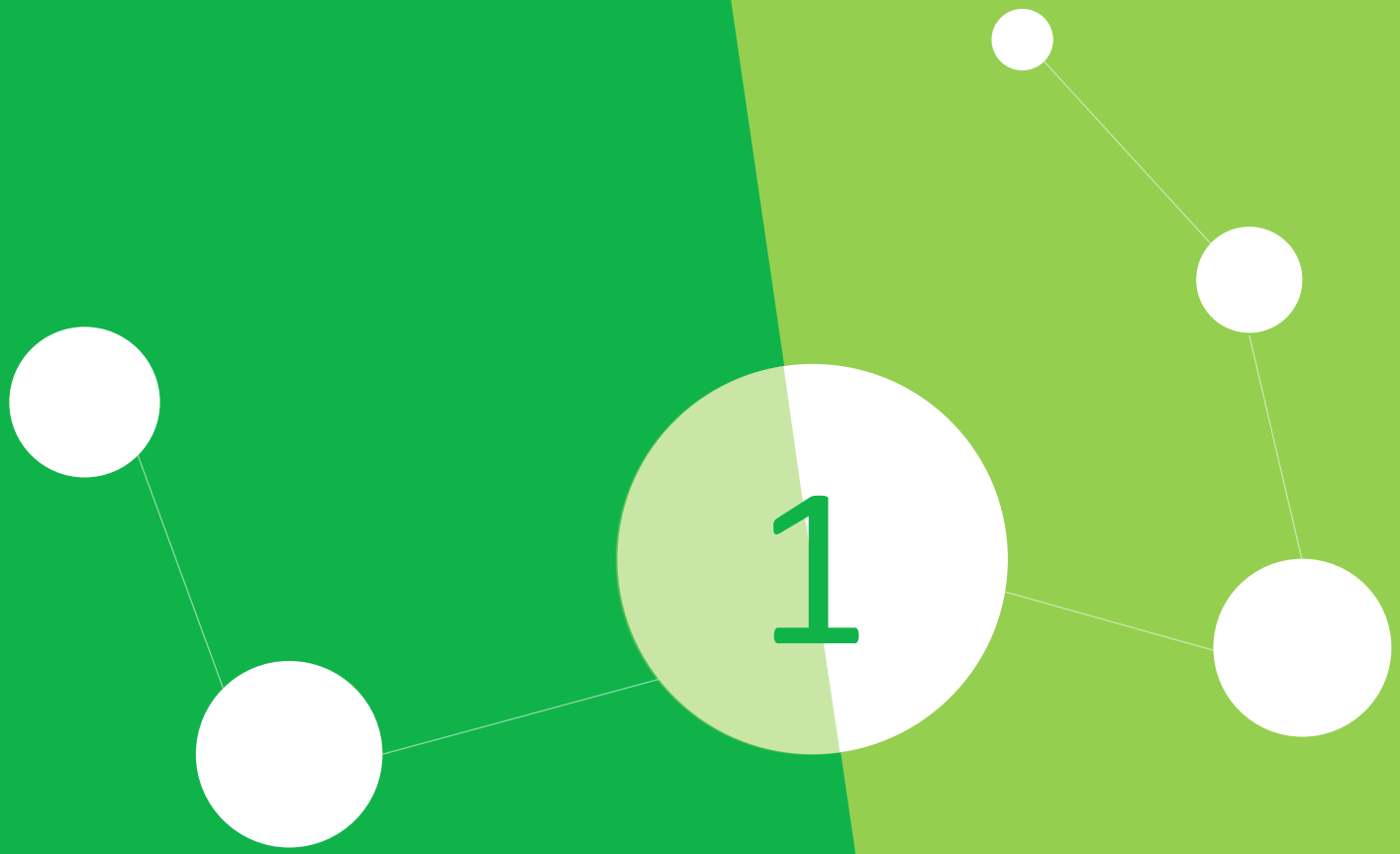
Meta Learning

With DP

Ning Wang

CONTENT


1

Introduction

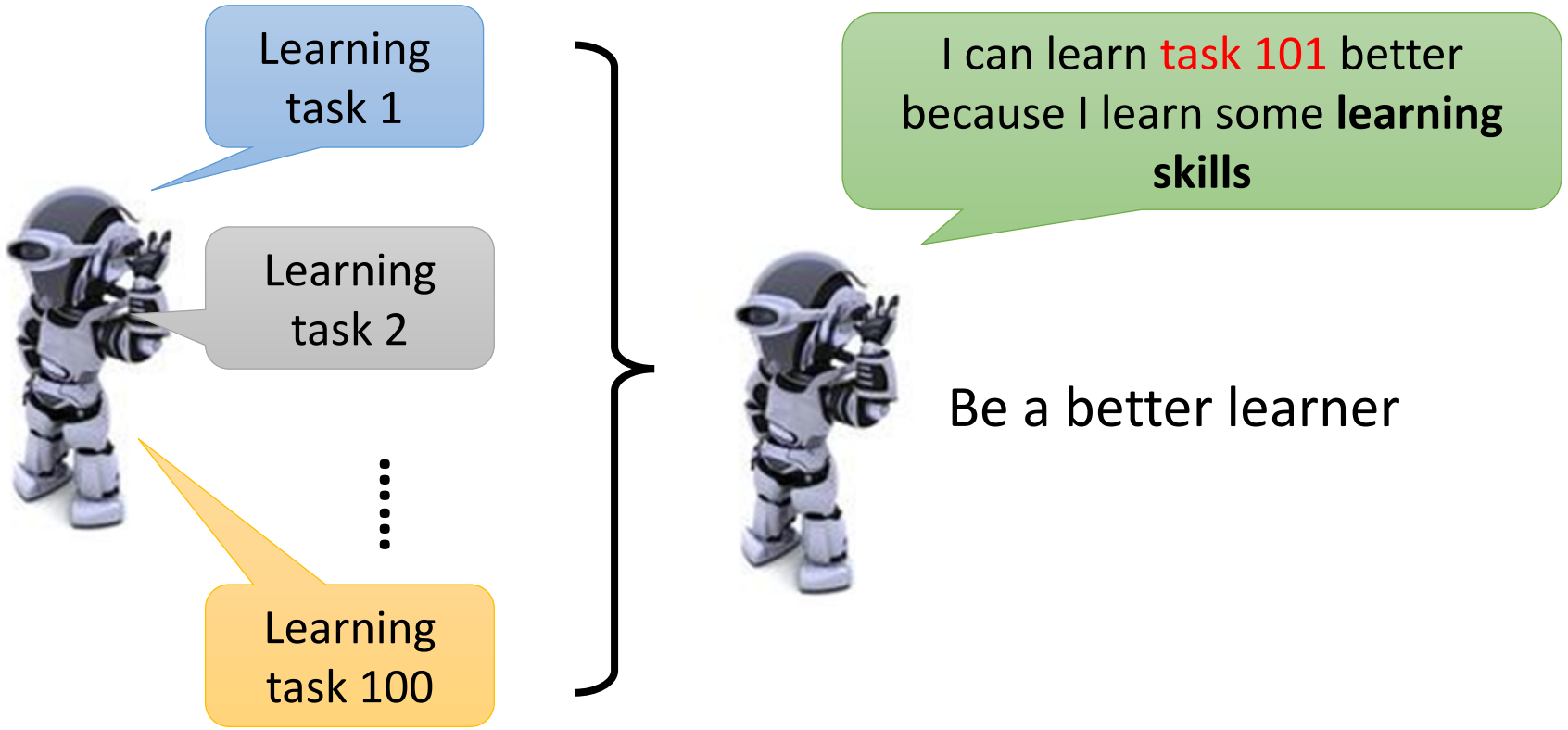
Task 1: speech recognition

Task 2: image recognition

⋮

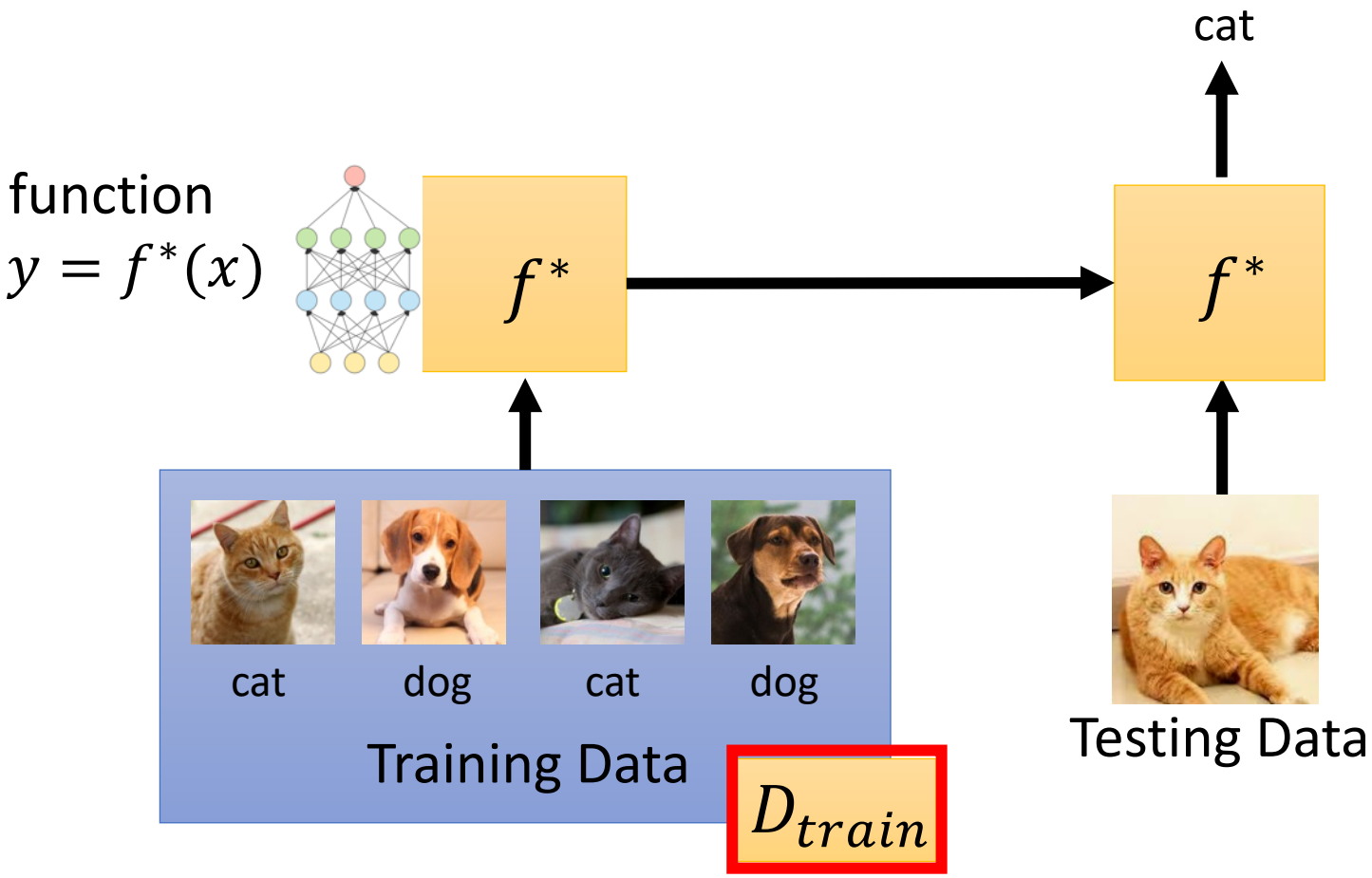
Task 101: text classification

- Meta learning = Learn to learn



Meta: How to learn a new model

Machine Learning



{The class of the test data} == {the classes of the training data}

Meta Learning

$$f^* = F(D_{train})$$

Can machine find F from data?

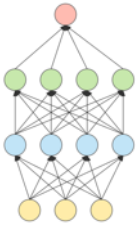
Designed by Developers

It is also a function.

Learning Algorithm

F

f^* function



F

Task n

...

Task 2

Task 1

Training Data

D_{train}

New task

cat dog cat dog

Testing Data

Meta Learning

Machine Learning ≈ find a model f

$f($  $) =$ "Cat"

Meta Learning

≈find a learning algorithm F, then be capable to get different f^* for different task

$F($ 

$) = f^*$

The literal meaning of meta learning

- **The interpretation of Meta from EuDic English Dictionary**

  Meta (from the Greek preposition and prefix *meta-* (μετά-) meaning "after", or "beyond") is a prefix used in English to indicate a concept which is an abstraction from another concept, used to complete or add to the latter.

- **The interpretation of Meta from Wikipedia**

Meta (from the Greek *meta-* μετά- meaning "after" or "beyond") is a prefix meaning more comprehensive or transcending.[1]

Meta learning == Learning to learn
Meta-learning, also known as “learning to learn”, intends to design models that can **learn new skills** or **adapt to new environments** rapidly with a **few training examples**.

Meta Learning

Every task is a traditional machine learning problem.

Task 1: Model f_1

…

Task n: Model f_n

…

Task N: Model f_N

Base model

Meta Learning goal: Not the model set $\{f_1, \ldots f_n, \ldots, f_N\}$

But a learning algorithm F $(f_n = F(D_n))$

Meta model

With F, we can get f^* for different tasks

Meta Learning

When do we use Meta Learning

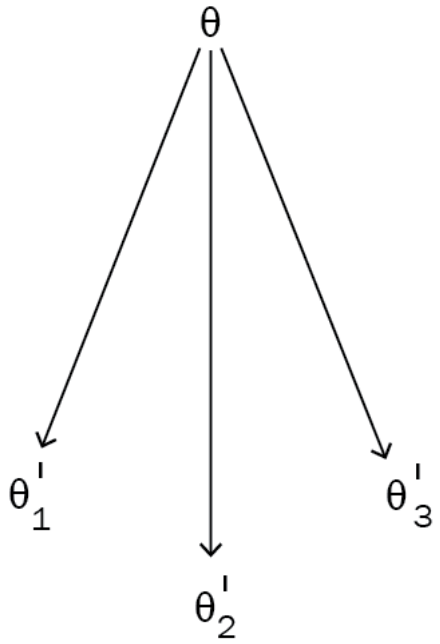
Training data set is too small (1-5 samples per concept) to train a good enough model for a task.

Property of Meta Learning

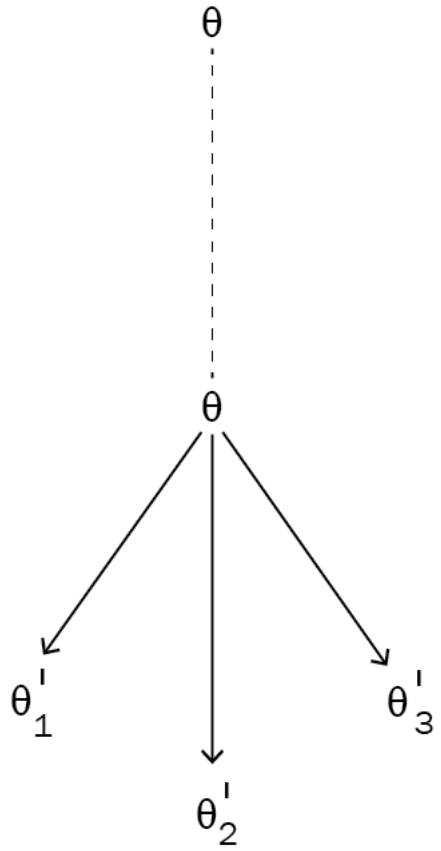
- Meta model F can learn across-tasks knowledge;
- Meta model F can acquire task-specific knowledge from new tasks using only a few samples.

Meta Learning

Essentially, meta learning is to find a good **initialization** that can adapt to different tasks with a few data samples.



Learn from scratch



Meta learning

PART TWO

~~Machine~~ Learning is Simple

Meta

01 Define a ~~function~~

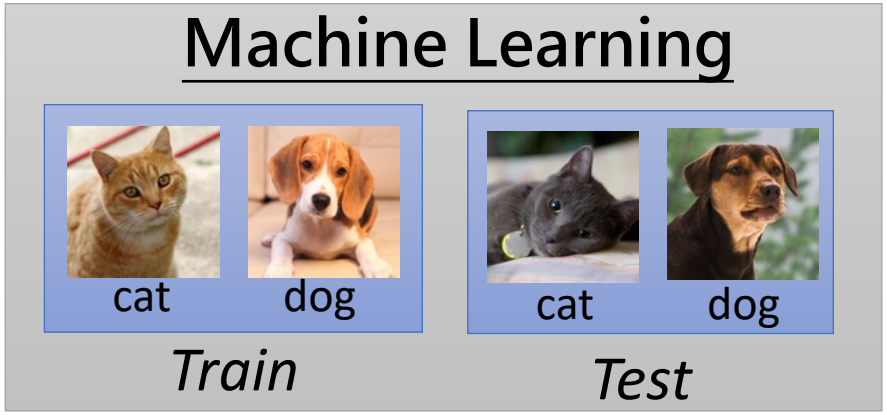
02 Goodness of the ~~function~~

03 Choose the best ~~function~~

Function f ➡ Learning algorithm F

Meta Learning

Widely considered in **few-shot learning**



Training Tasks

Task 1

Support set

Train

cat

dog

Query set

Test

cat

dog

Task 2

Train

apple

orange

Test

apple

orange

Sometimes you need validation tasks

Testing Tasks

Train

bike

car

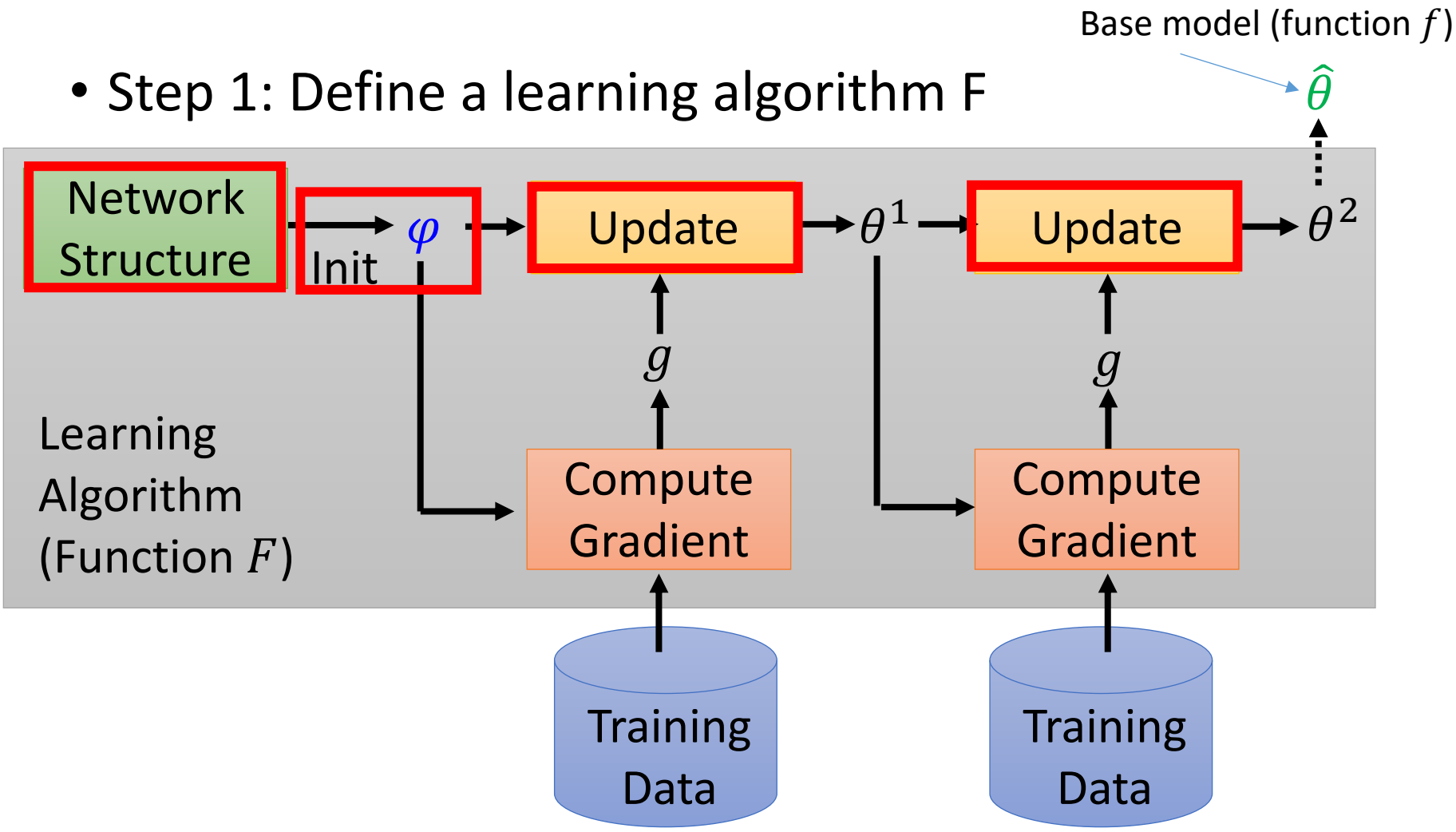
Test

bike

car

Meta Learning

- Step 1: Define a learning algorithm F



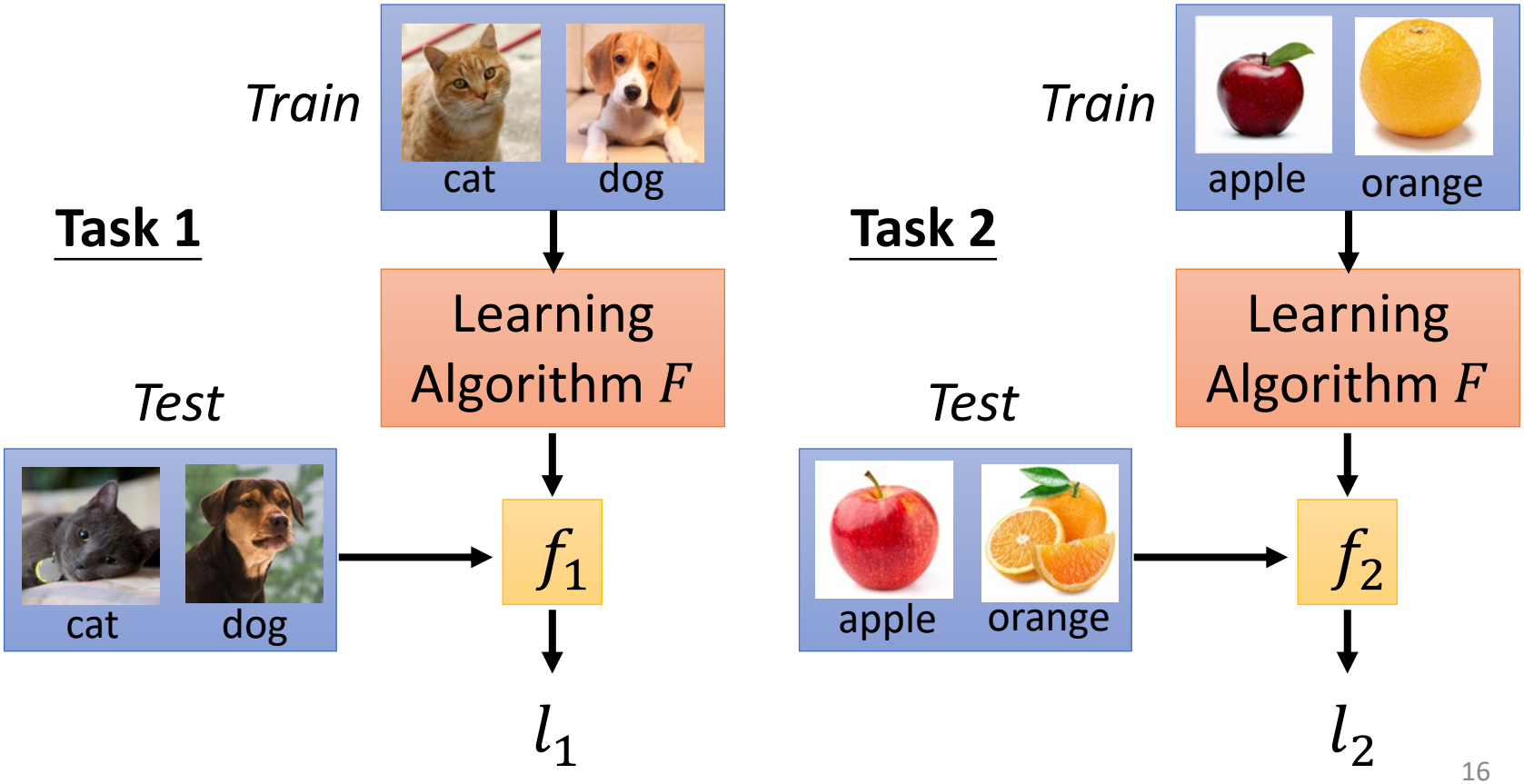
Meta Learning

$$L(F) = \sum_{n=1}^{N} l_n$$

N: N tasks

l_n: Testing loss for task n after training

- Step 2: Defining the goodness of a function F



Meta Learning

- Step2: Defining the goodness of a function F

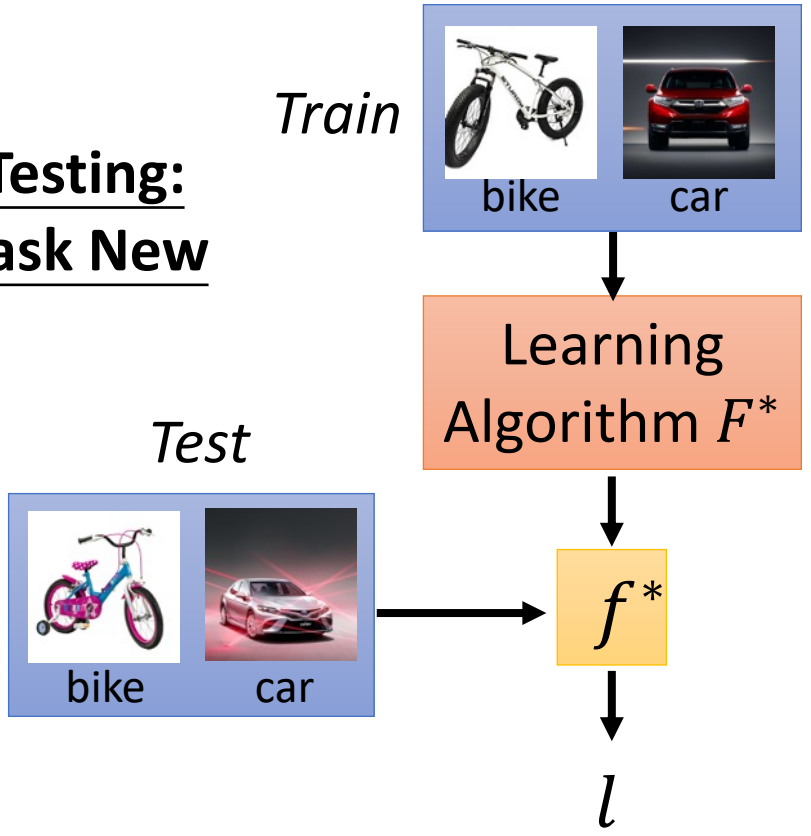
$$L(F) = \sum_{n=1}^{N} l^n$$

- Step3: Find the best function F^*

$$F^* = arg \min_F L(F)$$

Testing:

Task New



Meta Learning

- **MAML**
  - Chelsea Finn, Pieter Abbeel, and Sergey Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”, ICML, 2017

MAML

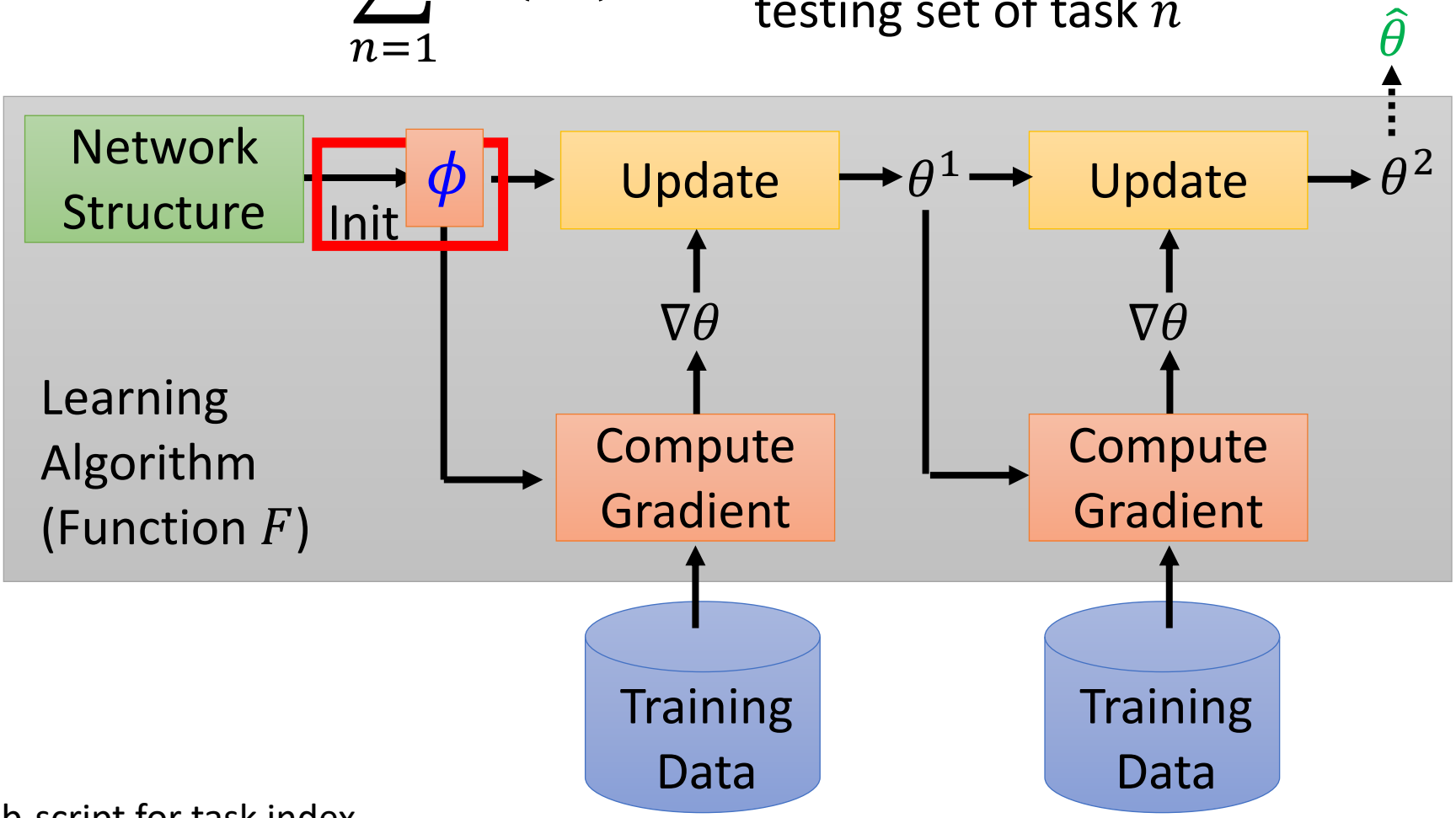
$\hat{\theta}_n$: model learned from task n

$\hat{\theta}_n$ depends on ϕ

Loss Function:

$$L(\phi) = \sum_{n=1}^{N} l_n(\hat{\theta}_n)$$

$l_n(\hat{\theta}_n)$: loss of task n on the testing set of task n



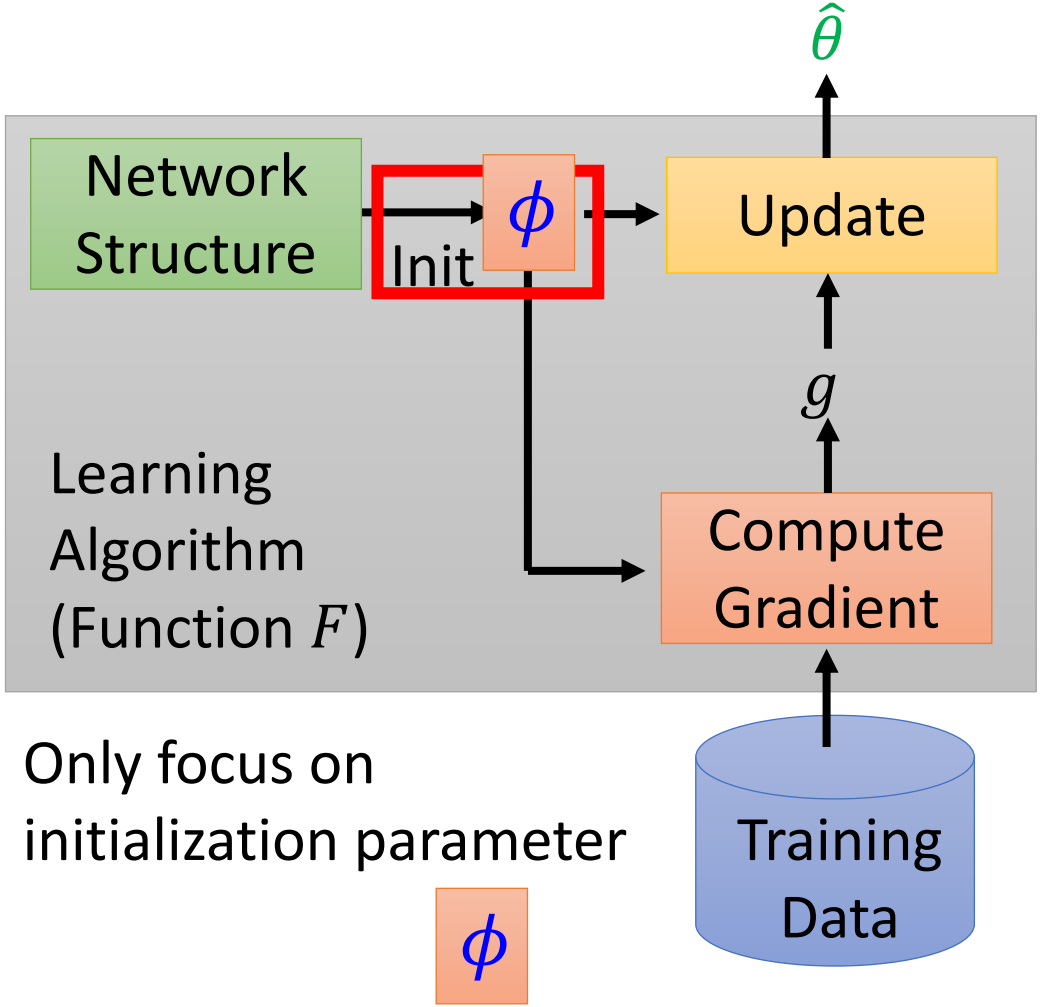
Sub-script for task index

Super-script for index in different time steps

MAML

Why one step?

- Fast
- Good to truly train a model with one step.
- Few-shot learning has limited data.(Over-fitting)



$$L(\phi) = \sum_{n=1}^{N} l^n(\hat{\theta}^n)$$

$$\phi \leftarrow \phi - \eta \nabla_\phi L(\phi)$$

Considering one-step training:

$$\hat{\theta} = \phi - \varepsilon \nabla_\phi l(\phi)$$

MAML

$$\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$$

$$L(\phi) = \sum_{n=1}^{N} l_n(\hat{\theta}_n)$$

$$\hat{\theta} = \phi - \varepsilon \nabla_{\phi} l(\phi)$$

MAML

- Two loops

  Inner loop, for Base model.

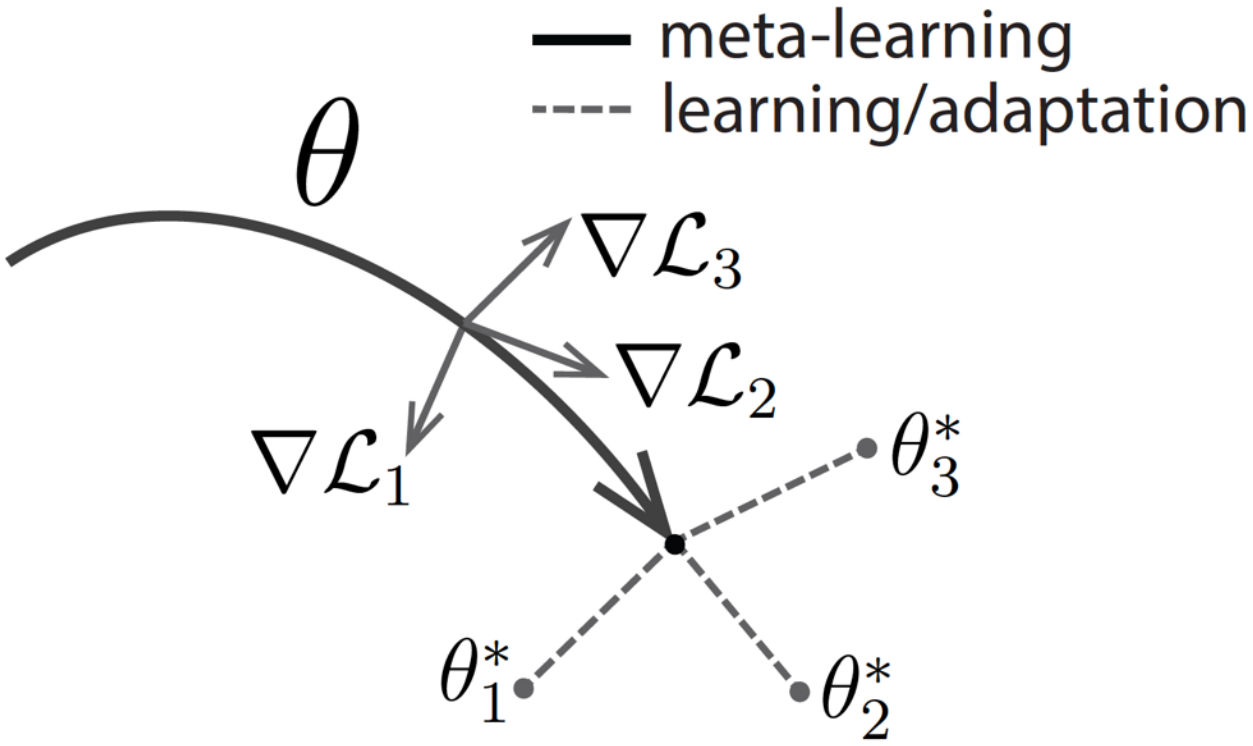
θ is weights of Meta Model;
θ_i is weights of Base Model i.

$$\theta_i = \theta - \alpha \nabla_\theta \frac{1}{M_{\text{train}}^i} \sum_{(x,y) \in \mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f_\theta(x), y) \; .$$

Outer loop. For Meta model.

$$\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i \in \mathbf{T}} \frac{1}{M_{\text{test}}^i} \sum_{(x,y) \in \mathcal{D}_i^{\text{test}}} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i}(x), y) \; .$$

MAML



Find θ which can easily adapt to a good model for different tasks.

MAML

$\hat{\theta}_n$: model learned from task n

$\hat{\theta}_n$ depends on ϕ

Loss Function:

$$L(\phi) = \sum_{n=1}^{N} l_n(\hat{\theta}_n)$$

$l_n(\hat{\theta}_n)$: loss of task n on the testing set of task n

How to minimize $L(\phi)$? Gradient Descent

$$\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$$

Model Pre-training

Widely used in transfer learning

Loss Function:

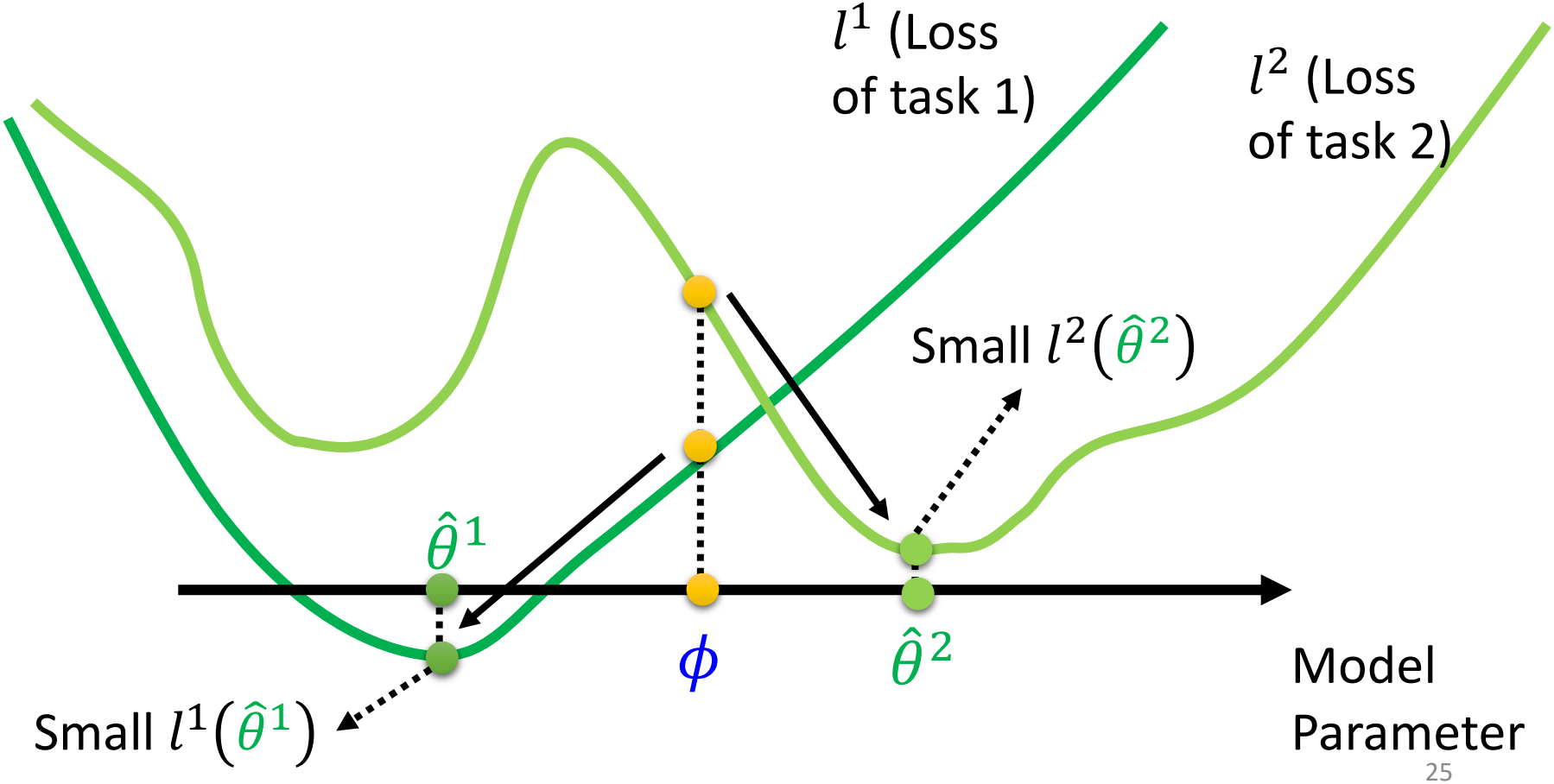
$$L(\phi) = \sum_{n=1}^{N} l^n(\phi)$$

MAML

$$L(\phi) = \sum_{n=1}^{N} l^n(\hat{\theta}^n)$$

No: goodness of ϕ in training task

Yes: **Goodness of ϕ** in finding good $\hat{\theta}^n$

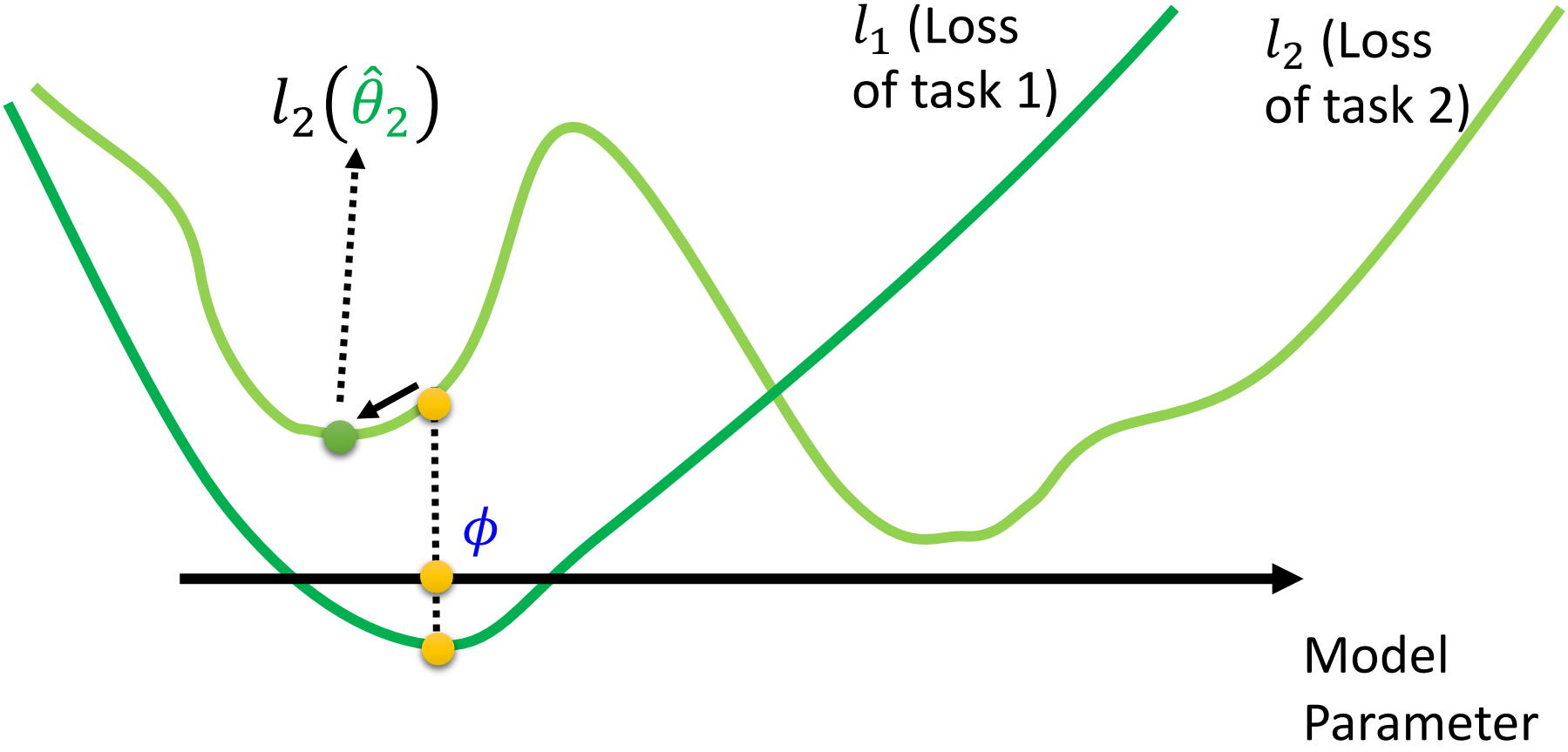


Model Pre-training

$$L(\phi) = \sum_{n=1}^{N} l_n(\phi)$$

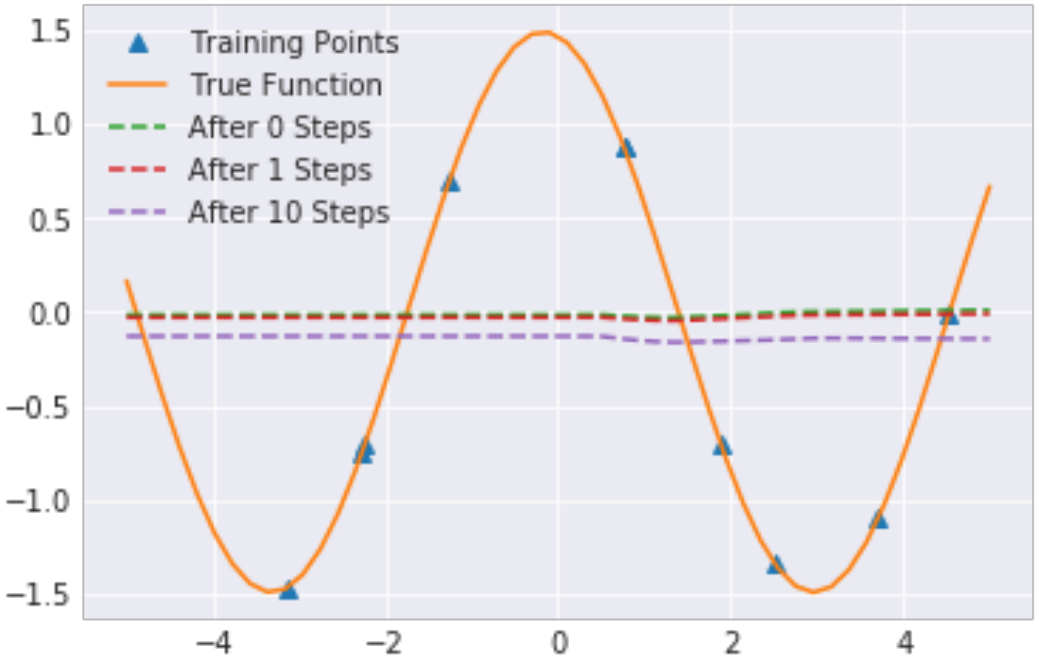
The optim ϕ

Can not guarantee good $\hat{\theta}_n$

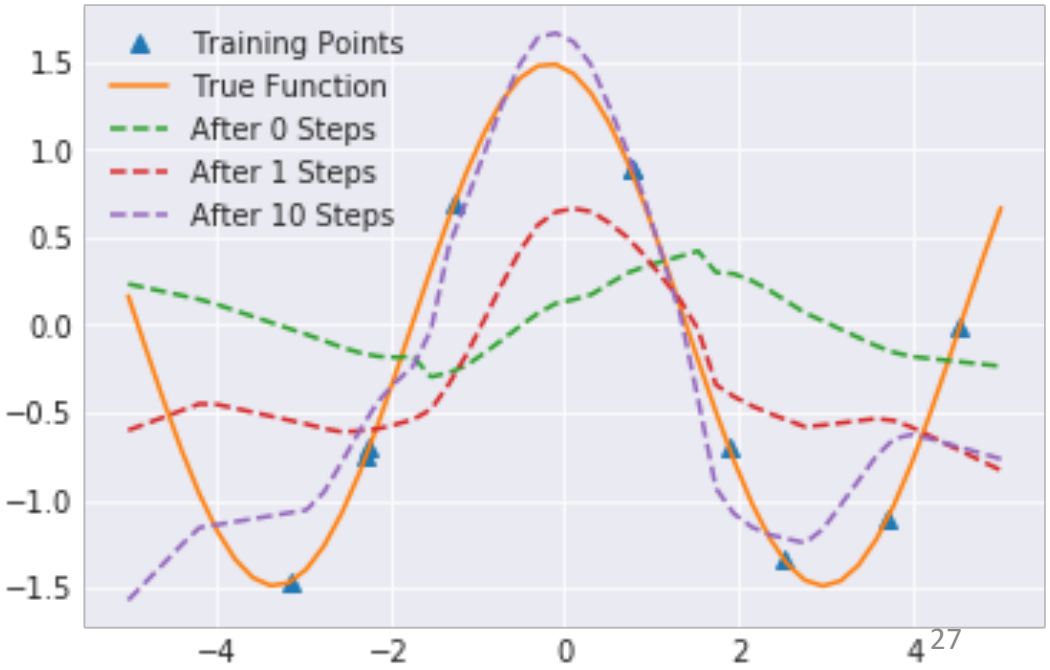


Performance

Model Pre-training



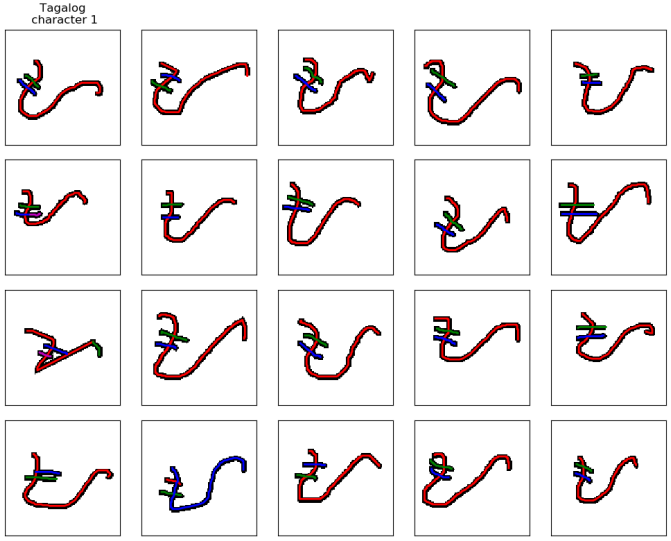
MAML



Omniglot

https://github.com/brendenlake/omniglot

- 1623 characters
- Each has 20 examples



Omniglot – Few-shot Classification

- N-ways K-shot classification: In each training and test tasks, there are N classes, each has K examples.

20 ways
1 shot

Each character represents a class

Testing set (Query set)

Training set (Support set)

- Split your characters into training and testing characters
  - Sample N training characters, sample K examples from each sampled characters → one **training task**
  - Sample N testing characters, sample K examples from each sampled characters → one **testing task**

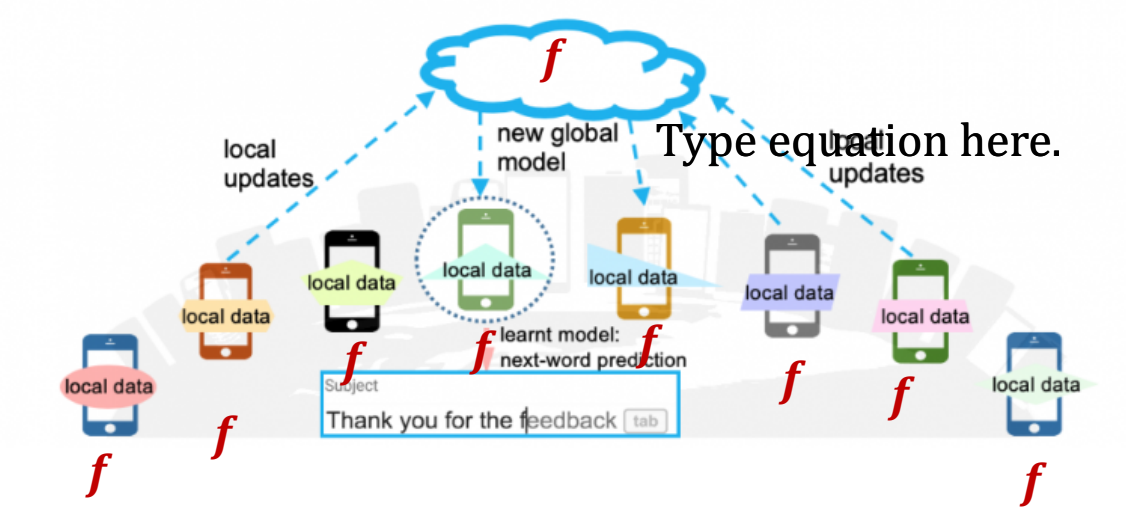
Omniglot

Omniglot (Lake et al., 2011)	5-way Accuracy		20-way Accuracy	
	1-shot	5-shot	1-shot	5-shot
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MAML, no conv (ours)	**89.7 ± 1.1%**	**97.5 ± 0.6%**	–	–
Siamese nets (Koch, 2015)	97.3%	98.4%	88.2%	97.0%
matching nets (Vinyals et al., 2016)	98.1%	98.9%	93.8%	98.5%
neural statistician (Edwards & Storkey, 2017)	98.1%	99.5%	93.2%	98.1%
memory mod. (Kaiser et al., 2017)	98.4%	99.6%	95.0%	98.6%
MAML (ours)	**98.7 ± 0.4%**	**99.9 ± 0.1%**	**95.8 ± 0.3%**	**98.9 ± 0.2%**


3

DP-Federated learning

- Recall **federated learning**

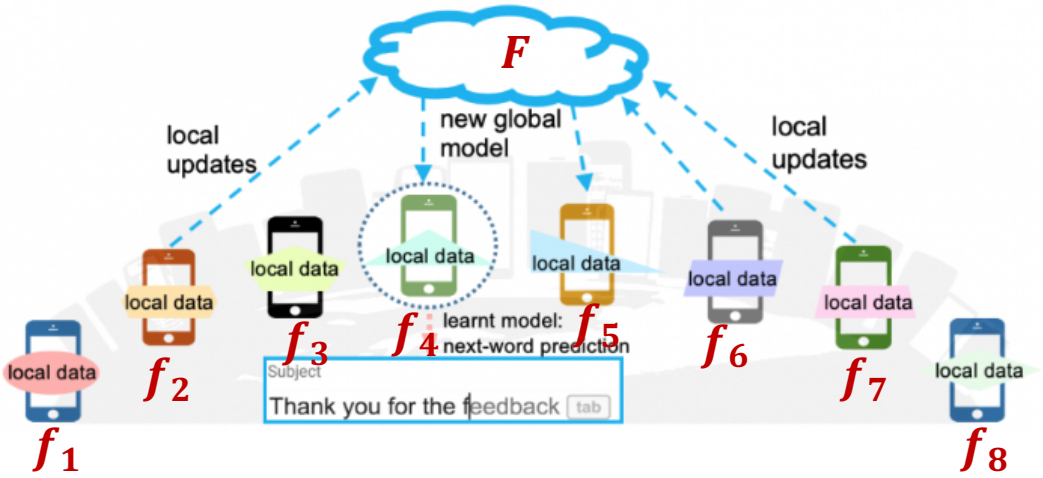


- **Goal:** train a central model f. Each client has the same model.
- Why **DP**?

  Users privacy consideration about whether to participate FL.

Our Target Work: DP Meta learning

- **Meta learning** scenario: Each user has his or her own task.



- **Goal**: train a learning algorithm F which can easily adapt to different f for different tasks.

- **DP:** Participants privacy consideration.

Our Target Work: DP Meta learning

- **Different level of privacy**

  - **Task level** (client level): whether a client participate the meta-training process cannot be inferred.

    Where: loss on the target set or the gradients of the meta model (sensitivity and adding noise)

  - **Example Level** (each data instance): whether a data sample is included in the training process can not be inferred.

    where: gradients of the base model (sensitivity and adding noise)

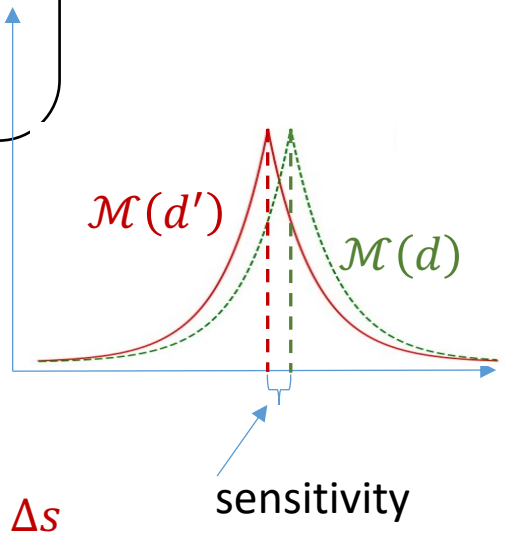
Recall DP

ε −differential privacy : A random mechanism $\mathcal{M}: \mathcal{D} \to \mathcal{R}$ satisfies ε −differential privacy if for any two adjacent inputs ($d, d' \in \mathcal{D}$) and for any subset of outputs $S \in \mathcal{R}$ it holds that

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S]$$

$$\mathcal{M}(d) = f(d) + noise$$

$$\text{sensitivity} \Delta s = \max_{d,d' \in \mathbb{N}^{|x|}} \|f(d) - f(d')\|_1$$



Three factor in DP: Noise σ, Privacy budget ϵ, sensitivity Δs

DP in Machine Learning

- **Perturbation Procedure**
  1. Choosing a subset of data records(Build the Database, adjacent Database).
  2. Clipping x (usually gradients)with a norm c. (build the sensitivity).
  3. Add noise σ to x.

- **Privacy Analysis**
  1. Keep a record of $z = \frac{\sigma}{\Delta s}$;
  2. Calculate total privacy spend $\varepsilon = f(z_1, z_2, \dots)$ with composition theory.

Work Plan

1. Implement different level DP in meta learning.
   - Client level
   - Sample level

2. Analyze the trade-off between utility and privacy.

THANK YOU