Lecture 1 - Introduction, Optimization

Advanced Machine Learning

Miroslav Čepek, Zdeněk Buk, Rodrigo da Silva Alves, Vojtěch Rybář, **Petr Šimánek**

FIT CTU

23. 2. 2023

Contacts

Miroslav Čepek	miroslav.cepek@fit.cvut.cz
Zdeněk Buk	zdenek.buk@fit.cvut.cz
Rodrigo da Silva Alves	rodrigo.alves@fit.cvut.cz
Vojtěch Rybář	vojtech.rybar@fit.cvut.cz
Petr Šimánek	petr.simanek@fit.cvut.cz

- https://courses.fit.cvut.cz/NI-AML

Agenda

- What to expect
- Assessment rules, exam, organization
- Showcases
- Optimization in deep neural networks

Evaluation

- Homeworks (up to 25 points, 5 points each, min 12 points)
- Semestral Projects (up to 50 points, min 25 points)
- Oral Exam (up to 25 points, min 12 points)
- Additional points: 10 points for a blog post

Homeworks

- Finish experiments and explore implementation on a given theme.
- Submission is a piece of code solving the problem and possibly a short report.
- You will have 2 weeks to finish each homework.
- Target is two hours per homework.

Semestral Project

- Topics available on course webpage: https://courses.fit.cvut.cz/NI-AML/semestral_projects.html
- Each topic can be solved in collaboration by a group of up to 3 people.
- Create a group and pick a topic by March 17. Let us know by email. Topics are assigned on a first-come-first-served principle.
- Outcomes:
  - Mid-term presentation
  - Final presentation
  - Report
  - Implemenetation
  - **Optionally** blogpost

Exam

- Each of the lecturers will open $\sim$ 6 slots.
- You will pick one lecturer and register with him.
- Expect 15-20 minute chat **mostly** about topics given by particular lecturer. It may also include questions about your project.

Grades

- Final grades are based on grand total of your points from the semester and exam.
- Using standard conversion table:

Grade	Points	Evaluation in words
A	90 and more	excellent
B	80 to 89	very good
C	70 to 79	good
D	60 to 69	satisfactory
E	50 to 59	sufficient
F	less than 50	failed

What I assume you know?

- We assume:
  - knowledge of Python.
  - familiarity with ML libraries (Torch/Tensorflow).
  - understanding of machine learning/AI principles and basic techniques.

Physics Informed ML

- Can we solve difficult physical problems with deep learning?
- How do we predict chaos?
- How does knowledge of physics help us predict the weather?

Computer as Painter - Stable Diffusion

Text $\rightarrow$ Picture

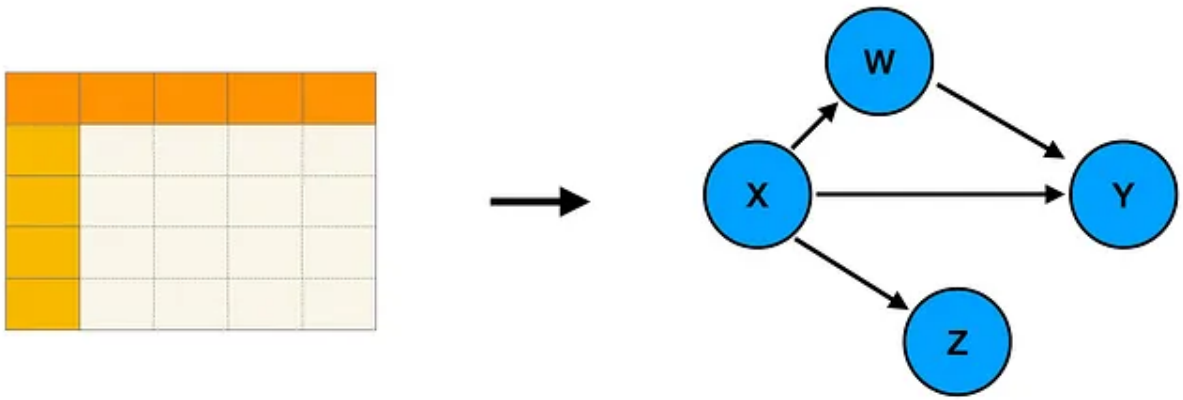
- explained
- other applications of denoising diffusion probabilistic models

You, have a cookie

Up-to-date research in recommender systems.

Causality

- Can we learn causal relations from observational data?



Drive like hell

- Learning to drive autonomous cars on FIT.

Alignment

- How can we avoid being completely annihilated by AI?

What is optimization in the context of Neural Nets?

What do we want from the optimization algorithm:

- converge most of the time
- converge with most initial weights
- converge fast
- generalize well
- be robust to small perturbations to the system
- small memory requirements
- no need to tune hyperparameters

Stochastic Gradient Descent

Deep neural net $f : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$.

$$f(x) = g_N \circ g_{N-1} \circ \cdots \circ g_1(x).$$

Where

$$g_j(x) = \sigma(W_j x + b_j), W_j \in \mathbb{R}^{d_x \times d_{j-1}}, b_j \in \mathbb{R}^{d_j}.$$

σ is the activation function (component-wise).

Supervised learning with loss L (Empirical risk minimization)

Stochastic Gradient Descent

Most common algorithm: Initialization: $W_1^0, \ldots, W_N^0$ Iterate for k $k = 0, 1, 2, \ldots$

$$W_j^{k+1} = W_j^k - \eta_k \nabla_{W_j} L(W_1^k, \ldots, W_N^k), j = 1, \ldots, N.$$

With some step sizes $\eta_0, \eta_1, \ldots$

Stochastic Gradient Descent

What do we want from the optimization algorithm:

- converge most of the time
- converge with most initial weights
- converge fast
- generalize well
- be robust to small perturbations to the system
- small memory requirements compared to what?
- no need to tune hyper-parameters

Momentum and adaptive gradient

- Often added to SGD.
- The goal is "not" to smooth convergence!
- SGD + Momentum oscillates too.
- Momentum allows larges batches.
- Adaptive gradient scales the gradients.
- Adam = momentum + adaptive gradient with the second moment (scales the gradient by gradient variance).

Standard goto algorithm:

- Adam/SGD + Nesterov Momentum for machine vision
- AdamW for NLP
- AdaGrad for recommenders

AdamW

- Adam often generalizes less than SGD. Why?
- Using Adam can hinder L2 regularization.
- The reason: Even the regularization term is "normalized".
- Solution: AdamW.

Generalization of SGD

CIFAR-10, 50k samples, trained with SGD

Architecture	n of parameters	Training loss	Test accuracy
MLP	1.2M	0	51%
Alexnet	1.4M	0	77%
Inception	1.65M	0	86%
Resnet	9M	0	88%

- Over-parameterized networks work better! - counter-intuitive
- No overfitting?
- This is not the case with many algorithms!
- Not understood well yet.

Zhang, 2017, Understanding Deep Learning Requires Rethinking Generalization.

Implicit Bias of SGD

- Linear NNs are driven to low-rank solutions.
- SGD finds a sparser solution.
- It seems like SGD implicitly finds "low complexity" optima.

Implicit Regularization of SGD

- Gradient descent finds a minimum that also minimizes not only the loss function but also its gradient.
- SGD optimizes a different function!
- SGD minimizes also the "variance" of mini-batch gradients.

Smith, 2021, On the Origin of Implicit Regularization in Stochastic Gradient Descent

Double Descent

Conventional thinking:

- Larger models are better.
- More data is better.
- Early stopping is good.

Model-wise/epoch-wise/sample-wise double descent.

Nakkiran, 2019, Deep Double Descent: Where Bigger Models and More Data Hurt.

2nd Order methods

- Minimization of 2nd order local approximation.
- Interesting also to inspect your model.
- BackPACK tool for computing Hessians and interesting other quantities.

Dangel, 2020, BackPACK: Packing more into Backprop

2nd order methods

What do we want from the optimization algorithm:

- converge most of the time
- converge with most initial weights
- converge fast
- generalize well - we don't know!
- be robust to small perturbations to the system - we don't know!
- small memory requirements
- no need to tune hyperparameters

2nd order methods

AdaHessian:

- Computes and stores only the diagonal of the Hessian
- Uses second-order momentum and other methods similarly as Adam
- Uses spatial averaging

Yao, 2020, ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning

Credit Assignment without Backpropagation

- Can we train without a backdrop?
- Backprop is very expensive and hard to parallelize
- Backprop is not biologically plausible - there is no such feedback in the brain
- Biologically-motivated:
  - asynchronous updating of weights at different layers of a network
  - reduced memory costs from having to store intermediate layer activation values
  - reduced synaptic wiring in the feedback path

The resulting computational efficiencies can be particularly great on neuromorphic hardware, where forward and backward network weights are represented by physically separate wiring on a circuit.

Forward Gradient

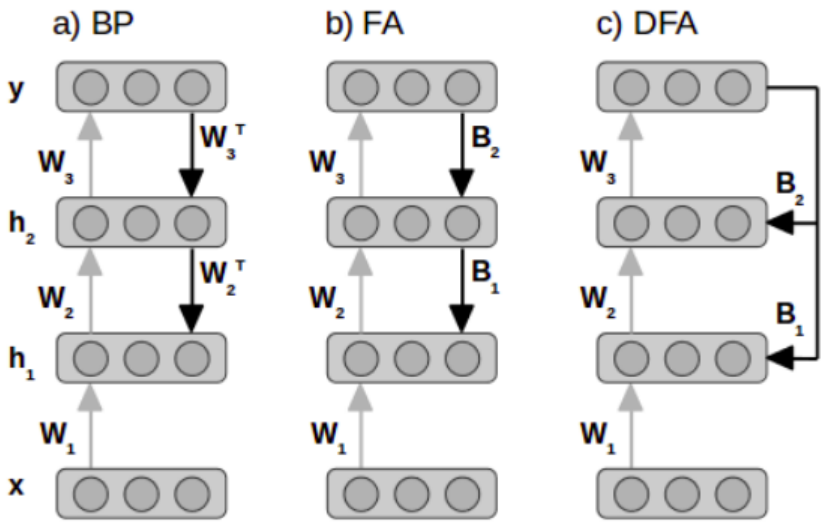
- We can estimate the gradient in forward mode.
- The forward gradient is an unbiased estimation of the standard gradient.
- Can be much faster than GD.

Baydin, 2022, Gradients without Backpropagation

Direct Feedback Alignment

Alignment Provides Learning in Deep Neural Networks the gradient of the last layer is computed and is distributed to all previous layers.

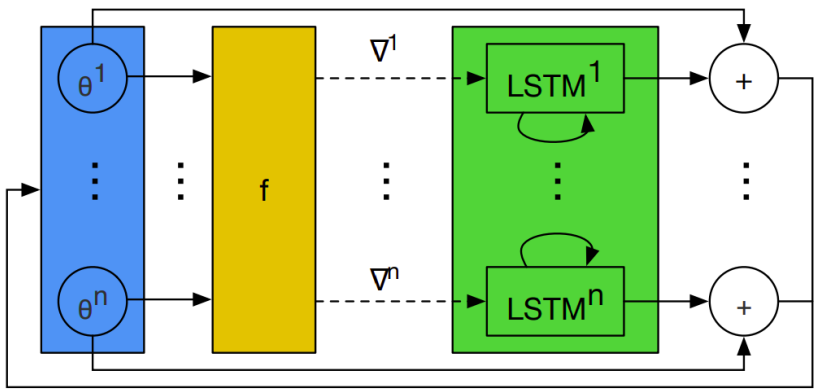
Can be used to solve real-life problems efficiently!



Nokland, 2016, Direct Feedback Alignment Provides Learning in Deep Neural Networks

Learning to Optimize

- Use meta-learning to find some "optimization algorithm".
- Two loops - inner loop optimizes a function, outer loop optimizes an optimizer (LSTM)



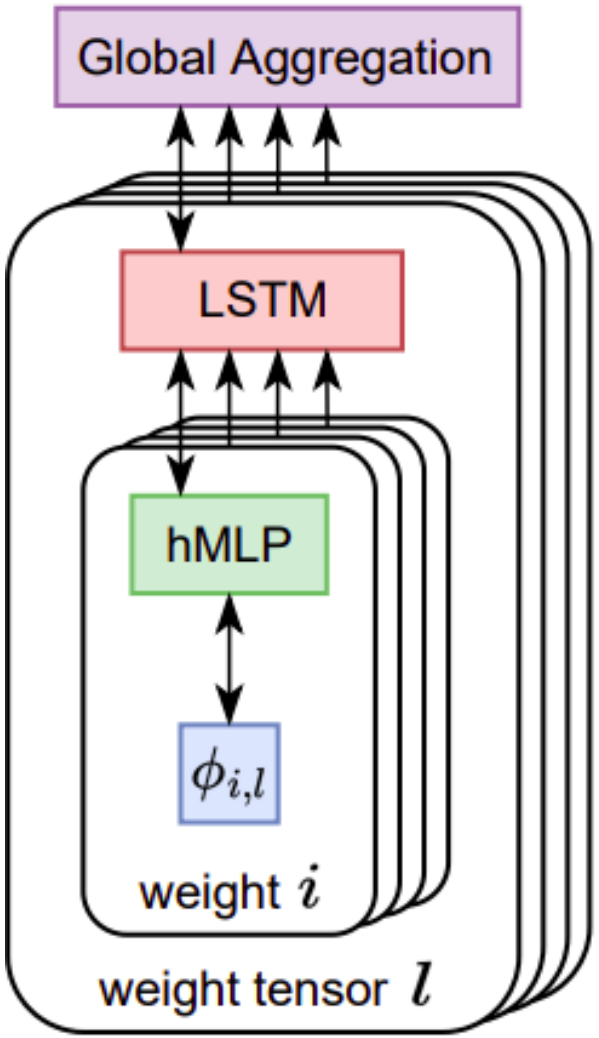
VeLO

First actually useful learned optimizer!

- Trained to solve many different optimization problems
- Uses hypernetworks
- Each hypernetwork ingests multiple features:
  - Exponential moving averages of the gradient and squared gradient
  - Mean and variance of weights and gradients
  - Training stage (info about training process).

Metz, 2022, VeLO: Training Versatile Learned Optimizers by Scaling Up.

VeLO


Global Aggregation
LSTM
hMLP
$\phi_{i,l}$
weight i
weight tensor l

VeLO

- Works very well for "smaller" networks (less than 500M)
- Allows much larger batches (10x)
- VeLO learns implicit learning rate scheduling
- Adapts to training horizon
- 2x memory overhead
- Fails after 200k iterations
- Sometimes fails out of distribution

LION

- Symbolic program assembly takes 45 common operations from numpy
- The program can access usual information - weight, gradient, learning step + some open variables
- Uses an evolutionary algorithm to create new optimization algos
- Uses many tricks - removal by wrong syntax, warm-start (AdamW)
- Funneling process to allow only the most promising algos to go from proxy tasks to large real problems
- 512 TPUs for days!

LION

Lion algorithm:

- Uniform updates to all weights! Adds a lot of noise $\rightarrow$ generalization
- Faster/less memory than AdamW, Adam, and adafactor. And often better.

Chen, 2023, Symbolic Discovery of Optimization Algorithms