ECEN 743: Reinforcement Learning

Meta Learning

Dileep Kalathil
Assistant Professor
Department of Electrical and Computer Engineering
Texas A&M University

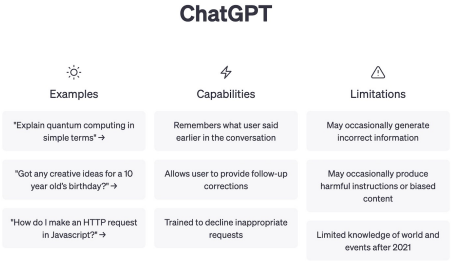
References

- Acknowledgment: Most of the materials for this lecture are taken from the lecture slides of Stanford CS 330: Deep Multi-Task and Meta Learning course by Prof. Chelsea Finn [Link], with permission.

- All the figures in the slides, unless otherwise stated, are also from the Stanford's CS 330 course.

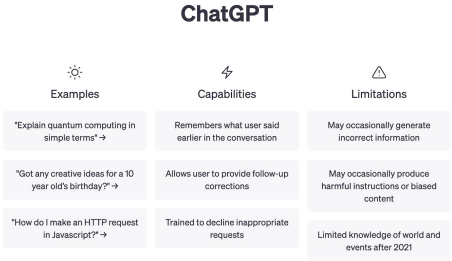
Deep Learning with Limited Data

- Standard deep learning algorithms require very large and diverse set of data samples, and large (deep) neural network models



Deep Learning with Limited Data

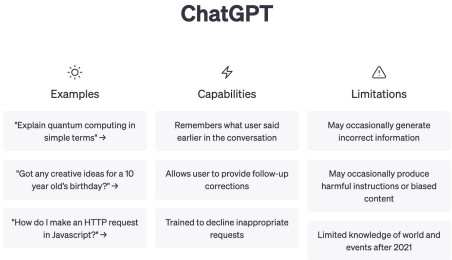
- Standard deep learning algorithms require very large and diverse set of data samples, and large (deep) neural network models



- What if we don't have a large dataset?

Deep Learning with Limited Data

- Standard deep learning algorithms require very large and diverse set of data samples, and large (deep) neural network models



- What if we don't have a large dataset?
  - medical imaging, robotics, personalized education, recommendations

Learning with Limited Data

- Humans learn new concepts and skills efficiently with very limited data

Learning with Limited Data

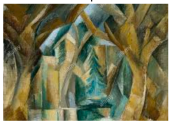
- Humans learn new concepts and skills efficiently with very limited data

Learning with Limited Data

- Humans learn new concepts and skills efficiently with very limited data

training data

Braque



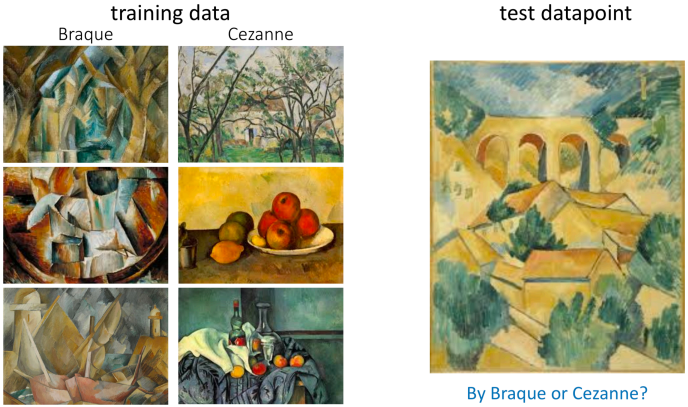
Cezanne

test datapoint

By Braque or Cezanne?

Learning with Limited Data

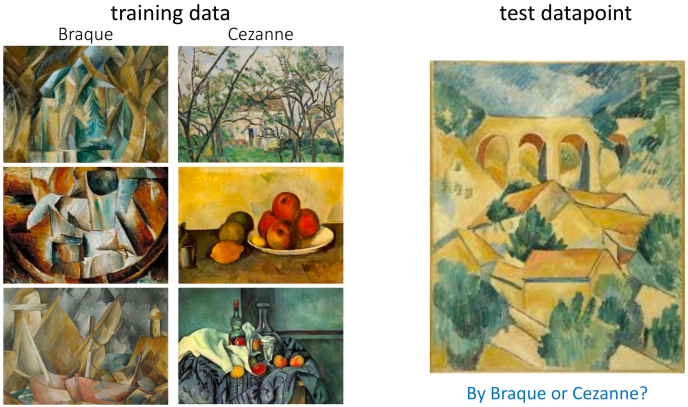
- Humans learn new concepts and skills efficiently with very limited data



- Arguably, human are leveraging on the prior experiences

Learning with Limited Data

- Humans learn new concepts and skills efficiently with very limited data



- Arguably, human are leveraging on the prior experiences
- Can we design ML algorithms that can learn new tasks with a few training examples by leveraging on prior experience/data from other tasks?

Tasks

- Consider a supervised learning problem: the goal is to predict the y given the input x, i.e, $y = f_\theta(x)$

Tasks

- Consider a supervised learning problem: the goal is to predict the y given the input x, i.e, $y = f_\theta(x)$
- Assume the data generating distributions $p_i(x)$ and $p_i(y|x)$; also denoted as $(x, y) \sim p_i$

Tasks

- Consider a supervised learning problem: the goal is to predict the y given the input x, i.e, $y = f_\theta(x)$
- Assume the data generating distributions $p_i(x)$ and $p_i(y|x)$; also denoted as $(x, y) \sim p_i$
- Data $\mathcal{D}_i = (\mathcal{D}_i^{\text{train}}, \mathcal{D}_i^{\text{test}})$

Tasks

- Consider a supervised learning problem: the goal is to predict the y given the input x, i.e, $y = f_\theta(x)$
- Assume the data generating distributions $p_i(x)$ and $p_i(y|x)$; also denoted as $(x, y) \sim p_i$
- Data $\mathcal{D}_i = (\mathcal{D}_i^{\text{train}}, \mathcal{D}_i^{\text{test}})$
- True loss function $L_i(\theta, p_i) = \mathbb{E}_{(x,y)\sim p_i}[\ell_i(f_\theta(x), y)]$; we typically assume that $\ell_i = \ell, \ \forall i$

Tasks

- Consider a supervised learning problem: the goal is to predict the y given the input x, i.e, $y = f_\theta(x)$
- Assume the data generating distributions $p_i(x)$ and $p_i(y|x)$; also denoted as $(x, y) \sim p_i$
- Data $\mathcal{D}_i = (\mathcal{D}_i^{\text{train}}, \mathcal{D}_i^{\text{test}})$
- True loss function $L_i(\theta, p_i) = \mathbb{E}_{(x,y)\sim p_i}[\ell_i(f_\theta(x), y)]$; we typically assume that $\ell_i = \ell, \ \forall i$
- Empirical loss function: $L_i(\theta, \mathcal{D}_i) = \mathbb{E}_{(x,y)\sim \mathcal{D}_i}[\ell(f_\theta(x), y)]$

Tasks

- Consider a supervised learning problem: the goal is to predict the y given the input x, i.e, $y = f_\theta(x)$
- Assume the data generating distributions $p_i(x)$ and $p_i(y|x)$; also denoted as $(x, y) \sim p_i$
- Data $\mathcal{D}_i = (\mathcal{D}_i^{\text{train}}, \mathcal{D}_i^{\text{test}})$
- True loss function $L_i(\theta, p_i) = \mathbb{E}_{(x,y)\sim p_i}[\ell_i(f_\theta(x), y)]$; we typically assume that $\ell_i = \ell, \ \forall i$
- Empirical loss function: $L_i(\theta, \mathcal{D}_i) = \mathbb{E}_{(x,y)\sim \mathcal{D}_i}[\ell(f_\theta(x), y)]$
- Objective: $\min_\theta \ L_i(\theta, \mathcal{D}_i)$

Tasks

- Consider a supervised learning problem: the goal is to predict the y given the input x, i.e, $y = f_\theta(x)$
- Assume the data generating distributions $p_i(x)$ and $p_i(y|x)$; also denoted as $(x, y) \sim p_i$
- Data $\mathcal{D}_i = (\mathcal{D}_i^{\text{train}}, \mathcal{D}_i^{\text{test}})$
- True loss function $L_i(\theta, p_i) = \mathbb{E}_{(x,y)\sim p_i}[\ell_i(f_\theta(x), y)]$; we typically assume that $\ell_i = \ell, \ \forall i$
- Empirical loss function: $L_i(\theta, \mathcal{D}_i) = \mathbb{E}_{(x,y)\sim \mathcal{D}_i}[\ell(f_\theta(x), y)]$
- Objective: $\min_\theta \ L_i(\theta, \mathcal{D}_i)$
- Task: $\mathcal{T}_i = \{p_i, \mathcal{D}_i, L_i\}$

Task Similarity

- Different tasks can vary based on different objects, people, animals, lighting condition, words, ...

Task Similarity

- Different tasks can vary based on different objects, people, animals, lighting condition, words, ...
- For doing multi-task/ transfer/meta learning, tasks should have some *similarity*

Task Similarity

- Different tasks can vary based on different objects, people, animals, lighting condition, words, ...
- For doing multi-task/ transfer/meta learning, tasks should have some *similarity*
- What is task similarity?

Task Similarity

- Different tasks can vary based on different objects, people, animals, lighting condition, words, ...
- For doing multi-task/ transfer/meta learning, tasks should have some *similarity*
- What is task similarity?
  - This can be characterized in a number of ways

Task Similarity

- Different tasks can vary based on different objects, people, animals, lighting condition, words, ...
- For doing multi-task/ transfer/meta learning, tasks should have some *similarity*
- What is task similarity?
  - This can be characterized in a number of ways
- Examples: per-language handwriting recognition, personalized recommendation, face attribute recognition, scene understanding, ...

Multi-Task Learning

- Basic vanilla multi-task learning objective: $\min_\theta \sum_{i=1}^{M} L_i(\theta, \mathcal{D}_i)$
  - Goal is to learn a single parameter for all tasks

Image from Ruder, 17. [Link]

Multi-Task Learning

- Basic vanilla multi-task learning objective: $\min_\theta \sum_{i=1}^{M} L_i(\theta, \mathcal{D}_i)$
  - Goal is to learn a single parameter for all tasks

- A better objective

$$\min_{\theta_{\text{sh}}, (\theta_i)_{i=1}^{M}} \sum_{i=1}^{M} L_i(\theta_{\text{sh}}, \theta_i, \mathcal{D}_i)$$

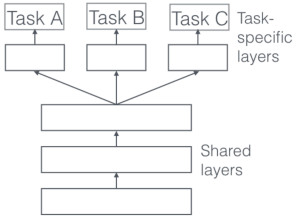


Figure 1: Hard parameter sharing for multi-task learning in deep neural networks

Image from Ruder, 17. [Link]

Transfer Learning

- Multi-task learning: solve multiple tasks at once, using the data from all tasks

Transfer Learning

- Multi-task learning: solve multiple tasks at once, using the data from all tasks
- Transfer Learning: Solve a target task after solving the source task

Transfer Learning

- Multi-task learning: solve multiple tasks at once, using the data from all tasks
- Transfer Learning: Solve a target task after solving the source task
  - We typically don't have the data from the source task while solving the target task

Transfer Learning

- Multi-task learning: solve multiple tasks at once, using the data from all tasks
- Transfer Learning: Solve a target task after solving the source task
  - We typically don't have the data from the source task while solving the target task
- Transfer learning is typically achieved by fine-tuning of a pre-trained model parameter

$$\phi_{\text{new-task},k+1} = \phi_{\text{new-task},k} - \alpha \nabla_{\phi} L(\phi, \mathcal{D}_{\text{new-task}})|_{\phi_{\text{new-task},k}}, \quad \text{with } \phi_{\text{new-task},0} = \theta_{\text{pre-trained}}$$

Transfer Learning

- Multi-task learning: solve multiple tasks at once, using the data from all tasks
- Transfer Learning: Solve a target task after solving the source task
  - We typically don't have the data from the source task while solving the target task
- Transfer learning is typically achieved by fine-tuning of a pre-trained model parameter

$$\phi_{\text{new-task},k+1} = \phi_{\text{new-task},k} - \alpha \nabla_{\phi} L(\phi, \mathcal{D}_{\text{new-task}})|_{\phi_{\text{new-task},k}}, \quad \text{with } \phi_{\text{new-task},0} = \theta_{\text{pre-trained}}$$

- Where do you get the pre-trained parameters?

Transfer Learning

- Multi-task learning: solve multiple tasks at once, using the data from all tasks
- Transfer Learning: Solve a target task after solving the source task
  - We typically don't have the data from the source task while solving the target task
- Transfer learning is typically achieved by fine-tuning of a pre-trained model parameter

$$\phi_{\text{new-task},k+1} = \phi_{\text{new-task},k} - \alpha \nabla_{\phi} L(\phi, \mathcal{D}_{\text{new-task}})|_{\phi_{\text{new-task},k}}, \quad \text{with } \phi_{\text{new-task},0} = \theta_{\text{pre-trained}}$$

- Where do you get the pre-trained parameters?
  - ImageNet classification models (VGG19, Inceptionv3, ResNet); LLMs (BERT, LLaMA), ...

Transfer Learning by Fine Tuning

$$\phi_{\text{new-task},k+1} = \phi_{\text{new-task},k} - \alpha \nabla_\phi L(\phi, \mathcal{D}_{\text{new-task}})|_{\phi_{\text{new-task},k}}, \quad \text{with } \phi_{\text{new-task},0} = \theta_{\text{pre-trained}}$$

Common design choices:

- Fine-tune with a smaller learning rate
- Smaller learning rate for earlier layers
- Freeze earlier layers, gradually unfreeze
- Reinitialize last layer
- Search over hyperparameters via cross-val
- Architecture choices matter (e.g. ResNets)

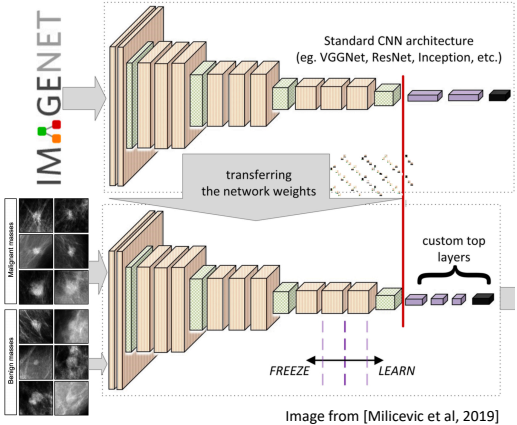
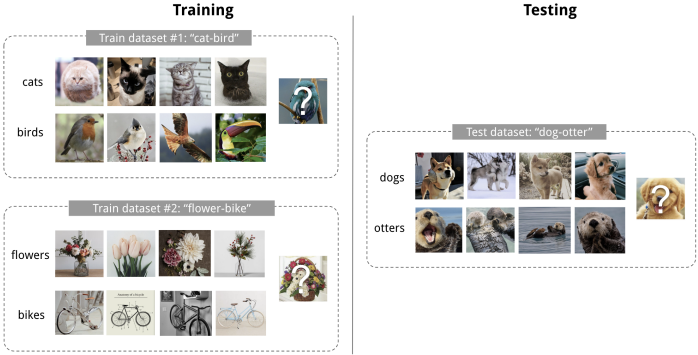


Image from [Milicevic et al, 2019]

Meta Learning

- Meta Learning: Given data from tasks $\mathcal{T}_1, \ldots, \mathcal{T}_n$, learn to solve a new task $\mathcal{T}_{\text{test}}$ with only few data samples

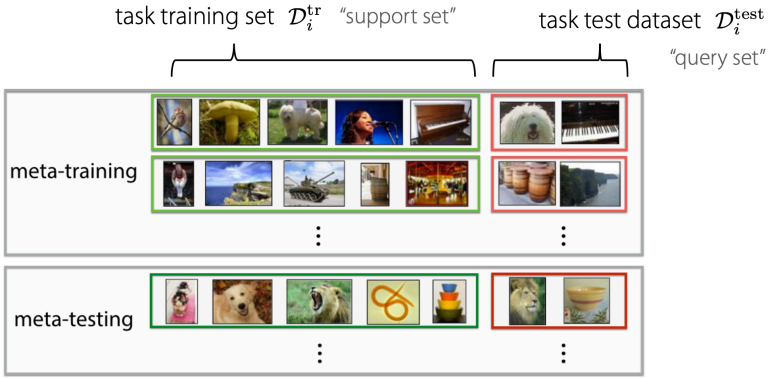


- k-shot learning: learning with k examples per class

Image from [Weng, 2018]

Meta Learning

- Meta Learning: Given data from tasks $\mathcal{T}_1, \ldots, \mathcal{T}_n$, learn to solve a new task $\mathcal{T}_{\text{test}}$ with only few data samples



- k-shot learning: learning with k examples per class

Meta Learning

- Multi-task learning: solving multiple tasks at the same time

Meta Learning

- Multi-task learning: solving multiple tasks at the same time
- Transfer learning: use the solution of the source task as an initialization to the target task

Meta Learning

- Multi-task learning: solving multiple tasks at the same time
- Transfer learning: use the solution of the source task as an initialization to the target task
- Can we optimize the training of the source tasks for transferability?

Meta Learning

- Multi-task learning: solving multiple tasks at the same time
- Transfer learning: use the solution of the source task as an initialization to the target task
- Can we optimize the training of the source tasks for transferability?
- Meta Learning: Given data from tasks $\mathcal{T}_1, \ldots, \mathcal{T}_n$, learn a model parameter θ that can be used to solve a new task $\mathcal{T}_{\text{test}}$ with only few data samples

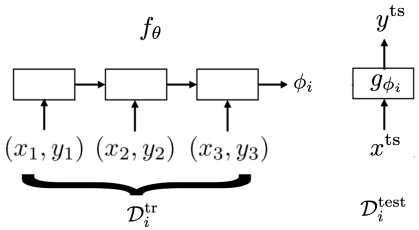
Meta Learning

- Multi-task learning: solving multiple tasks at the same time
- Transfer learning: use the solution of the source task as an initialization to the target task
- Can we optimize the training of the source tasks for transferability?
- Meta Learning: Given data from tasks $\mathcal{T}_1, \ldots, \mathcal{T}_n$, learn a model parameter θ that can be used to solve a new task $\mathcal{T}_{\text{test}}$ with only few data samples
- Meta-learning can be through as a transfer Learning with many source tasks where the real objective is to solve a new test task using only a small number of samples

Black-Box Adaptation Meta Learning

Key idea:

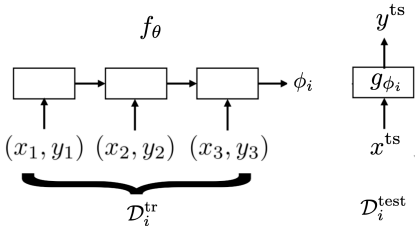
- Train a neural network θ to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$
- Use ϕ_i as the parameter for test data points $y^{\text{ts}} = g_{\phi_i}(x^{\text{ts}})$



Black-Box Adaptation Meta Learning

Key idea:

- Train a neural network θ to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$
- Use ϕ_i as the parameter for test data points $y^{\text{ts}} = g_{\phi_i}(x^{\text{ts}})$



- Training objective:

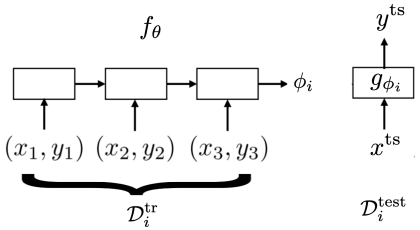
$$\min_\theta \sum_{\mathcal{T}_i} \underbrace{\sum_{(x,y)\sim\mathcal{D}_i^{\text{test}}} -\log g_{\phi_i}(y \mid x)}_{\mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})}$$

$$\min_\theta \sum_{\mathcal{T}_i} \mathcal{L}(f_\theta(\mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

Black-Box Adaptation Meta Learning

Key idea:

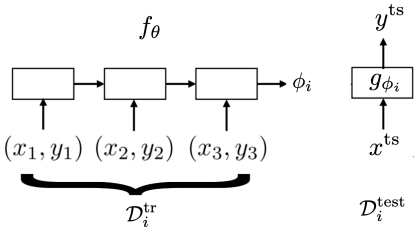
- Train a neural network θ to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$
- Use ϕ_i as the parameter for test data points $y^{\text{ts}} = g_{\phi_i}(x^{\text{ts}})$



Black-Box Adaptation Meta Learning

Key idea:

- Train a neural network θ to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$
- Use ϕ_i as the parameter for test data points $y^{\text{ts}} = g_{\phi_i}(x^{\text{ts}})$



$$\min_\theta \sum_{\mathcal{T}_i} \underbrace{\sum_{(x,y)\sim\mathcal{D}_i^{\text{test}}} -\log g_{\phi_i}(y \mid x)}_{\mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})}$$

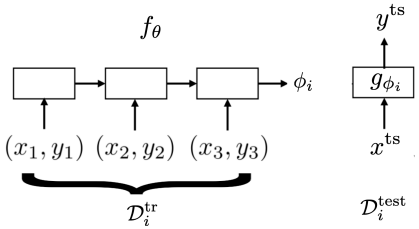
$$\min_\theta \sum_{\mathcal{T}_i} \mathcal{L}(f_\theta(\mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$
3. Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\text{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$

Black-Box Adaptation Meta Learning

Key idea:

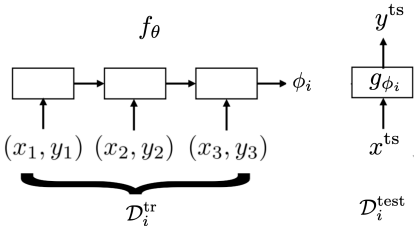
- Train a neural network θ to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$
- Use ϕ_i as the parameter for test data points $y^{\text{ts}} = g_{\phi_i}(x^{\text{ts}})$



Black-Box Adaptation Meta Learning

Key idea:

- Train a neural network θ to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$
- Use ϕ_i as the parameter for test data points $y^{\text{ts}} = g_{\phi_i}(x^{\text{ts}})$

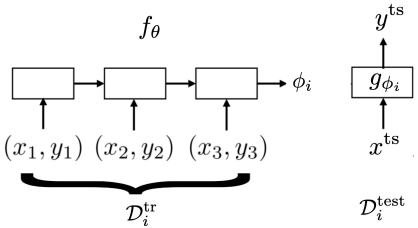


- Example: Meta-Learning with Memory-Augmented Neural Networks [Santoro et al., 2016]

Black-Box Adaptation Meta Learning

Key idea:

- Train a neural network θ to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$
- Use ϕ_i as the parameter for test data points $y^{\text{ts}} = g_{\phi_i}(x^{\text{ts}})$

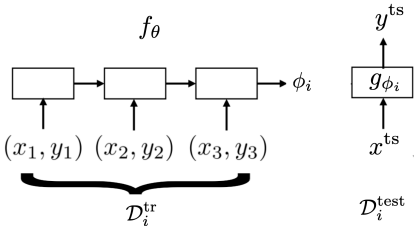


- Example: Meta-Learning with Memory-Augmented Neural Networks [Santoro et al., 2016]
- Outputting all neural net parameters may not be scalable

Black-Box Adaptation Meta Learning

Key idea:

- Train a neural network θ to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$
- Use ϕ_i as the parameter for test data points $y^{\text{ts}} = g_{\phi_i}(x^{\text{ts}})$



- Example: Meta-Learning with Memory-Augmented Neural Networks [Santoro et al., 2016]
- Outputting all neural net parameters may not be scalable
- The optimization problem is typically challenging and often data-inefficient

Optimization based Meta Learning

- Model Agnostic Meta Learning (MAML)[3]

$$\min_{\theta} \sum_{i} L(\phi_i(\theta), \mathcal{D}_i^{\text{ts}}), \quad \text{where,} \quad \phi_i(\theta) = \theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}})$$

[3] Finn et al, "Model-agnostic meta-learning for fast adaptation of deep networks", ICML, 2017

Optimization based Meta Learning

- Model Agnostic Meta Learning (MAML)[3]

$$\min_{\theta} \sum_i L(\phi_i(\theta), \mathcal{D}_i^{\text{ts}}), \quad \text{where,} \quad \phi_i(\theta) = \theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}})$$

- Equivalently

$$\min_{\theta} \sum_i L(\theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

[3] Finn et al, "Model-agnostic meta-learning for fast adaptation of deep networks", ICML, 2017

Optimization based Meta Learning

- Model Agnostic Meta Learning (MAML)[3]

$$\min_{\theta} \sum_i L(\phi_i(\theta), \mathcal{D}_i^{\text{ts}}), \quad \text{where,} \quad \phi_i(\theta) = \theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}})$$

- Equivalently

$$\min_{\theta} \sum_i L(\theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

- Optimize for a representation θ that can quickly adapt to new tasks

[3] Finn et al, "Model-agnostic meta-learning for fast adaptation of deep networks", ICML, 2017

Optimization based Meta Learning

- Model Agnostic Meta Learning (MAML)[3]

$$\min_{\theta} \sum_i L(\phi_i(\theta), \mathcal{D}_i^{\text{ts}}), \quad \text{where,} \quad \phi_i(\theta) = \theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}})$$

- Equivalently

$$\min_{\theta} \sum_i L(\theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

- Optimize for a representation θ that can quickly adapt to new tasks

[3] Finn et al, "Model-agnostic meta-learning for fast adaptation of deep networks", ICML, 2017

Optimization based Meta Learning

- Model Agnostic Meta Learning (MAML)[3]

$$\min_{\theta} \sum_i L(\phi_i(\theta), \mathcal{D}_i^{\text{ts}}), \quad \text{where,} \quad \phi_i(\theta) = \theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}})$$

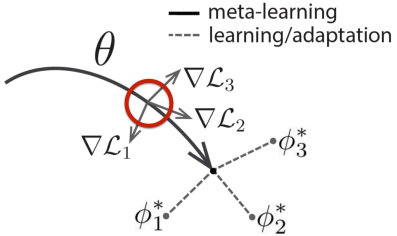
- Equivalently

$$\min_{\theta} \sum_i L(\theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

- Optimize for a representation θ that can quickly adapt to new tasks

θ parameter vector being meta-learned

ϕ_i^* optimal parameter vector for task i

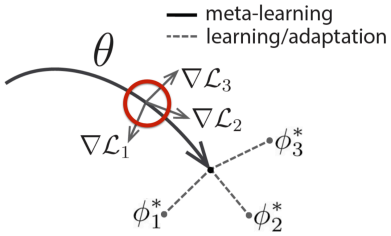


[3]Finn et al, "Model-agnostic meta-learning for fast adaptation of deep networks", ICML, 2017

Model Agnostic Meta Learning (MAML)

Algorithm Model-Agnostic Meta-Learning

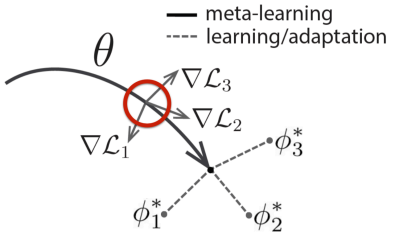
1: **for** $i = 1, 2, \ldots$ **do**
2: Sample a task $\mathcal{T}_i \sim p(\mathcal{T})$
3: Sample disjoint datasets $\mathcal{D}_i^{\text{train}}$ and $\mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$
4: Update $\phi_i \leftarrow \theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{train}})$
5: Update $\theta \leftarrow \theta - \beta \nabla_\theta L(\phi_i, \mathcal{D}_i^{\text{test}})$
6: **end for**



Model Agnostic Meta Learning (MAML)

Algorithm Model-Agnostic Meta-Learning

1: **for** $i = 1, 2, \ldots$ **do**
2: Sample a task $\mathcal{T}_i \sim p(\mathcal{T})$
3: Sample disjoint datasets $\mathcal{D}_i^{\text{train}}$ and $\mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$
4: Update $\phi_i \leftarrow \theta - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{train}})$
5: Update $\theta \leftarrow \theta - \beta \nabla_\theta L(\phi_i, \mathcal{D}_i^{\text{test}})$
6: **end for**



- Test time adaptation of MAML:
  Given the new task $\mathcal{T}_{\text{new}}$, the data $\mathcal{D}_{\text{new}}^{\text{tr}}, \mathcal{D}_{\text{new}}^{\text{ts}}$, and the trained meta-parameter θ_{meta}, perform one (or a few) gradient update to get the task specific parameter

$$\phi_{\text{new}} = \theta_{\text{meta}} - \alpha \nabla_\theta L(\theta, \mathcal{D}_i^{\text{tr}})|_{\theta=\theta_{\text{meta}}}$$

MAML Example

- Regression: Each task involves regressing from the input to the output of a sinsoidal signal

MAML Example

- Regression: Each task involves regressing from the input to the output of a sinsoidal signal
  - Amplitude and phase of the sinusoid are varied between tasks

MAML Example

- Regression: Each task involves regressing from the input to the output of a sinsoidal signal
  - Amplitude and phase of the sinusoid are varied between tasks
  - Amplitude varies within $[0.1, 5.0]$ and the phase varies within $[0, \pi]$

MAML Example

- Regression: Each task involves regressing from the input to the output of a sinsoidal signal
  - Amplitude and phase of the sinusoid are varied between tasks
  - Amplitude varies within $[0.1, 5.0]$ and the phase varies within $[0, \pi]$
  - Datapoints x are sampled uniformly from $[-5.0, 5.0]$

MAML Example

- Regression: Each task involves regressing from the input to the output of a sinsoidal signal
  - Amplitude and phase of the sinusoid are varied between tasks
  - Amplitude varies within $[0.1, 5.0]$ and the phase varies within $[0, \pi]$
  - Datapoints x are sampled uniformly from $[-5.0, 5.0]$
- K-shot regression using MAML

MAML Example

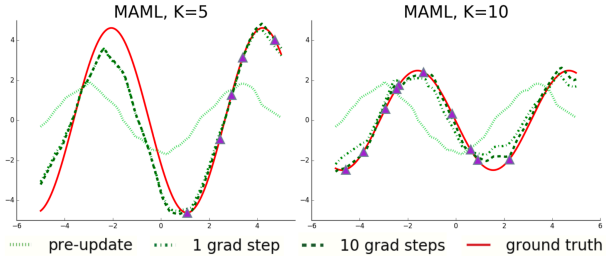
- Regression: Each task involves regressing from the input to the output of a sinsoidal signal
  - Amplitude and phase of the sinusoid are varied between tasks
  - Amplitude varies within $[0.1, 5.0]$ and the phase varies within $[0, \pi]$
  - Datapoints x are sampled uniformly from $[-5.0, 5.0]$
- K-shot regression using MAML
  - Only K samples of the new task is available

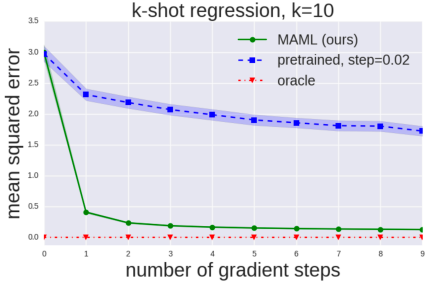
MAML Example

- Regression: Each task involves regressing from the input to the output of a sinsoidal signal
  - Amplitude and phase of the sinusoid are varied between tasks
  - Amplitude varies within $[0.1, 5.0]$ and the phase varies within $[0, \pi]$
  - Datapoints x are sampled uniformly from $[-5.0, 5.0]$
- K-shot regression using MAML
  - Only K samples of the new task is available

MAML Example

- Regression: Each task involves regressing from the input to the output of a sinsoidal signal
  - Amplitude and phase of the sinusoid are varied between tasks
  - Amplitude varies within $[0.1, 5.0]$ and the phase varies within $[0, \pi]$
  - Datapoints x are sampled uniformly from $[-5.0, 5.0]$
- K-shot regression using MAML
  - Only K samples of the new task is available





MAML Example

- Regression: Each task involves regressing from the input to the output of a sinsoidal signal
  - Amplitude and phase of the sinusoid are varied between tasks
  - Amplitude varies within $[0.1, 5.0]$ and the phase varies within $[0, \pi]$
  - Datapoints x are sampled uniformly from $[-5.0, 5.0]$
- K-shot regression using MAML
  - Only K samples of the new task is available

MAML Example

- Regression: Each task involves regressing from the input to the output of a sinsoidal signal
  - Amplitude and phase of the sinusoid are varied between tasks
  - Amplitude varies within $[0.1, 5.0]$ and the phase varies within $[0, \pi]$
  - Datapoints x are sampled uniformly from $[-5.0, 5.0]$
- K-shot regression using MAML
  - Only K samples of the new task is available

- See [Finn et al., 2017] for more examples of MAML in supervised learning (image classification) and reinforcement learning

Improved MAML Style Algorithms

- Challenge: Bi-level optimization problem

Improved MAML Style Algorithms

- Challenge: Bi-level optimization problem
- Meta Learning can be viewed as a bi-level optimization problem

Improved MAML Style Algorithms

- Challenge: Bi-level optimization problem
- Meta Learning can be viewed as a bi-level optimization problem
  - The inner optimization represents adaptation to a given task

Improved MAML Style Algorithms

- Challenge: Bi-level optimization problem
- Meta Learning can be viewed as a bi-level optimization problem
  - The inner optimization represents adaptation to a given task
  - The outer optimization represents adaptation to the meta objective

Improved MAML Style Algorithms

- Challenge: Bi-level optimization problem
- Meta Learning can be viewed as a bi-level optimization problem
  - The inner optimization represents adaptation to a given task
  - The outer optimization represents adaptation to the meta objective
- Bi-level optimization algorithms can exhibit oscillations and non-convergence when tried to solve through gradient descent approaches

Improved MAML Style Algorithms

- Challenge: Bi-level optimization problem
- Meta Learning can be viewed as a bi-level optimization problem
  - The inner optimization represents adaptation to a given task
  - The outer optimization represents adaptation to the meta objective
- Bi-level optimization algorithms can exhibit oscillations and non-convergence when tried to solve through gradient descent approaches
- There are many heuristic solutions now available that can address this issue: Meta-SGD [Li et al., 2017], CAVIA [Zintgraf et al., 2019], MAML++ [Antoniou et al., 2018]

Improved MAML Style Algorithms

- Challenge: Higher order derivatives

Improved MAML Style Algorithms

- Challenge: Higher order derivatives
  - MAML requires taking the gradient of many inner gradient steps

Improved MAML Style Algorithms

- Challenge: Higher order derivatives
  - MAML requires taking the gradient of many inner gradient steps
  - This is computationally expensive. It also requires a large memory.

Improved MAML Style Algorithms

- Challenge: Higher order derivatives
  - MAML requires taking the gradient of many inner gradient steps
  - This is computationally expensive. It also requires a large memory.
- Let $\phi_i = \text{Alg}(\theta, \mathcal{D}_i)$, which indicates multiple gradient steps

Improved MAML Style Algorithms

- Challenge: Higher order derivatives
  - MAML requires taking the gradient of many inner gradient steps
  - This is computationally expensive. It also requires a large memory.
- Let $\phi_i = \text{Alg}(\theta, \mathcal{D}_i)$, which indicates multiple gradient steps
- MAML [Finn et al., 2017]

$$\theta \leftarrow \theta - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L_i(\text{Alg}(\theta, \mathcal{D}_i), \mathcal{D}_i)$$

Improved MAML Style Algorithms

- Challenge: Higher order derivatives
  - MAML requires taking the gradient of many inner gradient steps
  - This is computationally expensive. It also requires a large memory.
- Let $\phi_i = \text{Alg}(\theta, \mathcal{D}_i)$, which indicates multiple gradient steps
- MAML [Finn et al., 2017]

$$\theta \leftarrow \theta - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L_i(\text{Alg}(\theta, \mathcal{D}_i), \mathcal{D}_i)$$

- First Order MAML [Finn et al., 2017]

$$\theta \leftarrow \theta - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\phi L_i(\phi, \mathcal{D}_i)|_{\phi=\text{Alg}(\theta, \mathcal{D}_i)}$$

Improved MAML Style Algorithms

- Challenge: Higher order derivatives
  - MAML requires taking the gradient of many inner gradient steps
  - This is computationally expensive. It also requires a large memory.
- Let $\phi_i = \text{Alg}(\theta, \mathcal{D}_i)$, which indicates multiple gradient steps
- MAML [Finn et al., 2017]

$$\theta \leftarrow \theta - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L_i(\text{Alg}(\theta, \mathcal{D}_i), \mathcal{D}_i)$$

- First Order MAML [Finn et al., 2017]

$$\theta \leftarrow \theta - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\phi L_i(\phi, \mathcal{D}_i)|_{\phi=\text{Alg}(\theta, \mathcal{D}_i)}$$

  - Ignores the effect of meta-parameters θ on task parameters ϕ_i in the meta- gradient computation

Improved Meta Learning Algorithms

- Challenge: Higher order derivatives
  - MAML requires taking the gradient of many inner gradient steps
  - This is computationally expensive. It also requires a large memory.
- Let $\phi_i = \text{Alg}(\theta, \mathcal{D}_i)$, which indicates multiple gradient steps

Improved Meta Learning Algorithms

- Challenge: Higher order derivatives
  - MAML requires taking the gradient of many inner gradient steps
  - This is computationally expensive. It also requires a large memory.
- Let $\phi_i = \text{Alg}(\theta, \mathcal{D}_i)$, which indicates multiple gradient steps
- Reptile [Nichol et al., 2018]
  - Uses the task- parameters as targets and slowly moves meta-parameters

$$\theta \leftarrow \theta - \beta \frac{1}{n} \sum_{i=1}^{n} (\phi_i - \theta)$$

Improved Meta Learning Algorithms

- Challenge: Higher order derivatives
  - MAML requires taking the gradient of many inner gradient steps
  - This is computationally expensive. It also requires a large memory.
- Let $\phi_i = \text{Alg}(\theta, \mathcal{D}_i)$, which indicates multiple gradient steps
- Reptile [Nichol et al., 2018]
  - Uses the task- parameters as targets and slowly moves meta-parameters

$$\theta \leftarrow \theta - \beta \frac{1}{n} \sum_{i=1}^{n} (\phi_i - \theta)$$

- Implicit MAML [Rajeswaran et al., 2019]
  - Derives an analytical meta-gradient using the implicit function theorem

Meta RL Formulation

- In Meta RL, each task $\mathcal{T}_i$ is an MDP $\mathcal{M}_i$
- Assumption: MDPs $\mathcal{M}_i$s are drawn according to a distribution $p(\mathcal{M})$

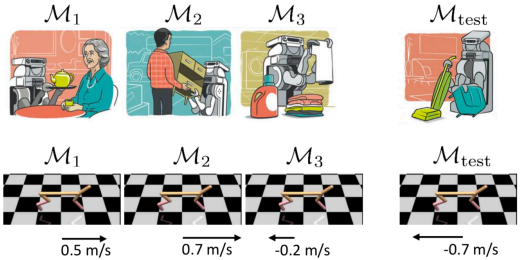
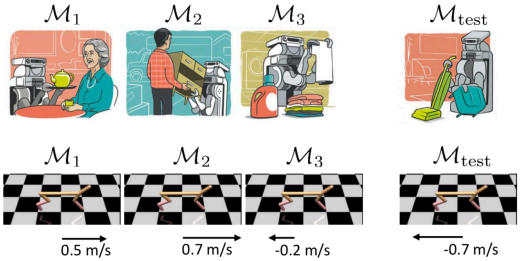


Image from UC Berkeley CS 285: Deep Reinforcement Learning course slides [Link]

Meta RL Formulation

- In Meta RL, each task $\mathcal{T}_i$ is an MDP $\mathcal{M}_i$
- Assumption: MDPs $\mathcal{M}_i$s are drawn according to a distribution $p(\mathcal{M})$



- Meta RL problem

$$\max_{\theta} \sum_{i=1}^{n} V_{\pi_{\phi_i}, \mathcal{M}_i} \quad \text{where,} \quad \pi_{\phi_i} = \text{Alg}(\theta, \mathcal{M}_i)$$

Image from UC Berkeley CS 285: Deep Reinforcement Learning course slides [Link]

MAML for RL

- Meta RL problem: $\max_\theta \sum_{i=1}^{n} V_{\pi_{\phi_i}, \mathcal{M}_i}$ where, $\pi_{\phi_i} = \text{Alg}(\theta, \mathcal{M}_i)$

MAML for RL

- Meta RL problem: $\max_\theta \sum_{i=1}^{n} V_{\pi_{\phi_i}, \mathcal{M}_i}$ where, $\pi_{\phi_i} = \text{Alg}(\theta, \mathcal{M}_i)$
- Change the notation as $\max_\theta \sum_{i=1}^{n} V(\phi_i, \mathcal{M}_i)$ where, $\phi_i = \text{Alg}(\theta, \mathcal{M}_i)$

MAML for RL

- Meta RL problem: $\max_\theta \sum_{i=1}^n V_{\pi_{\phi_i}, \mathcal{M}_i}$ where, $\pi_{\phi_i} = \text{Alg}(\theta, \mathcal{M}_i)$
- Change the notation as $\max_\theta \sum_{i=1}^n V(\phi_i, \mathcal{M}_i)$ where, $\phi_i = \text{Alg}(\theta, \mathcal{M}_i)$
- In MAML, we get ϕ_i by one (or multiple) gradient steps starting from θ

$$\phi_i = \theta + \alpha \nabla_\theta V(\theta, \mathcal{M}_i)$$

MAML for RL

- Meta RL problem: $\max_\theta \sum_{i=1}^n V_{\pi_{\phi_i}, \mathcal{M}_i}$ where, $\pi_{\phi_i} = \text{Alg}(\theta, \mathcal{M}_i)$
- Change the notation as $\max_\theta \sum_{i=1}^n V(\phi_i, \mathcal{M}_i)$ where, $\phi_i = \text{Alg}(\theta, \mathcal{M}_i)$
- In MAML, we get ϕ_i by one (or multiple) gradient steps starting from θ

$$\phi_i = \theta + \alpha \nabla_\theta V(\theta, \mathcal{M}_i)$$

- $\nabla_\theta V(\theta, \mathcal{M}_i)$ is the standard policy gradient step

MAML for RL

- Meta RL problem: $\max_\theta \sum_{i=1}^n V_{\pi_{\phi_i}, \mathcal{M}_i}$ where, $\pi_{\phi_i} = \text{Alg}(\theta, \mathcal{M}_i)$
- Change the notation as $\max_\theta \sum_{i=1}^n V(\phi_i, \mathcal{M}_i)$ where, $\phi_i = \text{Alg}(\theta, \mathcal{M}_i)$
- In MAML, we get ϕ_i by one (or multiple) gradient steps starting from θ

$$\phi_i = \theta + \alpha \nabla_\theta V(\theta, \mathcal{M}_i)$$

- $\nabla_\theta V(\theta, \mathcal{M}_i)$ is the standard policy gradient step
- Meta parameter update is done as

$$\theta \leftarrow \theta + \beta \nabla_\theta \sum_{i=1}^n V(\theta + \alpha \nabla_\theta V(\theta, \mathcal{M}_i), \mathcal{M}_i)$$

References I

Lilian Weng. Meta-learning: Learning to learn fast. *lilianweng.github.io/lil-log*, 2018. URL http://lilianweng.github.io/lil-log/2018/11/29/meta-learning.html.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702. PMLR, 2019.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

References II

Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. *arXiv preprint arXiv:1909.04630*, 2019.