Machine learning for vision and multimedia

(01URPOV)

Lab 02 – Hyper-parameter optimization
Francesco Manigrasso

2025-2026

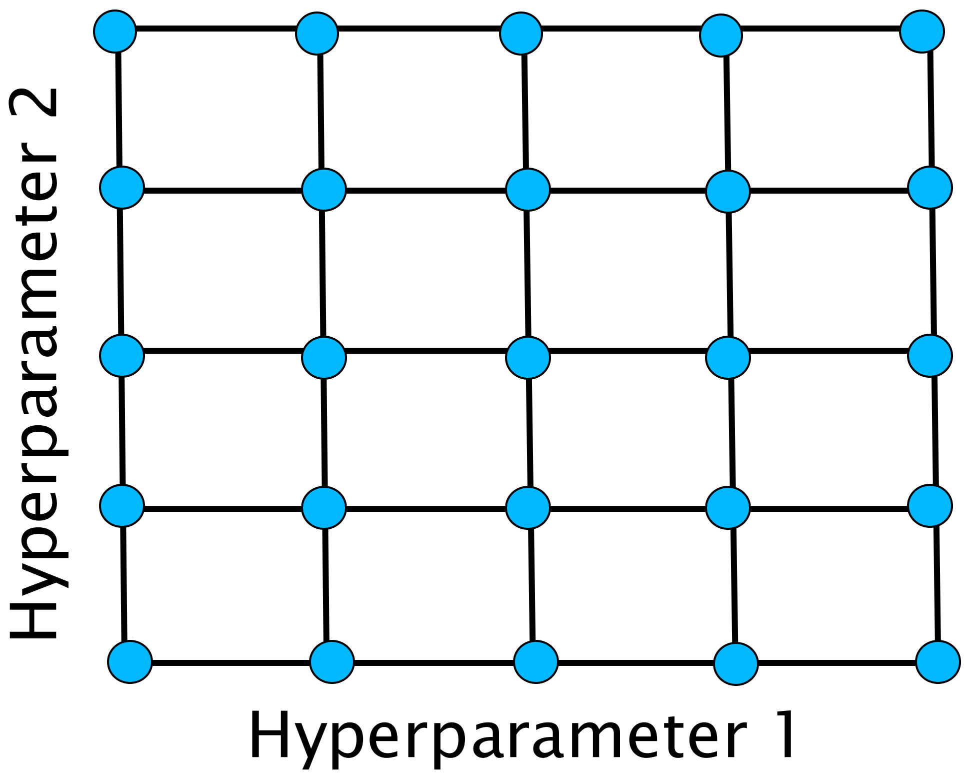
POLITECNICO
DI TORINO

Tuning hyper-parameters

- (Experience guided) experimentation
  - See Evaluating learning algorithms

Tuning hyper-parameters

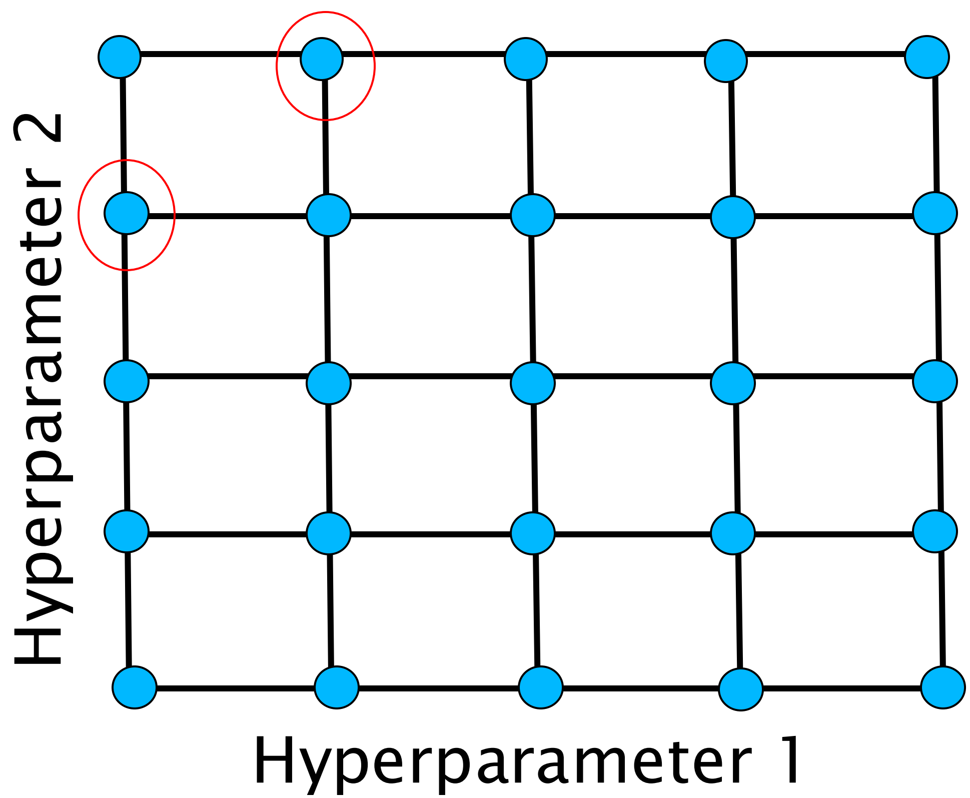
- (Experience guided) experimentation
  - See Evaluating learning algorithms
- Grid search



Tuning hyper-parameters

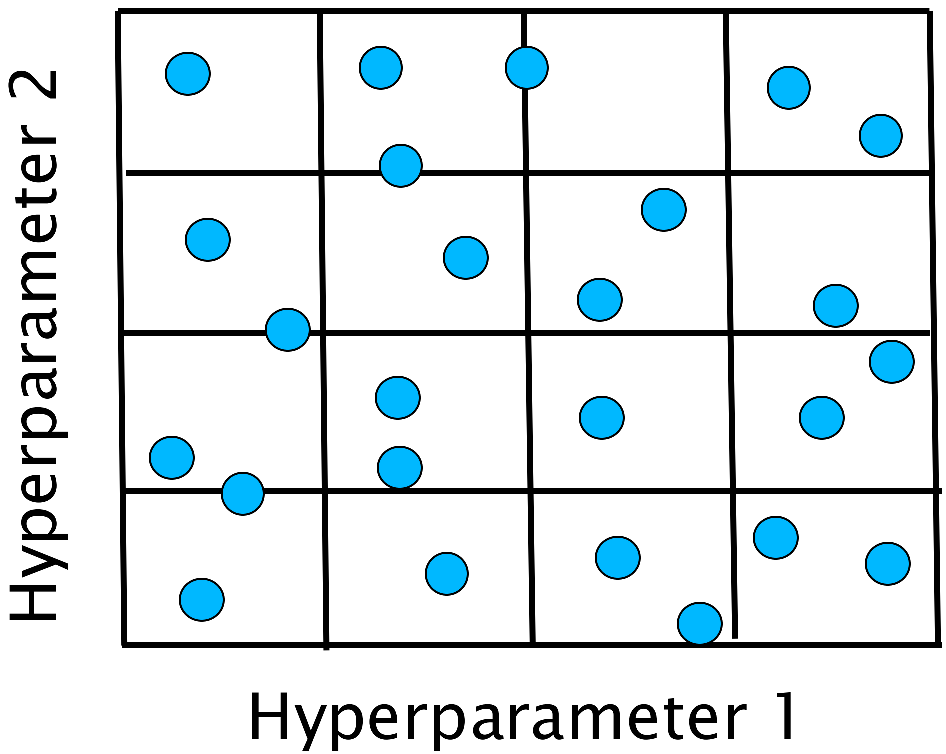
- (Experience guided) experimentation
  - See Evaluating learning algorithms
- Grid search

Pay attention when the best value falls on one of the extremes – the optimal value may not be covered!



Tuning hyper-parameters

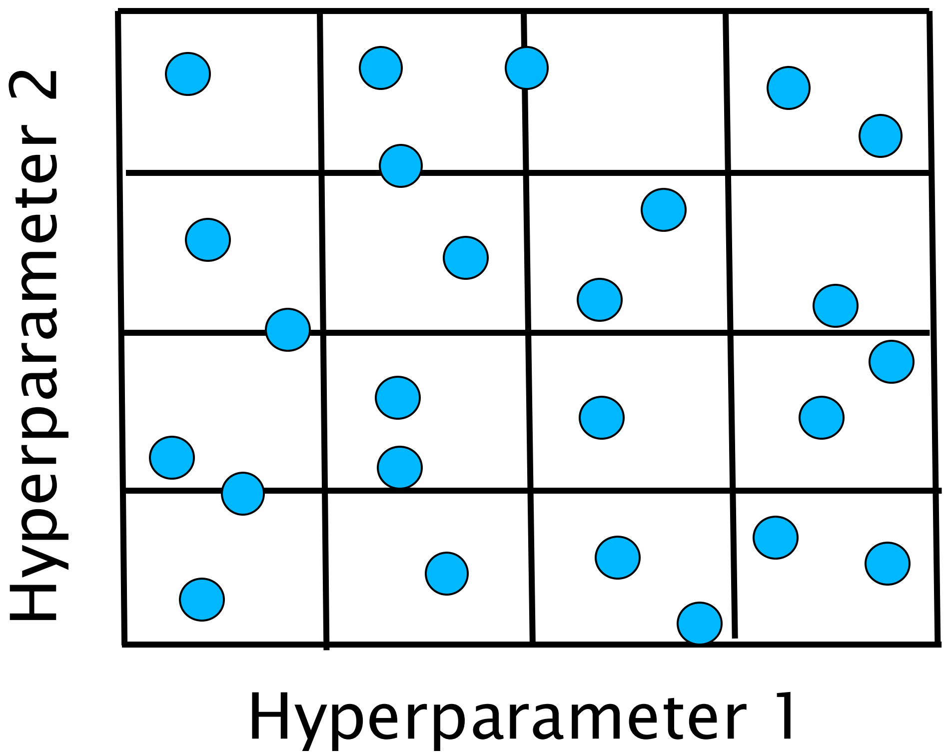
- (Experience guided) experimentation
  - See Evaluating learning algorithms
- Grid search
- Random search



Tuning hyper-parameters

- (Experience guided) experimentation
  - See Evaluating learning algorithms
- Grid search
- Random search

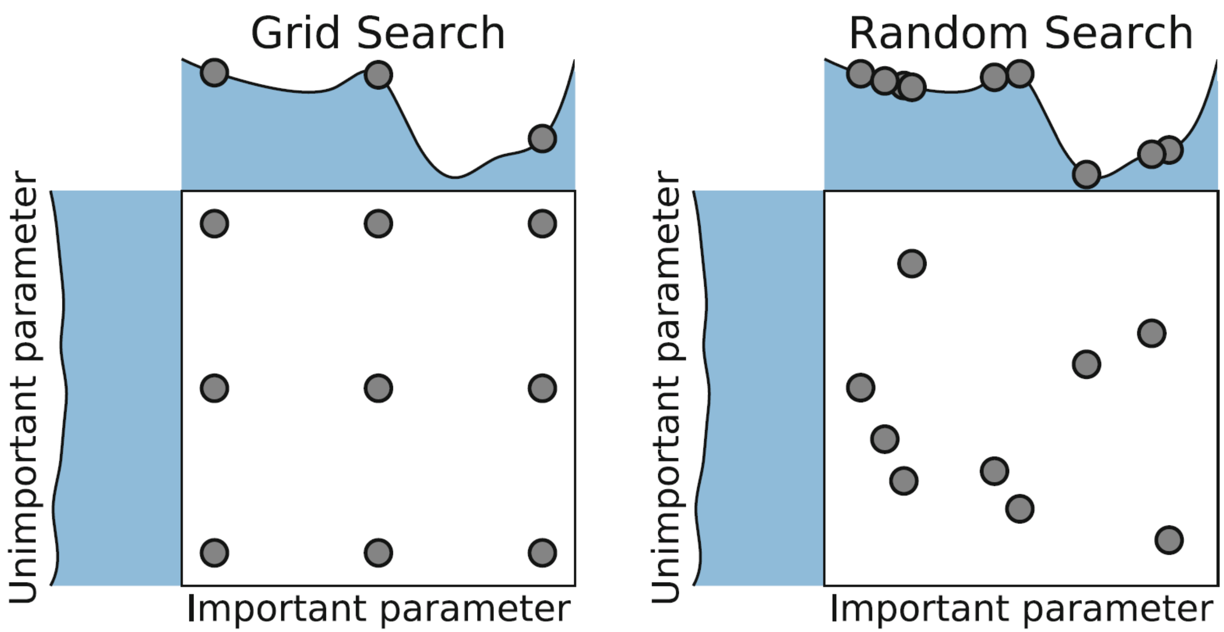
For deep learning, random search often achieves comparable results



Tuning hyper-parameters

Grid search:
- Fixed budget
- Small # of parameters
- Global coverage

Random search:
- Flexible budget
- Large # of parameters
- Works well if a single parameter is important



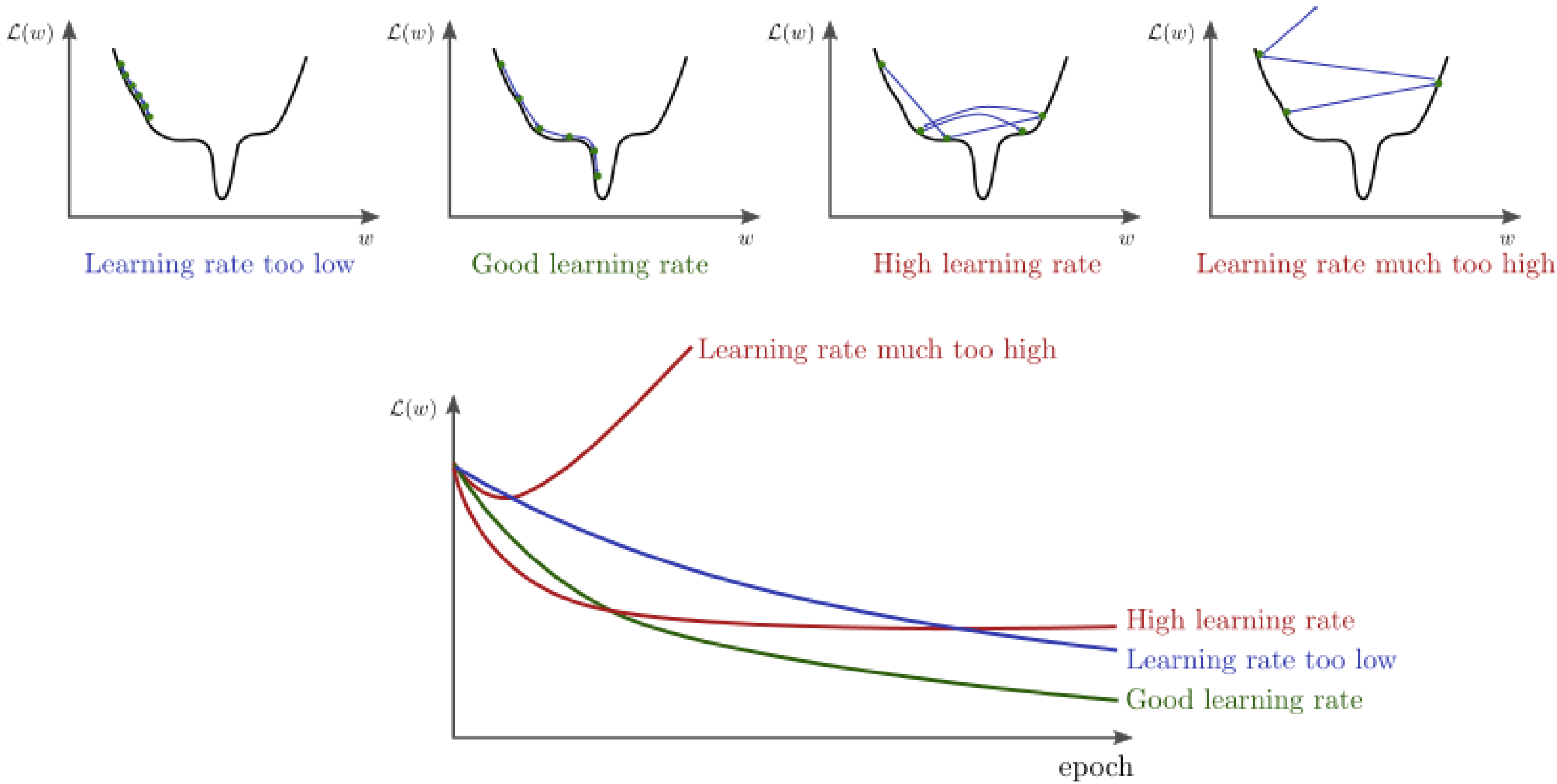
Source: AutoML: Methods, Systems, Challenges, Chapter 1

Learning rate

“The learning rate is perhaps the most important hyperparameter. If you have time to tune only one hyperparameter, tune the learning rate.”
Goodfellow, Bengio & Courville, Deep learning

- However, it is not possible to know in advance the optimal learning rate for a given problem
  - importance of experimenting on a proper validation set
- The learning rate depends:
  - On the batch size: batch size ↑
  - On the initialization: random initialization ↑
    transfer learning ↓
  - On the optimizer

What does a good learning rate look like?



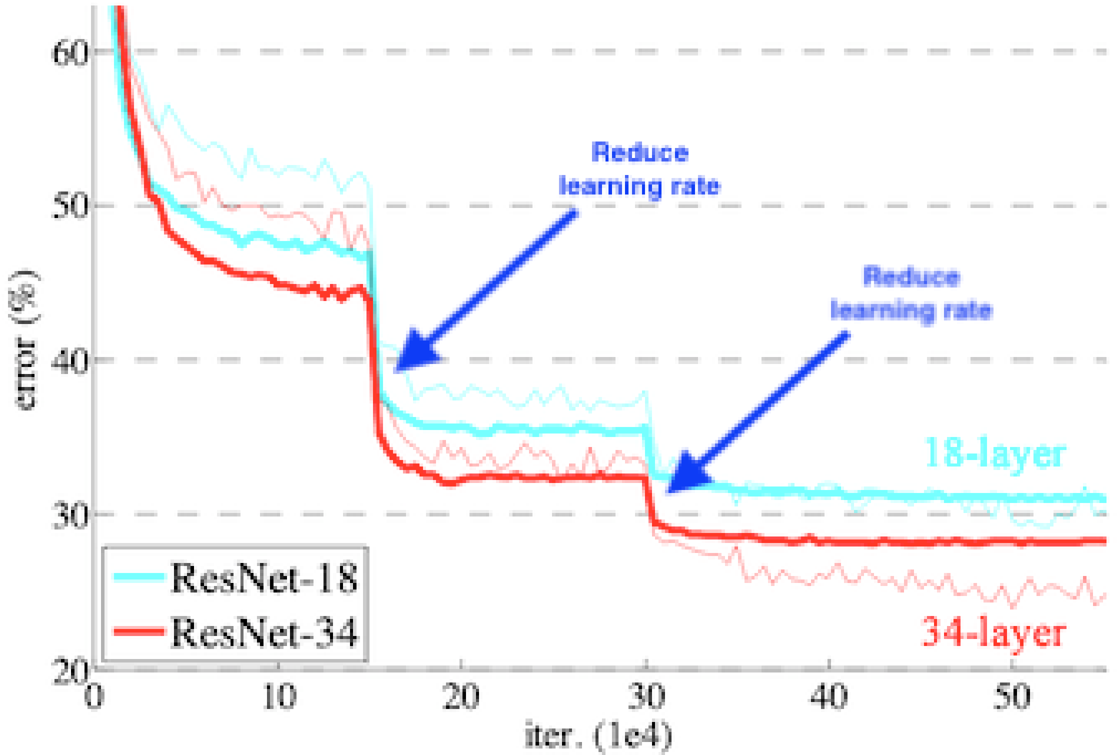
cite: [Stanford cs231]

Tips & tricks

- Change the learning rate by order of magnitude, first, and then fine-tune
  - 0.1, 0.01, 0.001 vs. 0.01, 0.02, 0.03, …
- Use a smaller learning rate when using transfer learning
  - Typically rule of thumb divide by a factor 10
- Consider the use of different learning schedules
  - Learning rate warmup
  - Learning rate decay
  - Reduce learning rate on plateau
  - Cyclical learning rate

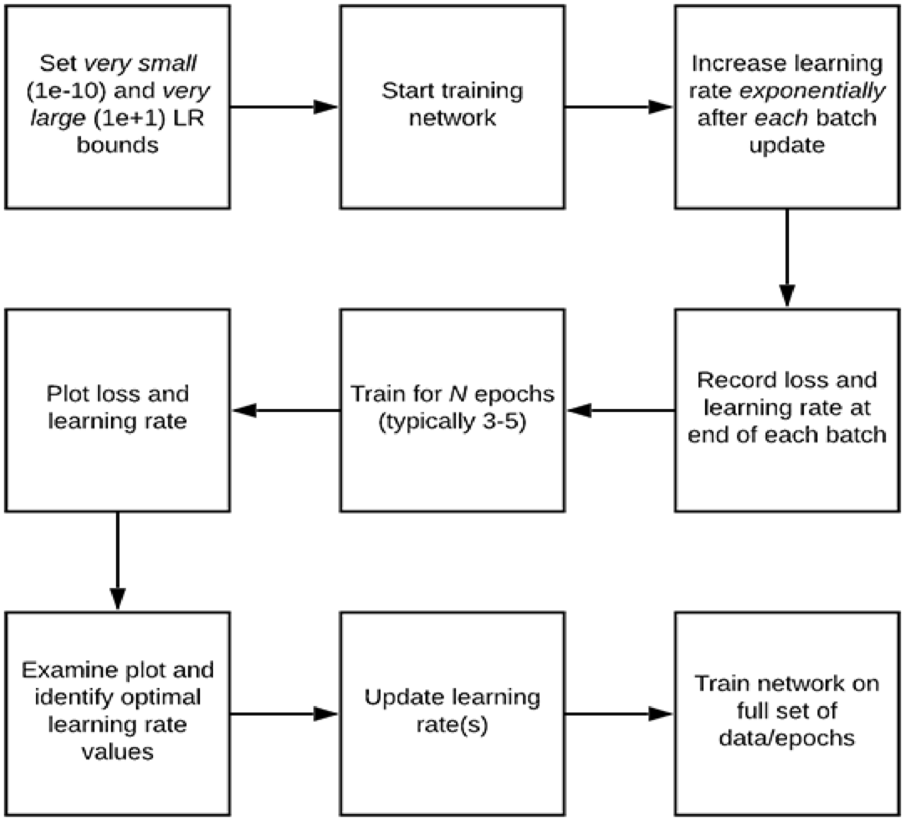
Reduce rate on plateau

- One of the most common “tricks”
- Implemented in `tf.keras.callbacks.ReduceLROnPlateau()`



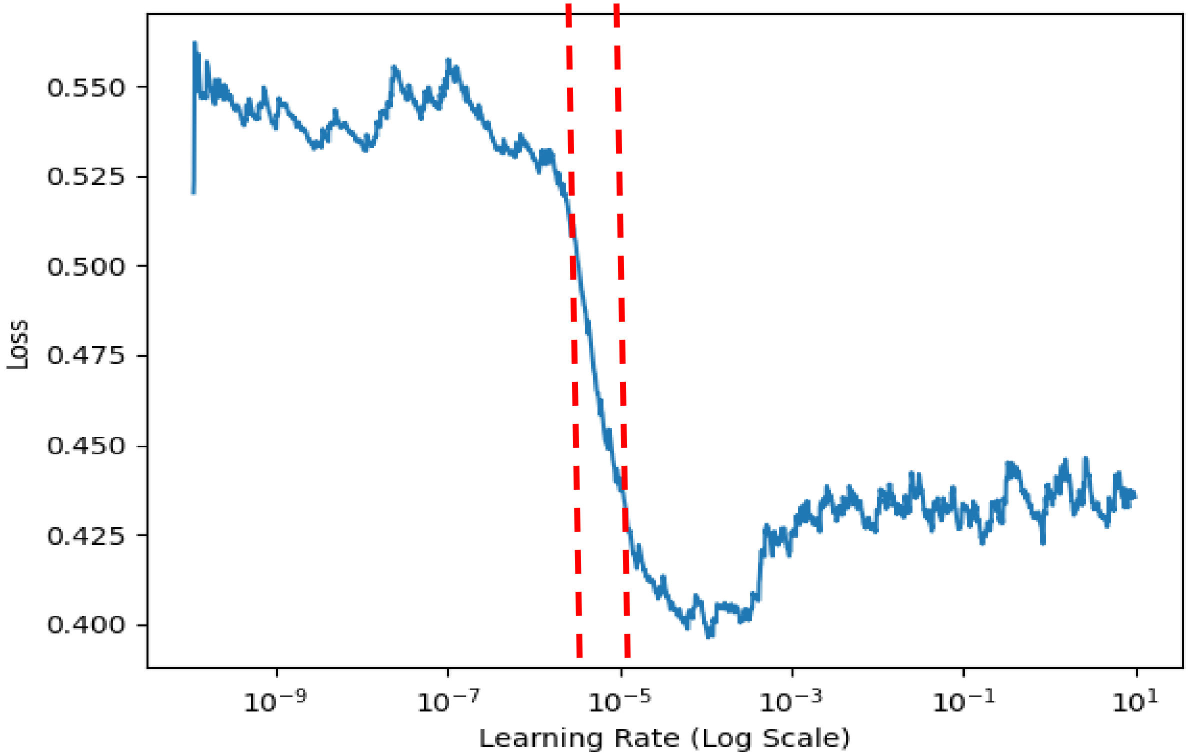
Learning rate finder

- A principled approach to finding the optimal learning rate



Learning rate finder

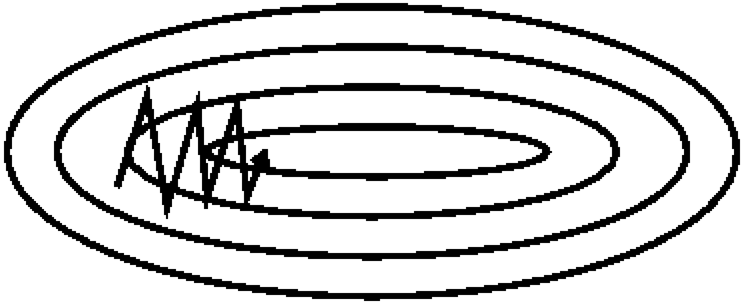
- Example of application of the learning rate finder on a ResNet50 with Adam optimizer



Optimizers: SGD + Momentum

- Plain SGD

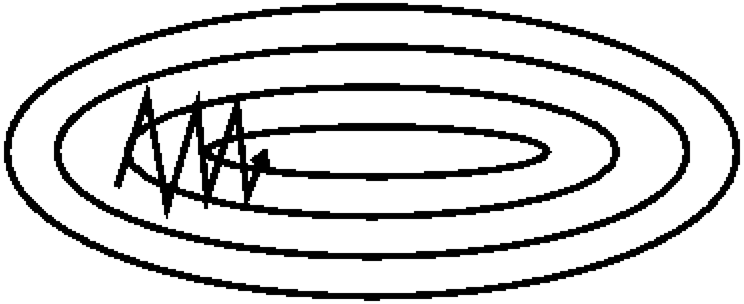
$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$



Optimizers: SGD + Momentum

- Plain SGD

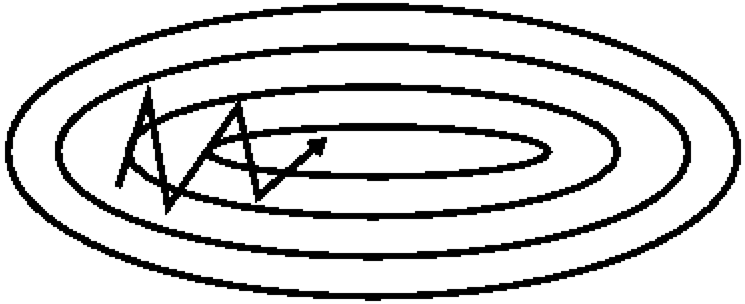
$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$



- SGD with momentum

$$w_{t+1} = w_t - \alpha v_t$$

$$v_t = \beta v_{t-1} + \frac{\partial L}{\partial w_t}$$



Momentum term
Analogy: pushing a ball down a hill
Usually $\beta \approx 0.9$

Optimizers: ADAM

- ADAM belongs to the family of adaptive gradient methods (Adagrad, Adadelta), that differentiate the learning rate for each parameter
- At each time t, it computes the moving average of the first moment (mean) and second moment (variance) of the gradients, with momentum

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) v_t$$

Optimizers: ADAM

- ADAM belongs to the family of adaptive gradient methods (Adagrad, Adadelta), that differentiate the learning rate for each parameter
- At each time t, it computes the moving average of the first moment (mean) and second moment (variance) of the gradients, with momentum

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) v_t$$

- Exponential weighting counteracts zero-initialization

$$\widetilde{m}_t = m_t / (1 - \beta_1^t)$$
$$\tilde{v}_t = v_t / (1 - \beta_2^t)$$

Optimizers: ADAM

- ADAM belongs to the family of adaptive gradient methods (Adagrad, Adadelta), that differentiate the learning rate for each parameter
- Use the bias-corrected first and second moment estimates to update the weights

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t} + \epsilon} m_t$$

Optimizers: ADAM

- ADAM belongs to the family of adaptive gradient methods (Adagrad, Adadelta), that differentiate the learning rate for each parameter
- Use the bias-corrected first and second moment estimates to update the weights

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\tilde{v}_t + \epsilon}} \widetilde{m}_t$$

Each parameter sees a different effective learning rate

Usually $\beta_1 \approx 0.9$, $\beta_2 \approx 0.9999$

Optimizers: Comparison

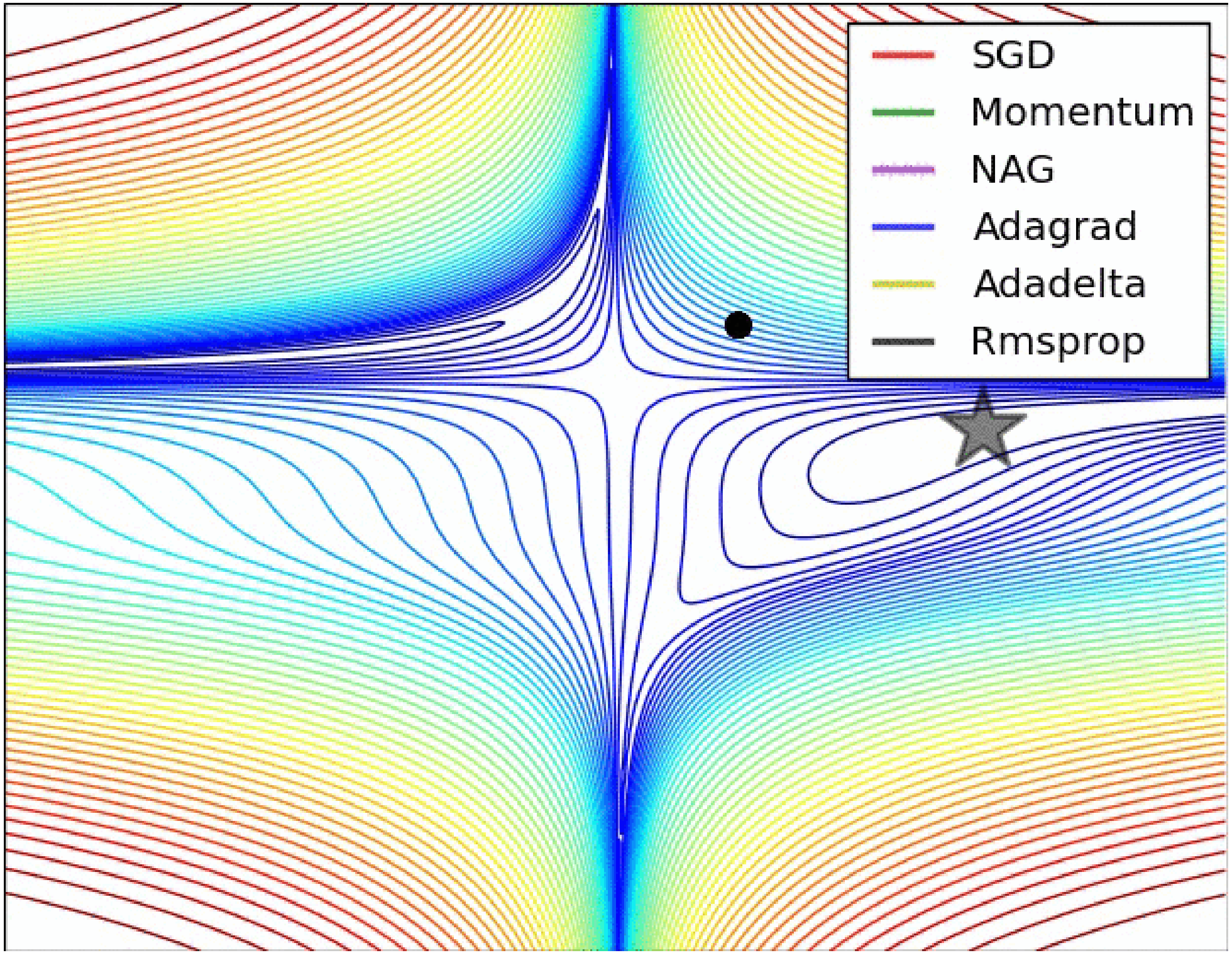


Image credit: https://cs231n.github.io/neural-networks-3/

Optimizers: Comparison

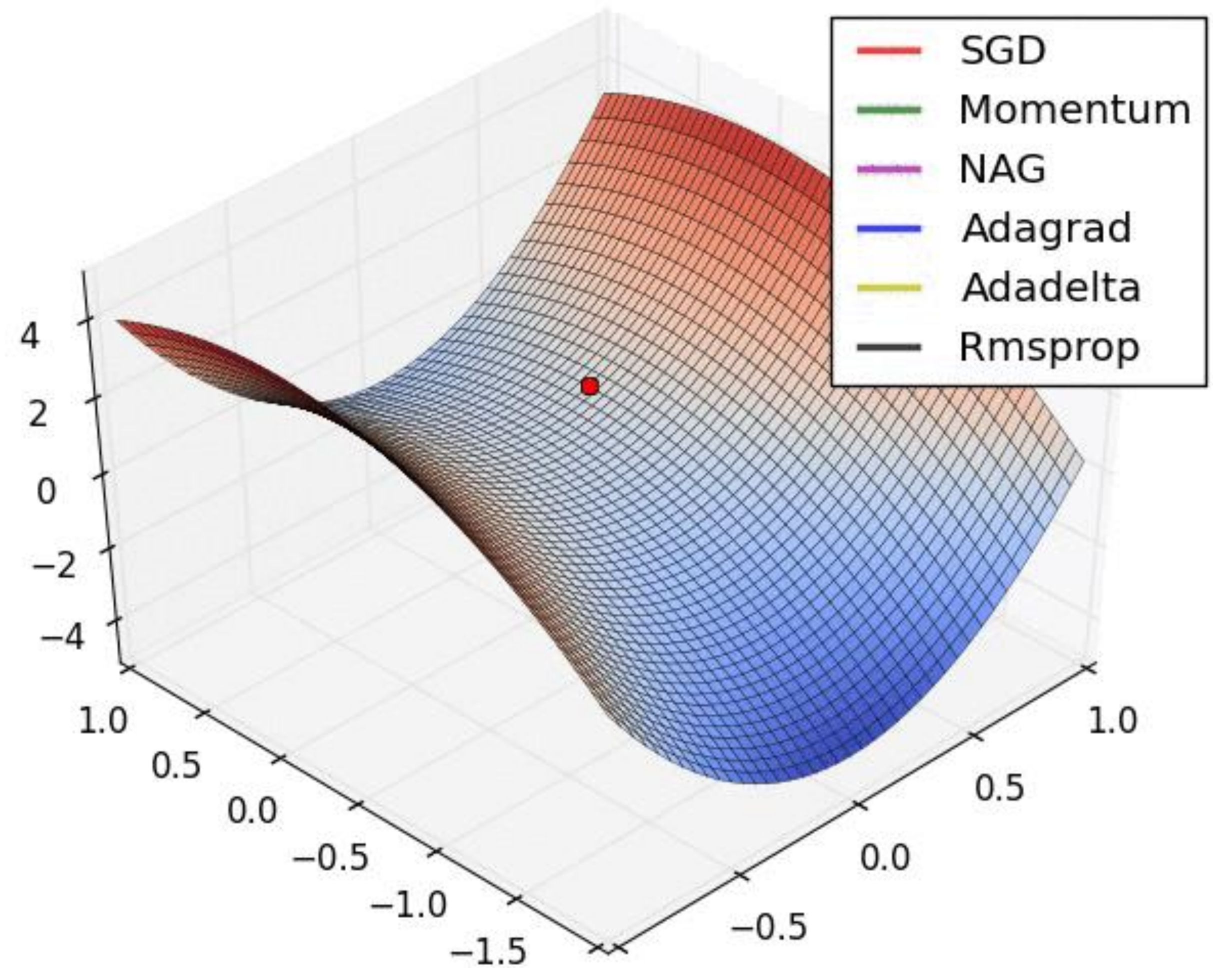


Image credit: https://cs231n.github.io/neural-networks-3/

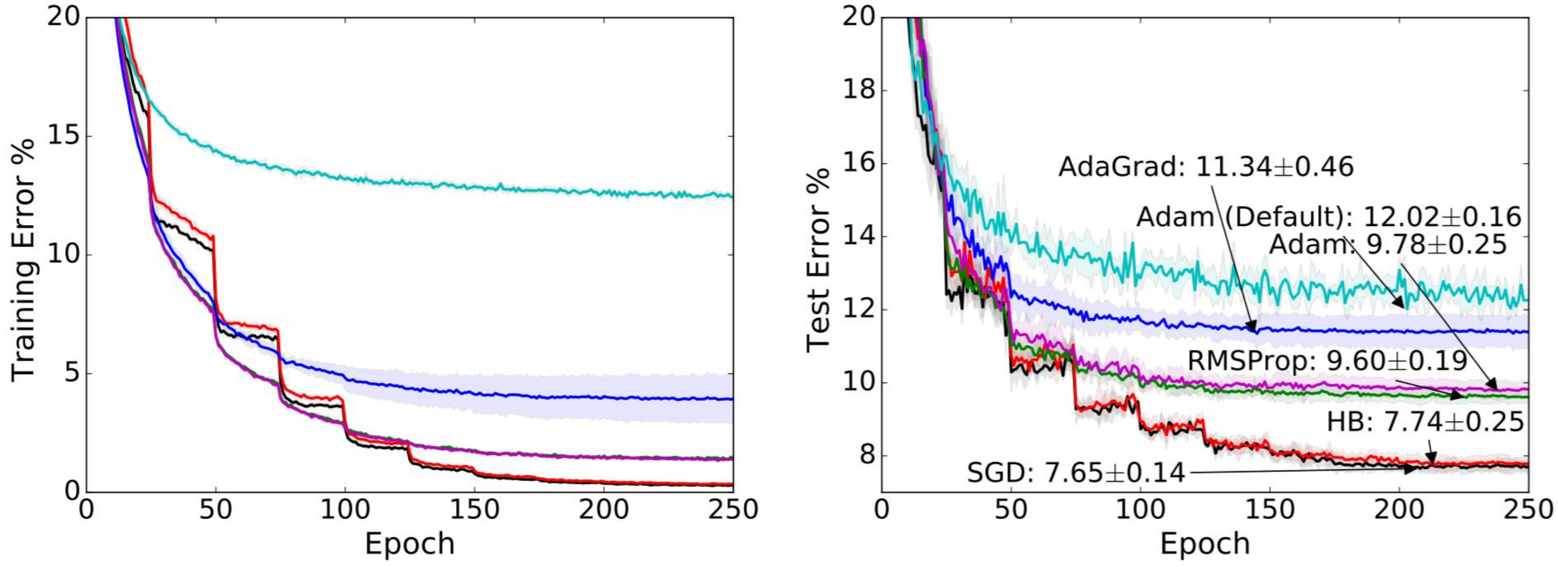
Adaptive vs. SGD

- Comparison of different optimizers on CIFAR-10
  - SGD vs. RMSProp, AdaGrad, Adam
  - Learning rate with decay: parameters α, decay rate
- Grid search
  - logarithmically-spaced grid of five learning rates
  - if the best performance was at one of the extreme, add grid points until the best performance is contained in the middle of two parameters

[Wilson, The Marginal Value of Adaptive Gradient Methods in Machine Learning, 2018]

Adaptive vs. SGD

“We observe that the solutions found by adaptive methods generalize worse (often significantly worse) than SGD, even when these solutions have better training performance”

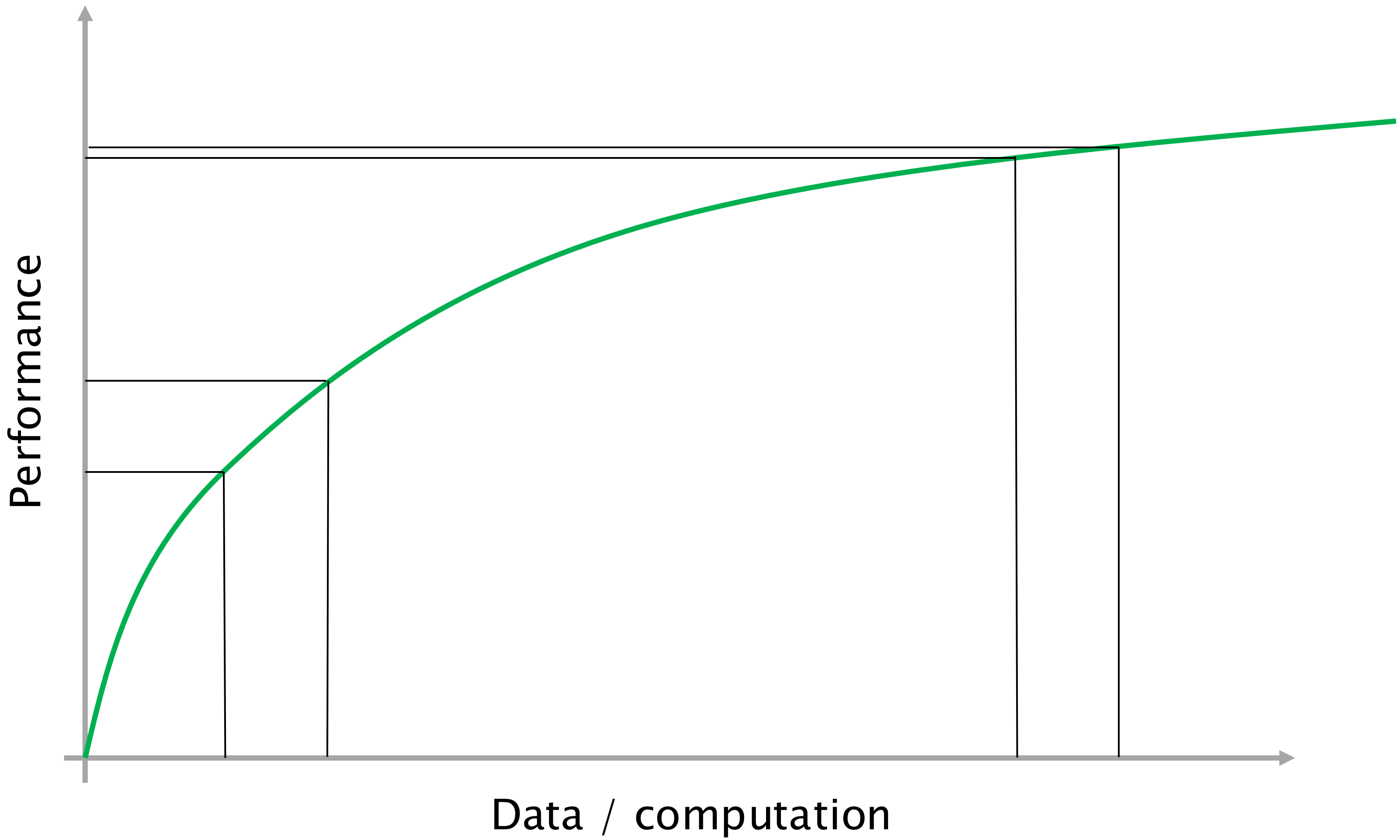


[Wilson, The Marginal Value of Adaptive Gradient Methods in Machine Learning, 2018]

In practice

- Data quality is often more important than hyper-parameters selection
- Consider your computational budget
  - Good enough vs. perfect solution
  - A faster method (to tune and to train) may be preferable
  - SGD requires a lot of tuning to work
- Automate whenever possible, log always
- Beware that automatic hyperparameter optimization at scale is prone to overfitting to the validation set
  - Final performance should be determined on the test set

Diminishing returns



References and tutorials

- F. Cholet, Deep Learning with Python, Manning Publications
- https://cs231n.github.io/neural-networks-3/
- https://www.pyimagesearch.com/2019/08/05/keras-learning-rate-finder/
- https://ruder.io/optimizing-gradient-descent/
- https://www.deeplearningbook.org/contents/optimization.html