CS 182 Lecture 21: Meta-Learning

7/20

Meta-learning: DL for low data. We "learn how to learn" via other (prior) tasks
↳ similar to multi-task learning.
↳ use plentiful prior tasks to "meta-train" a model.
supervised learning: $f(x) \to y$
supervised meta-learning: $f(\mathcal{D}^{tr}, x) \to y$ ← set of images + labels. (training set)
↳ can read in training set ($\mathcal{D}^{tr}$) via RNN, ex.

"Generic" meta-learning:

$$\theta^* \leftarrow \arg\min_{\theta} \sum_{i=1}^{n} \mathcal{L}(\phi_i, \mathcal{D}_i^{ts})$$

$$\phi_i = f_\theta(\mathcal{D}_i^{tr}).$$

Meta-learning methods.
→ black-box meta-learning: supervised learning approach, can read in entire (few-shot) $\mathcal{D}^{tr}$.
→ non-parametric meta-learning: unsupervised, nearest neighbors query, ex. matching networks, prototypical networks.
clustering techniques

basic idea for NPML:
find closest embeddings in meta-learned feature space.

→ gradient-based meta-learning: how we adapt to certain tasks ≈ finetuning gradient descent

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \underbrace{\mathcal{L}(f_\theta(\mathcal{D}_i^{tr}), \mathcal{D}_i^{ts})}_{\text{learned nearest neighbor classifier}} = -\sum_{i=1}^{n}\sum_{j=1}^{m} \log p_\theta(y_j^{ts} \mid x_j^{ts}, \mathcal{D}_i^{tr})$$

↑ all tasks ↑ all test points in task.

proportional.

$$p_{nearest}(x_k^{tr} \mid x_j^{ts}) \propto \exp(\phi(x_k^{tr})^T \phi(x_j^{ts})).$$

"softmax".

leverages pre-training as a means of obtaining features.
↳ framed as optimization problem.

$$p_\theta(y_j^{ts} \mid x_j^{ts}, \mathcal{D}_i^{tr}) = \sum_{k: y_k^{tr} = y_j^{ts}} p_{nearest}(x_k^{tr} \mid x_j^{ts})$$

meta-learning rate

$$f_\theta(\mathcal{D}_i^{tr}) = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{tr}).$$

$$\theta \leftarrow \theta - \beta \sum_i \nabla_\theta \mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{tr}), \mathcal{D}_i^{ts})$$

↳ in practice of MAML: ≈ 5-10 grad steps.

$$f_{MAML}(\mathcal{D}^{tr}, x) = f_{\theta'}(x)$$

$$\theta' = \theta - \alpha \sum_{(x,y) \in \mathcal{D}^{tr}} \nabla_\theta \mathcal{L}(f_\theta(x), y)$$

← can easily be implemented. (PyT, TF, etc.)

↓
favorable inductive bias (aligns well)
in short: $\theta \to \nabla_\theta \mathcal{L} \to \theta'$

requires more tuning, second derivatives

Matching networks: different nets (f_θ and g_θ) to embed x^{tr} and x^{ts}
↳ f_θ / g_θ conditioned on $\mathcal{D}_i^{tr}$.

$g(x_k^{tr}, \mathcal{D}_i^{tr}) \to$ bidirectional LSTM (≈ ELMo).
$f(x_j^{ts}, \mathcal{D}_i^{tr}) \to$ attentional LSTM

Prototypical networks:
1. Construct prototype for each class

$$p_\theta(y \mid x_j^{ts}, \mathcal{D}_i^{tr}) \propto \exp(c_y^T f(x_j^{ts}))$$

$$c_y = \frac{1}{N_y} \sum_{k: y_k^{tr} = y} g(x_k^{tr})$$ ← "average" embedding

2. Get rid of all the "complex junk" → no complex embeddings.

Universality: meta-learning can learn any "algo"
↳ a universal M-L method: NN.
↳ applies to both black-box and MAML.

"Generic" RL:

$$\theta^* = \arg\max_{\theta} E_{\pi_\theta(\tau)}[R(\tau)]$$ ← reward

$$= f_{RL}(\mathcal{M})$$ ← MDP. $\{S, A, P, r\}$

Meta-RL:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)] \qquad \phi_i = f_\theta(\mathcal{M}_i).$$

← every task.

$\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$ ← meta-training MDPs
$\mathcal{M}_i \sim p(\mathcal{M})$
meta test-time: sample $\mathcal{M}_{test} \sim p(\mathcal{M})$, $\phi_i = f_\theta(\mathcal{M}_{test})$

same as general meta-learning.

behavior of $f_\theta(\mathcal{M}_i)$?
1. improve policy w/ experience from $\mathcal{M}_i$
2. (new) choose how to interact; choose a_t
↳ "choose" how to explore!

} i.e. train an RNN policy (black-box form).

→ maximize reward over <u>meta-episodes</u> (concatenations of episodes!)

Meta-RL architectures:
→ standard RNN (LSTM) architecture.
→ attention + temporal convolution
→ parallel permutation-invariant context encoder

gradient-based (MAML) meta-RL?
↳ works largely the same as general MAML!