Meta-Learning

— an idiosyncratic tutorial

Yee Whye Teh
Statistics @ Oxford
DeepMind
http://csml.stats.ox.ac.uk/people/teh/

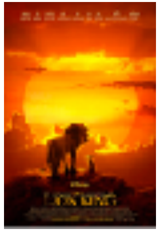
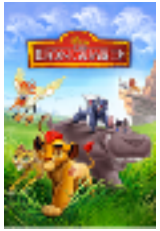
Small Data Problems

- Few Shot Learning

Small Data Problems

- Recommender Systems


?


?

Small Data Problems

- Robotics

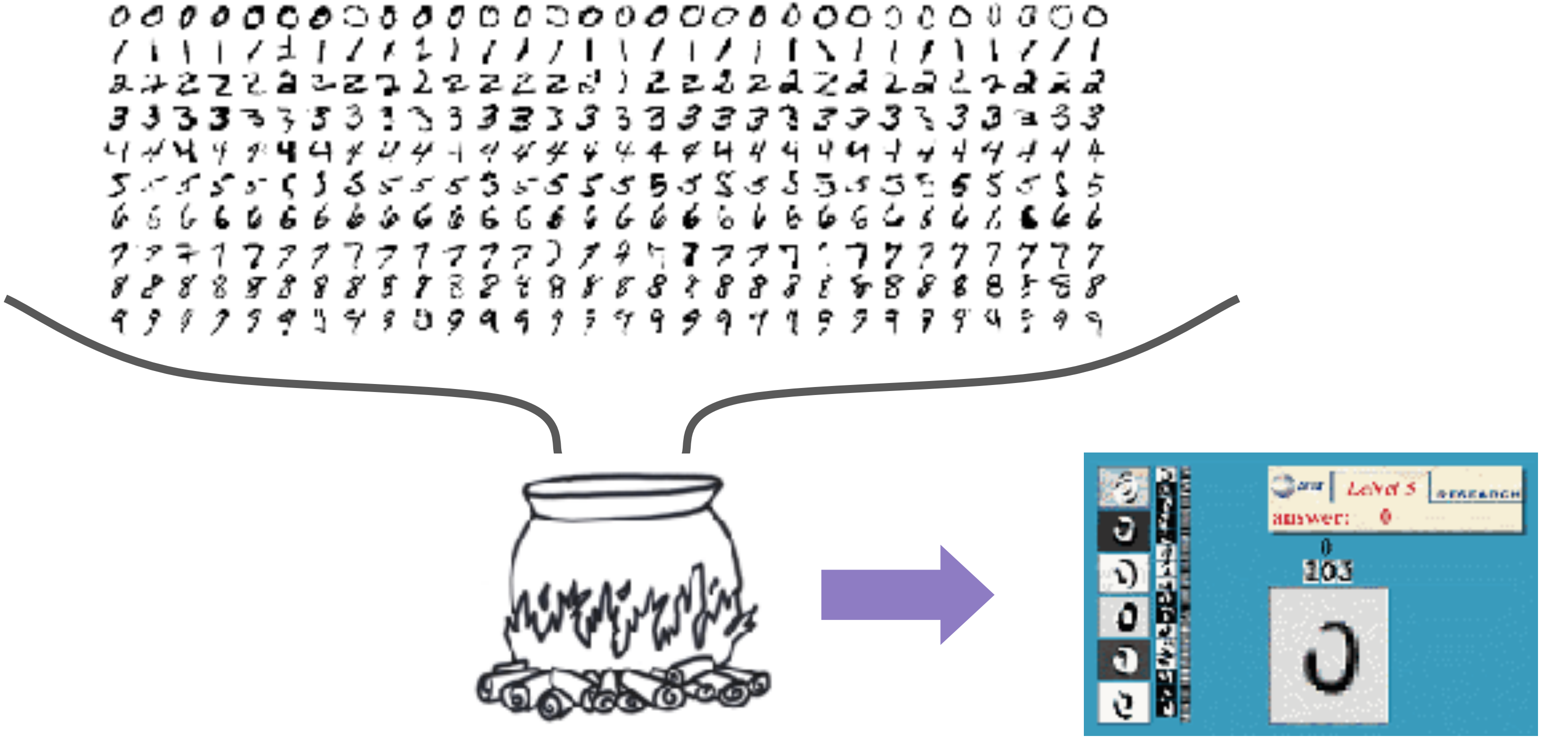
Credit: Sadeghi et al CVPR 2018

Small Data Problems

- Artificial General Intelligence

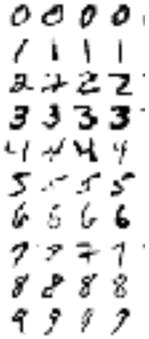
Credit: Getty Images, BBC

Machine Learning with Big Data



Credit: Yann LeCun, MNIST

Machine Learning with Small Data



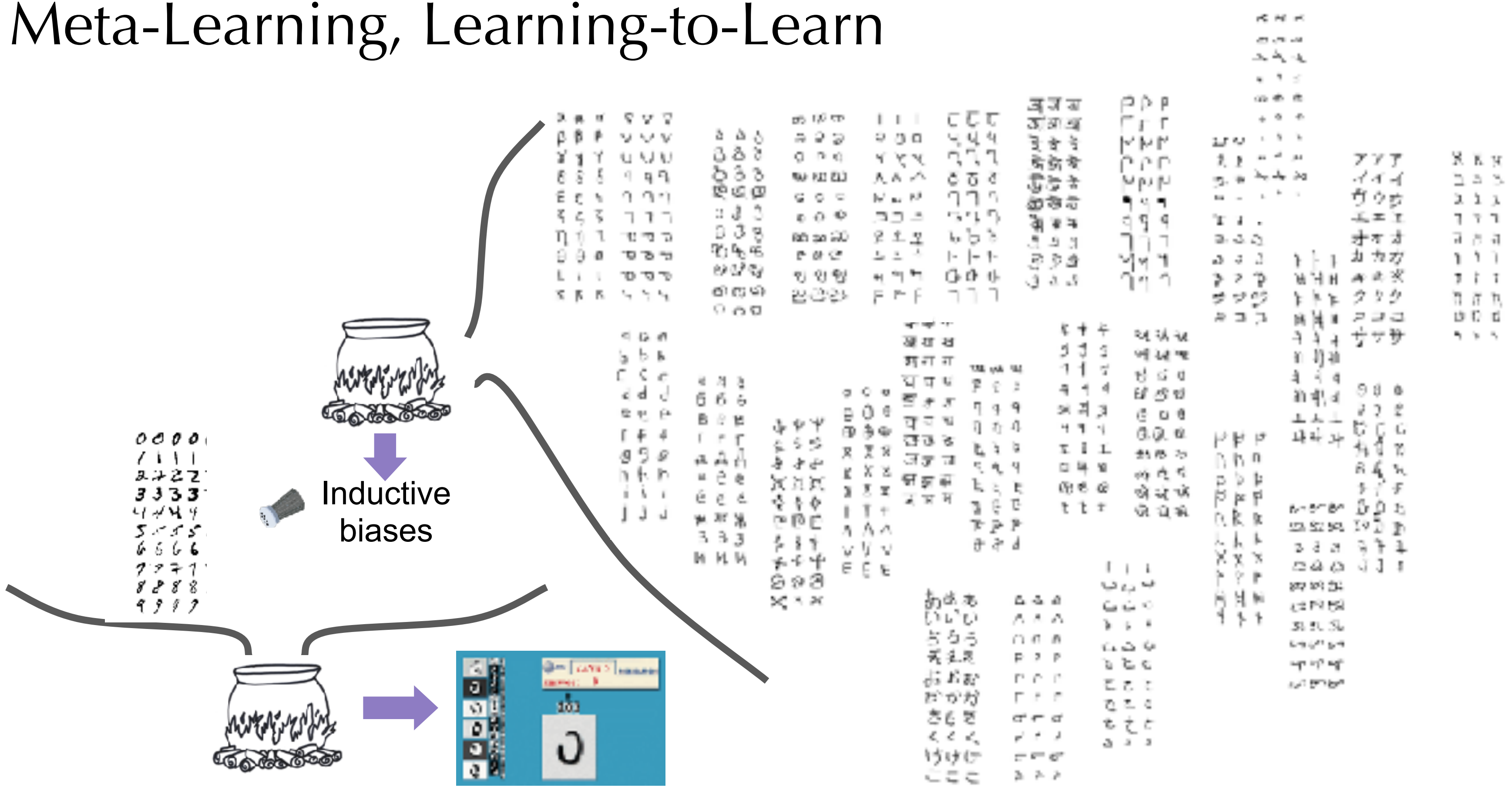
Inductive biases

- Preprocessing
- Feature engineering
- Data augmentation
- Model architectures
- Regularisation

Yee Whye Teh

Credit: Yann LeCun, MNIST

Meta-Learning, Learning-to-Learn



Meta-Learning: an idiosyncratic tutorial

- Optimisation perspective on meta-learning
  - Optimisation-based meta-learning
  - Black-box meta-learning

- Probabilistic perspective on meta-learning
  - Stochastic processes
  - Neural processes
  - Uncertainty in meta-learning

- Probabilistic symmetries and neural architectures

- Note: no meta reinforcement learning (meta-RL)

Meta-Learning: an idiosyncratic tutorial

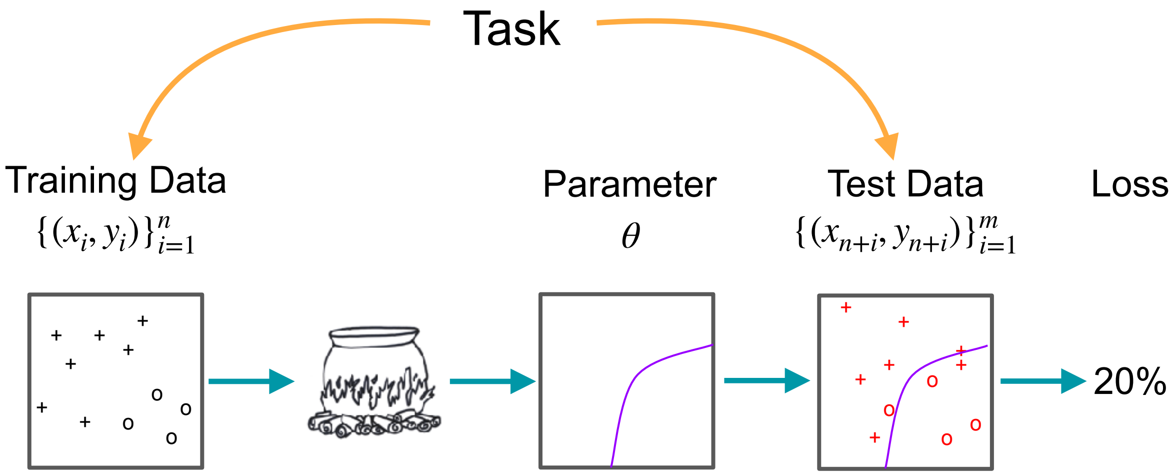
- **Optimisation perspective on meta-learning**
  - Optimisation-based meta-learning
  - Black-box meta-learning

- Probabilistic perspective on meta-learning
  - Stochastic processes
  - Neural processes
  - Uncertainty in meta-learning

- Probabilistic symmetries and neural architectures

- Note: no meta reinforcement learning (meta-RL)

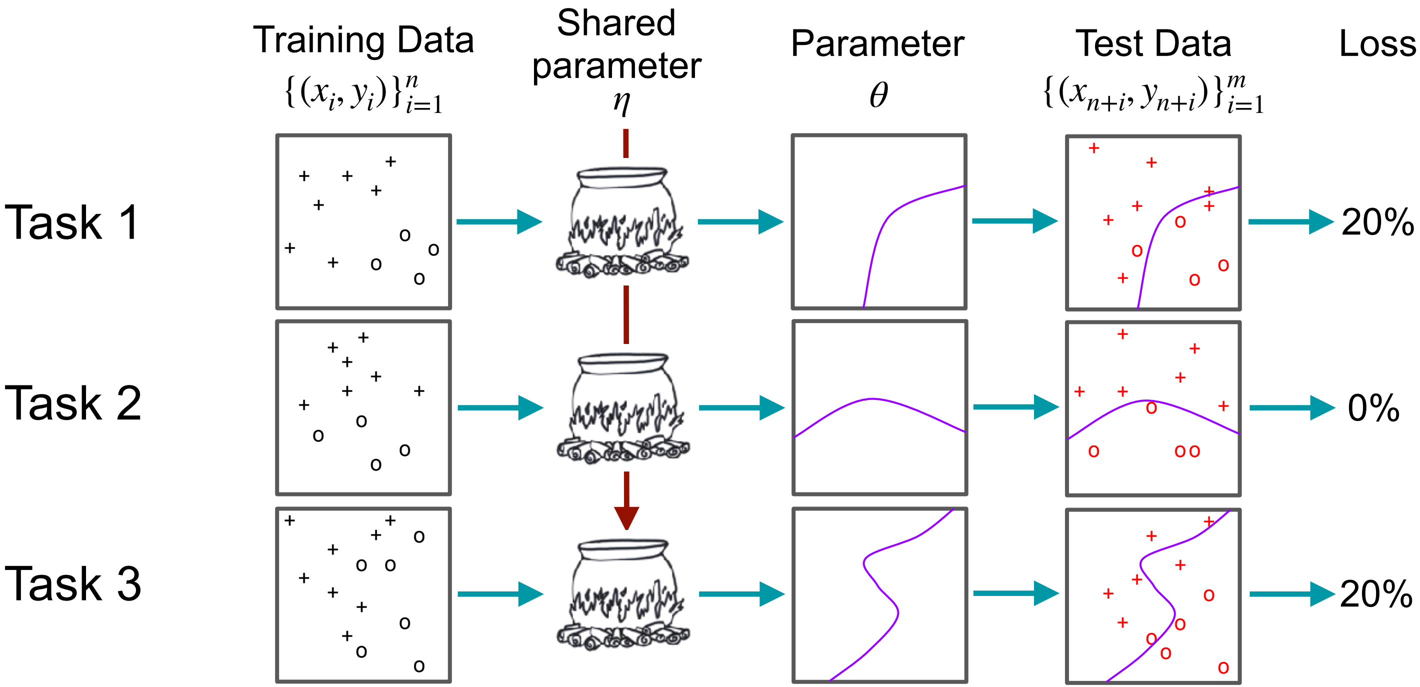
Single-Task Learning



Empirical Risk Minimisation

$$\arg\min_{\theta} \sum_{i=1}^{n} \mathrm{Loss}(f_{\eta,\theta}(x_i), y_i) + \mathrm{Regulariser}(\theta)$$

Multi-Task Learning



$$\arg\min_{\eta, \{\theta_j\}} \sum_{j=1}^{T} \sum_{i=1}^{n} \text{Loss}(f_{\eta,\theta_j}(x_{ji}), y_{ji}) + \text{Regulariser}(\theta_j)$$

Yee Whye Teh

Meta-Learning and Learning-to-Learn

- Is it possible to learn to generalise from training to test data?
- Learn:

$$\theta_j = \arg\min_{\theta_j} \sum_{i=1}^{n} \mathsf{Loss}(f_{\eta,\theta_j}(x_{ji}), y_{ji}) =: \mathsf{Learner}(\eta, \mathsf{TrainData}_j)$$

- Test performance:

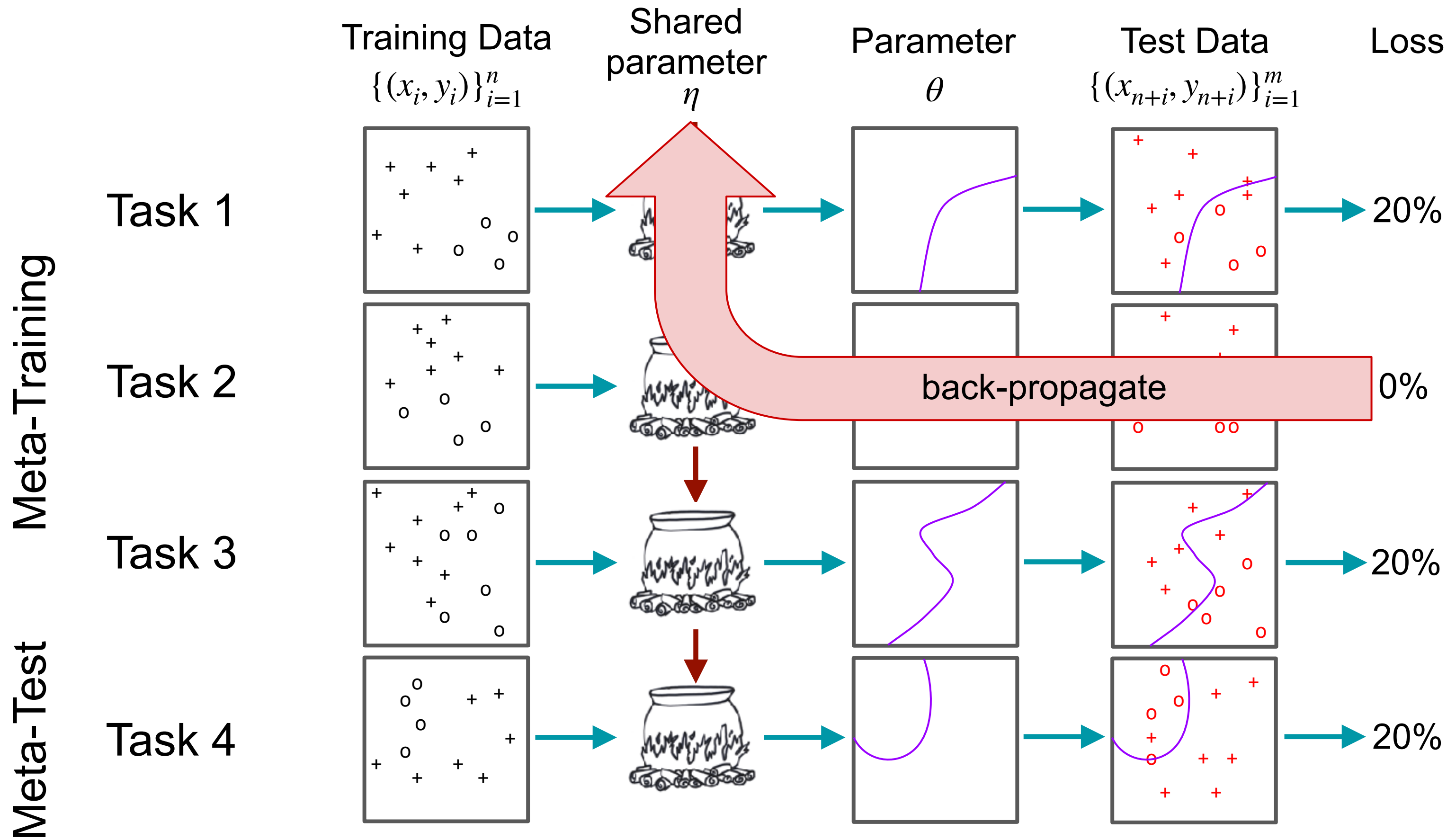
$$\sum_{i=n+1}^{n+m} \mathsf{Loss}(f_{\eta,\theta_j}(x_{ji}), y_{ji}) =: \mathscr{L}(\eta, \theta_j, \mathsf{TestData}_j)$$

- Optimise shared parameters for test performance:

$$\arg\min_{\eta} \sum_{j=1}^{T} \mathscr{L}(\eta, \theta_j, \mathsf{TestData}_j) = \arg\min_{\eta} \sum_{j=1}^{T} \sum_{i=n+1}^{n+m} \mathsf{Loss}(f_{\eta,\mathsf{Learner}(\eta,\mathsf{TrainData}_j)}(x_{ji}), y_{ji})$$

- "Our training procedure is based on a simple machine learning principle: test and train conditions must match" — [Vinyals et al NeurIPS 2016]

Meta-Learning



Yee Whye Teh

Meta-Learning

- For each iteration:
  - Pick (minibatch of) training task j:
  - Base learner:

$$\theta_j = \mathsf{Learner}(\eta, \mathsf{TrainData}_j) = \arg\min_{\theta_j} \sum_{i=1}^{n} L(f_{\eta,\theta_j}(x_{ji}), y_{ji})$$

  - Test performance:

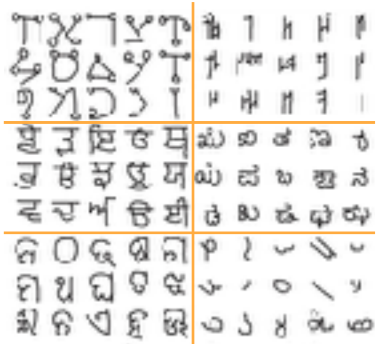
$$\mathscr{L}(\eta, \theta_j, \mathsf{TestData}_j) = \sum_{i=n+1}^{n+m} \mathsf{Loss}(f_{\eta,\theta_j}(x_{ji}), y_{ji})$$

  - Meta-learner: optimise shared parameters for test performance:

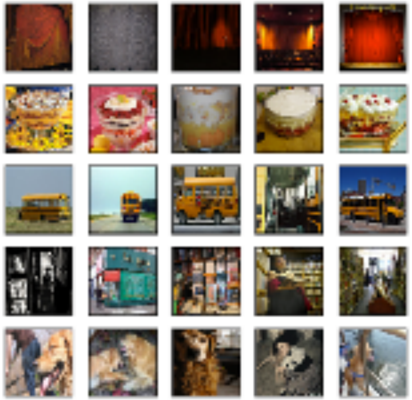
$$\eta \leftarrow \eta - \epsilon \frac{d}{d\eta} \sum_{i=n+1}^{n+m} \mathsf{Loss}(f_{\eta,\mathsf{Learner}(\eta,\mathsf{TrainData}_j)}(x_{ji}), y_{ji})$$

Image Datasets

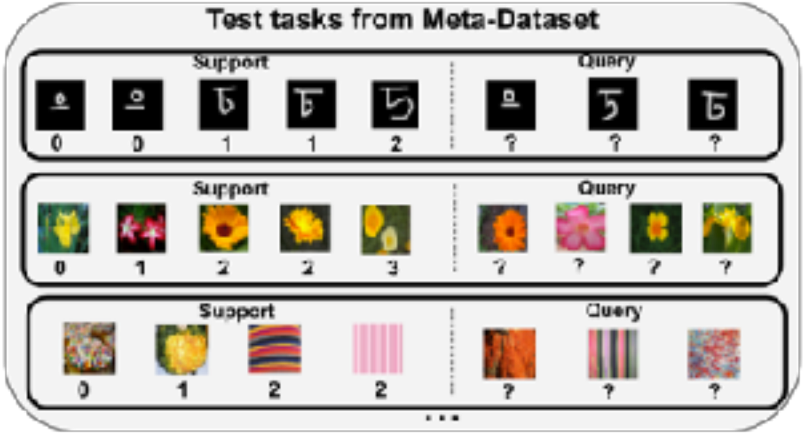
[Omniglot]



[mini-ImageNet]



[Meta Dataset]

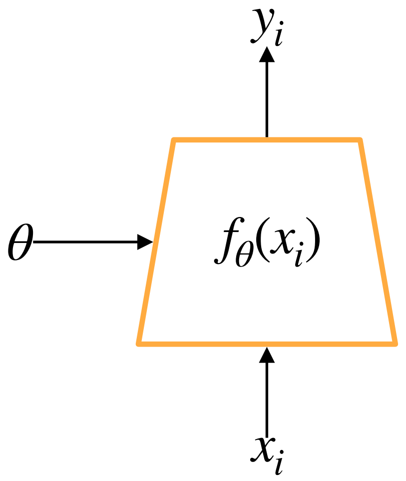


Dataset (other names)
ilsvrc_2012 (ImageNet, ILSVRC) [instructions]
omniglot [instructions]
aircraft (FGVC-Aircraft) [instructions]
cu_birds (Birds, CUB-200-2011) [instructions]
dtd (Describable Textures, DTD) [instructions]
quickdraw (Quick, Draw!) [instructions]
fungi (FGVCx Fungi) [instructions]
vgg_flower (VGG Flower) [instructions]
traffic_sign (Traffic Signs, German Traffic Sign Recognition Benchmark, GTSRB) [instructions]
mscoco (Common Objects in Context, COCO) [instructions]
Total (All datasets)

Different Kinds of Meta-Learning Methods

- Different choices of Learner and function class f lead to different methods.
  - Optimisation-based
  - black-box
  - metric-based
  - memory-based
  - Bayesian…
- Different choices correspond to different **inductive biases**.
  - Psychological theories
  - Symmetries, invariances and equivariances
  - Computational and learnability constraints
  - Intuitions and experimental findings

Optimisation-based Meta-Learning

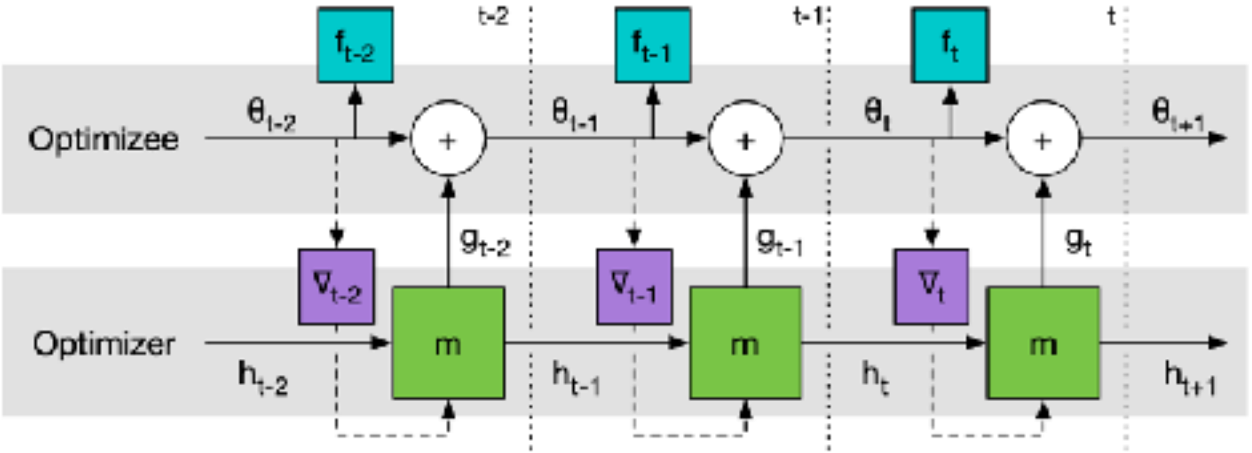


- Optimisation-based base learner.

$$
\begin{aligned}
&\theta_0 \\
&\theta_1 \leftarrow \theta_0 + \mathsf{Update}_\eta(\nabla_{\theta_0} L(f_{\theta_0}(X_1), Y_1)) \\
&\theta_2 \leftarrow \theta_1 + \mathsf{Update}_\eta(\nabla_{\theta_1} L(f_{\theta_1}(X_2), Y_2)) \\
&\vdots
\end{aligned}
$$

- Meta-learn meta-parameters θ_0, η.

Learning to Learn by Gradient Descent by Gradient Descent & LSTM Meta-Learner



Task index j suppressed

Stop gradient

$$f_t = f_{\theta_t}(X_t)$$

$$\nabla_t = \frac{d}{d\theta_t} L(f_t, Y_t)$$

X_t : training inputs in iter t

Y_t : training targets in iter t

ADAM

$$\mu_t = \beta_1 \mu_{t-1} + (1 - \beta_1)\nabla_t$$

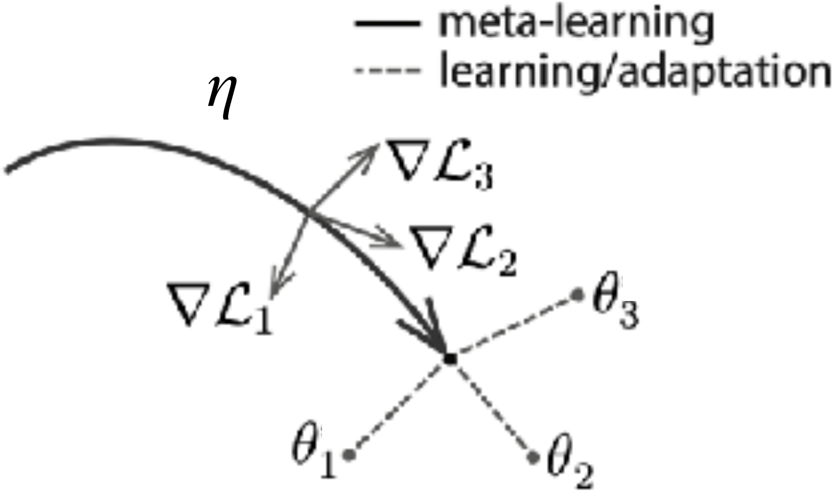
$$V_t = \beta_2 V_{t-1} + (1 - \beta_2)\nabla_t^2$$

$$g_t = -\frac{\epsilon}{\sqrt{V_t} + \delta}\mu_t$$

$$\theta_{t+1} = \theta_t + g_t$$

[Andrychowicz NeurIPS 2016, Ravi & Larochelle ICLR 2017]

Model-Agnostic Meta-Learning (MAML)



Single (or few) step gradient descent learner:

$$\theta_j = \eta - \epsilon \nabla_\eta L(f_\eta(X_j), Y_j)$$

Meta-learn shared initialisation η:

$$\eta \leftarrow \eta - \epsilon_0 \frac{d}{d\eta} \sum_j L(f_{\theta_j}(X_{jt}), Y_{jt})$$

Compute gradient through learner gradient step.

[Finn et al ICML 2017]

Model-Agnostic Meta-Learning (MAML)

$$\theta_j = \eta - \epsilon \nabla_\eta L(f_\eta(X_j), Y_j)$$

$$\begin{aligned}\frac{d}{d\eta} L(f_{\theta_j}(X_{jt}), Y_{jt}) &= \nabla_{\theta_j} L(f_{\theta_j}(X_{jt}), Y_{jt}) \frac{d\theta_j}{d\eta} \\ &= \nabla_{\theta_j} L(f_{\theta_j}(X_{jt}), Y_{jt}) \left(I - \epsilon \nabla^2_\eta L(f_\eta(X_{jt}), Y_{jt}) \right)\end{aligned}$$

- Difficulty is in computing $\nabla_{\theta_j} L(f_{\theta_j}(X_{jt}), Y_{jt}) \nabla^2_\eta L(f_\eta(X_{jt}), Y_{jt})$.
- Vector-Hessian products can be automatically computed in linear time.

[Pearlmutter NECO 1994]

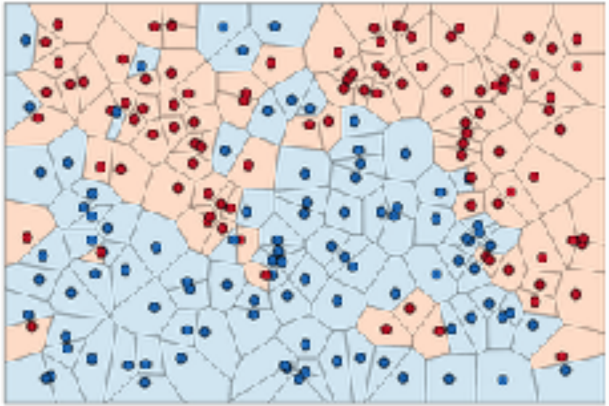
Optimisation-based Meta-Learning

- Two level optimisation problem
  - Flexible and applicable to wide range of neural architectures
  - Positive inductive bias
- Challenges
  - sensitive to neural architectures
  - can be expensive and unstable
- Difficulties of back-propagating through many base learner iterations.
    - If base learner optimisation can be analytically solved, gradients can be computed exactly [Harrison et al WAFC 2018, Lee et al CVPR 2019].
    - Alternatively, use implicit function theorem [Rajeswaran et al NeurIPS 2019].

Black-box Meta-Learning

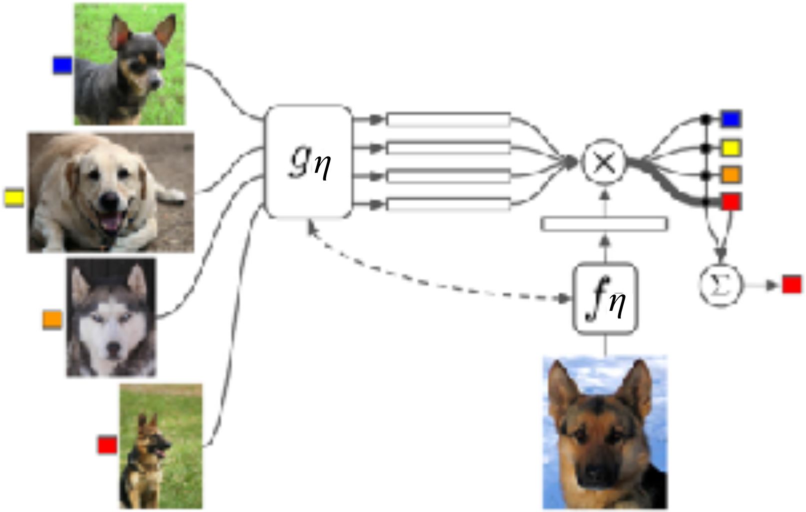
- So far, the base learner is an **optimisation-based** learner.
- Not all learning algorithms are optimisation-based.

- For example, in *k*-nearest neighbour, learner simply "memorises" training data, and matches test inputs with memory to make predictions.



- **Black-box meta-learning**: Implement the base learner using a differentiable programming framework.

Matching Networks



- Embed each train input $x_i \mapsto g_\eta(x_i)$.
- Embed test input $x_i \mapsto f_\eta(x_i)$

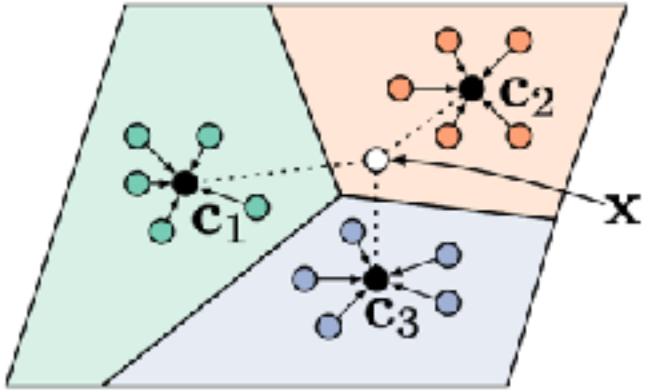
- "Softened" 1-NN classifier:

$$p(\cdot \mid x_{n+j}, \text{TrainData}) = \sum_{i=1}^{n} \frac{e^{g_\eta(x_i)^\top f_\eta(x_{n+j})}}{\sum_{l=1}^{n} e^{g_\eta(x_l)^\top f_\eta(x_{n+j})}} y_i$$

- Memory-based meta-learning
- Metric-based meta-learning

[Vinyals et al NeurIPS 2016]

Prototypical Networks



- Embed each input $x_i \mapsto f_\eta(x_i)$.
- Form "prototypes":

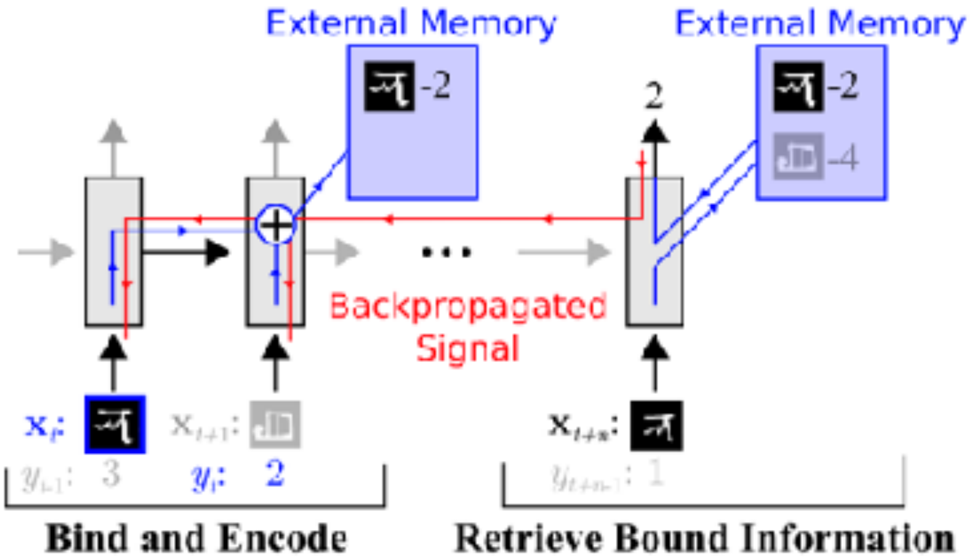
$$c_k = \frac{\sum_{i:y_i=k} f_\eta(x_i)}{\sum_{i:y_i=k} 1}$$

- Predict:

$$p(y_{n_j} = k \,|\, x_{n+j}, \text{TrainData}) = \frac{e^{-\|f_\eta(x_{n+j})-c_k\|^2/\sigma^2}}{\sum_{l=1}^{K} e^{-\|f_\eta(x_{n+j})-c_l\|^2/\sigma^2}}$$

[Snell et al NeurIPS 2017]

Memory-Augmented Neural Networks (MANNs)



Each task: sequence of input/output pairs $(x_1, y_1), (x_2, y_2), \ldots$

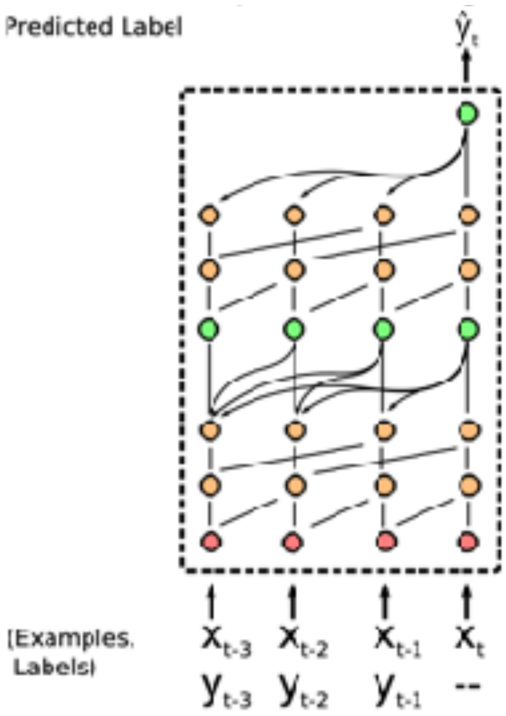
Base learner directly predicts

$$p_\eta(y_t \mid (x_i, y_i)_{i=1}^{t-1}, x_t)$$

Neural Turing machine with external memory.

[Santoro et al ICML 2016]

Simple Neural AttentIve Learner (SNAIL)



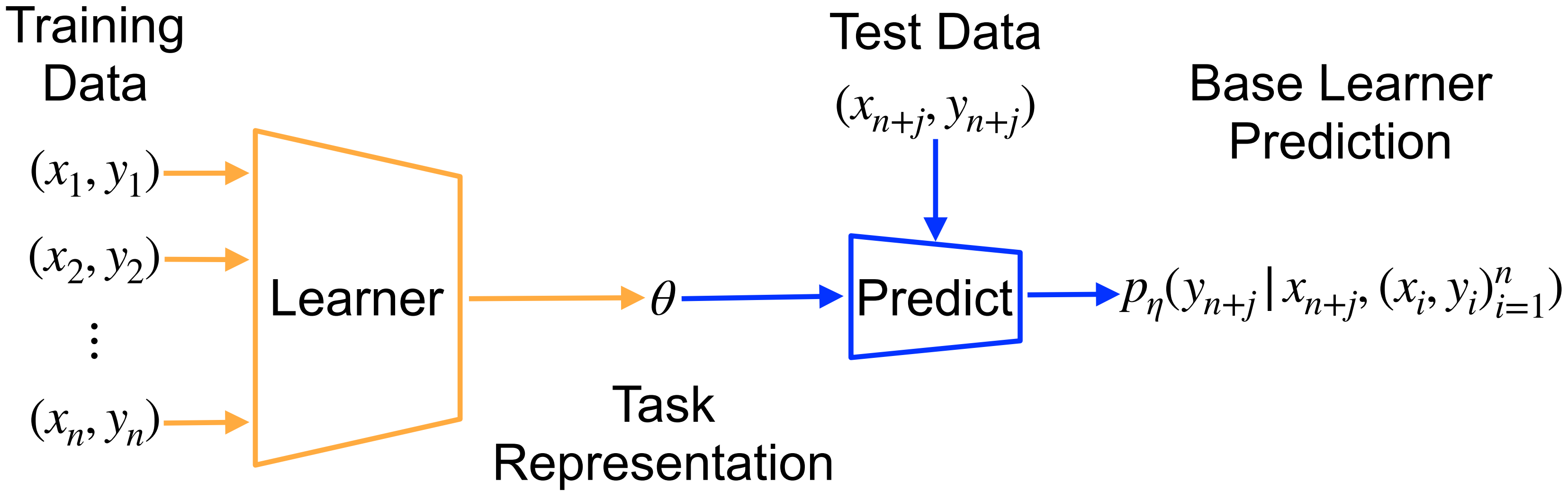
$$p_\eta(y_t \,|\, (x_i, y_i)_{i=1}^{t-1}, x_t)$$

Simple:

- treats base learner problem as sequential prediction
- Use (causal) convolution and attention layers.

[Mishra et al ICLR 2018]

Base Learner Architecture and Task Representation

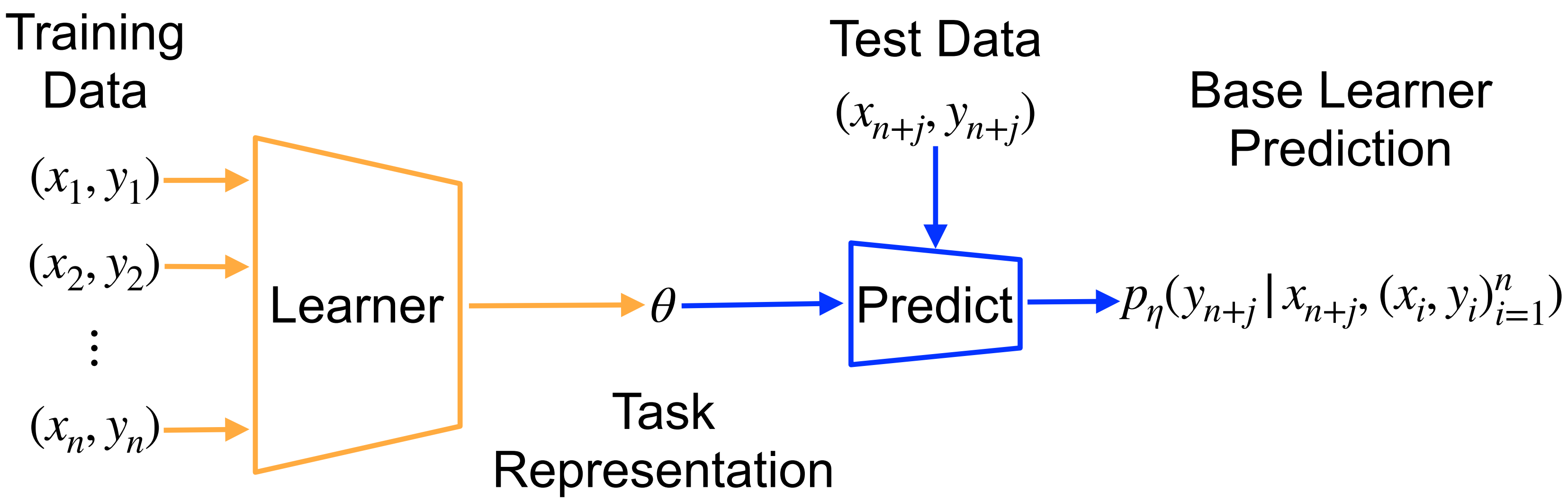


Black-box Meta-Learning

- Learn a differentiable function to map from training data to test predictions.
  - Simple: reduces meta-learning back to supervised learning.
  - Broad and flexible framework.

- Challenges:
  - Harder to learn as no inductive bias for base learner to optimise on training data.
  - Less able to generalise out of meta-training distribution.

Base Learner Architecture and Task Representation



Permutation Invariance

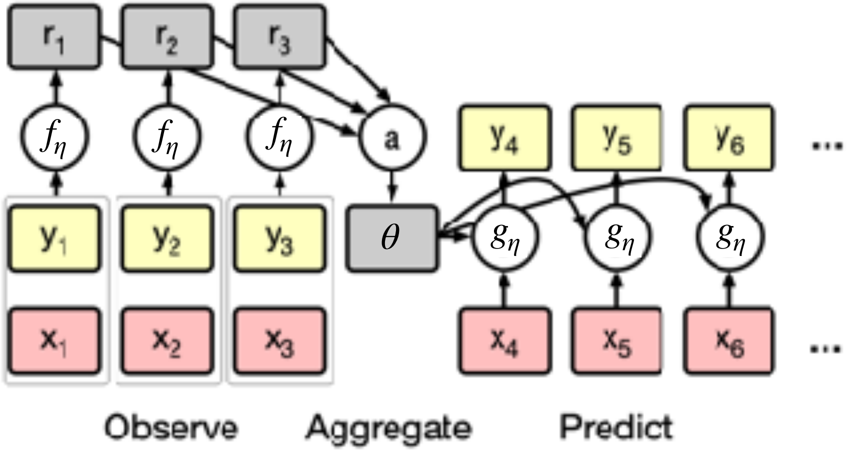
- Assumed that data items within each task are iid.
- Learner function should be invariant to permutations of the training data:

$$\text{Learner}(\eta, \{(x_1, y_1), \ldots, (x_n, y_n)\}) = \text{Learner}(\eta, \{(x_{\pi(1)}, y_{\pi(1)}), \ldots, (x_{\pi(n)}, y_{\pi(n)})\})$$

- MAML, Prototypical Nets (and simpler version of Matching Nets) are.
- LSTM meta-learner, MANN, SNAIL (and a version of Matching Nets) are not permutation invariant.

- Permutation invariance is an inductive bias.
- How to design neural architectures for permutation invariance? [more later]

Conditional Neural Processes



- Embed input/output pairs

$$(x_i, y_i) \mapsto f_\eta(x_i, y_i)$$

- Aggregate embeddings

$$\theta = \frac{1}{n} \sum_{i=1}^{n} f_\eta(x_i, y_i)$$

- Permutation invariant.
- Interpret θ as a task representation.

[Garnelo et al ICML 2018, Garnelo et al ICML DeepGen Workshop 2018]

Meta-Learning: an idiosyncratic tutorial

- Optimisation perspective on meta-learning
  - Optimisation-based meta-learning
  - Black-box meta-learning

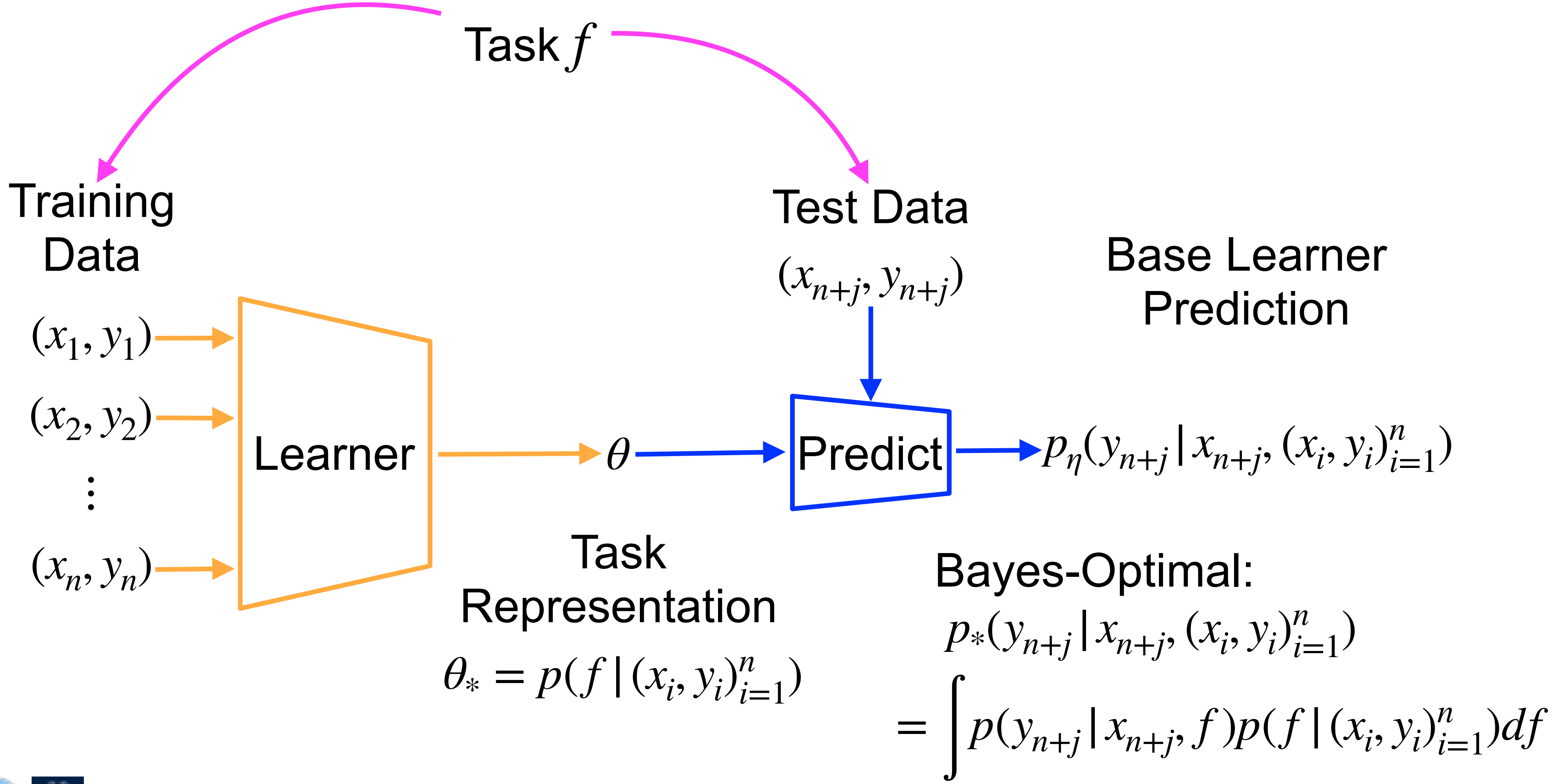
- **Probabilistic perspective on meta-learning**
  - Stochastic processes
  - Neural processes
  - Uncertainty in meta-learning

- Probabilistic symmetries and neural architectures

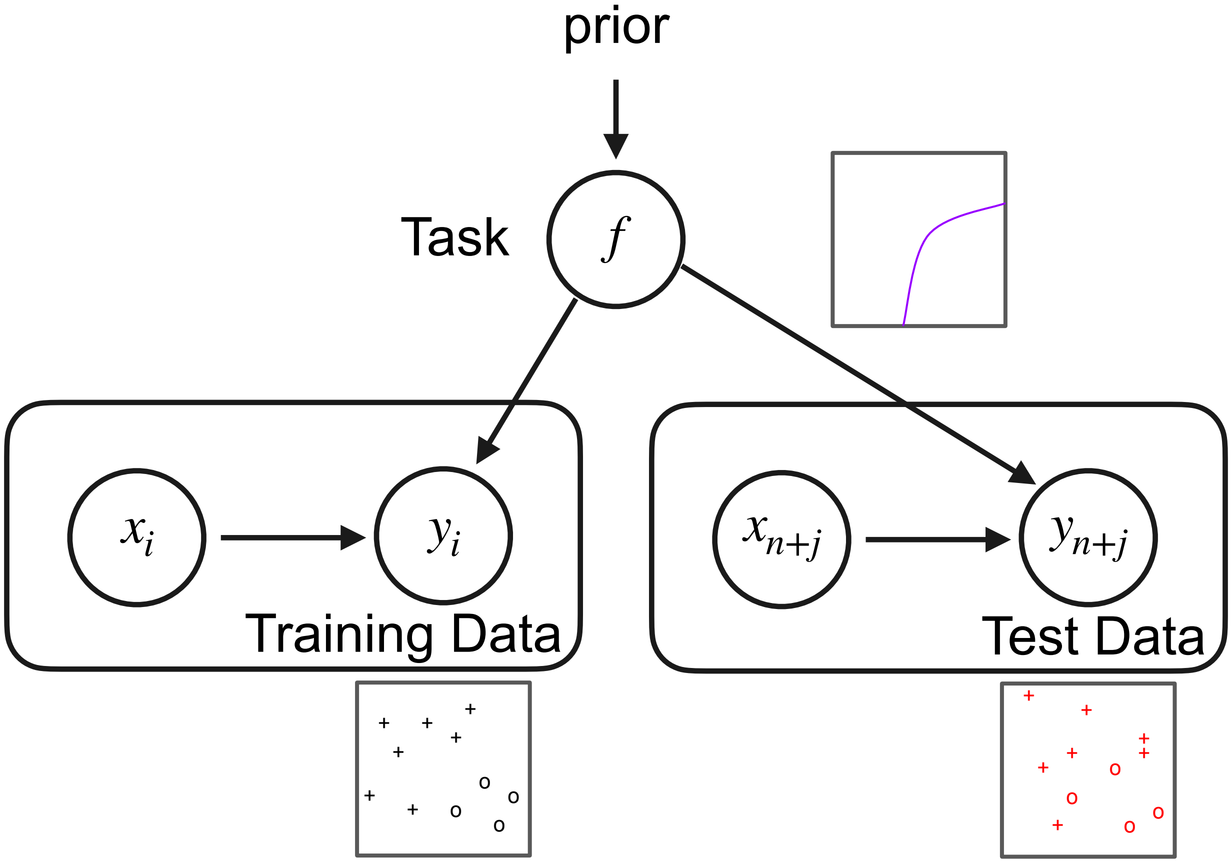
- Note: no meta reinforcement learning (meta-RL)

Yee Whye Teh

Probabilistic Perspective on Meta-Learning



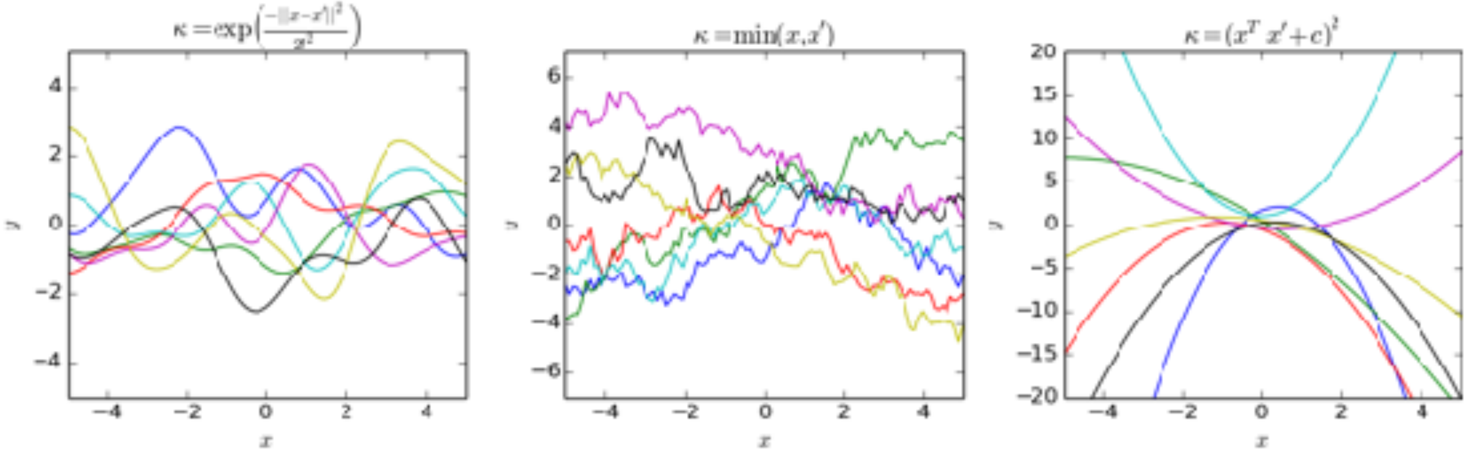
Generative Process



Specifying Stochastic Processes

- Gaussian processes are typically described via marginal distributions:

$$\begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_t) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_t) \end{pmatrix}, \begin{pmatrix} K(x_1,x_1) & K(x_1,x_2) & \cdots & K(x_1,x_t) \\ K(x_2,x_1) & K(x_2,x_2) & \cdots & K(x_2,x_t) \\ \vdots & & \ddots & \vdots \\ K(x_t,x_1) & K(x_t,x_2) & \cdots & K(x_t,x_t) \end{pmatrix} \right)$$



Yee Whye Teh

[Rasmussen & Williams 2006]

Specifying a Stochastic Process

- A stochastic process is a joint distribution over an infinite collection of random variables $(f(x))_{x \in \mathcal{X}}$.

- Kolmogorov Extension Theorem:
  - Constructs a stochastic process by specifying its finite dimensional marginal distributions.

  - Family of finite dimensional joint distributions $\rho_{x_{1:n}}$, one for each $n \in \mathbb{N}$and finite sequence $x_{1:n} \in \mathcal{X}$. We will want these to form the marginals:

$$\rho_{x_{1:n}}(y_{1:n}) = p(f(x_1) = y_1, \ldots, f(x_n) = y_n))$$

Exchangeability and Consistency

- Exchangeability: for each $n, x_{1:n}$, and permutation π of $\{1,\ldots,n\}$

$$\rho_{x_1,\ldots,x_n}(y_1,\ldots,y_n) = \rho_{x_{\pi(1)},\ldots,x_{\pi(n)}}(y_{\pi(1)},\ldots,y_{\pi(n)})$$

- Consistency: for each $n, m, x_{1:n+m}$

$$\rho_{x_1,\ldots,x_n}(y_1,\ldots,y_n) = \int \rho_{x_{1:n+m}}(y_{1:n+m})dy_{n+1:n+m}$$

Bayesian nonparametrics

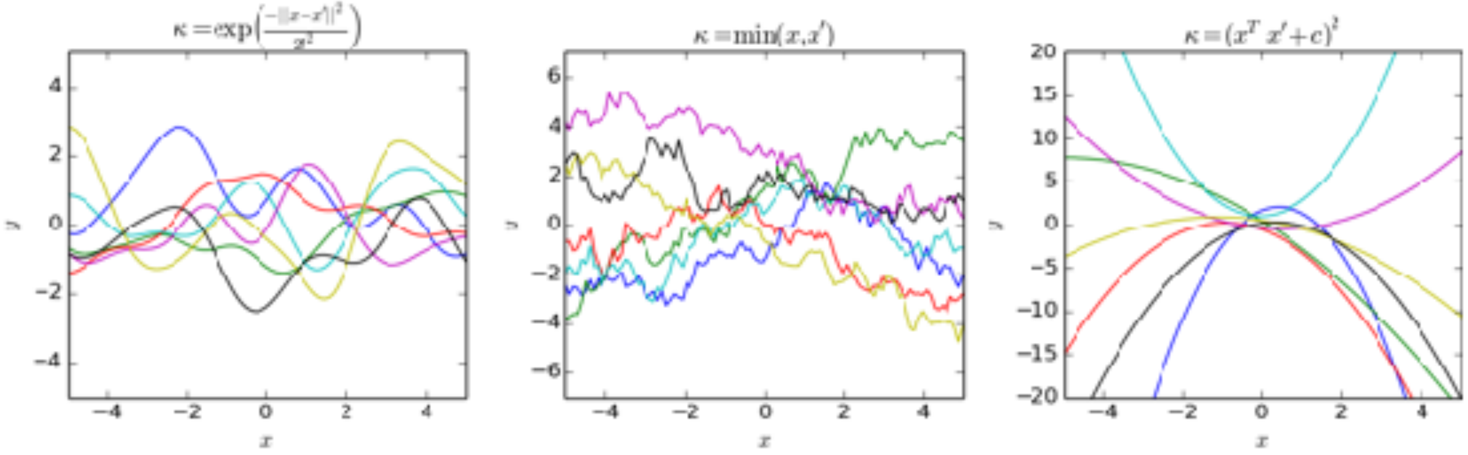
- Gaussian processes: regression, classification [Neal 1994, Rasmussen & Williams 2006]
- Dirichlet processes: infinite mixture models, clustering [Neal JCGS 1999, Rasmussen NeurIPS 2000]
- Hierarchical Dirichlet processes: topic models and HMMs [Teh et al JASA 2006]
- Pitman-Yor and Poisson-Kingman processes: power laws and language models [Teh ACL 2006], species discovery problems [Favaro et al Biometrics 2015]
- Sparse networks models and power-law structures [Caron & Fox JRSSB 2017]

[Hjort et al 2011, Ghosal & van der Vaart 2017, Jordan & Teh 20XX]

Specifying Stochastic Processes

- Gaussian processes are typically described via marginal distributions:

$$\begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_t) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_t) \end{pmatrix}, \begin{pmatrix} K(x_1,x_1) & K(x_1,x_2) & \cdots & K(x_1,x_t) \\ K(x_2,x_1) & K(x_2,x_2) & \cdots & K(x_2,x_t) \\ \vdots & & \ddots & \vdots \\ K(x_t,x_1) & K(x_t,x_2) & \cdots & K(x_t,x_t) \end{pmatrix} \right)$$



Specifying Stochastic Processes

- Gaussian processes can equivalently be described via its conditional distributions:

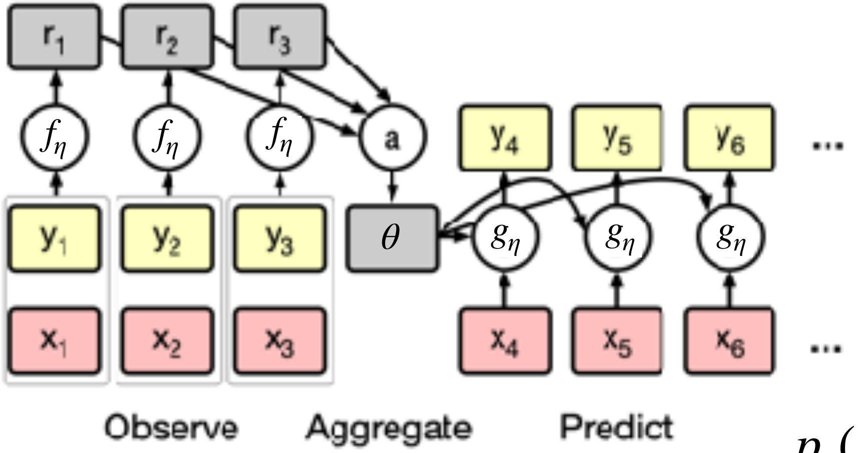
$$f(x_{t+1})|f(x_1) = y_1, \ldots, f(x_t) = y_t$$
$$\sim\mathcal{N}\left(\mu(x_{t+1}) + K_{t+1,1:t}K^{-1}_{1:t,1:t}y_{1:t},\ K_{t+1,t+1} - K_{t+1,1:t}K^{-1}_{1:t,1:t}K_{1:t,t+1}\right)$$

- In general, stochastic processes can also be described using a consistent family of conditional distributions:

$$p(f(x_{n+j}) = y_{n+j} \,|\, f(x_1) = y_1, \ldots, f(x_n) = y_n)$$

for training dataset $\{(x_i, y_i)\}_{i=1}^n$ and test data (x_{n+j}, y_{n+j}).

Conditional Neural Processes



- Embed input/output pairs

$$(x_i, y_i) \mapsto f_\eta(x_i, y_i)$$

- Aggregate embeddings

$$\theta = \frac{1}{n}\sum_{i=1}^{n} f_\eta(x_i, y_i)$$

- Predict

$$p_\eta(y_{n+j} \,|\, x_{n+j}, (x_i, y_i)_{i=1}^n) = \mathcal{N}(y_{n+j}; g_\eta(\theta, x_{n+j}))$$

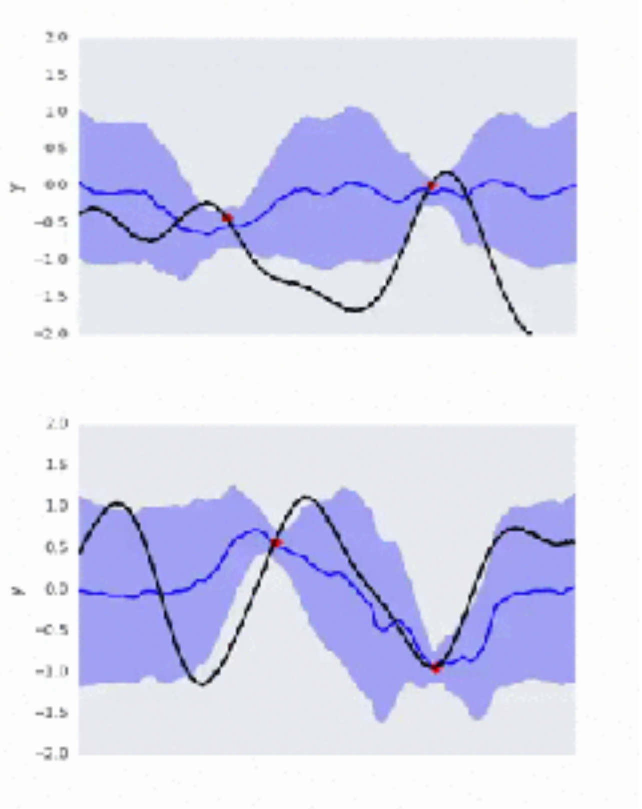
- Meta-learning learns the stochastic process!

[Garnelo et al ICML 2018]

Conditional Neural Processes

Task = Function on 1D space.

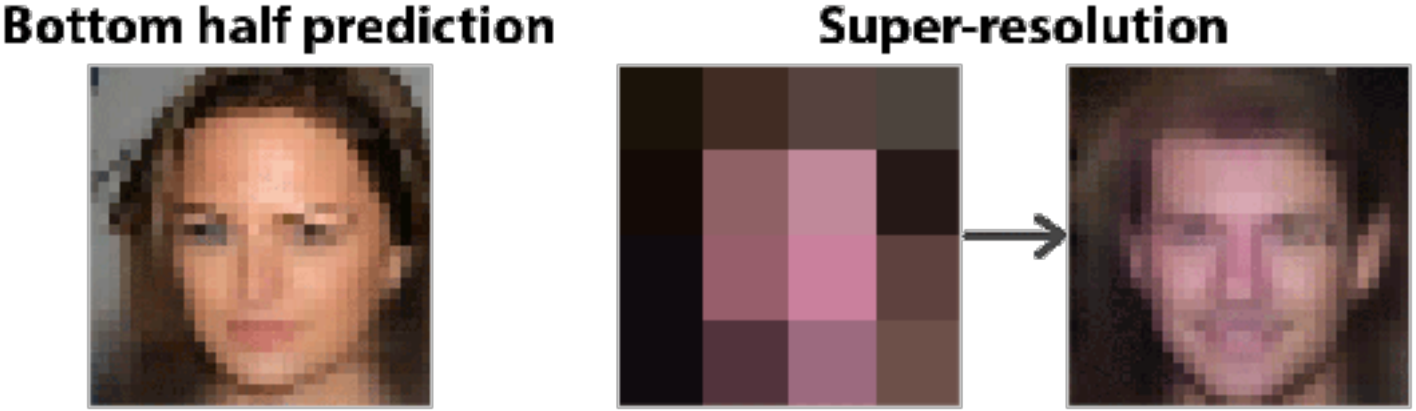
Given training points, use neural processes to predict mean and std of function values at other locations.



[Garnelo et al ICML 2018]

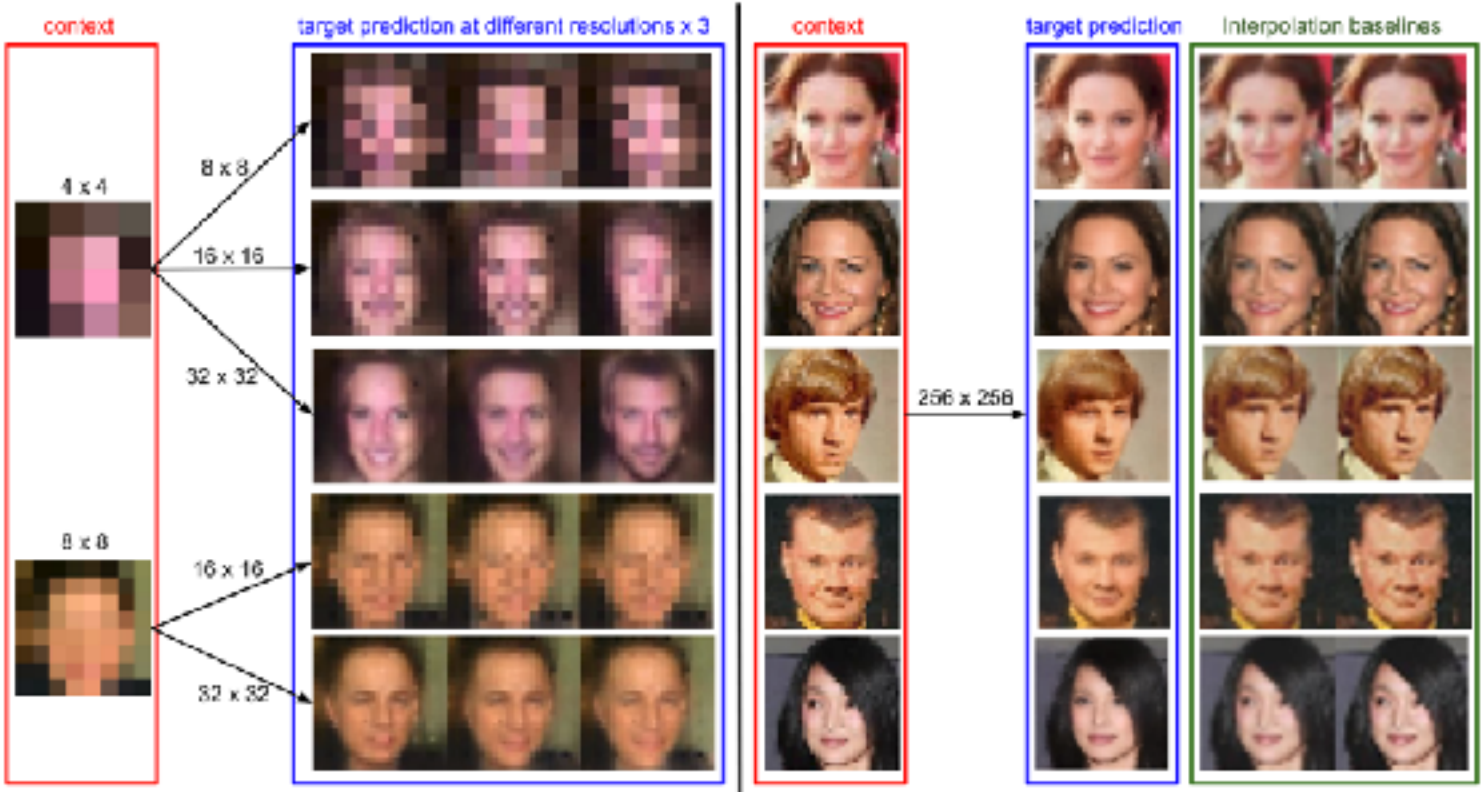
Image Completion and Super-resolution

Task = Image = Function on 2D space.

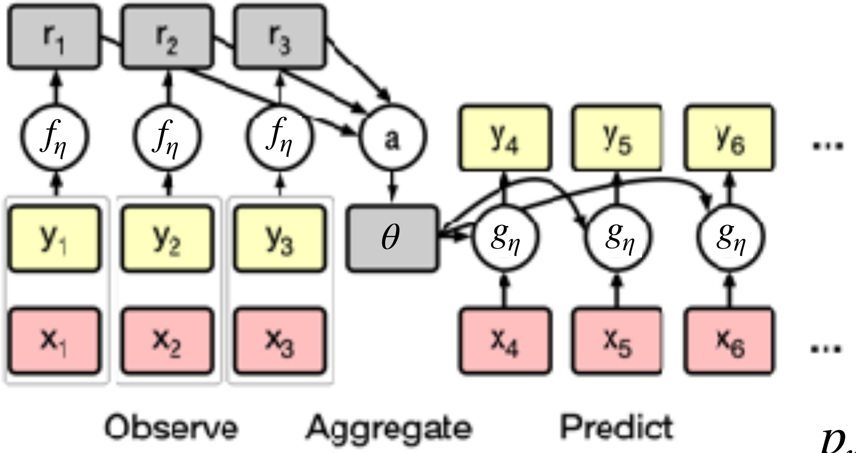


[Kim et al ICLR 2019]

Image Super-resolution



Conditional Neural Processes



- Embed input/output pairs

$$(x_i, y_i) \mapsto f_\eta(x_i, y_i)$$

- Aggregate embeddings

$$\theta = \frac{1}{n} \sum_{i=1}^{n} f_\eta(x_i, y_i)$$

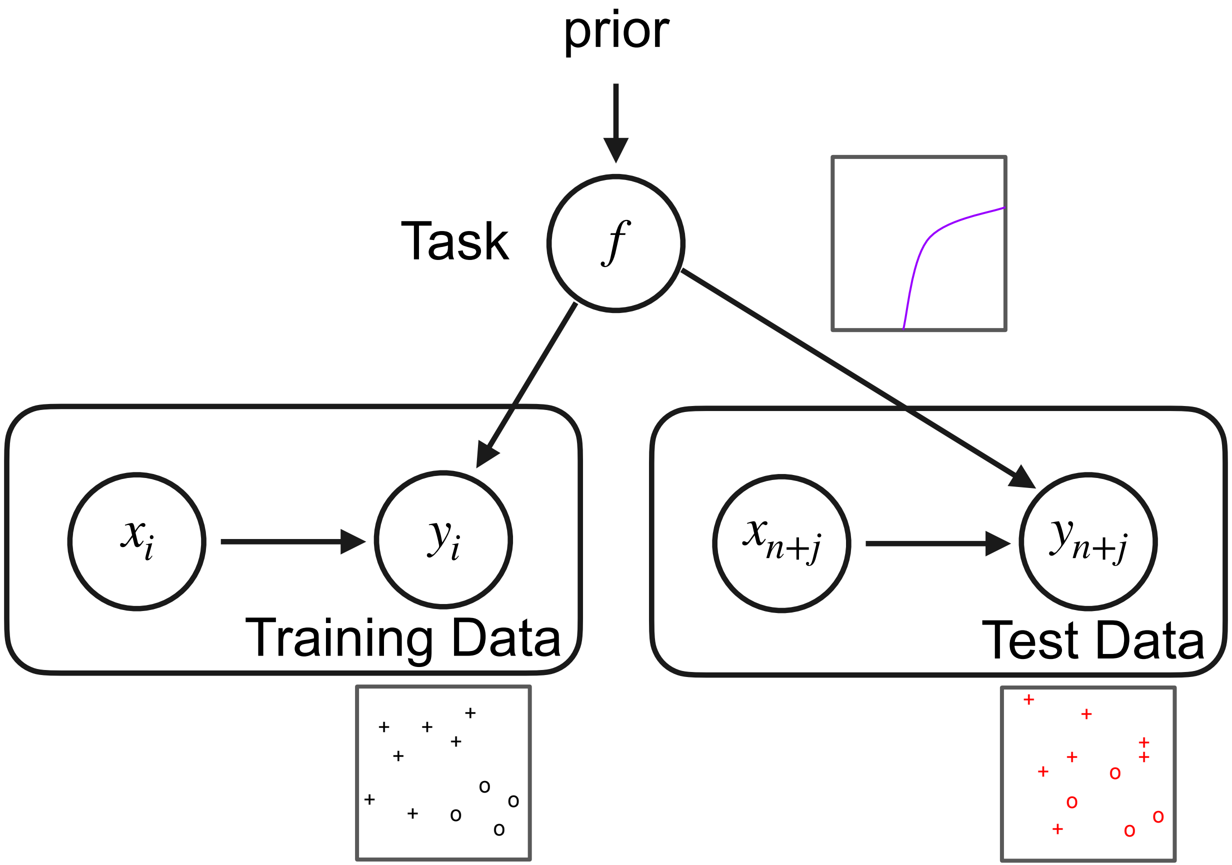
- Predict

$$p_\eta(y_{n+j} \mid x_{n+j}, (x_i, y_i)_{i=1}^n) = \mathcal{N}(y_{n+j}; g_\eta(\theta, x_{n+j}))$$

- But: architecture cannot model dependence among test outputs!

[Garnelo et al ICML 2018]

Generative Process



Latent Variable Model

- Generative **model**:

$$p_\eta(z, y_{1:n+m} | x_{1:n+m}) = p_\eta(z) \prod_{i=1}^{n+m} p_\eta(y_i | z, x_i)$$

$$p_\eta(z) = \mathcal{N}(z; 0, I)$$

$$p_\eta(y_i | z, x_i) = \mathcal{N}(y_i; g_\eta(z, x_i))$$

- Variational learning objective:

$$\log p(y_{1:n+m} | x_{1:n+m})$$

$$\geq \mathbb{E}_{q(z|x_{1:n+m}, y_{1:n+m})} \left[\sum_{i=1}^{n+m} \log p_\eta(y_i | z, x_i) + \log p(z) - \log q(z | x_{1:n+m}, y_{1:n+m}) \right]$$

$$q(z | x_{1:t}, y_{1:t}) = \mathcal{N}\left(z; s_\eta \left(\frac{1}{t} \sum_{i=1}^{t} f_\eta(x_i, y_i) \right) \right)$$

[Garnelo et al ICML DeepGen Workshop 2018]

Alternative Learning Objective

- "our training procedure is based on a simple machine learning principle: test and train conditions must match" — [Vinyals et al NeurIPS 2016]
- Alternative objective:

$$\log p(y_{n+:n+m} \mid x_{1:n}, y_{1:n}, x_{n+1:n+m})$$

$$\geq \mathbb{E}_{q(z|x_{1:n+m}, y_{1:n+m})}\left[\sum_{i=1}^{n+m} \log p_\eta(y_i \mid z, x_i) + \log p(z \mid x_{1:n}, y_{1:n}) - \log q(z \mid x_{1:n+m}, y_{1:n+m})\right]$$

$$\approx \mathbb{E}_{q(z|x_{1:n+m}, y_{1:n+m})}\left[\sum_{i=1}^{n+m} \log p_\eta(y_i \mid z, x_i) + \log q(z \mid x_{1:n}, y_{1:n}) - \log q(z \mid x_{1:n+m}, y_{1:n+m})\right]$$

$$q(z \mid x_{1:t}, y_{1:t}) = \mathcal{N}\left(z; s_\eta\left(\frac{1}{t}\sum_{i=1}^{t} f_\eta(x_i, y_i)\right)\right)$$

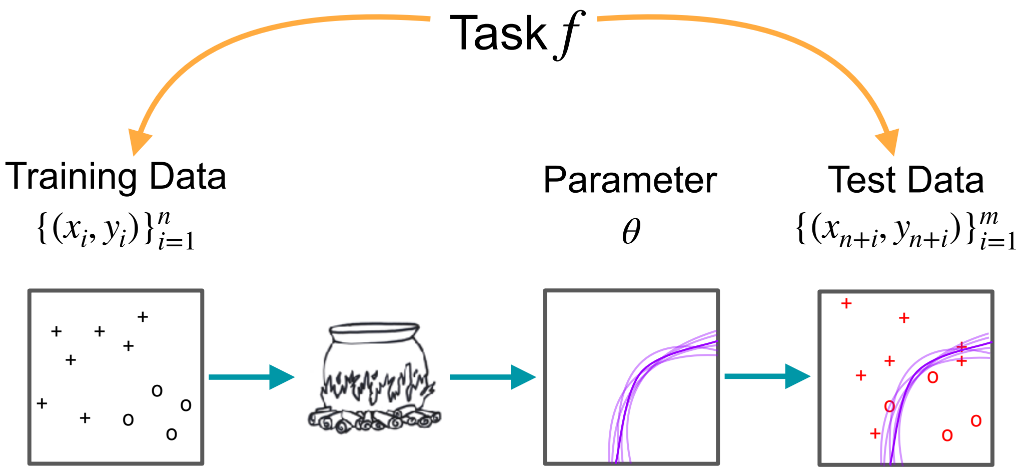
[Kim et al ICLR 2019, Le et al NeurIPS BDL 2018]

Probabilistic Perspective on Meta-Learning

- Meta-learning as learning a stochastic process:
  - Meta-learning a prior over functions -> meta-learning inductive biases from meta-training set!
  - Base learner can be thought of as **amortized learning**.

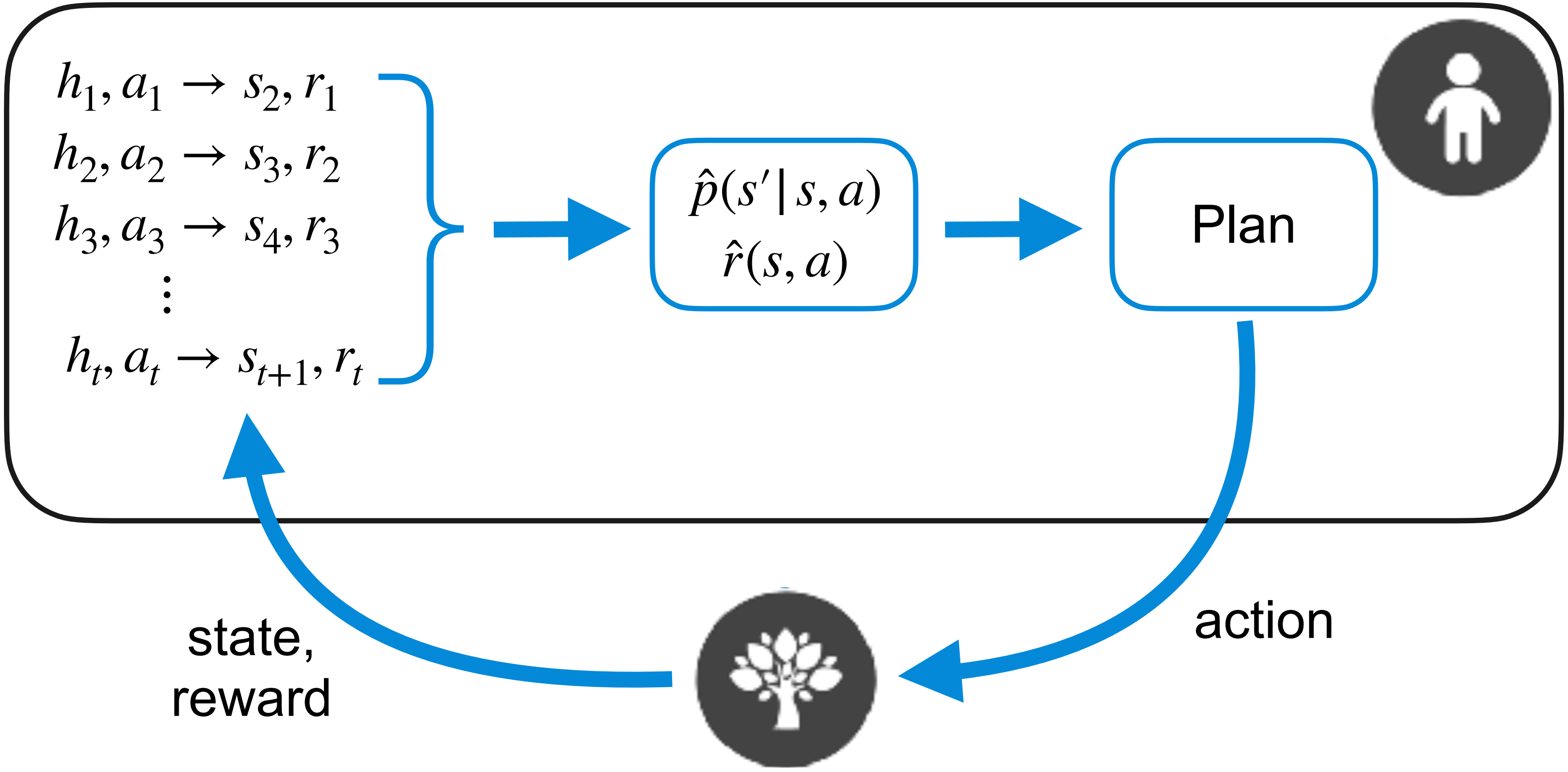
- Uncertainties are important in meta-learning applications:
  - Active learning
  - Bayesian optimisation
  - Reinforcement learning

Uncertainty in Meta-Learning



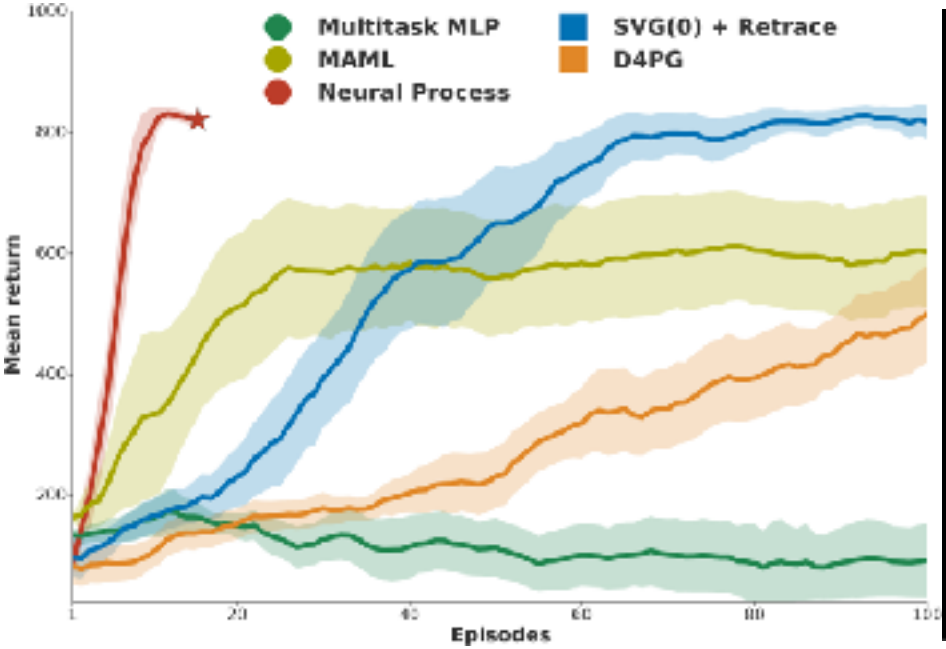
- Small training sets in meta-learning → uncertainty in task inference!
- Important for:
  - active learning,
  - Bayesian optimisation,
  - reinforcement learning

Efficient Model-based Reinforcement Learning



Yee Whye Teh

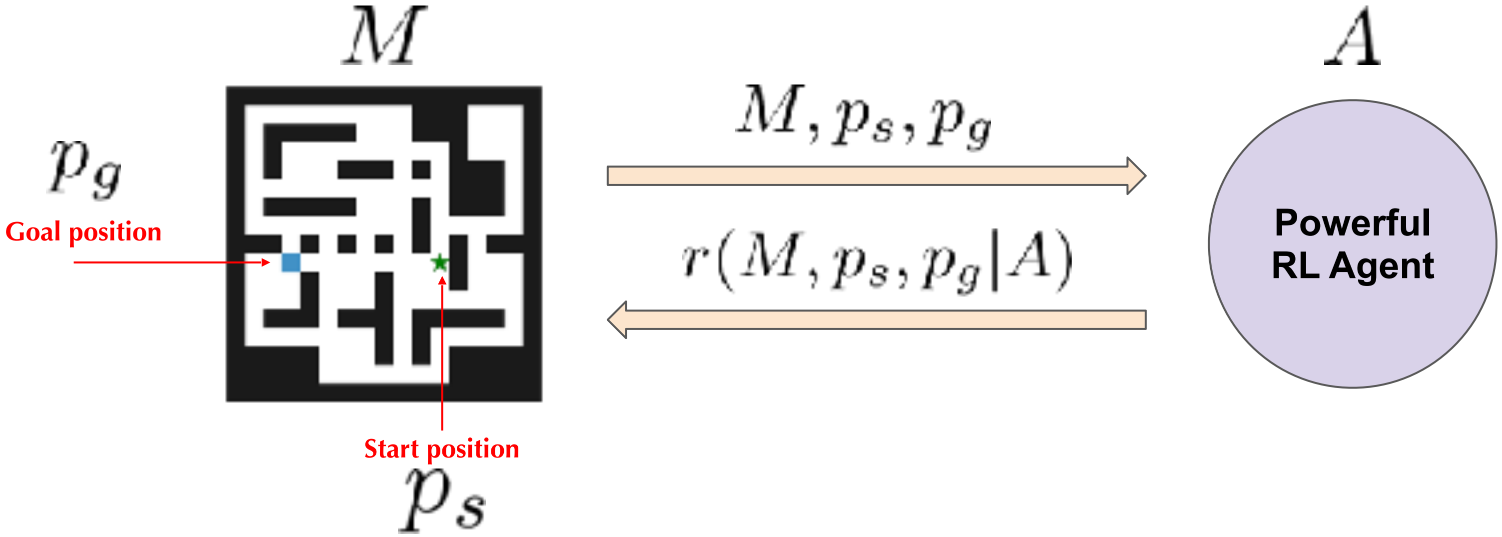
Cart Pole



Yee Whye Teh

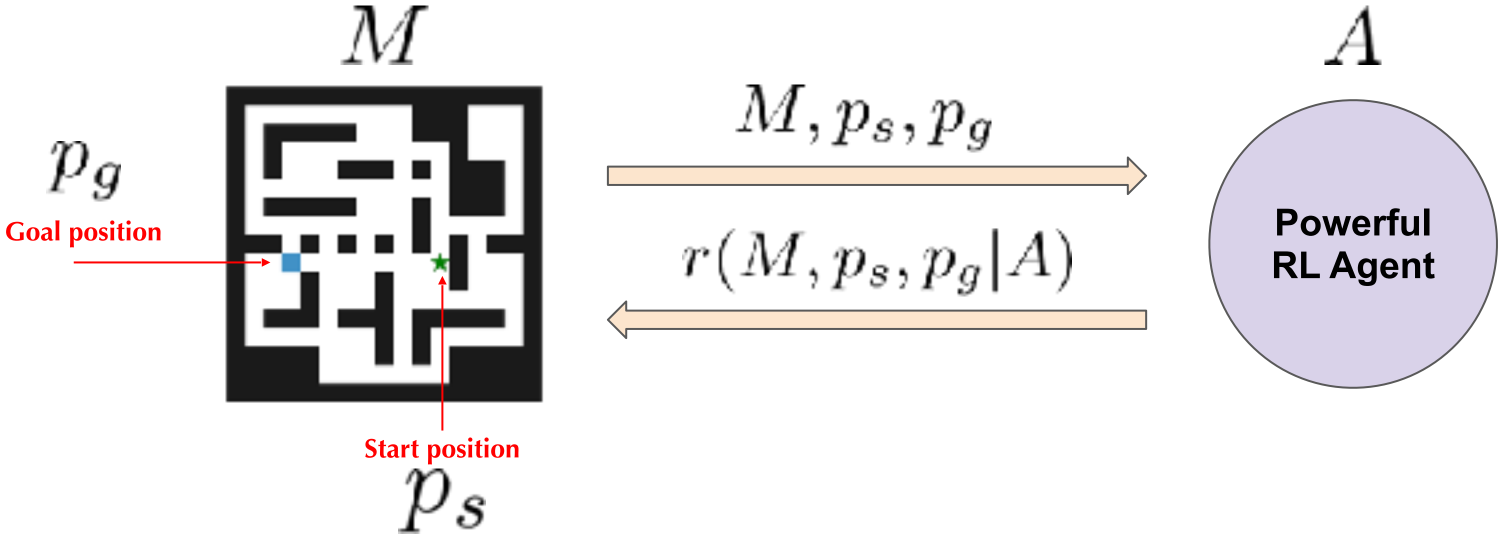
[Galashov et al 2019]

Adversarial Testing of RL Agents



[Galashov et al 2019]

Adversarial Testing of RL Agents

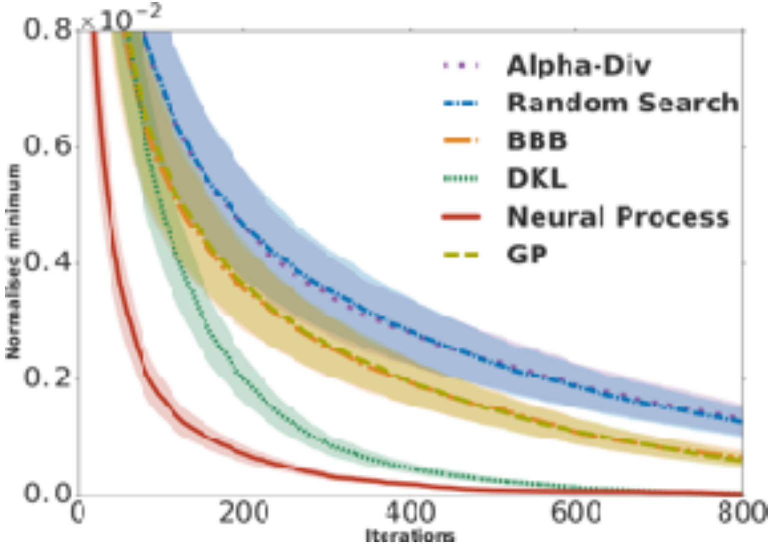


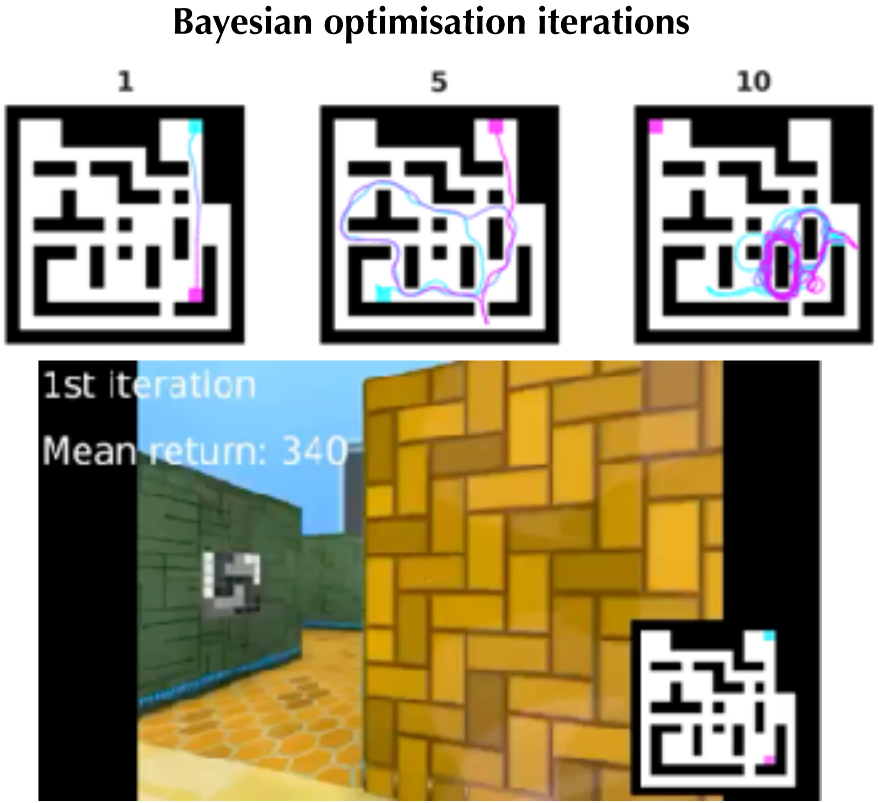
Bayesian Optimizaion $\min_{M, p_s, p_g} r(M, p_s, p_g | A)$

$(M, p_s, p_g, A) \sim p(\mathcal{T})$ - training & holdout samples (agents, mazes, positions)

[Galashov et al 2019]

Adversarial Testing of RL Agents





Meta-Learning: an idiosyncratic tutorial

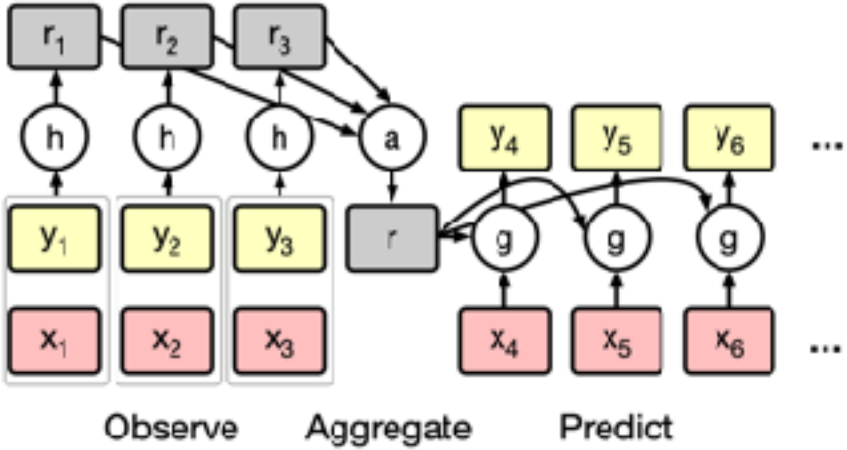
- Optimisation perspective on meta-learning
  - Optimisation-based meta-learning
  - Black-box meta-learning

- Probabilistic perspective on meta-learning
  - Stochastic processes
  - Neural processes
  - Uncertainty in meta-learning

- **Probabilistic symmetries and neural architectures**

- Note: no meta reinforcement learning (meta-RL)

Permutation-Invariance in Neural Processes



- Embed input/output pairs

$$(x_i, y_i) \mapsto f_\eta(x_i, y_i)$$

- Aggregate embeddings

$$\theta = \frac{1}{n}\sum_{i=1}^{n} f_\eta(x_i, y_i)$$

- Predict

$$p_\eta(y_{n+j} \,|\, x_{n+j}, (x_i, y_i)_{i=1}^n) = \mathcal{N}(y_{n+j}; g_\eta(\theta, x_{n+j}))$$

Characterising Permutation-Invariant Functions

- Function $h : \mathscr{X}^n \to \mathscr{Y}$ is permutation-invariant,

$$h(\pi \cdot (x_1, \ldots, x_n)) = h(x_{\pi(1)}, \ldots, x_{\pi(n)}) = h(x_1, \ldots, x_n)$$

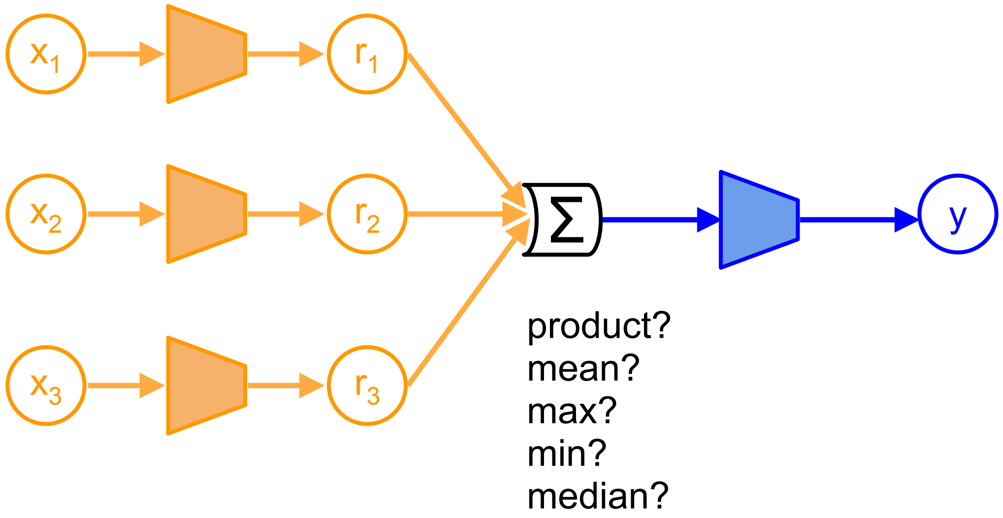
- Can we characterise the class of permutation-invariant functions?

- If we use neural networks to parameterise permutation-invariant functions, how should we choose the architecture?

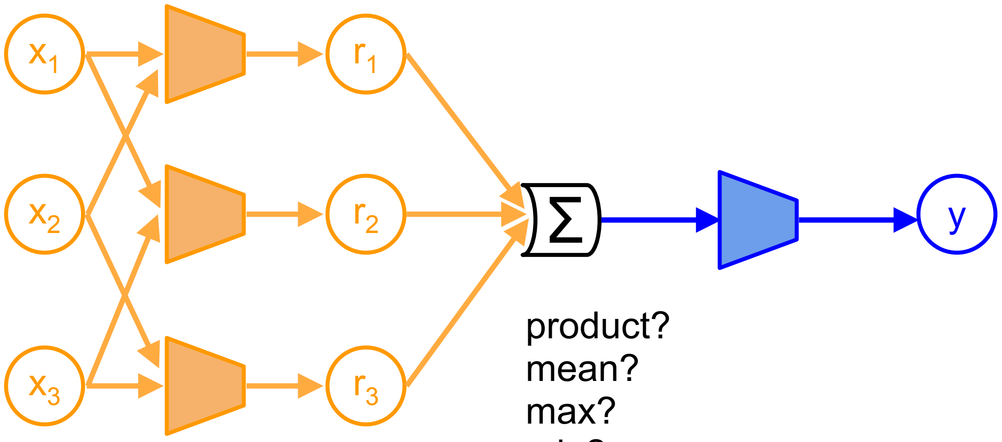
- Given an architecture choice, can the neural network approximate well any arbitrary permutation-invariant function?

[Zaheer et al NeurIPS 2017, Bloem-Reddy & Teh JMLR 2020]

Characterising Permutation-Invariant Functions



Characterising Permutation-Invariant Functions



product?
mean?
max?
min?
median?

Functional Symmetry Properties

- Function $h : \mathscr{X}^n \to \mathscr{Y}^n$ is **permutation-equivariant**,

$$h(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$$

$$h(\pi \cdot (x_1, \ldots, x_n)) = \pi \cdot (y_1, \ldots, y_n) = \pi \cdot h(x_1, \ldots, x_n)$$

- **Group** G acting on input space $\mathscr{X}$ and output space $\mathscr{Y}$.
  - G**-invariant**:

$$h(g \cdot x) = h(x)$$

  - G**-equivariant**:

$$h(g \cdot x) = g \cdot h(x)$$

Probabilistic Symmetries

- A distribution P for a random sequence $\mathbf{X}_n = (X_1, \ldots, X_n)$ is **exchangeable** if

$$P(X_1, \ldots, X_n) = P(\pi \cdot (X_1, \ldots, X_n))$$
$$P(X_1 \in B_1, \ldots, X_n \in B_n) = P(X_{\pi(1)} \in B_1, \ldots, X_{\pi(n)} \in B_n))$$

- Exchangeability is permutation-invariance of P.

- $\mathbf{X}_{\mathbb{N}}$ is infinitely exchangeable if all length n prefixes are exchangeable.
- **de Finetti's Theorem**:

$$\mathbf{X}_{\mathbb{N}} \text{ is infinitely exchangeable} \Leftrightarrow X_i \,|\, Q \sim_{iid} Q \text{ for some random } Q.$$

Probabilistic Symmetries for Conditional Distributions

- A conditional distribution $P(Y|X)$ is a stochastic relaxation for a function $Y = h(X)$.

- $P(Y|X)$ is G-invariant if:

$$P(Y|X) = P(Y|g \cdot X)$$

$$P(Y \in B | X \in A) = P(Y \in B | g \cdot X \in A)$$

- $P(Y|X)$ is G-equivariant if:

$$P(Y|X) = P(g \cdot Y | g \cdot X)$$

- Can we characterise the class of permutation-invariant conditional distributions?

Empirical Measure

- de Finetti's Theorem may fail for finitely exchangeable sequences.
- The **empirical measure** of $\mathbf{X}_n$ is
$$\mathbb{M}_{\mathbf{X}_n}(\,\cdot\,) = \sum_{i=1}^{n} \delta_{X_i}(\,\cdot\,)$$
- The empirical measure is a **sufficient statistic**: P is exchangeable iff
$$P(\mathbf{X}_n \in \cdot \mid \mathbb{M}_{\mathbf{X}_n} = m) = \mathbb{U}_m(\,\cdot\,)$$
where $\mathbb{U}_m$ is the uniform distribution over all sequences $(x_1, \ldots, x_n)$ with empirical measure m.

Noise Outsourcing

- If X and Y are random variables in "nice" (e.g. Borel) spaces $\mathscr{X}$ and $\mathscr{Y}$, then there are a random variable $\eta \sim U[0,1]$ with $\eta \perp\!\!\!\perp X$ and a function $h : [0,1] \times \mathscr{X} \mapsto \mathscr{Y}$ such that

$$(X, Y) =_{a.s.} (X, h(\eta, X))$$

- Furthermore, if there is an **adequate statistic** $S(X)$ with $X \perp\!\!\!\perp Y \,|\, S(X)$, then

$$(X, Y) =_{a.s.} (X, h(\eta, S(X)))$$

Probabilistic Permutation-Invariance

- Now suppose we have random variables $\mathbf{X}_n$ and Y.
  - Y is conditionally permutation-invariant given $\mathbf{X}_n$.
  - $\mathbf{X}_n$ is marginally permutation-invariant (exchangeable).
- The empirical measure is a sufficient statistic for $\mathbf{X}_n$.

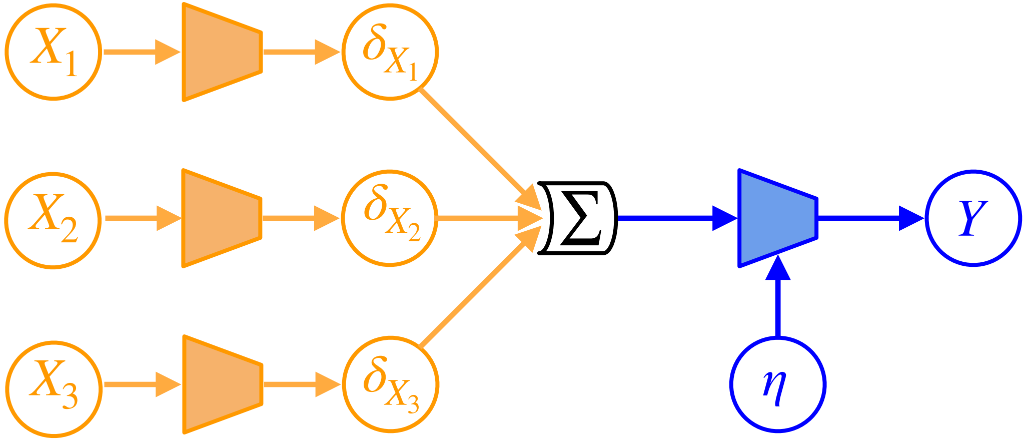
- It is also an adequate statistic for Y given $\mathbf{X}_n$:

$$P(Y \mid \mathbf{X}_n = \mathbf{x}_n) = P(Y \mid \mathbb{M}_{\mathbf{X}_n} = \mathbb{M}_{\mathbf{x}_n})$$

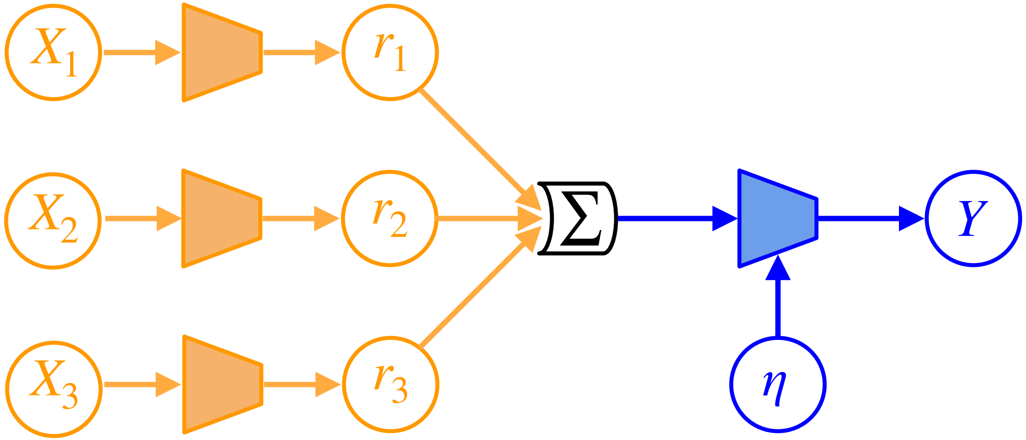
  We have the conditional independence $\mathbf{X}_n \perp\!\!\!\perp Y \mid \mathbb{M}_{\mathbf{X}_n}$.
- Noise outsourcing…

$$(\mathbf{X}_n, Y) =_{a.s.} (\mathbf{X}_n, h(\eta, \mathbb{M}_{\mathbf{X}_n}))$$

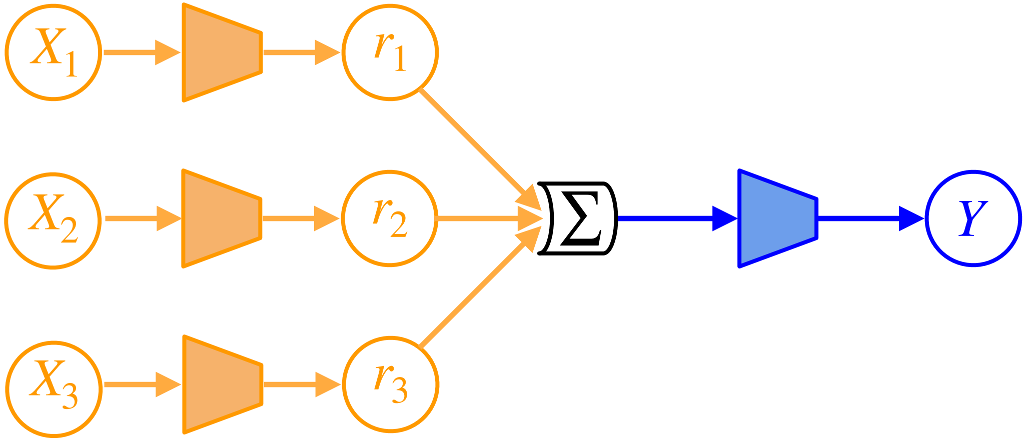
Probabilistic Permutation-Invariance



Probabilistic Permutation-Invariance



Functional Permutation-Invariance



Probabilistic Permutation-Equivariance

- Now suppose we have random sequences $\mathbf{X}_n$ and $\mathbf{Y}_n$.
  - $\mathbf{Y}_n$ is conditionally permutation-equivariant given $\mathbf{X}_n$.
  - $\mathbf{X}_n$ is marginally permutation-invariant (exchangeable).

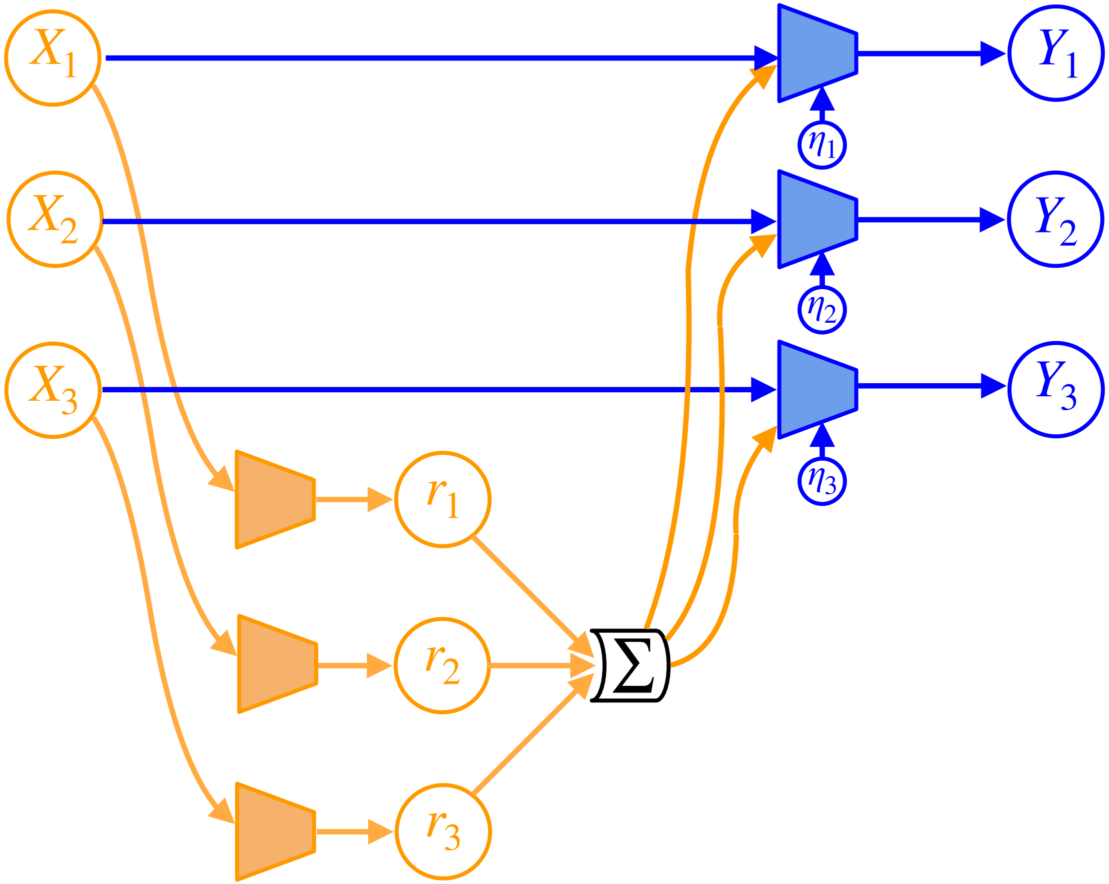
- Also suppose that $Y_i \perp\!\!\!\perp \mathbf{Y}_n \backslash Y_i \,|\, \mathbf{X}_n$ for each i.

- Then:

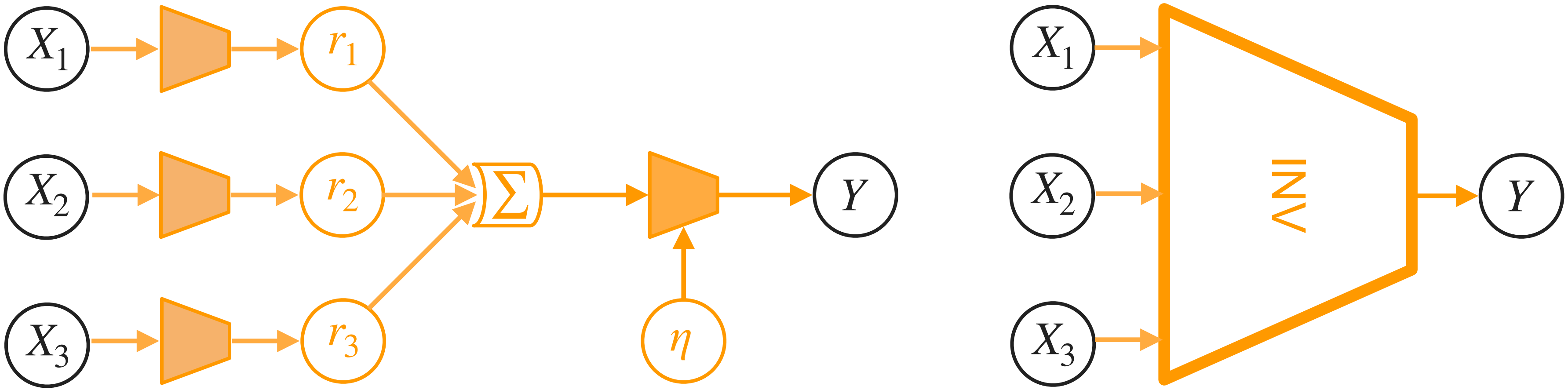
$$(\mathbf{X}_n, (Y_1, \ldots, Y_n)) =_{a.s.} (\mathbf{X}_n, (h(\eta_1, X_1, \mathbb{M}_{\mathbf{X}_n}), \ldots, h(\eta_n, X_n, \mathbb{M}_{\mathbf{X}_n}))$$

for outsourced noise (η_i) that are identical, mutually independent and independent of $\mathbf{X}_n$.

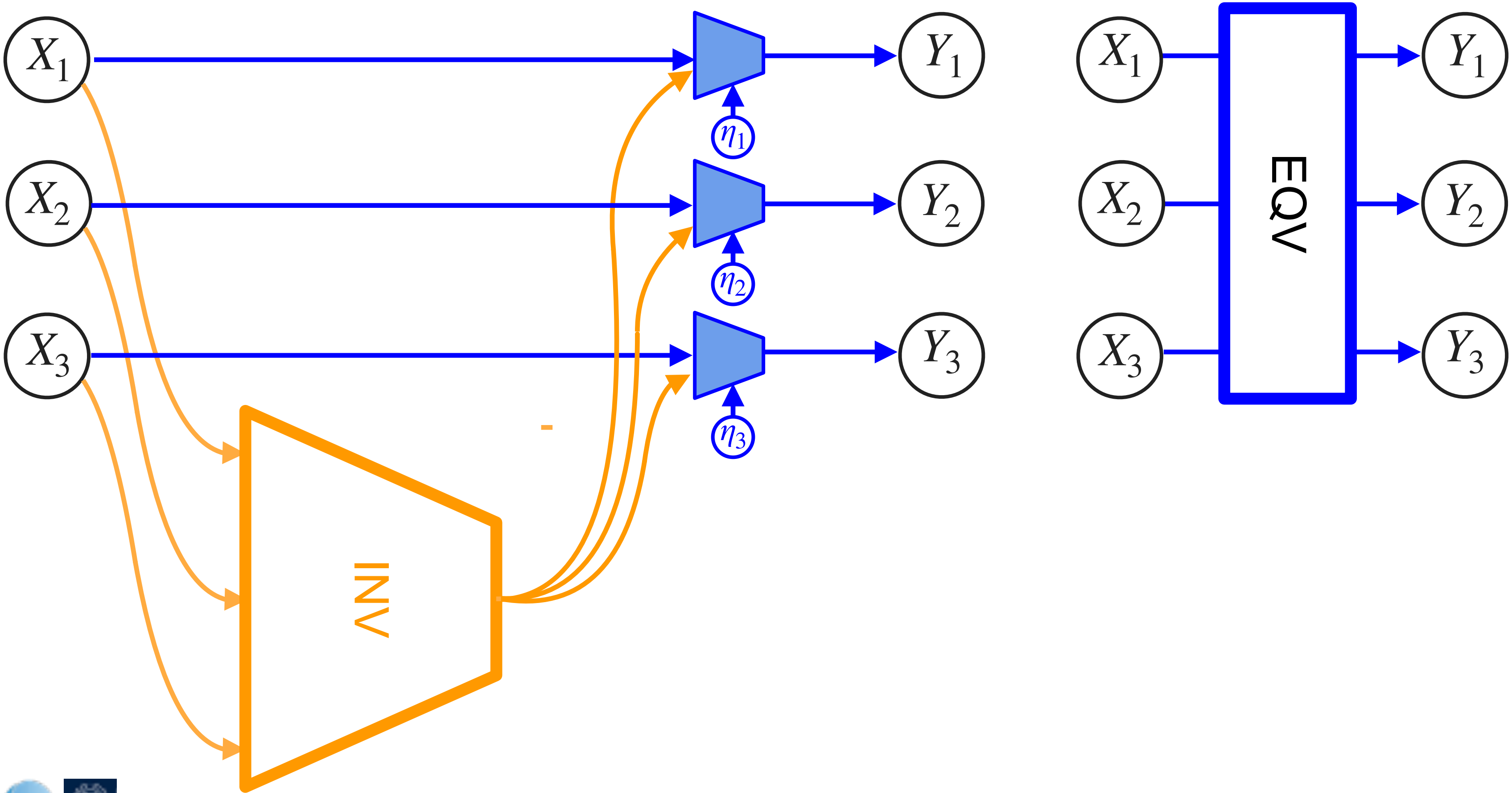
Probabilistic Permutation-Equivariance



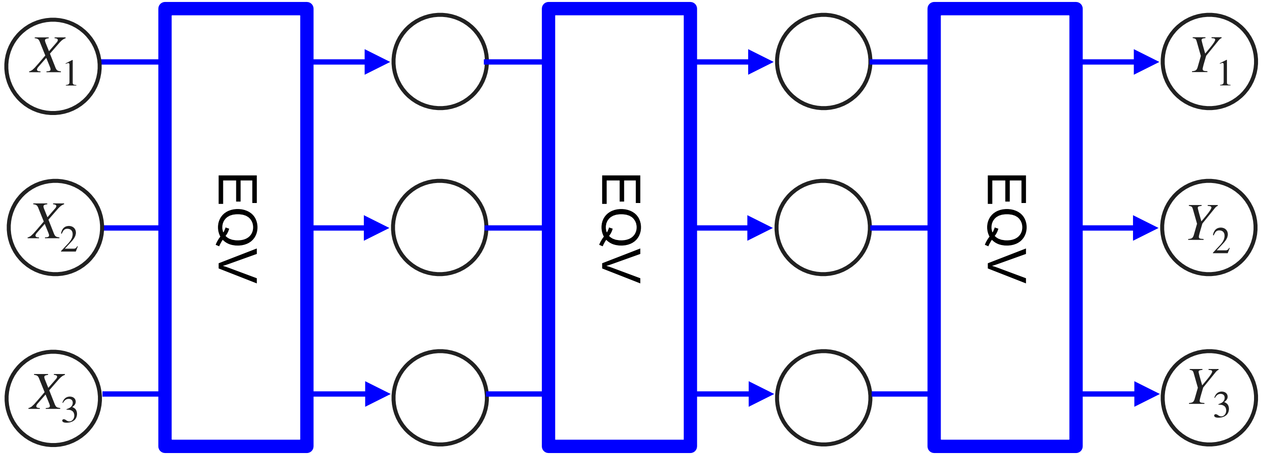
Composing Invariant and Equivariant Modules



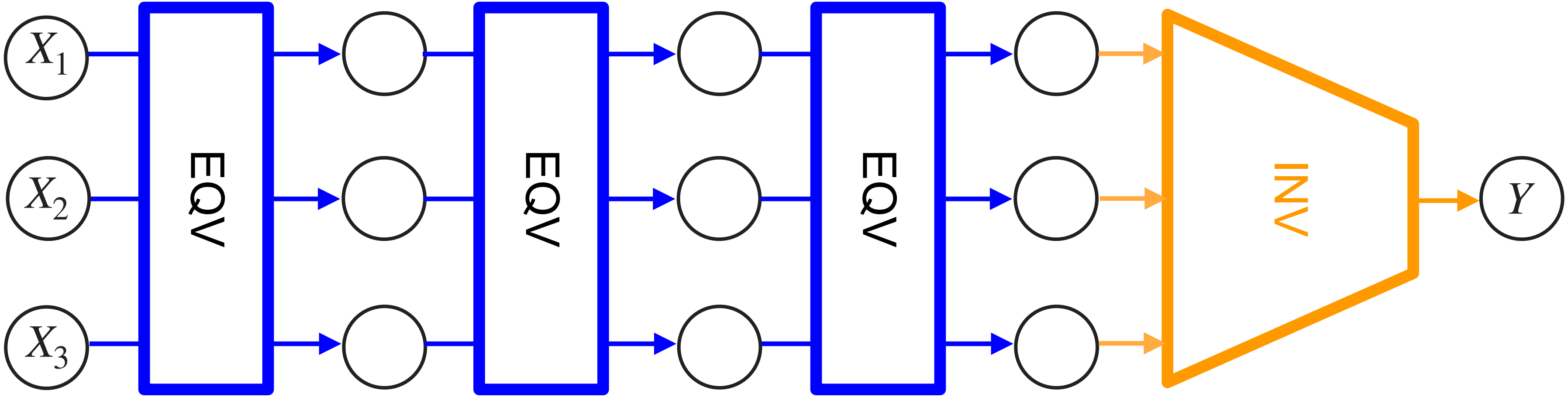
Composing Invariant and Equivariant Modules



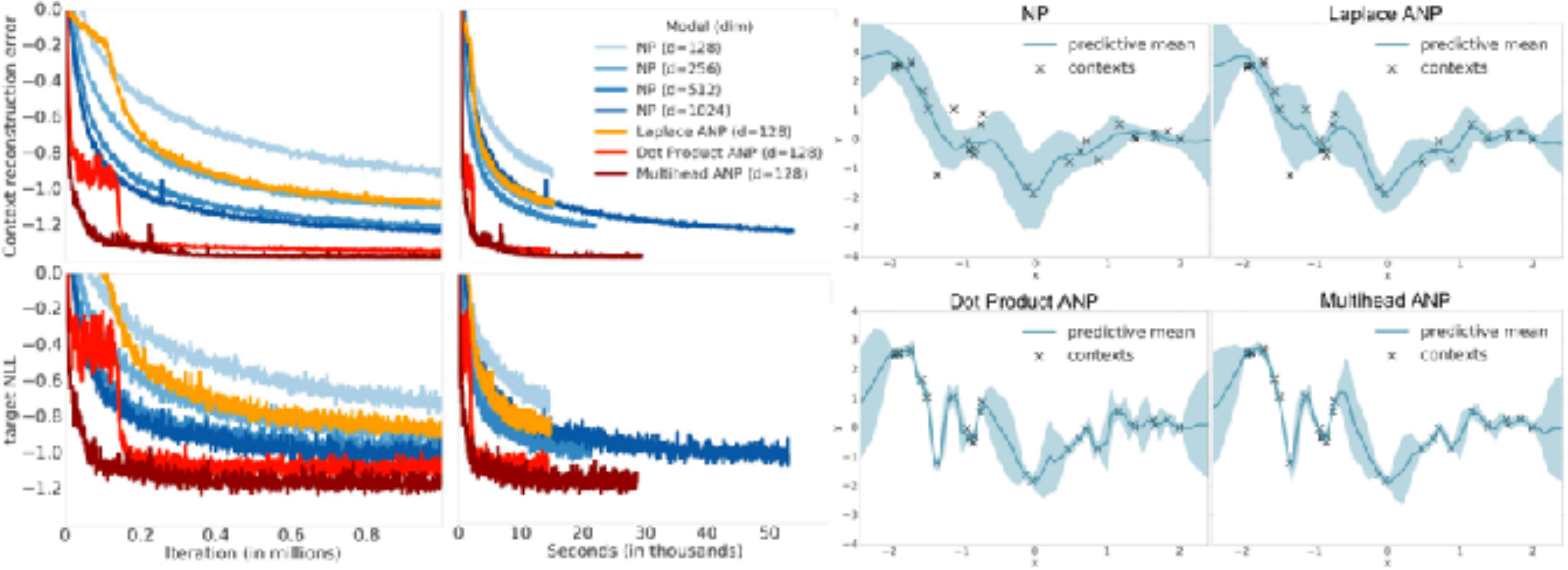
Composing Invariant and Equivariant Modules



Composing Invariant and Equivariant Modules



Attentive Neural Processes



[Kim et al ICLR 2019, Lee et al ICML 2019]

Maximal Invariant and Maximal Equivariant

- Let G be a compact group.
- A maximal invariant is a statistic $M : \mathcal{X} \mapsto \mathcal{S}$ such that

$$M(g \cdot x) = M(x) \forall g \in G, x \in \mathcal{X}$$

$$M(x_1) = M(x_2) \Rightarrow \exists g \in G : x_1 = g \cdot x_2$$

- A maximal equivariant $\tau : \mathcal{X} \mapsto G$ satisfies

$$\tau(g \cdot x) = g \cdot \tau(x)$$

Probabilistic and Functional Symmetries

- Let G be a compact group and X be marginally G-invariant.

- Let M be a maximal invariant, then

$$Y \text{ is conditionally } G\text{-invariant given } X \Leftrightarrow (X, Y) =_{a.s.} (X, h(\eta, M(X)))$$

  for outsourced noise η independent of X and a function h.

- If a maximal equivariant τ exists and $G_X \subset G_Y$ a.s., then

$$Y \text{ is conditionally } G\text{-equivariant given } X \Leftrightarrow (X, Y) =_{a.s.} (X, h(\eta, X))$$

  for outsourced noise η independent of X and a function h that is G-equivariant in its second argument.

Probabilistic Symmetries

- Tools from probabilistic symmetry, sufficiency and adequacy allowed us to answer questions about neural architectures under symmetry.
  - Framework extends to graph and array structured data with node exchangeability.
  - Continuous groups?
  - How to relax assumptions of conditional independence of outputs?

[Kallenberg 2002, Kallenberg 2005]

Meta-Learning: an idiosyncratic tutorial

- Optimisation perspective on meta-learning
  - Optimisation-based meta-learning
  - Black-box meta-learning

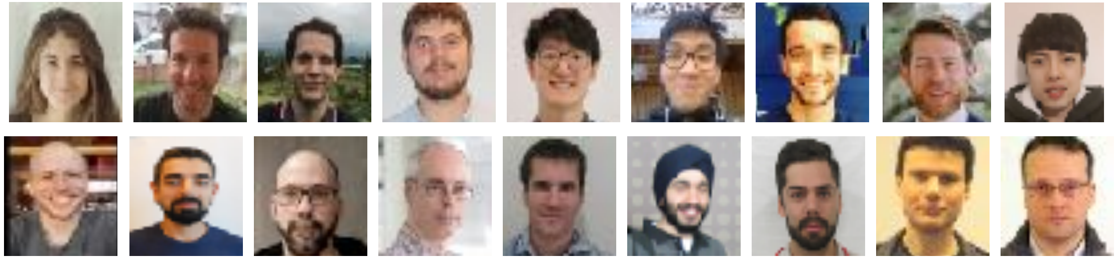
- Probabilistic perspective on meta-learning
  - Stochastic processes
  - Neural processes
  - Uncertainty in meta-learning

- Probabilistic symmetries and neural architectures

- Note: no meta reinforcement learning (meta-RL)

Thank You!

- Collaborators, colleagues, mentors



European Research Council

- You!

- Questions?

http://csml.stats.ox.ac.uk/people/teh