Meta-learning for dGAP

Presenter: Zhe Wang

https://qdata.github.io/deep2Read

Zhe Wang

201909

The motivations:

- Learning a dependency graph
- Incorporate the dependency graph into downstream tasks
  Two choices:
  1. Improve meta learning (learning a dependency graph across new task and knowledge set)
  2. Improve tasks (learning a dependency graph across limited tasks)

Learning a dependency graph across tasks.

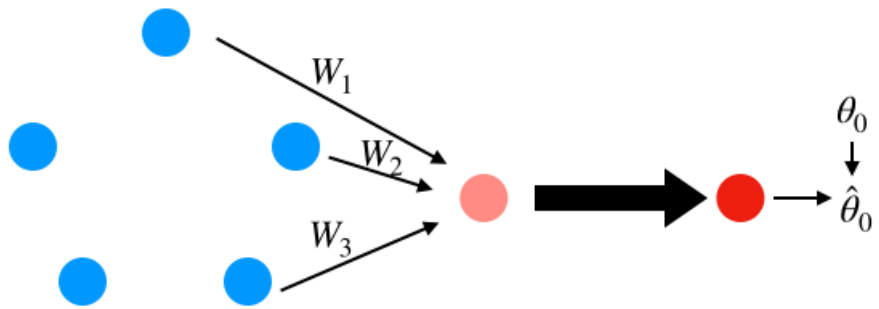
- Encode each task into a vector of dimension d.
- Learn a dependency graph with structure learner.

The input to tasks are different, but the output are the same.

Related tasks: meta-learning for heterogeneous tasks. [Automated relational meta-learning]

Main idea: learn a customized initialization for each task.

- Maintain a knowledge set: containing k knowledge basis.
- For each new task embedding, tap into the knowledge set to enhance the task embedding. (Via structure learner)
- Use enhanced task embedding to customize the task-specific initialization.



Goal: use the graph structure to improve meta learning

During meta-training:

- for the inner loop, loss consists of training loss and structure loss
- for the outer loop, loss is the validation loss.

Dataset:
Real Dataset: MiniImagenet, omniglot.
Synthetic Dataset: 1D, 2D curves from different distributions.

Baselines:

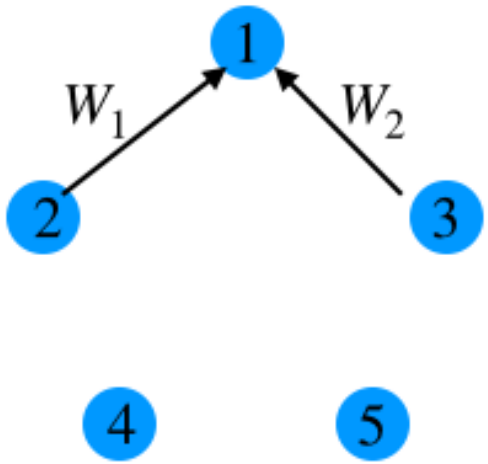
- Homogeneous meta-learning models: maml, meta-sgd.
- Heterogeneous meta-learning models: relational meta learning, Protonet, HSML.

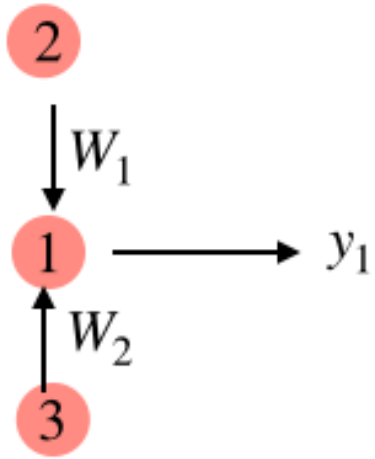
If we go into this direction:

- State-of-the-art accuracy
- Relational structure provides interpretations
- Do we need a distribution for the structure? What's the benefit of the probabilistic framework?
- A theory/ graphical diagram/information bottleneck explanation of the model will be a contribution.

The input to tasks are the same, but the output are different.
Related tasks: Multi-task learning.

Setting:
training set $\{x_i||y_{i1}, y_{i2}, \cdots y_{ik}\}$, test set $\{x_j\}$





Inner loop: update the dependency graph and enhance the individual task representation.
Outer loop: minimizing the loss w.r.t all tasks.

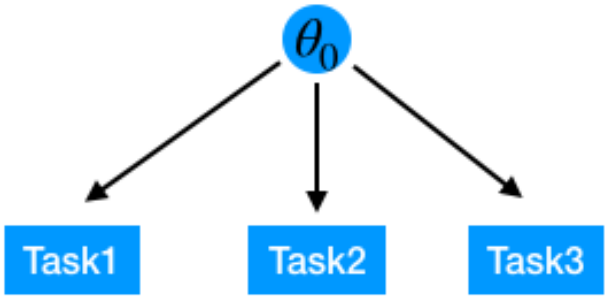
Dataset: ??

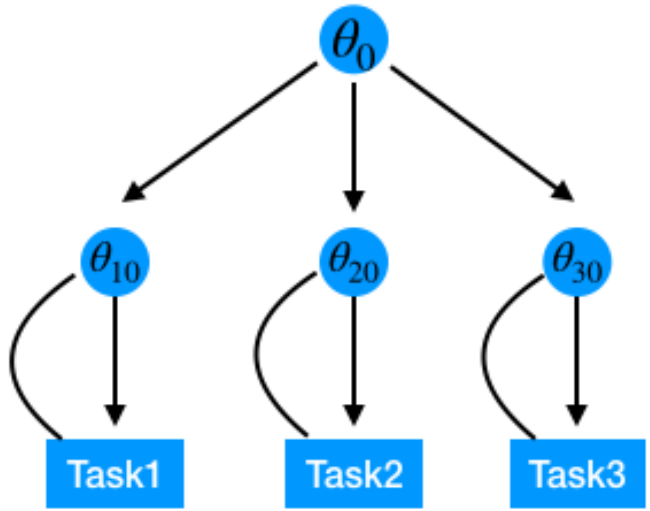
Baselines: multi-task learning models

Shared questions:

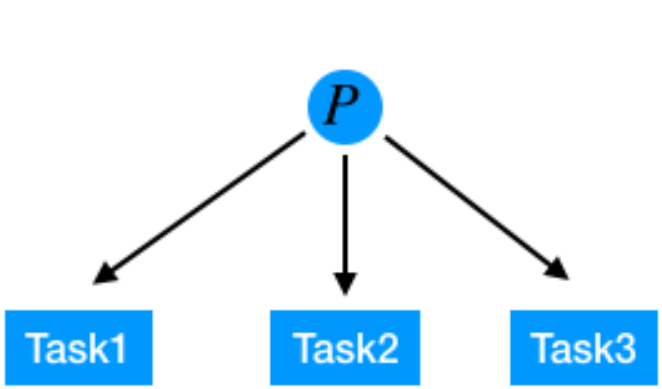
- How to encode a task?
- Do we want a binary relational structure? probabilistic model?

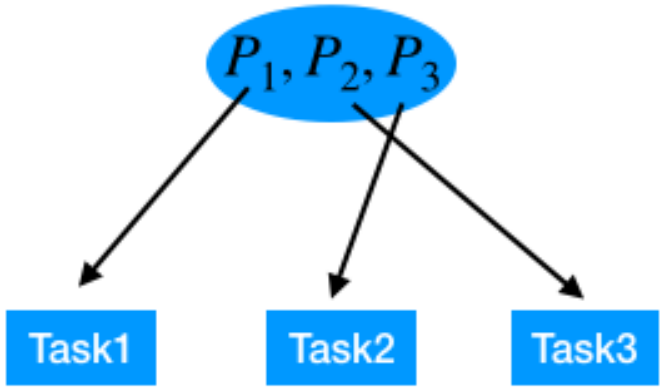
The shared initialization versus customized initialization.



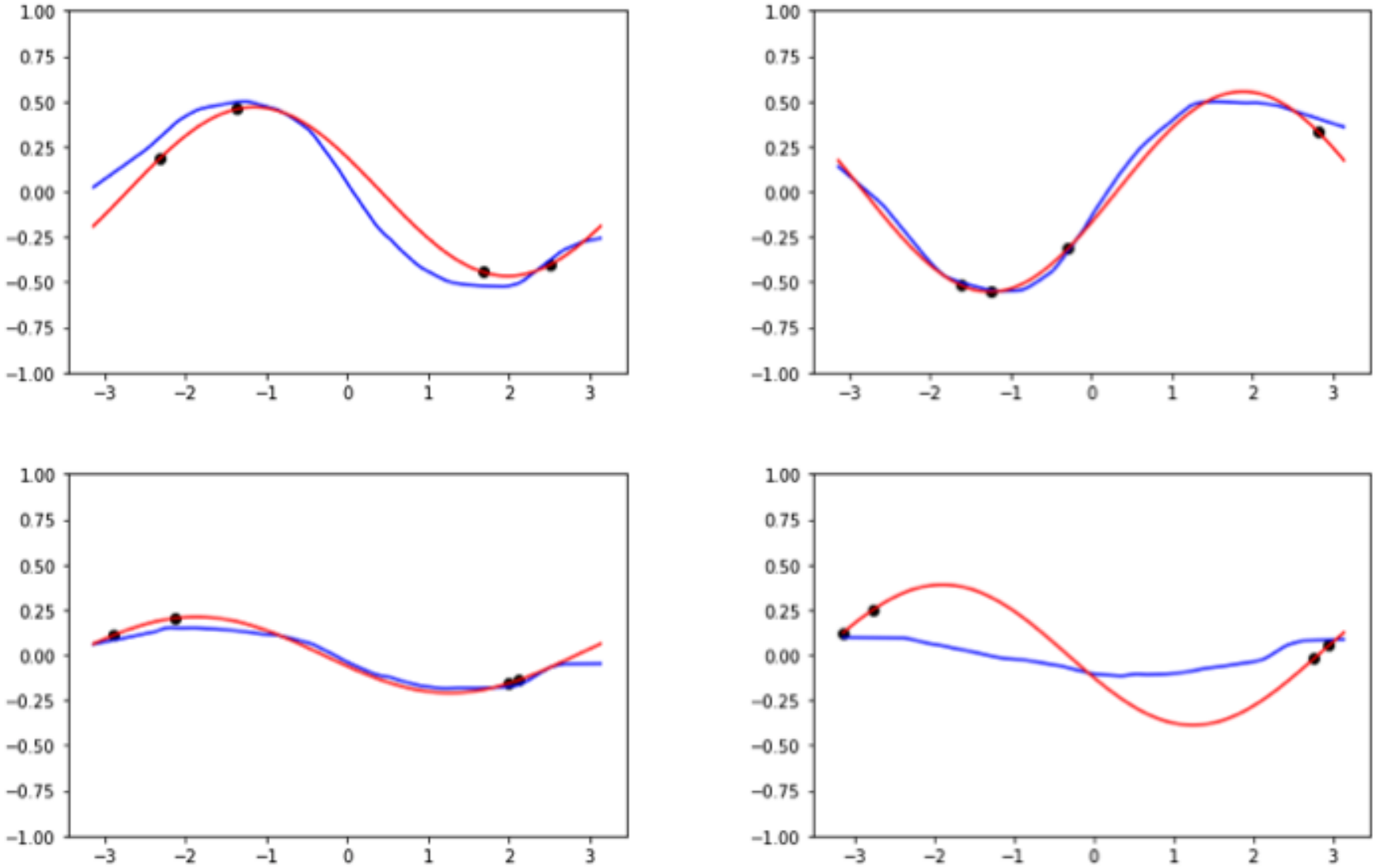


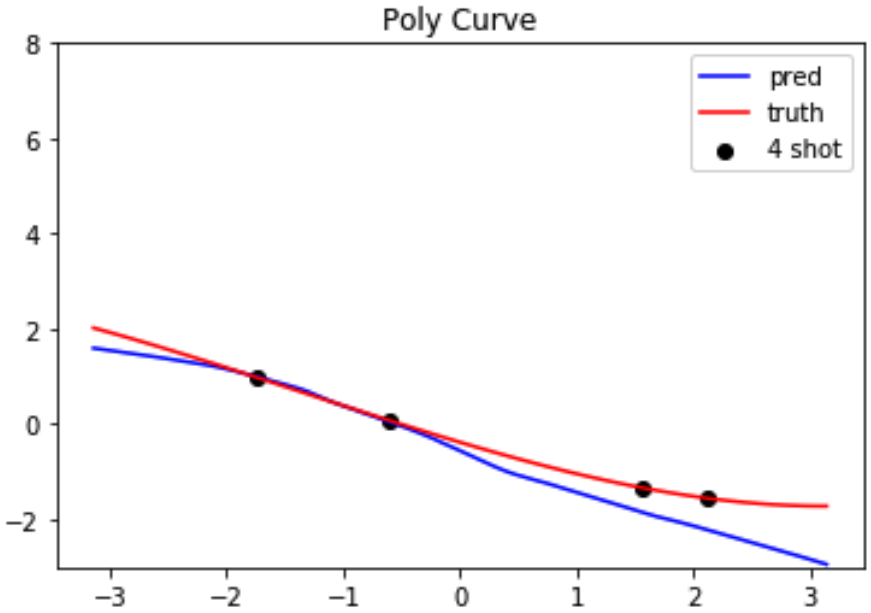
Homogeneous meta-learning versus heterogeneous meta-learning.

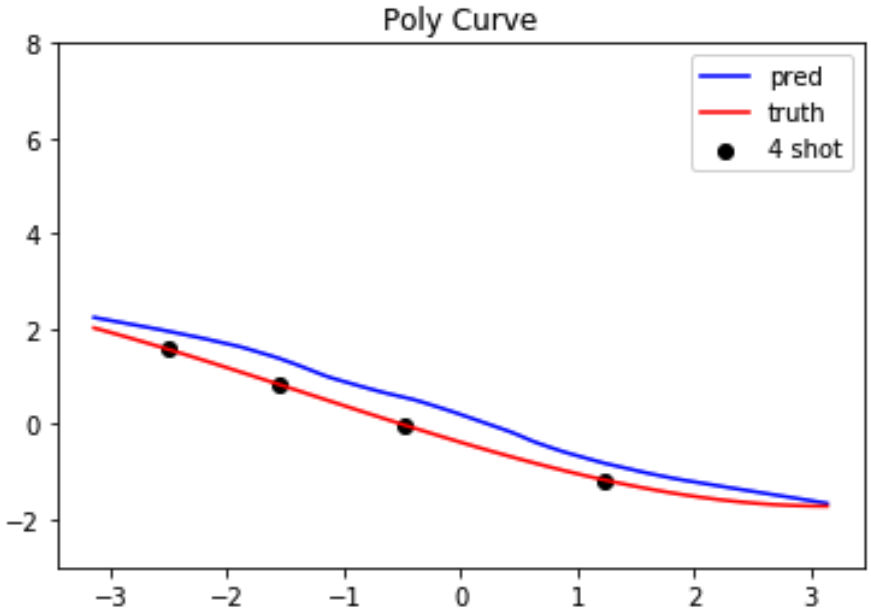


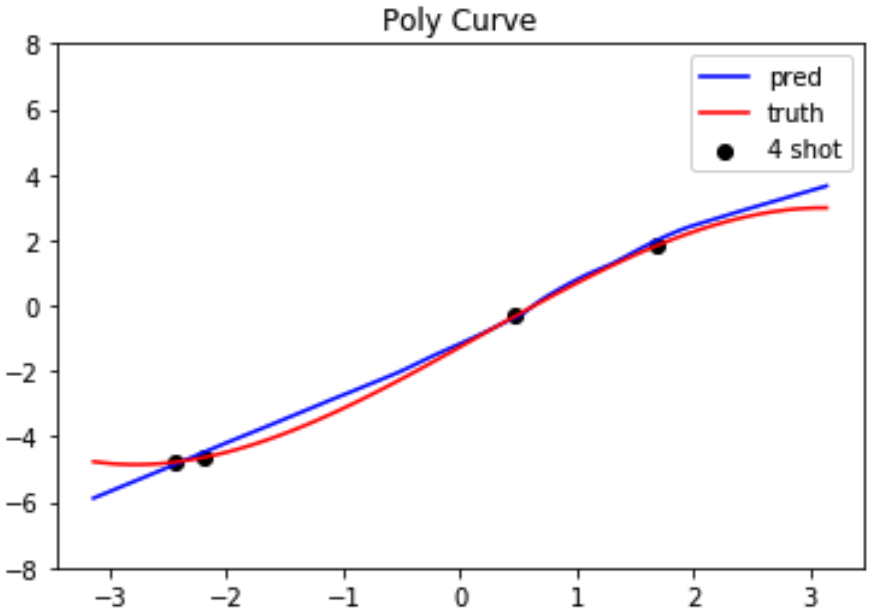


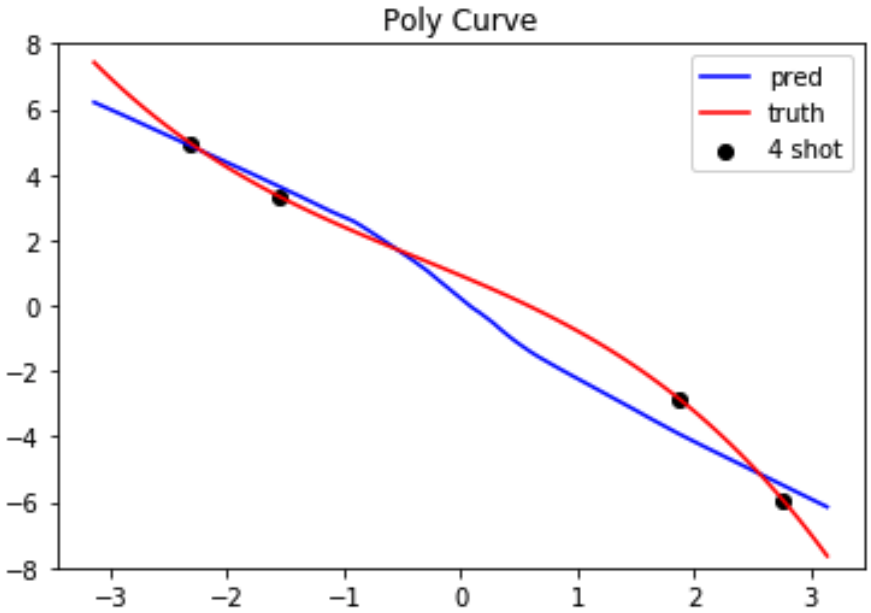
Homogeneous meta-learning with shared initialization.



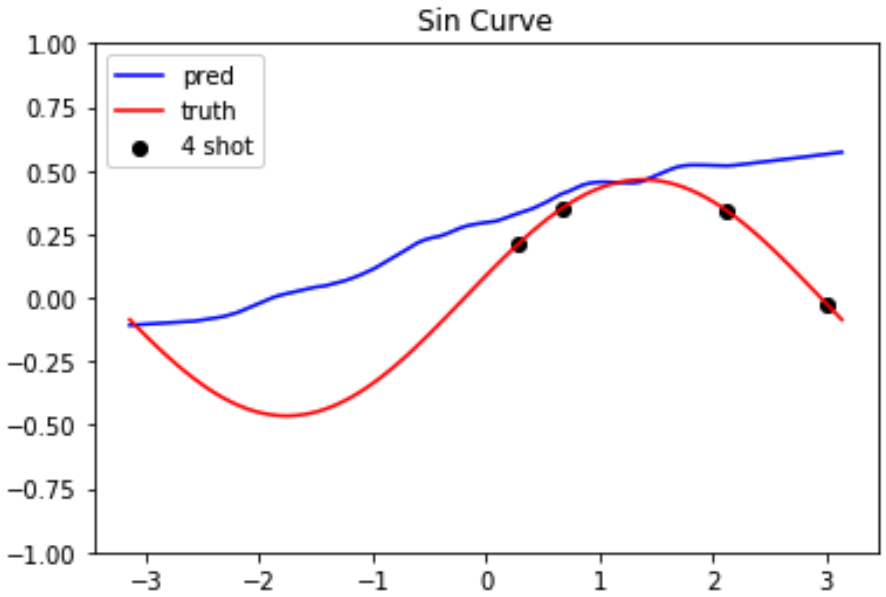

Poly Curve
pred
truth
4 shot

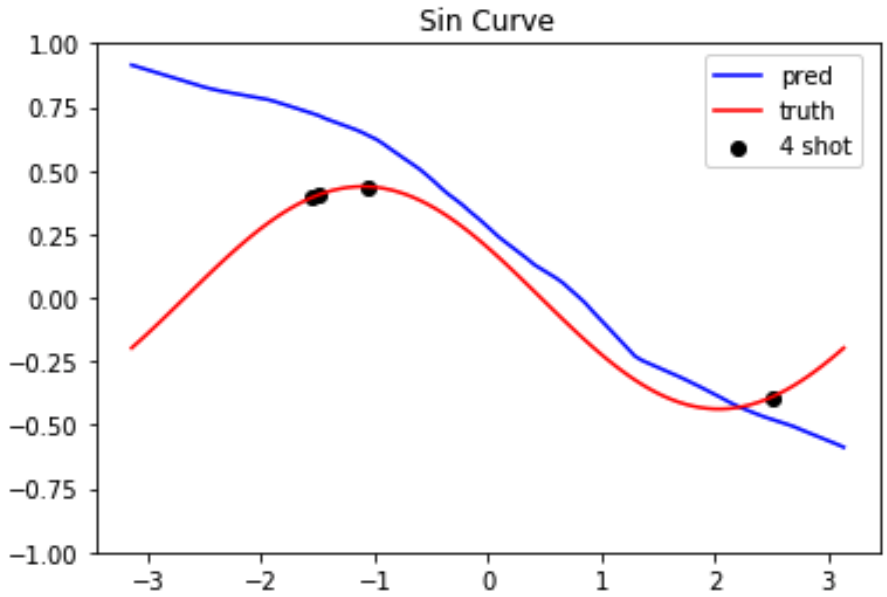

Poly Curve
pred
truth
4 shot

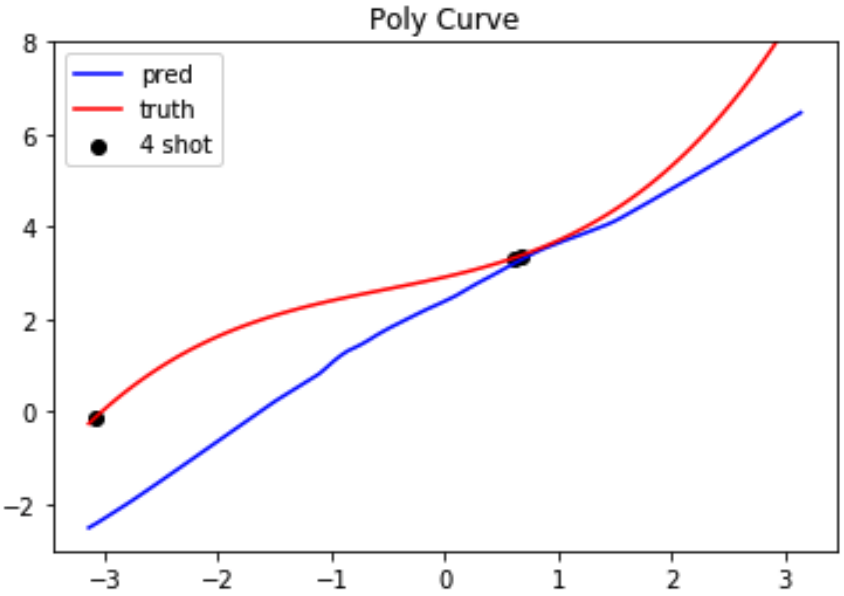

Poly Curve
pred
truth
4 shot

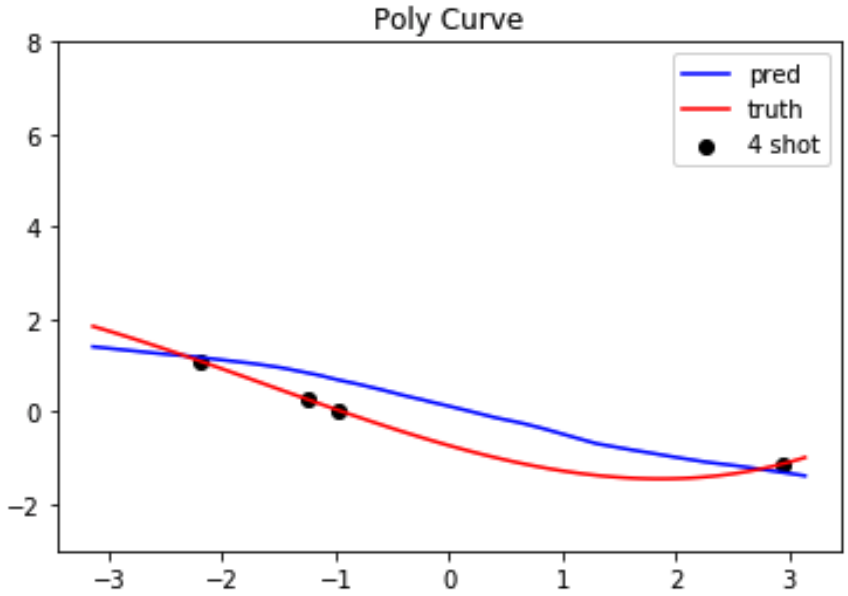

Poly Curve
pred
truth
4 shot

Heterogeneous meta-learning with shared initialization.









Next step

- Customized initialization for homogeneous and heterogeneous task.
- Challenge: How to encode a task?

Customized initialization

- Encode a task in order to customize the initialization.
- Training step.

Task Encoder

For task T_j, we have a support set $\{x^j_{s,i}, y^j_{s,i}\}_{i=1}^n$:

- Encode each pair of input and output:

$$c^j_i = MLP([x^j_{s,i}, y^j_{s,i}])$$

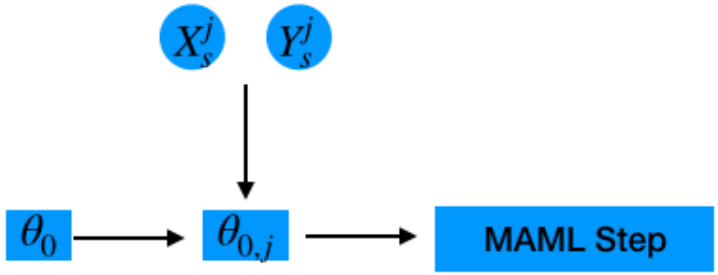
- Permutation invariant aggregator:

$$c^j = mean(c^j_i)$$

- Encode the initialization:

$$\theta_{0,j} = Enc(\theta_0, c^j)$$

Training step:



In the inner loop, the global initialization θ_0 will be optimized.
In the outer loop, all parameters will be optimized.

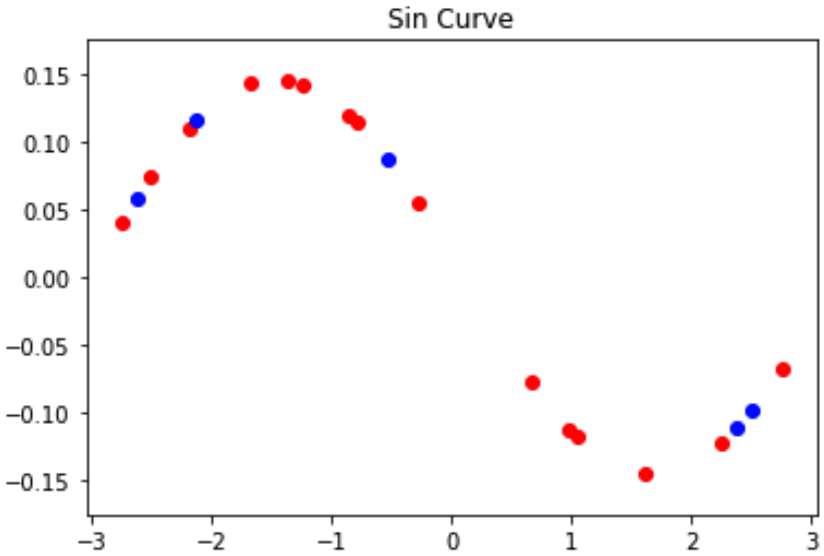
Experiment setting

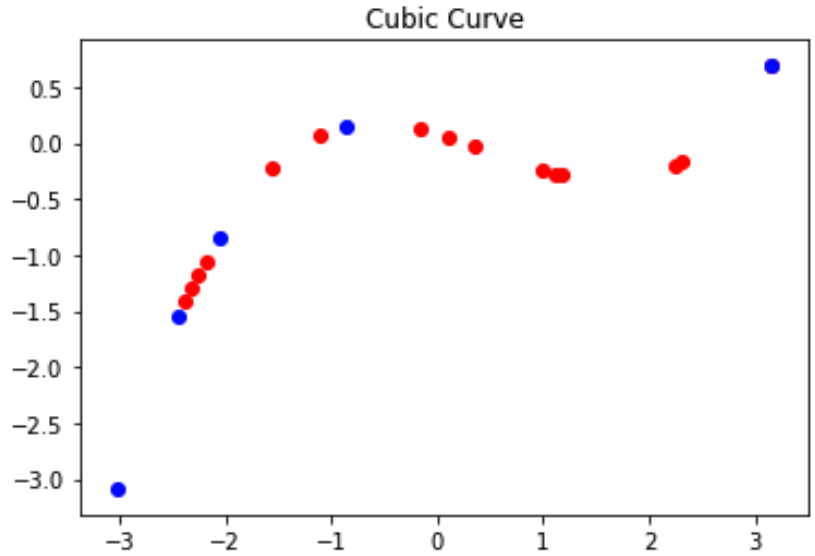
Few shot regression:

For each task T_j, the training(support) set is $\{x^j_{s,i}, y^j_{s,i}\}, i = 1, \cdots, 5$, the validation(query) set is $\{x^j_{v,k}, y^j_{v,k}\}, i = k, \cdots, 15$

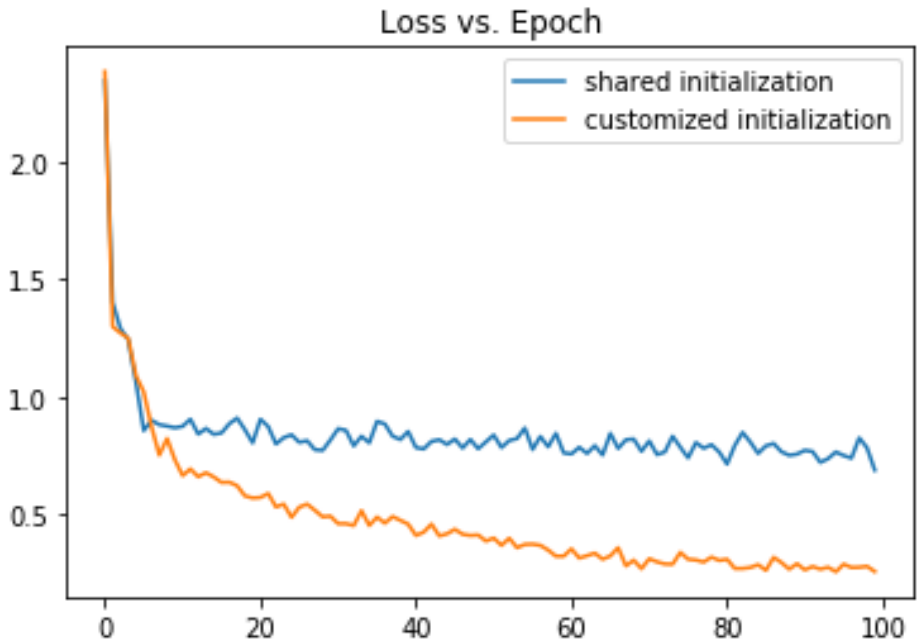
In heterogeneous setting, T_j are sampled from different distributions: sinusoidal curve family and cubic curve family.

In homogeneous setting, T_j are sampled from the same distribution.

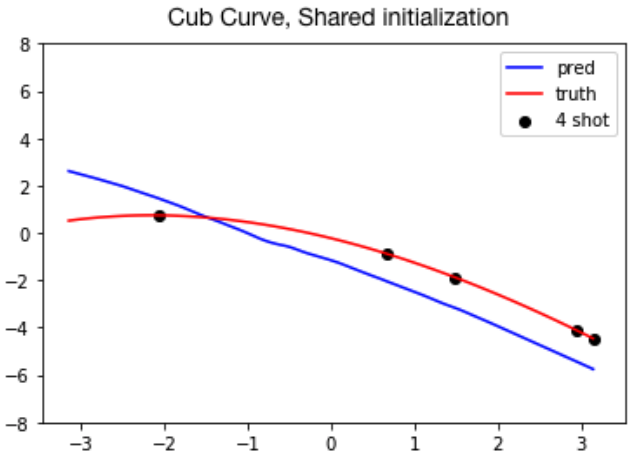

Sin Curve
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−3
−2
−1
0
1
2
3

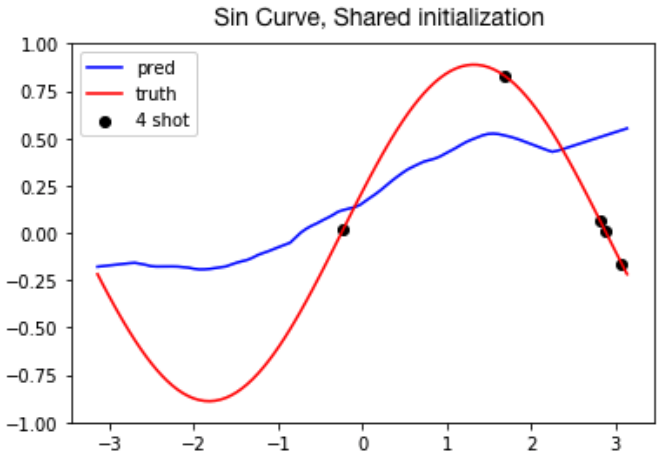

Cubic Curve
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3
−2
−1
0
1
2
3

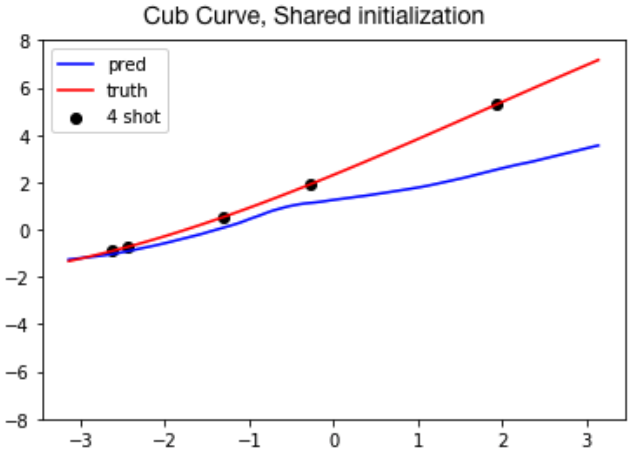
For better comparison, we fix the predictor as MLP $(1 \to 40 \to 40 \to 1)$, fix the optimization methods.
Loss curve:

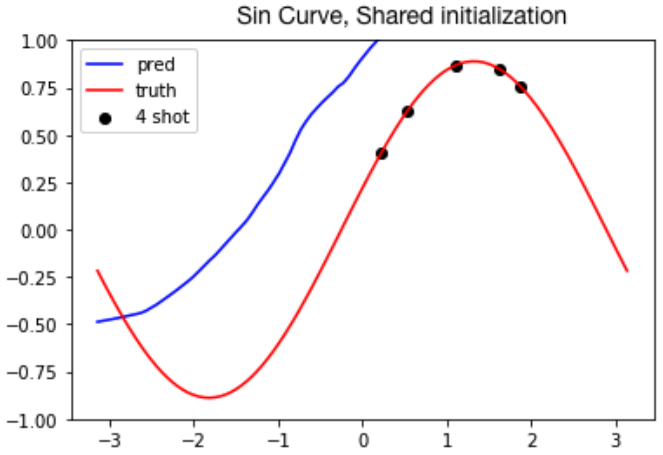


During meta test, for each task T_j, the training(support) set is $\{x^j_{s,i}, y^j_{s,i}\}, i = 1, \cdots, 5$, the validation(query) set is $\{x^j_{v,k}\}, i = k, \cdots, 1000$
Heterogeneous task with shared initialization.



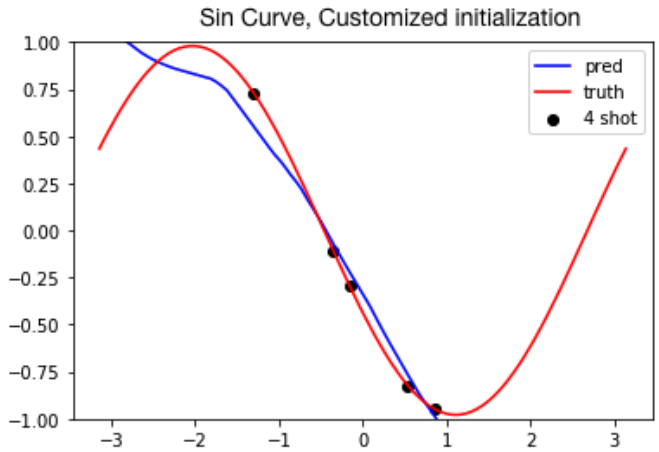


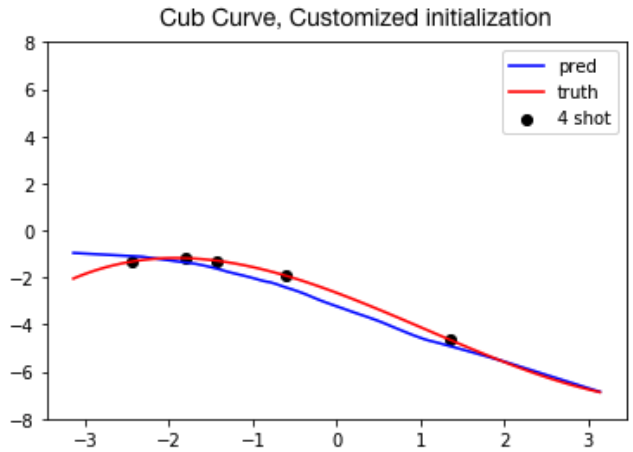


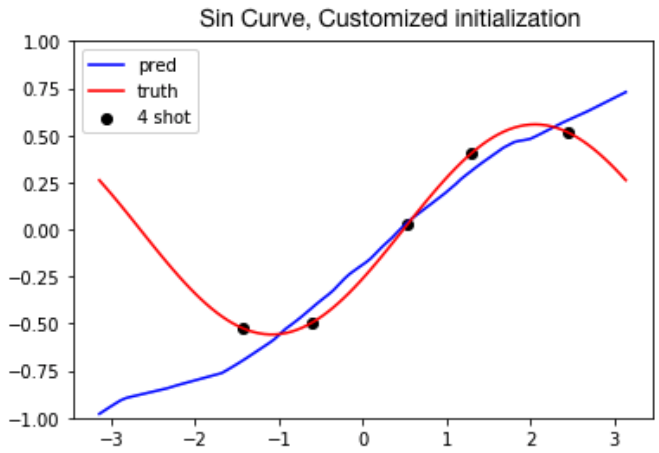


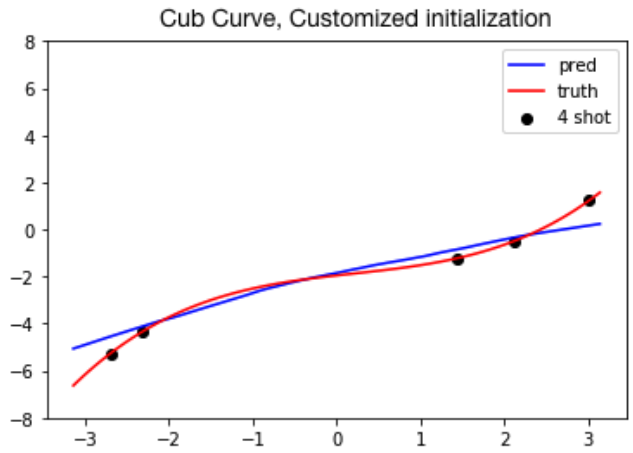
Average MSE loss: 0.79

Heterogeneous task with customized initialization.









Average MSE loss: 0.22

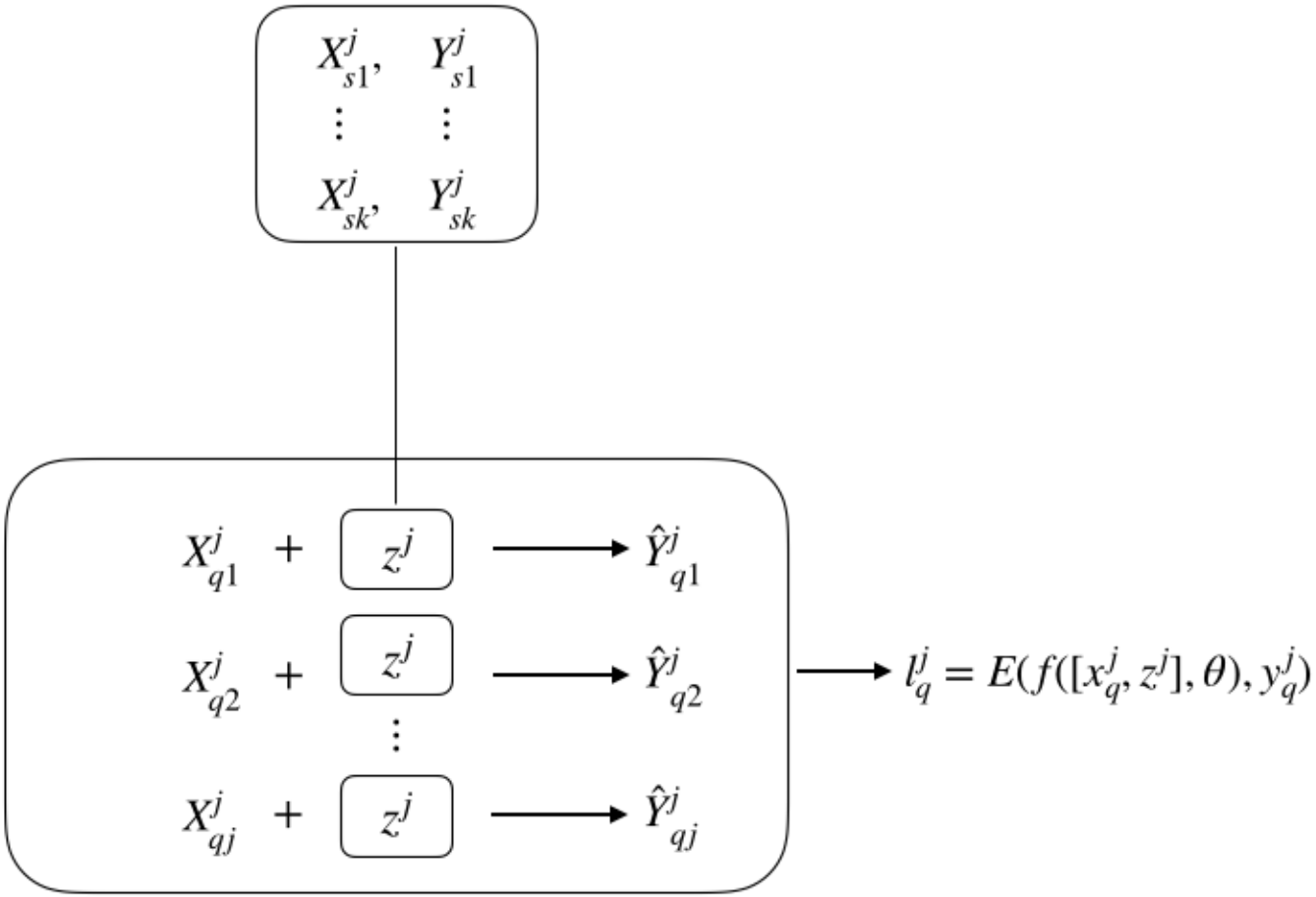
Summary:

Main idea:
Rather than using the shared initialization, for each specific task, the initialization is customized.

Conclusion:
Customized initialization will benefit meta learning tasks , including heterogeneous setting and homogeneous setting.

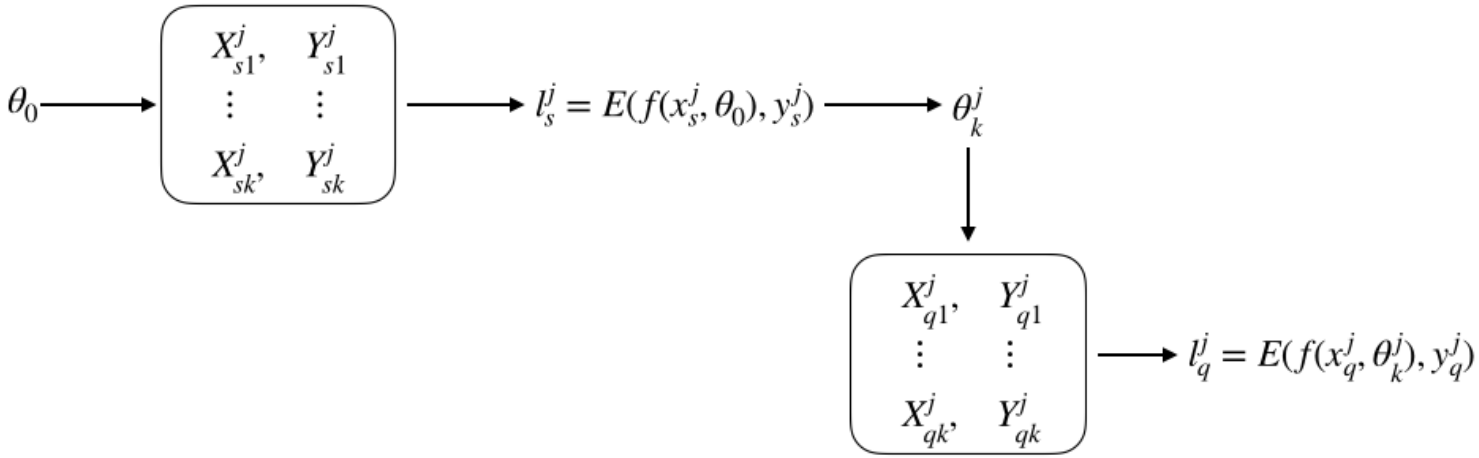
Model Comparison

Neural Processes:



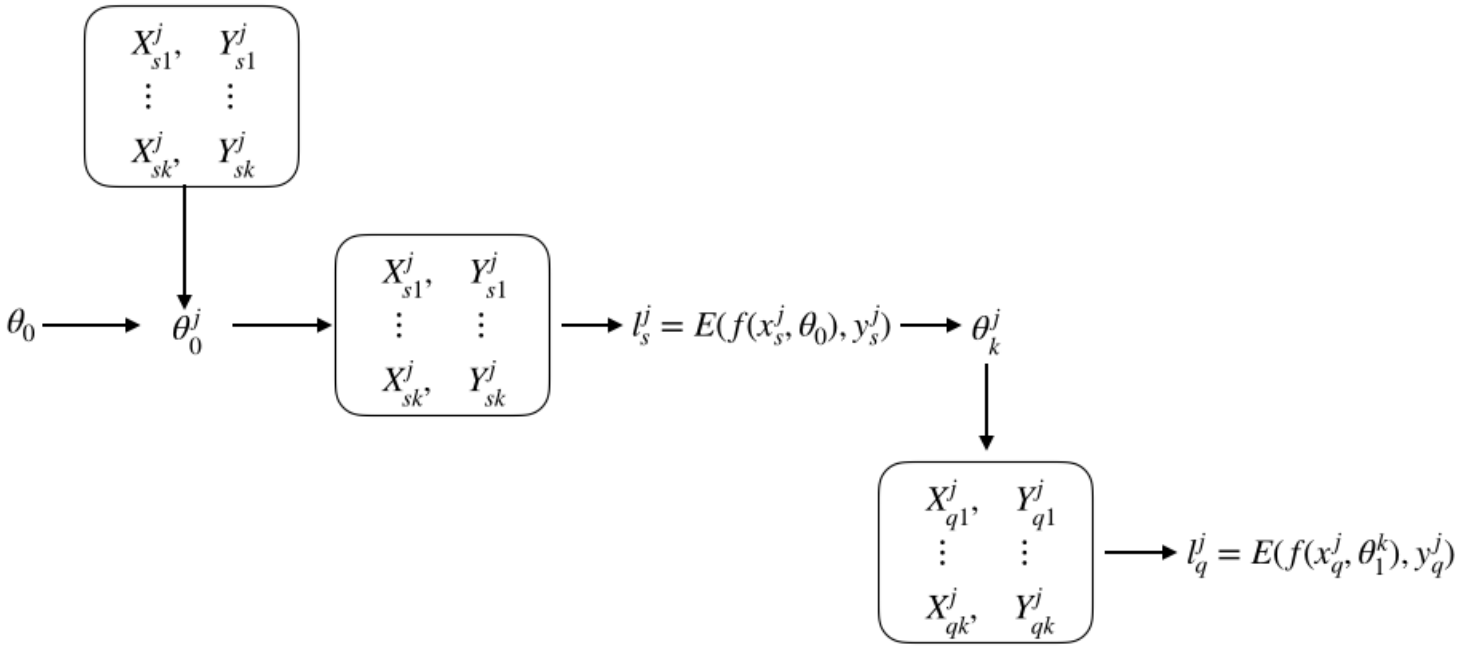
Model Comparison

MAML:



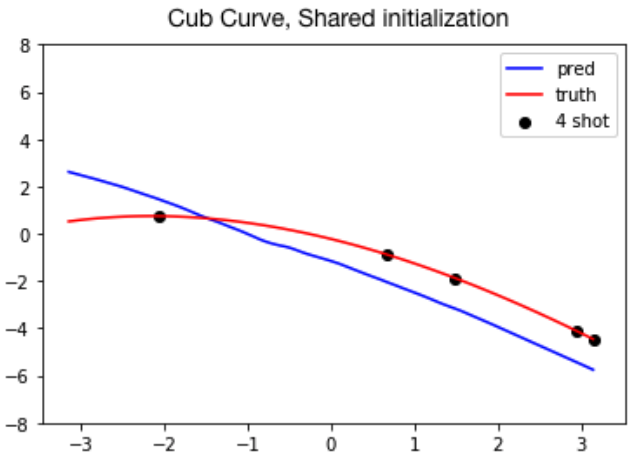
Model Comparison

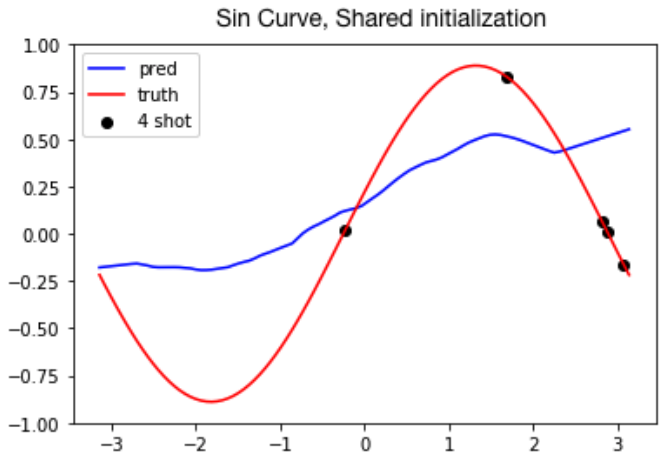
Customized MAML:

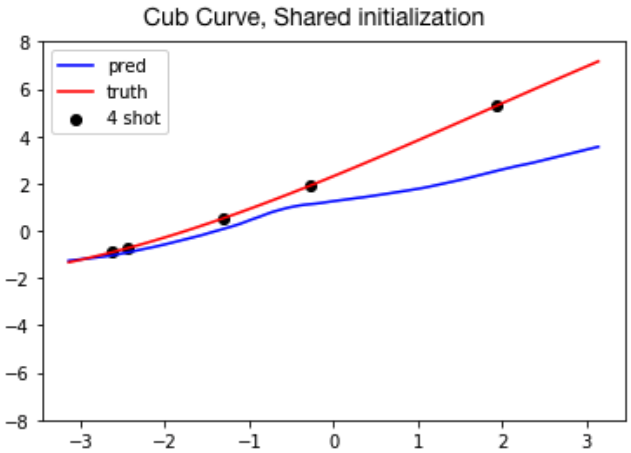


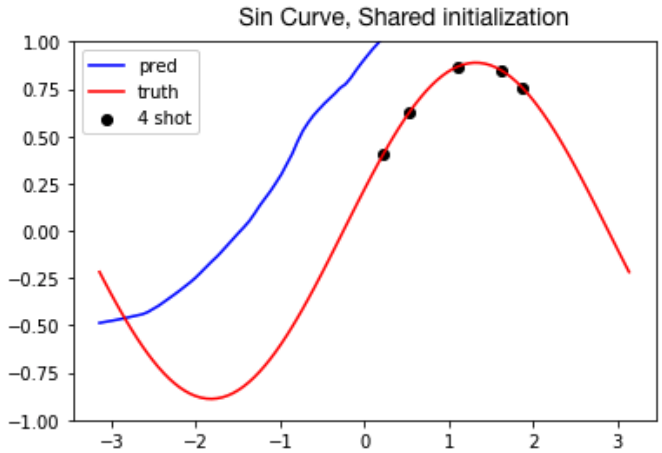
Model Comparison

MAML:



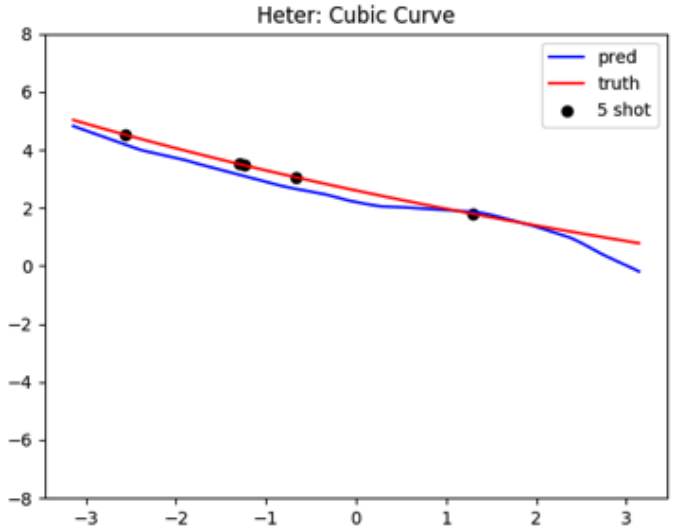


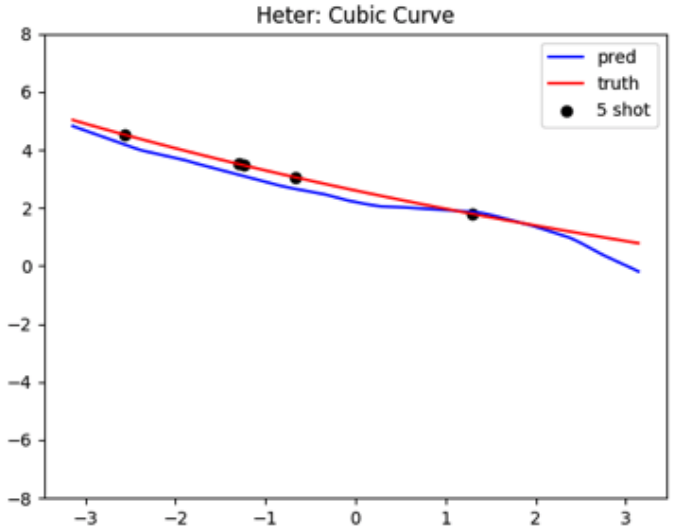


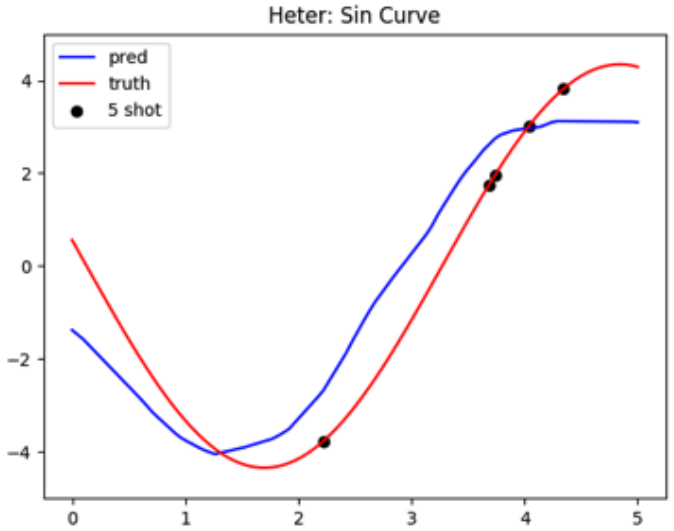


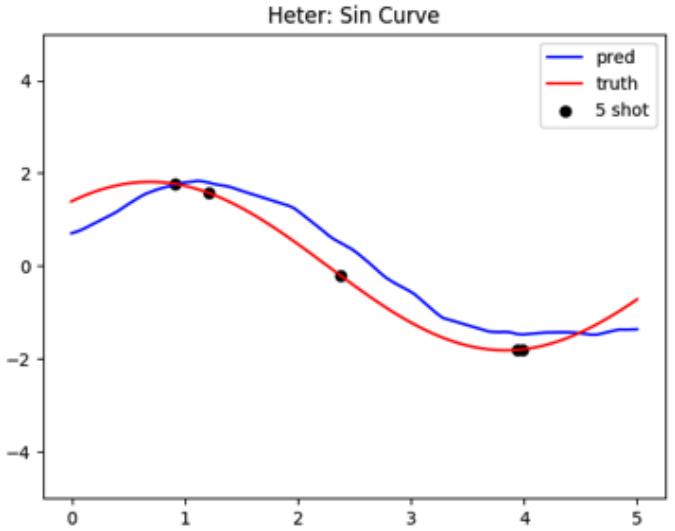
MSE: 0.79

Neural Processes:
Two distributions, no noise.



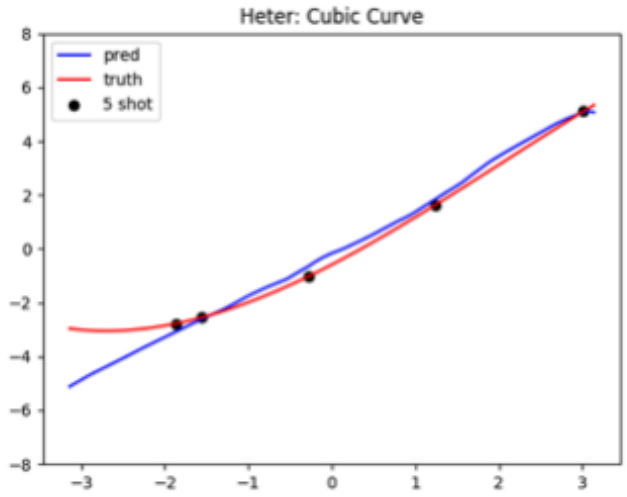


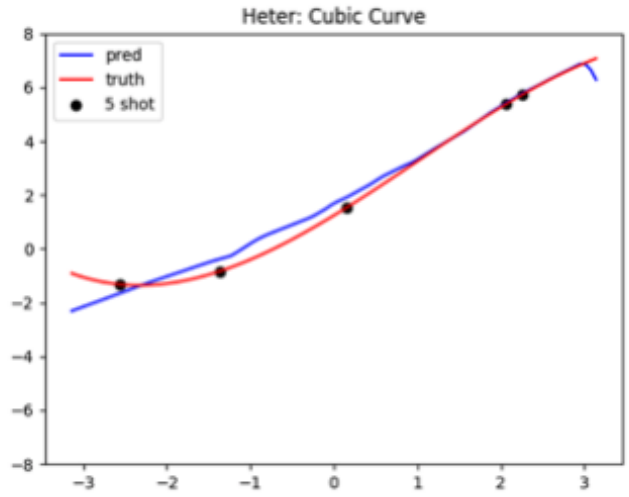


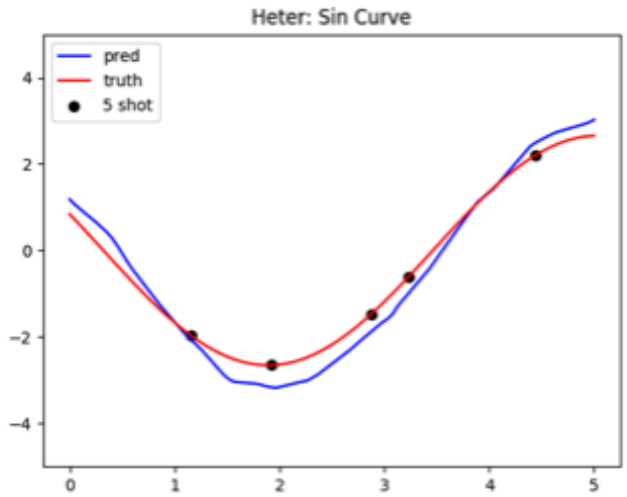


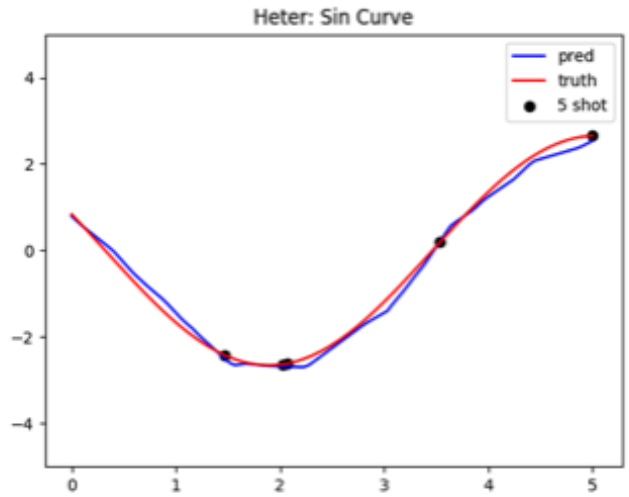
MSE: 0.58

Customized MAML:
Two distributions, no noise.









MSE: 0.32

Three distributions, with gaussian noise.
MAML: 1.231, Multi MAML: 0.713 MMAML: 0.444, Cutomized MAML: 0.501

Two ways to improve the model:

- Use more complex model, add tricks for training process
- Introduce task-task relation into the model

Heterogeneous Meta Learning with Neural Processes

Meta learning requires knowledge generalizing from previous tasks to future task.

Homogeneous meta learning assumes tasks are sampled from the same distribution.

A critical challenge is the task heterogeneity, where tasks are sampled from different distributions.

Two solutions including: meta learning, neural processes

Stochastic Process

Consider the supervised setting, the prior information that a practitioner may have about f is specified via the architecture of model, the loss function, or the training details.

Limitations:

- extent of prior knowledge that can be expressed is limited
- the hand designing of the prior knowledge
- hard to transfer the knowledge across different tasks
- need large training set for each task.

Stochastic process

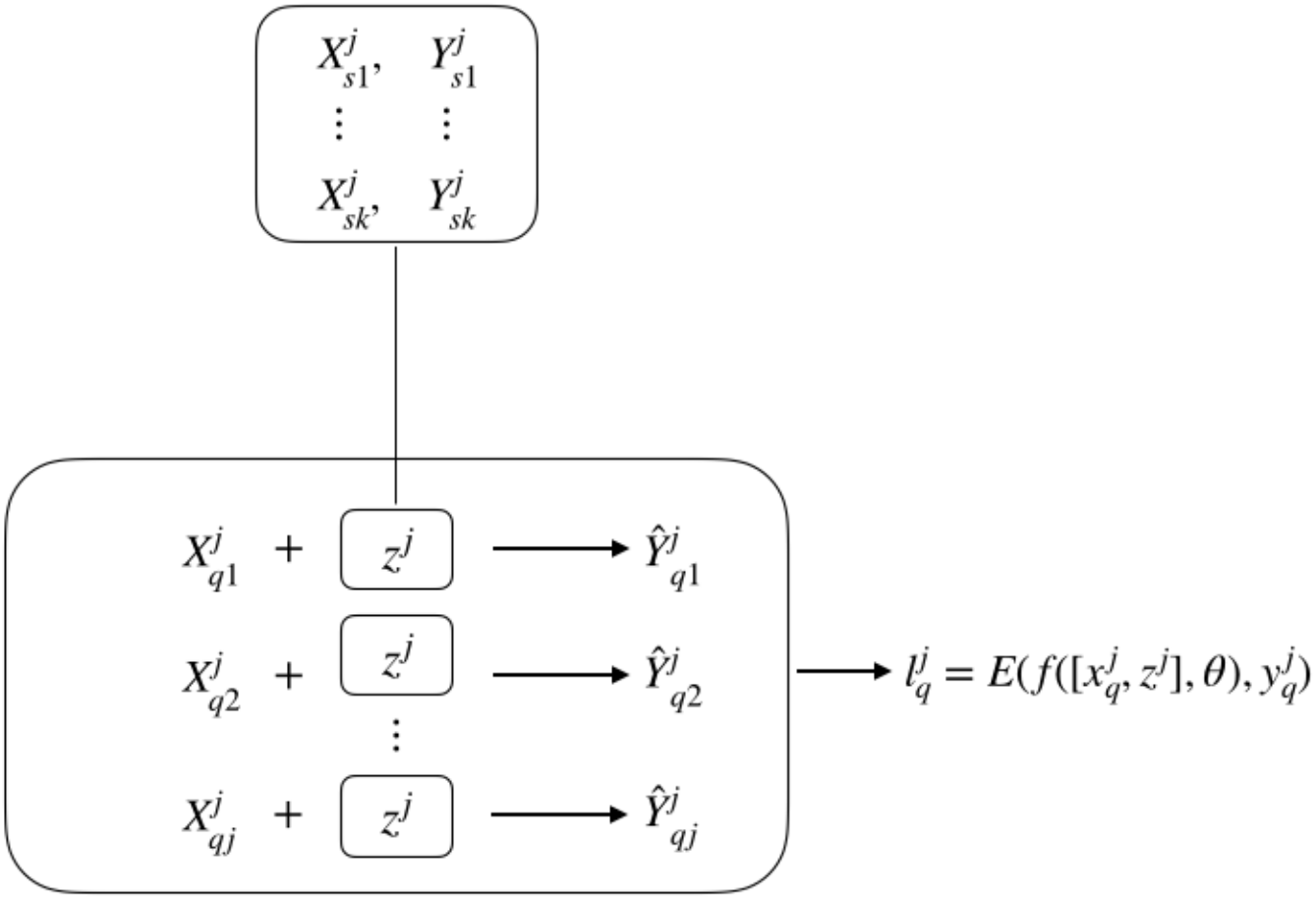
Stochastic process is another way where we assume a distribution for the mapping f. The prior knowledge is specified by the prior distribution and the learning corresponds to Bayesian inference.

Friendly to limited training set, Gaussian Process is one example. Specify the mean and kernel function, it is data-efficient.

Prior: $P(f(Tr), f(Te))$, Posterior: $P(f(Te)|X_{Te}, f(Tr))$

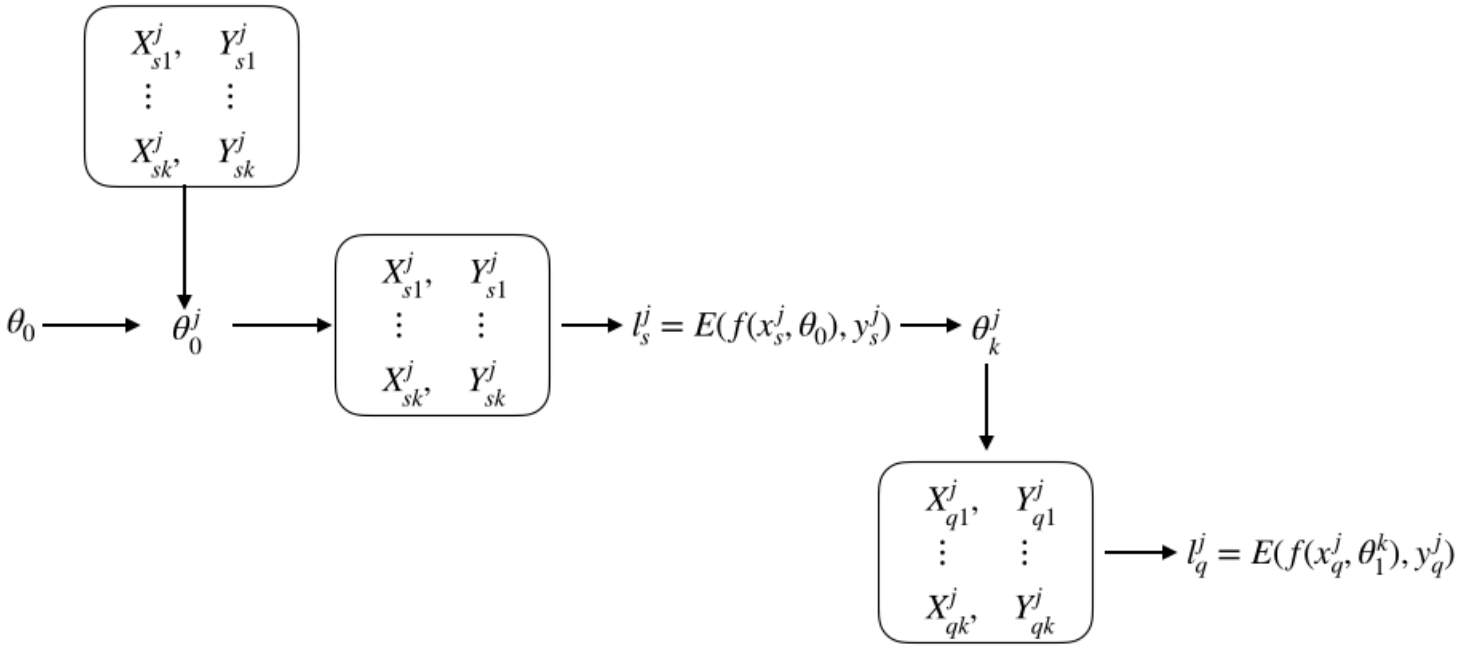
Model Comparison

Neural Processes:

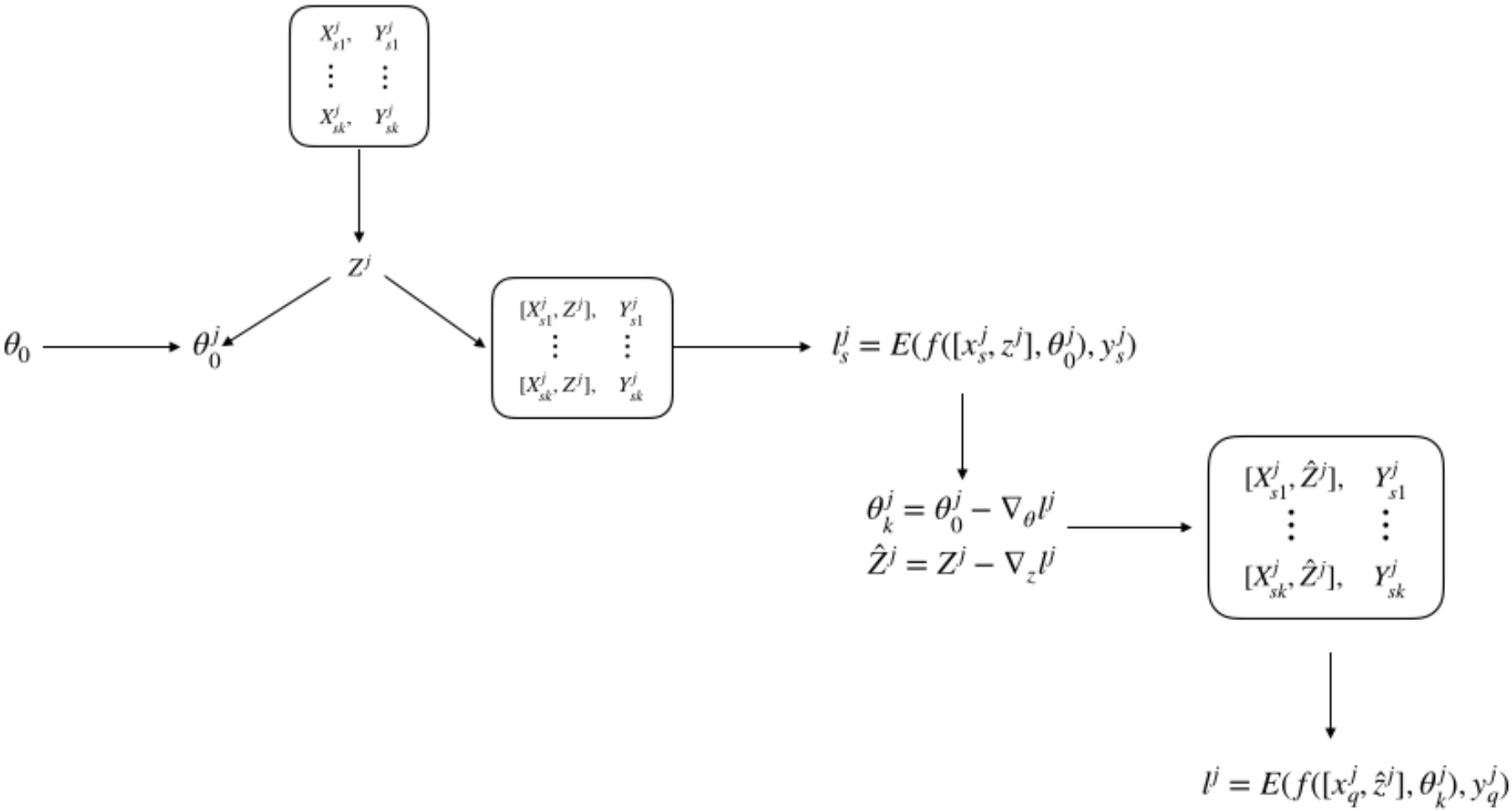


Model Comparison

Customized MAML:



Model


$\begin{bmatrix} X^j_{s1}, & Y^j_{s1} \\ \vdots & \vdots \\ X^j_{sk}, & Y^j_{sk} \end{bmatrix}$
Z^j
θ_0
θ^j_0
$\begin{bmatrix} [X^j_{s1}, Z^j], & Y^j_{s1} \\ \vdots & \vdots \\ [X^j_{sk}, Z^j], & Y^j_{sk} \end{bmatrix}$
$l^j_s = E(f([x^j_s, z^j], \theta^j_0), y^j_s)$
$\theta^j_k = \theta^j_0 - \nabla_\theta l^j$
$\hat{Z}^j = Z^j - \nabla_z l^j$
$\begin{bmatrix} [X^j_{s1}, \hat{Z}^j], & Y^j_{s1} \\ \vdots & \vdots \\ [X^j_{sk}, \hat{Z}^j], & Y^j_{sk} \end{bmatrix}$
$l^j = E(f([x^j_q, \hat{z}^j], \theta^j_k), y^j_q)$

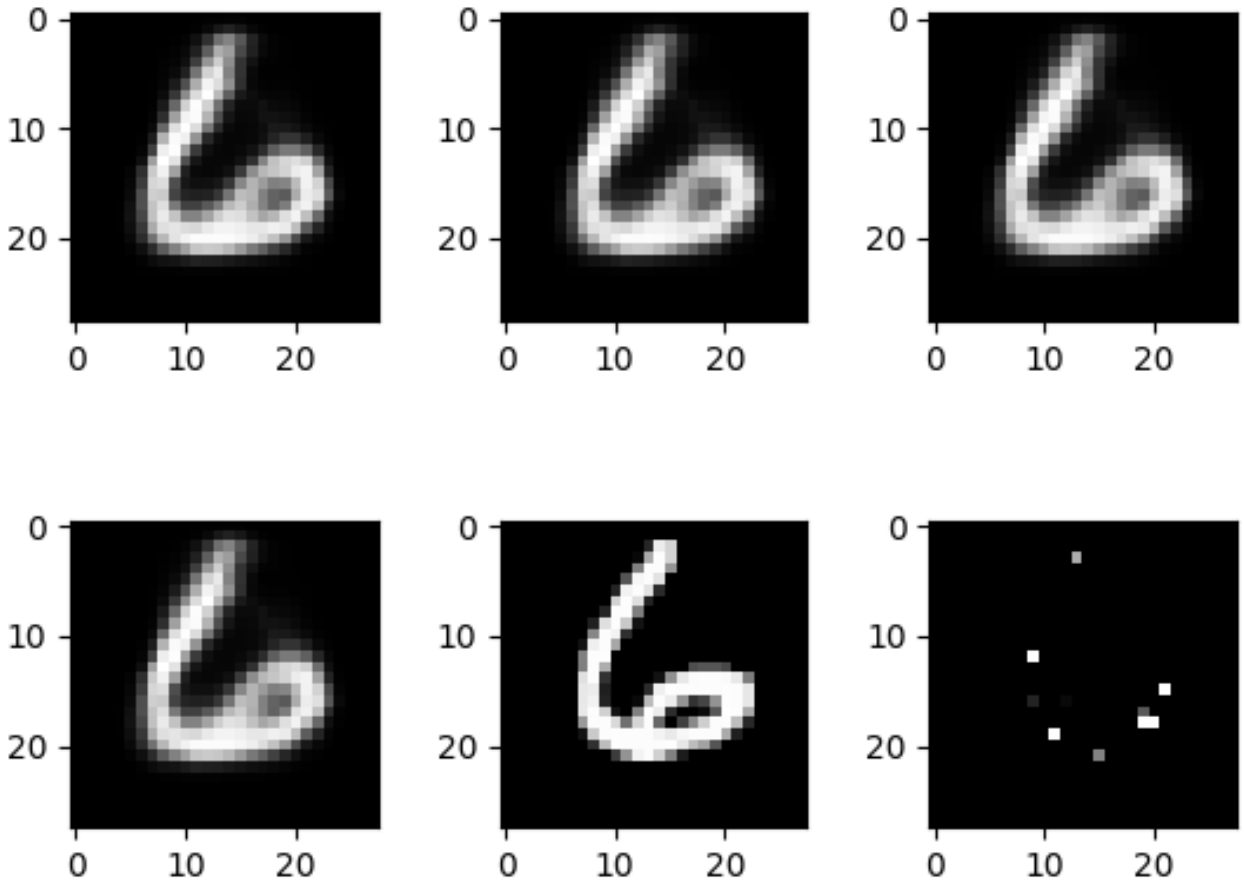
Experiments

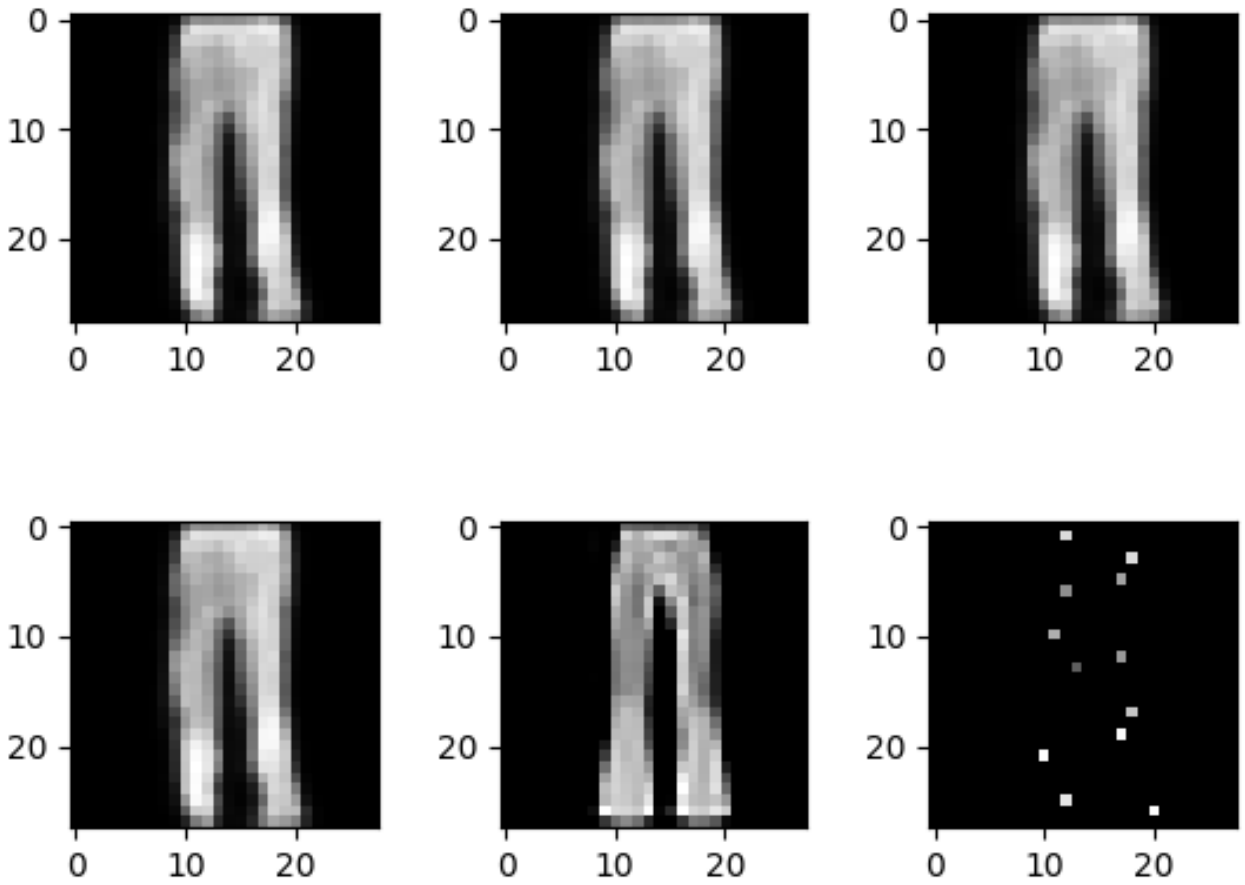
- Function regression, baselines: NP, HML, MMAML, ARML
- Image completion, baselines: NP, CNP
- Classification, baselines: HML, MMAML, ARML

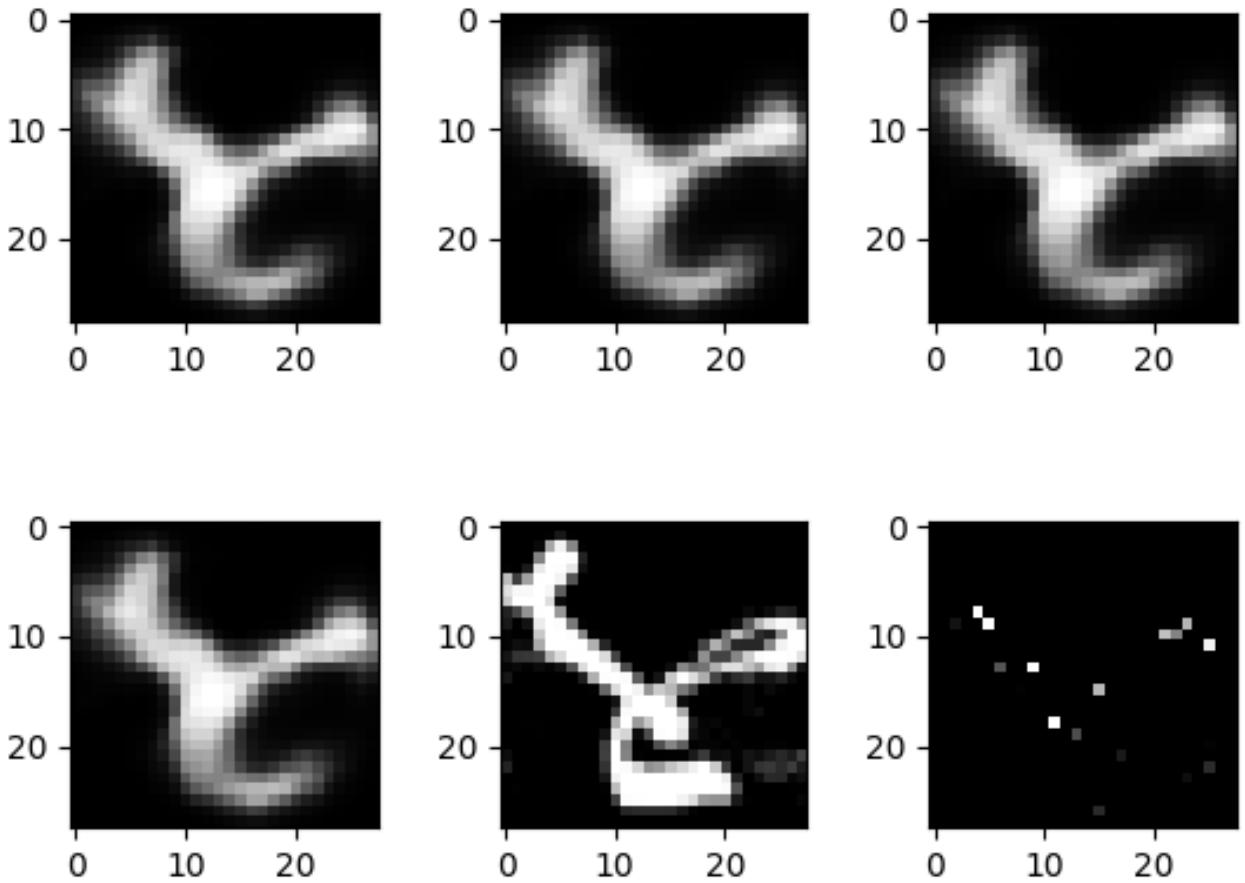
Regression results

Modes	MAML	MultiMAML	MUMOMAML	ARML	ML+NP
2	1.085	0.433	0.336		0.21
3	1.231		0.444		0.30
6	2.29	2.91	0.52	0.44	0.39

Data are sampled from three different distributions: MNIST, KMNIST, FMNIST.




0
10
20
0
10
20


0
10
20
0
10
20

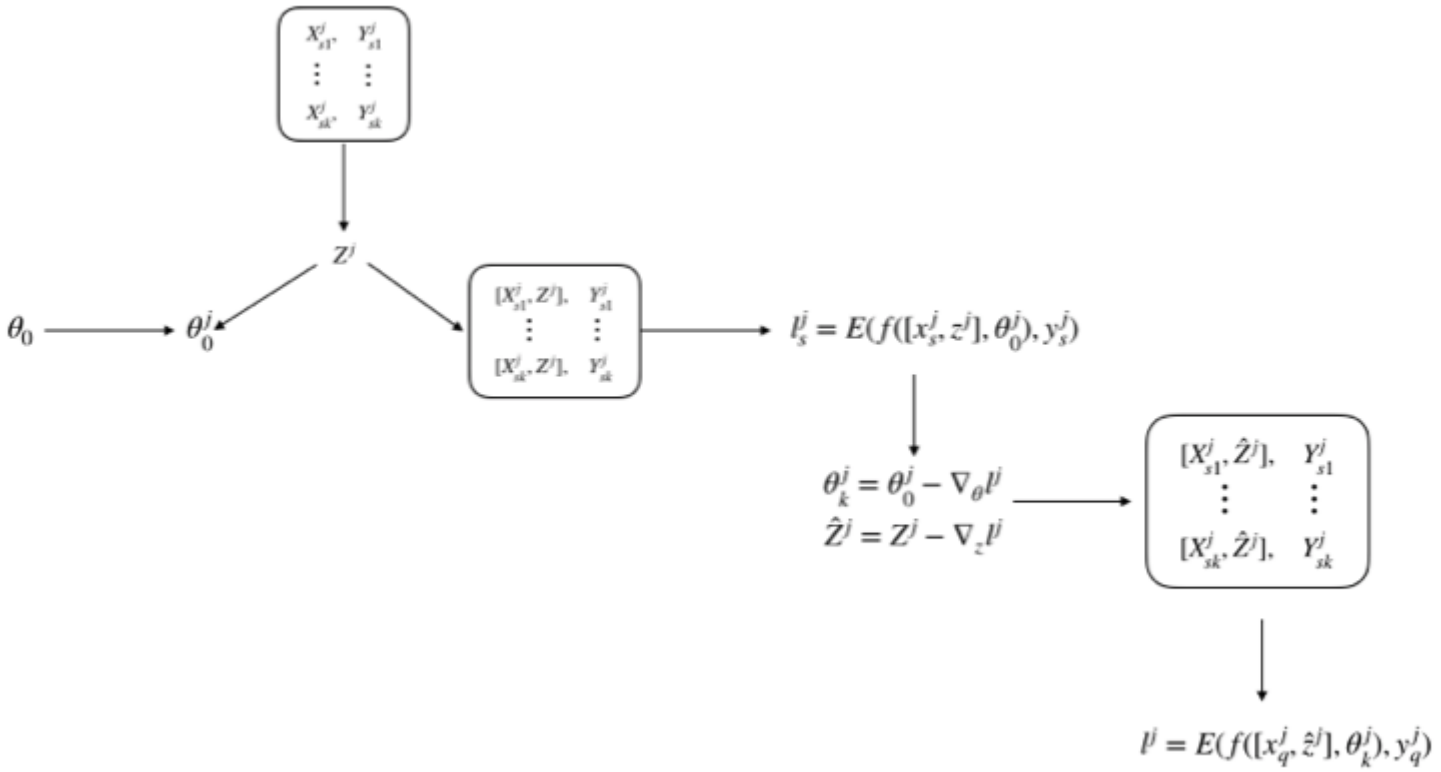
Probabilistic Model

Task ambiguity is a critical challenge for few shot learning

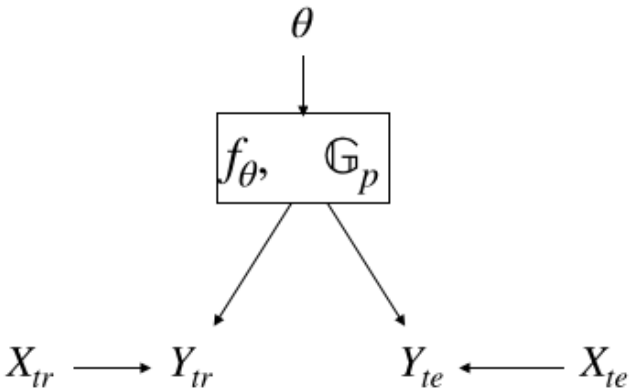
A single model can not explain the diversity of the data, the challenge is even more critical under heterogeneous setting.

Probabilistic Model

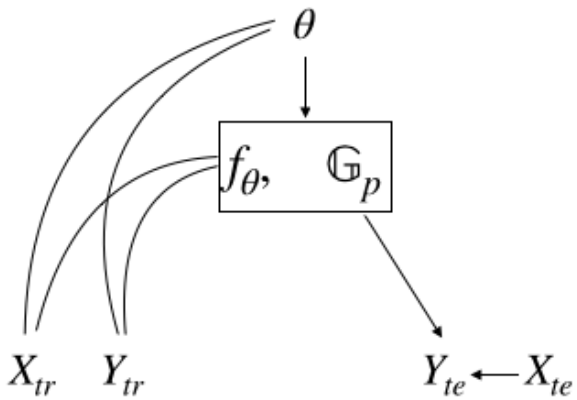
Solution:
Probabilistic version of the proposed model, which allows for model sampling.
Model architecture:



Graphical Model



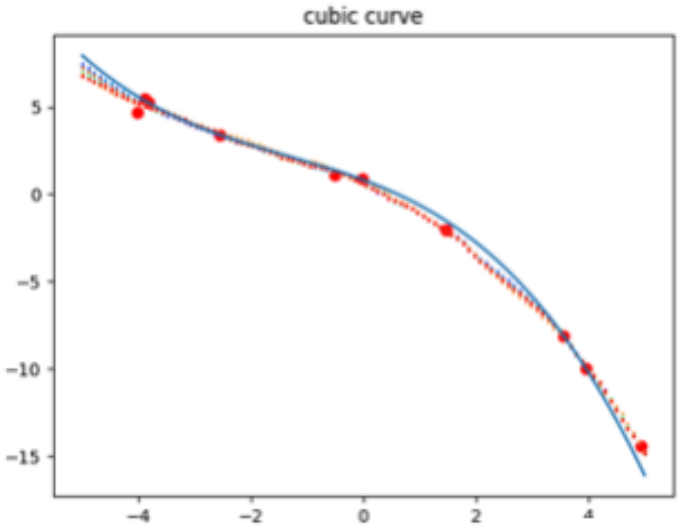
Original graphical model



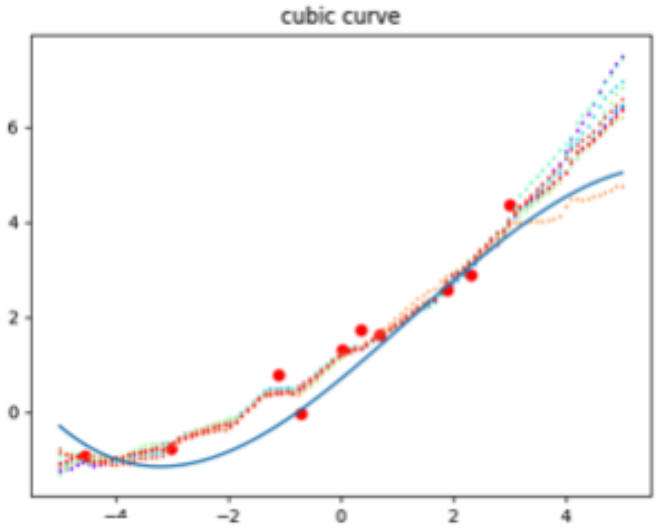
Deterministic graphical model

Probabilistic graphical model

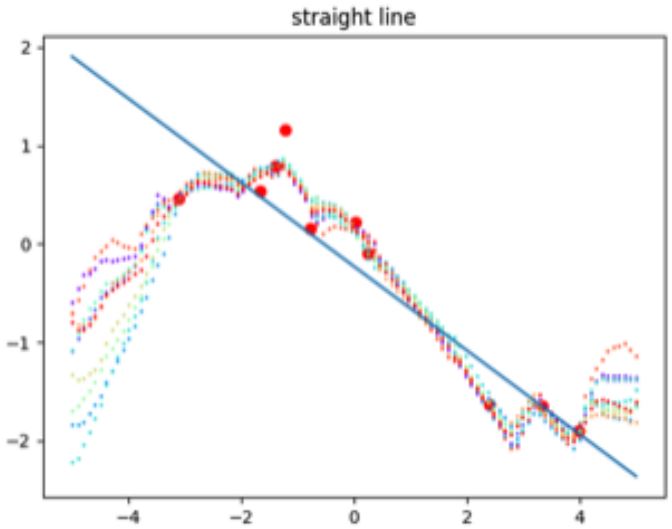
MSE for deterministic model: 0.389
MSE for probabilistic model: 0.415



noise=0.1



noise=0.2



noise=0.6

- If the noise is small, the sampled models are quite similar and are close to the underlying truth.
- If the noise is relatively large, the sampled models are diverse and show characteristics of different functions.
- If the keep increasing the noise, the trained model will generate non-smooth predictions to minimise MSE.

Regression on real world dataset
BCE loss for deterministic model: 0.270
BCE loss for probabilistic model: 0.273
BCE loss for NP: 0.295
BCE loss for CNP: 0358

Visualization

Inter-class variation

Cross-domain variation

Inter-domain variation