AIBC2021

Application of Natural Gradient Descent on the Iris Classification Problem

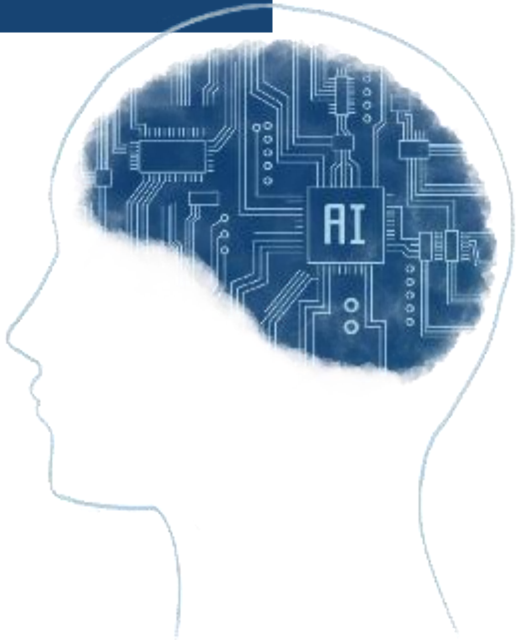
Shuhan Zheng

University of Tokyo

CSHA

NIP LAB

HUAWEI



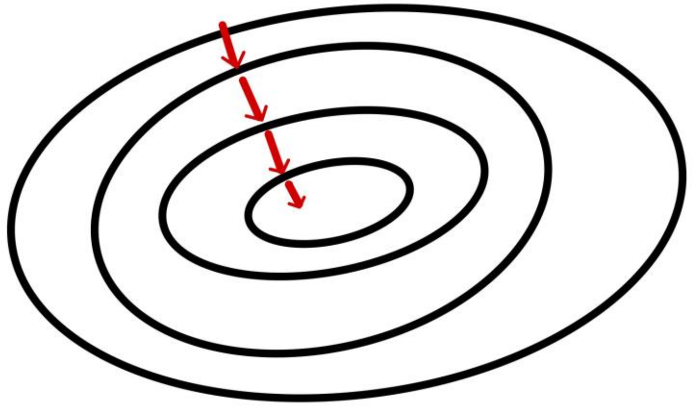
Outline

- Project background
- Implementation and Experiment setup
- Results
- Discussion

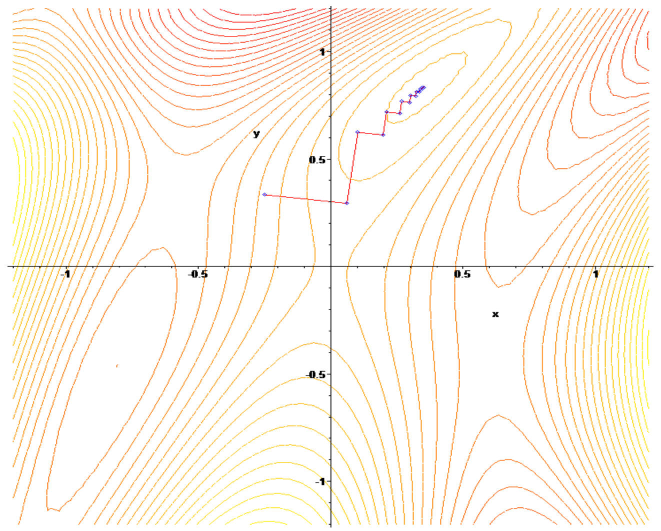
Project background

- To optimize network parameters, ordinary gradient descent update parameters in the Euclidian space by

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \frac{\partial l}{\partial \boldsymbol{\theta}}$$



√



An alternative approach?

Project background

We consider KL divergence as the metric (rather than Euclidean distance) between $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ in the distribution space. The optimization goal is:

$$d^* = \arg\min L(\theta + d) \qquad \text{with } KL[p(\theta)||p(\theta + d)] = c$$

The KL divergence can be approximated by the Taylor series:

$$KL[p(\theta)||p(\theta + d)] = \cancel{KL[p(\theta)||p(\theta)]} + \cancel{\left(\nabla_{\theta'} KL\left[p(\theta)\middle|\middle|p(\theta')\right]\right)^T d} + \frac{1}{2} d^T F d$$

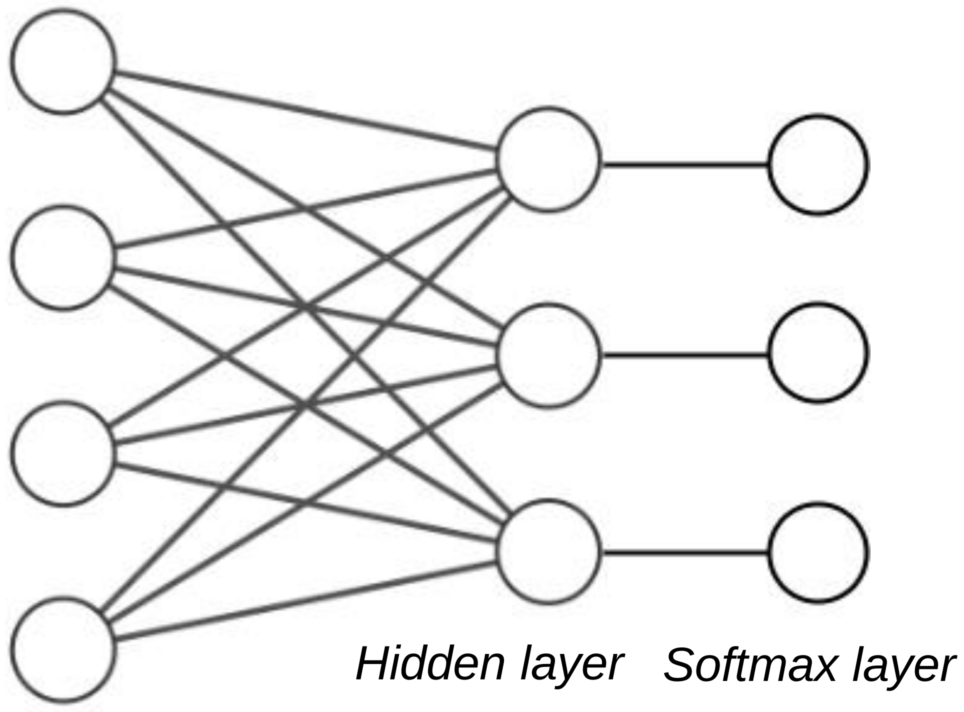
By applying Lagrange multiplier:

$$\begin{aligned} d^* &= \arg\min_{\mathrm{d}} (L(\theta + d) + \lambda(\underline{KL[P(\theta)||p(\theta + d)]} - c) \\ &= \arg\min_{\mathrm{d}} (L(\theta) + \nabla L(\theta)^T d + \underline{\tfrac{1}{2}\lambda d^T F d} - \lambda c) \end{aligned}$$

$$\frac{\partial}{\partial d}\left(L(\theta) + \nabla L(\theta)^T d + \frac{1}{2}\lambda d^T F d - \lambda c\right) = \nabla L(\theta) + \lambda F d = 0 \Longrightarrow d = -\frac{1}{\lambda} F^{-1} \nabla_\theta(\mathrm{L}(\theta))$$

Implementation & Experimental setup

- Neural network structure for the Iris classification problem



Input layer

We use cross entropy loss function:

$$loss(X,Y) = -\sum_{i=1}^{N} \ln\left(\frac{e^{w_{yi}\cdot x_i}}{\sum_{l=1}^{k} e^{w_l \cdot x_i}}\right)$$

N is the number of samples and k is the number of class.

For this task, one hidden layer is enough.

Implementation & Experimental setup

- The implemented optimization algorithm: Natural gradient descent with the empirical fisher information matrix

$$F(\theta) := \frac{1}{N}\sum_n \mathrm{E}_{\mathrm{p}_\theta(\mathrm{y}|\mathrm{x}_\mathrm{n})} \nabla_\theta \log p_\theta(y|x_n)\, \nabla_\theta \log p_\theta(y|x_n)^T$$

$$\Big\downarrow$$

$$\tilde{F}(\theta) := \frac{1}{N}\sum_n \nabla_\theta \log p_\theta(y_n|x_n)\, \nabla_\theta \log p_\theta(y_n|x_n)^T$$

y_n is a training label and not a sample from the model's predictive distribution $p_\theta(y|x_n)$.

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \tilde{F}(\boldsymbol{\theta})^{-1} \frac{\partial l}{\partial \boldsymbol{\theta}}$$

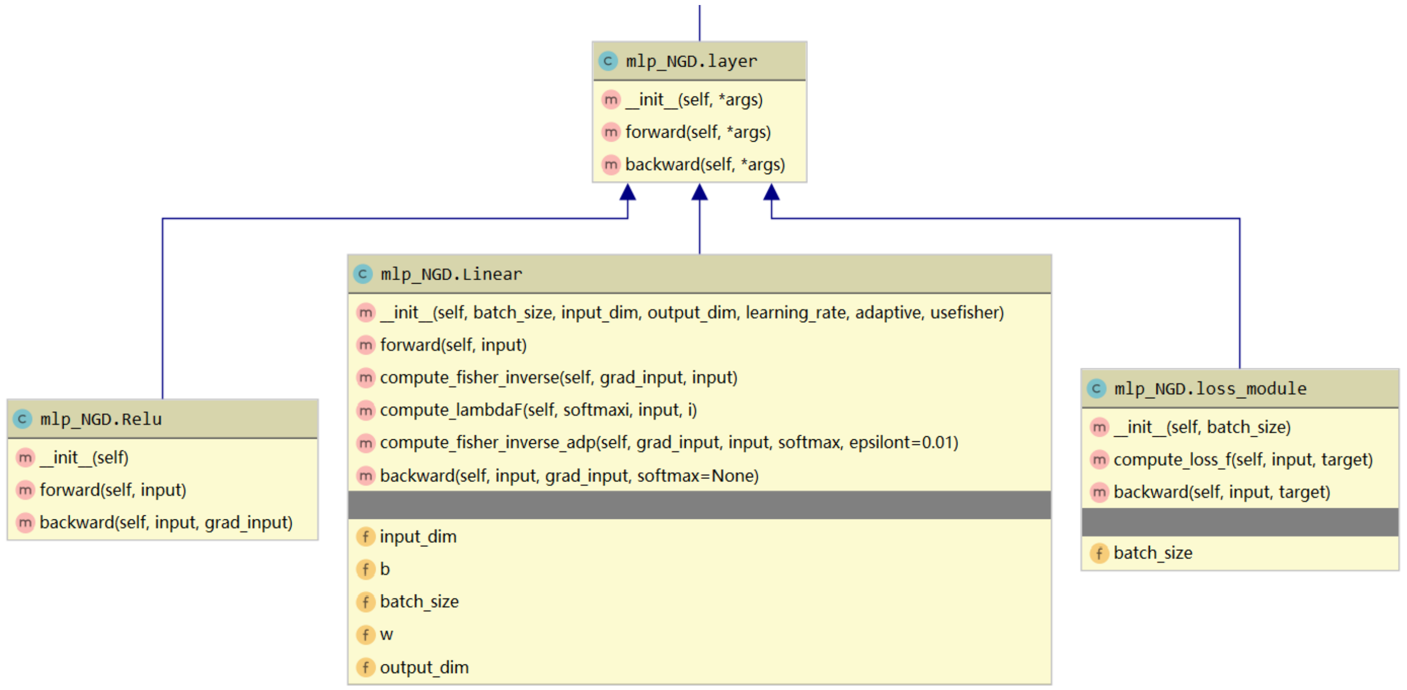
Optimization rule

Implementation & Experiment Setup

- Python implementation and Experiment Configuration

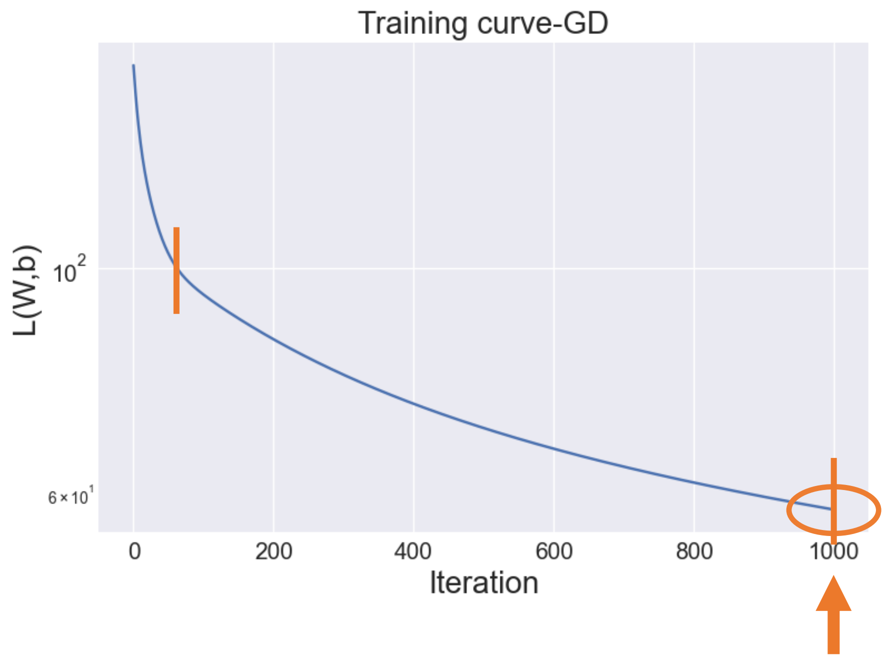
......

Training set size: 100 samples
Test set size: 50 samples
Batch size:100
Learning rate: 0.0025
Epoch number: 1000
CPU: Intel(R) Core(TM) i5-10210U
CPU @ 1.60GHz   2.11 GHz

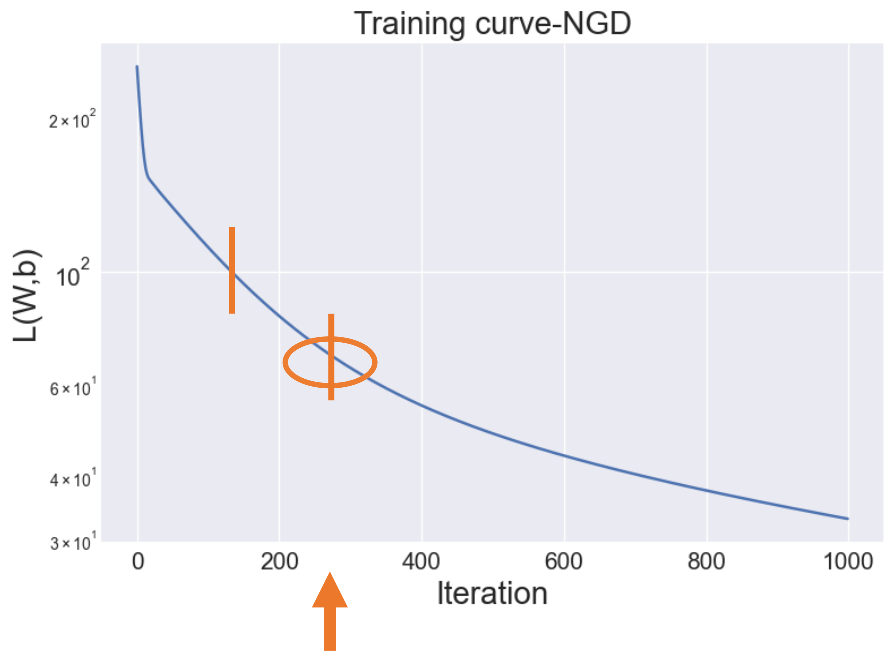

mlp_NGD.layer
__init__(self, *args)
forward(self, *args)
backward(self, *args)
mlp_NGD.Linear
__init__(self, batch_size, input_dim, output_dim, learning_rate, adaptive, usefisher)
forward(self, input)
compute_fisher_inverse(self, grad_input, input)
compute_lambdaF(self, softmaxi, input, i)
compute_fisher_inverse_adp(self, grad_input, input, softmax, epsilont=0.01)
backward(self, input, grad_input, softmax=None)
input_dim
b
batch_size
w
output_dim
mlp_NGD.Relu
__init__(self)
forward(self, input)
backward(self, input, grad_input)
mlp_NGD.loss_module
__init__(self, batch_size)
compute_loss_f(self, input, target)
backward(self, input, target)
batch_size

Results

- Natural gradient descent (NGD) shows faster convergence than the ordinary gradient descent (GD).



Average Test Accuracy: 0.78

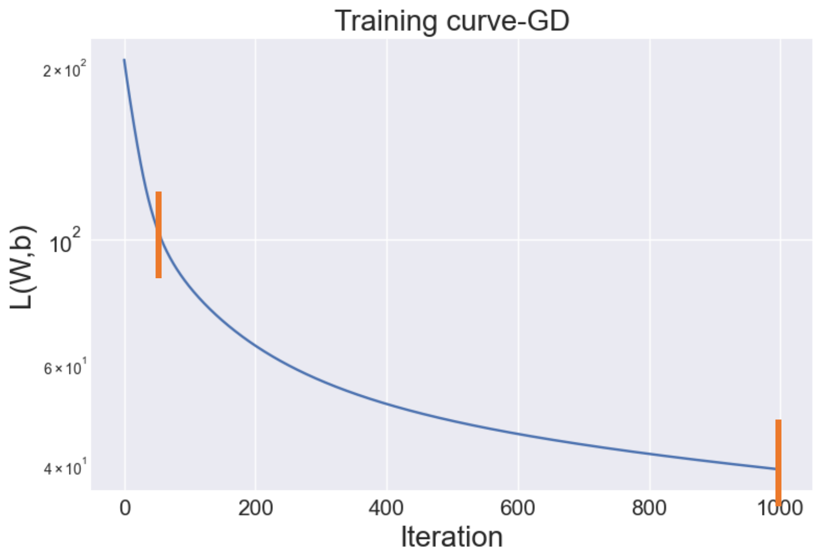


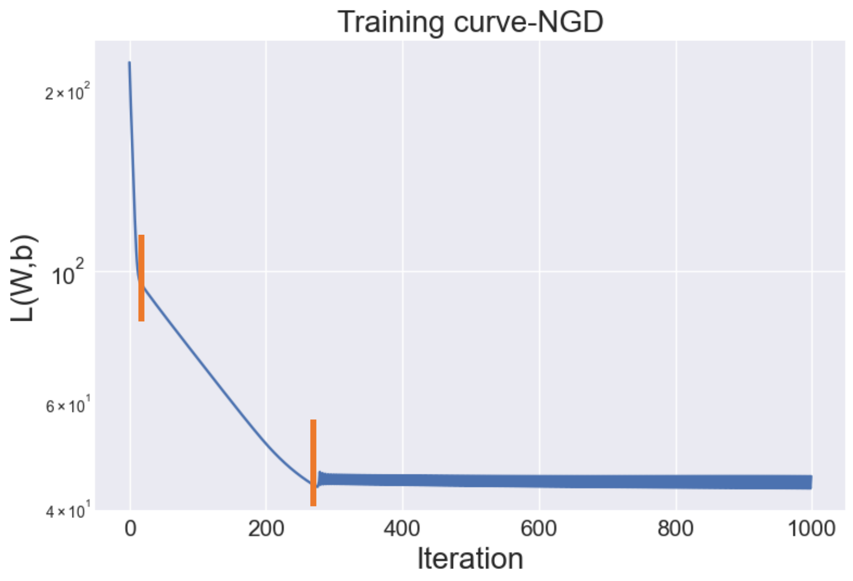
Average Test Accuracy: 0.88

Results

- Optimization method comparisons on the neural network with more layers (4x5,5x3)

learning_rate = 0.002



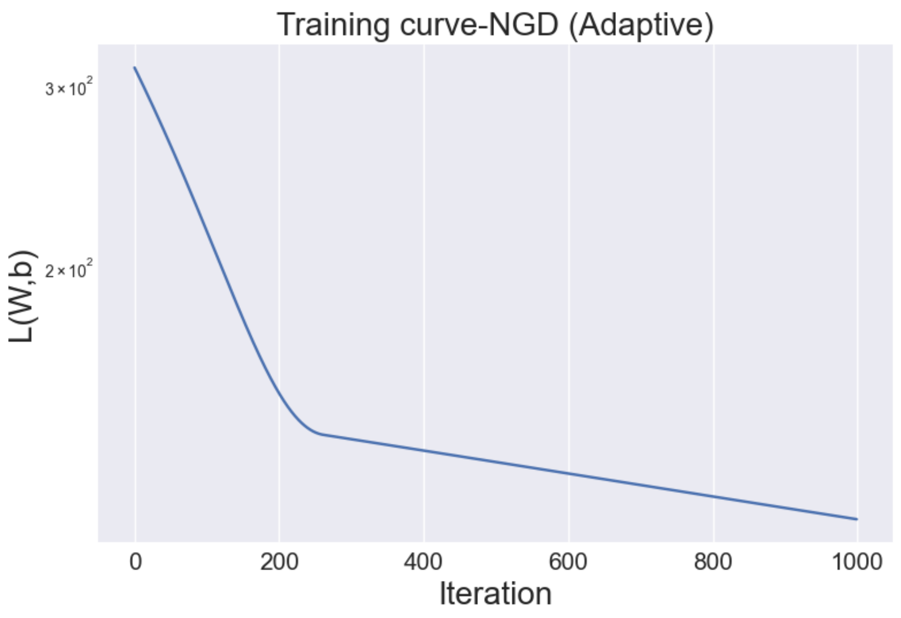


Discussion

- Implementation of Adaptive Natural Gradient Descent Method

$$\boldsymbol{G_t^{-1}} \approx (\left(1-\frac{1}{t+1}\right) G_{t-1} + \frac{1}{t+1} G_t)^{-1} \approx \left(1-\frac{1}{t+1}\right)\boldsymbol{G_{t-1}^{-1}} - \frac{1}{t+1}\boldsymbol{G_{t-1}^{-1}} G_t \boldsymbol{G_{t-1}^{-1}}$$

In our implementation, we update G_t^{-1} using every sample in a batch. We update the parameter when all samples have been visited.



Discussion

- Performance of empirical fisher(EF) approximation

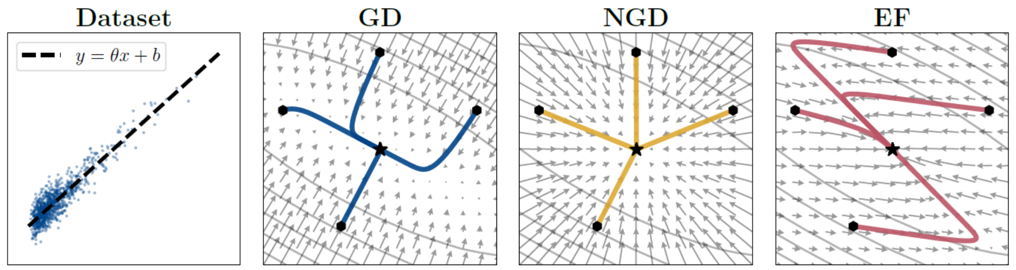


Figure 1: Fisher vs. empirical Fisher as preconditioners for linear least-squares regression on the data shown in the left-most panel. The second plot shows the gradient vector field of the (quadratic) loss function and sample trajectories for gradient descent. The remaining plots depict the vector fields of the natural gradient and the "EF-preconditioned" gradient, respectively. NGD successfully adapts to the curvature whereas preconditioning with the empirical Fisher results in a distorted gradient field.

Figuer1, Kunstner et al.

EF may fail in some cases.
Then why EF approach is still practical successful? Two reasons!

1. Empirical Fisher approaches the Fisher if the model "is a good fit for the data"
2. It follows the definition of a generalized Gauss-Newton matrix, making it an approximate curvature matrix in its own right.

Acknowledgment

Park, Hyeyoung, S-I. Amari, and Kenji Fukumizu. "Adaptive natural gradient learning algorithms for various stochastic models." *Neural Networks* 13.7 (2000): 755-764.

Amari, Shun-ichi, Ryo Karakida, and Masafumi Oizumi. "Fisher information and natural gradient learning in random deep networks." *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019.

Park, Hyeyoung, and Kwanyong Lee. "Adaptive natural gradient method for learning of stochastic neural networks in mini-batch mode." *Applied Sciences* 9.21 (2019): 4568.

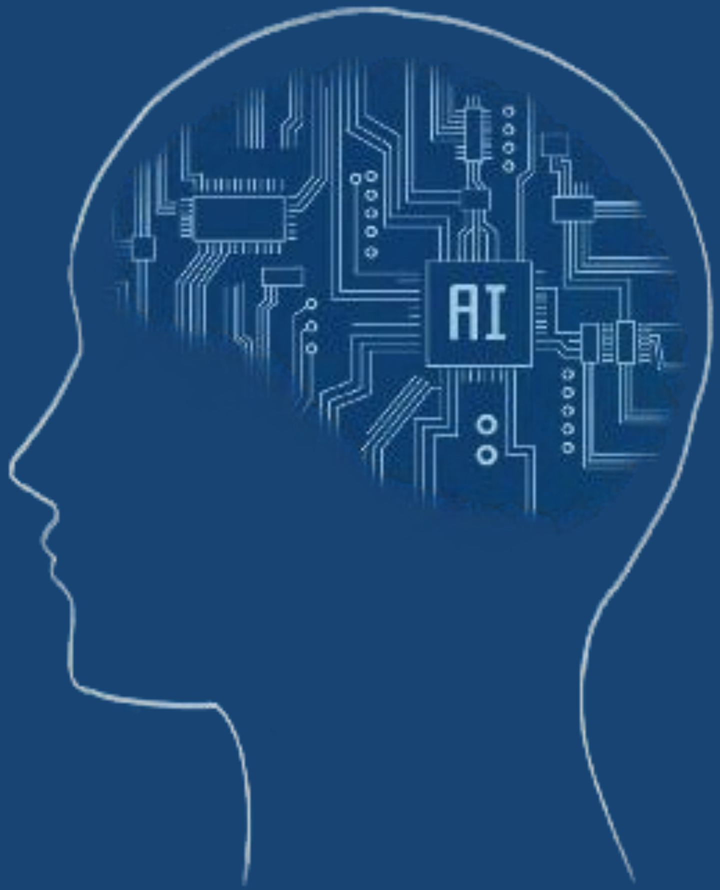
Kunstner, Frederik, Lukas Balles, and Philipp Hennig. "Limitations of the empirical fisher approximation for natural gradient descent." *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2019.

Say thanks to Xiaolong Zou and Xiaohan Lin for preparing tutorial sessions

Say thanks to the Organizers of AIBC 2021

AI and Brain Computation

Q & A



August 9 - August 22, 2021