Meta Learning

Machine Learning for Jet Physics, 2017

Luke de Oliveira
Vai Technologies & Berkeley Lab

@lukede0
@lukedeo
lukedeo@vaitech.io
https://ldo.io

Overview

- Traditional learning paradigm
- Learning to learn
    - Paradigms
- Utility in Jet Physics

Traditional Learning

Traditional (supervised) Learning

- Have some data (X, Y)
- Would like to learn a hypothesis to map X to Y via $h(\mathrm{X}, \theta)$
- More data, X, and we would like to know Y, but settle for h(X)
- We use tools (optimization, splitting, etc.) to solve for $\theta^* = \mathrm{argmin}_{\theta}\, L(\mathrm{X}, \mathrm{Y}; \theta)$

Traditional Generative Modeling

- Have some data X
- Would like to learn to sample from $\mathrm{P(X)}$ or perhaps evaluate likelihood of X
- Modern tools (GANs, VAEs, Autoregressive Models, etc.) optimize various information theoretic criteria to do this

Standard Setup

- Training data, validation data, test data
- Single task (i.e., *b*-tagging in a specific p_T range, learn a GAN on jet images, etc.)
- Minimize risk / loss / error *in expectation* on test data using only training data

Learning *how* to Learn

Why does the standard setup fail?

- Datasets are always changing
- May have limited data for a specific problem (hard-to-simulate region of phase space)
- Deployment domain ≠ training domain
- Question: can we have models learn the most efficient way to learn?

Meta Learning

- Given a new dataset / problem, can I learn how to efficiently learn to solve the task?

Meta Learning

- Given a new dataset / problem, can I learn how to efficiently learn to solve the task?
- Key solutions:

Meta Learning

- Given a new dataset / problem, can I learn how to efficiently learn to solve the task?
- Key solutions:
    - Replace manual engineering

Meta Learning

- Given a new dataset / problem, can I learn how to efficiently learn to solve the task?
- Key solutions:
    - Replace manual engineering
    - Adapt to new domains

Meta Learning

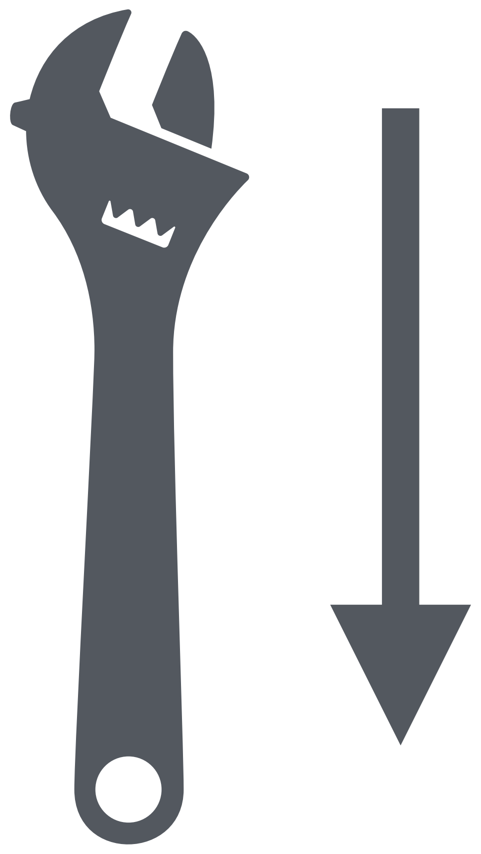
- Given a new dataset / problem, can I learn how to efficiently learn to solve the task?
- Key solutions:
    - Replace manual engineering
    - Adapt to new domains
    - Reuse Data

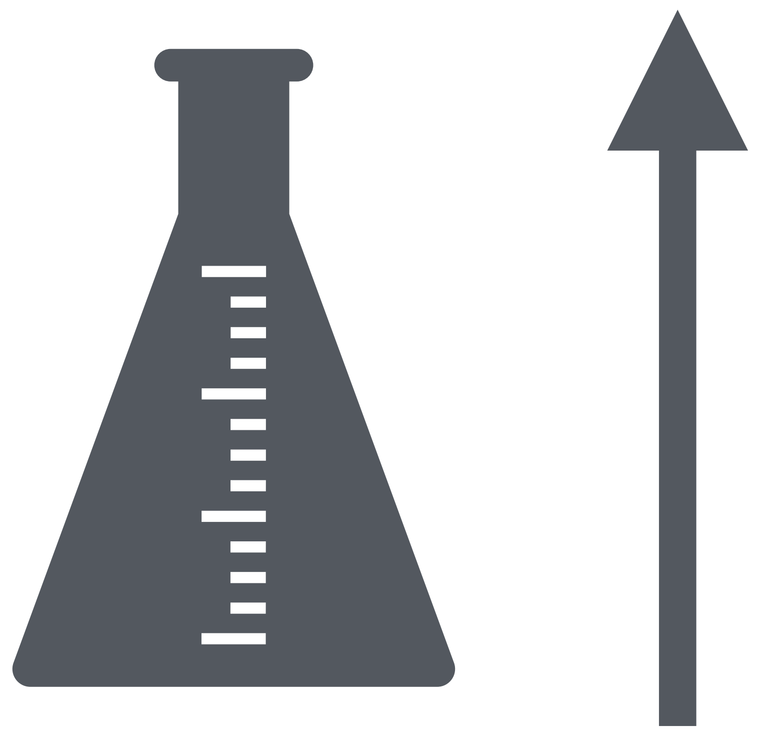
Meta Learning

- Given a new dataset / problem, can I learn how to efficiently learn to solve the task?
- Key solutions:
    - Replace manual engineering
    - Adapt to new domains
    - Reuse Data
    - Weak supervision

Meta Learning

- Given a new dataset / problem, can I learn how to efficiently learn to solve the task?
- Key solutions:
    - Replace manual engineering
    - Adapt to new domains
    - Reuse Data
    - Weak supervision





Designing a Supervised Meta Learner

Designing a Supervised Meta Learner

- Supervised Learning

$$X, y \sim p_{\mathcal{D}}, \{(X_i, y_i)\}_{i=1}^{K} \longrightarrow y = f(X, \theta)$$

Designing a Supervised Meta Learner

- Supervised Learning

$$X, y \sim p_{\mathcal{D}}, \{(X_i, y_i)\}_{i=1}^{K} \longrightarrow y = f(X, \theta)$$

- Meta Supervised Learning

$$\mathcal{D}_{\text{train}} = \{\mathcal{D}_j\}_{j=1}^{J} \longrightarrow y = f(X_{\text{test}}, \mathcal{D}_{\text{train}}; \theta),$$

$$\mathcal{D}_j = \{(X_i, y_i)\}_{i=1}^{K}$$

Designing a Supervised Meta Learner

- Supervised Learning

$$X, y \sim p_{\mathcal{D}}, \{(X_i, y_i)\}_{i=1}^{K} \longrightarrow y = f(X, \theta)$$

- Meta Supervised Learning

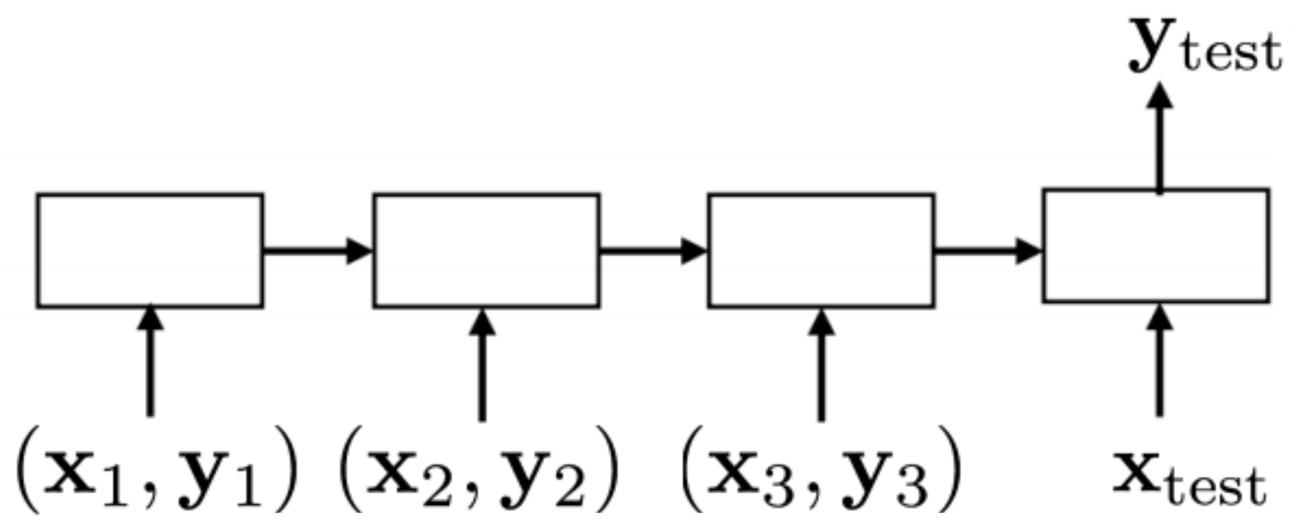
$$\mathcal{D}_{\text{train}} = \{\mathcal{D}_j\}_{j=1}^{J} \longrightarrow y = f(X_{\text{test}}, \mathcal{D}_{\text{train}}; \theta),$$

$$\mathcal{D}_j = \{(X_i, y_i)\}_{i=1}^{K}$$

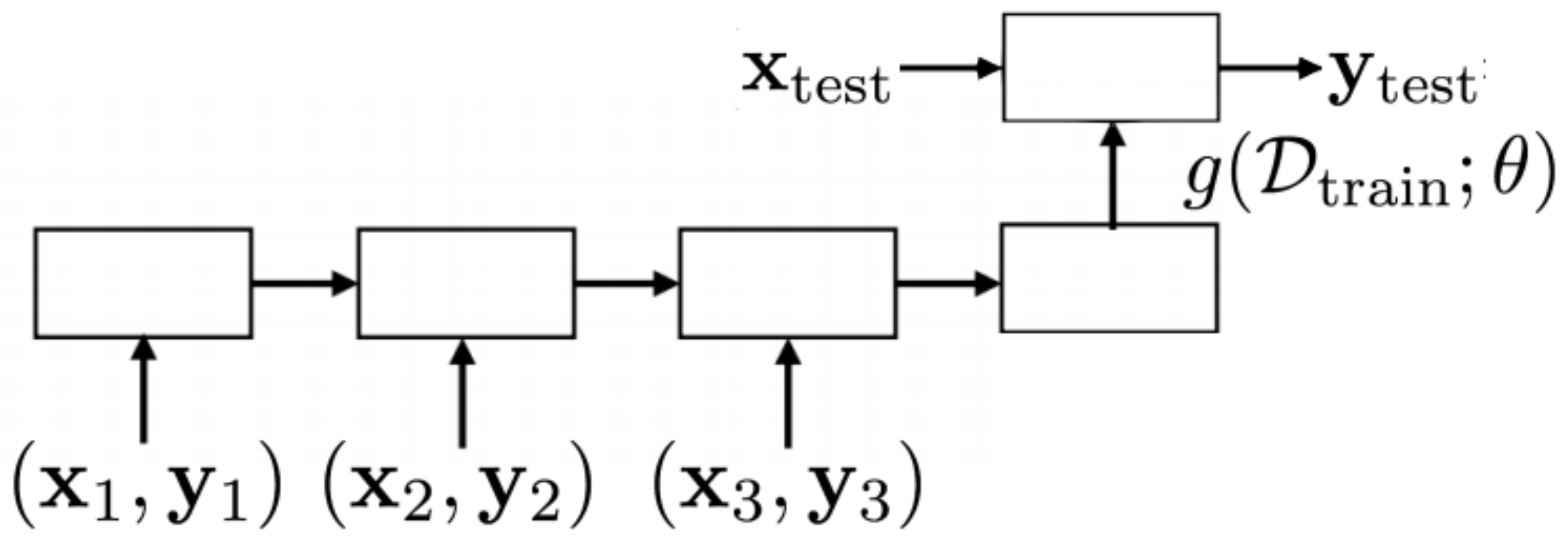
- New notion - meta train & meta test sets

How does one learn a Meta Learner?

- IOW, how can we ingest training sets to learn how to learn?
- Recurrence



- Learn an optimizer



- Learn an initialization

$$\mathbf{y}_{\text{test}} = f(\mathbf{x}_{\text{test}}; \theta - \alpha \nabla_\theta \mathcal{L}(\mathcal{D}_{\text{train}}))$$

- Learn a model architecture

Fig. Credit. C. Finn, '17

One example: Model Agnostic Meta Learning

- Idea: meta learning as a prior over space of models
- Learn a good initialization for new tasks

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

Meta-training: Compute what should be a good parameter value for network at that task

Use all task losses to learn a good starting point, and repeat

Fig. Credit. C. Finn, '17

One example: Model Agnostic Meta Learning

- Idea: meta learning as a prior over space of models
- Learn a good initialization for new tasks

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Evaluate ∇_θ L_{T_i}(f_θ) with respect to K examples
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i})
end while
```

Meta-training: Compute what should be a good parameter value for network at that task

Use all task losses to learn a good starting point, and repeat

These losses can be different! (Weak supervision)

Fig. Credit. C. Finn, '17

Utility in Jet Physics

- Prior over models can help with weak supervision (data-driven tuning)
- Data efficient (i.e., retain memory) over different sections of phase space
- Can automate much of model training

Thanks!

VAI

https://vaitech.io