Learning Step Size Controllers for Robust Neural Network Training

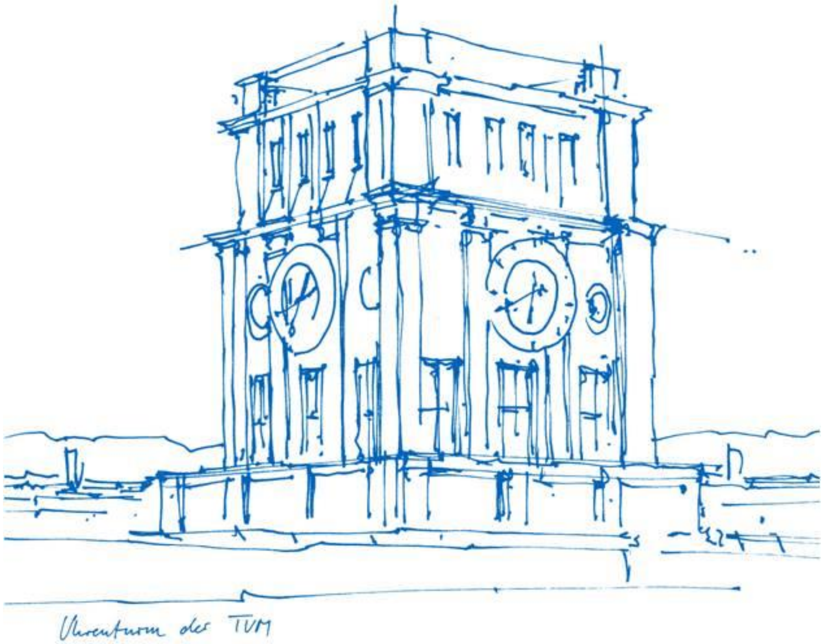
Technische Universität München

Fakultät für Informatik

Recent Trends in Automated Machine Learning

Zoom, 16th June 2021

Maximilian Karpfinger



Structure of the presentation

- Motivation
- Method
- Experiments & Results
- Conclusions

Motivation

- Many methods in DL rely on learning rate
- Cumbersome and sometimes hard to find
- Manual search, training schedule, learning rate decay, etc.
- Learn a control policy that adjusts the learning rate

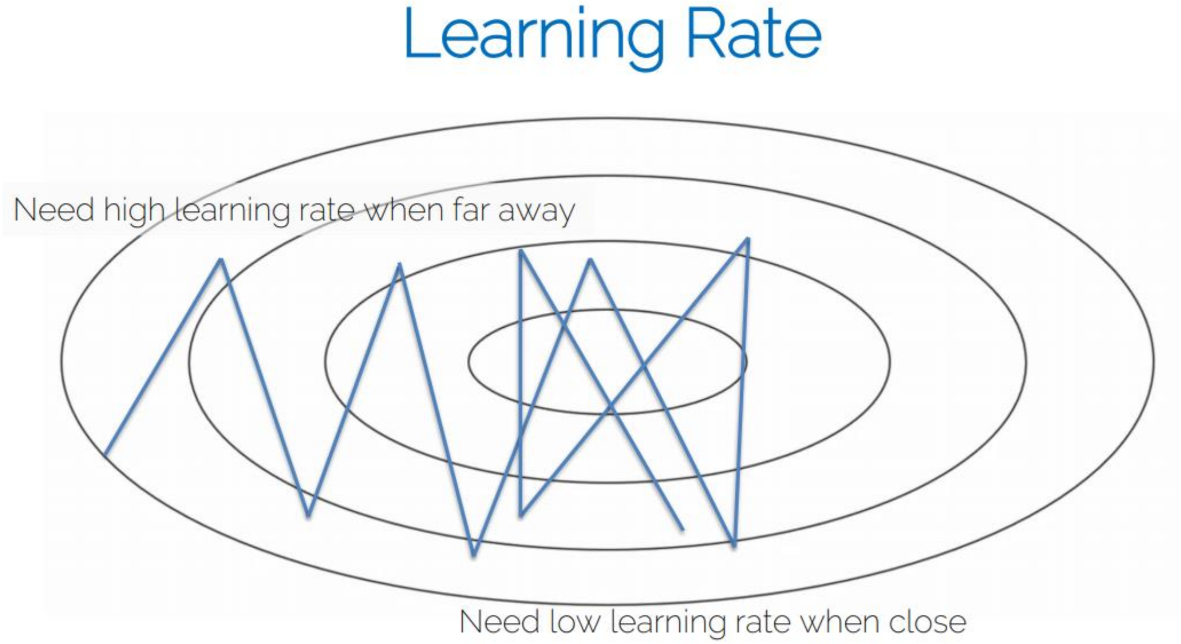


Image taken from I2DL 2020 lecture slides

Related Work

- Waterfall scheme
- Exponential scheme
- Power scheduling

→ All methods require additional hyperparameters

- TONGA, natural gradient

→ too expensive

Method

Background (what the controller wants to learn)

1.Find optimal weights

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} F(\boldsymbol{X}; \boldsymbol{\omega})$$

2.Function values F(x)

$$F(\boldsymbol{X}; \boldsymbol{\omega}) = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i; \boldsymbol{\omega})$$

3.Optimization operator T

$$\Delta\boldsymbol{\omega} = T(\nabla F, \boldsymbol{\rho}, \boldsymbol{\xi})$$

4.Weight update (SGD)

$$\boldsymbol{\omega} = \boldsymbol{\omega} - \alpha \nabla F$$

Learning a Controller

- Learn a policy using RL techniques
- Learn parameters $\boldsymbol{\theta}$ of controller

$$\xi = g(\phi; \theta)$$

- Policy will only set the parameters for the controller in the beginning of each training run
- Distribution of the policy for an optimal controller

$$\pi^*(\theta) = \arg\max_{\pi} \int p(\phi)\pi(\theta)r(g(\phi;\theta),\phi)d\phi d\theta$$

- REPS (policy update remains close to older step with KL divergence)

$$D_{KL}(\pi(\theta)||q(\theta)) \leq \epsilon$$

- Update step

$$\pi(\theta) \propto q(\theta)\exp(\frac{r(\theta)}{\eta})$$

State Features

Requirements:

- Shall be informative about current state
- Generalize across different tasks and architectures
- Computational complexity
- Memory requirements

Thoughts about the state features

Background:

Overall gradient composed of individual gradients

$$\nabla F = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i$$

Not all individual function values will improve by the same amount -> evaluating agreement likely to be informative

Use gradients to approximate change in function values by first order Taylor expansion

$\tilde{f}(x_i; \omega + \Delta\omega) = f(x_i; \omega) + \nabla f_i^T \Delta\omega$ (needed for first feature)

Introducing the Base State Features

1. Predictive change in function value ϕ_1

Variance of improvement of function values

$$\Delta\tilde{f}_i = \tilde{f}_i - f_i$$

$$\phi_1 = \log\left(Var(\Delta\tilde{f}_i)\right)$$

2. Disagreement of function values ϕ_2

Variance of current function values

$$\phi_2 = \log(Var(f(x_i;\omega)))$$

Mini Batch Setting

- Training set is split up
- Increases efficieny but also adds noise
- 2 countermeasures for both base features ϕ_1, ϕ_2

1. Discounted Average

- Running average for state features
- $\hat{\phi}_i \leftarrow \gamma\phi_i + (1-\gamma)\phi_i$

2. Uncertainty Estimate

- Estimate of noise level for state features
- $\hat{\phi}_{K+i} \leftarrow \gamma\hat{\phi}_{K+i} + (1-\gamma)(\phi_i - \hat{\phi}_i)^2$

Experiments & Results

- Datasets: MNIST & CIFAR-10
- SGD & RMSprop
- Learn parameters of controller with MNIST!
- Sample from “random“ CNNs and data-subsets
- Structure of the CNN c-p-c-p-c-r-c-s (convolution,pooling,rectified linear, softmax)

Back to Learning the Controller

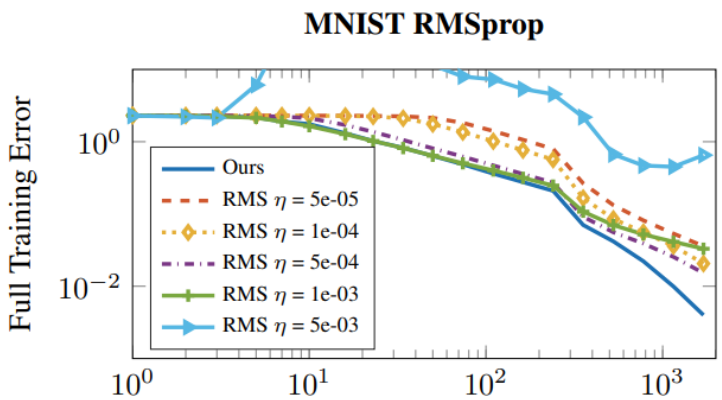
- Policy $\pi(\theta)$ initialized to a Gaussian with isotropic covariance (REPS, $\epsilon = 1$)
- In each learning iteration sample a parameter vector from the policy, a network and a training set
- Parametrized controller

$$g\left(\hat{\phi};\theta\right) = \exp(\theta^T \hat{\phi})$$

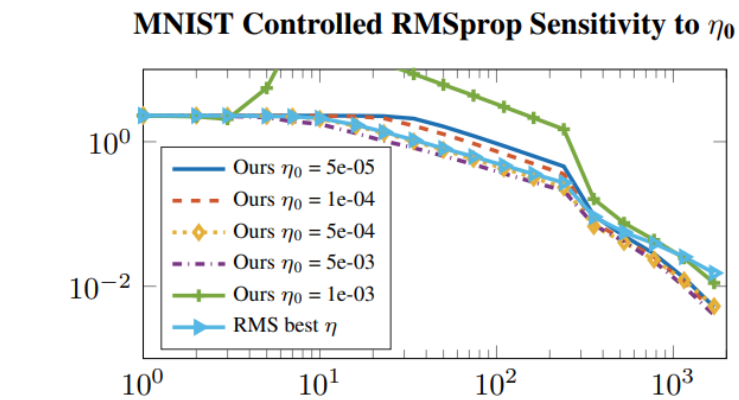
- Reward function for controller

$$r = -\frac{1}{S-1}\sum_{s=2}^{S}(\log(E_s) - \log(E_{s-1}))$$

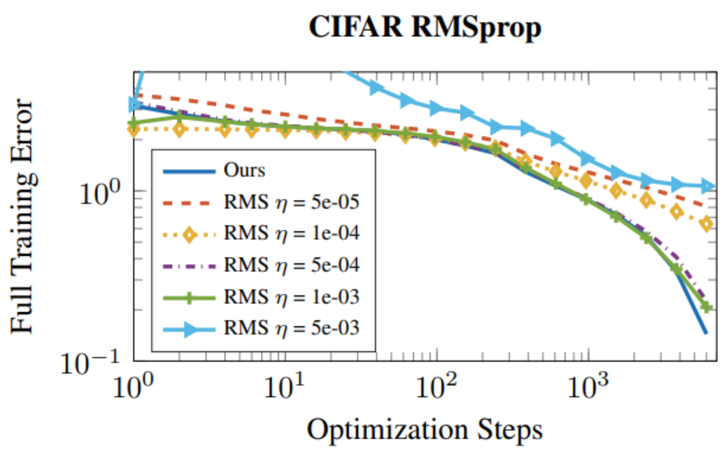
Results for RMSprop



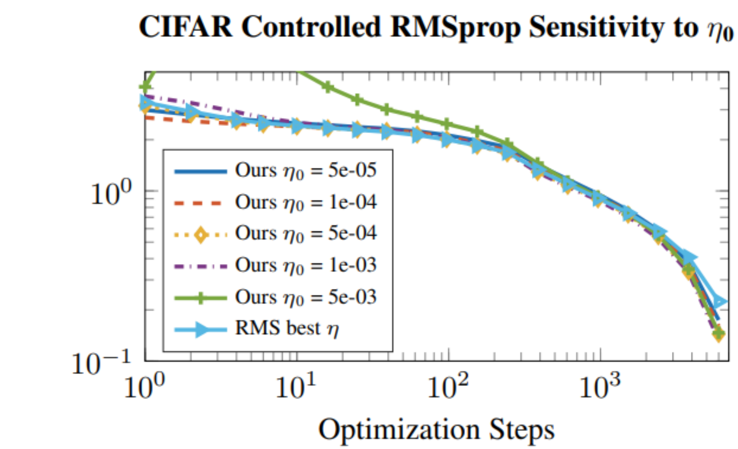
(a) Sensitivity analysis of static step sizes on MNIST.



(b) Sensitivity analysis of the proposed approach on MNIST.

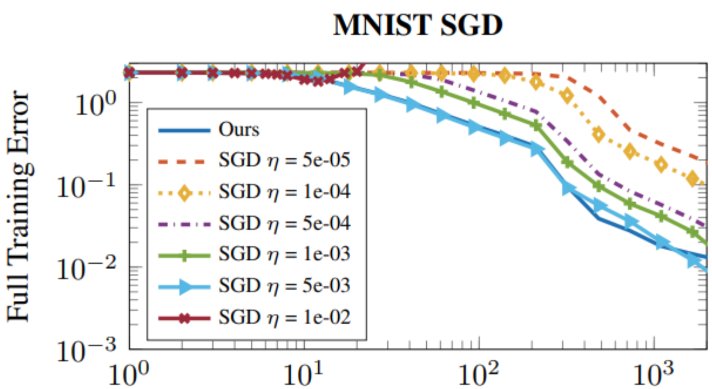


(c) Sensitivity analysis of static step sizes on CIFAR.

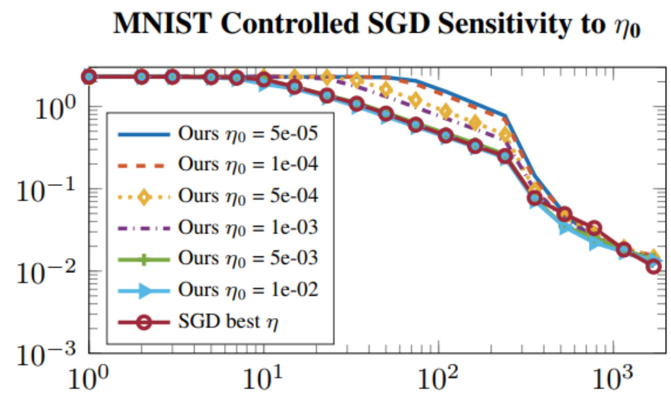


(d) Sensitivity analysis of the proposed approach on CIFAR.

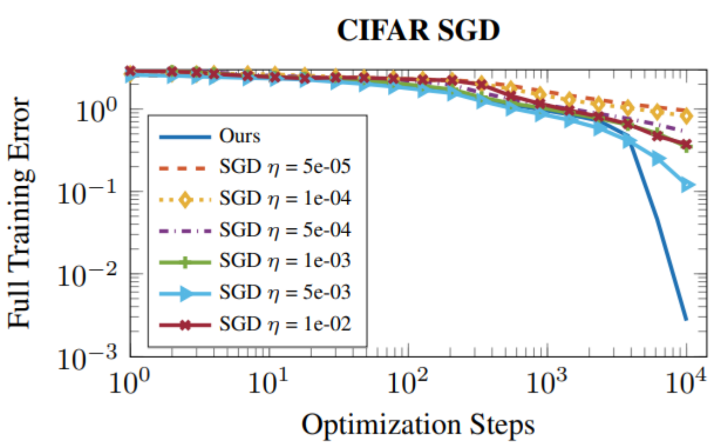
Results for SGD



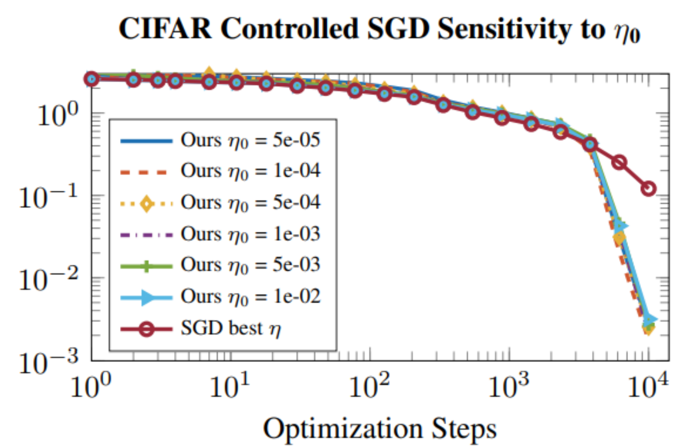
(a) Sensitivity analysis of static step sizes on MNIST.



(b) Sensitivity analysis of the proposed approach on MNIST.

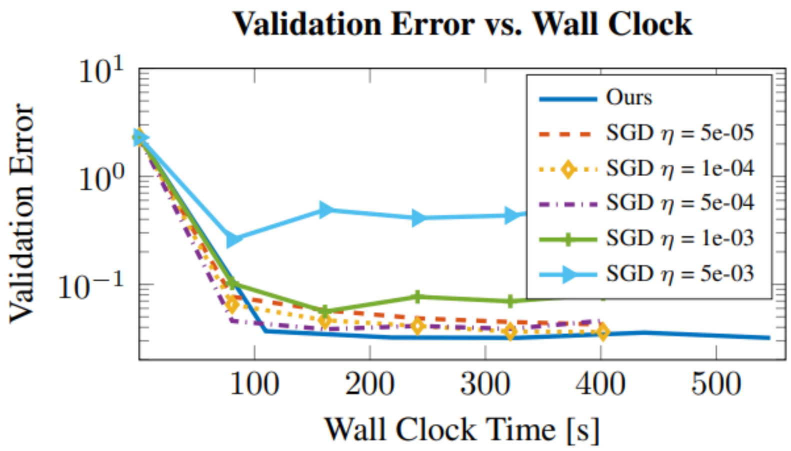


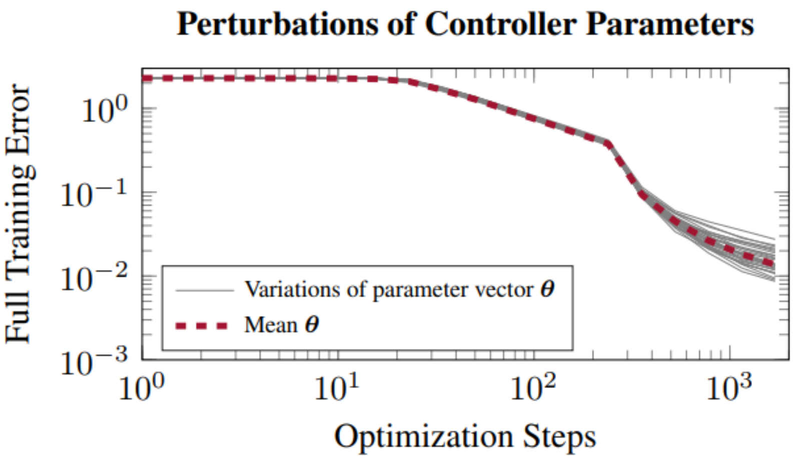
(c) Sensitivity analysis of static step sizes on CIFAR.



(d) Sensitivity analysis of the proposed approach on CIFAR.

Runtime & Robustness





- Small computational overhead of 36%
- six hours of training of the RL algorithm

- Learned controller robust to small changes
- Varied parameters in 20% range around learned values

Conclusion & Discussion

Strenghts

- Generalizes to larger networks (CNN) and full MNIST dataset
- Generalizes to different dataset (CIFAR-10)
- Robust to initial values of learning rate
- Small computational overhead (36%)

Weakness

- Can it generalize to different types of architectures? (RNN)
- Experimental section
- No comparison to other learning rate adapting techniques (learning rate decay)
- Black box view on learning rate (how does it change?)
- Cannot generalize across different optimizers

References

[1] Christian Daniel; Jonathan Taylor and Sebastian Nowozin 2016. Learning Step Size Controllers for Robust Neural Network Training
[2] REPS Peters, Mülling and Altun 2010
[3] TONGA Lex Roux, Bengio and Fitzgibbon 2012
[4] Waterfall scheme, Senior et. Al 2013
[5] Exponentiaol scheme, Sutton 1992

Questions?