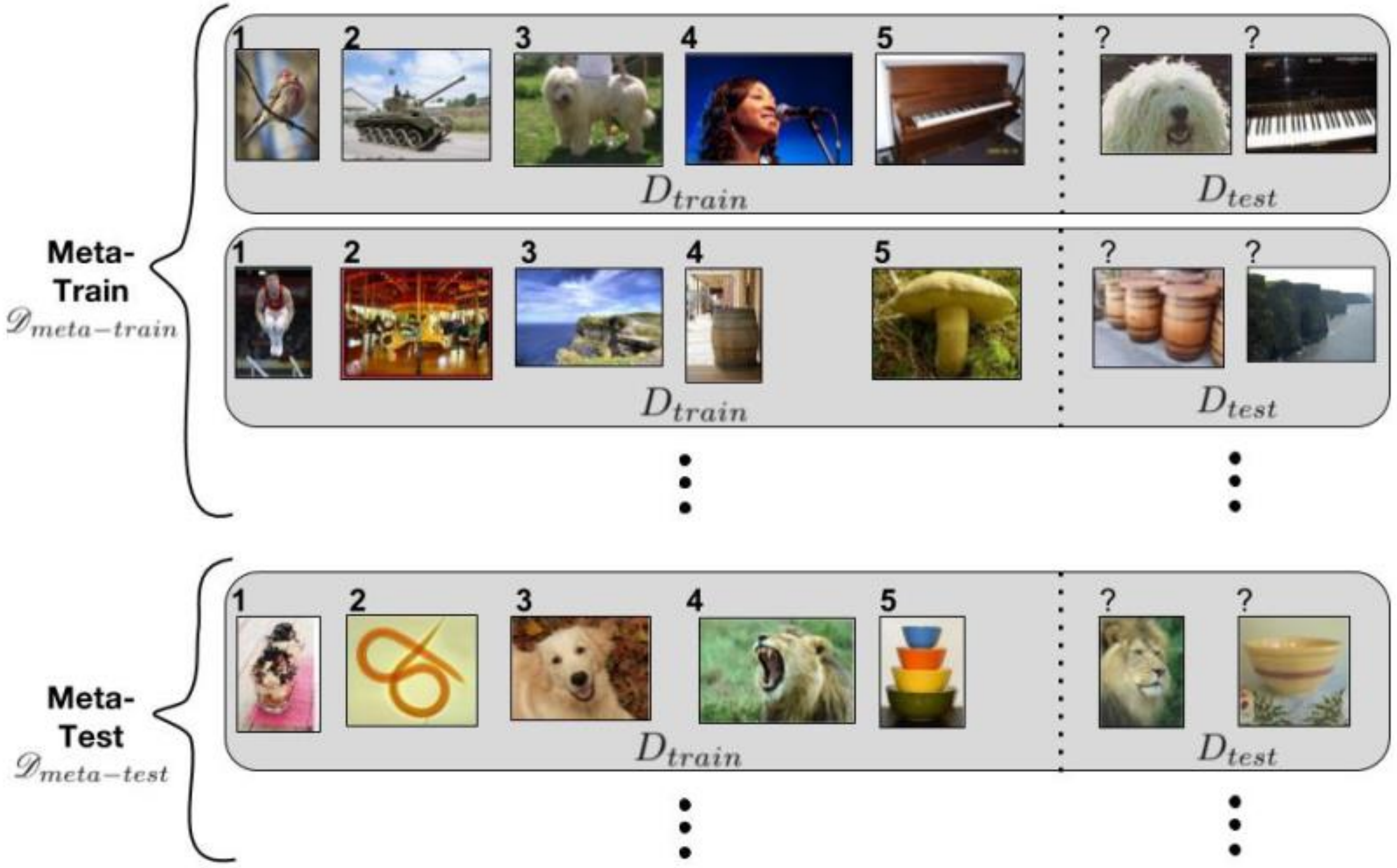
Probabilistic meta learning

Wu, 2019/02/26

Overview

- Review
  - Meta learning
  - MAML
- Probabilistic meta learning
  - Bayesian meta learning
  - Meta learning based on extended PAC-Bayes theory
- Some meta learning applications
  - Meta learning for item cold-start recommendations
  - Learn what to transfer

Meta learning example: few-shot learning



Ravi S, Larochelle H. Optimization as a model for few-shot learning[J]. ICLR.2017.

Problem formulation

- Supervised learning

$$f(x) \rightarrow y$$

- Supervised meta learning
  - Learner

$$f(D_i^{train}, x) \rightarrow y$$

  - Meta learner
    - Learn a learner given a task

MAML

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**

Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks[C]//ICML, 2017

Bayesian MAML

- Probabilistic interpretation of MAML
-

Line 6

$$p(\mathcal{D}_{\mathcal{T}}^{\text{val}} \mid \theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}}) = \prod_{\tau \in \mathcal{T}} p(\mathcal{D}_{\tau}^{\text{val}} \mid \underline{\theta'_{\tau} = \theta_0 + \alpha \nabla_{\theta_0} \log p(\mathcal{D}_{\tau}^{\text{trn}} \mid \theta_0)}),$$

$$\text{where } p(\mathcal{D}_{\tau}^{\text{val}} | \theta'_{\tau}) = \prod_{i=1}^{|\mathcal{D}_{\tau}^{\text{val}}|} p(y_i | x_i, \theta'_{\tau}).$$

Line 8

Kim T, Yoon J, Dia O, et al. Bayesian model-agnostic meta-learning[J]. arXiv preprint arXiv:1806.03836, 2018.

Bayesian MAML

- More general probabilistic interpretation

$$p(\mathcal{D}_{\mathcal{T}}^{\text{val}} \mid \theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}}) = \prod_{\tau \in \mathcal{T}} \left(\underbrace{\int p(\mathcal{D}_{\tau}^{\text{val}} \mid \theta_\tau) p(\theta_\tau \mid \mathcal{D}_{\tau}^{\text{trn}}, \theta_0) \mathrm{d}\theta_\tau}_{} \right)$$

Integrate over posterior distribution instead of point estimation approximation

Bayesian MAML

- Use a more flexible task-train posterior while maintaining the efficiency. How?

- Obtaining M samples from the task-train posterior using SVGD

$$p(\mathcal{D}_{\mathcal{T}}^{\text{val}} \mid \Theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}}) \approx \prod_{\tau \in \mathcal{T}} \left(\frac{1}{M} \sum_{m=1}^{M} p(\mathcal{D}_{\tau}^{\text{val}} \mid \theta_{\tau}^{m}) \right) \quad \textbf{where} \;\; \theta_{\tau}^{m} \sim p(\theta_{\tau} \mid \mathcal{D}_{\tau}^{\text{trn}}, \Theta_0)$$

Bayesian MAML

- SVGD: iteratively transports a set of particles to match the target distribution

Algorithm 1 Bayesian Inference via Variational Gradient Descent

Input: A target distribution with density function $p(x)$ and a set of initial particles $\{x_i^0\}_{i=1}^n$.
Output: A set of particles $\{x_i\}_{i=1}^n$ that approximates the target distribution $p(x)$.
for iteration ℓ **do**

$$x_i^{\ell+1} \leftarrow x_i^\ell + \epsilon_\ell \hat{\phi}^*(x_i^\ell) \quad \text{where} \quad \hat{\phi}^*(x) = \frac{1}{n}\sum_{j=1}^{n}\left[k(x_j^\ell, x)\nabla_{x_j^\ell}\log p(x_j^\ell) + \nabla_{x_j^\ell}k(x_j^\ell, x)\right], \quad (8)$$

where ϵ_ℓ is the step size at the ℓ-th iteration.
end for

Liu Q, Wang D. Stein variational gradient descent: A general purpose bayesian inference algorithm[C]//Advances In Neural Information Processing Systems. 2016: 2378-2386.

Bayesian MAML

- Traditional Loss, can be overfitting

$$\log p(\mathcal{D}^{\text{val}}_{\mathcal{T}_t} | \Theta_0, \mathcal{D}^{\text{trn}}_{\mathcal{T}_t})$$

$$\approx \sum_{\tau \in \mathcal{T}_t} \mathcal{L}_{\text{BFA}}(\Theta_\tau(\Theta_0); \mathcal{D}^{\text{val}}_\tau) \ \text{ where } \ \mathcal{L}_{\text{BFA}}(\Theta_\tau(\Theta_0); \mathcal{D}^{\text{val}}_\tau) = \log \left[\frac{1}{M} \sum_{m=1}^{M} p(\mathcal{D}^{\text{val}}_\tau | \theta^m_\tau) \right]$$

Bayesian MAML

- Chaser loss

$$\arg\min_{\Theta_0} \sum_{\tau} d_p(p_\tau^n \parallel p_\tau^\infty) \approx \arg\min_{\Theta_0} \sum_{\tau} d_s(\Theta_\tau^n(\Theta_0) \parallel \Theta_\tau^\infty).$$

$$p_\tau^n \equiv p_n(\theta_\tau | \mathcal{D}_\tau^{\text{trn}}; \Theta_0)$$

$$p_\tau^\infty \equiv p(\theta_\tau | \mathcal{D}_\tau^{\text{trn}} \cup \mathcal{D}_\tau^{\text{val}})$$

$$\mathcal{L}_{\text{BMAML}}(\Theta_0) = \sum_{\tau \in \mathcal{T}_t} d_s(\Theta_\tau^n \parallel \Theta_\tau^{n+s}) = \sum_{\tau \in \mathcal{T}_t} \sum_{m=1}^{M} \|\theta_\tau^{n,m} - \theta_\tau^{n+s,m}\|_2^2.$$

Bayesian MAML

Algorithm 3 Bayesian Meta-Learning with Chaser Loss (BMAML)

1: Initialize Θ_0
2: **for** $t = 0, \ldots$ until converge **do**
3: Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
4: **for** each task $\tau \in \mathcal{T}_t$ **do**
5: Compute chaser $\Theta_\tau^n(\Theta_0) = \mathrm{SVGD}_n(\Theta_0; \mathcal{D}_\tau^{\mathrm{trn}}, \alpha)$
6: Compute leader $\Theta_\tau^{n+s}(\Theta_0) = \mathrm{SVGD}_s(\Theta_\tau^n(\Theta_0); \mathcal{D}_\tau^{\mathrm{trn}} \cup \mathcal{D}_\tau^{\mathrm{val}}, \alpha)$
7: **end for**
8: $\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} d_s(\Theta_\tau^n(\Theta_0) \parallel \mathrm{stopgrad}(\Theta_\tau^{n+s}(\Theta_0)))$
9: **end for**

Experiments

- sinusoidal function regression
-

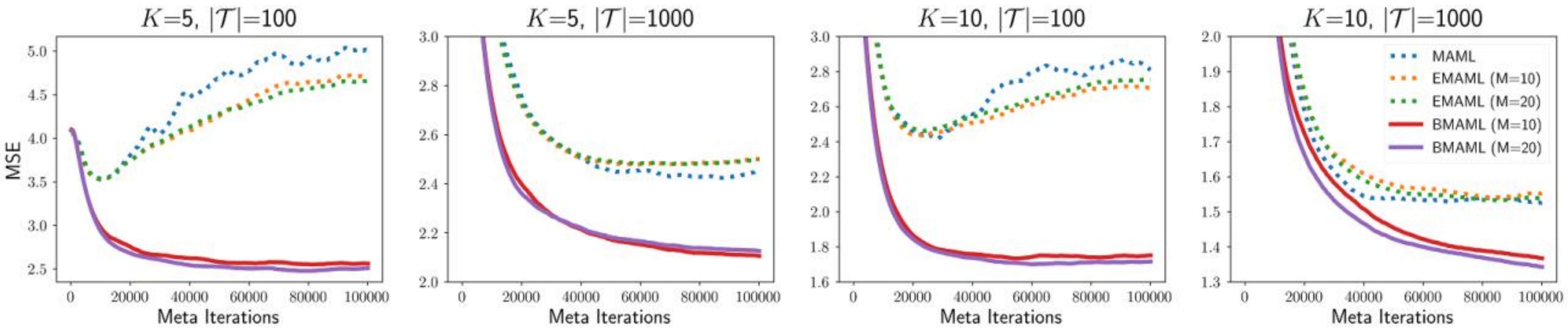
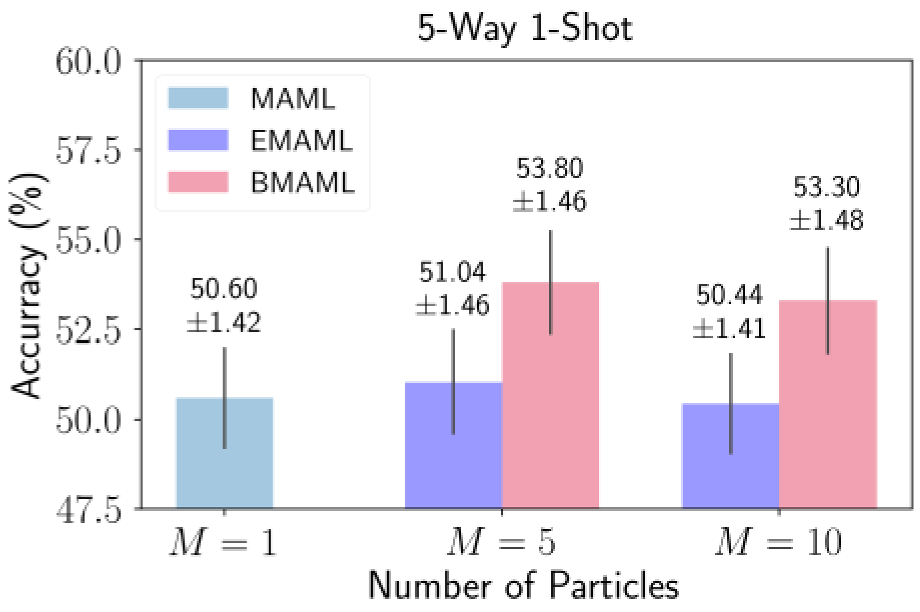


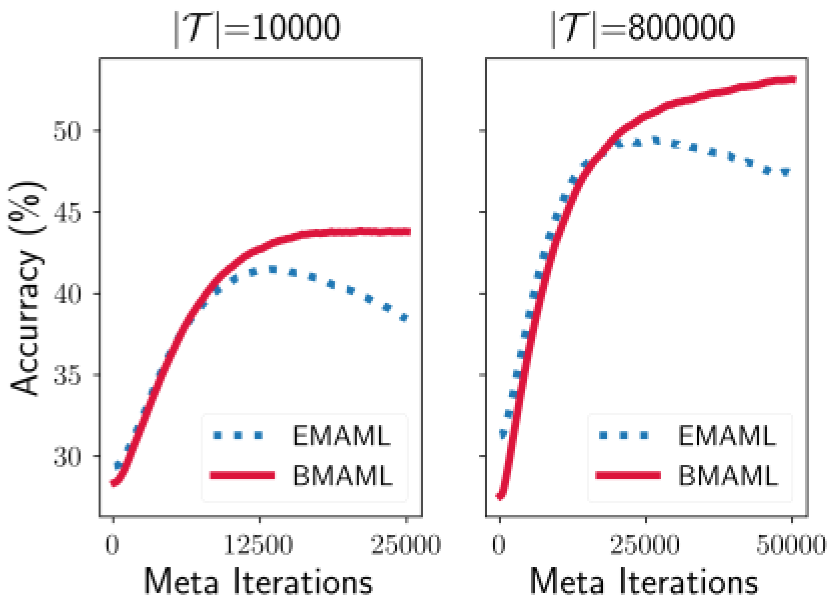
Figure 1: Sinusoidal regression experimental results (meta-testing performance) by varying the number of examples (K-shot) given for each task and the number of tasks $|\mathcal{T}|$ used for meta-training.

Experiments

- Classification on miniImagenet



(a)



(b)

PAC-Bayes Theory for single task problem

- For a hypothesis/algorithm h :
  - generalization error: $er(h, \mathcal{D}) \triangleq \mathbb{E}_{z \sim \mathcal{D}} \ell(h, z)$
  - empirical error: $\widehat{er}(h, S) \triangleq (1/m) \sum_{j=1}^{m} \ell(h, z_i)$
- For a (posterior) distribution Q on the hypothesis space:
  - Generalization error: $er(Q, \mathcal{D}) \triangleq \mathbb{E}_{h \sim Q} er(h, \mathcal{D})$
  - Empirical error: $\widehat{er}(Q, S) \triangleq \mathbb{E}_{h \sim Q} \widehat{er}(h, S)$

PAC-Bayes Generalization Bound

Theorem 1 (McAllester's single-task bound) *Let* $P \in \mathcal{M}$ *be some prior distribution over* $\mathcal{H}$. *Then for any* $\delta \in (0,1]$, *the following inequality holds uniformly for all posteriors distributions* $Q \in \mathcal{M}$ *with probability at least* $1-\delta$,

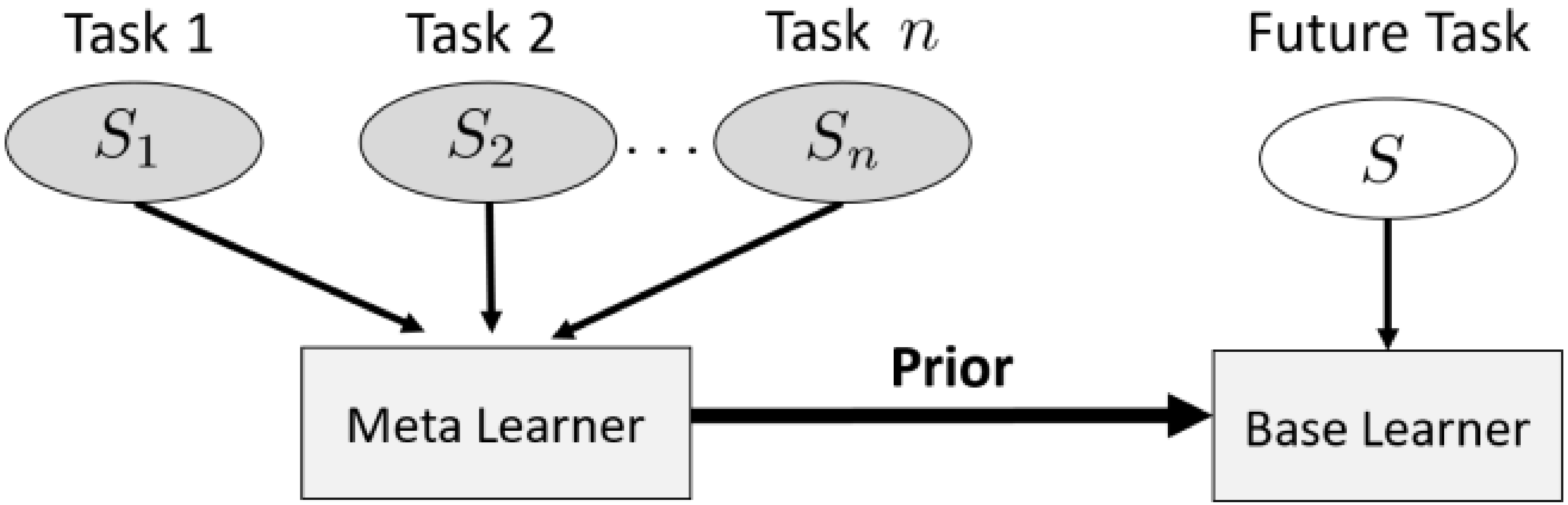
$$er\left(Q,\mathcal{D}\right) \leq \widehat{er}\left(Q,S\right) + \sqrt{\frac{D(Q||P) + \log\frac{m}{\delta}}{2(m-1)}}$$

where $D(Q||P)$ is the Kullback-Leibler (KL) divergence, $D(Q||P) \triangleq \mathop{\mathbb{E}}_{h\sim Q} \log\frac{Q(h)}{P(h)}$.

McAllester, D. A. PAC-Bayesian model averaging. In Conference on Computational Learning Theory (COLT).1999

PAC-Bayes Meta-Learning

- In single task problem, we need to define the prior distribution P
- In meta learning setting, can we learn the distribution of prior distribution P by making use of previous similar tasks?



Amit R, Meir R. Meta-learning by adjusting priors based on extended PAC-Bayes theory[J]. ICML, 2018

Meta-Learning Problem Formulation

- We assume all tasks share the
  - sample space: $\mathcal{Z}$,
  - hypothesis space: $\mathcal{H}$
  - loss function (model architecture): $\ell : \mathcal{H} \times \mathcal{Z} \rightarrow [0, 1]$
- For each task:
  - Unknown data distribution: D_i
  - Observed dataset $S_i \sim \mathcal{D}_i^{m_i}$
  - m_i is the number of data
  - All data distribution D_i are generated from same unknown tasks distribution τ

Meta-Learning Problem Formulation

- To measure the quality of a prior P:

$$er\,(P,\tau) \triangleq \mathop{\mathbb{E}}_{(\mathcal{D},m)\sim\tau}\ \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}\ \mathop{\mathbb{E}}_{h\sim Q(S,P)}\ \mathop{\mathbb{E}}_{z\sim\mathcal{D}}\ \ell(h,z).$$

- For meta learning problem, we define: 'hyper-prior' $\mathcal{P}(P)$,

- observes the training tasks, and then outputs a distribution over priors 'hyper-posterior' $\mathcal{Q}(P)$.

Meta-Learning Problem Formulation

- To measure the performance of 'hyper-posterior' $\mathcal{Q}(P)$.

- Transfer error(generalization error)

$$er(\mathcal{Q},\tau) \triangleq \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} er(P,\tau).$$

- Empirical error

$$\widehat{er}(\mathcal{Q},S_1,...,S_n) \triangleq \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \frac{1}{n}\sum_{i=1}^{n}\widehat{er}(Q(S_i,P),S_i)$$

Meta-learning PAC-Bayes bound

Theorem 2 (Meta-learning PAC-Bayes bound) *Let $Q : \mathcal{Z}^m \times \mathcal{M} \to \mathcal{M}$ be a base learner, and let $\mathcal{P}$ be some pre-defined hyper-prior distribution. Then for any $\delta \in (0, 1]$ the following inequality holds uniformly for all hyper-posterior distributions $\mathcal{Q}$ with probability at least $1 - \delta$,* [4]

$$er(\mathcal{Q}, \tau) \leq \frac{1}{n} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \hat{er}_i(Q_i, S_i) \qquad (4)$$

$$+\frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{D(\mathcal{Q}||\mathcal{P}) + \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} D(Q_i||P) + \log\frac{2nm_i}{\delta}}{2(m_i-1)}}$$

$$+\sqrt{\frac{D(\mathcal{Q}||\mathcal{P}) + \log\frac{2n}{\delta}}{2(n-1)}},$$

where $Q_i \triangleq Q(S_i, P)$.

Implementation

文中选择的超后验分布就是常用的高斯分布。定义如下，其中 $k_{\mathbb{Q}} > 0$ 是人工设定的超参数。

$$\mathbb{Q}_\theta = N(\theta, k_{\mathbb{Q}}^2 I_{N_P X N_P})$$

超先验分布定义为以 0 为均值的高斯分布，如下：

$$\mathbb{P} = N(0, k_{\mathbb{P}}^2 I_{N_P X N_P})$$

这样上图泛化误差上界中 KL 散度可以求得为：$D(\mathbb{Q}_\theta \| \mathbb{P}) = \frac{1}{2k_{\mathbb{P}}^2}\|\theta\|_2^2$。同时，与 VAE 中相同，从 $\mathbb{Q}_\theta$ 中采样一个分布 P 的参数 $\hat{\theta}$ 可以视为在参数 θ 上加一个随机采样的噪声，如下：

$$\hat{\theta} = \theta + \epsilon_P, \epsilon_P \sim N(0, k_{\mathbb{Q}}^2 I_{N_P X N_P})$$

采样 K 个样本 $\hat{\theta}_1, ..., \hat{\theta}_K$，就可以对上图中误差上界中的 $E_{P\sim\mathbb{Q}}$ 做一个估计，如下：

$$\mathbb{E}_{P\sim\mathbb{Q}_\theta} \hat{er}_i(Q_i, S_i) \approx \frac{1}{K}\sum_{j=1}^{K} \hat{er}_i(Q_i(S_i, P_{\hat{\theta}_j}), S_i)$$

Implementation

- Q_i for each task is also defined as parameterized Gaussian distribution

$$Q_{\phi_i}(w) = \prod_{k=1}^{d} \mathcal{N}\left(w_k; \mu_{i,k}, \sigma_{i,k}^2\right)$$

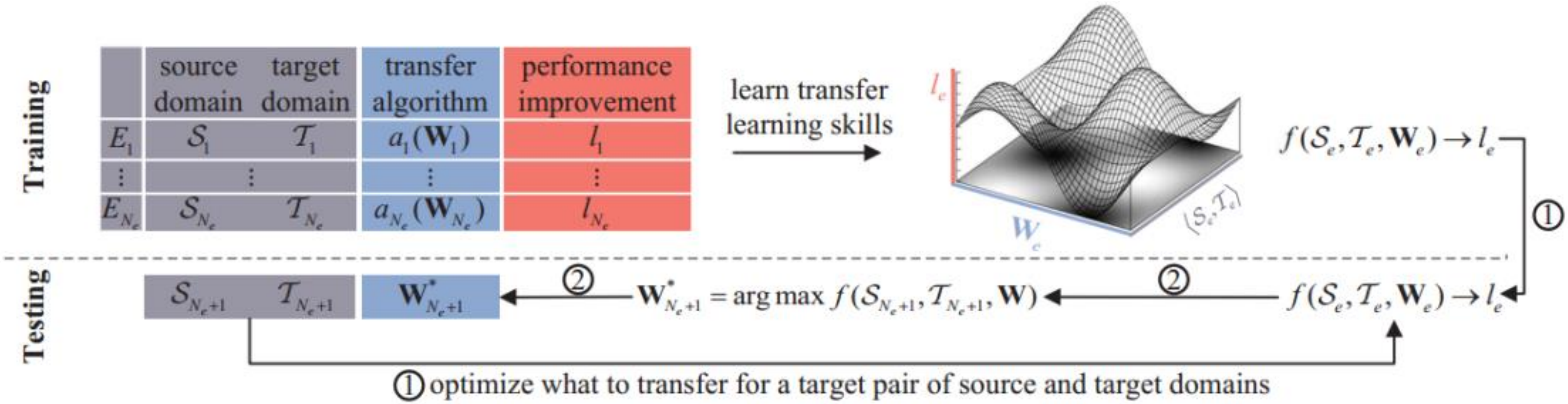
Experiments

- Permute labels and permute pixels

Method	Permuted Labels	Permuted Pixels
Scratch-S	2.27 ± 0.06	7.92 ± 0.22
Scratch-D	2.82 ± 0.06	7.65 ± 0.22
Warm-start	1.07 ± 0.03	7.95 ± 0.39
Oracle	0.69 ± 0.04	6.57 ± 0.32
MLAP-M	0.908 ± 0.04	3.4 ± 0.18
MLAP-S	0.75 ± 0.03	3.54 ± 0.2
MLAP-PL	82.8 ± 5.26	74.9 ± 4.03
MLAP-VB	0.85 ± 0.03	3.52 ± 0.17
Averaged	2.72 ± 0.08	7.63 ± 0.36
MAML	1.16 ± 0.07	3.77 ± 0.8

Learn what to transfer

- Each task is a transfer learning problem



Design function a

- All in all, function a can be approximated by a linear embedding matrix W
  - $a(W, x) = Wx$

Design function f

- The Difference between a Source and a Target Domain
  - maximum mean discrepancy (MMD)

$$\hat{d}_e^2(\mathbf{X}_e^s\mathbf{W}_e, \mathbf{X}_e^t\mathbf{W}_e)$$

  - Variance

$$\sigma_e^2(\mathbf{X}_e^s\mathbf{W}_e, \mathbf{X}_e^t\mathbf{W}_e)$$

- The Discriminative Ability of a Target Domain

$$\tau_e = \mathrm{tr}(\mathbf{W}_e^T\mathbf{S}_e^N\mathbf{W}_e)/\mathrm{tr}(\mathbf{W}_e^T\mathbf{S}_e^L\mathbf{W}_e)$$

- f:

$$1/f = \boldsymbol{\beta}^T\hat{\mathbf{d}}_e + \lambda\boldsymbol{\beta}^T\hat{\mathbf{Q}}_e\boldsymbol{\beta} + \frac{\mu}{\boldsymbol{\beta}^T\boldsymbol{\tau}_e} + b$$

Learn during meta train

$$
\begin{aligned}
&\boldsymbol{\beta}^*, \lambda^*, \mu^*, b^* = \\
&\arg\min_{\boldsymbol{\beta},\lambda,\mu,b} \sum_{e=1}^{N_e} \mathcal{L}_h \left(\boldsymbol{\beta}^T \hat{\mathbf{d}}_e + \lambda \boldsymbol{\beta}^T \hat{\mathbf{Q}}_e \boldsymbol{\beta} + \frac{\mu}{\boldsymbol{\beta}^T \boldsymbol{\tau}_e} + b, \frac{1}{l_e} \right) \\
&\qquad\qquad + \gamma_1 R(\boldsymbol{\beta}, \lambda, \mu, b), \\
&\text{s.t.} \quad \beta_k \geq 0,\ \forall k \in \{1, \cdots, N_k\},\ \lambda \geq 0,\ \mu \geq 0,
\end{aligned}
\tag{2}
$$

Learn a for a new task

$$
\begin{aligned}
\mathbf{W}^*_{N_e+1} &= \arg\max_{\mathbf{W}} \; f(\mathcal{S}_{N_e+1}, \mathcal{T}_{N_e+1}, \mathbf{W}; \boldsymbol{\beta}^*, \lambda^*, \mu^*, b^*) - \gamma_2 \|\mathbf{W}\|_F^2 \\
&= \arg\min_{\mathbf{W}} \, (\boldsymbol{\beta}^*)^T \hat{\mathbf{d}}_{\mathbf{W}} + \lambda^* (\boldsymbol{\beta}^*)^T \hat{\mathbf{Q}}_{\mathbf{W}} \boldsymbol{\beta}^* + \mu^* \frac{1}{(\boldsymbol{\beta}^*)^T \boldsymbol{\tau}_{\mathbf{W}}} \\
&\qquad + \gamma_2 \|\mathbf{W}\|_F^2, \qquad (3)
\end{aligned}
$$

Experiments

- Task: image classification
- Dataset: Caltech-256 (Griffin et al., 2007) and Sketches (Eitz et al., 2012)
- The model and baseline models learn representations of images
- Use nearest neighbor classifier based on representations learned by different algorithms

Experiments

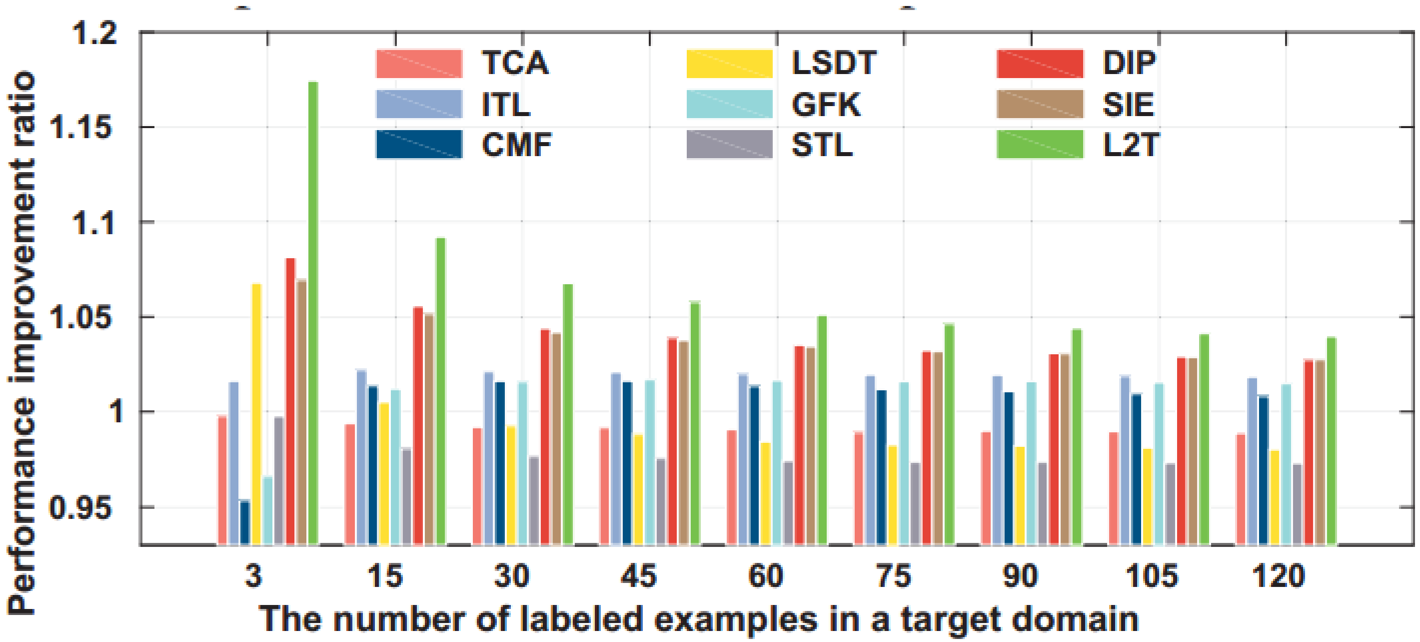
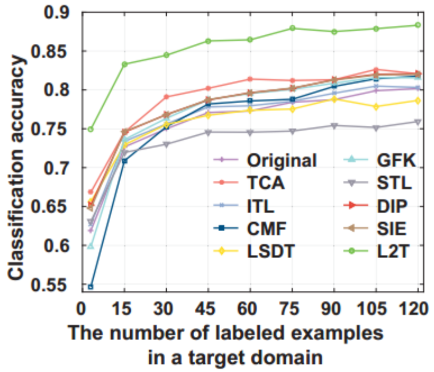
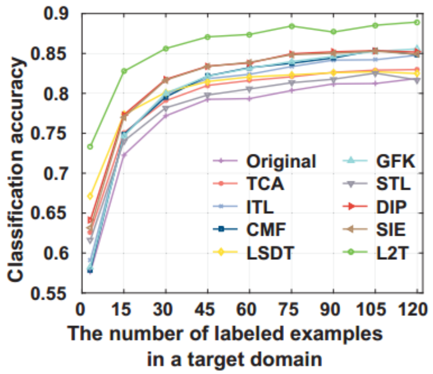


Figure 4. Average performance improvement ratio comparison over 500 testing pairs of source and target domains.

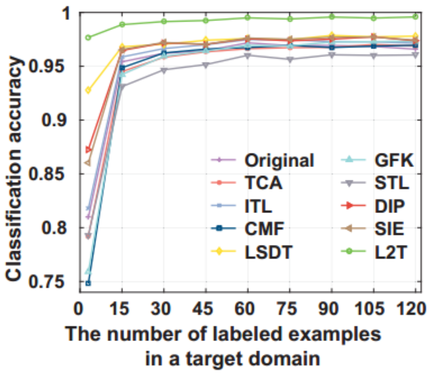
Experiments



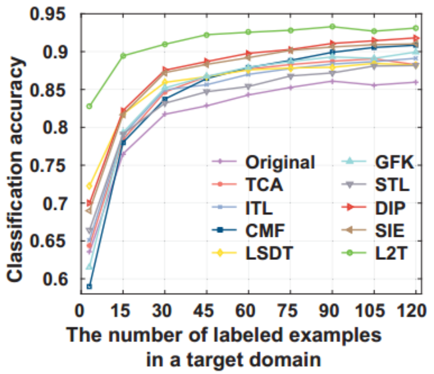
(a) galaxy / harpsichord / saturn → kangaroo / standing-bird / sun



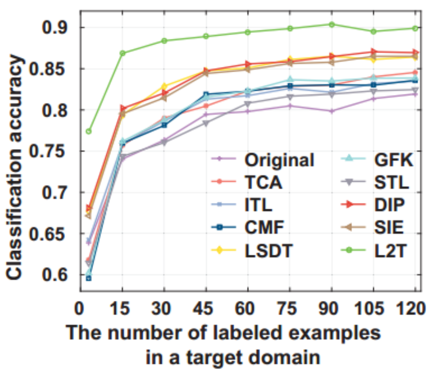
(b) bat / mountain-bike / saddle → bush / person / walkie-talkie



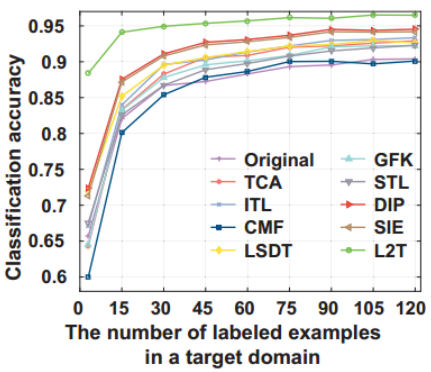
(c) microwave / spider / watch → spoon / trumpet / wheel



(d) bridge / harp / traffic-light → door-handle / hand / present



(e) bridge / helicopter / tripod → key / parrot / traffic-light



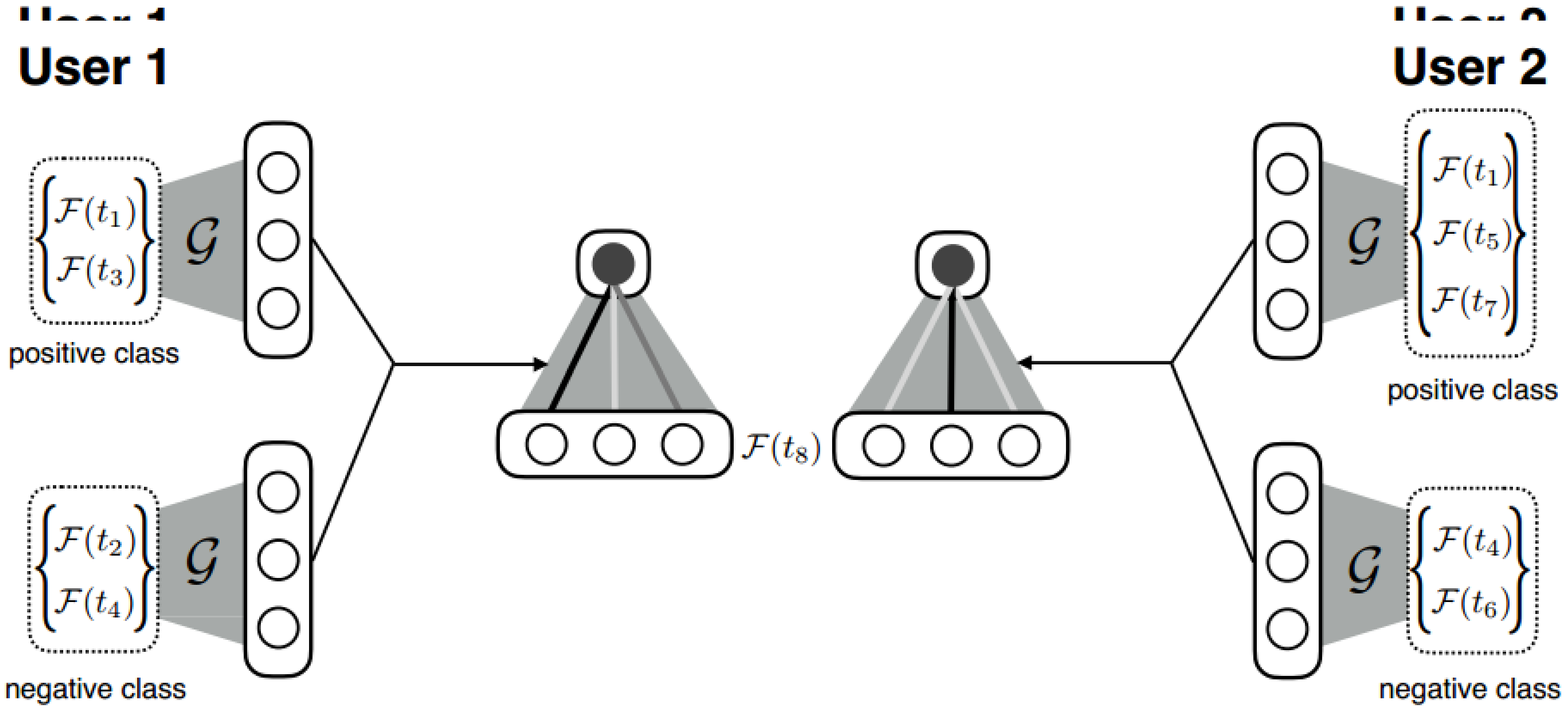
(f) caculator / straw / french-horn → doorknob / palm-tree / scissors

Meta learning for item cold-start recommendations

- Recommending items for a user is regarded as a task
- All items share the same embedding function F.
- Each user has 2 class representation embeddings calculated as follows:

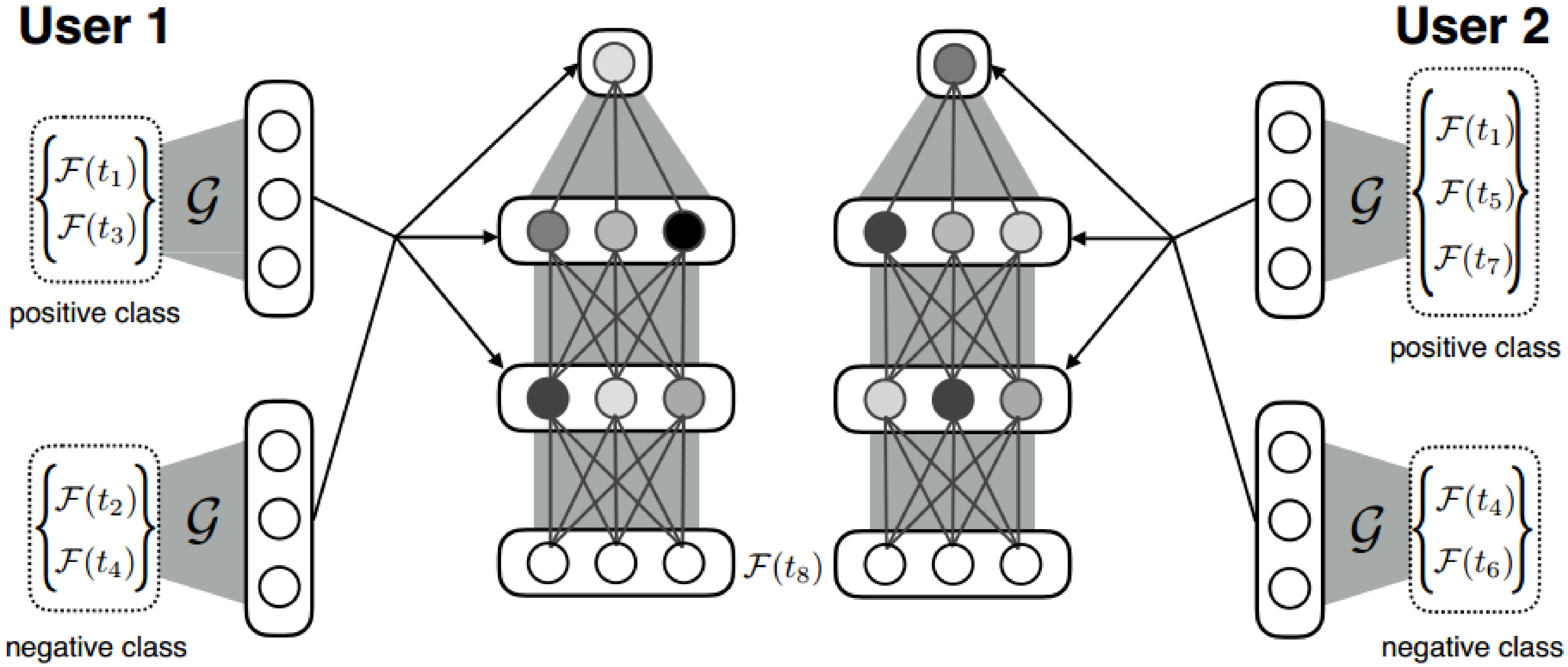
$$R_j^c = \mathcal{G}(\{\mathcal{F}(t_m)\} : t_m \in T_j \wedge (e_{mj} = c))$$

Linear Classifier with Task-dependent weights



$$\Pr(e_{ij}=1|t_i,u_j) = \sigma(b + \mathcal{F}(t_i) \cdot (w_0 R_j^0 + w_1 R_j^1))$$

Non-linear Classifier with Task-dependent Bias



Experiments

Model	AUROC	AUROC (w/CTR)
MF (shallow)	+0.22%	+1.32%
MF (deep)	+0.55%	+1.87%
PROD-BEST	+2.54%	**+2.54%**
LWA	+1.76%	+2.43%
LWA*	+1.98%	+2.31%

Table 1: Performance with LWA

Model	AUROC	AUROC (w/CTR)
MF (shallow)	+0.22%	+1.32%
MF (shallow)	+0.55%	+1.87%
PROD-BEST	+2.54%	+2.54%
NLBA	+2.65%	**+2.76%**

Table 2: Performance with NBLA

Thank you