Meta-learning

Gautam Shroff

Spring 2021

Outline

Weeks 1 and 2 (HWs 1,2,3, Q1)

1. Meta-learning: Basic Techniques, Some Theory and Extensions

Week 3 (HW 4, Q2)

2. Meta-Learning for Transfer, Continual and Reinforcement Learning

Week 4 (Q3)

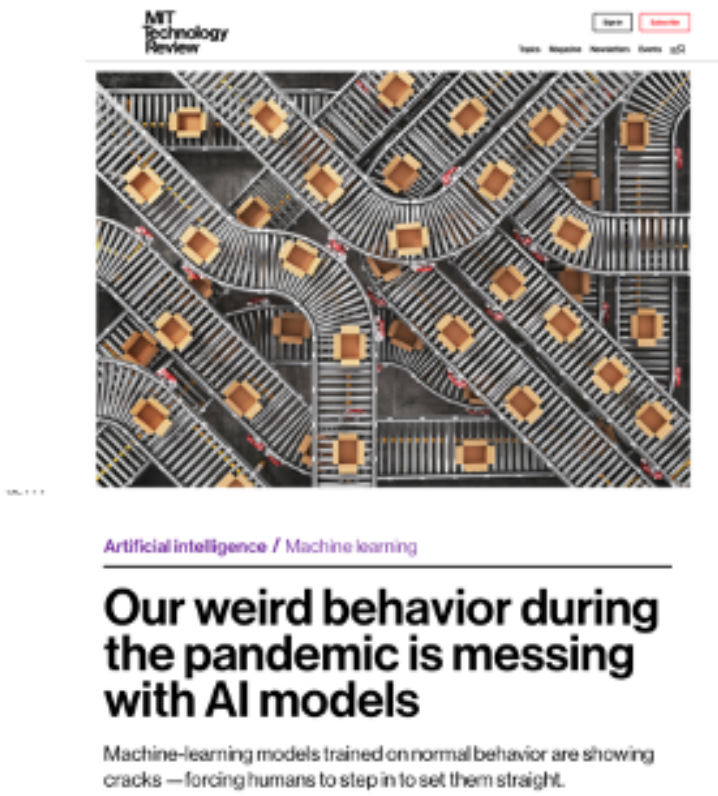
3. Learning from almost No Data
4. Selected applications of Meta-learning

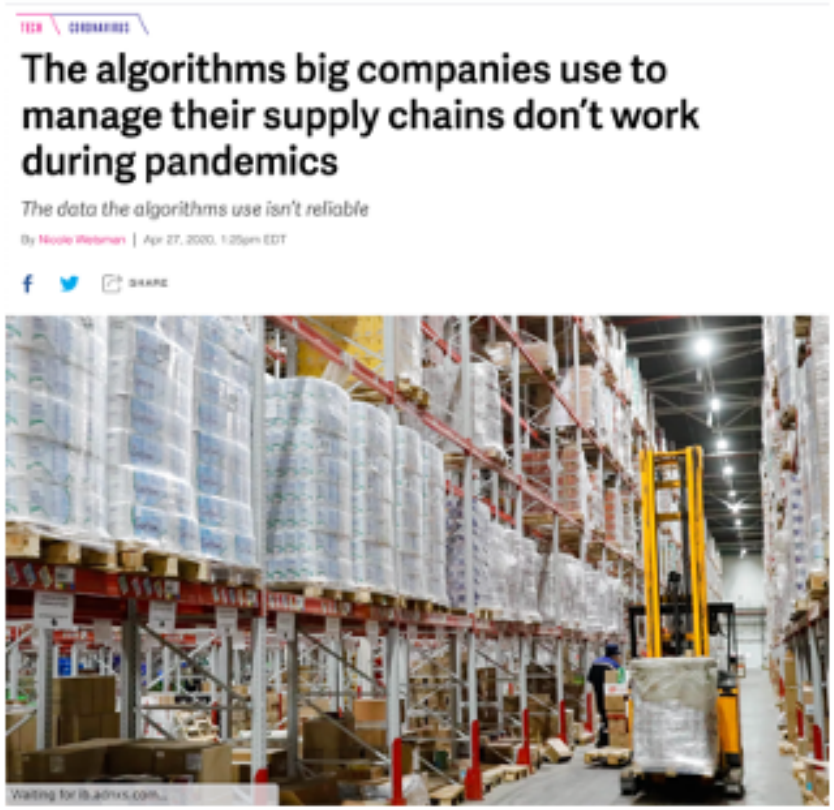
Week 5 (Q4)

5. Causality, Disentangled Representations and Causal RL

Motivation 1: Learning from Less Data

Motivation 2: A Changing World





1. Basics of Meta-learning

Machine Learning

A supervised learning *task* T is defined by the following:

Given 'training' data: $D = \{\mathrm{x}_i, y_i\}$ from some distribution $\mathcal{D}$,
'learn' a 'model' $y = f_{\mathrm{w}}(\mathrm{x})$, with parameters w that are chosen so as to:
minimise some loss $\mathcal{L}(\mathrm{w}, D)$, e.g., squared error: $\sum_i \|y_i - f_{\mathrm{w}}(\mathrm{x_i})\|^2$. or
negative log-likelihood: $\sum_i \sum_c y_i^c \log f_{\mathrm{w}}^c(\mathrm{x_i})$

Ideally the model should 'generalise well', so that the *expected loss* for 'test data' sampled from $\mathcal{D}$ is also small.

Machine-learning Examples

1. Linear regression:
$$y = f_{\mathrm{w}}(\mathrm{x}) = \mathrm{x}^T\mathrm{w} = [\mathrm{w}^T\mathrm{x}]^T, \mathrm{x}, \mathrm{w} \in \mathbb{R}^N$$

2. Linear classifier:
$$y = f_{\mathrm{W}}(\mathrm{x}) = \mathtt{softmax}([\mathrm{W}\mathrm{x}]^T), \mathrm{W} \in \mathbb{R}^{CxN}$$

2. Deep learning:
$$y = f_{\mathrm{w}}(\mathrm{x}) = [g(W_n \ldots g(W_2(g(W_1\mathrm{x}))\ldots)]^T$$

"multi-layer perceptron (MLP) with non-linearity g, e.g., $g = ReLU()$"

loss $\mathcal{L}(\{W_i\}, D)$ minimised by e.g. 'stochastic gradient descent'

gradient of *loss* w.r.t. parameters $\{W_i\}$ computed using 'back-propagation' = 'chain-rule for derivatives together with some dynamic programming to reuse intermediate results'

Meta-learning: Motivation & 'Definition'

Many, possibly different, tasks T, are experienced (by humans/machines).

What is a 'task'?
$D = \{D_{Train}, D_{Test}\} \sim \mathcal{D}$, a data distribution; and loss $\mathcal{L}$
where $D_{Train} = (X_{Train}, y_{Train})$ and $D_{Test} = (X_{Test}, y_{Test})$. So a task $T = (D, \mathcal{L})$. We assume a *distribution* $\mathcal{T}$ over such tasks.

Meta-learning, or 'learning to learn', attempts to learn from learning experiences across tasks, so as to learn 'better' on future tasks.
'Better' could mean with greater accuracy and/or with smaller amounts of data and/or faster exploration of functional forms of f.

Learning to discover better learning 'architectures' (f) is called **AutoML**
Learning across different data distributions is called **transfer learning**
Learning to learn faster and with less data is called **few-shot learning**
There are other (non meta-learning) techniques for few-shot/transfer learning: e.g. deductive learning/program synthesis

Learning from prior tasks via Pre-training

Reference - *Greedy layer-wise pre-training*

Layer-wise pre-training, e.g. train using layer 1 plus an output layer only, throw away output layer and use output of layer 1 to train layer 2 ... and so on. Target can be supervised, i.e., to predict labels, or unsupervised, i.e., to reconstruct the input (either directly or from a noisy version).

Reference - *Why Does Unsupervised Pre-training Help Deep Learning?*

Helps initialise weights to a region close to a local optima so that later training ('fine tuning') does better.

Reference - *How transferable are features in deep neural networks?*

Transferring weights from lower layers of one network to another, to see how general are the features being learned.

However, none of these actually explored meta-learning using pre-trained networks, i.e., whether training on many different datasets or tasks helps with new tasks and/or learning from less data.

Pre-training 2: time-series & NLP; how to pre-train better

Reference - *TimeNet: Pre-trained deep recurrent neural network for time series classification*

Timenet: sequence-to-sequence LSTM model trained to reconstruct diverse time-series; embeddings used as features to train SVM. Classifiers using Timenet embeddings beat traditional and deep baseline classifiers.

Reference - *ConvTimeNet: A Pre-trained Deep Convolutional Neural Network for Time Series Classification*

Conv-Timenet: convolutional (in-time) network trained on diverse classification tasks transfers to new data (with fine-tuning) beating baselines as well as itself trained from scratch on target dataset.

Reference - *Improving Language Understanding by Generative Pre-Training*

Transformer network trained for *language modelling*; pre-trained layers used for a variety of NLP tasks and beating baselines.

Reference - *Domain Adaptive Transfer Learning with Specialist Models*

Selecting data to pre-train on using importance sampling improves over using all data for pre-training.

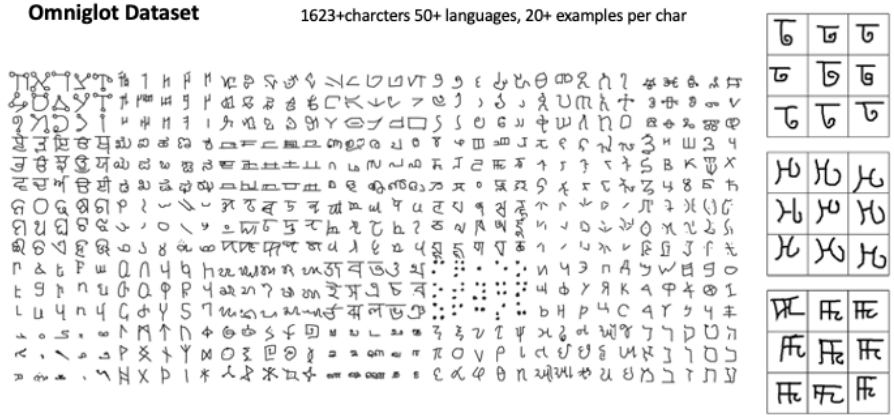
Early work in meta-learning and few-shot learning

Reference - *Schimduber's "Godel Machine"*

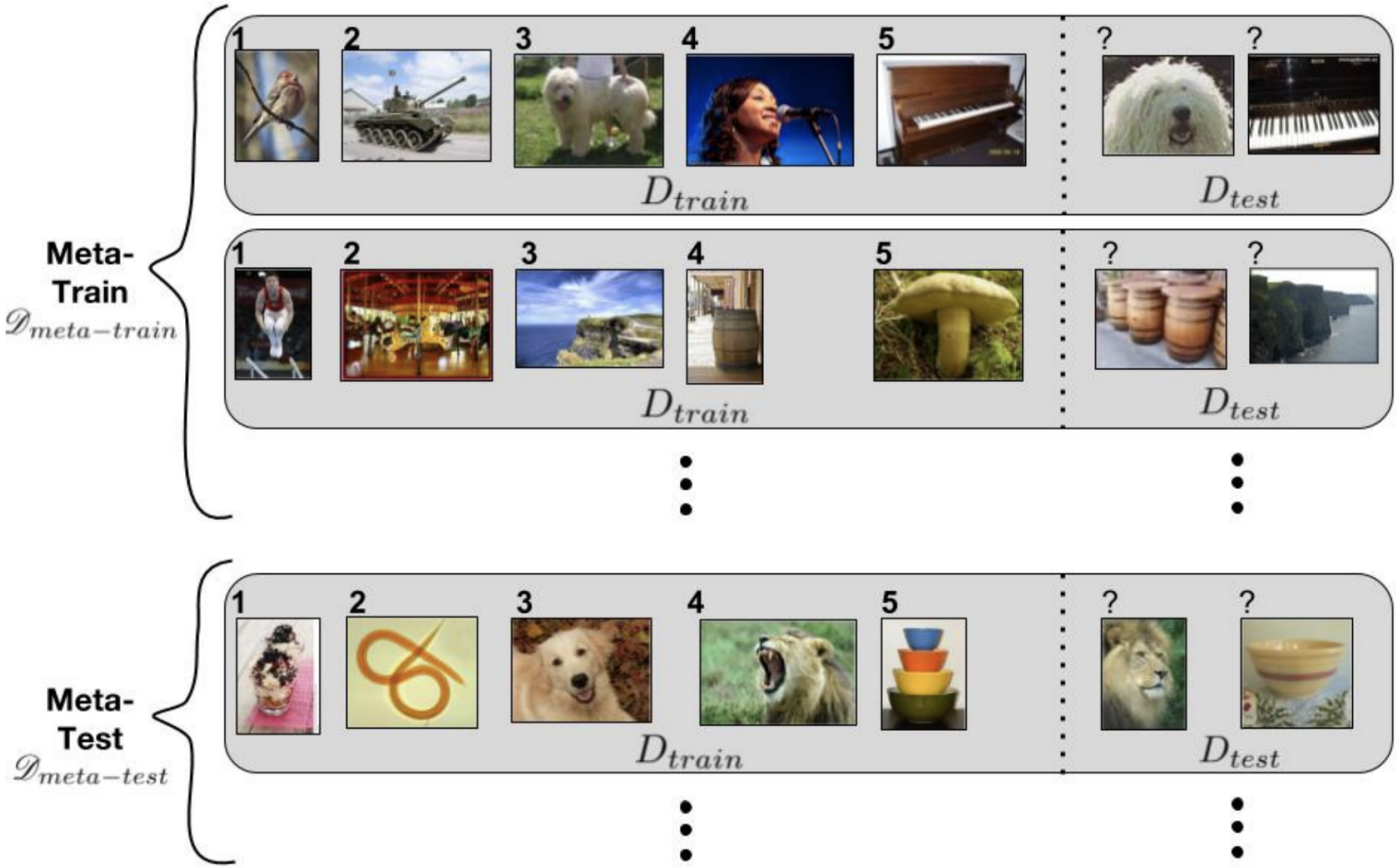
Architecture for a general-purpose meta-learner, where the inner learner could be a neural network or even a Turing machine. Key idea of optimisation across tasks introduced.

Reference - *Siamese Neural Networks for One-shot Image Recognition*

Use a 'siamese' network to learn a distance metric between images, and use it for few-shot nearest-neighbour classification.



Definitions: via Few-shot Learning



Key Idea in Meta-learning

Machine learning via empirical risk minimisation:

$$\mathsf{w} = \text{argmin}_{\mathsf{w}}\, \mathbb{E}_{D_{Train} \sim \mathcal{D}} \mathcal{L}(f_{\mathsf{w}}, D_{Train})$$

and *hope* that

$$\mathbb{E}_{D_{Test} \sim \mathcal{D}} \mathcal{L}(f_{\mathsf{w}}, D_{Test}) \text{ is also small.}$$

Loss $\mathcal{L}$ depends on **problem**: classification NLL, regression MSE, RL ...

Key idea in meta-learning:

If the objective is to minimise $\mathbb{E}_{D_{Test} \sim \mathcal{D}} \mathcal{L}(f_{\mathsf{w}}, D_{Test})$,

Why not minimise it directly?

Instead of $\bar{y} \leftarrow f_{\mathsf{w}}(x)$, use $f : \underline{\bar{y} \leftarrow f_{\mathsf{w}}(x, D_{Train})}$.

Generic Meta-learning Algorithm

Given task distribution $\mathcal{T}$, meta-learner f_{w}

for $e \leftarrow 1$ to N_{epochs} **do**
  for $i \leftarrow 1$ to N_{steps} **do**
    Sample $\{D_{Train}, D_{Test}\} \sim \mathcal{T}$
    $\Delta_i = \mathcal{L}(f_{\mathrm{w}}(D_{Test}, D_{Train}))$
  end
  $\mathrm{w} \leftarrow \mathrm{w} - \eta \nabla \sum_i \Delta_i$
end

Algorithm 1: MetaLearn

Different kinds of Meta-learners

1. **Optimization-based** meta-learners:
   f_w *adapts* a base learner g_w.

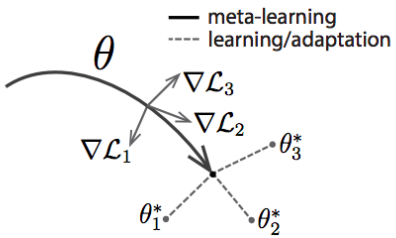
2. **Model-based** meta-learners:
   f_w is a (e.g., recurrent) network taking D_{Train} as input.

3. **Metric-based** meta-learners:
   f_w computes (learned) distances to elements of D_{Train}.

Optimisation-based Meta-learning

Reference - *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*

Choose model parameters w such that taking one or few gradient-steps on an unknown task (i.e., dataset) is maximally optimal: For each task, *adapt* $g_{\theta_0=\mathrm{w}}$ via gradient-step(s) using *test* loss on tasks, i.e., $\theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_i}(g_\theta)$, over a number of tasks $\mathcal{T}_i$ sampled from a distribution. Note: f_w requires similar *adaptation* at (meta)-test time also. Also, as formulated MAML applies both to supervised as well as other ML tasks, e.g. reinforcement learning. Thus:



$$f_\mathrm{w}^{MAML}(x, D_{Train}, g) = g_{\mathrm{w}-\alpha\nabla\mathcal{L}(g_\mathrm{w}(D_{Train}))}(x) \equiv g_\phi(x)$$

In practice, more than one gradient steps are taken: *fast adaptation*.
Note that training f^{MAML}, i.e., optimizing for w across meta-training tasks, requires second-order derivatives, i.e., we need $\nabla_w\mathcal{L}(g_{\hat{\phi}}, D_{Test})$
$= (I - \alpha\nabla_w^2\mathcal{L}(g_w(D_{Train}))\nabla_\phi\mathcal{L}(g_\phi, D_{Test})|_{\phi=\hat{\phi}}$, where $\hat{\phi} = \mathrm{w} - \alpha\nabla\mathcal{L}(g_\mathrm{w}(D_{Train}))$