

Interactive Session: Practical Insights into Deep Learning Optimization

Berkant Turan
turan@zib.de

Zuse Institute Berlin

MATH+ Summer School · September 12, 2023



Berlin Mathematics Research Center



Outline

1. Brief Overview of Deep Learning

- Paradigms of Learning

- Pillars of Deep Learning

2. Interactive Session - First-Order Methods in Deep Learning

- A Leap Forward with Momentum

- Nesterov Accelerated Gradient (NAG)

- The AdaGrad Approach

- Adam: Adaptive Moment Estimation

- Questions and Discussion

3. Final Thoughts and Conclusion

Brief Overview of Deep Learning

Deep Learning Paradigms

1. Supervised Learning

- Requires labeled data.
- Model learns to map input to given label.
- Examples: image classification, regression, object detection etc.

2. Self-Supervised Learning

- Unlabeled data - can be applied to vast amount of data.
- Learning algorithm for *Generative Modeling*
- Examples: *Stable Diffusion*, *GPTs*, *Midjourney* etc.

3. Reinforcement Learning

- No explicit labels; learns from reward signals.
- Model interacts with an environment to maximize cumulative reward.
- Examples: game playing, robotics etc.

Yann Lecun's Cake Analogy

Y. LeCun

Learning Paradigms: information content per sample

- ▶ **"Pure" Reinforcement Learning (cherry)**
 - ▶ The machine predicts a scalar reward given once in a while.
 - ▶ **A few bits for some samples**
- ▶ **Supervised Learning (icing)**
 - ▶ The machine predicts a category or a few numbers for each input
 - ▶ Predicting human-supplied data
 - ▶ **10 → 10,000 bits per sample**
- ▶ **Self-Supervised Learning (cake génoise)**
 - ▶ The machine predicts any part of its input for any observed part.
 - ▶ Predicts future frames in videos
 - ▶ **Millions of bits per sample**



Source: <https://leshouches2022.github.io/SLIDES/compressed-yann-1.pdf>

The Pillars of Deep Learning



Model



Data



Optimizer



Loss

Perspectives

- **Mathematics:** "The model acts like a *universal function approximator* and the data is the *ground truth*."
- **Computer Science:** "The model acts like a *compiler* and the data is analogous to the *source code*."

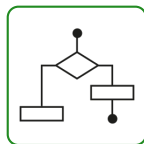
The Pillars of Deep Learning



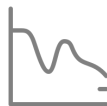
Model



Data



Optimizer



Loss

Perspectives

- **Mathematics:** "The model acts like a *universal function approximator* and the data is the *ground truth*."
- **Computer Science:** "The model acts like a *compiler* and the data is analogous to the *source code*."

Importance of Optimization in Deep Learning

Risk Minimization:

Ideal objective: minimize expected loss over true data distribution p_{data} :

$$R(\theta) = \mathbb{E}_{(x,y) \sim p_{\text{data}}} [L(f(x; \theta), y)]$$

Empirical Risk Minimization:

Find $\hat{\theta}$ that minimizes the average loss:

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N L(f(x_i; \theta), y_i)$$

Gradient Descent:

Iteratively update parameters:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} \left(\frac{1}{N} \sum_{i=1}^N L(f(x_i; \theta_t), y_i) \right)$$

Mini-Batch Gradient Descent:

Update using a subset of data:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} \left(\frac{1}{B} \sum_{i=1}^B L(f(x_i; \theta_t), y_i) \right)$$

Today's Interactive Session

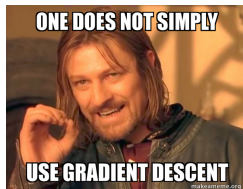
Goals

1. Understand the foundational concepts in optimization for deep learning.
2. Get hands-on experience in implementing and experimenting with various first-order optimization algorithms in deep learning.
3. Learn about common pitfalls and best practices when optimizing deep learning models.

Jupyter Notebook:

Repository @ <https://github.com/b-turan/math-plus-summer-school>

If you have any questions, please feel free to contact me. You are invited to discuss your results and insights with the other participants.



A Leap Forward with Momentum

Concept of Momentum

- **Basic Idea:** Utilize previous gradients to gain 'momentum' and speed up convergence.
- **Update Rule:**

$$\mathbf{v}_{t+1} = \gamma \mathbf{v}_t + \eta \nabla_{\theta} L(\theta_t)$$

$$\theta_{t+1} = \theta_t - \mathbf{v}_{t+1}$$

- **Role of γ :** Sets the weight of 'memory' of past gradients. Common value: 0.9.
- **Intuition:** Think of a ball rolling downhill; it picks up speed (and hence momentum) as it goes.

Nesterov Accelerated Gradient (NAG)

The "look-ahead" Strategy

- **Improvement over Momentum:** Takes a "look-ahead" gradient step.
- **Update Rule:**

$$\mathbf{v}_{t+1} = \gamma \mathbf{v}_t + \eta \nabla_{\theta} L(\theta_t - \gamma \mathbf{v}_t)$$

$$\theta_{t+1} = \theta_t - \mathbf{v}_{t+1}$$

- **Intuition:** Imagine you are rolling down while looking ahead to adjust your path.
- **Benefit:** Faster and potentially more accurate convergence.

Adaptive Gradient (AdaGrad)

Introducing Adaptivity

- **Adapting Learning Rates:** Adjusts the learning rates for each parameter during training.
- **Update Rule:**

$$G_{t+1,ii} = G_{t,ii} + (\nabla_{\theta} L(\theta_t))_i^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_{t+1}} + \epsilon} \odot \nabla_{\theta} L(\theta_t)$$

- ϵ : A small constant to prevent division by zero, typically 1×10^{-8} .
- **Benefit:** Good for sparse data and features that appear infrequently.

Adaptive Moment Estimation (Adam)

Best of Both Worlds

- **Combination of Momentum and AdaGrad:** Keeps track of past gradients and their squares.
- **Update Rule:**

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) \nabla_{\theta} L(\theta_t)$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) (\nabla_{\theta} L(\theta_t))^2$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 - \beta_1^{t+1}}$$

$$\hat{v}_{t+1} = \frac{v_{t+1}}{1 - \beta_2^{t+1}}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_{t+1} + \epsilon}} \hat{m}_{t+1}$$

- **Common values:** $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$.
- **Benefit:** Incorporates adaptivity and momentum. Often recommended as default optimizer in deep learning.

Questions and Discussion

1. Which Optimizer Outperformed Others?

- What might be the reasons?
- Which optimizer do you usually choose?
- Do you usually experiment with different optimizers?
- Share your experience with the us.

2. Areas for Improvement?

- How can we improve the optimization results?

3. Further Exploration

- Why do we rarely use second-order methods in deep learning?
- Does *Learning Rate Scheduling* make sense for adaptive methods?
- What about *Gradient-Clipping* for Momentum?

Final Thoughts and Conclusion

Summary

- Successfully implemented and compared (1) SGD, (2) Momentum, (3) Nesterov, (4) AdaGrad, (5) Adam on MNIST.
- Each has merits and drawbacks, which we discussed.

Limitations

- This is not an exhaustive list; many other algorithms exist, such as *Adadelata*, *RMSProp*, and *Nadam*.
- Importance of hyperparameter tuning for achieving the best results.

Key Takeaways

- Optimization algorithm nuances and choice are key to model training.
- Tuning hyperparameters are critical steps, not just 'set-and-forget' decisions.
- Better understanding boosts confidence in optimizer choice.

Thank You!

Thank You For Your Attention!



Any Questions?