Advances in Neural Information Processing Systems*, 2019.

[54] Sam Witty, Jun Ki Lee, Emma Tosch, Akanksha Atrey, Michael Littman, and David Jensen. Measuring and characterizing generalization in deep reinforcement learning. *arXiv preprint arXiv:1812.02868*, 2018.

[55] Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.