Investigating Latent Space and Conditional Generation of Variational Autoencoder

Koyoshi Shindo and Xueying Ding

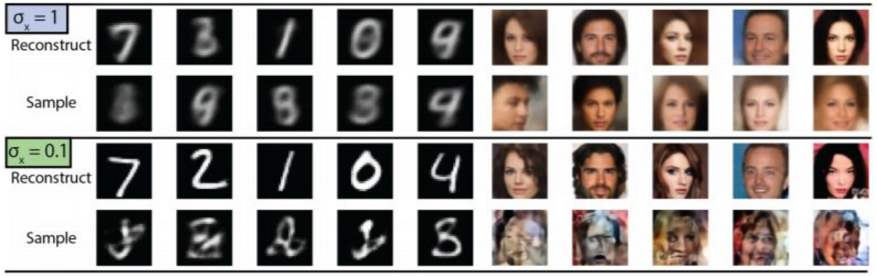
Why Conditional Generation

- Unconditional generation is arguably not very useful in real life: if one needs a random image, why not just sample it from the dataset?

- Such approach would less likely work for conditional generation, especially when more attributes are specified.

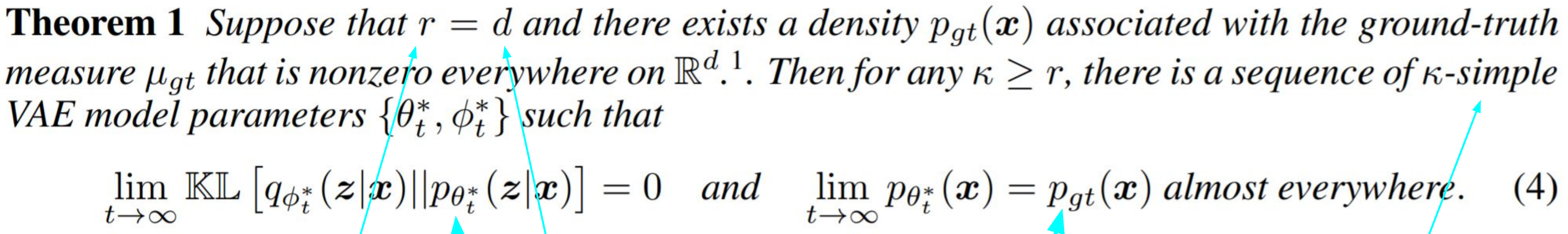
Challenges with Conditional Generation Using VAE

- ➢ Challenge 1: VAE suffers from the trade-off between high-quality reconstruction and realistic samples.
- ➢ Challenge 2. Conditional VAE needs to be retrained when new attributes are specified.
- ➢ We can **solve both challenges with an emphasis on the latent space** of VAE.



Engel et al '17, https://arxiv.org/abs/1711.05772

Challenge 1. Trade-off between Reconstruction and Samples



Theorem 1 *Suppose that* $r = d$ *and there exists a density* $p_{gt}(\boldsymbol{x})$ *associated with the ground-truth measure* μ_{gt} *that is nonzero everywhere on* $\mathbb{R}^d$.[1]. *Then for any* $\kappa \geq r$, *there is a sequence of* κ*-simple VAE model parameters* $\{\theta_t^*, \phi_t^*\}$ *such that*

$$\lim_{t\to\infty} \mathbb{KL}\left[q_{\phi_t^*}(\boldsymbol{z}|\boldsymbol{x})||p_{\theta_t^*}(\boldsymbol{z}|\boldsymbol{x})\right] = 0 \quad \textit{and} \quad \lim_{t\to\infty} p_{\theta_t^*}(\boldsymbol{x}) = p_{gt}(\boldsymbol{x}) \textit{ almost everywhere.} \qquad (4)$$

manifold dimension

ambient dimension

VAE with κ-dimension latent space

- ➢ If r = d, VAE can both **reconstruct well** and **recover manifold density**, no trade-off.
- ➢ If r < d, VAE can reconstruct well but may not recover manifold density, **source of trade-off**.

Challenge 1. Trade-off between Reconstruction and Samples

Theorem 2

$$2\mathbb{KL}\left[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\right] \equiv \text{trace}\left[\boldsymbol{\Sigma}_z\right] + \|\boldsymbol{\mu}_z\|_2^2 - \log|\boldsymbol{\Sigma}_z| \approx -\hat{r}\log\gamma + O(1) \quad \text{in the } \gamma \to 0 \text{ regime}$$

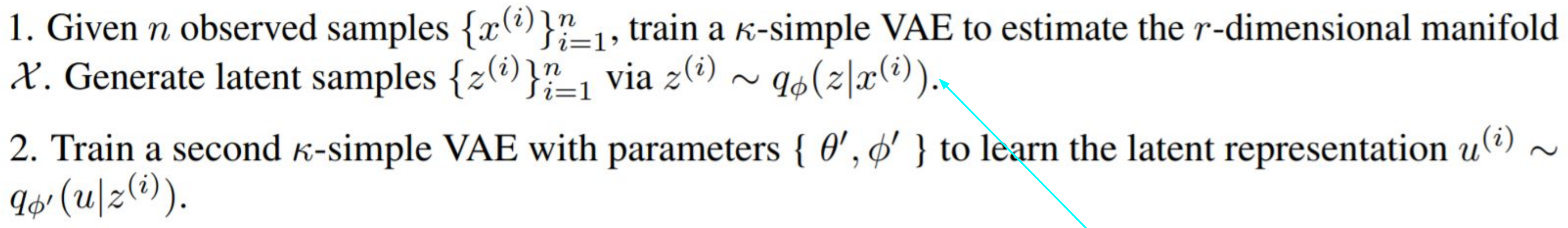
Hyperparameter of VAE

part of ELBO encoder variance encoder mean number of low-noise latent dimension

- ➢ Smallest possible $\hat{r}$ is r.
- ➢ If $\kappa > \mathrm{r}$, optimal VAE will learn r low-noise latent dimensions and fill in random Gaussian noise for the remaining (κ - r) dimensions to minimize $\mathbb{KL}\left[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\right]$
- ➢ This means the **latent space of VAE $q_\phi(\boldsymbol{z})$ does not lie on any lower dimensional manifolds**.

Dai et al '19, https://arxiv.org/abs/1903.05789

Challenge 1. Trade-off between Reconstruction and Samples

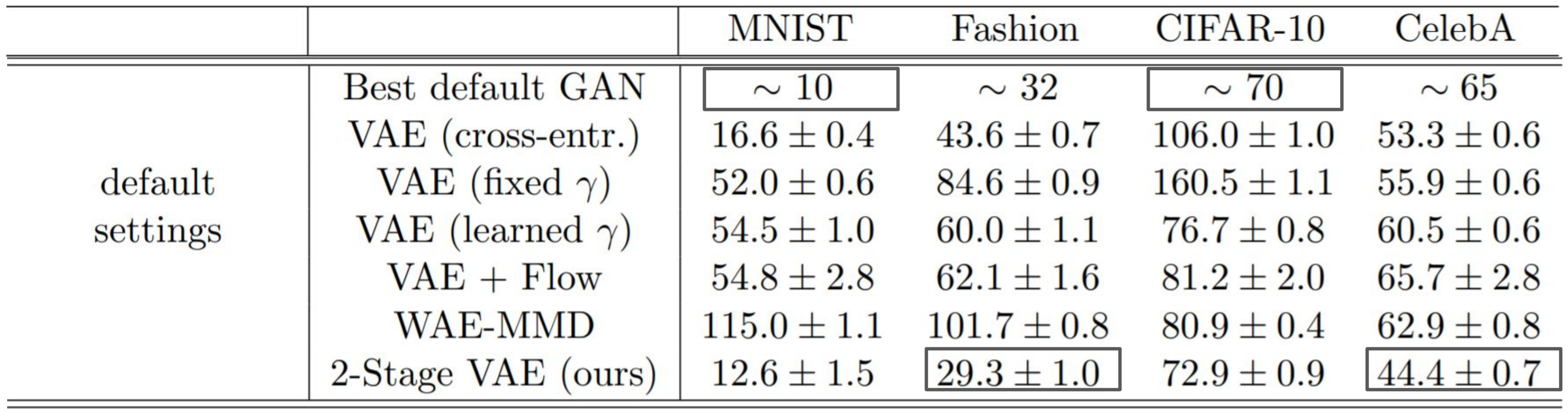

1. Given n observed samples $\{x^{(i)}\}_{i=1}^{n}$, train a κ-simple VAE to estimate the r-dimensional manifold $\mathcal{X}$. Generate latent samples $\{z^{(i)}\}_{i=1}^{n}$ via $z^{(i)} \sim q_\phi(z|x^{(i)})$.

2. Train a second κ-simple VAE with parameters $\{\ \theta', \phi'\ \}$ to learn the latent representation $u^{(i)} \sim q_{\phi'}(u|z^{(i)})$.

manifold dim = ambient dim

➢ Referred to as "2-Stage VAE".

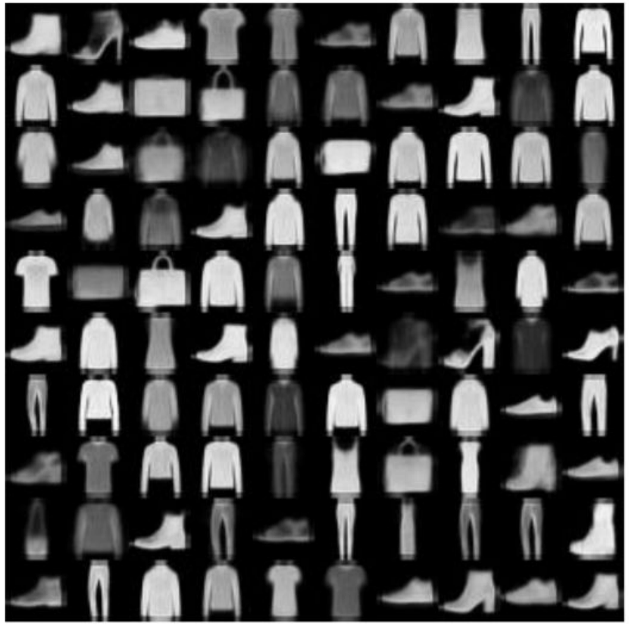
Challenge 1. Trade-off between Reconstruction and Samples

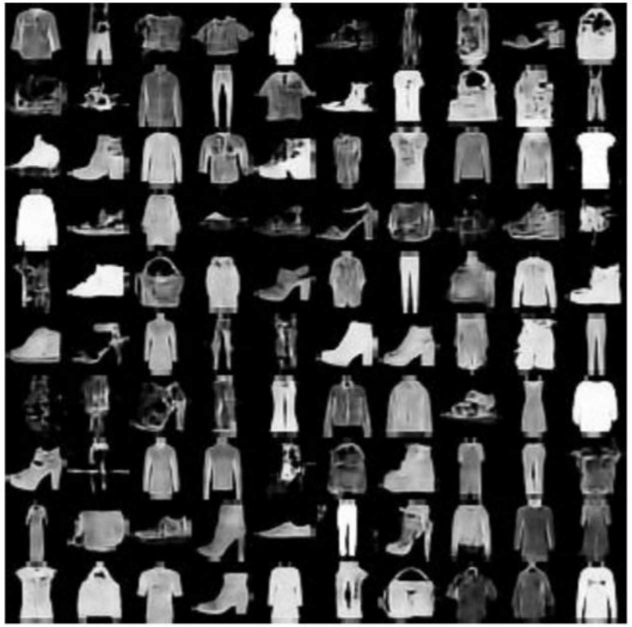


		MNIST	Fashion	CIFAR-10	CelebA
default settings	Best default GAN	$\boxed{\sim 10}$	~ 32	$\boxed{\sim 70}$	~ 65
	VAE (cross-entr.)	16.6 ± 0.4	43.6 ± 0.7	106.0 ± 1.0	53.3 ± 0.6
	VAE (fixed γ)	52.0 ± 0.6	84.6 ± 0.9	160.5 ± 1.1	55.9 ± 0.6
	VAE (learned γ)	54.5 ± 1.0	60.0 ± 1.1	76.7 ± 0.8	60.5 ± 0.6
	VAE + Flow	54.8 ± 2.8	62.1 ± 1.6	81.2 ± 2.0	65.7 ± 2.8
	WAE-MMD	115.0 ± 1.1	101.7 ± 0.8	80.9 ± 0.4	62.9 ± 0.8
	2-Stage VAE (ours)	12.6 ± 1.5	$\boxed{29.3 \pm 1.0}$	72.9 ± 0.9	$\boxed{44.4 \pm 0.7}$

Dai et al ‘19, https://arxiv.org/abs/1903.05789

Challenge 1. Trade-off between Reconstruction and Samples


VAE


2-Stage VAE

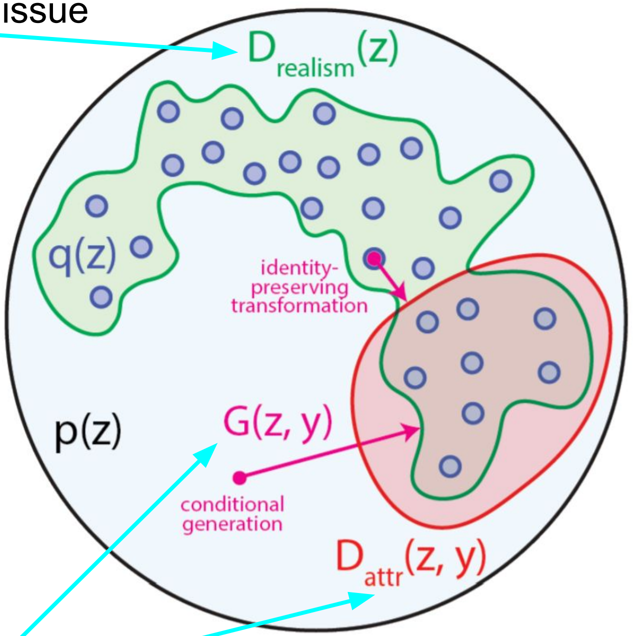
Dai et al ‘19, https://arxiv.org/abs/1903.05789

Challenge 2. Expensive Retraining of CVAE

➢ Idea: training generative models on images is expensive, **but training generative models on latent vectors is not as expensive**.

Challenge 2. Expensive Retraining of CVAE

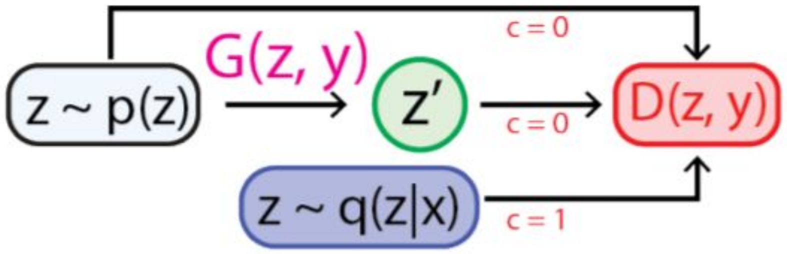
Not necessary, 2-Stage VAE solves this issue



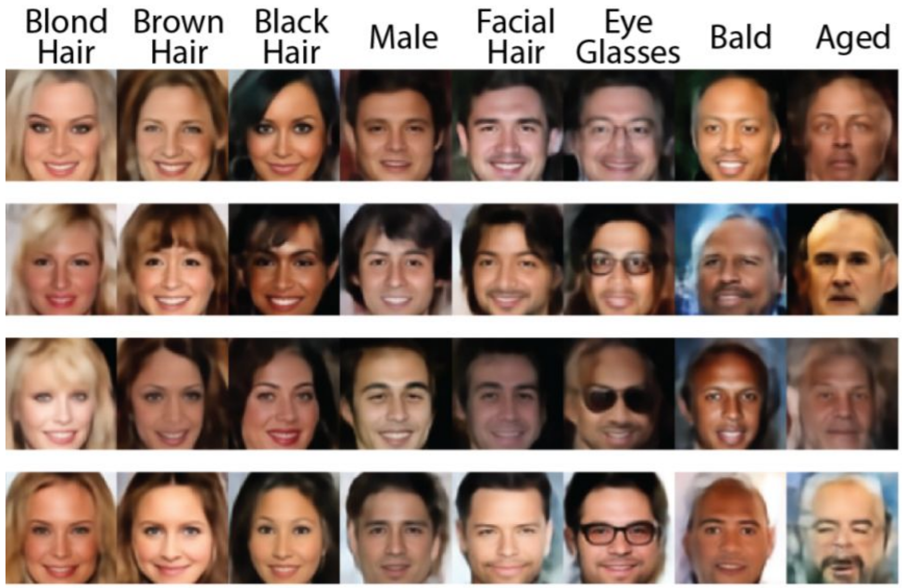
CGAN

- Train a CGAN on latent vectors z separately.
- **Use 96✕ ~ 2884✕ fewer FLOPs/iteration** with equally good performance.

Actor/Critic Training

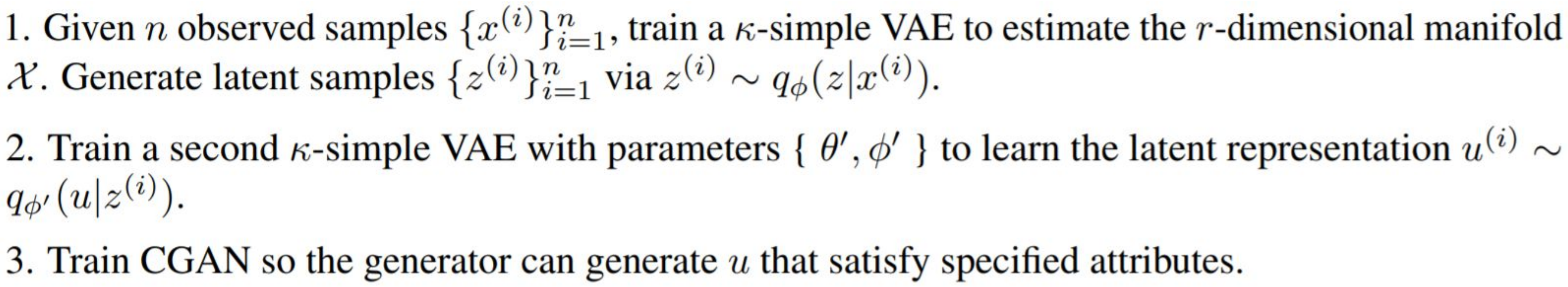


Challenge 2. Expensive Retraining of CVAE



Engel et al ‘17, https://arxiv.org/abs/1711.05772

Final Model Architecture


1. Given n observed samples $\{x^{(i)}\}_{i=1}^{n}$, train a κ-simple VAE to estimate the r-dimensional manifold $\mathcal{X}$. Generate latent samples $\{z^{(i)}\}_{i=1}^{n}$ via $z^{(i)} \sim q_\phi(z|x^{(i)})$.

2. Train a second κ-simple VAE with parameters $\{\ \theta', \phi'\ \}$ to learn the latent representation $u^{(i)} \sim q_{\phi'}(u|z^{(i)})$.

3. Train CGAN so the generator can generate u that satisfy specified attributes.

- ➢ Challenge 1: Trade-off between reconstruction and samples
- ➢ Challenge 2: Expensive retraining of CVAE

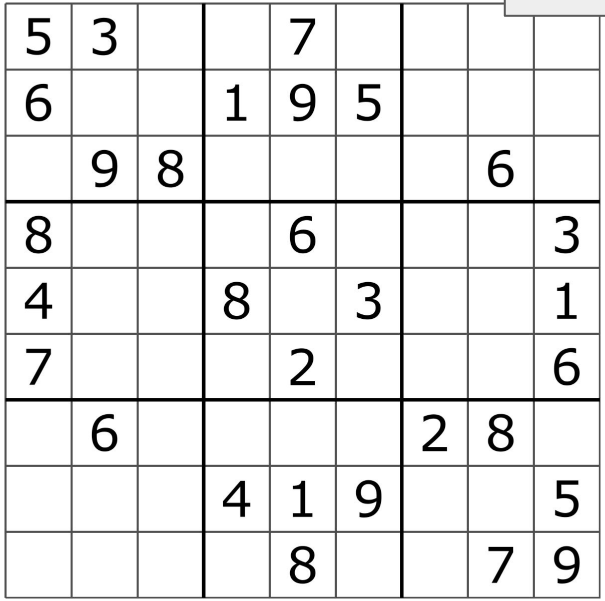
Thank you

Differentiable SAT Solver

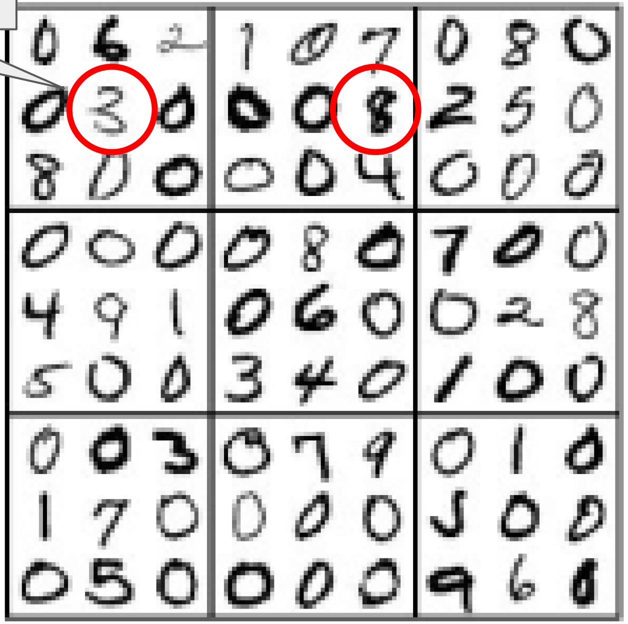
Haoping Bai and Jianing Yang

Motivation: Deep Learning + Logical Inference

Solved by Deep Learning classifiers



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



Traditional SAT problem

Solved by SAT algorithms

Differentiable SAT problem

Problem Definition

- SAT:

$$v_1 \wedge (v_3 \vee \neg v_1) \wedge (v_1 \vee \neg v_2)$$

Configuration of variables: $S = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}$

- Max SAT:

$$(x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1) \wedge (\neg x_0 \vee \neg x_1)$$

The above may NOT be satisfiable
-> Find the configuration of variables such that max # of clauses turn to be True.

Differentiable Continuous Relaxation

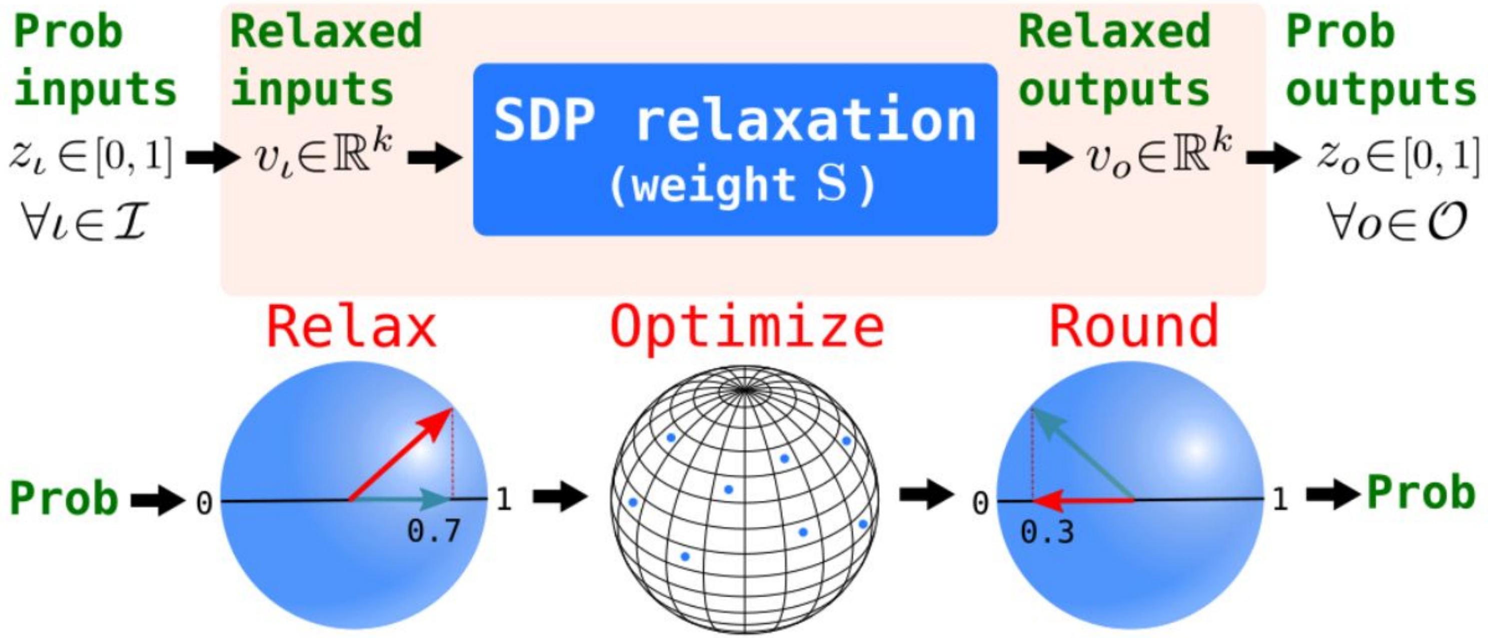
- Relax the binary variables to smooth and continuous parameterization

$$v_i \in \{-1, +1\} \xrightarrow{\text{relax}} ||v_i|| = 1,\ v_i \in \mathbb{R}^k$$

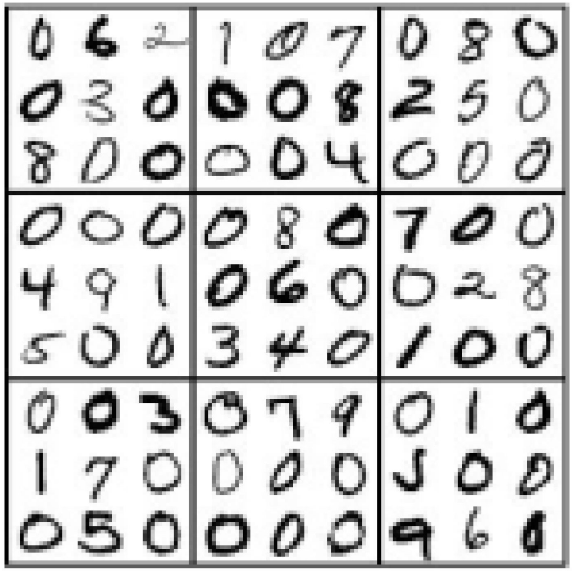
- Semidefinite Programing (SDP) Relaxation (Goemans-Williamson, 1995):

$$\begin{aligned} \min_{V \in \mathbb{R}^{k \times (n+1)}} \quad & \operatorname{tr}(V^\top V S^\top S) \\ \text{s.t.} \quad & V^\top V \succeq 0, \\ & \operatorname{diag}(V^\top V) = 1 \end{aligned} \qquad V = [v_1\, v_2\, \cdots v_n] \in \mathbb{R}^{k \times (n+1)}$$

Satisfiability Solving as a Layer



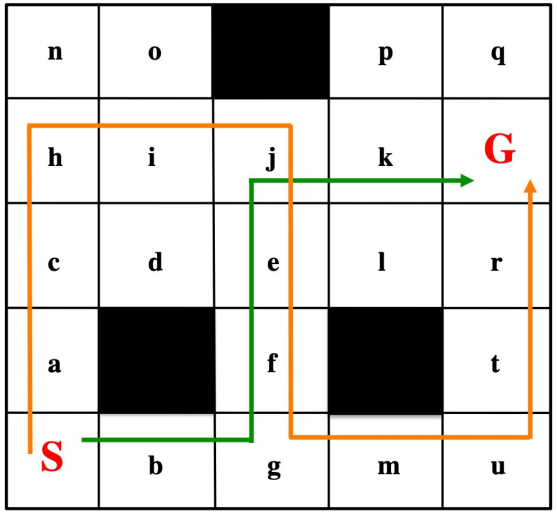
Applications & Experiments


MNIST Sudoku (original paper)

$\left\{ \begin{array}{l} \text{CNN: 0\%} \\ \\ \\ \text{SATNet: 63.2\%} \end{array} \right.$

Applications & Experiments

Game Map Generation
(constraints on edges)



Constrained Shortest Path

References

- SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver (https://arxiv.org/pdf/1905.12149.pdf)

Information Bottleneck

A modern theory of generalization for deep learning

Cinnie Hsiung Allen Zhu
10-716 Project Presentation
Carnegie Mellon University

How do deep nets generalize?

Overparameterized networks can fit random labels

- 100% train accuracy
- 0% test accuracy

Big Idea

Complexity ~ **information content** in the trained weights.

The IB Lagrangian

Cross entropy loss decomposes as:

$$H_{p,q}(y|x,w) = \underbrace{H(y|x,\theta)}_{\text{intrinsic error}} + \underbrace{I(\theta;y|x,w)}_{\text{sufficiency}} + \mathbb{E}_{x,w}\underbrace{(p(y|x,w)||q(y|x,w))}_{\text{accuracy}} - \underbrace{I(y;w|x,\theta)}_{\text{overfitting}}.$$

Optimal regularization:

$$L(q(w|D)) = H_{p,q}(y|x,w) + I(y;w|x,\theta)$$

The IB Lagrangian

Loss:

$$L(q(w|D)) = H_{p,q}(y|x,w) + I(y;w|x,\theta)$$

- Intractable in practice
- So constrain an approximation

Generalized IB Lagrangian:

$$L(q(w|\mathcal{D})) = H_{p,q}(y|x,w) + \beta \tilde{I}(w;\mathcal{D})$$

From overfitting to underfitting

$$L(q(w|\mathcal{D})) = H_{p,q}(y|x,w) + \beta \tilde{I}(w;\mathcal{D})$$

With a good approximation,

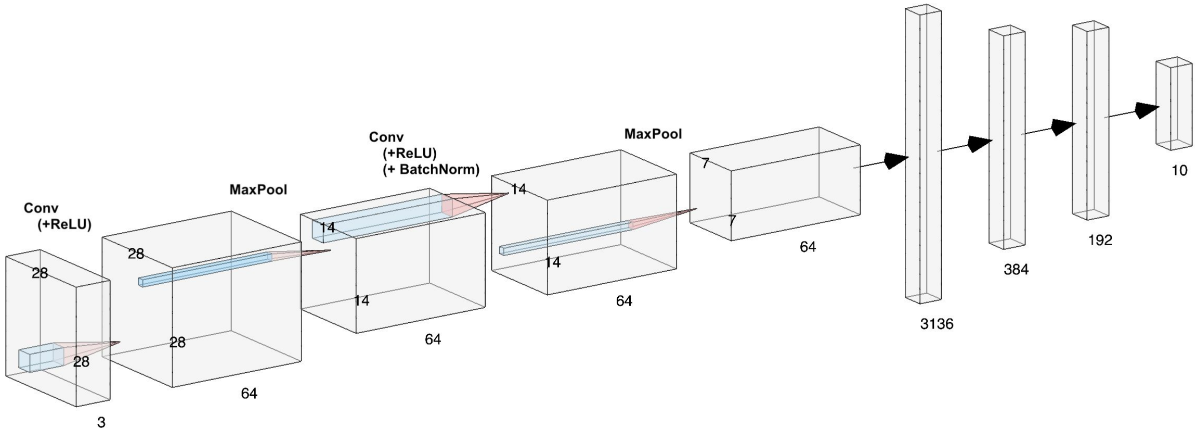
- For $\beta < 1$, we overfit
- For $\beta > 1$, we underfit

Hence, $\beta = 1$, is the critical value where we observe a phase transition between overfitting and underfitting

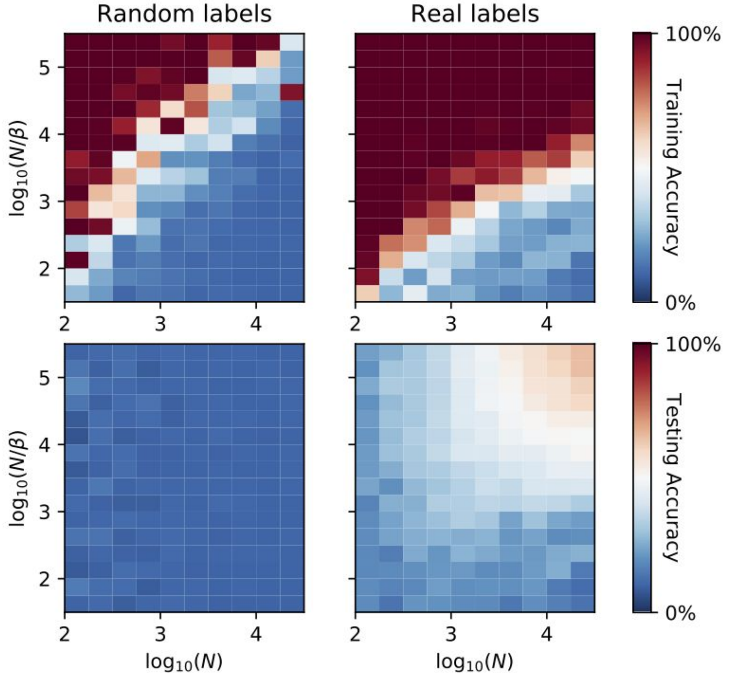
Experiments

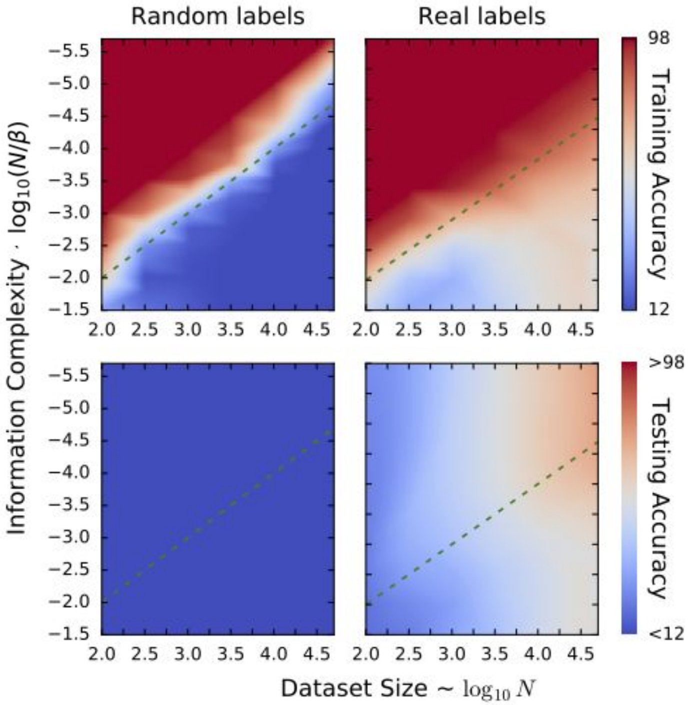
Predict the **transition from overfitting to underfitting**

- Train modified AlexNet on CIFAR-10
- Vary number of samples and regularization



Experimental Results





Thank you!

Bayesian Discovery of Pairwise Interactions in High Dimensions

Wuwei Lin

April 28, 2020

Introduction

Learn linear regression with augmented feature space

$$\Phi(x) = [1, x_1, .., x_d, x_1 x_2, f .., x_{d-1} x_d, x_1^2, .., x_d^2]^T, x \in \mathbb{R}^d$$
$$y = \theta^T \Phi(x) + \epsilon$$

Bayesian learning to find θ that satisfies:

- Sparsity: $||\theta||_0 \leq s$
- Strong hierarchy prior: $\theta_{x_i x_j} \neq 0$ iff $\theta_{x_i} \neq 0, \theta_{x_j} \neq 0$

Bayesian Learning

Our priors can be modeled as:

$$\begin{aligned}\tau &\sim p(\tau)\\ \sigma^2 &\sim p(\sigma^2)\\ \theta|\tau &\sim \mathcal{N}(0, \Sigma_\tau)\\ y^{(n)}|x^{(n)}, \theta, \sigma^2 &\sim \mathcal{N}(\theta^T \Phi(x^{(n)}), \sigma^2)\end{aligned}$$

Estimating or sampling posterior $\mathbb{E}_{p(\theta|D)}[f(\theta)]$ directly is difficult
Additionally sample $p(\tau|D)$ with MCMC, which requires computing $p(D|\tau, \sigma^2)$: computing posterior Gaussian requires $O(d^2N^2 + N^3)$ time

Re-parameterize with Gaussian Process

Our Bayesian model follows the weight-space view of Gaussian Process

$$\begin{aligned}
\tau &\sim p(\tau) \\
g|\tau &\sim GP(0, k_\tau), k_\tau(x, x') = \Phi(x)^T \Sigma_\tau \Phi(x') \\
\sigma^2 &\sim p(\sigma^2) \\
y^{(n)}|x^{(n)}, g, \sigma^2 &\sim \mathcal{N}(g(x^{(n)}), \sigma^2)
\end{aligned}$$

Theorem 1: $g(\cdot) = \theta^T \Phi(\cdot)$ in distribution
Theorem 2: For any diagonal Σ_τ, k_τ can be written as a weighted sum of polynomial kernels of the form $k_{poly}(x, x') = (x^T x' + c)^d$, which is $O(d)$ in time

Kernel Interaction Trick

Now we have efficient sampling of $p(\tau|D)$, we want to compute $\mathbb{E}_{p(\theta|D,\tau)}[f(\theta)]$
To learn θ_{x_i}, choose i-th unit vector $e_i = [0, 0, ..., 1, 0, 0]^T$

$$g(e_i) = \theta_{x_i} + \theta_{x_i^2}$$
$$g(-e_i) = -\theta_{x_i} + \theta_{x_i^2}$$

We can recover θ_{x_i} as:

$$\theta_{x_i} = \frac{1}{2}(g(e_i) - g(-e_i))$$

We can find the top k main effects in this way, and then compute their pairwise interactions. Total number of possible iterations is $\Theta(k^2)$, smaller than $\Theta(d^2)$.

Overparameterization in Deep Generative Models

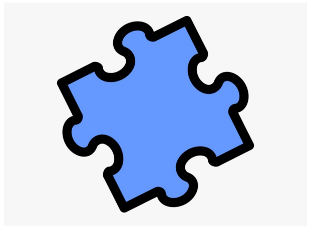
Saurabh Garg & Tanya Marwah

CARNEGIE MELLON UNIVERSITY
PITTSBURGH PENNSYLVANIA 1900

April 28th 2020

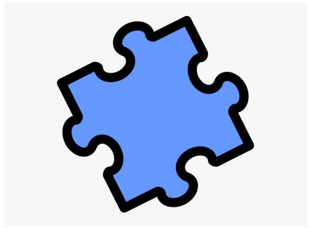
Introduction

- Zhang et. al. [2017] pointed out the paradox with overparameterization in deep neural networks
  - Reasonably large neural network can fit random labels
  - **Generalization puzzle:** Even though they achieve good generalization performance



Introduction

- Zhang et. al. [2017] pointed out the paradox with overparameterization in deep neural networks
  - Reasonably large neural network can fit random labels
  - **Generalization puzzle:** Even though they achieve good generalization performance

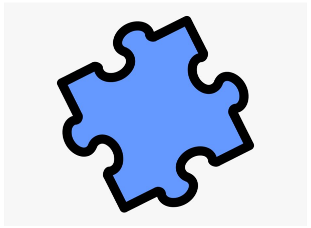


- Since then, many attempts to resolve this:
  - Neural Tangent Kernel [Jacot et. al. 2018]

$$\partial_t f(t) = -\nabla_K C|_{f(t)}.$$

Introduction

- Zhang et. al. [2017] pointed out the paradox with overparameterization in deep neural networks
  - Reasonably large neural network can fit random labels
  - **Generalization puzzle:** Even though they achieve good generalization performance



- Since then, many attempts to resolve this:
  - Neural Tangent Kernel [Jacot et. al. 2018]

$$\partial_t f(t) = -\nabla_K C|_{f(t)}.$$

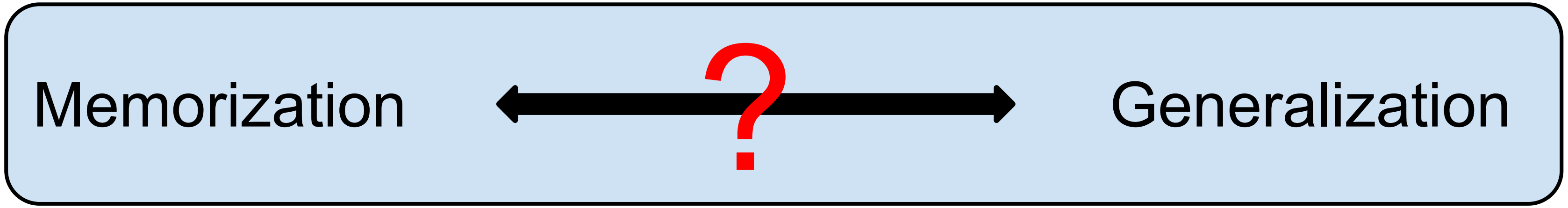
  - Convergence to global minima for 2-layered network [Du et. al. 2018]

$$\|\mathbf{u}(t) - \mathbf{y}\|_2^2 \leq \exp(-\lambda_0 t) \|\mathbf{u}(0) - \mathbf{y}\|_2^2 .$$

  - and many more ...

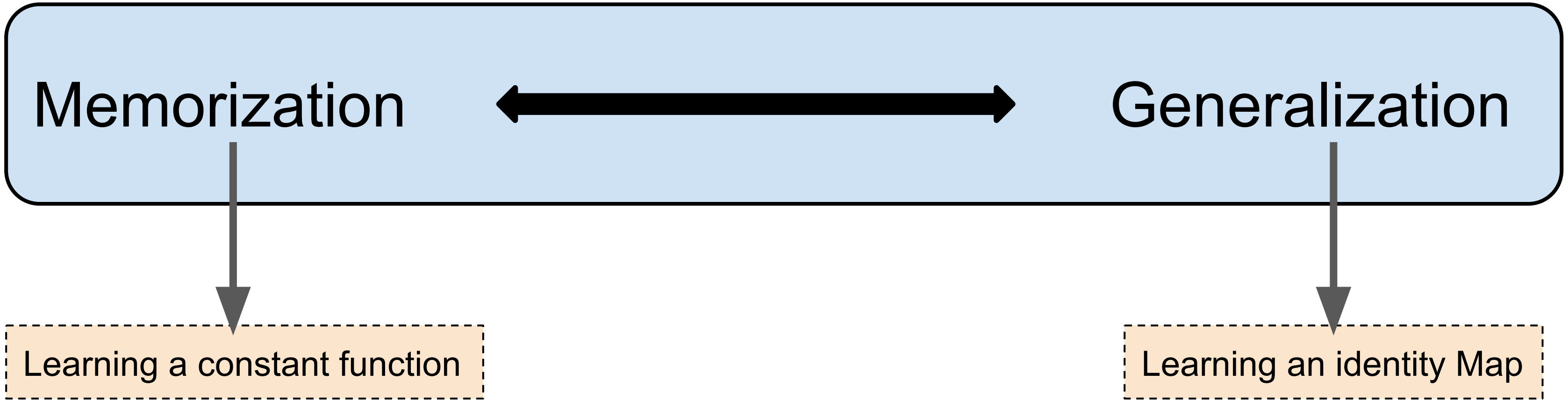
Motivation: Generative Models

- How does the generalization puzzle transfer in unsupervised learning of generative models ?



Motivation: Generative Models

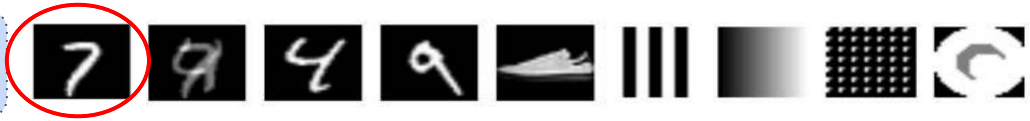
- How does the generalization puzzle transfer in unsupervised learning of generative models ?



Generalization Puzzle: Example*

Training Samples

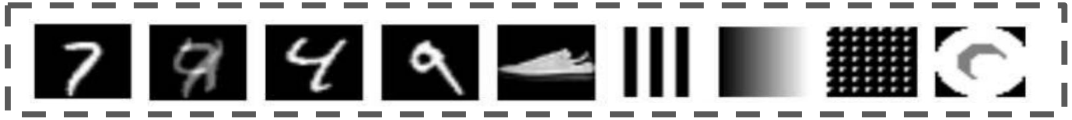
(Just images of 7)



* Zhang et. al. Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698* (2019).

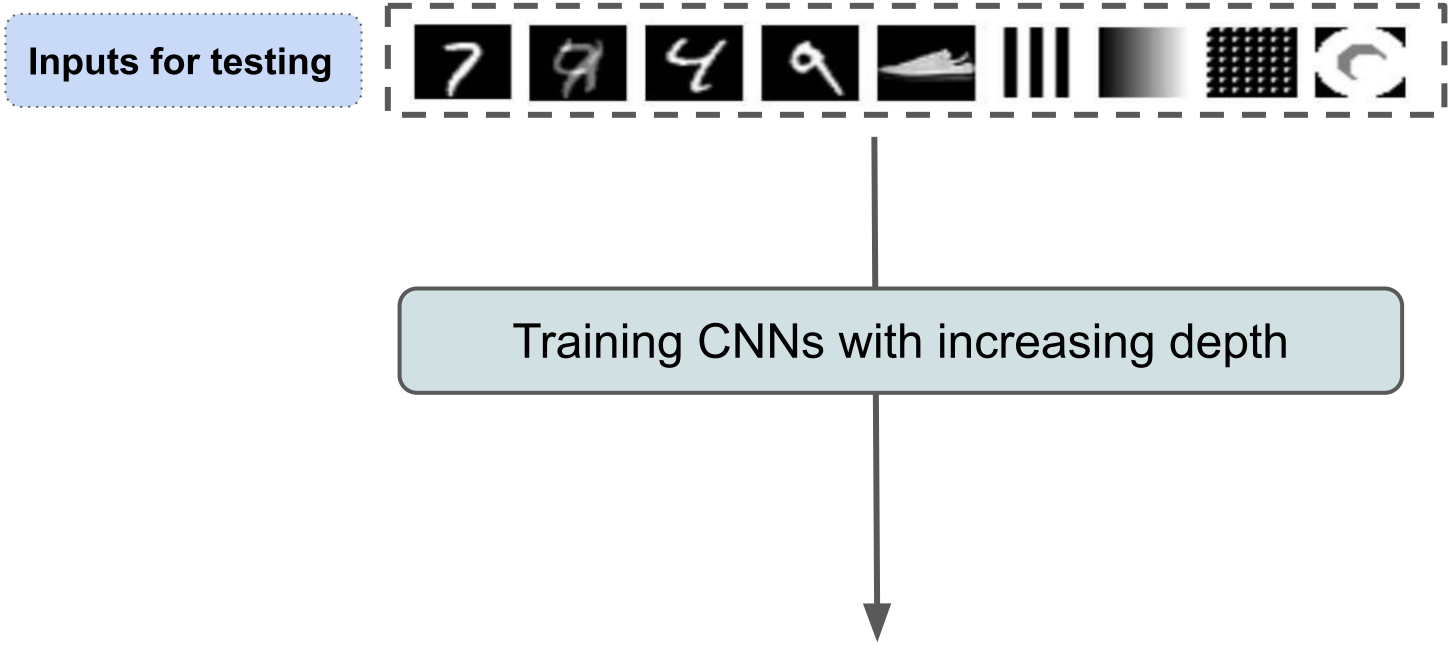
Generalization Puzzle: Example*

Inputs for testing



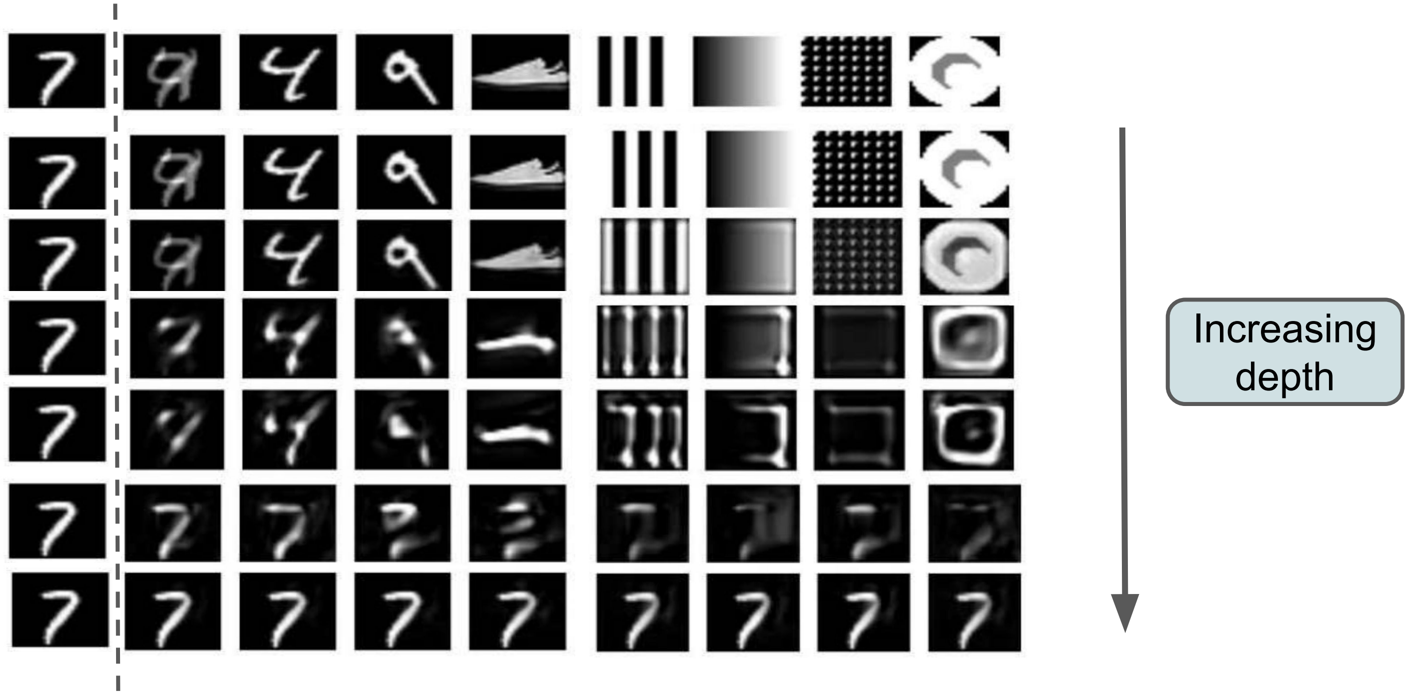
* Zhang et. al. Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698* (2019).

Generalization Puzzle: Example*



* Zhang et. al. Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698* (2019).

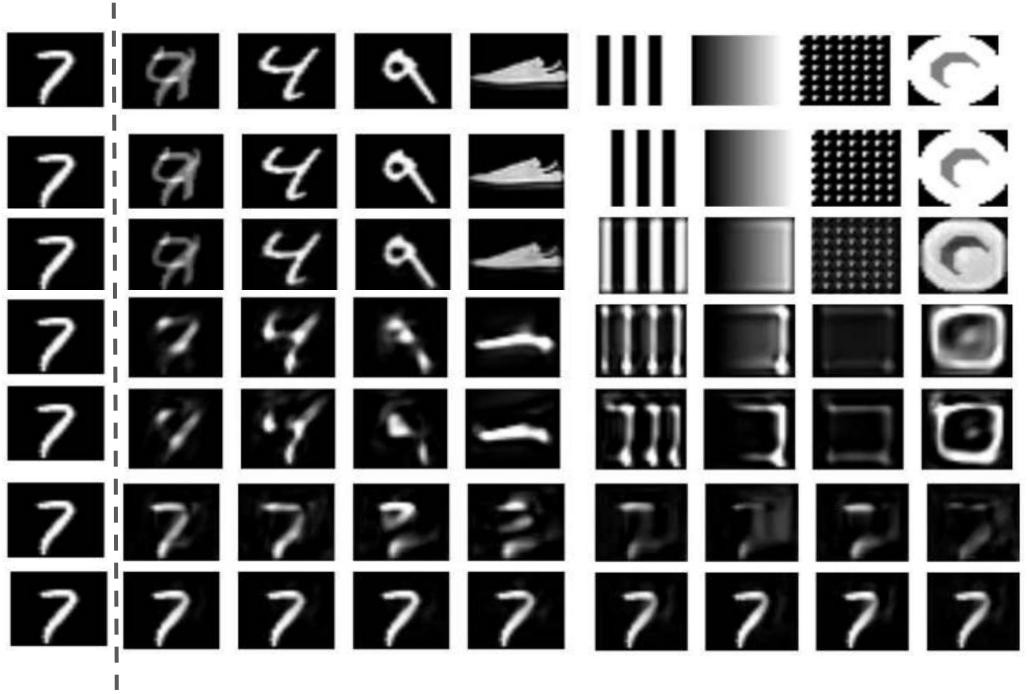
Generalization Puzzle: Example*



* Zhang et. al. Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698* (2019).

Generalization Puzzle: Example*

Note as we increase depth it fails to learn an **identity map**, instead it learns to **memorize**



Increasing depth

* Zhang et. al. Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698* (2019).

Benefits of Jointly Training Autoencoders: An Improved Neural Tangent Kernel Analysis*

*Nguyen et. al. Benefits of Jointly Training Autoencoders: An Improved Neural Tangent Kernel Analysis. *arXiv preprint arXiv:1911.11983* (2019).

Key Takeaways

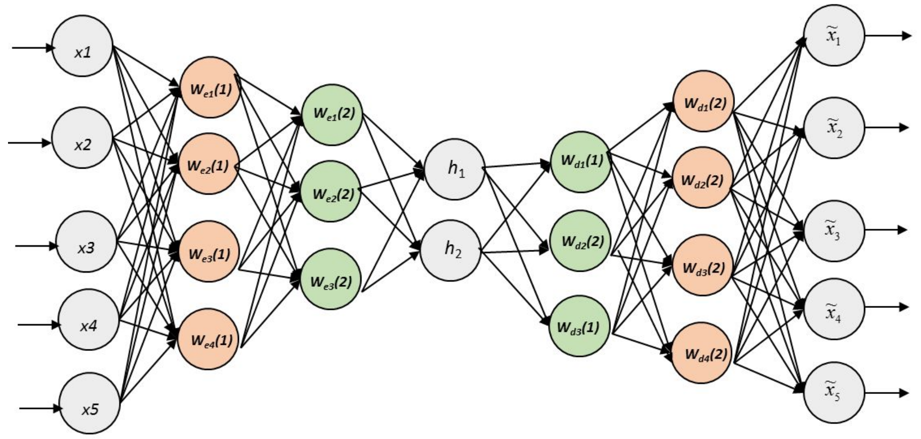
Under a linearization assumption, an Autoencoder’s reconstruction for a given input can be written down as a linear combination of the training samples weighted by kernel scores.

Key Takeaways

Under a linearization assumption, an Autoencoder’s reconstruction for a given input can be written down as a linear combination of the training samples weighted by kernel scores.

Study the gradient dynamics of a two layer autoencoder and obtain a bound on the number of hidden neurons (i.e., level of over-parameterization) required to achieve linear convergence of gradient descent, starting from random initialization, to global optimality.

Problem Setup

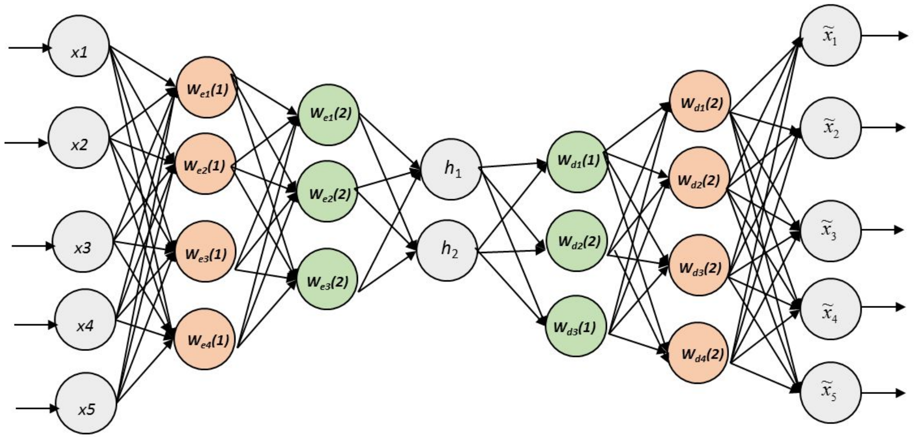


Two Layer Autoencoder defined as,

$$u = \frac{1}{\sqrt{md}} A\phi(W^T x)$$

Where $A = [a_1, \ldots, a_m]$ and $W = [w_1, \ldots, w_m]$ and $w_i \sim N(0, I)$ and $a_i \sim \text{Unif}\{-1,1\}$

Problem Setup



Two Layer Autoencoder defined as,

$$u = \frac{1}{\sqrt{md}} A\phi(W^T x)$$

Where $A = [a_1, \ldots, a_m]$ and $W = [w_1, \ldots, w_m]$ and $w_i \sim N(0, I)$ and $a_i \sim \text{Unif}\{-1,1\}$

Loss Function: MSE

$$\mathcal{L}(W, A) = \frac{1}{2} \sum_{i=1}^{n} \|x_i - \frac{1}{\sqrt{md}} A\phi(W^T x_i)\|_2$$

Main Result 1: Inductive Bias

Under a linearization assumption, an autoencoder when trained with multiple samples the closer the new test input x is to the span of the training data X, the more its **reconstruction concentrates** around the seen points.

Theorem. Let K be the tangent kernel. Assume $\lambda_{\min}(K) \geq 0$ and let $\eta_{\text{critical}} = 2(\lambda_{\max}(K) + \lambda_{\min}(K))^{-1}$. Under gradient descent with learning rate $\eta \leq \eta_{\text{critical}}$, for every normalized $x \in \mathbb{R}^d$ as the width $m \to \infty$, the autoencoder output $f_t(x)$ at step t converges to $\mu_t(x) + \gamma_t(x)$, where:

$$\mu_t(x) \to \sum_{i=1}^{n} \Lambda_i x_i$$

$$\gamma_t(x) \to f_0(x) - \sum_{i=1}^{n} \Lambda_i f_0(x_i)$$

Main Result 2: Parameter Bounds

Weak Training: Only train the encoder, # parameters should be greater than

$$O_p(n^5)$$

Theorem. Consider an autoencoder that computes output $u = \frac{1}{\sqrt{md}} A\phi(W^T x)$ where the weight vectors are initialized with independent vectors $w_r \sim \mathcal{N}(0, I)$ and $a_r \sim \text{Unif}(\{-1, 1\}^d)$ for all $r \in [m]$. For any $\delta \in (0, 1)$ and $m \geq C \frac{n^5 d^4 \lambda_{\max}}{\lambda_{\min}^4 \delta^3}$ for some large enought constant C, the gradient descent over W linearly converges to a global minimizer with probability at least $1 - \delta$ over the randomness in the initialization.

Main Result 2: Parameter Bounds

Joint Training: Only train the encoder, # parameters should be greater than

$$O_p(n)$$

Theorem. Consider an autoencoder that computes output $u = \frac{1}{\sqrt{md}} A\phi(W^T x)$ where the weight vectors are initialized with independent vectors $w_r \sim \mathcal{N}(0, I)$ and $a_r \sim \text{Unif}(\{-1, 1\}^d)$ for all $r \in [m]$. For any $\delta \in (0, 1)$ and $m \geq C \frac{nd\lambda_{\max}^3}{\lambda_{\min}^4 \delta^2}$ for some large enought constant C, the gradient descent over W and A linearly converges to a global minimizer with probability at least $1 - \delta$ over the randomness in the initialization.

Main Result 2: Parameter Bounds

Joint Training: Only train the encoder, # parameters should be greater than

$$O_p(n)$$

Theo ectors
are in r any
$\delta \in (0$ early
conve on.

A significant Improvement from the Weak Training case.

Bayesian Hyperparameter Optimization

Ruochi Zhang, Hongyu Zheng

Computational Biology Department | Carnegie Mellon University School of Computer Science

Background / Setup

- Hyperparameter tuning is usually treated as “black art” and human labor intensive.
- Given an algorithm with tunable hyperparameters, with a fixed budget (ex. # of attempts), find the best hyperparameter for running the algorithm

- Prior methods (evolutionary algorithms etc) uses **local / near-term** information.
- By modeling hyperparameter tuning as an exploration in a *nonparametric* Bayesian setup, we can utilize **global / full** information from past attempts.

Approach / Analysis

- Let f be the algorithm we evaluate, and x be a hyperparameter.
- Gaussian process: $f(x) \sim GP(0, k(x, x'))$ induces posterior $f(x \mid D) \sim N(\mu(x), \sigma^2(x))$
- Determine next attempt: maximize **acquisition function**
  - Expected Improvement: $a(x, D) = \mathbb{E}_{f \sim N(\mu,\sigma)} \max(0, f - f^*)$
  - Confidence Upper Bound: $a(x, D) = \mu + \kappa\sigma$ for (2-sided) confidence $2\Phi(\kappa) - 1$

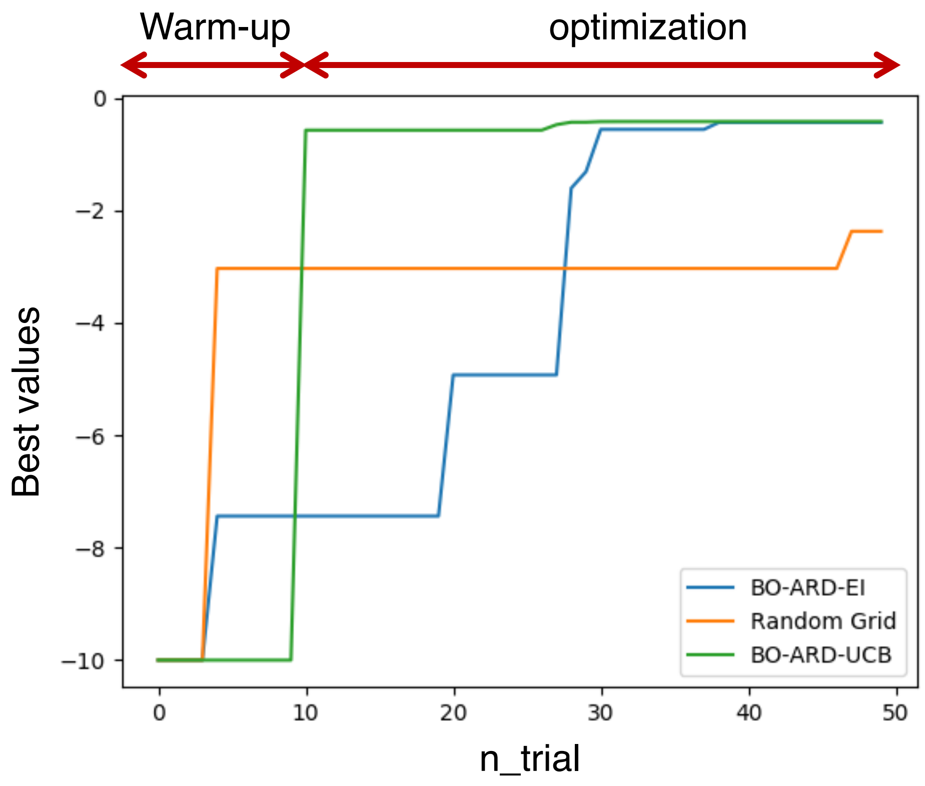
- Theoretical analysis based on **regret minimization**: instantaneous regret approaches 0 = obtains optimal solution asymptotically

Experiment 1: Branin function

- A highly nonconvex function

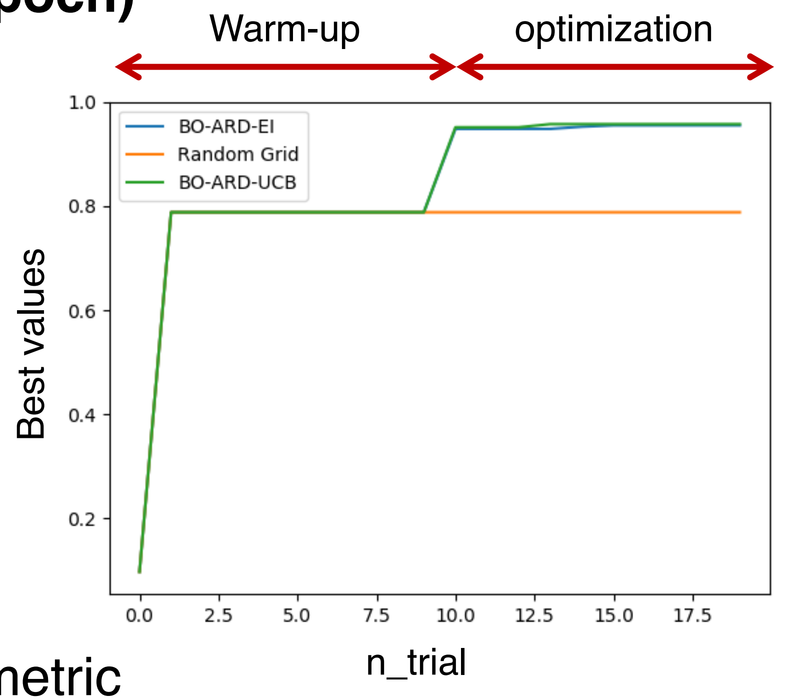
$$f(x,y) = \left(y - \frac{5.1}{4\pi^2}x^2 + \frac{5}{\pi}x - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x) + 10 + \sigma_e$$

- $\text{argmax}_{x,y}\ -f(x,y)$
- $x \in [-5,10], y \in [0,15], \sigma_e \sim N(0, 0.1^2)$
- All optimization methods can evaluate on the function for 50 times
- The two Bayesian method consistently outperforms the random grid search



Experiment 2: Neural network classification on MNIST

- Fit an MLP model to make classification on MNIST **(1 epoch)**
- Parameters to optimize:
  - Number of hidden layers (0 - 4)
  - Number of neurons in the hidden layer (64 - 256)
  - Dropout rate (0.5 - 1.0)
  - Learning rate (0.001 – 0.1)
- The accuracy on the test set is used as the evaluation metric
- All optimization methods can evaluate on the function for 20 times
- The first 10 trial (warm-up) are the same



Summary

- We implemented two types of Bayesian optimization method (**Expected Improvement** and **Confidence Upper Bound**)
- We tested these two methods on two tasks: optimizing a highly non-convex function and tuning hyperparameters for a neural network
- We observed that the Bayesian optimization method consistently outperforms the baseline model

Thank you!

Computational Biology Department

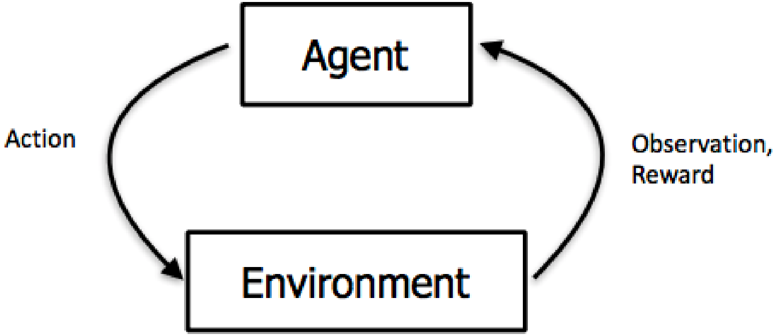
Carnegie Mellon University School of Computer Science

Efficiency of Q-learning for solving Markov Decision Processes

Dhruv Malik & Vishwak Srinivasan

What is Reinforcement learning (RL)?

Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment.



Markov Decision Processes (MDPs)

We model this problem using an MDP. An MDP consists of

- S, a set of **states**
- $\mathcal{A}$: a set of **actions** that one can take at a state
- $\mathcal{T} : S \times \mathcal{A} \to \{P_a(\cdot|s) : (s, a) \in (S, \mathcal{A})\}$: a **transition** function mapping state-action pairs to probability distribution over next states
- $\mathcal{R} : S \times \mathcal{A} \to [0, r_{\max}]$: a **reward** function mapping state-action pairs to scalar rewards
- $\gamma < 1$: a **discount** factor that decreases the amount of reward receives over time

A **policy** $\pi : S \to \mathcal{A}$ is a function that returns the action to be taken at a given state.

Goal

In modern RL, we assume that the transition distributions and reward function are unknown. Instead, we are only given oracle access to samples of rewards. The goal is to find the policy π^* that maximizes expected sum of discounted rewards.

$$\pi^* \in \operatorname*{argmax}_{\pi} \underbrace{\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \pi(s_t)) \Big| s_0 = s, \pi(s_0) = a\right]}_{\theta^{\pi}(s,a)}$$

The function $\theta^{\pi}(s, a)$ maps state-action pairs to their value under policy π is called the Q-value function.

Classic theory on MDPs have shown that when the transition and reward functions are known, the optimal deterministic policy can be found by computing the fixed point of the Bellman operator $\mathcal{B}$:

$$\mathcal{B}(\theta)(s, a) := \mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim P_a(\cdot|s)}\left[\max_{a' \in \mathcal{A}} \theta(s', a')\right]$$

However, in modern RL we assume that the transitions and rewards are unknown.

Q-learning for (Synchronous) RL

Synchronous Q-learning: at each timestep $t = 1, 2, \ldots$ and for each state-action pair (s, a), we observe sample states s' drawn from the distribution $P_a(\cdot|s)$, and the corresponding rewards.

The goal of Q-learning is learn the quantity θ^{π^*}, which immediately gives the optimal policy. It does so by attempting to mimic the action of the Bellman operator $\mathcal{B}$. Since the transitions and rewards are unknown, and the oracle only gives samples of states and rewards, it uses the empirical version of the Bellman operator, denoted as $\widehat{\mathcal{B}}$.

This gives rise to the following algorithm. For a choice of stepsizes $\{\lambda_k\}_{k=0}^{\infty}$, the iterates $\{\theta_k\}_{k\geq 1}$ satisfy the recursion:

$$\theta_{k+1} = (1 - \lambda_k)\theta_k + \lambda_k \hat{\mathcal{B}}_k(\theta_k)$$

Past Work

Azar et al. [2013] show that the minimax rate in this setting requires at least $\Omega\left(\frac{r_{\max}^2}{\epsilon^2}\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^3}\log\left(\frac{|\mathcal{S}||\mathcal{A}|}{\delta}\right)\right) \equiv \Omega\left(\frac{1}{(1-\gamma)^3}\right)$ samples. However, the algorithms that have been shown to achieve this rate are not the standard Q-learning algorithm.

Moreover, Wainwright [2019a] shows that the classical Q-learning algorithm cannot achieve this rate, since it can require at least $\Omega\left(\frac{1}{(1-\gamma)^4}\right)$ samples. A natural question is then whether there is a simple variant of Q-learning that can achieve this rate.

Variance reduced Q-learning vs. Original Q-learning

To close the gap between minimax lower and upper bounds, Wainwright [2019b] proposes a new algorithm called *variance reduced Q-learning*.

The key difference between these algorithms is the way in which we perform the update:

Original Q-learning update	Variance reduced Q-learning update
$\theta_{k+1} = (1-\lambda_k)\theta_k + \lambda_k \hat{\mathcal{B}}_k(\theta_k)$	$\theta_{k+1} = (1-\lambda_k)\theta_k$ $+\lambda_k \left\{ \hat{\mathcal{B}}_k(\theta_k) - \hat{\mathcal{B}}_k(\bar{\theta}) + \tilde{\mathcal{B}}_N(\bar{\theta}) \right\}$

where $\bar{\theta}$ is the result obtained after every M iterations and $\tilde{\mathcal{B}}_k(\theta)$ is a form of rolling average.

One can immediately draw parallels to update made in SVRG [Johnson and Zhang, 2013] in the form of a control variate.

References

Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349, 2013.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

Martin J. Wainwright. Stochastic approximation with cone-contractive operators: Sharp ℓ_∞-bounds for q-learning, 2019a.

Martin J. Wainwright. Variance-reduced q-learning is minimax optimal, 2019b.

Nonparametric Stochastic Volatility

Yan Gao

10716 Project Presentation

Bandi, Federico M., and Roberto Renò. "Nonparametric stochastic volatility." *Econometric Theory* 34.6 (2018): 1207-1255.

Price-Volatility System

$$\mathrm{d}\log p_t = \mu(\sigma_t^2)\mathrm{d}t + \sigma_t \mathrm{d}W_t^r + \mathrm{d}J_t^r,$$
$$\mathrm{d}f(\sigma_t^2) = m_{f(\sigma^2)}(\sigma_t^2)\mathrm{d}t + \Lambda_{f(\sigma^2)}(\sigma_t^2)\mathrm{d}W_t^\sigma + \mathrm{d}J_t^\sigma,$$

- W: independent, standard Brownian motions; J: independent Poisson jump processes; f: monotonically non-decreasing transformation of variance;
- Nested and generalized system. Different f and assumptions.
- Two-step work:
  1. Estimate spot variance by localizing a high-frequency estimate of integrated variance robust to the presence of price jumps. i.e., $\int_t^{t+\phi} \sigma_s^2 \mathrm{d}s$
  2. Use the resulting spot variance estimates to identify the parameters in the price-volatility system.
- The second stage requires controlling the estimation error introduced by the first-step spot variance estimates.

Main Contribution

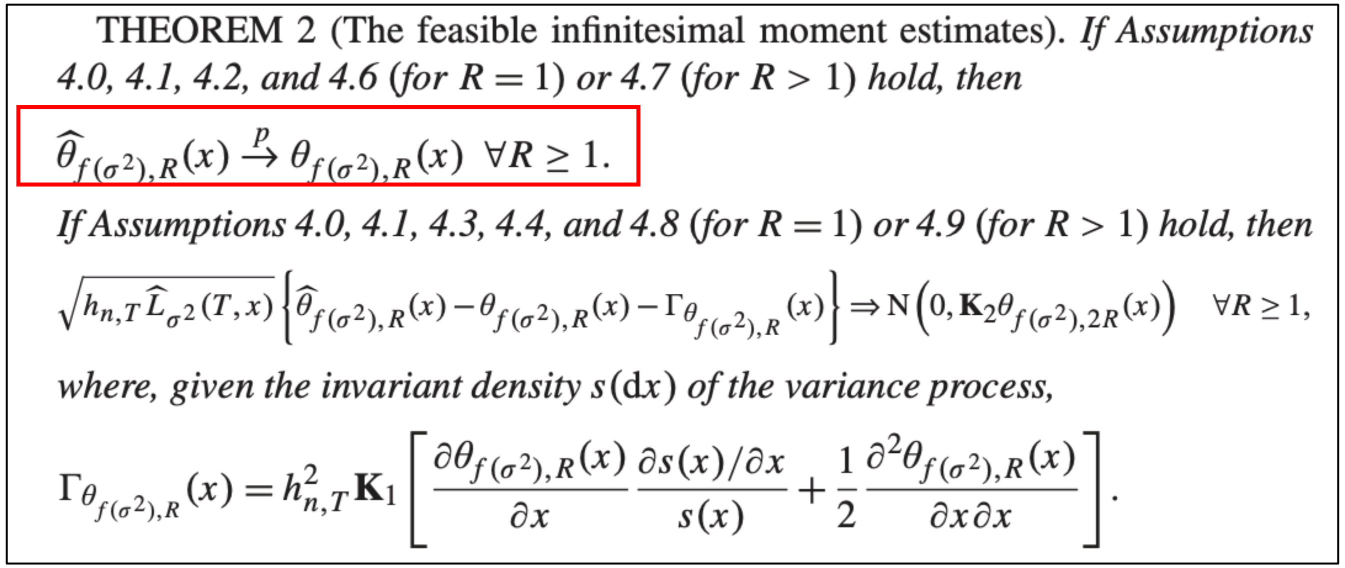
- Use non-parametric kernel methods to estimate the price-volatility system's *infinitesimal moments*, and then derive parameters.

$$\theta_{f(\sigma^2),R}(x) = \lim_{\Delta \to 0} \frac{1}{\Delta} \mathrm{E}\left[\left(f(\sigma^2_{t+\Delta}) - f(\sigma^2_t) \right)^R | \sigma^2_t = x \right] \qquad R = 1, \ldots,$$

$$\widehat{\theta}_{f(\sigma^2),R}(x) = \frac{1}{\Delta_{n,T}} \frac{\sum_{i=1}^{n-1} \mathbf{K}\left(\frac{\widehat{\sigma}^2_{i\Delta_{n,T}} - x}{h_{n,T}} \right) \left(f(\widehat{\sigma}^2_{(i+1)\Delta_{n,T}}) - f(\widehat{\sigma}^2_{i\Delta_{n,T}}) \right)^R}{\sum_{=1}^{n} \mathbf{K}\left(\frac{\widehat{\sigma}^2_{i\Delta_{n,T}} - x}{h_{n,T}} \right)}$$

$$\widehat{\sigma}^2_{i\Delta_{n,T}} = \frac{1}{\phi} \sum_{j=1}^{k} r^2_{i,j} \, \mathbf{1}_{\{r^2_{i,j} \leq \vartheta\}},$$

$$r_{i,j} = \log p_{i\Delta_{n,T} - \phi + j\phi/k} - \log p_{i\Delta_{n,T} - \phi + (j-1)\phi/k}$$



THEOREM 2 (The feasible infinitesimal moment estimates). *If Assumptions 4.0, 4.1, 4.2, and 4.6 (for $R = 1$) or 4.7 (for $R > 1$) hold, then*

$$\boxed{\widehat{\theta}_{f(\sigma^2),R}(x) \xrightarrow{p} \theta_{f(\sigma^2),R}(x) \;\; \forall R \geq 1.}$$

If Assumptions 4.0, 4.1, 4.3, 4.4, and 4.8 (for $R = 1$) or 4.9 (for $R > 1$) hold, then

$$\sqrt{h_{n,T} \widehat{L}_{\sigma^2}(T,x)} \left\{ \widehat{\theta}_{f(\sigma^2),R}(x) - \theta_{f(\sigma^2),R}(x) - \Gamma_{\theta_{f(\sigma^2),R}}(x) \right\} \Rightarrow \mathrm{N}\left(0, \mathbf{K}_2 \theta_{f(\sigma^2),2R}(x)\right) \quad \forall R \geq 1,$$

where, given the invariant density $s(\mathrm{d}x)$ of the variance process,

$$\Gamma_{\theta_{f(\sigma^2),R}}(x) = h^2_{n,T} \mathbf{K}_1 \left[\frac{\partial \theta_{f(\sigma^2),R}(x)}{\partial x} \frac{\partial s(x)/\partial x}{s(x)} + \frac{1}{2} \frac{\partial^2 \theta_{f(\sigma^2),R}(x)}{\partial x \partial x} \right].$$

Main Contribution

- Use non-parametric kernel methods to estimate the price-volatility system's *infinitesimal moments*, and then derive parameters.

$$\theta_{f(\sigma^2),R}(x) - \frac{\sum_{i=1}^{n-1} \mathbf{K}_{i\Delta_{n,T}} \left(\Delta f_{i\Delta_{n,T}}\right)^R}{\Delta_{n,T} \sum_{i=1}^{n} \mathbf{K}_{i\Delta_{n,T}}} = \underbrace{\frac{\sum_{i=1}^{n-1} \mathbf{K}_{i\Delta_{n,T}} \left(\Delta f_{i\Delta_{n,T}}\right)^R}{\Delta_{n,T} \sum_{i=1}^{n} \widehat{\mathbf{K}}_{i\Delta_{n,T}}} - \frac{\sum_{i=1}^{n-1} \mathbf{K}_{i\Delta_{n,T}} \left(\Delta f_{i\Delta_{n,T}}\right)^R}{\Delta_{n,T} \sum_{i=1}^{n} \mathbf{K}_{i\Delta_{n,T}}}}_{\Pi_{1,n,T,k,\phi}} \qquad O_p(1)\, O_p\left(\frac{\mathbf{M}_{n,k,T,\phi}}{h_{n,T}}\right).$$

$$+ \underbrace{\frac{\sum_{i=1}^{n-1} \widehat{\mathbf{K}}_{i\Delta_{n,T}} \left(\Delta f_{i\Delta_{n,T}}\right)^R}{\Delta_{n,T} \sum_{i=1}^{n} \widehat{\mathbf{K}}_{i\Delta_{n,T}}} - \frac{\sum_{i=1}^{n-1} \mathbf{K}_{i\Delta_{n,T}} \left(\Delta f_{i\Delta_{n,T}}\right)^R}{\Delta_{n,T} \sum_{i=1}^{n} \widehat{\mathbf{K}}_{i\Delta_{n,T}}}}_{\Pi_{2,n,T,k,\phi}} \qquad O_p\left(\frac{\mathbf{M}_{n,k,T,\phi}\,\Delta^*_{n,T}}{\Delta^2_{n,T}}\right) \qquad O_p\left(\frac{\mathbf{M}_{n,k,T,\phi}\,\Delta^*_{n,T}}{h_{n,T}\,\Delta^{3/2}_{n,T}}\right).$$

$$+ \underbrace{\frac{\sum_{i=1}^{n-1} \widehat{\mathbf{K}}_{i\Delta_{n,T}} \left(\widehat{\Delta f}_{i\Delta_{n,T}}\right)^R}{\Delta_{n,T} \sum_{i=1}^{n} \widehat{\mathbf{K}}_{i\Delta_{n,T}}} - \frac{\sum_{i=1}^{n-1} \widehat{\mathbf{K}}_{i\Delta_{n,T}} \left(\Delta f_{i\Delta_{n,T}}\right)^R}{\Delta_{n,T} \sum_{i=1}^{n} \widehat{\mathbf{K}}_{i\Delta_{n,T}}}}_{\Pi_{3,n,T,k,\phi}}. \qquad O_p(\mathbf{M}_{n,k,T,\phi}/\Delta_{n,T}). \quad O_p\left(\mathbf{M}_{n,k,T,\phi}\,\Delta^*_{n,T}/\Delta^{3/2}_{n,T}\right).$$

$$\mathbf{M}_{n,k,T,\phi} = \max_{1 \le i \le n} \left| \widehat{\sigma}^2_{i\Delta_{n,T}} - \sigma^2_{i\Delta_{n,T}} \right| = O_p(g(n,T,k,\phi) h_{n,T}). \qquad g(n,T,k,\phi)\frac{h_{n,T}}{\Delta_{n,T}} \to 0$$

$$g(n,T,k,\phi) = \frac{(\log n \log nk)^{1/2}}{h_{n,T} k^{1/2}} + \frac{\phi^{1/2} (\log n)^{1/2}}{h_{n,T}} + \frac{n\phi}{h_{n,T}} \to 0 \qquad g(n,T,k,\phi)\frac{1}{\Delta^{1/2}_{n,T}} \to 0$$

Modified Bounded Domain

$$\mathrm{d}\log p_t = \mu(\sigma_t^2)\mathrm{d}t + \sigma_t \mathrm{d}W_t^r + \mathrm{d}J_t^r,$$

$$\mathrm{d}f(\sigma_t^2) = \boxed{m_{f(\sigma^2)}(\sigma_t^2)}\mathrm{d}t + \boxed{\Lambda_{f(\sigma^2)}(\sigma_t^2)}\mathrm{d}W_t^\sigma + \mathrm{d}J_t^\sigma,$$

$$\widetilde{\sigma}_t^2 = \overline{\sigma}^2\left(1 - \exp\{-\sigma_t^2/\overline{\sigma}^2\}\right),$$

Assumption 1. (a) Define the system in equations (1)–(2) on a filtered probability space $(\Omega, (\mathcal{F}_t)_{0\le t<\infty}, \mathcal{P})$ satisfying the usual conditions. Let $\mu(.)$, $m_{f(\sigma^2)}(.)$, $\Lambda_{f(\sigma^2)}(.)$, $\rho(.)$, $\lambda^r(.)$ and $\lambda^{f(\sigma^2)}(.)$ be uniformly bounded and, at least, twice continuously-differentiable with bounded derivatives of any order.

- where $\bar{\sigma}^2$ is a positive constant, $0 \le \tilde{\sigma}_t{}^2 \le \bar{\sigma}^2$
- For any *fixed* σ_t^2, $\tilde{\sigma}_t{}^2$ is arbitrarily close to σ_t^2 for a sufficiently large $\bar{\sigma}^2$

$$\widetilde{\sigma}_t^2 = \overline{\sigma}^2\left(1 - \left(1 - \sigma_t^2/\overline{\sigma}^2 + \sigma_t^4 O\left(\frac{1}{\overline{\sigma}^4}\right)\right)\right) = \sigma_t^2 + O\left(\frac{1}{\overline{\sigma}^2}\right)$$

- The modified process has a bounded domain and can be made arbitrarily close to the original system by letting σ2 be sufficiently large.

$$\mathrm{d}\log p_t = \mu(\widetilde{\sigma}_t^2)\mathrm{d}t + \widetilde{\sigma}_t \mathrm{d}W_t^r + \mathrm{d}J_t^r,$$

$$\mathrm{d}f(\widetilde{\sigma}_t^2) = m'_{f(\widetilde{\sigma}^2)}(\widetilde{\sigma}_t^2)\mathrm{d}t + \Lambda'_{f(\widetilde{\sigma}^2)}(\widetilde{\sigma}_t^2)\mathrm{d}W_t^\sigma + \mathrm{d}J_t^{\prime,\sigma},$$

Moments and Parameters Estimates

- Moments $\theta_{\tilde{\sigma}_t{}^2,R}(x)$ of the bounded process $\tilde{\sigma}_t{}^2$ (for a sufficiently large $\bar{\sigma}^2$)
- When $f(\sigma_t^2) = \sigma_t^2$ and $dJ_t^\sigma = \xi^\sigma dN_t^\sigma$, where $\xi^\sigma \overset{\mathrm{d}}{=} \exp(\mu_\xi)$.

$$\theta_{\tilde{\sigma}^2,1}(x) \asymp m_{\sigma^2}(x) + \mu_\xi(x)\lambda^{\sigma^2}(x),$$
$$\theta_{\tilde{\sigma}^2,2}(x) \asymp \Lambda^2_{\sigma^2}(x) + 2\mu_\xi^2(x)\lambda^{\sigma^2}(x),$$
$$\theta_{\tilde{\sigma}^2,3}(x) \asymp 6\mu_\xi^3(x)\lambda^{\sigma^2}(x),$$
$$\theta_{\tilde{\sigma}^2,4}(x) \asymp 24\mu_\xi^4(x)\lambda^{\sigma^2}(x),$$

$$\widehat{\mu}_\xi(x) = \frac{\widehat{\theta}_{\sigma^2,4}(x)}{4\widehat{\theta}_{\sigma^2,3}(x)},$$
$$\widehat{\lambda}^{\sigma^2}(x) = \frac{\widehat{\theta}_{\sigma^2,4}(x)}{24\widehat{\mu}_\xi^4(x)},$$
$$\widehat{\Lambda}^2_{\sigma^2}(x) = \widehat{\theta}_{\sigma^2,2}(x) - 2\widehat{\mu}_\xi^2(x)\widehat{\lambda}^{\sigma^2}(x),$$
$$\widehat{m}_{\sigma^2}(x) = \widehat{\theta}_{\sigma^2,1}(x) - \widehat{\mu}_\xi(x)\widehat{\lambda}^{\sigma^2}(x).$$

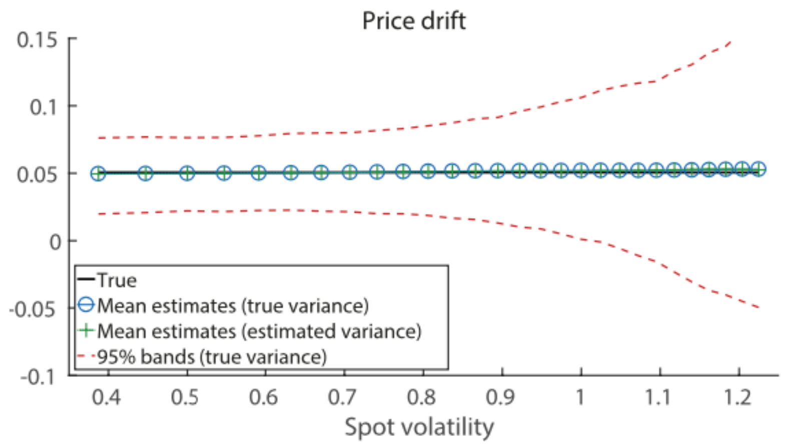
$$\widehat{\theta}_{f(\sigma^2),R}(x) = \frac{1}{\Delta_{n,T}} \frac{\sum_{i=1}^{n-1} \mathbf{K}\left(\frac{\widehat{\sigma}^2_{i\Delta_{n,T}} - x}{h_{n,T}}\right)\left(f(\widehat{\sigma}^2_{(i+1)\Delta_{n,T}}) - f(\widehat{\sigma}^2_{i\Delta_{n,T}})\right)^R}{\sum_{i=1}^{n} \mathbf{K}\left(\frac{\widehat{\sigma}^2_{i\Delta_{n,T}} - x}{h_{n,T}}\right)} \quad R = 1, \ldots,$$

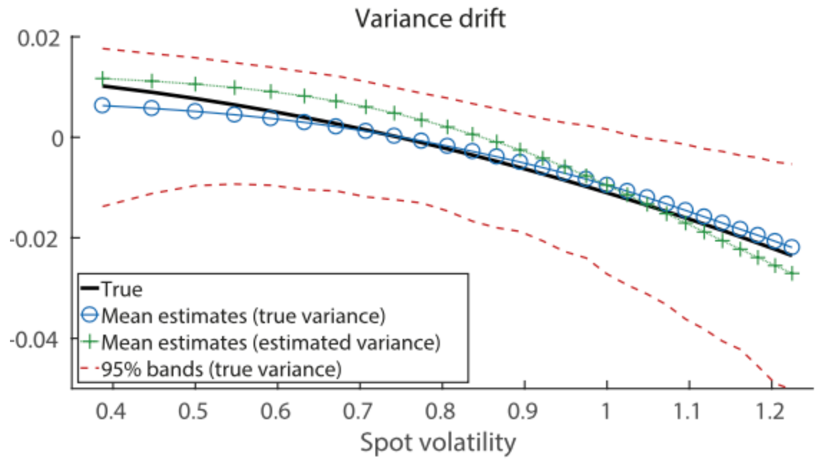
Simulation

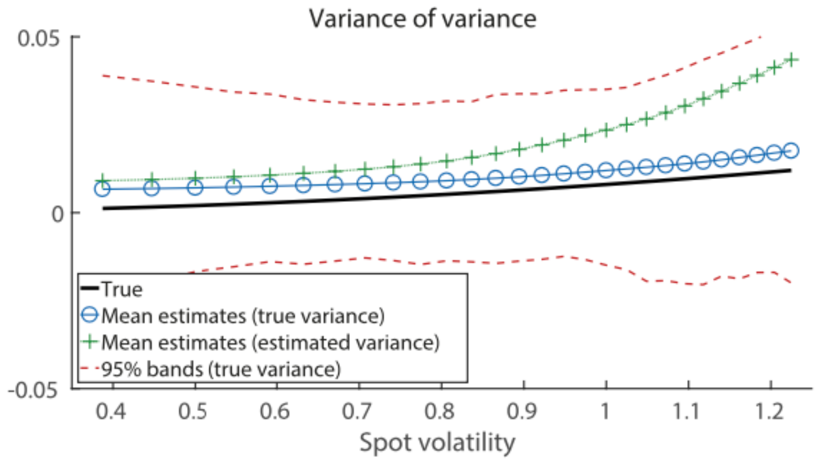
- S&P500, 1,000 replications, 5,000 days, mimicking 10-second observations sampled over a 1-hour time interval; Epanechnikov kernel for all infinitesimal moments;

$$\log p_{t+\Delta_t} - \log p_t = b\Delta_t + \sqrt{\sigma_t^2 \Delta_t}\varepsilon_t^r + \psi^r J_t^r,$$

$$\sigma_{t+\Delta_t}^2 - \sigma_t^2 = \kappa(\theta - \sigma_t^2)\Delta_t + \sigma_v\sqrt{\sigma_t^2 \Delta_t}\varepsilon_t^\sigma + \xi^\sigma J_t^\sigma,$$







Eraker, B., M. Johannes, & N. Polson (2003) The impact of jumps in volatility and returns. *Journal of Finance* 58, 1269–1300.

Reference

- Bandi, Federico M., and Roberto Renò. "Nonparametric stochastic volatility." *Econometric Theory* 34.6 (2018): 1207-1255.
- Federico M Bandi and Thong H Nguyen. On the functional estimation of jump–diffusion models. Journal of Econometrics, 116(1-2):293–328, 2003.
- Michael Johannes. The statistical and economic role of jumps in continuous-time interest rate models. The Journal of Finance, 59(1):227–260, 2004.
- Darrell Duffie, Jun Pan, and Kenneth Singleton. Transform analysis and asset pricing for affine jump-diffusions. *Econometrica*, 68(6):1343–1376, 2000.
- Bjørn Eraker, Michael Johannes, and Nicholas Polson. The impact of jumps in volatility and returns. *The Journal of Finance*, 58(3):1269–1300, 2003.
- Eric Jacquier, Nicholas G Polson, and Peter E Rossi. Bayesian analysis of stochastic volatility models. *Journal of Business & Economic Statistics*, 20(1):69–87, 2002.

Thanks

- Yan Gao
- 10716 Project Presentation

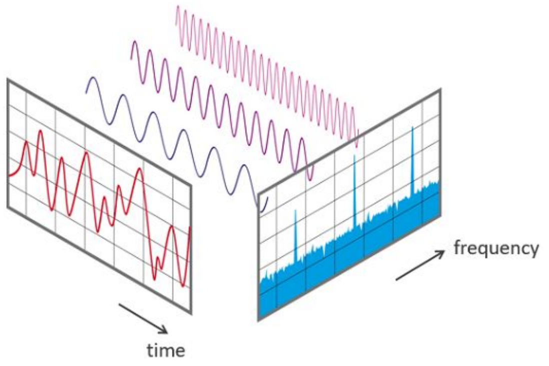
- Bandi, Federico M., and Roberto Renò. "Nonparametric stochastic volatility." *Econometric Theory* 34.6 (2018): 1207-1255.

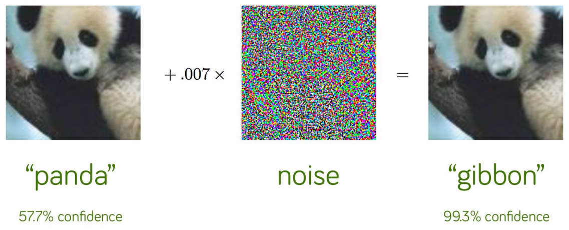
Towards Frequency-based Explanation for Predictions from CNN-based classifiers

Varun Rawal, Zihao Ding

Background

- Convolutional Neural Network (CNN)
  - Interpretability
    - Human-understandable input features
    - Frequency components decomposition
  - Performance
    - Powerful in classification
    - Vulnerability





Objective

1. Attack pre-trained CNN model with adversarial attacks
2. Analyze adversarial attacks in frequency domain
3. Generate robust dataset
4. Determine frequency vulnerability

CNN and adversarial attack

- Update weights in the model

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i$$

$$\theta = \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

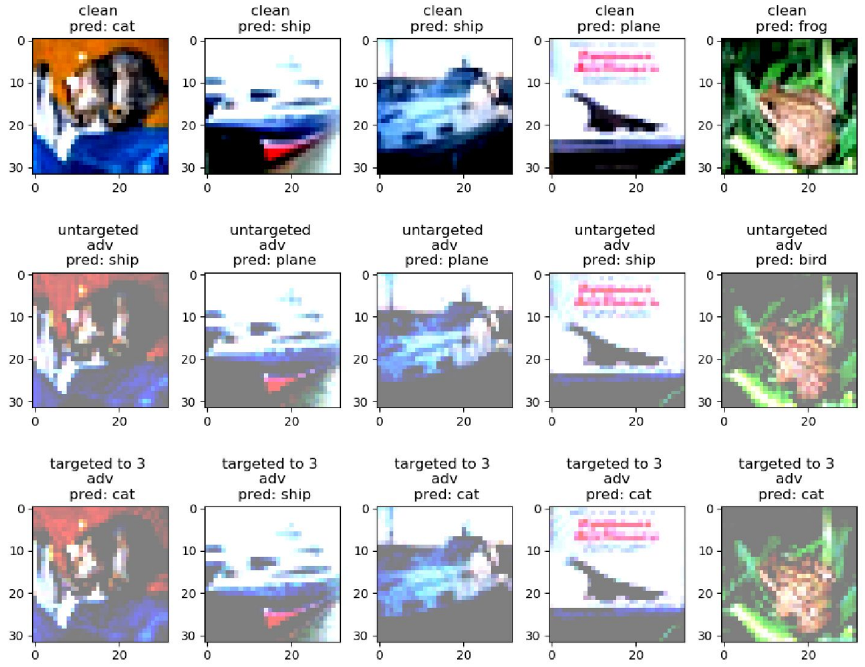
- Update input

$$\mathcal{L}(x) = -y_i \log f_\theta(x)_i$$

$$x = x + \eta \nabla_x \mathcal{L}(x)$$

- Attacking methods used:

  L-BFGS; FGSM; PGD; CW



Performance of pre-trained VGG-Cifar10 model under targeted/untargeted attack

Frequency analysis

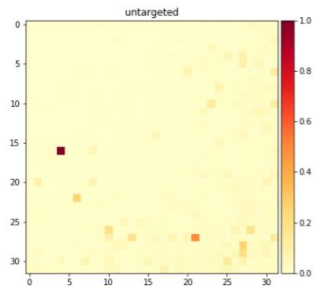
- Discrete cosine transform (DCT):

$$F(u) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \Lambda(i).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] f(i)$$

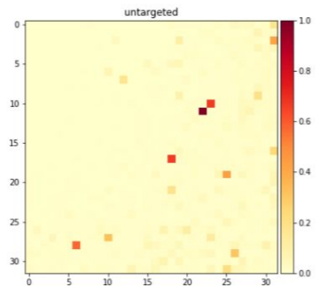
- DCT-2D:
    - dct(dct(a.T).T)
- Relative change (RCT):

$$\mathrm{RCT} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \left|\frac{\mathcal{F}(x_{\mathrm{i}}') - \mathcal{F}(x_{\mathrm{i}})}{\mathcal{F}(x_{\mathrm{i}})}\right|$$

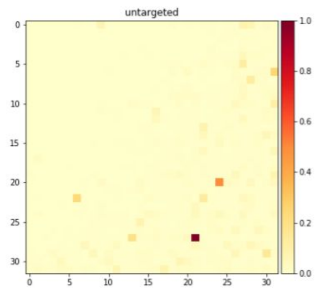
Hypothesis 1: *Low-frequency features are more robust than the high-frequency features.*



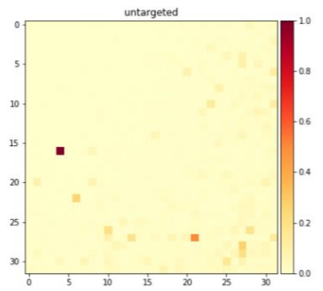
(a) PGD Attack



(b) L-BFGS attack



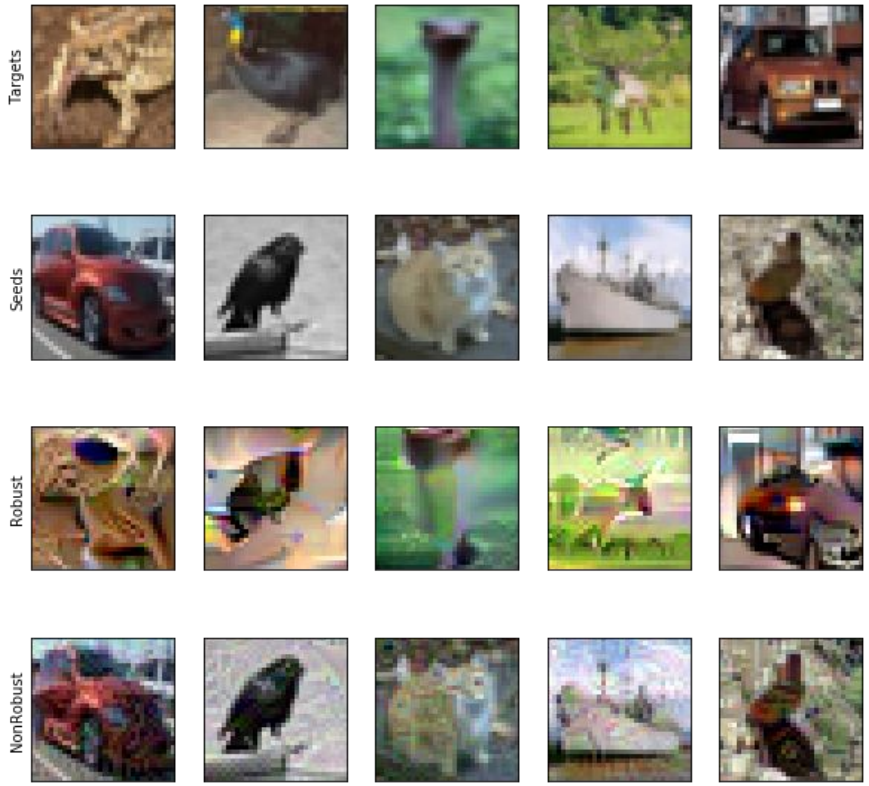
(c) CW attack



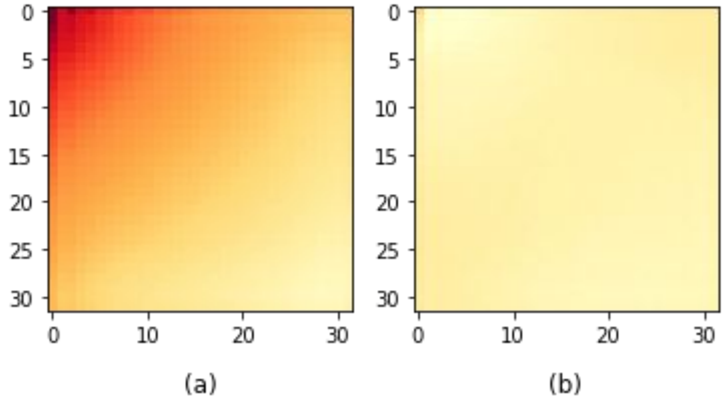
(d) FGSM attack

RCT maps for different adversarial attacks

Robust Dataset in Frequency Domain



DCT Analysis



Robust

Non-Robust

Frequency attribution

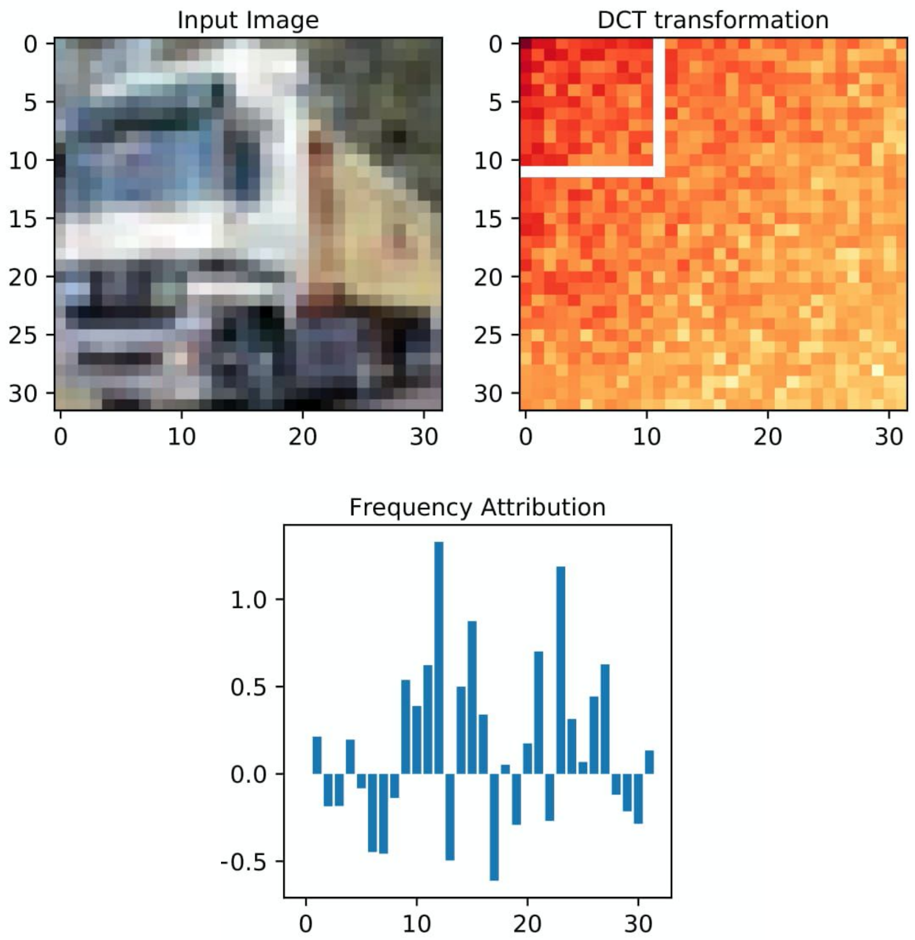
Definition 1 (Occluded Frequency (OF)) *Given a model $y = f(x)$, a class of interest c, let S be a DCT function and S^{-1} be its inverse function. Define $H(i)$ as a matrix with the same shape of $S(x)$ whose each entry h_k is*

$$h_k = \mathbb{I}[k = i]\sigma(S(x)_k) + \mathbb{I}[k \neq i]S(x)_k \qquad (6)$$

where $\sigma(\cdot)$ is a function that transforms an frequency component $S(x)_k$ to a baseline version, e.g. zeros or random signal. Therefore, the occlusion score $O(S(x)_i)$ for frequency component $S(x)_i$ towards class c is defined as

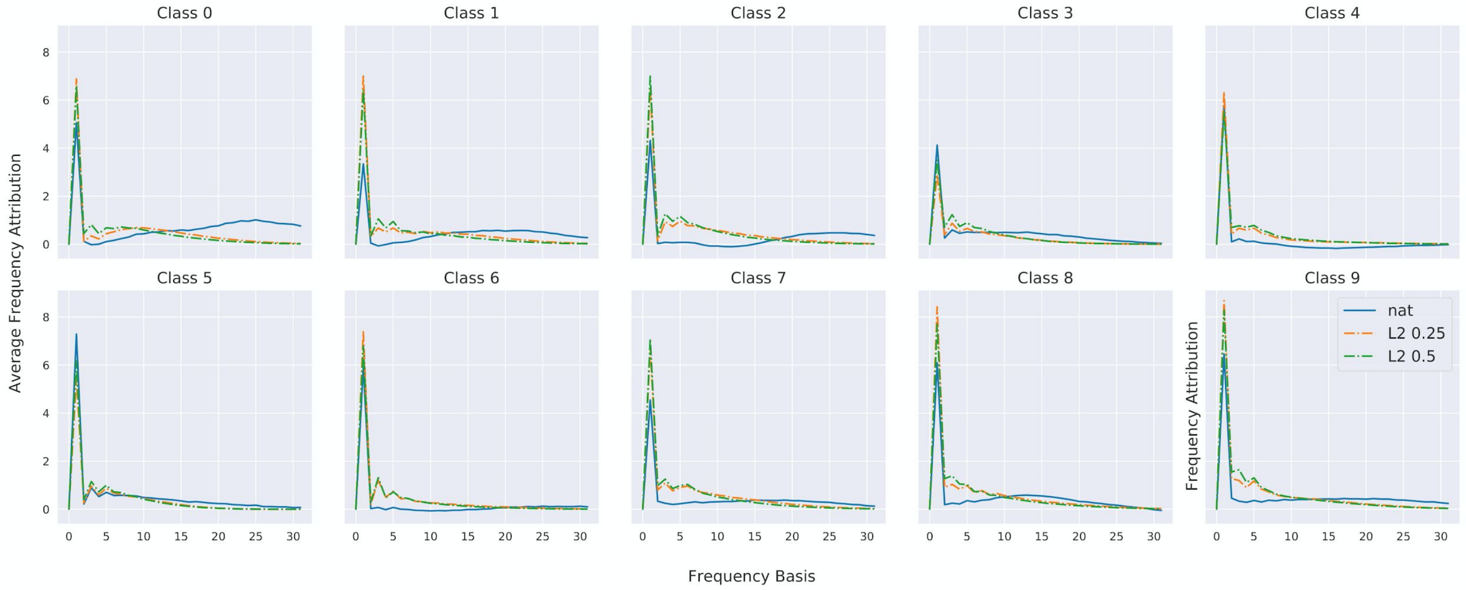
$$O(S(x)_i) = f^c(x) - f^c(S^{-1}(H(i)))$$

where $f^c(\cdot)$ is the logit score of class c.



Average Attribution scores for each frequency components on each subset of CIFAR-10 dataset. We compute the attribution scores on three ResNet models. (Blue : natural, others : Robust)

Robust models tend to shift the attribution scores from the high frequency range to the low frequency range, compared to the naturally trained models.



Hypothesis 2: *A model is more vulnerable to the current adversarial attacks if the relevant features towards the prediction are not from low-frequency range.*

Hierarchical Learning in Neural Networks

Jun-Ting Hsieh, Bingbin Liu

Understanding Neural Networks

Neural tangent kernel (NTK): approximate NNs well in the infinite width limit:

For large width, learning a multi-layer NN with SGD is close to a kernel method with the specific kernel defined as

$$K(x, y) = \mathbb{E}_\theta \langle \nabla_\theta f(\theta, x), \nabla_\theta f(\theta, y) \rangle$$

However: in practice, NNs usually outperform kernel methods.

Can we provably separate the power of NNs and kernel methods?

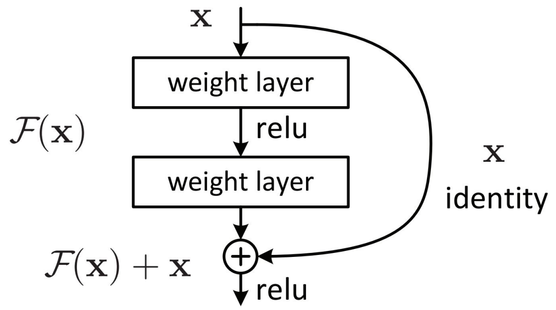
3-layer Residual Networks

[AL19] What Can ResNet Learn Efficiently, Going Beyond Kernels?

Hierarchical Learning

Key observations:

1. Neural networks with **residual connections** perform **hierarchical learning**.



2. Traditional methods such as kernel methods are **non-hierarchical**.

Hierarchical Function Class

Consider the function

$$\mathcal{H}(x) = \mathcal{F}(x) + \alpha\mathcal{G}(\mathcal{F}(x))$$

Learning $\mathcal{F}, \mathcal{G}$ separately is “easy”, but learning $\mathcal{F} \circ \mathcal{G}$ directly is “hard”.

Example: $\mathcal{F}(x) = x_1 + x_2,\ \mathcal{G}(y) = y^4,\ \mathcal{H}(x) = x_1 + x_2 + \alpha(x_1 + x_2)^4$

Goal: prove that

1. **3-layer ResNets** can learn this function class efficiently.
2. Kernel methods cannot.

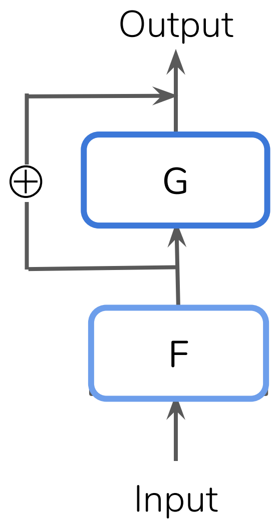
Intuition

3-layer ResNet

- ResNets automatically **distribute the learning** of $\mathcal{F}, \mathcal{G}$ between layers.
- The residual link is crucial (otherwise it is equivalent to kernels).
- Sample complexity ~ $C_{\mathcal{F}}^2 + C_{\mathcal{G}}^2$ (for some complexity notion).

Kernels

- Kernels cannot separate $\mathcal{F}, \mathcal{G}$.
- Sample complexity ~ $C_{\mathcal{G}\circ\mathcal{F}}^2$, which can be much larger.



Main Result

Separation in learning efficiency.

1. 3-layer ResNets learn any hierarchical function $\mathcal{H}$ up to $O(\alpha^2)$ generalization error with sample complexity $O(C_{\mathcal{F}}^2 + C_{\mathcal{G}}^2)/\alpha^4$.
2. For **any** kernel method, there is some hierarchical function $\mathcal{H}$ such that achieving $o(\alpha)$ generalization error requires sample complexity $O(C_{\mathcal{G}\circ\mathcal{F}}^2)$.

Conceptual messages:

- ResNets perform implicit **hierarchical learning**.
- Kernels must learn from scratch.

Going beyond 3 layers

[AL20] Backward Feature Correction: How Deep Learning Performs Deep Learning

Hierarchical Learning in deep networks

Note: the previous analysis only applies to 3-layer neural networks.

Joint learning of deep networks

Goal: an L-layer NN can efficiently learn concept classes not learnable by

1. Any network with at most (L-1)-layers; or

Going deeper $\longrightarrow$ *more efficient learning*

2. Any kernel methods.

Hierarchical learning $\longrightarrow$ *more efficient learning*

Concept Class

DenseNet with skip connections

$$G(x) = \sum_{\ell} u_\ell^\top G_\ell(x) \in \mathbb{R} \longrightarrow \boxed{\text{scalar-valued function with coordinate sum}}$$

where

$$G_0(x) = x \in \mathbb{R}^d, \qquad G_1(x) = \sigma(x) - \mathbb{E}[\sigma(x)] \in \mathbb{R}^d$$

$$G_\ell(x) = \sigma\left(\sum_{j \in \mathcal{J}_\ell} \mathbf{W}_{\ell,j} G_j(x)\right) \in \mathbb{R}^{k_\ell} \qquad \text{for } \ell \geq 2$$

(※) $\sigma(x) = x^2$: L layer → polynomial of degree 2^L.

(easier to measure the representation power)

Main Result

Separation in learning efficiency: for input in any $\mathbb{R}^d$ and $L = o(\log \log d)$,

to get a ε generalization error, the sample complexity required is

- $poly(d/\epsilon)$ ... for the L-layer DenseNet
- $d^{\omega(1)}$ ... for any kernel methods, linear regression, or 2-layer NN.

Conceptual messages:

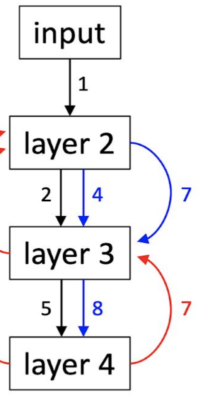
- Overparameterization: providing a rich set of features for subsequent layers.
- Backward feature correction: important for hierarchical learning.

Hierarchical learning: Backward Feature Correction

Joint training is required: higher layers correct the features learned by lower layers.

e.g. ground truth $f(x) = x_1^4 + x_2^4 + 0.3(x_1^4 + x_2^4)^2$

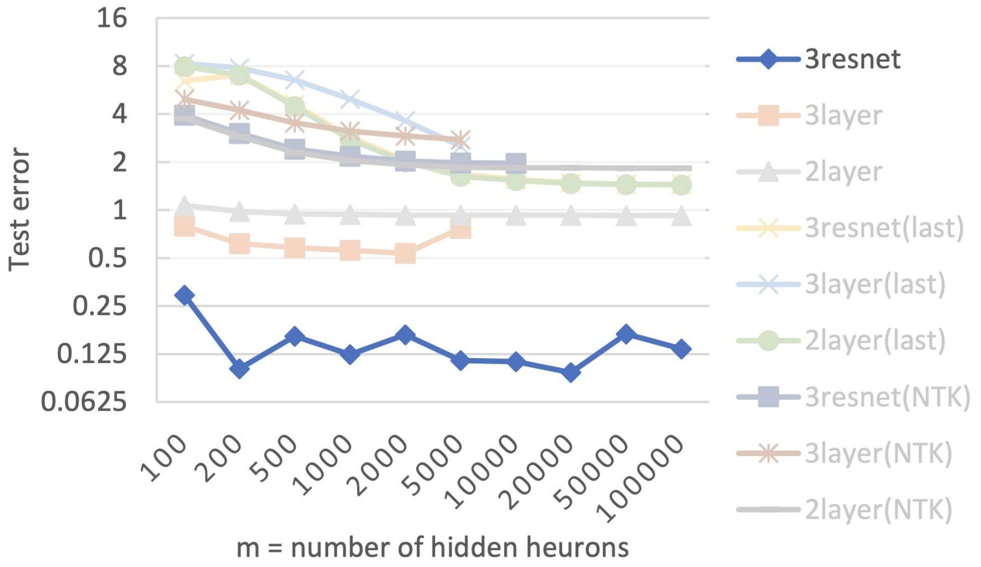
- Learning with degree-4 polynomials will likely not recover the right "base" $x_1^4 + x_2^4$.



1) layer 2 begins to learn
2) layer 3 begins to learn using features given by layer 2
3) layer 3 helps layer 2 correct its features by reducing over-fitting
4) layer 3 improves since layer 2 now feeds in better features to layer 3
5) layer 4 begins to learn
6) layer 4 helps layer 2 correct its features by reducing over-fitting
7) layer 3 improves since
   a) layer 4 helps layer 3 correct its features by reducing over-fitting, and
   b) layer 2 now feeds in better features to layer 3
8) layer 4 improves since layer 3 now feeds in better features

Experiments

Synthetic dataset: only the jointly trained 3-layer ResNet achieves non-trivial error.



Summary

- Neural tangent kernels cannot explain neural network's practical success.
- Hierarchical learning is key to the efficient learning of neural networks, by:
  - Implicitly **distributing learning** into different layers; and
  - Finding the right "base" features through **backward feature correction**.

Thank you!

References

1. Neural tangent kernel [arXiv:1806.07572]
2. On Exact Computation with an Infinitely Wide Neural Net [arXiv:1904.11955]
3. What Can ResNet Learn Efficiently, Going Beyond Kernels? [arXiv:1905.10337]
    a. Deep Residual Learning for Image Recognition [arXiv:1512.03385]
4. Backward Feature Correction: How Deep Learning Performs Deep Learning [arXiv:2001.04413]
    a. Densely Connected Convolutional Networks [arXiv:1608.06993]

Task Relatedness in Multi-task Learning

Helen Ren Ruohan Li

April 27, 2020

Introduction

Background

- Multi-task learning (MTL) tries to jointly learn useful representation via multiple related tasks to help improve the generalization performance of the tasks.
- Successfully applied to solve data and annotation insufficiency problem.
- Broad applications in spam filtering, face authentication etc.
- Usually, multiple tasks share some commonalities.

Motivation

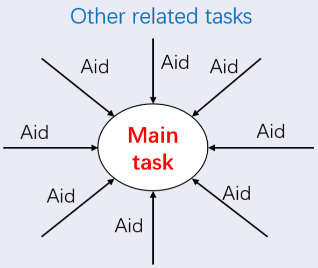
- Any theoretical guarantees for the MTL's success?
- What kinds of tasks can be jointly learned?
- How much information can be shared across multiple tasks?
- Is there any guidelines on the sufficient sample size for each individual task?
- Task relatedness between tasks is the key.

Key Results

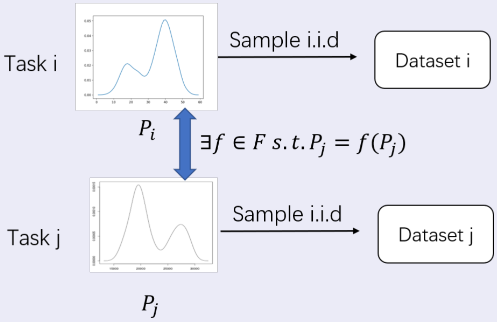
Shai *et al.* 2008: Mapping data generation mechanism

- Sub-domain: ”One focus of interest”
- Key assumption: Pairwise F-relatedness
- Guarantee: Provided lower bound on sample size that depends on a generalized VC-dimension parameter.

One focus of interest



Pairwise F-relatedness $\forall i \neq j$



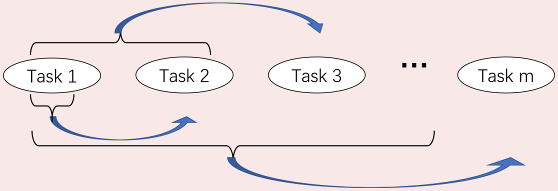
Key Results

Baxter *et al.* 1997: Modeling inductive bias

- Simultaneous MTL
- The task relatedness can be considered as the existence of a sub-hypothesis space/inductive hyper bias which contains the best hypothesis/bias for each task.

Mahmud *et al.* 2009: Information-based approach

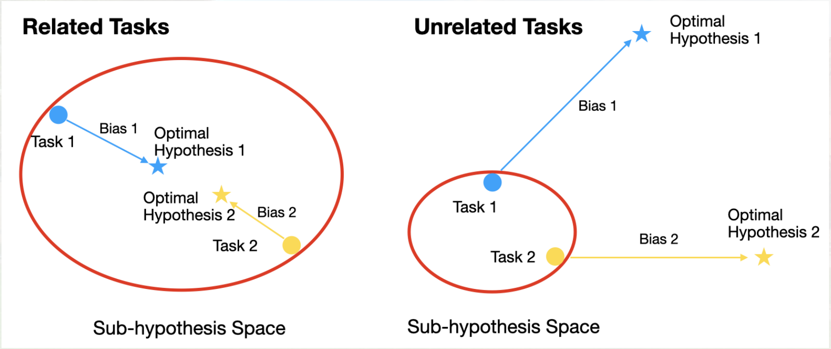
- Sequential Transfer Learning
- Proposed universal measures of task relatedness from the Algorithmic Information Theory perspective.
- Developed universally optimal Bayesian transfer learning methods, which could provide guidance to construct practical transfer learning algorithms.



Experiments

By Baxter's theory, two tasks can be learned together when:

- they have the same input space
- they require a common set of low dimensional representations
- they are included in a sub-hypothesis space that contains the optimal hypothesis



Experiments

Building **Related** Task Pair

- Input: $\{X, y\}_{i=1}^{N}$, where $y = \text{poly}(X)$
- Model: linear regression with SGD
- 1000 training samples and 200 testing samples per task

Mean Square Error

		Single Task Learning	Multi-task Learning
Related Tasks	Task 1	3.7793	**3.7416**
	Task 2	1.6277	**1.5517**
Unrelated Tasks	Task 1	**3.4498**	3.5025
	Task 2	**5.4768**	5.6758

Conclusion

Comparison of Representative Works

	Sub-domain of MTL	Assumption
Shai 2008	Learning for One Target Task	F-Relatedness[1]
Baxter 1997	Simultaneous MTL	True Prior in Common Set[1]
Mahmud 2009	Sequential Transfer Learning	Semi-computability
	Notion of Task Relatedness	**Relatedness Measurement**
Shai 2008	Clearly Defined	Discriminate Measurement
Baxter 1997	Generally Included	N/A
Mahmud 2009	Clearly Defined	Continuous Measurement
	Generalization Guarantee	**Applications**
Shai 2008	Sample Size	Tasks w/ Slightly Domain Shifts
Baxter 1997	Sample Size	Tasks w/ Common LDRs[2]
Mahmud 2009	Error Bound	Most Learning Tasks

[1] Strong assumption.

[2] LDR stands for low dimensional representation.

Thank You!

GANs and Image Super Resolution

John Fang & Niles Christensen

“On Unifying Deep Generative Models” (Hu et al. 2018)

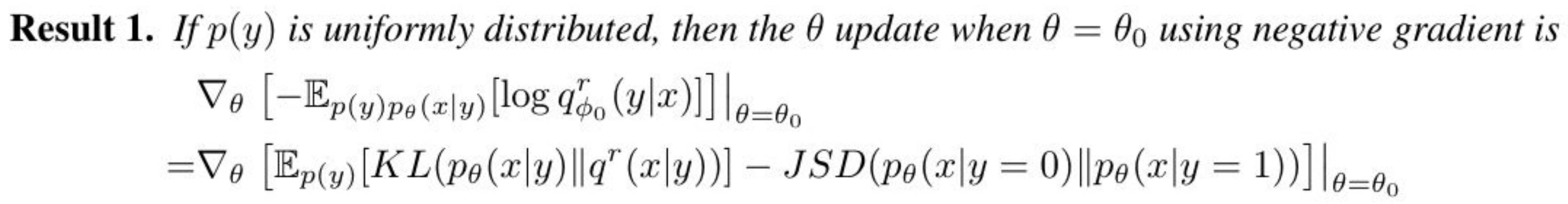
- Wake-sleep algorithm:

$$\max_{\psi} \mathbb{E}_{p_{data}(x) q_{\lambda}(z|x)} [\log p_{\psi}(x|z)]$$
$$\max_{\lambda} \mathbb{E}_{p_z(z) p_{\psi}(x|z)} [\log q_{\lambda}(z|x)]$$

- GAN objective:

$$\mathcal{L}_{\phi} = \mathbb{E}_{p(y) p_{\theta}(x|y)} [\log q_{\phi}(y|x)]$$
$$\mathcal{L}_{\theta} = \mathbb{E}_{p(y) p_{\theta}(x|y)} [\log q^{r}_{\phi}(y|x)]$$

"On Unifying Deep Generative Models" (Hu et al. 2018)



Result 1. *If $p(y)$ is uniformly distributed, then the θ update when $\theta = \theta_0$ using negative gradient is*

$$\nabla_\theta \left[-\mathbb{E}_{p(y)p_\theta(x|y)}[\log q^r_{\phi_0}(y|x)]\right]\Big|_{\theta=\theta_0}$$
$$=\nabla_\theta \left[\mathbb{E}_{p(y)}[KL(p_\theta(x|y)\|q^r(x|y))] - JSD(p_\theta(x|y=0)\|p_\theta(x|y=1))\right]\Big|_{\theta=\theta_0}$$

Insights:

- Shows mathematically that generator's distribution pushed to the distribution of the data
- Optimizing reverse-KL divergence: explains missing-mode phenomena

Wasserstein GANs (Arjovsky et al. 2017)

- Uses Wasserstein (Earth Mover's) distance as loss

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

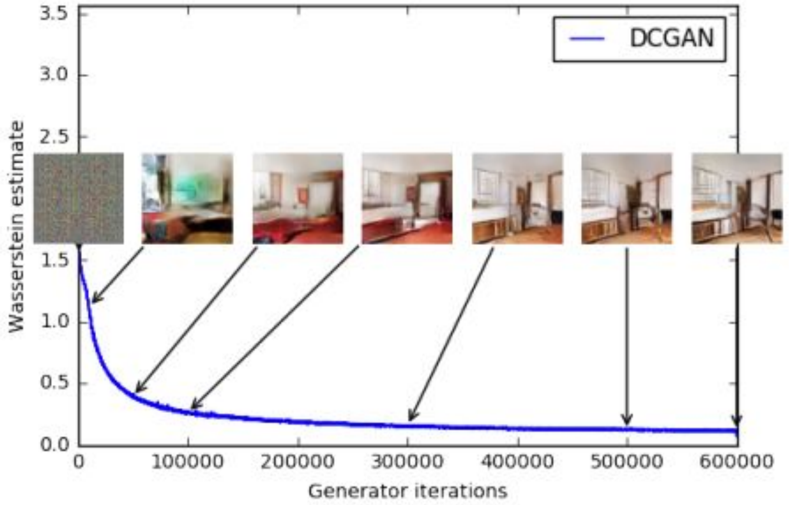


Image Super Resolution

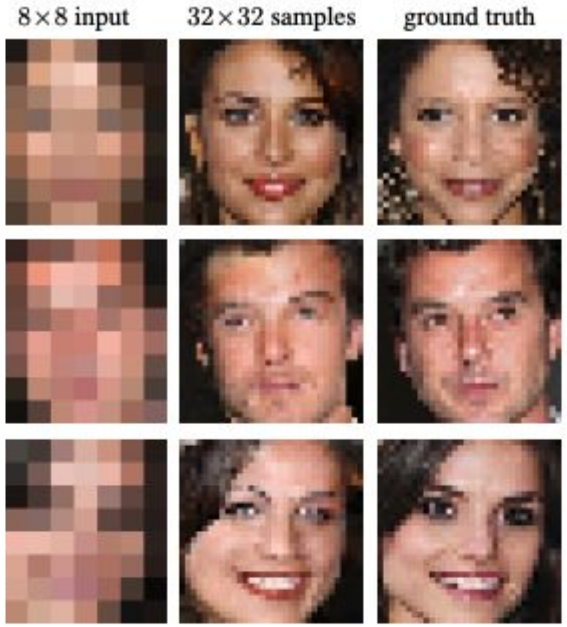


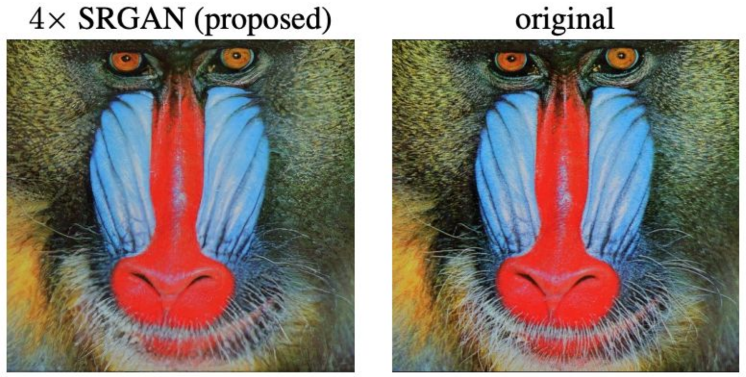
Image from Dahl et al. 2017

Supervised Approaches

- SRGAN (Ledig et al. 2017) - a classic paper
  - “first framework capable of inferring photo-realistic natural images for 4× upscaling factors”
- Utilizes both adversarial loss and content less (using VGG19)

$$l_{Gen}^{SR} = \sum_{n=1}^{N} - \log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

$$l_{VGG/i.j}^{SR} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$



Unsupervised Approaches

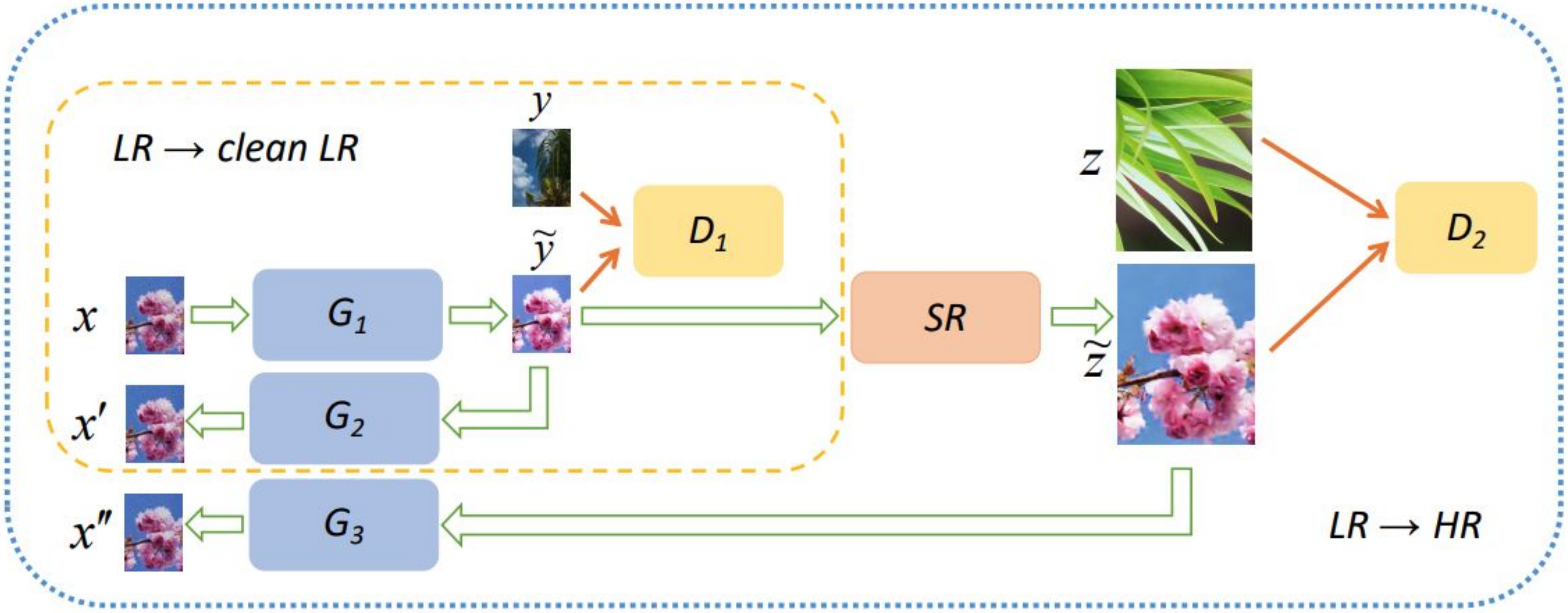


Image from Yuan et al. 2018

Convergence of Overparametrized Neural Networks

George Cai   Irene Li

Overview

- Convergence of training error [*A Convergence Theory for Deep Learning via Over-Parametrization*]
    - Main Theorem - Convergence of SGD
    - Important Lemmas - Convex Geometry of Loss Landscape

- Generalization/Performance on Test set [*Learning and Generalization in Overparametrized Neural Networks, Going Beyond Two Layers*]

Convergence of SGD

Main Theorem

For overparametrized ($m \geq \text{poly}(n, L, \delta^{-1})$) feed-forward ReLU networks
If data is non-degenerate ($\|x_i\|_2 = 1$ and $\|x_i - x_j\|_2 \geq \delta$)

Then SGD finds training global minima (up to error ϵ) in

$$T = \frac{\text{poly}(n, L) \cdot \log^2 m}{\delta^2} \cdot \log \frac{1}{\epsilon}$$

Iterations for ℓ_2-regression

*: n=number of training samples,
m=number of hidden neurons
L=number of layers

Convergence of SGD

Main Theorem

For overparametrized ($m \geq \text{poly}(n, L, \delta^{-1})$) feed-forward ReLU networks
If data is non-degenerate ($\|x_i\|_2 = 1$ and $\|x_i - x_j\|_2 \geq \delta$)

Then SGD finds training global minima (up to error ϵ) in

$$T = \frac{\text{poly}(n, L) \cdot \log^2 m}{\delta^2} \cdot \log \frac{1}{\epsilon}$$

Iterations for ℓ_2-regression.

The polynomial dependence on L is possible because ReLU kills half of the neurons, thus preventing exponential gradient explosion/vanishing

*: n=number of training samples,
m=number of hidden neurons
L=number of layers

Convergence of SGD

Main Theorem

For overparametrized ($m \geq \text{poly}(n, L, \delta^{-1})$) feed-forward ReLU networks
If data is non-degenerate ($\|x_i\|_2 = 1$ and $\|x_i - x_j\|_2 \geq \delta$)

Then SGD f

Iterations fo

In paper, the result is generalized to more scenarios:

1. Neural networks with different number of neurons per layer;
2. Other smooth losses such as cross entropy loss;
3. Other architectures such as ResNet, CNN, etc.

*: n=number of training samples,
m=number of hidden neurons
L=number of layers

Important Lemmas: Convex Geometry

Lemma 1 (no critical point). If loss is large, then the gradient norm is also large.

$$\|\nabla F(\vec{\mathbf{W}})\|_F^2 \geq \Omega(F(\vec{\mathbf{W}}) \cdot \frac{\delta m}{dn^2})$$

Important Lemmas: Convex Geometry

Lemma 1 (no critical point). If loss is large, then the gradient norm is also large.

$$\|\nabla F(\vec{\mathbf{W}})\|_F^2 \geq \Omega(F(\vec{\mathbf{W}}) \cdot \frac{\delta m}{dn^2})$$

Lemma 2 (semi-smoothness). The objective is semi-smooth.

$$F(\vec{\mathbf{W}} + \vec{\mathbf{W}}') = F(\vec{\mathbf{W}}) + \langle \nabla F(\vec{\mathbf{W}}), \vec{\mathbf{W}}' \rangle \pm \mathrm{poly}(L, n) \|\vec{\mathbf{W}}'\|_F$$

Important Lemmas: Convex Geometry

Lemma 1 (no critical point). If loss is large, then the gradient norm is also large.

$$\|\nabla F(\vec{\mathbf{W}})\|_F^2 \geq \Omega(F(\vec{\mathbf{W}}) \cdot \frac{\delta m}{dn^2})$$

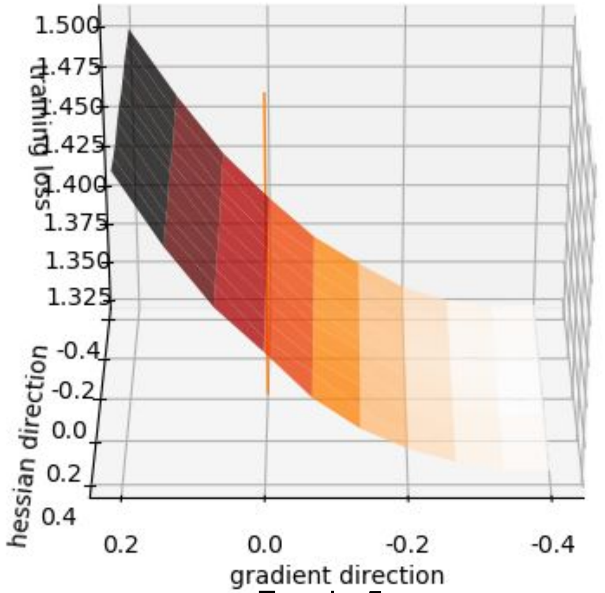
Lemma 2 (semi-smoothness). The objective is semi-smooth.

$$F(\vec{\mathbf{W}} + \vec{\mathbf{W}}') = F(\vec{\mathbf{W}}) + \langle \nabla F(\vec{\mathbf{W}}), \vec{\mathbf{W}}' \rangle \pm \mathrm{poly}(L, n) \|\vec{\mathbf{W}}'\|_F$$

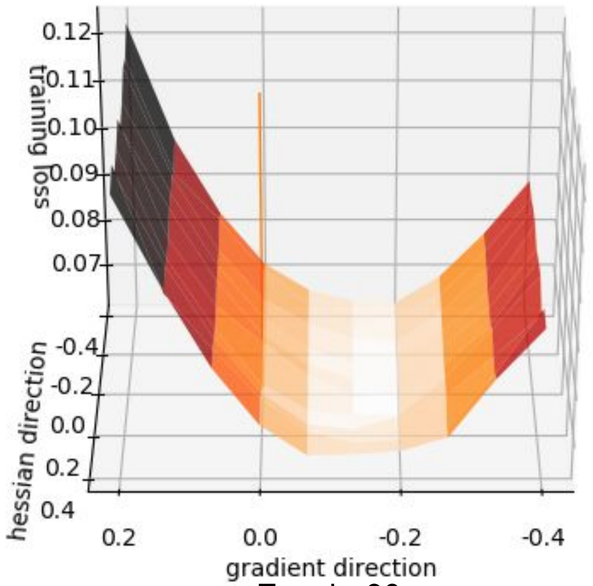
If we apply the above semi-smoothness and gradient bound on $\mathbb{E}[F(\vec{\mathrm{W}}^{(t+1)})]$ for every iteration t, we can derive the main theorem.

Important Lemmas: Convex Geometry

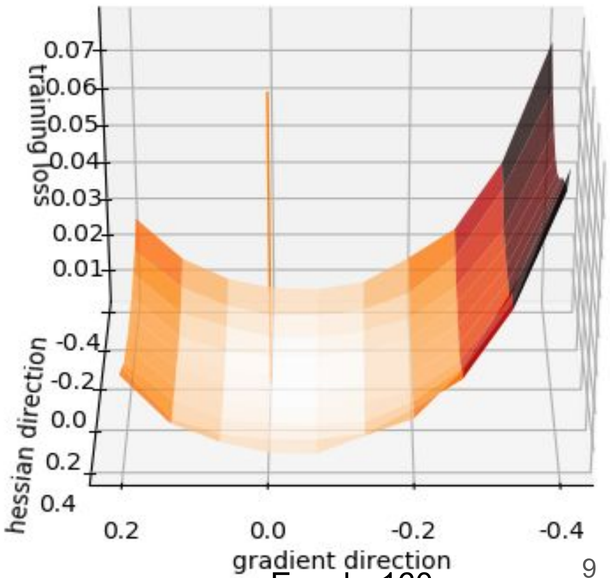
VGG19_bn trained on CIFAR10 with SGD training trajectory.
The orange vertical sticks represents parameter at current iteration.
The loss landscape is plotted along gradient direction and the most negative (least convex) hessian eigenvector direction.



Epoch=5



Epoch=90



Epoch=160

Generalization: 2-Layer/3-Layer Neural Networks

Main Result

Some notable target functions can be efficiently learned by 3-layer/2-layer ReLU neural networks using polynomially many samples.

Generalization: 2-Layer/3-Layer Neural Networks

Main Result

Some notable target functions can be efficiently learned by 3-layer/2-layer ReLU neural networks using polynomially many samples.

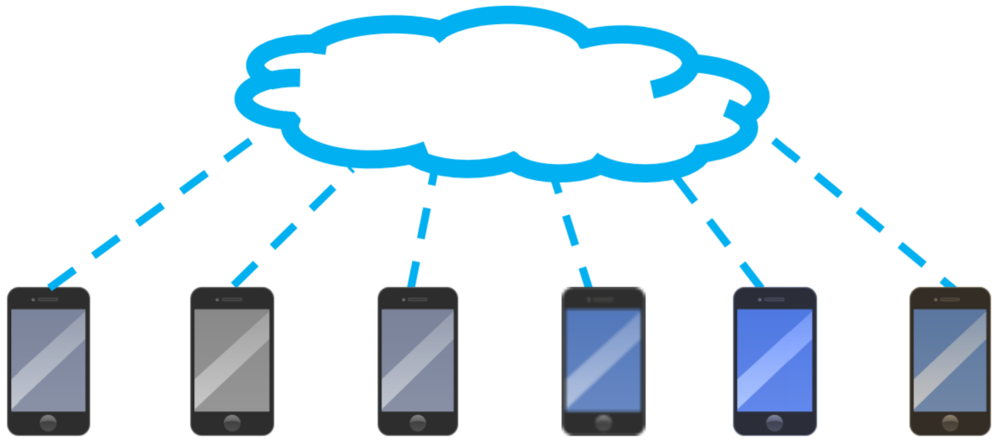
Target functions that contain 3-layer (resp. 2-layer) neural networks with smooth activations can be efficiently learned up to additive error ϵ by 3-layer (resp. 2-layer) ReLU neural networks of size greater than a fixed polynomial in 1) the size of the target network; 2) $1/\epsilon$; 3) ‘complexity’ of the activation function in the target function using polynomially many samples.

Fairness in Federated Learning

Srinivasa Pranav

Federated Learning

- Example: next-word prediction
- p devices connected to the cloud
- Each device k has m_k training samples from a different distribution D_k
- Each device trains a neural network h_k locally and sends neural network parameters
- Cloud combines neural network parameters to minimize loss over mixture of distributions D_k
- Cloud forces devices to use its parameters



$$\overline{\mathcal{U}} = \sum_{k=1}^{p} \frac{m_k}{\sum_{k=1}^{p} m_k} \mathcal{D}_k$$

Picture from: Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2019). Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873.*

Agnostic Federated Learning (AFL)

- We don’t really know the mixture of distributions
- Replace specific mixture of distributions with arbitrary convex combination of distributions
- *Agnostic* to the true mixture weights
- *Good-intent Fairness*

$$\overline{\mathcal{U}} = \sum_{k=1}^{p} \frac{m_k}{\sum_{k=1}^{p} m_k} \mathcal{D}_k$$

$$\Downarrow$$

$$\mathcal{D}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{D}_k$$

$$h_{\mathcal{D}_\Lambda} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \max_{\lambda \in \Lambda} \mathcal{L}_{\mathcal{D}_\lambda}(h)$$

Mohri, M., Sivek, G. Suresh, A.T.. (2019). Agnostic Federated Learning. Proceedings of the 36th International Conference on Machine Learning, in PMLR 97:4615-4625

q-Fair Federated Learning (q-FFL)

- AFL’s minimax-like notion of fairness may be too strict
- Inspired by α-fairness in wireless networks
- At each time step dynamically reweight empirical loss (changing objective function)
- More complicated parameter updates
- Trades off performance for fairness

$\mathcal{D}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{D}_k \quad \mathcal{L}_{\mathcal{D}_\lambda}(h)$

$$\min_h \sum_{k=1}^{p} \lambda_k \mathcal{L}_{\mathcal{D}_k}^{q+1}(h)$$

Li, T., Sanjabi, M., Smith, V. (2019). Fair resource allocation in federated learning. arXiv preprint arXiv:1905.10497

Analysis of Orthogonal Matching Pursuit for Sparse Linear Regression

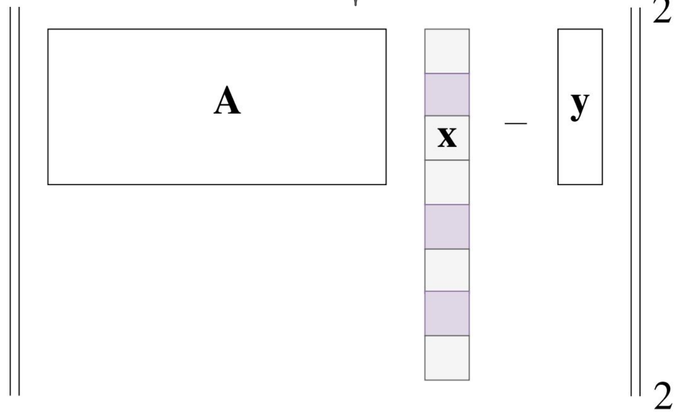
10-716 Spring 2020 Course Project

Vineet Jain, Chirag Pabbaraju

Sparse Linear Regression (SLR)

Model: $\mathbf{y} = \mathbf{A}\bar{x} + \boldsymbol{\eta}$

Objective: $\min_{\mathbf{x}\in\mathbb{R}^d} \underbrace{\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2}$ subject to $\|\mathbf{x}\|_0 \leq s^*$



Applications: Compressed sensing, bioinformatics, etc.

Goals in SLR

1. Support Recovery:

   Find $\hat{\mathbf{x}}$ such that $\mathbf{S}^* \subseteq supp(\hat{\mathbf{x}})$ and $\|\hat{\mathbf{x}} - \bar{\mathbf{x}}\|_\infty$ is small

2. Generalization Error:

   Find $\hat{\mathbf{x}}$ such that $\frac{1}{n}\|\mathbf{A}(\hat{\mathbf{x}} - \bar{\mathbf{x}})\|_2$ is small

Problem Setting

- Restricted Strong Convexity (RSC):

$$\|\mathbf{A}\boldsymbol{\Delta}\|_2^2 \geq \rho_s^- \|\boldsymbol{\Delta}\|_2^2 \; \forall \; \|\boldsymbol{\Delta}\|_0 \leq \boldsymbol{s}$$

- Restricted Smoothness (RSS):

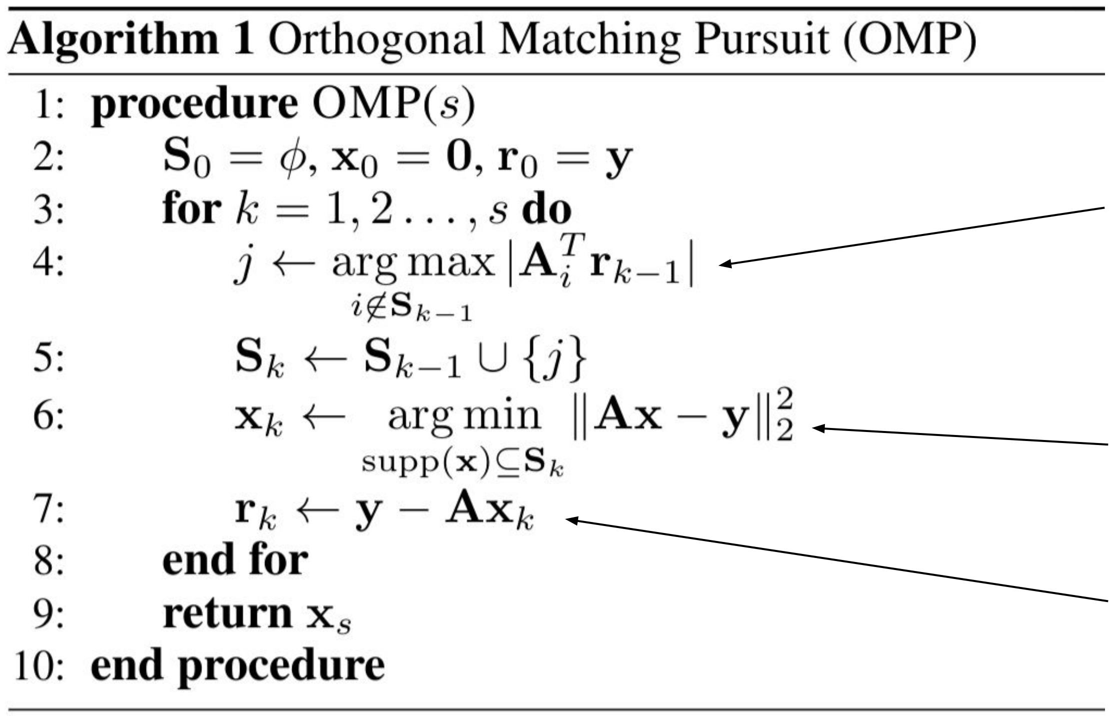
$$\|\mathbf{A}\boldsymbol{\Delta}\|_2^2 \leq \rho_s^+ \|\boldsymbol{\Delta}\|_2^2 \; \forall \; \|\boldsymbol{\Delta}\|_0 \leq \boldsymbol{s}$$

- Restricted Condition Number:

$$\tilde{\boldsymbol{\kappa}}_s = \frac{\rho_s^+}{\rho_s^-}$$

- The noise $\boldsymbol{\eta}$ is sub-Gaussian with parameter σ^2

Orthogonal Matching Pursuit



Algorithm 1 Orthogonal Matching Pursuit (OMP)

```
1:  procedure OMP($s$)
2:      $\mathbf{S}_0 = \phi, \mathbf{x}_0 = \mathbf{0}, \mathbf{r}_0 = \mathbf{y}$
3:      for $k = 1, 2 \ldots, s$ do
4:          $j \leftarrow \arg\max_{i \notin \mathbf{S}_{k-1}} |\mathbf{A}_i^T \mathbf{r}_{k-1}|$
5:          $\mathbf{S}_k \leftarrow \mathbf{S}_{k-1} \cup \{j\}$
6:          $\mathbf{x}_k \leftarrow \arg\min_{\text{supp}(\mathbf{x}) \subseteq \mathbf{S}_k} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$
7:          $\mathbf{r}_k \leftarrow \mathbf{y} - \mathbf{A}\mathbf{x}_k$
8:      end for
9:      return $\mathbf{x}_s$
10: end procedure
```

Extract column of $\mathbf{A}$ that has maximum correlation with residual $\mathbf{r_{k-1}}$

Solve least squares problem with support $\mathbf{S_k}$

Update residual

Upper Bound: Support Recovery

For the problem setting described before, say we run OMP for s iterations, then if:

1. $|\bar{x}_{min}| \simeq \sigma/\sqrt{n}$
2. $s \simeq s^* \tilde{\kappa}_{s+s^*} \log \tilde{\kappa}_{s+s^*}$

Then we have full support recovery, that is $\mathbf{S}^* \subseteq supp(\hat{\mathbf{x}})$ and $\|\hat{\mathbf{x}} - \bar{\mathbf{x}}\|_\infty$ is small with high probability

Upper Bound: Generalization Error

For the setting described before, let $\hat{\mathbf{x}}_s$ be the output of OMP after s iterations, then if $s \simeq s^* \tilde{\kappa}_{s+s^*} \log \tilde{\kappa}_{s+s^*}$,

$$\frac{1}{n} \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{A}\bar{\mathbf{x}}\|_2 \leq \sigma^2 s^* \tilde{\kappa}_{s+s^*} \log \tilde{\kappa}_{s+s^*}$$

with high probability

Upper Bound: Key idea

$$f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$$

- If *any* support $\mathbf{S}^*$ is unrecovered, then there is a large additive decrease in objective
- $f(\mathbf{x}) \geq 0 \implies$ support recovery will happen soon
- Recovery will small support $\implies$ small generalization error

Lower bound

There exists a matrix $\mathbf{M}$ such that:

Noiseless Case: OMP has to indeed run for d iterations to recover support completely

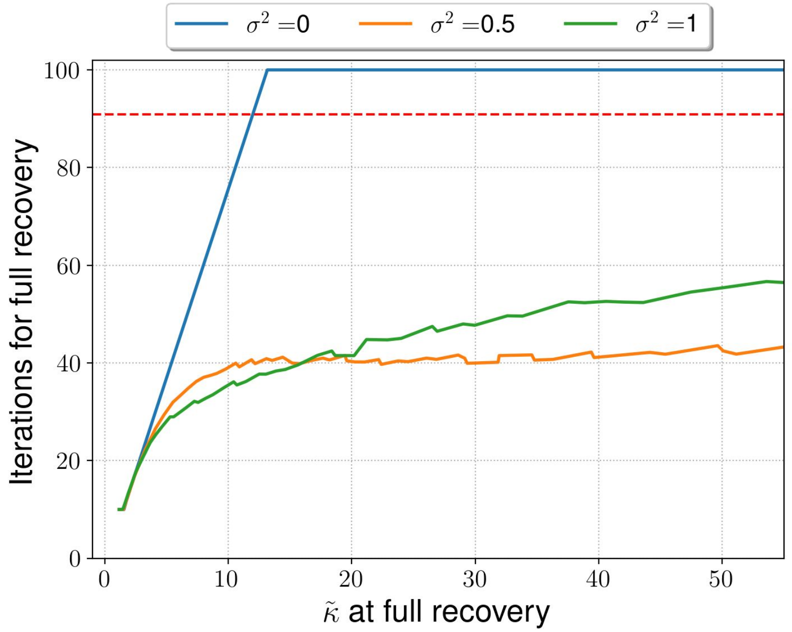
Noisy Case: Even if OMP is run for $s^* \tilde{\kappa}_{s+s^*}$ iterations:

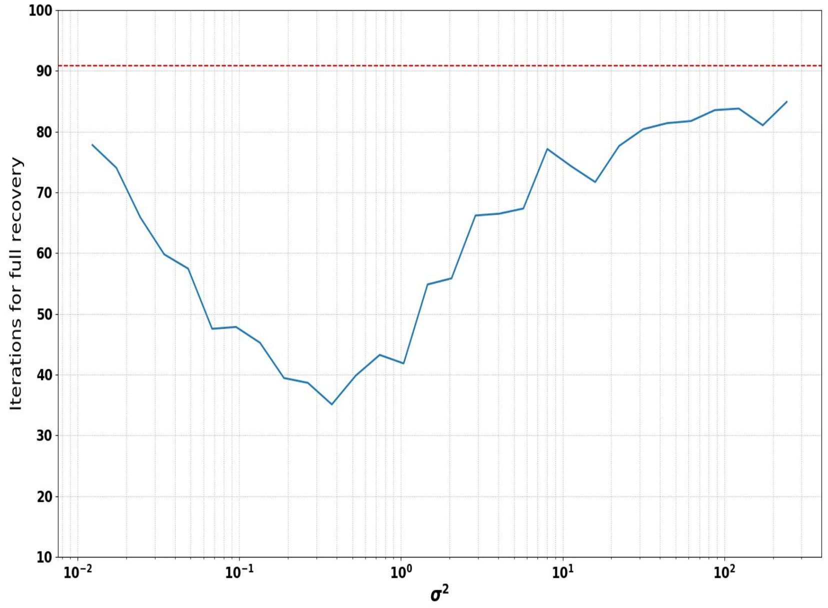
1) The support $\hat{\mathbf{x}}_s$ of OMP is disjoint from $\mathbf{S}^*$
2) The generalization error of OMP is large i.e. on the same order as that in upper bound (upto log factors)

Lower bound: Key idea

- The matrix $\mathbf{M}$ is constructed to have columns that are average of the columns corresponding to $\mathbf{S}^*$ with some additional noise
- Since OMP is a greedy algorithm, it gets fooled in choosing coordinates corresponding to these columns instead

Experiments





Thank you

10-716 Final Project: Influence and Normalizing Flow

Gopaljee Atulya (gatulya)
Sungjun Choi (sungjun2)

Normalizing Flow Models

Generative models where NLL can be minimized directly. Similar to GAN and VAE

Mold Priors/Proxy variables into target distribution using invertible transformations

Computation of Jacobian term is expensive, hence lot of research into NN architectures that allow for faster computation of jacobian.

Flow models have been proved to approximate any target density provided sufficient number of transformation layers.

Change of Variable with Function Composition

$$p_x(x) = p_u\Big(f^{-1}(x)\Big)\Big|J(f^{-1}(x))\Big|$$

$$p_x(x) = p_u\Big(f_1^{-1} \circ f_2^{-1} \cdots \circ \dots f_K^{-1}(x)\Big) \prod_{k=1}^{K} \Big|J_{f_k^{-1}} f_{k+1}^{-1}\Big|$$

$$\log(p_x(x)) = \log\Big(p_u\Big(f_1^{-1} \circ f_2^{-1} \cdots \circ \dots f_K^{-1}(x)\Big)\Big) + \sum_{k=1}^{K} \log\Big(\Big|J_{f_k^{-1}} f_{k+1}^{-1} \circ f_{k+2}^{-1} \cdots \circ \dots f_K^{-1}(x)\Big|\Big)$$

Affine coupling layer

Identity transformation on random variables 1:d

$$f(x_{(d+1):D}) = x_{(d+1):D} \odot e^{s(x_{1:d})} + t(x_{1:d})$$

Scaling and Translation on random variables d+1:D

$$f^{-1}(x_{(d+1):D}) = (x_{(d+1):D} - t(x_{1:d})) \odot e^{-s(x_{1:d})}$$

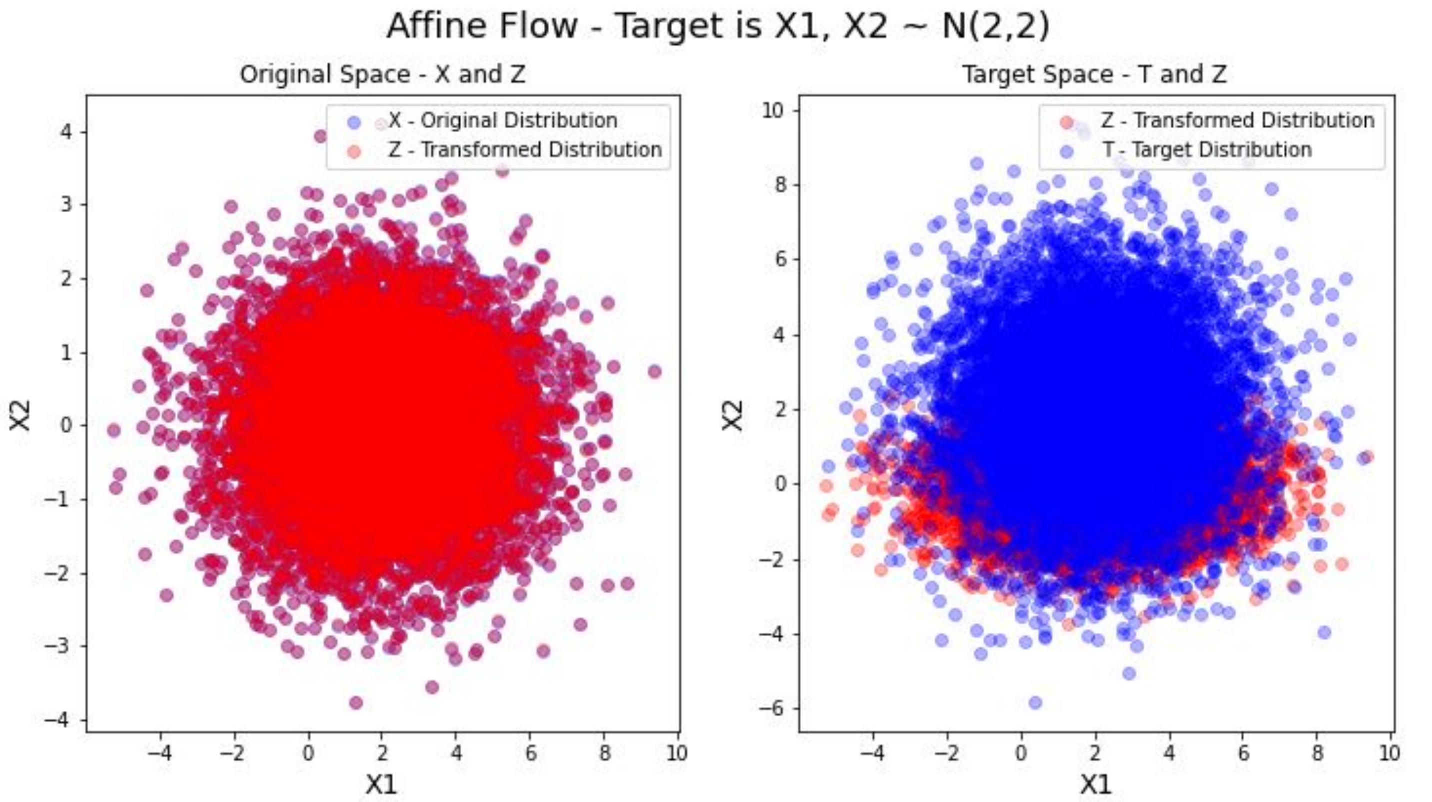
Invert Forward Transformations

$$p_x(x) = p_u\Big(f^{-1}(x)\Big)\Big|J(f^{-1}(x))\Big|$$

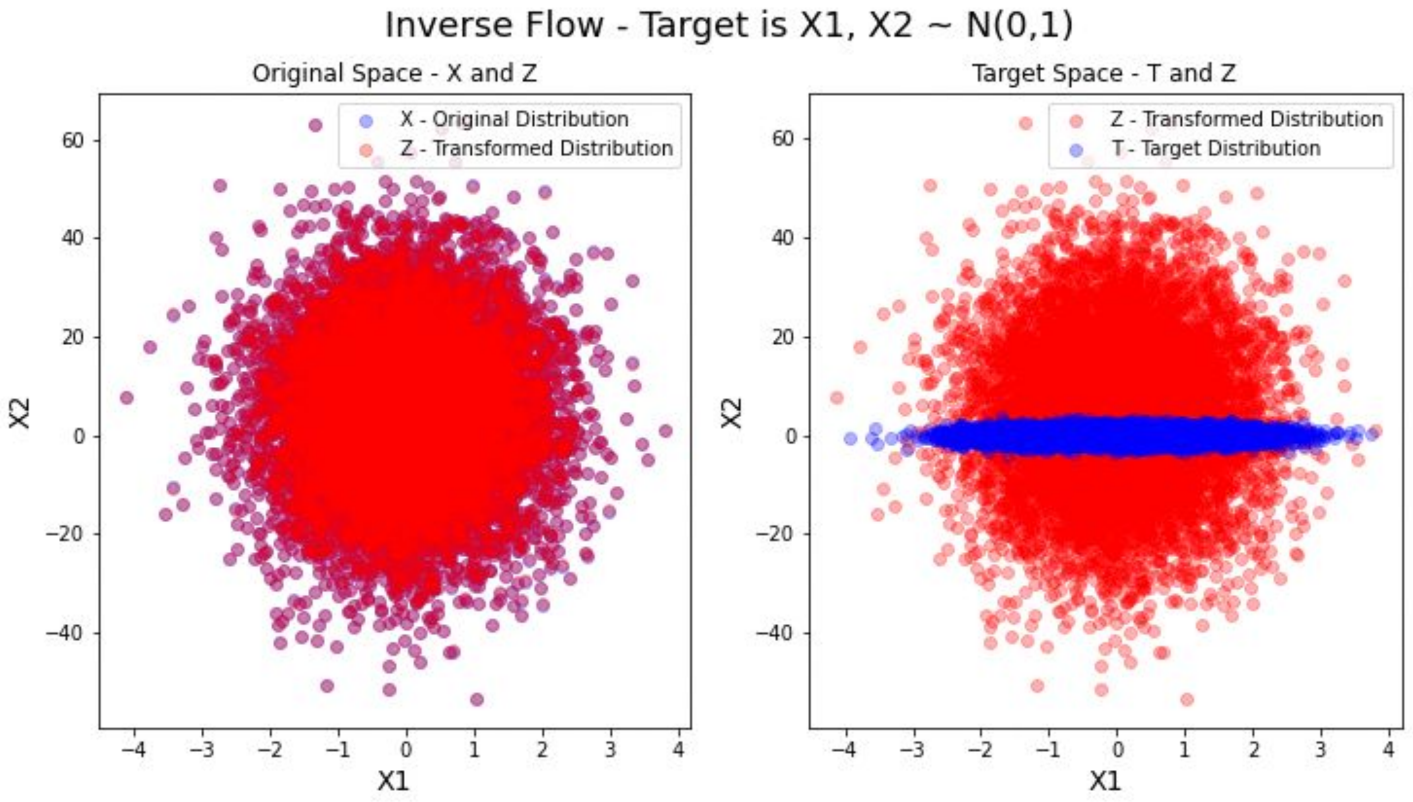
Fast Jacobian Determinant

$$\begin{bmatrix} \frac{\partial f^{-1}_{1:d}(x)}{\partial x_{1:d}} & \frac{\partial f^{-1}_{1:d}(x)}{\partial x_{(d+1):D}} \\ \frac{\partial f^{-1}_{(d+1):D}(x)}{\partial x_{1:d}} & \frac{\partial f^{-1}_{(d+1):D}(x)}{\partial x_{(d+1):D}} \end{bmatrix} = \begin{bmatrix} \mathbb{I} & 0 \\ \frac{\partial f^{-1}_{(d+1):D}(x)}{\partial x_{1:d}} & \operatorname{diag}(e^{-s(x_{1:d})}) \end{bmatrix}$$

Result: Affine Flow - Single Coupling Layer



Result: Inverse Flow - Single Coupling Layer



Influence Functions

Tool from semi-parametric statistics for analyzing robustness of models (understanding how "influential" each training point is to the model)

Consider model parameters and test loss as functionals of data distribution

e.g. with data distribution *F*:

Mean: $T(F) = \int x \, dF(x)$

Variance: $T(F) = \int (x - \mu)^2 \, dF(x)$

Influence Functions

To consider the “influence” of a single training point x on parameters or the loss, consider upweighting the point in the data distribution by small ε:

$$L(x) = \lim_{\epsilon \to 0} \left[\frac{T\{(1-\epsilon)F + \epsilon\delta_x\} - T(F)}{\epsilon} \right]$$

where

- F: data distribution, δ_x : Dirac-delta distribution at x
- T: parameter functional

Influence Functions

It is known that influence functions can be linearly approximated as follows

$$\mathcal{I}_{\text{up,params}}(z) \overset{\text{def}}{=} \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\Big|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\,\nabla_\theta L(z,\hat{\theta})$$

Applying chain rule to Gâteaux derivative allows chaining to other quantities, specifically the test loss:

$$\begin{aligned}\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) &\overset{\text{def}}{=} \frac{dL(z_{\text{test}}, \hat{\theta}_{\epsilon,z})}{d\epsilon}\Big|_{\epsilon=0} \\ &= \nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\Big|_{\epsilon=0} \\ &= -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})\end{aligned}$$

(Koh & Liang, 2017)

Analysis with Influence Functions

Many applications introduced in (Koh & Liang, 2017):

- Analyzing model behavior for individual test points
- Generating adversarial test examples
- Fixing mislabeled examples

We aim to use influence functions to analyze normalizing flows in different parts of the support and use the information to guide further training

Discussion and Potential for Novel Contribution

NF models have an Expressiveness vs Computation trade-off.

NF models prefer modelling high density regions.

Influence functions can be used to select trainings sample to balance density estimation.

References

Kingma, Durk P., and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." *Advances in Neural Information Processing Systems*. 2018.

Huang, Chin-Wei, et al. "Neural autoregressive flows." *arXiv preprint arXiv:1804.00779* (2018).

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." *arXiv preprint arXiv:1605.08803* (2016).

Cook, R. Dennis, and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

Agarwal, Naman, Brian Bullins, and Elad Hazan. "Second-order stochastic optimization for machine learning in linear time." *The Journal of Machine Learning Research* 18.1 (2017): 4148-4187.

Dinh, Laurent, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation." *arXiv preprint arXiv:1410.8516* (2014).

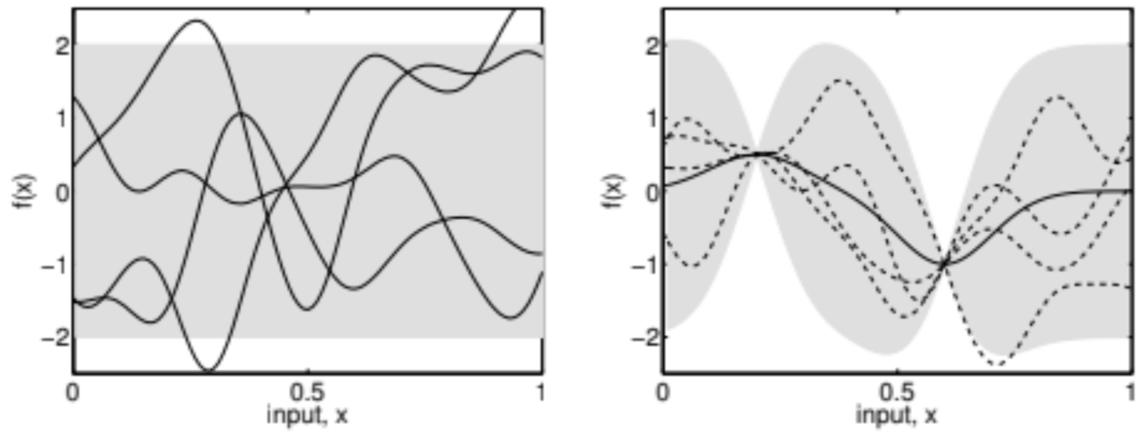
Pang Wei Koh and Percy Liang. “Understanding black-box predictions via influence functions.” *International Conference on Machine Learning*. 2017.

Gaussian Processes for Bayesian Inference

Christopher Kottke

Cathy Su

Gaussian Process models are widely used for Bayesian Learning



(a), prior (b), posterior

- GPs is collection of random variables with a joint Gaussian distribution
- Applied to both classification and regression supervised learning problems
- However, what about adversarial examples?

Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006

Probabilistic local robustness guarantees put bound on GP predictions

- For the GP *z(x)*, introduce new test points *x**
- Probabilistic safety metric:

$$\phi_1^i(x^*, T, \delta|\mathcal{D}) = P(\exists x' \in T \text{s.t.} (z^i(x^*) - z^i(x') > \delta|\mathcal{D})$$

- Probabilistic invariance metric:

$$\phi_2^i(x^*, T, \delta|\mathcal{D}) = P(\exists x' \in T \text{s.t.} \|z^i(x^*) - z^i(x')\|_d > \delta|\mathcal{D})$$

- We want to bound each of these metrics by using the properties of linearity and symmetry of GPs

Cardelli, L., Kwiatkowska, M., Laurenti, L., & Patane, A. (2019). Robustness Guarantees for Bayesian Inference with Gaussian Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(Dudley 1967), 7759–7768. https://doi.org/10.1609/aaai.v33i01.33017759

Theorem 1: showing a tight upper bound on the probabilistic safety metric

$$P(\exists x \in T \, s.t. \left(\mathbf{z}^{(i)}(x^*) - \mathbf{z}^{(i)}(x) > \delta \,|\, \mathcal{D}\right)$$

$$=P\left(\sup_{x \in T} \mathbf{z}^{o,(i)}(x^*, x) > \delta\right)$$

Cardelli, L., Kwiatkowska, M., Laurenti, L., & Patane, A. (2019). Robustness Guarantees for Bayesian Inference with Gaussian Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(Dudley 1967), 7759–7768. https://doi.org/10.1609/aaai.v33i01.33017759

Theorem 1: showing a tight upper bound on the probabilistic safety metric

$$
\begin{aligned}
&P(\exists x \in T\, s.t. \big(\mathbf{z}^{(i)}(x^*) - \mathbf{z}^{(i)}(x) > \delta \,|\, \mathcal{D}\big) \\
=&P\big(\sup_{x \in T} \mathbf{z}^{o,(i)}(x^*, x) > \delta\big) \\
=&P\big(\sup_{x \in T} \hat{\mathbf{z}}^{o,(i)}(x^*, x) + \mathbb{E}[\mathbf{z}^{o,(i)}(x^*, x)] > \delta\big)
\end{aligned}
$$

Cardelli, L., Kwiatkowska, M., Laurenti, L., & Patane, A. (2019). Robustness Guarantees for Bayesian Inference with Gaussian Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(Dudley 1967), 7759–7768. https://doi.org/10.1609/aaai.v33i01.33017759

Theorem 1: showing a tight upper bound on the probabilistic safety metric

$$
\begin{aligned}
&P(\exists x \in T\, s.t. \left(\mathbf{z}^{(i)}(x^*) - \mathbf{z}^{(i)}(x) > \delta \,|\, \mathcal{D}\right) \\
=&P\Big(\sup_{x\in T} \mathbf{z}^{o,(i)}(x^*, x) > \delta\Big) \\
=&P\Big(\sup_{x\in T} \hat{\mathbf{z}}^{o,(i)}(x^*, x) + \mathbb{E}[\mathbf{z}^{o,(i)}(x^*, x)] > \delta\Big) \\
\leq&P\Big(\sup_{x\in T} \hat{\mathbf{z}}^{o,(i)}(x^*, x) > \delta - sup_{x_1\in T}\mathbb{E}[\mathbf{z}^{o,(i)}(x^*, x_1)]\Big)
\end{aligned}
$$

Cardelli, L., Kwiatkowska, M., Laurenti, L., & Patane, A. (2019). Robustness Guarantees for Bayesian Inference with Gaussian Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(Dudley 1967), 7759–7768. https://doi.org/10.1609/aaai.v33i01.33017759

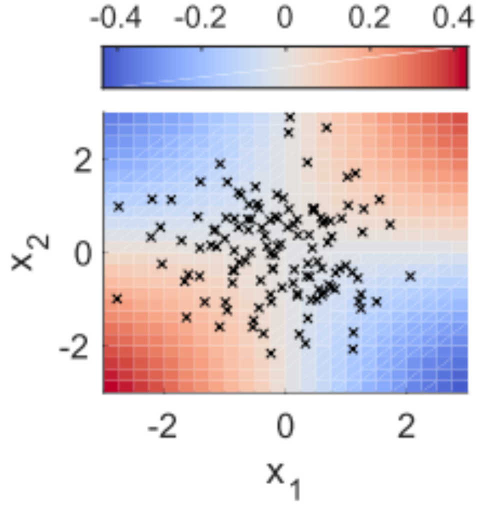
Theorem 1: showing a tight upper bound on the probabilistic safety metric

$$
\begin{aligned}
&P(\exists x \in T\, s.t. \left(\mathbf{z}^{(i)}(x^*) - \mathbf{z}^{(i)}(x) > \delta \,|\, \mathcal{D}\right) \\
=&P\left(\sup_{x \in T} \mathbf{z}^{o,(i)}(x^*, x) > \delta\right) \\
=&P\left(\sup_{x \in T} \hat{\mathbf{z}}^{o,(i)}(x^*, x) + \mathbb{E}[\mathbf{z}^{o,(i)}(x^*, x)] > \delta\right) \\
\leq&P\left(\sup_{x \in T} \hat{\mathbf{z}}^{o,(i)}(x^*, x) > \delta - sup_{x_1 \in T}\mathbb{E}[\mathbf{z}^{o,(i)}(x^*, x_1)]\right)
\end{aligned}
$$

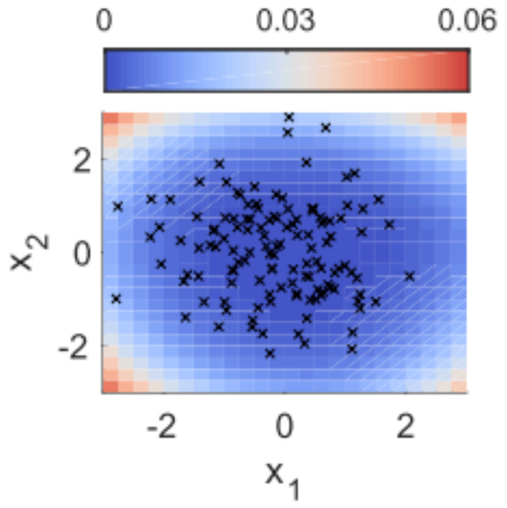
$$P(sup_{x \in T} \hat{\mathbf{z}}(x) > u) \leq e^{\frac{(u - \mathbb{E}[sup_{t \in T}\hat{\mathbf{z}}(x)])^2}{2\sigma_T^2}}$$

Cardelli, L., Kwiatkowska, M., Laurenti, L., & Patane, A. (2019). Robustness Guarantees for Bayesian Inference with Gaussian Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(Dudley 1967), 7759–7768. https://doi.org/10.1609/aaai.v33i01.33017759

Introduced metrics bound the sampled values obtained for 2d GP dataset in simulation



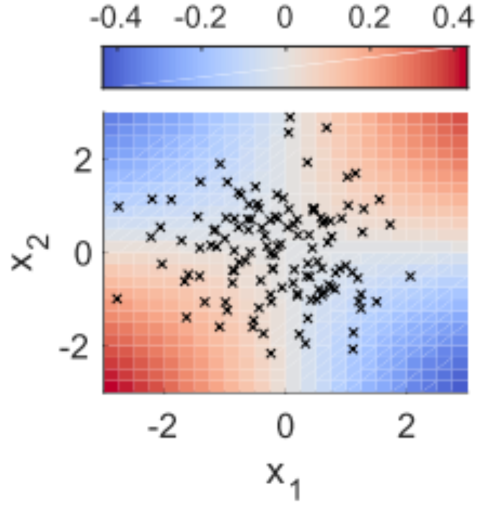
(a) Mean.



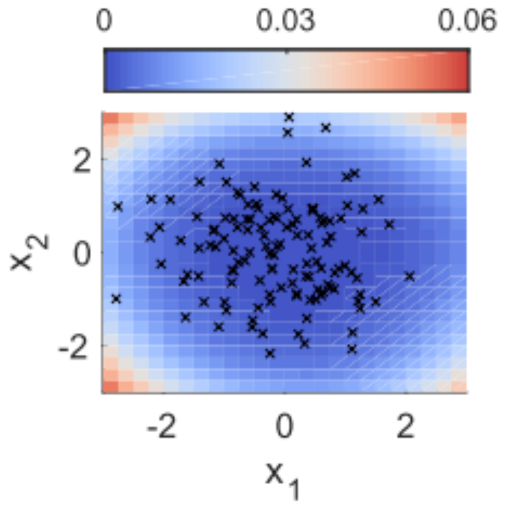
(b) Variance.

Cardelli, L., Kwiatkowska, M., Laurenti, L., & Patane, A. (2019). Robustness Guarantees for Bayesian Inference with Gaussian Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(Dudley 1967), 7759–7768. https://doi.org/10.1609/aaai.v33i01.33017759

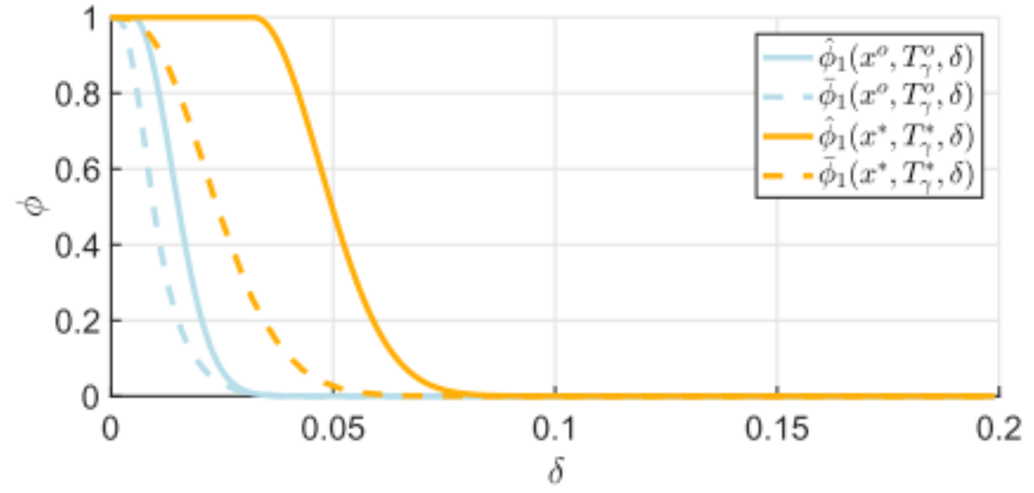
Introduced metrics bound the sampled values obtained for 2d GP dataset in simulation



(a) Mean.

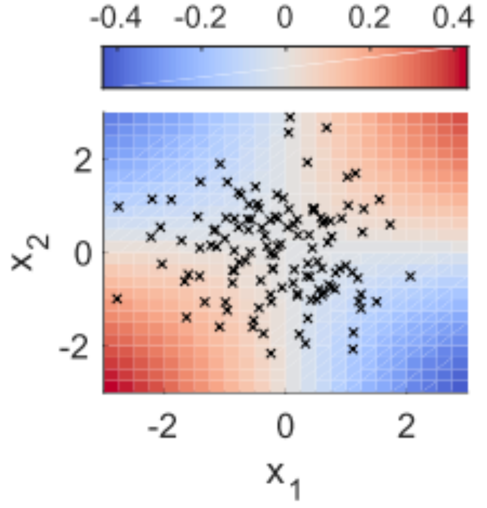


(b) Variance.

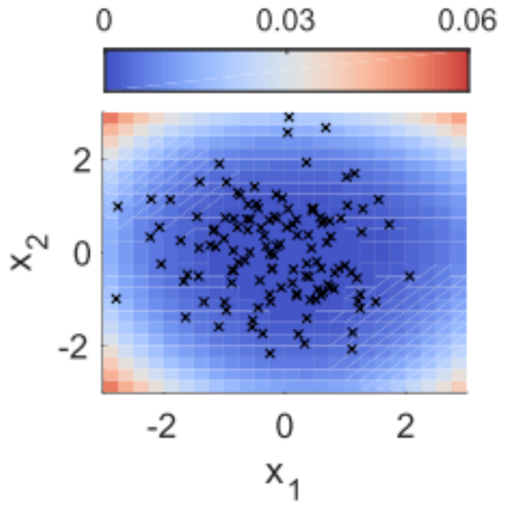


Cardelli, L., Kwiatkowska, M., Laurenti, L., & Patane, A. (2019). Robustness Guarantees for Bayesian Inference with Gaussian Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(Dudley 1967), 7759–7768. https://doi.org/10.1609/aaai.v33i01.33017759

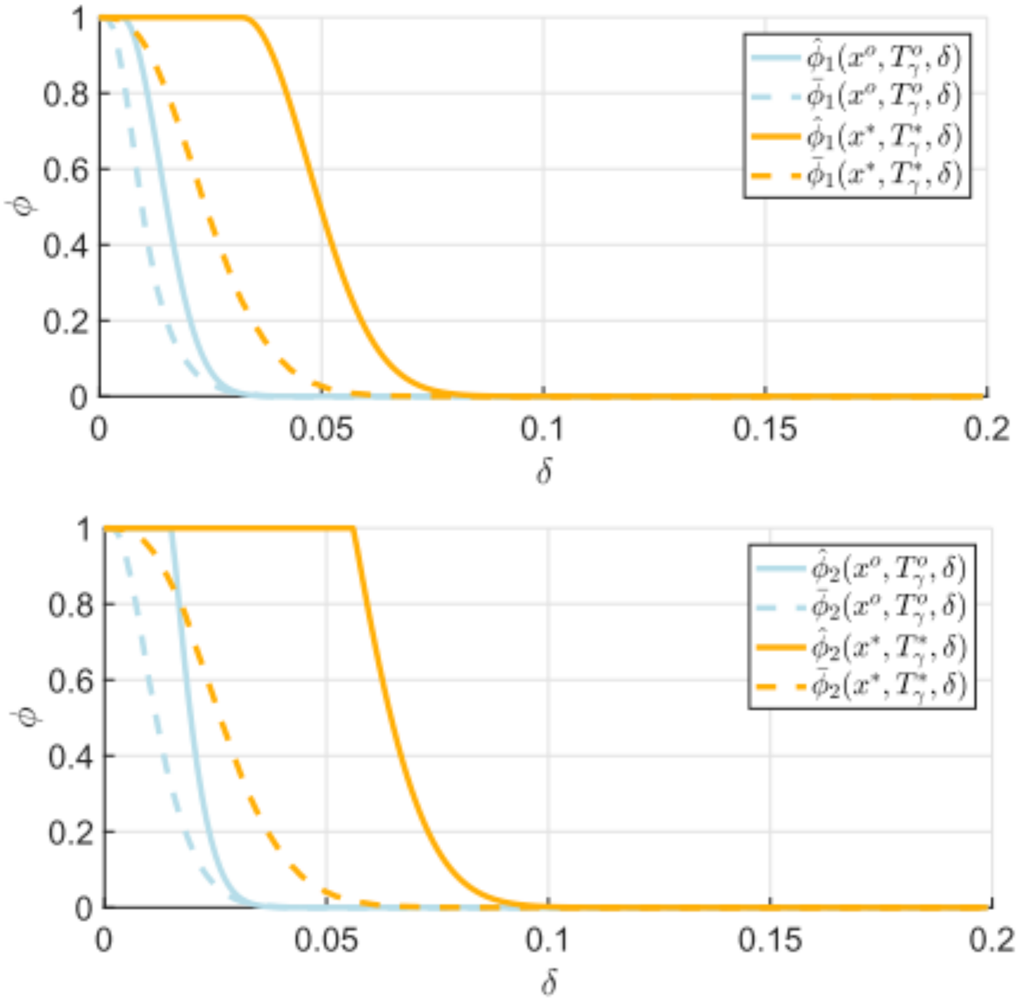
Introduced metrics bound the sampled values obtained for 2d GP dataset in simulation



(a) Mean.



(b) Variance.



Cardelli, L., Kwiatkowska, M., Laurenti, L., & Patane, A. (2019). Robustness Guarantees for Bayesian Inference with Gaussian Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(Dudley 1967), 7759–7768. https://doi.org/10.1609/aaai.v33i01.33017759

Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design

Seeger et. al.

- Using a GP to optimize some outcome often incurs some cost
- How can we select GP kernels and acquisition functions to minimize this cost?

Information Gain

$$x_t = \underset{x \in D}{argmax}\ \sigma_{t-1}(x)$$

Pure Exploration

GP Upper Confidence Bound

$$x_t = \underset{x \in D}{argmax}\ \mu_{t-1}(x) + \sqrt{2\log(\left|D\right| t^2\pi^2/(6\delta))}\sigma_{t-1}(x)$$

Exploration/Exploitation Tradeoff

Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design

Seeger et. al.

Under the GP-UCB acquisition function, we can bound the regret at step T:

$$\Pr\left\{ R_T \leq \sqrt{2C_1 T \log(\left|D\right| t^2\pi^2/(6\delta))\, \gamma_T} \right\} \geq 1 - \delta$$

Specific bounds are achieved for each kernel by upper bounding the information gain at a given step for that kernel

Kernel	**Linear**	**RBF**	**Matern**
Regret Bound			

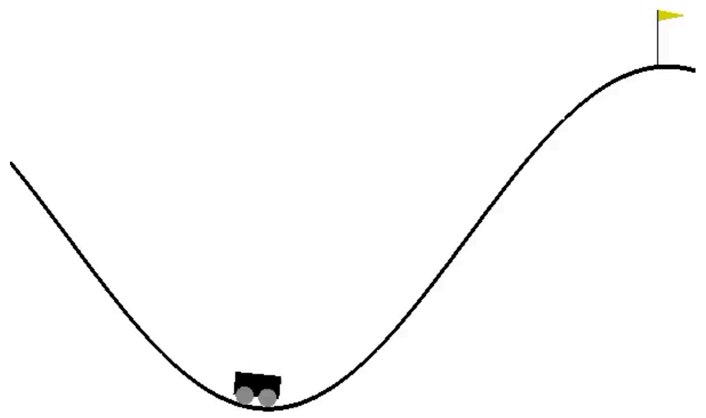
Open AI Mountain Car Test

Using DDQN learning, we optimize training hyperparameters for speed and stability.

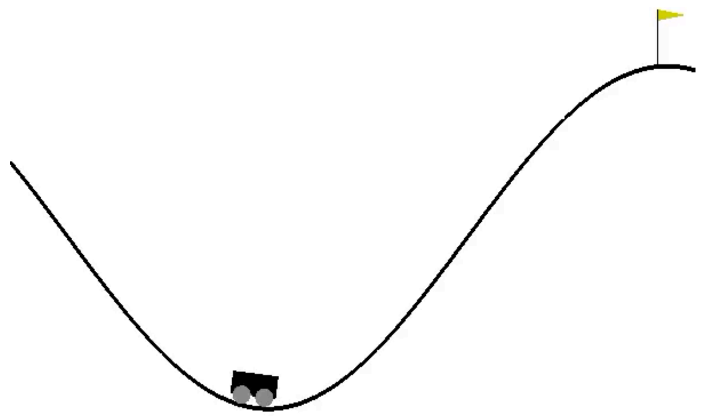
- Learning Rate
- Initial Exploration Rate
- Final Exploration Rate
- Exploration Rate Decay
- TD Discount Factor
- Target Network Update Frequency

Compared the following acquisition functions

- Expected Improvement
- Information Gain
- GP Upper Confidence Bound



Untrained



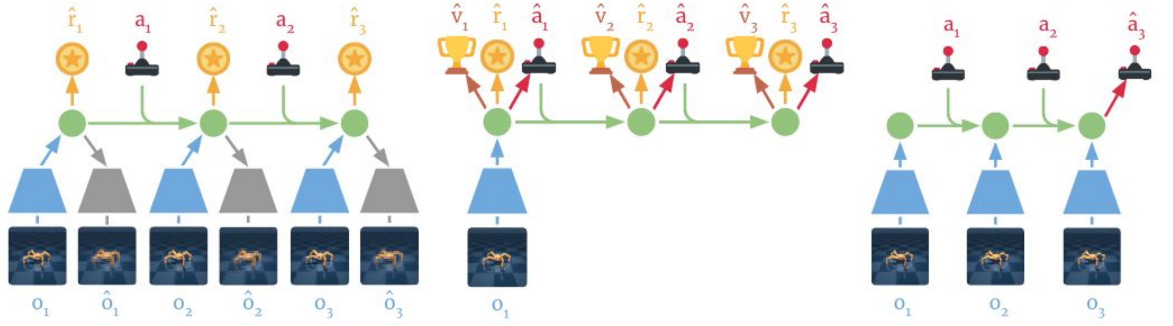
Solved

Sequential Normalizing Flows for Model-Based Reinforcement Learning

Raunaq Bhirangi, Ben Freed

Sequential Normalizing Flows for MBRL

- Goal: Model learning for an autonomous agent to predict future observations and rewards conditioned on past observations, rewards, actions, and past & future actions.
- Normalizing flow enables exact inference with complex, multi-modal distributions, while RNN captures temporal dynamics.

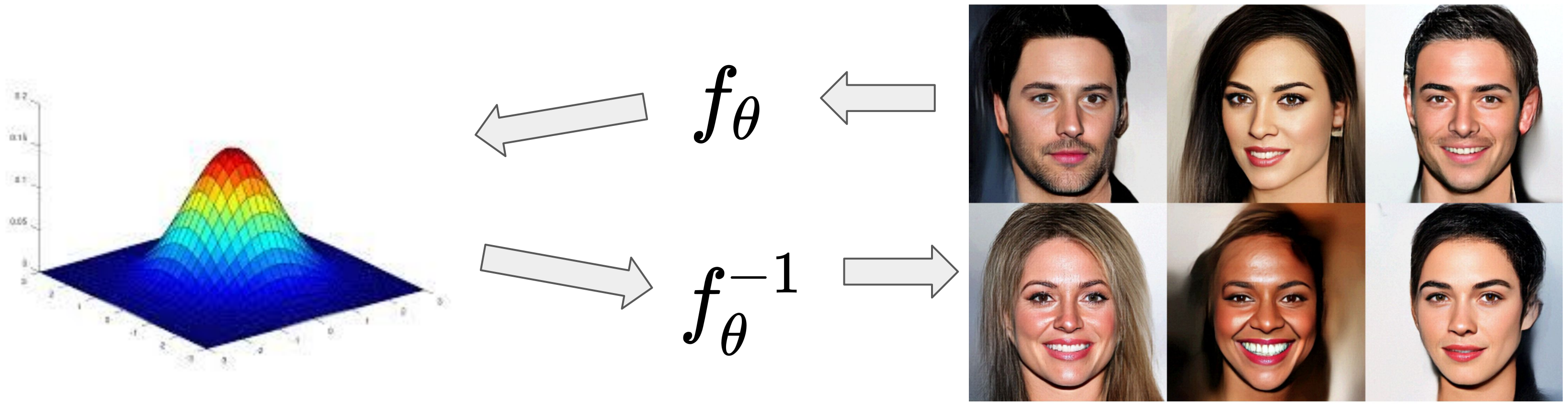


(a) Learn dynamics from experience (b) Learn behavior in imagination (c) Act in the environment

Hafner et. al., 2020

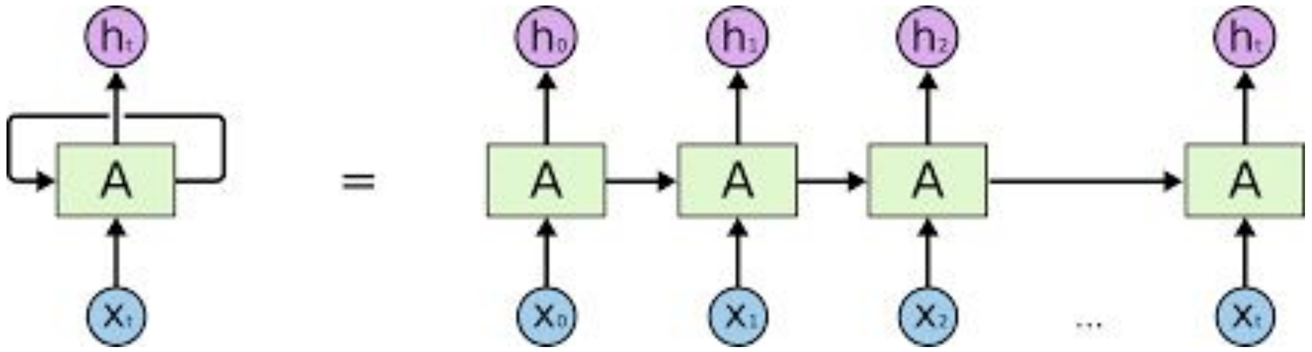
Normalizing Flows

- Enable sampling from complex distributions by mapping samples from a simple distribution through a complex invertible mapping

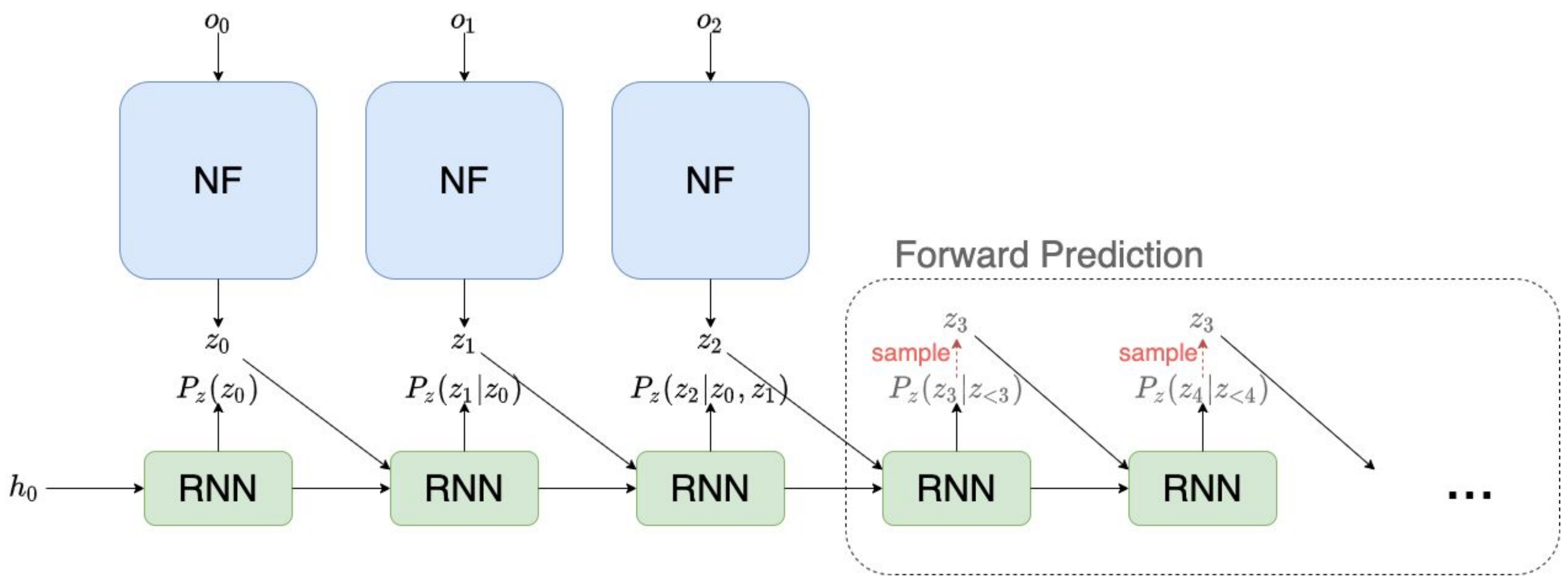


RNNs

- Output conditioned on previous sequence of inputs by via conditioning on a hidden state

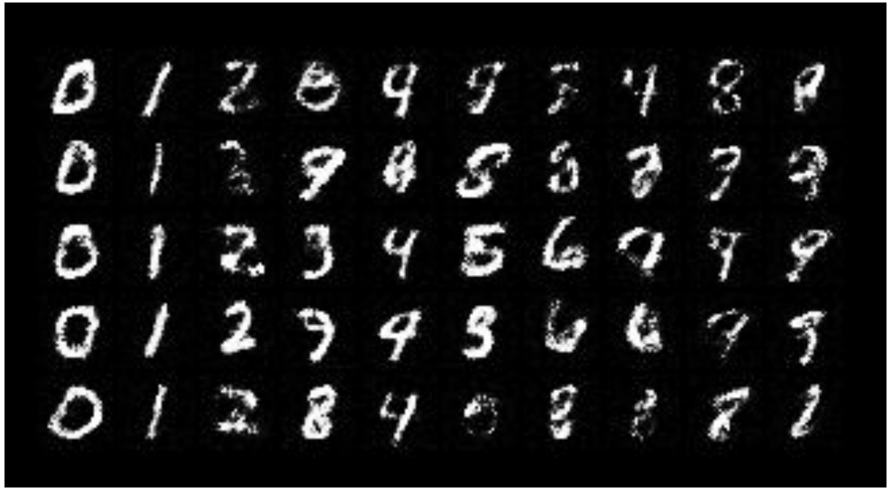


Sequential Normalizing Flows



Results on Sequential MNIST

- Task: generate sequence of MNIST digits [0,1,...,9].



Future Work

- Incorporate reward prediction and action conditioning
- Incorporate policy optimization pipeline that uses trained model to derive a policy that optimizes predicted future rewards

Forecasting seasonals and non linear shared trends with ESNN

Cristian Challu

Background

- Deep Learning applications on time series forecasting have been limited.
- On the 2018 M4 forecasting competition [1] the first place was a novel multivariate hybrid ML-time series model called

Exponential Smoothing Recurrent Neural Network (ESRNN)

[1] Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos. "The M4 Competition: Results, findings, conclusion and way forward." International Journal of Forecasting 34.4 (2018): 802-808.

ESRNN model

The hybrid model learns a **shared function trend for the time series.** This makes the assumption that this information is common across series and estimating them using all the series is beneficial [2].

$$\begin{array}{ll} \textit{Level} & l_t = \alpha(y_t/s_t) + (1-\alpha)l_{t-1} \\ \textit{Seasonal} & s_{t+m} = \gamma(y_t/l_t) + (1-\gamma)s_t \\ & \\ \textit{Prediction} & \hat{Y}_{t+h} = \boxed{RNN(X_t)} * l_t * s_{t+1 \ldots t+h} \end{array}$$

Holt-Winters

Trends

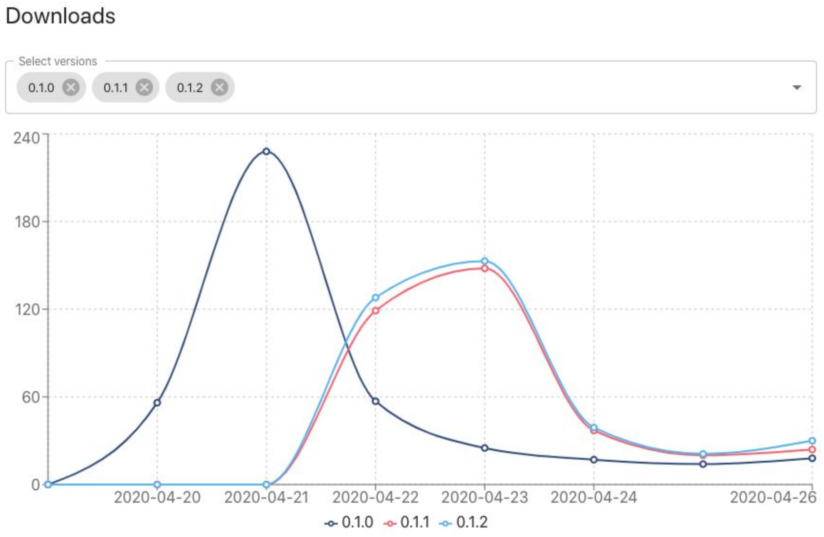
[2] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 competition:100,000 time series and 61 forecasting methods. International Journal of Forecasting, 2019.

Goals of this project

1. Implement the ESRNN model in PyTorch.
2. Extend the model with different ML components.
3. Study the reason and conditions for the success of the ESRNN model.

Implementation

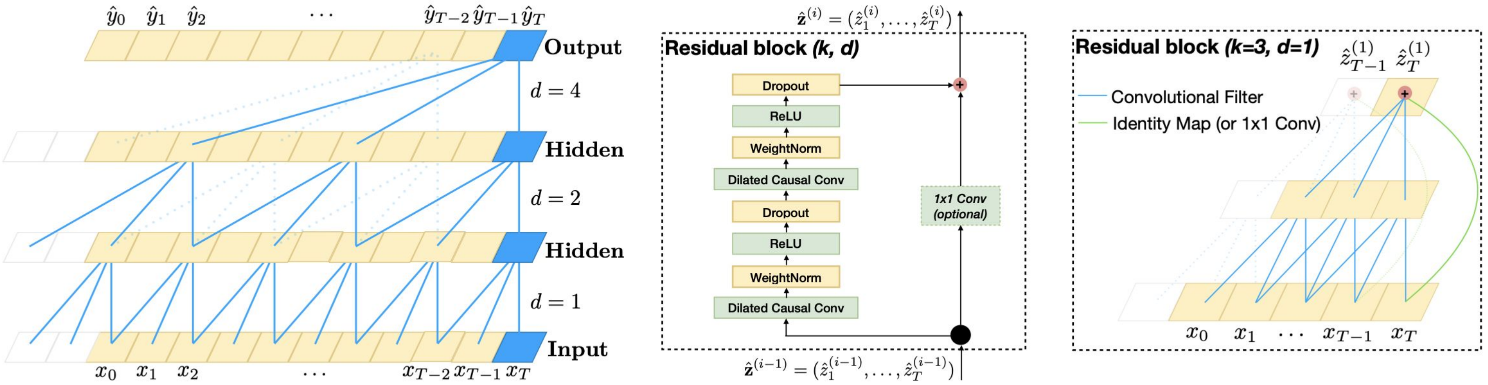
- First public ESRNN implementation in PyTorch.
    - Sklearn fashion, has general fit and predict methods.
    - Released in PyPI on 04/20, with 1,100+ downloads so far.
    - 300x speedup vs original C++ implementation (Dynet)



Implementation

- I extended the model with Temporal Convolutional Neural networks to the:

Exponential Smoothing Convolutional Neural Network (ESCNN)



$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ $\dots$ $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 $\dots$ x_{T-2} x_{T-1} x_T
$\hat{\mathbf{z}}^{(i)} = (\hat{z}_1^{(i)}, \dots, \hat{z}_T^{(i)})$
Residual block (k, d)
Dropout
ReLU
WeightNorm
Dilated Causal Conv
Dropout
ReLU
WeightNorm
Dilated Causal Conv
1x1 Conv (optional)
$\hat{\mathbf{z}}^{(i-1)} = (\hat{z}_1^{(i-1)}, \dots, \hat{z}_T^{(i-1)})$
Residual block (k=3, d=1)
$\hat{z}_{T-1}^{(1)}$ $\hat{z}_T^{(1)}$
Convolutional Filter
Identity Map (or 1x1 Conv)
x_0 x_1 $\dots$ x_{T-1} x_T

[3] Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv preprint arXiv:1803.01271 (2018).

Performance of ESRNN and ESCNN

Performance of ESRNN and ESCNN by dataset of the M4 competition

Overall Weighted Average (OWA):

$$OWA = \frac{1}{2}\left[\frac{sMAPE}{sMAPE_{Naive2}} + \frac{MASE}{MASE_{Naive2}}\right]$$

DATASET	ESRNN	ESCNN	M4
Yearly	0.785	0.801	0.778
Quarterly	0.879	0.902	0.847
Monthly	0.872	0.895	0.836
Hourly	0.615	0.876	0.920
Weekly	0.952	0.986	0.920
Daily	0.968	0.970	0.920

Similarity of local trends

- Multiplicative decomposition of time series.

$y_t = \tau_t * l_t * s_t$ where τ_t is the trend, l_t the level and s_t the seasonality

- Models such as ESRNN estimate future local trends based on previous local trends

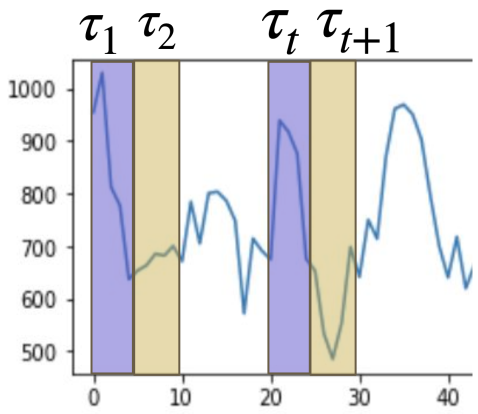
$F : T_t \rightarrow T_{t+1}$, where $T \subseteq \mathbb{R}^d$ is a local trend of length d.

Is function F across time series the **same or similar**?

Similarity of local trends

Is function F across time series the **same or similar**?

For each serie i, compute the set $\{(\tau_t^{(i)}, \tau_{t+1}^{(i)})\}$ which contains local consecutive local trends.

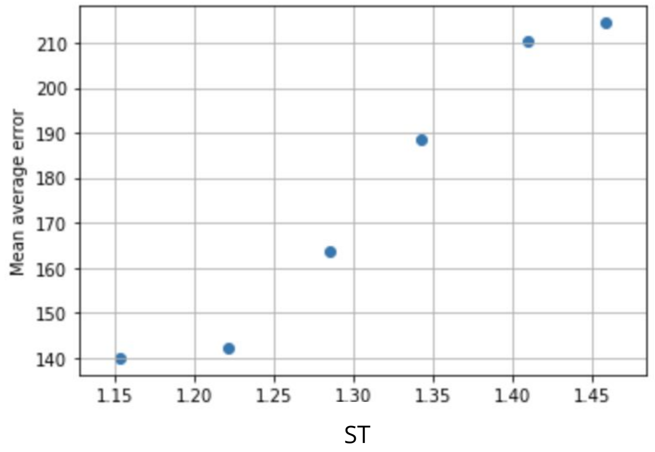


- Similarity test of F between two series:

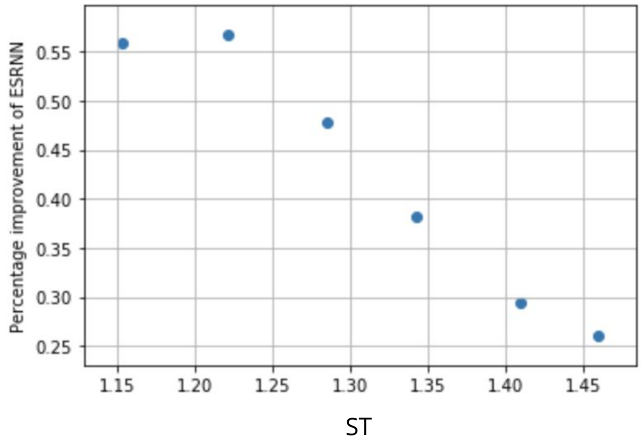
$$\hat{S} = \frac{1}{n_1 * n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} |\tau_{t+1}^{(i)} - \tau_{t+1}^{(j)}| K\left(||\tau_t^{(i)} - \tau_t^{(j)}||_2^2 \right)$$

Empirical results: simulated data

Similarity Test vs Error of ESNN



Similarity Test vs Improvement of ESNN over univariate ES

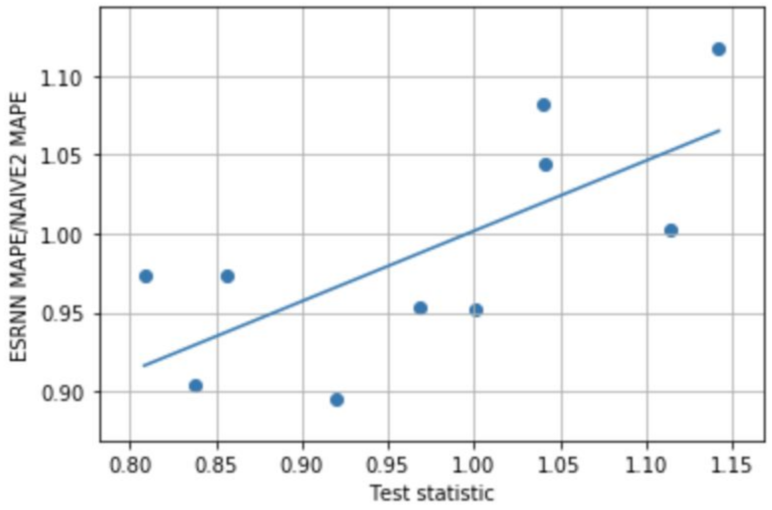


Similarity test proposed vs error and improvement of the ESRNN over an univariate ES model.

The data was simulated controlling for the shared information between time series.

Empirical results: M4 data

Similarity Test vs Improvement of ESNN over Naive2



Similarity test proposed vs improvement (relative MAPE) of the ESRNN over an univariate Naive2 model.

Different sets of time series where randomly selected from two categories of data (Demographics and Finance), varying the proportion sampled from each category.

Contribution and Future Work

1. Novel useful test of asynchronous shared local trends.
2. Empirical correlation between test and relative performance of ESRNN.
3. Potential applications:
   a. Model selection
   b. Improve models by incorporating the test for clustering
   c. Anomaly detection

Multi-Agent Adversarial Inverse Reinforcement Learning[1]

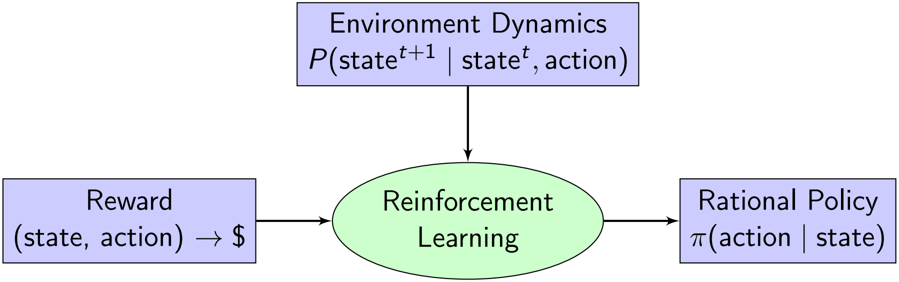
Matt Battifarano

Department of Civil and Environmental Engineering
10-716 Advanced Machine Learning: Theory and Methods
Carnegie Mellon University

28 April 2020

[1] L. Yu, J. Song, and S. Ermon, "Multi-agent adversarial inverse reinforcement learning", in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, Sep. 2019, pp. 7194–7201. [Online]. Available: http://proceedings.mlr.press/v97/yu19e.html.

What is it?

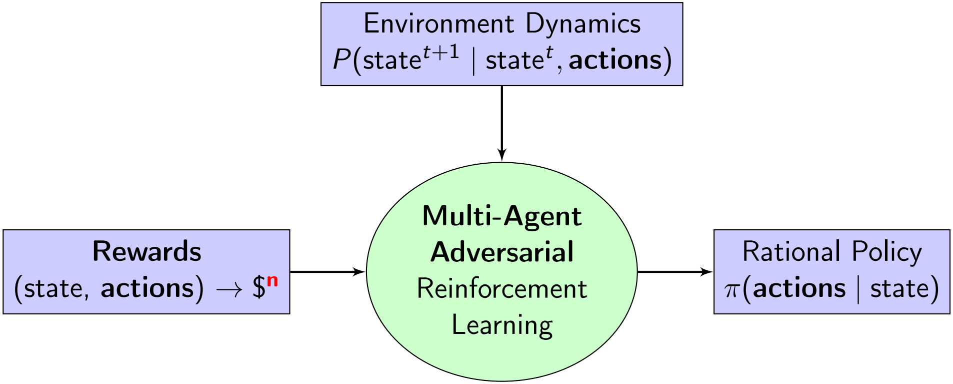


Environment Dynamics
$P(\text{state}^{t+1} \mid \text{state}^{t}, \text{action})$
Reward
(state, action) $\rightarrow$ \$
Reinforcement Learning
Rational Policy
$\pi(\text{action} \mid \text{state})$

A Rational policy

is one that **maximizes** the expected reward.

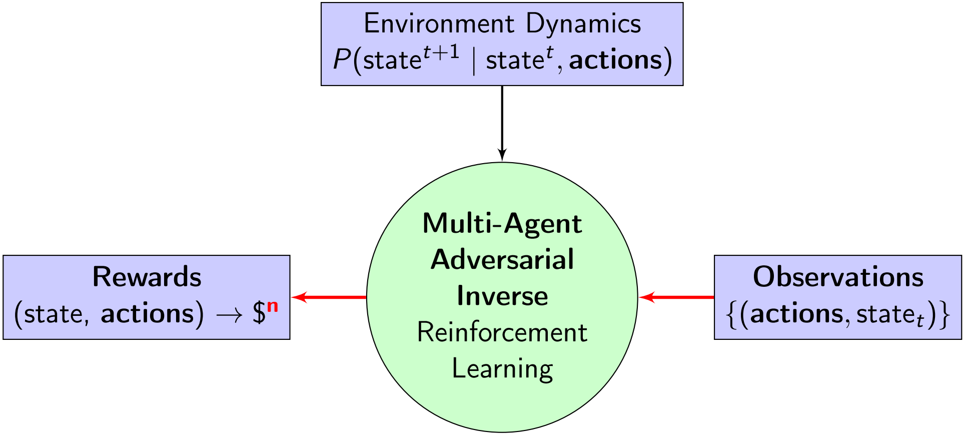
What is it?


Environment Dynamics
$P(\text{state}^{t+1} \mid \text{state}^t, \textbf{actions})$
Rewards
$(\text{state}, \textbf{actions}) \to \$^{\mathbf{n}}$
Multi-Agent Adversarial Reinforcement Learning
Rational Policy
$\pi(\textbf{actions} \mid \text{state})$

A Rational Policy

is one in which **no agent can increase their reward by unilaterally altering their own policy**.

What is it?



Environment Dynamics
$P(\text{state}^{t+1} \mid \text{state}^{t}, \textbf{actions})$
Multi-Agent Adversarial Inverse Reinforcement Learning
Rewards
$(\text{state}, \textbf{actions}) \rightarrow \$^{\mathbf{n}}$
Observations
$\{(\textbf{actions}, \text{state}_t)\}$

Multi-Agent Adversarial Inverse Reinforcement Learning:

- **recover** each agents reward function,
- **given** observations of expert behavior *and* the environment dynamics.

Why do we care?

- Many real-world systems can be understood as the result of a set **agents competing and/or cooperating to achieve their own goals**.
- It is often very **difficult to measure the reward function** of each agent directly.
- It is often comparatively **simple to measure states and actions**.
- Useful to *rationalize* a set of observed behaviors.

Example: A routing game

- We observe the GPS traces of vehicles on a road network.
- Assume that each driver observes road network (state) and attempts to select their route (action) to maximize their own expected utility.
- Goal: find the parameters of the reward function $r_\theta(s, \mathbf{a})$, which *best rationalizes* the observed gps traces.

How does it work?

Key Challenges

- Characterize the **joint trajectory distribution** induced by the reward parameters.
- Handle **bounded rationality** of the observed trajectories.

How does it work?

Key Ideas

- **Logistic Stochastic Best Response Equilibrium** (LSBRE): each agent, in turn, optimizes their stochastic policy[2] with all other actions *fixed*. Repeat until convergence. (Think Gibbs sampling)
- Each agent's conditional policy is the *softmax* over their *value function*: **higher value actions are selected with higher probability**

Thm 1 The **trajectory distribution** of the conditional policies in LSBRE are close (in KL) to a distribution **exponential in the sum of rewards**.

Thm 2 Approximate the joint likelihood by the pseudolikelihood and the psuedolikelihood (via theorem 1) by the **sum of rewards** with an (intractable) partition function.

[2]McKelvey and Palfrey, "Quantal response equilibria for normal form games"; McKelvey and Palfrey, "Quantal response equilibria for extensive form games".

How does it work?

Implementation

- Use a **generative adversarial**[3] framework to simultaneously estimate the partition function *and* the reward function.
- The **generators** estimate the **partition function** (optimal policy) for each agent in order to produce realistic trajectories
- The **discriminator**, estimates the **reward function** for each agent, which it used to evaluate how realistic a given trajectory is.

[3]Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio, "Generative adversarial nets"; Ziebart, Maas, Bagnell, and Dey, "Maximum entropy inverse reinforcement learning."

Thank You!

References

L. Yu, J. Song, and S. Ermon, "Multi-agent adversarial inverse reinforcement learning", in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, Sep. 2019, pp. 7194–7201. [Online]. Available: `http://proceedings.mlr.press/v97/yu19e.html`.

R. D. McKelvey and T. R. Palfrey, "Quantal response equilibria for normal form games", *Games and economic behavior*, vol. 10, no. 1, pp. 6–38, 1995.

——,"Quantal response equilibria for extensive form games", *Experimental economics*, vol. 1, no. 1, pp. 9–41, 1998.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets", in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.", in *AAAI*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.

Constrained Clustering

Euxhen Hasanaj, Dennis Li

Carnegie Mellon University

Introduction

Feature Based Clustering:

- Pointwise semi-supervision
  - User provides labels for a subset of the data
- **Pairwise semi-supervision**
  - **Must-Link constraints**
  - **Cannot-Link constraints**

Idea 1: Modified EM

Must-Link Constraints

- Modify the E-step to only consider assignments $\mathbf{Y}$ that comply with the given constraint

$$\gamma^{(t+1)} = \mathbb{E}_{\mathbf{Y}|\mathbf{X},\Theta^{(t)},E_{\mathcal{ML}}}[\mathcal{LL}(\Theta; \mathbf{X}, \mathbf{Y}, E_{\mathcal{ML}})]$$

- Expected log likelihood:

$$\begin{aligned}&\mathbb{E}_{\mathbf{Y}|\mathbf{X},\Theta^{(t)},E_{\mathcal{ML}}}[\mathcal{LL}(\Theta; \mathbf{X}, \mathbf{Y}, E_{\mathcal{ML}})]\\ &= \sum_{g=1}^{G}\sum_{c=1}^{C}\sum_{x_i \in X_c} \log p(x_i \mid g, \Theta) \cdot p(Y_c = g \mid X_c, \Theta^{(t)}) + \sum_{g=1}^{G}\sum_{c=1}^{C} \log \alpha_g^{(t)} \cdot p(Y_c = g \mid X_c, \Theta^{(t)})\end{aligned}$$

Must-Link Constraints: M-step

- To get the update rules differentiate $\mathbb{E}_{\mathbf{Y}|\mathbf{X},\Theta^{(t)},E_{\mathcal{ML}}}[\mathcal{LL}(\Theta;\mathbf{X},\mathbf{Y},E_{\mathcal{ML}})]$ to get [1]:

$$\alpha_g^{(t+1)} = \frac{1}{C}\sum_{c=1}^{C} p(Y_c = g \mid X_c, \Theta^{(t)})$$

$$\mu_g^{(t+1)} = \frac{\sum_{c=1}^{C} \overline{X}_c \; p(Y_c = g \mid X_c\Theta^{(t)}) \; |X_c|}{\sum_{c=1}^{C} p(Y_c = g \mid X_c, \Theta^{(t)}) \; |X_c|}$$

$$\Sigma_g^{(t+1)} = \frac{\sum_{c=1}^{C} \Sigma_{cg}^{(t+1)} \; p(Y_c = g \mid X_c, \Theta^{(t)}) \; |X_c|}{\sum_{c=1}^{C} p(Y_c = g \mid X_c, \Theta^{(t)}) \; |X_c|}$$

- Constrained EM is essentially treating every chunklet as a single point, but weighted according to the number of points in that chunklet.

Cannot-Link Constraints

- Must-link constraints satisfy **transitivity**:

$$(a, b) \in \overline{\mathcal{ML}}, (b, c) \in \overline{\mathcal{ML}} \rightarrow (a, c) \in \overline{\mathcal{ML}}$$

- Cannot-link constraints do not satisfy **transitivity**:

$$(a, b) \in \mathcal{CL}, (b, c) \in \mathcal{CL} \not\rightarrow (a, c) \in \mathcal{CL}$$

Cannot-Link Constraints

- Likelihood:

$$p(\mathbf{X}, \mathbf{Y} \mid \Theta, E_{\mathcal{CL}}) = \frac{1}{Z} \prod_{i=1}^{L} (1 - \mathbf{I}(y_{a_i} = y_{b_i})) \prod_{n=1}^{N} p(y_n \mid \Theta) \; p(x_n \mid y_n, \Theta)$$

- Can be described by a Markov network with potentials $p(y_n \mid \Theta), p(x_n \mid y_n, \Theta)$ and $1 - \mathbf{I}(y_{a_i} = y_{b_i})$.

- Calculating the posteriors $p(\mathbf{Y} \mid \mathbf{X}, \Theta^{(t)}, E_{\mathcal{CL}})$ and the updated α requires calculating Z.
  - Have $O(n)$ cannot-link constraints and use inference algorithms such as Pearl's junction tree algorithm.

Combine Must-Link and Cannot-Link Constraints

- Extend Cannot-Link likelihood equation with new potentials for the must-link constraints.
- Use a single Markov network with likelihood function [1]:

$$
\begin{aligned}
& p(\mathbf{X}, \mathbf{Y} \mid \Theta, E_{\mathcal{ML}}, E_{\mathcal{CL}}) \\
=& \frac{1}{Z} \prod_{c=1}^{C} \mathbf{I}(y_{X_C}) \prod_{i=1}^{L} (1 - \mathbf{I}(y_{a_i} = y_{b_i})) \ \prod_{n=1}^{N} p(y_n \mid \Theta) \ p(x_n \mid y_n, \Theta)
\end{aligned}
$$

Idea 2: EM with Posterior Constraints

- Typically EM maximizes an auxiliary lower bound:

$$\mathcal{L}(\Theta; \mathbf{X}, \mathbf{Y}) = \mathbb{E}\left[\log \sum_{\mathbf{Y}} q(\mathbf{Y} \mid \mathbf{X}) \frac{p(\mathbf{X}, \mathbf{Y}; \Theta)}{q(\mathbf{Y} \mid \mathbf{X})}\right] \geq \mathbb{E}\left[\sum_{\mathbf{Y}} q(\mathbf{Y} \mid \mathbf{X}) \log \frac{p(\mathbf{X}, \mathbf{Y}; \Theta)}{q(\mathbf{Y} \mid \mathbf{X})}\right] = F(q, \Theta).$$

  which can be made tight by maximizing over q.
- Posterior constraints: $p(\mathbf{Y} \mid \mathbf{X}, \Theta) \in \mathcal{Q}(\mathbf{X})$, constraint set $\mathcal{Q}$
  - E.g., Cannot-Link: posterior satisfies $\mathbb{E}[z_{ag} + z_{bg}] \leq 1$.
- Differences:
  - Variational EM: constrain to a smaller tractable subspace $\mathcal{Q}$ in the original intractable space
  - Here: we assume the original space is tractable, imposing constraints on posteriors to enforce semantics not captured by the simpler model

EM with Posterior Constraints

- Instead of penalizing p directly, penalize the distance from p to $\mathcal{Q}$. Can show that this can be achieved by restricting q to be in $\mathcal{Q}$ instead:

$$q^{(t+1)} = \arg\max_{q \in \mathcal{Q}} F(q, \Theta^{(t)}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q(\mathbf{Y} \mid \mathbf{X}) \mid\mid p(\mathbf{Y} \mid \mathbf{X}, \Theta^{(t)})$$
$$\text{such that } \mathbb{E}[f(\mathbf{X}, \mathbf{Y})] \leq b.$$

  Downside: no longer guaranteed that the local maxima of the constrained problem are local maxima of the log-likelihood.

EM with Posterior Constraints

Proposition 1 [2]

The local maxima of $F(q, \Theta)$, subject to $q(\mathbf{Y} \mid \mathbf{X}) \in \mathcal{Q}(\mathbf{X})$ are local maxima of

$$\mathbb{E}[\log p(\mathbf{X}; \Theta))] - \mathbb{E}\left[\mathrm{KL}(\mathcal{Q}(\mathbf{X}) \mid\mid p(\mathbf{Y} \mid \mathbf{X}, \Theta))\right].$$

- Trades off likelihood and distance to the desired posterior subspace
- KL projection onto $\mathcal{Q}$ can be solved via its dual

$$\arg\max_{\lambda \geq 0} \left(\lambda^T \mathbf{b} - \log \sum_{\mathbf{Y}} p(\mathbf{Y} \mid \mathbf{X}, \Theta) \exp\{\lambda^T f(\mathbf{X}, \mathbf{Y})\} \right)$$

- Can be solved using projected GD.

Experiments

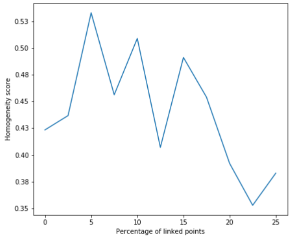


Figure: MNIST. Homogeneity-Score.

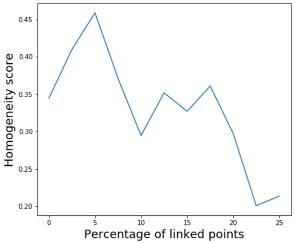


Figure: GE. Homogeneity-Score.

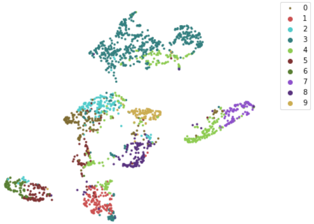


Figure: MNIST. UMAP visualization.

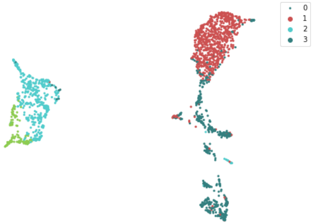


Figure: GE. UMAP visualization.

Sources

N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall, "Computing Gaussian mixture models with EM using equivalence constraints," *Advances in Neural Information Processing Systems*, 2004.

J. V. Graça and B. Taskar, "Expectation Maximization and Posterior Constraints," *NIPS 2007, Advances in Neural Information Processing Systems 20*, pp. 569–576, 2008.

Constraint vs Score: What's the tradeoff?

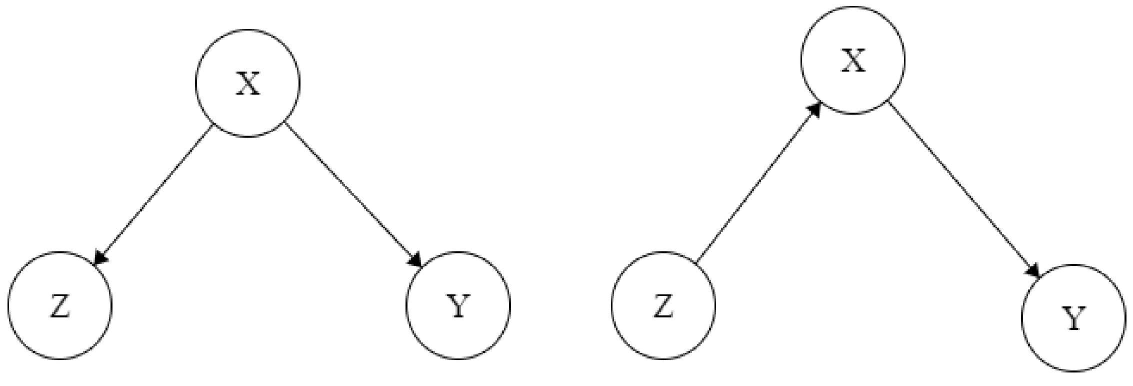
— a basic survey of Bayesian network learning algorithm

Shuyan Wang

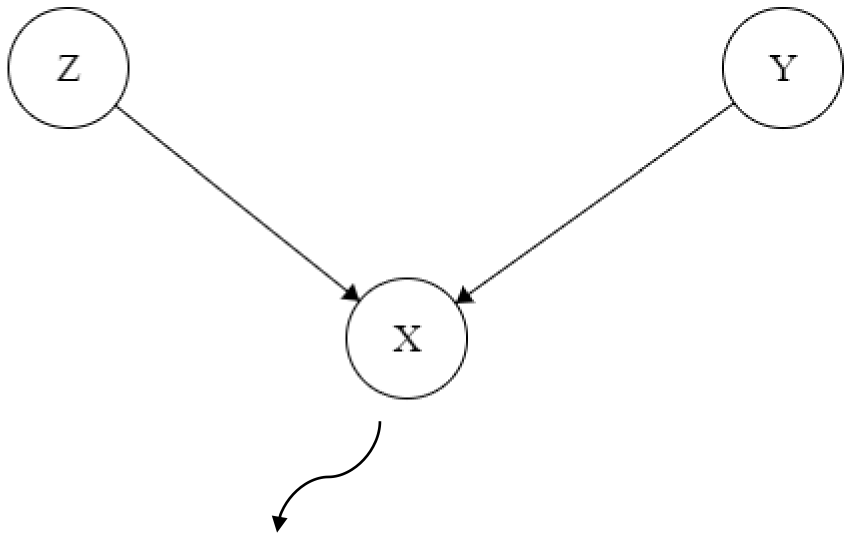
Causal Graph and Algorithm

- 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

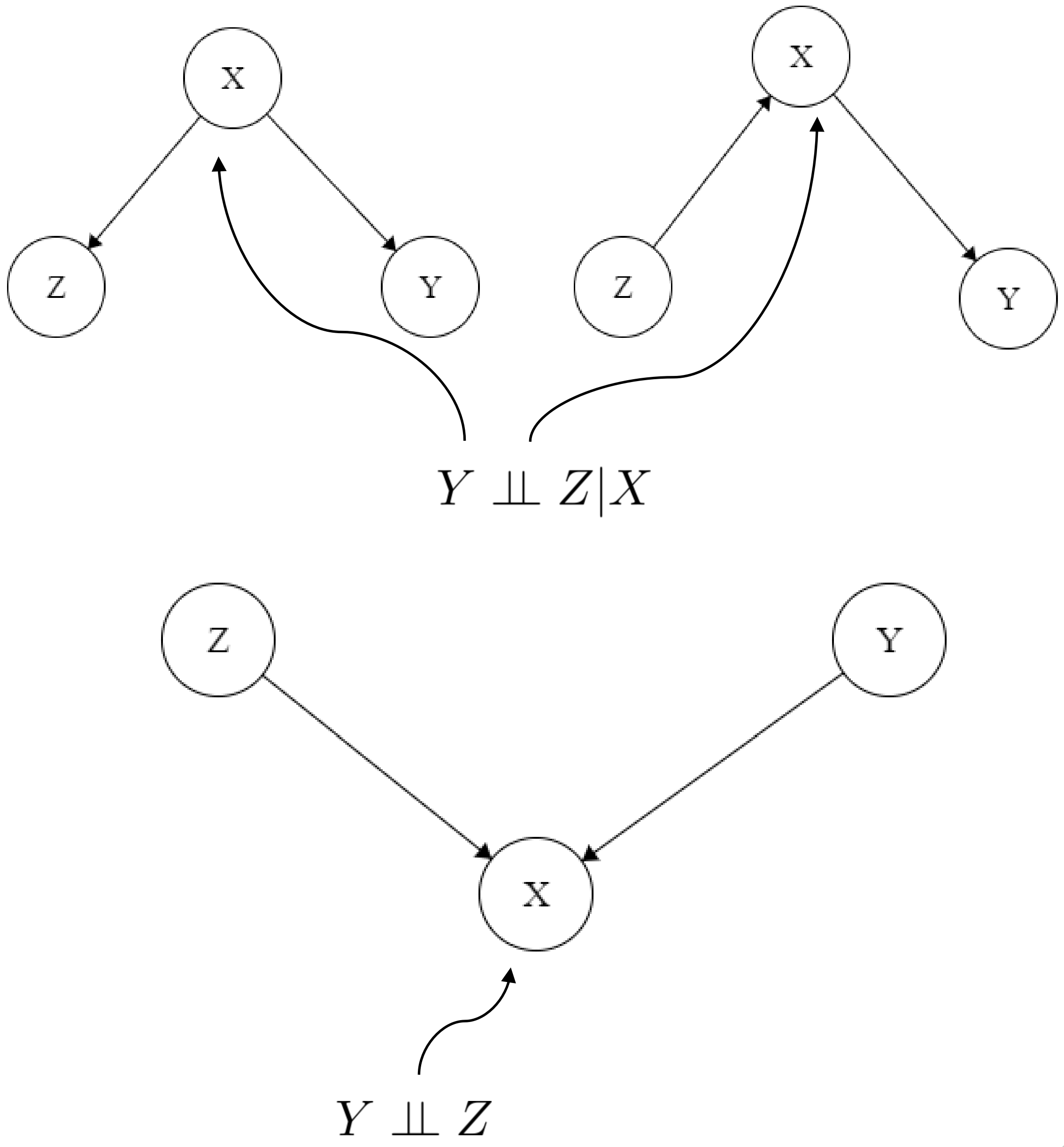


$$Y \perp\!\!\!\perp Z | X$$



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

Faithfulness Assumption



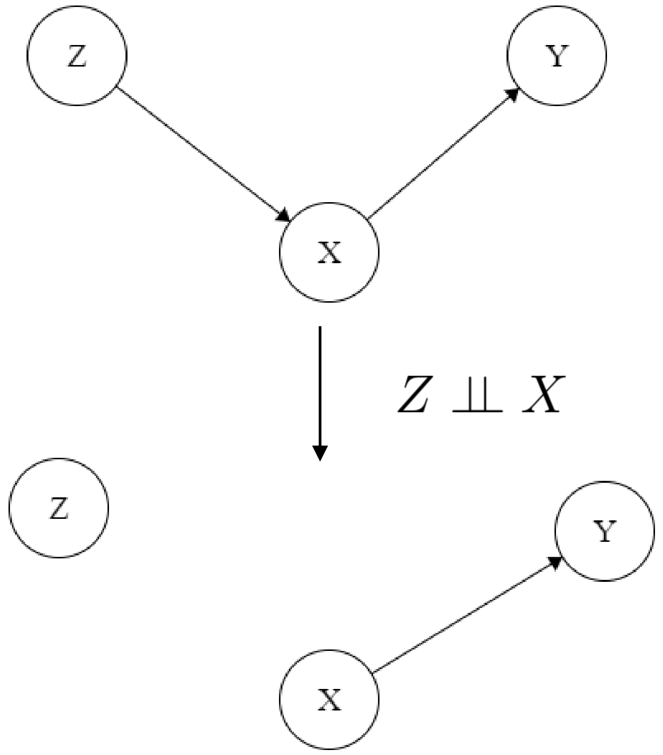
Meek Conjecture

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

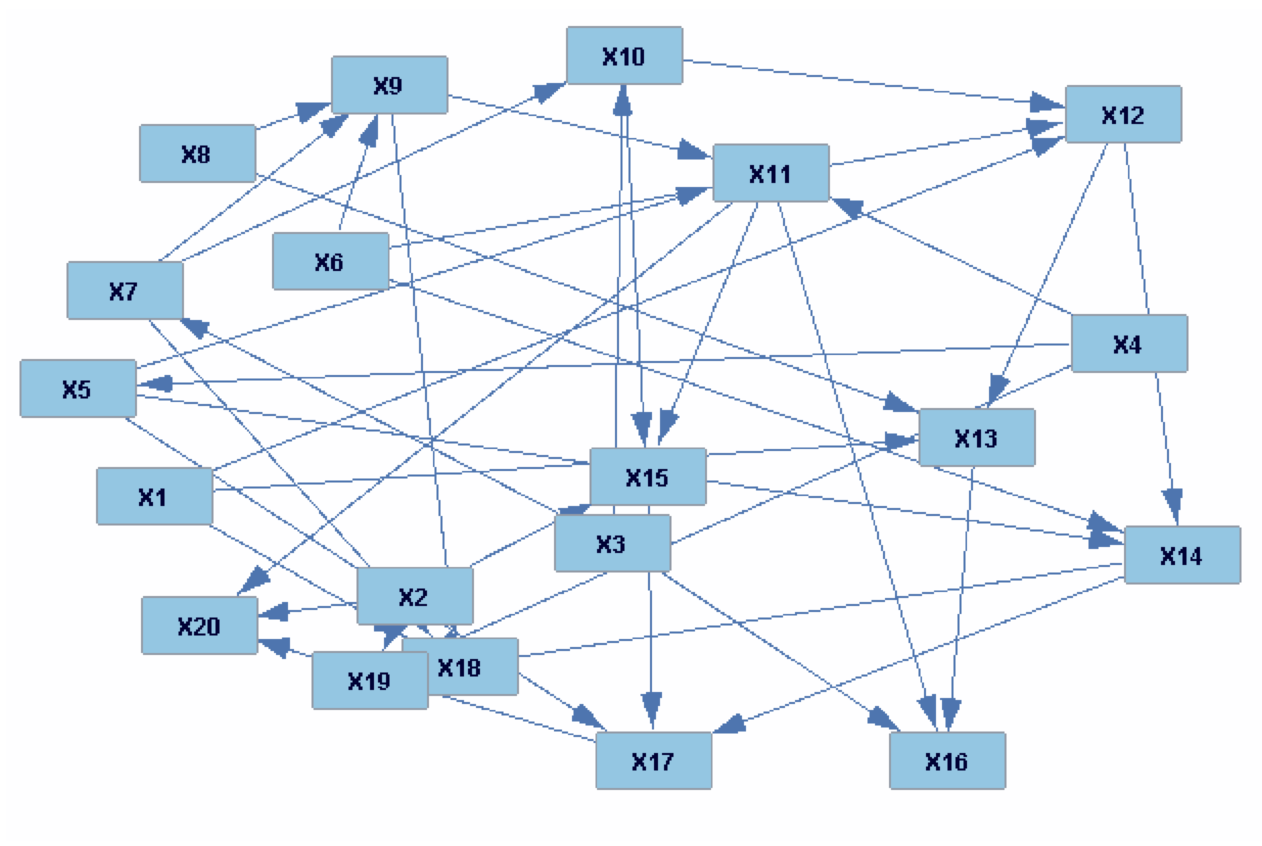
- Linear
- Gaussian
- 20 variables
- Average degree:4, 8, 12
- Constraint-based:FCI
- Score-based:FGES
- Combination: GFCI

Simulation Uses Linear Gaussian Model

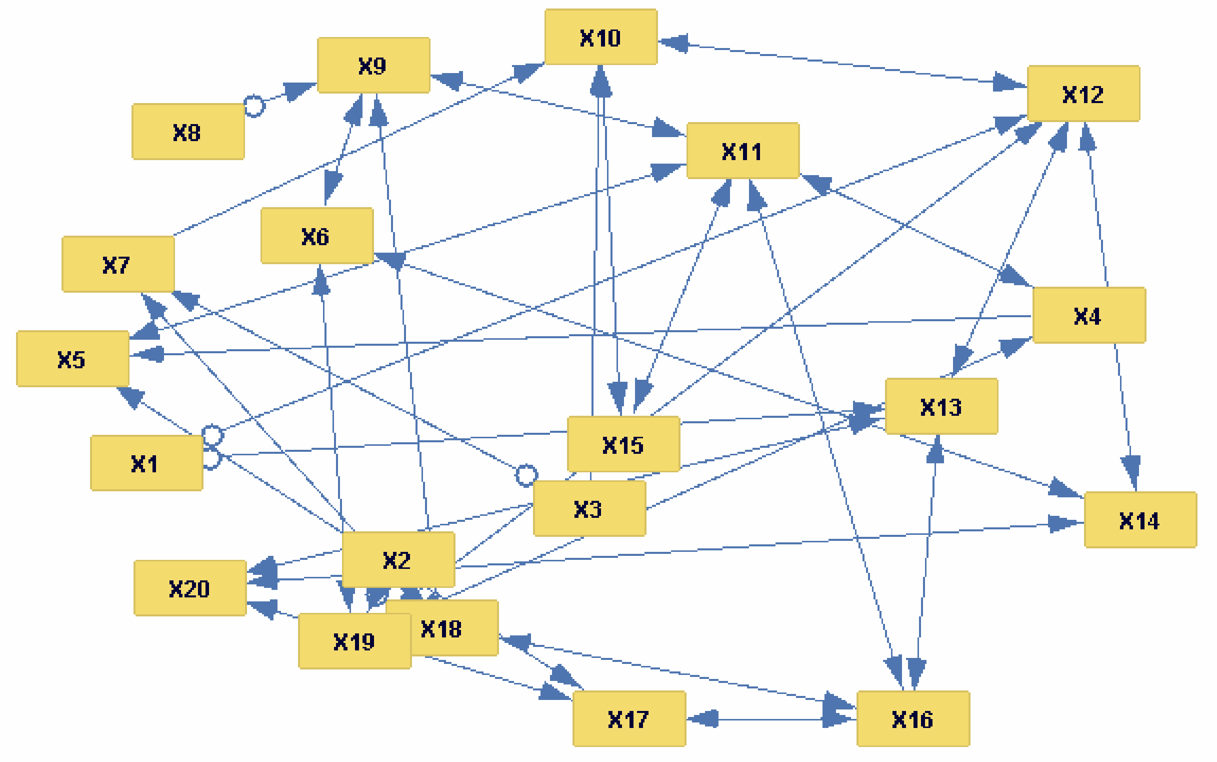
$$\begin{tikzcd}
Z \arrow[dr, "a_1"'] & & Y \arrow[dl, "a_2"] \\
& X &
\end{tikzcd}$$

$$\mathrm{Data}(X,Y,Z) \sim P(X,Y,Z) \sim X = a_1 Z + a_1 Y + e$$

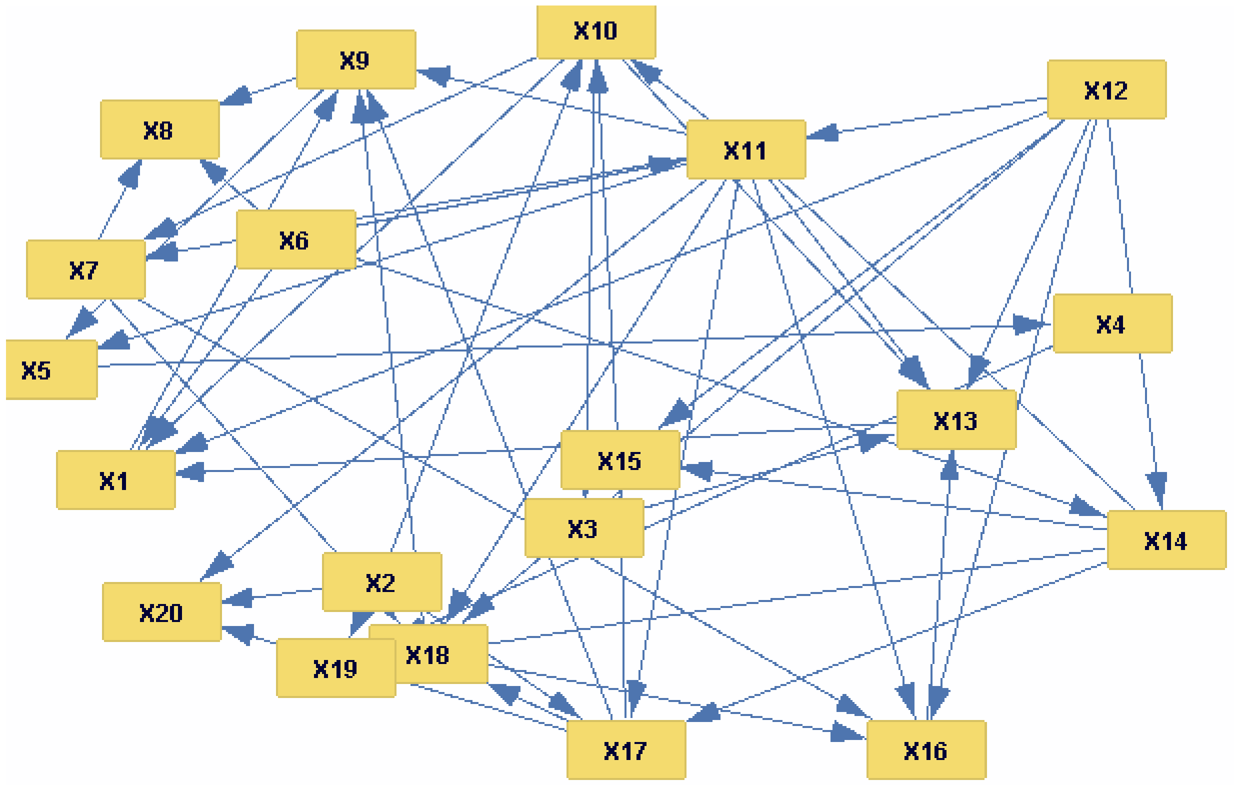
20 variables with ave. degree of 4



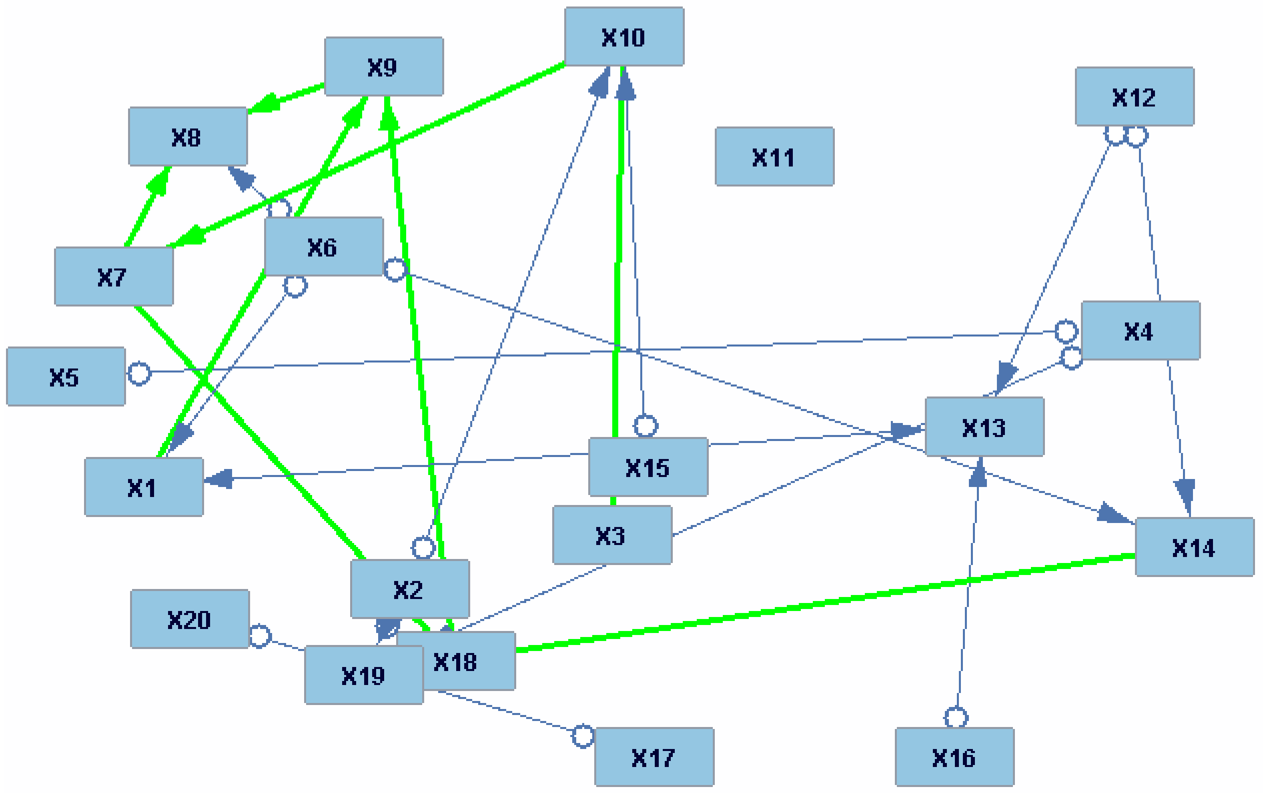
FCI


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

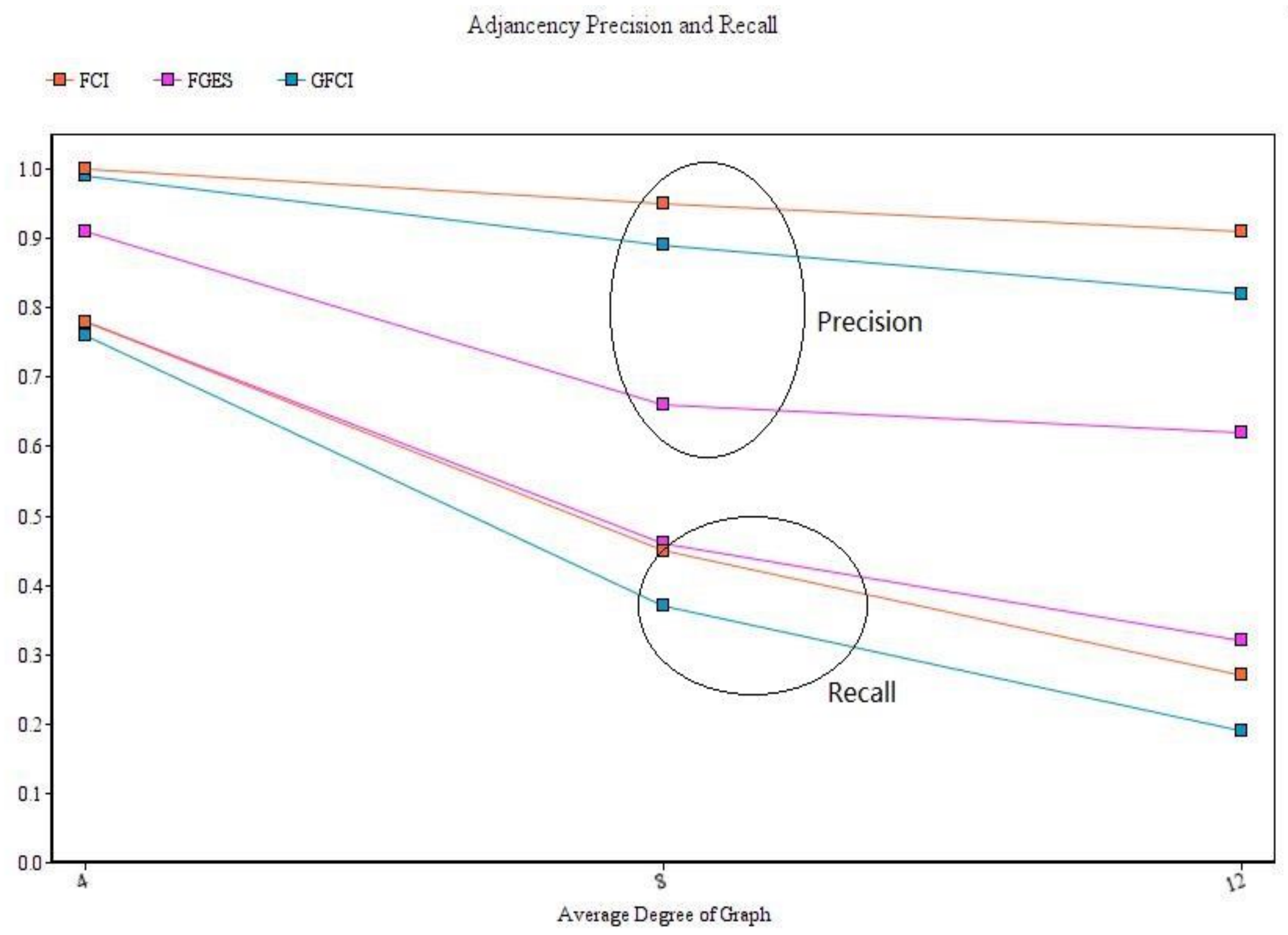
GFCI


X10
X9
X12
X8
X11
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

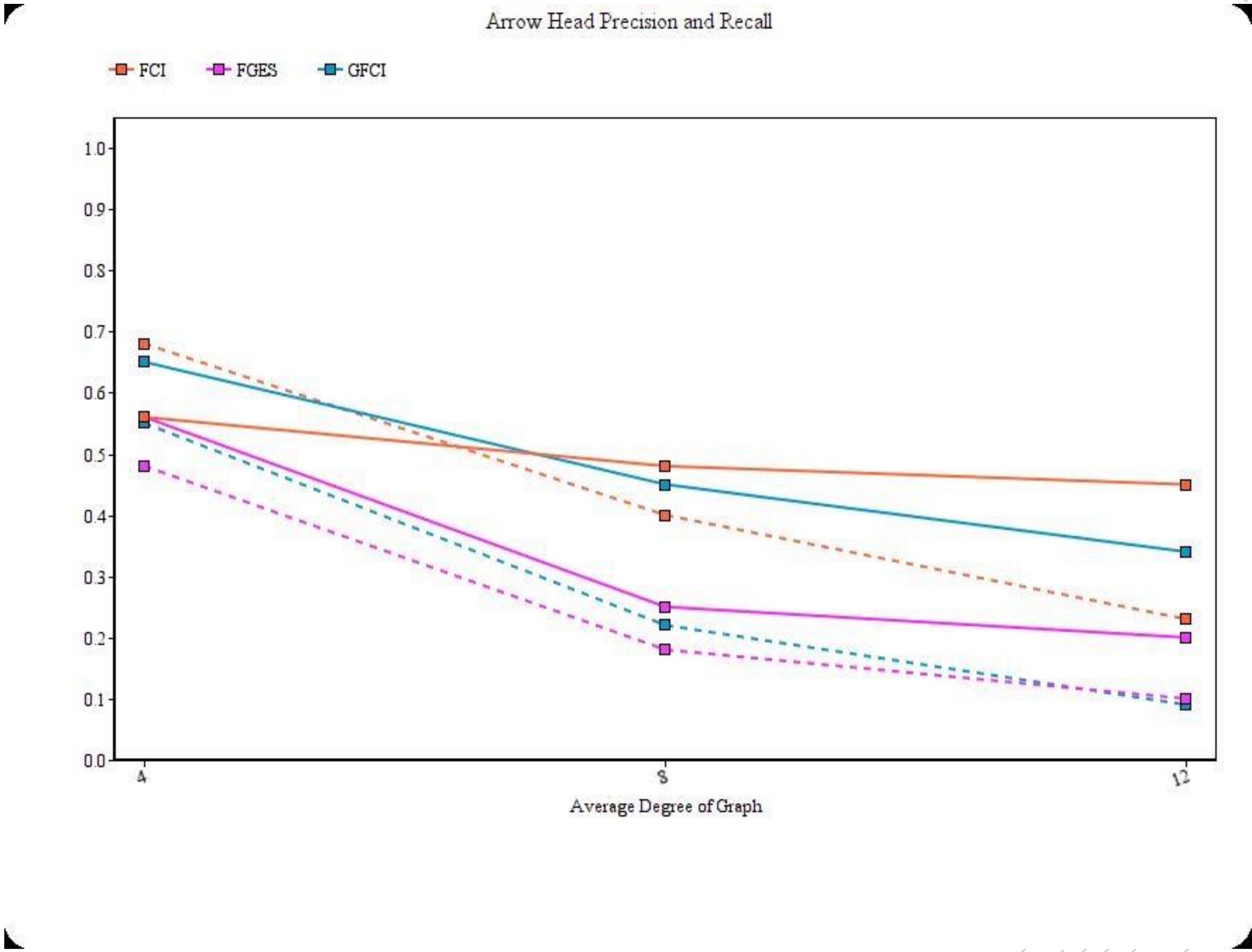
Adjancency Precision & Recall



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