Constraint vs Score: What's the tradeoff?

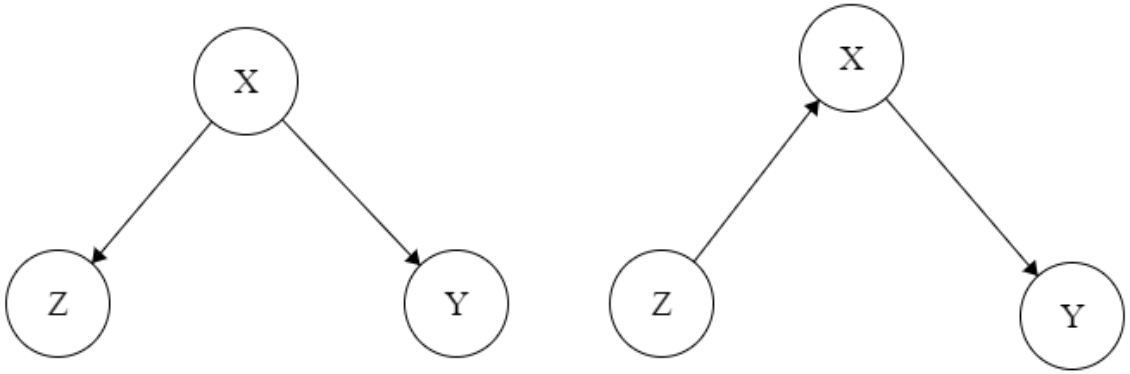
— a basic survey of Bayesian network learning algorithm

Shuyan Wang

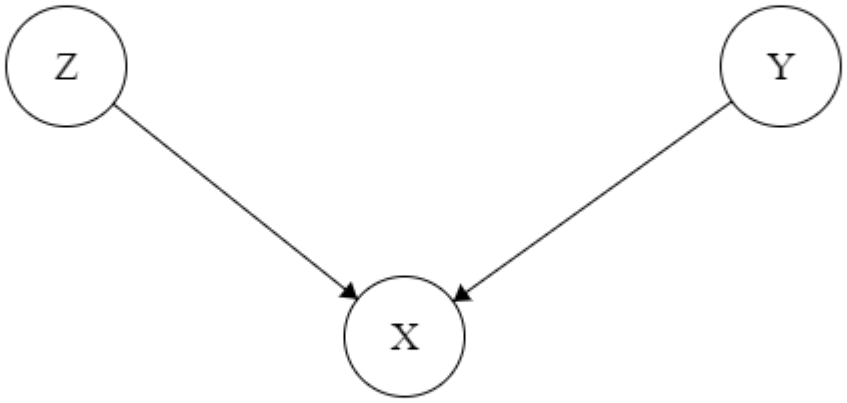
Causal Graph and Algorithm

- X is a cause of Y if **intervening/manipulating** the state of X changes the distribution of Y
- Directed acyclic graph (DAG) are often used to represent causal relations
- Constraint-based and score-based algorithms searching causal graphs
- Guarantee of Consistency: Markov Condition and Faithfulness Assumption

Markov Condition

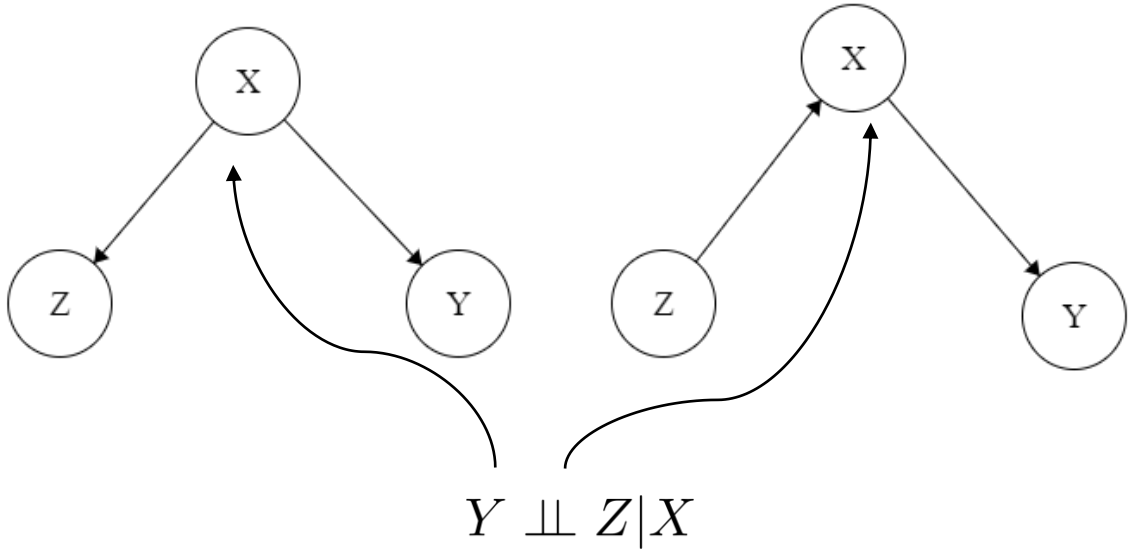


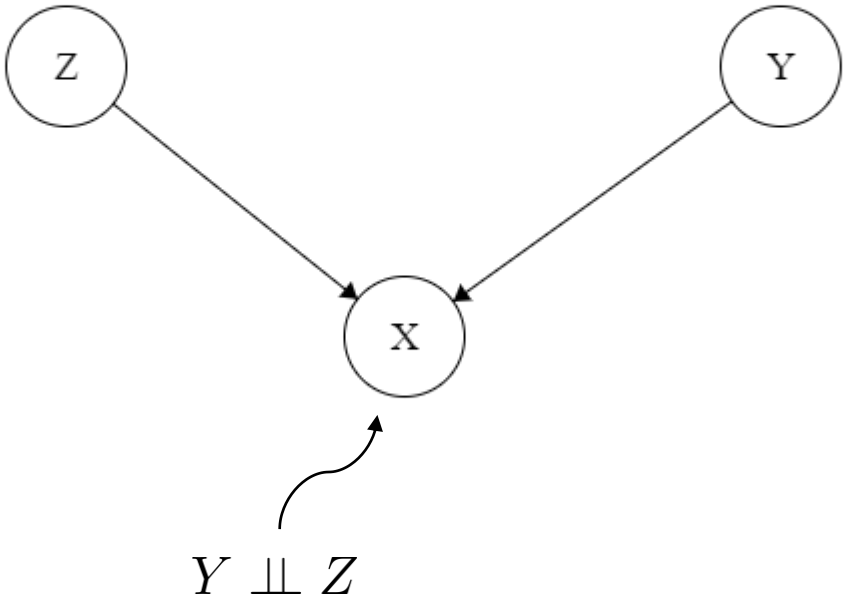
$$Y \perp\!\!\!\perp Z | X$$



$$Y \perp\!\!\!\perp Z$$

Faithfulness Assumption





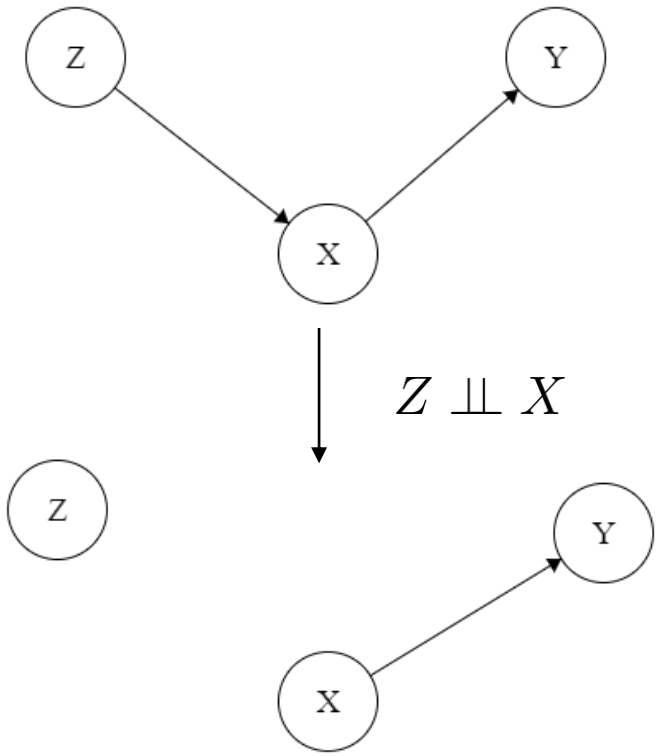
Meek Conjecture

If a DAG $\mathcal{H}$ is an independence map of another DAG $\mathcal{G}$, then there exists a finite sequence of edge additions and covered edge reversals in $\mathcal{G}$ such that (1) after each edge modification $\mathcal{H}$ remains an independence map of $\mathcal{G}$ and (2) after all modifications $\mathcal{H} = \mathcal{G}$.

$$Id(H) \subset Id(G) \implies$$

It takes finitely many steps, each changing one edge, to turn G into H.

Meeks Conjecture



Score-based

- Starting with an empty graph

- add edges/dependencies that improves the score mostly

- if adding edges does not improve score anymore, remove edges that improves the score mostly

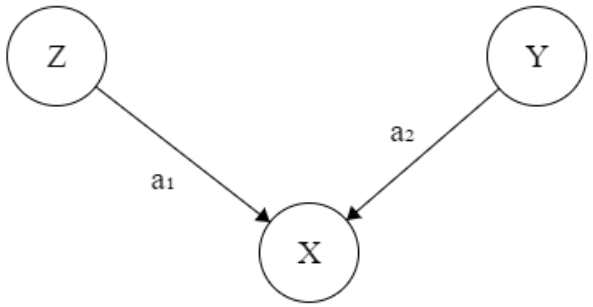
$$S(\mathcal{G}, \mathbf{D}) = \sum_{i=1}^{n} s(X_i, \mathbf{Pa}_i^{\mathcal{G}})$$

$$S_B(\mathcal{G}, \mathbf{D}) = \log p(\mathcal{G}^h) + \log p(\mathbf{D}|\mathcal{G}^h)$$

Simulation

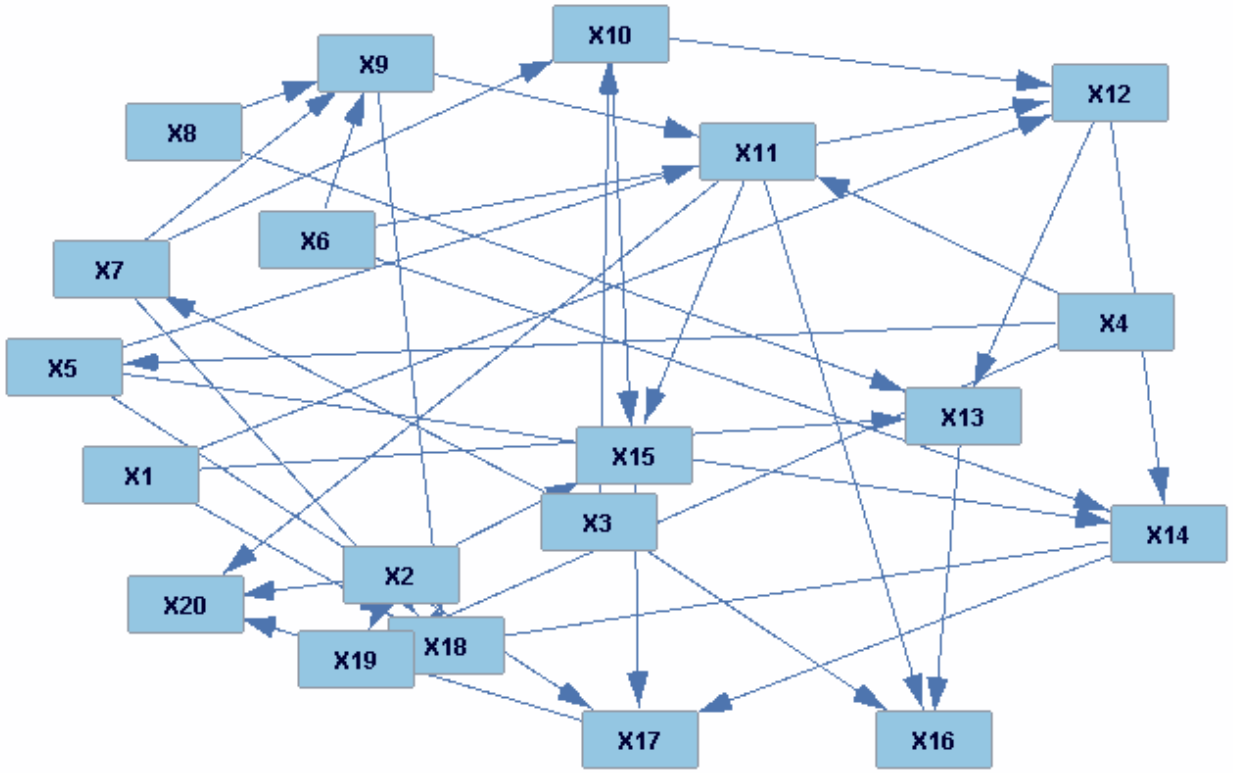
- Linear
- Gaussian
- 20 variables
- Average degree:4, 8, 12
- Constraint-based:FCI
- Score-based:FGES
- Combination: GFCI

Simulation Uses Linear Gaussian Model

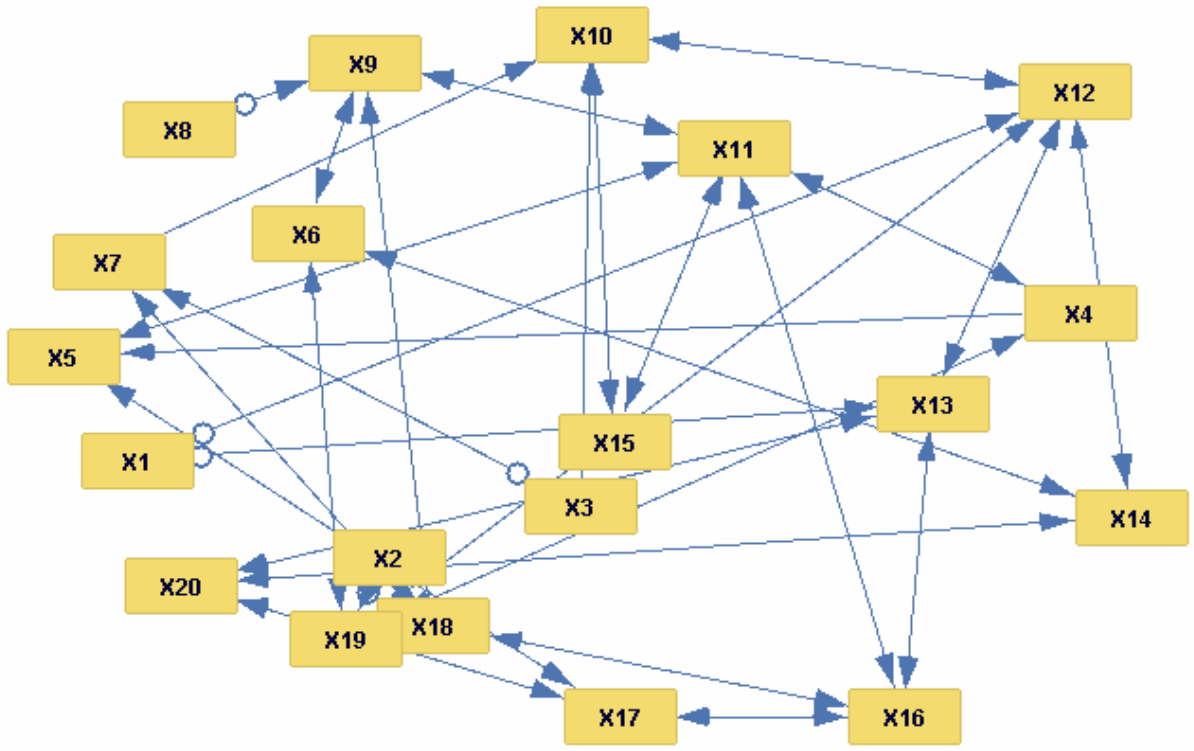


Data(X,Y,Z) ~ P(X,Y,Z) ~ $X = a_1 Z + a_1 Y + e$

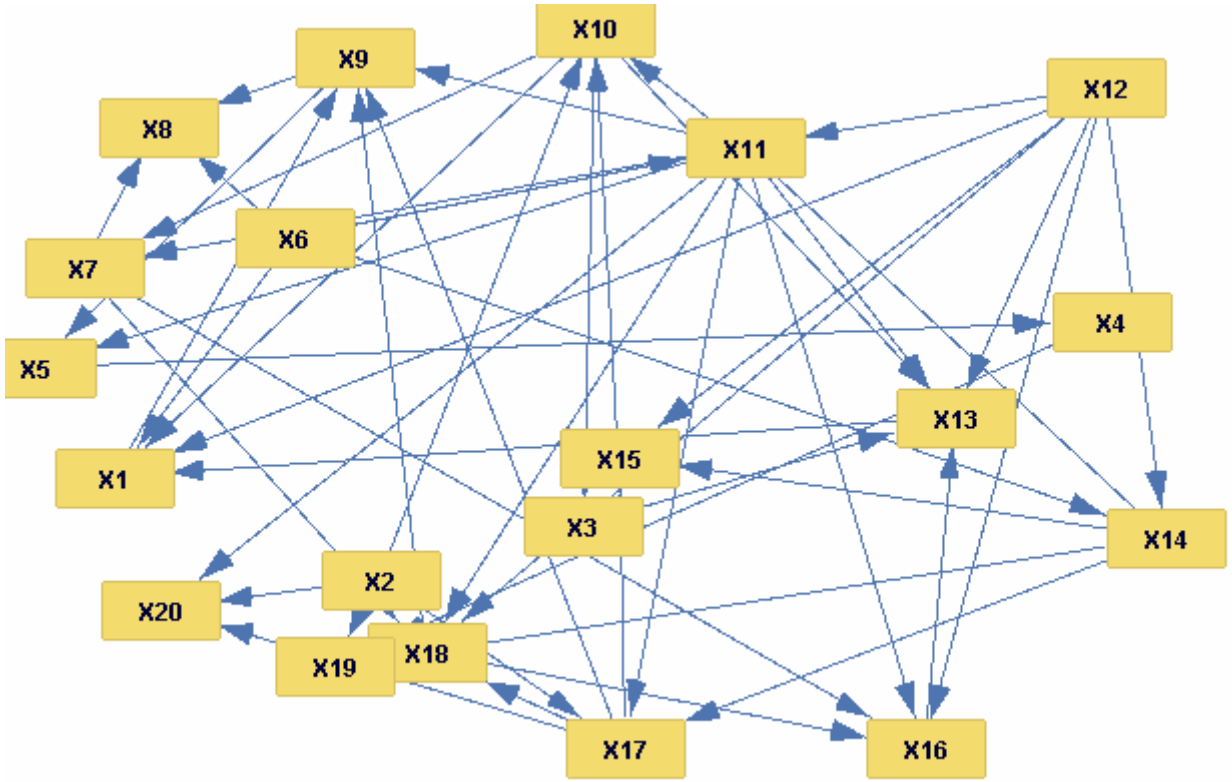
20 variables with ave. degree of 4



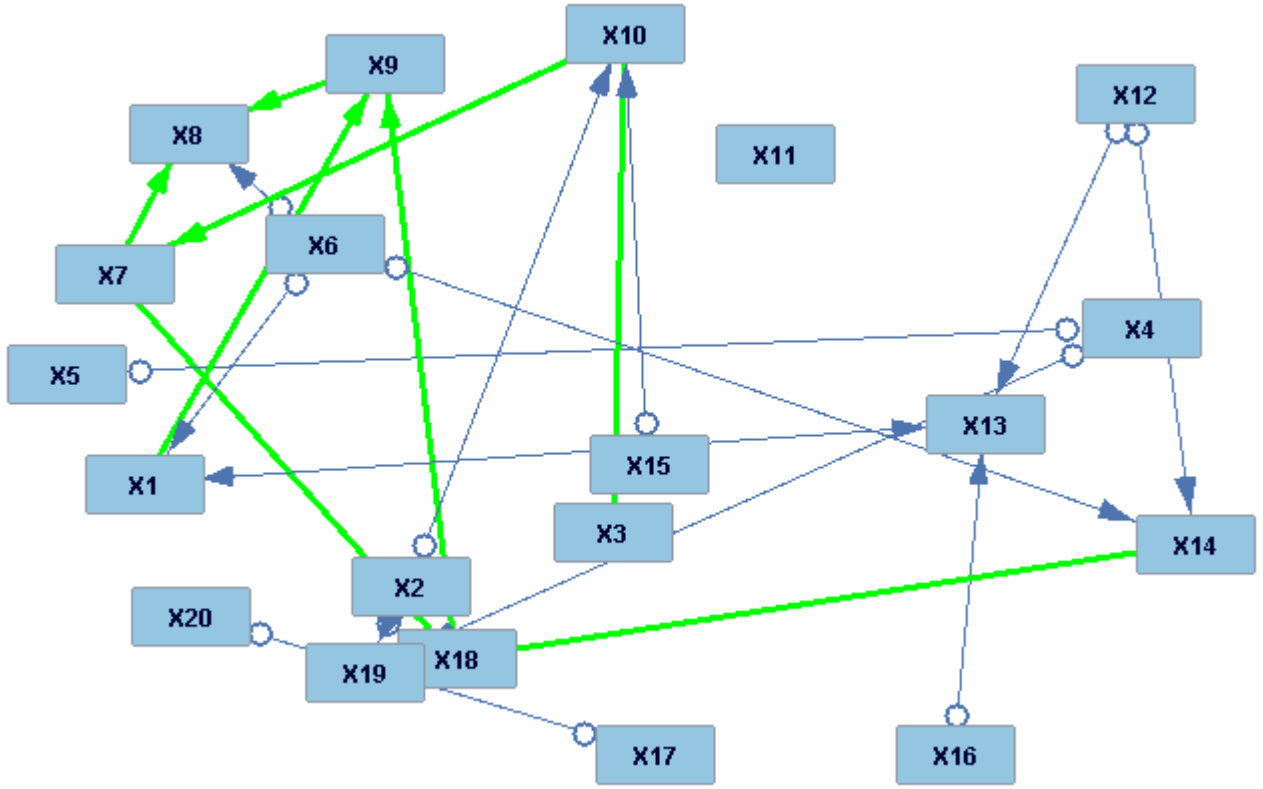
FCI


X10
X9
X12
X8
X11
X6
X7
X4
X5
X13
X15
X1
X3
X14
X2
X20
X18
X19
X17
X16

FGES


X10
X9
X12
X8
X11
X6
X7
X4
X5
X13
X15
X1
X3
X14
X2
X20
X18
X19
X17
X16

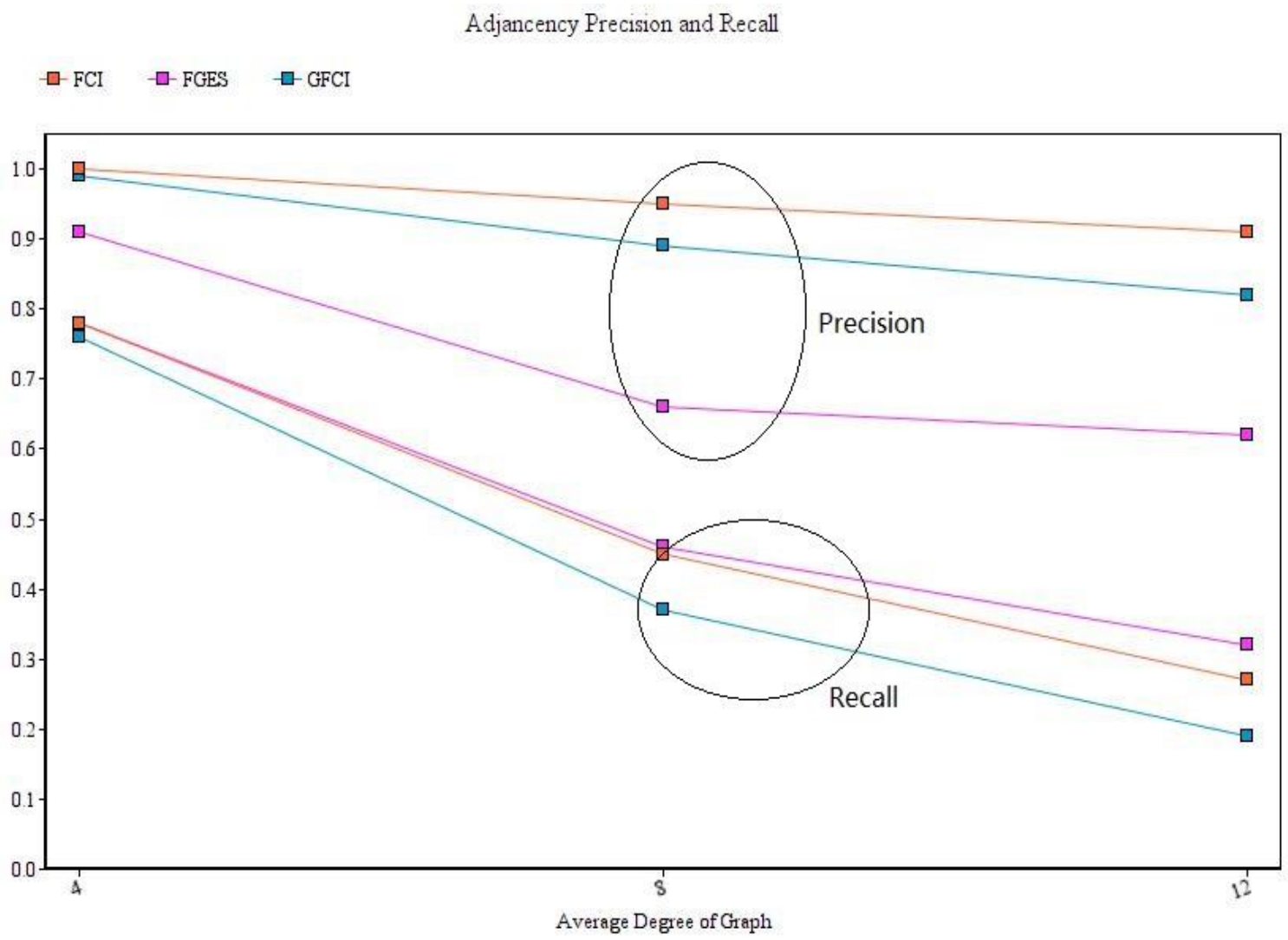
GFCI


X10
X9
X8
X11
X12
X6
X7
X4
X5
X13
X15
X1
X3
X14
X2
X20
X18
X19
X17
X16

Adjancency Precision & Recall

- Precision – percentage of edges in the output graph that are in the true graph
- Recall – percentage of edges in the true graph are in the output graph

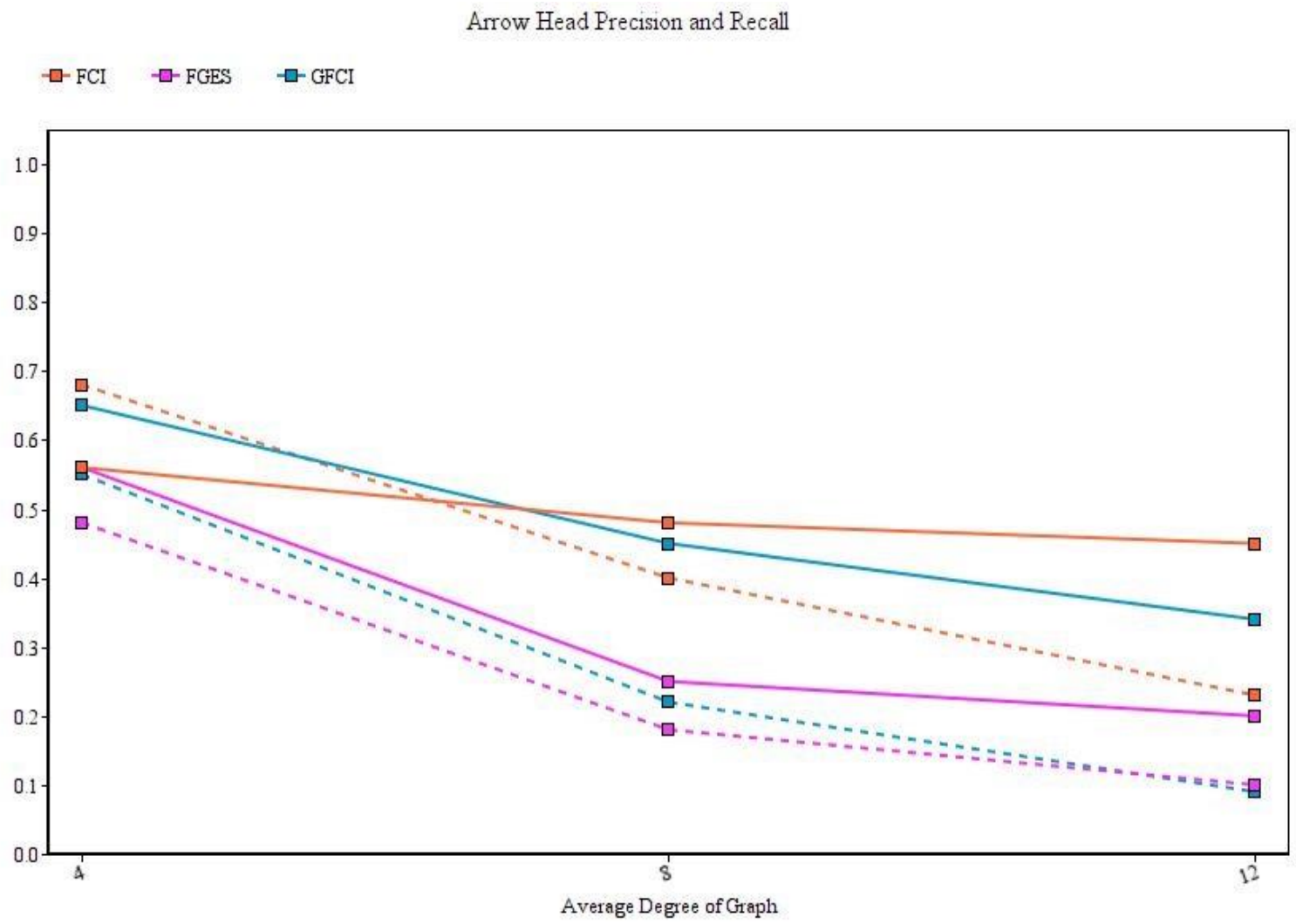
Adjancency Precision & Recall



Arrow Head Precison & Recall

- Precision – percentage of edges in the output graph pointing at the correct direction that are in the true graph
- Recall – percentage of edges in the true graph are in the output graph pointing at the correct direction

Arrow Head Precison & Recall



Summary

- Constraint-based algorithm produces fewer extra edges and is more accurate about the direction of the edges
- Score-based algorithm are more sensitive detecting edges but less accurate

Improvement

- Aiming at minimizing score can help relaxing faithfulness assumption
- Adding direction rules used in the constraint-based algorithm into FGES

Analyze Pairwise Comparison Data Using Stochastically Transitive Model

Weichen Wu & Xinyu Yao

Carnegie Mellon University

Apr 30th, 2020

Analyze Pairwise Comparison Data

- Data: Pairwise comparison among items
  e.g. Wins and losses among tennis players
- Goal: Rank the items correspondingly
  e.g. Rank the players according to their game record

The Model

- n items: $1, 2, ..., n$
- Underlying complete ordering $\pi^* : [n] \to [n]$
  $\pi^*(i) < \pi^*(j)$ means i is better than j
- Matrix of underlying comparison probability $M^* \in \mathbb{R}^{n \times n}$:

$$M^*_{ij} = \mathbb{P}(i \text{ beats } j)$$

- Observation: $Y \in \mathbb{R}^{n \times n}$, in which

$$Y_{ij} = \mathbb{I}(i \text{ beats } j) \sim \text{Ber}(M^*_{ij})$$

- Goal: Estimate M^* using Y

Related Work: Parametric Model

- "Scores" for each item: $w_1^*, w_2^*, ..., w_n^*$
- Assumption: $M_{ij}^* = F(w_i^* - w_j^*)$
- $\mathbb{C}_{PAR}$: The set of all parametric matrices
- Bradley-Terry-Luce(BTL) model: $F =$ sigmoid function
- Thurstone model: $F =$ standard normal CDF
- Estimation method: MLE of w^*
- Limitation: Fit poorly to real-world data due to strict single-factor assumption

Stochastically Transitive Models

- Shah, Balakrishnan, Guntuboyina, Wainwright, 2015: *Stochastically Transitive Models for Pairwise Comparisons: Statistical and Computational Issues*
- Strong Stochastic Transitivity(SST) condition: if $\pi^*(i) < \pi^*(j)$, then for any $k \neq i, j$, $M^*_{ik} \geq M^*_{jk}$
- Parametric model as a special case

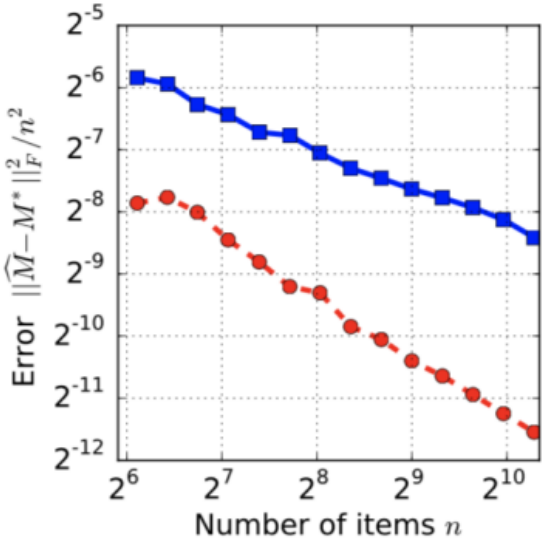
Algorithm: Singular Value Thresholding (SVT)

- Singular Value Decomposition(SVD) of Y : $Y = UDV^\top$
- Soft thresholding of singular values: $T_{\lambda_n}(D) = \max(D - \lambda_n, 0)$
- Estimator: $\widehat{M}_{\lambda_n} = UT_{\lambda_n}(D)V^\top$
- Error bound: Take $\lambda_n = 2.1\sqrt{n}$, then with high probability,

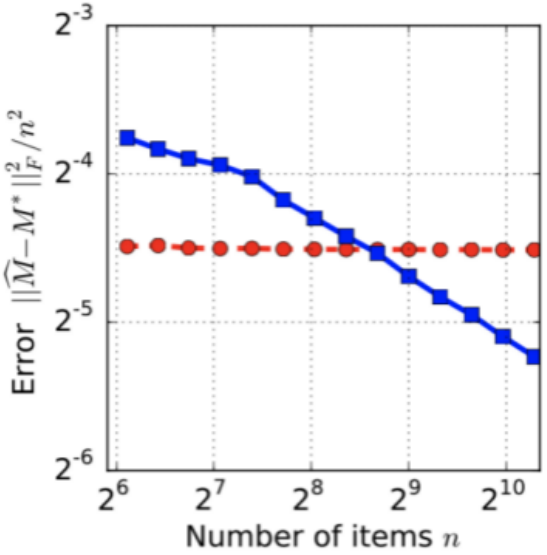
$$MSE(\widehat{M}_{\lambda_n}, M^*) \leq Cn^{-1/2}$$

Experiments and Results

- Data Generating models:
  1. Thurstone model: $M^*_{ij} = \Phi(w^*_i - w^*_j)$
  2. High SNR model: $\left| M^*_{ij} - \frac{1}{2} \right| \geq \gamma$
- Estimation method:
  1. Thurstone model + MLE
  2. Singular Value Thresholding



Data: Thurstone



Data: High SNR

Exploring Theory and Application of Attention Model with Reinforcement Learning and Supervision

Yijie Sun, Haiguang Liao

Introduction

- Recent work by (Kool et al., 2018) applied attention-based REINFORCE to learn generalizable heuristics efficiently for combinatorial optimization problems such as Travelling Salesman Problem(TSP)

- Reinforcement Learning algorithm makes the attention-based model's training unstable and sample

Question we focused

- Can we leverage supervised data based on non-learning methods to guide the training process of attention-based model?

- Specifically, can genetic algorithm (GA) that gives incrementally better solutions make such guidance even easier?

Theoretical Contribution (Lipschitz)

- Why GA can guide RL model? *Pirotta, M.,2015*

- the Lipschitz continuity properties for Markov Decision Processes to safely speed up policy-gradient algorithms

Theoretical Contribution (Lipschitz)

- Goal: Both the expected return of a policy and its gradient are Lipschitz continuous w.r.t. policy parameters.

- $L(\theta-\theta') < K\,|\theta-\theta'|$, where K is some constant, L is some proper loss, θ and θ' are the parameters of successive iterations.

Theoretical Contribution (Lipschitz)

Assumptions:

1. the Lipschitz continuity of the (parameterized) state-transition model, the reward function, and the policies considered in the learning process.

   mild assumptions in realistic setup

Theoretical Contribution (Lipschitz)

Assumptions:

2. the Lipschitz gradient of (parameterized) policy logarithm

$$\forall (s,a) \in S \times A, \theta \in \Theta, i, |\nabla_{\theta_i} \log \pi^{\theta}(a|s)| \leq M_{\theta_i}$$

Theoretical Contribution (Lipschitz)

Main results:

$$|J_\mu^\theta - J_\mu^{\theta'}| = \frac{1}{1-\gamma}|E_{(s,a)\sim\xi_\mu^\theta}[R(s,a)] - E_{(s,a)\sim\xi_\mu^{\theta'}}[R(s,a)]| \leq \frac{L_R}{1-\gamma}\kappa(\xi_\mu^\theta, \xi_\mu^{\theta'})$$

where κ is the Kantorovich distance.

Theoretical Contribution (Policy Gradient)

By policy gradient theorem, the loss gradient of REINFORCE (Kool et al. 2018):

$\nabla L(\theta|s) = E_{p_\theta(\pi|s)}[(L(\pi) - b(s))\nabla log p_\theta(\pi|s)]$

$p_\theta = \prod_{t=1}^{n} p_\theta(\pi_t|s, \pi_{1:t-1})$

Supervised Model Loss based on KL-Divergence:

$L = p_\theta^T(log p_\theta - log q)$

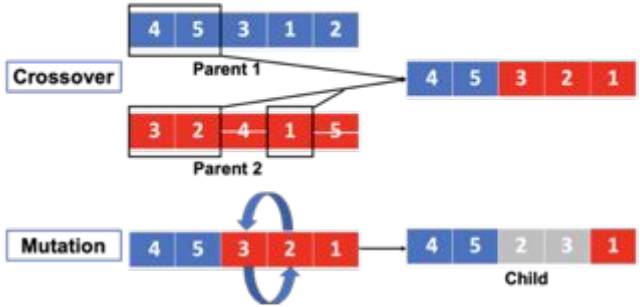
$q = \prod_{t=1}^{n} q_t$ (Distribution obtained from GA results)

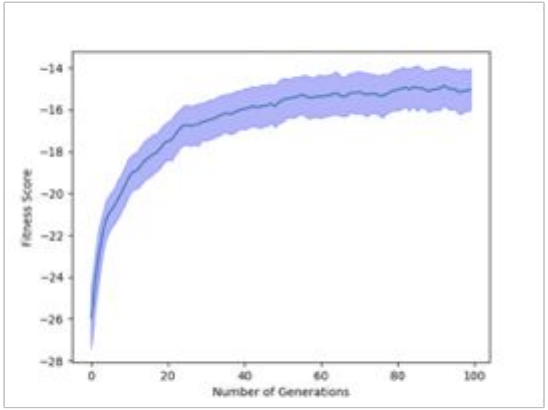
$\nabla L = C\nabla(log p_\theta)$ (which is consistent with REINFORCE loss)

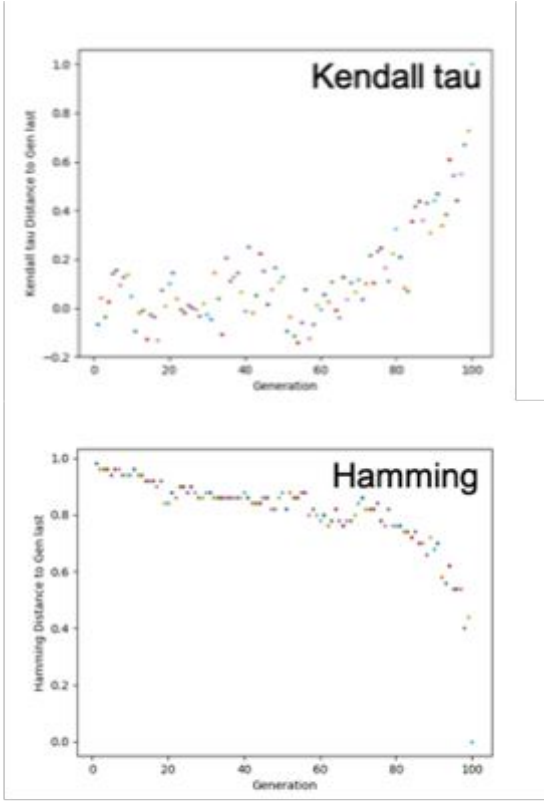
Incremental GA-guided training (i: epoch number):

$$L_i = p_\theta^T(log p_\theta - log q_{Genetation:10^i})$$

Experimental Results







Key Finding: GA-based results incrementally get closer to optimum results

Experimental Results

First epoch result:

```
(base) D:\GA_RL_TSP_Theory-master\GA_RL_TSP_Theory-master\Repo_AttentionTSP>python eval.py data/tsp/tsp50_test_seed1234.
pkl --model outputs/tsp_50/run_20200426T181558/epoch-0.pt --decode_strategy greedy
  [*] Loading model from outputs/tsp_50/run_20200426T181558/epoch-0.pt
 10%|                                                                                                  | 1/10 [00:01<00:14,   1
 20%|                                                                                                          | 2/10 [00:01<0
 30%|                                                                                                                  | 3/10
 40%|
 50%|
 60%|
 70%|
 80%|
 90%|
100%|
100%|                                                                                                                  | 10/10 [00:05<00:00,   1.91it/s]
Average cost: 19.469135284423828 +- 0.07356663227081299
Average serial duration: 0.5186470405578614 +- 0.007113419311778707
Average parallel duration: 0.0005064912505447865
Calculated total duration: 0:00:05
```

Experimental Results

Second epoch result:

```
(base) D:\GA_RL_TSP_Theory-master\GA_RL_TSP_Theory-master\Repo_AttentionTSP>python eval.py data/tsp/tsp50_test_seed1234.
pkl --model outputs/tsp_50/run_20200426T181558/epoch-1.pt --decode_strategy greedy
  [*] Loading model from outputs/tsp_50/run_20200426T181558/epoch-1.pt
 10%|                                                                                                  | 1/10 [00:01<00:14,  1
 20%|                                                                                                          | 2/10 [00:01<0
 30%|                                                                                                                  | 3/10
 40%|
 50%|
 60%|
 70%|
 80%|
 90%|
100%|
100%|                                                                                                                  | 10/10 [00:05<00:00,  1.89it/s]
Average cost: 12.48619556427002 +- 0.05998109340667725
Average serial duration: 0.5335053829193115 +- 0.007127382472011064
Average parallel duration: 0.0005210013505071401
Calculated total duration: 0:00:05
```

Experimental Results

Third epoch result:

```
(base) D:\GA_RL_TSP_Theory-master\GA_RL_TSP_Theory-master\Repo_AttentionTSP>python eval.py data/tsp/tsp50_test_seed1234.
pkl --model outputs/tsp_50/run_20200426T181558/epoch-2.pt --decode_strategy greedy
  [*] Loading model from outputs/tsp_50/run_20200426T181558/epoch-2.pt
 10%|                                                                                                  | 1/10 [00:01<00:14,  1
 20%|                                                                                                          | 2/10 [00:01<0
 30%|                                                                                                                  | 3/10
 40%|
 50%|
 60%|
 70%|
 80%|
 90%|
100%|
100%|                                                                                                                  | 10/10 [00:05<00:00,  1.90it/s]
Average cost: 14.9197359085083 +- 0.046204800605773 93
Average serial duration: 0.5285226825714111 +- 0.007139521012438624
Average parallel duration: 0.0005161354321986437
Calculated total duration: 0:00:05
```

Experimental Results

Fourth epoch result:

```
(base) D:\GA_RL_TSP_Theory-master\GA_RL_TSP_Theory-master\Repo_AttentionTSP>python eval.py data/tsp/tsp50_test_seed1234.
pkl --model outputs/tsp_50/run_20200426T181558/epoch-3.pt --decode_strategy greedy
  [*] Loading model from outputs/tsp_50/run_20200426T181558/epoch-3.pt
 10%|                                                                                                  | 1/10 [00:01<00:14,  1
 20%|                                                                                                  | 2/10 [00:01<0
 30%|                                                                                                  | 3/10
 40%|
 50%|
 60%|
 70%|
 80%|
 90%|
100%|
100%|                                                                                                  | 10/10 [00:05<00:00,  1.91it/s]
Average cost: 16.987154006958008 +- 0.05102144241333008
Average serial duration: 0.5203445774078369 +- 0.0070201242908623975
Average parallel duration: 0.0005081490013748407
Calculated total duration: 0:00:05
```

Experimental Results

- Promising results we obtained so far

  Loss relatively low considering the GA training size

  Efficiency monotonically increases w.r.t epochs

- Challenge we encountered so for

  Original REINFORCE has much larger training size while we have to generate our own GA data

Conclusions

- Theoretically, under mild assumptions, RL policy model has Lipschitz property; GA-based solution incrementally gets closer to optimum. Thus, GA-based supervision can effectively guide the training of attention-based policy model.
- The above theoretical results is proved by some primary experimental results
- Robustness of the GA-based guidance requires larger training sets and parameter tunings; better loss design in the supervised stage is needed

References

Kool, Wouter, Herke Van Hoof, and Max Welling. "Attention, learn to solve routing problems!." arXiv preprint arXiv:1803.08475 (2018).

Liao, H., Dong, Q., Dong, X., Zhang, W., Zhang, W., Qi, W., ... & Kara, L. B. (2020). Attention Routing: track-assignment detailed routing using attention-based reinforcement learning. arXiv preprint arXiv:2004.09473.

Towards Understanding the State of Sparsity of Deep Neural Networks

Ting-Wu Chin

Why sparse neural networks?

- Deploying deep neural networks onto resource-constrained devices often requires model compression
- Sparsifying the weights of neural networks is a popular approach for model compression
- Goal: understand the state of sparsity of deep neural networks

Methods for group sparsity

Method	Lasso / Trimmed Lasso (Wen et al. 2016; Yun et al. 2019)	ℓ_0 stochastic relaxation (Louizos et al. 2018)
Objective	$\widehat{L(\theta)} = L(\theta) + \lambda \sum_{j=1}^{p} \left\| \theta_{G_j} \right\|_2$ $\widehat{L(\theta, w)} = L(\theta) + \lambda \sum_{j=1}^{p} w_j \left\| \theta_{G_j} \right\|_2$ $\text{s.t.} \ \|w\|_1 = p - h$	$\widehat{L(\theta)} = \frac{1}{N} \sum_{i=1}^{N} L\left(\theta \odot z^{(i)}\right) + \lambda \sum_{j=1}^{p} \left(1 - P\left(s_j \leq 0 \middle\| \phi\right)\right)$ $z_{G_j}^{(i)} = \min\left(0, \max\left(1, s_j\right)\right), s_j = f\left(\phi, \epsilon^{(i)}\right)_j, \epsilon^{(i)} \sim p(\epsilon)$
Algorithm	Minimize $\widehat{L(\theta)}$ w/ SGD Minimize $\widehat{L(\theta, w)}$ w/ block coordinate descent	Minimize $\widehat{L(\theta)}$ w/ SGD

Statistical analysis for Lasso and Trimmed-Lasso

Lasso	Trimmed Lasso
Restricted eigenvalue condition $\frac{1}{n}\|Xv\|_2^2 \geq \phi_0^2\|v\|_2^2$ for all $S \subseteq \{1,2,\dots,p\}$ and $\|S\| = k$ and all $v \in \mathcal{C}(S;3)$	Restricted strong convexity $\frac{1}{n}\|X(\beta-\beta')\|_2^2 \geq \kappa_l\|\beta-\beta'\|_2^2 - \tau_l\frac{\log p}{n}\|\beta-\beta'\|_1^2$ for all $\beta, \beta' \in \mathbb{R}^p$
$\|\beta-\beta^*\|_2 \lesssim C\sqrt{\frac{k\log p}{n}}$	$\|\beta-\beta^*\|_2 \lesssim C\sqrt{\frac{\log p}{n}}\left(\frac{\sqrt{k}}{2}+\sqrt{k-h}\right)$ $\quad h < k$ $\|\beta-\beta^*\|_2 \lesssim C\sqrt{\frac{h\log p}{n}}$ $\quad h \geq k$

Sequential Transfer in Multi-armed Bandit with Finite Set of Models

Mohammad Gheshlaghi Azar *
School of Computer Science
CMU

Alessandro Lazaric †
INRIA Lille - Nord Europe
Team SequeL

Emma Brunskill *
School of Computer Science
CMU

Presenters: Audrey Huang & Naveen Shankar

Problem Setting

In each episode $j = 1, \ldots, J$ we interact with unknown task $\theta \in \Theta$

- Finite set of tasks $\Theta = \{\theta_1, \ldots, \theta_M\}$
- Each $\theta \in \Theta$ is a K-armed bandit specified by $\theta = \{\theta^1, \ldots, \theta^K\}$

Objective: minimize cumulative regret over tasks

Problem Setting

In each episode $j = 1, \ldots, J$ we interact with unknown task $\theta \in \Theta$

- Finite set of tasks $\Theta = \{\theta_1, \ldots, \theta_M\}$
- Each $\theta \in \Theta$ is a K-armed bandit specified by $\theta = \{\theta^1, \ldots, \theta^K\}$

Objective: minimize cumulative regret over tasks

We present 2 algorithms which achieve better performance than UCB:

- **mUCB**: Θ is known
- **tUCB**: Θ is unknown

Notation

$\mu_i(\theta)$ - mean reward of arm *i* in task θ

$\mu_*(\theta)$ - mean reward of optimal arm in task θ

$i^*(\theta)$ - optimal arm of task θ

$\Delta_i(\theta)$ - optimality gap of arm *i* in task θ

$\gamma_i(\theta, \theta')$ - difference in arm *i* reward between tasks θ, θ'

mUCB

Assume we know **Θ** a priori, and have unknown task $\theta \in \Theta$.

Input: models Θ and timesteps T

mUCB

Assume set of tasks **Θ** is known, and have unknown task $\theta \in \Theta$.

Input: models *Θ* and timesteps *T*

For *t = 1, …, T*:

Compute estimates $\hat{\mu}_{i,t}$ of each arm *i*

Compute confidence band for each arm $\epsilon_{i,t} \propto \frac{1}{\#\text{ times arm } i \text{ has been pulled}}$

mUCB

Assume set of tasks **Θ** is known, and have unknown task $\theta \in \Theta$.

Input: models *Θ* and timesteps *T*

For *t = 1, …, T*:

Compute estimates $\hat{\mu}_{i,t}$ of each arm *i*

Compute confidence band for each arm $\epsilon_{i,t} \propto \frac{1}{\#\text{ times arm } i \text{ has been pulled}}$

Build set of compatible models $\Theta_t = \{\theta : \forall i, |\mu_i(\theta) - \hat{\mu}_{i,t}| \leq \varepsilon_{i,t}\}$

mUCB

Assume set of tasks **Θ** is known, and have unknown task $\theta \in \Theta$.

Input: models *Θ* and timesteps *T*

For *t = 1, …, T*:

Compute estimates $\hat{\mu}_{i,t}$ of each arm *i*

Compute confidence band for each arm $\epsilon_{i,t} \propto \frac{1}{\#\text{ times arm } i \text{ has been pulled}}$

Build set of compatible models $\Theta_t = \{\theta : \forall i, |\mu_i(\theta) - \hat{\mu}_{i,t}| \leq \varepsilon_{i,t}\}$

Let $\theta_t = \arg\max_{\theta \in \Theta_t} \mu_*(\theta)$ be the task with highest overall reward

mUCB

Assume set of tasks **Θ** is known, and have unknown task $\theta \in \Theta$.

Input: models *Θ* and timesteps *T*

For *t = 1, …, T*:

Compute estimates $\hat{\mu}_{i,t}$ of each arm *i*

Compute confidence band for each arm $\epsilon_{i,t} \propto \frac{1}{\#\text{ times arm } i \text{ has been pulled}}$

Build set of compatible models $\Theta_t = \{\theta : \forall i, |\mu_i(\theta) - \hat{\mu}_{i,t}| \leq \varepsilon_{i,t}\}$

Let $\theta_t = \arg\max_{\theta \in \Theta_t} \mu_*(\theta)$ be the task with highest overall reward

Pull $i^*(\theta_t)$ and observe reward

mUCB: Regret

$$\mathbb{E}[R_{UCB}(T,\theta)] \leq O(\frac{\log T}{\min_i \Delta_i(\theta)})$$

$$\mathbb{E}[R_{mUCB}(T,\theta)] \leq O(\frac{\log T}{\min_i \min_{\theta' \in \Theta_+} \gamma_i(\theta,\theta')})$$

where $\Theta_+ = \{\theta' \in \Theta : \mu_*(\theta') \geq \mu_*(\theta)\}$

mUCB: Regret

$$\mathbb{E}[R_{UCB}(T,\theta)] \leq O(\frac{\log T}{\min_i \Delta_i(\theta)})$$

$$\mathbb{E}[R_{mUCB}(T,\theta)] \leq O(\frac{\log T}{\min_i \min_{\theta' \in \Theta_+} \gamma_i(\theta,\theta')})$$

where $\Theta_+ = \{\theta' \in \Theta : \mu_*(\theta') \geq \mu_*(\theta)\}$

For all arms *i*, arm gap is at least the optimality gap:

$$\min_{\theta' \in \Theta_+} \gamma_i(\theta,\theta') \geq \Delta_i(\theta)$$

→ mUCB has better upper bound

tUCB

We do not know set of tasks **Θ**, and for each episode *j* have unknown $\theta_j \in \Theta$.

Input: # tasks *M*, # arms *K*, episodes *J***,** timesteps *T*

For episodes *j = 1, …, J*:

Compute MoM estimate of tasks so far $\Theta_j = \{\theta_1, \ldots, \theta_j\}$ using **Robust Tensor Power Method** (Anandkumar et al. 2013)

Run **mUCB** on estimated models Θ_j

tUCB: Regret

- Cumulative pseudoregret is never worse than UCB
- After a certain episode *j*, tUCB has the same performance as mUCB and improvement over UCB

mUCB and tUCB Are Never Worse than UCB

	Arm1	Arm2	Arm3	Arm4	Arm5	Arm6	Arm7
θ_1	0.9	0.75	0.45	0.55	0.58	0.61	0.65
θ_2	0.75	0.89	0.45	0.55	0.58	0.61	0.65
θ_3	0.2	0.23	0.45	0.35	0.3	0.18	0.25
θ_4	0.34	0.31	0.45	0.725	0.33	0.37	0.47
θ_5	0.6	0.5	0.45	0.35	0.95	0.9	0.8

Table 1: Models.

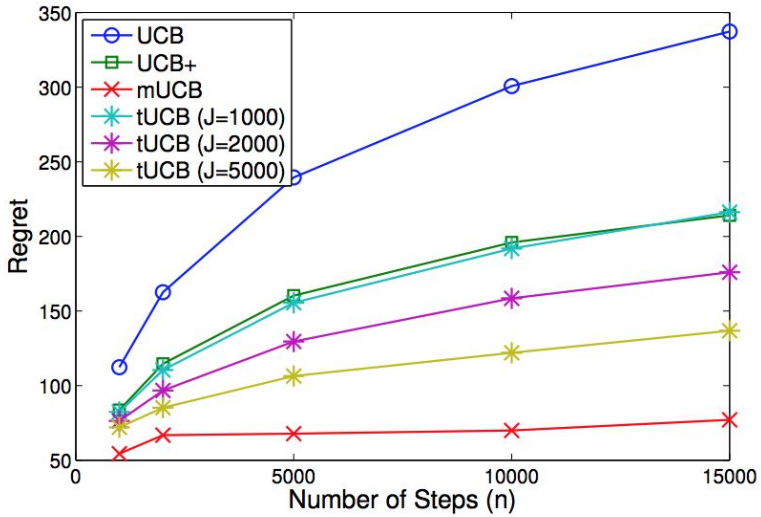


Figure 7: Regret of *UCB*, *UCB+*, *mUCB*, and *tUCB* (avg. over episodes) vs episode length.

Regret of tUCB approaches mUCB

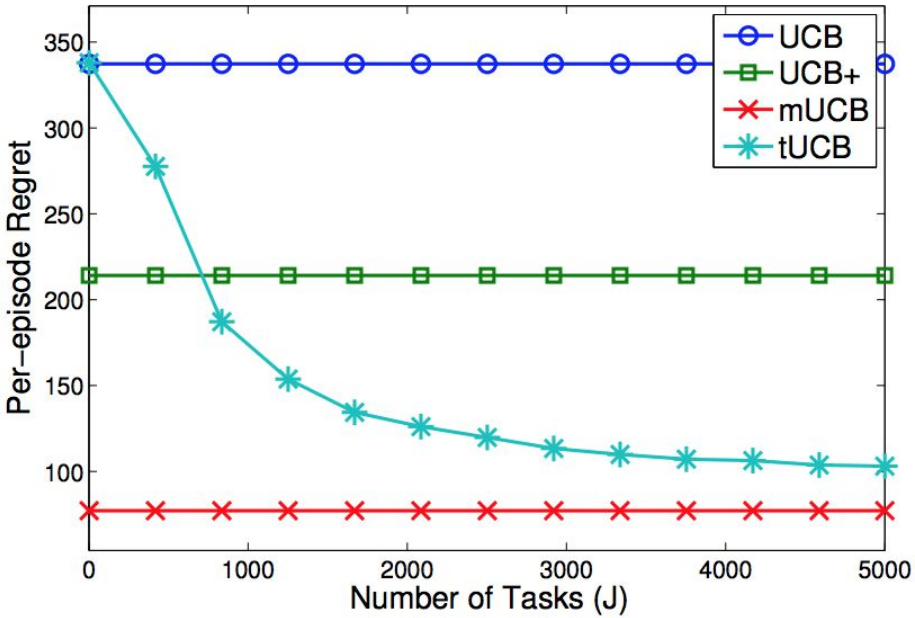


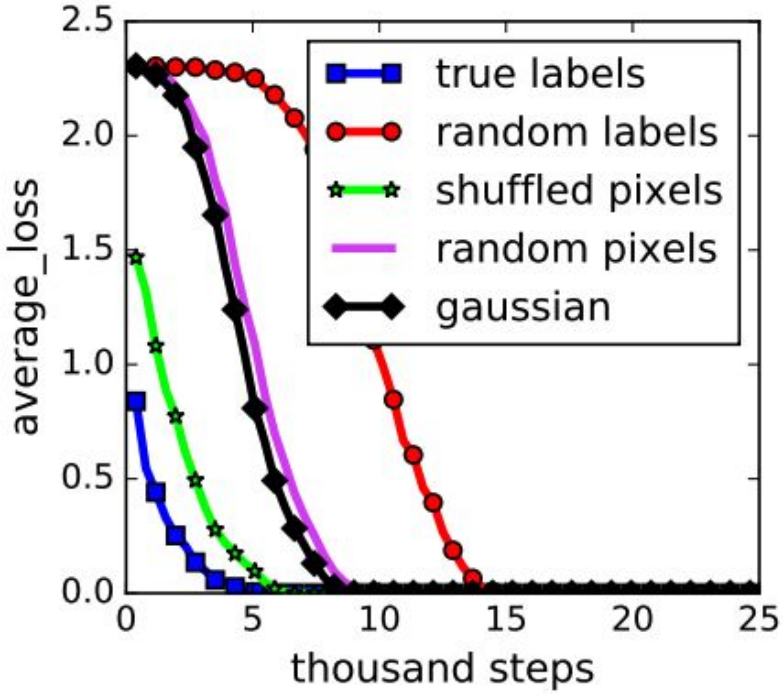
Figure 8: Per-episode regret of *tUCB*.

Generalizability of Interpolating Schemes

10716 Project

Danlei Zhu

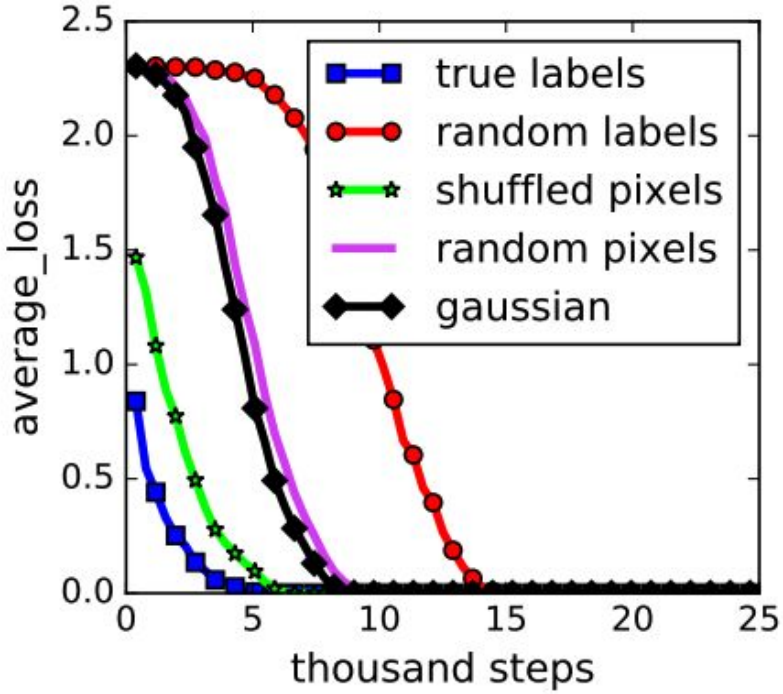
Introduction



model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61

Source: Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Introduction



model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61

Source: Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Uniform Bounds

Given n independent samples (x_i, y_i) (from some distribution P) and f* be the optimal bayes classifier, we have below the uniform bound

$$\mathbb{E}(L(f^*, y)) \leq \frac{1}{n}\sum_{i=1}^{n} L(f^*(\mathbf{x}_i), y_i) + \mathcal{O}^*(\sqrt{\frac{c}{n}})$$

Uniform Bounds

$$\mathbb{E}(L(f^*, y)) \leq \frac{1}{n}\sum_{i=1}^{n} L(f^*(\mathbf{x}_i), y_i) + \mathcal{O}^*(\sqrt{\frac{c}{n}})$$

Test loss

Training loss

Model complexity:
VC dimension,
Rademacher etc

Validity of some kernels

Given Nadaraya-Watson estimator (Nadaraya, 1964)

$$f_n(x) = \frac{\sum_{i=1}^{n} Y_i K\left(\frac{x-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right)}$$

Where K is singular kernel such as

$$K(u) = ||u||^{-a} I\{||u|| \leq 1\}$$

Validity of some kernels

(Belkin et al, 2018) MSE decays to 0 at a rate of $n^{-\frac{2\beta}{2\beta+d}}$

$$\mathbb{E}\,\|f_n - f\|^2_{L_2(P_X)} \triangleq \mathbb{E}(f_n(X) - f(X))^2 \leq C n^{-\frac{2\beta}{2\beta+d}}$$

Belkin, M., Rakhlin, A., & Tsybakov, A. B. (2018). Does data interpolation contradict statistical optimality?. *arXiv preprint arXiv:1806.09471*.

Validity of some kernels

(Belkin et al, 2018)

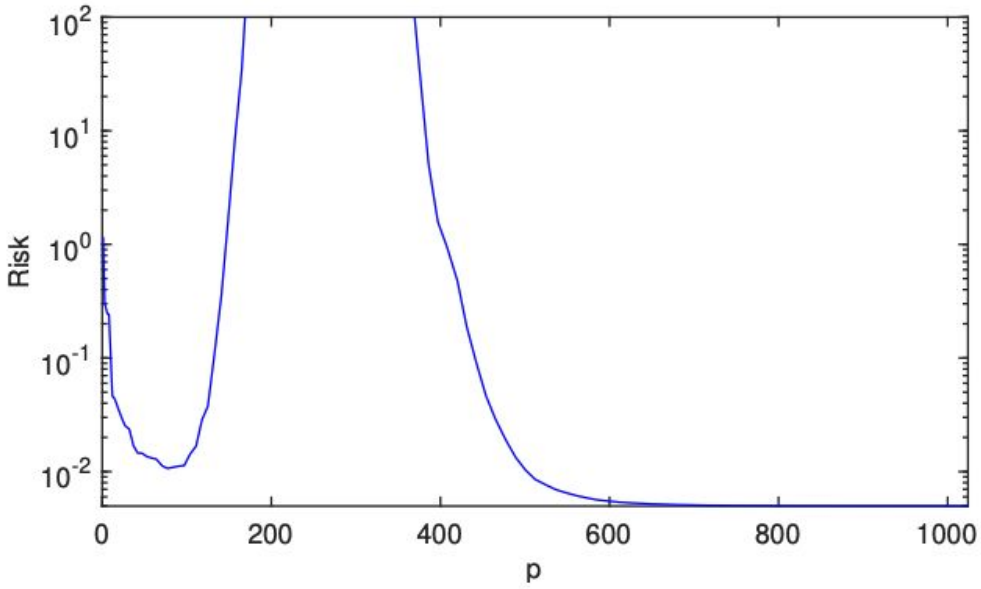
- Asymptotic MSE for interpolating 1-Nearest-Neighbor with triangulation scales like O(1/d), d is the dimension of data. (asymptotic excess risk for classification is exponentially small in d)
- Weighted & Interpolated Nearest Neighbor(WiNN) MSE converges at rate $n^{-2\alpha/(2\alpha+\bar{d})}$

Belkin, M., Hsu, D. J., & Mitra, P. (2018). Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In *Advances in neural information processing systems* (pp. 2300-2311).

What about interpolating general methods?

1. What does empirical knowledge tell us?
2. Dependence of generalization on model complexity?

Double Descent Curve Dependence of generalization on model complexity?



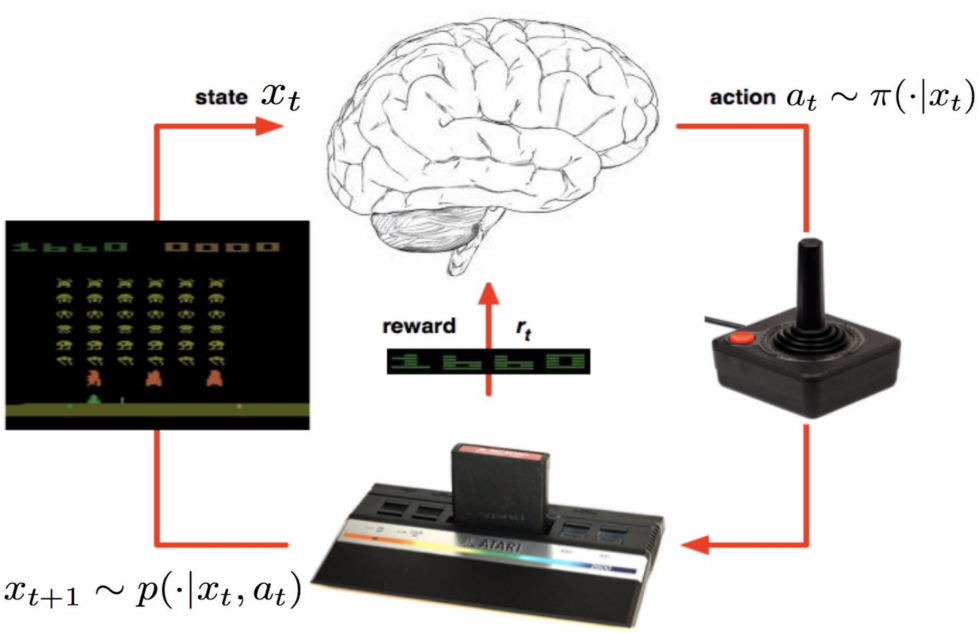
Belkin, M., Hsu, D., & Xu, J. (2019). Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*.

Distributional Reinforcement Learning

By Tianwei Ni, Shangbang Long

Apr 30, 2020

Conventional Q learning



Value function Q^π:

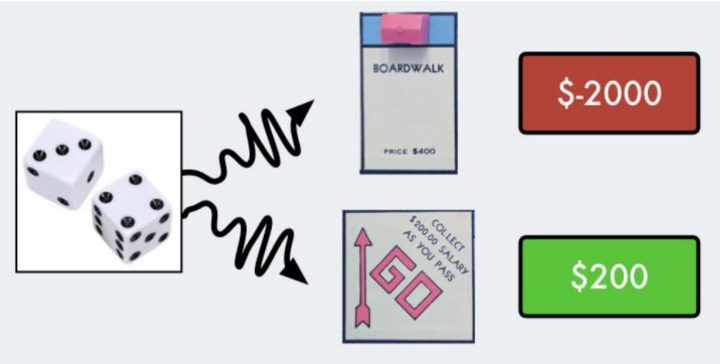
the **expected** sum of future discounted rewards induced by the policy π

$$Q^\pi(x, a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t \middle| x, a, \pi\right]$$

Intrinsic Problem?

Problem of value function: Expectation

- Consider a simple dice game:



Random variable reward:

$$R(x) = \begin{cases} -2000 \text{ w.p. } 1/36 \\ 200 \text{ w.p. } 35/36 \end{cases}$$

$$\mathbb{E}[R(x)] = \frac{1}{36} \times (-2000) + \frac{35}{36} \times (200) = 138.88$$

- Return is actually a **random variable**, may be **multi-modal**
- Using expectation (i.e. value function) is not fully representative

Stochasticity in Value Distribution

Consider **value distribution** instead:

$$Z^\pi(x,a) = \sum_{t\geq 0} \gamma^t r(x_t, a_t)\Big|_{x_0=x, a_0=a, \pi} \qquad Q^\pi(x,a) = \mathbb{E}[Z^\pi(x,a)]$$

Randomness:

- Immediate reward
- Stochastic dynamics $x' \sim p(\cdot|x,a)$
- Stochastic policy $a' \sim \pi(\cdot|x')$

Distributional Q learning

- **Distributional** Bellman Equation for value *distribution* Z:

$$Z^\pi(x,a) \stackrel{D}{=} R(x,a) + \gamma Z^\pi(x',a')$$

$$\text{where } x' \sim p(\cdot|x,a) \text{ and } a' \sim \pi(\cdot|x')$$

-

- **Distributional** Bellman Operator

Contraction?

$$T^\pi Z(x,a) = R(x,a) + \gamma Z(x',a')$$

- Recall: *Traditional* Bellman Equation for value function

$$Q^\pi(x,a) = r(x,a) + \gamma \mathbb{E}_{x'}\left[\sum_{a'} \pi(a'|x') Q^\pi(x',a') \middle| x,a\right]$$

Theories of Contraction (I) - Wasserstein Metric

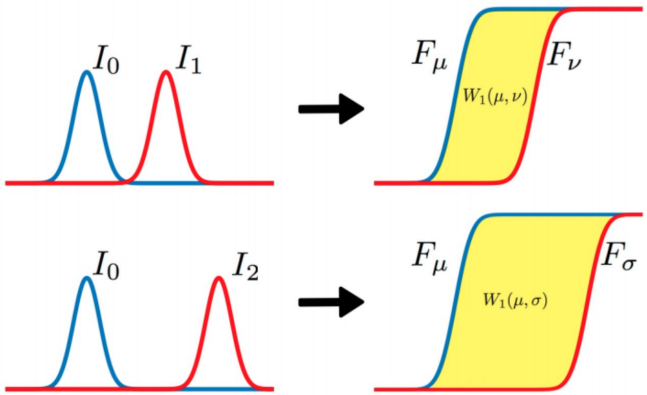
To study contraction, we introduce the **maximal** form of Wasserstein metric:

$$\bar{d}_p(Z_1, Z_2) := \sup_{x,a} d_p(Z_1(x,a), Z_2(x,a))$$

where Wasserstein distance in 1D distribution by inverse CDF:

$$d_p(F_U, F_V) = \left(\int_0^1 |F_U^{-1}(\omega) - F_V^{-1}(\omega) d\omega \right)^{1/p}$$

$$F_U^{-1}(\omega) := \inf\{x \in \mathbb{R} : \omega \leq F_U(x)\}$$



Theories of Contraction (II) - Policy Evaluation

Recall: Distributional Bellman operator $\mathcal{T}^\pi : \mathcal{Z} \to \mathcal{Z}$

$$T^\pi Z(x,a) = R(x,a) + \gamma Z(x',a')$$

Theorem: it's a **contraction** in maximal form of Wasserstein metric. [1]

Iterate $Z \leftarrow T^\pi Z$ will converge to unique fixed point Z^π

Hints of proof: discount rate **shrinks** the support size

$$\begin{aligned} d_p(aU, aV) &\leq |a| d_p(U,V) \\ d_p(A+U, A+V) &\leq d_p(U,V) \\ d_p(AU, AV) &\leq \|A\|_p d_p(U,V) \end{aligned}$$

Theories of Contraction (III) - Control

Distributional Bellman **Optimality** operator:

$$TZ(x,a) \stackrel{D}{=} r(x,a) + \gamma Z(x', \pi_Z(x'))$$

$$\text{where } x' \sim p(\cdot|x,a) \text{ and } \pi_Z(x') = \arg\max_{a'} \mathbb{E}[Z(x',a')]$$

Theorem: If optimal policy is **unique**, then iterate $Z_{k+1} \leftarrow TZ_k$ converge to Z^{π^*}

Proposition: Optimality operator is **not** a contraction. [1]

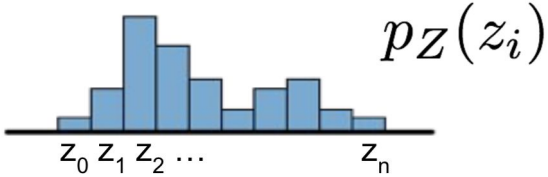
Intuition: Optimality operator preserves the **mean** Q^*, but in general there exist *many* optimal value distributions.

Approximate Distributional RL

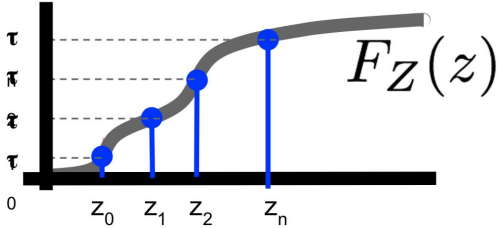
How to represent value distributions?

How to apply it to Deep RL?

- Categorical Distribution:
  - **Categorical DQN [1]**

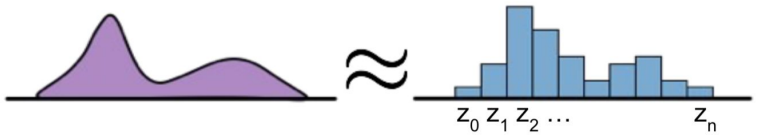


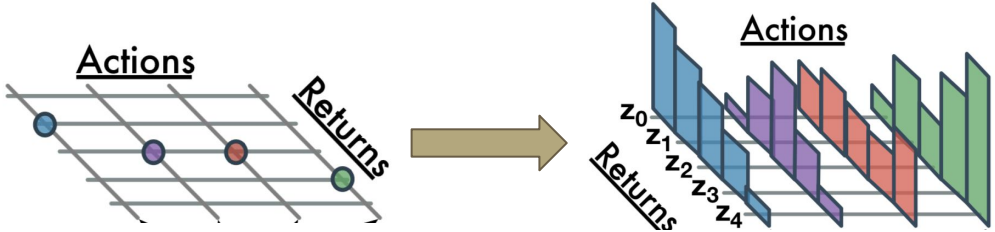
- Inverse CDF for specific quantiles:
  - **Quantile Regression DQN [2]**

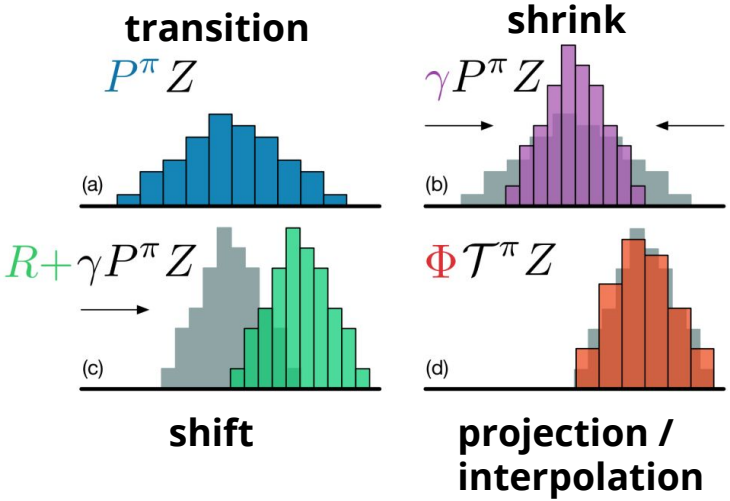


Categorical DQN

- Categorical Distribution
  - Fixed support
- Project target value distribution onto the support
  - By linear interpolation
- Contraction in Cramer distance [3]
- From DQN to Categorical DQN:
  - Difference in network's output

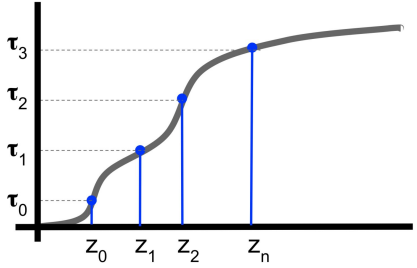




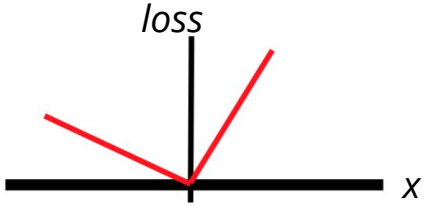


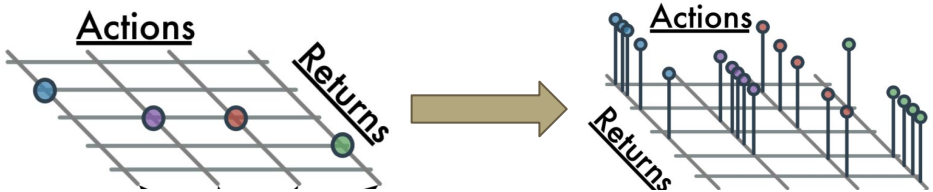
Quantile Regression DQN

- From Categorical to Quantile:
  - Cat: Fixed Support and Learned Probabilities
  - Quant: Fixed Probabilities and Learned Support
- Use *quantile regression loss* to learn many quantiles
- Contraction in Wasserstein distance [2]
- From DQN to Quantile Regression DQN:
  - Difference in network's output



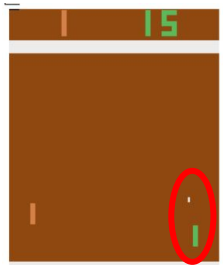
$$loss = \begin{cases} \tau x, & \text{for } x \geq 0 \\ (\tau - 1)x, & \text{for } x \geq 0 \end{cases}$$

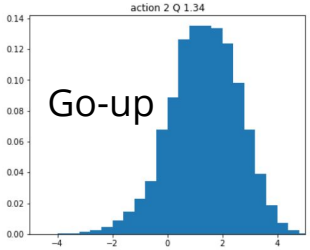


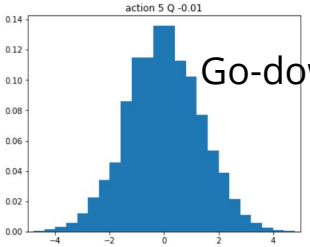


Visualization of value distribution

Pong: about to hit the ball

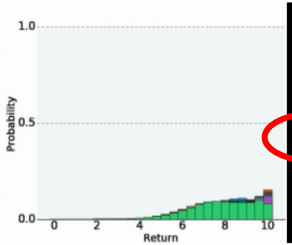




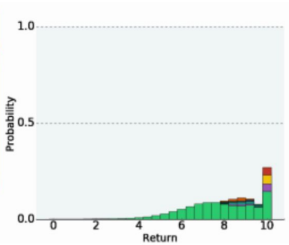


Go-up has *left-skewed* value distribution (high confidence) apart from larger mean.

Seaquest: about to hit the fish



uncertain



affirmative

How to utilize value distribution?

- We only use the **mean** of value distribution finally?
- Distributional RL has some non-trivial advantages:
  - Richer representation
  - Training: lower-variance gradient
  - Risk-aware: risk-averse/seeking agent
  - Exploration: uncertainty

References

1. Marc G Bellemare, Will Dabney, and Rémi Munos. *A distributional perspective on reinforcement learning*. ICML 2017.
2. Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. *Distributional reinforcement learning with quantile regression*. AAAI 2018.
3. Mark Rowland, Marc G Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. *An analysis of categorical distributional reinforcement learning*. AISTATS 2018.
4. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
5. Distributional Reinforcement Learning. Rémi Munos. Talk at PGMO workshop, 2019.

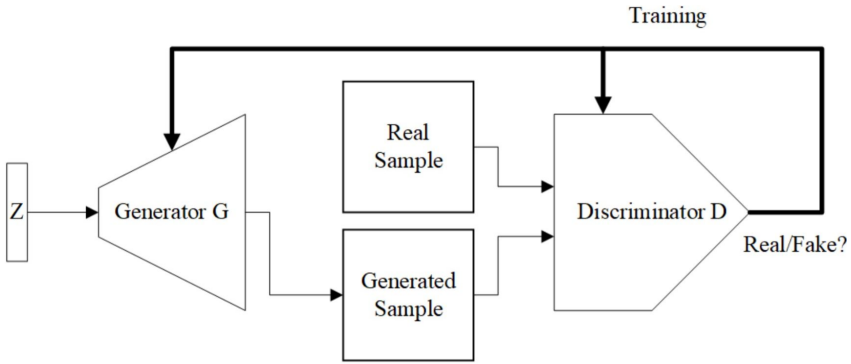
Thanks!

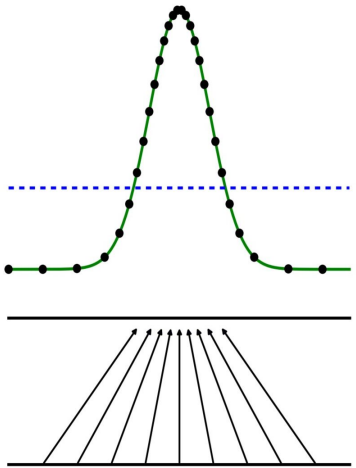
GAN Theory and Objectives

Andrew Luo, Matthew Ho

10-716 - Advanced Machine Learning Theory and Methods

Problem Overview





- Consists of Generator ***(G)*** and Discriminator ***(D)***
- Generator transforms latent z into data space $\mathcal{X}$
- Discriminator trained to provide guidance to ***G***
- Adversarial training can be highly unstable
  - Vanishing gradients
  - Mode collapse
  - Memorization

Goal of a generator is to model **true distribution (black dotted)** with a **generated distribution (green)** by transforming a **latent (black solid)**

Image src:
Paired 3D Model Generation with Conditional Generative Adversarial Networks (left figure of GAN)
Generative Adversarial Nets (right figure of distribution matching)

Original GAN objective (Goodfellow 2014)

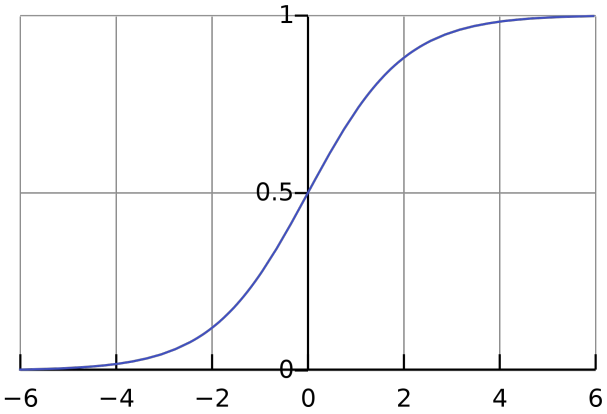
$$\min_G \max_D \{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]\}$$

- Suffers from vanishing gradient when discriminator is too good / generator is poor
- Obvious when we take <u>derivative with respect to generator weights</u>

$$\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\left(\frac{-D'(G(\mathbf{z}))}{1 - D(G(\mathbf{z}))}\right)\nabla_{\theta_G} G(\mathbf{z})\right]$$

- Optimal Discriminator:

$$D^*(x) = \frac{p_r(x)}{p_g(x) + p_r(x)}$$



Modified GAN objective (Goodfellow 2014)

- Discriminator uses the original loss:

$$\max_D \{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]\}$$

- Generator modified to maximize:

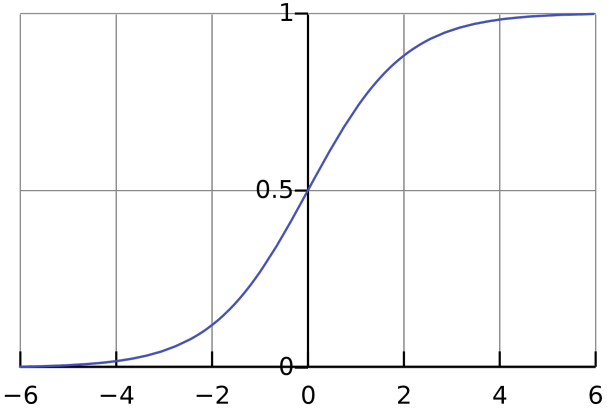
$$\max_G \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(D(G(\boldsymbol{z})))]$$

- Slightly better derivative (when generator is poor). Still not ideal.

$$\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left[\left(\frac{D'(G(\mathbf{z}))}{D(G(\mathbf{z}))}\right)\nabla_{\theta_G} G(\mathbf{z})\right]$$

- Optimal Discriminator:

$$D^*(x) = \frac{p_r(x)}{p_g(x) + p_r(x)}$$



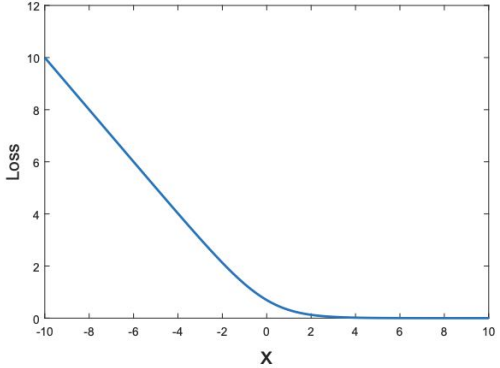
Least Squares GAN (Mao 2017)

$$\min_D \left\{ \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[(D(\boldsymbol{x})-1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\left[(D(G(\boldsymbol{z})))^2\right] \right\}$$

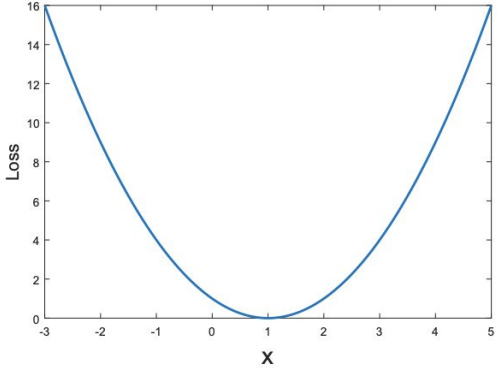
$$\min_G \left\{ \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\left[(D(G(\boldsymbol{z}))-1)^2\right] \right\}$$

- Penalizes data which are on the correct side of, but far from the discriminator boundary

Sigmoid Cross-entropy



Least Squares



$$\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\left[(D(G(\boldsymbol{z}))-1)D'(G(\mathbf{z}))\nabla_{\theta_G}G(\mathbf{z})\right]$$

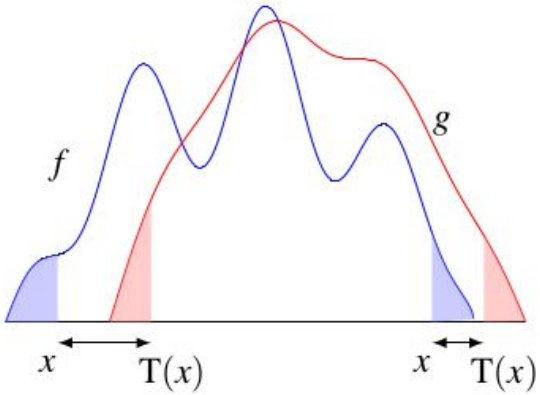
Wasserstein GAN (Arjovsky 2017)

$$\max_D \{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[D(G(\boldsymbol{z}))]\}$$

$$\max_G \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[D(G(\boldsymbol{z}))]$$

+ weight clipping

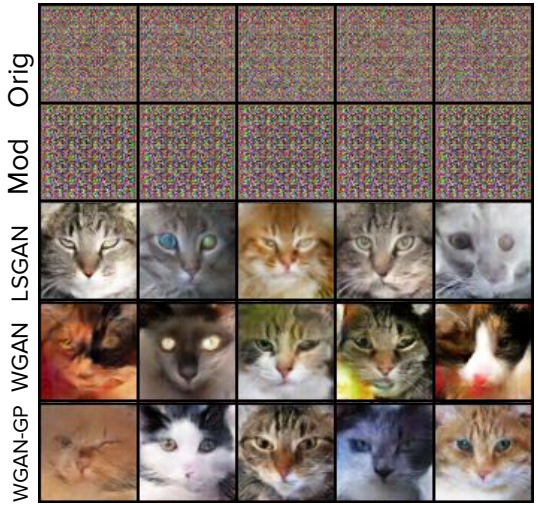
- Implements Wasserstein metric to measure discriminator performance
- Includes <u>weight-clipping</u> to keep parameter space compact and keep discriminator space Lipschitz
- Improves cases of mode collapse
- Well-defined derivatives solve vanishing gradients



WGAN-GP (Gulrajani 2017) adds weight norm penalty to replace weight clipping.

Empirical results and Conclusions

- Convolution GAN trained on 9k images of aligned cat images
- Original (Goodfellow 2014), Modified (Goodfellow 2014), LSGAN, WGAN, WGAN-GP
- 2e-4 lr used with Adam optimizer for first 3; 1e-4 used for WGAN & WGAN-GP
- Weights clipped to [-0.01, +0.01] for WGAN, λ_{GP}= 1 for WGAN-GP



	Orignal	Modified	LSGAN	WGAN	WGAN-GP
Time to train	37 min	37 min	38 min	38 min	59 min
FID score	376.6	397.9	21.35	24.74	27.57

References

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

Mao, Xudong, et al. "Least squares generative adversarial networks." *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein gan." *arXiv preprint arXiv:1701.07875* (2017).

Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." *Advances in neural information processing systems*. 2017.

Lucic, Mario, et al. "Are gans created equal? a large-scale study." *Advances in neural information processing systems*. 2018.

Dong, Hao-Wen, and Yi-Hsuan Yang. "Towards a deeper understanding of adversarial losses." *arXiv preprint arXiv:1901.08753* (2019).

Meta-(supervised)-learning through a probabilistic lens

Oscar Li

CMU MLD

April 27, 2020

Meta-learning problem formulation

Goal: Learn a general algorithm that can learn predictive models for a range of tasks (Learning to learn)

- distribution over tasks $\mathcal{P}(\mathcal{T})$
- each task $\mathcal{T}$ with data generating distribution $p_{\mathcal{T}}$ over $\mathcal{X} \times \mathcal{Y}$ is represented by a task-specific training and test set pair $(D_{\mathcal{T}}^{\mathrm{Tr}}, D_{\mathcal{T}}^{\mathrm{Tr}})$.
- loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.
- an algorithm $A : \{D^{\mathrm{Tr}}\} \to \mathcal{M}$ maps a task's training set to a predictive model in the model space
- meta-learning as finding a good algorithm through optimization:

$$\hat{A} \in \arg\min_{A \in \mathcal{A}} \sum_{t=1}^{T} \frac{1}{|D_t^{\mathrm{Te}}|} \sum_{(x^{\mathrm{Te}}, y^{\mathrm{Te}}) \in D_t^{\mathrm{Te}}} L(\underbrace{\underbrace{A(D_t^{\mathrm{Tr}})}_{\text{a model}}(x^{\mathrm{Te}})}_{\text{a prediction}}, y^{\mathrm{Te}}).$$

Going probabilistic

$$\hat{A} \in \arg\min_{A \in \mathcal{A}} \sum_{t=1}^{T} \frac{1}{|D_t^{\mathrm{Te}}|} \sum_{(x^{\mathrm{Te}}, y^{\mathrm{Te}}) \in D_t^{\mathrm{Te}}} L(\underbrace{\underbrace{A(D_t^{\mathrm{Tr}})}_{\text{a model}}(x^{\mathrm{Te}})}_{\text{a prediction}}, y^{\mathrm{Te}}).$$

Potential Problems: When $|D_t^{\mathrm{Tr}}|$ is small, algorithm A only produces **one** model estimate despite a high level of **uncertainty** in many possible model explanations for the training data D_t^{Tr}.

Solution:
Make A a stochastic algorithm: given a training set D^{Tr}, A returns models sampled from a distribution over the model space $\mathcal{M}$.

Now this sounds like running a posterior inference:
sample models conditioned on the observations (D^{Tr})!

Probabilistic setup [1]
parameterized model class $\mathcal{M} = \{m_\phi : \phi \in \Phi\}$

- First, for each task t, sample the true task model parameter ϕ_t from a prior distribution $p(\cdot; \theta)$
- Then, generate the task specific training D_t^{Tr} and test set D_t^{Te} by sampling inputs from a distribution over $\mathcal{X}$ and labelling them with model m_{ϕ_t}

This is a latent variable problem because the underlying model ϕ_t is never observed. We want to infer ϕ_t.

Naive Goal: find $p(\phi_t | D_t^{\mathrm{Tr}})$ for each t through variational inference

- too many tasks to be scalable
- cannot use for a new task

Useful Goal: by amortized variational inference, find a parameterized function mapping from a task's training set D^{Tr} to the task's approximate posterior distribution $p(\phi | D^{\mathrm{Tr}})$ (**very similar to** what the stochastic algorithm A mentioned before)

Examples of Amortized VI approaches

- parametric approximate posterior distribution [2]
  - let the approximate posterior distributions be chosen from the class of Gaussian distributions
  - find the amortized posterior mapping function $f : \{D^{\mathrm{Tr}}\} \to \mathcal{P}_\Phi$ through optimizing ELBO

$$\max_f \sum_{t=1}^{T} \left[E_{\phi \sim f(D^{\mathrm{Tr}})} \log p(D_t^{\mathrm{Te}}, D_t^{\mathrm{Tr}} | \phi)] - \mathrm{KL}(f(D_t^{\mathrm{Tr}}) \parallel p(\cdot; \theta)) \right]$$

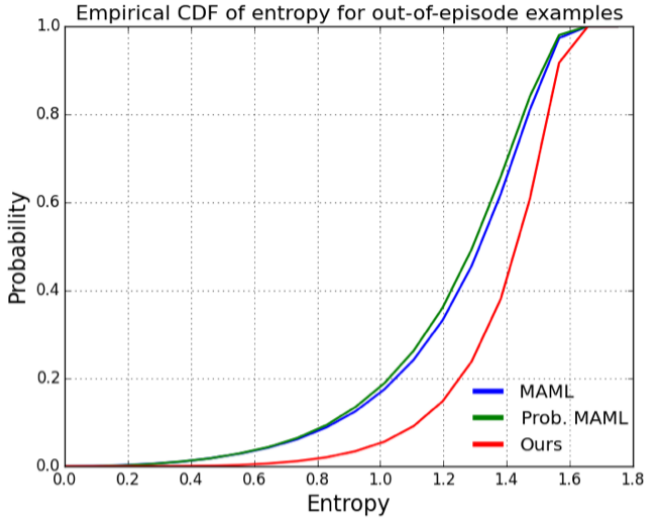
- nonparametric approximate posterior distribution [3]
  - Use n-step Stein Variational Gradient Descent (SVGD)[4] with kernel function $k : \Phi \times \Phi \to \mathbb{R}$ to produce samples from the posterior distribution conditioned on D_t^{Tr}
    - initialize M particles $\{\psi_m^{(0)}\}_{m=1}^M \subset \Phi$ in the parameter space
    - run SVGD for n steps:

$$\psi_m^{(i+1)} \leftarrow \psi_m^{(i)} + \epsilon_i \left[\frac{1}{M} \sum_{j=1}^{M} k(\psi_j^{(i)}, \psi_m^{(i)}) \nabla_{\phi_t} p(\phi_t = \psi_j^{(i)} | D_t^{\mathrm{Tr}}) + \nabla_{\psi_j^{(i)}} k(\psi_j^{(i)}, \psi_m^{(i)}) \right], \forall m \in [M]$$

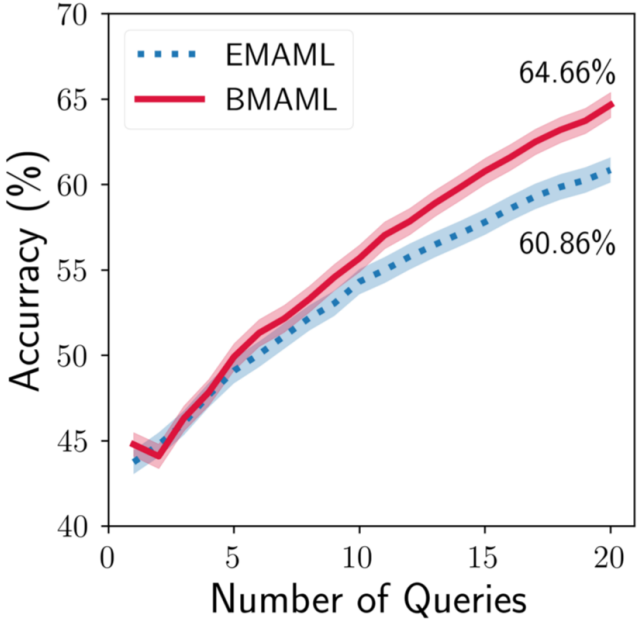
  - no parametric form for the approximate posterior (directly produces samples from it)
  - learn the particle initializations $\{\psi_m^{(0)}\}_{m=1}^M$ through a novel *Chaser loss*

Benefits gained

- the posterior distribution gives us uncertainty estimation



- allows us to conduct better active learning



References

[1] Grant, E., Finn, C., Levine, S., Darrell, T., & Griffiths, T. (2018). Recasting gradient-based meta-learning as hierarchical bayes. arXiv preprint arXiv:1801.08930.

[2] Ravi, S., & Beatson, A. (2018). Amortized bayesian meta-learning.

[3] Kim, T., Yoon, J., Dia, O., Kim, S., Bengio, Y., & Ahn, S. (2018). Bayesian model-agnostic meta-learning. arXiv preprint arXiv:1806.03836.

[4] Liu, Q., & Wang, D. (2016). Stein variational gradient descent: A general purpose bayesian inference algorithm. In Advances in neural information processing systems (pp. 2378-2386).

Verifiable Random Forests vs. Neural Networks in Safety Critical Systems

Jack H. Good

jhgood@andrew.cmu.edu

April 30, 2020

Introduction and Motivation

- Neural networks can display unpredictable behavior, which is problematic in safety-critical applications.
  - Lack of adversarial robustness
  - Inability to verify requirements
- SMT techniques are used to solve these challenges, but they:
  - do not scale well
  - often require approximations or highly restrictive assumptions
  - do not support non-linear arithmetic (e.g. sigmoid activations)
- Many approaches to improve performance are being explored [1] [3] [4] [5] [6].
- Tree based models, such as random forests, are also applied in safety-critical applications, but their formal verification is less studied.
- Trees and voting-based tree ensembles can be represented with SAT only.

Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks [6]

Extends the simplex algorithm to support the non-convex ReLUs.

Key results:

- Reluplex is much faster than vanillia SMT for verifying DNNs
- Reluplex is sound and complete
  (others are not, or assume strong conditions)
- Reluplex guarantees termination
- Verifying properties of DNNs with ReLUs is NP-Complete

Reluplex Experiments on ACAS-Xu

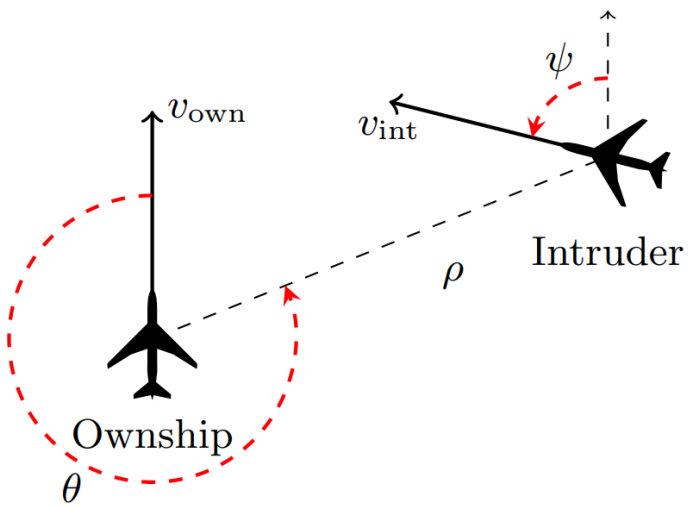
ACAS-Xu: Aircraft Collision Avoidance System (unmanned).
$\sim$ 2GB lookup table obtained by solving MDP on a discrete space.

7 inputs:

- distance to intruder
- angle to intruder
- heading of intruder
- ownship speed
- intruder speed
- time until loss of vertical separation
- previous advisory

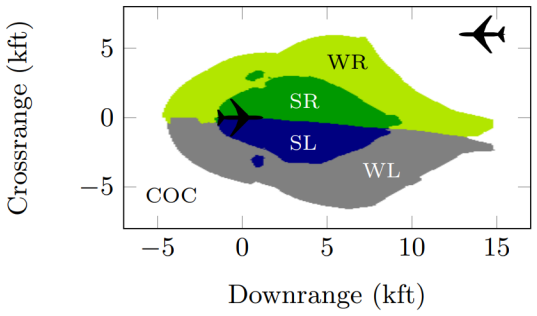
5 outputs:

- clear of conflict
- weak right
- strong right
- weak left
- strong left



Reluplex Experiments on ACAS-Xu

Lookup table (2GB) is too large; compress with neural networks.



Advisories for a head-on encounter with $a_{\text{prev}} = \text{COC}, \tau = 0\,\text{s}$.

45 networks, 6 hidden layers & 300 ReLU each, $\sim$ 3MB.

Use Reluplex to verify or find counterexamples to:

- several policy design specifications

  e.g. "If the intruder is directly ahead and is moving towards the ownship, the score for COC will not be minimal."

- pointwise adversarial robustness

  NN is δ-locally-robust at $\mathbf{x}$ iff $\forall \mathbf{x}', \|\mathbf{x} - \mathbf{x}'\|_\infty \leq \delta \implies$ same prediction.

Verification of Tree-based Models

Discrete decision logic means we can use SAT instead of SMT to accomplish the same verification tasks. This is **much** more scalable than SMT.

Because trees are less prone to unpredictable behavior, their formal verification is understudied; however, verification and adversarial example generation are often desirable [2].

We offer a SAT framework for verifying tree-based models. Currently, only vote-based ensembles are supported.

MAX-SAT offers additional capabilities.

Repeating the Reluplex Experiments with Trees and SAT

Case study: ACAS-Xu experiments for Reluplex neural net (3MB), tree classifier (8MB), and tree regressor (34MB, WIP).

Time in seconds to verify ten properties:

Source	Prop. 1	Prop. 2	Prop. 3	Prop. 4	Prop. 5
Reluplex	`TIMEOUT`	82882	28156	12475	19355
Tree, classifier	N/A	N/A	0.0003	0.9423	4.015

Source	Prop. 6	Prop. 7	Prop. 8	Prop. 9	Prop. 10
Reluplex	180288	`TIMEOUT`	40102	99634	19944
Tree, classifier	4.447	3.399	4.060	4.455	0.0002

Time to find or disprove adversarial example (5 trials, $\delta = 0.1$):

Source	min	median	max
Reluplex	2	863	14560
Tree, classifier	3.523	3.701	3.863

Conclusions

References

Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi.
Measuring neural net robustness with constraints.
In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2613–2621. Curran Associates, Inc., 2016.

Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh.
Robust decision trees against adversarial examples, 2019.

Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A. Mann, and Pushmeet Kohli.
A dual approach to scalable verification of deep networks.
In *UAI*, 2018.

Rüdiger Ehlers.
Formal verification of piece-wise linear feed-forward neural networks.
In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis*, pages 269–286, Cham, 2017. Springer International Publishing.

Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu.
Safety verification of deep neural networks.
In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 3–29, Cham, 2017. Springer International Publishing.

Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer.
Reluplex: An efficient smt solver for verifying deep neural networks.
In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 97–117, Cham, 2017. Springer International Publishing.

Maximum Mean Discrepancy for Unsupervised Domain Adaptation

Saswati Ray

saswatir@andrew.cmu.edu

Unsupervised Domain Adaptation

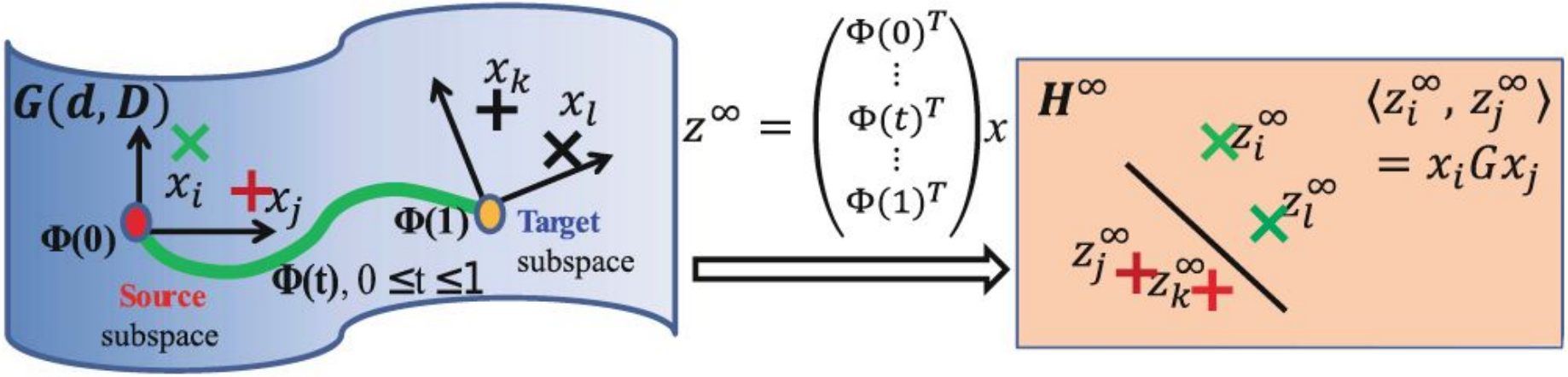
Problem: Build robust models using source data for mismatched source (labeled) and target (unlabeled) distributions.

Unsupervised Domain Adaptation

How do we achieve this?

Learn models in a domain-invariant feature space using geodesic flow kernel (GFK) and source dataset landmarks (most useful instances)

Geodesic Flow Kernel (GFK)



P_S, P_T : Basis of PCA subspaces for source and target datasets D_S, D_T respectively

$$\phi(0) = P_S \text{ and } \phi(1) = P_T$$

$$Z^\infty = \{\phi(t)^T x : t \in [0, 1]\}$$

$$\langle Z_i^\infty, Z_j^\infty \rangle = \int_0^1 (\phi(t)^T x_i)^T (\phi(t)^T x_j) dt = x_i^T \mathbf{G} x_j$$

GFK G can be used to extract domain-invariant feature space!

Gong, B., Shi, Y., Sha, F., and Grauman, K. "Geodesic flow kernel for unsupervised domain adaptation." In CVPR, 2012

MMD-based Domain Adaptation

Three-step approach

1) Selection of source dataset landmarks using weighted MMD for different bandwidths (scaling factors)
2) Constructing the auxiliary tasks by moving landmarks from source to target dataset
3) Learning final class-discriminative optimized kernel from all auxiliary GFKs and landmarks (for different scaling factors).

MMD-based Domain Adaptation

Step1: Selection of source-domain landmark instances

- Learn source domain weights using weighted MMD.

$$\mathrm{MMD}(D_S, D_T) = \| 1/N \sum_n \Phi(x_n) - 1/M \sum_m \Phi(x_m) \|^2_{\mathcal{H}}$$

$$\text{Optimal } W = \mathrm{argmin}_W W^T K_{SS} W - 2/M\, W^T K_{ST}$$

$$K(x_i, x_j) = \exp(-(x_i - x_j)^T G (x_i - x_j)/\sigma^2_q)$$

- For each scaling factor σ_q, landmarks L_q are chosen to be high-weighted instances.

MMD-based Domain Adaptation

Step2: Constructing the auxiliary tasks by moving landmarks from source to target dataset

- For each scaling factor σ_q, learn GFK G_q using new pair of datasets

$$D_{Sq} = D_S \setminus L_q \text{ and } D_{Tq} = D_T \cup L_q$$

$$KL(P_S(X)||P_{Tq}(X)) \leq KL(P_S(X)||P_T(X))$$

MMD-based Domain Adaptation

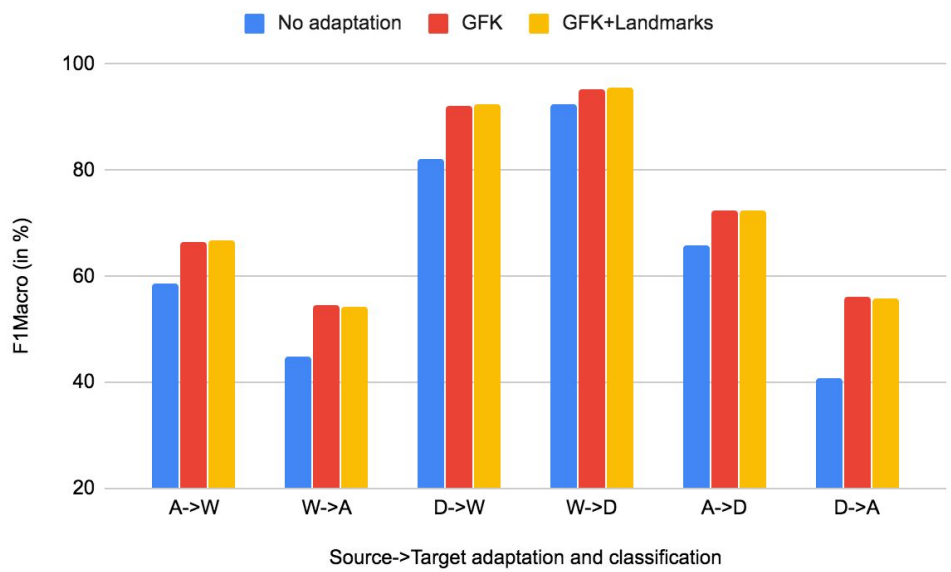
Step3: Learning final class-discriminative optimized kernel

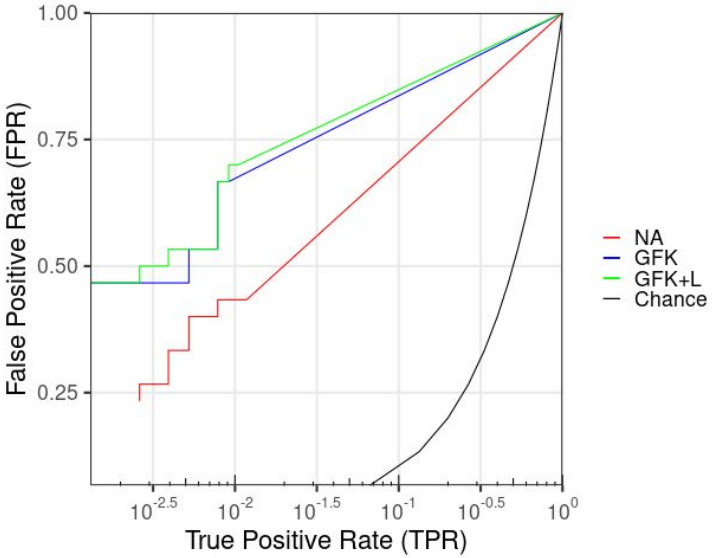
- Learn convex combination of all kernels G_q
- Final kernel, $F = \sum_q w_q G_q$ s.t. $\sum_q w_q = 1$ and $w_q \geq 0$
- Use L_q labels to optimize learning of w_q by minimizing prediction error on landmarks.

Experimental Results on Image Classification Datasets

Model	A->W	W->A	D->W	W->D	A->D	D->A
No adaptation	58.52	44.96	82.28	92.39	65.82	40.84
GFK	66.36	**54.46**	92.21	95.29	72.31	**56**
GFK+Landmarks	**66.72**	54.36	**92.52**	**95.59**	**72.51**	55.74

Classification metrics (F1Macro in %) reported for source -> target dataset pairs.
A: Amazon, W: Webcam and D: DSLR





Receiver Operating Characteristic (ROC) plot for laptop_computer class for A->W pair

Thank You!

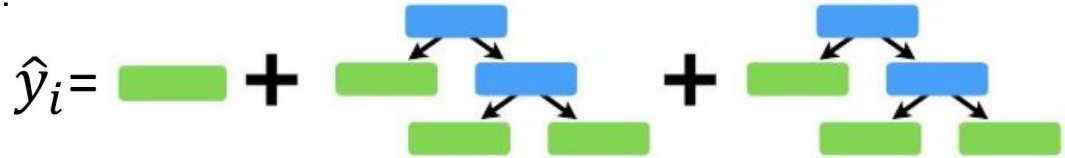
From Gradient Tree Boosting to XGBoost

Tong Lin Wentai Zhang

Carnegie Mellon University

Gradient Tree Boosting

- A tree based additive model that learns the residual produced from the previous base learner.



$\hat{y}_i =$

$\hat{y}_i^1$ Learns y_i

$\hat{y}_i^2$ Learns $y_i - \hat{y}_i^1$

$\hat{y}_i^3$ Learns $y_i - \hat{y}_i^1 - \hat{y}_i^2$

- More concisely,

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F}$$

Where,

$$\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}(q : \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T)$$

T is the # of leaf nodes for the tree.
w is the leaf node values for the tree.

Rule Of Leaf Splitting

- XGBoost uses **the first and the second order loss gradient** before and after a leaf node split to evaluate the quality of the split.
- The larger the evaluation L_{split}, the better the split is.

$$\mathcal{L}_{split} = \frac{1}{2}\left[\underbrace{\boxed{\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda}}}_{\text{loss after split}} - \underbrace{\boxed{\frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}}}_{\text{loss before split}}\right] - \gamma$$

- Loss function: $L^k = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(k-1)} + f_k(x_i)) + \Omega(f_k)$ where $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$
- ***γ*** regulates # of leaf nodes. **λ** regulates the values of leaf nodes.
- The g_i and the h_i are the 1st and the 2nd order loss gradients, resulting from the 2nd Taylor series approximation of the loss function.

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{(k-1)})}{\partial \hat{y}_i^{(k-1)}}$$

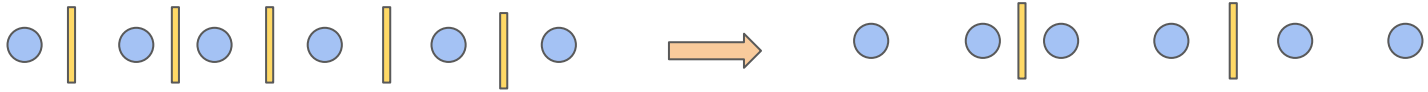
$$h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(k-1)})}{\partial (\hat{y}_i^{(k-1)})^2}$$

Note: **g_i and h_i are constant** since y_i and $\hat{y}_i^{(k-1)}$ are all known.

Split Finding Algorithm

- Exact Greedy Algorithm for Split Finding
  Calculate L_{spit} for each possible split of the given training sets. Then choose the argmax.
  Not efficient and not suitable for distributed settings
- Approximation using the weighted quantile sketch

Sorted data points:



Try all possible splits
(Exact)

Only propose quantile splits
(Approximate)

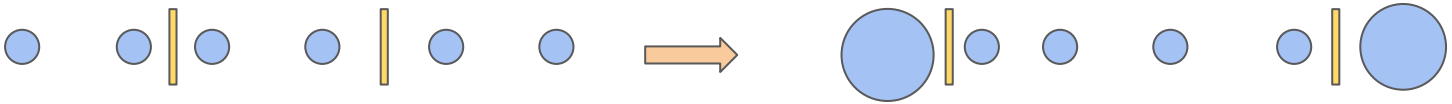
- "weighted"

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

$$-g_i/h_i$$

$$\sum_{i=1}^{n} \frac{1}{2} h_i (f_t(\mathbf{x}_i) - g_i/h_i)^2 + \Omega(f_t) + constant,$$

Weighted squared loss (weights are hi's)



⅓ quantile splits
(unweighted)

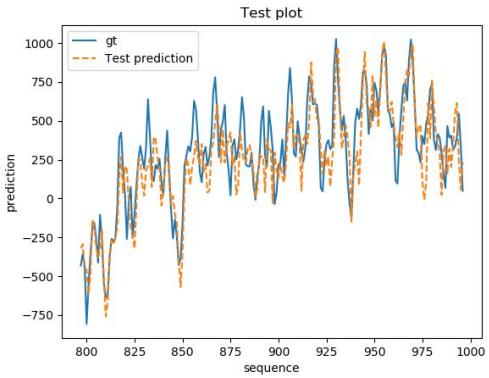
⅓ quantile splits
(weighted)

Experiments

Dataset: Supply chain demand data from our funding source

Features
(eg. customer sales,
social sentiment...
in total ~100)

Customer
demands (scalar)



Base case: XGBoost (n_est = 30, max_dept = 6, reg_lambda = 1.0)

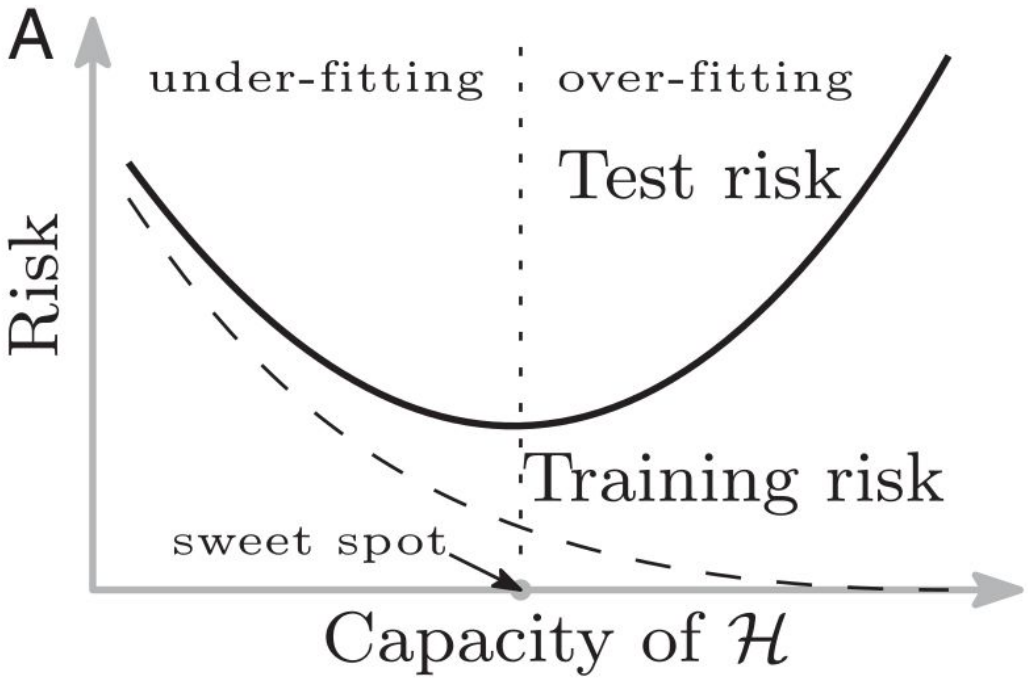
Condition	Train MAE	Test MAE	Comment
Base	24.12	126.87	OPT
n_est=10	54.28	129.74	underfitting
n_est=60	12.57	127.42	overfitting
max_dept=5	35.96	130.94	Structure too coarse
max_dept=7	15.23	145.11	Structure too fine
Lambda=0.0	20.28	149.59	No regu
lambda=2.0	27.65	131.58	Over-regu

Double Descent in High-Dimensional Least-Squares

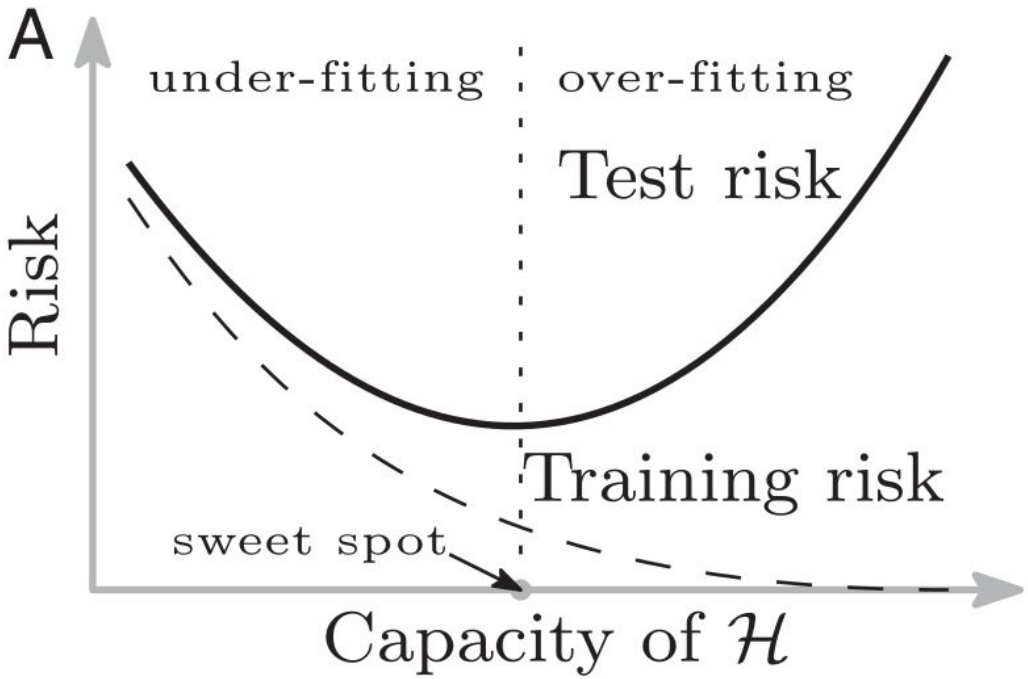
(10-716 S20 Final Project)

Neil Xu

The classical bias-variance tradeoff

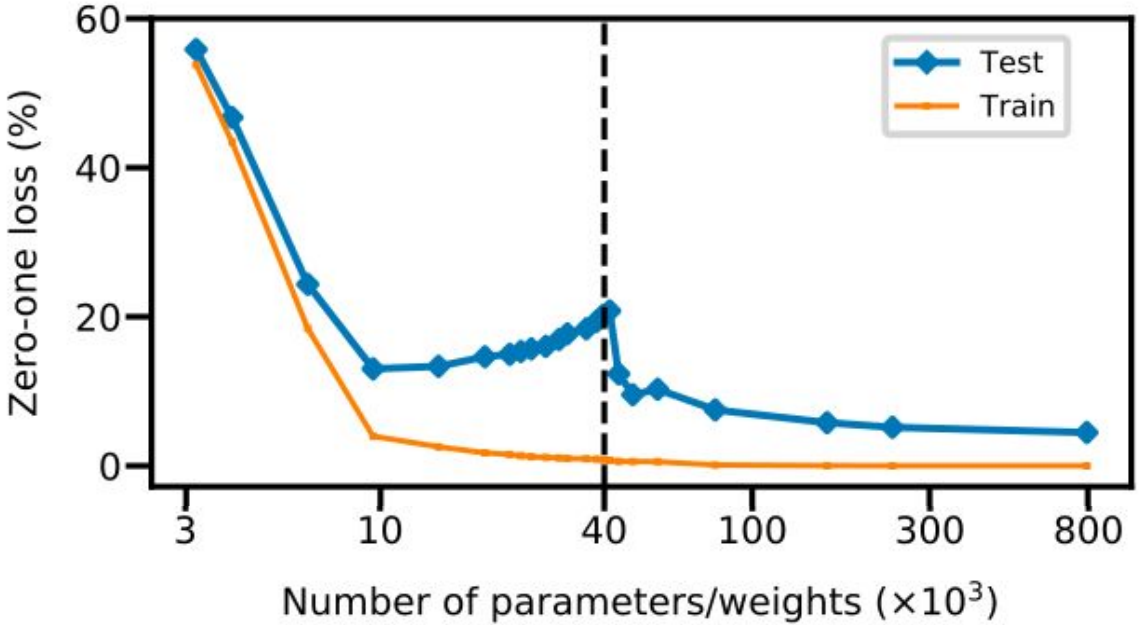


The classical bias-variance tradeoff



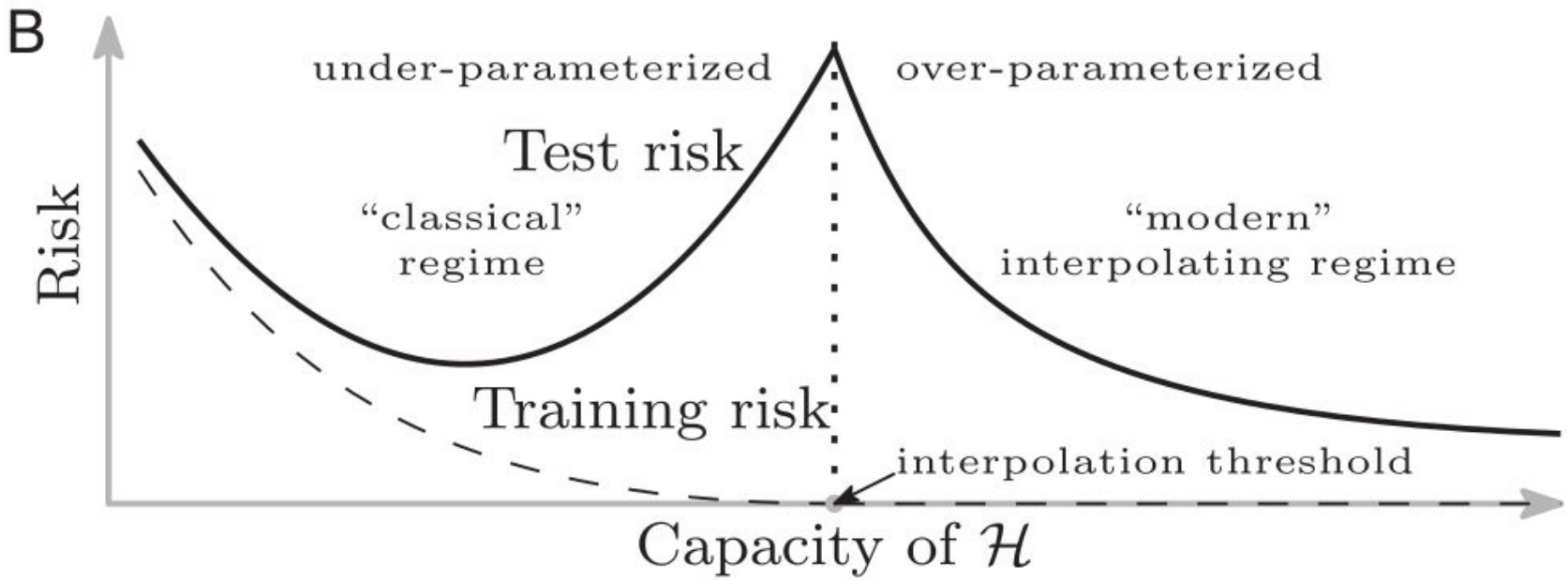
$$\text{Risk} = \text{Bias}^2 + \text{Variance}$$

Deep learning does not adhere to the tradeoff



Fully connected 2 layer neural network on MNIST classification

Double descent

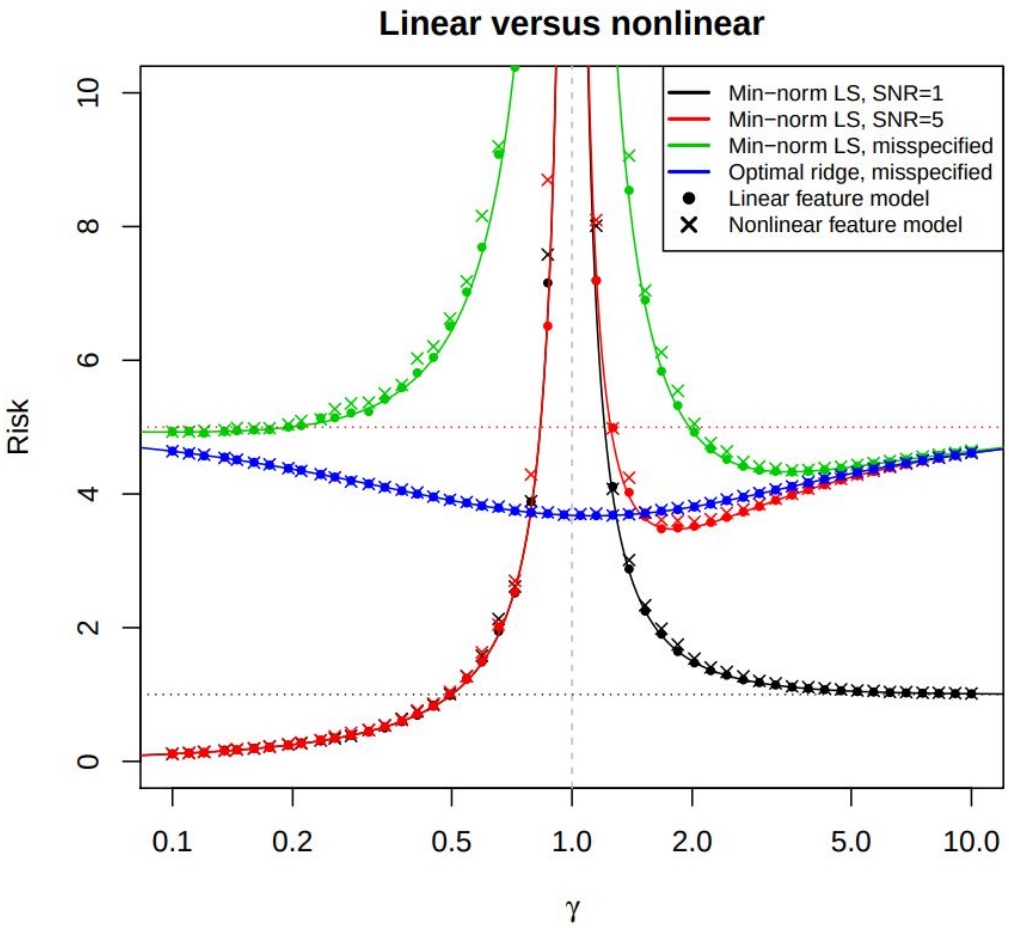


“Surprises in High-Dimensional Ridgeless Least Squares Interpolation”

Trevor Hastie (Stanford), Andrea Montanari (Stanford), Saharon Rosset (Tel Aviv), Ryan J. Tibshirani (CMU)

Main contributions:

1. For high-dimensional least-squares regression, in the overparameterized regime (# of features > # of examples) the risk can have a global minimum.
2. This holds true in a nonlinear model as well i.e. when the features are generated by a two layer NN w/ random weights.



$$\gamma = p/n$$

$$\mathrm{SNR} = \|\beta^*\|_2^2/\sigma^2$$

(signal-to-noise ratio)

Misspecification: only a subset of features are observed

Thanks!

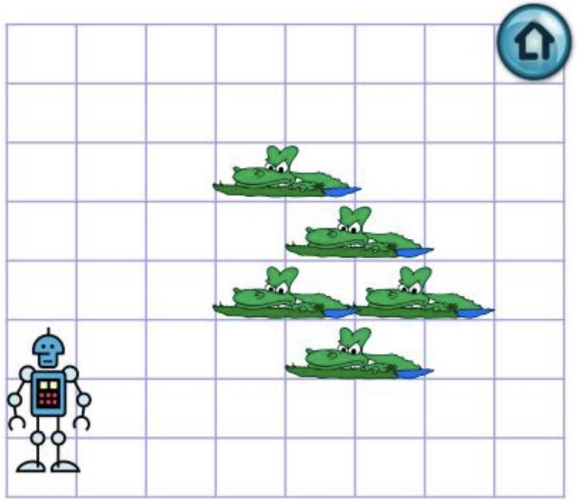
Exploring Partial Observability in Reinforcement Learning and Planning

Siddharth Ancha, sancha@cs.cmu.edu
Stephanie Milani, smilani@cs.cmu.edu

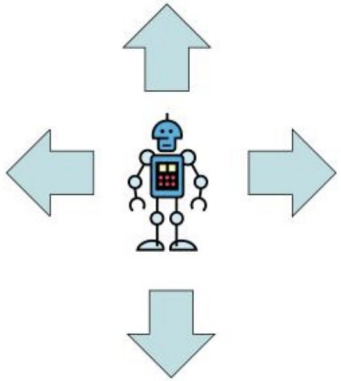
Traditional RL: Markov Decision Processes (MDPs)

State: $s \in S$
(true location of robot, crocs, goal)

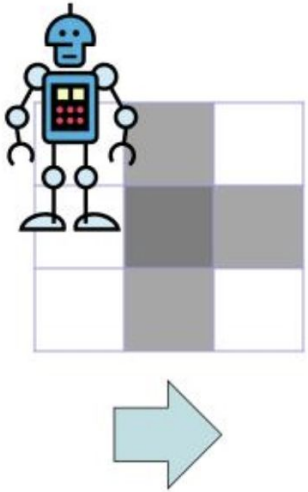
Reward: $R(s, a)$: +100 for reaching goal
-100 for reaching croc



Actions: $a \in A$
$a \in$ (let, right, up, down)



Transition:
$T(s, a, s') = Pr(s' \mid s, a)$



Policy: $\pi: S \rightarrow A$

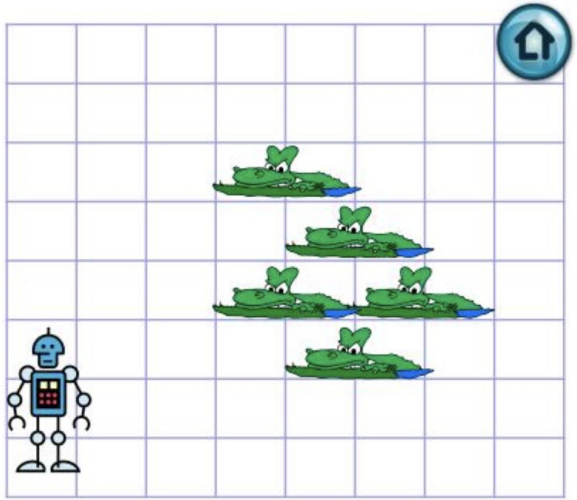
*figures adapted from Lee Wee Sun, National University of Singapore

Traditional RL: Markov Decision Processes (MDPs)

State: $s \in S$

(true location of robot, crocs, goal)

Policy: $\pi: S \rightarrow A$??



- Real autonomous agents seldom have access to the true state of the world. They need to make decisions based on ***partial observations***.

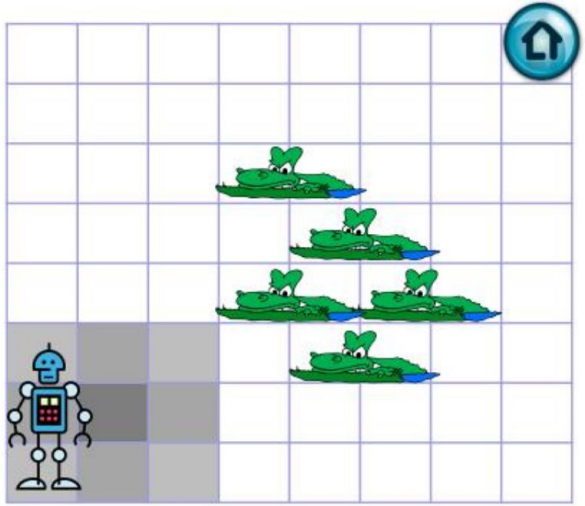
- Partially Observable Markov Decision Processes (POMDPs) provide a nice framework that models observations of states, and policies as functions of observations.

*figures adapted from Lee Wee Sun, National University of Singapore

Traditional RL: Markov Decision Processes (MDPs)

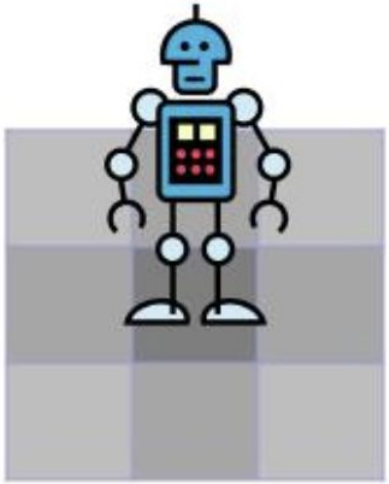
State: $s \in S$

(true location of robot, crocs, goal)



Observations: $o \in \Omega$

$O(o, a, s') = Pr(o|a, s')$



Policy: $\pi(o1, \ldots, ot) = a$

*figures adapted from Lee Wee Sun, National University of Singapore

Contributions

- Survey theoretical results and important proofs in POMDPs.
- Survey methods (exact and approximate) to solve POMDPs.
- Run experiments using state-of-the-art POMDP solvers on benchmark POMDP environments.

Key Theoretical Results on POMDPs

- POMDPs are much harder to solve them MDPs
  [Papadimitriou & Tsitsiklis, 1987]
  - MDPs are known to be solvable in P, whereas POMDPs are PSPACE-complete. Even less likely to be solved in poly-time than NP-complete problems.

- POMDPs are converted to belief MDPs to be solved optimally.
  [Smallwood & Sondik, 1971]

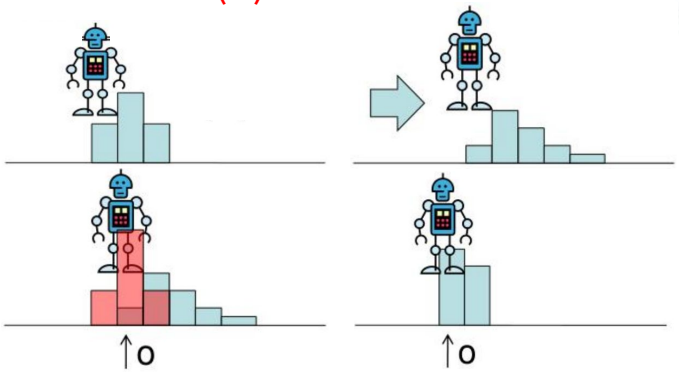
POMDP → Belief MDP

Belief = probability distribution over states

- **Belief is a sufficient statistic of the history** of observations.
  [Smallwood & Sondik, 1971] **Policy: $\pi : (b) = a$** where **$b \in \mathscr{P}(S)$**

- Belief can be computed
  via **Bayesian Estimation**

$$b_t(s_t) = \frac{O(o_t, a_{t-1}, s_t) \sum_{s_{t-1}} T(s_{t-1}, a_{t-1}, s_t) b_{t-1}(s_{t-1})}{\Pr(o_t | a_{t-1}, b_{t-1})}$$

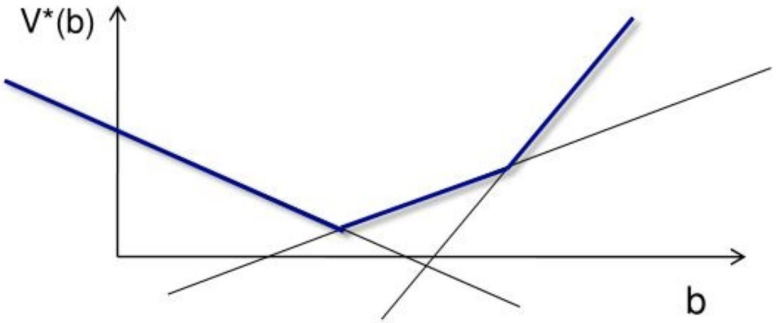


- Even for finite {states, actions, observations}, belief space is uncountably infinite! How do we represent **$\pi : \mathscr{P}(S) \rightarrow A$** ?

*figures adapted from Lee Wee Sun, National University of Singapore

First exact algorithm to solve POMDPs [Sondik, 1978]

- Represent belief policies as upper envelope of finite set of linear functions.



$$V(b) = \max_{\alpha \in V} b \cdot \alpha$$

- Value iteration under this representation can exactly solve small POMDPs, but the number of *alpha* vectors grows exponentially!

$$V' = \bigcup_{a \in A} V^a$$

$$V^a = \bigoplus_{o \in \Omega} V^{a,o}$$

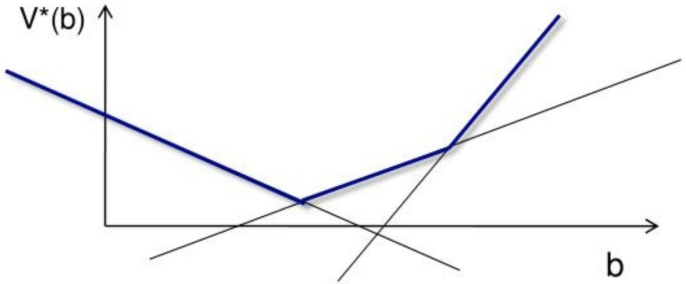
$$V^{a,o} = \{\frac{1}{|\Omega|} r_a + \alpha^{a,o} : \alpha \in V\}$$

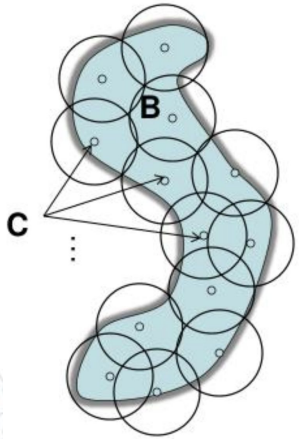
$$\alpha^{a,o}(s) = \sum_{s' \in S} O(a, s', o) T(s, a, s') \alpha(s')$$

*figures adapted from Lee Wee Sun, National University of Singapore

Offline Approximation: Point-Based Methods

- Remove *alpha* vectors that don't form upper envelope for **reachable beliefs**
  - PBVI [Pineau, Gordon, Thrun, 2003]
  - HSVI [Smith & Simmons, 2004]
  - Perseus [Spaan, Vlassis, 2005]
  - **SARSOP** [Kurniawati, Hsu, Lee, 2008]



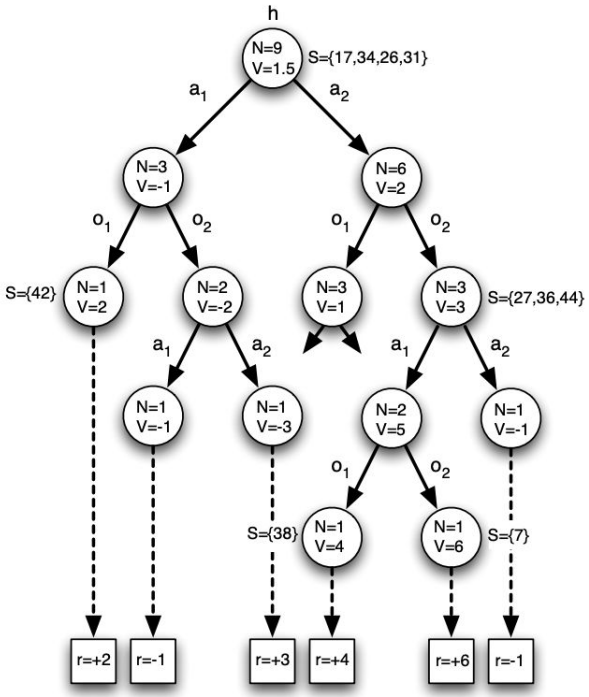


- Theoretical results
  - Point-based VI gets sufficiently close to optimal value when sampling done over *entire reachable belief space*. [Pineau, Gordon, Thrun, 2006]
  - POMDPs can be efficiently approximated when reachable belief space is small. [Hsu, Lee, Nan, 2007].

*figures adapted from Lee Wee Sun, National University of Singapore

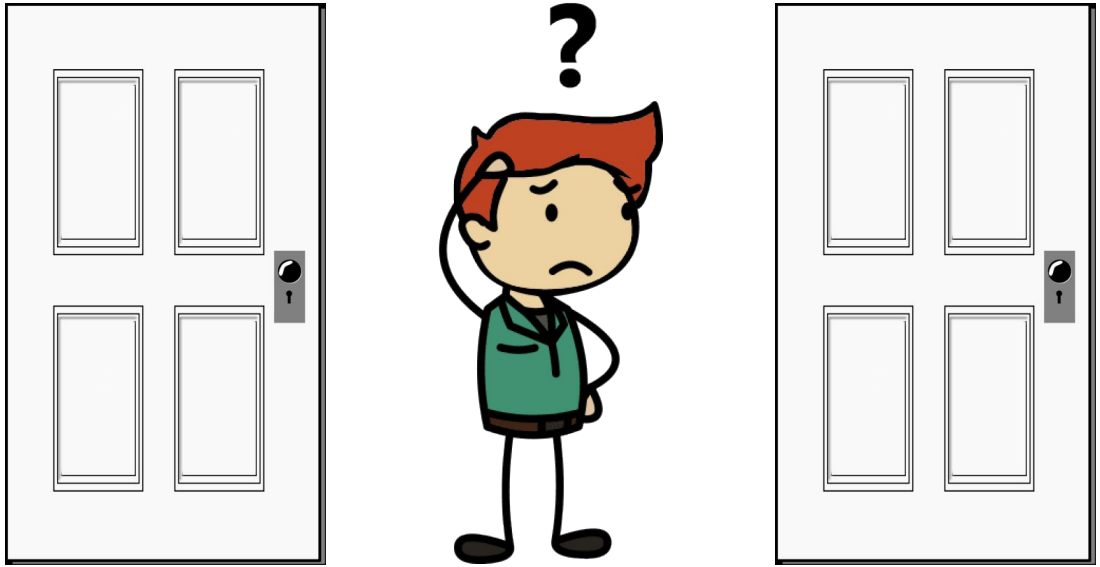
Online Approximation: Monte-Carlo Methods

- **POMCP**: Monte-Carlo Planning in Large POMDPs [Silver, Veness, 2010]

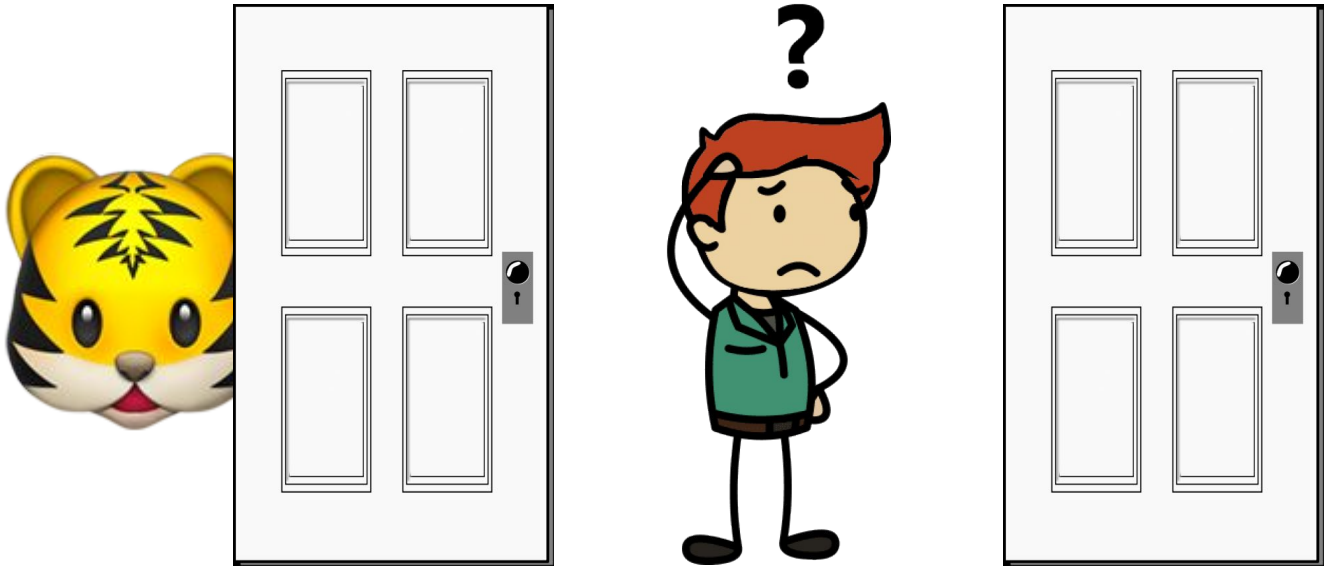


- Applies Monte-Carlo Tree Search (MCTS), where each node represents a *history of past observations* instead of state.
- Each node also stores visitation counts & value estimates.
- UCB1 is used to decide which action to take at each node, trading off exploration and exploitation. $V(ha) + c\sqrt{\frac{\log N(h)}{N(ha)}}$
- Only requires a black-box simulator of the environment. A particle filter is used to represent beliefs for each history. Particles (states) are sampled from the belief to perform simulation.
- Works very well in practice, often outperforming offline methods!

Tiger



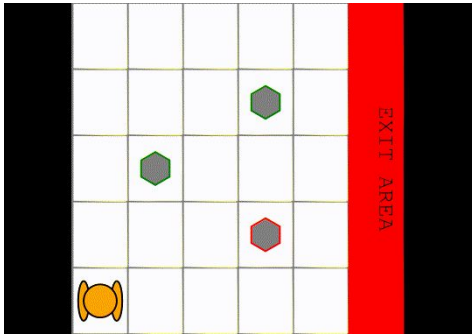
Tiger



Results: Tiger

	Average Cumulative Discounted Reward (4 trials, 30 steps of policy)
SARSOP	16.81
QMDP	25.53
POMCPOW (extension of POMCP)	21.92

Results: 5x5 RockSample



	Final Discounted Reward	Time	Beliefs
SARSOP	**33.772**	**0.07**	**69**

Thanks!

Any questions?

Carnegie Mellon University

Enhance the Bandgap Classifier for Organic Molecular Crystals with Batch Mode Active Learning

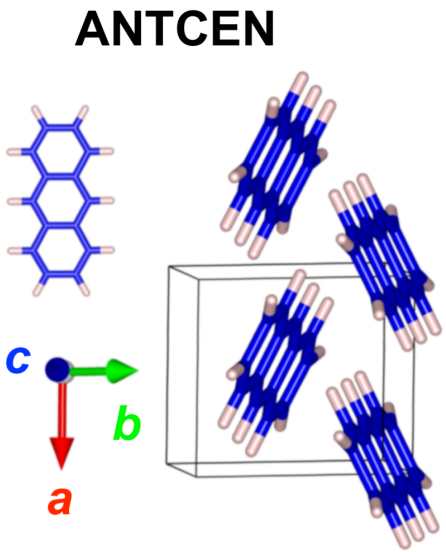
Bo Lei
Department of Materials Science and Engineering

Xingyu Liu
Department of Materials Science and Engineering

Problem

- Band gap is an essential electronic property of organic semiconductors, those with relative smaller band gap (<1.5 eV) are preferred. However, obtaining this value is expensive.

- In order to limit the cost of calculating band gaps, training a band gap classifier using crystal structure information is getting popular.

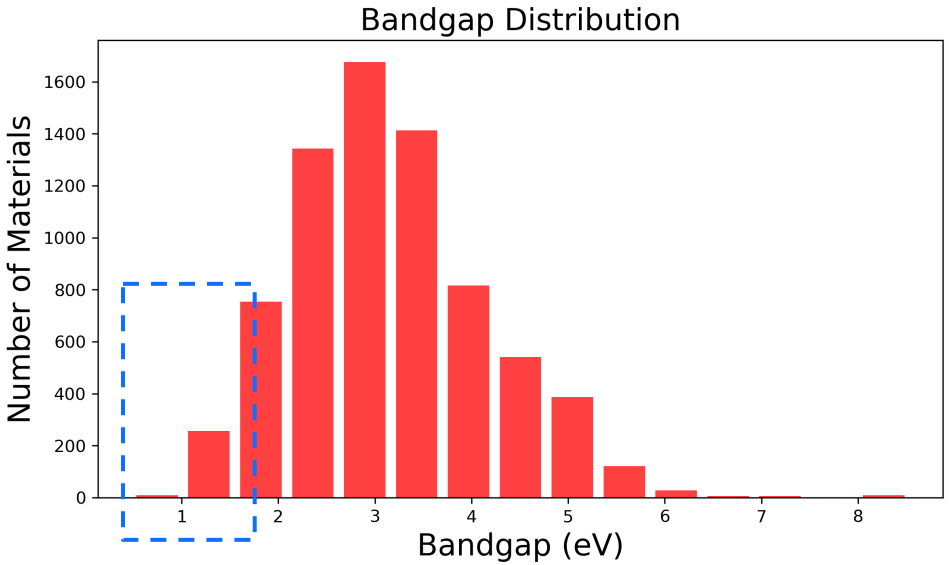


Carnegie Mellon University

Problem

- No sufficient training data with preferred class of small-gap materials. Less than 5% of dataset are target class.

- In order to improve the accuracy of band gap classifier, and efficiently utilize the computing resources, batch-mode active learning is implemented.



Carnegie Mellon University

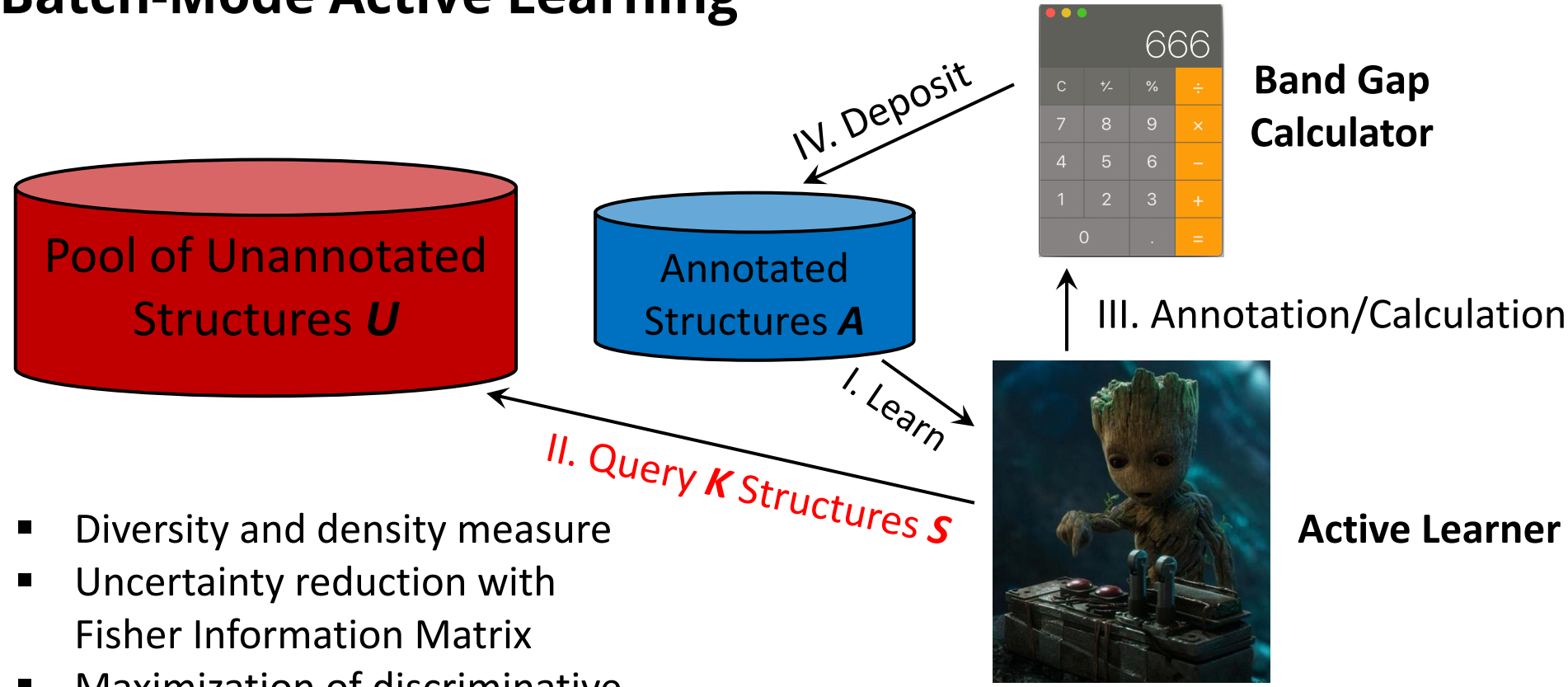
Batch-Mode Active Learning

- Active learning is a method used to select the next unlabeled data to annotate, so that a better-performed model can be achieved more efficiently and economically.

- However, single target selection is not practical as parallel annotation can be performed.

- Batch-mode active learning is proposed based on active learning to maximize the rate of model improvement.

Batch-Mode Active Learning



- Diversity and density measure
- Uncertainty reduction with Fisher Information Matrix
- Maximization of discriminative classification performance

Carnegie Mellon University

Diversity and density measure

- General Idea: maximize the difference between annotated structures and next batch of unlabeled structures

$$S = 0$$

$$\text{For } i = 1, 2, ..., k$$

$$s_i = \text{argmax}_{s_i \in U \notin A}[\alpha \cdot relevance(s_i) + \beta \cdot density(s_i) + (1 - \alpha - \beta) \cdot diversity(s_i, S)]$$

$$S = S \cup s_i$$

$$\text{relevance}(s_i) = KL(\hat{\theta}_S || \hat{\theta}_A)$$

$$\text{density}(s_i) = \frac{-1}{|A|} \sum_{s_j \in A} J(s_i || s_j)$$

$$\text{diversity}(s_i, S) = J(s_i || s_j),\ s_j = \text{argmin}_{s_j \in S} |s_i - s_j|$$

Carnegie Mellon University

Uncertainty reduction with Fisher Information Matrix

- General Idea: Search for a set of examples which can most efficiently reduce Fisher Information. Use Fisher information matrix to represents the uncertainty of a maximum likelihood estimation.

- The set of examples that can most efficiently reduce the uncertainty of classification model is found by minimizing the ratio between the two Fisher information matrices

$$S^* = \text{argmin}_S \, \text{tr}\big(I_S(\alpha)^{-1} I_U(\alpha)\big)$$

α : classifier parameter

I_S : Fisher information for selected data

I_U : Fisher information for all unlabeled data

- In logistic regression settings:
- Require efficient algorithm for set search

$$I_U(\hat{\alpha}) = \frac{1}{|U|} \sum_{x \in U} \pi(x)(1 - \pi(x)) x x^T + \delta I_d$$

$$I_S(S, \hat{\alpha}) = \frac{1}{|S|} \sum_{x \in S} \pi(x)(1 - \pi(x)) x x^T + \delta I_d$$

$$\pi(x) = p(-|x) = \frac{1}{1 + exp(\hat{\alpha^T} x)}$$

Carnegie Mellon University

Maximization of discriminative classification performance

- General Idea: Formulate active learning as an optimization problem. Maximize the log likelihood of labeled instances and minimize the uncertainty of unlabeled instances. Use entropy to represent uncertainty.

$$f(S) = \sum_{i \in A \cup S} \log P(y_i | x_i, w) - \alpha \sum_{j \in U \setminus S} H(y | x_j, w) \qquad w \text{ is trained on } A \cup S$$

- Since label of *S* is unavailable, use the best *f*(*S*) score *S* can achieve over all possible label configurations y_S.

$$S^* = \max_S \max_{y_S} \sum_{i \in A \cup S} \log P(y_i | x_i, w) - \alpha \sum_{j \in U \setminus S} H(y | x_j, w)$$

- In practice, transfer to a linear programming problem and use optimization techniques.

Carnegie Mellon University

Thank You

SGD Variants: How are they better?

Zejie Ai, Max Chen

Agenda

- Vanilla SGD
- Limitations of Vanilla SGD
- SGD with Momentum
- Elastic Averaging SGD
- Evolutionary SGD

Vanilla SGD

- Performs a parameter update for each training example

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

- SGD does away with the redundancy problem that Batch Gradient Descent (GD) suffers from
- Thus SGD is usually much faster than Batch GD and works well in the online setting

Limitations of Vanilla SGD

- SGD performs frequent updates with a high variance that might cause the loss function to fluctuate heavily
- SGD has trouble navigating ravines, which are common around local optima
- SGD can be slow when the dataset gets large
- SGD may not be suitable for complex optimization problems
    - e.g. non-linear, non-convex, non-smooth

SGD with Momentum

- Include a momentum term when updating the parameters

$$\Delta w_t = -\epsilon \nabla_w J(w) + p \Delta w_{t-1}$$

- The momentum term has a nice physical interpretation

$$m \frac{d^2 w}{dt^2} + \mu \frac{dw}{dt} = -\nabla_w E(w) \approx -H(w - w_0)$$

- Under a similarity transformation, the system now simplifies to

$$m \frac{d^2 w_i'}{dt^2} + \mu \frac{dw_i'}{dt} = -k_i w_i'$$

SGD with Momentum

- The Vanilla SGD update rule is a special case when setting m = 0. Then the solution is $w_i'(t) = ce^{\lambda_{i,0}t}$, $\lambda_{i,0} = -\frac{k_i}{\mu}$
- In the general case, when m ≠ 0, the solution becomes $w_i'(t) = c_1 e^{\lambda_{i,1}t} + c_2 e^{\lambda_{i,2}t}$ as $\lambda_{i,\{1,2\}} = -\frac{\mu}{2m} \pm \sqrt{\frac{\mu}{m}(\frac{\mu}{4m} - \frac{k_i}{\mu})}$
- The speed of convergence of the system is determined by the magnitude of the real parts of the eigenvalues λ_i's

Momentum Helps

Theorem: For positive m, μ and k_i ,the inequality $|\mathrm{Re}\,(\lambda_{i,1})| > |\mathrm{Re}\,(\lambda_{i,0})|$ holds, if and only if $k_i < \frac{\mu^2}{2m}$. Therefore the momentum term improves the rate of convergence.

The proof deals with k_i's by three cases: $(0, \mu^2 / 4m]$, $(\mu^2 / 4m, \mu^2 / 2m)$ and $[\mu^2 / 2m, +\infty)$.

Elastic Averaging SGD (EASGD)

- Vanilla SGD can be slow when the dataset gets large.
- Running SGD asynchronously is faster, but suboptimal communication between workers can lead to poor convergence.
- EASGD links the parameters of the workers of asynchronous SGD with a center variable stored by the parameter server, allowing more exploration in the parameter space.
    - This center variable is used as an anchor when updating the local models.

Elastic Averaging SGD (EASGD)

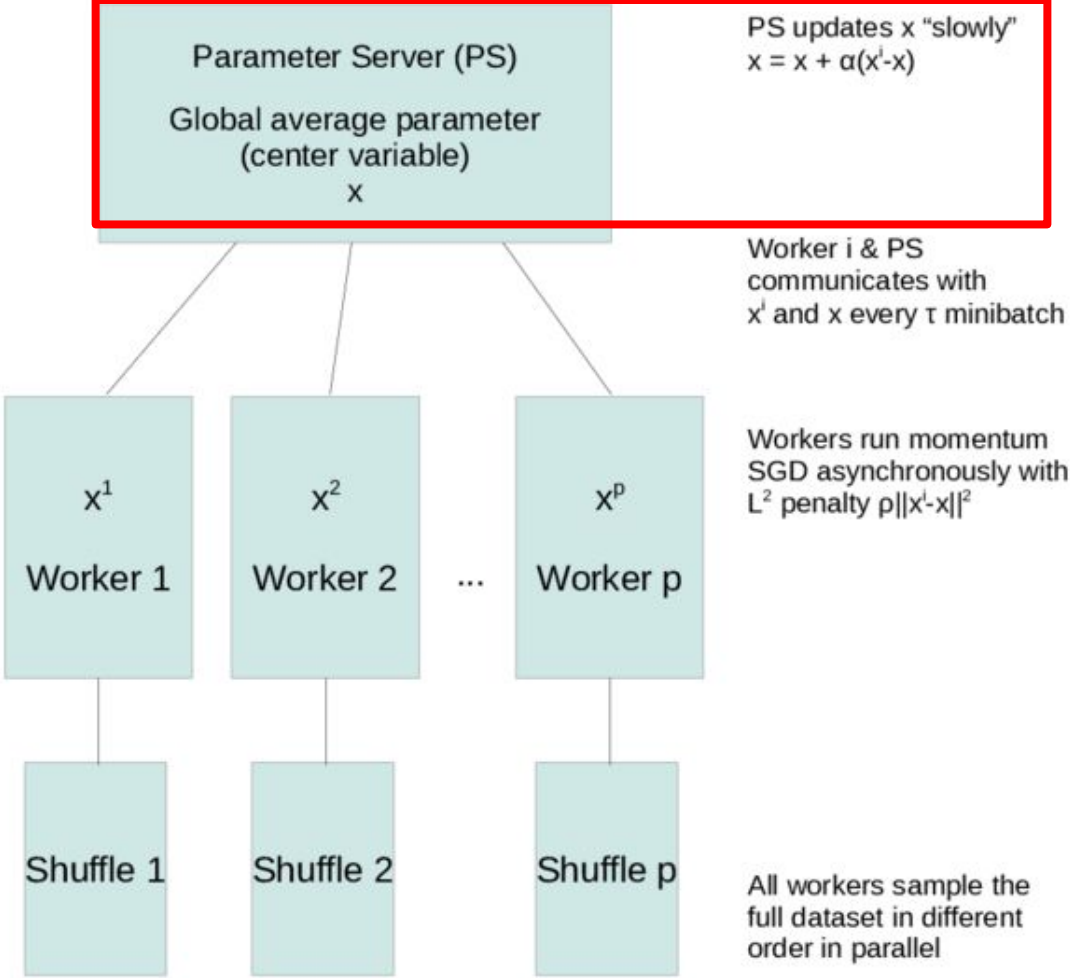
Center Variable

- Optimization Problem:

$$\min_{x^1,\ldots,x^p,\tilde{x}} \sum_{i=1}^{p} \mathbb{E}[f(x^i,\xi^i)] + \frac{\rho}{2}\|x^i - \tilde{x}\|^2$$

- Update Rule:

$$\begin{aligned} x_{t+1}^i &= x_t^i - \eta(g_t^i(x_t^i) + \rho(x_t^i - \tilde{x}_t)) \\ \tilde{x}_{t+1} &= \tilde{x}_t + \eta \sum_{i=1}^{p} \rho(x_t^i - \tilde{x}_t), \end{aligned}$$



Features:

** EASGD has similar implementation to the Asynchronous SGD method*

** EASGD is very stable using large communication period τ*

** Workers spend most of their time on computation*

** Data communication is much less expensive than parameter communication*

Distributed stochastic optimization for deep learning (thesis) - Scientific Figure on ResearchGate.
https://www.researchgate.net/figure/The-big-picture-of-EASGD_fig1_302588432

Evolutionary SGD (ESGD)

- Combining gradient-free evolutionary algorithms (EAs) and SGD helps optimization on large, distributed networks.
  - EA are population-based so computation is intrinsically parallel.
- ESGD uses a model back-off and elitist strategy.
  - theoretical guarantee that the best model in the population will never degrade.

Evolutionary SGD (ESGD)

- Given a population of randomly initialized parameter vectors, ESGD searches for the ones which give the lowest empirical risks.

$$J = \mathbb{E}_\theta[R_n(\theta)] = \mathbb{E}_\theta[\mathbb{E}_\omega[l_\omega(\theta)]]$$

- The best offsprings are selected through **m-elitist average fitness.**
  - the average fitness of the best m individuals from ranking them in ascending order

$$J_{\bar{m}:\mu} = \frac{1}{m}\sum_{k=1}^{m} f(\theta_{k:\mu})$$

Algorithm 1: Evolutionary Stochastic Gradient Descent (ESGD)

Input: generations K, SGD steps K_s, evolution steps K_v, parent population size μ, offspring population size λ and elitist level m.

Initialize population $\Psi_\mu^{(0)} \leftarrow \{\theta_1^{(0)}, \cdots, \theta_\mu^{(0)}\}$;

// K generations

for $k = 1 : K$ **do**

  Update population $\Psi_\mu^{(k)} \leftarrow \Psi_\mu^{(k-1)}$;

  // in parallel

  for $j = 1 : \mu$ **do**

    Pick an optimizer $\pi_j^{(k)}$ for individual $\theta_j^{(k)}$;

    Select hyper-parameters of $\pi_j^{(k)}$ and set a learning schedule;

    // K_s SGD steps

    for $s = 1 : K_s$ **do**

      SGD update of individual $\theta_j^{(k)}$ using $\pi_j^{(k)}$;

      If the fitness degrades, the individual backs off to the previous step $s-1$.

    end

  end

  // K_v evolution steps

  for $v = 1 : K_v$ **do**

    Generate offspring population $\Psi_\lambda^{(k)} \leftarrow \{\theta_1^{(k)}, \cdots, \theta_\lambda^{(k)}\}$;

    Sort the fitness of the parent and offspring population $\Psi_{\mu+\lambda}^{(k)} \leftarrow \Psi_\mu^{(k)} \bigcup \Psi_\lambda^{(k)}$;

    Select the top m ($m \leq \mu$) individuals with the best fitness (m-elitist);

    Update population $\Psi_\mu^{(k)}$ by combining m-elitist and randomly selected $\mu - m$ non-m-elitist candidates;

  end

end

SGD
back-off

EA
m-elitist

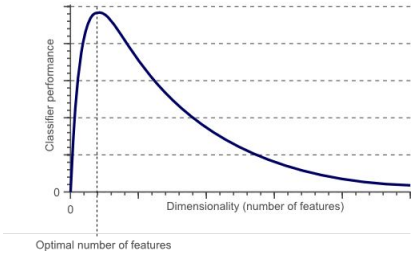
Thanks!

Dimensionality Reduction

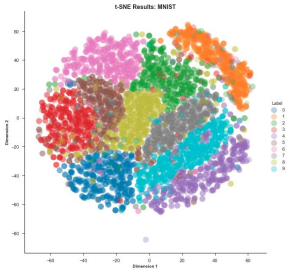
Jiaxian Sheng, Ye Yuan

Why Dimensionality Reduction

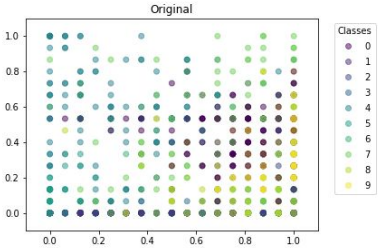
- Curse of dimensionality



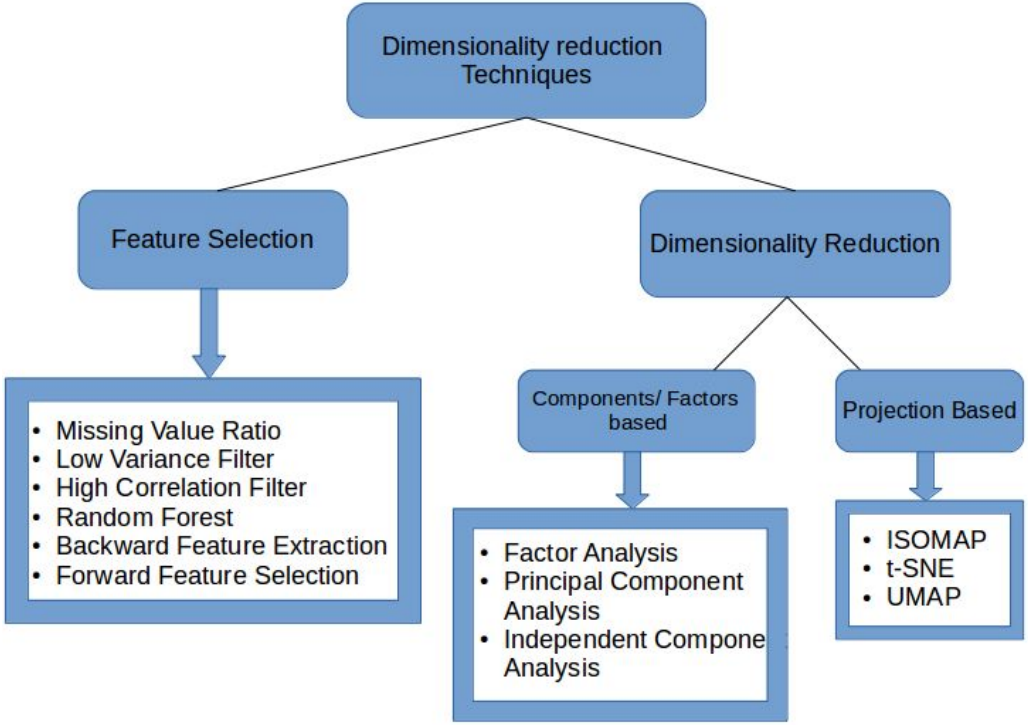
- Understanding your data



- Visualization



Methods


Dimensionality reduction Techniques
Feature Selection
Dimensionality Reduction
Missing Value Ratio
Low Variance Filter
High Correlation Filter
Random Forest
Backward Feature Extraction
Forward Feature Selection
Components/ Factors based
Projection Based
Factor Analysis
Principal Component Analysis
Independent Compone Analysis
ISOMAP
t-SNE
UMAP

Components/ Factor Based

Algorithms	Data	Objective	Need label	Time complexity
PCA	Linear subspace	Minimize reconstruction error, maximize variance	No	$O(ND^2+D^3)$
Kernel PCA	Linear after projection	Minimize error, maximize variance in projection space	No	
LLE	Non-linear	Preserve local geometry	No	$O(Dlog(k)Nlog(N) + DNk^3 + DN^2)$
Independent Component Analysis	Each is mixture of independent components	Maximize projections' statistical independence	No	$O(D(D+1)NT)$
LDA	Multivariate normality, homoscedasticity, multicollinearity, independence	Find a linear combination of features that characterizes or separates two or more classes of objects or events	Yes	$max(O(ND^2), O(D^3))$

Projection Based

Algorithms	Data	Objective	Need label	Time complexity
MDS	Nonlinear	Between-object distances are preserved as well as possible	No	Classical MDS: $O(N^3)$ per step
Isomap	Nonlinear, manifold	Estimate the intrinsic geometry of a data manifold based on an estimate of neighbors	No	$O[D\log(k)N\log(N)] + O[N^2(k+\log(N))] + O(DN^2)$
t-SNE	Nonlinear	Similar objects are modeled by nearby points	No	$O(DN^2)$

Experiments

- MNIST(60,000 x 784), Fashion MNIST (60,000 x 784)
- Leukemia (72 x 7128)
- Metrics:
  - Trustwo

$$M_1(k) = 1 - \frac{2}{Nk(2N-3k-1)} \sum_{i=1}^{N} \sum_{j \in U_k(i)} (r(i,j) - k)$$

$$M_2(k) = 1 - \frac{2}{Nk(2N-3k-1)} \sum_{i=1}^{N} \sum_{j \in V_k(i)} (\hat{r}(i,j) - k)$$

  - Residual Variance
  - Classification Error Rate

Appendix

References

- https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.visiondummy.com%2F2014%2F04%2Fcurse-dimensionality-affect-classification%2F&psig=AOvVaw3_h5eP8ADYB1NVE9uBcK3k&ust=1588040109410000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCPDBvMvEh-kCFQAAAAAdAAAAABAV
- https://jeanselme.github.io/
- https://www.google.com/url?sa=i&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FDimensionality_reduction&psig=AOvVaw300cxdDL2_epuvqqwotIAt&ust=1588040357359000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLjr7r7Fh-kCFQAAAAAdAAAAABAD
- https://www.google.com/url?sa=i&url=https%3A%2F%2Ftowardsdatascience.com%2Fan-introduction-to-t-sne-with-python-example-5a3a293108d1&psig=AOvVaw242fxioApDzI4-o48jpyKD&ust=1588041608597000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCKjJuaDKh-kCFQAAAAAdAAAAABAD
- https://research.cs.aalto.fi/pml/papers/wsom05_lmds.pdf
- https://pdfs.semanticscholar.org/25c1/d0e6609df3483f13007b36f603f20a40de9e.pdf
- https://web.stanford.edu/~hastie/CASI_files/DATA/leukemia.html

Best Arm Identification in Multi-armed Bandit

Saket Dingliwal, Divyansh Pareek

Carnegie Mellon University

April 30, 2020

Problem Formulation

- Best Arm Identification problem
  - Learner outputs arm J_n after n rounds
  - Minimize $e(n) := \mathbb{P}(J_n \neq k^*)$
- Adversarial Rewards (ADV)
  - Adversary chooses reward matrix $g \in \mathbb{R}^{K \times n}$
  - Best arm $k_g^* = \arg\max_{k \in [K]} \sum_{i=1}^{n} g_{k,i}$
- Stochastic Rewards (STO)
  - Rewards sampled w/ mean vector $\mu \in \mathbb{R}^K$
  - Best arm $k_\mu^* = \arg\max_{k \in [K]} \mu_k$
- Best of Both Worlds (BOB)
  - Can a learner achieve optimal rates in both worlds?

Theorem (Upper Bound for Adversarial Rewards)

Uniform Learner (Rule) [Abbasi-Yadkori et al., 2018] for all n, given rewards g, outputs an arm with error

$$e_{ADV(g)}(n) \leq K \cdot \exp\left(-\frac{3n}{28H_{UNIF}(g)}\right)$$

Theorem (Upper Bound for Stochastic Rewards)

Sequential Halving [Karnin et al., 2013] for all n, for any stochastic reward generating process μ, outputs an arm with error

$$e_{STO(\mu)}(n) \leq 3\log K \cdot \exp\left(-\frac{n}{8H_{SR}(\mu)\cdot \log K}\right)$$

- Adversary easily fool Sequential Halving: choosing high rewards for arm rejected in early phase : Should not Reject!
- Pulling uniformly perform poorly in Stochastic: incur a large variance of reward estimates of order K : Reduce Variance!

Theorem (Lower bound for BOB setting)

For any learner, if there exists a stochastic problem $STO(\mu)$, such that for any reasonable n probability of error is upper bounded by

$$e_{STO(\mu)}(n) \leq \frac{1}{64} \exp(-\frac{2048n}{H_{BOB}(\mu)})$$

then there exists an adversarial problem g, that makes the learner suffer a constant error ie $e_{ADV(g)}(n) \geq 1/16$

P1 algorithm: parameter free algorithm

Sort and rank arms at each step based on the estimate $\tilde{G}_{\cdot,t-1}$ in descending order, where $\tilde{G}_{k,t} = \sum_{t'=1}^{t} \frac{g_{k,t'}}{p_{k,t'}} \mathbf{1}(I_{t'} = k)$

Pull arm with rank k with probability $\frac{1}{k\overline{\log K}}, \forall k \in [K]$

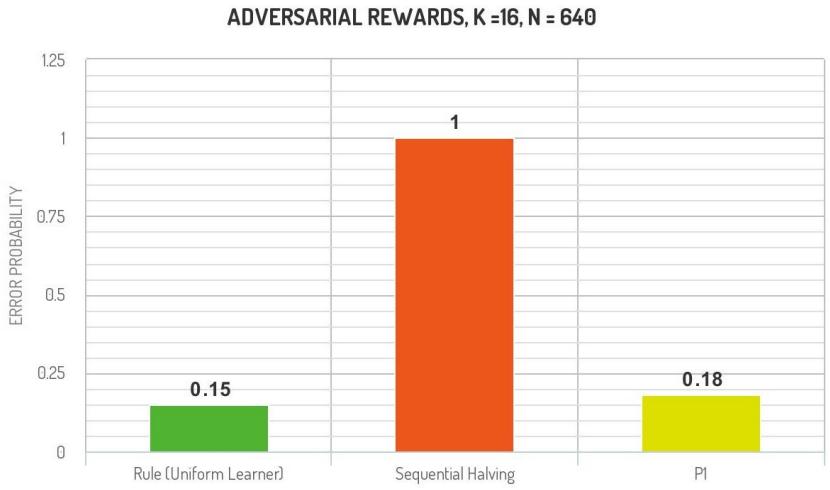
Theorem (Upper Bound for P1)

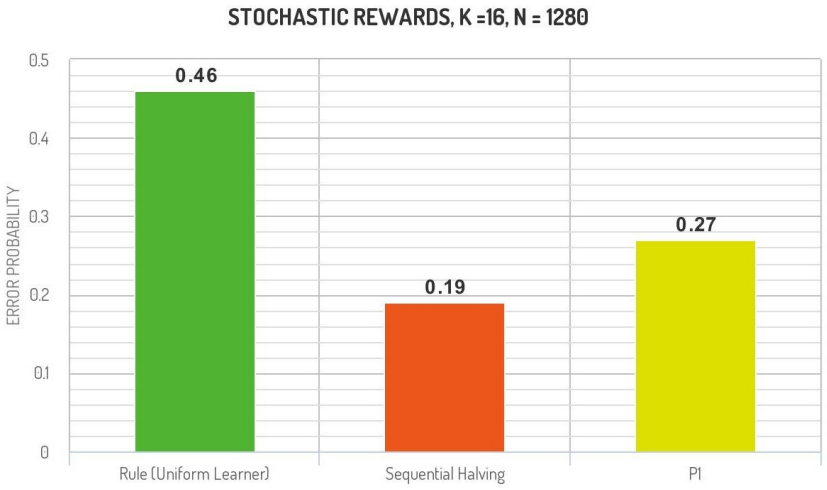
For any stochastic problem $STO(\mu)$, for any adversarial problem $ADV(g)$, the probability of error of P1 in respective environments

$$e_{STO(\mu)}(n) \leq 2K^3 n \cdot \exp(-\frac{n}{128 H_{P1}(\mu)})$$

$$e_{ADV(g)}(n) \leq K \cdot \exp\left(-\frac{3n}{40\overline{\log}(K) H_{UNIF}(g)}\right)$$

Experiments





References

Abbasi-Yadkori, Y., Bartlett, P., Gabillon, V., Malek, A., and Valko, M. (2018).
Best of both worlds: Stochastic & adversarial best-arm identification.
In *Conference On Learning Theory*, pages 918–949.

Karnin, Z., Koren, T., and Somekh, O. (2013).
Almost optimal exploration in multi-armed bandits.
In *International Conference on Machine Learning*, pages 1238–1246.

Domain Adaptation in Classification

Brian Lu

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.

Outline

- Motivation
- Formal Setup
- Theoretical Results
- Empirical Results

Motivation

- Statistical learning works well when the training and testing samples come from the same distribution.
- However, it is often the case that the distribution we sample from during training and testing are different.
  e.g. spam filtering, sentiment analysis.
- So it is important that
  1) We understand the behavior of our source-trained models on the target domain.
  2) We develop methods that can do well on target domain when most of the data is from the source.

Formal Setup

- Classification Task
- Input X from some domain D
- Output Y related to X by label function f
- A domain is the pair <D,f>
- Domain changes from training to testing

  Source $<D_S,f_S>$ to Target $<D_T,f_T>$
- Two questions:
  - When do source-trained classifiers perform well on target?
  - How do we mix source and target training to minimize target error?

Main Results I

Error bound for target domain

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Very Large Data Bases*, pages 180-191, 2004.

Main Results I

Error bound for target domain

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

where

$$\epsilon_D(h) = \mathbb{E}_{x \sim D}[|h(x) - f(x)|]$$

$$\mathcal{H}\Delta\mathcal{H} := \{g | g(x) = h(x) \oplus h(x)', h, h' \in \mathcal{H}\}$$

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{h \in \mathcal{H}\Delta\mathcal{H}} |P_{D_T}(I(h)) - P_{D_T}(I(h))|$$

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Very Large Data Bases*, pages 180-191, 2004.

Main Results I Error bound for target domain

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

where

$$\epsilon_D(h) = \mathbb{E}_{x\sim D}[|h(x) - f(x)|]$$
$$\mathcal{H}\Delta\mathcal{H} := \{g | g(x) = h(x) \oplus h(x)', h, h' \in \mathcal{H}\}$$
$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{h\in\mathcal{H}\Delta\mathcal{H}} |P_{D_T}(I(h)) - P_{D_T}(I(h))|$$

And with probability $1 - \delta$

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) \leq d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_S, \mathcal{U}_T) + 4\sqrt{\frac{2d \log m' + \log \frac{2}{\delta}}{m'}}$$

where $\mathcal{U}_S$ and $\mathcal{U}_T$ are the empirical distributions of the m' samples from $\mathcal{D}_S$ and $\mathcal{D}_T$ respectively, and d is the VC dimension of $\mathcal{H}$.

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Very Large Data Bases*, pages 180-191, 2004.

Main Results I Error bound for target domain

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda$$

where

$$\epsilon_D(h) = \mathbb{E}_{x\sim D}[|h(x) - f(x)|]$$
$$\mathcal{H}\Delta\mathcal{H} := \{g | g(x) = h(x) \oplus h(x)', h, h' \in \mathcal{H}\}$$
$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{h\in\mathcal{H}\Delta\mathcal{H}} |P_{D_T}(I(h)) - P_{D_T}(I(h))|$$
$$\lambda = \min_{h\in\mathcal{H}} \epsilon_S(h) + \epsilon_T(h)$$

And with probability $1 - \delta$

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) \leq d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_S, \mathcal{U}_T) + 4\sqrt{\frac{2d \log m' + \log \frac{2}{\delta}}{m'}}$$

where $\mathcal{U}_S$ and $\mathcal{U}_T$ are the empirical distributions of the m' samples from $\mathcal{D}_S$ and $\mathcal{D}_T$ respectively, and d is the VC dimension of $\mathcal{H}$.

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Very Large Data Bases*, pages 180-191, 2004.

Main Results II

Optimal Mixing Value

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \alpha \hat{\epsilon}_T(\hat{h}) + (1-\alpha)\hat{\epsilon}_S(\hat{h})$$

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.

Main Results II

Optimal Mixing Value

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \alpha \hat{\epsilon}_T(\hat{h}) + (1-\alpha)\hat{\epsilon}_S(\hat{h})$$

$$\epsilon_T(\hat{h}) \leq \epsilon_T(h_T^*) + 4\sqrt{\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}}\sqrt{\frac{2d\log(2(m+1)) + 2\log(\frac{8}{\delta})}{m}}$$

$$+ 2(1-\alpha)\left(\frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_S, \mathcal{U}_T) + 4\sqrt{\frac{2d\log(2m') + \log(\frac{8}{\delta})}{m'}} + \lambda\right).$$

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.

Main Results II

Optimal Mixing Value

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \alpha \hat{\epsilon}_T(\hat{h}) + (1-\alpha)\hat{\epsilon}_S(\hat{h})$$

$$\epsilon_T(\hat{h}) \leq \epsilon_T(h_T^*) + 4\sqrt{\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}}\sqrt{\frac{2d\log(2(m+1)) + 2\log(\frac{8}{\delta})}{m}}$$

$$+ 2(1-\alpha)\left(\frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_S, \mathcal{U}_T) + 4\sqrt{\frac{2d\log(2m') + \log(\frac{8}{\delta})}{m'}} + \lambda\right).$$

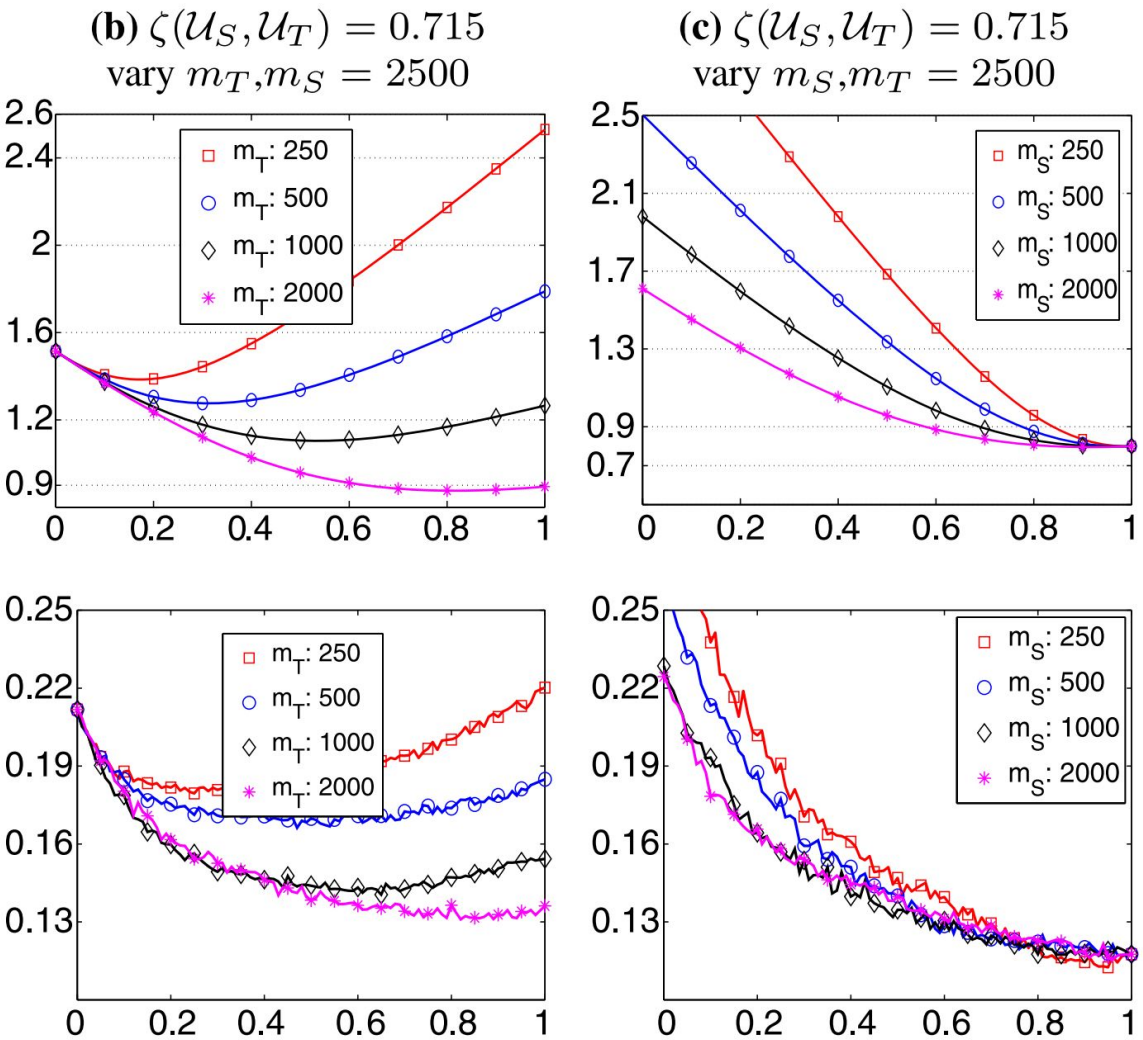
$$\alpha^*(m_T, m_S; D) = \begin{cases} 1 & m_T \geq D^2 \\ \min\{1, \nu\} & m_T \leq D^2, \end{cases}$$

where

$$\nu = \frac{m_T}{m_T + m_S}\left(1 + \frac{m_S}{\sqrt{D^2(m_S + m_T) - m_S m_T}}\right).$$

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.

Empirical Results [from the paper]



Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.

References

- Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. JMLR, 79, 2010.
- Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Very Large Data Bases*, pages 180-191, 2004.

10-716 Project
Invariant Risk Minimization (IRM)

Amrit Singhal (amritsin)
Shantanu Gupta (shantang)

Spring 2020

Setup

- High level goal - learn a predictor that generalized to unseen environments.
- Assume that the data generating causal process remains the same. As an example, consider the following:

$$
\begin{aligned}
X_1 &\leftarrow \text{Gaussian}(0, \sigma^2), \\
Y &\leftarrow X_1 + \text{Gaussian}(0, \sigma^2), \\
X_2 &\leftarrow Y + \text{Gaussian}(0, 1).
\end{aligned}
$$

- Multiple environments are generated by intervening on different subsets of variables (say X_1 in one and X_2 in another).
- Using data from from several environments, generalize to unseen all unseen environments.

The IRM Objective

- Invariant Predictor - one that ignores spurious correlations across environments.
- Let $\Phi : \mathcal{X} \to \mathcal{H}$ be a data representation. A classifer $w : \mathcal{H} \to \mathcal{Y}$ is invariant if $w \in \arg\min_{\tilde{w}} R^e(\tilde{w} \circ \Phi) \forall e \in \mathcal{E}$.
- This can be phrased as the following constrained optimization problem (also known as IRM):

$$\min_{\Phi, w} \sum_{e \in \mathcal{E}_{\text{tr}}} R^e(w \circ \Phi)$$
$$\text{subject to } \; w \in \arg\min_{w'} R^e(w' \circ \Phi), \; \forall e \in \mathcal{E}_{\text{tr}}.$$

- Impractical to optimize the objective directly.

IRMv1

- A relaxation of the IRM objective (known as IRMv1) is proposed:

$$\min_{\Phi} \sum_{e \in \mathcal{E}_{\text{tr}}} \underbrace{R^e(\Phi)}_{\text{ERM}} + \lambda \underbrace{\|\nabla_{w|w=1.0} R^e(w \cdot \Phi)\|^2}_{\text{invariance penalty}},$$

  where Φ becomes the invariant predictor, $w = 1.0$ is a scalar fixed *dummy* classifier and $\lambda \in [0, \infty)$ is a hyperparameter.
- Φ can be a non-linear predictor.
- The IRMv1 objective can be optimized using gradient based methods.

Obtaining IRMv1

- Convert the hard constraints in IRM to a penalty $D(w, \Phi, e)$ in the loss. This penalty should capture how well w minimizes the environment risk.
- In the case of linear classifiers w, the authors propose the penalty

$$D_{\text{lin}}(w, \Phi, e) = \|\mathbb{E}_{X^e}[\Phi(X^e)^T \Phi(X^e)]w - \mathbb{E}_{X^e, Y^e}[\Phi(X^e)^T Y^e]\|^2$$

- We can write the predictor $w \circ \Phi = (w \circ \Psi) \circ (\Psi^{-1} \circ \Phi)$. So, keeping the predictor $w \circ \Phi$ to be the same, we can fix w to any constant $\tilde{w}$ and perform optimisation over just Φ.
- More generally, rewrite the penalty as $D(w = 1.0, \Phi, e) = \|\nabla_{w|w=1.0} R^e(w.\Phi)\|^2$.

Generalisation for IRM

- IRM gives us invariant predictors with low error across all *training* environments. What about *all* environments?
- We need that our training environments have "sufficient" diversity.
- For linear case, this is formalised by requiring the training environments to be in a *linear general position*, which basically limits the extent to which the training environments can be co-linear.
- This constraint is not very restrictive, as the set that doesn't satisfy this condition has a measure zero.

Theorem

A representation Φ of rank r leads to an invariant predictor across all training environments, lying in a linear general position of rank r, *iff* the predictor is invariant across *all* environments.

Causation as invariance

- Invariance is promoted as the main feature of causation.
- Show that a predictor is invariant across all unseen environment if and only if it uses the direct causal parents of Y as input.
- The other variables are responsible for spurious correlations that hinder generalization.

Experiments

- MSE on Synthetic Data (Linear case)

Dataset	ERM Validation	IRM Validation
Fully observed	3.067	2.112
Partially observed	4.719	4.212
Heteroskedastic	3.489	3.130

- In our experiments, we found that IRM did not work well with many spurious correlations.
- Accuracy on MNIST dataset

Setup	Training Acc.	Testing Acc.
IRM, Colored MNIST	0.7036	0.6678
ERM, colored MNIST	0.8742	0.1681
ERM, Grayscale MNIST	0.7358	0.7302

Thank You

Augmenting Generative Adversarial Networks through Application-Specific Knowledge

Maxwell B. Wang

MD/PhD Student

GANs are great but...

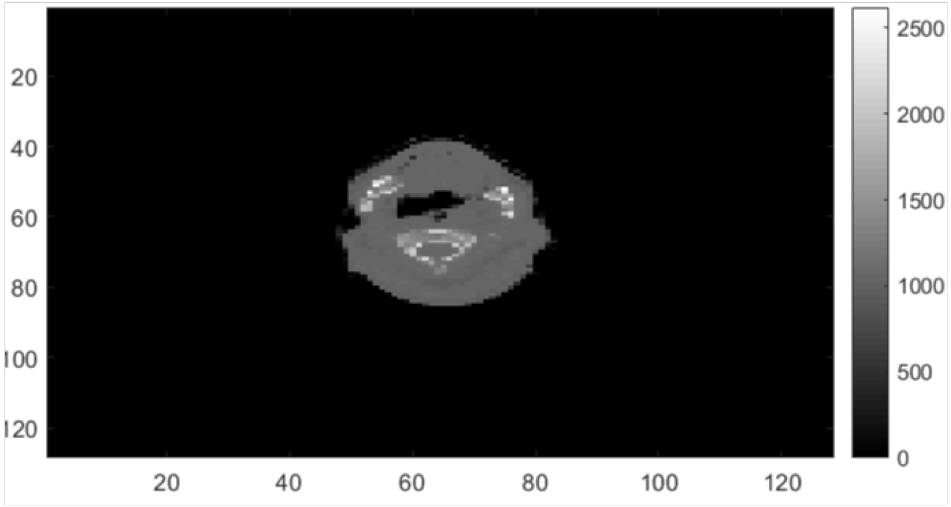
- GANs have seen incredible progress in generating samples with similar statistics from a given dataset

GANs are great but...

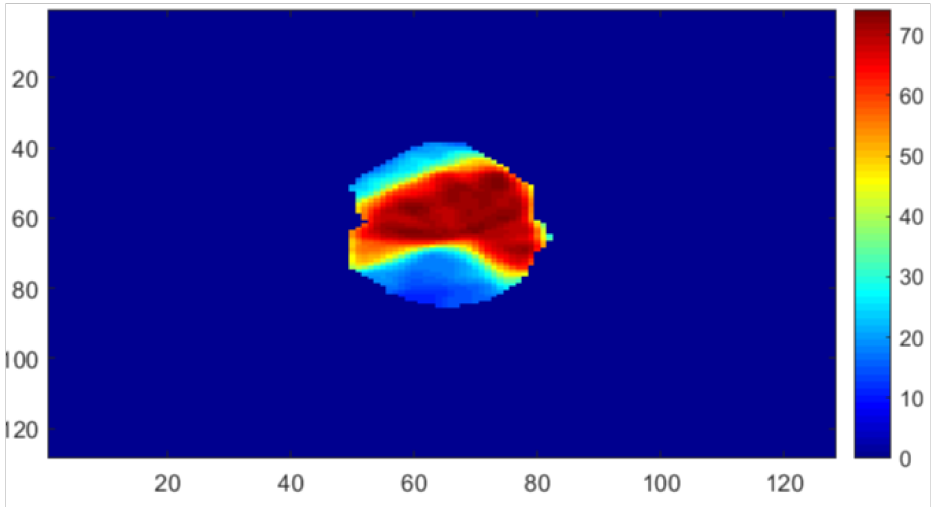
- GANs have seen incredible progress in generating samples with similar statistics from a given dataset
- Being CNNs, they require large datasets to train

Motivating Example

CT Image of Tumor



Optimal Radiation Dosage Planning

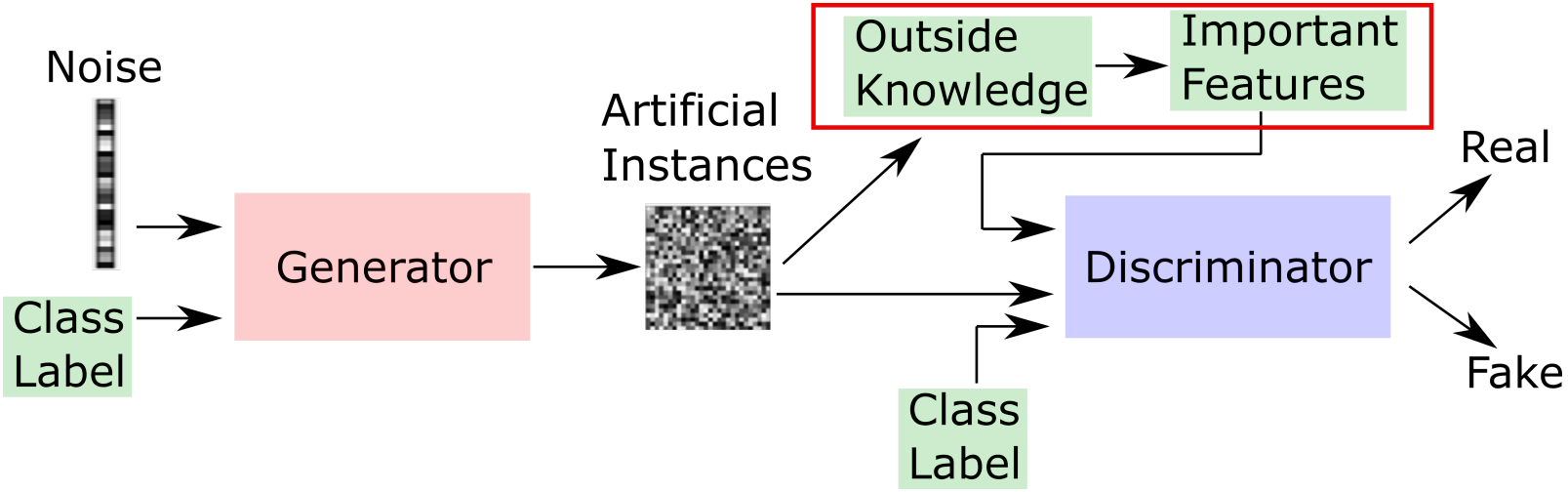


Goal

Hypothesis

Can we improve GAN performance in the low-sample size regime by incorporating application-specific knowledge/features into the training process?

Augmented GAN Architecture



Assessment

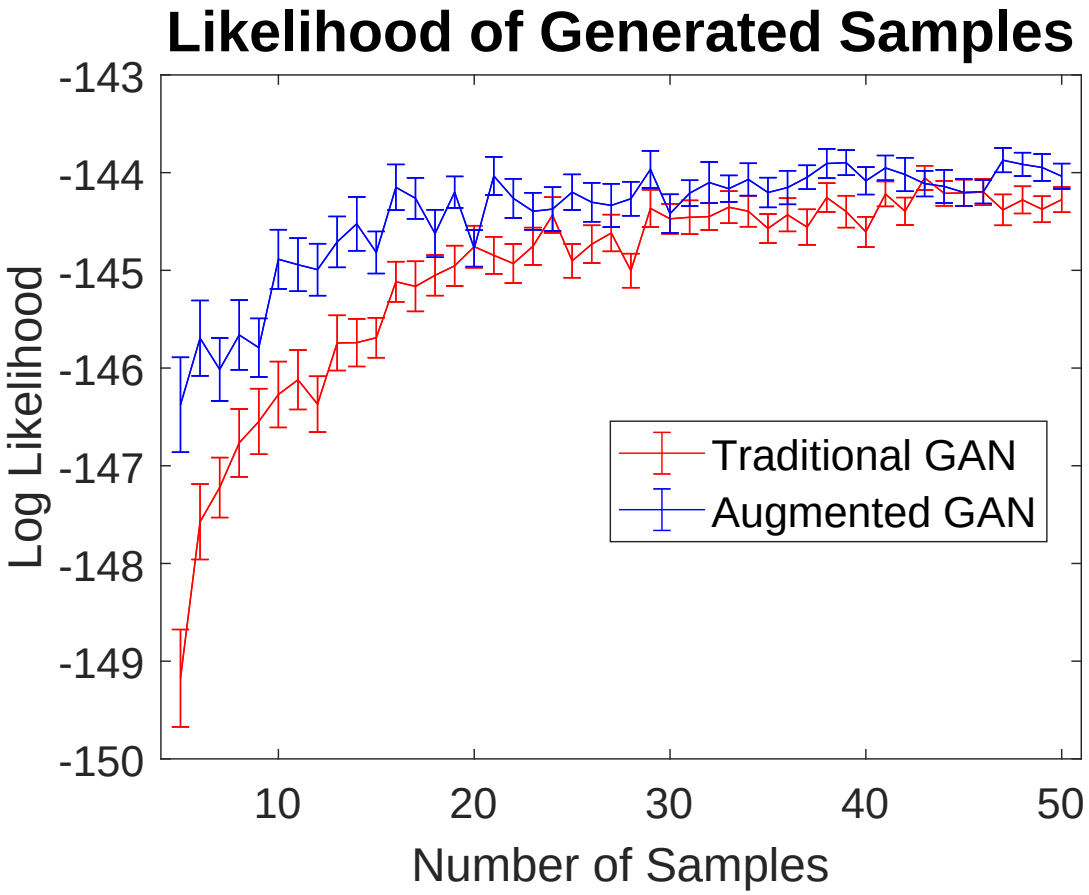
- Simulated data from a probability distribution inspired by radiation beam physics
- OpenKBP Challenge of Radiation Planning: 200 CT images w/ marked tumors and vulnerable organs

Simulated Test

- Gaussian Bayesian Network where a quarter of the edges were known as our "application-specific knowledge"

Simulated Test

- Gaussian Bayesian Network where a quarter of the edges were known as our "application-specific knowledge"



Acknowledgements

- Ian Char
- Kartik Gupta
- Tom Yan
- Sean Jin
- Prof. Pradeep Ravikumar
- Profs. Avniel Ghuman, Max G'Sell

A-DNN: From Additive Model to Additive Dense Neural Network

Yuning Wu[1]

[1]Carnegie Mellon University

Background & Questions

- Most current research focus on performance of neural networks. In many cases, architecture of neural network is assigned (# of nodes, # of layers, etc.), and logic behind such assignment is unclear.

- Current neural networks are static and un-adaptive.

- Research on general additive models (GAM) has proposed a more flexible schema in both model structure and learning update (back-fitting).

Key Research Points

- Apply methods and philosophy of additive models to neural networks, current scope being dense neural net.

- Expand notion of “additive” into neural nets’ structural additivity, i.e., how to “grow” a neural network from void using building blocks like unit node and layer, instead of arbitrarily assigning or tuning for a final static structure.

- Performance and efficiency evaluation of such additive model design schema.

Research Progress

- Design and testing of proposed additive model schema.

- Theoretical
  - Adaptation of theoretical background in GAM into the designed schema.

- Experimental
  - Classical regression scenarios.
  - AutoML experiments.

Carnegie Mellon University

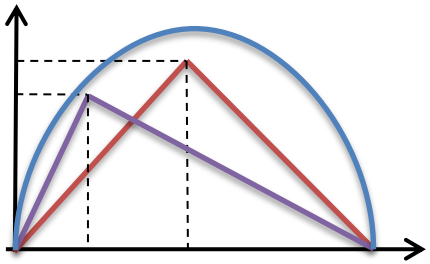
Thank you.

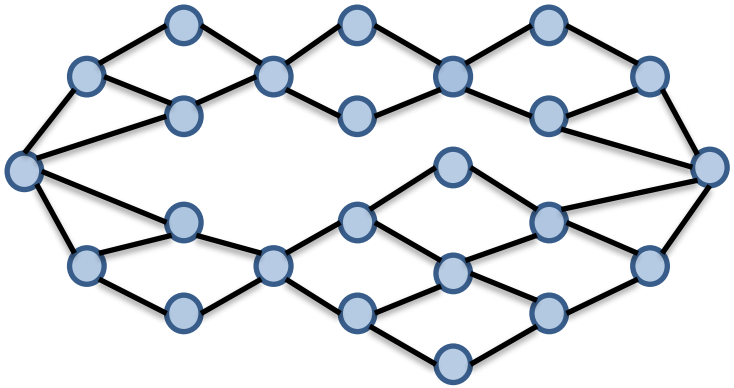
10716 Advanced Machine Learning - April 30, 2020

Optimal Decision Making of Inspection and Maintenance in Network Systems

Chaochao Lin

Carnegie Mellon University





Exploration of networked systems

Assumptions:

A networked system composed of binary components.

P_i is the probability of malfunctioning component i;

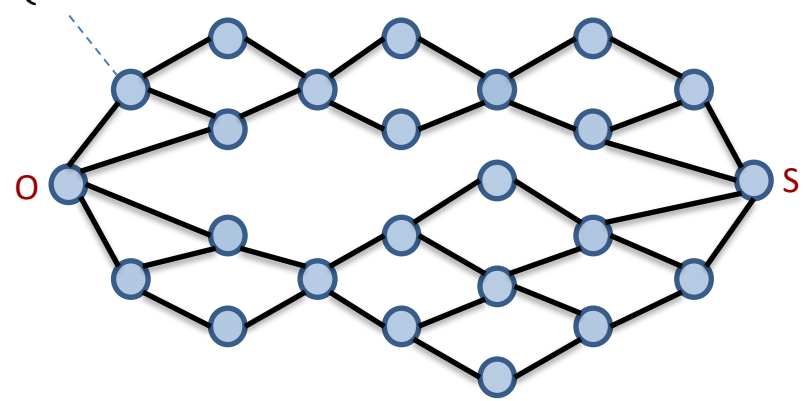
The system's state is also binary: it works only if there is an intact path from origin to sink.

P_s is the system's failure probability.

Management of transportation and energy networks can be improved by placing sensors and installing monitoring systems.

Question: Identify metric M so that $M_i > M_j$ if it is more appropriate inspecting component j.

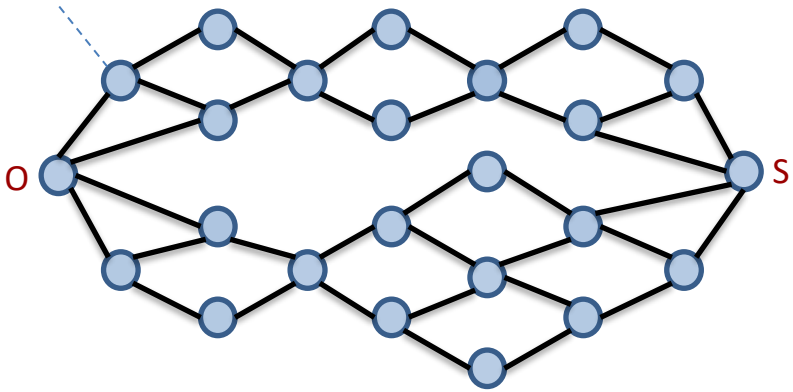
$$s_i = \begin{cases} 0 & \text{if work.} \\ 1 & \text{if not work.} \end{cases}, \quad P_i = \mathbb{P}[s_i = 0]$$



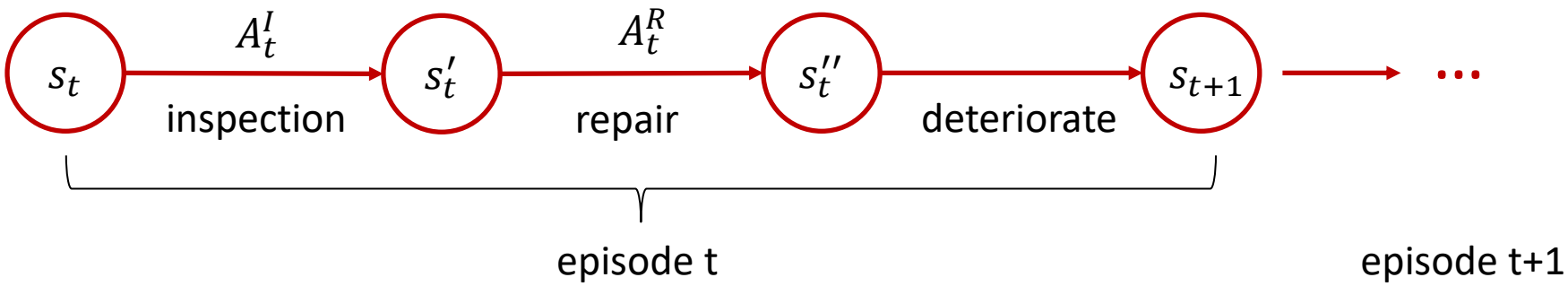
Exploration of networked systems

Methods:

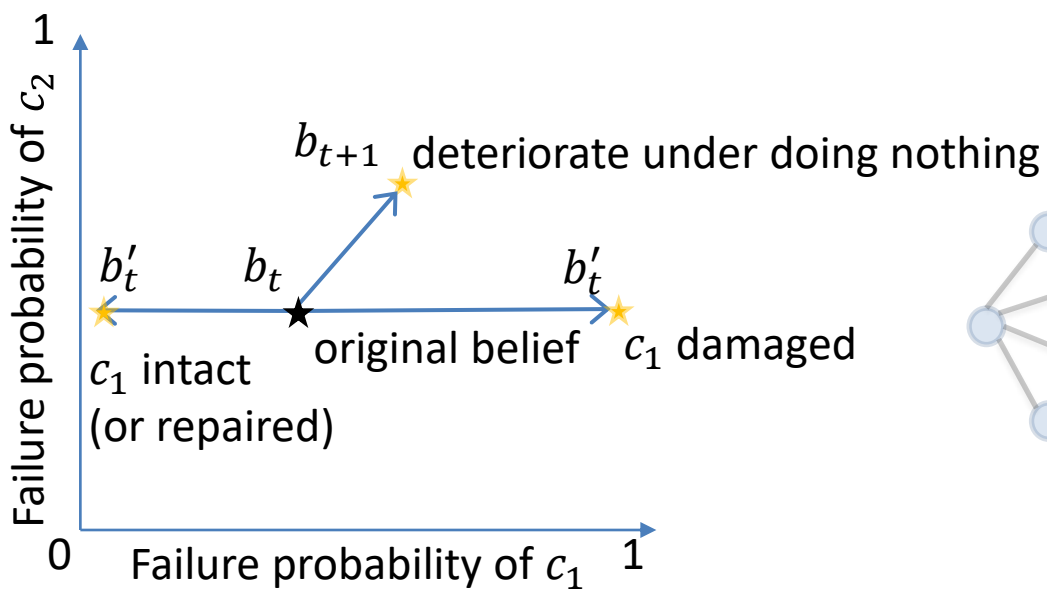
1. Partially Observable Markov Decision Process model
2. Bayesian prior-posterior analysis based on Knowledge Gradient

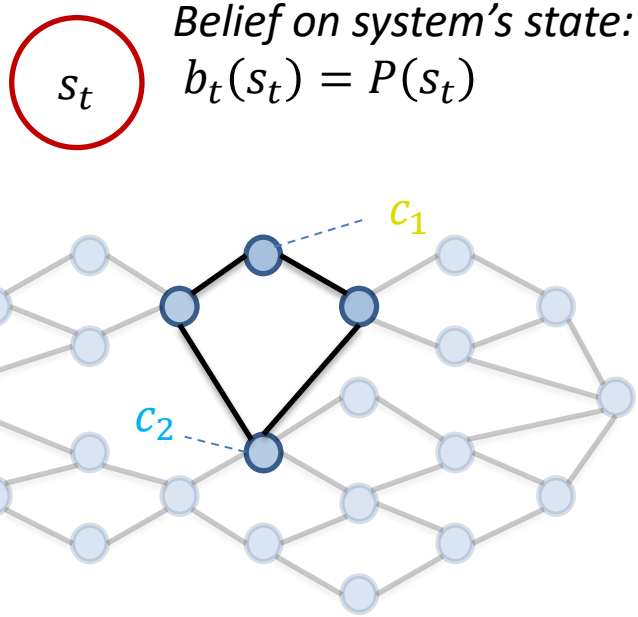


Partially Observable Markov Decision Process model



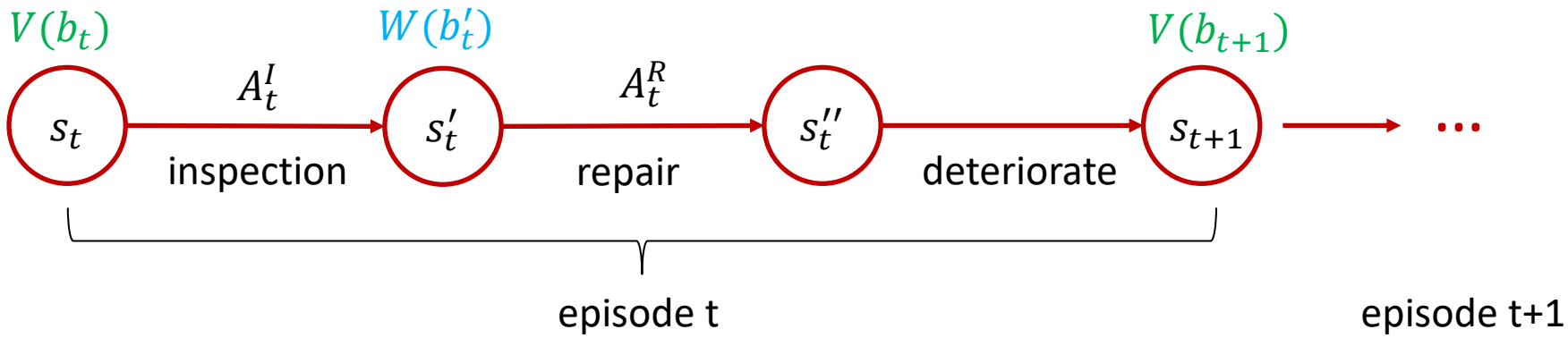
Belief transition for c_1





Belief on system's state:
$b_t(s_t) = P(s_t)$

Belief Markov Decision Process model



Value functions:

$V(b_t)$ is the value at the beginning of each step, before inspection;

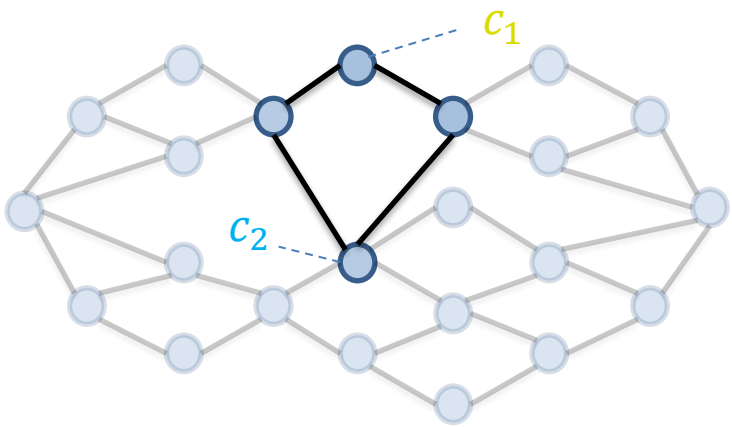
$W(b_t')$ is the value of the state s_t' after inspection and before repairing.

Bellman Equations:

$$W(b_t') = \min_{A_t^R}[C(A_t^R) + \gamma\, V(b_{t+1})]$$

$$V(b_{t+1}) = \min_{A_t^I} \mathbb{E}[W(b_{t+1}')]$$

estimation via SARSOP method.

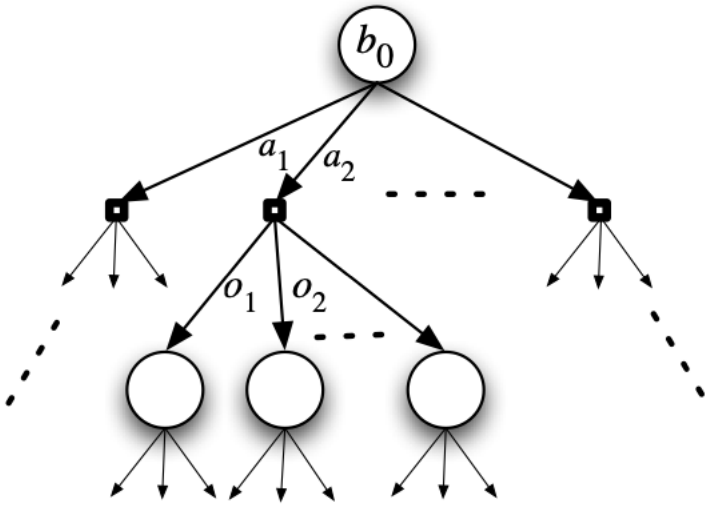


Successive Approximation of the Reachable Space under Optimal Policies: SARSOP

- SARSOP was first introduced to solve robotic tasks: navigation, grasping, target tracking, etc.
- Key idea is to sample a representative set of points from the belief space and use it as an approximate representation of the space.

SARSOP iterates over main functions:

1. SAMPLE
2. BACKUP
3. PRUNE



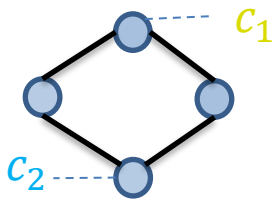
Kurniawati, H., Hsu, D. and Lee, W.S., 2008, June. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems* (Vol. 2008).

Successive Approximation of the Reachable Space under Optimal Policies: SARSOP

Ratio between the costs $C_F/C_{R_i} = 5$

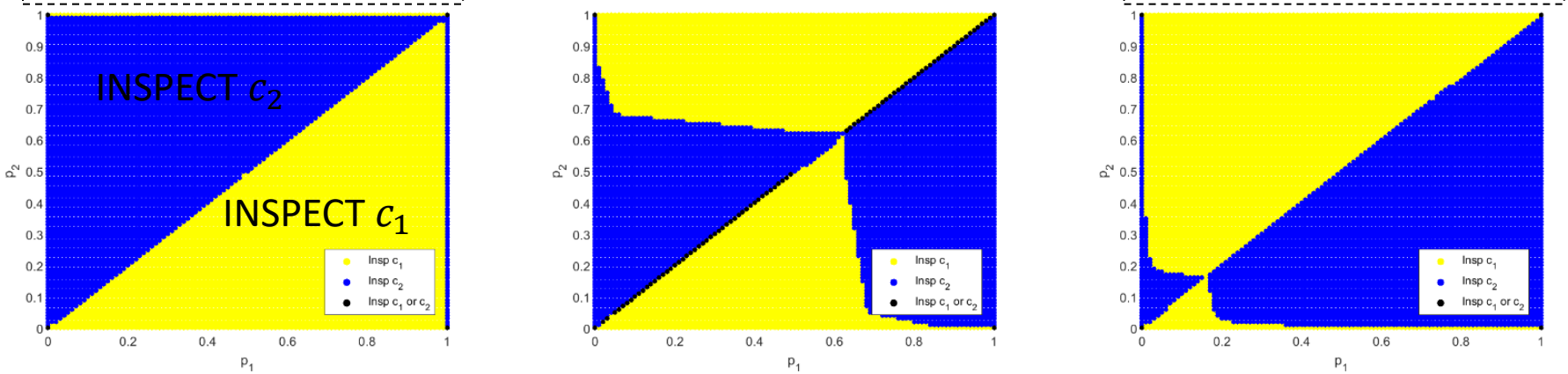
Deterioration rate $\delta_1 = \delta_2 = 1\%$

Inspection type 1 and type 2 error rate are both 0.01



Inspect the more vulnerable component

Inspect the more reliable component



(a) $\gamma = 0.99$ **(b)** $\gamma = 0.96$ **(c)** $\gamma = 0.80$

Kurniawati, H., Hsu, D. and Lee, W.S., 2008, June. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems* (Vol. 2008).

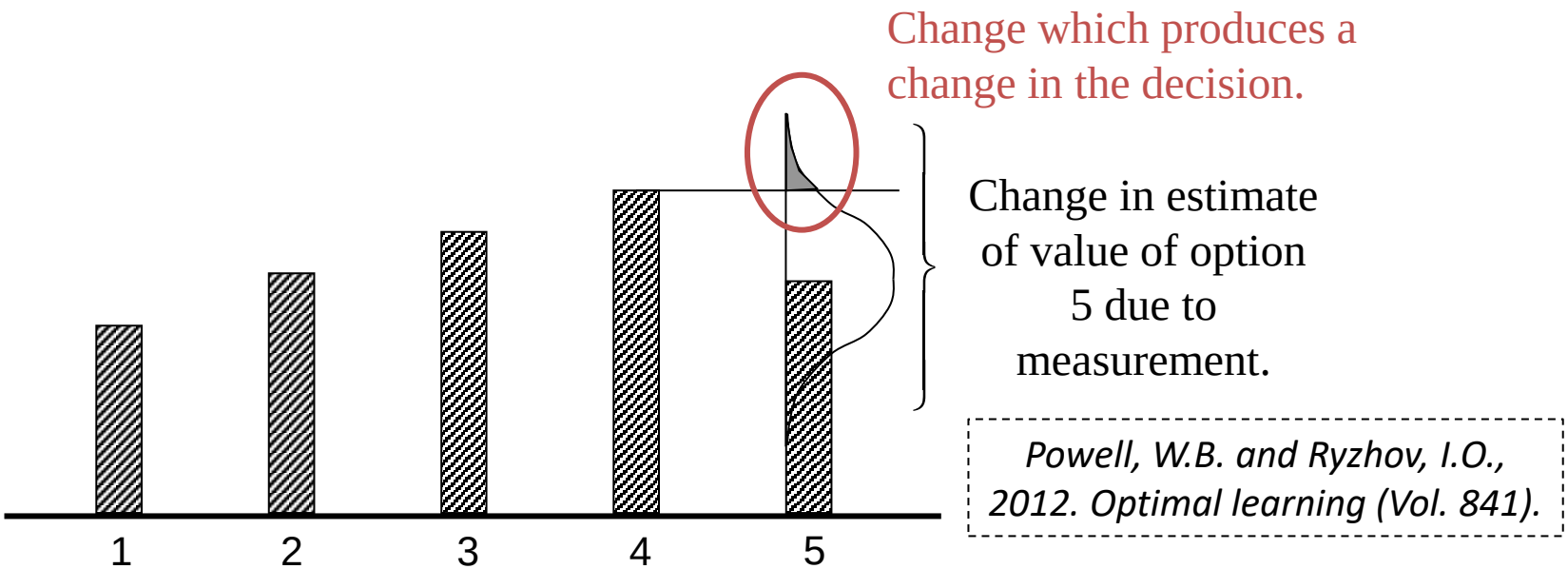
Bayesian prior-posterior analysis with Knowledge Gradient Method

Basic principle:

Assume you can make only one measurement, after which you have to make a final choice (the implementation decision).

What choice would you make now to maximize the expected value of the implementation decision?

$$\nu_x^{KG,n} = E\left\{\max_y F(y, K^{n+1}(x))\right\} - \max_y F(y, K^n)$$



Powell, W.B. and Ryzhov, I.O., 2012. Optimal learning (Vol. 841).

Bayesian prior-posterior analysis with Knowledge Gradient Method

Assume that all components have a Bayesian prior as: $p_i \sim Beta(\alpha_i^0, \beta_i^0)$

At state $s^n = (\alpha^n, \beta^n)$, the true failure probability $p_i \sim Beta(\alpha_i^n, \beta_i^n)$

The inspection x^n gives outcome $W_{x^n}^{n+1} \mid s^n \sim Bernoulli(p_{x^n})$

Since Beta is a conjugate prior for the Bernoulli, the posterior belief on $p_{\{x^n\}}$:

$$p_{x^n} \mid s^n, W_{x^n}^{n+1} \sim Beta(\alpha_{x^n} + W_{x^n}^{n+1}, \beta_{x^n} + 1 - W_{x^n}^{n+1})$$

Different value function form:

1. $V^n(s^n) = \max_i \frac{\alpha_i}{\alpha_i + \beta_i}$, which can be interpreted as to identify the most vulnerable component in a system. Then:

- If $C_{x^n}^n \leq \frac{\alpha_{x_n}^n}{\alpha_{x_n}^n + \beta_{x_n}^n + 1}$ or $C_{x^n}^n \geq \frac{\alpha_{x_n}^n + 1}{\alpha_{x_n}^n + \beta_{x_n}^n + 1}$, $\nu_{x_n}^{KG,n} = 0$;
- If $\frac{\alpha_{x_n}^n}{\alpha_{x_n}^n + \beta_{x_n}^n + 1} \leq C_{x^n}^n \leq \frac{\alpha_{x_n}^n}{\alpha_{x_n}^n + \beta_{x_n}^n}$, $\nu_{x_n}^{KG,n} = \frac{\beta_{x_n}^n}{\alpha_{x_n}^n + \beta_{x_n}^n}\left(C_{x_n}^n - \frac{\alpha_{x_n}^n}{\alpha_{x_n}^n + \beta_{x_n}^n + 1}\right)$;
- If $\frac{\alpha_{x_n}^n}{\alpha_{x_n}^n + \beta_{x_n}^n} \leq C_{x^n}^n \leq \frac{\alpha_{x_n}^n + 1}{\alpha_{x_n}^n + \beta_{x_n}^n + 1}$, $\nu_{x_n}^{KG,n} = \frac{\alpha_{x_n}^n}{\alpha_{x_n}^n + \beta_{x_n}^n}\left(\frac{\alpha_{x_n}^n + 1}{\alpha_{x_n}^n + \beta_{x_n}^n + 1} - C_{x_n}^n\right)$;

Algorithmic Framework for Model-based Deep RL with Theoretical Guarantees

Arundhati Banerjee Ashwini Pokle

Machine Learning Department
Carnegie Mellon University

30th April 2020

Motivation

- Model-based RL (MB-RL) algorithms are empirically known to be sample efficient (Nagabandi *et al.*'17, Kurutach *et al.*'17) but their theoretical understanding is limited.
- We analyze a framework for MB-RL from Luo *et al.*[1] that guarantees monotonic convergence to a local maximum of the reward.

[1] Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees, Yuping Luo and Huazhe Xu and Yuanzhi Li and Yuandong Tian and Trevor Darrell and Tengyu Ma, *ICLR*, 2019

Notations and Preliminaries

We assume the standard RL set up for continuous state and action space

- Value Function

$$V^{\pi,M}(s) = \mathop{\mathrm{E}}_{\substack{\forall t \geq 0, A_t \sim \pi(\cdot|S_t) \\ S_{t+1} \sim M(\cdot|S_t, A_t)}} \Big[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) | S_0 = s \Big]$$

  where, $V^{\pi,M}(s)$ is value function at state s under the estimated dynamical model, M is the estimated dynamical model, and Π is a family of parameterized policies.

- State distribution induced by policy

$$\rho^{\pi} = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \cdot p_{S_t^{\pi}}$$

Challenges in Model Based RL

What loss to use for learning a dynamical model?

- Existing Approach: Mean Squared Error

$$||M(s,a) - s'||_2^2$$

- Issue: Not variant to change of representation
- Proposed: Good loss for $M \approx$ error for predicting future rewards

$$|V^{\pi,M} - V^{\pi,M*}|$$

Plan

- Upper bound the reward discrepancy and use it as a loss function

$$|V^{\pi,M} - V^{\pi,M^*}| \leq \mathrm{E}_{(s,a,s')\sim\pi,M^*}\Big[\underbrace{|V^{\pi,M}(M(s,a)) - V^{\pi,M}(s')|}_{\text{Loss function for learning M}}\Big]$$

- **Cons**: Requires samples from environment M^* to estimate loss
- **Pros**: Invariant to state representation

Modified Discrepancy Bound Design

$\forall\pi$ close to π_{ref},

$$|V^{\pi,M} - V^{\pi,M^*}| \leq \mathrm{E}_{(s,a,s')\sim\pi_{\text{ref}},M^*}\Big[\underbrace{|V^{\pi,M}(M(s,a)) - V^{\pi,M}(s')|}_{\text{Discrepancy bound } D_{\pi_{\text{ref}}}(M,\pi)}\Big]$$

Meta-Algorithm for Model-based RL

Algorithm 1: Meta-Algorithm for Model-based RL

Input: Initial policy π_0. Discrepancy bound D and distance function d

for $k = 0$ to T **do**

$$\pi_{k+1}, M_{k+1} = \underset{\pi \in \Pi, M \in \mathcal{M}}{\arg\max} \; V^{\pi,M} - D_{\pi_k,\delta}(M, \pi) \tag{1}$$

s.t. $d(\pi, \pi_k) \leq \delta$

end for

Convergence Guarantee

Theorem

Suppose that $M^* \in \mathcal{M}$ and the optimization problem in equation (1) is solvable at each iteration. Then, Algorithm 1 produces a sequence of policies $\pi_0, \ldots, \pi_T$ with monotonically increasing values:

$$V^{\pi_0, M^*} \leq V^{\pi_1, M^*} \leq \ldots \leq V^{\pi_T, M^*} \tag{2}$$

Moreover, as $k \to \infty$, the value V^{π_k, M^*} converges to some $V^{\bar{\pi}, M^*}$, where $\bar{\pi}$ is a local maximum of V^{π, M^*} in domain Π.

Empirical Results

- We analyzed Stochastic Lower Bound Optimization (SLBO) algorithm which is a practical implementation of the proposed algorithmic framework
- No Squared loss in model training is found to be crucial
- SLBO achieves near-optimal reward using model-based RL with a **single** model

Empirical Results

- SLBO achieves state-of-the-art performance with under 1 million samples

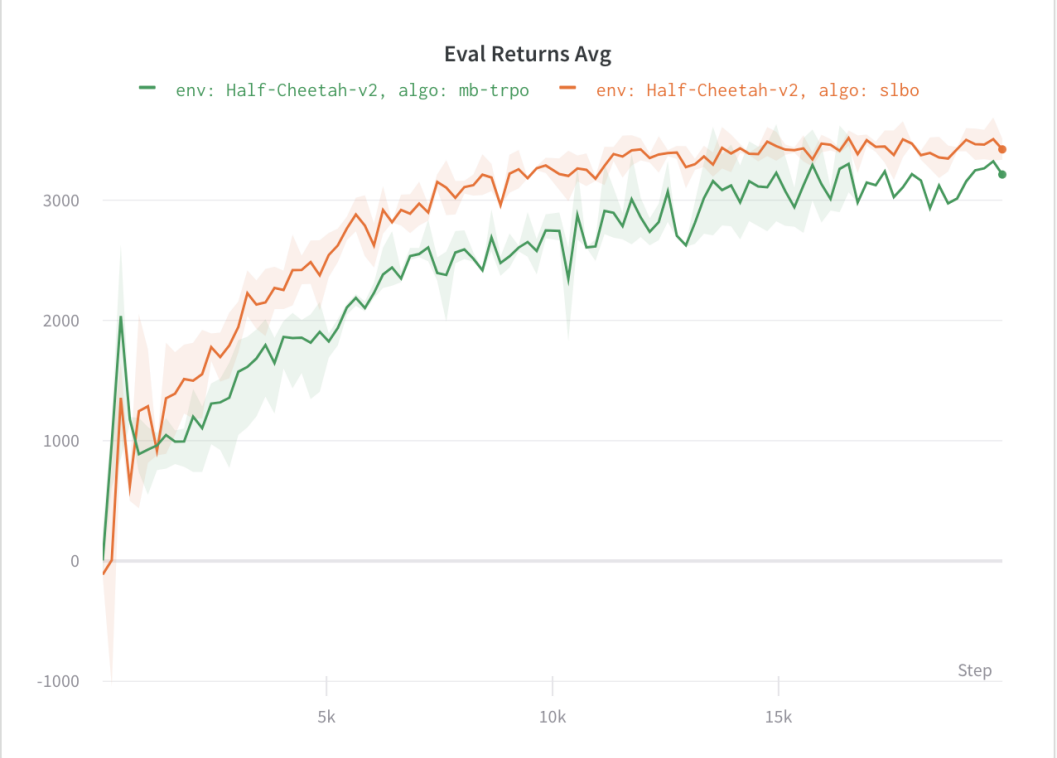


Figure: Total reward on Half-cheetah for SLBO and Model-based TRPO (MB-TRPO) plotted averaged over 3 runs for 20k steps. The dotted line is the mean and the shaded areas indicate the variance.

Key Takeaways

- Learn dynamical model via iterative optimization of the discrepancy bound
- Proposed framework ensures monotonic convergence to a local maximum of the reward
- No use of confidence interval for uncertainty estimation. It is implicit in the error bound.
- Joint optimization of M and π extends optimism-in-face-of-uncertainty principle to non-linear dynamic models.

A Comparative Analysis of Manifold Learning Algorithms

Jerry Ding
Carnegie Mellon University
Pittsburgh, April 27 2020

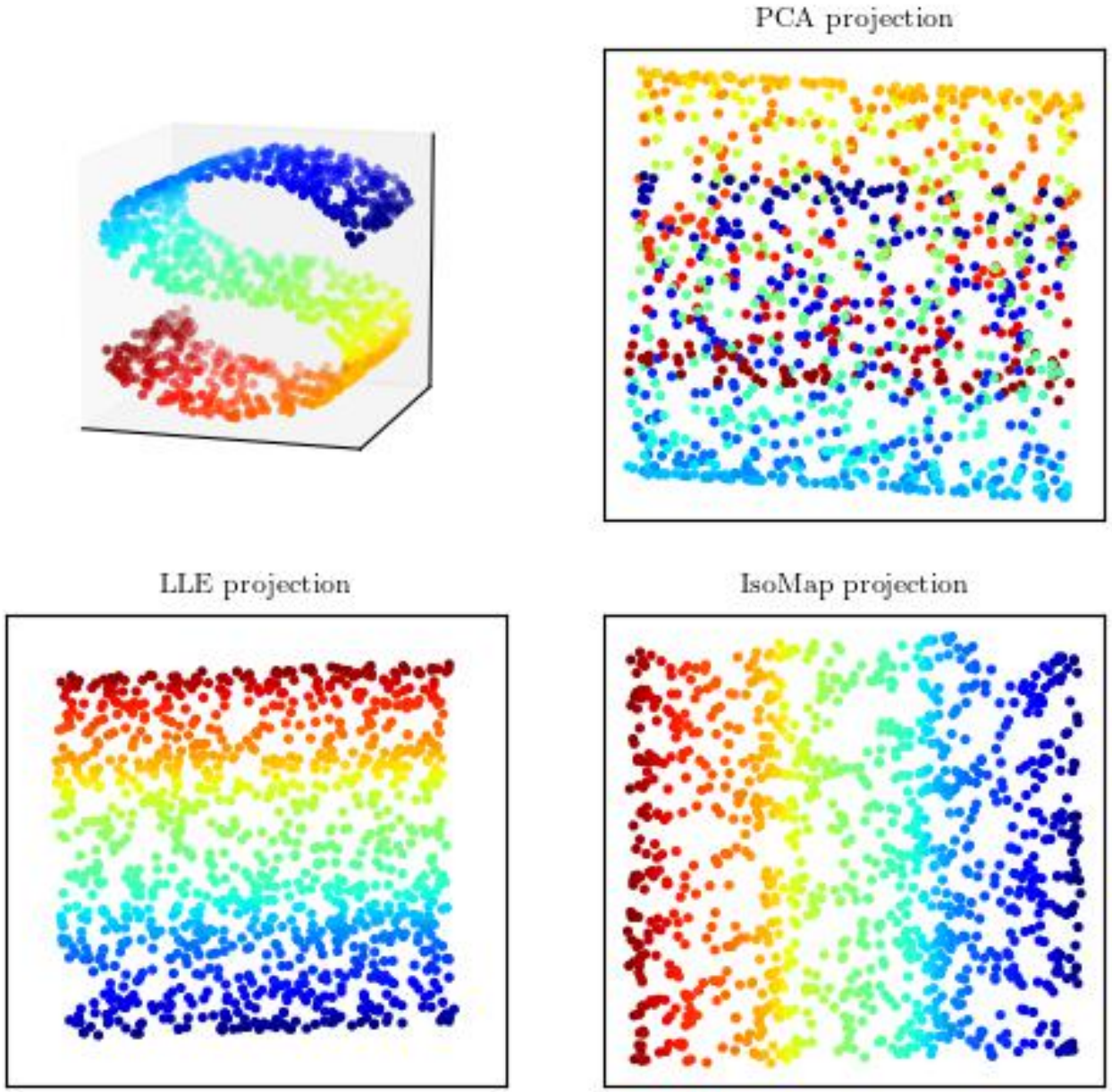
Manifold Learning

- Given data X such that $X = f(Y), f\ .\ \mathbb{R}^d \to \mathbb{R}^D \in \mathcal{C}_2$ injective
  - In other words, the data lies on a d-dimensional manifold immersed in D-dimensional space
- Want to learn the manifold structure and recover Y
  - At least up to a continuous & invertible change of coordinates
- Used primarily for visualization, we'll assume $d = 2$ or 3

Manifold Learning Algorithms

- Isomap
  - Run a graph search on k-nearest neighbors to estimate geodesic distances between all pairs of points
    - Geodesic means “shortest path lying on the manifold”
  - Use MDS on distance matrix to estimate coordinates

- LLE
  - Fit local linear combination of each point to k-nearest neighbors
  - Reconstruct points in d-dim space to minimize the reconst. error, subject to zero mean and identity covariance

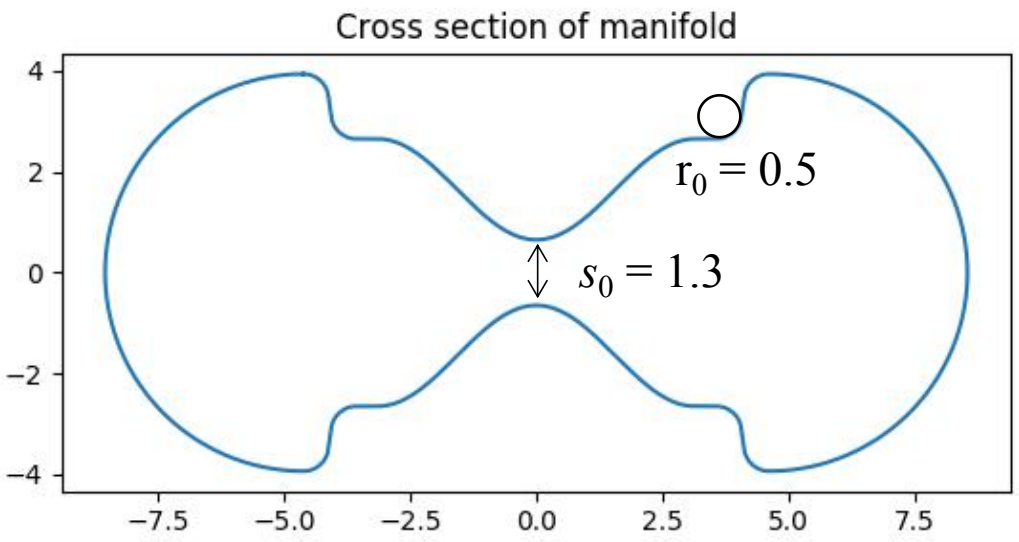

PCA projection
LLE projection
IsoMap projection

Round 1: Embedding Quality

- Isomap aims to find:
  1. Approximation of true geodesic distance between points
  2. Points in Euclidean space that minimizes the stress, i.e. difference between true distances and embedding's distances
- MDS finds the globally optimal solution to #2
- Even with perfect distance estimates from #1, final output distances may still be inaccurate due to curvature and topology, since the output lives in Euclidean space!

Result for Isomap

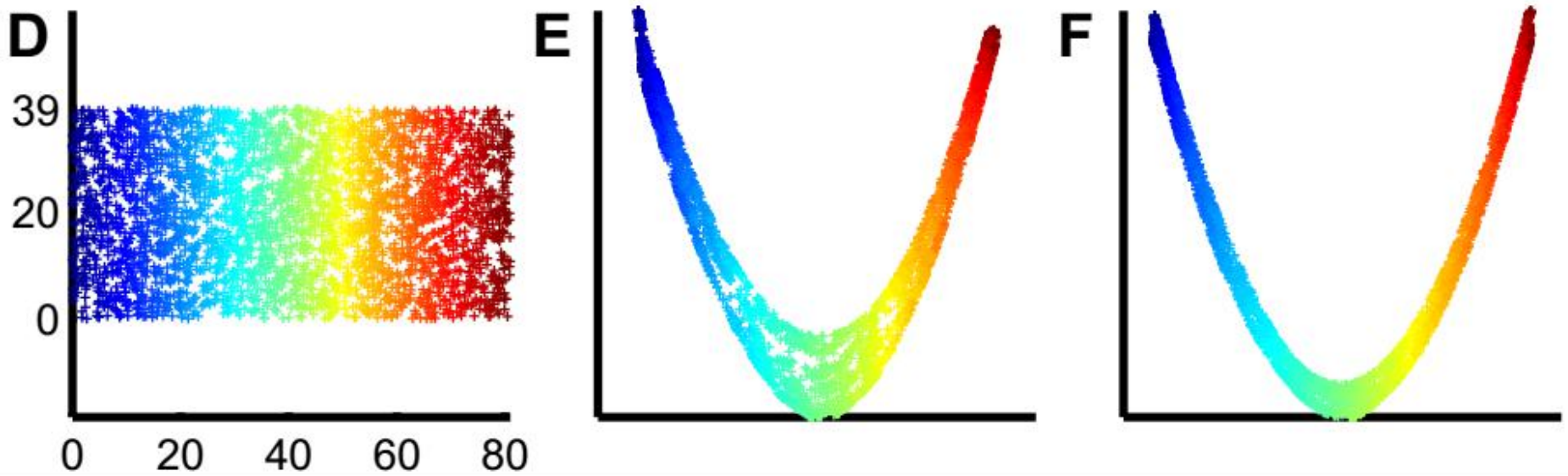
- From the paper “Graph Approximations to Geodesics on Embedded Manifolds” by Bernstein et al.
- If radius of curvature r_0 & branch distance s_0 is nonzero, and manifold is geodesically convex
  - i.e. manifold is compact and has no “cut-out” holes
- Then the geodesic distance estimates are consistent
- Experimental tests confirm the error bounds by creating manifolds with specific values of r_0, s_0




```
Using radius=0.50, separation=1.30, scale=1.00, closed=1,
count=10000, dims=2

**** Experiment 0 ****

** Theoretical results: **
delta must be at least 0.160974
epsilon must be between 0.643897 and 1.300000
Separation ensures lambda_1 <= 0.694985

* With the epsilon rule: *
epsilon = 1.161206 seems like a reasonable choice
The distances will be accurate within multiplicative factors
(0.445493, 1.554507)

* With the k rule: *
k needed to be between 290 and 417
k = 348 seems like a reasonable choice
The distances will be accurate within multiplicative factors
(0.445903, 1.689295)


** Test results: **

* With the epsilon rule: *
1.000000 dm <= ds <= 1.000601 dm
0.829387 ds <= dg <= 1.000000 ds
0.829387 dm <= dg <= 1.000403 dm

* With the k rule: *
1.000000 dm <= ds <= 1.001932 dm
0.855209 ds <= dg <= 1.000000 ds
0.855209 dm <= dg <= 1.001234 dm
```

Result for LLE

- From the paper “Manifold Learning: The Price of Normalization” by Goldberg et al.
- If the manifold has very different lengths along different dimensions, then a whole class of manifold learning algorithms (which includes LLE) will fail
- Simplest illustration: a sufficiently wide rectangle
  - Failure result: algorithm favors one-dimensional output even though the intrinsic dimensionality is 2


D
39
20
0
0
20
40
60
80
E
F

Round 2: Algorithm complexity

- For Isomap, graph search cost usually dominates
  - Using Dijkstra's algorithm: $O(n^2 (k + \log n))$
  - Using Floyd-Warshall's algorithm: $O(n^3)$
    - Could be faster in practice with large k

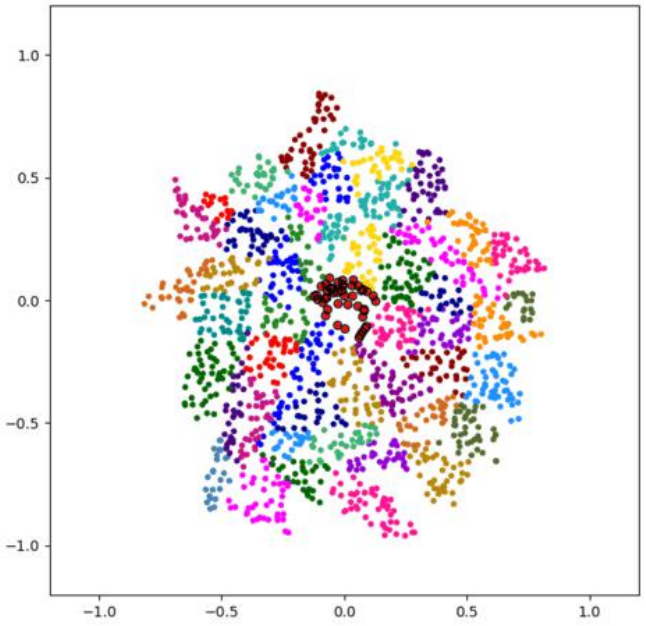
- For LLE, cost of steps are:
  - Finding nearest neighbors, $O(n^2 D)$ when $n << 2^D$
    - Can be significantly less with approximate nearest neighbors alg.
  - Computing local weights, $O(n (D k^2 + k^3))$
  - Reconstruction problem, $O(n^2)$

A new approach

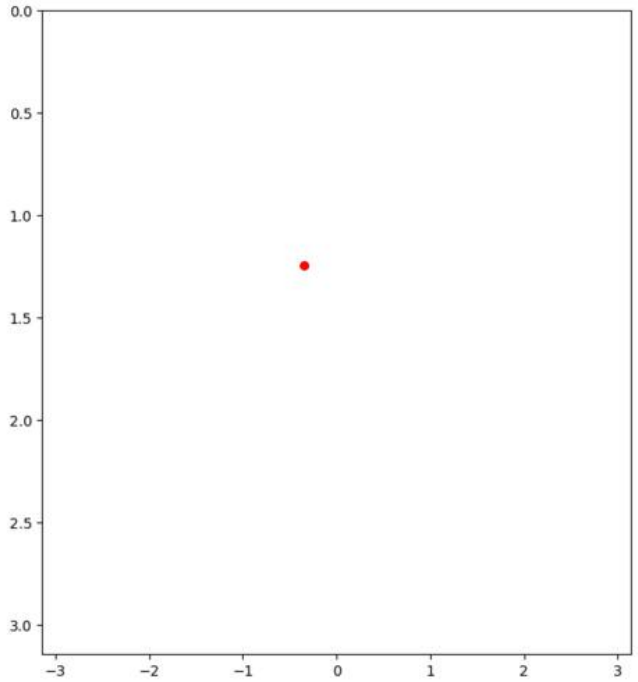
- Proposed algorithm (Jigsaw Isomap):
  - Use divide & conquer alg. to partition data into tiles
    - `O(n log n (k + log n))`
  - Embed each tile + its neighbors with PCA
    - `O(n D)`, when choosing # of points per tile to be small
  - For each tile, assemble neighboring tiles to form a view
    - `O(n m^3)` with avg `m` tiles in view, likely cheaper due to matrix sparsity
- Avoids graph search between all pairs of points
- Visualize curvature / topology by hopping between views

Demo: points on a sphere

Embedding view



Latitude & longitude of central tile



The viewport travels in a loop around the starting red tile. The overall rotation of the viewport at the end is an effect of the sphere's curvature

Sphere GIF: https://www.shorturl.at/tzLM4
Plane GIF: https://www.shorturl.at/hrDQ9
Hyperbolic plane GIF: https://www.shorturl.at/tCGKO

Final comparison table

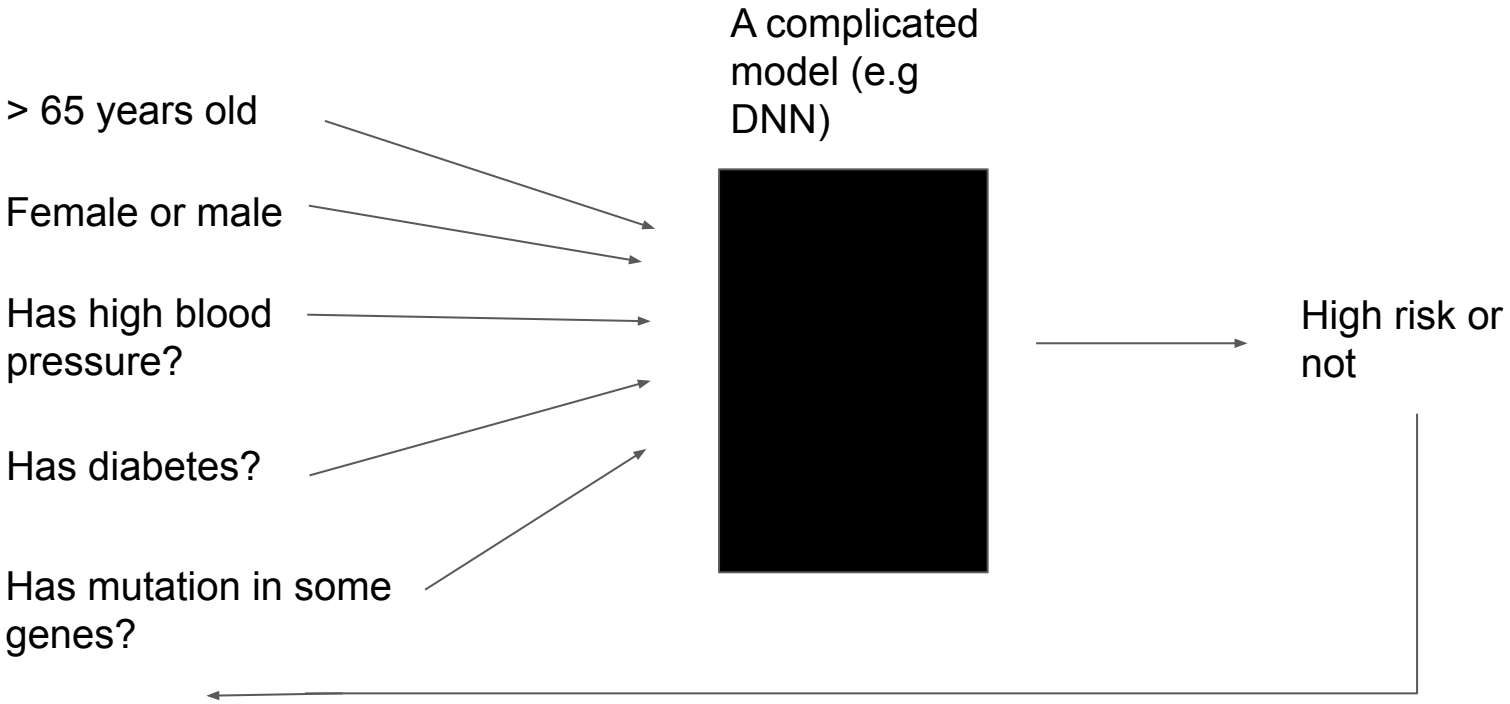
	Isomap	LLE	Jigsaw Isomap
Embedding quality	Consistent distance est. Optimal reconstruction	Fails with trivial cases	Consistent distance est.
Attempt to preserve geodesic distances	Yes, direct	No	Yes, indirect
Runtime complexity	Slow (super-quadratic)	Fast (kNN + quadratic)	Fast (kNN + sub-quadratic)
Curvature artifacts	Distortion near boundary	Distortion near boundary	Viewport rotation
Topology artifacts	Self-intersection in output	Self-intersection in output	Upper limit on viewport size

Thank you!

Understanding How Complex Models Make Predictions

Tianming Zhou (tianming), Dongshunyi Li (dongshul)

Motivation - who is most at risk for COVID-19?



...

How do we know which feature contributing most to high risk?

Related methods

- LIME (Ribeiro et al. 2016)
- DeepLIFT (Shrikumar et al, 2016; 2017)
- Layer-Wise Relevance Propagation
- SHAP (Lundberg & Lee, 2017)

SHAP as a unified framework

- Local methods: explain a prediction f on a single input x
- Each feature in x is either observed or missing

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \qquad z' \in \{0,1\}^M$$

Contribution of feature i

Whether feature i is observed

- The marginal contribution of feature i to f(x) depends on the states of other features, i.e., whether other features are observed/missing
- φ_i: The **average marginal contribution** of feature i after all combinations of other features' states are considered

Main results

- Proved the uniqueness and derived the analytical form of the Shapley value
  - In the context of feature explanation
  - Expand to other interpretation
- Implemented a SHAP algorithm adapted to multi-class tasks
- Experimentally validated the reasoning provided by Shapley value

Uniqueness & Analytical Form

- **Local accuracy**: the contributions of all features sum to f(x)
- **Symmetry**: the contribution of feature i is independent of its feature index, i
- **Strong monotonicity**: if the marginal contribution of feature i to f(x) is always greater than that to f'(x), then feature i contributes more to f(x) to f'(x)

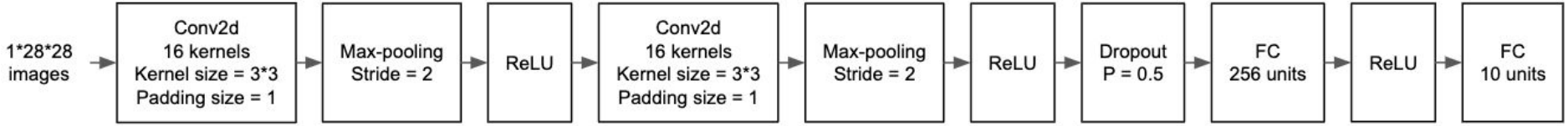
- The way to allocate contribution to features that satisfies the above 3 properties is **unique**

$$\forall i \in U,\ \phi_i^*(f,x) = \sum_{S: i \in S \subseteq U} \frac{(|S|-1)!(M-|S|)!}{M!} \left(f_x(S) - f_x(S \backslash \{i\})\right)$$

$$\phi_0^*(f,x) = f_x(\emptyset).$$

Setup for the prediction task

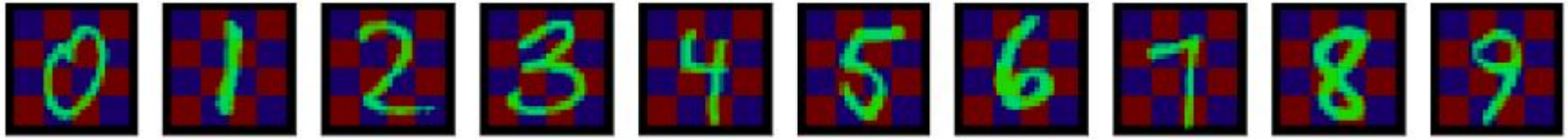
- Implemented a CNN model for MNIST prediction



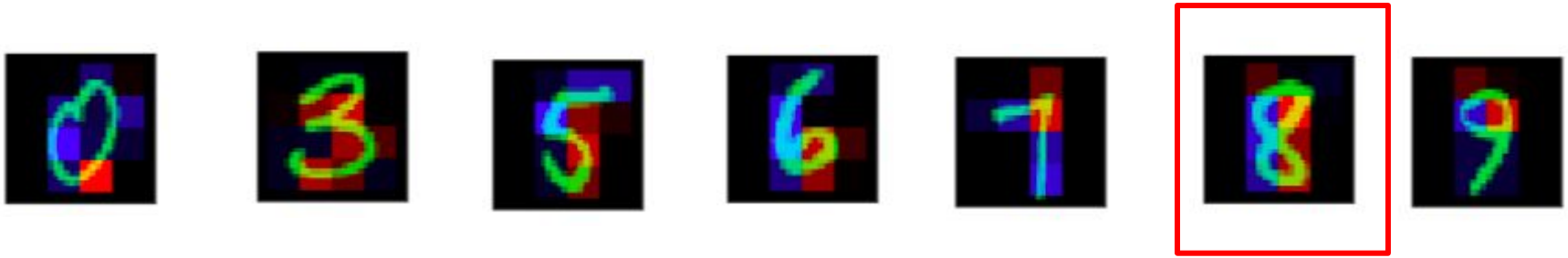
- One-versus-rest auROC for any digit (>= 99.99%)

Experiment results

- Generate super-pixels based on the original 28*28 pixels



- What super-pixels (marginally) contributing to being predict as **3**?



Acknowledgements

- Professor Pradeep Ravikumar
- TAs:
  - Ian Char
  - Kartik Gupta
  - Tom Yan
  - Sean Jin

Reference

- Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one, 10(7).
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).
- Shrikumar, A., Greenside, P., & Kundaje, A. (2017, August). Learning important features through propagating activation differences. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 3145-3153). JMLR. org.

- Shrikumar, A., Greenside, P., & Kundaje, A. (2017, August). Learning important features through propagating activation differences. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 3145-3153). JMLR. org.

- Young, H. P. (1985). Monotonic solutions of cooperative games. International Journal of Game Theory, 14(2), 65-72.