Modern Topics on Statistical Learning Theory Spring 2023

Lecture 8 — Meta Learning

Prof. Qi Lei *Scribe: Lehan Li, Kristi Topollai, Qiwen Zhang*

1 Meta learning and few shot learning

1.1 Overview

Now we enter the second half of the lecture.

"Modern topics" of statistical learning theory
↓
unit 1
↓
unit 2 and unit 3
↓
Descriptions of settings (how to apply statistical learning theory)

- For unit 1, we primarily focus on statistical learning theory.
- For unit 2 and 3, we focus on modern topics, to apply statistical learning theory from unit 1.

Today's Objective meta-learning (learning to learn) / few-shot learning

$$\text{previous dataset } D = \begin{cases} \text{train} & \text{shared training and testing} \\ \text{test} & \text{we need lots of samples} \end{cases} \tag{1}$$

- Previously, the dataset contained training and testing data, where the data had the same distribution, marginal distribution of the input, and label classes.
- We need lots of samples to fully train the model.
- In reality, human learning process is not the same as above.

1.2 Motivating Examples

We are presented with paintings from two different people (Alice and Bob). And each of them have 3 paintings (3-shots). Thus the training samples are limited for this scenario. Then another new

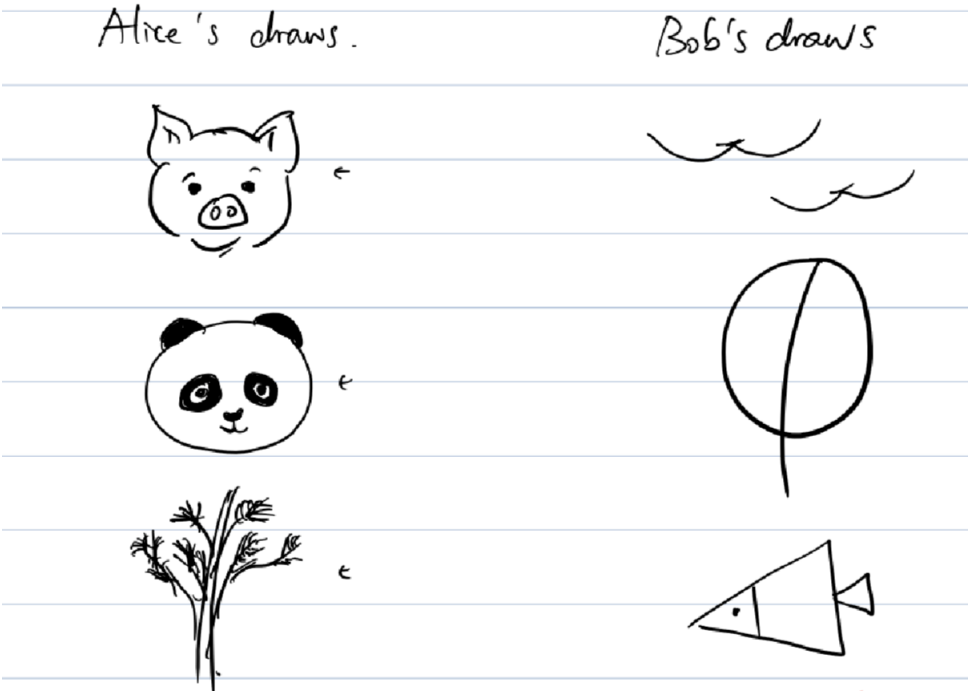

Figure 1: Caption

painting is presented and the task is to recognize if the painting is done by Alice or Bob.

The new painting  $\longrightarrow$ obviously belongs to Bob

From this motivating examples, the key takeaways are

- People use prior knowledge in performing a new task instead of learning from scratch.
- **Similarly, can we use prior information to train Machine Learning models faster and more sample efficiently? (goal of meta and few-shot learning)**
- Train a model on different (but relevant) learning tasks such that it could help solve new tasks with only a few samples.

1.3 Few-shot Classification Problem

One application of meta-learning is the few-shot classification problem. Where the objective is to learn a model to recognize unseen targets during training with limited labeled examples.

- Input $\begin{cases} D_1^s, D_2^s, ..., D_e^s & \text{source} \\ D^T & \text{target, which is the new task for the model} \end{cases}$

  To look inside each D: $\begin{cases} D_i^s = \{x_j^{(i)}, y_j^{(i)}\}_{j=1}^{n_s} \\ D^T = \{x_j^T, y_j^T\}_{j=1}^{n_t} \end{cases}$

  Where we denote by e the number of different tasks. n_s refers to the samples for each source task, n_t refers to the samples for target task. And n_t is much smaller than n_s.

- L-way Classification: $y_i \in [L] = \{1, 2, ..., L\}$
  It means that in each of the individual tasks in the training data set, there are L different categories in the y_i label

- K-shot Learning: k samples per class in each task

- Motivating example: for the example above, it should be a 2-way 3-shot classification problem.

One question regarding to Few-shot Classification Problem is that how to use **structures** learned from D_i^s, i = 1, ..., e to adopt to new task (sample) efficiently?

The solution is to **learn a meta parameter θ, such that**

$$f_i = A(D_i^s, \theta)$$

is good for D_i^s where f_i refers to task specific model parameters, $A(\cdot, \cdot)$ refers to the 'base learner'.

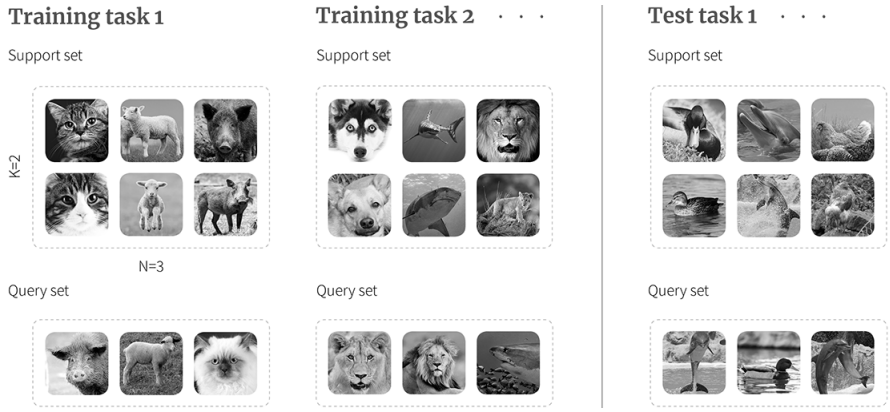

Figure 2: Few shot classification tasks

1.4 Training procedure on some tasks

Training

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{e} \hat{L}(D_i^s, f_i) \text{ such that } f_i = A(D_i^s, \theta)$$

The function minimizes the sum of training losses for each task using the task specific parameters, the task specific parameters are obtained from a meta parameter θ which is shared across all tasks.

Testing

$$f^T = A(D^T, \hat{\theta})$$

The base learner uses trained $\hat{\theta}$ and the test task data set D^T to obtain the test task parameters f^T

1.5 Model Agnostic Meta Learning [4]

Perhaps the most popular meta learning algorithm is Model Agnostic Meta Learning (cit)

- The general idea behind MAML is to find a model initialization $\hat{\theta}$, so that, for a new task, a small number of gradient steps and a small amount of training data are sufficient to produce good generalization performance on that task.
- The function $A(\cdot, \cdot)$ is a gradient based method, and it is basically finetuning the learned θ. In MAML, it consists of a few gradient descent steps, usually 5 to 10.
- MAML training involves two levels of gradient descent. The outer loop trains the model with respect to the shared parameter θ, and the inner loop trains the model with task specific parameters f_i.

In particular, assuming only one gradient step of adaptation, then if θ is the meta solution/initialization and $p(\mathcal{T})$ is a distribution over tasks, the training procedure is formulated as follows

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}\right)$$

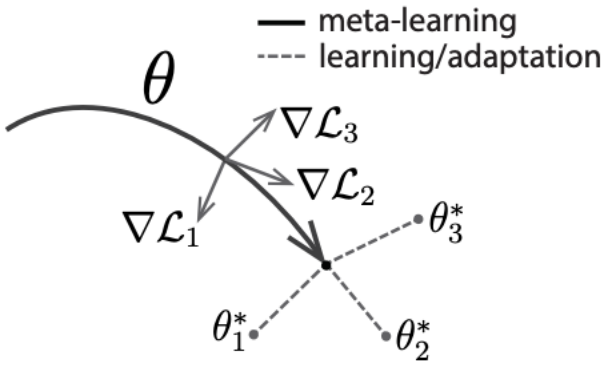

Figure 3: MAML

2 Finding a shared representation

Another approach to meta-learning is to find a good shared representation [1]), for example, the prototypical network [5]. An illustration of the network can be referred to figure 4:

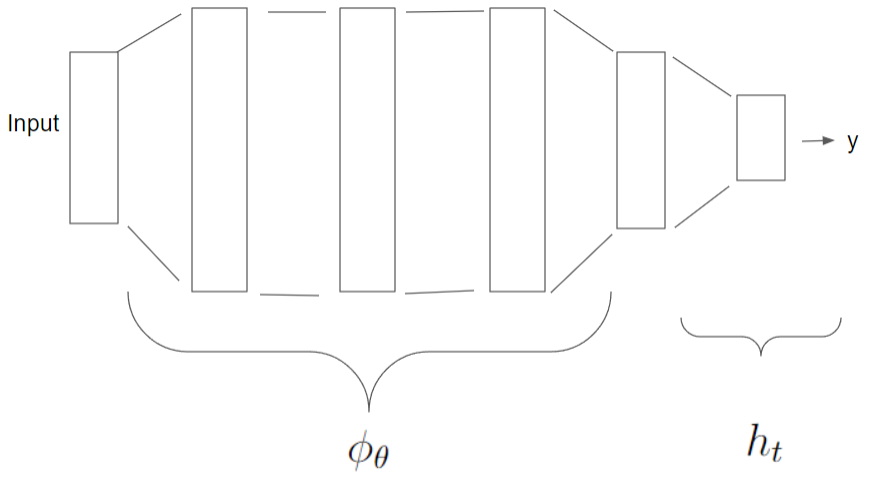

Figure 4

where ϕ_θ represents the shared weight matrix that's shared in all tasks 1 ... e and h_t is task-specific.

In this example, only 4 hidden layers are present but theoretically there could be arbitrarily many. In practice, ϕ_θ can be either fixed or fine-tuned. Fixing the ϕ_θ means that when applying the learned ϕ_θ to a new task, only h_t is learned. Fine-tuning means that both ϕ_θ and h_t are updated when applying the network on a new task.

Optimization

- From source datasets, the shared weights are derived by:

$$\hat{\phi} \leftarrow \min_{\phi} \min_{h_1,\ldots,h_t} \sum_{t=1}^{e} \hat{\mathcal{L}_t^S}(h_t \circ \phi)$$

  where

$$\hat{\mathcal{L}_t^S}(h_t \circ \phi) = \sum_{(x,y)\sim D_t^S} l(h_t(\phi(x)), y)$$

  In this optimization task, ϕ are the shared parameters, $h_1, \ldots, h_t$ are the weights in the last layer for different tasks. l is the loss function. D_t^S is the source dataset for the given task.

- From source datasets, assuming $\hat{\phi}$ is fixed:

$$\hat{h^T} \leftarrow \min_{h} \mathcal{L}^T(h \circ \phi)$$

- Then the output predictor is:

$$\hat{h} \circ \hat{\phi}$$

Intuitively, the reason why this can potentially make the learning procedure sample-efficient is that for the source task, we have abundant number of samples to learn the complicated part and if the learned representation is good, in the target task, only the last layer is learned which is also sample-efficient due to its simplicity.

One question to consider is whether the learned ϕ has a good expressive power. For example, if the learned representation function is just an identity mapping or a constant, then it would not help or even become no longer expressive when making a prediction. Hence, even though in this case, the learning procedure is sample-efficient. Bias is very high due to its weak expressive ability.

To conclude, the purpose of the learned shared representation is to make itself adapt to new tasks quickly which depends on whether f_θ^T can be expressed as $h_*^T \circ \hat{\phi}$, where we want to maintain a good expressive ability in ϕ.

3 Decomposition of generalization error in meta-learning

3.1 General form

If we see the output predictor as a single function:

$$\hat{f} := \hat{h} \circ \hat{\phi}$$

the classical bias-variance reduction, or in statistical learning we call bias as approximation error and variance as estimation error, can be decomposed as follows:

$$\begin{aligned}
&\mathcal{L}(\hat{f}) - \mathcal{L}_* \\
=&\mathcal{L}(\hat{f}) - \min_{f,cont.} \mathcal{L}(f) \\
=&\mathcal{L}(\hat{f}) - \min_{f\in\mathcal{F}} \mathcal{L}(f) + \min_{f\in\mathcal{F}} \mathcal{L}(f) - \min_{f,cont.} \mathcal{L}(f)
\end{aligned}$$

where $\mathcal{L}$ is the loss function and $\mathcal{F}$ is our choices of function class.

Now we can observe that $\mathcal{L}(\hat{f}) - \min_{f\in\mathcal{F}} \mathcal{L}(f)$ is the estimation error and $\min_{f\in\mathcal{F}} \mathcal{L}(f) - \min_{f,cont.} \mathcal{L}(f)$ is the approximation error.
More specifically,

- $\mathcal{L}(\hat{f}) - \min_{f\in\mathcal{F}} \mathcal{L}(f)$ is the estimation error because this error comes from using a finite number samples to estimate a function compared with knowing data concrete distribution to estimate.
- $\min_{f\in\mathcal{F}} \mathcal{L}(f) - \min_{f,cont.} \mathcal{L}(f)$ is the approximation error because it's about how expressive our choice of function class is. For example, choosing a wide and deep neural network would result in a lower approximation error compared to a quadratic function.

3.2 Fixing $\hat{\phi}$

Knowing that $\hat{f} := \hat{h} \circ \hat{\phi}$, we can add two extra terms in the decomposition obtained above, which is:

$$\mathcal{L}^T(\hat{h} \circ \hat{\phi}) - \min_{h \in \mathcal{H}} \mathcal{L}^T(h \circ \hat{\phi}) + \min_{h \in \mathcal{H}} \mathcal{L}^T(h \circ \hat{\phi}) - \min_{f \in \mathcal{F}} \mathcal{L}^T(f) + \min_{f \in \mathcal{F}} \mathcal{L}^T(f) - \mathcal{L}^T_*$$

where

- $\mathcal{L}^T(\hat{h} \circ \hat{\phi}) - \min_{h \in \mathcal{H}} \mathcal{L}^T(\hat{h} \circ \hat{\phi})$ is the adaptation error (III). Error caused by not learning the last layer optimally due to the limit of finite samples.
- $\min_{h \in \mathcal{H}} \mathcal{L}^T(\hat{h} \circ \hat{\phi}) - \min_{f \in \mathcal{F}} \mathcal{L}^T(f)$ is the approximation error (II) due to fixed representation. This error comes from fixing $\hat{\phi}$ as $\hat{\phi}$ can be freely selected in the second term but not in the first.
- $\min_{f \in \mathcal{F}} \mathcal{L}^T(f) - \mathcal{L}^T_*$ is the approximation error (I) due to architecture. For example, the error due to choosing the architecture to be ResNet, VGG...

4 Some upper bounds on the generalization error

4.1 Assumptions

We will now attempt to provide upper bounds for the generalization error in the context of multitask learning via shared representations [3, 6]

We again assume the availability of T source tasks, where each task $t \in T$ is described by n_s i.i.d samples which can be expressed as an input matrix $X_t \in \mathbb{R}^{n_s \times d}$ and a target vector $y_t \in \mathbb{R}^{n_s}$

The shared representation between the different tasks, which describes their common properties, is a function $\phi \in \Phi : \mathbb{R}^d \to Z$. Which as mentioned previously, maps the input to some feature space $z \subseteq \mathbb{R}^k$. To adapt to different tasks, this shared representation is composed with a task specific function $h \in \mathcal{H} : Z \to \mathbb{R}$ to produce the outputs.

To be able to derive acceptable upper bounds, we make two important assumptions

- Attainability of the Bayes optimal predictor.
- Task diversity

The first assumption guarantees the existence of a shared representation by which the task specific Bayes predictor f_i^* is attainable.

$$\exists \phi^* \in \Phi \text{ such that } f^* \in \mathcal{H} \circ \phi^*$$

where

$$\mathcal{H} \circ \phi^* = \{h \circ \phi^* | h \in \mathcal{H}\}$$

The second assumption can be intuitively described as follows, given a set of source tasks, the information acquired from that set should allow for new tasks to be solved. Specifically, in case of

linear functions h, let $h_1, \ldots, h_T$, be the source task specific prediction functions, then for any new test task, its prediction function h_{test} must be expressible as a linear combination of $h_1, \ldots, h_T$.

$$h_{\text{test}} \in \text{span}(h_1, \ldots, h_T)$$

4.2 Exactly shared structure

For the usual formulation of an exact shared structure across different tasks the following bounds can be attained

$$\mathcal{E} = \underbrace{\mathcal{L}^T(\hat{h} \circ \hat{\phi}) - \min_{h \in \mathcal{H}} \mathcal{L}^T(h \circ \hat{\phi})}_{\text{Adaptation error}} + \underbrace{\min_{h \in \mathcal{H}} \mathcal{L}^T(h \circ \hat{\phi}) - \min_{f \in \mathcal{F}} \mathcal{L}^T(f)}_{\text{Representation Error}} + \underbrace{\min_{f \in \mathcal{F}} \mathcal{L}^T(f) - \mathcal{L}^T_*}_{\text{Approximation Error}}$$

From our first assumption we know that there is no approximation error, and depending on the family of representation functions Φ and a complexity measure $\mathcal{C}$ we have the following

Linear representations and linear predictor

$$\mathcal{E} \leq \frac{dk}{n_s T} + \frac{k}{n_t}$$

Non-Linear representations and linear predictor

$$\mathcal{E} \leq \frac{\mathcal{C}(\Phi)}{n_s T} + \frac{k}{n_t}$$

Non-Linear representations and non-linear predictor

$$\mathcal{E} \leq \frac{\mathcal{C}(\Phi)}{n_s T} + \frac{\mathcal{C}(\mathcal{H})}{n_t}$$

which are all acceptable bounds with a fast convergence rate.

4.3 Fine tuning the shared structure

One issue of the above approach is the assumption of the existence of an exact shared structure among different tasks. This problem becomes apparent in the presence of misspecifications

$$y_i = h_i \circ \phi + w_n$$

where w_n can be noise. One straightforward approach [2] is to also allow for adaptation on the representation. Therefore, on a new task, instead of solving

$$\min_h \frac{1}{n_t} \sum_{i=1}^{n_t} l(y_i, h^T \phi_{\theta^*}(x_i))$$

as before, we solve for the training

$$\min_{\theta_0} \min_{\substack{\theta_t, h_t \\ \|\theta_t - \theta_0\| \leq \gamma}} \frac{1}{n_s} \frac{1}{T} \sum_{i=1}^{n_s} l(y_{i,t}, h_t^T \phi_{\theta_t}(x_i))$$

and for the testing step

$$\min_{\substack{\theta \\ \|\theta - \theta_0\| \leq \gamma}} \min_{h} \frac{1}{n_t} \sum_{i=1}^{n_t} l(y_i, h^T \phi_{\theta}(x_i))$$

This approach has obvious similarities with MAML since all model parameters are adaptable, however unlike MAML, the degree at which the model can adapt is constrained since the representation component can only adapt inside a γ-ball centered at θ_0. Another difference is that no train-validation splits are used, as is widespread in MAML.

In addition, regarding the generalization error upper bound in this case, if we refer to the previous upper bound as $\mathcal{E}_{\text{exactly-shared}}$ then

$$\mathcal{E} \leq \mathcal{E}_{\text{exactly-shared}} + \frac{\gamma}{\sqrt{n_T}}$$

Intuition behind the disadvantages of having a shared representation

First assume that the representation is linear, thus it can be described by a matrix $\Phi \in \mathbb{R}^{k \times d}$. If the representation is shared, then using the usual notation, the optimization problem is

$$\hat{\Phi} = \arg\min_{\Phi} \min_{h_1, \ldots, h_T} \frac{1}{2n_s T} \sum_{i=1}^{T} \|y_t - X_t \Phi h_t\|_2^2 \xrightarrow{n_s \to \infty} \frac{1}{2T} \sum_{t \in [T]} \|\Phi^* h_t^* - \Phi h_t\|_{\Sigma}^2$$

the obvious problem in this case, which is not present when finetuning the representation, is that the representation is based on the prediction space norm instead of the parameter space norm, resulting in potentially completely failing to find a useful Φ. Fine tuning, thus allows to properly handle misspecifications and to effectively meta learn.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. *Representation Learning: A Review and New Perspectives*. 2014. arXiv: `1206.5538 [cs.LG]`.

[2] Kurtland Chua, Qi Lei, and Jason D. Lee. *How Fine-Tuning Allows for Effective Meta-Learning*. 2021. arXiv: `2105.02221 [cs.LG]`.

[3] Simon S. Du et al. *Few-Shot Learning via Learning the Representation, Provably*. 2021. arXiv: `2002.09434 [cs.LG]`.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1126–1135. URL: `https://proceedings.mlr.press/v70/finn17a.html`.

[5] Jake Snell, Kevin Swersky, and Richard S. Zemel. *Prototypical Networks for Few-shot Learning*. 2017. arXiv: 1703.05175 [cs.LG].

[6] Nilesh Tripuraneni, Michael I. Jordan, and Chi Jin. *On the Theory of Transfer Learning: The Importance of Task Diversity*. 2020. arXiv: 2006.11650 [cs.LG].