The Many Faces of Model Selection

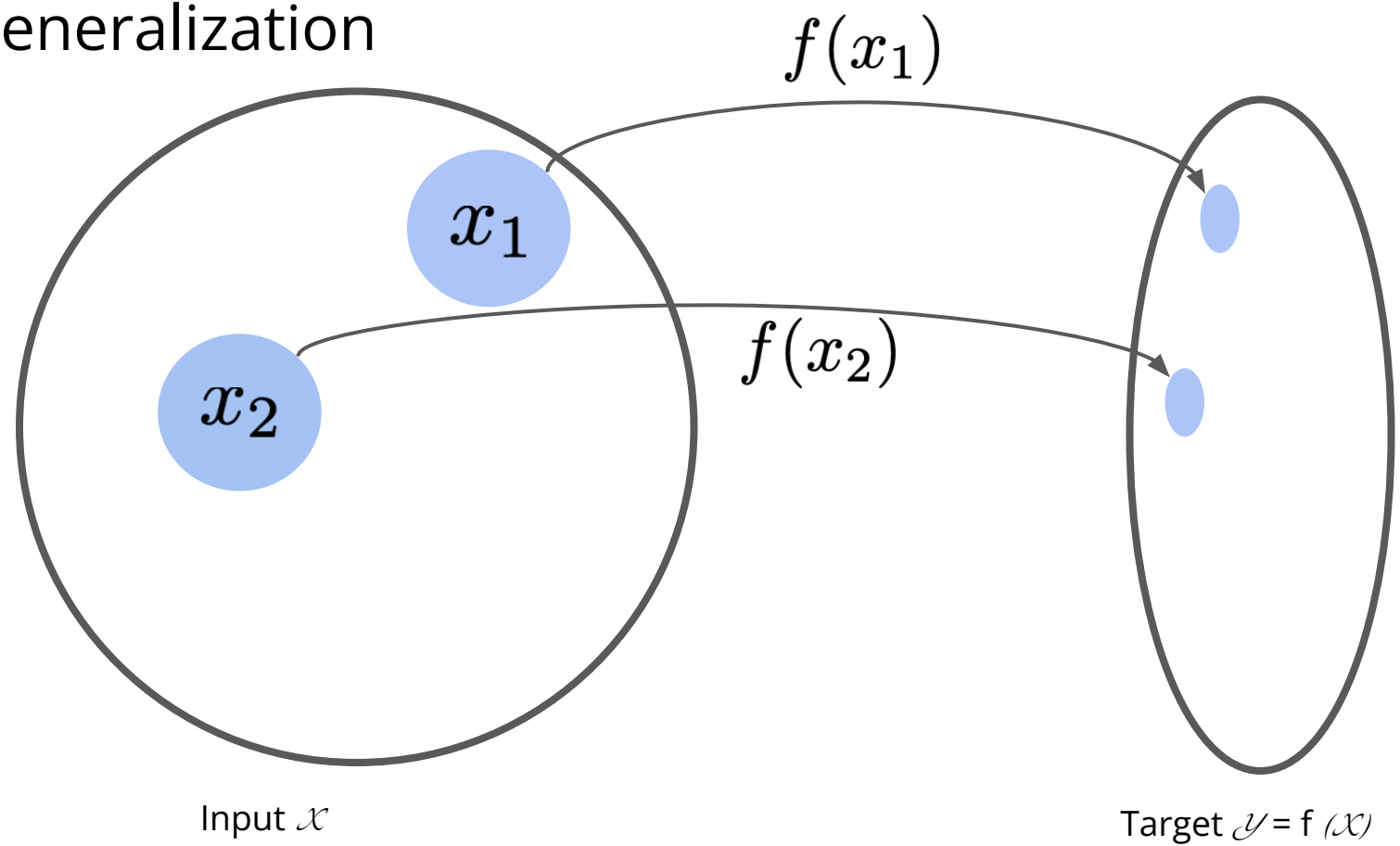
Clare Lyle
November 2021

Talk Outline

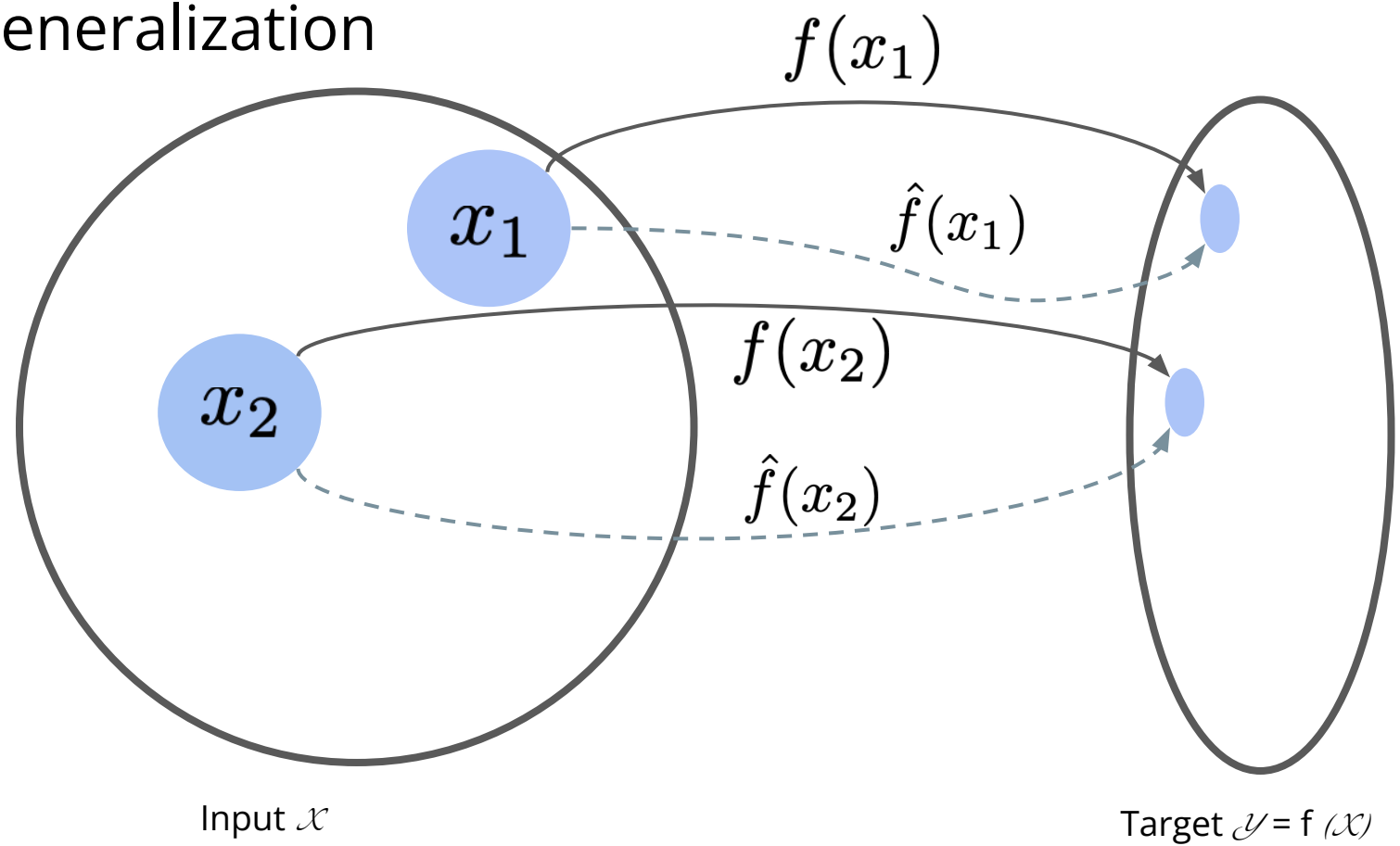
- Introduction: model selection and generalization
- Model selection in the i.i.d. setting
  - Bayesian model selection and training speed
  - Applications in neural architecture search
- Causal perspectives on generalization
  - Identifying causal structure for multi-environment generalization in RL
- Concluding thoughts

Generalization

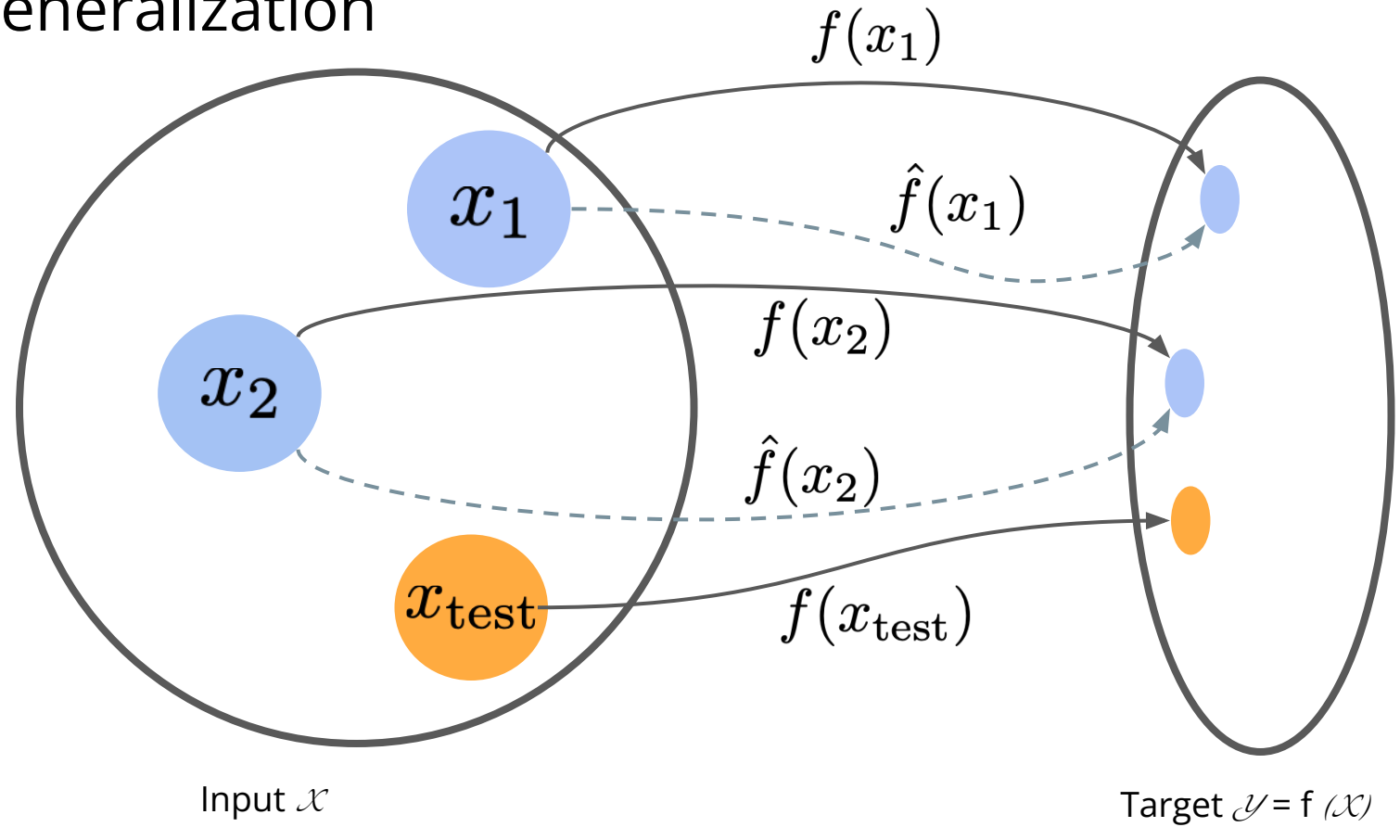
Generalization



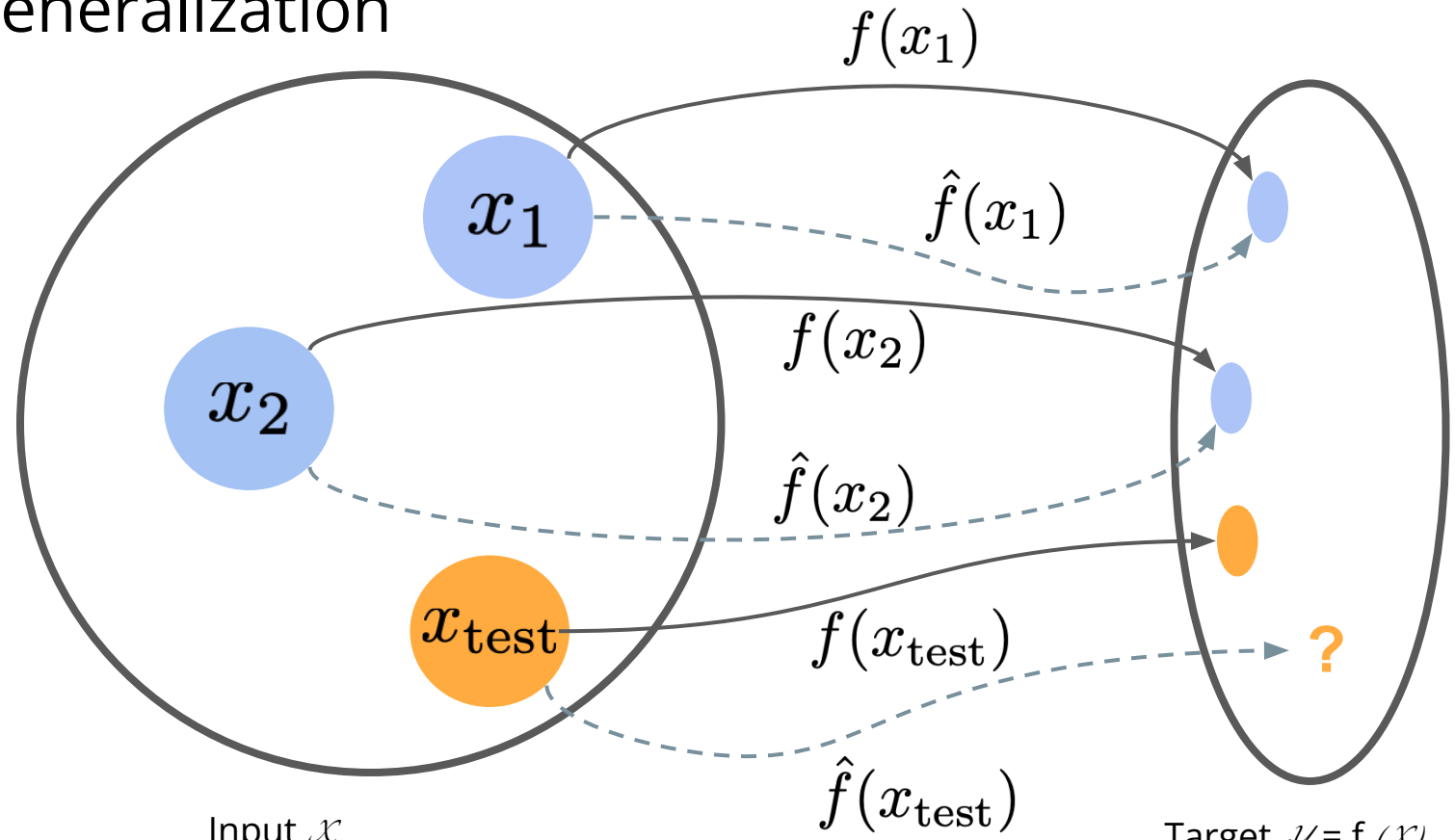
Generalization



Generalization

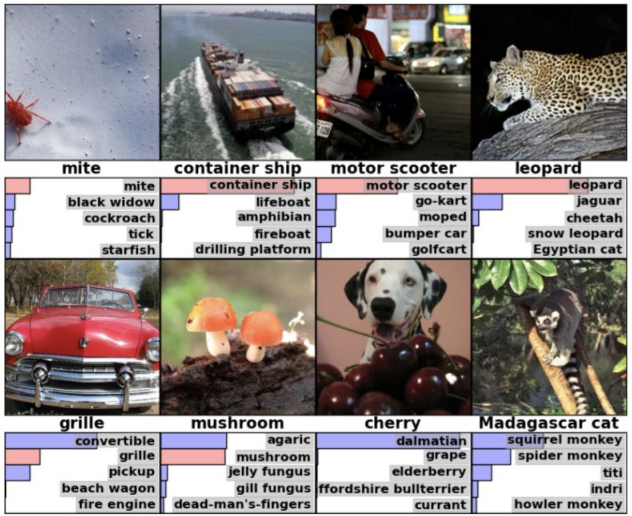


Generalization



Risk minimization: formalism

- Given: a finite set of inputs X_{train} and outcomes Y_{train}
- Want to learn to predict $P(Y|X)$, with X sampled from some test distribution $P(X_{test})$.
  - In generative modelling (not the focus of this talk), may want to also model P(X)
- Example: image classification.
  - X_{train}, Y_{train} are the training set of image-label pairs
  - $P(Y|X)$ = categorical distribution (e.g. softmax) over labels
  - X_{test}, Y_{test} = the data the model sees at deployment



IID vs OOD

Different learning frameworks depend on the type of shared structure between X_{train} and X_{test}

- Same distribution: i.i.d. setting
- Same causal graph: causal structure identification
- Same conditional $P(Y|X)$, different X: covariate shift
- Same world dynamics, different objective: multi-task RL
- Nebulous “shared structure”: meta-learning, transfer learning

We are focusing on settings where the nature of the shared structure can be expressed mathematically .

IID

Train

Test

OOD (Out-of-distribution)

Train

Test

Adversarially OOD

Train

Test

IID vs OOD

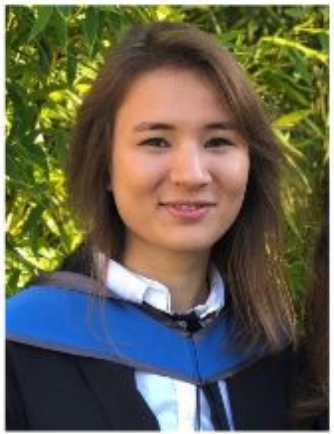
Different learning frameworks depend on the type of shared structure between X_{train} and X_{test}

- **Same distribution:** expected risk minimization
- **Same causal graph:** causal structure identification
- Same conditional $P(Y|X)$, different X: covariate shift
- Same world dynamics, different objective: multi-task RL
- Nebulous “shared structure”: meta-learning, transfer learning

We are focusing on settings where the nature of the shared structure can be expressed mathematically .

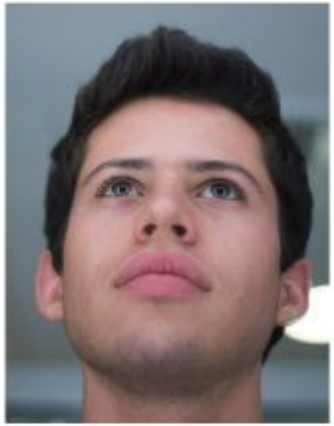
Within-distribution generalization

Credits



Lisa Schut

Robin Ru

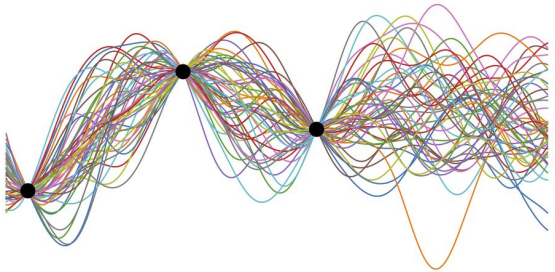


Yarin Gal

Mark van der Wilk

“A Bayesian Perspective on Training Speed and Model Selection”, NeurIPS 2020
“Speedy Performance Estimation for Neural Architecture Search”, NeurIPS 2021

Bayesian Inference 101



Bayesian Model M = prior over parameters + likelihood function

Examples: Bayesian linear regression, Gaussian Processes, Bayesian neural networks

Prior distribution P(w|M) is defined over parameters w

Likelihood function P(D|w) maps parameters to a distribution over data D.

Parameter inference updates a posterior belief over which parameters are likely to have generated the data using Bayes' rule.

$$P(w|\mathcal{D}) = \frac{P(\mathcal{D}|w)P(w|\mathcal{H})}{P(\mathcal{D})}$$

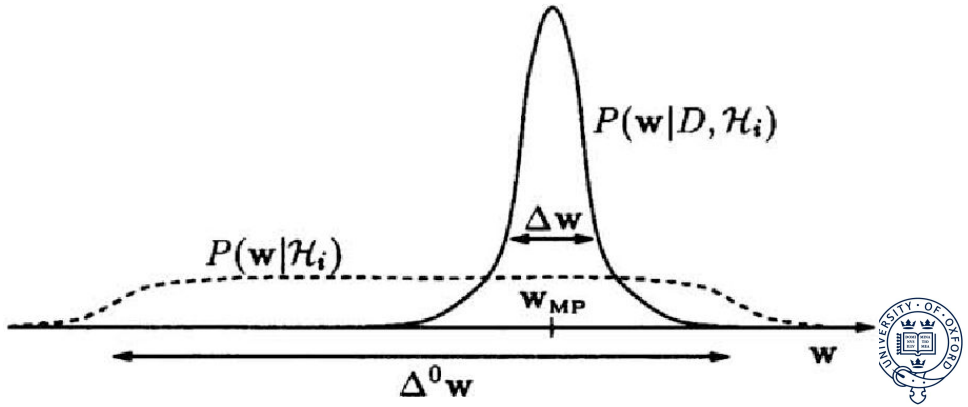
Bayesian Inference 101

Parameter inference updates a posterior belief over which parameters are likely to have generated the data using Bayes' rule.

Model inference updates posterior belief over which model is most likely to have generated the data.

$$P(w|\mathcal{D}) = \frac{P(\mathcal{D}|w)P(w|\mathcal{H})}{P(\mathcal{D})}$$

$$P(\mathcal{H}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{H})P(\mathcal{H})}{P(\mathcal{D})}$$

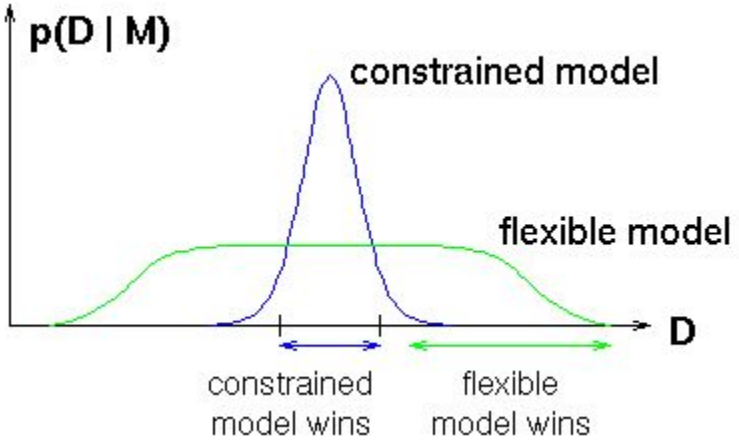


The Marginal Likelihood

Marginal likelihood = Likelihood of data given **model**

(marginalized over parameters)

$$P(\mathcal{D}|\mathcal{H}) = \int_w P(\mathcal{D}|w)P(w|\mathcal{H})dw$$



Type II Bayesian Inference

Can use the marginal likelihood to infer a posterior belief over the model H via Bayes' Rule

$$P(\mathcal{H}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{H})P(\mathcal{H})}{P(\mathcal{D})}$$

In practice do *Type II Maximum Likelihood*: pick the model from a given set with the highest marginal likelihood

$$\mathcal{H}^* = \max_{\mathcal{H}} \frac{P(\mathcal{D}|\mathcal{H})P(\mathcal{H})}{P(\mathcal{D})} \approx \max_{\mathcal{H}} P(\mathcal{D}|\mathcal{H})$$

Generalization and Model Selection

Generalization: pick the model that will *make the best predictions at deployment*

(Bayesian) Model Selection: pick the model that is *most likely to have generated my training data*

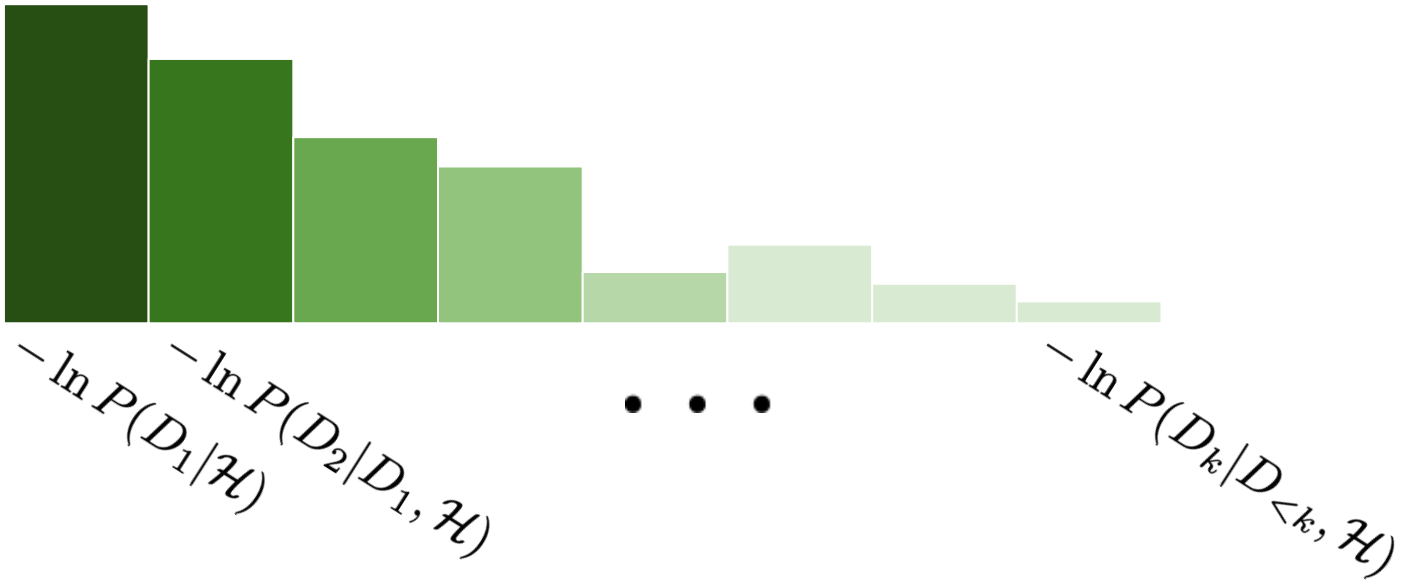
In practice, models with high marginal likelihood tend to generalize well.

Prequential Coding View

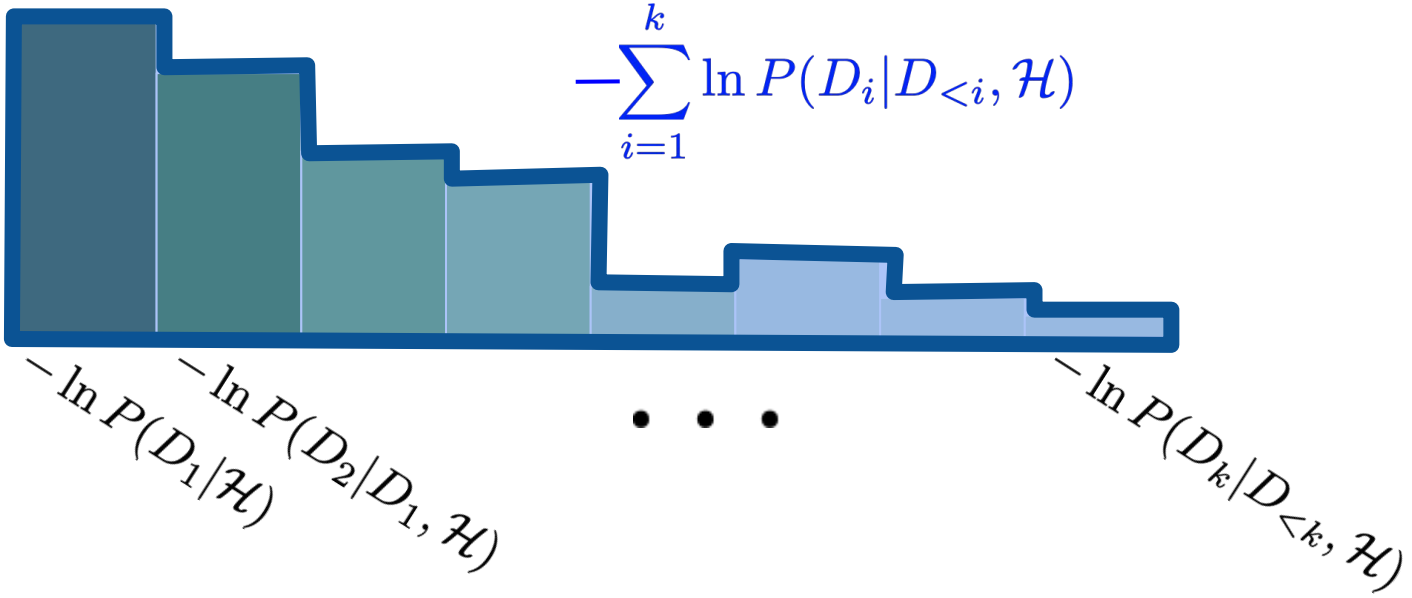
We can also view the marginal likelihood as measuring how well each data point explains the data that the model sees after it.

$$\log P(\mathcal{D}|\mathcal{H}) = \sum_{i=1}^{n} \log P(\mathcal{D}_i|\mathcal{D}_{<i}, \mathcal{H})$$

Bayesian updating as a training curve



Marginal Likelihood = Area Under Curve



Prequential Coding View

We can also view the marginal likelihood as measuring how well each data point explains the data that the model sees after it.

$$\log P(\mathcal{D}|\mathcal{H}) = \sum_{i=1}^{n} \underbrace{\log P(\mathcal{D}_i|\mathcal{D}_{<i}, \mathcal{H})}$$

Can be estimated via posterior samples for each index i!

Estimating the ML via Training Speed

Estimator for $\log P(\mathcal{D}_i|\mathcal{D}_{<i}, \mathcal{H})$

Can compute a lower bound on the ML using Jensen's inequality using posterior samples θ_j

$$\hat{\mathcal{L}}(\mathcal{D}) = \sum_{i=1}^{n} \frac{1}{k} \sum_{j=1}^{k} \log P(\mathcal{D}_i|\theta_j)$$

Less biased estimator for $\log P(\mathcal{D}_i|\mathcal{D}_{<i}, \mathcal{H})$

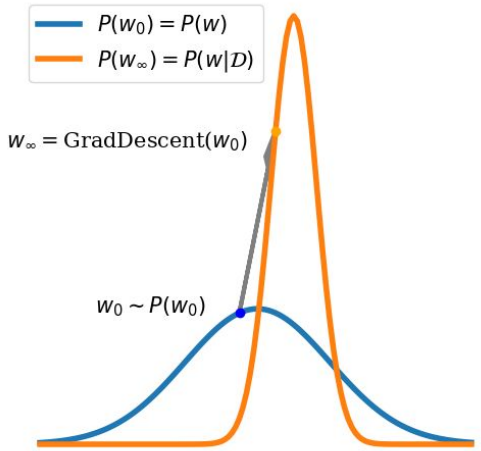
To reduce the gap between LB and ML, average over many samples before applying concave function

$$\hat{\mathcal{L}}_k(\mathcal{D}) = \sum_{i=1}^{n} \log \frac{1}{k} \sum_{j=1}^{k} P(\mathcal{D}_i|\theta_j)$$

Estimating the ML via Gradient Descent

Matthews et al. and Osband et al. both show how to get posterior samples out of linear models by following gradient descent.

This means we can follow an iterative GD procedure and use a subset of the model's training losses to estimate its marginal likelihood!



Algorithm

Algorithm 1: Marginal Likelihood Estimation for Linear Models

Input: A dataset $\mathcal{D} = (x_i, y_i)_{i=1}^n$, parameters $\mu_0, \sigma_0^2, \sigma_N^2$
Result: An estimate of $\mathcal{L}(\mathcal{D})$

$\theta_t \leftarrow \theta_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$; $\quad \tilde{Y} \leftarrow Y + \epsilon \sim \mathcal{N}(0, \sigma_N^2)$; $\quad$ sumLoss $\leftarrow 0$;

$\ell(\mathcal{D}_{\leq i}, w) \leftarrow \|\tilde{Y}_{\leq i} - \theta^\top X_{\leq i}\|_2^2 + \frac{\sigma_N^2}{\theta_0^2}\|\theta - \theta_0\|_2^2$;

for $\mathcal{D}_i \in \mathcal{D}$ **do**

$\quad$ sumLoss $=$ sumLoss $+ \frac{(\theta_t^\top x_i - y_i)^2}{2\sigma_N^2}$;

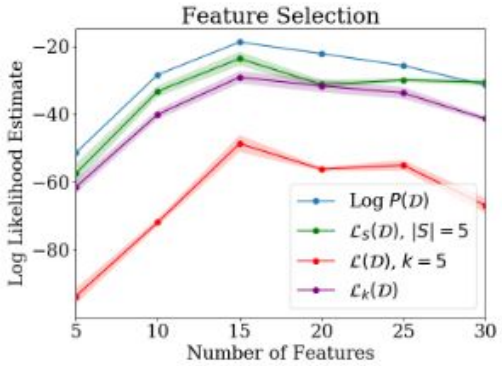
$\quad \theta_t \leftarrow$ GradientDescent$(\ell, \theta_t, \mathcal{D}_{\leq i})$;

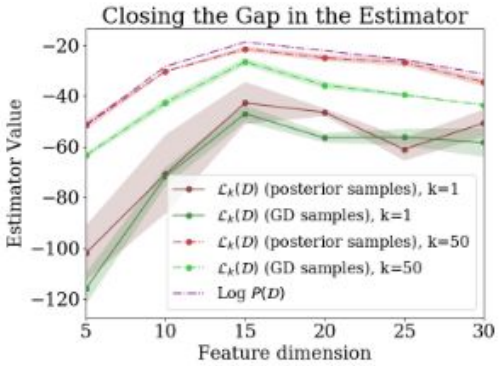
end

return sumLoss

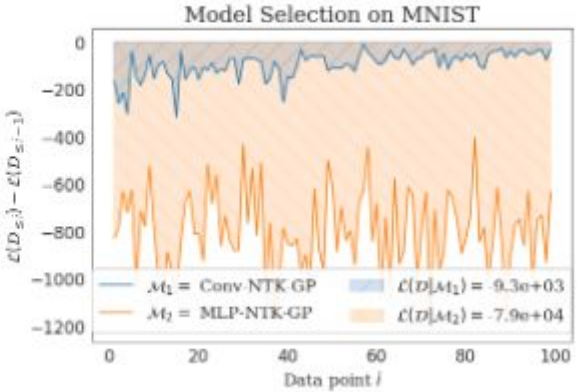
Results

Posterior sampling estimator correlates reasonably well with real log ML on some interpretable model selection problems





This approach also allows us to obtain marginal likelihood estimates for infinite-width versions of neural network architectures



(finite width)

What about neural networks?

Measuring Training Speed in DNNs

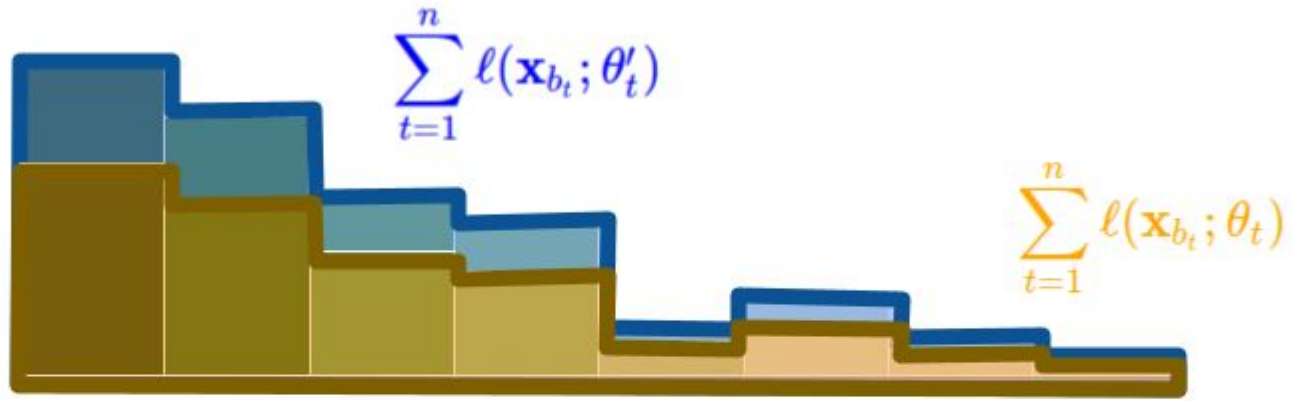
What happens if instead of summing over data points, we sum training losses over optimization steps in an SGD trajectory?

$$\log P(\mathcal{D}|\mathcal{H}) = \sum_{i=1}^{n} \overbrace{\log P(\mathcal{D}_i|\mathcal{D}_{<i}, \mathcal{H})}^{\text{sum of log likelihoods}}$$

$$\text{TSE} = \sum_{t=1}^{T} \left[\frac{1}{B} \sum_{i=1}^{B} \overbrace{\ell\left(f_{\theta_{t,i}}(\mathbf{X}_i), \mathbf{y}_i\right)}^{\text{sum of training losses}} \right]$$

$$\text{TSE-EMA} = \sum_{t=1}^{T} \gamma^{T-t} \left[\frac{1}{B} \sum_{i=1}^{B} \ell\left(f_{\theta_{t,i}}(\mathbf{X}_i), \mathbf{y}_i\right) \right]$$

Training Speed and AUC



How to evaluate a model

Problem: want to find the best architecture for a problem, but don't want to train the set of all possible NNs to find the best one.

Need a cheap way of **estimating** how good an architecture will be.

Ranking vs Prediction

Note: a good performance estimator lets us **rank** models

It doesn't necessarily need to predict the final trained performance! For neural architecture search, we just need to know which architectures we should invest in training.

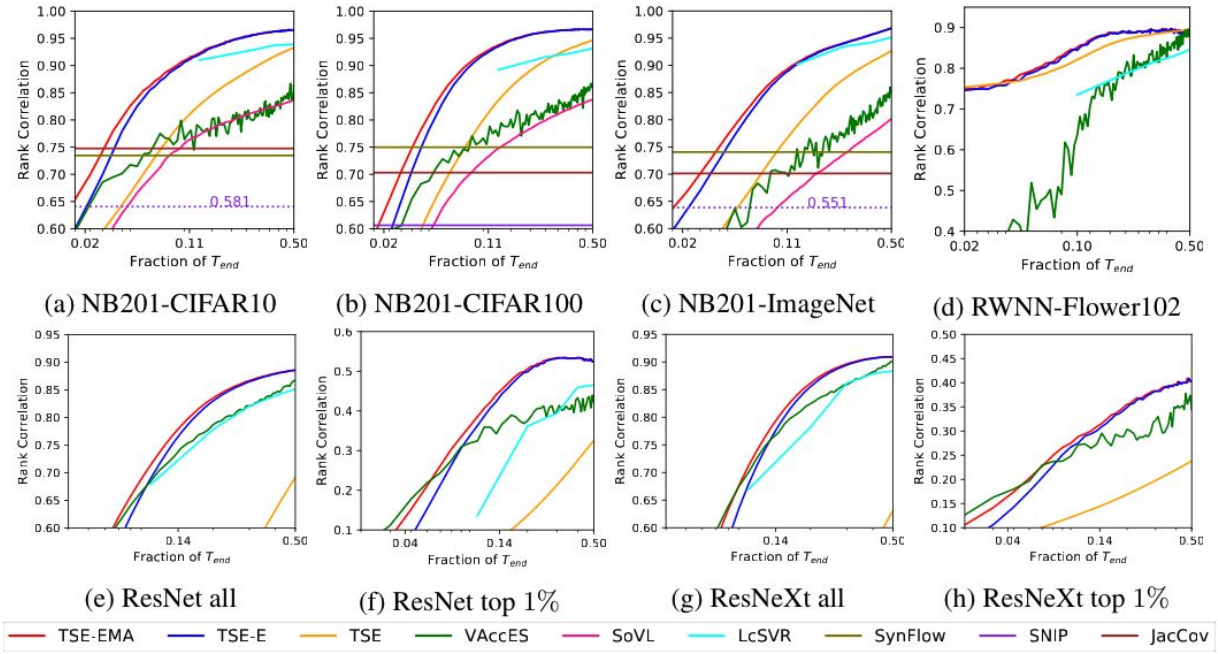
We therefore focus on **rank correlation** between a performance estimator evaluated early in training, and the final trained model's performance.

Ranking of model's final performance

Ranking of model's performance estimator value

$$r_s = \rho_{\mathrm{R}(X),\mathrm{R}(Y)} = \frac{\mathrm{cov}(\mathrm{R}(X),\mathrm{R}(Y))}{\sigma_{\mathrm{R}(X)}\sigma_{\mathrm{R}(Y)}}$$

Rank Correlation with Generalization



Performance vs Validation Accuracy

B	Estimator	Rank Correlation				Average Accuracy of Top 10 Architectures			
		NB201-CIFAR10			DARTS	NB201-CIFAR10			DARTS
		RandNAS	FairNAS	MultiPaths	RandNAS	RandNAS	FairNAS	MultiPaths	RandNAS
100	TSE	**0.70 (0.02)**	**0.84 (0.01)**	**0.83 (0.01)**	**0.30(0.04)**	**92.67 (0.12**	**92.7 (0.1)**	**92.63 (0.12)**	**93.64(0.04)**
	Val Acc	0.44 (0.15)	0.56 (0.17)	0.67 (0.05)	0.11(0.04)	91.47 (0.31)	91.73 (0.21)	91.77 (0.78)	93.20(0.04)
200	TSE	**0.70 (0.03)**	**0.850 (0.01)**	**0.83 (0.01)**	**0.32(0.04)**	**92.70 (0.00)**	**92.77 (0.06)**	**92.73 (0.06)**	**93.55(0.04)**
	Val Acc	0.41 (0.10)	0.56 (0.17)	0.53 (0.11)	0.09(0.02)	91.53 (0.55)	92.40 (0.10)	92.23 (0.23)	93.34(0.02)
300	TSE	**0.71 (0.03)**	**0.851 (0.00)**	**0.82 (0.01)**	**0.34(0.04)**	**92.70 (0.00)**	**92.77 (0.06)**	**92.70 (0.00)**	**93.65(0.04)**
	Val Acc	0.44 (0.04)	0.62 (0.08)	0.59 (0.71)	0.06(0.02)	91.20 (0.35)	92.10 (0.50)	91.43 (0.72)	93.31(0.02)

Results of performance estimators in one-shot NAS setting over 3 supernetwork training initialisations. For each supernetwork, we randomly sample 500 random subnetworks for DARTS and 200 for NB201, and compute their TSE, Val Acc after inheriting the supernetwork weights and training for B additional mini-batches. Rank correlation measures the estimators' correlation with the rankings of the true test accuracies of subnetworks when *trained from scratch independently*, and we compute the average test accuracy of the top 10 architectures identified by different estimators from all the randomly sampled subnetworks.

Differentiable NAS Search Spaces

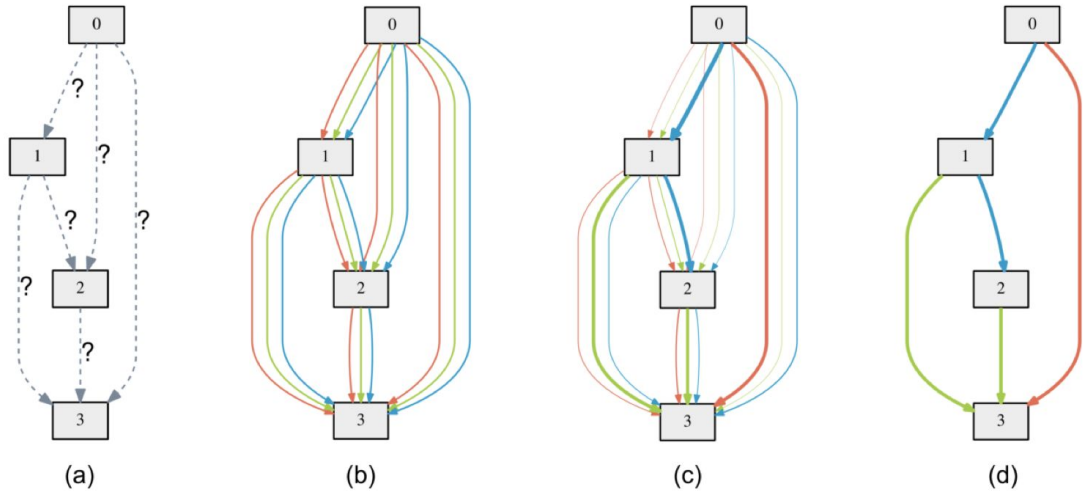


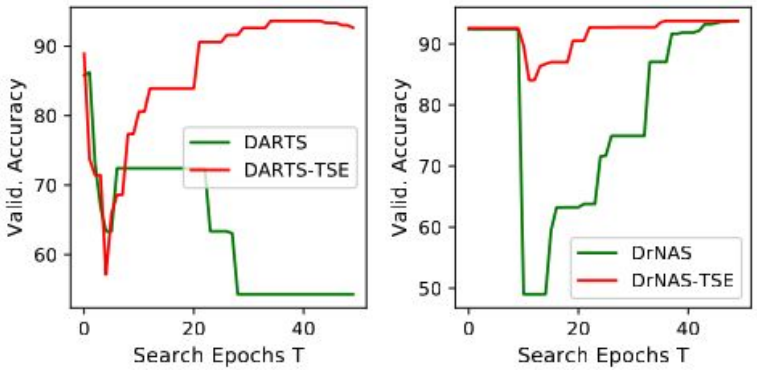
Figure 1: An overview of DARTS: (a) Operations on the edges are initially unknown. (b) Continuous relaxation of the search space by placing a mixture of candidate operations on each edge. (c) Joint optimization of the mixing probabilities and the network weights by solving a bilevel optimization problem. (d) Inducing the final architecture from the learned mixing probabilities.

Node = latent representation

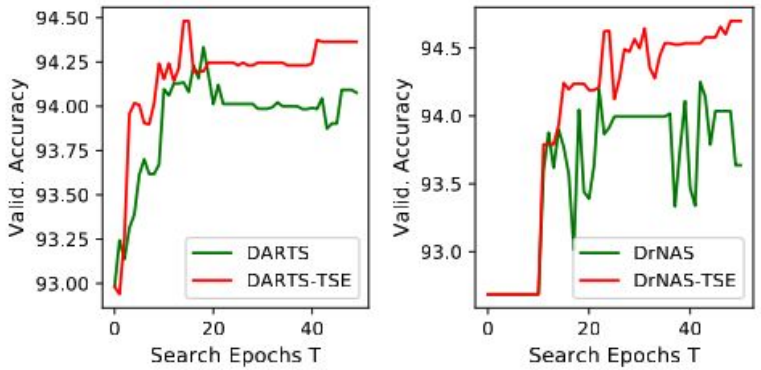
Edge = operation, e.g. on CIFAR-10 :

{3 × 3 and 5 × 5 separable convolutions, 3 × 3 and 5 × 5 dilated separable convolutions, 3 × 3 max pooling, 3 × 3 average pooling, identity, and zero}

Differentiable NAS Performance



(a) NB201-CIFAR10



(b) DARTS-NB301

Model Selection: Conclusions

- In some settings, training speed can be used to estimate the marginal likelihood of a Bayesian model
- Even in non-Bayesian settings, training speed seems to be highly predictive of final generalization performance -- better than ES validation!
  - Needs fixed hyperparameter set
  - Needs fixed data generating distribution (no data augmentation)
  - Assumes fixed training procedure (can’t compare dropout or BN vs no dropout)

Generalizing Out-of-Distribution

Credits

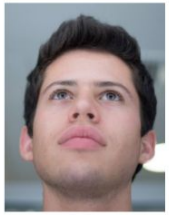
Amy Zhang

Shagun Sodhani

Angelos Filos

Marta Kwiatkowska

Joelle Pineau

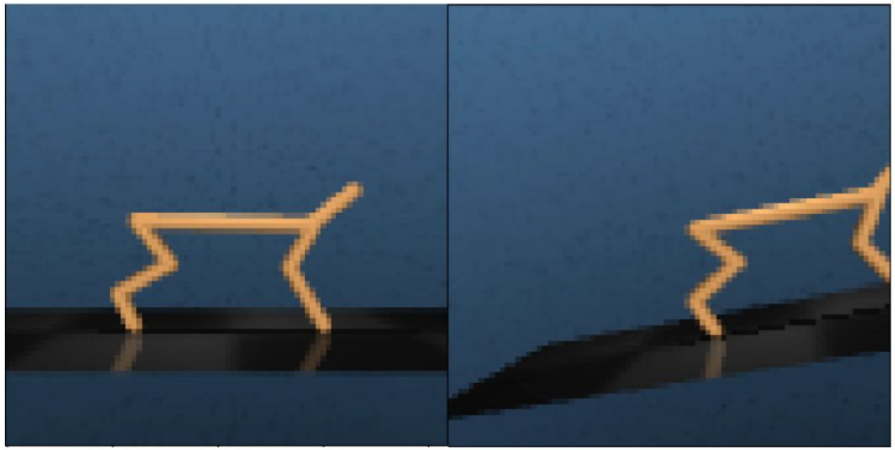


Yarin Gal

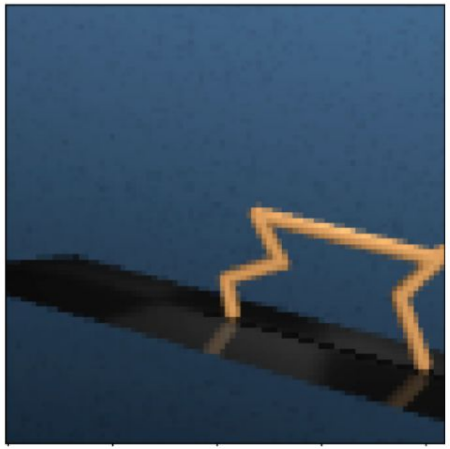
Doina Precup

"Invariant Causal Prediction for Block MDPs", ICML 2020

Problem Setting



Train RL agent on environments $\mathcal{E}_1$, $\mathcal{E}_2$



Deploy RL agent on environment $\mathcal{E}_3$

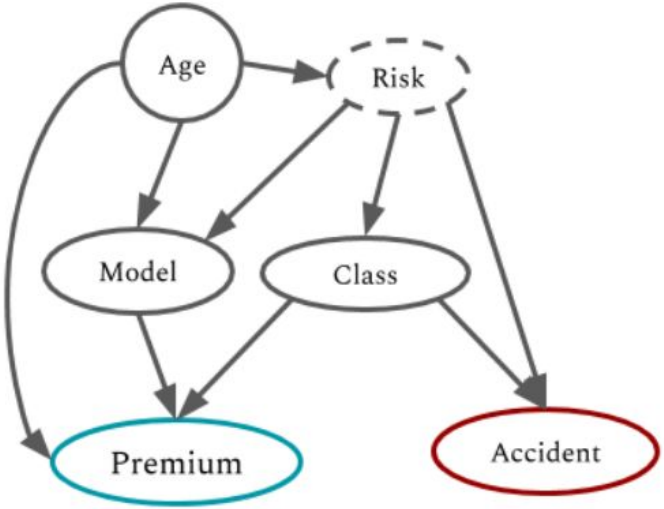
Summary

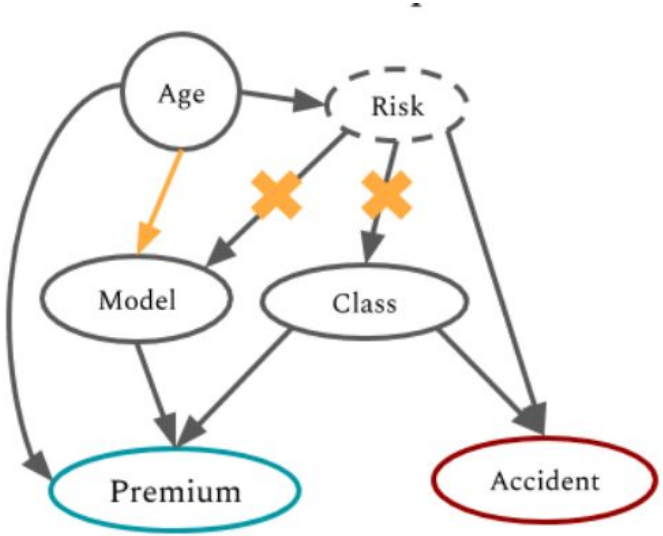
1. Want RL agents that generalize to new environments where **the underlying MDP structure is the same**
2. So we find a *representation* that maps equivalent observations from different environments to the same abstract state
3. We use an idea from the causal inference literature called **invariant prediction** to find such a representation.
4. This extra structure makes the problem easier than meta- or transfer-learning and gives us nicer guarantees.

Formalizing OOD Generalization

Want to formalize **shared structure** between training and test distributions

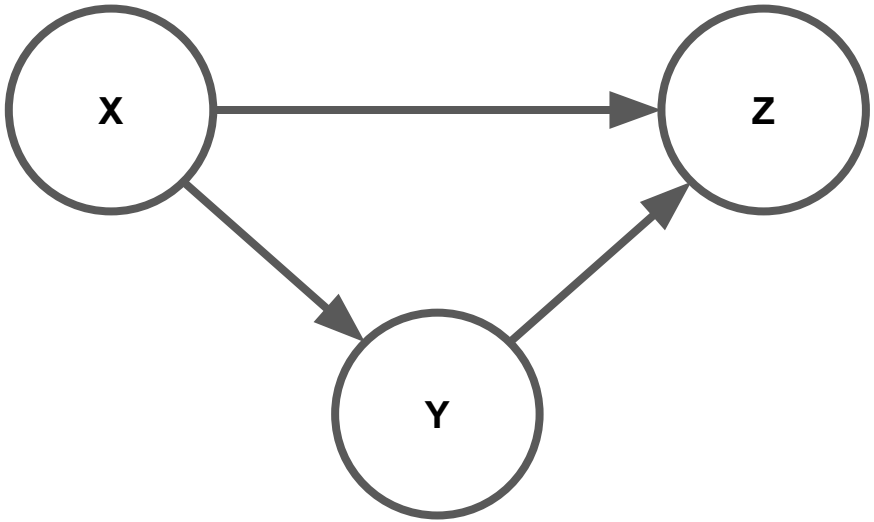
Solution: consider changes induced by interventions on a **causal graph**!





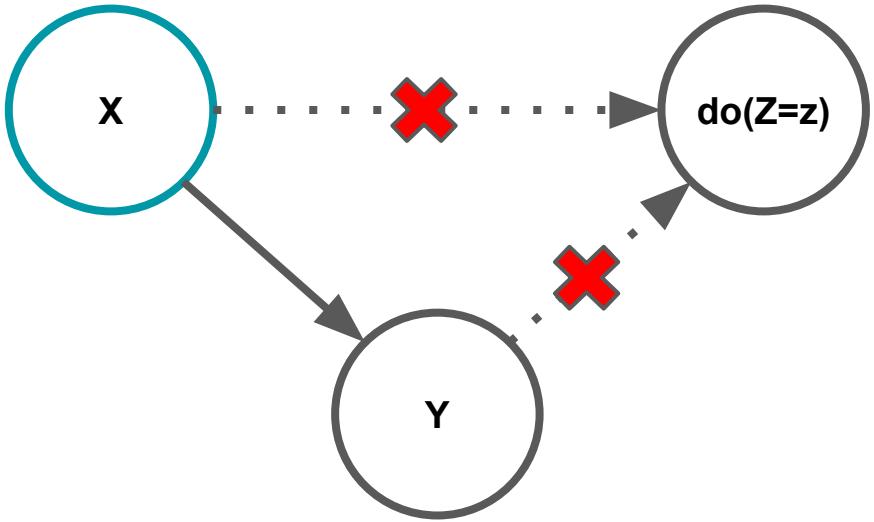
Generalization Under Interventions

Key idea: generalizing to new environments is easy if the predictor learns to depend on causal ancestors of the target.



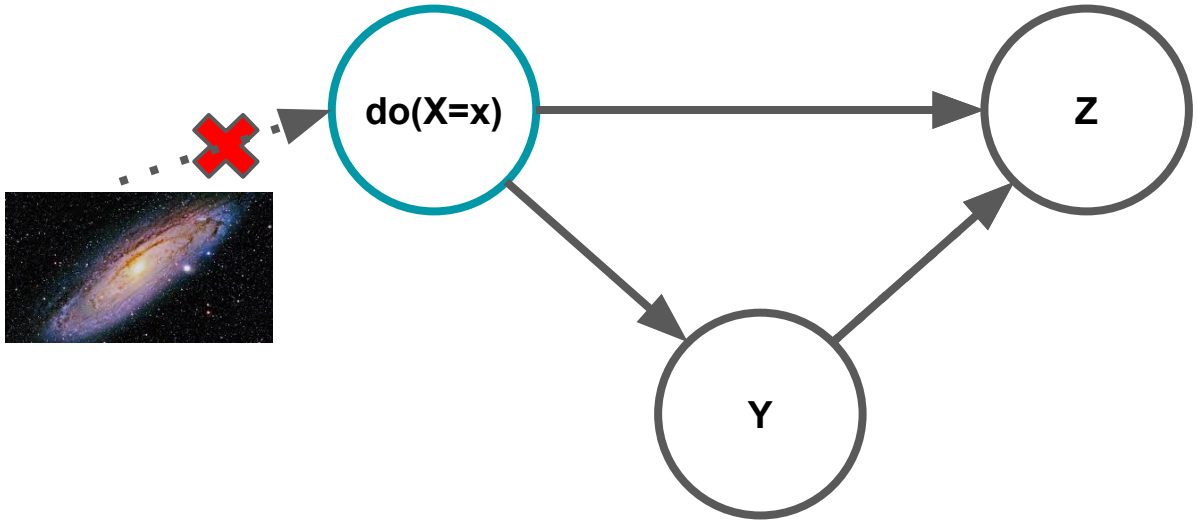
Generalization Under Interventions

Key idea: generalizing to new environments is easy if the predictor learns to depend on causal ancestors of the target.



Generalization Under Interventions

Key idea: generalizing to new environments is easy if the predictor learns to depend on causal ancestors of the target.



Prior Work: Causal Inference & Invariant Prediction

Predictors which depend on causal parents of the return will satisfy certain invariance properties when evaluated under different interventions (environments).

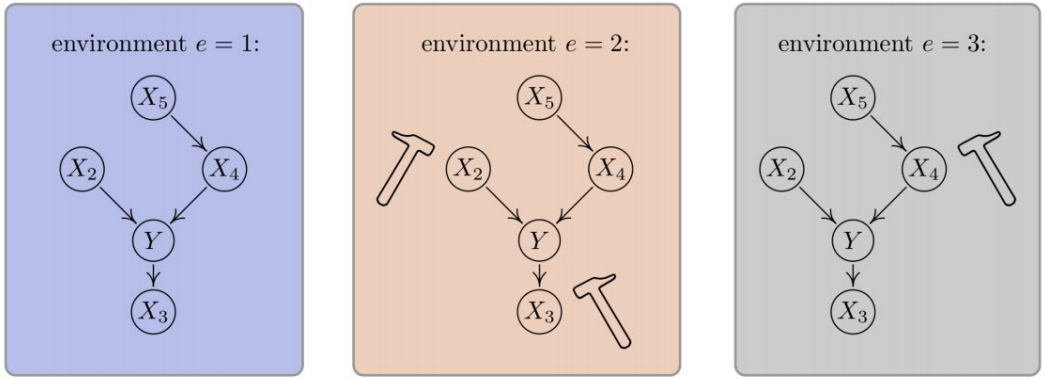
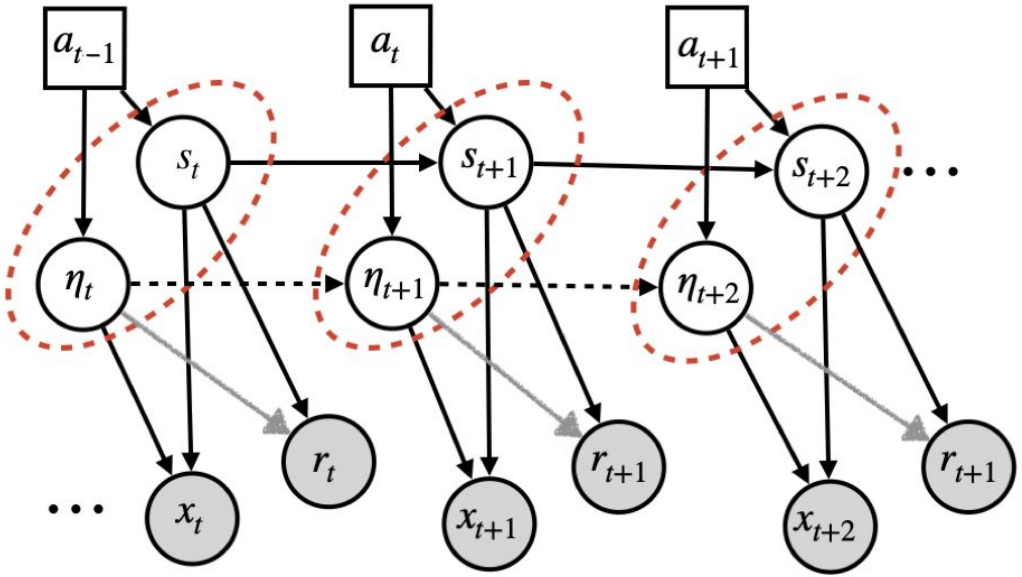


Figure 1: An example including three environments. The invariance (1) and (2) holds if we consider $S^* = \{X_2, X_4\}$. Considering indirect causes instead of direct ones (e.g. $\{X_2, X_5\}$) or an incomplete set of direct causes (e.g. $\{X_4\}$) may not be sufficient to guarantee invariant prediction.

Peters et al., 2016

Formalizing Causal Structure in MDPs

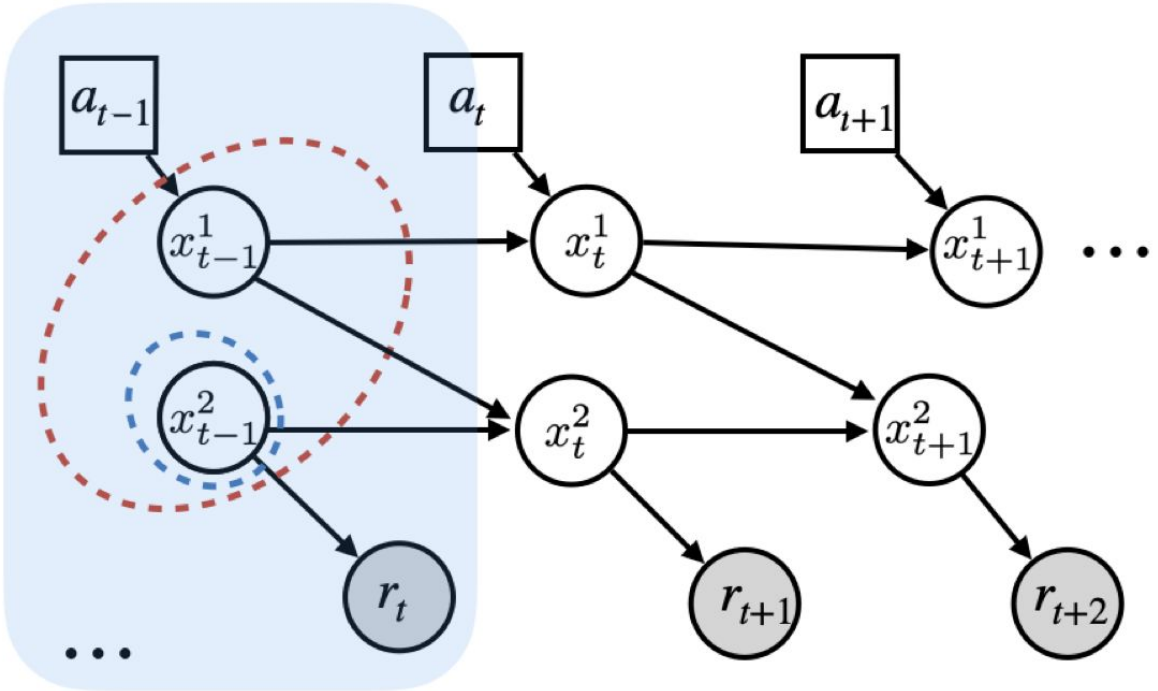
We'll be interested in reinforcement learning, where an agent interacts with an environment (MDP) M so as to maximize cumulative reward.



Dealing with multiple timesteps

In RL, we're interested in the sum of rewards over time.

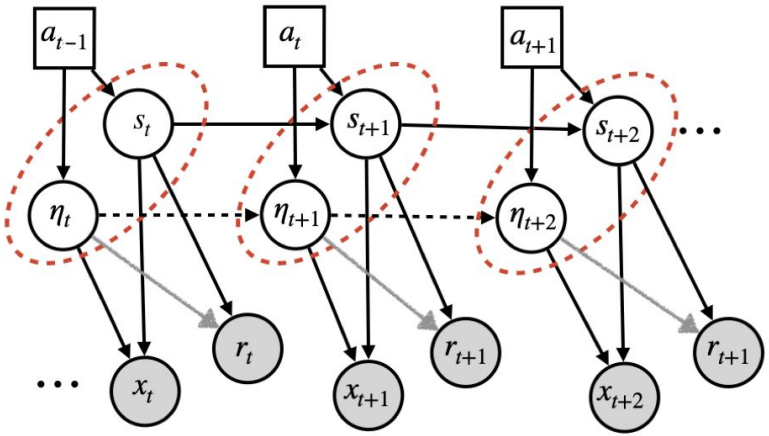
It's not sufficient to look at just the causal parents of the reward when making predictions -- we also need to look at the parents, and those parents' parents, and so on.



Assumptions

Assumption 1: the training and test environments we consider satisfy the block MDP property (i.e. they share “equivalent structure”)

Assumption 2: The training environments correspond to interventions on variables in the graphical model that defines the MDP transition dynamics



State Abstractions

State abstractions are functions of the form

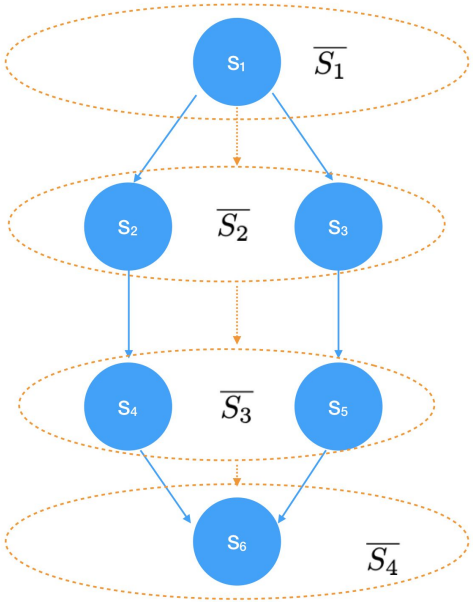
$$\phi : \mathcal{S} \to \bar{\mathcal{S}}$$

which map the state space to a simpler abstract state space to make it easier for an agent to learn and plan.

A model irrelevance state abstraction satisfies the following consistency properties:

$$\phi(s) = \phi(s') \implies R(s) = R(s')$$

$$\text{and} \sum_{s'' \in \phi^{-1}(\bar{s}')} p(s''|s) = \sum_{s'' \in \phi^{-1}(\bar{s}')} p(s''|s')$$



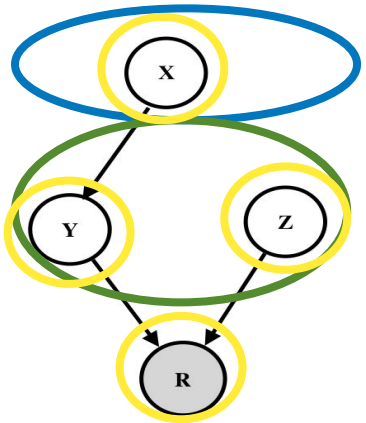
Causal Structure and State Abstractions

Theorem 1

Let $S_R \subseteq \{1, \ldots, k\}$ be the set of variables such that the reward $R(x, a)$ is a function only of $[x]_{S_R}$ (x restricted to the indices in S_R). Then let $S = \mathbf{AN}(R)$ denote the ancestors of S_R in the (fully observable) causal graph corresponding to the transition dynamics of $M_{\mathcal{E}}$. Then the state abstraction $\phi_S(x) = [x]_S$ is a *model-irrelevance* abstraction for every $e \in \mathcal{E}$.

Linear Setting

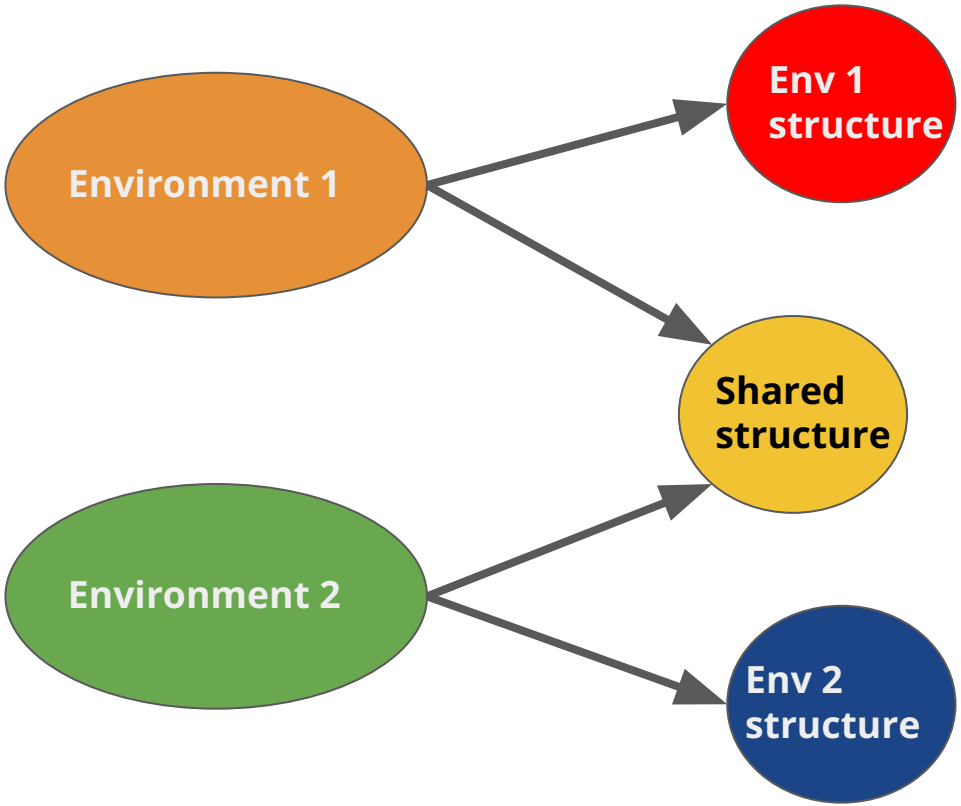
Idea: iteratively apply the ICP algorithm (Peters et al., 2016) to identify the causal ancestors of the reward.



Algorithm 1 Linear MISA: Model-irrelevance State Abstractions

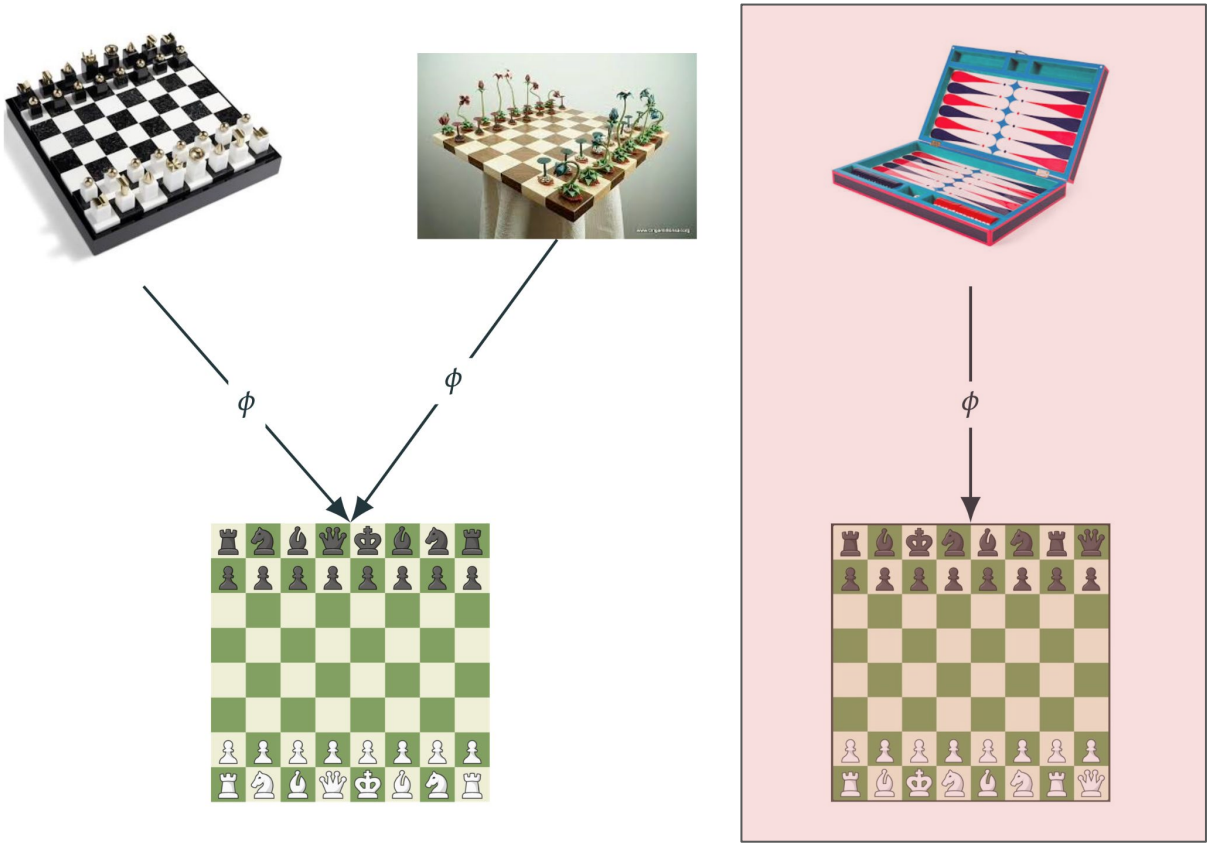
Result: $S \subset \{1, \ldots, k\}$, the causal state variables
Input: α, a confidence parameter, $\mathcal{D}$, an replay buffer with observations $\mathcal{X}$.
$S \leftarrow \emptyset$
stack $\leftarrow$ r
while *stack is not empty* **do**
  v = stack.pop()
  if $v \notin S$ **then**
    $S' \leftarrow$ `ICP`(v, $\mathcal{D}$, $\frac{\alpha}{\dim(\mathcal{X})}$)
    $S \leftarrow S \cup S'$
    stack.push(S')
  end
end

Rich observations: intuition



Want a representation that only depends on the **shared** structure between environments

State Abstractions with Rich Observations



Only care about generalization to “reasonable” inputs

Method

Minimize MI between representation and environment

$\phi : \mathcal{X} \mapsto \mathcal{Z}$

Invariant Encoder

Task-Specific Encoder

$\psi : \mathcal{X} \mapsto \mathcal{H}$

Unique encoder for each task

$r : \mathcal{Z} \times \mathcal{A} \times \mathcal{Z} \mapsto \mathbb{R}$

Invariant Reward Model

$$J_R(\phi, R) = \sum \mathbb{E}_{\pi_{b_i}} \left[(R(\phi(x_i), a, \phi(x_i')) - r_i')^2 \right]$$

Invariant representation contains reward information

a

Invariant Model

$f_s : \mathcal{A} \times \mathcal{Z} \mapsto \mathcal{Z}$

Task-Specific Model

a

$f_\eta : \mathcal{A} \times \mathcal{H} \mapsto \mathcal{H}$

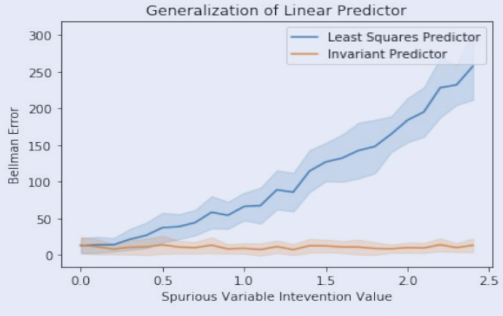
$\phi^{-1} : \mathcal{Z} \times \mathcal{H} \mapsto \mathcal{X}$

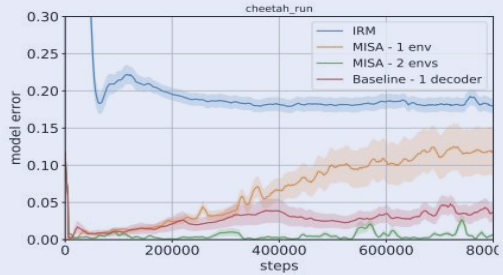
Decoder

$$J_D(\phi, \psi, f_s, f_\eta) = \sum_i \mathbb{E}_{\pi_{b_i}} \left[(\phi^{-1}(f_s(a, \phi(x_i)), f_\eta(a, \psi(x_i))) - x_i')^2 \right]$$

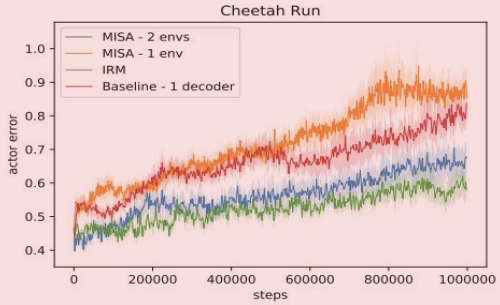
Ground representations using transition loss

Results

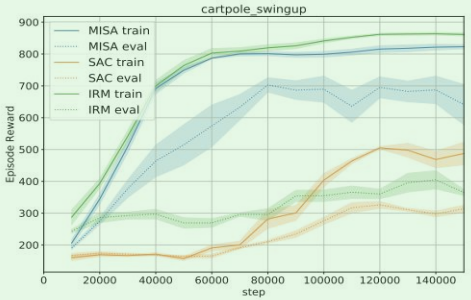




Model Learning



Imitation Learning



Reinforcement Learning

Conclusions

Key take-aways

- Identifying the right inductive bias for a set of tasks can be made arbitrarily challenging, but the problem is tractable for a rich class of problems defined by causal manipulations to the data-generating process.

- Two simple ideas, invariance and training speed, provide surprisingly powerful tools with theoretical guarantees for a wide range of model selection problems.

- Scalable methods inspired by these tools lead to improved generalization in high-dimensional input spaces with complex dynamics.

Food for thought: invariance

- When we explicitly enforce invariance, we encourage the representation to identify when different inputs are equivalent
- Implicitly, training speed is also measuring this equivalence but at a the “gradient update level”
  - If all inputs map to the same gradient update, then the training loss will decrease rapidly because every gradient step lowers the loss of the entire training set
- Is there a unifying framework under which these phenomena are two sides of the same coin?

Thanks!