Meta-learning and its applications to NLP

Katia Shutova

ILLC
University of Amsterdam

23 April 2021

Deep learning in NLP

Deep learning models have achieved much success in NLP, but...

- using large datasets for training
- the resulting models are not easily adaptive
- unrealistic to have such large datasets for every possible task, application scenario, domain or language

We need models that are adaptive and can learn from a few examples.

Self-supervised pre-training

- general-purpose word and sentence encoding models
- with self-supervised pre-training (e.g. BERT, GPT-2)
- provide a good starting point for task-specific fine-tuning

and yet...

- to perform well in a given task
- need to fine-tune on a large task-specific dataset

Do not enable few-shot learning or model adaptation.

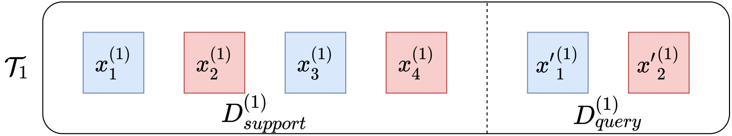
Meta-learning

Meta-learning, aka "learning to learn"

- a framework to train models to perform fast adaptation from a few examples
- a different learning paradigm: episodic learning
- many promising results in computer vision
- and slowly but surely making its way to NLP

Episodic learning

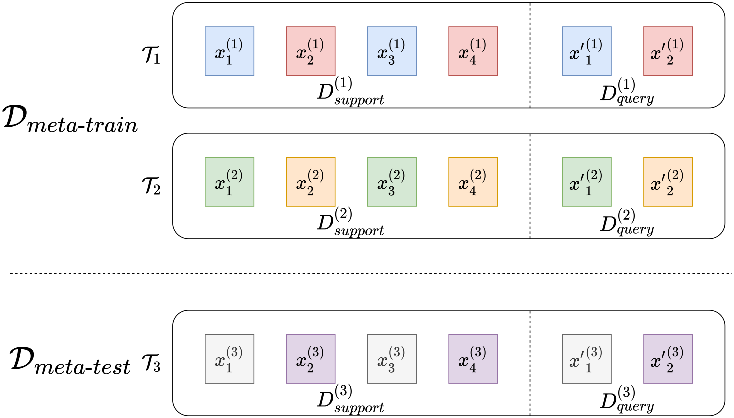
Learning from a collection of few-shot tasks, called episodes



Each episode has its own

- training set = support set
- test set = query set

Meta-training and meta-test sets



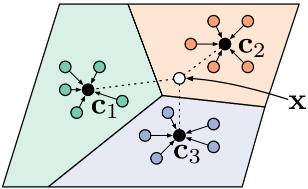
Meta-learning methods

1. **Metric-based**
   - embed examples in each episode using a neural network
   - compute probability distribution over labels for all query examples
   - based on their similarity with the support examples.
2. **Model-based**
   - achieve rapid learning directly through their architectures.
3. **Optimisation-based**
   - explicitly include generalizability in their objective function.

Metric-based method: Prototypical networks

Snell et al 2017. *Prototypical Networks for Few-shot Learning*. NIPS.

- use an embedding function f_θ to encode each input into a vector
- compute a prototype feature vector for every class k
- as the **mean vector** of the embedded **support examples** in this class.



$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\theta(x_i)$$

Prototypical networks

For a given query input x:

- compute the distance between its embedding and each of the prototype vectors
- pass through a **softmax**
- to get the **distribution over classes**

$$P(y=k|x) = softmax(-d_\phi(f_\theta(x), c_k)) = \frac{exp(-d_\phi(f_\theta(x), c_k))}{\sum_{k'} exp(-d_\phi(f_\theta(x), c_{k'}))}$$

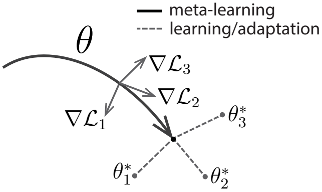
where d_ϕ is the distance function

- Snell et al. use squared Euclidean distance
- The loss function is the negative log-likelihood.

Optimisation-based method: Model-agnostic meta-learning

Finn et al. 2017. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. ICML.

- General and model-agnostic method
- applicable to **any learning problem**
- and **any model architecture** (trainable with gradient descent)



Model-agnostic meta-learning (MAML)

Key intuition:

- learn a good parameter initialisation
- such that the model has **maximal performance** on a new task
- after the parameters have been updated in a few gradient steps
- computed with **a small amount of data** from that new task.

Essentially, the goal is to learn internal representations that are broadly suitable for many tasks.

MAML overview

The learner model f_θ, parametrized by θ

- e.g. a sentence encoder, such as an LSTM or Transformer.

The meta-learning algorithm

1. **Adapt** to a new task $\mathcal{T}_i$, given the task objective
   - computing the loss on the **support set**
2. Perform **meta-optimisation** over a batch of tasks (episodes)
   - computing the loss on the **query sets**.

MAML algorithm

1. Adapt to a new task $\mathcal{T}_i$, given the task objective:
   - compute updated parameters θ_i' using the **support set**

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

2. Perform meta-optimisation over a batch of tasks (episodes)
   - minimise meta-objective across tasks, on the **query sets**:

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

   - perform a meta-update of shared parameters θ

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

MAML algorithm

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
end while

First-order approximation of MAML

- Computing second-order gradients is computationally expensive
- Finn et al. proposed a first order approximation of MAML
- compute the gradients with respect to the updated parameters θ'_i rather than the initial parameters θ

$$\theta \leftarrow \theta - \beta \nabla_{\theta'_i} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$$

Hybrid method: ProtoMAML

Triantafillou et al. 2020. *Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples*. ICLR.

- Prototypical networks with Euclidean distance are **equivalent to a linear model** with a particular parameterization

$$-||f_\theta(x) - c_k||^2 = -f_\theta(x)^T f_\theta(x) + 2{c_k}^T f_\theta(x) - {c_k}^T c_k$$

$f_\theta(x)^T f_\theta(x)$ is constant with respect to class k

$$2{c_k}^T f_\theta(x) - {c_k}^T c_k = {w_k}^T f_\theta(x) + b_k$$

w_k and b_k are the weights and biases for the output unit corresponding to class k.

ProtoMAML

Key idea:

- initialise the final layer of the learner classifier in each episode
- with prototypical network-equivalent weights and biases
- and continue to learn with MAML.

Benefits:

- combines the strength of prototypical networks and MAML
- extends MAML beyond N-way, K-shot scenario.

Meta-learning in NLP

1. Address one NLP task (e.g. focus on learning new classes)
   - **Tasks addressed**: relation classification, entity typing, text classification, word sense disambiguation
2. Apply meta-learning across multiple NLP tasks
   - Bansal et al. 2020 – to be discussed later in this session
3. Apply meta-learning across languages
   - **fast cross-lingual adaptation**: machine translation, dependency parsing, document classification
   - **zero-shot x-lingual transfer**: NLI and question answering (Nooralahzadeh et al. 2020) – to be discussed next Friday

Meta-learning in NLP: Methods

- Model architectures:
  - feed-forward networks
  - graph convolutional networks
  - recurrent networks (LSTM, GRU)
  - transformers
- Meta-learning methods:
  - First-order MAML (the most popular)
  - several extensions thereof proposed
  - Prototypical networks
  - ProtoMAML