We are starting at 14:00!

Grab a seat and get ready

Saturdays.AI
Kigali

Saturdays.AI
Kigali

#2 Deep Linear Neural Networks

AI Saturdays Kigali

Agenda

14:00 - 14:45: Gradient descent

14:45 - 15:30: Pytorch Autograd

15:00 - 16:00: Pytorch Neural Net module

16:00 - 16:30: Break

16:30 - 17:00: Learning Hyperparameters

17:30 - 18:00: Challenges & Next steps

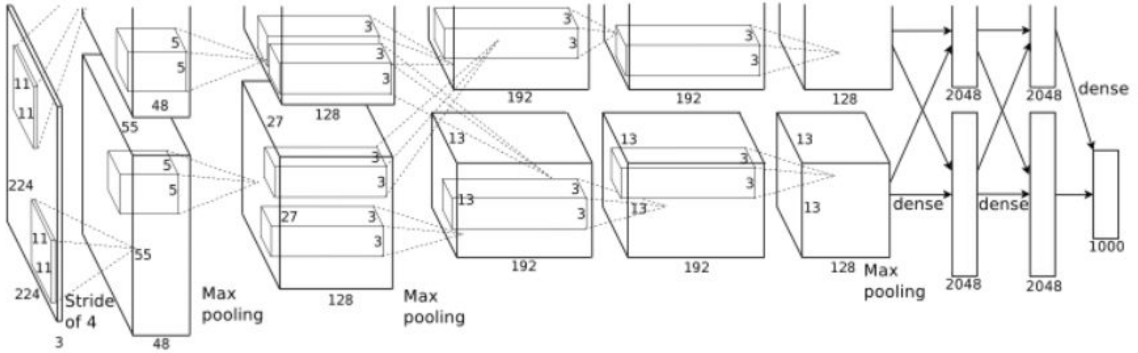
Saturdays.AI
Kigali

Gradient Descent

Saturdays.AI
Kigali

Gradient descent

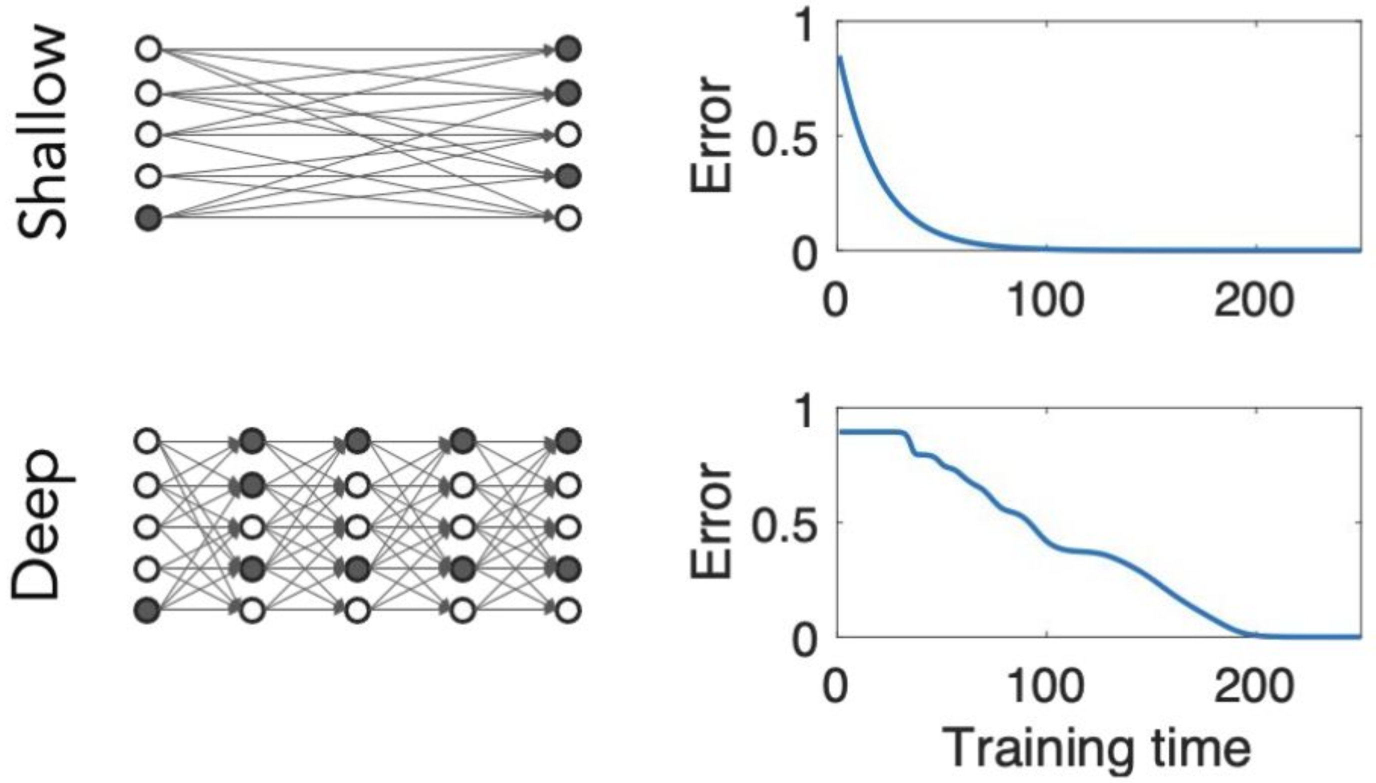
How can we change parameters to make the overall system work better?



Output:
Cat
Target:
Dog

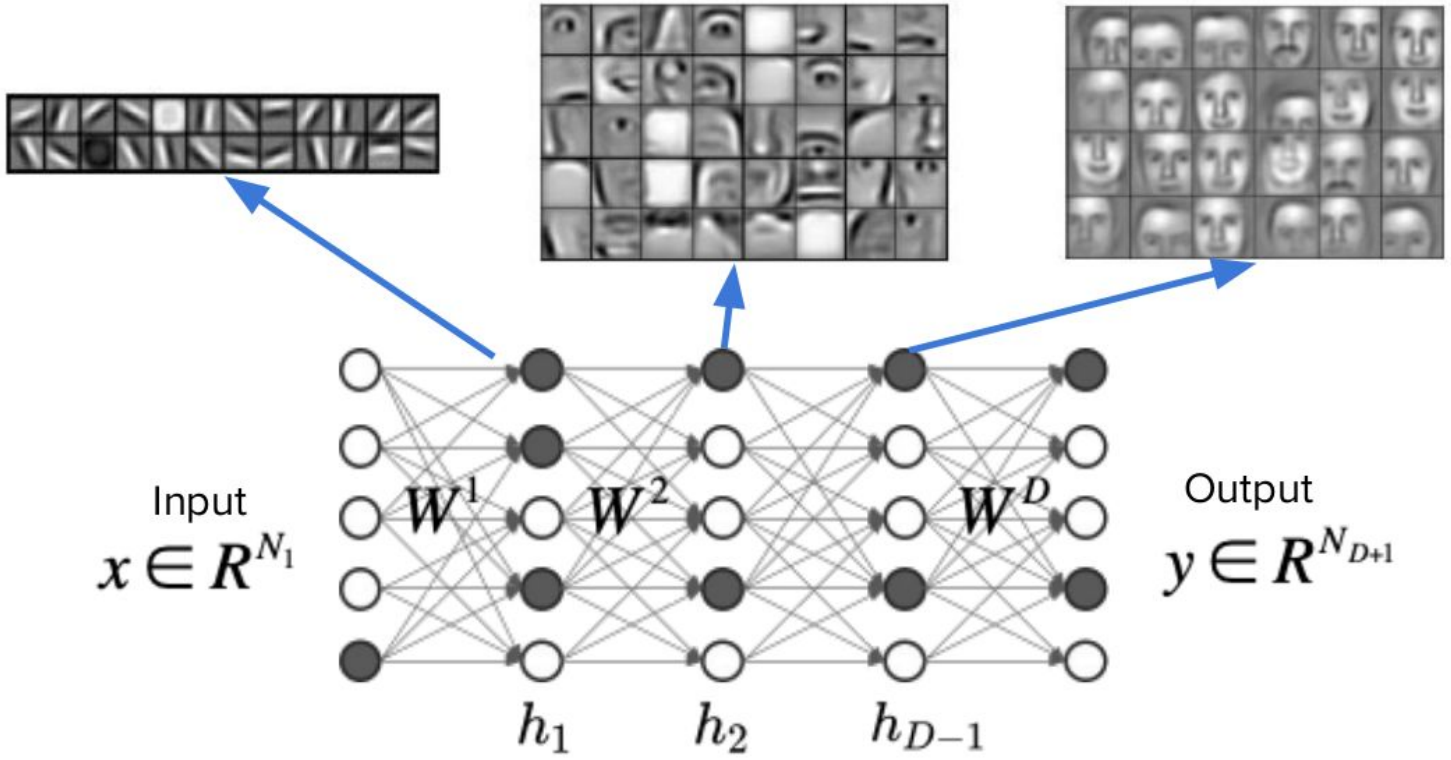
Saturdays.AI
Kigali

The effect of depth



Saturdays.AI
Kigali

Representation Learning

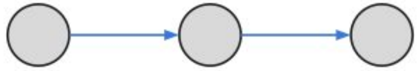


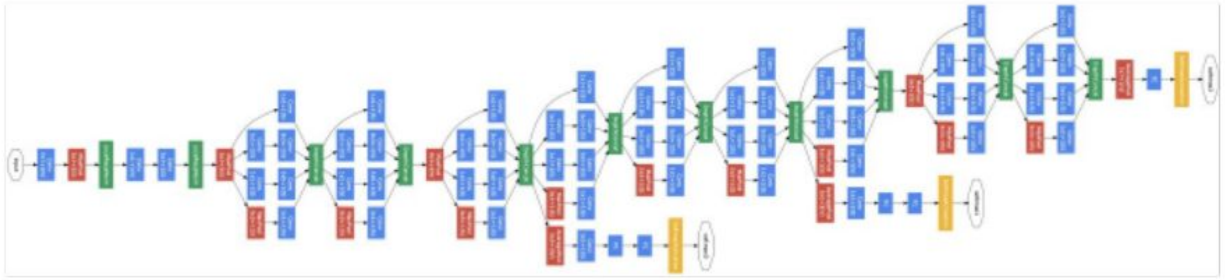
Saturdays.AI
Kigali

Simple models

Today

The rest of your career





Saturdays.AI
Kigali

Deep learning: key components

The designer specifies:

1. Objective function
2. Learning rule
3. Architecture
4. Initialisation
5. Environment

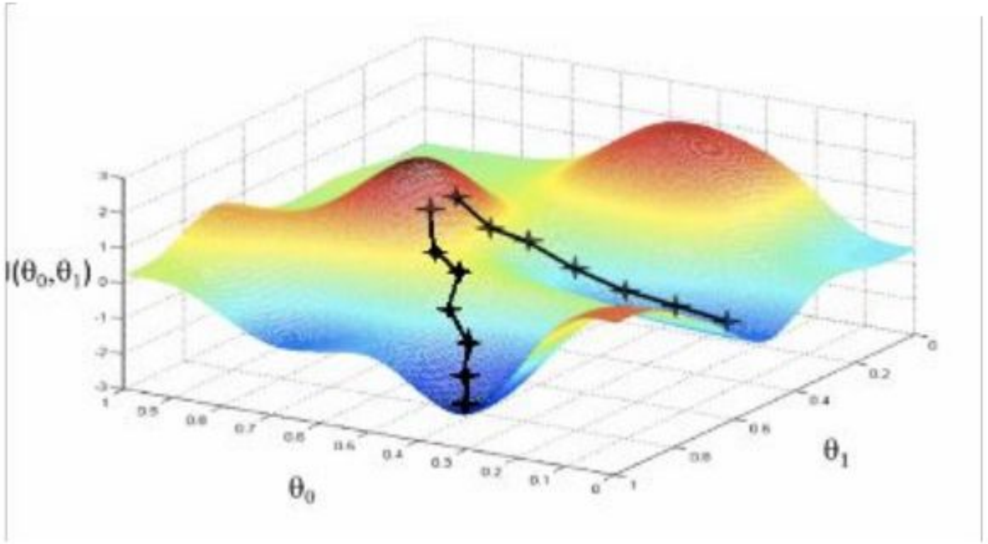
Saturdays.AI
Kigali

Deep learning: key components

What is the goal of the computation?

The designer specifies:

1. **Objective function**
2. Learning rule
3. Architecture
4. Initialisation
5. Environment



Saturdays.AI
Kigali

Deep learning: key components

The designer specifies:

1. Objective function
2. **Learning rule**
3. Architecture
4. Initialisation
5. Environment

How will the weight change to improve the objective function?

$$\Delta W = -\eta \frac{\partial L}{\partial W}$$

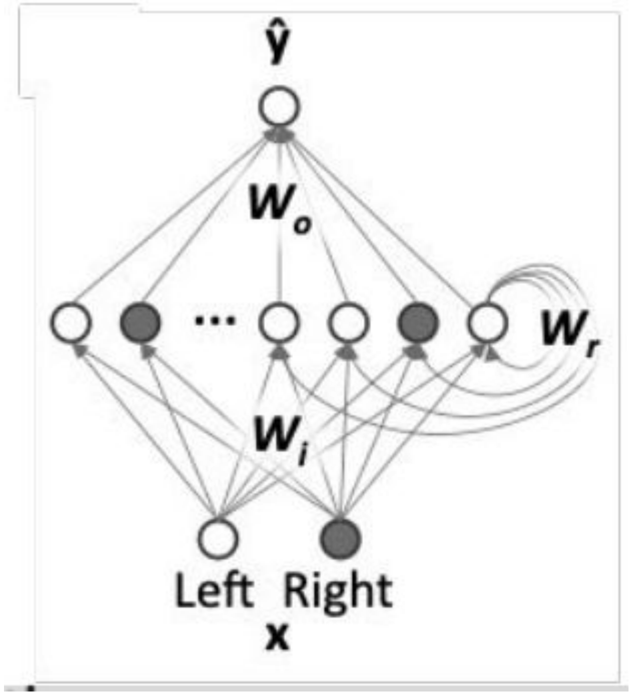
Saturdays.AI
Kigali

Deep learning: key components

What are the components and connectivity?

The designer specifies:

1. Objective function
2. Learning rule
3. **Architecture**
4. Initialisation
5. Environment



Deep learning: key components

The designer specifies:

1. Objective function
2. Learning rule
3. Architecture
4. **Initialisation**
5. Environment

What are the initial weight values?

$$W(0) \sim N(0, \sigma^2)$$

Saturdays.AI
Kigali

Deep learning: key components

The designer specifies:

1. Objective function
2. Learning rule
3. Architecture
4. Initialisation
5. **Environment**

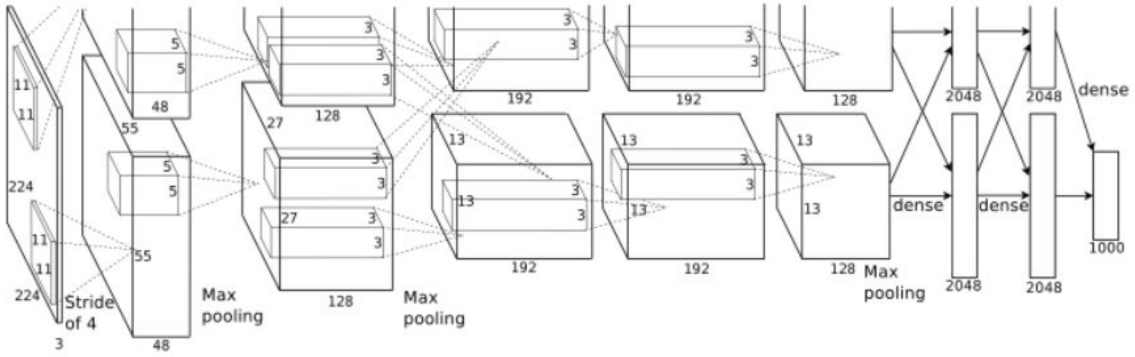
What is the data provided during learning?

IMAGENET

Saturdays.AI
Kigali

Example

- Objective function: Cross entropy loss
- Learning rule: Gradient descent with momentum
- Architecture: Deep convolutional ReLU network
- Initialisation: He et al. (Scaled Gaussian)
- Environment: ImageNet dataset



Output: Cat

Target: Dog

Learning as optimization

1. An input-output function (an ANN): $y = f_w(x)$

2. A loss function: $L = \ell(y, data)$

3. Optimization problem: $w^* = \text{argmin}_w \ell(f_w(x), data)$

Saturdays.AI
Kigali

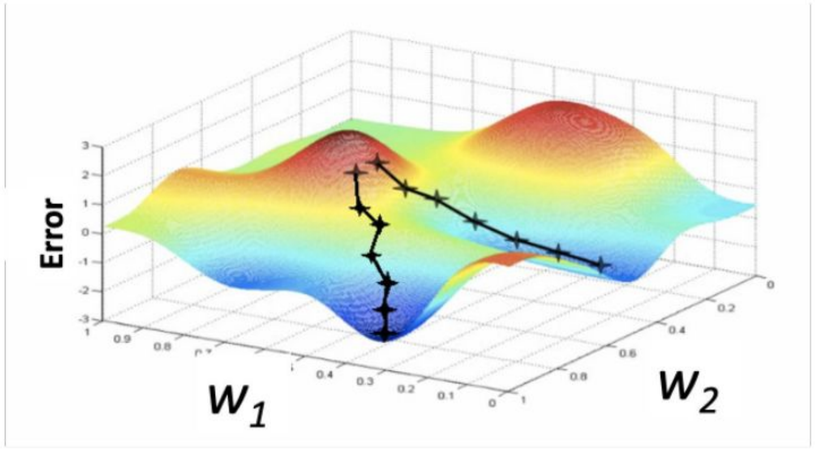
The workhorse algorithm: Gradient descent

So important we will understand it at different levels:

- Conceptually
- By taking derivatives by hand *Just once - I promise!*
- Through automatic differentiation in PyTorch

Saturdays.AI
Kigali

Gradient descent



http://blog.datumbox.com/wp-content/uploads/2013/10/gradient-descent.png

Minimize function by taking many small steps, each pointing downhill

Saturdays.AI
Kigali

Gradient

“how much would loss change if I changed a parameter just a tiny bit”

$$\nabla L(w) = \left[\frac{\partial L}{\partial w_1} \; \frac{\partial L}{\partial w_2} \cdots \frac{\partial L}{\partial w_N} \right] \Bigg|_{w}$$

Saturdays.AI
Kigali

Practice

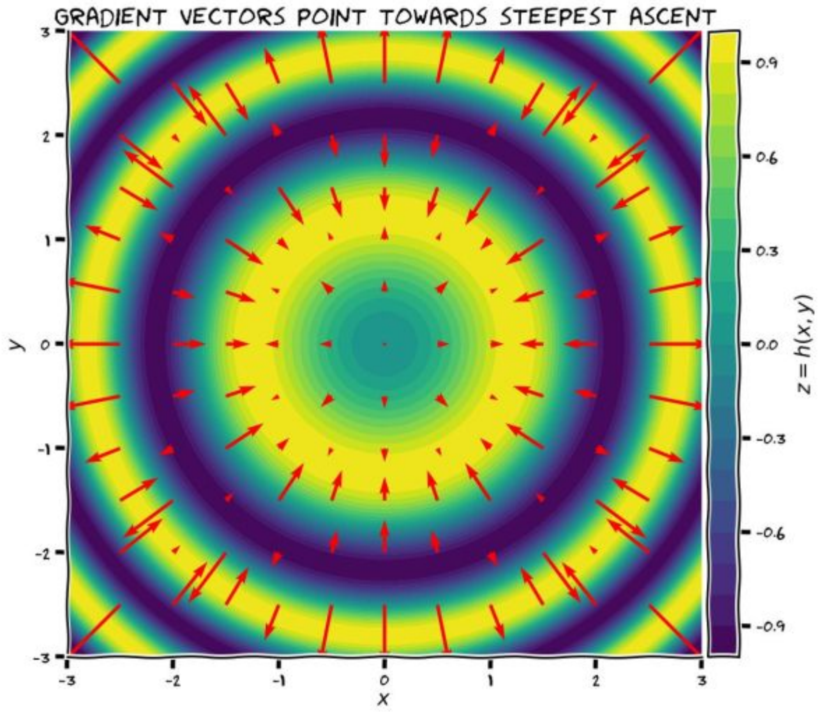
Derive the gradient by hand

Let's start by investigating what directions the gradient points in

Analytical Exercise 1.1

Saturdays.AI
Kigali

Gradient



The gradient points in the direction of steepest ascent

Saturdays.AI Kigali

Gradient descent

“Make small change in weights that most rapidly improves task performance”

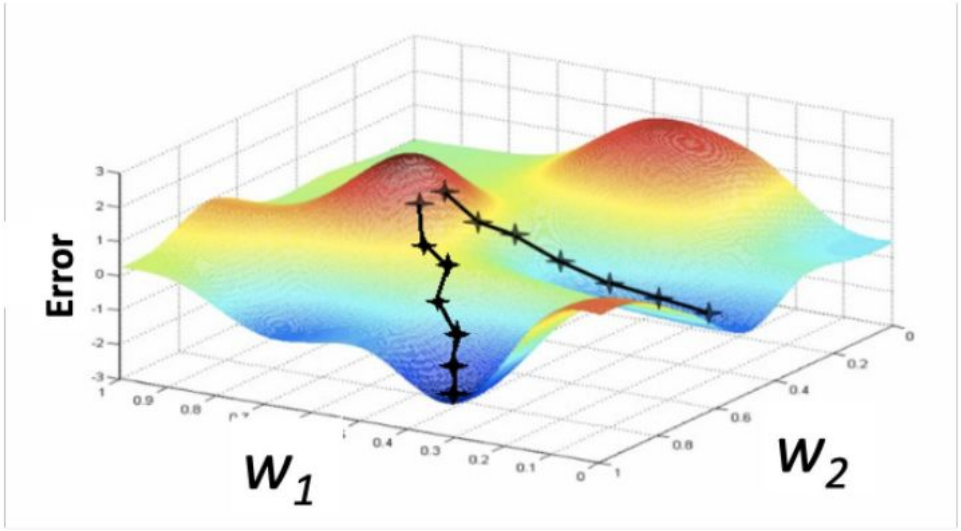
$$\Updownarrow$$

Change each weight in proportion to the negative gradient of the loss

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla L(\mathbf{w}^{(t)})$$

Saturdays.AI
Kigali

Gradient descent



http://blog.datumbox.com/wp-content/uploads/2013/10/gradient-descent.png

Initialize: $\mathbf{w}^{(0)} = \mathbf{w}_0$

For t=0 to T:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla L(\mathbf{w}^{(t)})$$

Saturdays.AI
Kigali

Why gradient descent

There are an infinite set of learning approaches that make us better

However, GD is the one that most rapidly reduces loss (for infinitesimal steps)

Saturdays.AI
Kigali

Practice

Derive the gradient by hand (again!)

- Let's try a slightly more complicated example
- Because it is so fundamental, you should do it at least once
- Basic tools: partial derivatives; chain rule

Analytical Exercise 1.2

Saturdays.AI
Kigali

Pytorch Auto Grad

Saturdays.AI
Kigali

Gradients via the computational graph

Deriving gradients ad hoc is hard and it’s easy to make mistakes

How can we simplify and systematize our approach?

Saturdays.AI
Kigali

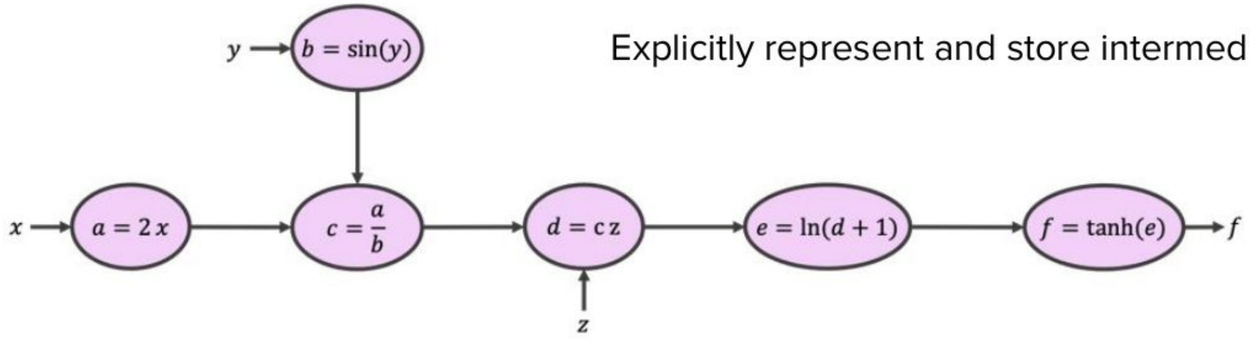
Computational Graph (forward)

$$f(x, y, z) = \tanh\left(\ln\left[1 + z\frac{2x}{sin(y)}\right]\right)$$

Saturdays.AI
Kigali

Computational Graph (forward)

$$f(x,y,z) = \tanh\left(\ln\left[1 + z\frac{2x}{sin(y)}\right]\right)$$



Explicitly represent and store intermediate variables *a*,*b*,*c*,*d*,*e*

Saturdays.AI Kigali

Computational Graph (forward)

$$f(x, y, z) = \tanh\left(\ln\left[1 + z\frac{2x}{sin(y)}\right]\right)$$

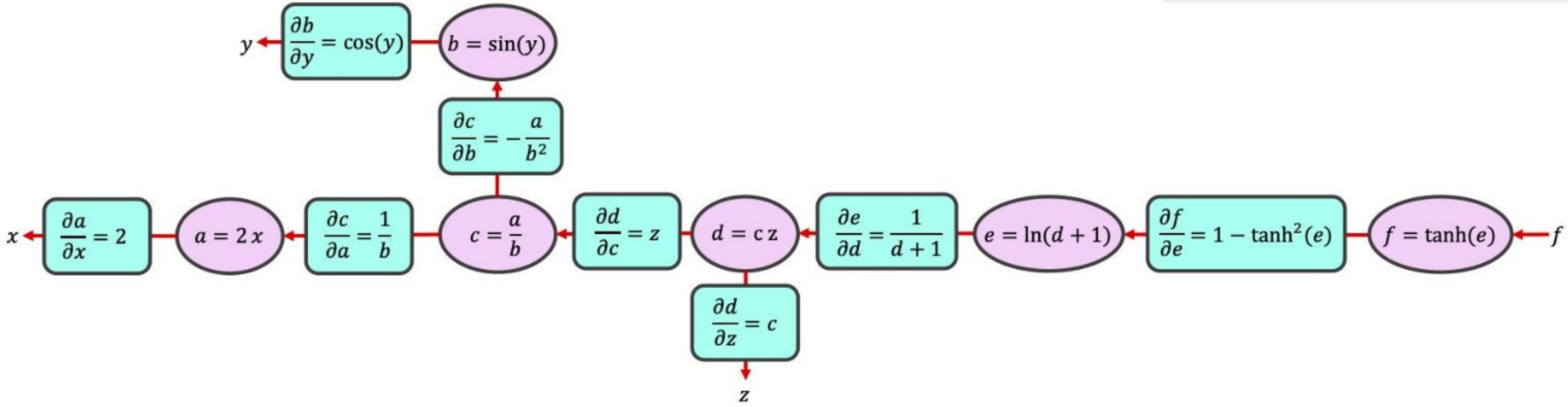
$$\begin{tikzcd}
 & y \arrow[r] & b = \sin(y) \arrow[d] & & & & \\
x \arrow[r] & a = 2\,x \arrow[r] & c = \frac{a}{b} \arrow[r] & d = c\,z \arrow[r] & e = \ln(d+1) \arrow[r] & f = \tanh(e) \arrow[r] & f \\
 & & & z \arrow[u] & & &
\end{tikzcd}$$

Explicitly represent and store intermediate variables *a*,*b*,*c*,*d*,*e*.
Nodes in the graph correspond to intermediate variables.

Saturdays.AI
Kigali

Computational Graph (backward)

Starting from the top, pass backward. Each edge stores partial derivative of the head of the edge with respect to the tail.

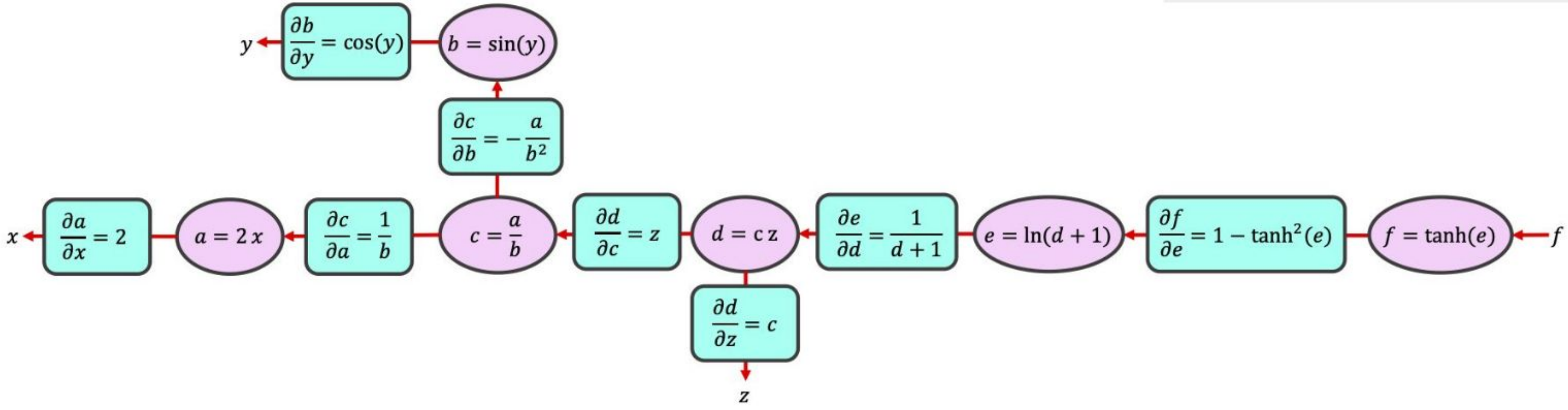


y
$\frac{\partial b}{\partial y} = \cos(y)$
$b = \sin(y)$
$\frac{\partial c}{\partial b} = -\frac{a}{b^2}$
x
$\frac{\partial a}{\partial x} = 2$
$a = 2x$
$\frac{\partial c}{\partial a} = \frac{1}{b}$
$c = \frac{a}{b}$
$\frac{\partial d}{\partial c} = z$
$d = cz$
$\frac{\partial d}{\partial z} = c$
z
$\frac{\partial e}{\partial d} = \frac{1}{d+1}$
$e = \ln(d+1)$
$\frac{\partial f}{\partial e} = 1 - \tanh^2(e)$
$f = \tanh(e)$
f

Saturdays.AI
Kigali

Computational Graph (backward)

Starting from the top, pass backward. Each edge stores partial derivative of the head of the edge with respect to the tail.



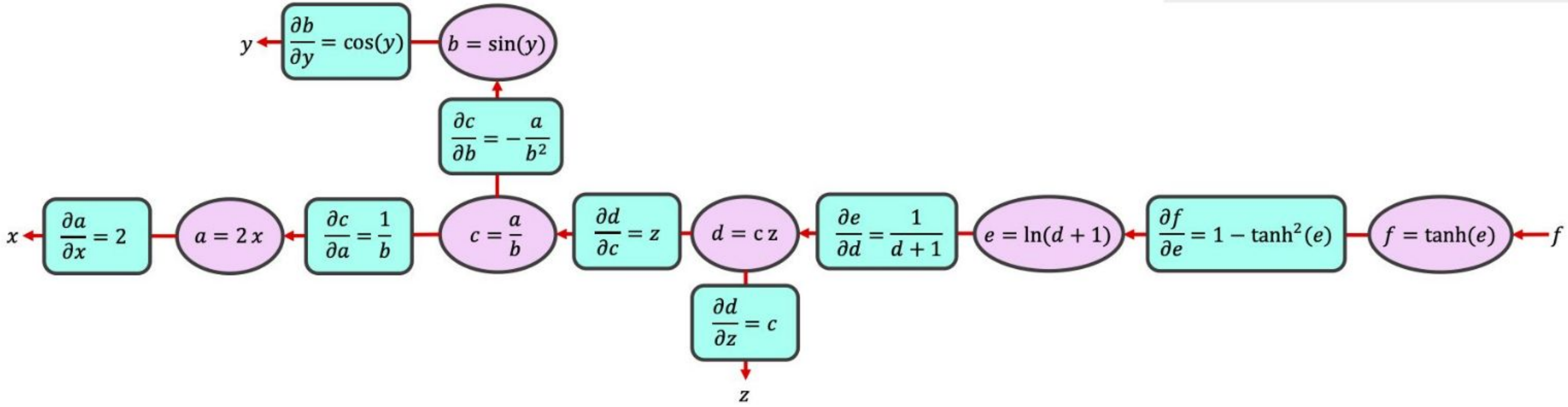
y
$\frac{\partial b}{\partial y} = \cos(y)$
$b = \sin(y)$
$\frac{\partial c}{\partial b} = -\frac{a}{b^2}$
x
$\frac{\partial a}{\partial x} = 2$
$a = 2x$
$\frac{\partial c}{\partial a} = \frac{1}{b}$
$c = \frac{a}{b}$
$\frac{\partial d}{\partial c} = z$
$d = cz$
$\frac{\partial e}{\partial d} = \frac{1}{d+1}$
$e = \ln(d+1)$
$\frac{\partial f}{\partial e} = 1 - \tanh^2(e)$
$f = \tanh(e)$
f
$\frac{\partial d}{\partial z} = c$
z

Conveniently, the partial derivatives can often be expressed using the intermediate variables calculated in the forward pass (a,b,c,d,e).

Saturdays.AI
Kigali

Computational Graph (gradients)

Starting from the top, pass backward. Each edge stores partial derivative of the head of the edge with respect to the tail.



y
$\frac{\partial b}{\partial y} = \cos(y)$
$b = \sin(y)$
$\frac{\partial c}{\partial b} = -\frac{a}{b^2}$
x
$\frac{\partial a}{\partial x} = 2$
$a = 2x$
$\frac{\partial c}{\partial a} = \frac{1}{b}$
$c = \frac{a}{b}$
$\frac{\partial d}{\partial c} = z$
$d = cz$
$\frac{\partial e}{\partial d} = \frac{1}{d+1}$
$e = \ln(d+1)$
$\frac{\partial f}{\partial e} = 1 - \tanh^2(e)$
$f = \tanh(e)$
f
$\frac{\partial d}{\partial z} = c$
z

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial a}\frac{\partial a}{\partial x} = (1 - \tanh^2(e)) \cdot \frac{1}{d+1} \cdot z \cdot \frac{1}{b} \cdot 2$$

Saturdays.AI Kigali

The magic of automatic differentiation

$$\begin{tikzcd}
\boxed{\texttt{f(x) \{...\};}} \arrow[r, "\text{Automatic differentiation}"] & \boxed{\texttt{df(x) \{...\};}} \\
\boxed{y = f(x)} \arrow[u, "\text{human programmer}"'] \arrow[r, dashed, "\text{symbolic differentiation (human/computer)}"'] & \boxed{y' = f'(x)} \arrow[u, dashed]
\end{tikzcd}$$

Saturdays.AI
Kigali

Building the computational graph

A data structure for storing intermediate values and partial derivatives needed to compute gradients.

- Node v represents variable
  - Stores value
  - Gradient
  - The function that created the node
- Directed edge from v to u represents the partial derivative of u w.r.t. v
- To compute the gradient ∂L/∂v, find the unique path from L to v and multiply the edge weights, where L is the overall loss

Saturdays.AI
Kigali

Building the computational graph

When we perform operations on PyTorch Tensors, PyTorch does not simply calculate the output

Instead, each operation is added to the computational graph

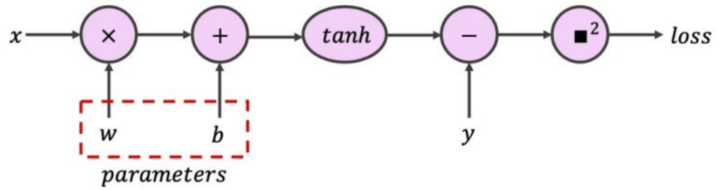
PyTorch can then do a forward and backward pass through the graph, storing necessary intermediate variables, and yield any gradients we need

Building the computational graph

Often only some parameters are trainable and require gradients.

We indicate tensors that require gradients by setting `requires_grad=True`

$$(y - \tanh(wx + b))^2$$



Saturdays.AI
Kigali

Building the computational graph

PyTorch can keep adding to the graph as your code winds through functions and classes

In essence, you write code to compute the loss L; AutoGrad does the rest

$$(y - \tanh(wx + b))^2$$

Saturdays.AI
Kigali

Practice

Building the computational graph

- Compute the gradient with respect to w and b the easy way!

[Coding Exercise 2.1]

Saturdays.AI
Kigali

PyTorch's Neural Net module

Saturdays.AI
Kigali

Putting it together: a simple ANN

PyTorch can differentiate through fairly arbitrary functions

But neural networks often make use of simple building blocks

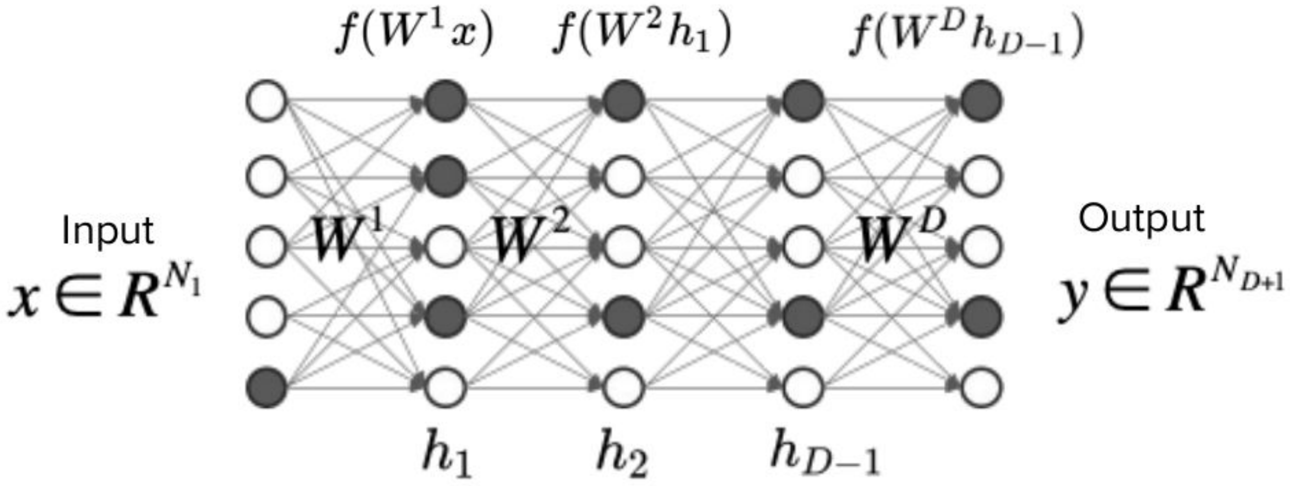
Let’s look at some ways that PyTorch makes building and training ANNs particularly simple

Aligning concepts and code

An input-output function (an ANN):	$y = f_w(x)$	nn.Module, nn.Sequential nn.Linear, nn.ReLU
A loss function:	$L = \ell(y, data)$	nn.MSELoss, nn.CrossEntropyLoss
Optimization problem:	$w^* = \text{argmin}_w \ell(f_w(x), data)$	torch.optim.SGD, torch.optim.ADAM

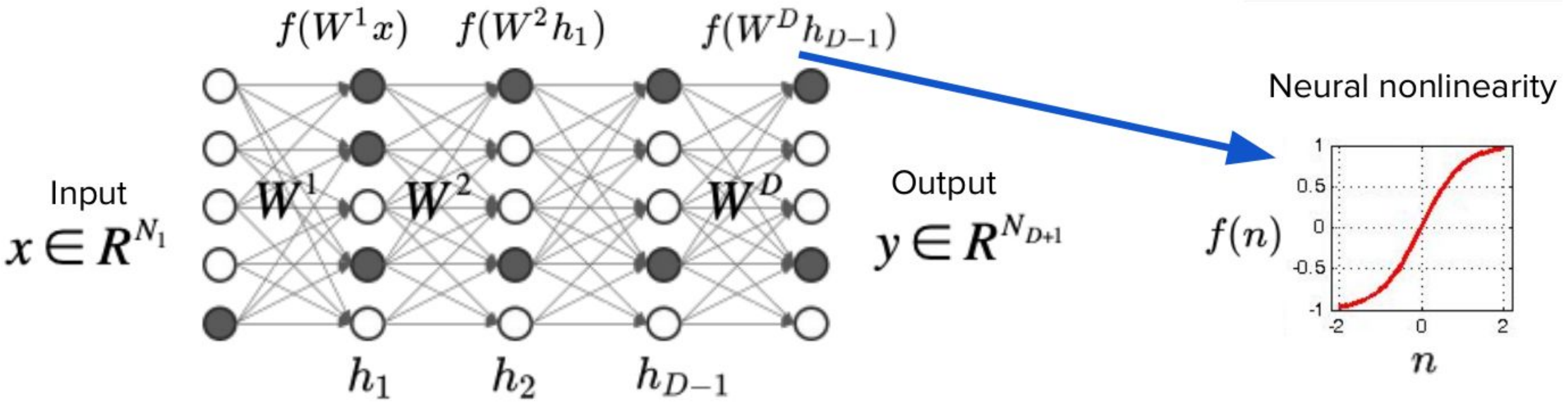
Saturdays.AI
Kigali

Deep Network



Saturdays.AI
Kigali

Deep Network



Saturdays.AI
Kigali

The canonical train loop in PyTorch

```
for i in range (n_epochs):
     optimizer.zero_grad() # Reset all gradients to zero
     prediction = neural_net(inputs) # Forward pass
     loss = loss_function(prediction, targets) # Compute the loss
     loss.backward() # Backward pass to build the graph and compute the gradients
     optimizer.step() # Update weights using gradient
```

Saturdays.AI
Kigali

Practice

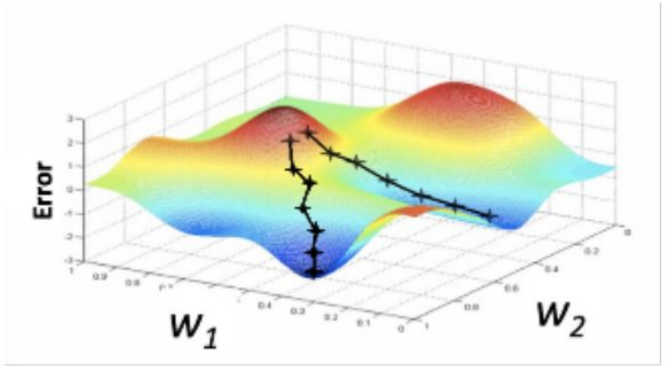
Build a neural network!

- Build a network using the nn.Module and nn components
- Compute predictions for some input data
- Calculate the loss
- Calculate derivatives with AutoGrad
- Run a step of the optimizer

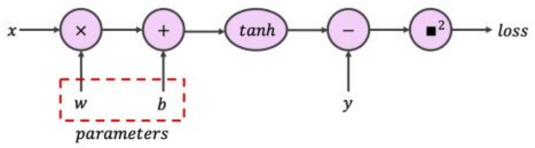
[Coding Exercise 3.1]

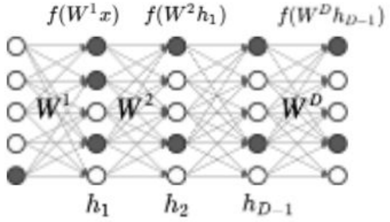
Saturdays.AI
Kigali

Wrap up: gradient descent



http://blog.datumbox.com/wp-content/uploads/2013/10/gradient-descent.png





Saturdays.AI
Kigali

Break

Saturdays.AI
Kigali

Learning Hyperparameters

Saturdays.AI
Kigali

Learning Hyperparameters

Now that we can implement a network, let’s understand some core learning behaviors and tradeoffs

The architecture, initialization, and **learning hyperparameters** all can change the performance of a network dramatically

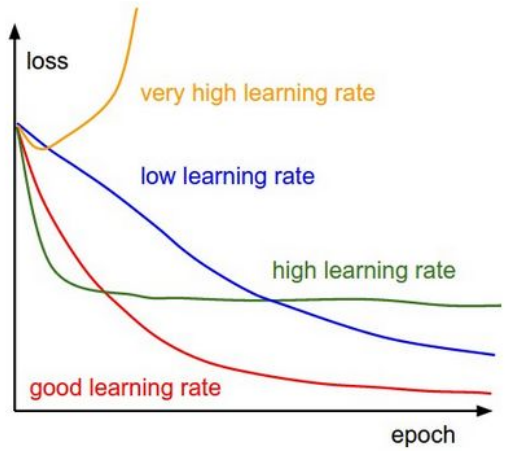
To be proficient at training deep networks, we have to build our intuition

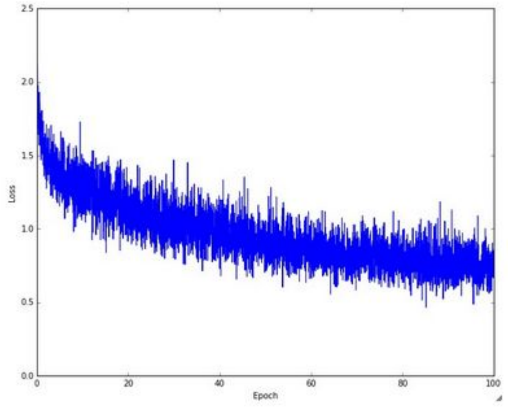
Saturdays.AI
Kigali

Learning rate

The learning rate is the most important hyper-parameter. There is a huge amount of material on how to choose a learning rate, how to modify the learning rate during the training, and how the wrong learning rate can completely ruin the model training.

Maybe you might have seen this famous graph (from Stanford's CS231n class) that shows how a learning rate that is too big or too small affects the loss during training.





Saturdays.AI
Kigali

Choosing a learning rate

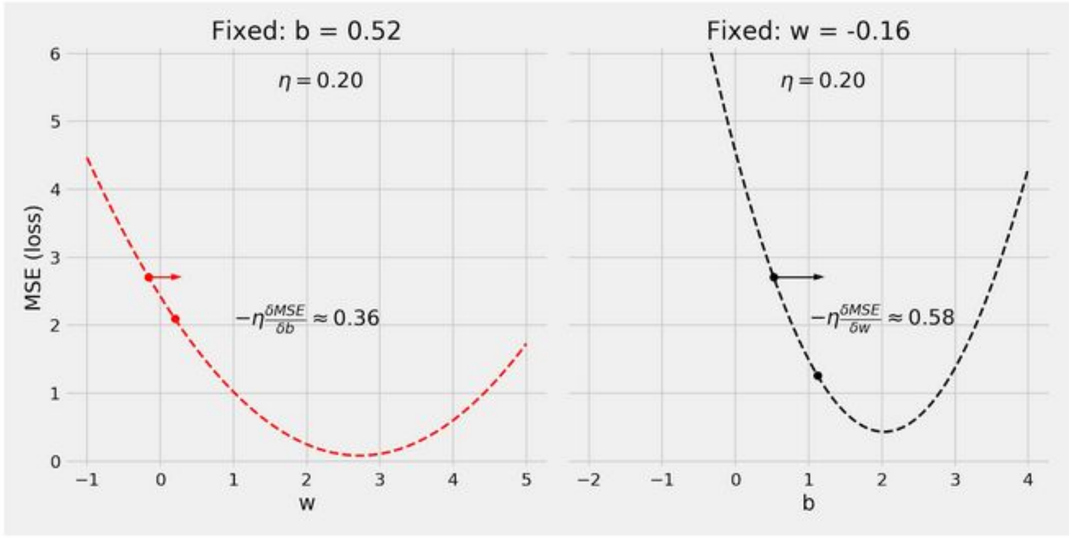
Let us further explore the only choice you have; **the size of your step or the learning rate**.

> "Choose your learning rate wisely."
>
> \- Grail Knight

Saturdays.AI
Kigali

Small learning rate

- It makes sense to start with baby steps, right? This means using a **small learning rate**.
- Small learning rates will likely get you to (some) minimum point eventually.
  - Unfortunately, time is money, especially when you're paying for GPU time in the cloud.
  - So, there is an incentive to try **bigger learning rates.**



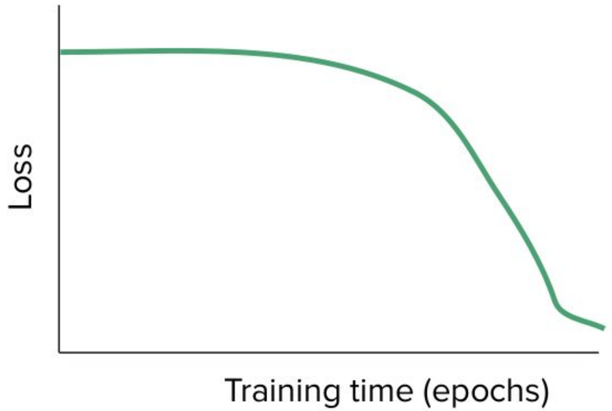
IMPORTANT: In real life, a learning rate of 0.2 is usually considered big

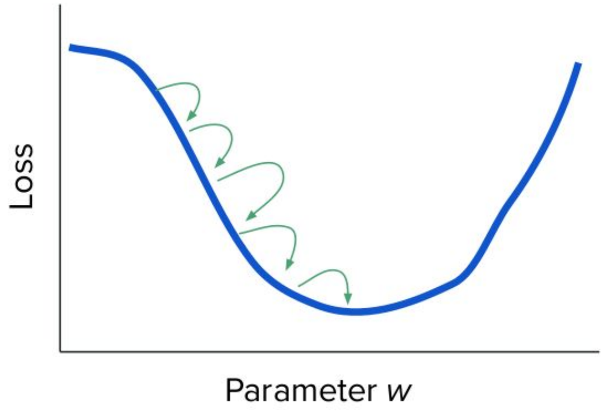
Saturdays.AI
Kigali

Small learning rate

How to pick η? $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla L(\mathbf{w}^{(t)})$

Slightly too small: works but slow

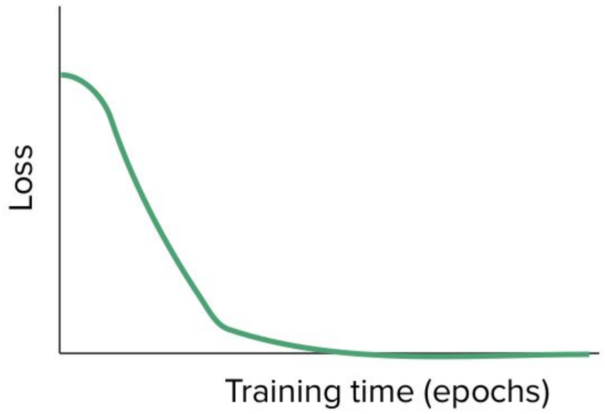


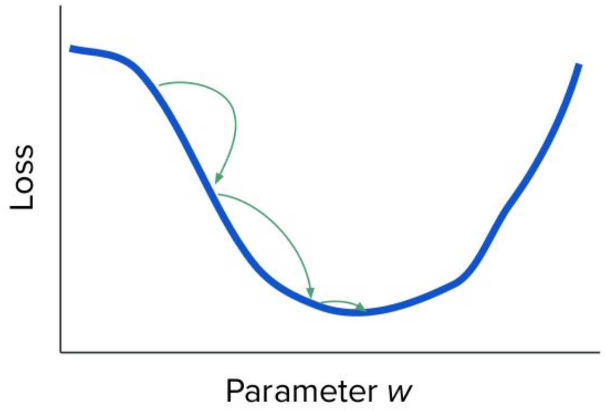


Small learning rate

How to pick η? $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla L(\mathbf{w}^{(t)})$

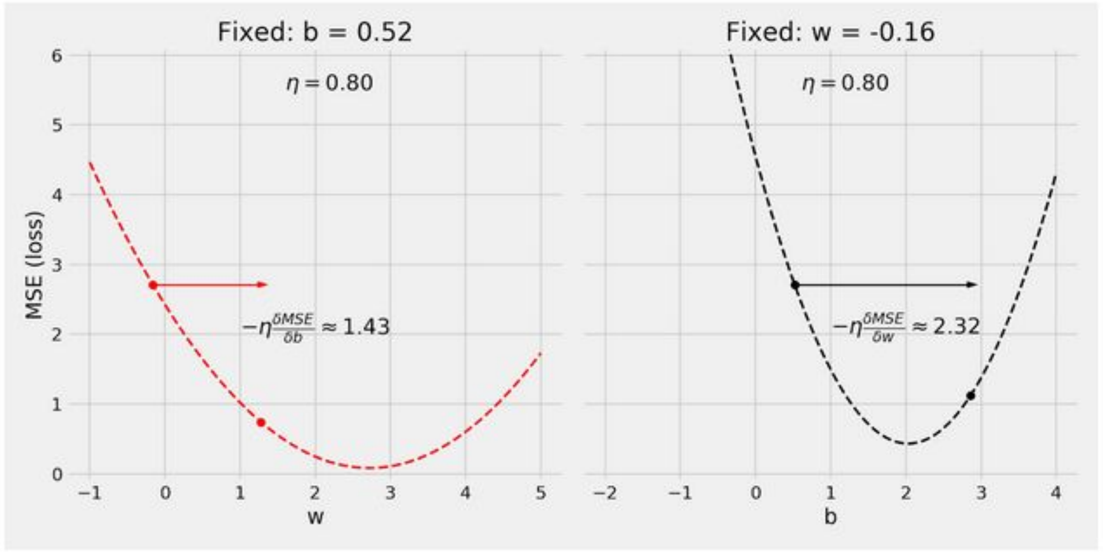
Just right: converges quickly and cleanly





Big learning rate

What would have happened if we had used a big learning rate instead, say, a step size of 0.8? As we can see in the plots below, we start to literally run into trouble.



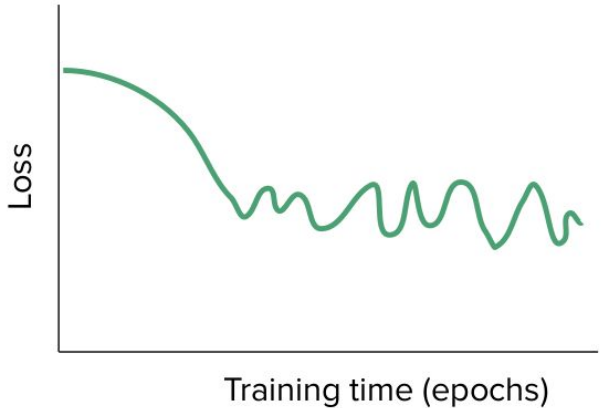
Even though everything is still okay on the left plot, the right plot shows us a completely different picture; we ended up on the other side of the curve. **That is not good.**

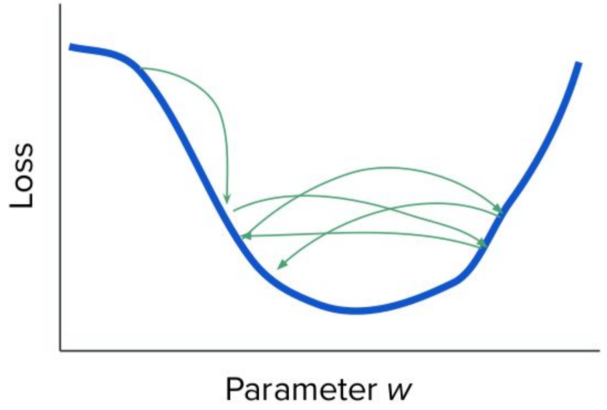
Saturdays.AI Kigali

Big learning rate

How to pick η? $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla L(\mathbf{w}^{(t)})$

Slightly too big: chaotic

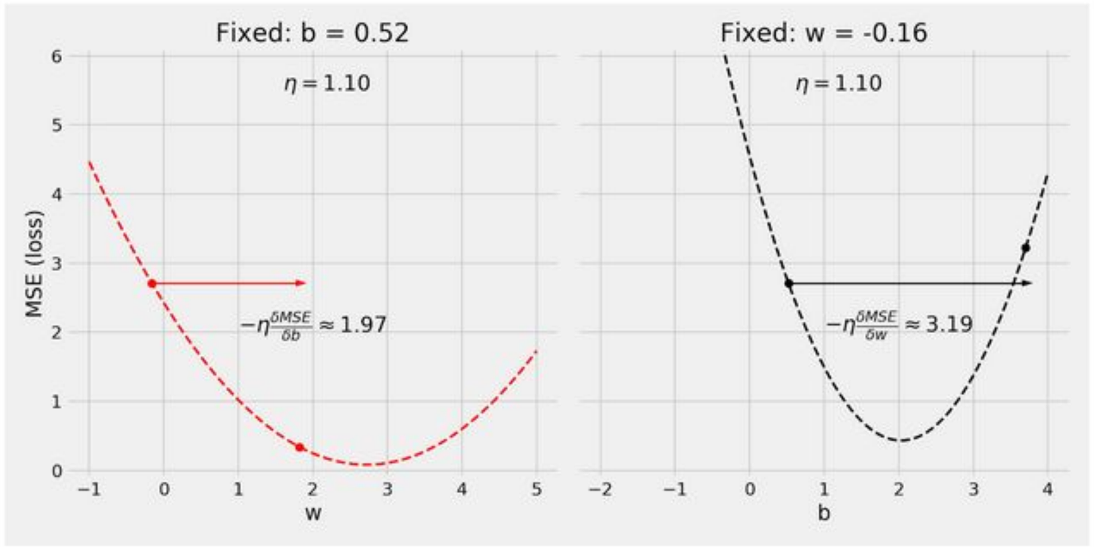




Saturdays.AI
Kigali

Very big learning rate

Wait, it may get worse than that. Let us use a really big learning rate, say, a step size of 1.1!



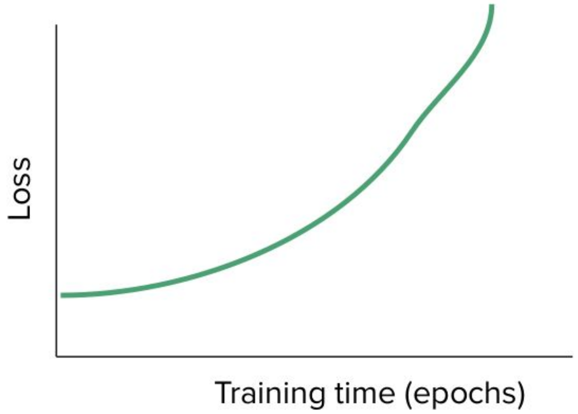
This is bad. On the right plot, not only did we end up on the other side of the curve again, but we actually climbed up. This means our loss increased instead of decreasing!

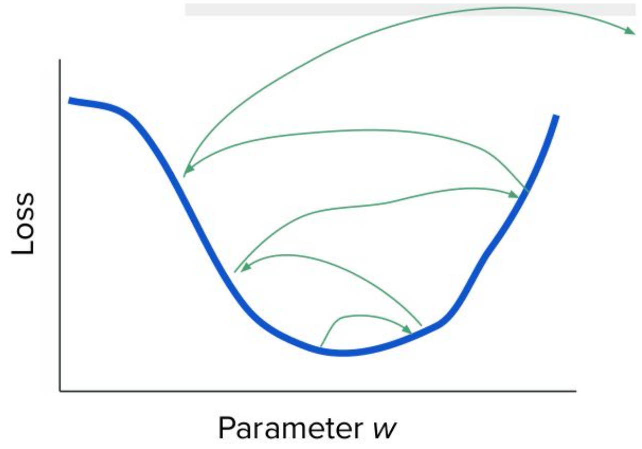
Saturdays.AI
Kigali

Very big learning rate

How to pick η? $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla L(\mathbf{w}^{(t)})$

Way
too big:
Divergence





Saturdays.AI
Kigali

Challenges & Next steps!

Saturdays.AI
Kigali

Kahoot!

Saturdays.AI
Kigali

Any questions?

Saturdays.AI
Kigali

Saturdays.AI
Kigali

THANKS

kigali@saturdays.ai
coming soon
coming soon