<basic setup>

single task $D=\{(x,y)_k\}$, $\min_\theta \mathcal{L}(\theta, D)$

$\mathcal{L}(\theta, D) = -\mathbb{E}_{(x,y)\sim D}[\log f_\theta(y|x)]$

task $\mathcal{T}_i \triangleq \{p_i(x), p_i(y|x), \mathcal{L}_i\}$

→ data generating distributions

D_i^{tr}의 shorthand로 D_i 사용.

- Multi-task classification : $\mathcal{L}_i$가 모든 task에서 동일.
- Multi-label learning : $\mathcal{L}_i$와 $p_i(x)$가 모든 task에서 동일
  (distribution over the inputs)

when might $\mathcal{L}_i$ vary across tasks?

- 이산, 연속 label이 섞여 있는 경우
- 한 task를 다른 task보다 중요하게 여기는 경우.

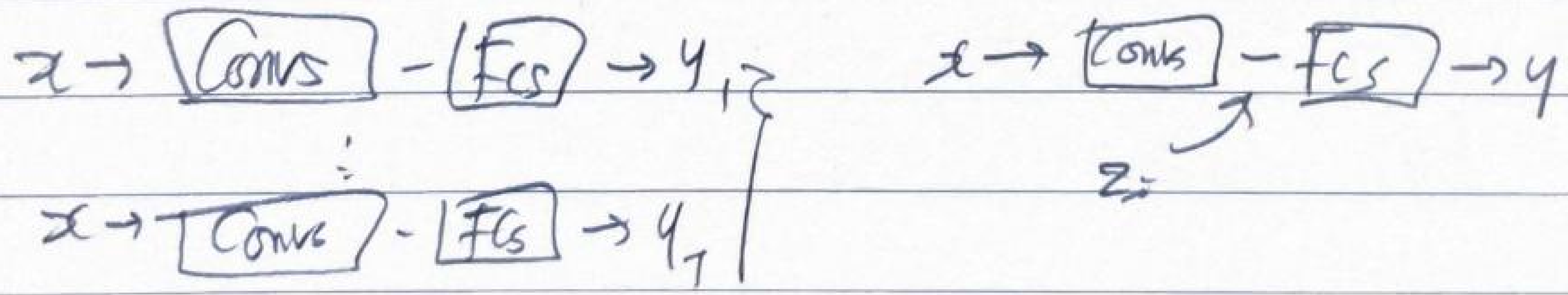
model $f_\theta(y|x, z_i)$ (task descriptor)

(z_i) → model design 과도 관계가 있음! ex> 이 정보를 파악해서 모델에 어떻게 넣어줄 것이냐?

① How should we design the model?

② How should we optimize the objective?

task별 일정 부분은 공유해야 ~~[illegible]~~ 가정이 성립된다고 저번 시간에 언급.



Parameter share를 어디로 할 것인가?!

⇒ design 문제.

$$\min_{\theta^{sh},\theta^1,\dots,\theta^T} \sum_{i=1}^{T} \mathcal{L}_i(\{\theta^{sh},\theta_i^s\}, \mathcal{D}_i) \quad \text{~} \square$$

How and Where to share parameter?

∴ Conditioning!
- Concatenate
- Additive (Conditional bias)

  } These are actually same thing!
  ↳ always same??
- Multi-Head architecture
- Multiplicative (Conditional scaling) ⤳ more expressive ~~recall~~

서로 다른 task에서 무엇을 공유하고 있는가?

(attention mechanism! → 보통 dot product! (교수님 생각)
  in this case, dimension summing이 아닌 element-wise multiplication

Neural Network is ~~UFA~~ Universal function Approximator)

π... these design decision → 직관과 지식 필요, 문제 의존적.

then, ②. Optimizing the objective

$$\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i)$$

$$\left(\hat{\mathcal{L}}(\theta, \mathcal{B}) = \sum_{\mathcal{T}_k \in \mathcal{B}} \mathcal{L}_k(\theta, \mathcal{D}_k^b) \right)$$

basic version

negative transfer ⇒ 공유하는 부분을 줄여라!! (∵ 에러 문제, 표현 용량 문제)

$$\square + \sum_{t'=1}^{T} \|\theta^t - \theta^{t'}\|$$

overfitting ⇒ 공유하는 부분을 늘려라!!

+ objective form을 바꾸는 경우도 존재!
(ex) weighted)

< Multi-task Learning → Solve multiple tasks $\mathcal{T}_1, \dots \mathcal{T}_T$ at once
Transfer Learning $\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i)$.

→ ($\mathcal{T}_a$로 부터 학습한 지식을 전이시켜)
$\mathcal{T}_a$를 푼 다음 $\mathcal{T}_b$를 푸는 것.
$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr})$. Fine-tuning

<u>Meta-Learning Basic</u>

two ways < Mechanistic View : easy to implement
✓ Probabilistic View : high-level

problem definition!

supervised learning $\arg\max_{\phi} \log p(\phi | \mathcal{D})$: maximum likelihood
(→ ϕ: model parameter, → $\mathcal{D}$: training data)

$$= \arg\max_{\phi} \{ \log p(\mathcal{D} | \phi) + \log p(\phi) \}$$

$$= \arg\max_{\phi} \{ \sum_i \log p(y_i | x_i, \phi) + \log p(\phi) \}$$

required labeled data!

Can we incorporate additional data?

$$\arg\max_{\phi} \log p(\phi | \mathcal{D}, \mathcal{D}_{meta\text{-}data})$$

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

$$\mathcal{D}_{meta\text{-}train} = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$$

$$\mathcal{D}_i = \{(x_j^i, y_j^i)\}_{j=1}^{K}$$ ⇒ 추가적인 data를 주는 것!!
(뭐여, 진짜 meta 여)

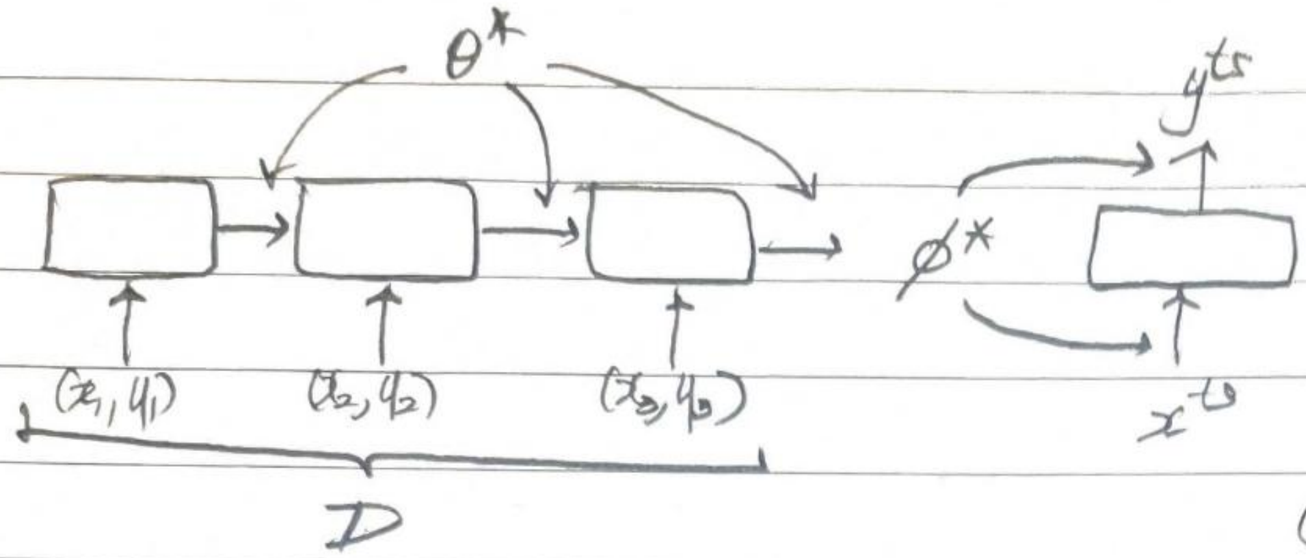
what if we don't to keep $\mathcal{D}_{meta\text{-}train}$ around forever?

$$\arg\max_{\phi} \log p(\phi | \mathcal{D}, \mathcal{D}_{meta\text{-}train}) \approx \arg\max_{\phi} \log p(\phi | \mathcal{D}, \theta^*)$$

where $\theta^* = \arg\max_{\theta} \log p(\theta | \mathcal{D}_{meta\text{-}train})$ → meta-learning

$$\phi^* = \arg\max_{\phi} \log p(\phi | \mathcal{D}, \theta^*)$$
→ adaptation

대충읽자

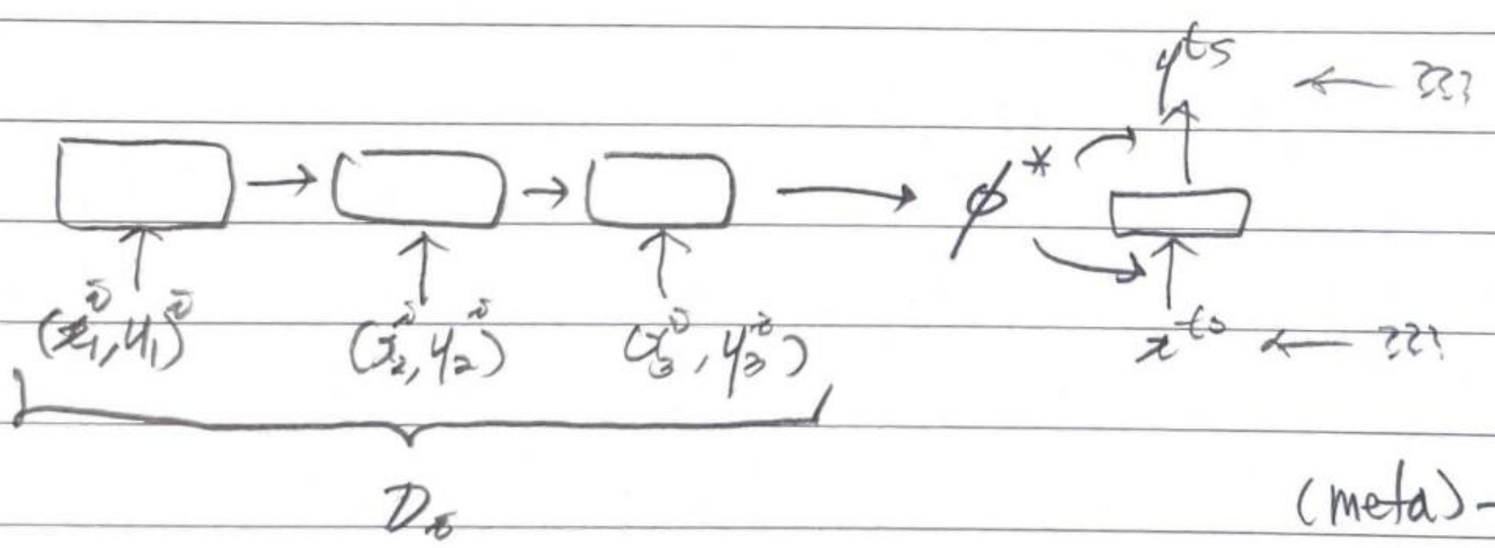


(meta) test-time

meta learning : $\theta^* = \arg\max_{\theta} \log p(\phi \mid D_{meta\text{-}train})$

adaptation : $\phi^* = \arg\max_{\phi} \log p(\phi \mid D_i, \theta^*)$

* our training procedure is based on a simple machine learning principle
: test and train conditions must match.



(meta) training time.

Reverse a test set for each task!!

D_1^{tr}, D_1^{ts}

* the Complete meta-learning optimization.

$$\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p(\theta | D_i)$$

$$\phi_i = \theta \ (\text{i.e., } f_\theta(D_i) = \theta)$$

$\Downarrow$ multi-task learning

meta learning : $\theta^* = \arg\max_\theta \log p(\theta | D_{\text{meta-train}})$

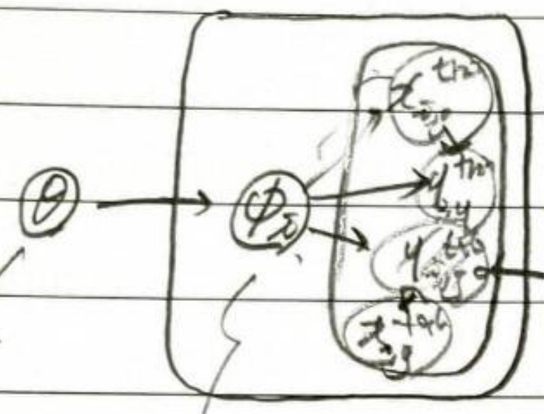
adaptation : $\phi^* = \arg\max_\phi \log p(\phi | D^{tr}, \theta^*)$

$\Updownarrow$

$\phi^* = f_{\theta^*}(D^{tr})$

learn θ s.t. $\phi = f_{\theta^*}(D_i^{tr})$ is good for D_i^{ts}.

$$\theta^* = \max_\theta \sum_{i=1}^{n} \log p(\phi_i | D_i^{ts}) \quad \text{where} \quad \phi_i = f_\theta(D_i^{tr})$$



unobserved at meta test time.

task-specific

task 마다 다른 Information !!

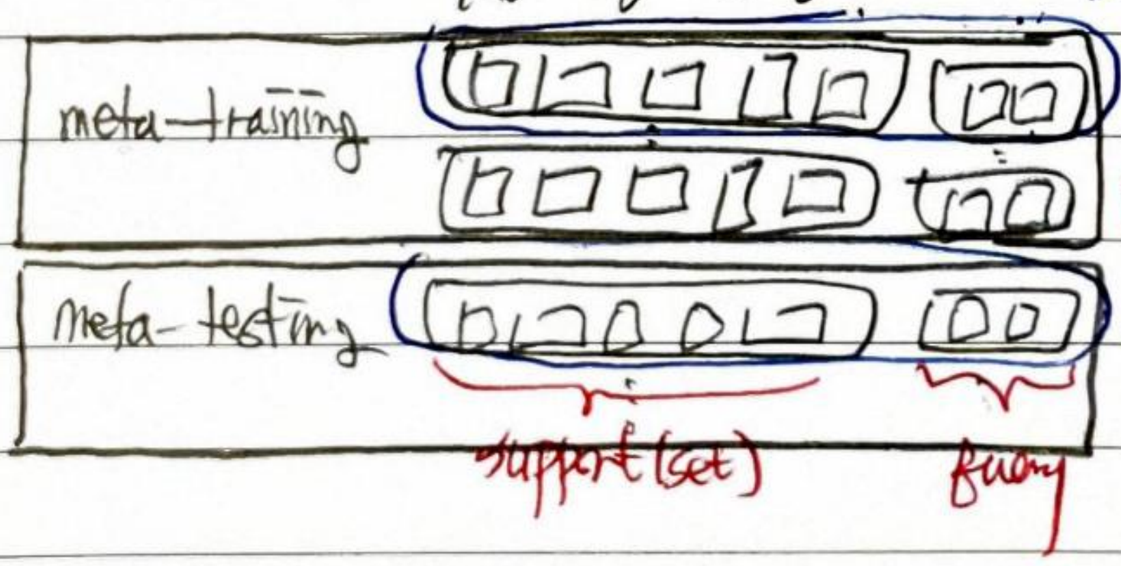
-X- hyperparameter optimization
Auto-ML
⇒ Can be cast as meta-learning !!

* Some meta-learning terminology

learn θ s.t. $\phi_i = f_\theta(D_i^{tr})$ is good D_i^{ts}

$$\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p(\phi_i | D_i^{ts})$$

where $\phi_i = f_\theta(D_i^{tr})$



$$D_{\text{meta-train}} = \{(D_1^{tr}, D_1^{ts}), \ldots, (D_n^{tr}, D_n^{ts})\}$$

$$\mathcal{T}_i = \left\{ D_i^{tr} = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\} \right.$$
$$\left. D_i^{ts} = \{(x_1^i, y_1^i), \ldots, (x_l^i, y_l^i)\} \right.$$

→ shot (i.e., k-shot, l-shot)

대충읽자