Training neural networks - 2

Today

- **Last class.** Setting up the training
  - Activation function, Data preprocessing, Normalization, Initialization

- **Today.** Tuning the training process
  - Learning rate & Batch size
  - Optimizers
  - Regularizers
  - Hyperparameter Tuning

Learning rate & Batch size

SGD

- Recall that SGD can be written as:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla_\theta \left(\sum_{i=1}^{B} \ell(y_i, f_\theta(\mathbf{x}_i)) \right)$$

- There are two key hyperparameters
  - Learning rate η
  - Bath size B

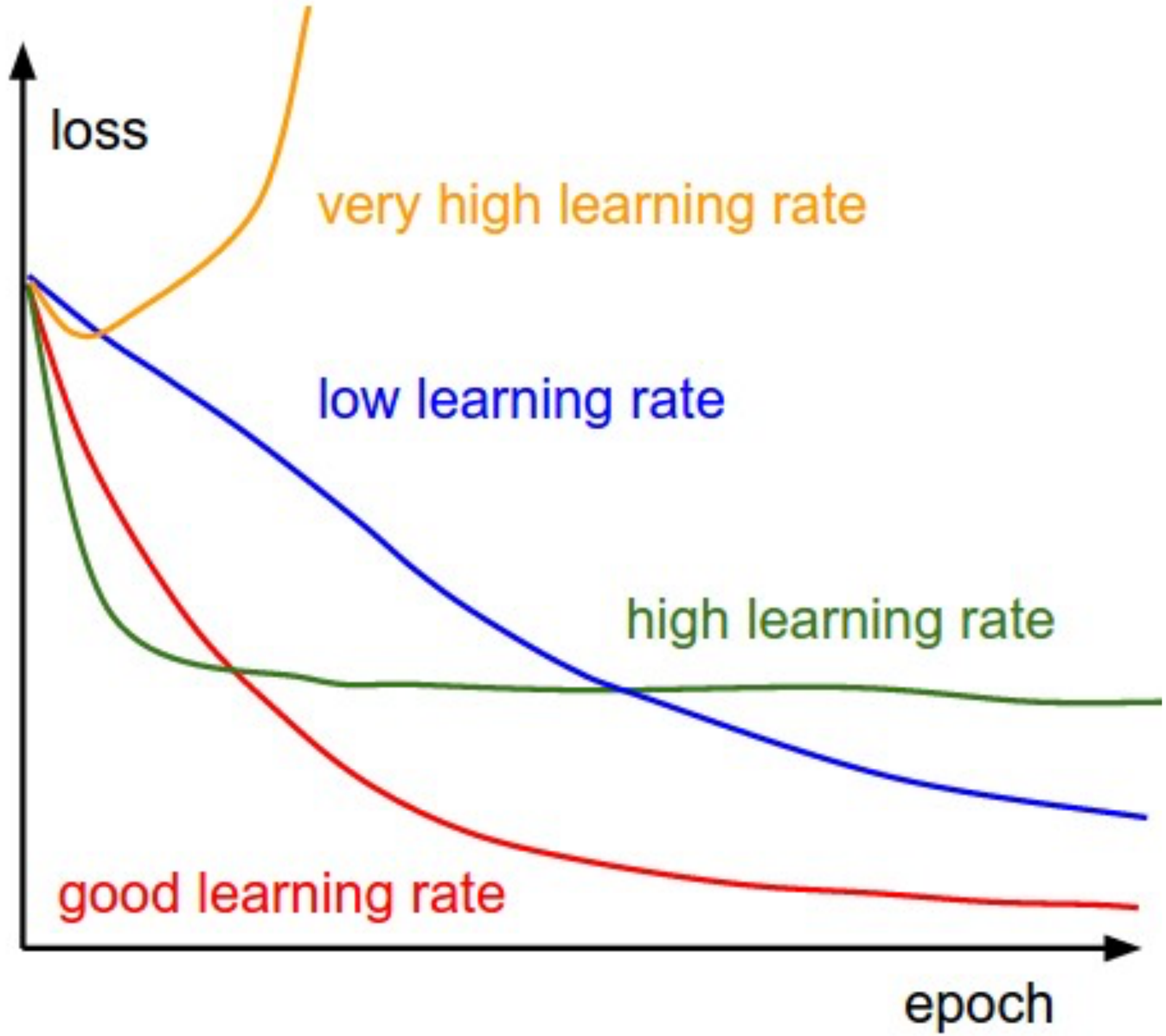
SGD

- **Question.** How do we tune these?
  - Answer. Usually by trial-and-error
    - Train with $(\eta_1, B_1), (\eta_2, B_2), \ldots$
    - Choose the one with the best validation accuracy
  - Tip. Choose the largest possible B, and tune η
    - Large B = less #steps/epoch = Faster training
    - Constrained by (1) RAM
      (2) Generalization

Learning rate vs. Loss

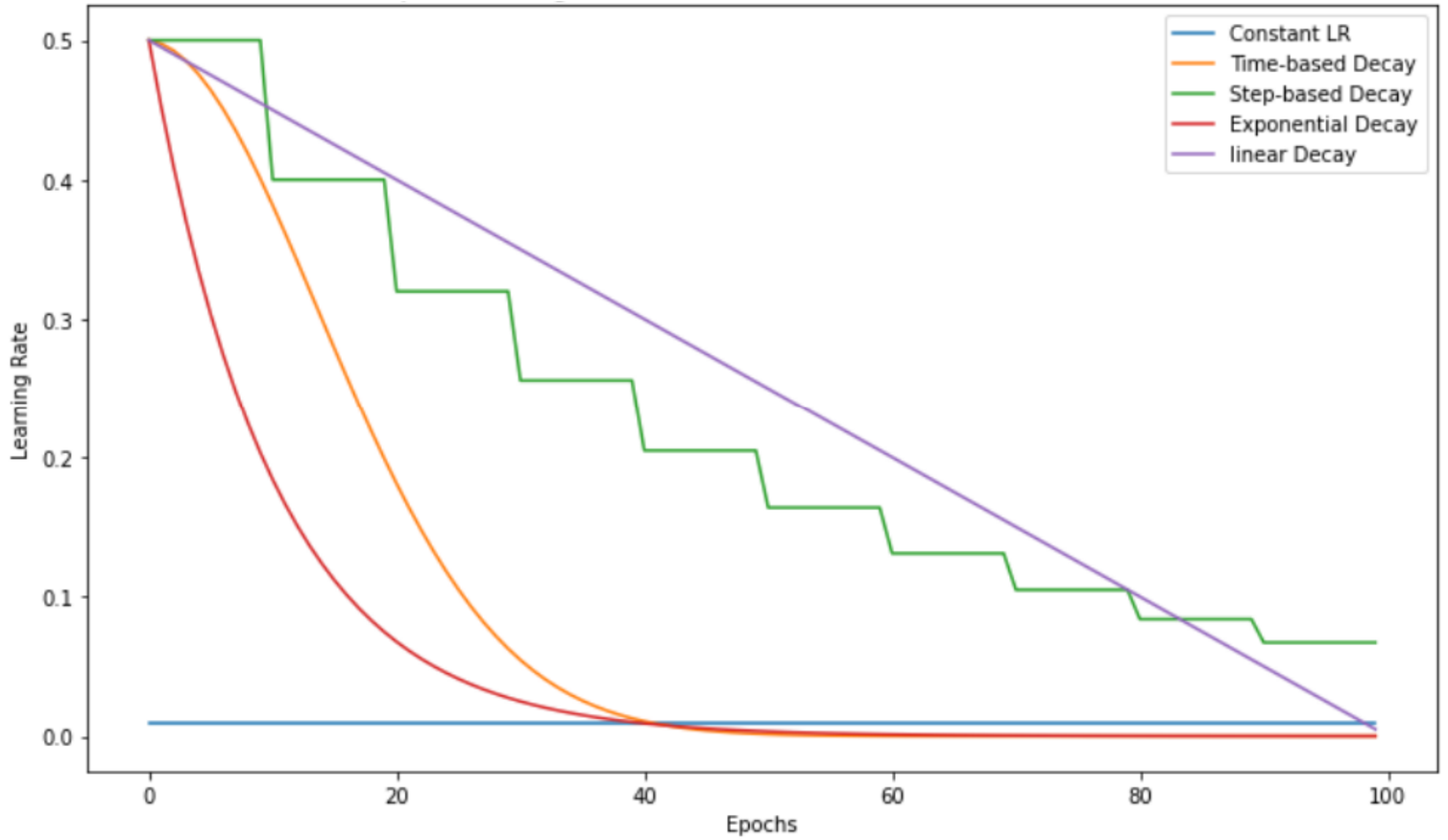
- **High LR.**
  - Fast Convergence
  - High Loss at Convergence
- **Low LR.**
  - Slow Convergence
  - Low Loss at Convergence

- **Question.**
  - Can we get the best of both worlds?



Learning rate scheduling

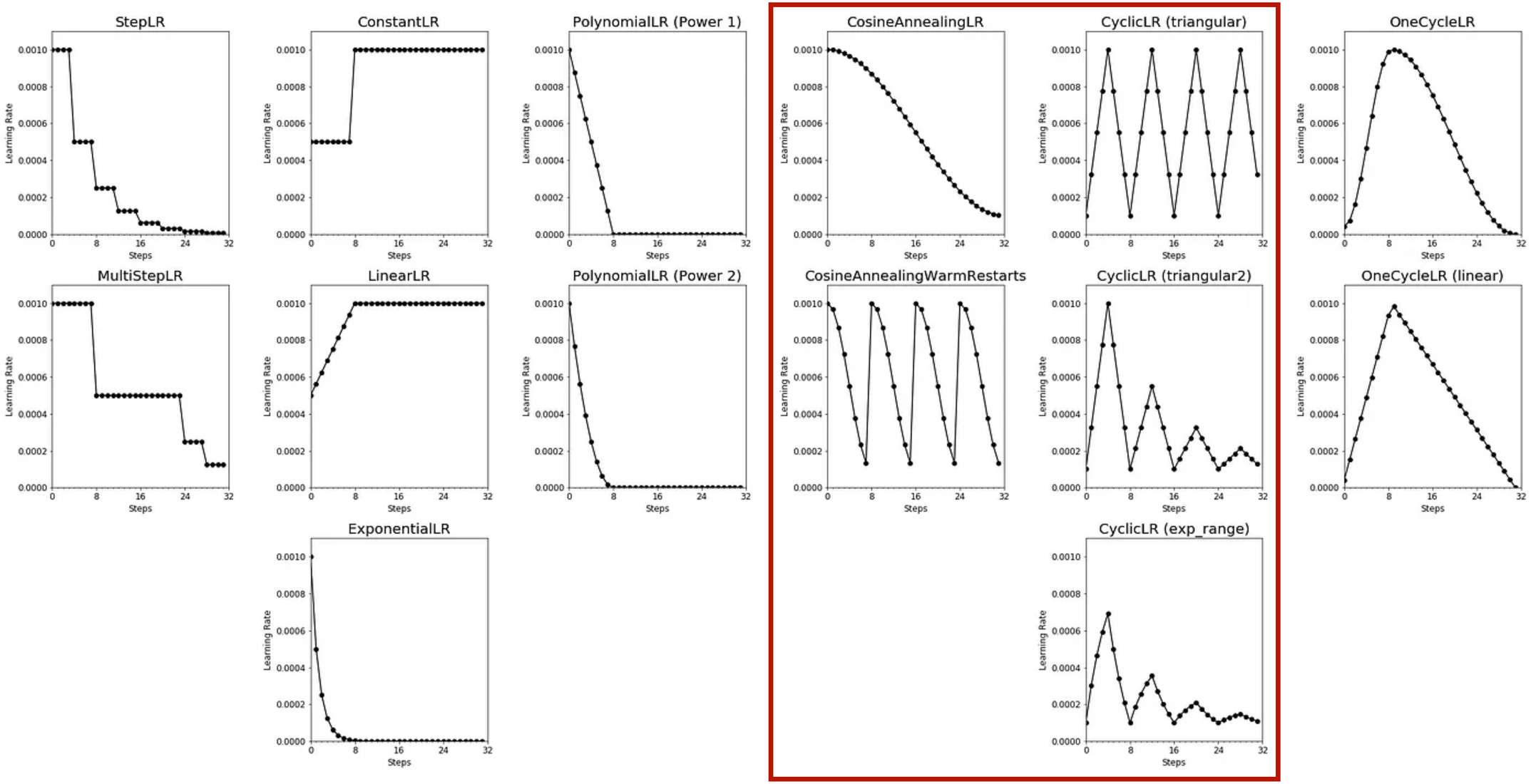
- **Idea.** Slowly decay the LR
  - Step
  - Linear
  - Exponential
  - …



- Note. Optimizers have different sensitivities
  - Vanilla SGD is very sensitive
  - Adam is not

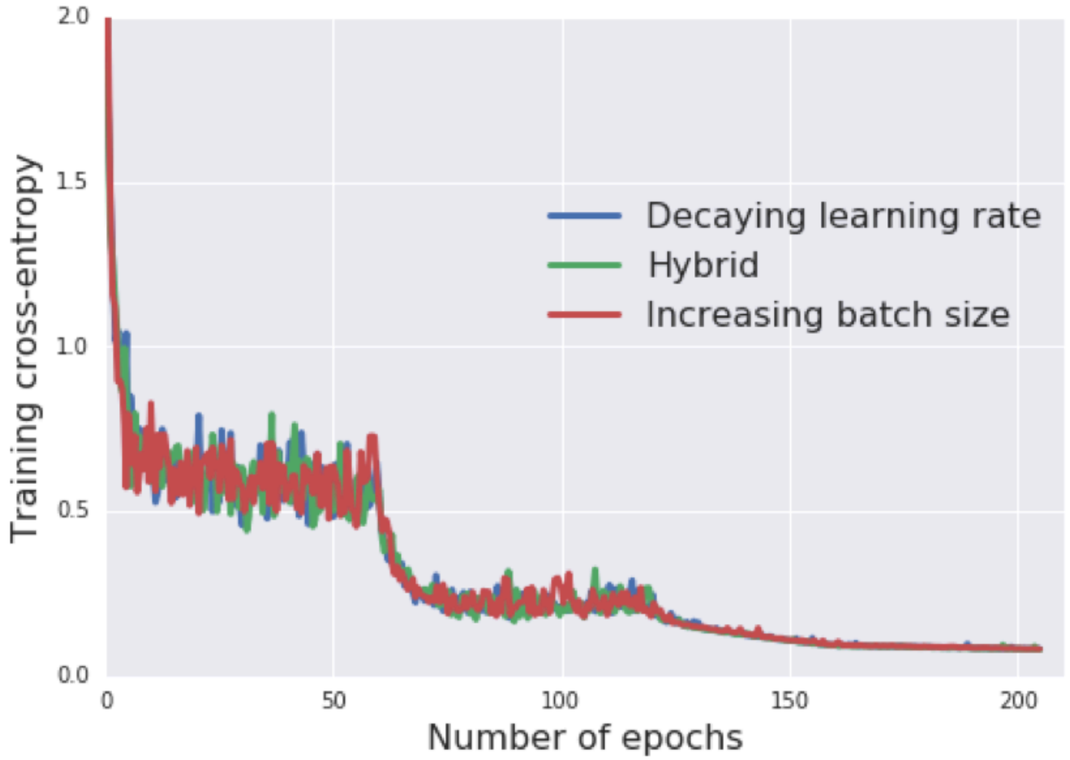
Learning rate scheduling

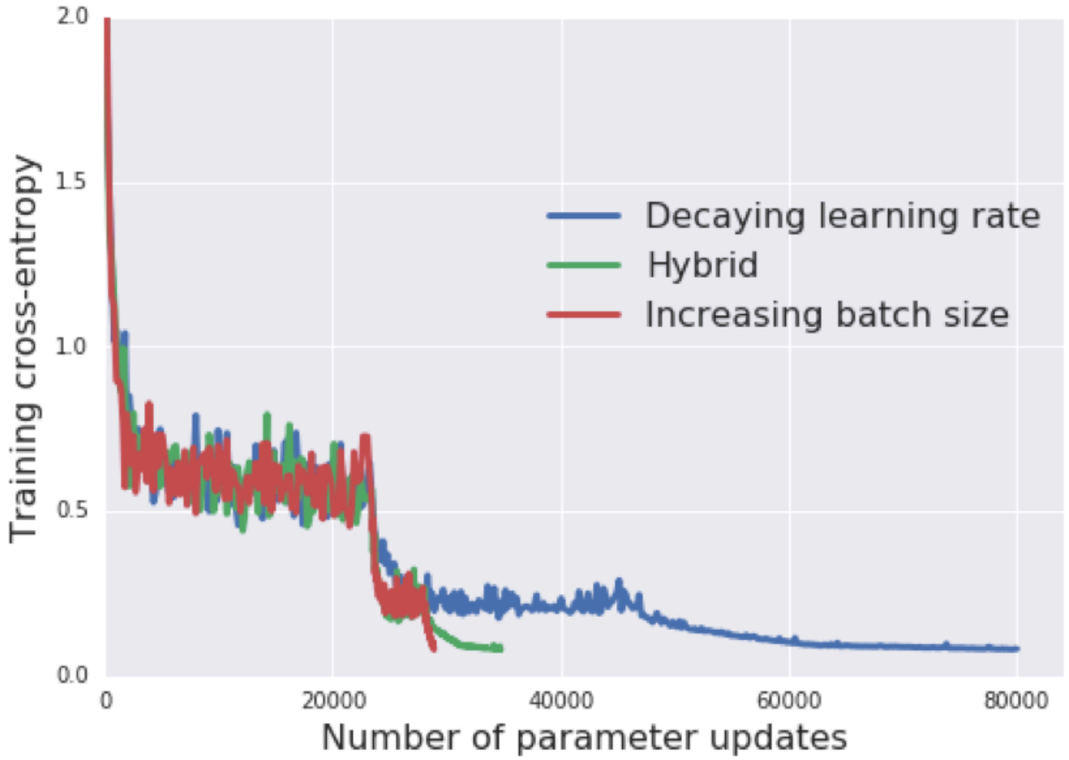
- **Popular.** Cosine annealing / Cyclic LR + Warmup



Learning rate vs. Batch size

- Empirically, increasing the batch size has a similar effect to decreasing the learning rate
  - **Idea.** Less “noise” added to the trajectory

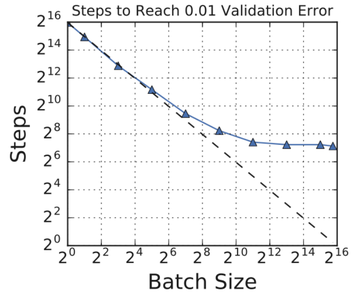




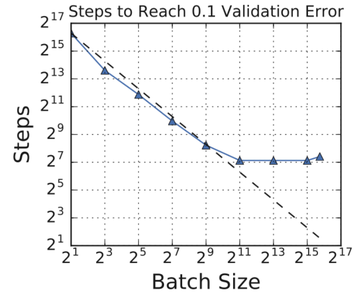
Smith et al., “Don’t decay the learning rate, increase the batch size,” ICLR 2018

Learning rate vs. Batch size

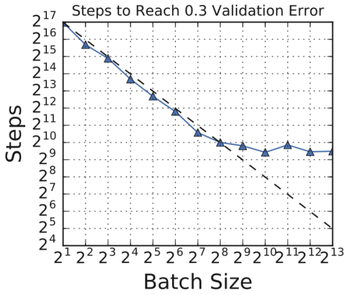
- With a larger batch, we expect:
  - Reduced #steps needed for achieving similar test accuracy
    - Optimal LR scales linearly with the batch size
    - The advantage saturates at some critical batch size



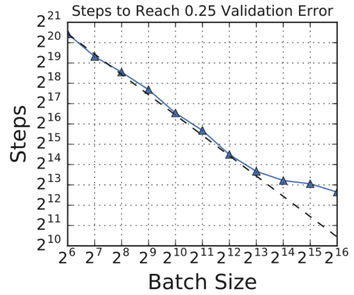
(a) Simple CNN on MNIST



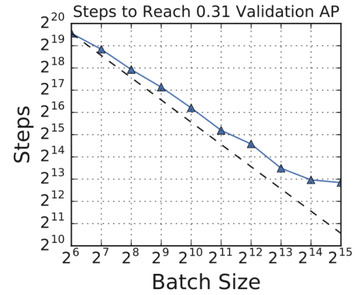
(b) Simple CNN on Fashion MNIST



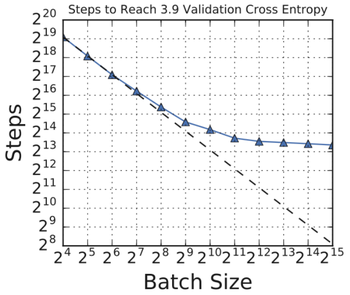
(c) ResNet-8 on CIFAR-10



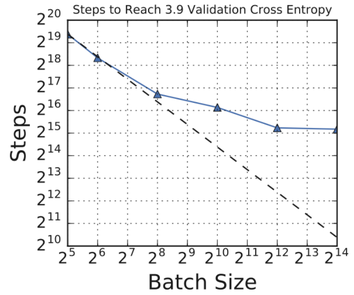
(d) ResNet-50 on ImageNet



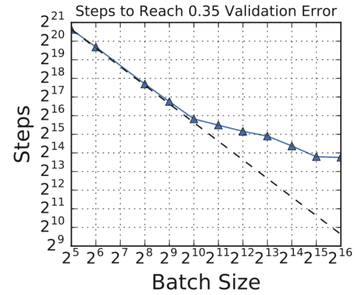
(e) ResNet-50 on Open Images



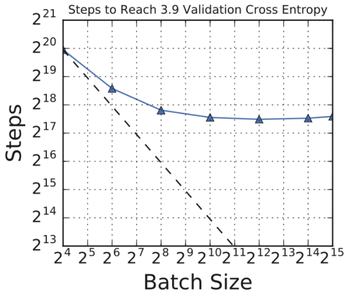
(f) Transformer on LM1B



(g) Transformer on Common Crawl



(h) VGG-11 on ImageNet



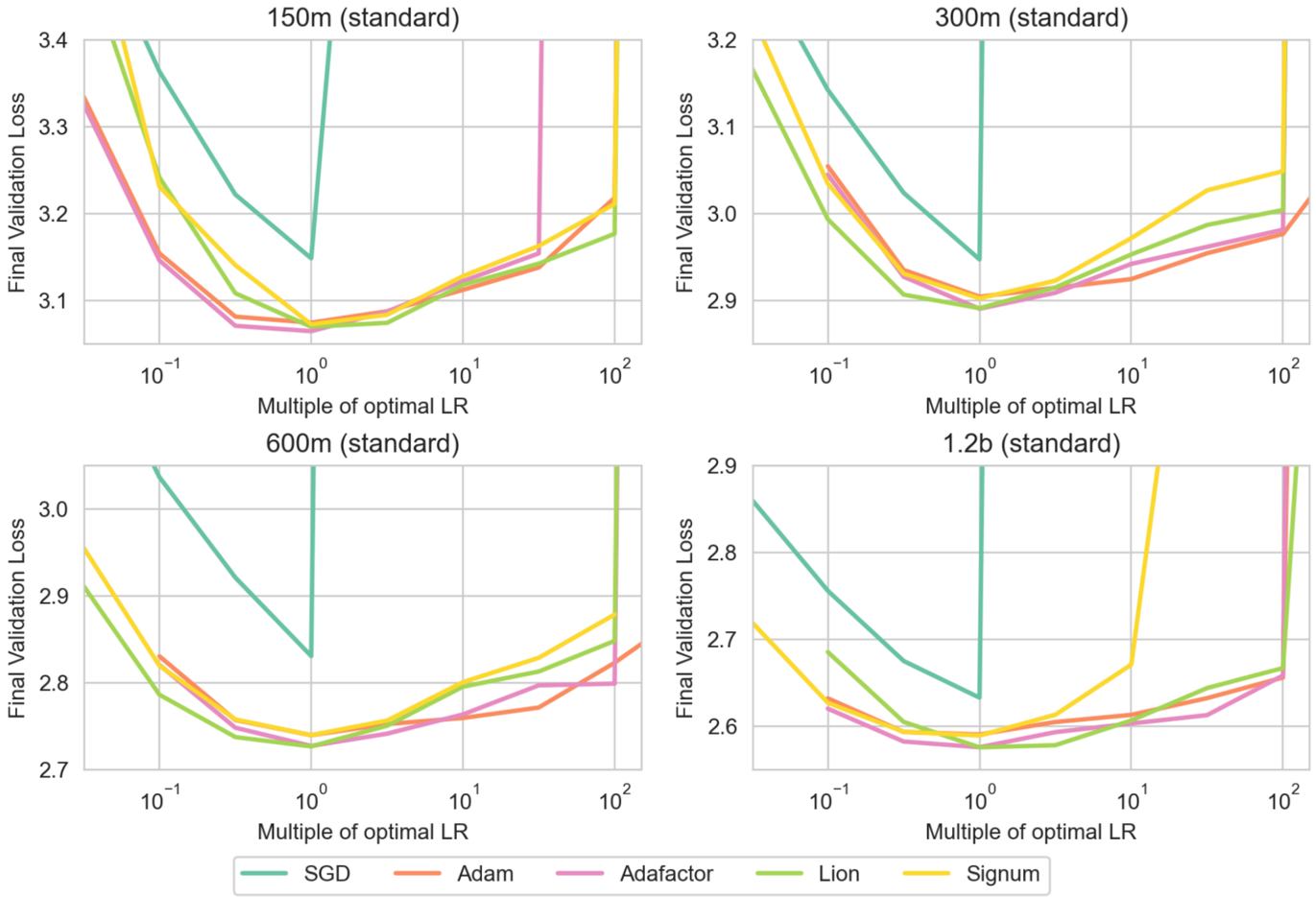
(i) LSTM on LM1B

Shallue et al., “Measuring the effects of data parallelism of neural network training,” JMLR 2019

Optimizers

Optimizers

- We rarely use the vanilla SGD!
  - AdaGrad
  - Adam
  - AdamW
  - RMSProp
  - LBFGS
  - Shampoo
  - Muon
  - (…)
- **Today.** Understand key concepts

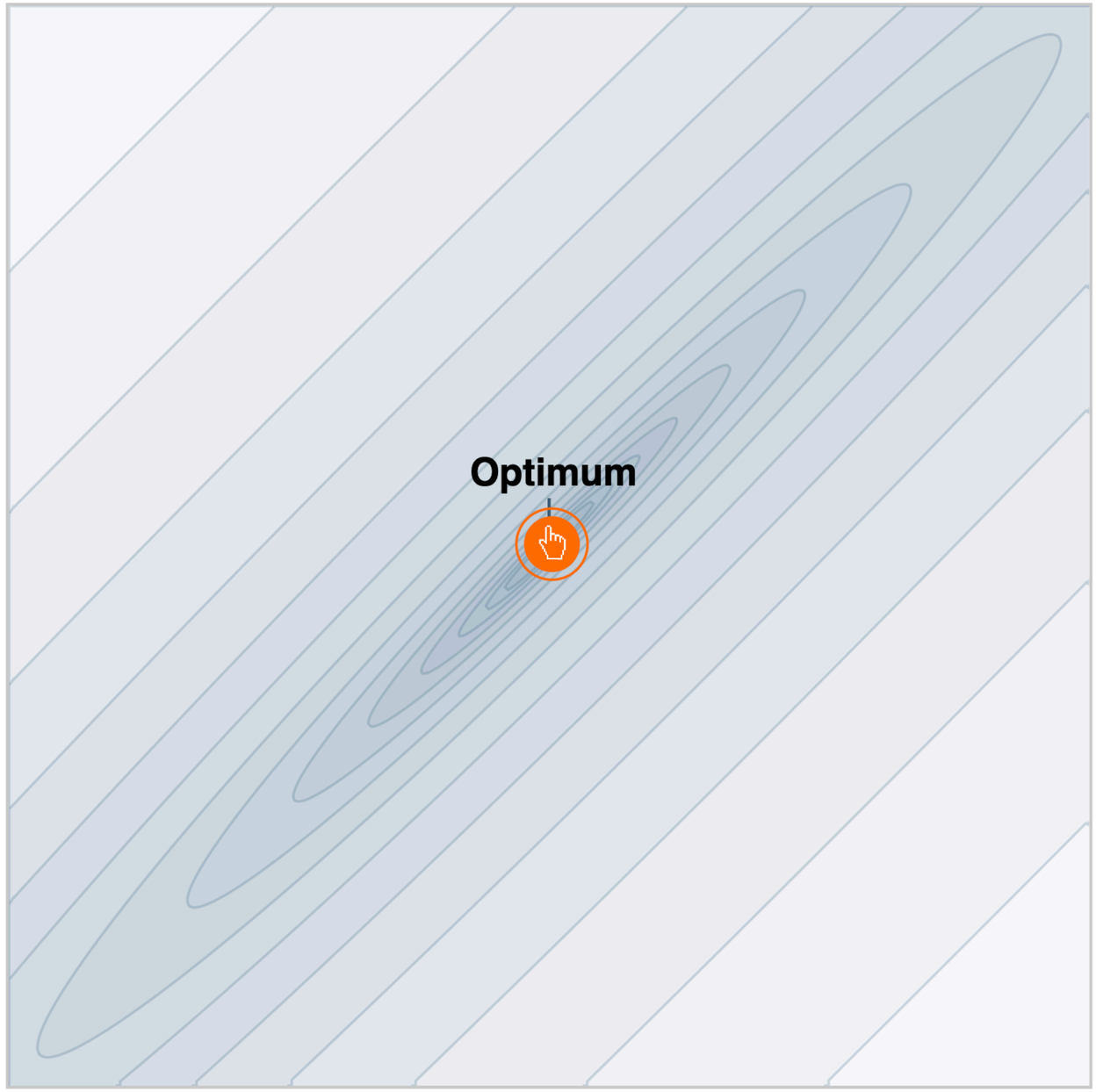


Zhao et al., “Deconstructing What Makes a Good Optimizer for Language Models,” arXiv 2024

Momentum

- Suppose that the risk
  - Changes fast in one direction
  - Changes slow in another

- **Question.** What will GD do?

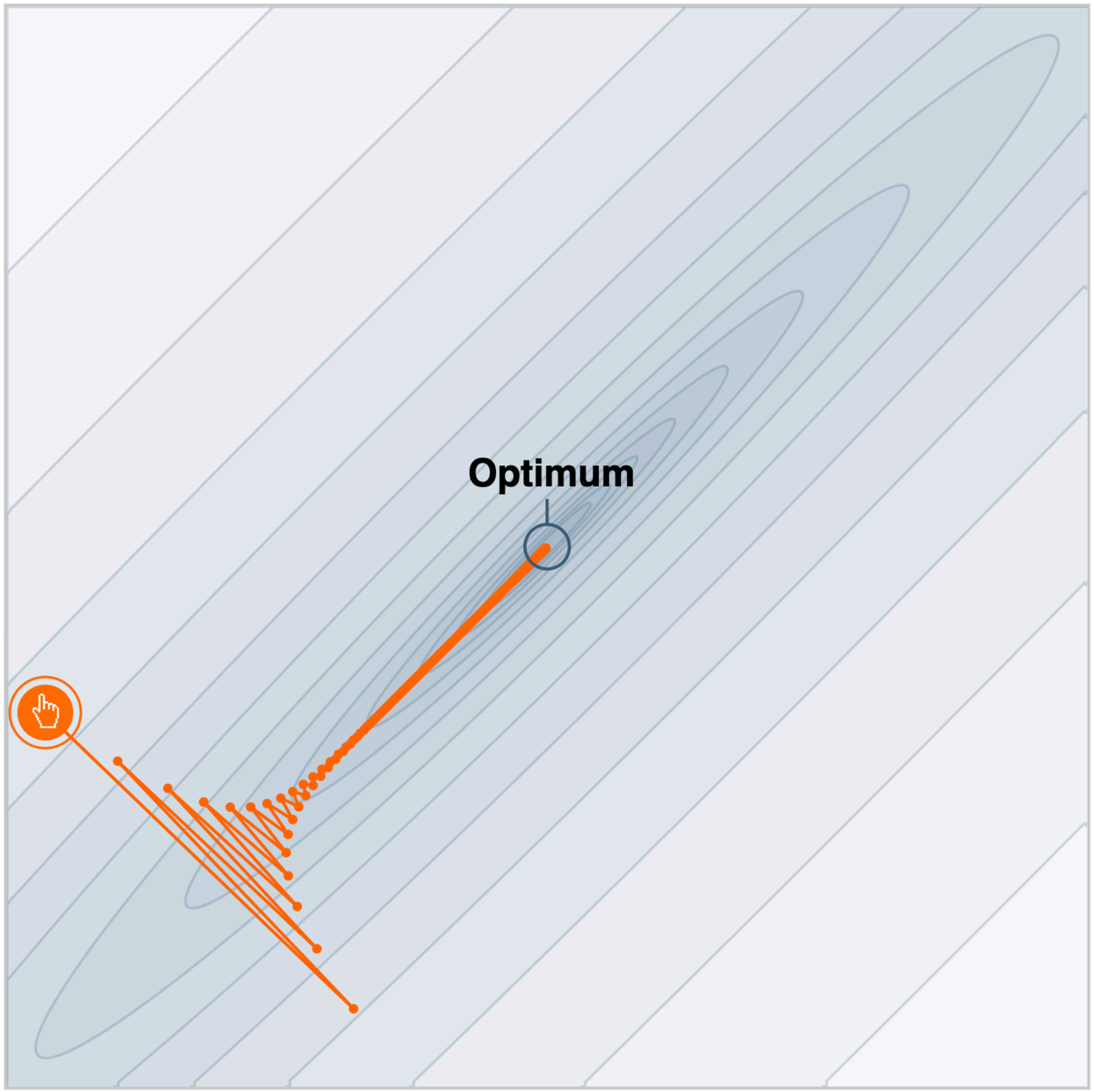


Momentum

- Suppose that the risk
  - Changes fast in one direction
  - Changes slow in another

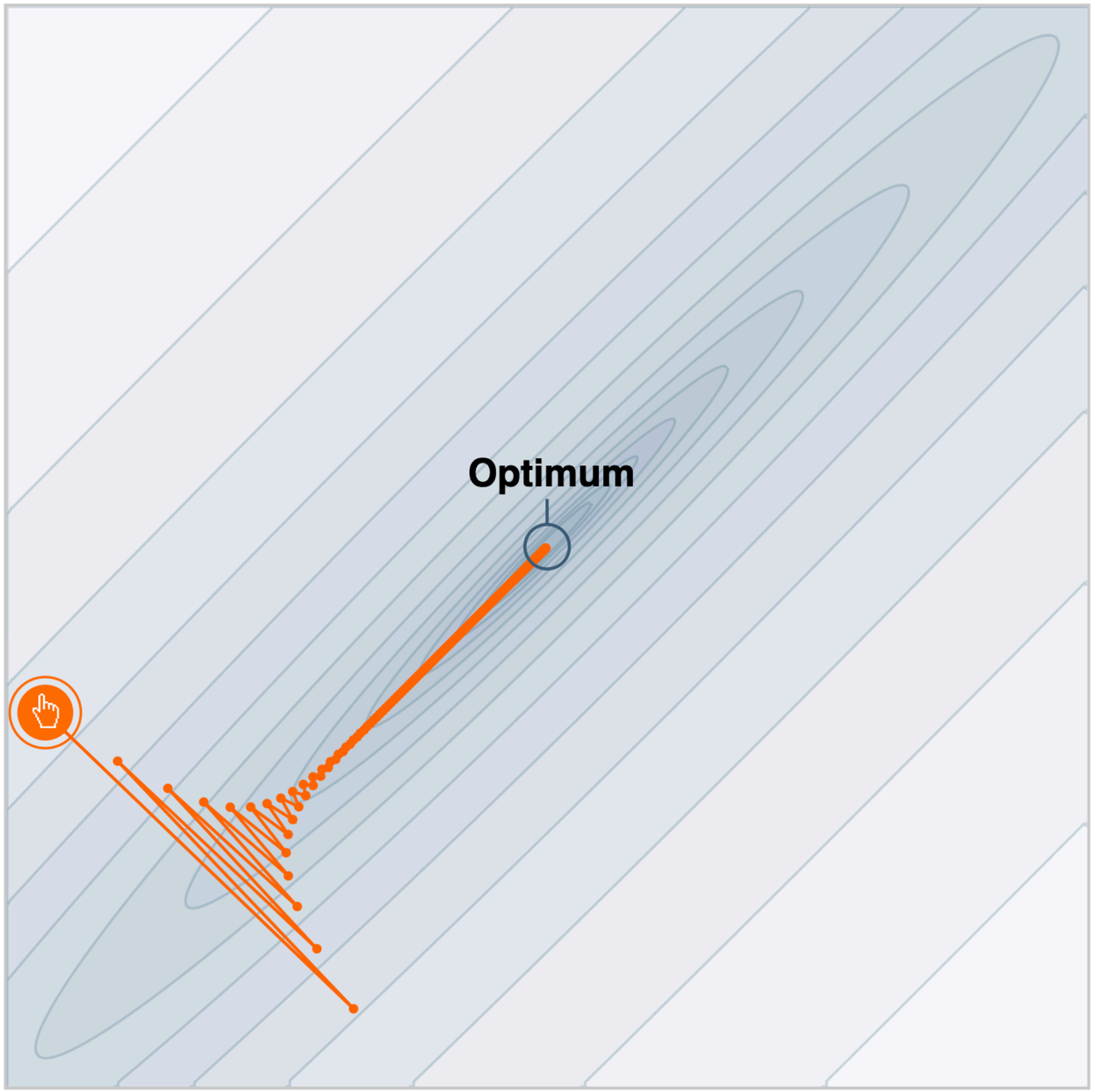
- **Question.** What will GD do?
  - Slow progress in shallow direction
  - High jitter in steep direction

- Note. The loss has a large "condition number"



Momentum

- **Idea.** Let GD have an inertia
  - If we moved to one direction consistently, move faster in that direction



- Original

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla_\theta L(\theta^{(t)})$$

- With Momentum

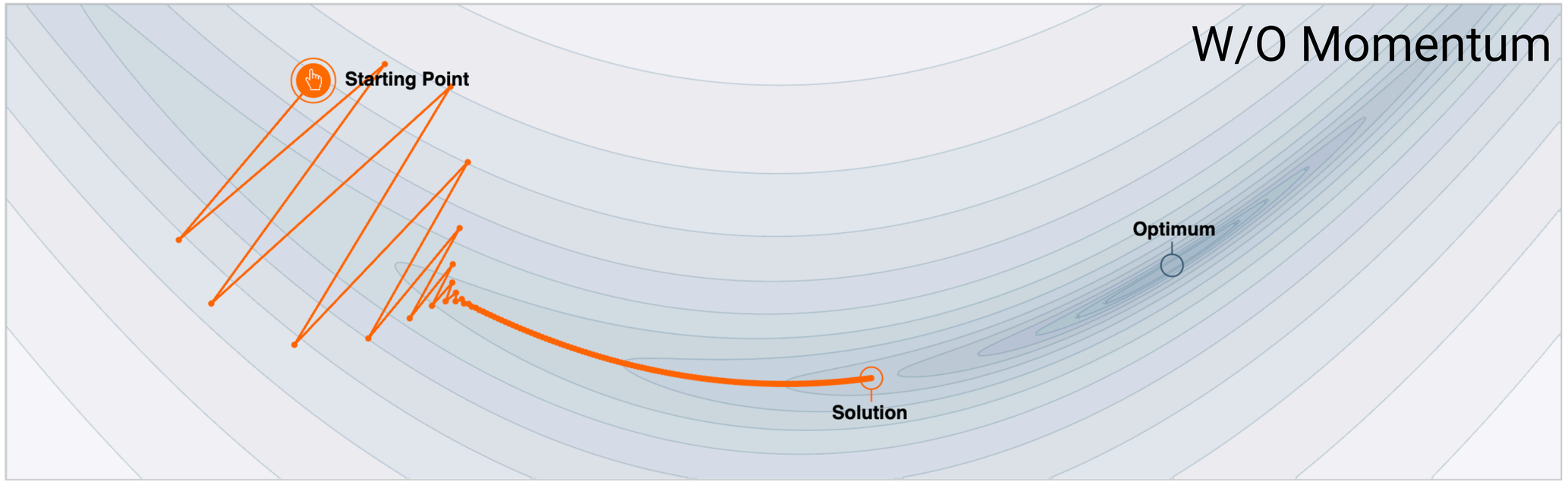
$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot v^{(t+1)}$$

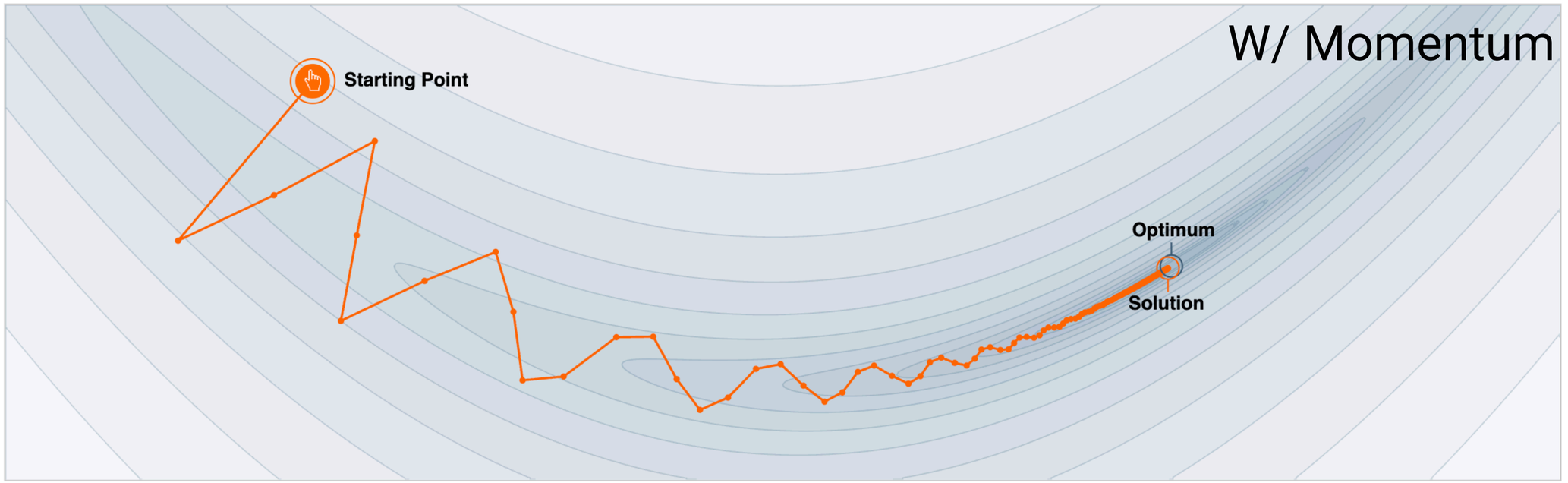
$$v^{(t+1)} = \beta \cdot v^{(t)} + \nabla_\theta L(\theta^{(t)})$$

Accumulated gradients

https://distill.pub/2017/momentum/

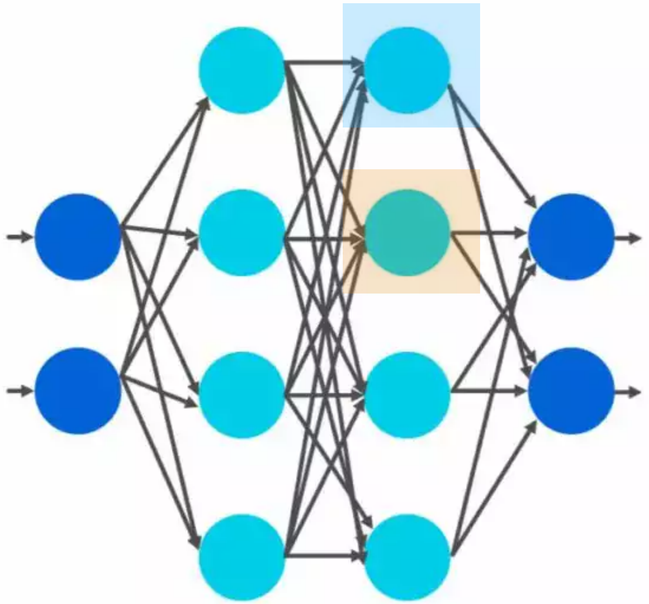
Momentum





Adaptive learning rate

- **Motivation.** Single learning rate may not work well for all parameters
  - Example. Suppose that we have a “cat neuron” and a “dog neuron”
  - If we see much less dogs than cats, then using higher LR for dogs may help run faster



Adaptive learning rate

- **RMSProp.** Conduct the following operation element-wise:
  - Keep the moving average of the gradient scale (squared)

$$g^{(t+1)} = \gamma \cdot g^{(t)} + (1 - \gamma) \cdot \left(\nabla_\theta L(\theta^{(t)}) \right)^2$$

  - Divide the LR by it

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\eta}{\sqrt{g^{(t+1)} + \varepsilon}} \cdot \nabla_\theta L(\theta^{(t)})$$

ε: tiny value, for avoiding division by zero

- **Adam.** RMSProp + Momentum

Adaptive learning rate

- Similar ideas are at the core of modern optimizers
  - **LARS / LAMB.** Layerwise adaptive learning rate
  - **Shampoo.** Does similar thing, but using Hessian matrix
  - **Muon.** Resolves momentum-like motivation in a more elaborate way

Remarks

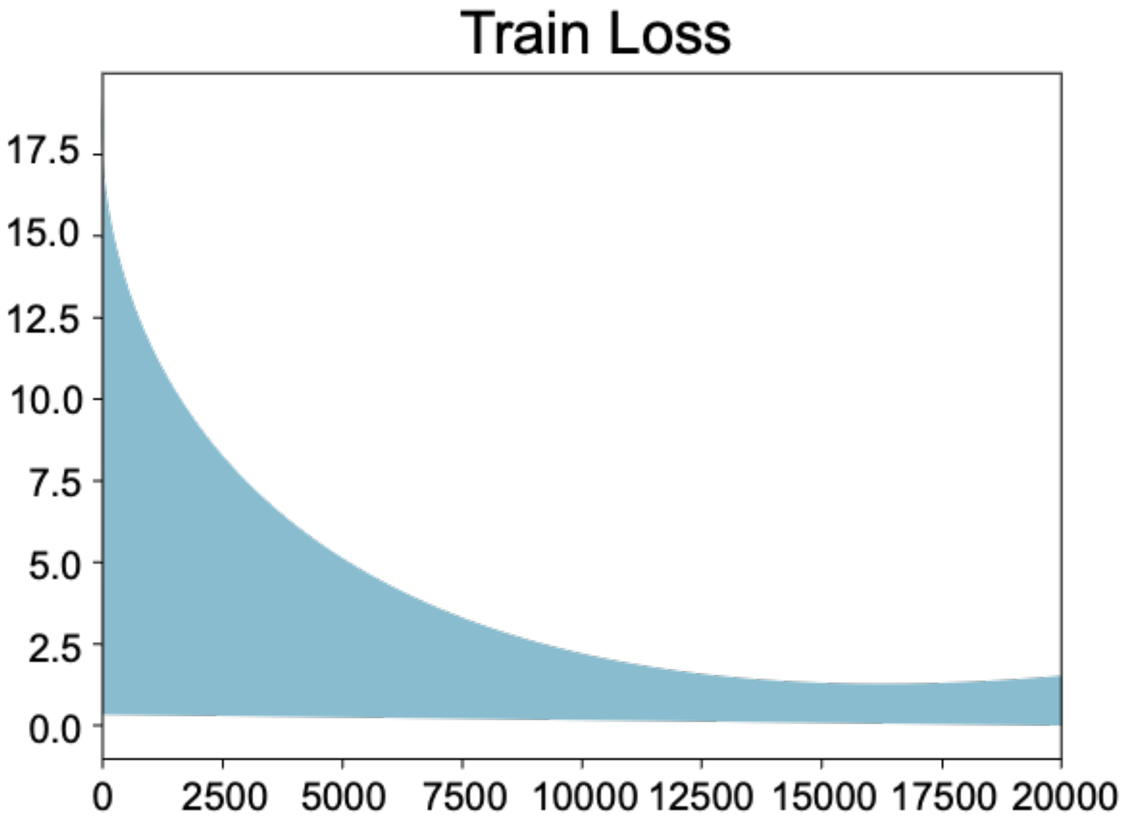
- **Memory.** Optimizer states should also be stored on memory!
  - For each 32bit parameter, we keep:
    - Gradient (32bit)
    - Momentum (32bit)
    - Adaptive LR (32bit)

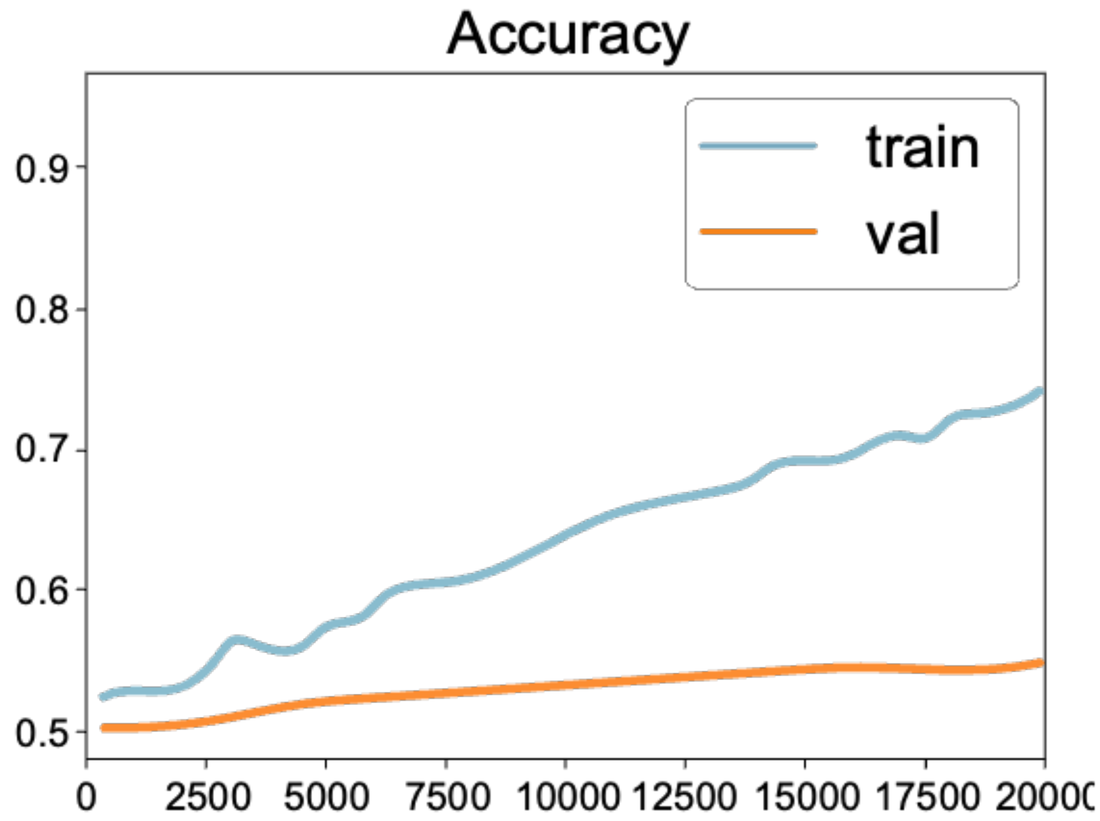
- **Tuning.** Advanced optimizers introduce more parameters to tune
  - Much training computation needed for the best performance

Regularization

Beyond training error

- Better optimization algorithms help reduce the training loss
  - But we actually care about the test performance — how can we reduce the gap?

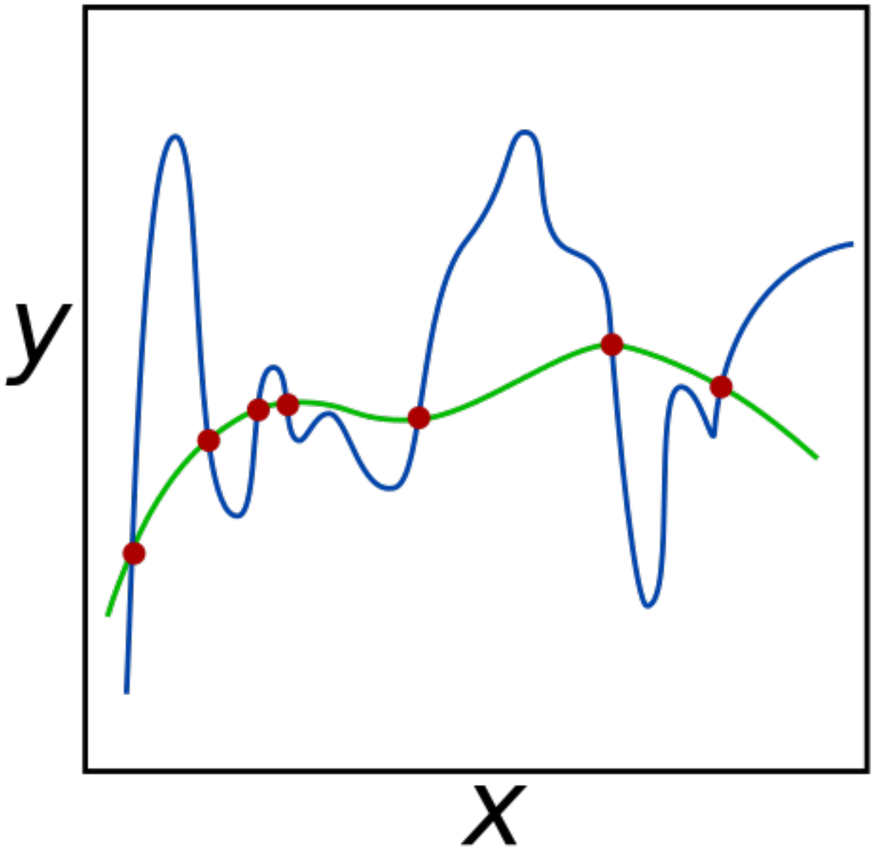




Beyond training error

- Most regularization methods follow the principle of Occam's razor
  - Whenever possible, use simpler models

- Simplicity may be:
  - Number of weight parameters
  - Scale of the weights
  - Prediction confidence …



Regularization through loss

- **Idea.** Add complexity term to the loss function

$$\frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_\theta(\mathbf{x}_i)) \quad + \quad \text{complexity}(\theta)$$

  - This corresponds to a Lagrangian relaxation:
    - Constraint given as:

$$\text{complexity}(\theta) \leq \tau$$

Regularization through loss

- **Example.** L2 regularization
  - Philosophy. Use smaller ℓ_2-norm solution if possible
    - Use the loss:

$$L(\theta) + \lambda \cdot \|\theta\|_2^2$$

    - The gradient update becomes:

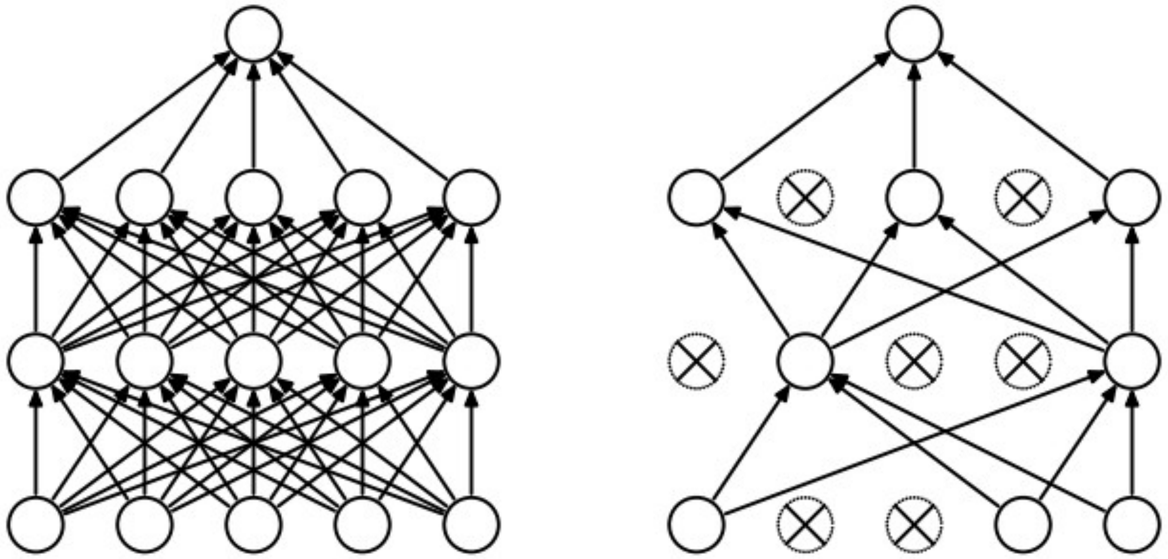
$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla_\theta (L(\theta) + \lambda \cdot \|\theta\|_2^2)$$

    - Equivalent to a simpler-to-implement form, called “weight decay”

$$\theta^{(t+1)} = (1 - 2\eta\lambda)\theta^{(t)} - \eta \cdot \nabla_\theta L(\theta)$$

Non-differentiable complexities

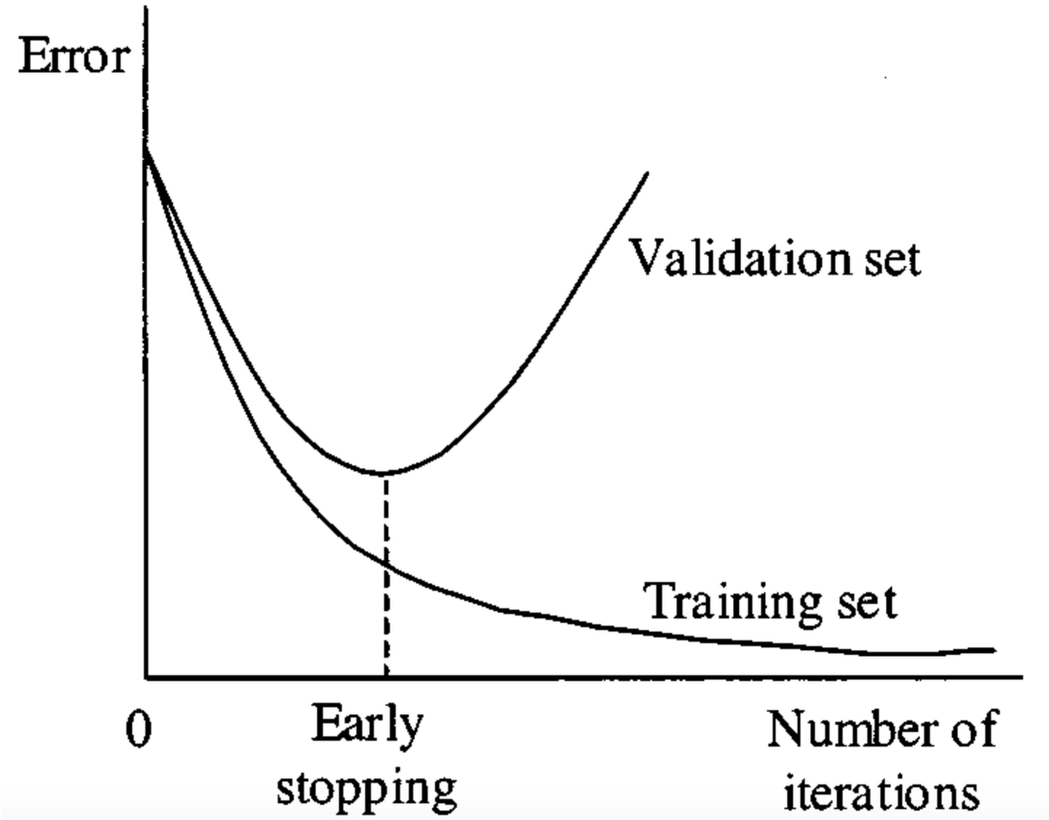
- For non-differentiable complexities, design a customized algorithm
- **Example.** Dropout
  - Philosophy. Use fewer parameters if possible
    - During the training, randomly remove each neuron w.p. p
    - For inference, rescale the weights back to $1/p$



(a) Standard Neural Net (b) After applying dropout.

Non-differentiable complexities

- **Example.** Early stopping
  - Philosophy. Use parameters that can be discovered in a shorter time
    - Pause training when validation error does not drop anymore



Remarks

- **Optimization.** Sometimes, regularization makes the optimization easier
- Example. Consider solving a least-square problem

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$$

  - Ordinary solution will give: $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$
    - No guarantee on invertibility
  - If we add ℓ_2 penalty:

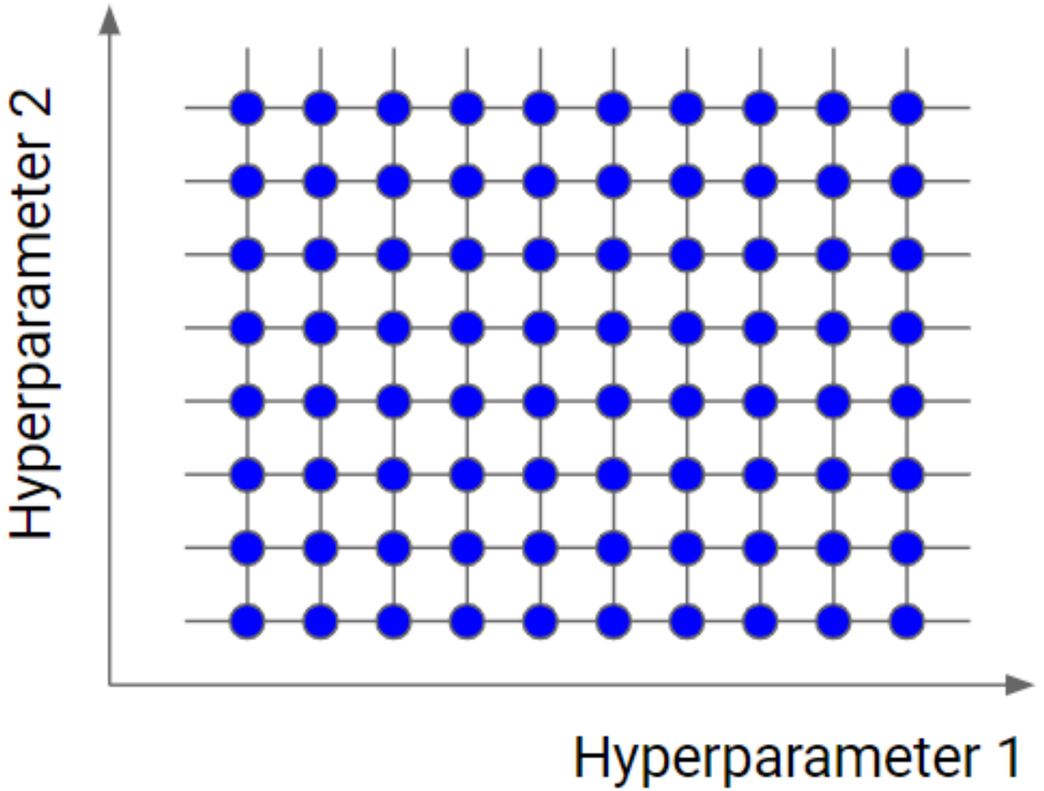
$$\min_{\mathbf{w}} \left(\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2 \right)$$

  - The solution will be: $\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_n)^{-1}\mathbf{X}^\top \mathbf{y}$
    - Always invertible

Hyperparameter tuning

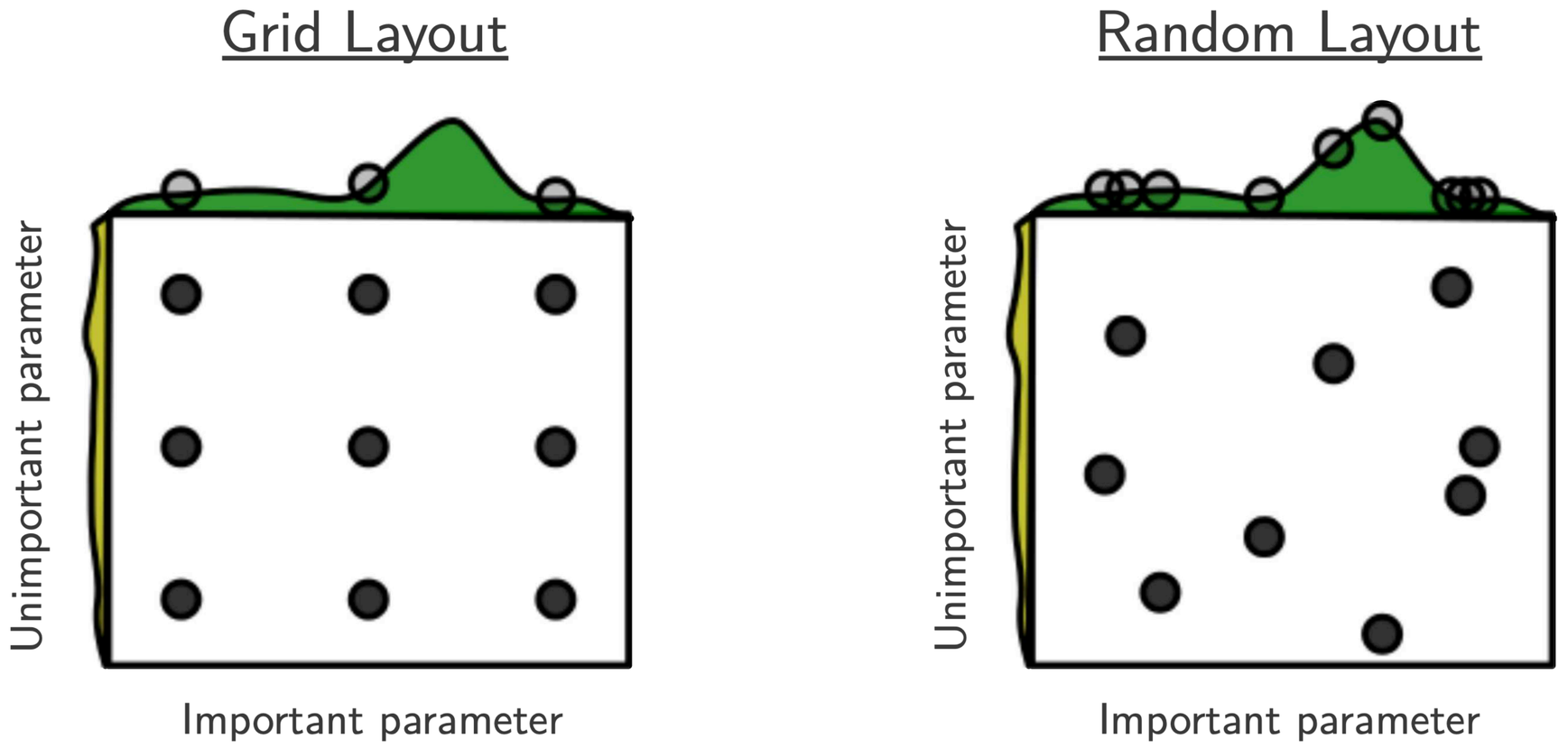
Strategy

- How do we select the hyperparameters?
- **Grid search.** Use coarse-to-fine grids, to reduce the number of trials
  - Sometimes, use log scales – e.g., $\{10^{-2}, 10^{-3}, 10^{-4}, \cdots\}$



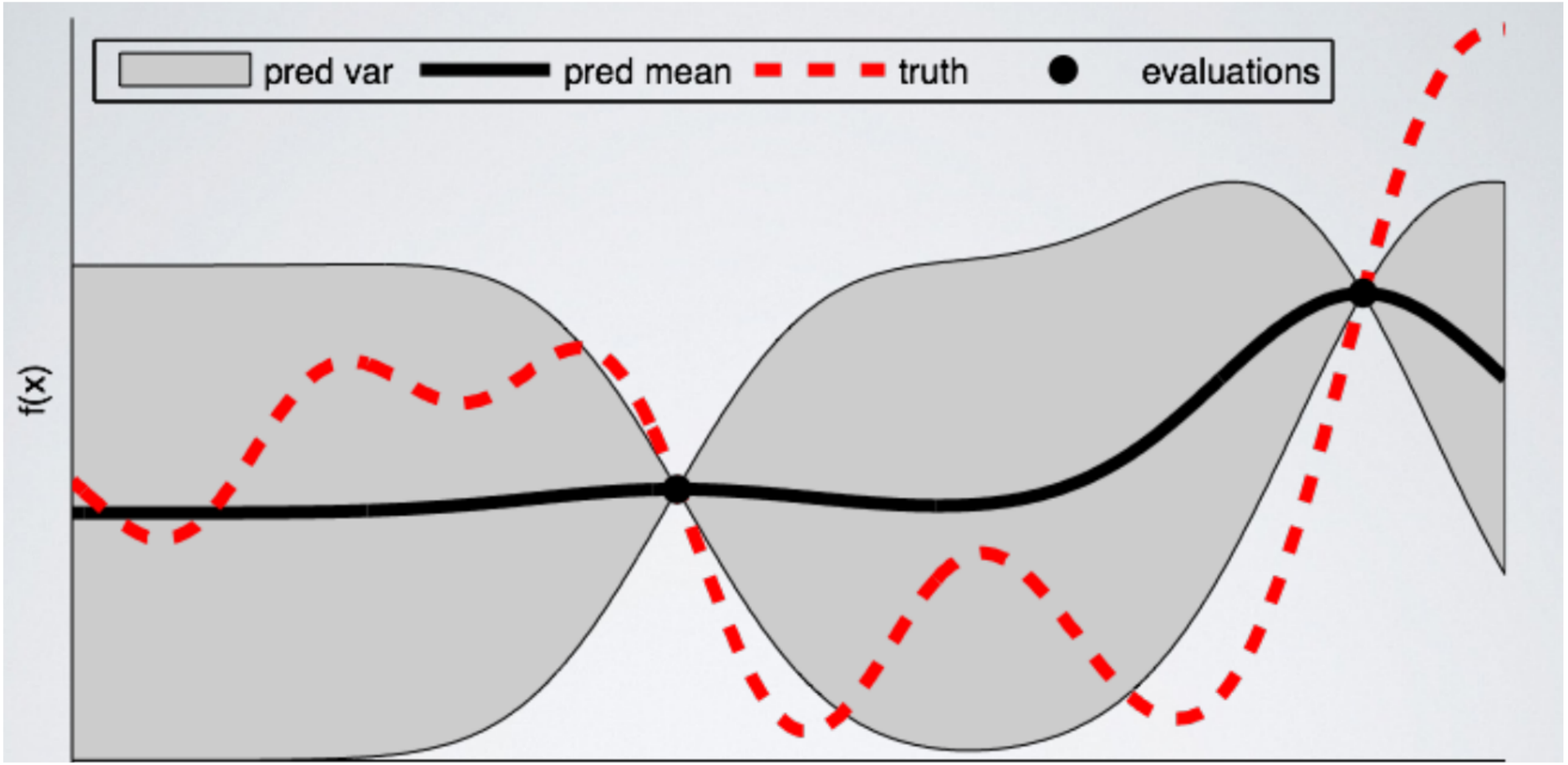
Strategy

- **Random search.** Use randomly sampled hyperparameters
  - Has a larger effective sample size, when more than two distinct hyperparameters should be tuned



Strategy

- **Bayesian approach.** Bayesian HP optimization is also used:
  - Model the performance-HP relationship with some smooth function



Strategy

- **Transfer.** In some cases, we can tune HP on a small model and use them for training the larger models
  - Requires a special parameterization (called μ-parameterization)

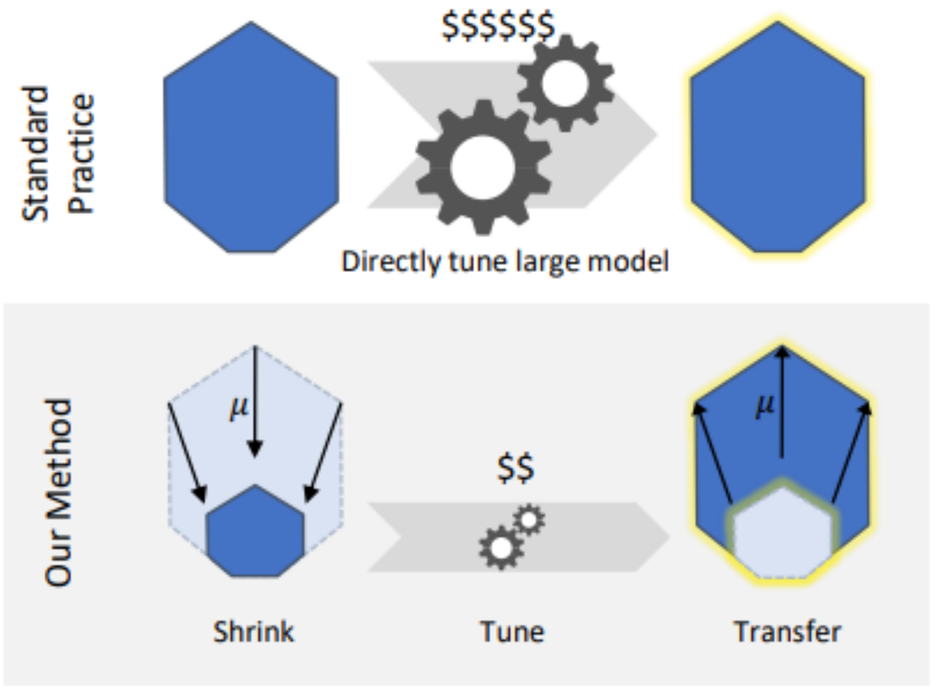


Figure 2: Illustration of μTransfer

</lecture 14>