Training Optimization

Danna Gurari
University of Colorado Boulder
Spring 2025

https://dannagurari.colorado.edu/course/neural-networks-and-deep-learning-spring-2025/

Review

- Last lecture:
  - Gradient descent: how neural networks learn
  - Mathematical foundation of gradient descent: derivatives
  - Applying gradient descent to train neural networks
  - Training example
- Assignments (Canvas):
  - Problem set 1 due earlier today
  - Problem set 2 due in 1 week
- Questions?

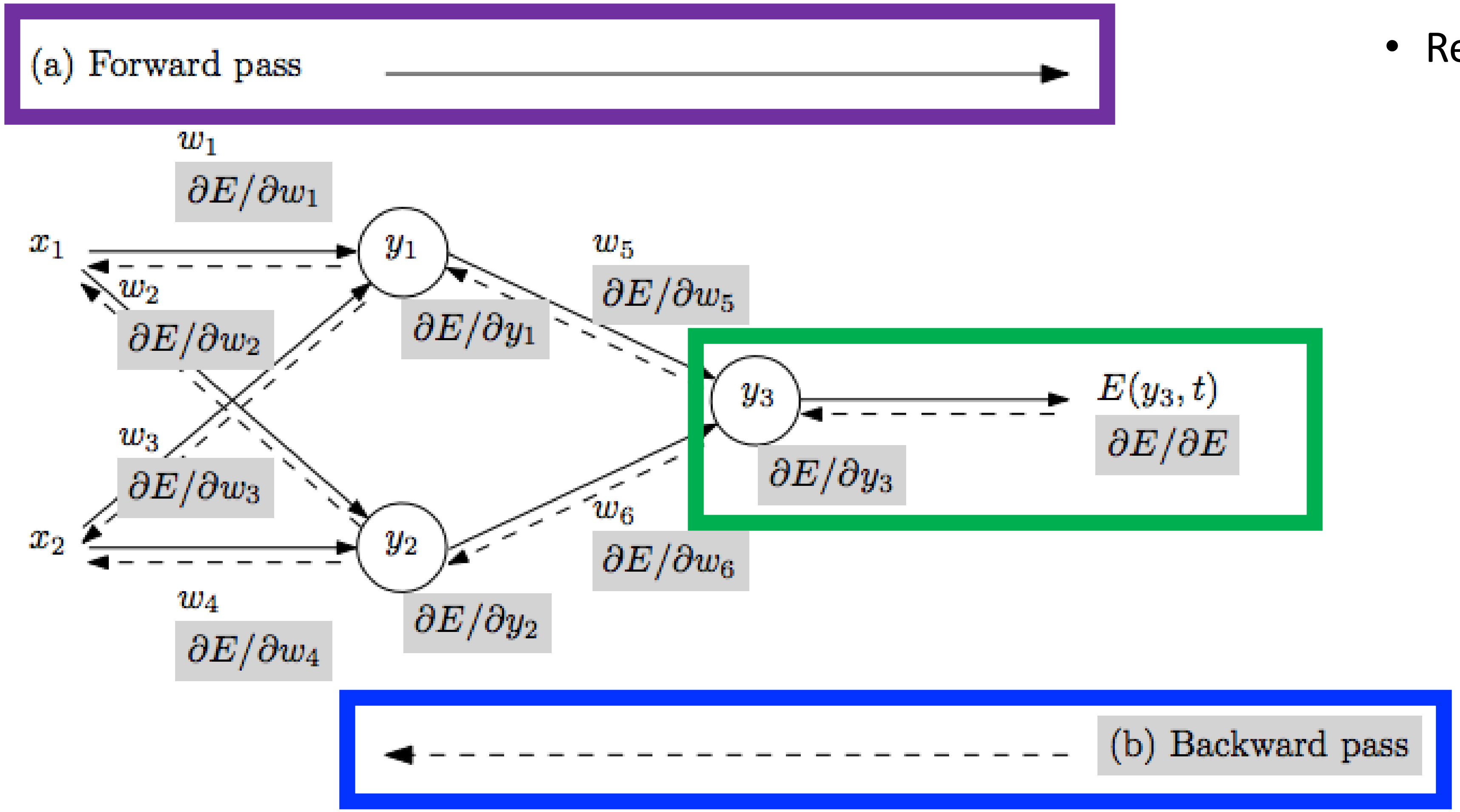
Today’s Topics

- Motivation: effective gradients for learning
- Initializing parameters
- Initializing data
- Following the gradient (optimization)
- Programming tutorial

Today's Topics

- Motivation: effective gradients for learning
- Initializing parameters
- Initializing data
- Following the gradient (optimization)
- Programming tutorial

Recall: Neural Network Training Approach



- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure error of the model's predictions on training data using a loss function
  3. **Backward pass**: calculate gradients to determine how each model parameter contributed to model error
  4. Update each parameter using calculated gradients

Key challenge: maintaining sufficient gradients for learning

Baydin et al. Automatic Differentiation in Machine Learning: a Survey. 2018

Today's Scope: "Looking Under the Hood" at How to Maintain Good Gradients

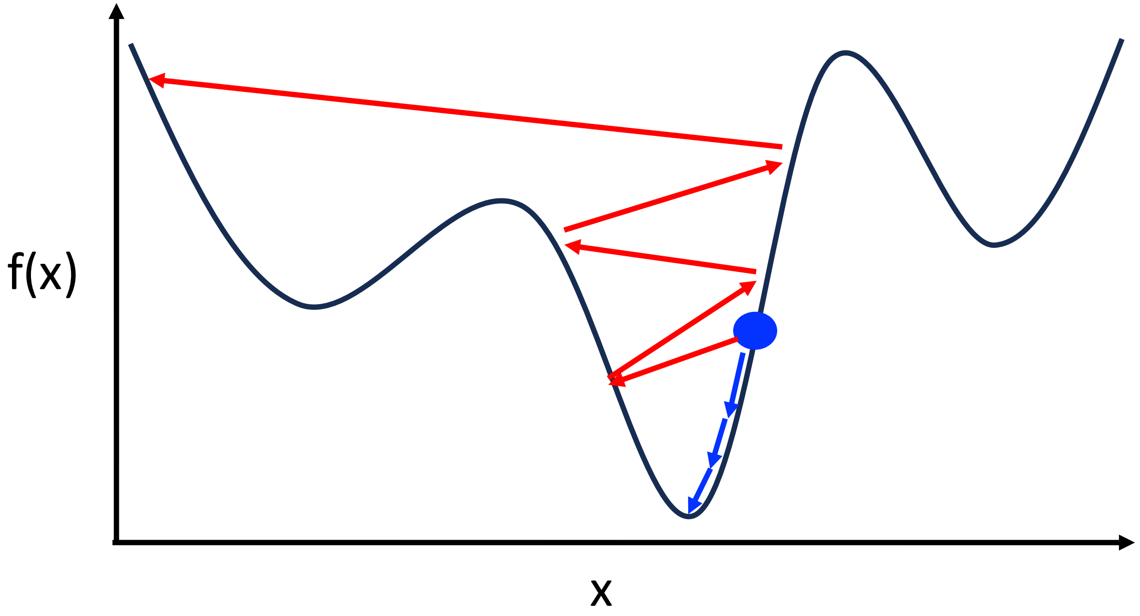
Recall: **algorithm** learns from **data** on a **processor** patterns for making predictions

Challenge: sufficient gradients (fuel) to learn (drive anywhere)

Today's scope:

https://www.etftrends.com/etfs-the-importance-of-looking-under-the-hood/

How Can We Arrive at the Global Loss?



1. Choose good starting point

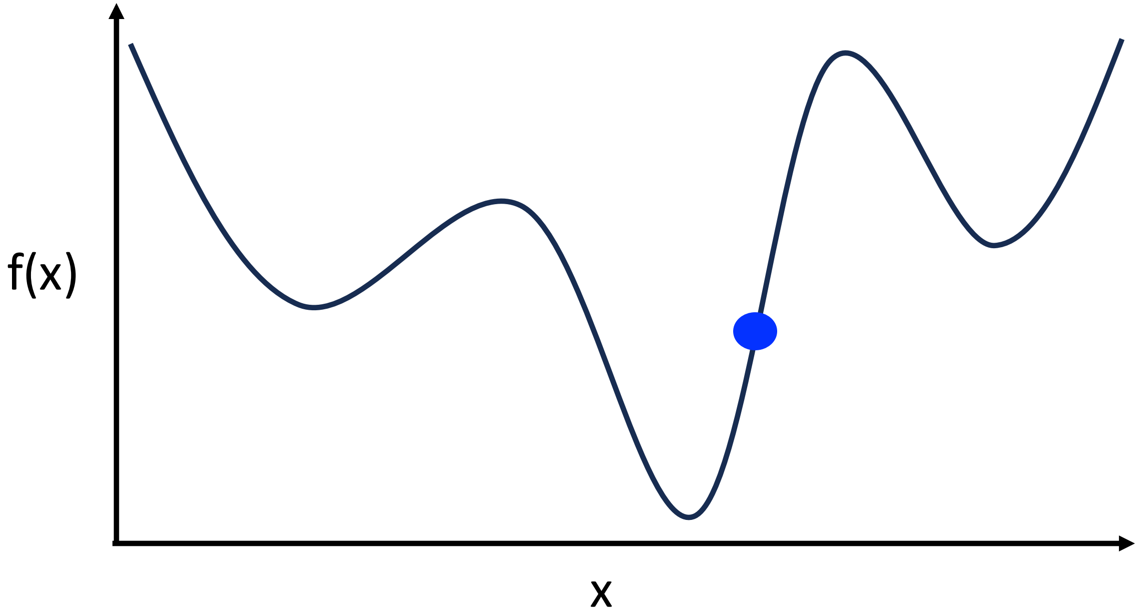
2. Choose good step sizes for following the gradient

   (or avoid bad step sizes)

Today’s Topics

- Motivation: effective gradients for learning
- Initializing parameters
- Initializing data
- Following the gradient (optimization)
- Programming tutorial

How Can We Arrive at the Global Loss?

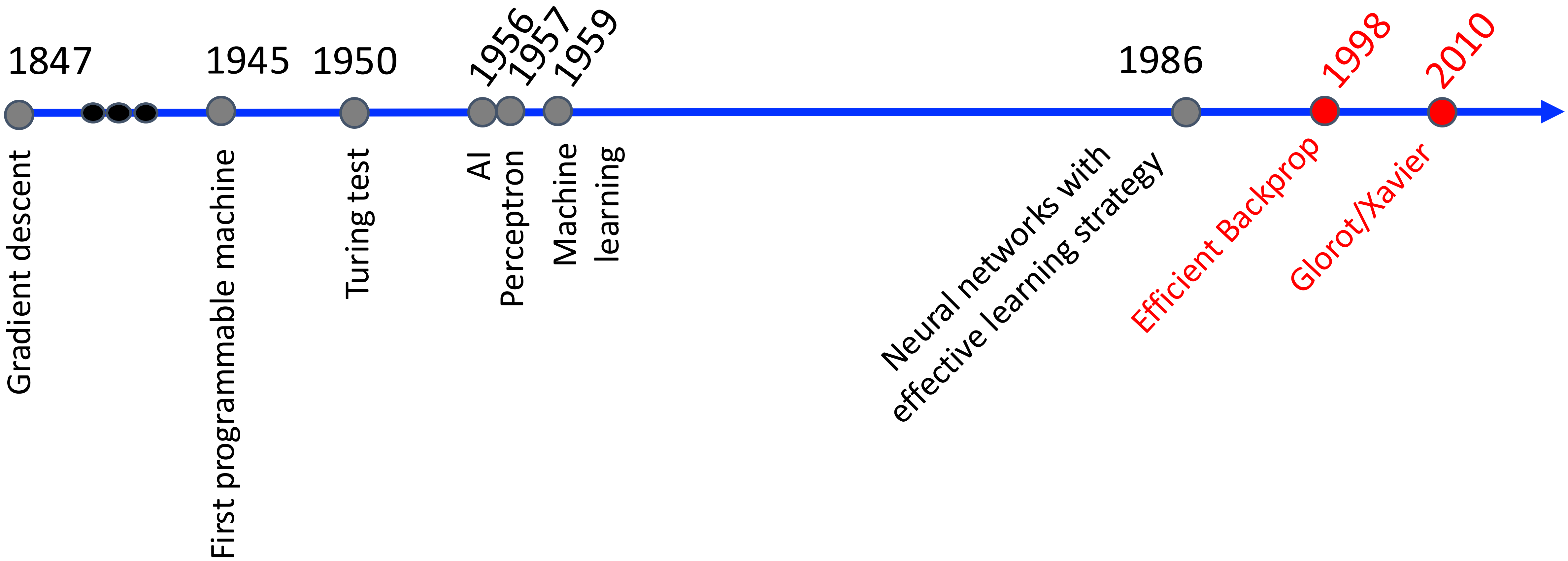


1. Choose good starting point

2. Choose good step sizes for following the gradient

   (or avoid bad step sizes)

Popular Initializations: Historical Context



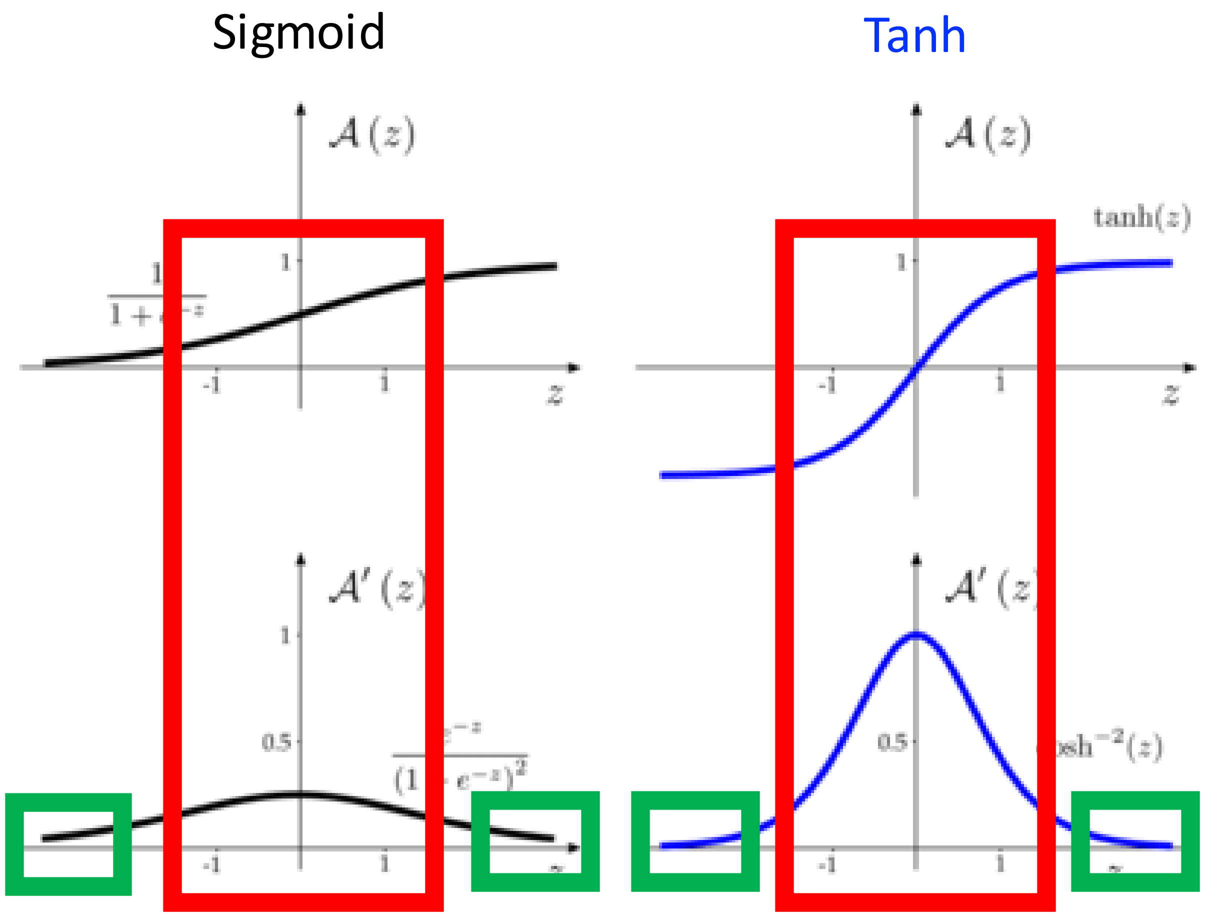
https://pyimagesearch.com/2021/05/06/understanding-weight-initialization-for-neural-networks/

Popular Initializations

Approach: enable suitable gradients for learning
- weights initialized to random, small values, where the scale of “small” is key
- biases set to 0

They avoid:
- weight symmetry, which prevents learning since neurons compute same functions
- large weights; why?

Idea: Choose Parameters that Facilitate Learning



Masi et al. Journal of the Mechanics and Physics of Solids. 2021

Units with very large or small "z" values have slow/no learning; why?

Small derivatives limit amount model parameters can change with gradient descent

Idea: normalize parameters so derivative lies in a "*good range*", where learning can occur

e.g., Xavier/Glorot Initialization

uniform distribution in
interval between two values

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

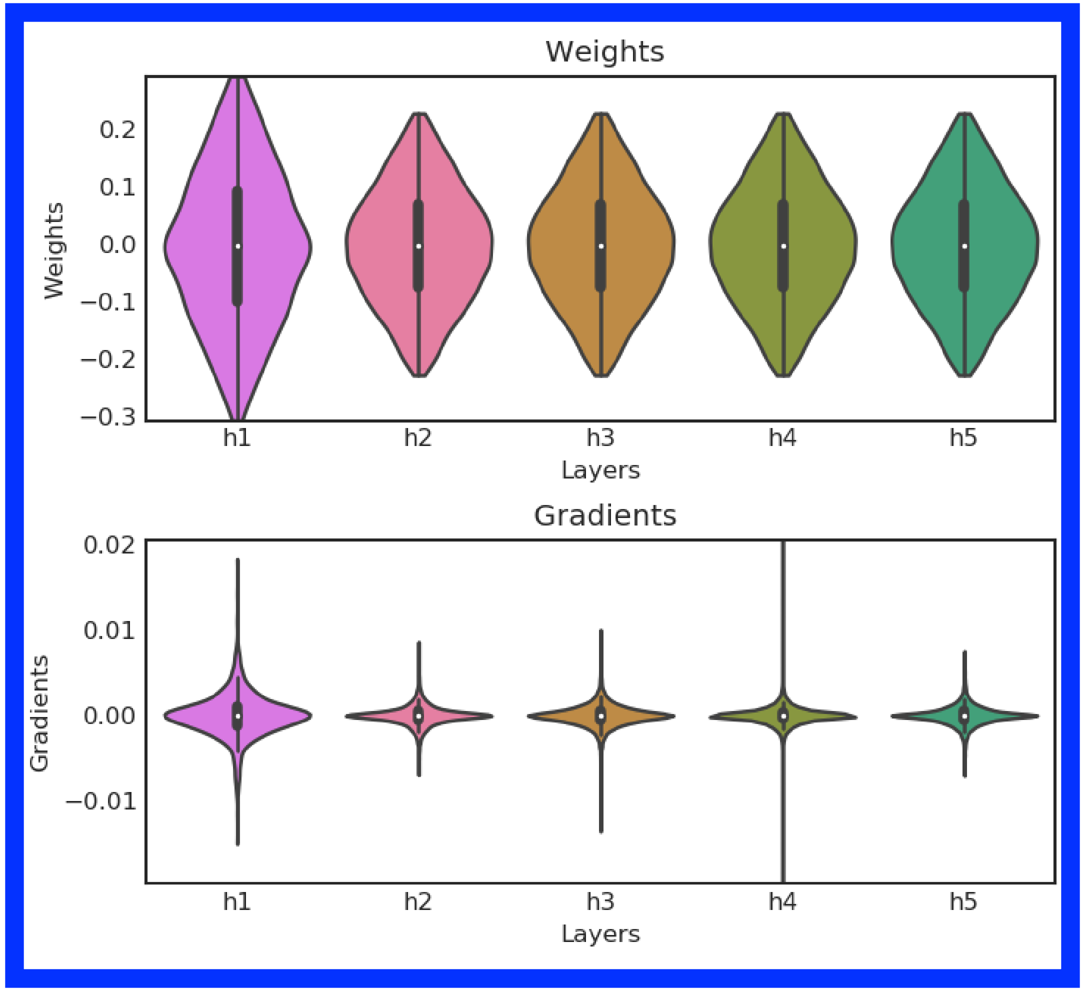
weight matrix between
layers *j* and *j+1*

fan in*:* # of neurons
entering layer *j+1*

fan out*:* # of neurons
leaving layer *j+1*

It is common for the scale of “small” for weights to be
determined by many neurons are entering or leaving a layer

e.g., Xavier/Glorot Initialization

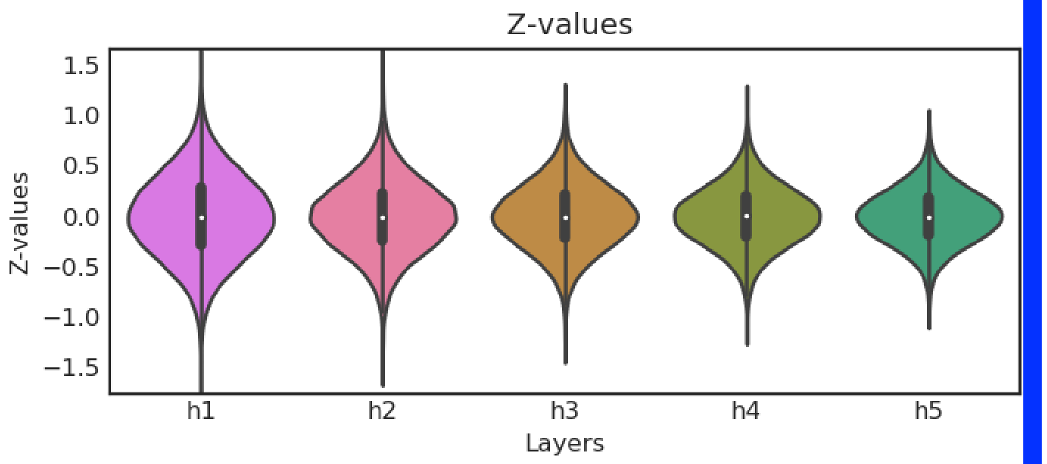


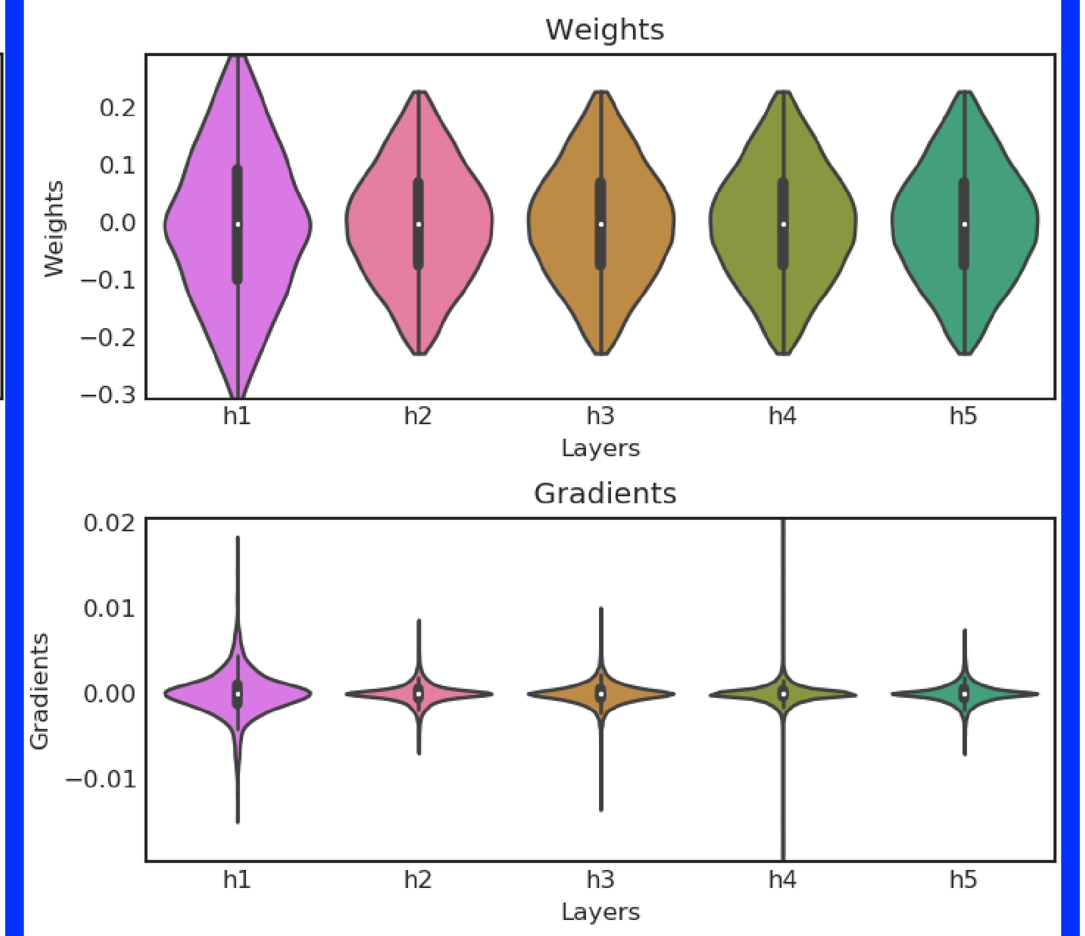
Weights set so resulting **gradients** can support learning, with similar variance across layers

https://towardsdatascience.com/hyper-parameters-in-action-part-ii-weight-initializers-35aee1a28404

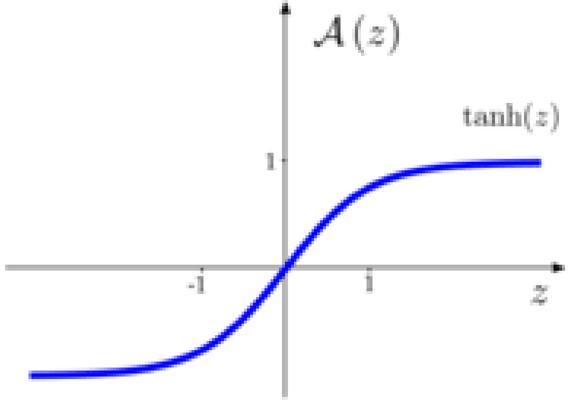
e.g., Xavier/Glorot Initialization

Activation: tanh - Initializer: Glorot Normal - Epoch 0





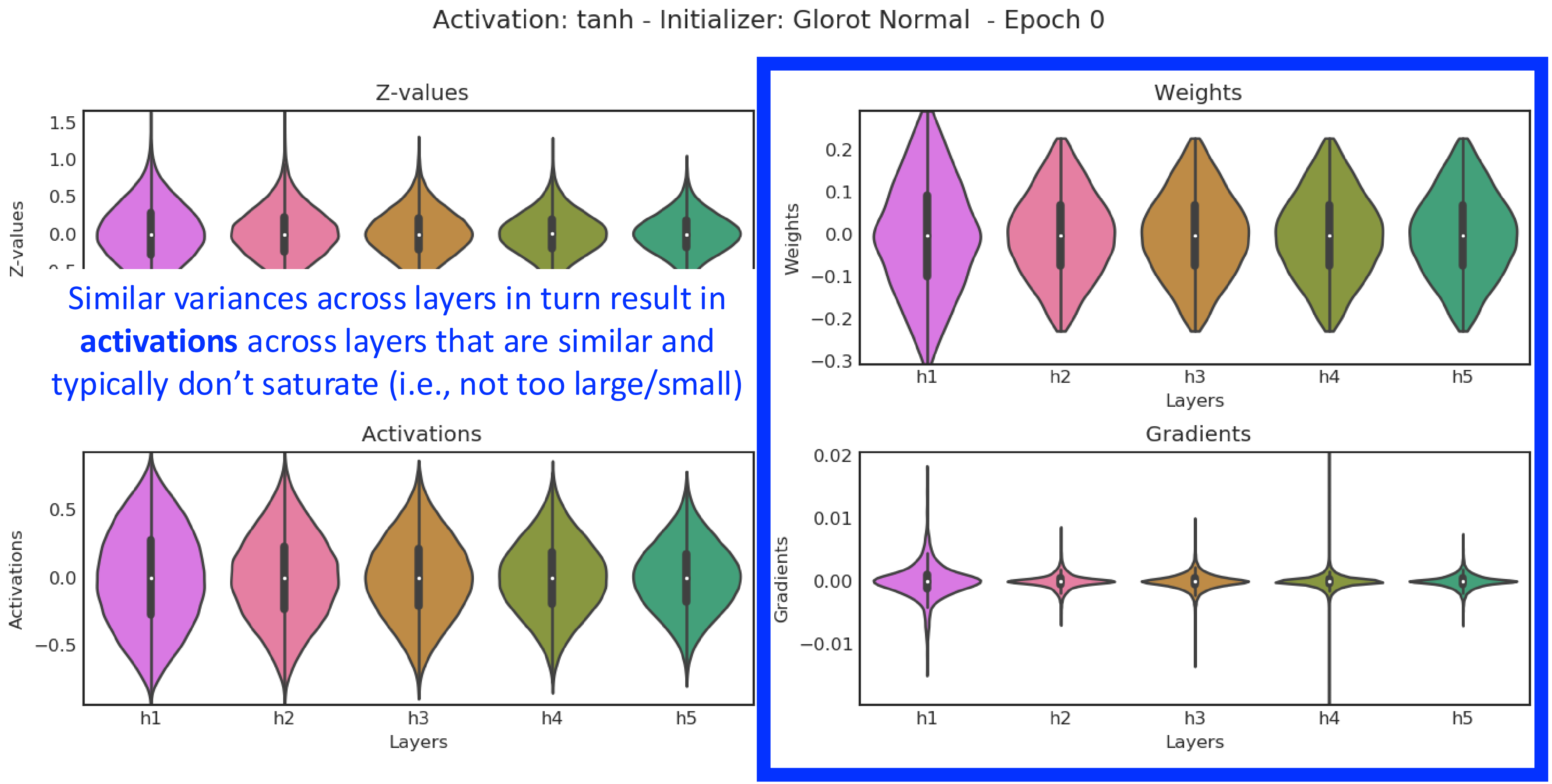
Initial weights cause **weighted sums** to have similar variance across layers in range of (1, -1)



Weights set so resulting **gradients** can support learning, with similar variance across layers

https://towardsdatascience.com/hyper-parameters-in-action-part-ii-weight-initializers-35aee1a28404

e.g., Xavier/Glorot Initialization

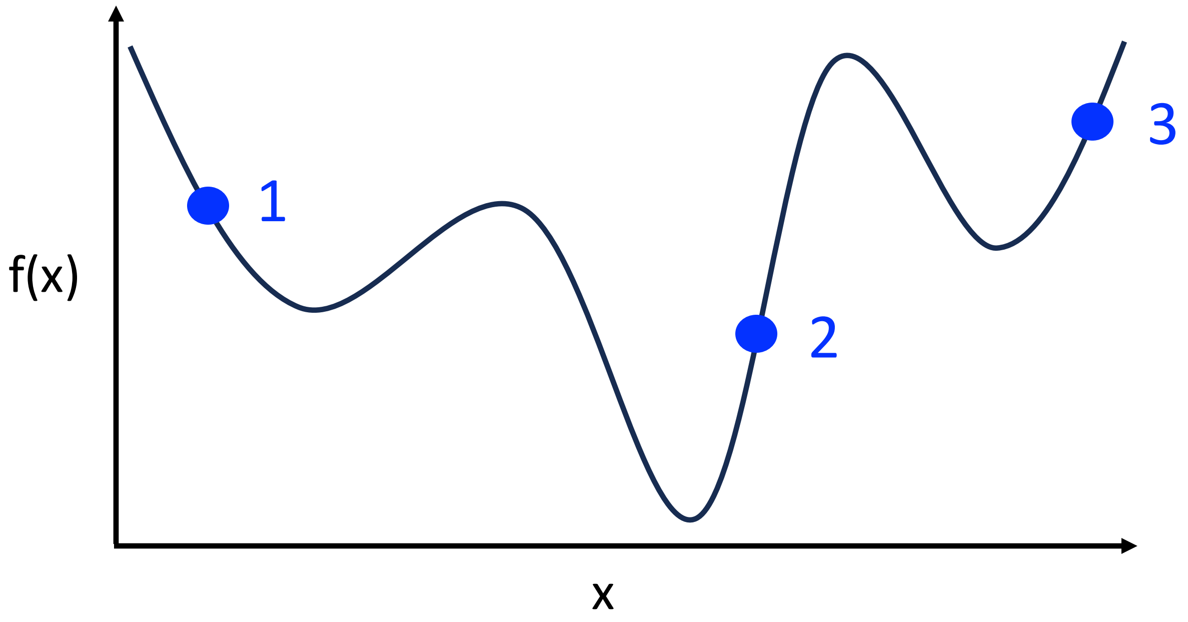


Similar variances across layers in turn result in **activations** across layers that are similar and typically don't saturate (i.e., not too large/small)

Weights set so resulting **gradients** can support learning, with similar variance across layers

https://towardsdatascience.com/hyper-parameters-in-action-part-ii-weight-initializers-35aee1a28404

Practical Note: Where to Start When Learning?

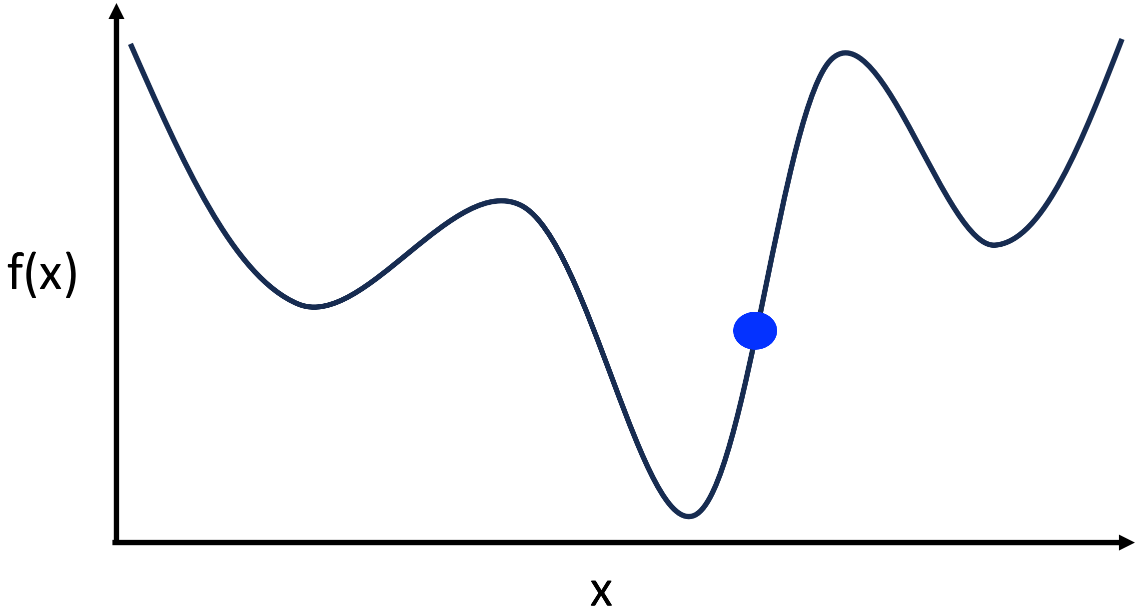


May need to repeat initialization to arrive closer to the target solution to accelerate learning and improve final performance

Today's Topics

- Motivation: effective gradients for learning
- Initializing parameters
- Initializing data
- Following the gradient (optimization)
- Programming tutorial

How Can We Arrive at the Global Loss?



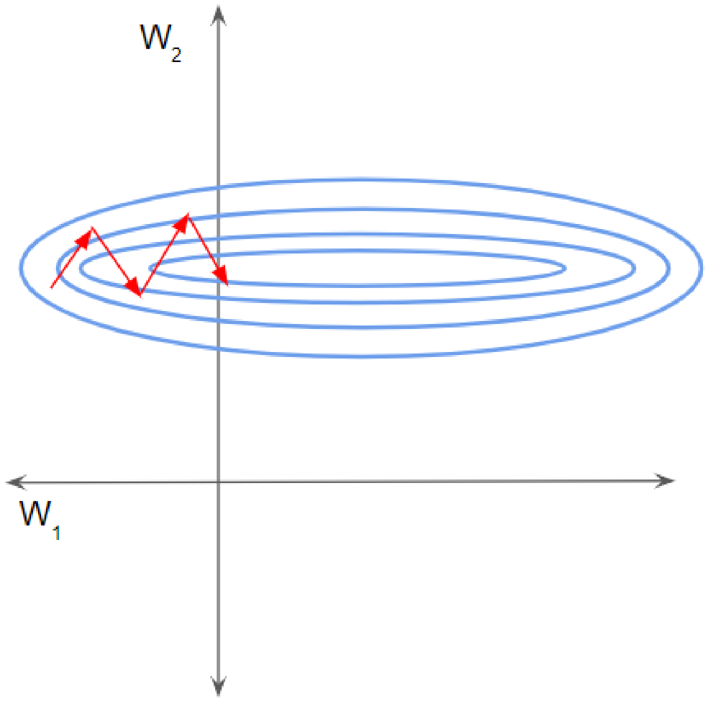
1. Choose good starting point

2. Choose good step sizes for following the gradient

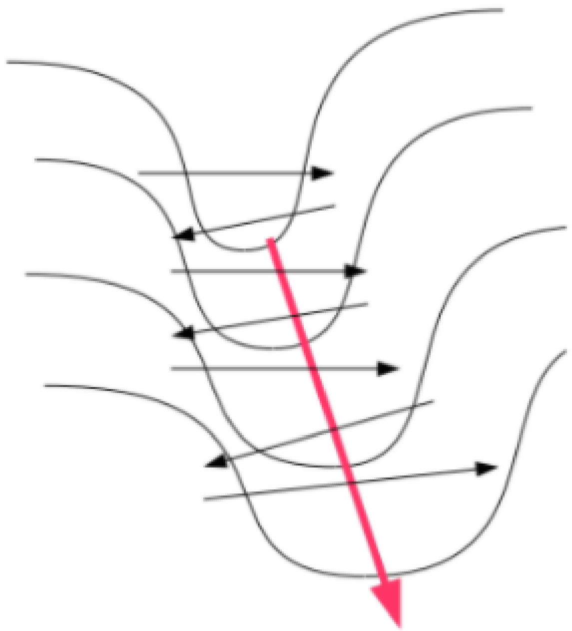
   (or avoid bad step sizes)

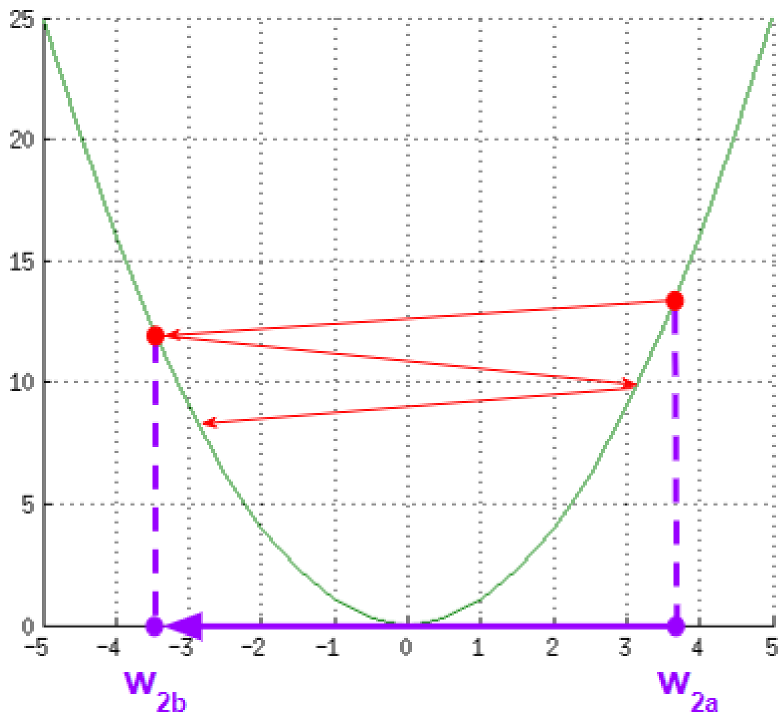
In Parallel, Data Typically Initialized So Features Have the Same Scales to Accelerate Learning

e.g., 2D loss function:



Inefficient bouncing can occur during learning when larger updates are needed for some weights to minimize the loss during gradient descent



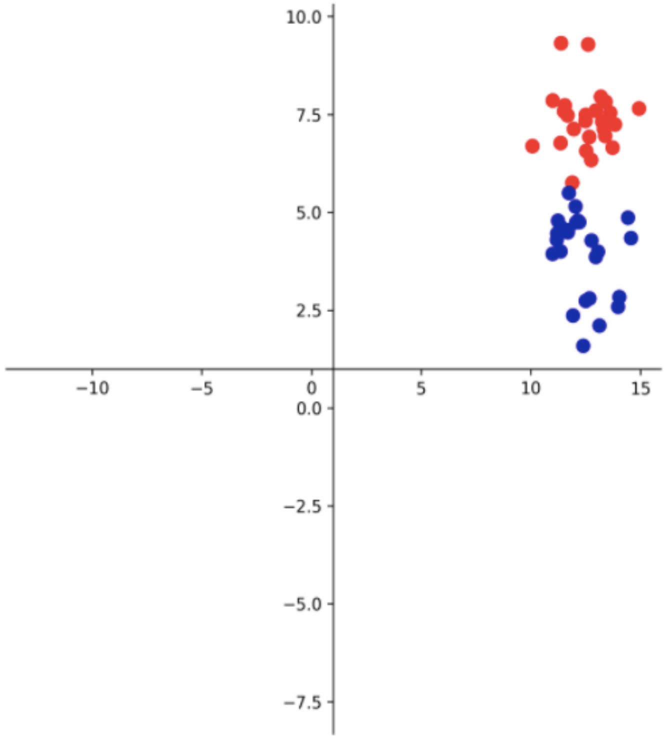


https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

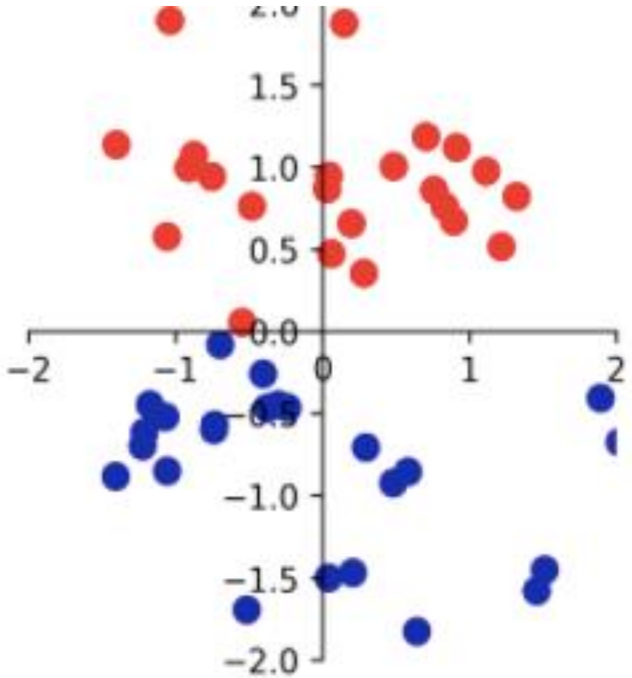
In Parallel, Data Typically Initialized So Features Have the Same Scales to Accelerate Learning

Learning simplified by standardizing input data so mean is 0 and standard deviation 1

Original data:

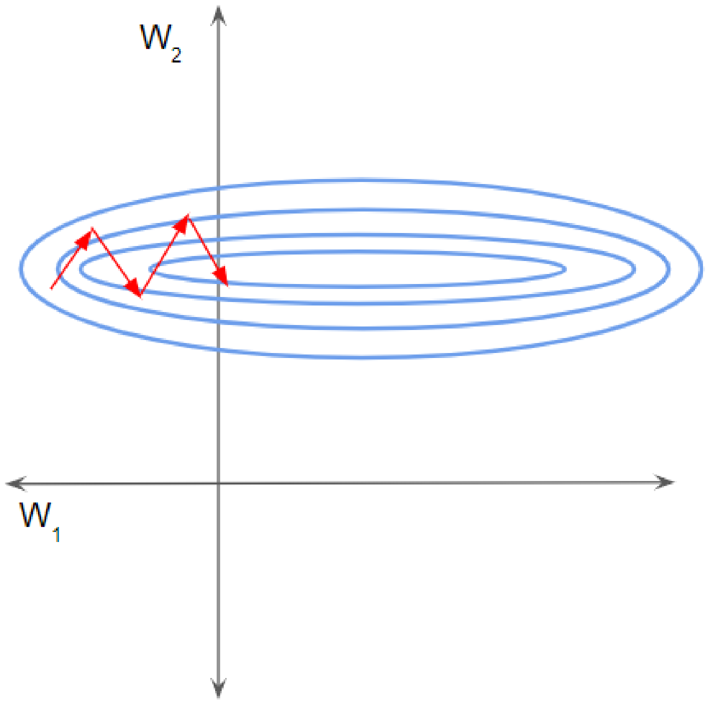


Standardized data:

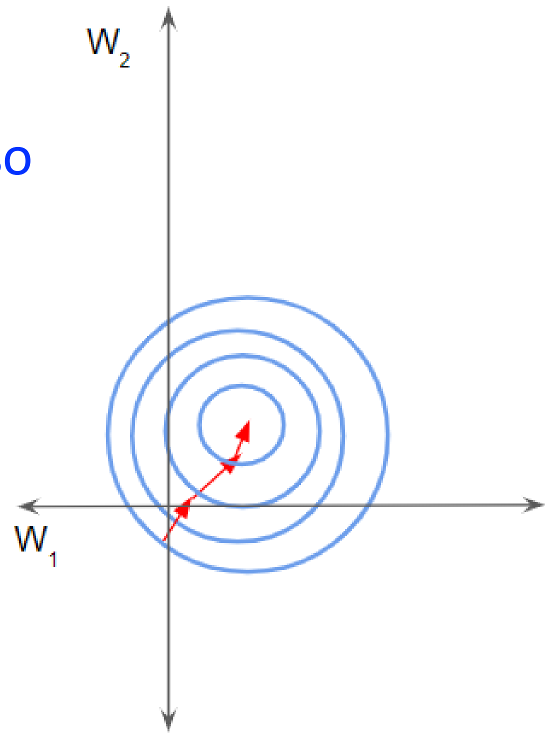


https://github.com/amueller/introduction_to_ml_with_python/blob/master/03-unsupervised-learning.ipynb

In Parallel, Data Typically Initialized So Features Have the Same Scales to Accelerate Learning



Standardization changes loss function so that the gradient descent can more smoothly arrive at the minimum!

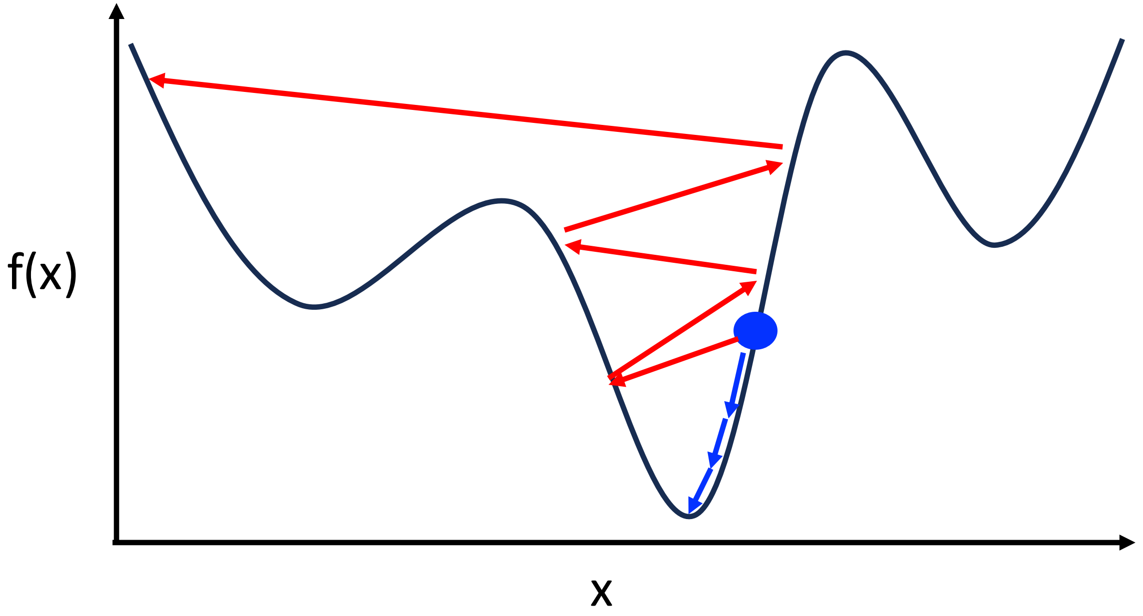


https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Today’s Topics

- Motivation: effective gradients for learning
- Initializing parameters
- Initializing data
- Following the gradient (optimization)
- Programming tutorial

How Can We Arrive at the Global Loss?

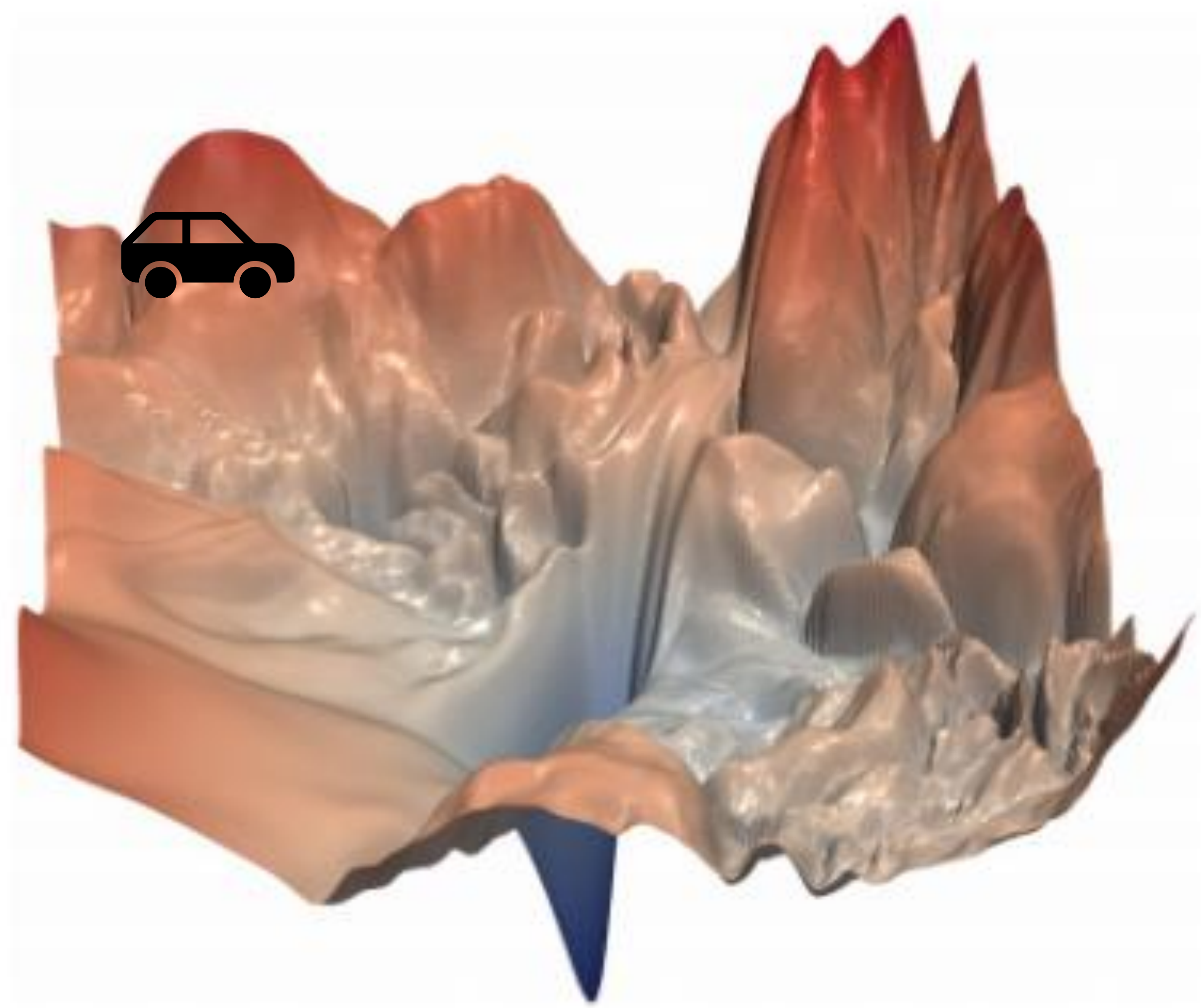


1. Choose good starting point

2. Choose good step sizes for following the gradient

   (or avoid bad step sizes)

How Can We Arrive at the Global Loss?



Example loss/error surface (vertical axis values) based on all possible weight pairs (two weights in horizontal plane)

What could go wrong when driving down the error surface?
- get stuck in a ditch (local optimum)
- zig-zag on a ravine (little gradient)
- arrive at a flat plateau (no gradient)

Many ways for trying to avoid these issues!

Li et al. Visualizing the Loss Landscape of Neural Nets. Neurips 2018.

How Can We Arrive at the Global Loss?

Example loss/error surface (vertical axis values) based on all possible weight pairs (horizontal plane with two weights)

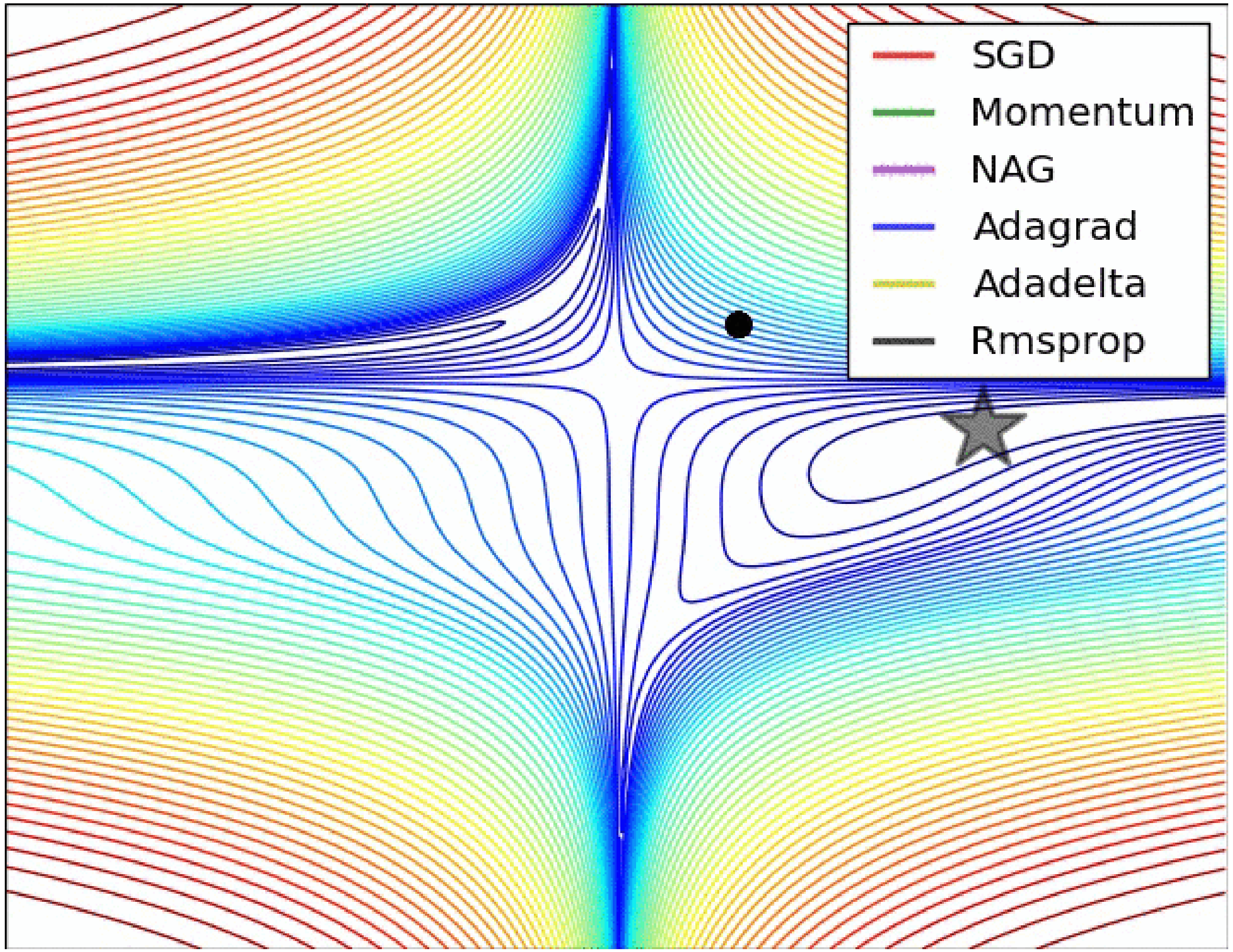
What could go wrong when driving down the error surface?

- get stuck in a ditch (local optimum)
- zig-zag on a ravine (little gradient)
- arrive at a flat plateau (no gradient)

Many ways for trying to avoid these issues!

Popular Optimization Methods

Trajectory of methods on contours of a loss surface:



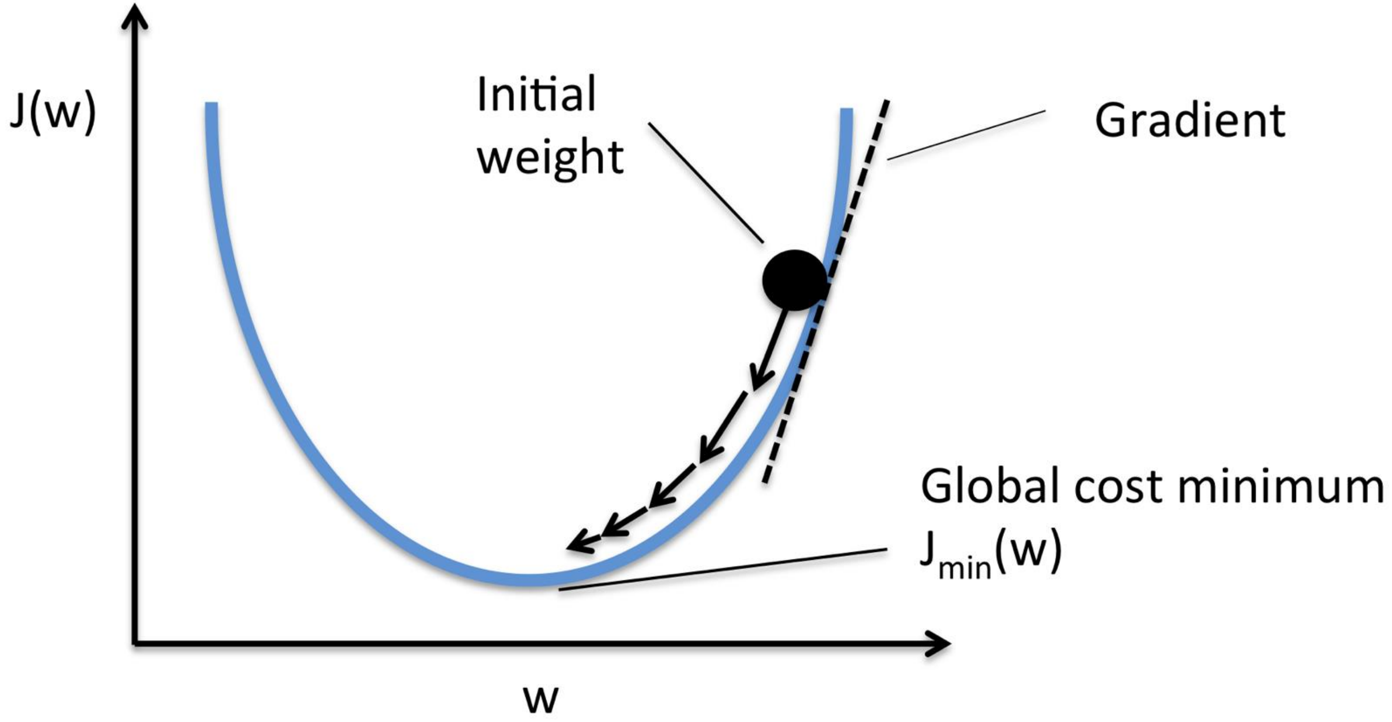
http://cs231n.github.io/neural-networks-3/#update

Vanilla Approach (Already Examined)

Parameters   Gradient

```
x += - learning_rate * dx
```

Inefficient since steps get smaller as gradient gets smaller



http://cs231n.github.io/neural-networks-3/#update
https://rasbt.github.io/mlxtend/user_guide/general_concepts/gradient-optimization/

Momentum Optimization

- Analogy: roll a ball down a hill and it will pick up momentum

Like friction; values range from 0 to 1 with larger being greater friction

Velocity vector captures cumulative direction of previous gradients; initialized to 0

Gradient not used for speed but instead acceleration

```
v = mu * v - learning_rate * dx # integrate velocity
x += v # integrate position
```

- What are advantages and disadvantages?
  - Can roll past local minima ☺
  - It may roll past optimum and oscillate around it ☹
  - Need to choose a mu value ☹

http://cs231n.github.io/neural-networks-3/#update

Other Optimization Methods

- Step decay:
  - Reduce the learning rate by some factor every few epochs
- Exponential decay
- 1/t decay
- Adapt learning rate per-parameter
  - e.g., AdaGrad, RMSprop, and Adam (i.e., adaptive momentum – very popular)

http://cs231n.github.io/neural-networks-3/#update

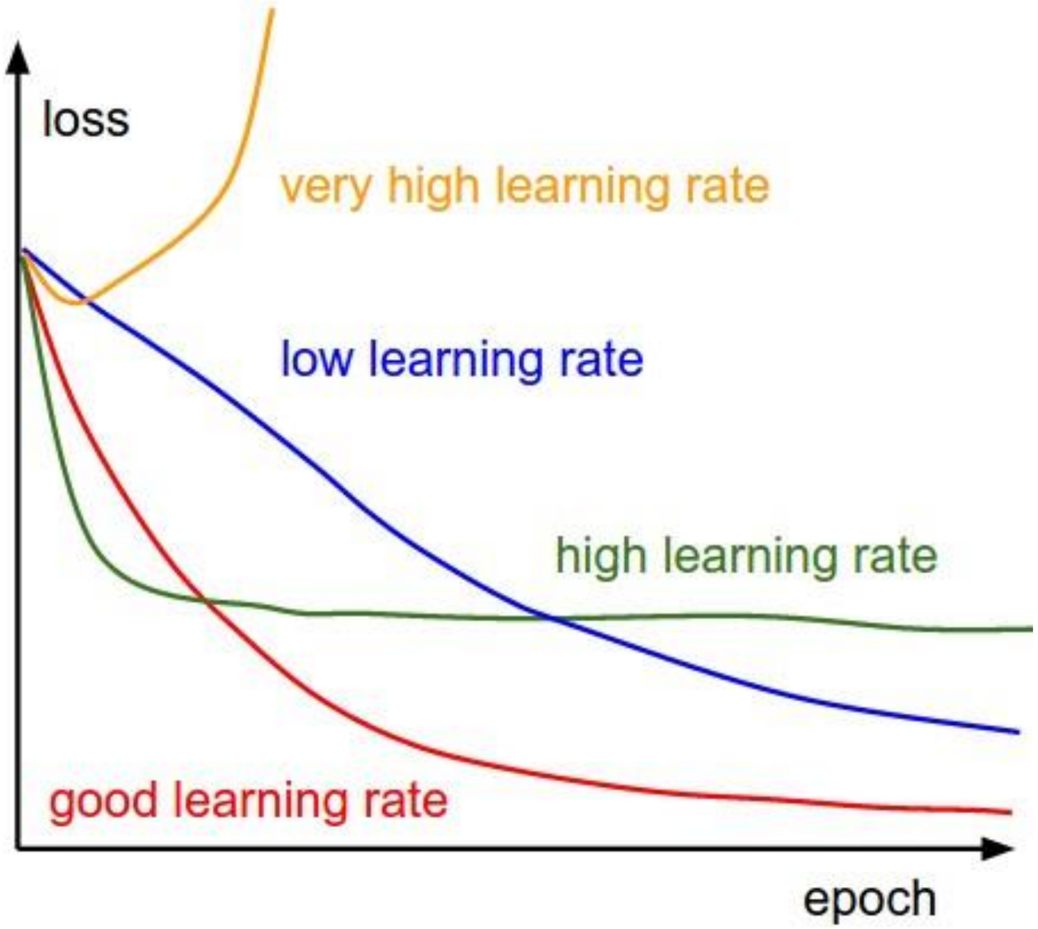
How Often to Update?

- Use mean gradients over ***all training examples*** (Batch gradient descent)
  - Less bouncing but can be slow or infeasible when dataset is large
- Use gradient from ***one training example*** (Stochastic gradient descent)
  - Fast to compute and can train using huge datasets (stores one instance in memory at each iteration) but updates are expected to bounce a lot
- Use mean gradients over ***subset of training examples*** (Mini-batch gradient descent)
  - Bounces less erratically than SGD and can train using huge datasets (store some instances in memory at each iteration) but can be slow or infeasible when dataset is large
- Often mini-batch gradient descent is used with maximum # of examples that fit in memory

Practical Note: Need Patience

Algorithm training can take hours, days, weeks, months, or more!

During Training, You Should Ask Yourself: What Does the Observed Loss Behavior Mean?



https://cs231n.github.io/neural-networks-3/#update

During Training, You Should Ask Yourself: What Does the Observed Loss Behavior Mean?

- Loss curves signal how well training is going
- Can address potential concerns by debugging the training process for each hypothesized issue one-by-one: e.g.,
  - learning rate too high
  - learning rate too low
  - too small of mini-batch size
  - too many dead neurons resulting from poor weight initialization

What is a Good Loss Value?

- 0… no error ☺
- In practice, a value better than the *expected* one for the loss function
  - e.g., What would be expected for the cross entropy loss?

Probability distribution of predicted class

Probability distribution of true class

Number of classes

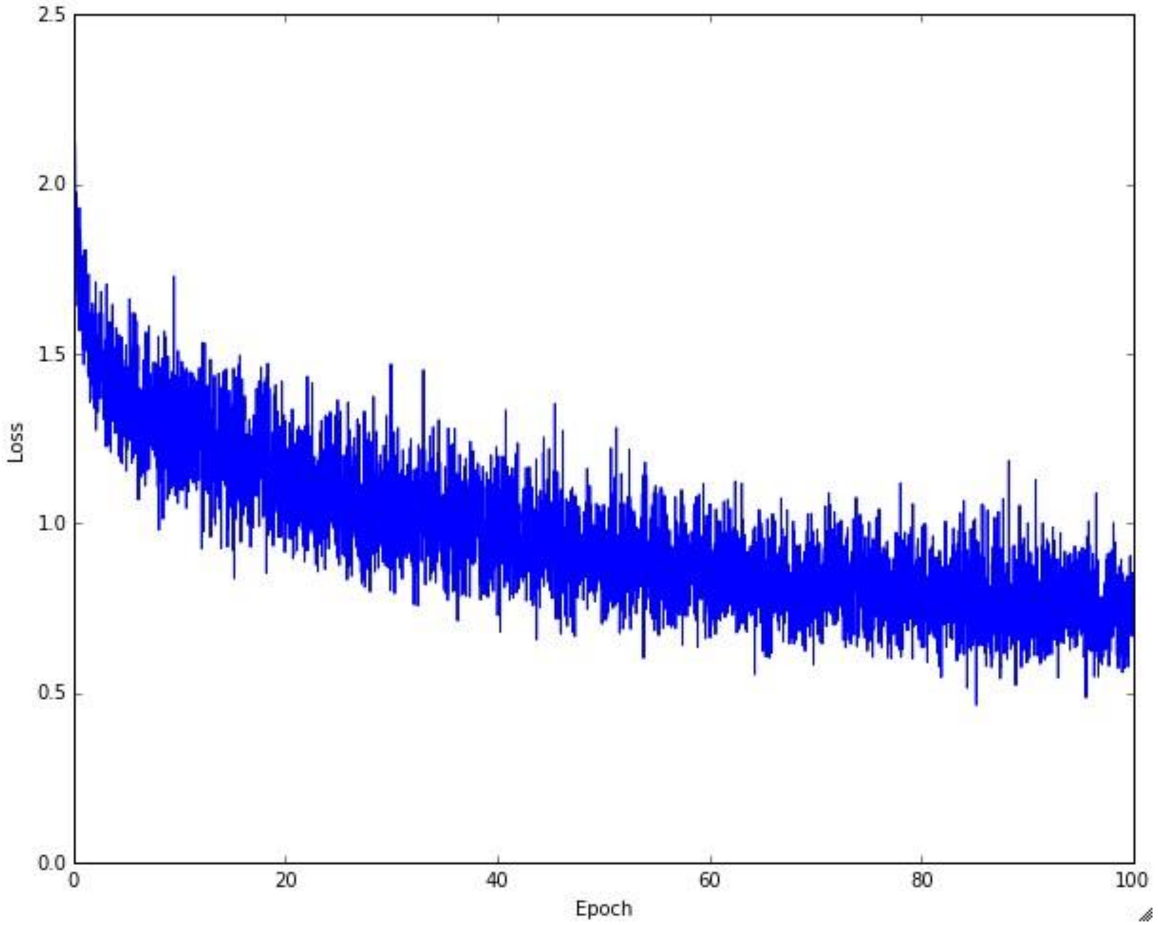
$$L_{\mathrm{CE}}(\hat{y}, y) = -\sum_{k=1}^{K} y_k \log \hat{y}_k = -\log \hat{y}_k, \quad \text{(where } k \text{ is the correct class)}$$

  - For a single example, loss from random guessing (equal probability per class) is:

Loss = - log (1/K) = log (K)

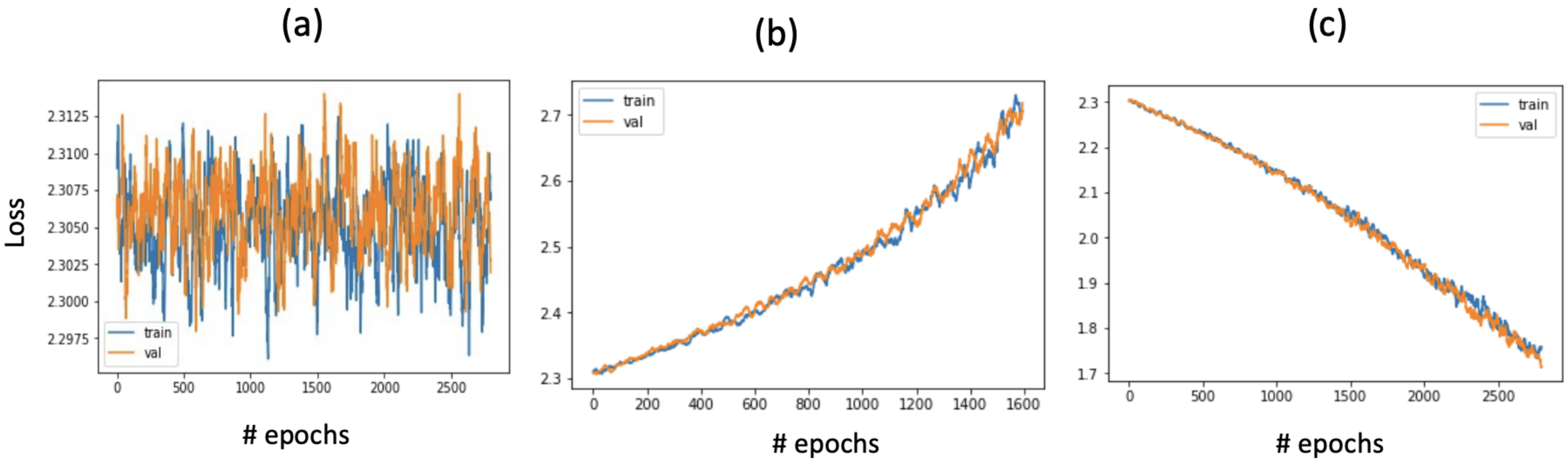
  - Assuming the dataset has a uniform true class distribution, we also get this value

Analysis: Why Might There Be Oscillations in the Learning Curve for the Training Loss?



https://cs231n.github.io/neural-networks-3/#update

Discussion: From These Learning Curves, What Do You Think Is Happening and What Might Be a Fix?



Feeling Bewildered By Your Learning Curves?

You may feel better when looking at this link:

https://lossfunctions.tumblr.com/

Today’s Topics

- Motivation: effective gradients for learning
- Initializing parameters
- Initializing data
- Following the gradient (optimization)
- Programming tutorial

Rise of “Deep Learning” Open Source Platforms

torch (Collobert et al.) — 2011

theano (Bastien et al.) — 2012

Caffe (Jia et al.) — 2013

2014

Keras

TensorFlow™ (Abadi et al.) — 2015

2016

PYTORCH (Paszke et al.) — 2017

idiap RESEARCH INSTITUTE

Université de Montréal

Berkeley UNIVERSITY OF CALIFORNIA

Google

Popular Options Today

Rise of "Deep Learning" Open Source Platforms

Course focus for programming tutorials

torch	theano	Caffe		Keras TensorFlow™		PYTORCH
(Collobert et al.)	(Bastien et al.)	(Jia et al.)		(Abadi et al.)		(Paszke et al.)
2011	2012	2013	2014	2015	2016	2017

idiap RESEARCH INSTITUTE

Université de Montréal

Berkeley UNIVERSITY OF CALIFORNIA

Google

Today's Programming Tutorial

Today’s Topics

- Motivation: effective gradients for learning
- Initializing parameters
- Initializing data
- Following the gradient (optimization)
- Programming tutorial


The End