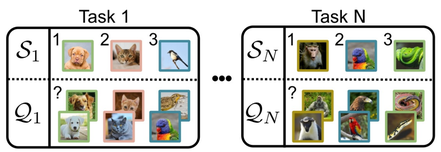
Bayesian Meta-Learning for the Few-Shot Setting via Deep Kernels

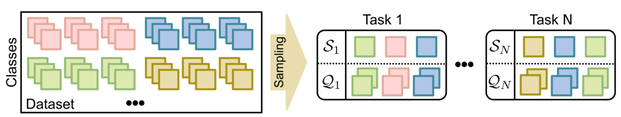
M. Patacchiola, J. Turner, E.J. Crowley, M. O'Boyle, & A. Storkey, University of Edinburgh

What is few-shot learning?

Setting: predict the class membership of data points in a query set Q given a few labeled points in a support set S. Query and support can be grouped in a task T={S,Q}. Shot: number of datapoints per class. Way: number of classes.

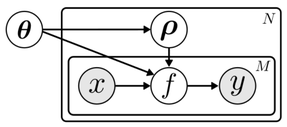


Training and evaluation: tasks sampled from a meta-training (base classes) or a meta-evaluation (unseen classes) dataset.



Bayesian hierarchical methods

Setting: those methods define a set of task-common parameters (shared) and a set of task-specific parameters. Differentiable versions (e.g. MAML, Finn et al. 2017) use two optimization loops (inner and outer) to find those parameters. This is the probabilistic graphical model (regression):



Problem: a full Bayesian treatment is cumbersome, because it is necessary to manage two level of inference. In differentiable methods (e.g. MAML) learning is unstable due to the two loops and the need of higher-order derivatives.

Proposed solution: we do a full integral of the task specific parameters, and optimize only for the cross-task parameters (ML-II). We do this implicitly using Gaussian processes by trasfering a deep kernel across tasks

$$P(\mathcal{T}_t^y|\mathcal{T}_t^x, \boldsymbol{\theta}, \boldsymbol{\phi}) = \int \prod_k P(y_k|x_k, \boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\rho}_t) d\boldsymbol{\rho}_t,$$

Deep Kernel Transfer (DKT)

Algorithm 1 Deep Kernel Transfer (DKT) in the few-shot setting, pseudocode for train and test functions.

Require: $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^N$ train dataset and $\mathcal{T}_* = \{\mathcal{S}_*, \mathcal{Q}_*\}$ test task.
Require: $\hat{\boldsymbol{\theta}}$ kernel hyperparameters and $\hat{\boldsymbol{\phi}}$ neural network weights.
Require: α, β: step size hyperparameters for the optimizers.

function TRAIN($\mathcal{D}$, α, β, $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\phi}}$)
  while not done **do**
    Sample task $\mathcal{T} = \{\mathcal{S}, \mathcal{Q}\} \sim \mathcal{D}$
    Assign $\mathcal{T}^x \leftarrow \forall x \in \mathcal{S} \cup \mathcal{Q}$ and $\mathcal{T}^y \leftarrow \forall y \in \mathcal{S} \cup \mathcal{Q}$
    $\mathcal{L} = -\log p(\mathcal{T}^y|\mathcal{T}^x, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}})$
    Update $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} - \alpha\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{L}$ and $\hat{\boldsymbol{\phi}} \leftarrow \hat{\boldsymbol{\phi}} - \beta\nabla_{\hat{\boldsymbol{\phi}}}\mathcal{L}$
  end while
  return $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\phi}}$
end function
function TEST($\mathcal{T}_*$, $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\phi}}$)
  Assign $\mathcal{T}_*^x \leftarrow \forall x \in \mathcal{S}_*$, $\mathcal{T}_*^y \leftarrow \forall y \in \mathcal{S}_*$, and $x_* \leftarrow x \in \mathcal{Q}_*$
  return $p(y_*|x_*, \mathcal{T}_*^x, \mathcal{T}_*^y, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}})$
end function

Regression

Training: maximize the marginal likelihood over kernel and network parameters given a single task

$$\log P(\mathcal{D}^y|\mathcal{D}^x, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}}) = \sum_t \underbrace{-\frac{1}{2}\mathbf{y}_t^\top[K_t(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}})]^{-1}\mathbf{y}_t}_{\text{data-fit}} \underbrace{-\frac{1}{2}\log|K_t(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}})|}_{\text{penalty}} + c.$$

Inference: the predictive distribution for unseen tasks is given in a closed form

$$\mathbb{E}[f_*] = \mathbf{k}_*{}^\top(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y},$$
$$\text{cov}(f_*) = k_{**} - \mathbf{k}_*{}^\top(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_*.$$

Classification

Training: consists of maximizing the marginal likelihood given by the sum of the marginals for each one of the C individual class outputs, treating classification as a regression problem

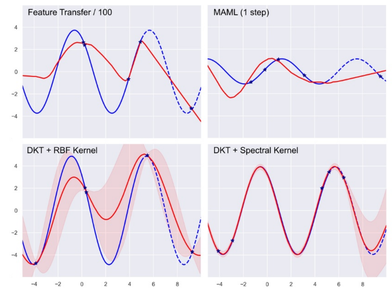
$$\log p(\mathbf{y}|\mathbf{x}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}}) = \sum_{c=1}^{C} \log p(\mathbf{y}_c|\mathbf{x}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}}).$$

Inference: over a new point is done by selecting the output with the highest probability, after passing each predictive mean through a Sigmoid function

$$c_* = \text{argmax}_c\big(\sigma(m_c(x_*))\big)$$

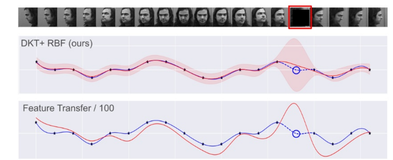
Experiments: regression

Method	in-range	out-of-range
Periodic functions		
ADKL	0.14	–
R2-D2	0.46	–
ALPaCA	0.14 ± 0.09	5.92 ± 0.11
Feature Transfer/1	2.94 ± 0.16	6.13 ± 0.76
Feature Transfer/100	2.67 ± 0.15	6.94 ± 0.97
MAML (1 step)	2.76 ± 0.06	8.45 ± 0.25
DKBaseline + RBF	2.85 ± 1.14	3.65 ± 1.63
DKBaseline + Spectral	2.08 ± 2.31	4.11 ± 1.92
DKT + RBF (ours)	1.38 ± 0.03	2.61 ± 0.16
DKT + Spectral (ours)	**0.08 ± 0.06**	**0.10 ± 0.06**



Left: quantitative results; **Right**: qualitative comparison, real function (blue) approximation (red) uncertainty (shadow)

Method	in-range	out-of-range
Head pose trajectory		
Feature Transfer/1	0.25 ± 0.04	0.20 ± 0.01
Feature Transfer/100	0.22 ± 0.03	0.18 ± 0.01
MAML (1 step)	0.21 ± 0.01	0.18 ± 0.02
DKT + RBF (ours)	0.12 ± 0.04	0.14 ± 0.03
DKT + Spectral (ours)	**0.10 ± 0.01**	**0.11 ± 0.02**



Left: quantitative results; **Right:** uncertainty estimation given a corrupted input.

Experiments: classification

Method	CUB	Omni→EMNIST	ImgNet→CUB
Feature Transfer	46.19 ± 0.64	64.22 ± 1.24	32.77 ± 0.35
Baseline++	61.75 ± 0.95	56.84 ± 0.91	39.19 ± 0.12
MatchingNet	60.19 ± 1.02	75.01 ± 2.09	36.98 ± 0.06
ProtoNet	52.52 ± 1.90	72.04 ± 0.82	33.27 ± 1.09
MAML	56.11 ± 0.69	72.68 ± 1.85	34.01 ± 1.25
RelationNet	62.52 ± 0.34	75.62 ± 1.00	37.13 ± 0.20
DKT + Linear (ours)	60.23 ± 0.76	**75.97 ± 0.70**	38.72 ± 0.42
DKT + CosSim (ours)	**63.37 ± 0.19**	73.06 ± 2.36	**40.22 ± 0.54**
DKT + BNCosSim (ours)	62.96 ± 0.62	75.40 ± 1.10	40.14 ± 0.18

Left table: mean accuracy on CUB and cross-domain settings;
Right table: mean accuracy on Mini-ImageNet.

Method	mini-ImageNet
ML-LSTM	43.44 ± 0.77
SNAIL	45.10
iMAML-HF	49.30 ± 1.88
LLAMA	49.40 ± 1.83
VERSA	48.53 ± 1.84
Amortized VI	44.13 ± 1.78
Meta-Mixture	49.60 ± 1.50
SimpleShot	49.69 ± 0.19
Feature Transfer	39.51 ± 0.23
Baseline++	47.15 ± 0.49
MatchingNet	48.25 ± 0.65
ProtoNet	44.19 ± 1.30
MAML	45.39 ± 0.49
RelationNet	48.76 ± 0.17
DKT + CosSim (ours)	48.64 ± 0.45
DKT + BNCosSim (ours)	**49.73 ± 0.07**

References

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C., and Huang, J.-B. (2019). A closer look at few-shot classification. In International Conference on Learning Representations.

Finn, C., Xu, K., and Levine, S. (2018). Probabilistic model-agnostic meta-learning. In Advances in NeuralInformation Processing Systems.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In Advances in NeuralInformation Processing Systems.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Acknowledgments: Huawei DDMPLab Innovation Research grant, and European Union's Horizon 2020 research and innovation programme (agreement No 732204, Bonseyes)