BessaGroup Mini-conference #1

March 18, 2022

Ozgur Taylan Turan

Expected Loss of Model-Agnostic Meta-Learning

- Meta-Learning
- MAML vs Biased Ridge
- Some Results/Conclusions
- What is next?

Intro

Learning

- Task $\rightarrow f : \mathbf{x} \mapsto y$
- Training experience $\rightarrow \mathcal{Z} = \{\mathbf{x}_i, y_i\}_{i=0}^{N}$
- Error measure $\rightarrow \mathcal{L} := \sum_j^M (\mathcal{M}_j - y_j)^2$

Learning-to-learn

- Family of Tasks $\rightarrow \{f_k : \mathbf{x} \mapsto y\}_{k=1}^{K}$
- Training experience for $f_k \rightarrow \mathcal{Z}_k$
- Error measure for each task $\rightarrow \mathcal{L}_k$

- Learning a function vs learning a functional (space of functions!)

Intro

Learning

- Task $\rightarrow f : \mathbf{x} \mapsto y$
- Training experience $\rightarrow \mathcal{Z} = \{\mathbf{x}_i, y_i\}_{i=0}^{N}$
- Error measure $\rightarrow \mathcal{L} := \sum_j^M (\mathcal{M}_j - y_j)^2$

Learning-to-learn

- Family of Tasks $\rightarrow \{f_k : \mathbf{x} \mapsto y\}_{k=1}^{K}$
- Training experience for $f_k \rightarrow \mathcal{Z}_k$
- Error measure for each task $\rightarrow \mathcal{L}_k$

- Learning a function vs learning a functional (space of functions!)

Intro

Learning

- Task $\rightarrow f : \mathbf{x} \mapsto y$
- Training experience $\rightarrow \mathcal{Z} = \{\mathbf{x}_i, y_i\}_{i=0}^{N}$
- Error measure $\rightarrow \mathcal{L} := \sum_j^M (\mathcal{M}_j - y_j)^2$

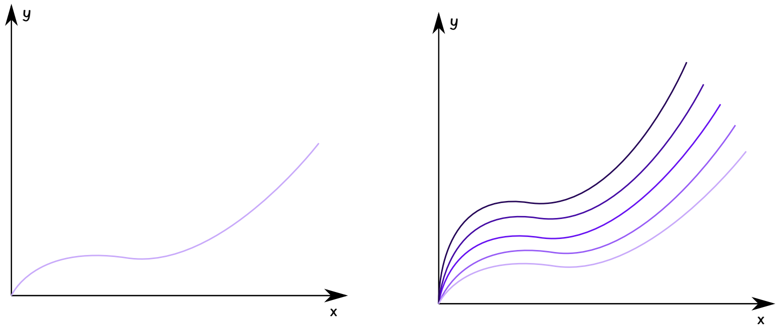
Learning-to-learn

- Family of Tasks $\rightarrow \{f_k : \mathbf{x} \mapsto y\}_{k=1}^{K}$
- Training experience for $f_k \rightarrow \mathcal{Z}_k$
- Error measure for each task $\rightarrow \mathcal{L}_k$

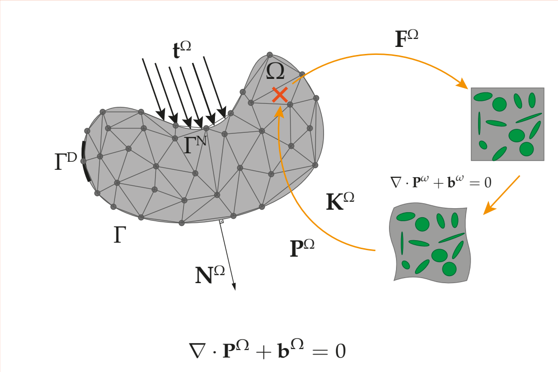
- Learning a function vs learning a functional (space of functions!)

Intro

Learning vs Learning-to-learn

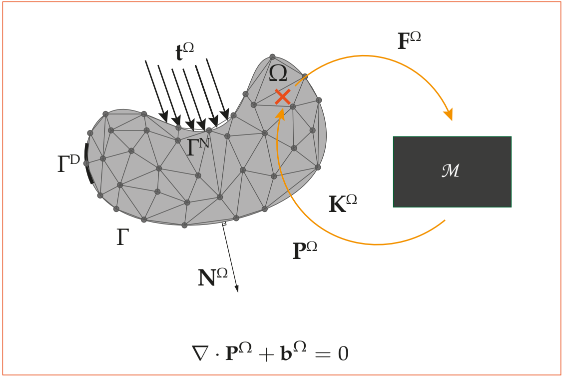


But for why bother?



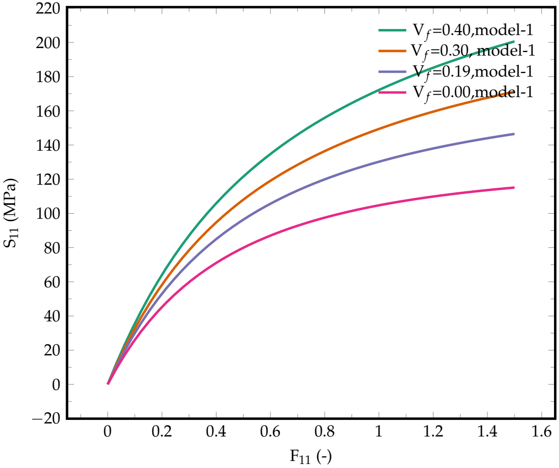
- Multi-scale composite modeling
- Computational homogenization (nested FEA)
- Computational expense is enormous
- ....
- Currently: $\mathcal{M}(\mathbf{F}^\Omega, \cdot, \cdot)$
- Computationally a material: $\mathbf{F}^\Omega \mapsto \mathbf{P}^\Omega$
- Why not exploit the similarities?

But for why bother?



- Multi-scale composite modeling
- Computational homogenization (nested FEA)
- Computational expense is enormous
- ....
- Currently: $\mathcal{M}(\mathbf{F}^{\Omega}, \cdot, \cdot)$
- Computationally a material: $\mathbf{F}^{\Omega} \mapsto \mathbf{P}^{\Omega}$
- Why not exploit the similarities?

For Example

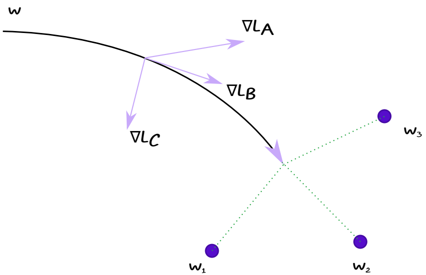


- Focus on: $\mathcal{M}(\mathbf{F}^{\Omega})$
- Treat other descriptors as different tasks

How to solve this problem?

- Pool of methods and algorithms!
- One really famous and one promising but "overlooked" method!

MAML[1]

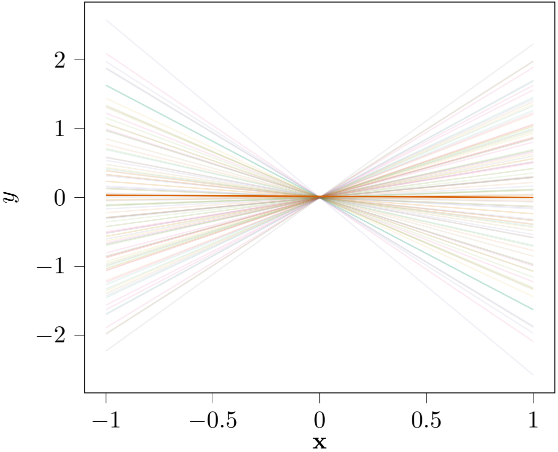


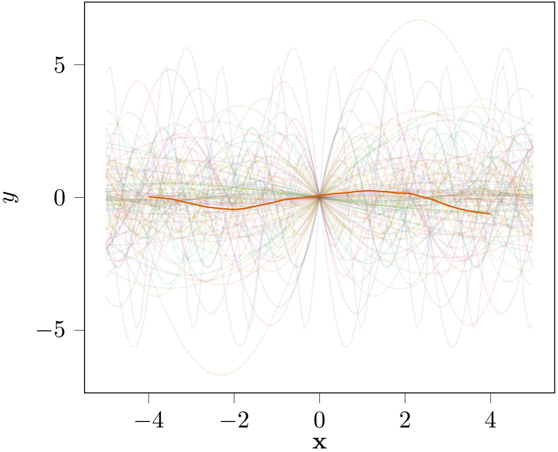
- Sample Tasks
- Sample training experiences from that tasks
- Check the possible loses
- Take average step

For a model $\mathcal{M}$ parametrized by $(\mathbf{w})$

[1] C. Finn, P. Abbeel, and S. Levine (2017). "Model-agnostic meta-learning for fast adaptation of deep networks". In: *34th International Conference on Machine Learning, ICML 2017* 3, pp. 1856–1868. arXiv: 1703.03400

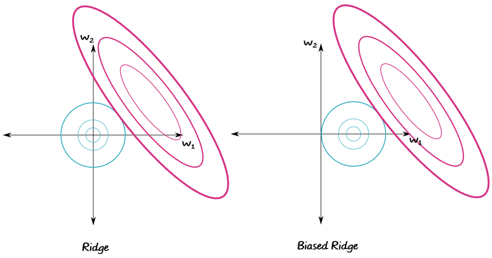
MAML[1]





[1] C. Finn, P. Abbeel, and S. Levine (2017). "Model-agnostic meta-learning for fast adaptation of deep networks". In: *34th International Conference on Machine Learning, ICML 2017* 3, pp. 1856–1868. arXiv: `1703.03400`

Biased Ridge[1]



- Sample Tasks
- Sample training experiences from that tasks
- adjust the bias

For a model $\mathcal{M}$ parametrized by $(\mathbf{w})$ minimize $\mathcal{L} + \lambda||\mathbf{w} - \mathbf{h}||_2^2$

[1] G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil (2018). "Learning to learn around a common mean". In: *Advances in Neural Information Processing Systems* 2018-Decem.NeurIPS, pp. 10169–10179. ISSN: 10495258

Biased Ridge[1]

- Sample Tasks
- Sample training experiences from that tasks
- adjust the bias

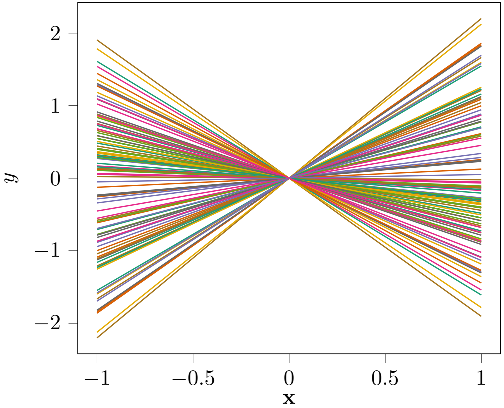
Kernel version: minimize $\mathcal{L} + \lambda||\mathcal{M}||^2_{\mathcal{H}_k}$

[1] G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil (2018). "Learning to learn around a common mean". In: *Advances in Neural Information Processing Systems* 2018-Decem.NeurIPS, pp. 10169–10179. ISSN: 10495258

Problem Setting-1

Linear Problem

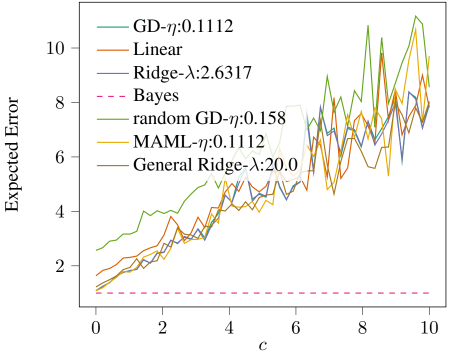
- $\mathbf{a} \in \mathbb{R}^d \to p_{\mathcal{T}} \sim \mathcal{N}(m\mathbf{1}, c\mathbf{I})$
- $\mathbf{x} \in \mathbb{R}^d \to p_{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, k\mathbf{1})$
- $\varepsilon \sim \mathcal{N}(0, \sigma^2)$
- $y = \mathbf{a}^{\mathsf{T}}\mathbf{x} + \varepsilon \quad \in \mathbb{R}$
- $\mathcal{Z} := ((x_i, y_i))_{i=1}^{N}$
- $\hat{\mathcal{M}} \to$ an estimator trained with $\mathcal{Z}$



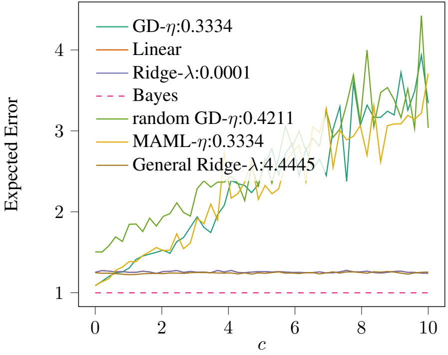
Expected Error for an estimator: $\mathcal{E} := \int \int \int (\hat{\mathcal{M}} - y)^2 p(x,y)\,dx\,dy\,p_{\mathcal{Z}}\,d\mathcal{Z}\,p_{\mathcal{T}}\,d\mathcal{T}$

Most Interesting Results

Limit the number of gradient steps for adaptation to 1 and other parameters regarding the problem is defaulted to 1 as well.



N=1



N=10

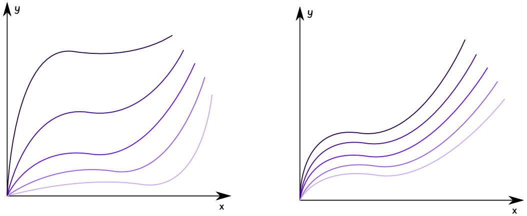
Most Interesting Results

Only consider the $c = [0, 1]$

	number of gradient steps									
	1	2	3	4	5	6	7	8	9	10
MAML	1.21	**1.19**	1.20	1.21	1.23	1.24	1.27	1.33	1.46	1.75

Conclusions

- Benefit of MAML only for $p_{\mathcal{T}}$ with small variance...
- Number of gradient steps taken is a clear regularizing effect...
- Not, a huge gain though???



Conclusions

- Benefit of MAML only for $p_{\mathcal{T}}$ with small variance...
- Number of gradient steps taken is a clear regularizing effect...
- Not, a huge gain though???

Method Development

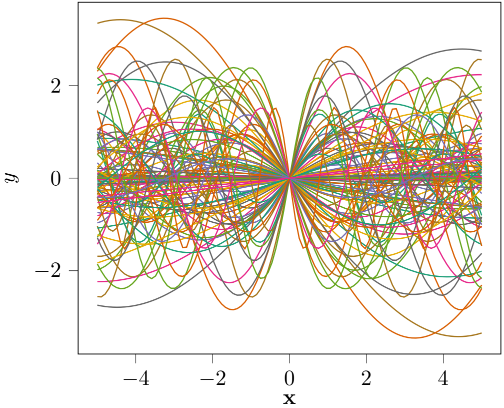
- Can we include additional bias for the functional norm in RKHS?
- Latent Variable Models for detecting underlying descriptors automatically?

Application

- Bit by bit start using real-data for experimentation!

Thanks!

Problem Setting-2



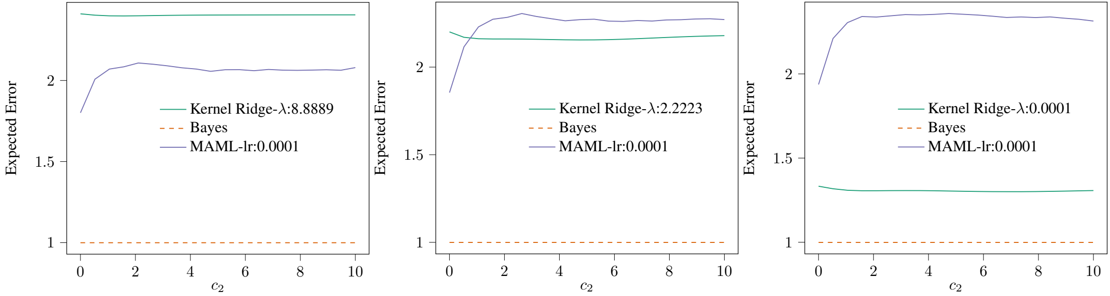
Nonlinear Problem

- $\mathbf{a} \in \mathbb{R}^d \rightarrow p_{\mathbf{a}} \sim \mathcal{N}(\mathbf{1}, c_1\mathbf{I})$
- $\boldsymbol{\phi} \in \mathbb{R}^d \rightarrow p_{\boldsymbol{\phi}} \sim \mathcal{N}(\mathbf{0}, c_2\mathbf{I})$
- $\mathbf{x} \in \mathbb{R}^d \rightarrow p_x \sim \mathcal{N}(\mathbf{0}, k\mathbf{1})$
- $\varepsilon \sim \mathcal{N}(0, \sigma^2)$
- $y = \mathbf{a}^{\mathsf{T}}\mathrm{sin}(\mathbf{x} + \phi) + \varepsilon \quad \in \mathbb{R}$
- $\mathcal{Z} := ((x_i, y_i))_{i=1}^N$
- $\hat{\mathcal{M}} \rightarrow$ an estimator trained with N training points

Expected Error for an estimator: $\mathcal{E} := \int\int\int (\hat{\mathcal{M}} - y)^2 p(x,y) dx dy p_{\mathcal{Z}} d\mathcal{Z} p_{\mathcal{T}} d\mathcal{T}$

Most Interesting Results

Every other parameter is defaulted to 1, and the adaptation steps used is 5.



N=1 N=10 N=50