Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN Faculty
Department of Informatics

Transfer Learning using Meta-learning

Nilesh Vijayrania

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

04. November 2019

Outline

Traditional DL Model vs Transfer Learning

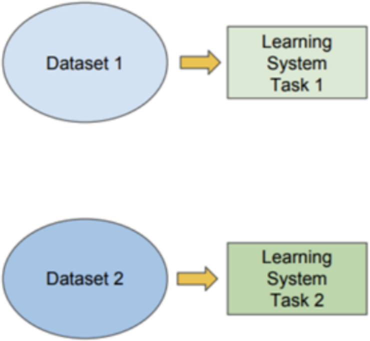


Image Credits: Sarkar D. for medium post on 'A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning'

Motivation

Traditional DL Model vs Transfer Learning

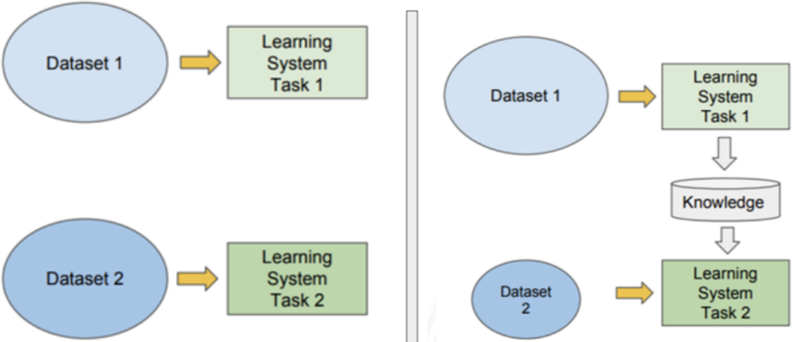


Image Credits: Sarkar D. for medium post on 'A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning'

Motivation

- Is Deep Learning Bio-inspired?
- Problems with DL
  - Data hungry
  - Long training time
  - No progressive learning like humans

How can we learn like Humans do?

- Learn with few samples(Few-shot Learning)
- Leverage prior knowledge

What is Transfer Learning?

- Transfer the knowledge from one task to another

Why Transfer Learning?

- Can help in learning quicker and with fewer examples
- Provides a way for few-shot learning

Few-Shot Learning

Using Data Augmentation *Using Meta Learning*

Transfer Learning

What is Transfer Learning?

- Transfer the knowledge from one task to another

Why Transfer Learning?

- Can help in learning quicker and with fewer examples
- Provides a way for few-shot learning

Few-Shot Learning

Using Data Augmentation *Using Meta Learning*

Transfer Learning

What is Transfer Learning?

- Transfer the knowledge from one task to another

Why Transfer Learning?

- Can help in learning quicker and with fewer examples
- Provides a way for few-shot learning

Few-Shot Learning

Using Data Augmentation *Using Meta Learning*

Transfer Learning

What is Transfer Learning?

- Transfer the knowledge from one task to another

Why Transfer Learning?

- Can help in learning quicker and with fewer examples
- Provides a way for few-shot learning

Few-Shot Learning

Using Data Augmentation *Using Meta Learning*

Transfer Learning

What is Transfer Learning?

- Transfer the knowledge from one task to another

Why Transfer Learning?

- Can help in learning quicker and with fewer examples
- Provides a way for few-shot learning

Few-Shot Learning

Using Data Augmentation *Using Meta Learning*

Transfer Learning

What is Transfer Learning?

- Transfer the knowledge from one task to another

Why Transfer Learning?

- Can help in learning quicker and with fewer examples
- Provides a way for few-shot learning

Few-Shot Learning

Using Data Augmentation *Using Meta Learning*

What is Meta-Learning?

- Meta Learning is learning from multiple tasks
- Also referred as "Learning To Learn"

Supervised Learning: $f(x) \rightarrow y$
Meta-Supervised Learning: $f(D_{train}, x) \rightarrow y$

Meta-Learning Model

1. Given Dataset D, split the tasks into D_{train} and D_{test}
2. Train the network for some epochs on D_{train}
3. Test the network on D_{test}

What is Meta-Learning?

- Meta Learning is learning from multiple tasks
- Also referred as "Learning To Learn"

Supervised Learning: $f(x) \rightarrow y$
Meta-Supervised Learning: $f(D_{train}, x) \rightarrow y$

Meta-Learning Model

1. Given Dataset D, split the tasks into D_{train} and D_{test}
2. Train the network for some epochs on D_{train}
3. Test the network on D_{test}

Meta Learning

What is Meta-Learning?

- Meta Learning is learning from multiple tasks
- Also referred as "Learning To Learn"

Supervised Learning: $f(x) \rightarrow y$
Meta-Supervised Learning: $f(D_{train}, x) \rightarrow y$

Meta-Learning Model

1. Given Dataset D, split the tasks into D_{train} and D_{test}
2. Train the network for some epochs on D_{train}
3. Test the network on D_{test}

What is Meta-Learning?

- Meta Learning is learning from multiple tasks
- Also referred as "Learning To Learn"

Supervised Learning: $f(x) \rightarrow y$
Meta-Supervised Learning: $f(D_{train}, x) \rightarrow y$

Meta-Learning Model

1. Given Dataset D, split the tasks into D_{train} and D_{test}.
2. Train the network for some epochs on D_{train}
3. Test the network on D_{test}

Meta Learning

What is Meta-Learning?

- Meta Learning is learning from multiple tasks
- Also referred as "Learning To Learn"

Supervised Learning: $f(x) \rightarrow y$
Meta-Supervised Learning: $f(D_{train}, x) \rightarrow y$

Meta-Learning Model

1. Given Dataset D, split the tasks into D_{train} and D_{test}.
2. Train the network for some epochs on D_{train}
3. Test the network on D_{test}

Meta Learning

What is Meta-Learning?

- Meta Learning is learning from multiple tasks
- Also referred as "Learning To Learn"

Supervised Learning: $f(x) \to y$
Meta-Supervised Learning: $f(D_{train}, x) \to y$

Meta-Learning Model

1. Given Dataset D, split the tasks into D_{train} and D_{test}.
2. Train the network for some epochs on D_{train}
3. Test the network on D_{test}

What is Meta-Learning?

- Meta Learning is learning from multiple tasks
- Also referred as "Learning To Learn"

Supervised Learning: $f(x) \rightarrow y$
Meta-Supervised Learning: $f(D_{train}, x) \rightarrow y$

Meta-Learning Model

1. Given Dataset D, split the tasks into D_{train} and D_{test}.
2. Train the network for some epochs on D_{train}
3. Test the network on D_{test}

A Simple View On Meta Learning Problem

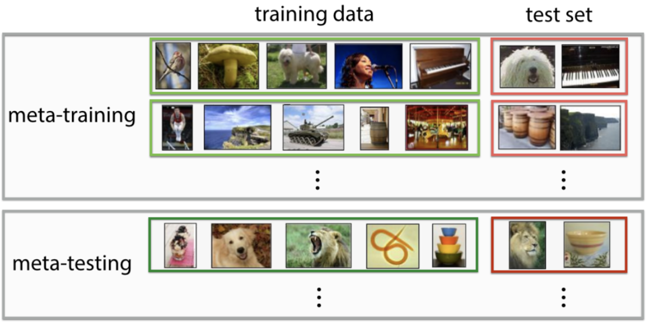


image credit: Ravi & Larochelle 2017

image credit: S. Levine. Taken from deeprlcourse-fa17 Lec 16

A Simple View On Meta Learning Problem

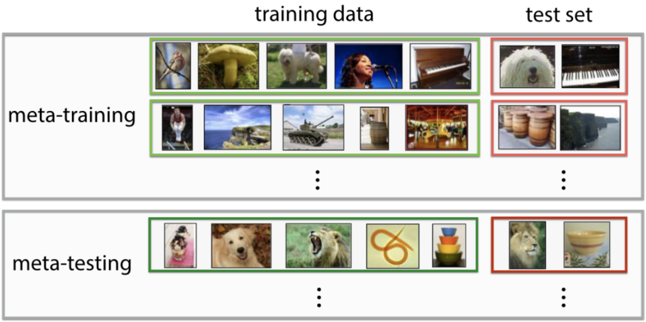


image credit: Ravi & Larochelle 2017

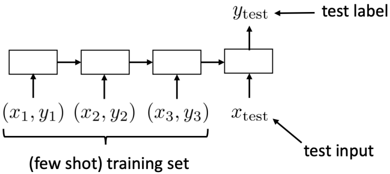


image credit: S. Levine. Taken from deeprlcourse-fa17 Lec 16

Meta Learning Approaches

- Three Approaches for Meta-Learning

- Meta Learning
  - *Metric Learning*
    - Learn embeddings
  - *Memory Based*
    - Store experience
  - *Gradient Based*
    - Using RNNs
    - MAML

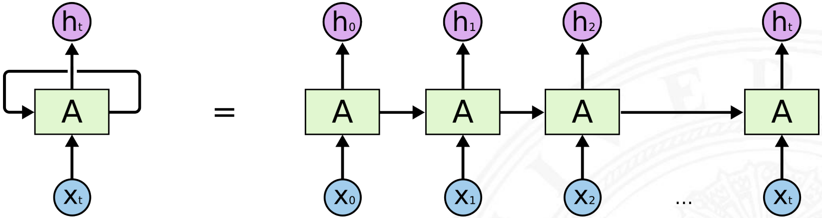
Meta Learning Approaches

- Three Approaches for Meta-Learning

Meta Learning
- *Metric Learning*
  - Learn embeddings
- *Memory Based*
  - Store experience
- *Gradient Based*
  - Using RNNs
  - MAML

Recurrent Neural Networks

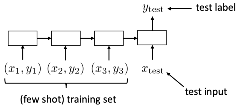
- RNNs consider info from prev timesteps
- Used for sequence modelling



Unfolded RNNs: Image taken from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Meta-Learning Using RNN

Learn the optimizer that guides the model to learn different tasks eg. Using RNN



Makes use of two deep networks

1. Meta-Learner(to learn the task independent features)
2. Base-Learner(to learn the task-dependent features)

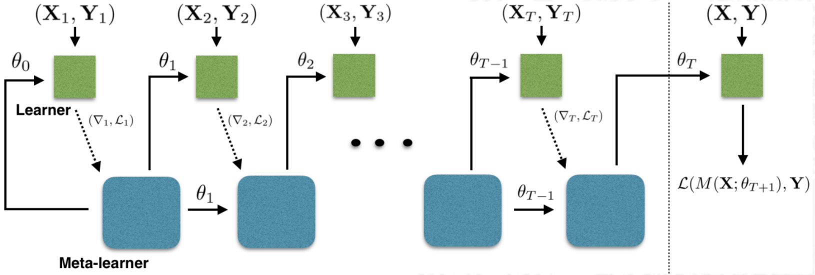


image credit: Ravi and Larochelle

Another Method: MAML

Learn the good initialization weights for the model which could be easily fine-tuned eg. using MAML

- Goal is to provide a model which once fine tuned on a particular task, can learn rapidly and can generalize well
- MAML, provides a good initialization for the model which needs to be fine tuned(similar to tranfer learning on ImageNet)

Image credits: Finn et al. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Another Method: MAML

Learn the good initialization weights for the model which could be easily fine-tuned eg. using MAML

- Goal is to provide a model which once fine tuned on a particular task, can learn rapidly and can generalize well
- MAML, provides a good initialization for the model which needs to be fine tuned(similar to tranfer learning on ImageNet)

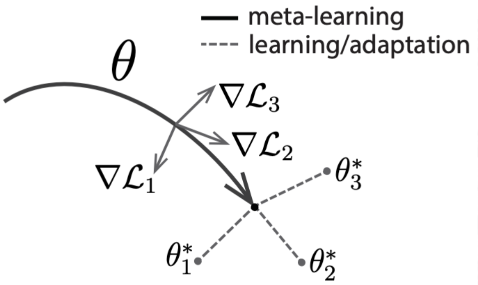


Image credits: Finn et al. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Pseudo code

Algorithm 1 Train MAML

Input:Data p(τ): Distribution over tasks, model params θ

$\theta \leftarrow$random initialization
while not done **do**:
  Sample batch of tasks $T_i \sim p(\tau)$
  for i=1 to N **do**:
    Sample K datapoints for Task T$_i$
    $\theta'_i \leftarrow \theta - \alpha \nabla_\theta L_{T_i}(f_\theta)$ ▷ Gradient wrt K examples
    Sample Datapoints $D' = \{x^j, y^j\}$ for meta-update
  end for
  $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau \sim p(\tau)} L_{T_i}(f_{\theta'_i})$ ▷ Update Meta-Learner wrt D'
end while

MAML: cont'd

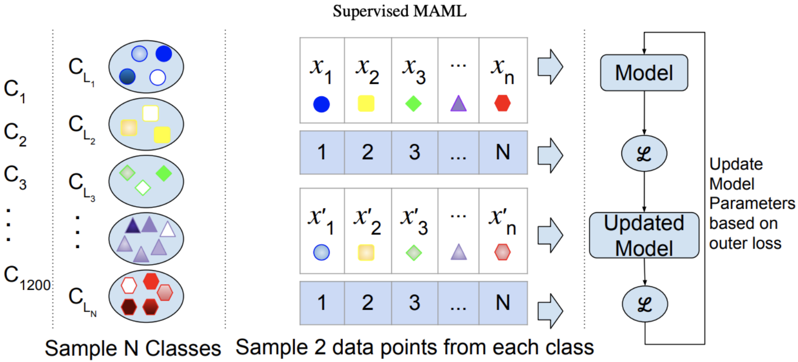


Image credits: Khodadadeh S. et al. Unsupervised Meta-Learning For Few-Shot Image and Video Classification

MAML: cont'd

- Suitable for any kind of ML problem , regression, classification and RL
- Need to adapt the loss function as per problem

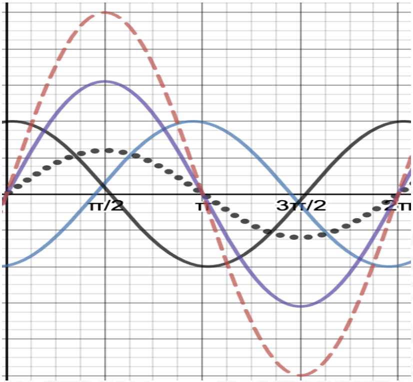
Common Loss Functions

- Regression loss: Mean squared error
  $$L_{T_i}(f_\phi) = \sum_{x^{(j)},y^{(j)} \sim T_i} ||f_\phi(x^{(j)}) - y^{(j)}||^2$$
- Classification loss: cross-entropy loss
  $$L_{T_i}(f_\phi) = \sum_{x^{(j)},y^{(j)} \sim T_i} y^{(j)} \log f_\phi(x^{(j)}) - (1 - y^{(j)}) \log(1 - f_\phi(x^{(j)}))$$
- RL loss: negative avg episodic reward
  $$L_{T_i}(f_\phi) = -\mathbb{E}_{x^t, a^t \sim f_{\phi, q_{T_i}}} \sum_{t=1}^{T} R_i(x_t, a_t)$$

Experiments-Simple Regressor

Training Dataset:

- 70000 random Sinusoid Wave
- amplitude $\in$ [0.1, 5.0]
- Phase varies within $[0, \pi]$
- datapoints x $\sim$ [-5.0, 5.0]



MAML training:

- take N=70000
- sample k=5 datapoints for each sinwave
- Train MAML learner

Meta-Testing:

- take random sine wave $\rightarrow$ [0.1, 5]
- sample k=5 datapoints for the selected sinwave
- Fine-tune the model for the task and measure performance

Experiments-Cont'd

Trained Baseline Models

- Oracle Model(fed with amplitude and phase beforehand)
- pre-trained on the randomly generated sine waves and fit a regressor

Evaluation:

- Select 600 points at random i.e.$x_{test} \in [-5, 5]$
- Calculate y_{test} for the selected task for meta-testing
- get y_{pred} from the models for x_{test}
- Calculate MSE loss between y_{test} and y_{pred}

Model Details:

- 2 layer NN with 40 neurons in each hidden Layer and Relu in between
- Trained with ADAM optimizer

Results

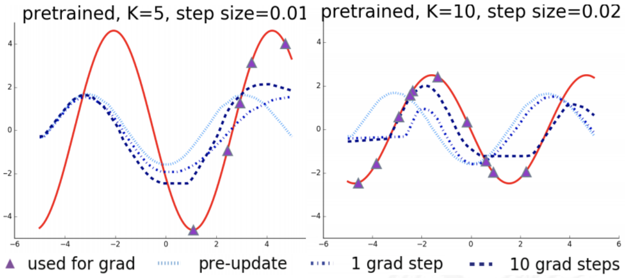


image courtesy: Finn C. et al [1]

Results

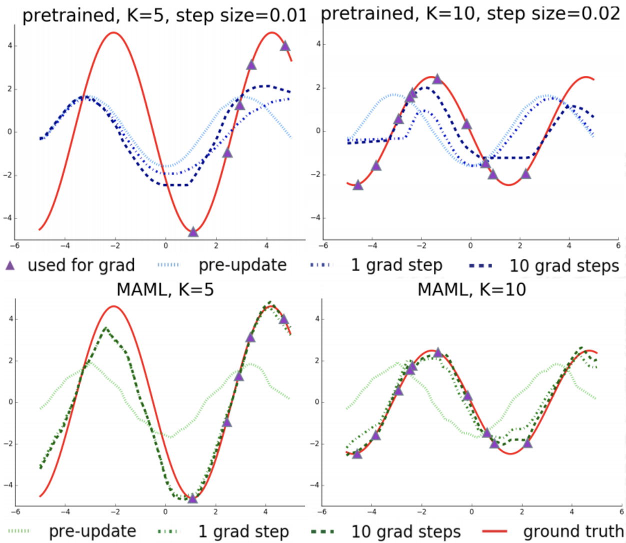


image courtesy: Finn C. et al [1]

Experiment-Robot Locomotion

Task:

- Robot(e.g. 2 legged cheetah or 4 legged ant) locomotion in simple 3D world
- Tasks are to attain a target speed or walk in a particular direction

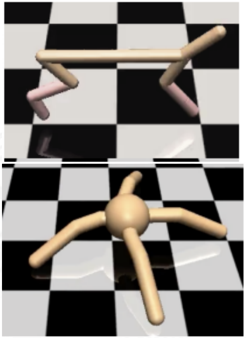


image courtesy: Finn C. et al [1]

MDP details:

- Observation space –> current coordinates
- Action space –> joint angles to move
- Rewards:
  - **For goal velocity tasks:** negative of absolute diff of goal velocity and current velocity
  - **For goal direction tasks:** magnitude of goal velocity in desired direction

Experiment-Robot Locomotion Cont'd

meta-training:

- Sample a task for goal velocity from $\sim [0.0, 2.0]$
- For each task, generate k=20/40 policy rollouts(samples) and fit the learner
- For each task, generate the meta-test observations and update the meta learner using loss on meta-testset.

meta-test:

- Sample a task for goal velocity from $\sim [0.0, 2.0]$
- Generate k=20/40 samples policy rollouts and fine tune the learner

Experiment-Robot Locomotion Results

1. Randomly trained policy
2. MAML trained policy

Conclusion

+ Meta-Learning shows potential for more human like learning
+ Works with only few samples(Saves effort on data labelling)
+ Algorithms like MAML show early success in field on related tasks
– Still in early phase, and requires the tasks to be related and similar.
– Requires large number of similar tasks
– Uses shallow networks to avoid overfitting which restricts the representational powers of model
– Need for more mature algorithms for more human like learning

Thank You For Your Attention!

Questions??

References

1. Finn C., Abbeel P., Levine S. in CoRR 2017, Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, http://arxiv.org/abs/1703.03400
2. Ravi S., Larochelle H. in ICLR 2017, Optimization As a Model for Few-shots learning https://openreview.net/forum?id=rJY0-Kcll
3. Finn Chelsea, Learning to Learn https://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/

Pseudo code

Algorithm 2 Train Meta-Learner

Input:Data D_{train}, Learner(M) params θ, Meta-learner(R) params ϕ

$\phi_0 \leftarrow$random initialization
for d=1,n **do**
  D_{train}, D_{test} gets random dataset from $D_{MetaTrain}$
  $\theta_0 \leftarrow c_0$ ▷ Initialize learner parameters
  for t=1,T **do**
    X_t, Y_t gets random batch from D_{train}
    $L_t \leftarrow L(M(X_t; \theta_{t-1}), Y_t)$ ▷ learner loss on train batch
    $c_t \leftarrow R((\nabla_{\theta_{t-1}} L_t, L_t); \phi_{d-1})$ ▷ output of meta-learner
    $\theta_t \leftarrow c_t$ ▷ Update learner parameters
  end for
  $X, Y \leftarrow D_{test}$
  $L_{test} \leftarrow L(M(X; \theta_t), Y)$ ▷ learner loss on test batch
  Update ϕ_t using $\nabla_{\Theta_{d-1}} L_{test}$ ▷ Update meta-learner params
end for