

Training Neural Networks

Florin Gogianu,
Bitdefender ML Team

February 23, 2022

Deep Learning Course Structure



Outline

Recap++

Understanding Neural Networks

Optimizing Neural Networks

Computational Graphs

Forward pass

Backward pass

Wild beasts and how to tame them?

Goals

1. Optimize large, deep neural networks...

Goals

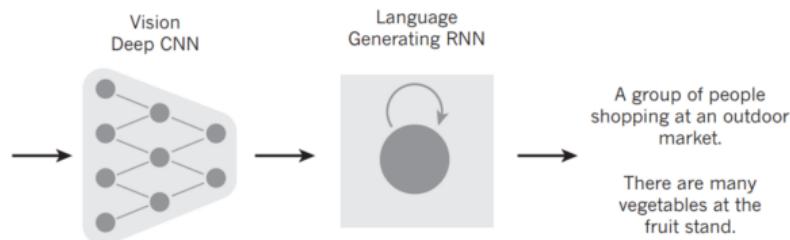
1. Optimize large, deep neural networks...
2. ...for learning *useful representations*.

Terminology

- ▶ **Supervised learning:** labeled data points of the correct behaviour.
- ▶ **Reinforcement learning:** receive some reward signal and try to maximize it by improving the model's behaviour
- ▶ **Unsupervised learning:** no labels - the aim is discovering interesing patterns in the data and usefull representations

Supervised Learning¹

Caption generation²



A woman is throwing a **frisbee** in a park.



A **dog** is standing on a hardwood floor.



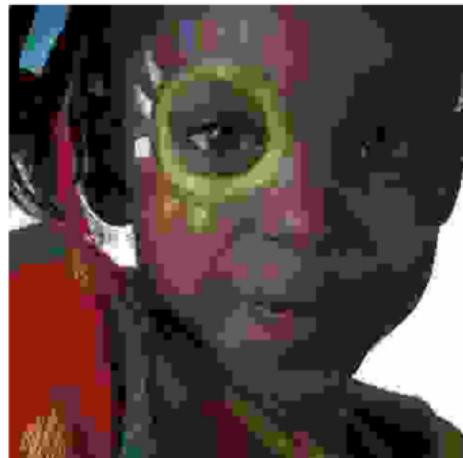
A **stop** sign is on a road with a mountain in the background

¹from Grosse, Ba, 2019

²Xu et al., 2015

Unsupervised Learning¹

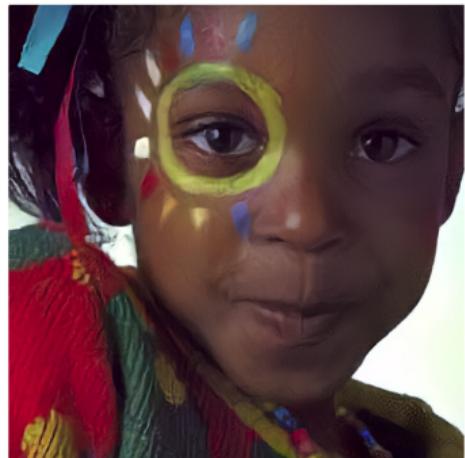
Image compression²



JPEG



JPEG 2000



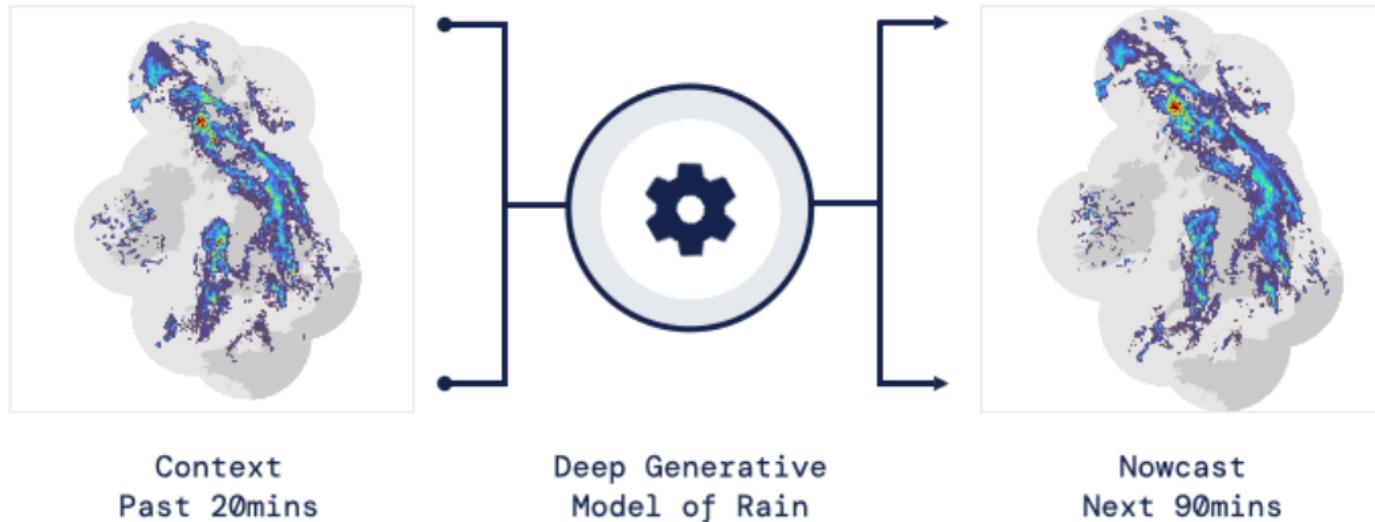
WaveOne

¹from CS294-158, 2019

²Rippel, Bourdev, 2017

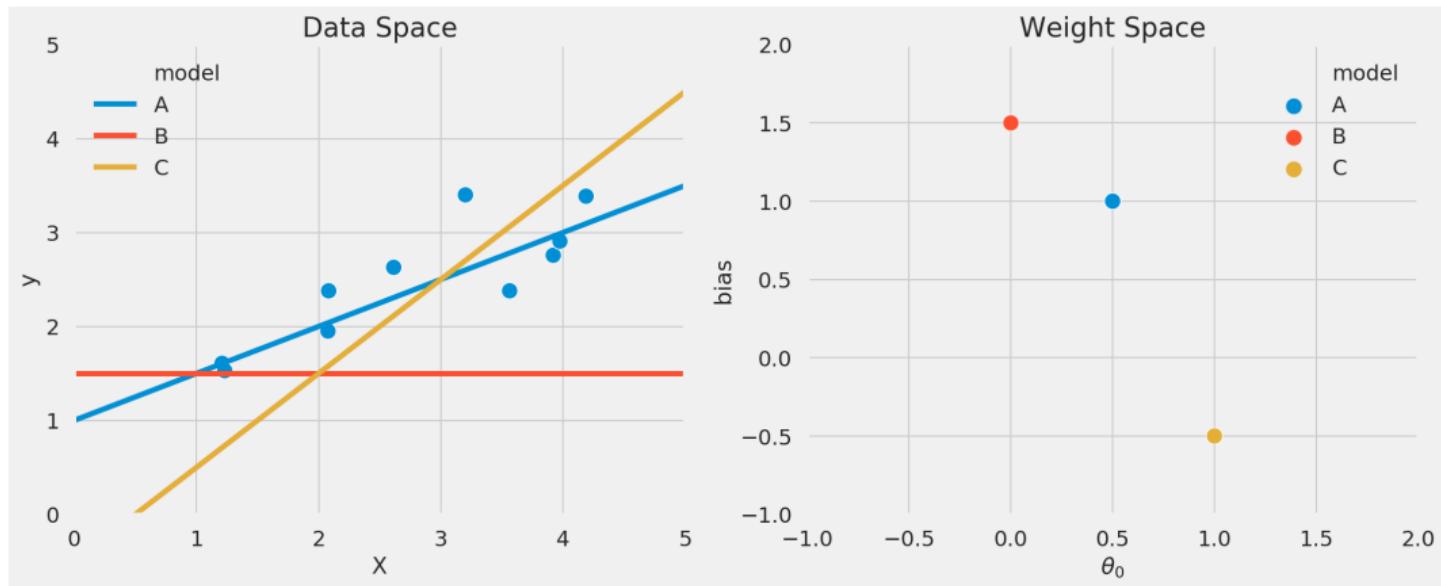
Unsupervised Learning

Nowcasting the next hour of rain ¹



¹DeepMind, 2022

Linear regression



Three different linear models in the data and weight space ¹.

$$y = \theta^T x + b$$

¹following Grosse, Ba - CS421, 2019

- ▶ Linear model $y = \theta^\top \phi(\mathbf{x})$, with Mean Squared Error objective function
 $\mathcal{L}(\theta) = (t - y)^2$

- ▶ Linear model $y = \theta^\top \phi(x)$, with Mean Squared Error objective function $\mathcal{L}(\theta) = (t - y)^2$
- ▶ Has a nice closed form solution: $(\Phi^\top \Phi)^{-1} \Phi^\top t$

- ▶ Linear model $y = \theta^\top \phi(\mathbf{x})$, with Mean Squared Error objective function $\mathcal{L}(\theta) = (t - y)^2$
- ▶ Has a nice closed form solution: $(\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$
- ▶ But the solution can also be found iteratively:
 - ▶ Compute the **gradient** of $\mathcal{L}(\theta)$ w.r.t. θ :

$$\nabla_{\theta} \mathcal{L} = (t - y) \phi(\mathbf{x})$$

- ▶ Linear model $y = \theta^\top \phi(\mathbf{x})$, with Mean Squared Error objective function $\mathcal{L}(\theta) = (t - y)^2$
- ▶ Has a nice closed form solution: $(\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$
- ▶ But the solution can also be found iteratively:
 - ▶ Compute the **gradient** of $\mathcal{L}(\theta)$ w.r.t. θ :

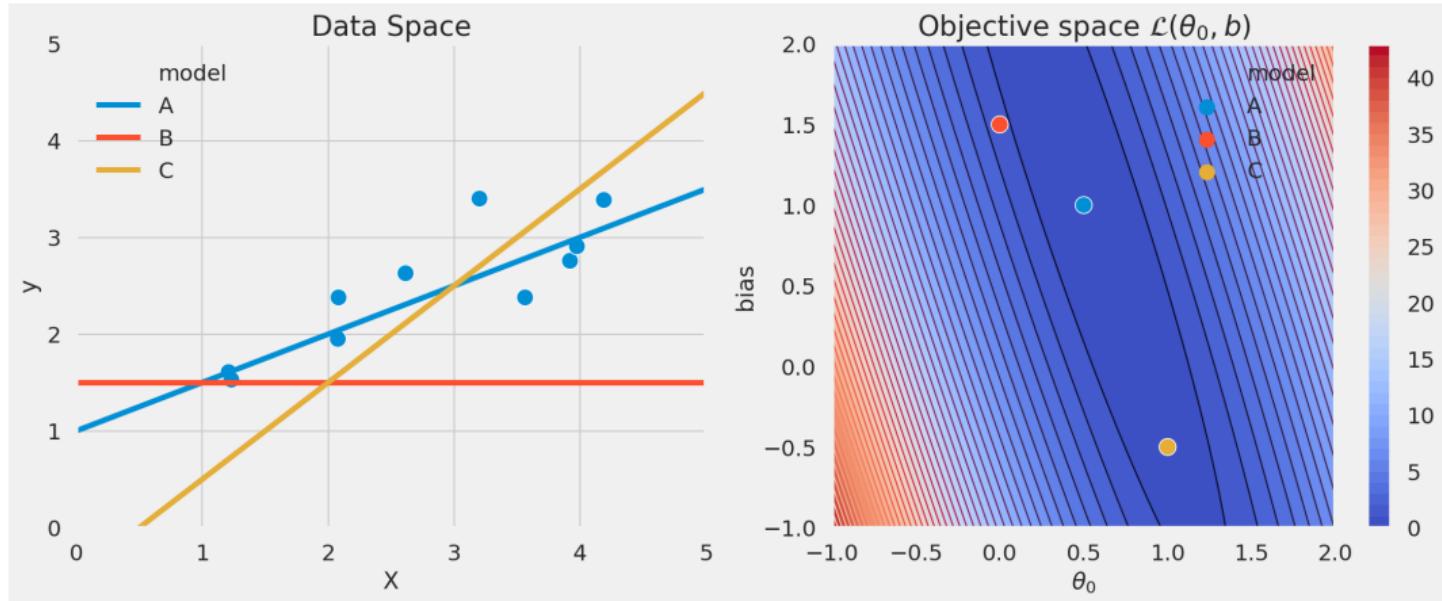
$$\nabla_{\theta} \mathcal{L} = (t - y) \phi(\mathbf{x})$$

- ▶ Perform **gradient descent**:

$$\theta_{j+1} \leftarrow \theta_j - \alpha \nabla_{\theta_j} \mathcal{L}$$

- ▶ Why do gradient descent if we can find the minimum analytically?

Linear models



Three different linear models in the data and objective $\mathcal{L}(\theta_0, b) = [t - f(x; \theta_0, b)]^2$ space.

Also discussed

- ▶ Model complexity
- ▶ Regularization
- ▶ Bias / Variance trade-off
- ▶ Over-fitting / Under-fitting

Outline

Recap++

Understanding Neural Networks

Optimizing Neural Networks

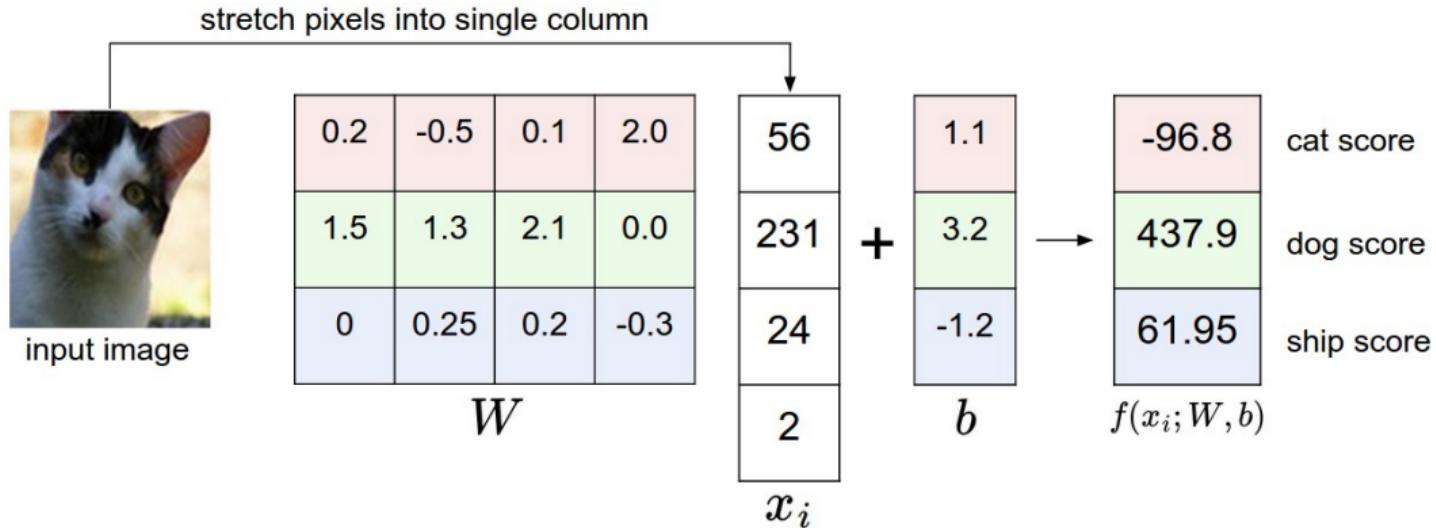
Computational Graphs

Forward pass

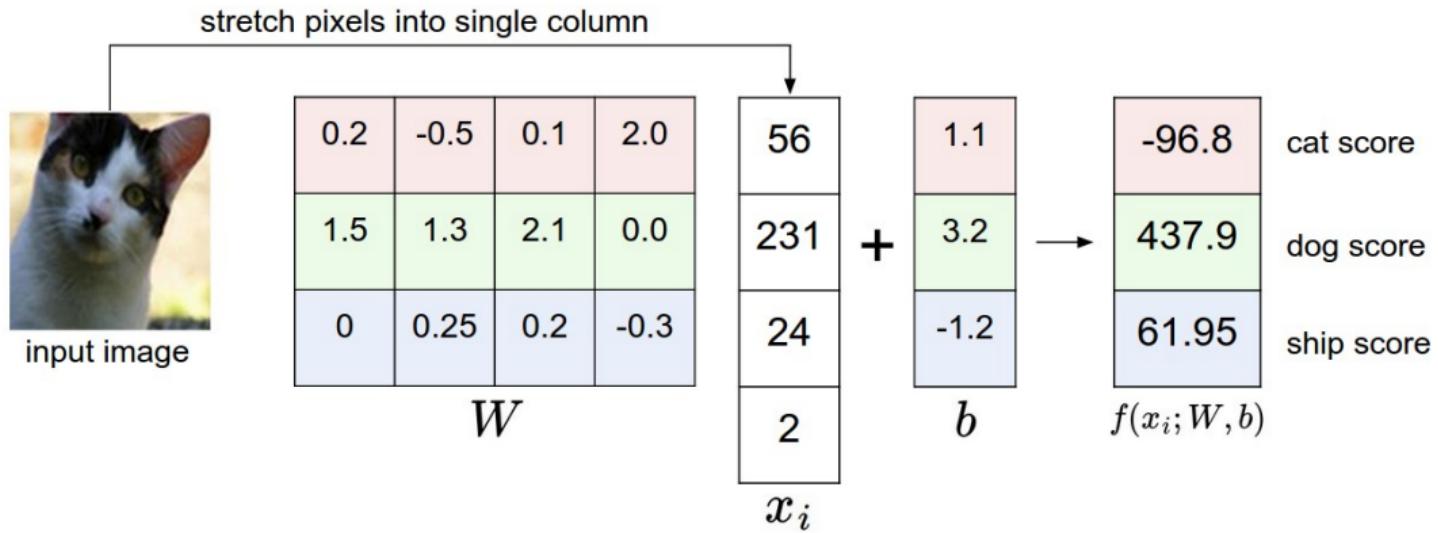
Backward pass

Wild beasts and how to tame them?

Linear classification



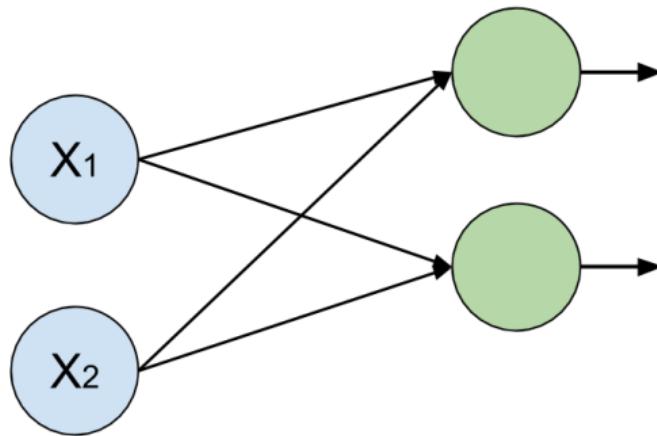
Linear classification



Learned weights:



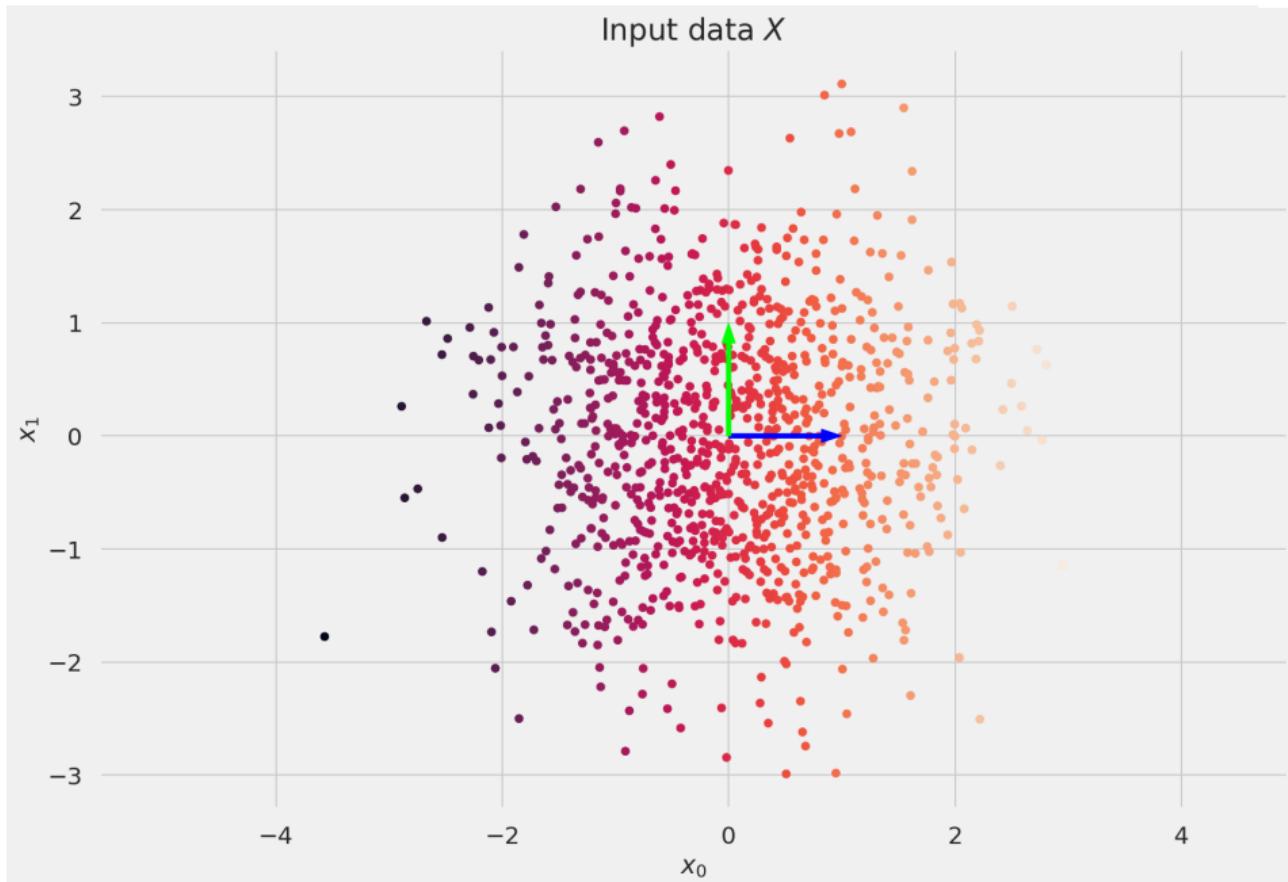
A simple transformation



A linear model with four weights. Also can be seen as two stacked "neurons" without activation functions.

Linear case: $\mathbf{y} = \mathbf{Wx}$

A simple transformation



A simple transformation

- ▶ $y = Wx$
- ▶ What is W actually doing to x ?

¹Mathematics for Machine Learning, ch. 4

A simple transformation

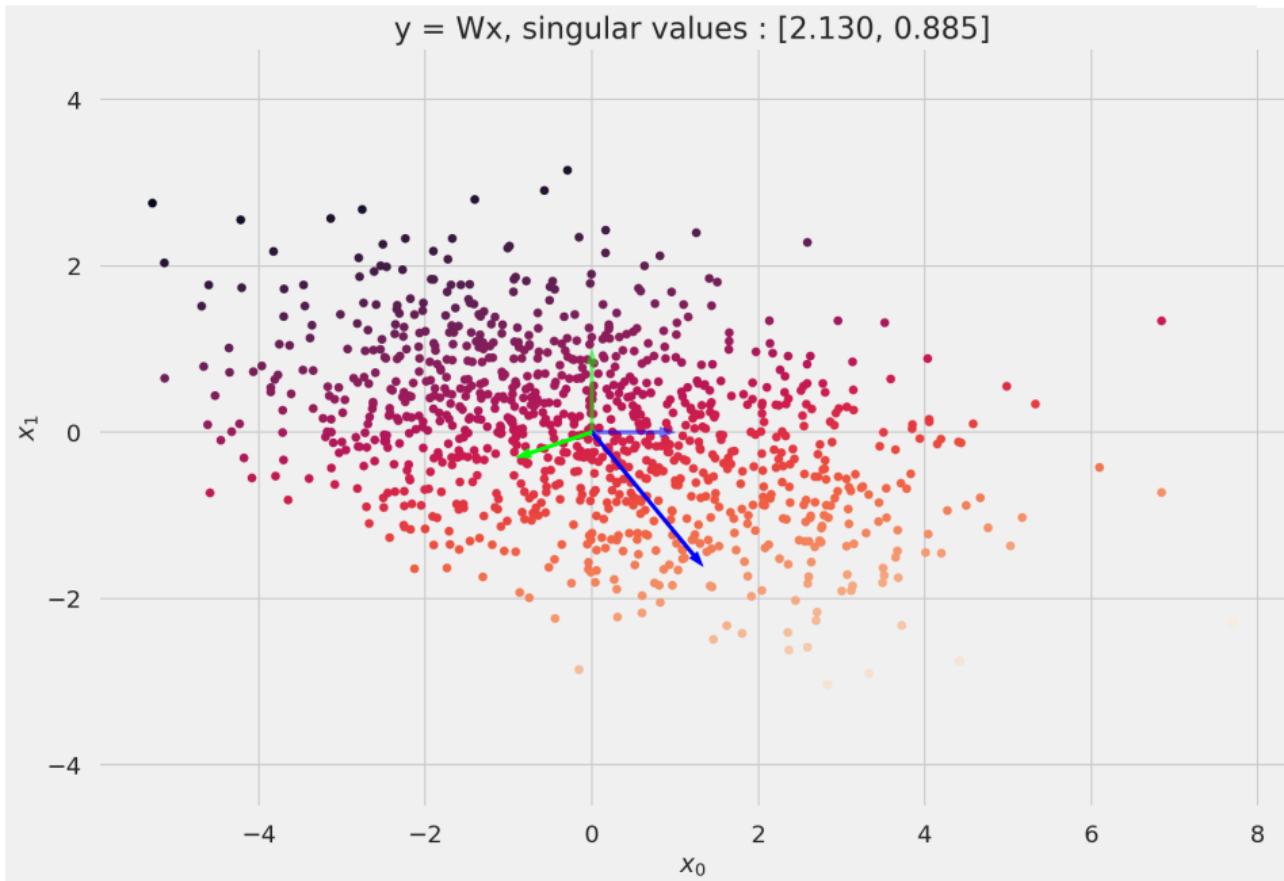
- ▶ $y = Wx$
- ▶ What is W actually doing to x ?
- ▶ Let's perform singular value decomposition¹ for some intuition:

$$W = U \times S \times V^T$$
$$\begin{bmatrix} 0.19 & 1.84 \\ -0.97 & -0.93 \end{bmatrix} = \underbrace{\begin{bmatrix} 0.83 & -0.55 \\ -0.55 & -0.83 \end{bmatrix}}_{\text{rotation}} \times \underbrace{\begin{bmatrix} 2.17 & 0.00 \\ 0.00 & 0.74 \end{bmatrix}}_{\text{scale}} \times \underbrace{\begin{bmatrix} 0.32 & 0.94 \\ 0.94 & -0.32 \end{bmatrix}}_{\text{reflection}}$$

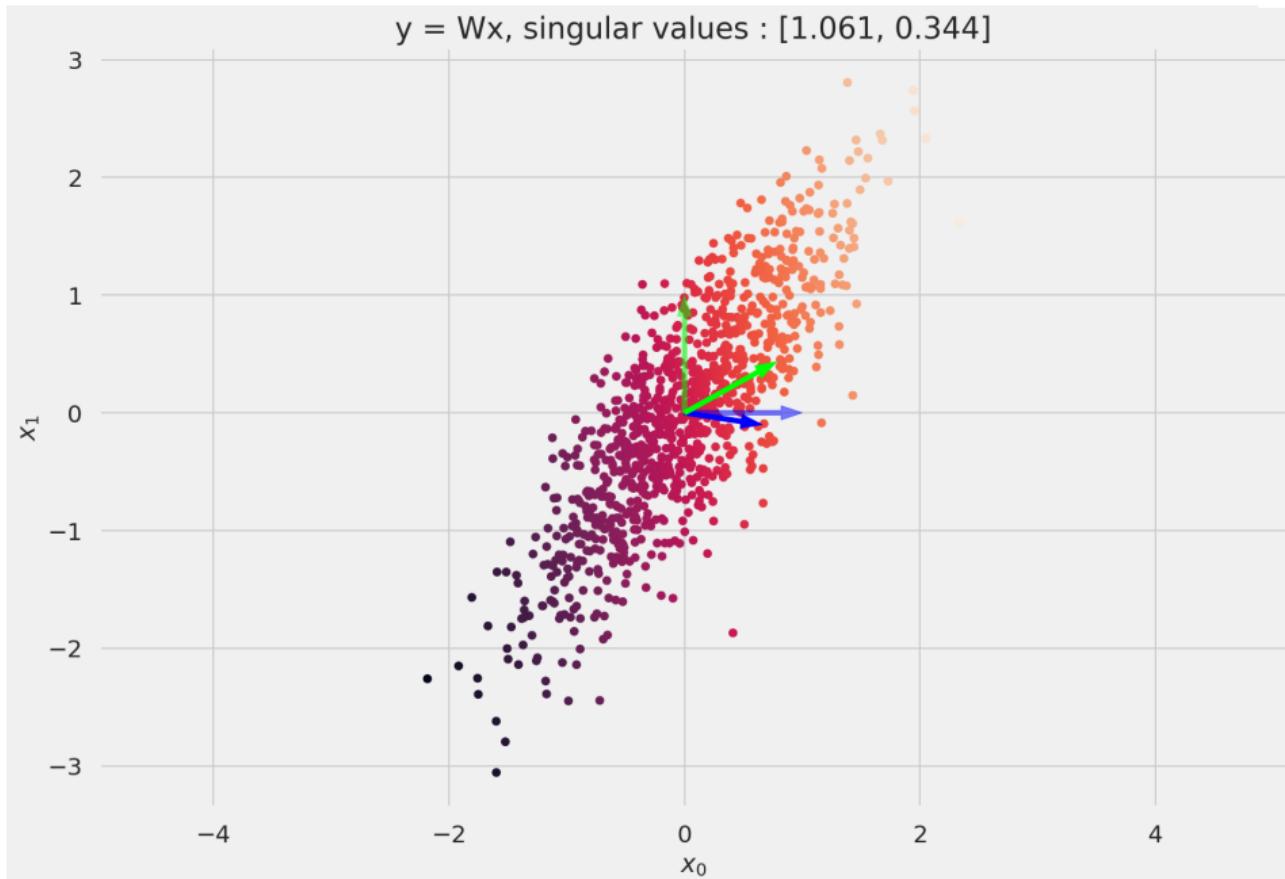
¹Mathematics for Machine Learning, ch. 4

A simple transformation

$y = Wx$, singular values : [2.130, 0.885]



A simple transformation



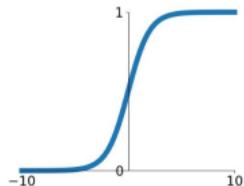
A simple transformation

We can **scale**, **rotate** and **reflect** data. Can we do **more**?

Non-linear functions¹

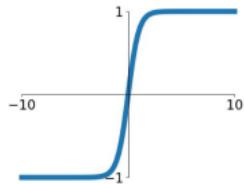
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



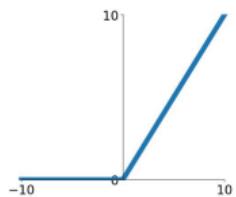
tanh

$$\tanh(x)$$



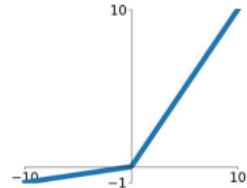
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

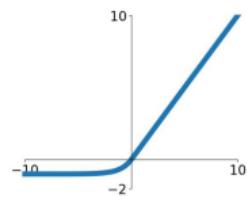


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

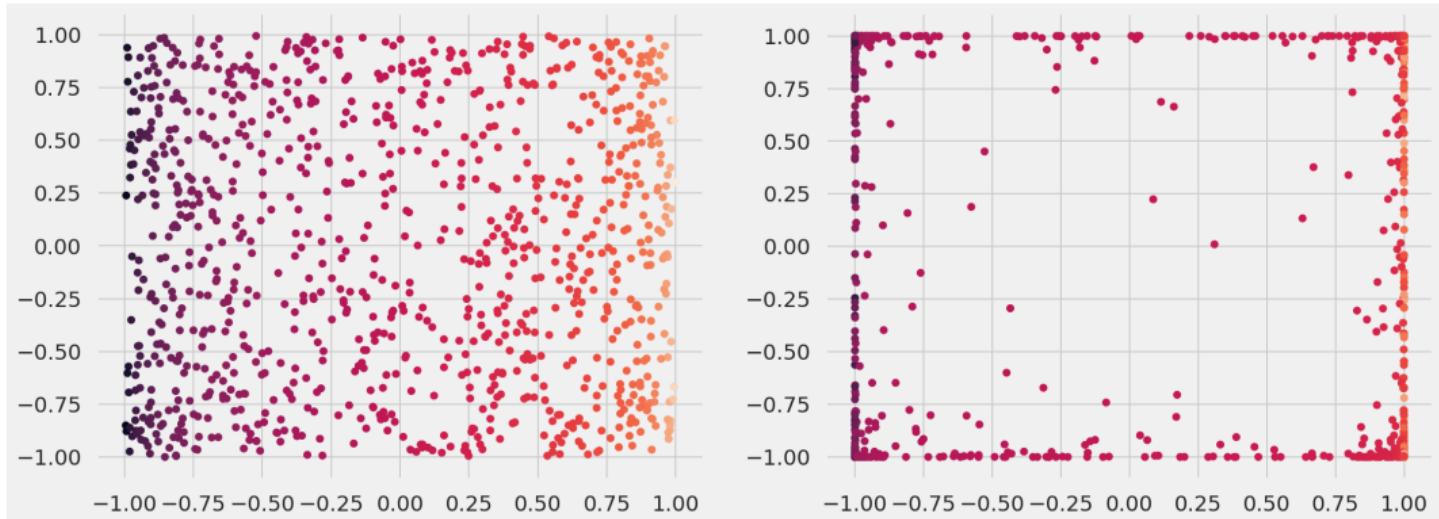
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



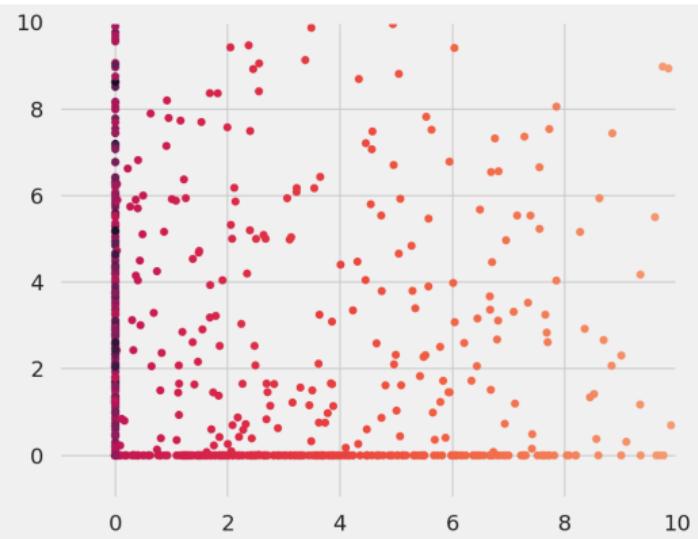
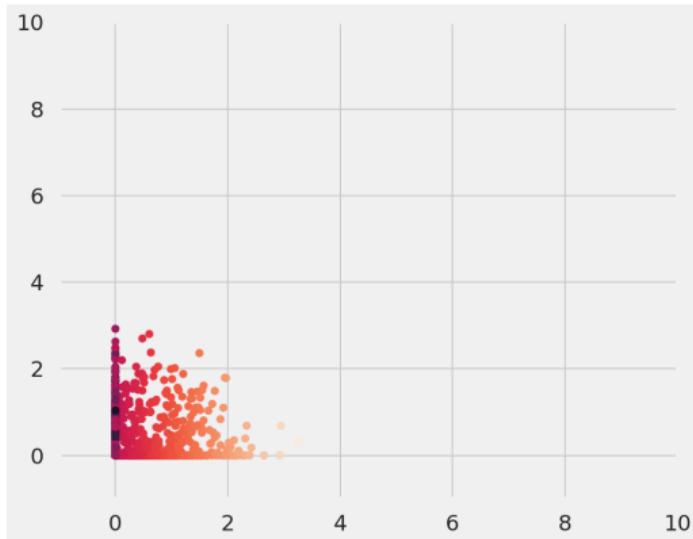
¹from CS231n, lecture 4, 2019

Non-linear transformation



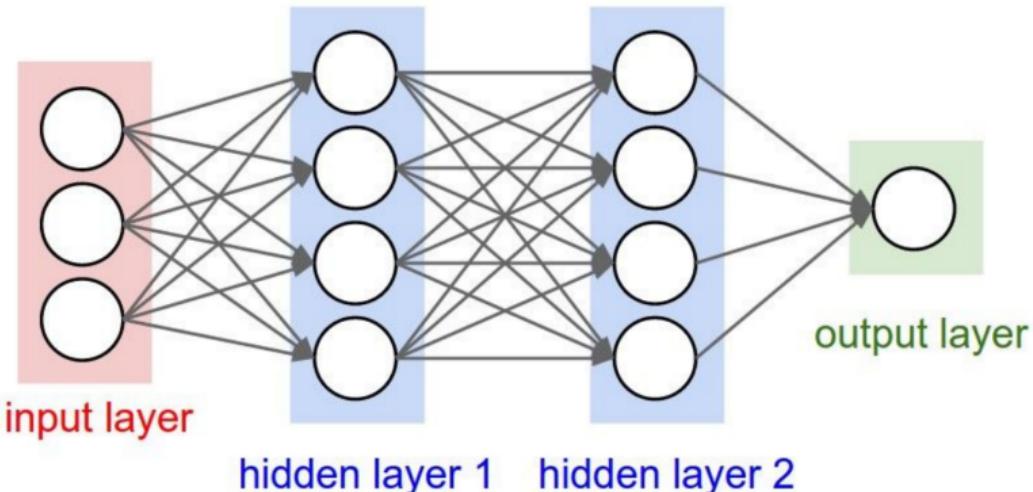
$\tanh(Sx)$ activation function with two different scale factors (left=1.0, right=5.0).

Non-linear transformation



$\text{ReLU}(\mathbf{S}\mathbf{x})$ activation function with two different scale factors (left=1.0, right=5.0).

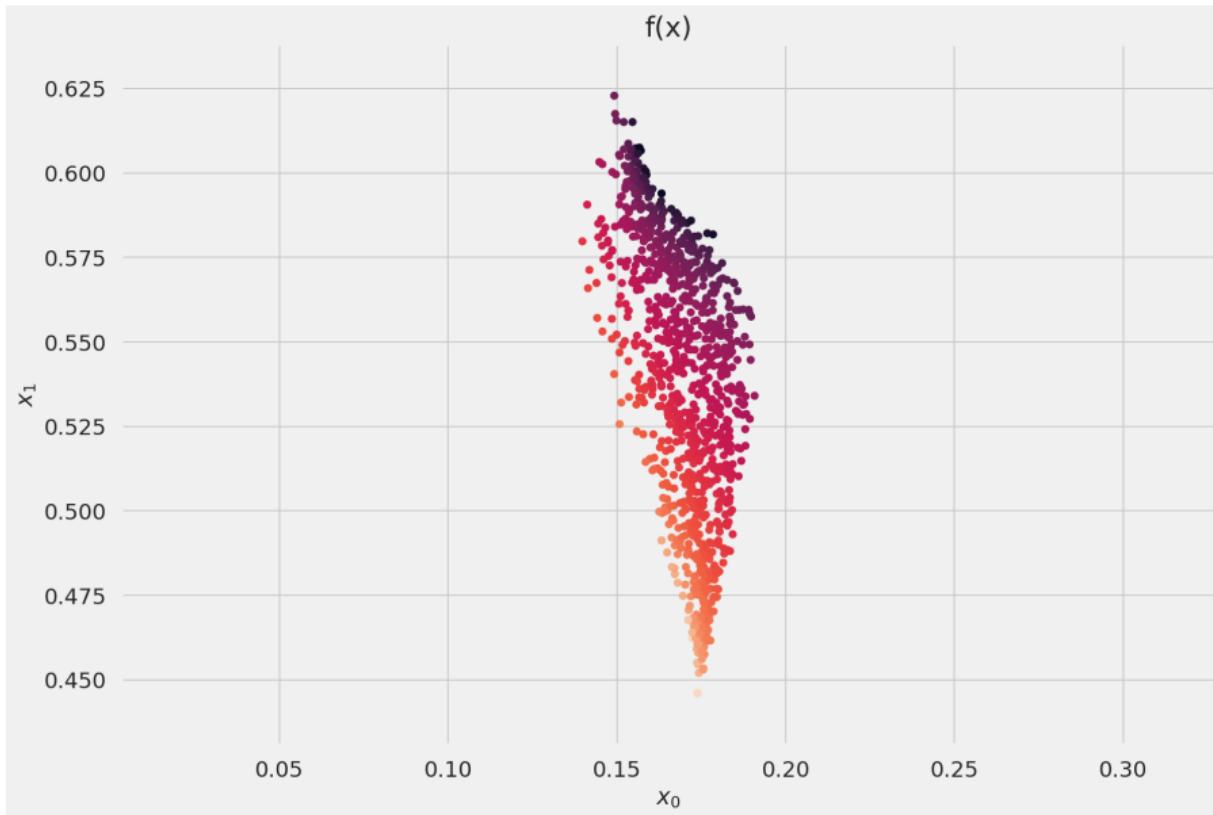
Neural Networks, finally



Typical "three-layer" or "two-hidden-layer" neural network

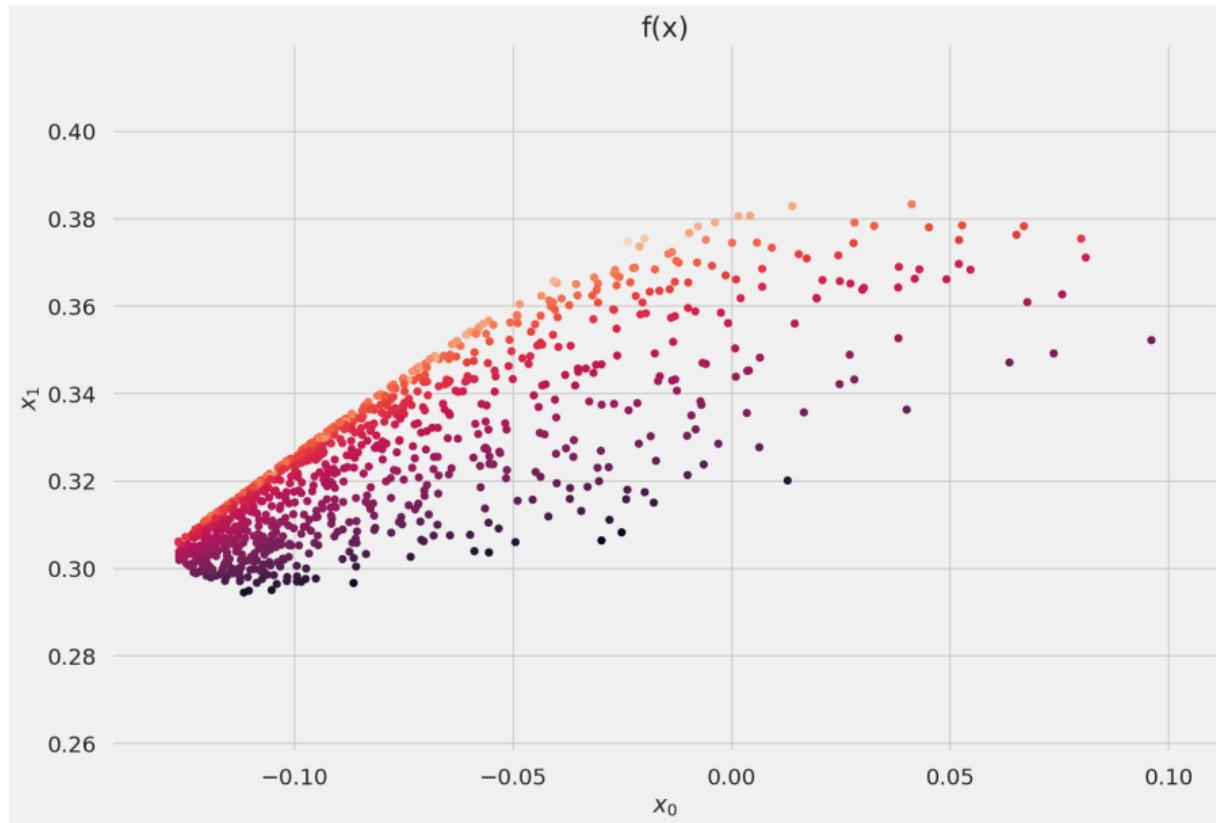
- ▶ Linear case: $f(\mathbf{x}) = \mathbf{W}\mathbf{x}$
- ▶ Neural Net: $f(\mathbf{x}) = \mathbf{W}_3 \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}))$

NN transformation



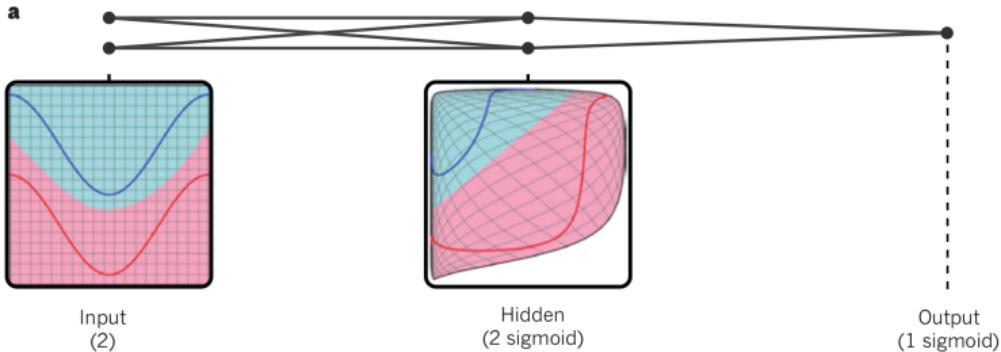
Transformation performed by a 5-layer **random** neural network

NN transformation



Transformation performed by a 5-layer **random** neural network

Warping space. Why is it usefull.



Space warping by a one hidden layer neural network for learning a linearized representation of the decision boundary¹.

¹from LeCun 2015, Nature.

Outline

Recap++

Understanding Neural Networks

Optimizing Neural Networks

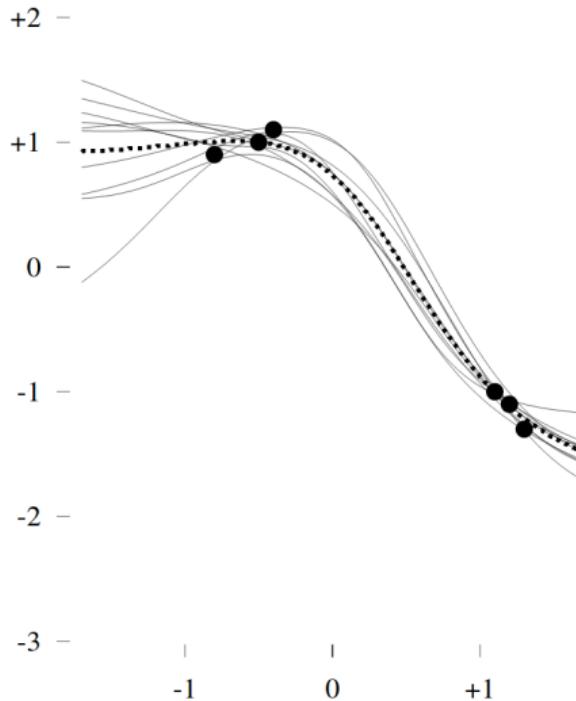
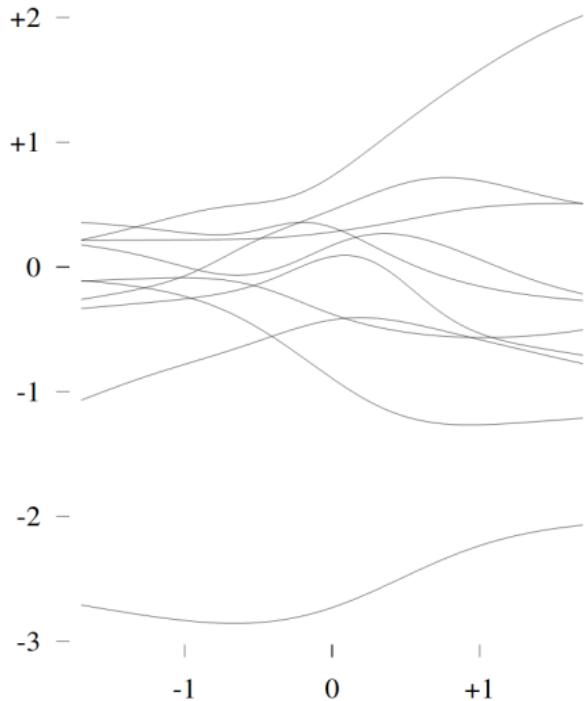
Computational Graphs

Forward pass

Backward pass

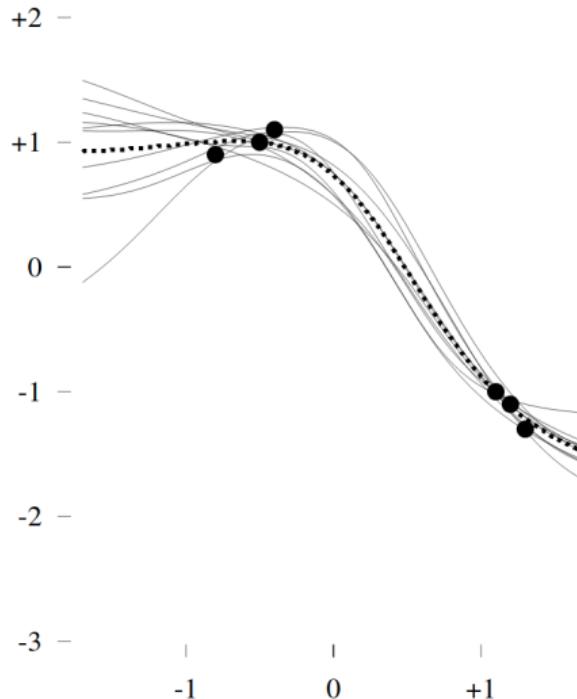
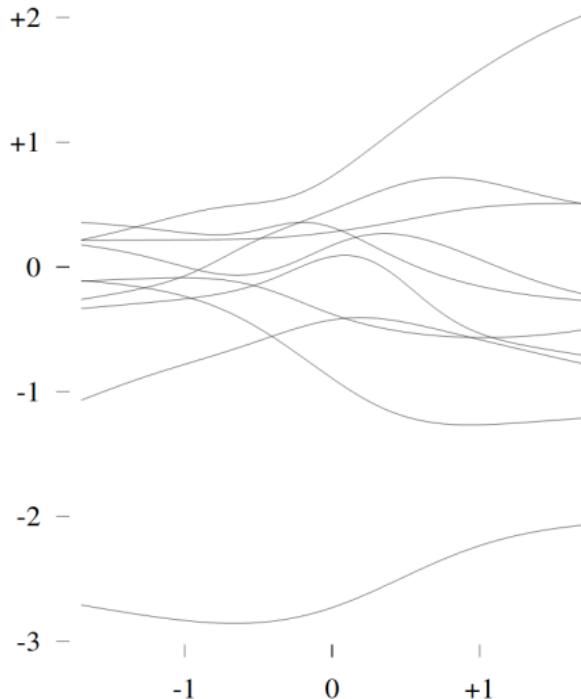
Wild beasts and how to tame them?

How to learn θ ?



¹ Radford M. Neal, Bayesian Learning for Neural Networks

How to learn θ ?



Sample 2.6 million networks and you're done!

¹ Radford M. Neal, Bayesian Learning for Neural Networks

How to learn θ ?



- ▶ Can we do better?

How to learn θ ?



- ▶ Can we do better?
- ▶ Use the lesson from the iterative linear regression

How to learn θ ?



- ▶ Can we do better?
- ▶ Use the lesson from the iterative linear regression
- ▶ Compute the **gradient** of $\mathcal{L}(\Theta)$ w.r.t. Θ ,

How to learn θ ?



- ▶ Can we do better?
- ▶ Use the lesson from the iterative linear regression
- ▶ Compute the **gradient** of $\mathcal{L}(\Theta)$ w.r.t. Θ ,
- ▶ Find the weights using gradient descent.

How to learn θ ?



- ▶ Can we do better?
- ▶ Use the lesson from the iterative linear regression
- ▶ Compute the **gradient** of $\mathcal{L}(\Theta)$ w.r.t. Θ ,
- ▶ Find the weights using gradient descent.
- ▶ It can get **tedious** and **inflexible**:
 - ▶ Additional cost functions, regularization loss, etc...
 - ▶ We would want to quickly experiment with different layers and activations.

But it's not that complicated¹

```
1 import numpy as np
2 from numpy.random import randn
3
4 N, D_in, H, D_out = 64, 1000, 100, 10
5 x, y = randn(N, D_in), randn(N, D_out)
6 w1, w2 = randn(D_in, H), randn(H, D_out)
7
8 for t in range(2000):
9     h = 1 / (1 + np.exp(-x.dot(w1)))
10    y_pred = h.dot(w2)
11    loss = np.square(y_pred - y).sum()
12    print(t, loss)
13
14    grad_y_pred = 2.0 * (y_pred - y)
15    grad_w2 = h.T.dot(grad_y_pred)
16    grad_h = grad_y_pred.dot(w2.T)
17    grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19    w1 -= 1e-4 * grad_w1
20    w2 -= 1e-4 * grad_w2
```

¹from CS231n, lecture 4, 2019

Outline

Recap++

Understanding Neural Networks

Optimizing Neural Networks

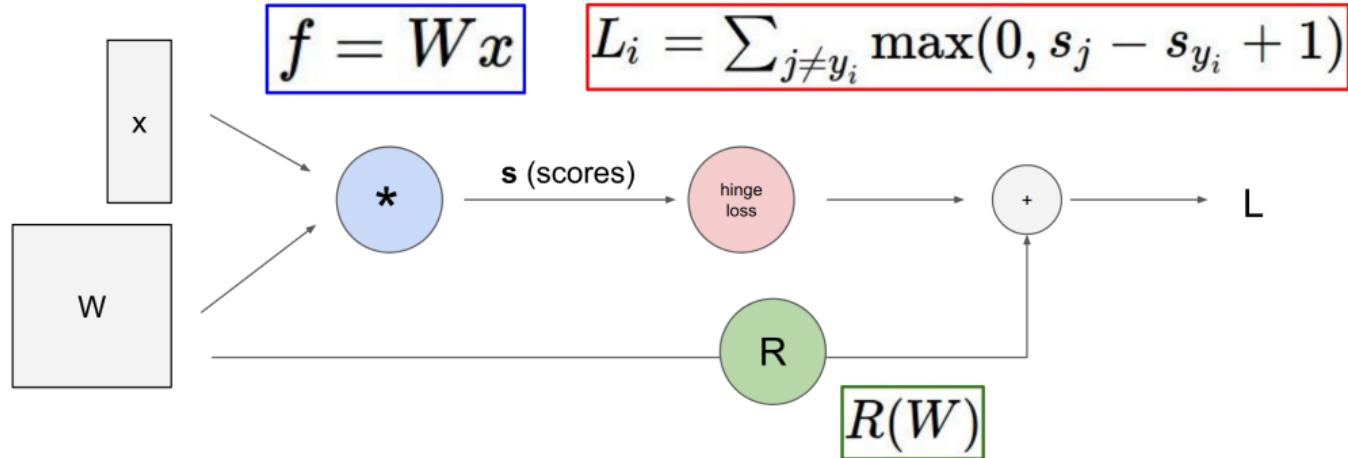
Computational Graphs

Forward pass

Backward pass

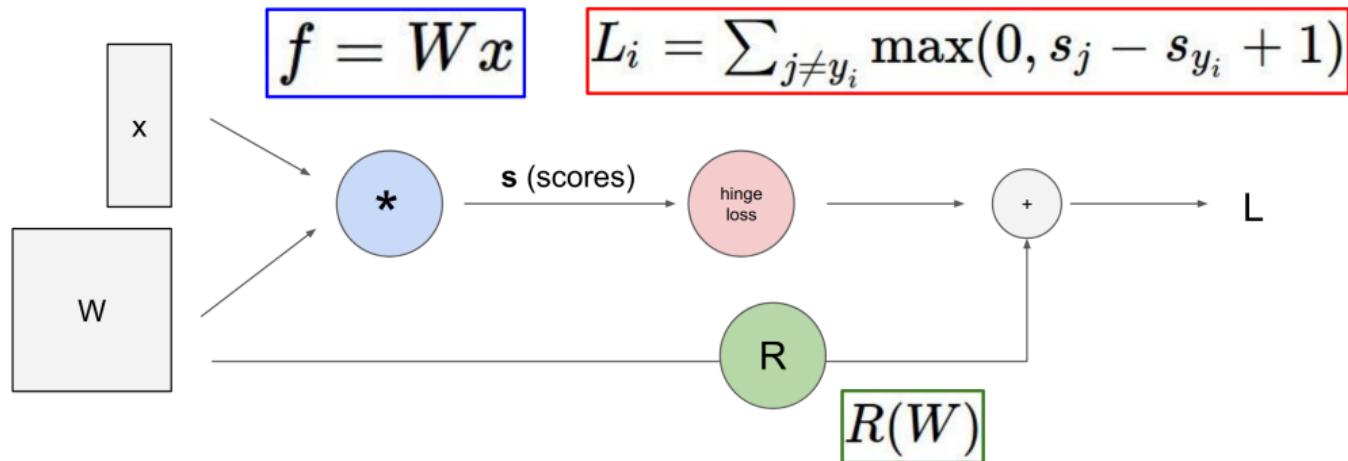
Wild beasts and how to tame them?

Computation graphs + automatic differentiation



¹Grosse, Ba, CS421

Computation graphs + automatic differentiation



Automatic differentiation: takes a program which computes a value, and automatically constructs a procedure for computing derivatives of that value¹.

¹Grosse, Ba, CS421

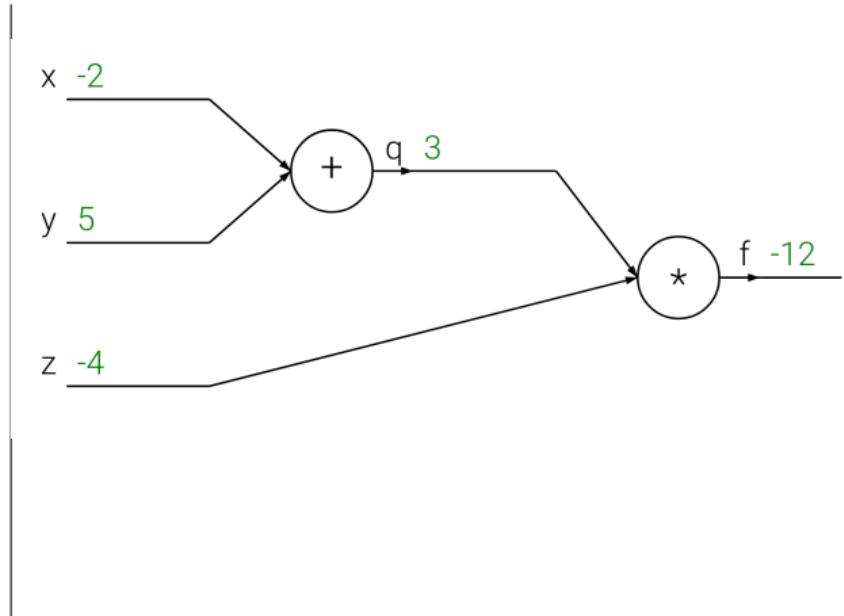
Simple example¹

$$f(x, y, z) = (x + y)z$$

¹from CS231n, lecture 4, 2019

Simple example¹

$$f(x, y, z) = (x + y)z$$



¹from CS231n, lecture 4, 2019

Simple example¹

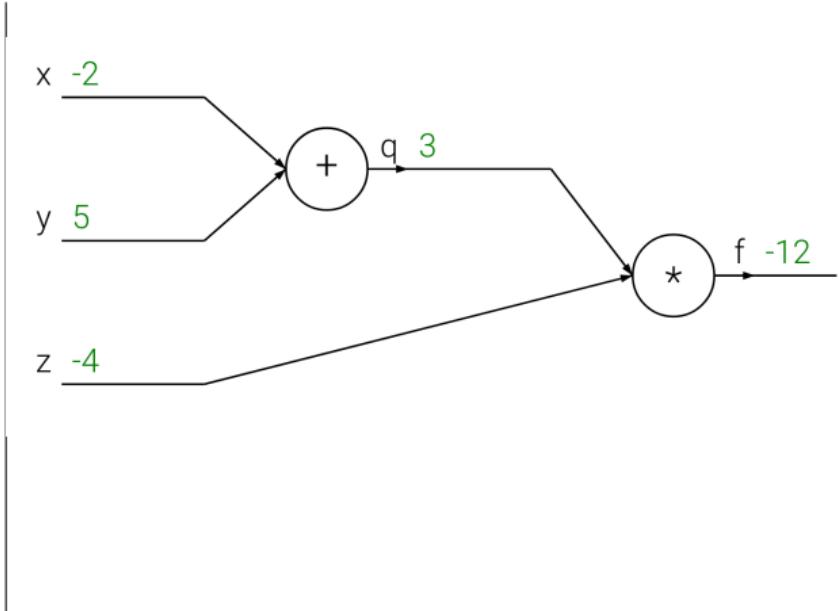
$$f(x, y, z) = (x + y)z$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

We want:

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$



¹from CS231n, lecture 4, 2019

Simple example¹

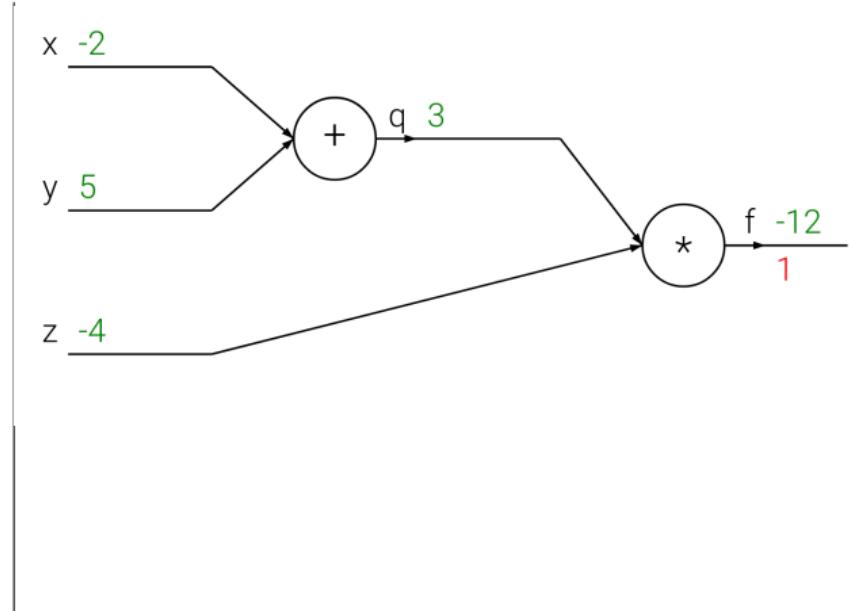
$$f(x, y, z) = (x + y)z$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

We want:

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$



¹from CS231n, lecture 4, 2019

Simple example¹

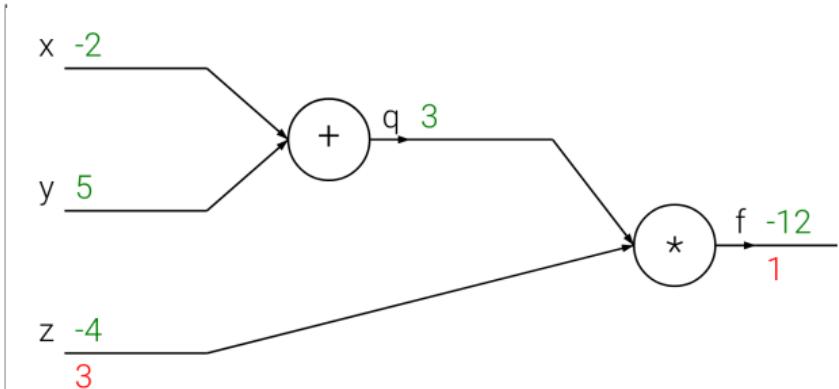
$$f(x, y, z) = (x + y)z$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

We want:

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$



¹from CS231n, lecture 4, 2019

Simple example¹

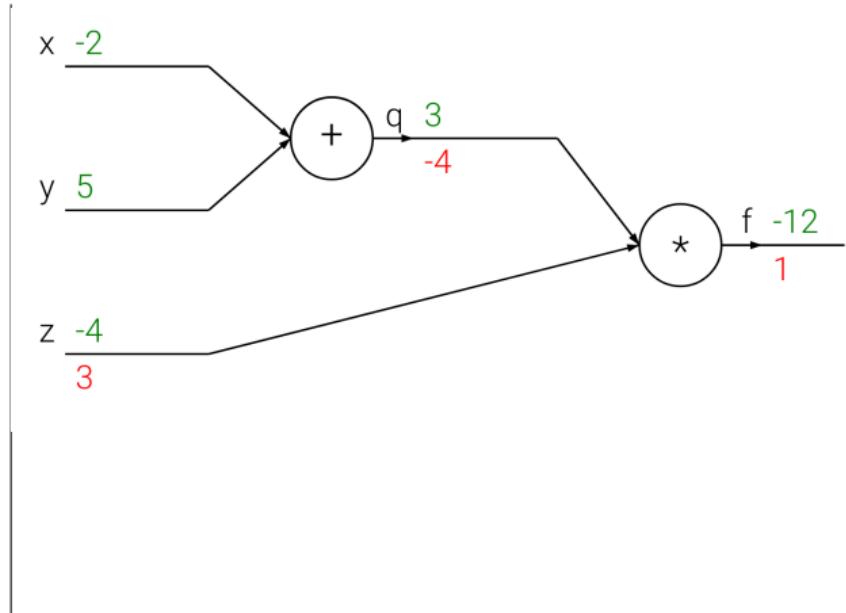
$$f(x, y, z) = (x + y)z$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

We want:

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$



¹from CS231n, lecture 4, 2019

Simple example¹

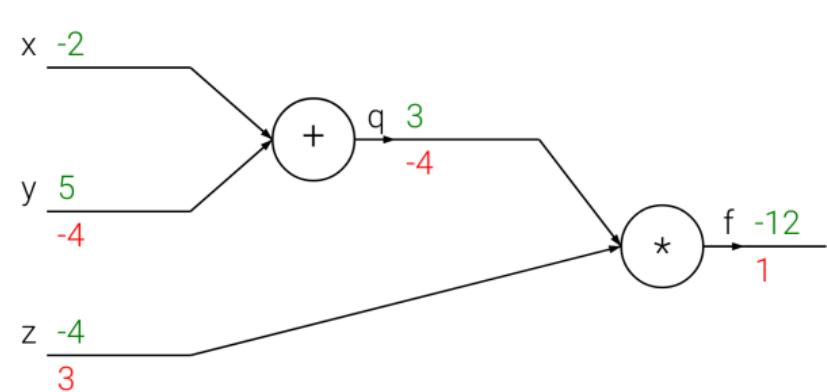
$$f(x, y, z) = (x + y)z$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

We want:

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$



$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

¹from CS231n, lecture 4, 2019

Simple example¹

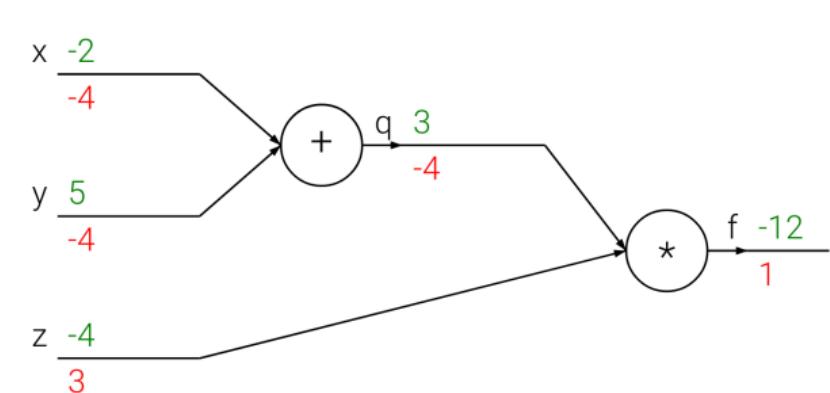
$$f(x, y, z) = (x + y)z$$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

We want:

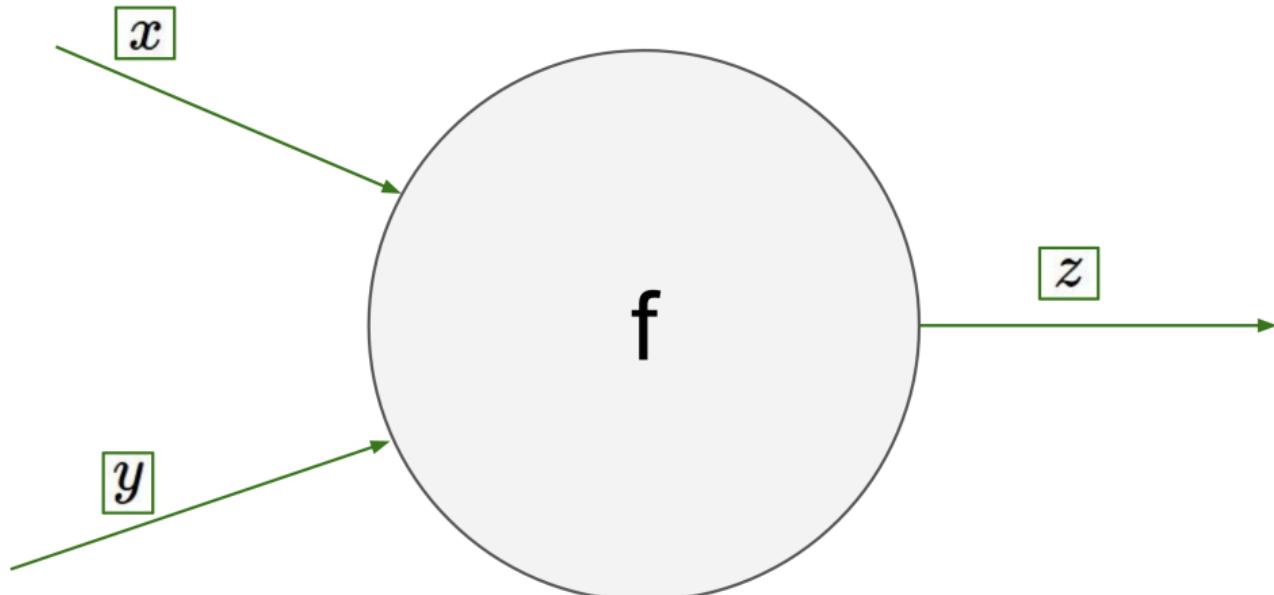
$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$



$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

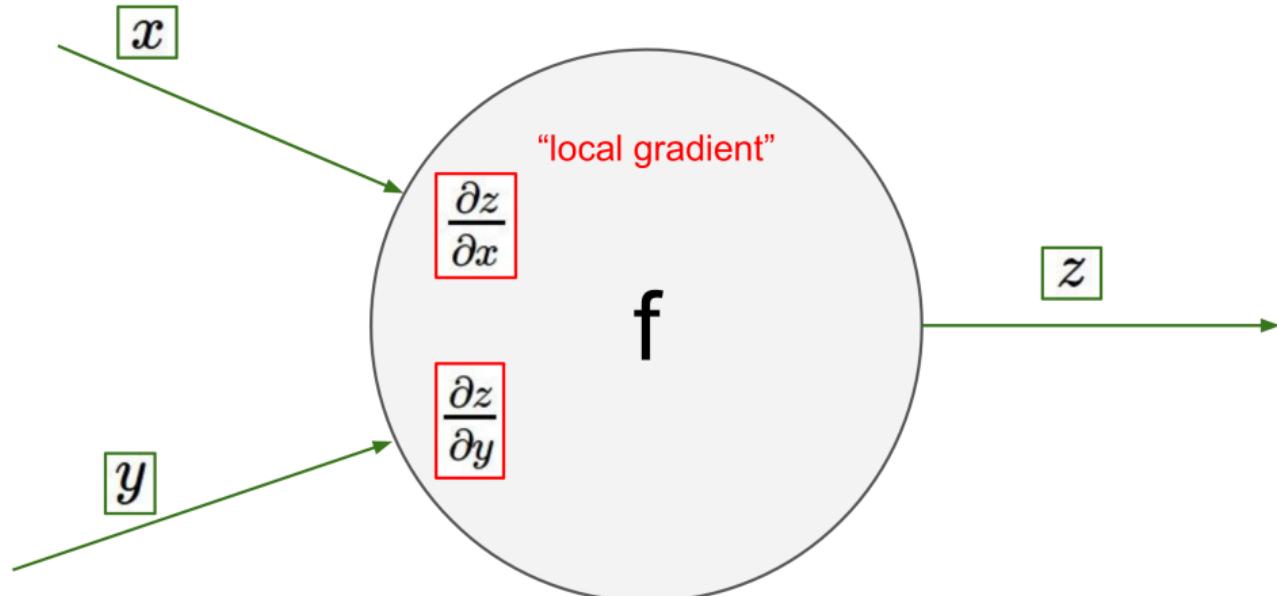
¹from CS231n, lecture 4, 2019

Backprop¹



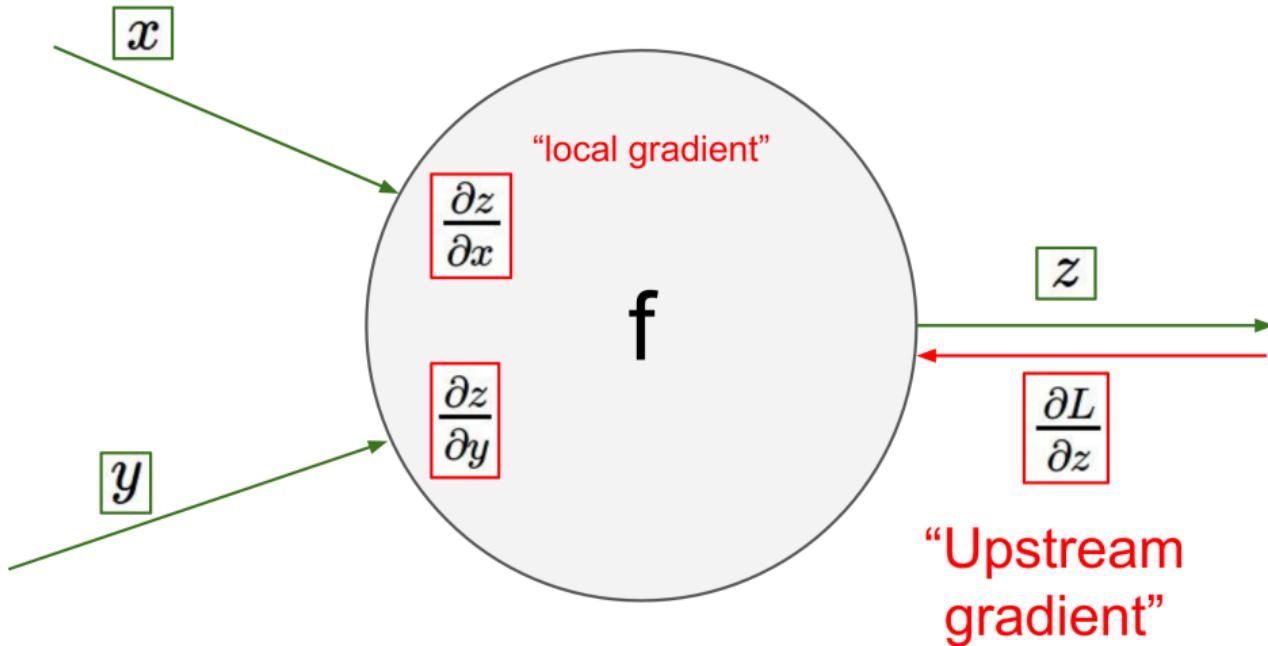
¹from CS231n, lecture 4, 2019

Backprop¹

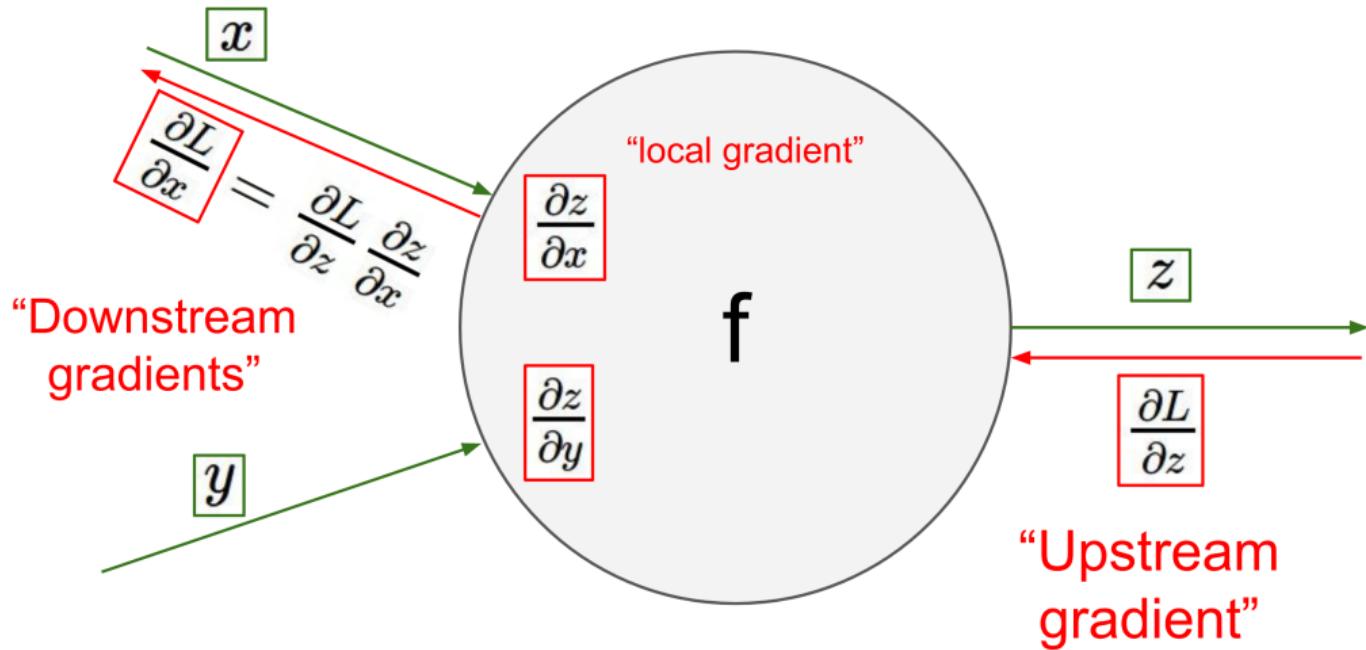


¹from CS231n, lecture 4, 2019

Backprop¹

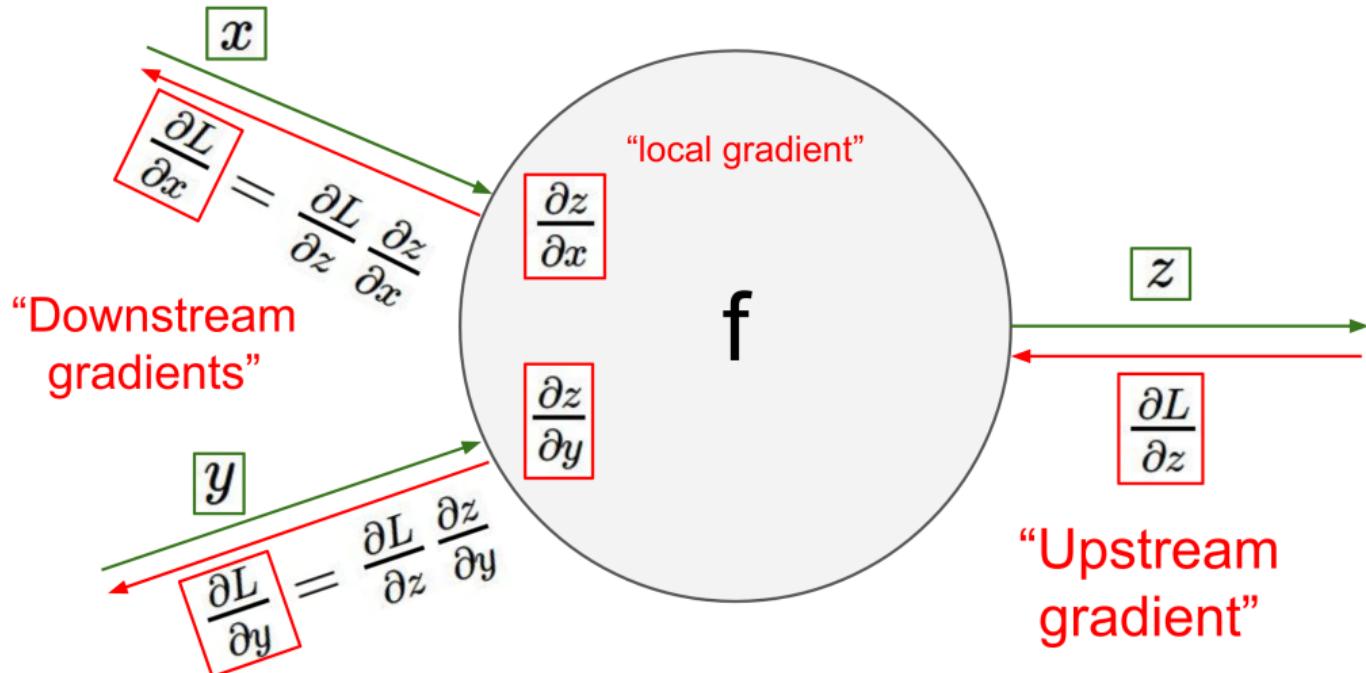


¹from CS231n, lecture 4, 2019

Backprop¹

¹from CS231n, lecture 4, 2019

Backprop¹



¹from CS231n, lecture 4, 2019

Outline

Recap++

Understanding Neural Networks

Optimizing Neural Networks

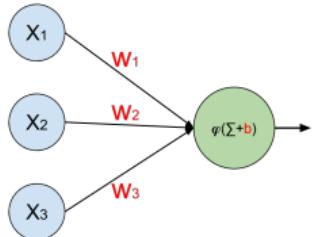
Computational Graphs

Forward pass

Backward pass

Wild beasts and how to tame them?

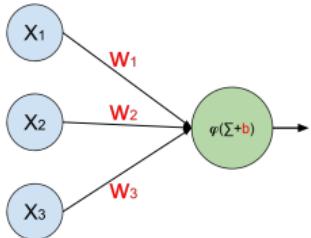
NN = Tensor Operations



Perceptron

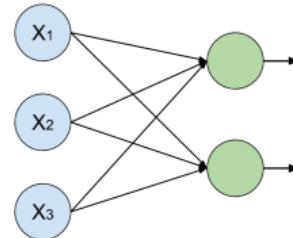
$$\begin{matrix} X1 & X2 & X3 \end{matrix} \bullet \begin{matrix} W1 \\ W2 \\ W3 \end{matrix} + \begin{matrix} B \end{matrix} = \begin{matrix} P \end{matrix}$$

NN = Tensor Operations



Perceptron

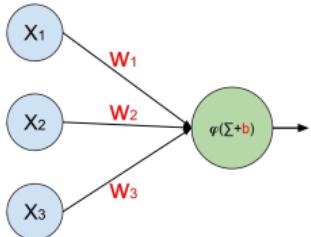
$$\begin{matrix} X1 & X2 & X3 \end{matrix} \bullet \begin{matrix} W1 \\ W2 \\ W3 \end{matrix} + \begin{matrix} B \end{matrix} = \begin{matrix} P \end{matrix}$$



Linear Layer

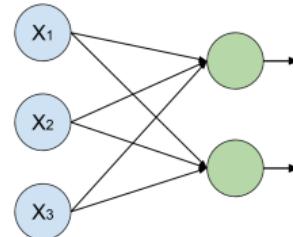
$$\begin{matrix} X1 & X2 & X3 \end{matrix} \bullet \begin{matrix} W11 & W12 \\ W21 & W22 \\ W31 & W32 \end{matrix} + \begin{matrix} B1 & B2 \end{matrix} = \begin{matrix} P1 & P2 \end{matrix}$$

NN = Tensor Operations



Perceptron

$$\begin{matrix} X_1 & X_2 & X_3 \end{matrix} \bullet \begin{matrix} W_1 \\ W_2 \\ W_3 \end{matrix} + \begin{matrix} B \end{matrix} = \begin{matrix} P \end{matrix}$$



Linear Layer

$$\begin{matrix} X_1 & X_2 & X_3 \end{matrix} \bullet \begin{matrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \end{matrix} + \begin{matrix} B_1 & B_2 \end{matrix} = \begin{matrix} P_1 & P_2 \end{matrix}$$

How about the case with multiple input tensors?

Softmax

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Input pixels, x



Feedforward output, y_i

	cat	dog	horse
cat	5	4	2
dog	4	2	8
horse	4	4	1

Forward propagation

Softmax output, $S(y_i)$

	cat	dog	horse
cat	0.71	0.26	0.04
dog	0.02	0.00	0.98
horse	0.49	0.49	0.02

Shape: (3, 32, 32)

Shape: (3,)

Shape: (3,)

Softmax ¹

¹[ljvmiranda921.github.io](https://jvmiranda921.github.io)

Loss Function

$$\frac{1}{n} \sum_i (h_\theta(x_i) - y_i)^2$$

$$\frac{1}{n} \sum_i -\log(h_\theta(x_i)_{y_i})$$

Input pixels, \mathbf{x}



Softmax output, $\mathbf{S}(\mathbf{y}_i)$

cat dog horse

0.71	0.26	0.04
0.02	0.00	0.98
0.49	0.49	0.02

Loss, $\mathbf{L}(a)$

NLL

0.34
0.02
0.71

$-\log(a)$ at the
correct classes

The correct class is
highlighted in red

Total: **1.07**

NLL ¹

¹[ljvmiranda921.github.io](https://jvmiranda921.github.io)

Outline

Recap++

Understanding Neural Networks

Optimizing Neural Networks

Computational Graphs

Forward pass

Backward pass

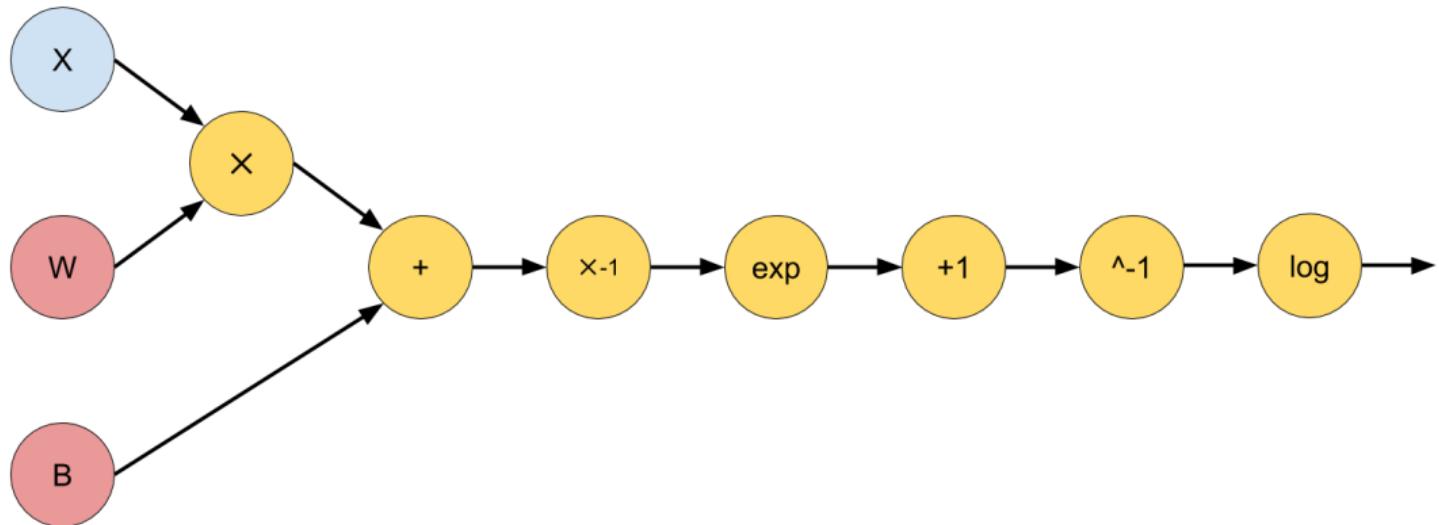
Wild beasts and how to tame them?

Computation Graph

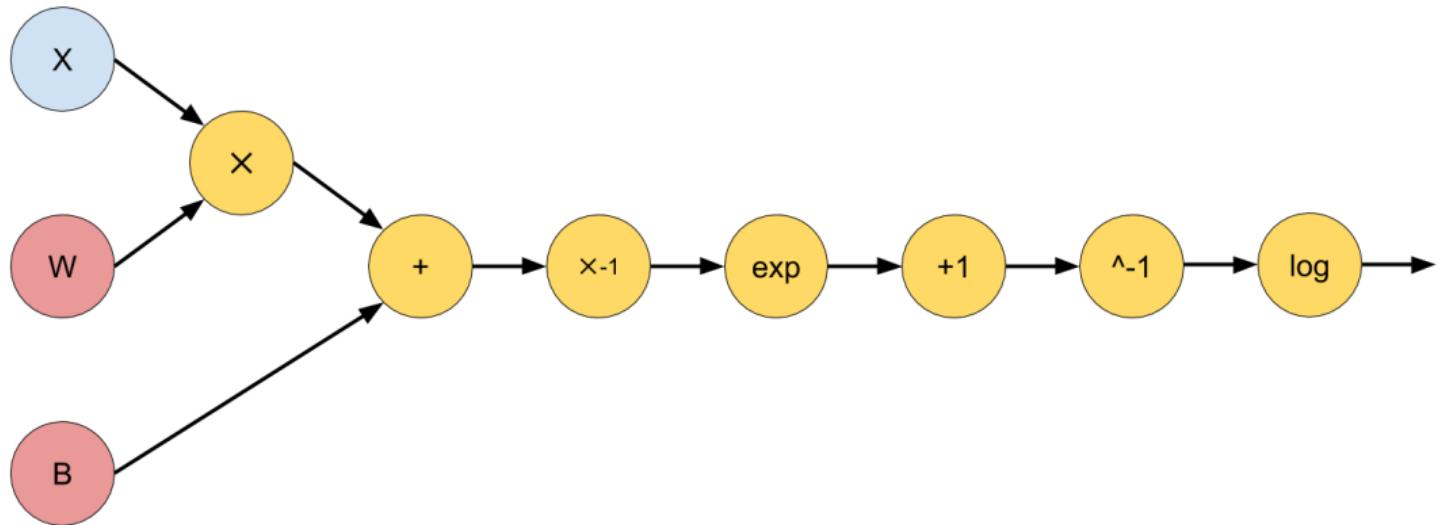
$$L(x, w, b) = \log \left(\frac{1}{e^{-(xw+b)} + 1} \right)$$

Computation Graph

$$L(x, w, b) = \log \left(\frac{1}{e^{-(xw+b)} + 1} \right)$$

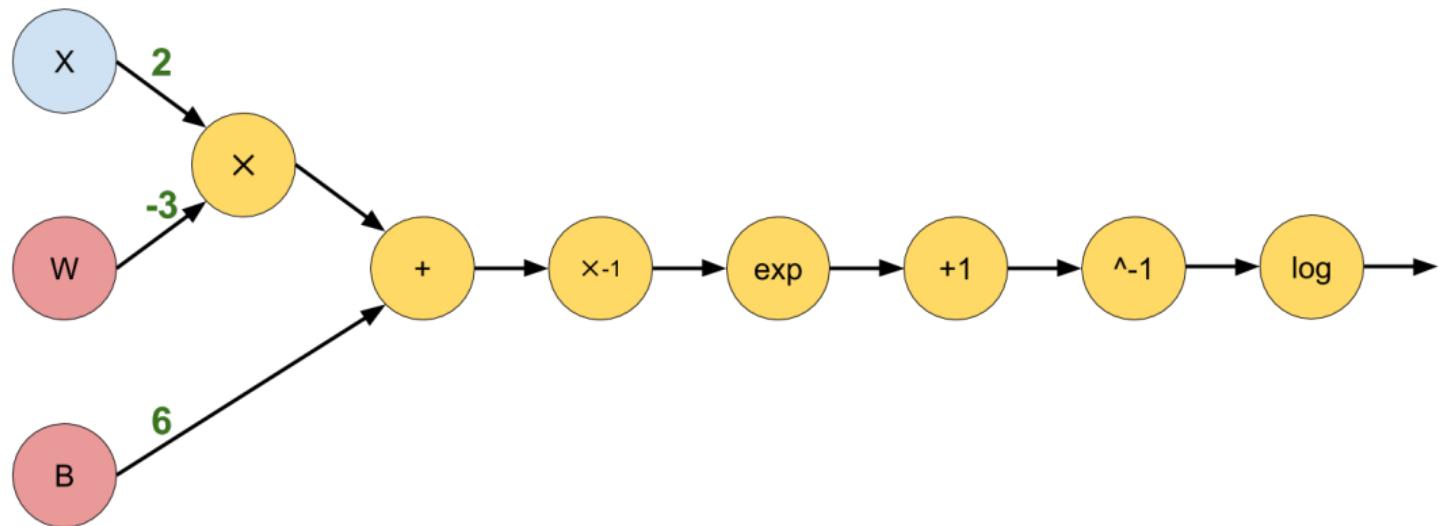


Forward



Task: Compute $L(x, w, b)$

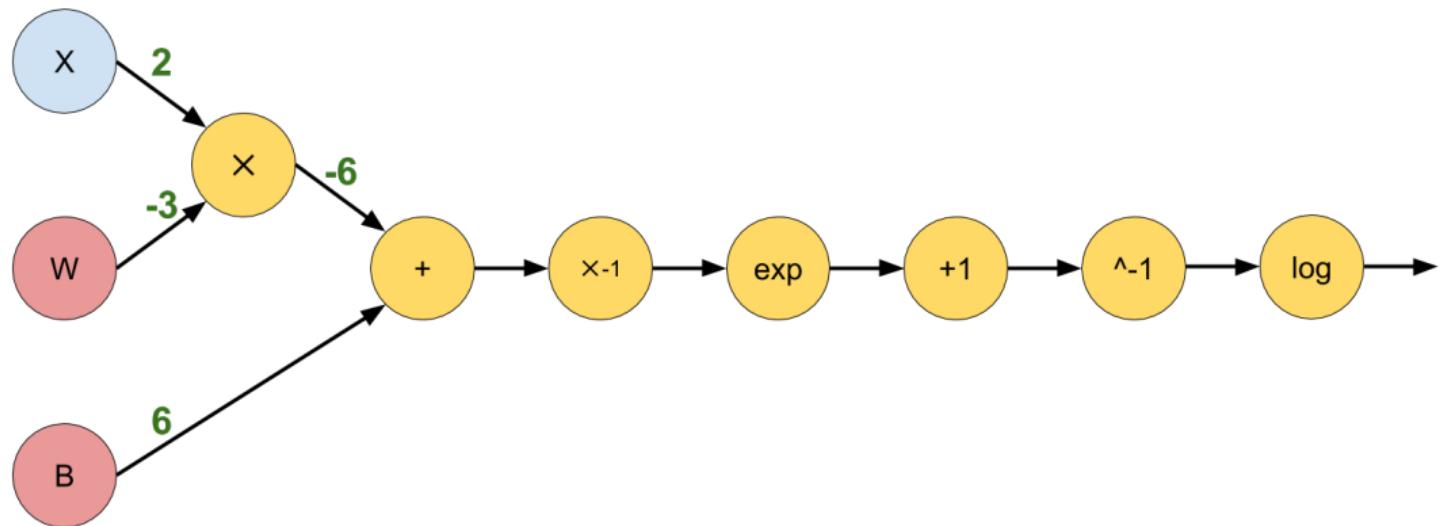
Forward



Task: Compute $L(x, w, b)$

$$x = 2, w = -3, b = 6$$

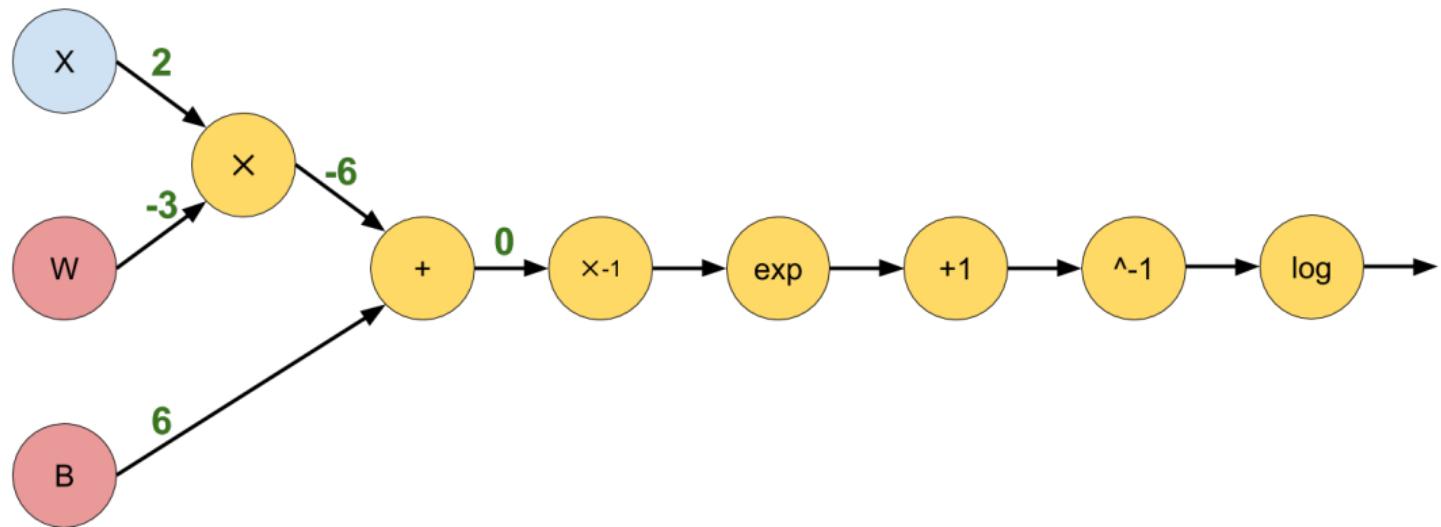
Forward



Task: Compute $L(x, w, b)$

$$x = 2, w = -3, b = 6$$

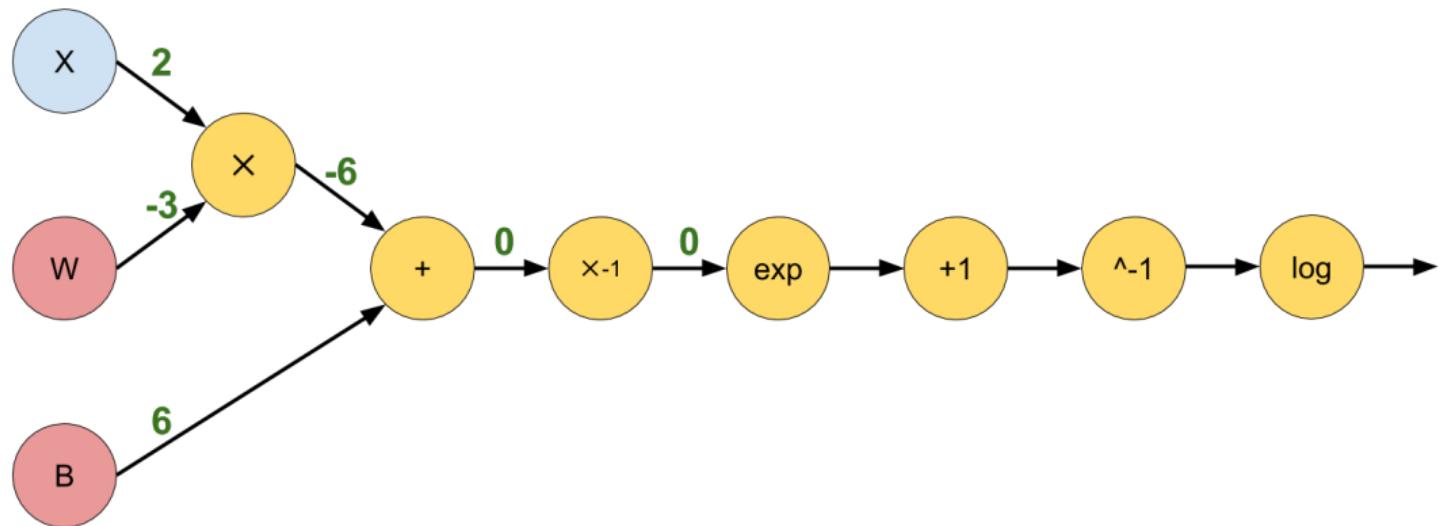
Forward



Task: Compute $L(x, w, b)$

$$x = 2, w = -3, b = 6$$

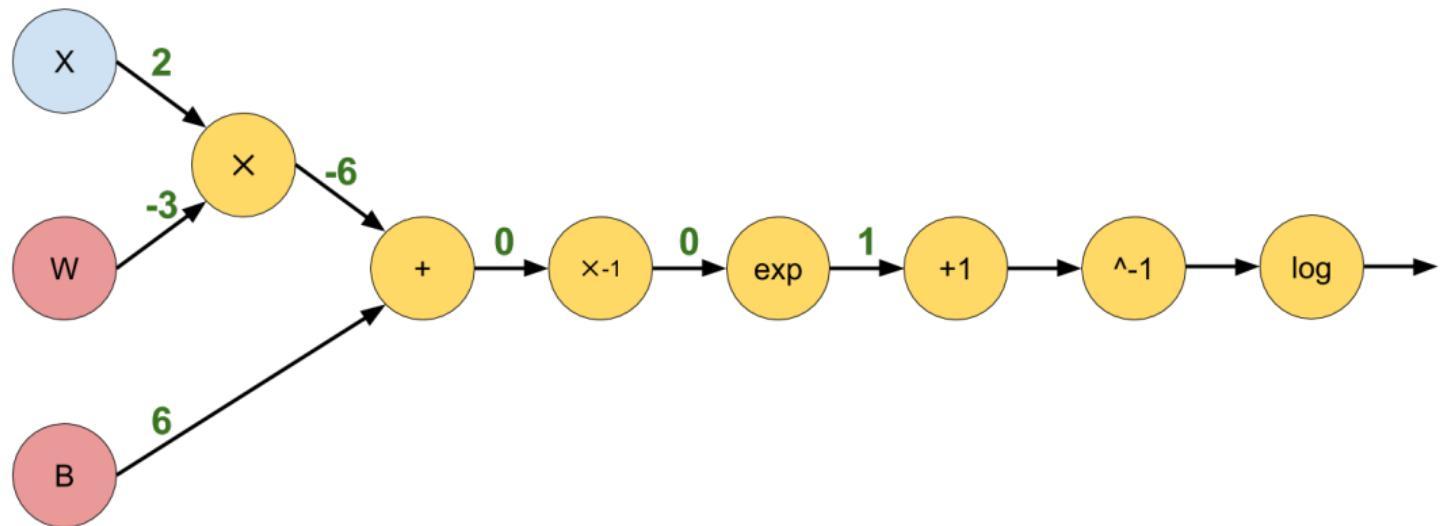
Forward



Task: Compute $L(x, w, b)$

$$x = 2, w = -3, b = 6$$

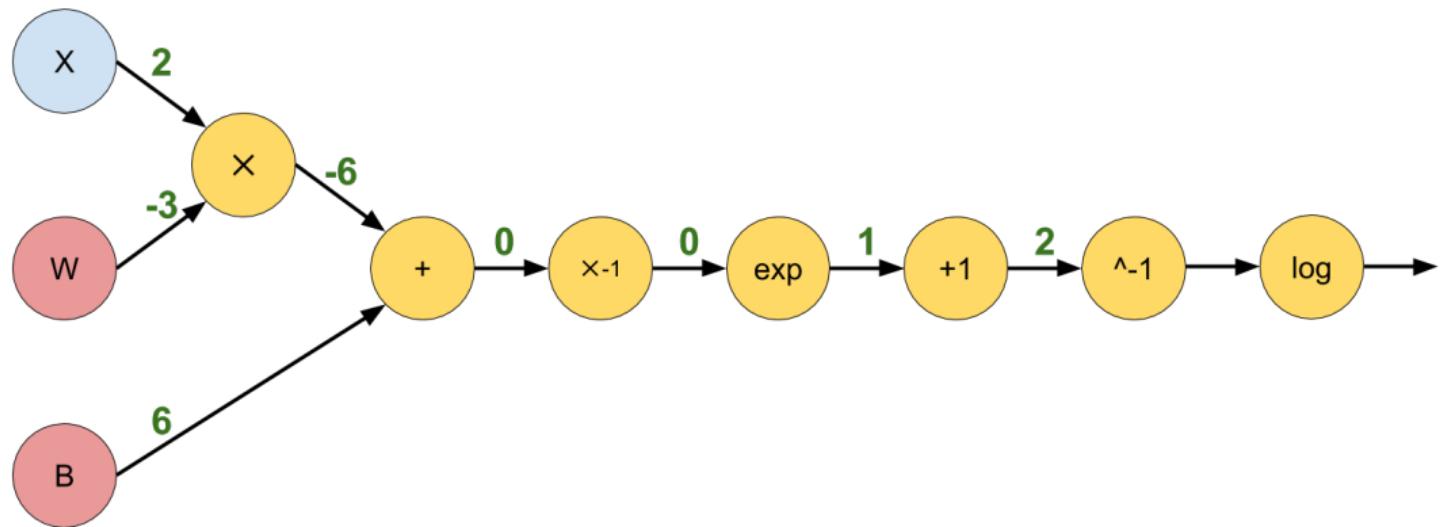
Forward



Task: Compute $L(x, w, b)$

$$x = 2, w = -3, b = 6$$

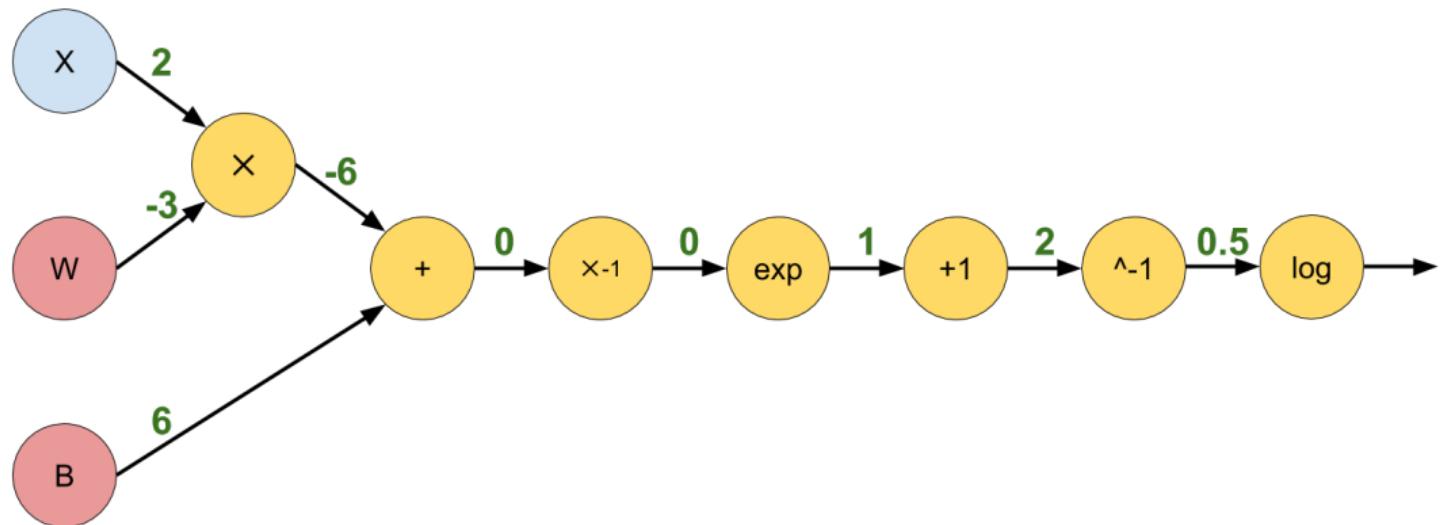
Forward



Task: Compute $L(x, w, b)$

$$x = 2, w = -3, b = 6$$

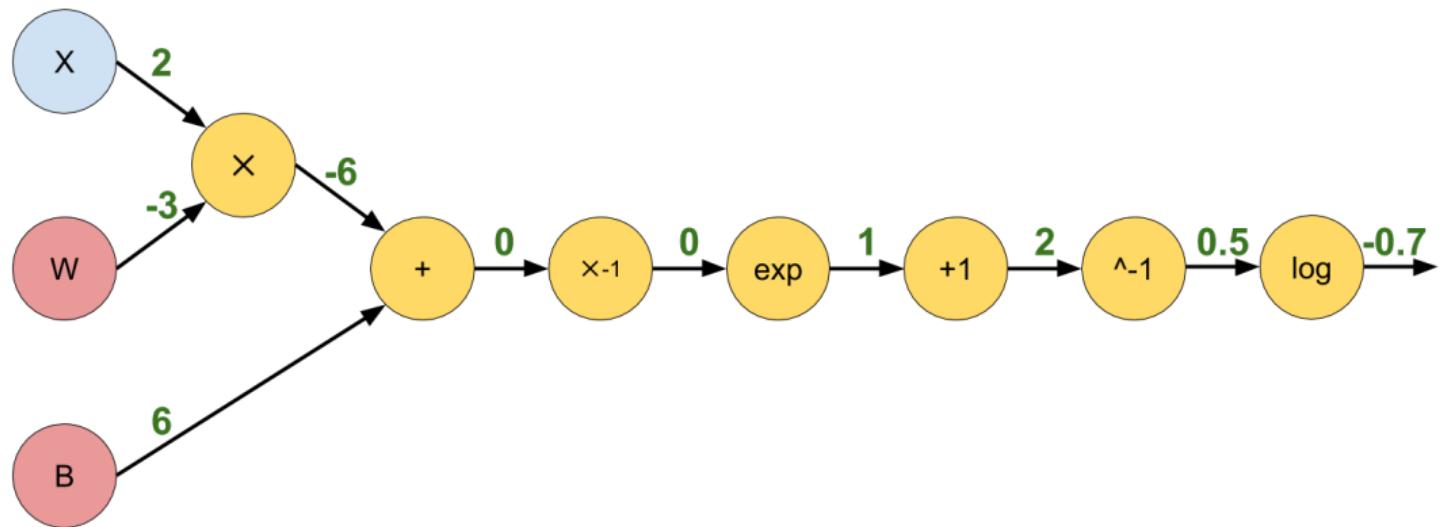
Forward



Task: Compute $L(x, w, b)$

$$x = 2, w = -3, b = 6$$

Forward

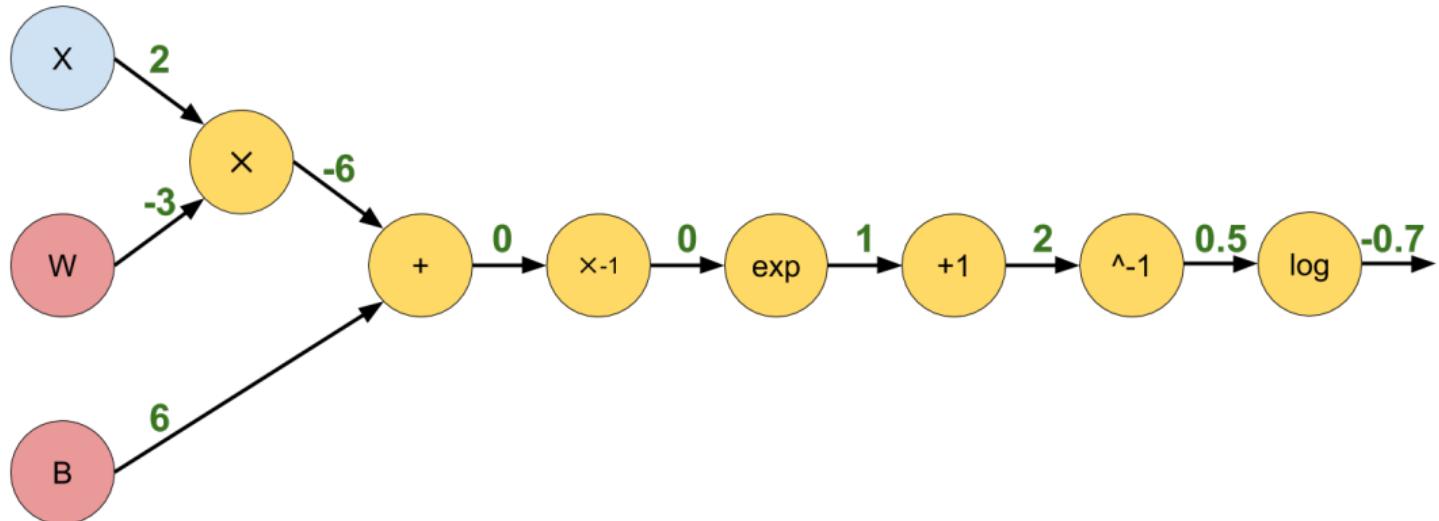


Task: Compute $L(x, w, b)$

$$x = 2, w = -3, b = 6$$

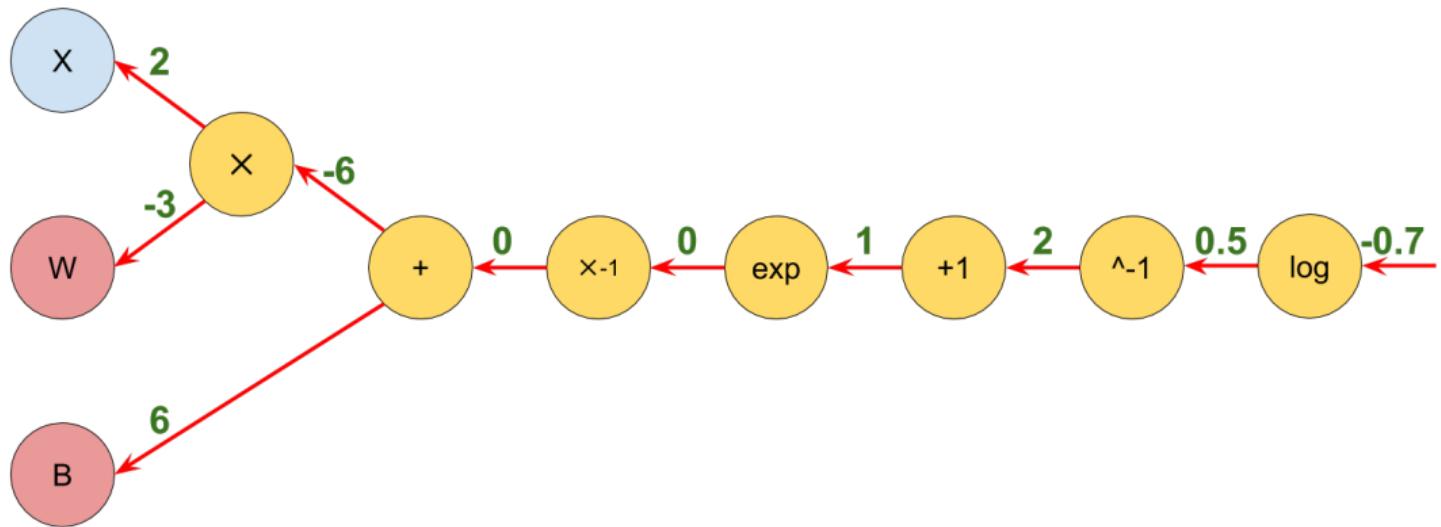
$$L(2, -3, 6) \simeq 0.7$$

Backward



Task: Compute $\nabla L = \left[\frac{\partial L}{\partial x}, \frac{\partial L}{\partial w}, \frac{\partial L}{\partial b} \right]$

Backward

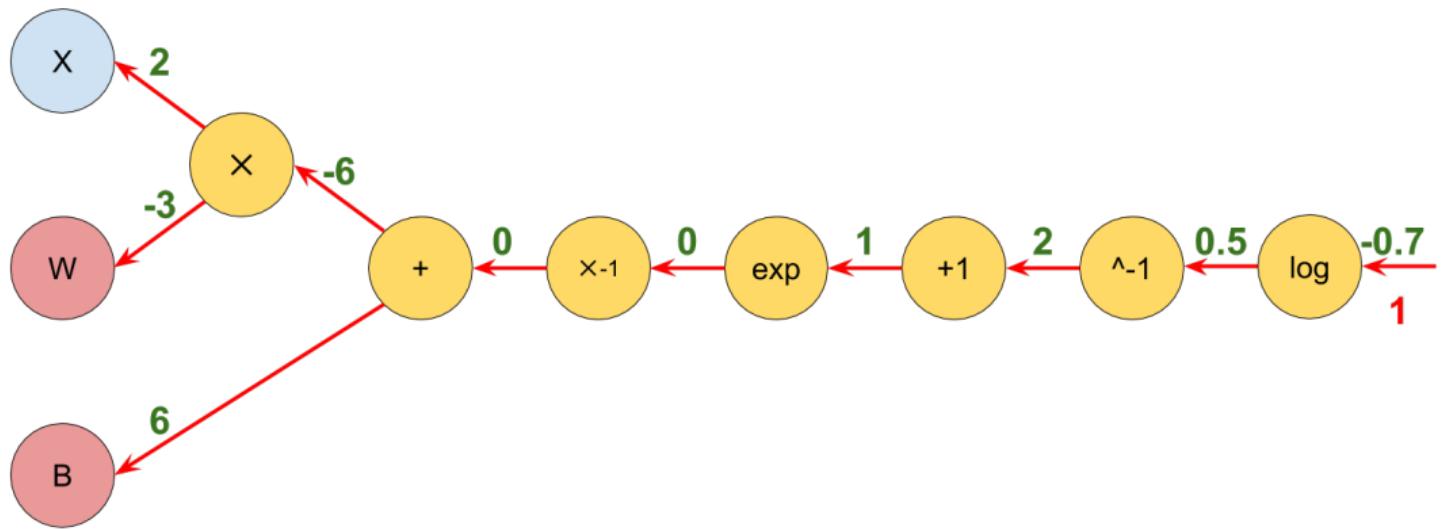


Task: Compute $\nabla L = \left[\frac{\partial L}{\partial x}, \frac{\partial L}{\partial w}, \frac{\partial L}{\partial b} \right]$

Chain-rule \implies each sub-module M :

- ▶ receives $\frac{\partial L}{\partial M(in)}$
- ▶ computes $\frac{\partial M(in)}{\partial in_k}$
- ▶ returns $\frac{\partial L}{\partial in_k} = \frac{\partial L}{\partial M(in)} \frac{\partial M(in)}{\partial in_k}$

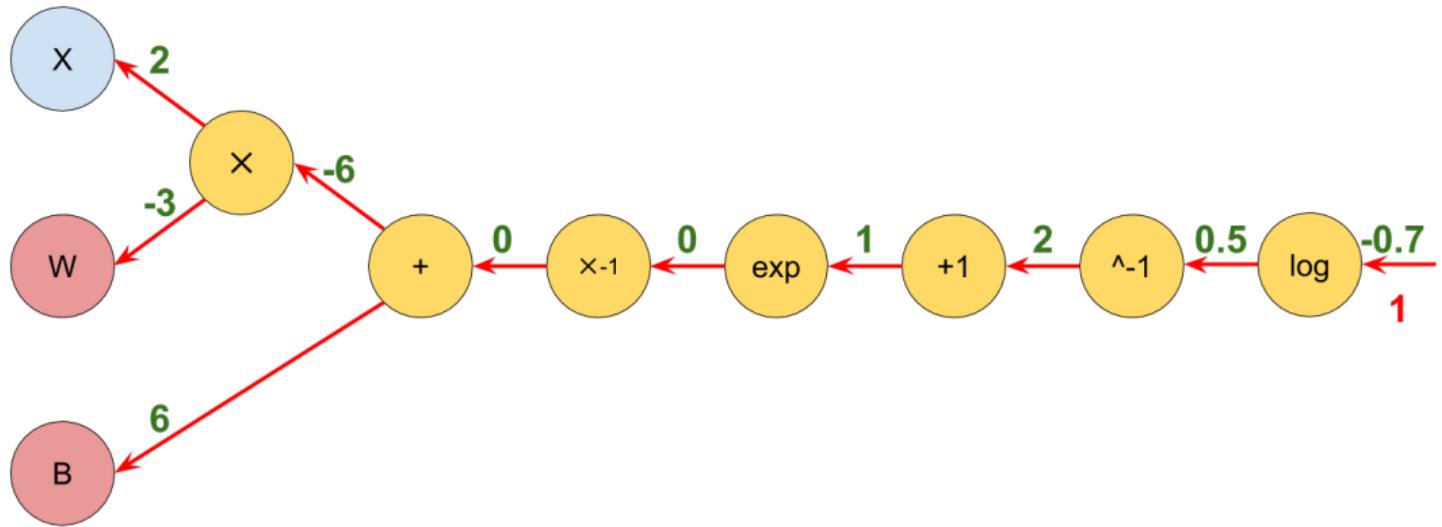
Backward



\log :

- ▶ receives $\frac{\partial L}{\partial \log(\text{in})} = 1$

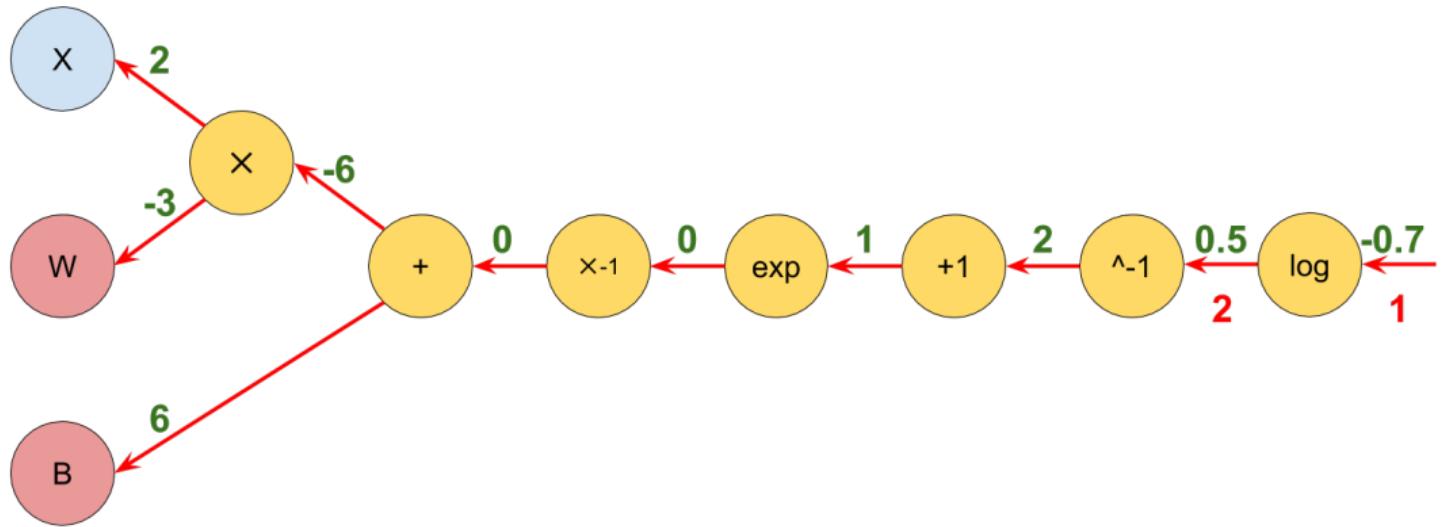
Backward



\log :

- ▶ receives $\frac{\partial L}{\partial \log(in)} = 1$
- ▶ computes $\frac{\partial \log(in)}{\partial in} = \frac{1}{in} = 2$

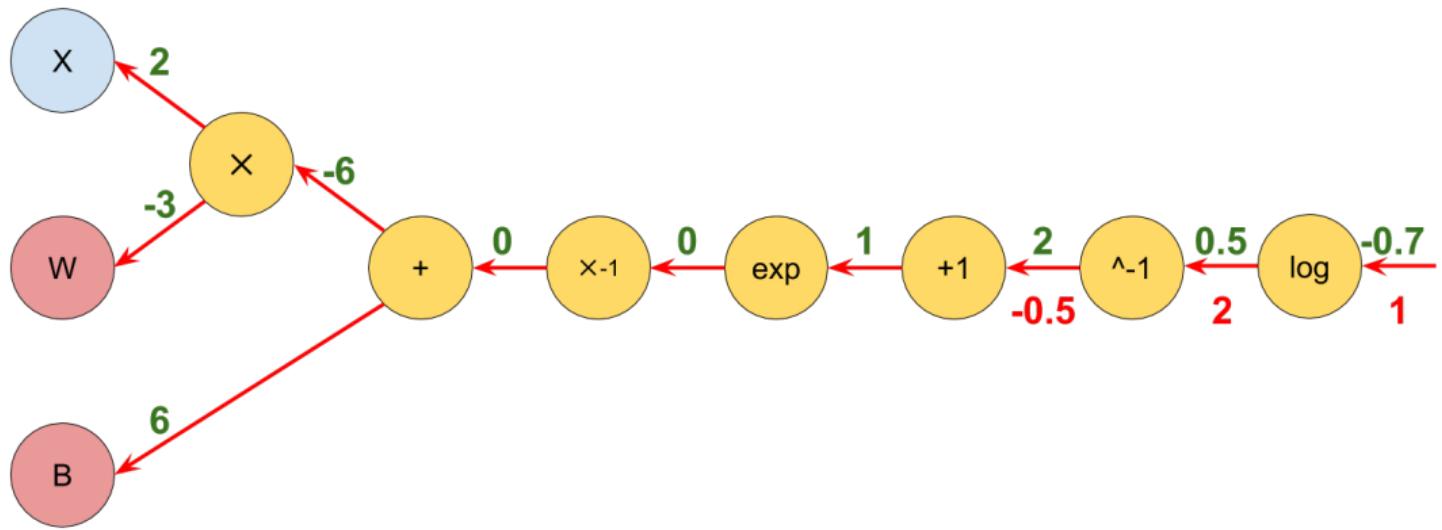
Backward



\log :

- ▶ receives $\frac{\partial L}{\partial \log(\text{in})} = 1$
- ▶ computes $\frac{\partial \log(\text{in})}{\partial \text{in}} = \frac{1}{\text{in}} = 2$
- ▶ returns $\frac{\partial L}{\partial \text{in}} = 1 * 2$

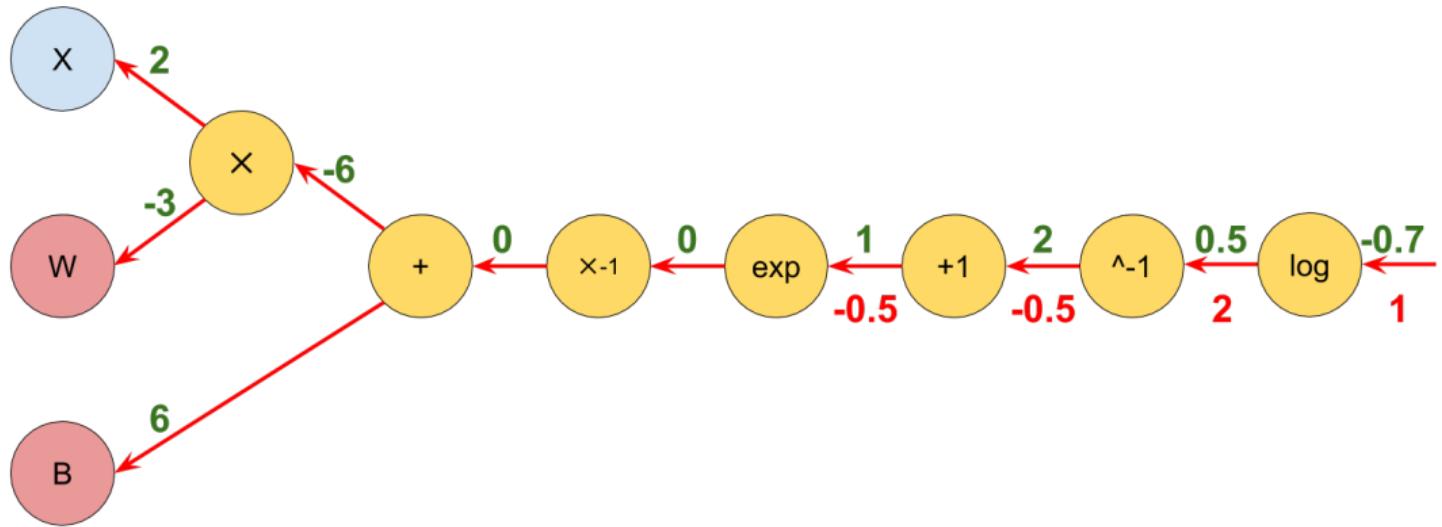
Backward



$\wedge - 1$:

- ▶ receives $\frac{\partial L}{\partial (\text{in})^{-1}} = 2$
- ▶ computes $\frac{\partial (\text{in})^{-1}}{\partial \text{in}} = -(\text{in})^{-2} = -0.25$
- ▶ returns $\frac{\partial L}{\partial \text{in}} = 2 * -0.25 = -0.5$

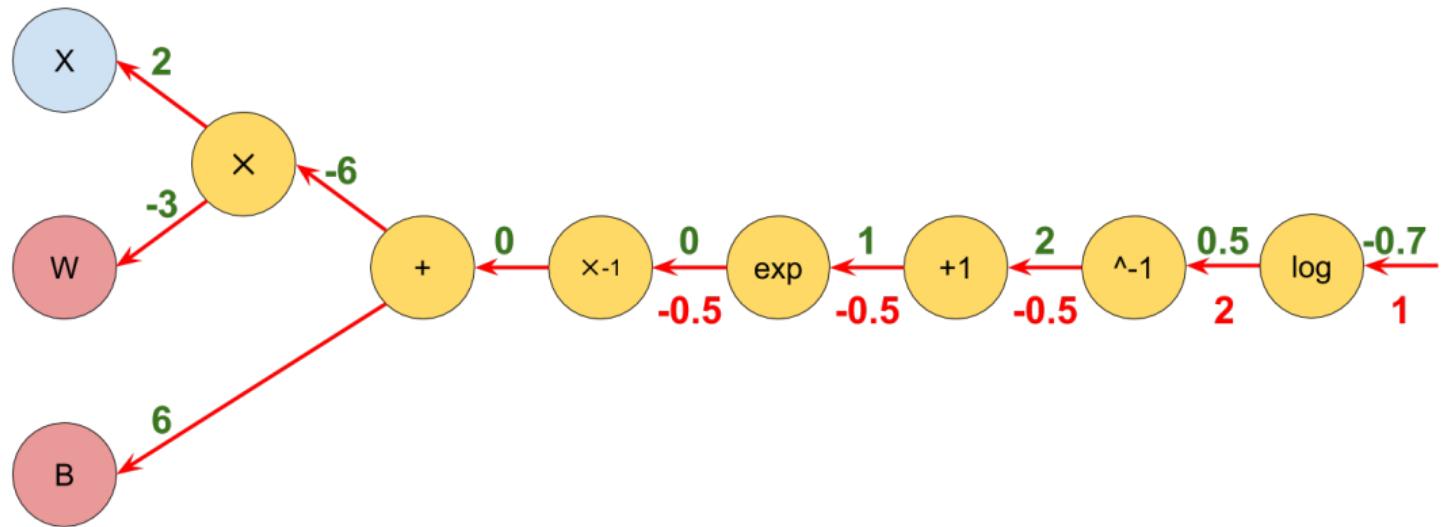
Backward



+1:

- ▶ receives $\frac{\partial L}{\partial (in+1)} = -0.5$
- ▶ computes $\frac{\partial (in+1)}{\partial in} = 1$
- ▶ returns $\frac{\partial L}{\partial in} = -0.5 * 1 = -0.5$

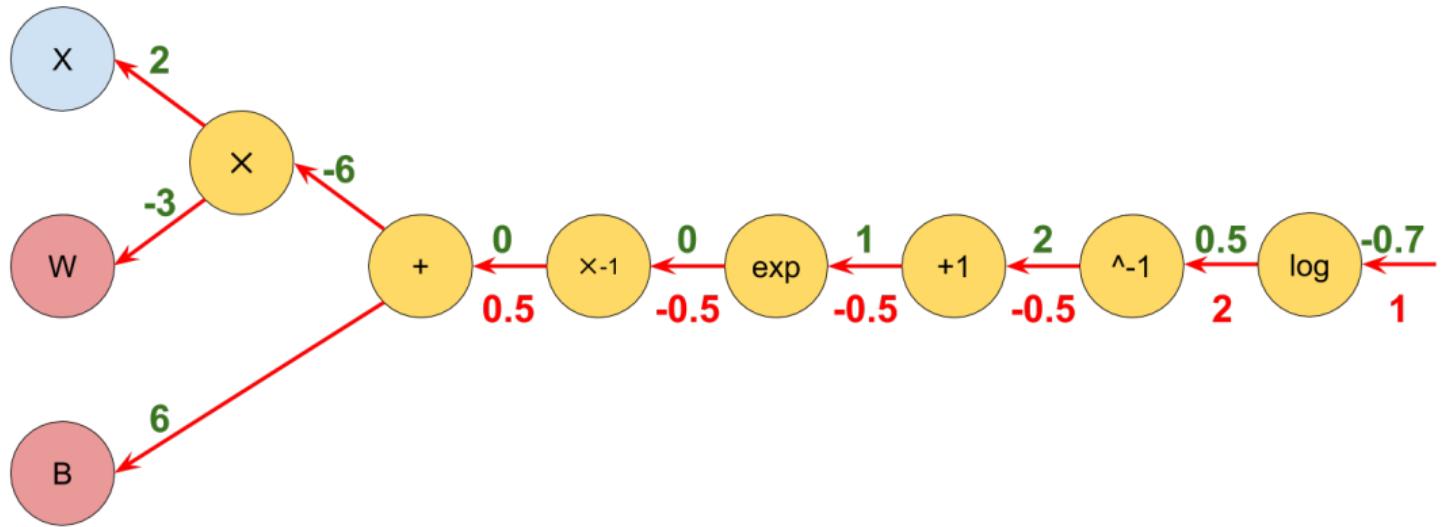
Backward



\exp :

- ▶ receives $\frac{\partial L}{\partial e^{in}} = -0.5$
- ▶ computes $\frac{\partial e^{in}}{\partial in} = e^{in} = 1$
- ▶ returns $\frac{\partial L}{\partial in} = -0.5 * 1 = -0.5$

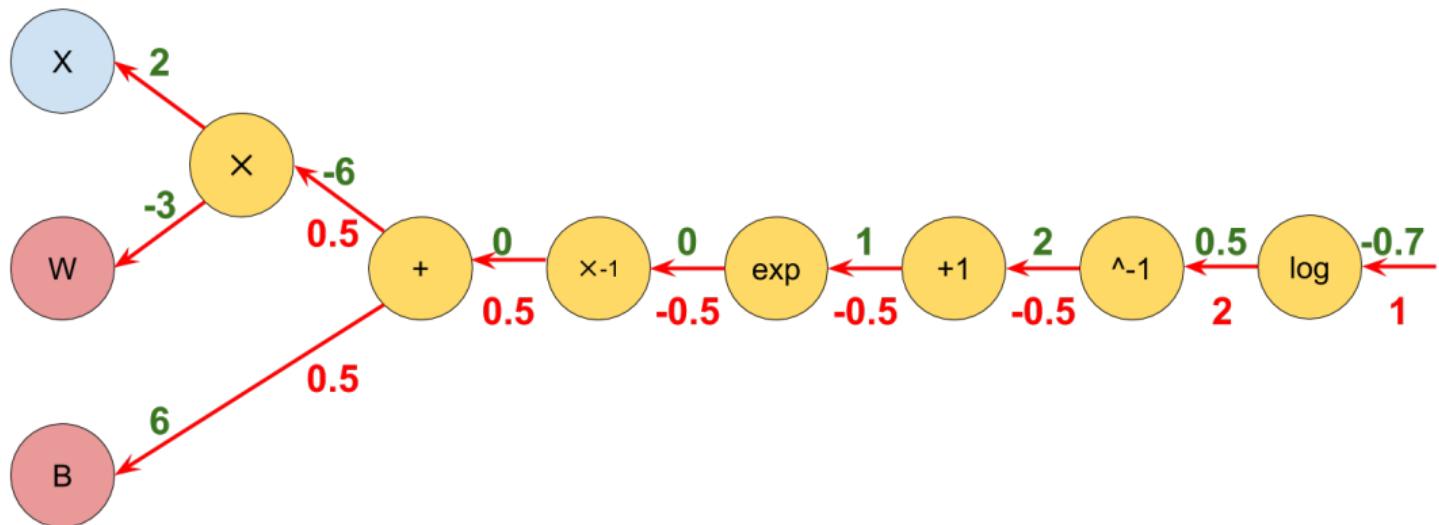
Backward



$\times - 1$:

- ▶ receives $\frac{\partial L}{\partial (-in)} = -0.5$
- ▶ computes $\frac{\partial (-in)}{\partial in} = -1$
- ▶ returns $\frac{\partial L}{\partial in} = -0.5 * -1 = 0.5$

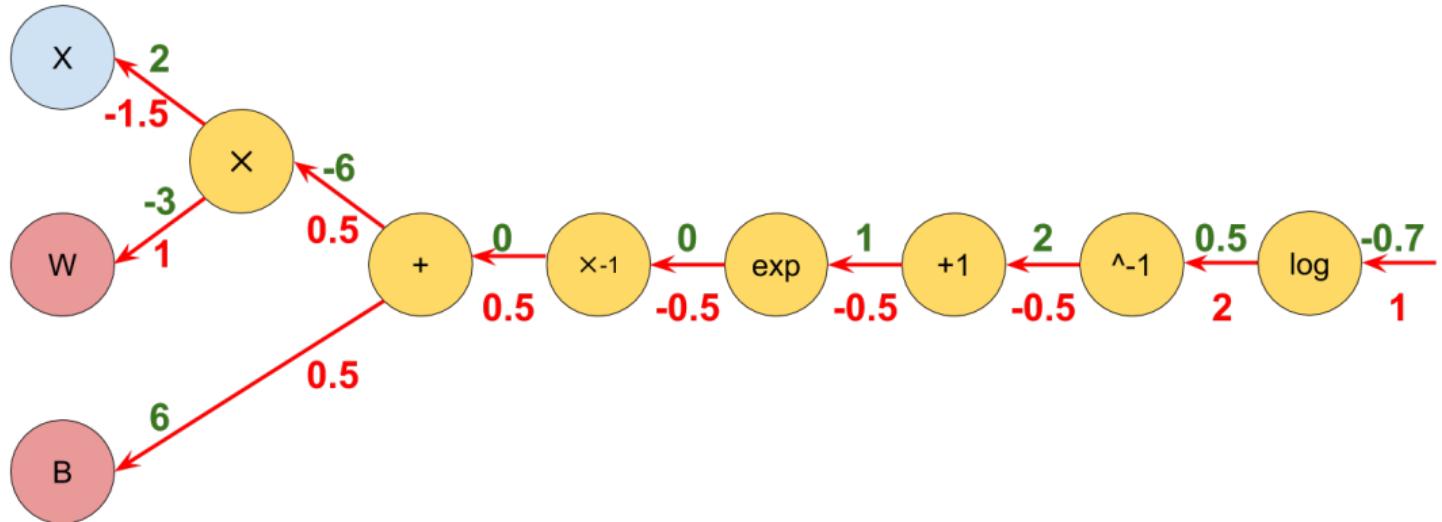
Backward



+

- ▶ receives $\frac{\partial L}{\partial (in_1 + in_2)} = 0.5$
- ▶ computes $(\frac{\partial (in_1 + in_2)}{\partial in_1}, \frac{\partial (in_1 + in_2)}{\partial in_2}) = (1, 1)$
- ▶ returns $(\frac{\partial L}{\partial in_1}, \frac{\partial L}{\partial in_2}) = (0.5, 0.5)$

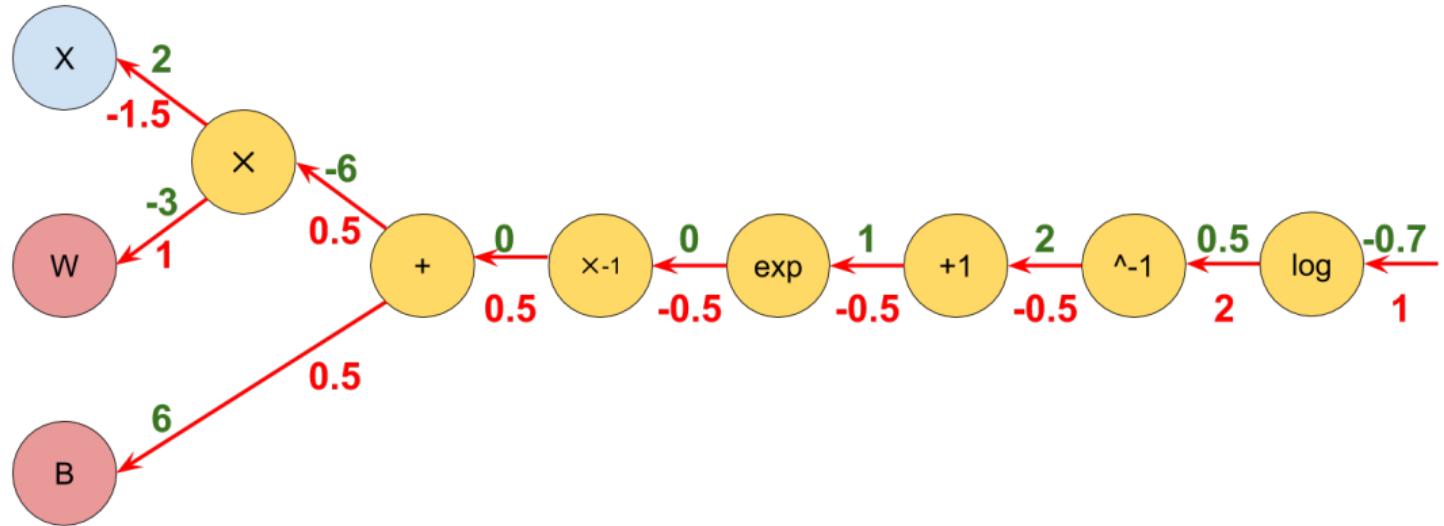
Backward



\times :

- ▶ receives $\frac{\partial L}{\partial (in_1 \times in_2)} = 0.5$
- ▶ computes $(\frac{\partial (in_1 \times in_2)}{\partial in_1}, \frac{\partial (in_1 \times in_2)}{\partial in_2}) = (in_2, in_1) = (-3, 2)$
- ▶ returns $(\frac{\partial L}{\partial in_1}, \frac{\partial L}{\partial in_2}) = 0.5 * (-3, 2) = (-1.5, 1)$

Backward



$$\nabla L = \left[\frac{\partial L}{\partial x}, \frac{\partial L}{\partial w}, \frac{\partial L}{\partial b} \right] = [-1.5, 1, 0.5]$$

Parameter updating

- ▶ Measure data fitting performance: loss function $L(D, \theta)$.

¹dsdeepdive.blogspot.com

Parameter updating

- ▶ Measure data fitting performance: loss function $L(D, \theta)$,
- ▶ Compute the gradient of the loss function with respect to weights,

¹dsdeepdive.blogspot.com

Parameter updating

- ▶ Measure data fitting performance: loss function $L(D, \theta)$,
- ▶ Compute the gradient of the loss function with respect to weights,
- ▶ Find incremental solutions that better explain the data.

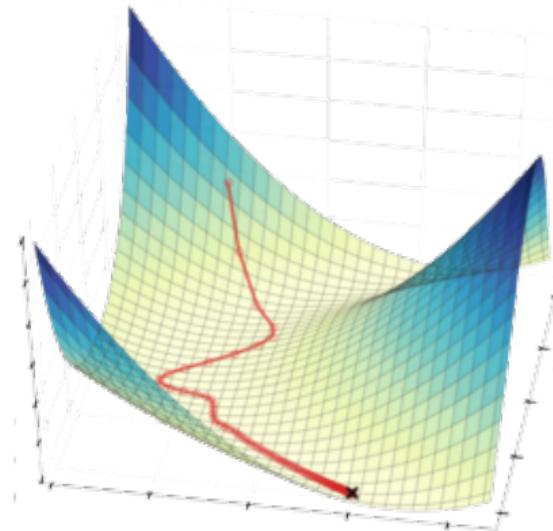
using:

$$\theta_{j+1} \leftarrow \theta_j - \alpha \nabla_{\theta_j} \mathcal{L}$$

sequentially find:

$$\theta_0, \theta_1, \dots, \theta_k$$

such that $L(D, \theta)$ decreases.



Sequential minimizing $L(D, \theta)$ ¹

¹dsdeepdive.blogspot.com

Outline

Recap++

Understanding Neural Networks

Optimizing Neural Networks

Computational Graphs

Forward pass

Backward pass

Wild beasts and how to tame them?

Universal Approximation Theorem



A fully connected network with one hidden layer followed by a non-linear function **can approximate any function** from one finite-dimensional space to another in the limit of the **width** of the hidden layer (**Hornik et al., 1990**).

Universal Approximation Theorem

A fully connected network with one hidden layer followed by a non-linear function **can approximate any function** from one finite-dimensional space to another in the limit of the **width** of the hidden layer ([Hornik et al., 1990](#)).

Potentially the hidden layer can get exponentially large. Eg.: no of possible binary functions on vectors $v \in \{0, 1\}^n$ is 2^{2^n} , requiring $O(2^n)$ units, each responding to a single input.

UAT and depth

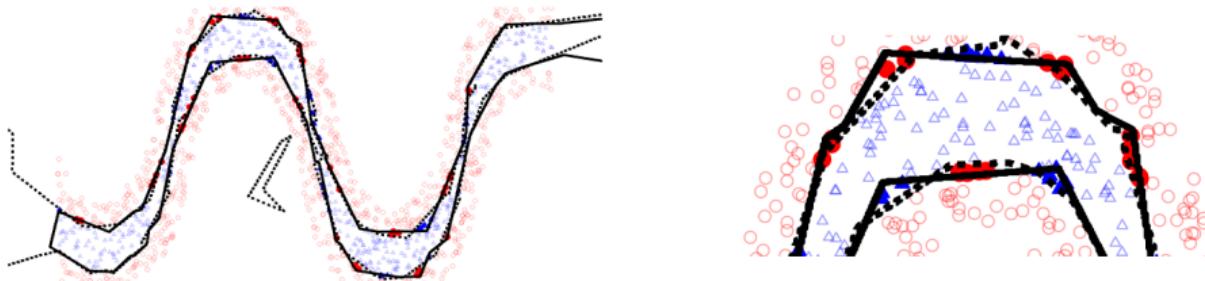
- ▶ How does **depth** relate to **UAT**?

UAT and depth

- ▶ How does **depth** relate to **UAT**?
- ▶ There exist families of functions which can be approximated efficiently (in numbers of parameters) by **deep** networks.

UAT and depth

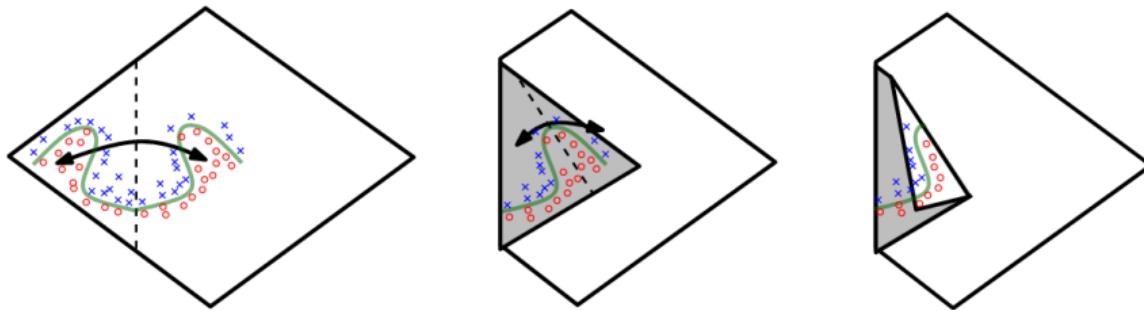
- ▶ How does **depth** relate to **UAT**?
- ▶ There exist families of functions which can be approximated efficiently (in numbers of parameters) by **deep** networks.
- ▶ Montufar et al. 2014:



A network with ReLU units learns piece-wise linear functions at each layer and deep networks reuse these computations exponentially more often than shallow networks. Solid line is a 20 units, one layer NN and dotted line is a 2 layers, 10 units each NN.

UAT and depth

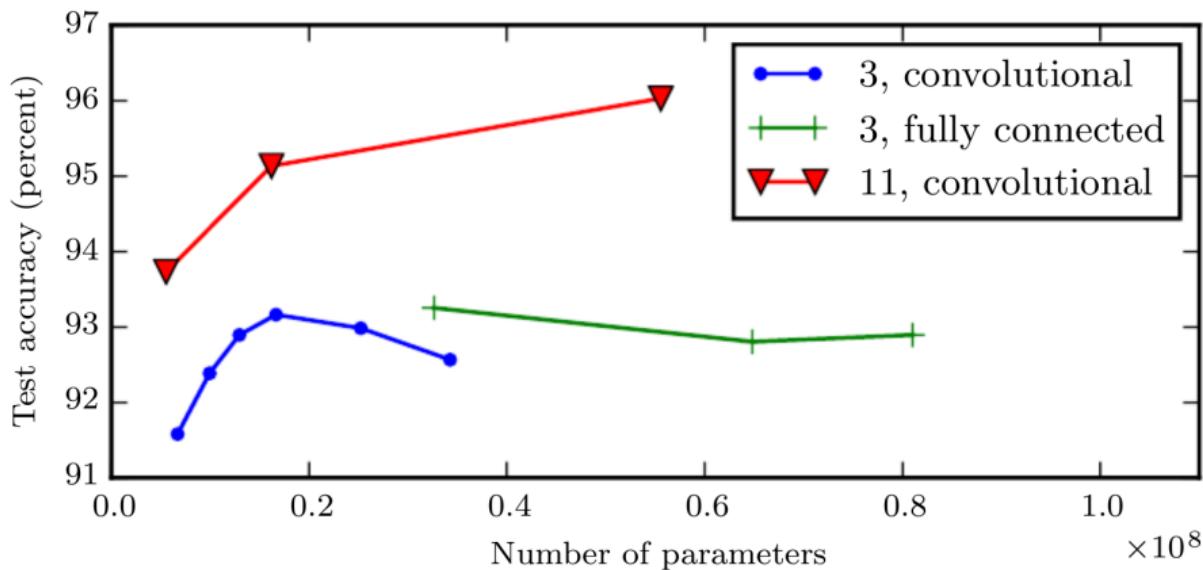
- ▶ How does **depth** relate to **UAT**?
- ▶ There exist families of functions which can be approximated efficiently (in numbers of parameters) by **deep** networks.
- ▶ Montufar et al. 2014:



A network with ReLU units creates mirror images of the function computed at the input of some hidden unit resulting in increasingly expressive networks

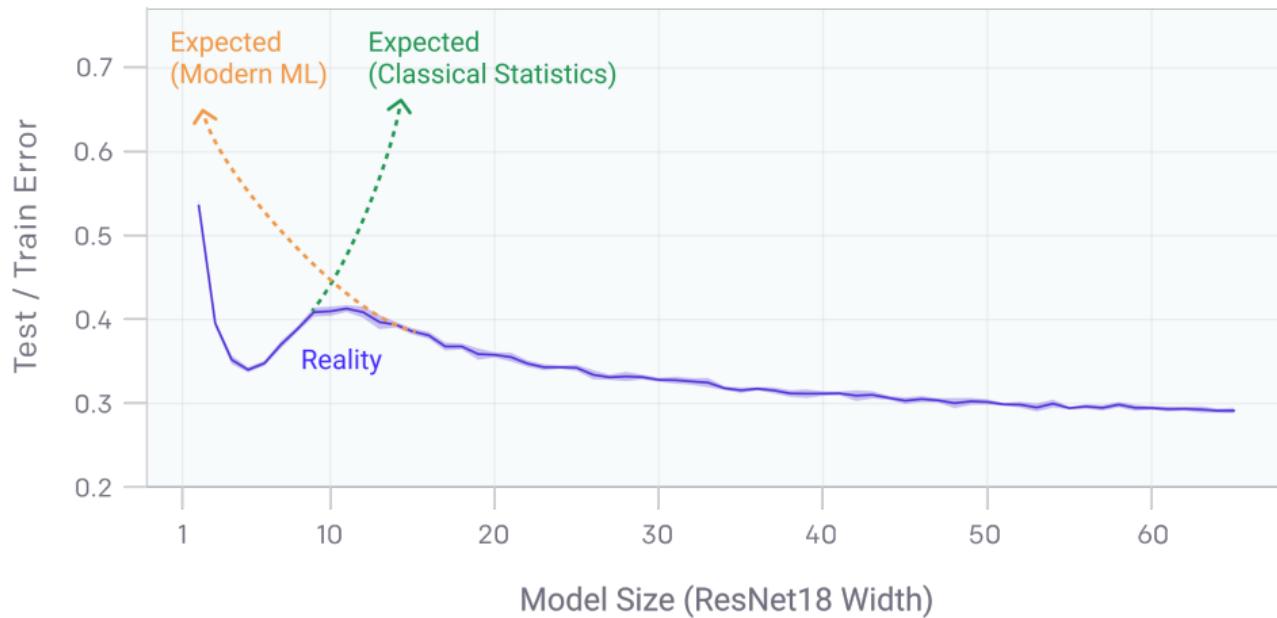
UAT and depth

- ▶ How does **depth** relate to **UAT**?
- ▶ There exist families of functions which can be approximated efficiently (in numbers of parameters) by **deep** networks.
- ▶ Goodfellow et al. 2014:



Generalization correlates with depth, not number of parameters.

The new generalization theory



Is the classic generalization theory still valid?