Metaparameters and metaoptimization

MIPT

2024

Hyperparameters

Definition

Prior for parameters $\mathbf{w}$ and structure $\boldsymbol{\Gamma}$ of the model $\mathbf{f}$ is a distrubution $p(\mathbf{W}, \boldsymbol{\Gamma}|\mathbf{h}) : \mathbb{W} \times \Gamma \times \mathbb{H} \to \mathbb{R}^+$, where $\mathbb{W}$ is a parameter space, Γ is a structure space.

Definition

Hyperparameters $\mathbf{h} \in \mathbb{H}$ of the models are the parameters of $p(\mathbf{w}, \boldsymbol{\Gamma}|\mathbf{h})$ (parameters of prior $\mathbf{f}$).

Metaparameters

Wikipedia

A parameter that controls the value of one or more others.

Definition

Metaparameters $\boldsymbol{\lambda}$ are the parameters of optimization problem.

Is dropout rate a metaparameter or hyperparameter?

Consider using Gumbel-Softmax as a variational distribution for model deep neural network structure. Is temperature a metaparameter or hyperparameter?

Is SGD learning rate a metaparameter or hyperparameter?

Gradient descent for evidence estimation

Statement

Let L be a Lipschitz function, and optimization operator be a bijection. Then entropy difference for two steps is:

$$\mathrm{S}(q'(\mathbf{w})) - \mathrm{S}(q(\mathbf{w})) \simeq \frac{1}{r}\sum_{g=1}^{r}\left(-\beta \mathrm{Tr}[\mathbf{H}(\mathbf{w}'^{g})] - \beta^2 \mathrm{Tr}[\mathbf{H}(\mathbf{w}'^{g})\mathbf{H}(\mathbf{w}'^{g})]\right).$$

Final estimation for the τ optimization step:

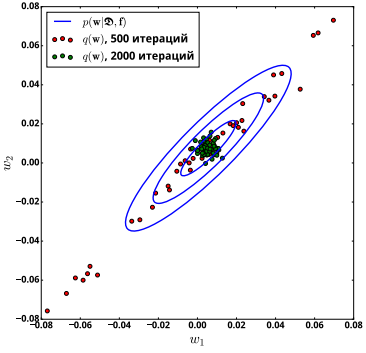
$$\log \hat{p}(\mathbf{Y}|\mathfrak{D}, \mathbf{h}) \sim \frac{1}{r}\sum_{g=1}^{r} L(\mathbf{w}_{\tau}^{g}, \mathfrak{D}, \mathbf{Y}) + \mathrm{S}(q^0(\mathbf{w})) +$$

$$+\frac{1}{r}\sum_{b=1}^{\tau}\sum_{g=1}^{r}\left(-\beta \mathrm{Tr}[\mathbf{H}(\mathbf{w}_{b}^{g})] - \beta^2 \mathrm{Tr}[\mathbf{H}(\mathbf{w}_{b}^{g})\mathbf{H}(\mathbf{w}_{b}^{g})]\right),$$

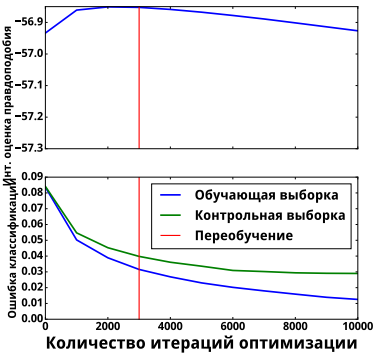
$\mathbf{w}_{b}^{g}$ is a parameter vector for optimization g on the step b, $\mathrm{S}(q^0(\mathbf{w}))$ is an initial entropy.

Overfitting, Maclaurin et. al, 2015

Gradient descent does not optimize KL-divergence $\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}|\mathfrak{D},\mathbf{h}))$. Evidence estimation gets worse while optimization tends to the optimal parameter values. This can be considered as a overfitting start.



Convergence

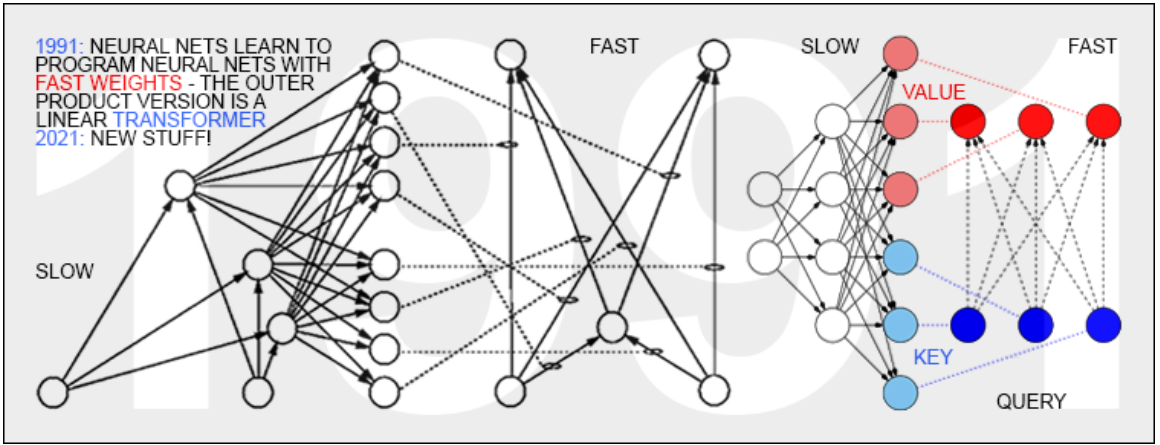


Overfitting start

A neural network that embeds its own meta-levels

A model is decomposed into submodels with different subtasks:

- "Normal" model: optimization and inference.
- Evaluation model: validation performance estimation.
- Analyzing model: analysis of model parameters.
- Modifiyng model: parameter modification.



https://people.idsia.ch/~juergen/fast-weight-programmer-1991-transformer.html

In-context learning with attention [1]

Attention

Definition(Unnormalized Dot Attention) let $\mathbf{K} = (\mathbf{k}_1, \dots \mathbf{k}_T) \in \mathbb{R}^{d_{\text{in}} \times T}$ and $\mathbf{V} = (\mathbf{v}_1, \dots \mathbf{v}_T) \in \mathbb{R}^{d_{\text{out}} \times T}$ denote matrices representing T key and value vectors. Let $\mathbf{q} \in \mathbb{R}^{d_{\text{in}}}$ denote a query vector. An unnormalised linear dot attention operation $\mathit{Attention}(\mathbf{K}, \mathbf{V}, \mathbf{q})$ computes the following weighted average of value vectors $\mathbf{v}_t$:

$$\mathit{Attention}(\mathbf{K}, \mathbf{V}, \mathbf{q}) = \sum_{t=1}^{T} \alpha_t \mathbf{v}_t \tag{1}$$

where the weights $\alpha_t = \mathbf{k}_t^\top \mathbf{q} \in \mathbb{R}$ are dot products between key $\mathbf{k}_t$ and query $\mathbf{q}$ vectors, and are called attention weights.

[1] See Sergey Skorik's talk, spring 2023

Idea

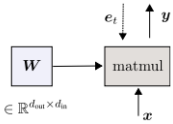


Figure 1. The primal form of a linear layer to be contrasted with the dual form in Figure 2.

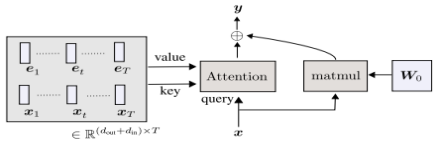
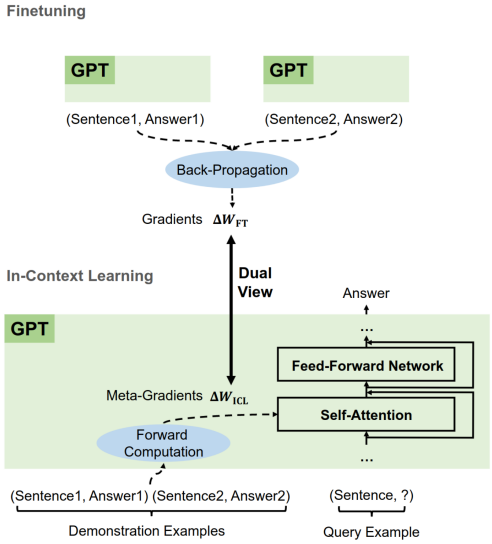


Figure 2. The dual form of a linear layer trained by gradient descent is a key-value memory with attention storing the entire training experience. Compare to the primal form in Figure 1.

Idea


Finetuning
GPT
GPT
(Sentence1, Answer1)
(Sentence2, Answer2)
Back-Propagation
Gradients ΔW_{FT}
In-Context Learning
Dual View
Answer
GPT
Feed-Forward Network
Meta-Gradients ΔW_{ICL}
Self-Attention
Forward Computation
(Sentence1, Answer1) (Sentence2, Answer2)
(Sentence, ?)
Demonstration Examples
Query Example

Duality

Corollary

The following systems S_1, S_2 are equivalent:

- S_1 (*Primal form*): A linear layer in a neural network trained by gradient descent in some error function using T training inputs to this layer $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ with $\mathbf{x}_t \in \mathbb{R}^{d_{in}}$ and corresponding (backpropagation) error signal $(\mathbf{e}_1, \ldots, \mathbf{e}_T)$ with $\mathbf{e}_t \in \mathbb{R}^{d_{out}}$ obtained by gradient descent. Its weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ is thus:

$$\mathbf{W} = \mathbf{W}_0 + \sum_{t=1}^{T} \mathbf{e}_t \otimes \mathbf{x}_t \tag{2}$$

  Where $\mathbf{W}_0 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ in is the initialisation. The layer transforms input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ to output $S_1(\mathbf{x}) \in \mathbb{R}^{d_{\text{out}}}$ as:

$$S_1(\mathbf{x}) = \mathbf{W}\mathbf{x} \tag{3}$$

Corollary

- S_2 (*Dual form*): A layer which stores T key-value pair $(\mathbf{x}_1, \mathbf{e}_1), \ldots (\mathbf{x}_T, \mathbf{e}_T)$ i.e., a key matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T) \in \mathbb{R}^{d_{\text{in}} \times T}$ and a value matrix $\mathbf{E} = (\mathbf{e}_1, \ldots, \mathbf{e}_T) \in \mathbb{R}^{d_{\text{out}} \times T}$, and a weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ which transforms input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ to output $S_2(\mathbf{x}) \in \mathbb{R}^{d_{\text{out}}}$ as:

$$S_2(\mathbf{x}) = \mathbf{W}_0 \mathbf{x} + \mathit{Attention}(\mathbf{X}, \mathbf{E}, \mathbf{x}) \tag{4}$$

Proof:

$$S_1(\mathbf{x}) = \mathbf{W}\mathbf{x} = \mathbf{W}_0 \mathbf{x} + \sum_{t=1}^{T} \mathbf{e}_t \otimes \mathbf{x}_t \cdot \mathbf{x} = \mathbf{W}_0 \mathbf{x} + \mathit{Attention}(\mathbf{X}, \mathbf{E}, \mathbf{x})$$

Remarks

Remark 1

Nothing is "forgotten":
The entire life of an NN is recorded and stored as a key matrix X and value matrix E. Roughly speaking, the only limitation of the model's capability to “remember” something is the limitation of the retrieval process

Remark 2

Unlimited memory size is not necessarily useful:
Systems which store everything in memory by increasing its size for each new event (S_2) are not necessarily better than those with a fixed size storage (S_1).

Remark 3

Non-Uniqueness:
The expression of a linear layer as an attention system is not unique.

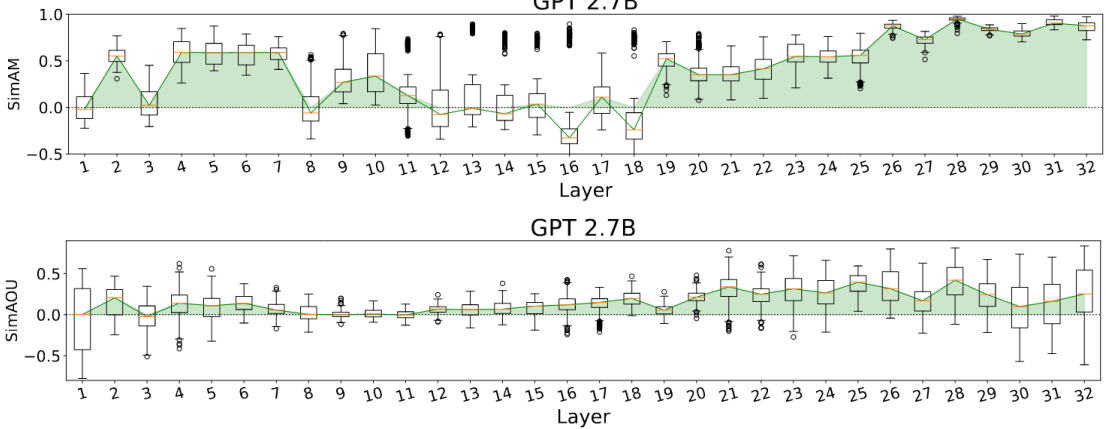
Exeperiment to compare Fine-tuning and ICL

- Fine-tune model using the same examples as in ICL;
- Use the following quality criteria:
  - Recall to Finetuning Predictions (**Rec2FTP**): percentage of queries that fine-tuning and ICL correctly predicts among queries that fine-tuning predicts correctly, but zero-shot not.
  - Similarity of Attention Output Updates (**SimAOU**): cosine between updates of output representations after fine-tuning and ICL.
  - Similarity of Attention Map (**SimAM**): cosine between attention maps before softmax operations.

Experiment Results

Model	Task	Rec2FTP	SimAOU	Random SimAOU	SimAM	ZSL SimAM
GPT 1.3B	CB	91.67	0.189	0.004	0.386	0.152
	SST2	86.32	0.128	0.003	0.608	0.555
	SST5	70.16	0.173	0.004	0.430	0.391
	Subj	84.39	0.070	0.004	0.504	0.378
	MR	92.14	0.188	0.003	0.513	0.398
	AGNews	85.41	0.155	0.003	0.536	0.152
GPT 2.7B	CB	100.00	0.184	-0.001	0.362	0.228
	SST2	93.87	0.113	0.003	0.687	0.687
	SST5	74.32	0.142	0.001	0.411	0.380
	Subj	90.46	0.100	0.004	0.375	0.346
	MR	95.44	0.120	0.001	0.346	0.314
	AGNews	87.48	0.210	-0.003	0.305	0.172

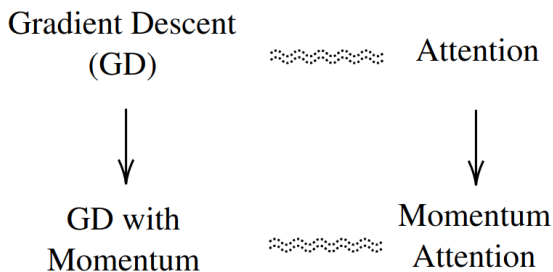
Experiment Results



Attention with momentum

Since we have established a relation of Attention layers with SGD-like optimization, we can plug momentum into attention:

$$\text{MoAttn}(\mathbf{V}, \mathbf{K}, \mathbf{v}) = \text{Attention}(\mathbf{V}, \mathbf{K}, \mathbf{v}) + \sum_{i=1}^{N-1} \lambda_{EMA}^{N-i} \mathbf{v}_i.$$



Attention with momentum: results

Model	Train_{1024}	Valid_{256}	Valid_{512}	Valid_{1024}
Transformer	17.61	19.50	16.87	15.14
$\text{Transformer}_{\text{MoAttn}}$	**17.55**	**19.37**	**16.73**	**15.02**

Table 4: Perplexity on the training set and validation sets with different input lengths for language modeling. Applying momentum to attention introduces a consistent perplexity improvement compared with the vanilla Transformer.

Model	SST5	IMDB	MR	CB	ARC-E	PIQA	Average
Transformer	25.3	64.0	61.2	43.9	48.2	68.7	51.9
$\text{Transformer}_{\text{MoAttn}}$	**27.4**	**70.3**	**64.8**	**46.8**	**50.0**	**69.0**	**54.7**

Table 5: Accuracy on six in-context learning tasks. Introducing momentum into attention improves the accuracy of the vanilla Transformer by 2.8 on average.

Learning to learn by gradient descent by gradient descent

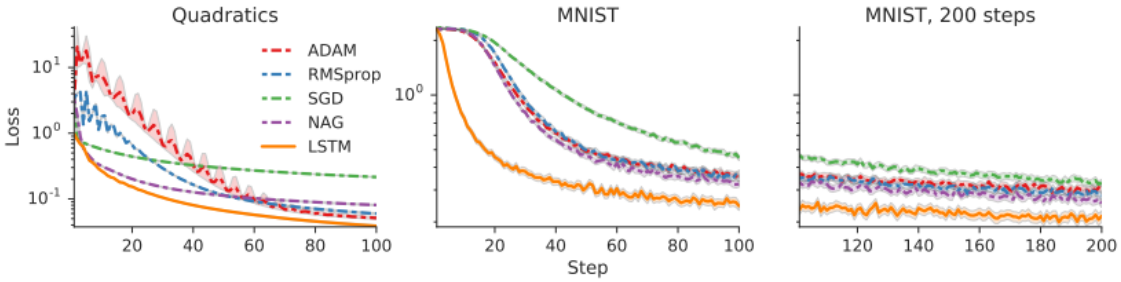
Idea: consider optimization T as a differential function:

$$T(\theta) = \text{LSTM}(\theta).$$

Optimization problem:

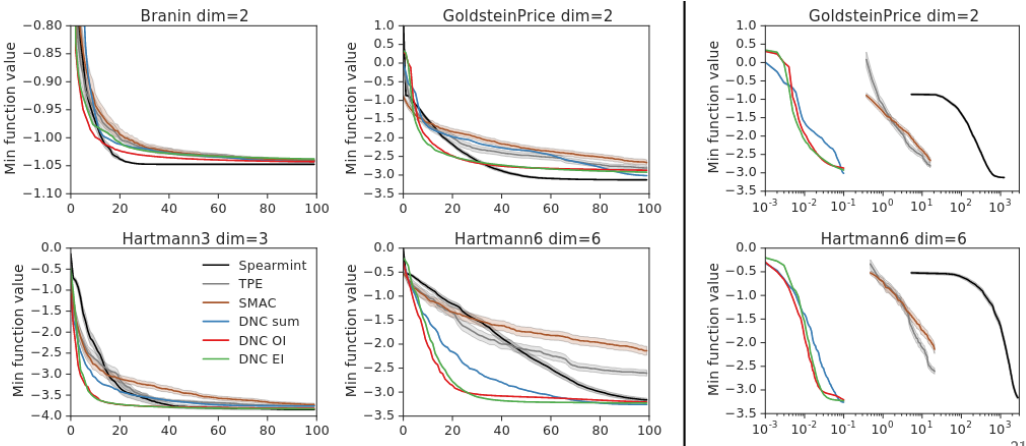
$$\sum_{t=t_0}^{t_\eta} L\left(T^t(\theta_{t_0})\right) \to \max.$$

LSTM has a small number of parameters, it shares parameters for each metaparameters.



Learning to Learn without Gradient Descent by Gradient Descent

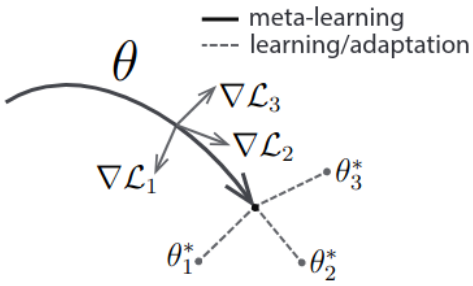
- Using the same idea, but for hyperparameters optimization (alternative to GP)
- Can be used with a combination of different optimization functions (e.g. EI)
- Does not require gradient at the «test» time



Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

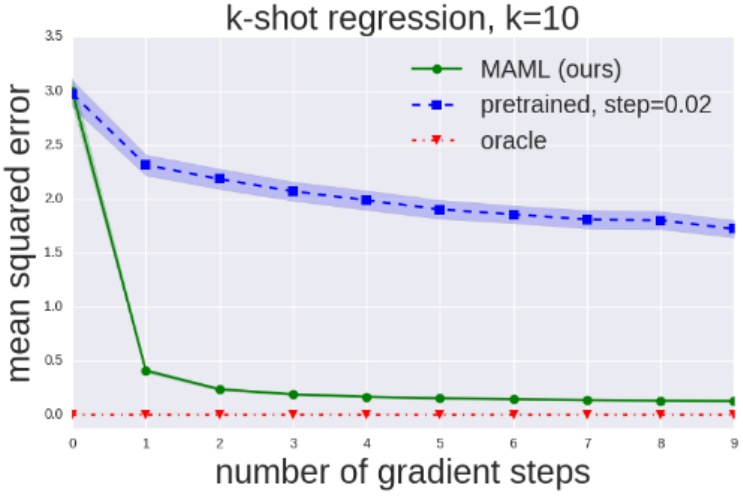
Algorithm 1 Model-Agnostic Meta-Learning

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**

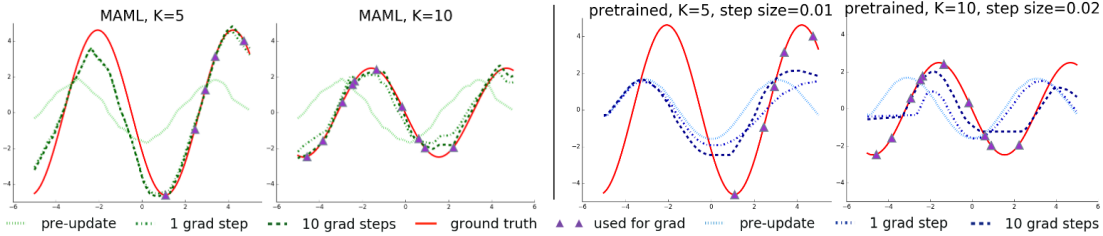


2

Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks



Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks



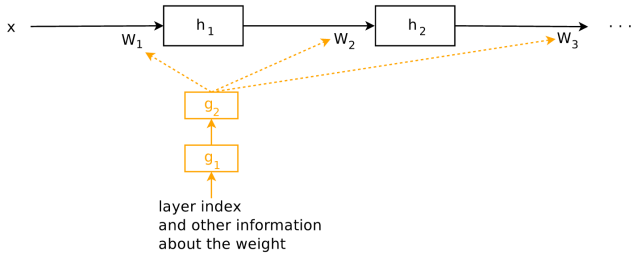
Hypernetworks

Definition

Given a set Λ.
Hypernetwork is a parametric mapping from Λ to set $\mathbb{R}^n$ of the model **f** parameters:

$$G : \Lambda \times \mathbb{R}^u \to \mathbb{R}^n,$$

where $\mathbb{R}^u$ is a set of hypernetwork parameters.



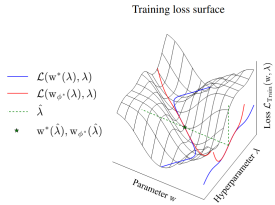
Ha et al., 2016

Stochastic Hyperparameter Optimization through Hypernetworks

$$\mathrm{E}_{\lambda}\left(-\log\ p(\mathfrak{D}|\mathbf{w}(\lambda)) + \lambda||\mathbf{w}(\lambda)||_2^2\right) \rightarrow \min$$

Theorem

Sufficiently powerful hypernetworks can learn continuous best-response functions, which minimizes the expected loss for all hyperparameter distributions with convex support.



Lorraine et al., 2016

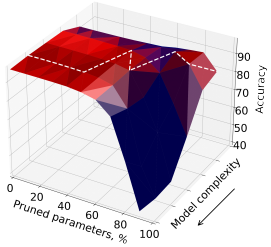
Deep learning model selection with parametric complexity control

Theorem (Grebenkova, 2023)

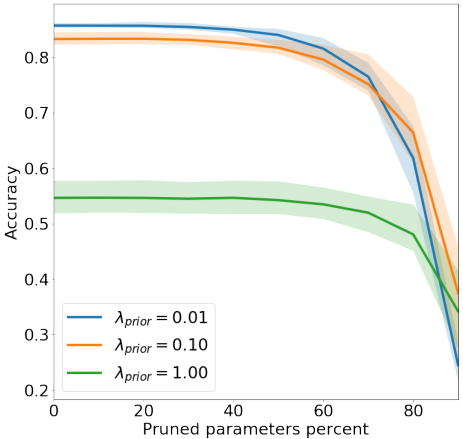
Hypernetowkr allows to learn not only the best-response functions, but also the statistical properties of the models.

Corollary

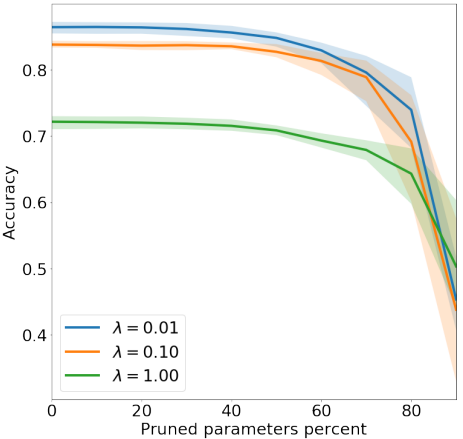
We can control the metaparameters, such as pruning rate or model complexity.



Example: CIFAR-10



CNN



CNN with hypernetwork

References

- Schmidhuber, Jürgen. "A neural network that embeds its own meta-levels."IEEE International Conference on Neural Networks. IEEE, 1993.
- Dougal Maclaurin et. al, Gradient-based Hyperparameter Optimization through Reversible Learning, 2015
- Andrychowicz M. et al. Learning to learn by gradient descent by gradient descent //Advances in neural information processing systems. – 2016. – С. 3981-3989.
- Chen Y. et al. Learning to learn without gradient descent by gradient descent //International Conference on Machine Learning. – PMLR, 2017. – С. 748-756.
- Ha D., Dai A., Le Q. V. Hypernetworks //arXiv preprint arXiv:1609.09106. – 2016.
- Lorraine J., Duvenaud D. Stochastic hyperparameter optimization through hypernetworks //arXiv preprint arXiv:1802.09419. – 2018.
- Гребенькова О. С., Бахтеев О. Ю., Стрижов В. В. Вариационная оптимизация модели глубокого обучения с контролем сложности //Информатика и её применения. – 2021. – Т. 15. – №. 1. – С. 42-49.
- Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks."International conference on machine learning. PMLR, 2017.
- Olga Grebenkova, Oleg Bakhteev, Vadim Strijov , Deep Learning Model Selection With Parametric Complexity Control , ICAART 2023.
- Dai, Damai, et al. "Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers."arXiv preprint arXiv:2212.10559 (2022).