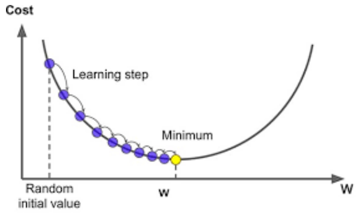
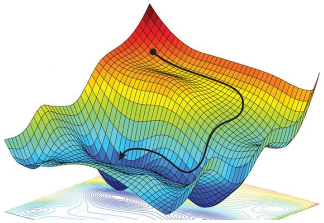
Hypernetwork approach to few-shot learning

P. Spurek

January 4, 2023

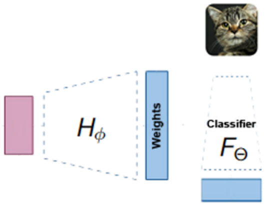
Gradient Optimization





- We have Neural Network F_Θ
- We have cost function $C_\Theta(X, y)$
- $\Theta := \Theta - \nabla_\Theta C_\Theta$

Gradient Optimization



- We have Neural Network (Target network) F_Θ
- We have Neural Network (Hyper network) $H_\phi : X \to \theta$
- Weights $\theta(\phi)$ depends on ϕ
- We have cost function $C_\phi(X, y)$
- $\phi := \phi - \nabla_\phi C_\phi$

Image continuous reprehension

- https://arxiv.org/pdf/1902.10404.pdf

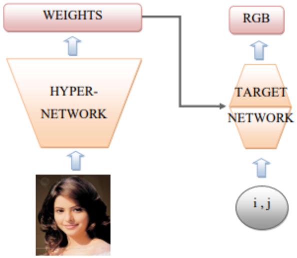
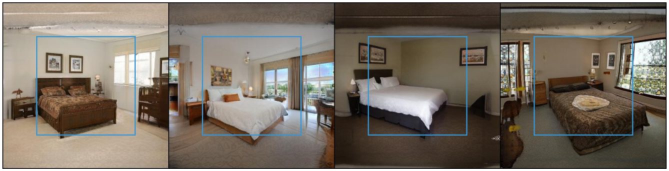
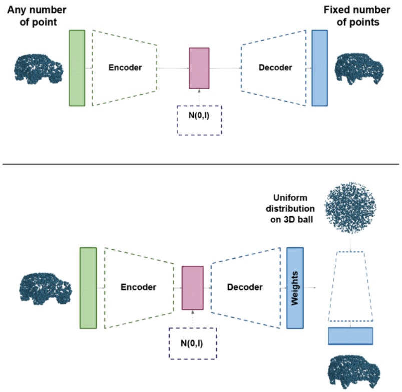


Image continuous reprehension

- https://arxiv.org/pdf/2011.12026.pdf



Hypernetwork approach to generating point clouds

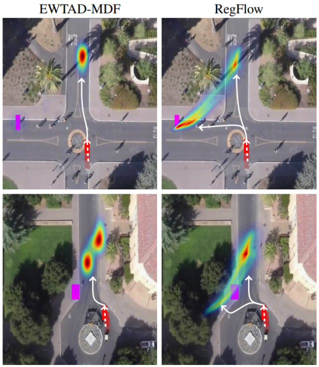


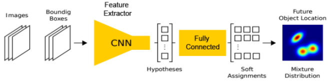
https://arxiv.org/pdf/2003.00802.pdf

Hypernetwork approach to generating point clouds

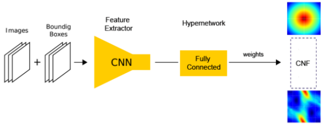
- https://arxiv.org/pdf/2003.00802.pdf
- https://arxiv.org/pdf/2108.01411.pdf
- https://arxiv.org/pdf/2102.05973.pdf
- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9627800
- https://ujchmura-my.sharepoint.com/:v:/r/personal/przemyslaw_spurek_uj_edu_pl/Documents/hyperflow-video.mp4?csf=1&web=1&e=GrysIB
- https://arxiv.org/pdf/2110.05770.pdf

Hypernetwork approach to regression





(a) EWTAD-MDF [36] architecture. In the first stage, it generates a hypotheses trained with the EWTA loss and the second part fits a mixture distribution by predicting soft assignments of the hypotheses to mixture components.



(b) RegFlow architecture. We use a one stage approach. Our feature extractor produces weights for the Continuous Normalizing Flow (CNF) module, which is able to model more complex distributions.

- `https://arxiv.org/pdf/2011.14620.pdf`
- `https://github.com/maciejzieba/regressionFlow`

HYPERNETWORKS
HyperNetwork
HyperPocket
HyperShot
HyperMAML
HyperColor
HyperCube
HyperReg

Few-shot learning

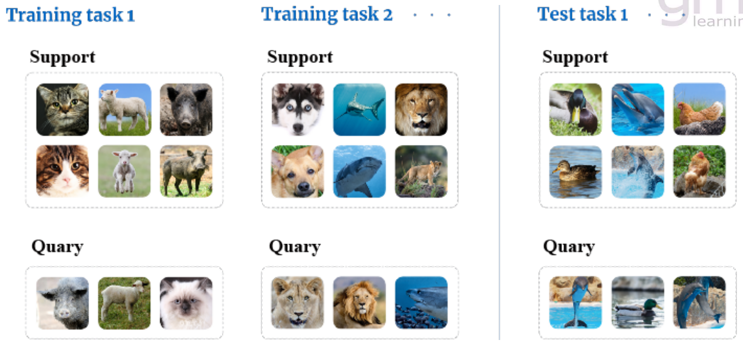


Figure: In a few shot-learning problems, known sets are called support sets and are used to classify elements from query sets. In training, we have labels in support and quarry sets. After training (in test time), our model has to correctly classify the query set using elements from the support set.

Few-shot learning

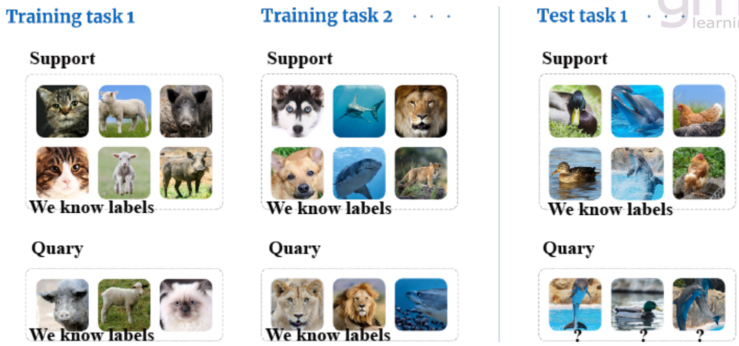
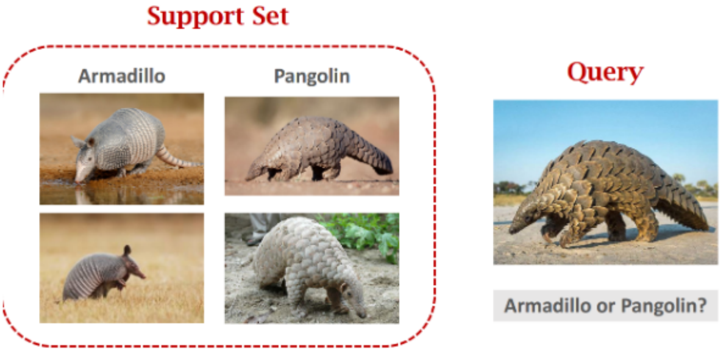


Figure: In a few shot-learning problems, known sets are called support sets and are used to classify elements from query sets. In training, we have labels in support and quarry sets. After training (in test time), our model has to correctly classify the query set using elements from the support set.

Few-shot learning



MAML

The main idea of MAML is to find the parameters of a model so that it can adapt to a new task in a few gradient steps or even a single gradient step, and produce good results on a new task, see Fig. 21.

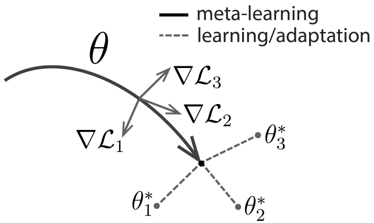


Figure: The MAML algorithm produces parameters θ, which can quickly adapt to many different tasks in a few gradient steps θ_i^*.

MAML

Let

$$S = \{(x_l, y_l)\}_{l=1}^{L}$$

be a support-set containing input-output pairs, with *L* equal to one (1-shot) or five (5-shot), and

$$Q = \{(x_m, y_m)\}_{m=1}^{M}$$

be a query-set (sometimes referred to in the literature as a target-set), with *M* typically one order of magnitude greater than *L*. For ease of notation the support and query sets are grouped in a task

$$T = \{S, Q\},$$

with the dataset

$$D = \{T_t\}_{t=1}^{N}$$

defined as a collection of such tasks. Models are trained on random tasks sampled from *D*

MAML

In MAML, the updated parameter vector θ_i' is computed using one or more gradient descent updates on task T_i. For example, when using one gradient update,

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{T_i}(f_\theta)$$

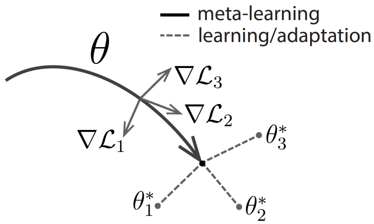
The step size α may be fixed as a hyperparameter or meta learner. For simplicity of notation, we will consider one gradient update for the rest of this section, but using multiple gradient updates is a straightforward extension.

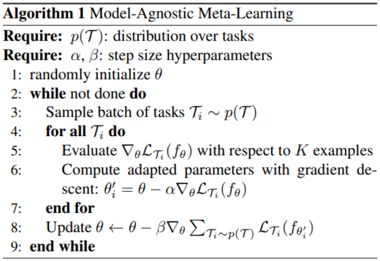
The meta-optimization across tasks is performed via stochastic gradient descent (SGD), such that the model parameters θ are updated as follows:

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

where β is the meta step size.

MAML



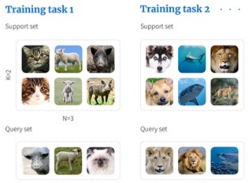

Algorithm 1 Model-Agnostic Meta-Learning

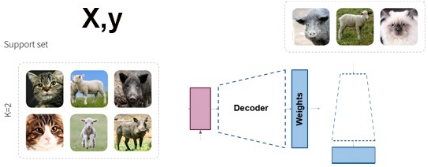
```
Require: $p(\mathcal{T})$: distribution over tasks
Require: $\alpha$, $\beta$: step size hyperparameters
1: randomly initialize $\theta$
2: while not done do
3:     Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4:     for all $\mathcal{T}_i$ do
5:         Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ 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
```

Figure: The MAML algorithm produces parameters θ, which can quickly adapt to many different tasks in a few gradient steps θ_i^*.

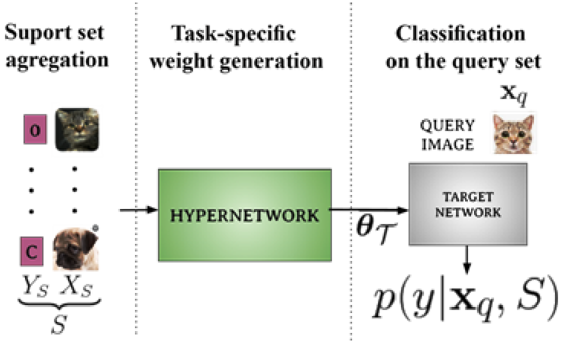
`https://arxiv.org/pdf/1703.03400.pdf`

Hypernetwork

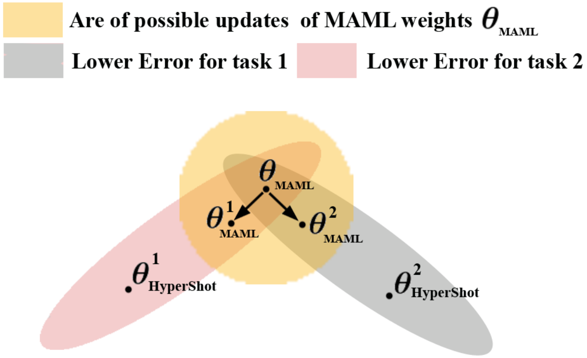




Hypernetwork

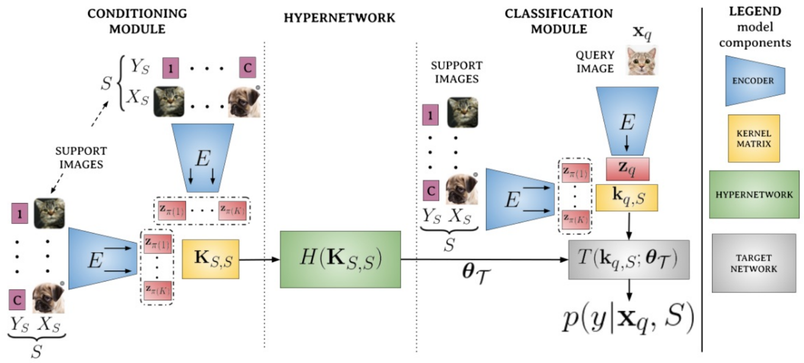


HyperShot

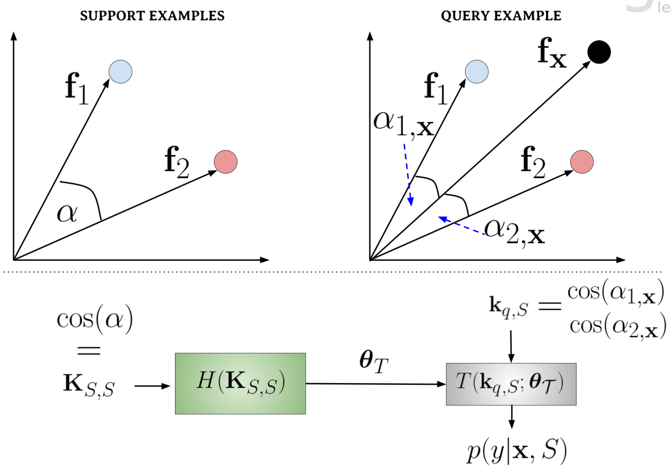


Are of possible updates of MAML weights θ_{MAML}
Lower Error for task 1
Lower Error for task 2
θ_{MAML}
θ^1_{MAML}
θ^2_{MAML}
$\theta^1_{HyperShot}$
$\theta^2_{HyperShot}$

HyperShot


CONDITIONING MODULE
HYPERNETWORK
CLASSIFICATION MODULE
LEGEND model components
S
Y_S
X_S
1
C
SUPPORT IMAGES
E
$\mathbf{z}_{\pi(1)}$
$\mathbf{z}_{\pi(K)}$
$\mathbf{K}_{S,S}$
$H(\mathbf{K}_{S,S})$
$\boldsymbol{\theta}_\mathcal{T}$
QUERY IMAGE
$\mathbf{x}_q$
$\mathbf{z}_q$
$\mathbf{k}_{q,S}$
$T(\mathbf{k}_{q,S};\boldsymbol{\theta}_\mathcal{T})$
$p(y|\mathbf{x}_q, S)$
ENCODER
KERNEL MATRIX
HYPERNETWORK
TARGET NETWORK

HyperShot

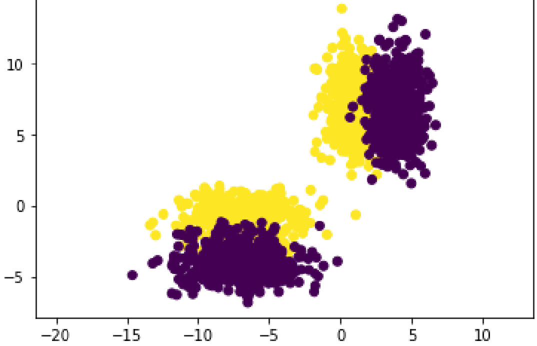

SUPPORT EXAMPLES
$\mathbf{f}_1$
$\mathbf{f}_2$
α
QUERY EXAMPLE
$\mathbf{f}_{\mathbf{x}}$
$\mathbf{f}_1$
$\mathbf{f}_2$
$\alpha_{1,\mathbf{x}}$
$\alpha_{2,\mathbf{x}}$
$\cos(\alpha) = \mathbf{K}_{S,S}$
$H(\mathbf{K}_{S,S})$
$\boldsymbol{\theta}_T$
$\mathbf{k}_{q,S} = \begin{matrix}\cos(\alpha_{1,\mathbf{x}}) \\ \cos(\alpha_{2,\mathbf{x}})\end{matrix}$
$T(\mathbf{k}_{q,S}; \boldsymbol{\theta}_T)$
$p(y|\mathbf{x}, S)$

HyperShot

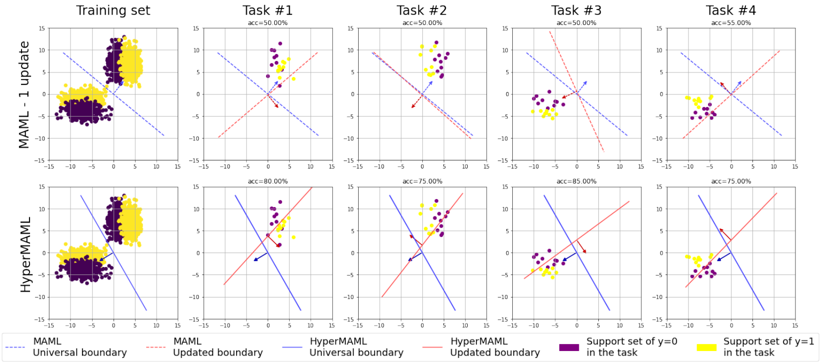
WACV 2022

`https://arxiv.org/pdf/2203.11378.pdf`

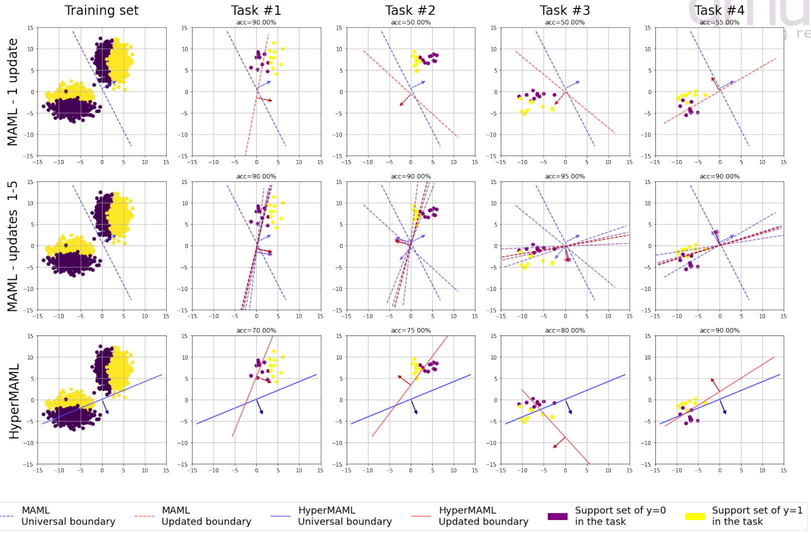
HyperMAML - Motivation



HyperMAML - Motivation



HyperMAML - Motivation



HyperMAML - in practice

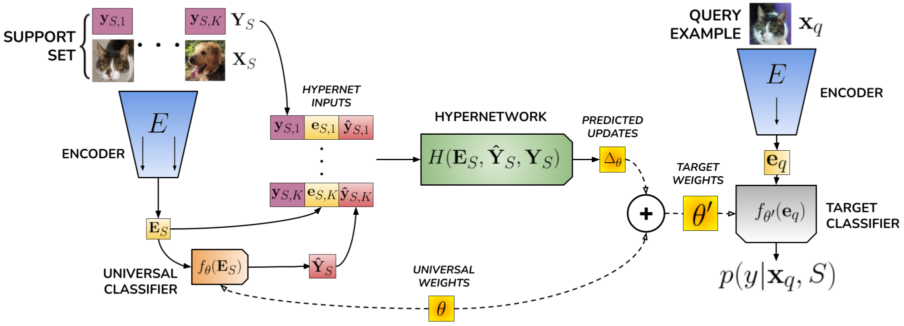
- Update in MAML

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{S}_i}(f_\theta), \tag{1}$$

- Update in HyperMAML

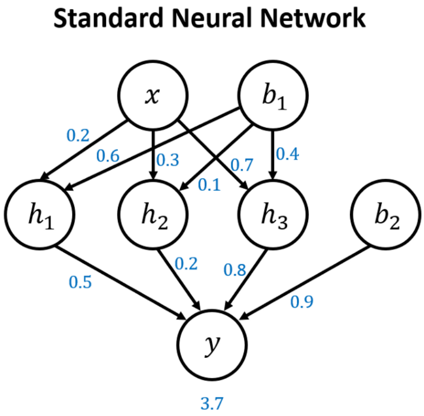
$$\theta' = \theta + \Delta_\theta = \theta + H(\mathbf{E}_{\mathcal{S}}, \hat{\mathbf{Y}}_{\mathcal{S}}, \mathbf{Y}_{\mathcal{S}}). \tag{2}$$

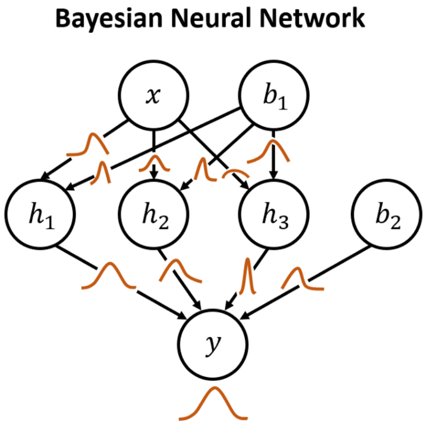
HyperMAML



SUPPORT SET
$\mathbf{y}_{S,1}$
$\mathbf{y}_{S,K}$
$\mathbf{Y}_S$
$\mathbf{X}_S$
ENCODER
E
$\mathbf{E}_S$
UNIVERSAL CLASSIFIER
$f_\theta(\mathbf{E}_S)$
$\hat{\mathbf{Y}}_S$
HYPERNET INPUTS
$\mathbf{y}_{S,1}$ $\mathbf{e}_{S,1}$ $\hat{\mathbf{y}}_{S,1}$
$\mathbf{y}_{S,K}$ $\mathbf{e}_{S,K}$ $\hat{\mathbf{y}}_{S,K}$
HYPERNETWORK
$H(\mathbf{E}_S, \hat{\mathbf{Y}}_S, \mathbf{Y}_S)$
PREDICTED UPDATES
Δ_θ
UNIVERSAL WEIGHTS
θ
TARGET WEIGHTS
θ'
QUERY EXAMPLE
$\mathbf{x}_q$
ENCODER
E
$\mathbf{e}_q$
$f_{\theta'}(\mathbf{e}_q)$
TARGET CLASSIFIER
$p(y|\mathbf{x}_q, S)$

Bayes By Back-propagation (BBB)





Bayes By Back-propagation (BBB)

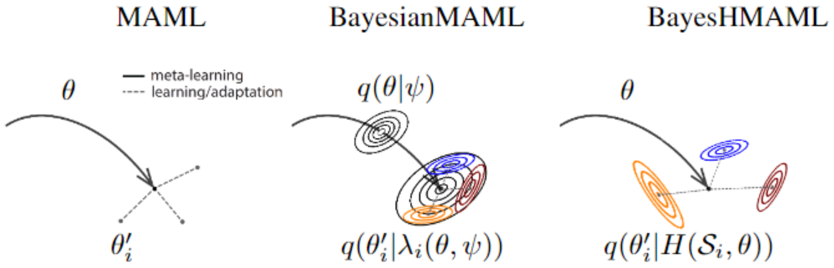
In the case of Bayesian Neural networks in Bayes By Backpropagation (BBB) we optimize the cost function

$$\mu^*, \Sigma^* = \operatorname{argmax}_{\mu,\sigma} \sum_{(x_i, y_i) \in D_{tr}} log[p(y_i | x_i, \theta)] - KL[p(\theta), p(\theta_0)]$$

where

$$\theta \sim N(\mu, \Sigma), \qquad \theta_0 \sim N(0, I)$$

HyperBMAML


MAML
BayesianMAML
BayesHMAML
meta-learning
learning/adaptation
θ
θ'_i
$q(\theta|\psi)$
$q(\theta'_i|\lambda_i(\theta,\psi))$
θ
$q(\theta'_i|H(\mathcal{S}_i,\theta))$

HyperBMAML

Hence, the variational inference along with reparametrization gradients (i.e., Bayes by backpropagation blundell2015weight) is typically used, and the following objective (evidence lower bound) maximized w.r.t variational parameters λ_i and ψ:

$$\mathcal{L}_{\mathcal{D}} = \mathcal{E}_{q(\theta|\psi)} \left[\underbrace{\sum_{i}^{N} \mathcal{E}_{q(\theta_i'|\lambda_i)} \left[\log p({}_i|\theta_i') - KL\left(q(\theta_i'|\lambda_i)|p(\theta_i'|\theta)\right)\right]}_{\mathcal{L}_i} \right] - KL(q(\theta|\psi)|p(\theta))$$

where $q(\theta_i'|\lambda_i)$ and $q(\theta|\psi)$ are respectively per-task posterior approximation and approximate posterior for the universal weights. They are tied together by the prior $p(\theta_i'|\theta)$.

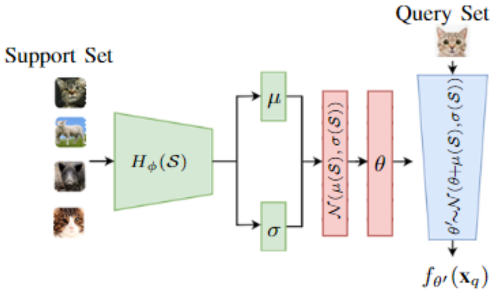
HyperBMAML

All these modifications simplify the optimization landscape and, taken together along with our Hypernetwork-based adjustment strategy (details below) should enable better optima for our objective:

$$\mathcal{L}_D^{our} = \sum_i^N \mathcal{E}_{q(\theta_i'|\lambda_i(\theta,\S_i))} \left[\log p(_i|\theta_i') - \gamma \cdot KL\left(q(\theta_i'|\lambda_i(\theta,\S_i))|p(\theta_i')\right)\right]$$

In practice, we use the standard normal priors for the weights of the neural network f, i.e., $p(\theta_i') = \mathcal{N}(\theta_i'|0,\mathbb{I})$, and the hyperparameter γ allows controlling impact of the priors and compensating for model mispecification.

HyperBMAML


Query Set
Support Set
$H_\phi(\mathcal{S})$
μ
σ
$\mathcal{N}(\mu(\mathcal{S}),\sigma(\mathcal{S}))$
θ
$\theta' \sim \mathcal{N}(\theta + \mu(\mathcal{S}), \sigma(\mathcal{S}))$
$f_{\theta'}(\mathbf{x}_q)$

Thank you
for your kind attention