Learning to Learn to Communicate

Osvaldo Simeone
Joint work with Sangwoo Park, Sharu Theresa Jose, Hyeryung Jang, and Joonhyuk Kang

MLCOM 2020, 7/9/2020
Meta-Learning to Communicate

Osvaldo Simeone
Joint work with Sangwoo Park, Sharu Theresa Jose, Hyeryung Jang, and Joonhyuk Kang

MLCOM 2020, 7/9/2020
Overview

- Motivation
- Meta-learning in a nutshell
- Algorithms and applications
Overview

• Motivation

• Meta-learning in a nutshell

• Algorithms and applications
Motivation

• In the Internet of Things (IoT), devices transmit sporadically using short packets with few pilot symbols.
Motivation

• Conventional model-based approach: estimate the (linear) fading channel and then use it in an optimal coherent demodulator.

• Model deficit (e.g., transmitter’s hardware imperfections) $\rightarrow$ machine learning (ML)
Motivation

• Conventional model-based approach: estimate the (linear) fading channel and then use it in an optimal coherent demodulator.

• Model deficit (e.g., transmitter’s hardware imperfections) → machine learning (ML)

• ML requires enough pilot data from each device:

  Can ML tools be useful with few pilots per device? (sample complexity)
Motivation

• **End-to-end training for communication on known channel model to tackle an algorithm deficit**
Motivation

• End-to-end training for communication on known channel model to tackle an algorithm deficit

Since the channel model is known, data can be generated at will, but training must be redone for each channel realization…
Motivation

- **End-to-end training for communication on known channel model** to tackle an algorithm deficit

- Since the channel model is known, data can be generated at will, but training must be redone for each channel realization…

Can ML tools be useful with few training iterations per channel realization? (iteration complexity)
Overview

• Motivation

• Meta-learning in a nutshell

• Algorithms and applications
Learning to Demodulate from Few Pilots via Offline and Online Meta-Learning
S. Park, H. Jang, O. Simeone, and J. Kang

Model-agnostic meta-learning for fast adaptation of deep networks
We propose an algorithm for meta-learning that is model-agnostic, in the sense that it is compatible with any model trained with gradient descent and applicable to a variety of different learning problems, including classification, regression, and reinforcement learning…

1997

Osvaldo Simeone
Meta-Learning to Communicate
Conventional Machine Learning

model class + training procedure

inductive bias

[Diagram of neural network and dataset]

Osvaldo Simeone

Meta-Learning to Communicate
Conventional Machine Learning

model class + training procedure

inductive bias

$D_k^{tr}$

training data

Meta-Learning to Communicate
Conventional Machine Learning

- model class
- + training procedure
- inductive bias
- model class
- model parameter
- \( \mathcal{D}_{k}^{tr} \)
- training data
Conventional Machine Learning

model class + training procedure

model class

inductive bias

model parameter

$D_k^{tr}$ training data

$D_k^{te}$ test data

Meta-Learning to Communicate
Conventional Machine Learning

- Model class
- + Training procedure

Model class

Inductive bias

Model parameter

Training procedure

$D^\text{tr}_k$

Training data

$D^\text{te}_k$

Test data

Requires enough data and training time for each new task…
Joint Learning

- Train a single model for a class of tasks
Joint Learning

• Train a single model for a class of tasks

inductive bias
Joint Learning

- Train a **single model** for a class of tasks
Joint Learning

- Train a single model for a class of tasks
Joint Learning

- Train a **single model** for a class of tasks
  - May require few data points for each task
  - Training time amortized over multiple tasks
Joint Learning

- Train a **single model** for a class of tasks
  - May require few data points for each task
  - Training time amortized over multiple tasks

- There may not be a **single model** that works well on all tasks
  - Unable to adapt to new tasks
Meta-Learning

\[ D_{tr}^k \quad D_{te}^k \]

training data

test data

meta-training data

Osvaldo Simeone
Meta-Learning

Meta-Learning to Communicate

D^tr_k
training data

D^te_k
test data

meta-training data

inductive bias
(\theta)

meta-training
Meta-Learning

- Meta-learning finds an inductive bias that enables the training of accurate specialized models from few samples and/or with little complexity on each of the meta-training tasks...
Meta-Learning

• … so that sample or iteration complexity for training new tasks are reduced
Meta-Learning

• … so that sample or iteration complexity for training new tasks are reduced

• Meta-training learns how to adapt, or how to learn
Meta-Learning

• Meta-learned inductive bias:
  ▪ representation (i.e., feature extraction) [Vinyals et al ’16]
  ▪ use of memory [Santoro et al ‘16]
  ▪ learning rate [Maclaurin et al ’15]
  ▪ non-linear gradient-based updates [Bengio et al ’90] [Wichrowska et al ’17]
  ▪ initialization [Finn et al ’17]
# Relationship with Transfer and Multi-Task Learning

<table>
<thead>
<tr>
<th>Learning Type</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer Learning</td>
<td>Task 1</td>
<td>Task 2</td>
</tr>
<tr>
<td>Multi-task Learning</td>
<td>Task 1</td>
<td>Task 1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Task N</td>
<td>Task N</td>
</tr>
<tr>
<td>Meta-learning</td>
<td>Task 1</td>
<td>Task N+1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task N</td>
<td></td>
</tr>
</tbody>
</table>
Overview

- Motivation
- Meta-learning in a nutshell
- Algorithms and applications

System Model

Key offline meta-learning parameters:

- $K$: number of meta-training devices
- $N$: number of pilots for meta-training data
- $\mathcal{D}$: meta-training dataset
- $\varphi$: neural network parameters
- $P$: number of pilots for meta-test data
- $\mathcal{D}_T$: meta-test dataset

End-to-end channel for a device $k$:

$$ y_k = h_k x_k + z_k $$

$$ x_k \sim p_k(\cdot | s_k) $$
System Model

\[ \text{End-to-end channel for a device } k \]
\[ y_k = h_k x_k + z_k \]
\[ x_k \sim p_k(\cdot | s_k) \]

Parameterized demodulator (neural network)
\[ p(s|y, \varphi) \]

Key offline meta-learning parameters:
- \( K \): \# of meta-training devices
- \( N \): \# of pilots for meta-training data
- \( D \): meta-training dataset
- \( \varphi \): neural network parameters
- \( P \): \# of pilots for meta-test data
- \( D_T \): meta-test dataset
System Model

Key offline meta-learning parameters

- \( K \): # of meta-training devices
- \( N \): # of pilots for meta-training data
- \( D \): meta-training dataset
- \( \varphi \): neural network parameters
- \( P \): # of pilots for meta-test data
- \( D_T \): meta-test dataset

Parameterized demodulator (neural network)

\[ p(s \mid y, \varphi) \]

Symbol: \( S_k \)

Transmitter’s distortion: \( p_k \)

Channel: \( h_k \)

Noise: \( z_k \)

IoT device \( k \)

End-to-end channel for a device \( k \)

\[ y_k = h_k x_k + z_k \]

\[ x_k \sim p_k \]

Meta-training devices

- Training set \( D_k^T \)
- Test set \( D_k^e \)

Meta-test device (i.e., new)

- Training set \( D_T \)
- Data symbols \( D_{\text{data}} \)

Meta-training dataset

\[ D = \{ D_k \}_{k=1,\ldots,K} \]

\[ D_k = \{ (s_k^{(n)}, y_k^{(n)}) : n = 1, \ldots, N \} \]

Meta-test dataset

\[ D_T = \{ (s^{(n)}, y^{(n)}) : n = 1, \ldots, P \} \]
System Model

Key offline meta-learning parameters:
- \( K \): # of meta-training devices
- \( N \): # of pilots for meta-training data
- \( D \): meta-training dataset
- \( \varphi \): neural network parameters
- \( P \): # of pilots for meta-test data
- \( D_T \): meta-test dataset

System Model Diagram:
- Base station
- Noise: \( Z_k \)
- Channel: \( h_k \)
- Symbol: \( S_k \)
- Transmitter’s distortion: \( p_k(\cdot \mid s_k) \)
- End-to-end channel for a device \( k \):
  \[ y_k = h_k x_k + z_k \]
  \( x_k \sim p_k(\cdot \mid s_k) \)
- Parameterized demodulator (neural network):
  \( p(s \mid y, \varphi) \)

Meta-training dataset:
\[ D = \{ D_k \}_{k=1,...,K} \]
\[ D_k = \{(s_k^{(n)}, y_k^{(n)}) : n = 1, \ldots, N\} \]

Meta-test dataset:
\[ D_T = \{(s^{(n)}, y^{(n)}) : n = 1, \ldots, P\} \]
Conventional Training

• Conventional training operates separately on the pilots of each device \( k \).

• Pilots can be used for the supervised learning of a demodulator as a classifier.

• The training procedure aims at minimizing the generalization cross-entropy (surrogate of the probability of error)

\[
L_k(\varphi) = \mathbb{E}_{(s,y) \sim p_k} \left[ -\log p(s \mid y, \varphi) \right]
\]

log-loss

(information-theoretic surprise)
Conventional Training

• Given a training data set $\mathcal{D}_k$, the ensemble loss is approximated by the training cross-entropy loss

$$L_{\mathcal{D}_k}(\varphi) = - \sum_{(s,y) \in \mathcal{D}_k} \log p(s|y, \varphi)$$

• Minimization is done via Stochastic GD (SGD)

$$\varphi \leftarrow \varphi - \eta \nabla_\varphi \log p(s|y, \varphi)$$
Conventional Training

- **Inductive Bias**: The model is trained on a specific dataset ($D_{tr}^k$) and then tested on a different dataset ($D_{te}^k$).
- **Model Class**: The model is parameterized by a fixed parameter set $\phi$.
- **Training Procedure**: The model is trained on the training data to achieve a specific performance on the test data.

The diagram illustrates the conventional training process with an inductive bias, where the model is trained on one dataset and tested on another, highlighting the separation between training and testing environments.
Joint Training

• In order to reduce the amount of data required for the new task, an intuitive solution would be to use joint training.

• Based on meta-training data \( \mathcal{D} \), joint training minimizes

\[
L_\mathcal{D}(\varphi) = - \sum_{(s,y) \in \mathcal{D}} \log p(s|y, \varphi)
\]

• Joint training finds a single model that should perform well on all meta-training tasks/devices.
Joint Training

inductive bias

model class

model parameter

training procedure

training data $D_{k}^{tr}$

$D_{k}^{te}$ test task
Meta-Learning

- Shared hyperparameter $\theta \rightarrow$ defines the inductive bias

- Task/device-specific parameter $\phi$
Meta-Learning

- Shared hyperparameter $\theta \rightarrow$ defines the inductive bias

  meta-learned using meta-training data $\mathcal{D}$

- Task/ device-specific parameter $\phi$

  from conventional training with inductive bias $\theta$
  using task-specific data
Meta-Learning

- Meta-learning algorithms can be derived as approximations of Expectation Maximization (EM) for hierarchical probabilistic models [Park et al ‘19].
- As for EM, they are organized around a nested loop.
for given meta-training devices, update device-dependent model parameter $\phi_k$
Meta-Learning

for given meta-training devices, update device-dependent model parameter $\phi_k$
given device-dependent updates, update shared hyperparameter $\theta$
Meta-Learning

for given meta-training devices, update device-dependent model parameter $\phi_k$

given device-dependent updates, update shared hyperparameter $\theta$

conventional (device-specific) training
Meta-Learning

meta-learning: acquisition of inductive bias

given device-dependent updates, update shared hyperparameter $\theta$

for given meta-training devices, update device-dependent model parameter $\phi_k$

conventional (device-specific) training
MAML

- Model-Agnostic Meta-Learning [Finn et al ’17]
  - Demodulator: \( p(s|y, \phi) \)
  - Device-specific parameter: \( \phi = \varphi \)
  - Shared hyperparameter: \( \theta = \text{initialization} \) of local updates

\[
\phi_k = \theta - \eta \nabla_\theta L_{D_{tr}^k} (\theta) \\
(m \text{ SGD steps, here } m=1)
\]
MAML

- Model-Agnostic Meta-Learning [Finn et al ’17]
  - Demodulator: \( p(s|y, \phi) \)
  - Device-specific parameter: \( \varphi = \phi \)
  - Shared hyperparameter: \( \theta = \text{initialization} \) of local updates

\[
\phi_k = \theta - \eta \nabla_\theta L_{D_{tr}^k}(\theta)
\]

\( (m \text{ SGD steps, here } m=1) \)
MAML

- Model-Agnostic Meta-Learning [Finn et al ’17]
  - Demodulator: \( p(s | y, \phi) \)
  - Device-specific parameter: \( \varphi = \phi \)
  - Shared hyperparameter: \( \theta = \text{initialization} \) of local updates

\[
\phi_k = \theta - \eta \nabla_{\theta} L_{\mathcal{D}_k^{tr}}(\theta)
\]

\((m \text{ SGD steps, here } m=1)\)

meta-update

\[
\min_{\theta} \sum_{k=1}^{K} L_{\mathcal{D}_k^{te}}(\phi_k)
\]
Learning to Demodulate from Few Pilots via Offline and Online Meta-Learning

S. Park, H. Jang, O. Simeone, and J. Kang

MAML

MAML

$\theta_{\text{MAML}}$

$\phi_1^*$

$m$

$\phi_k^*$

$\phi_{\text{joint}}$

$\phi_k^*$

$\phi_k^*$

Meta-Learning to Communicate
MAML

- Meta-update

\[
\min_{\theta} \sum_{k=1}^{K} L_{D_k}^{te}(\phi_k)
\]

\[
- \eta \nabla_{\theta} L_{D_k}^{tr}(\theta)
\]
- **Meta-update**

\[
\min_{\theta} \sum_{k=1}^{K} L_{D^{te}_k}(\phi_k)
\]

\[
\nabla_{\theta} \sum_{k=1}^{K} L_{D^{te}_k}(\phi_k) - \eta \nabla_{\theta} L_{D^{tr}_k}(\theta)
\]
MAML

- Meta-update

\[ \min_{\theta} \sum_{k=1}^{K} L_{D_{k}^te}(\phi_k) \]

\[ \nabla_{\theta} \sum_{k=1}^{K} L_{D_{k}^te}(\phi_k) \]

\[ - \eta \nabla_{\theta} L_{D_{k}^tr}(\theta) \]

\[ \theta \leftarrow \theta - \kappa \sum_{k=1}^{K} (I - \eta \nabla_{\theta}^2 L_{D_{k}^tr}(\theta)) \nabla_{\phi_k} L_{D_{k}^te}(\phi_k) \]
FOMAML

- First-Order Model-Agnostic Meta-Learning [Finn et al ’17]
  - Meta-update

\[
\theta \leftarrow \theta - \kappa \nabla_{\theta} L_{D^*_k} (\theta) - \eta \nabla_{\theta} L_{D^*_k} (\theta) \nabla_{\phi_k} L_{D^*_k}(\phi_k) \sum_{k=1}^{K} L_{D^*_k}(\phi_k)
\]
REPTILE

- REPTILE [Nichol et al ’18]
  - Meta-update

\[
- \eta \nabla_{\theta} L_{D_k^{tr}}(\theta)
\]

\[
\theta \leftarrow (1 - \kappa)\theta - \kappa \sum_{k=1}^{K} \phi_k
\]
REPTILE

- REPTILE [Nichol et al ’18]
  - Meta-update

\[ -\eta \nabla_{\theta} L_{D_k}^{tr}(\theta) \]

\[ \theta \leftarrow (1 - \kappa) \theta - \kappa \sum_{k=1}^{K} \phi_k \]

… equivalent to Federated Averaging [McMahan et al ’17]
- fast Context Adaptation VIA meta-learning [Zintgraf et al ’19]
  - Demodulator: $p(s|\tilde{y}, \theta)$, $\tilde{y} = [y, \phi]$, $\phi$: additional input
  - Shared parameter: $\varphi = \theta$

\[ \begin{align*}
\phi &= \phi_1, \ldots, \phi_D \\
y &= \{W, b\} \\
\theta_1 &= \{W_1, b_1\} \\
h &= \ldots
\end{align*} \]
... And Many More

- Very active field with daily updates: T-MAML [Liu et al ’19], modular meta-learning [Chen et al ’19], implicit gradients [Rajeswaran et al ’19], zeroth-order MAML [Song et al ’19],…

- Probabilistic approach [Finn et al ’18], [Ravi and Beatson ’19], [Gordon et al ’19], [Nguyen et al ’19]:
  - Instead of point estimate, approximate posterior distribution in E-step
  - Can add prior for shared parameter
Experiments

- I/Q imbalance at the transmitters and Rayleigh fading channels with 16-QAM
Experiments

![Graph showing symbol error rate vs. number of pilots for meta-test (P).](image)

- MMSE channel estimator + ML demodulator
- Optimal demodulator

- Symbol error rate
- Number of pilots for meta-test (P)
Experiments

- MMSE channel estimator + ML demodulator
- Conventional learning
- Optimal demodulator

Symbol error rate vs. number of pilots for meta-test ($P$)
Experiments

- MMSE channel estimator + ML demodulator
- Conventional training (long adaptation)
- MAML
- CAVIA
- FOMAML
- REPTILE
- Optimal (perfect knowledge)

![Graph showing symbol error rate vs. number of pilots for meta-test (P)]
Experiments

The graph illustrates the performance of different demodulation methods as a function of the number of pilots for meta-test ($P$). The methods compared are:

- **MMSE channel estimator + ML demodulator**
- **Conventional learning**
- **CAVIA**
- **REPTILE**
- **Optimal demodulator**
- **MAML**

The x-axis represents the number of pilots, and the y-axis shows the symbol error rate. The graph shows how the symbol error rate decreases as the number of pilots increases, with each method having a different slope and intercept, indicating varying performance characteristics.
Experiments

![Graph showing the comparison of different demodulation methods with varying number of meta-training devices (K)].

- **Conventional Learning**
- **MMSE Channel Estimator + ML Demodulator**
- **Optimal Demodulator**

The graph plots symbol error rate against the number of meta-training devices (K).
Experiments

![Graph showing symbol error rate vs. number of meta-training devices (K).]

- **Joint training**
- **Conventional learning**
- **MMSE channel estimator + ML demodulator**
- **Optimal demodulator**
Experiments

![Graph showing comparison between different methods for symbol error rate as a function of the number of meta-training devices (K). The methods compared include joint training, MAML, conventional learning, MMSE channel estimator + ML demodulator, optimal demodulator, and CAVIA. The graph illustrates the effectiveness of these methods in reducing the symbol error rate as the number of meta-training devices increases.]
Learning to Demodulate from Few Pilots via Offline and Online Meta-Learning

key online meta-learning parameters

- \( t \): current time slot
- \( P_t \): # of pilots for current time slot
- \( \mathcal{D}^{t-1} \): meta-training dataset
- \( t-1 \): # of meta-training devices
- \( P \): maximum # of pilots
- \( \mathcal{D}_t \): meta-test dataset

![Diagram of meta-training and test devices at slot \( t \)]

Meta-training dataset:
\[
\mathcal{D}^{t-1} = \{ \mathcal{D}_{t'} \}_{t'=1}^{t-1}
\]

Meta-test dataset:
\[
\mathcal{D}_t = \{ (s_t^{(n)}, y_t^{(n)}) : n = 1, \ldots, P_t \}
\]
Online Meta-Learning

key online meta-learning parameters

- \( t \): current time slot
- \( P_t \): # of pilots for current time slot
- \( \mathcal{D}^{t-1} \): meta-training dataset
- \( t - 1 \): # of meta-training devices
- \( P \): maximum # of pilots
- \( \mathcal{D}_t \): meta-test dataset

Parameterized demodulator (neural network)

\[ p(s_t | y_t, \varphi_t) \]

Meta-training dataset

\[ \mathcal{D}^{t-1} = \{ \mathcal{D}_{t'} \}_{t'=1}^{t-1} \]
\[ \mathcal{D}_t = \{(s_{t}^{(n)}, y_{t}^{(n)}) : n = 1, \ldots, P_t \} \]

Meta-test dataset

\[ \mathcal{D}_t = \{(s_{t}^{(n)}, y_{t}^{(n)}) : n = 1, \ldots, P_t \} \]
Online Meta-Learning

Parameterized demodulator (neural network)

\[ p(s_t | y_t, \varphi_t) \]

Online meta-training is also known as lifelong learning

Meta-training dataset

\[ \mathcal{D}^{t-1} = \{ \mathcal{D}_{t'} \}_{t'=1}^{t-1} \]
\[ \mathcal{D}_t = \{(s_t^{(n)}, y_t^{(n)}) : n = 1, \ldots, P_t\} \]

Meta-test dataset

\[ \mathcal{D}_t = \{(s_t^{(n)}, y_t^{(n)}) : n = 1, \ldots, P_t\} \]

key online meta-learning parameters

- \( t \): current time slot
- \( P_t \): # of pilots for current time slot
- \( \mathcal{D}^{t-1} \): meta-training dataset
- \( t - 1 \): # of meta-training devices
- \( P \): maximum # of pilots
- \( \mathcal{D}_t \): meta-test dataset

Online meta-training is also known as lifelong learning.
Adaptive Pilot Allocation

- Based on reliability check with different number of pilots in current time slot, determine number of pilots for the next time slot:

\[
\text{reliability check: } - \sum_{y \in \mathcal{D}_t^{\text{data}}} \max_s \log p(s | y, \phi_t^{(p)}, \theta_t)
\]
Experiments

![Graph showing the performance of different demodulation methods. The x-axis represents the average number of pilots, and the y-axis represents the average symbol error rate. The graph compares MMSE channel estimator + ML demodulator, MAML, CAVIA, and an optimal demodulator.](image)
Concluding Remarks

• Meta-learning techniques can benefit communication systems with few pilots or when fast training is necessary.

• Reduction of sample or iteration complexity by transferring knowledge from related tasks.

• Online meta-learning may yield novel adaptive resource allocation.

• How many tasks should we observe? Information theoretic analysis [Jose and Simeone ‘20]

• Other potential applications of meta-learning:
  - Channel estimation and prediction
  - Precoding in multi-antenna systems
  - …
Extra Slides

• Some theory

Statistical Learning Theory

- inductive bias

- model class

- model parameter

- training procedure

- \( D_{tr}^k \) training data

- \( D_{te}^k \) test data

~ same data distribution
For a given task $k$, the learner can compute the **training loss** $L_{D_k}(\phi_k)$. 
Statistical Learning Theory

- For a given task $k$, the learner can compute the training loss $L_{D_k}(\phi_k)$.

- Test performance is measured by the (unknown) test loss $L_k(\phi_k)$. 

![Diagram showing the relationship between training data, test data, model parameters, and loss functions.](image)
For a given task $k$, the learner can compute the training loss $L_{D_k}(\phi_k)$.

Test performance is measured by the (unknown) test loss $L_k(\phi_k)$.

Test performance can be guaranteed if the generalization gap $L_k(\phi_k) - L_{D_k}(\phi_k)$ is small.
Statistical Learning Theory

- Under suitable assumptions [Xu-Raginsky '17],

\[
E_{\text{train algo}}[L_k(\phi_k) - L_{D_k}(\phi_k)] \leq \sqrt{\frac{2\sigma^2}{\text{#train samples}} I(\phi_k; D_k)}
\]
Statistical Learning Theory

- Under suitable assumptions [Xu-Raginsky ’17],

\[
E_{\text{train algo}} [L_k(\phi_k) - L_{\mathcal{D}_k}(\phi_k)] \leq \sqrt{\frac{2\sigma^2}{\#\text{train samples}}} I(\phi_k; \mathcal{D}_k)
\]

“sensitivity” of training procedure to training data
Statistical Meta-Learning Theory

~ same environment (task) distribution
The meta-learner can compute the meta-training loss $L_D(\theta)$. 
The meta-learner can compute the meta-training loss $L_D(\theta)$.

Test performance is measured by the (unknown) meta-test loss $L(\theta)$. 
The meta-learner can compute the meta-training loss $L_D(\theta)$.

Test performance is measured by the (unknown) meta-test loss $L(\theta)$.

Test performance can be guaranteed if the meta-generalization gap $L(\theta) - L_D(\theta)$ is small.
Statistical Meta-Learning Theory

- Under suitable assumptions [Jose and Simeone ‘20],

\[ E_{\text{meta-train algo}} [L(\theta) - L_{\mathcal{D}}(\theta)] \leq \sqrt{\frac{2\sigma^2}{\#\text{meta-train tasks}}} \quad \text{I(\theta; \mathcal{D})} \]

“sensitivity” of meta-training procedure to meta-training data
Under suitable assumptions [Jose and Simeone ‘20],

\[ E_{\text{meta-train algo}} [L(\theta) - L_D(\theta)] \leq \sqrt{\frac{2\sigma^2}{\#\text{meta-train tasks}}} I(\theta; D) \]

“sensitivity” of meta-training procedure to meta-training data

[Yin et al ‘19]