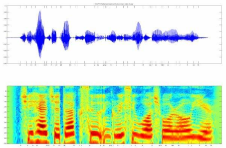
MADMO
Introduction to …
Deep Learning

Taras Khakhulin
Deep Learning Engineer Samsung AI Center
Skoltech and MIPT Master Student

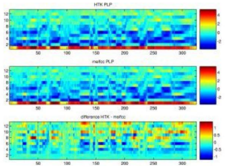
t.khakhulin@gmail.com
https://github.com/khakhulin/
https://twitter.com/t_khakhulin

Real world problems

Audio Features

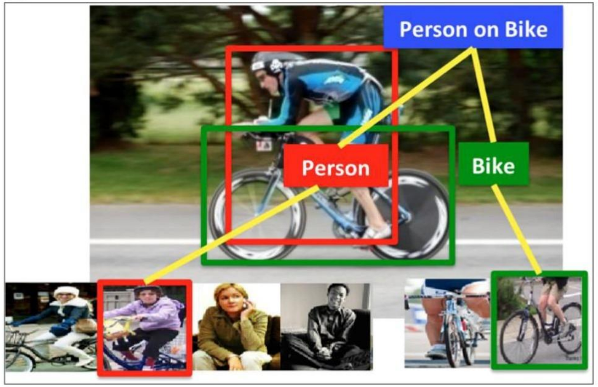


Spectrogram



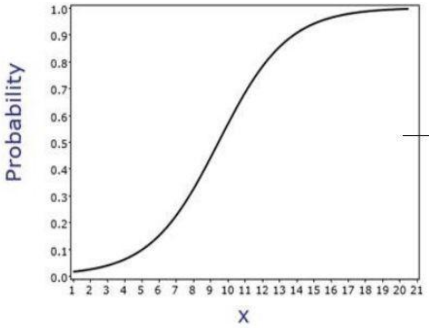
MFCC

- Object detection
- Action classification
- Image captioning
- …



"man in black shirt is playing guitar."

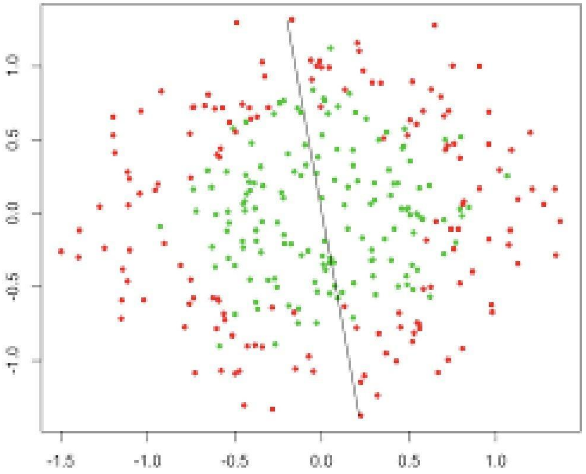
Logistic regression

X → Wx +b →  → P(y)

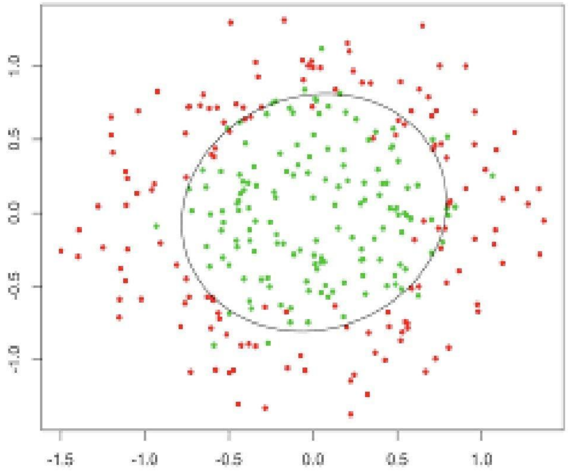
$$P(y|x) = \sigma(w \cdot x + b)$$

$$L = -\sum_i y_i \log P(y|x_i) + (1 - y_i) \log(1 - P(y|x_i))$$

Problem: nonlinear dependencies



What we have

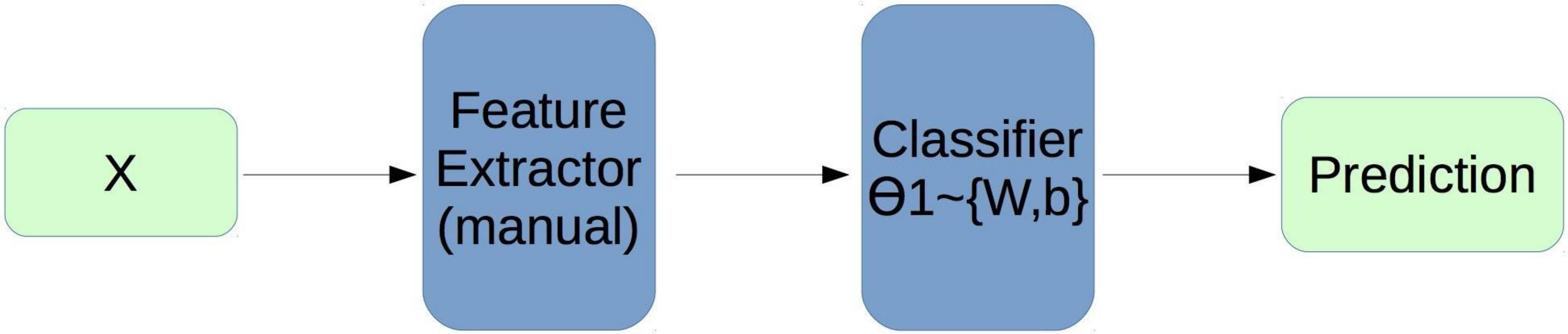


What we want

Logistic regression (generally, linear model) need feature engineering to show good results.

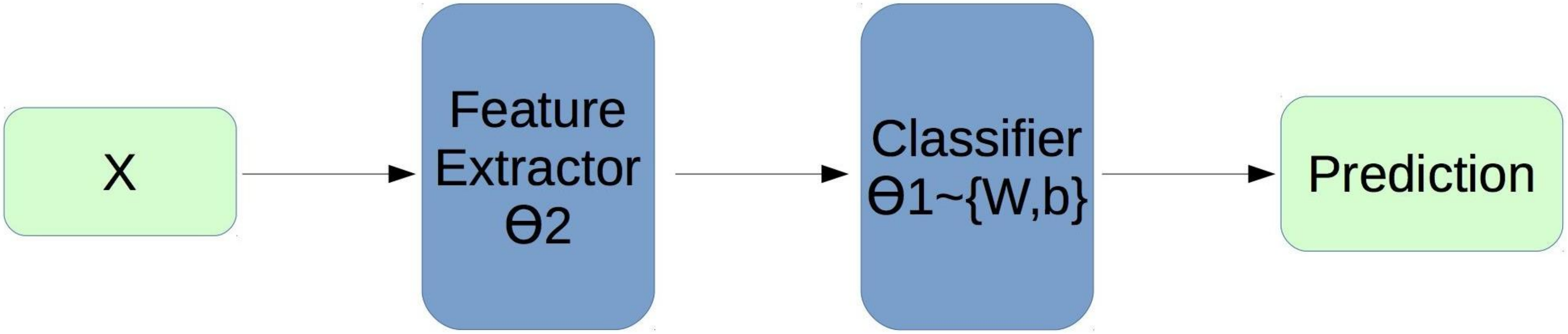
And feature engineering is an *art*.

Classic pipeline



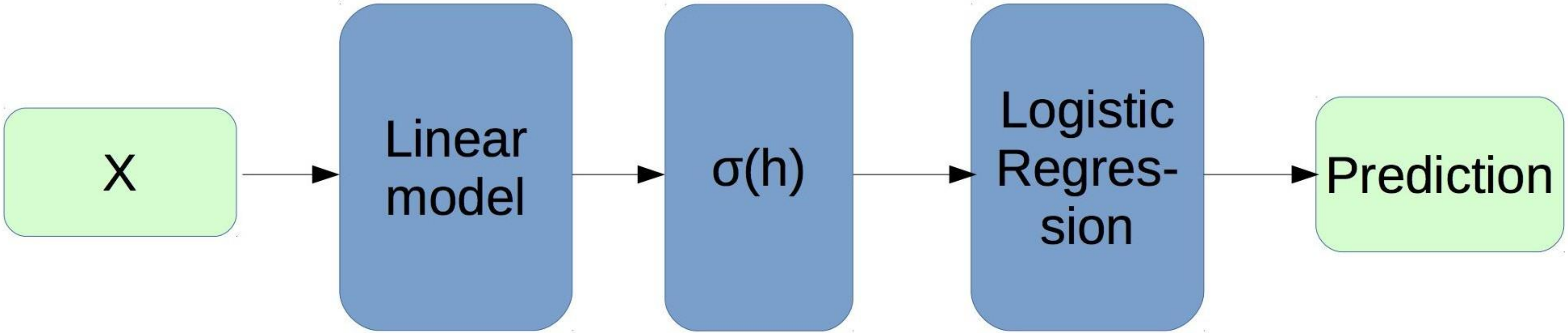
Handcrafted features, generated by experts.

NN pipeline



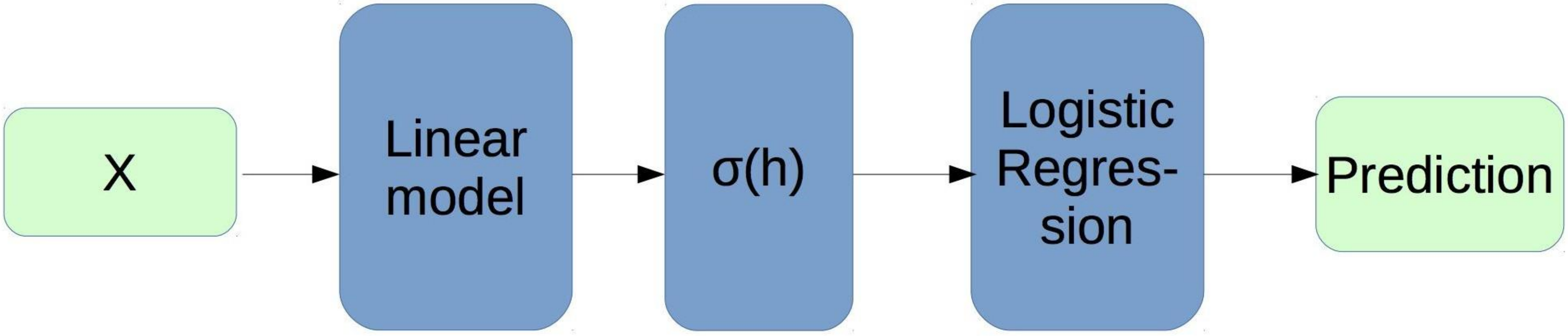
Automatically extracted features.

NN pipeline: example



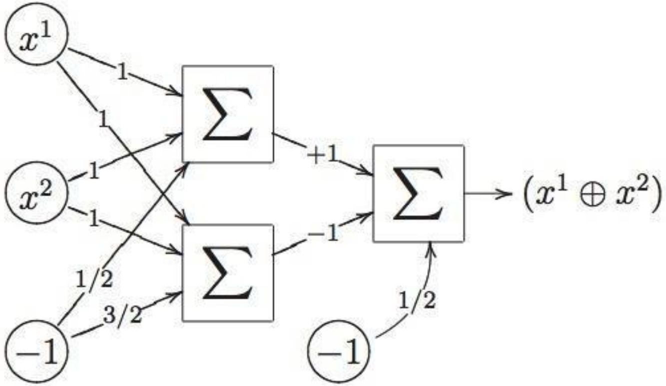
E.g. two logistic regressions one after another.

NN pipeline: example



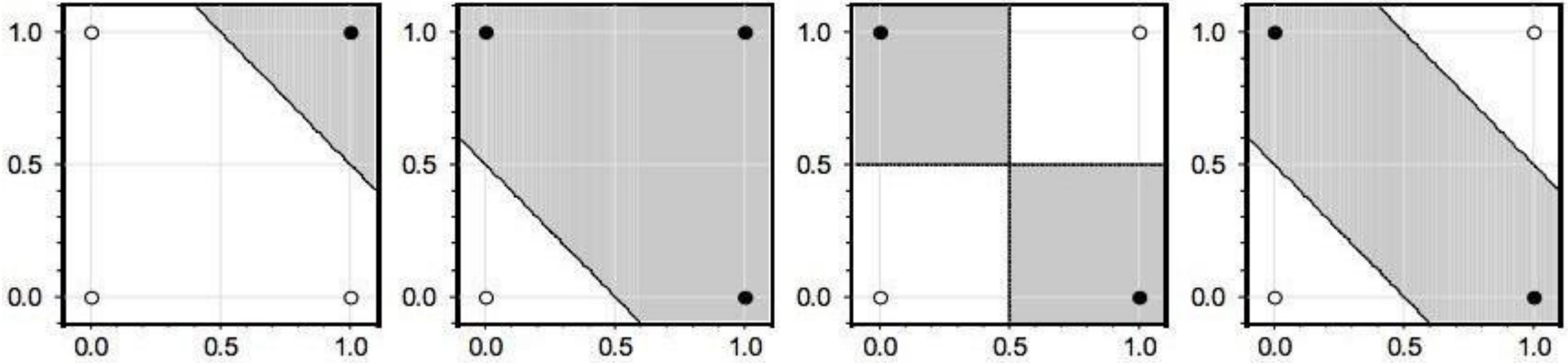
Actually, it's a neural network.

XOR problem



This 2-layer NN (on the left) implements XOR with only x1 and x2 features.

1-layer NN also can succeed, but only with extra feature x1*x2.



AND OR XOR(with x1*x2) XOR

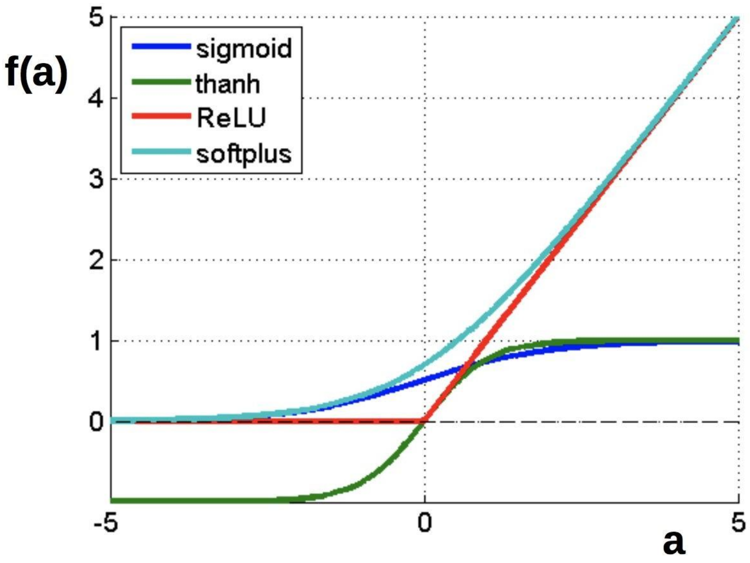
Activation functions: nonlinearities

$$f(a) = \frac{1}{1 + e^{a}}$$

$$f(a) = \tanh(a)$$

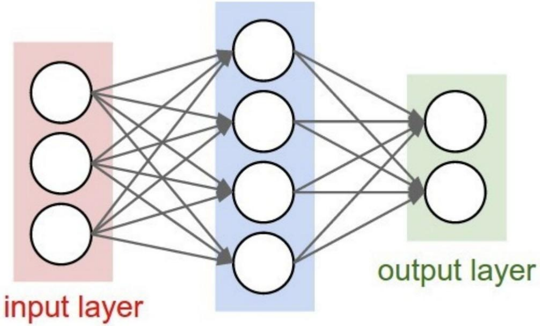
$$f(a) = \max(0, a)$$

$$f(a) = \log(1 + e^{a})$$



Some generally accepted terms

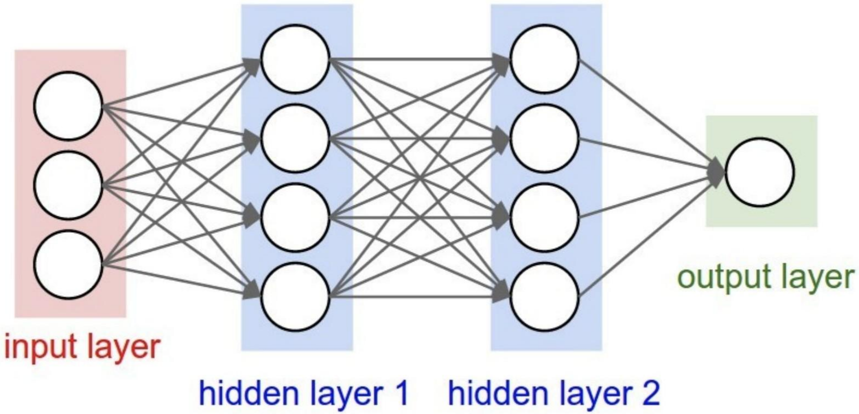
- Layer – a building block for NNs :
    - Dense layer: $f(x) = Wx+b$
    - Nonlinearity layer: $f(x) = \sigma(x)$
    - Input layer, output layer
    - A few more we will cover later
- Activation function – function applied to layer output
    - Sigmoid
    - tanh
    - ReLU
    - Any other function to get nonlinear intermediate signal in NN
- Backpropagation – a fancy word for “chain rule”



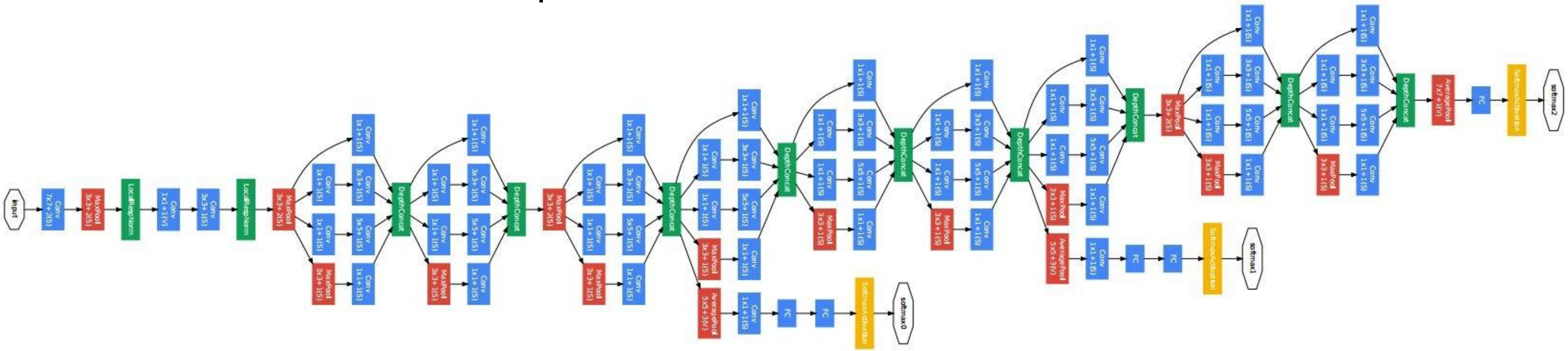
Dense layer
with a nonlinearity

“Train it via backprop!”

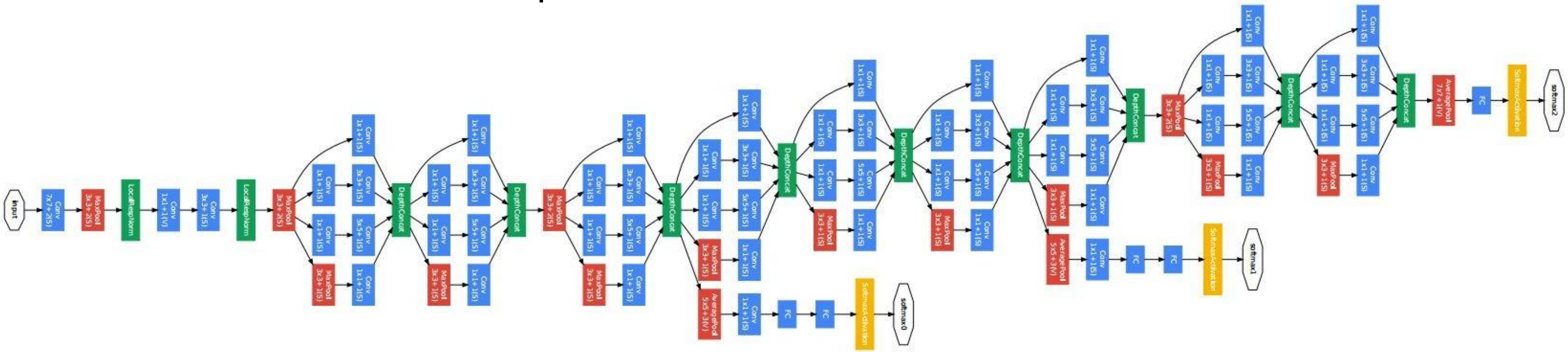
Actually, it can be deeper



Much
deeper...



Much
deeper...

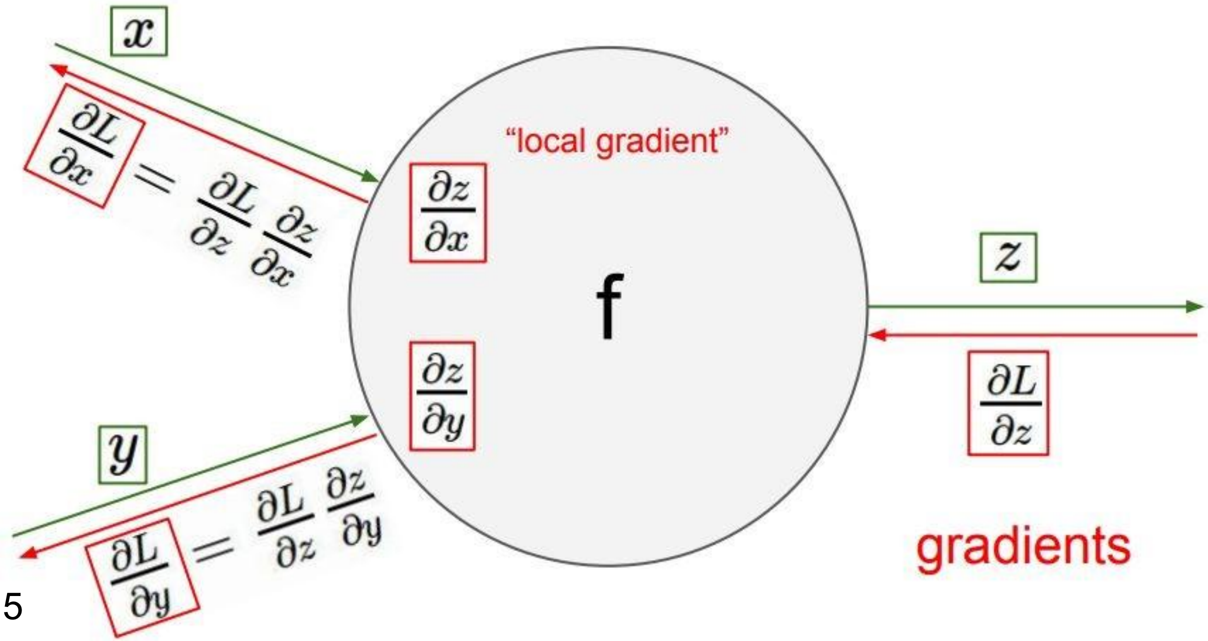


How to train it?

Backpropagation and chain rule

Chain rule is just simple math: $$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

Backprop is just way to use it in NN training.

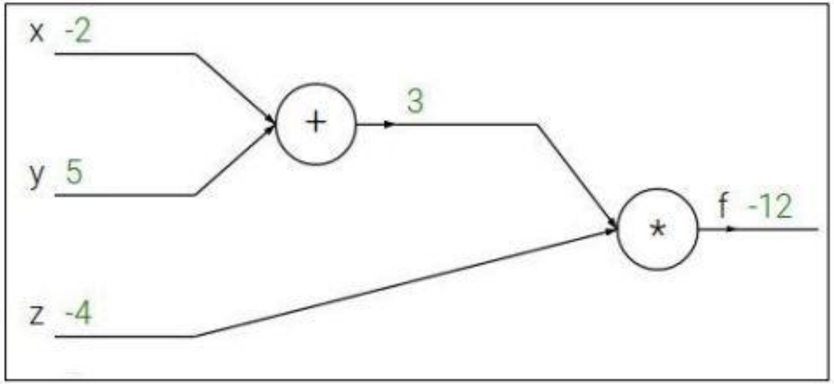


source: http://cs231n.github.io

Backpropagation example

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

e.g. x = -2, y = 5, z = -4



source: http://cs231n.github.io

Backpropagation example

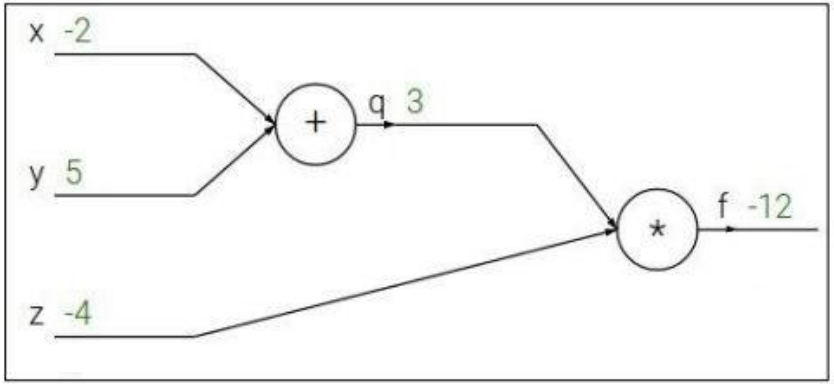
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



source: http://cs231n.github.io

Backpropagation example

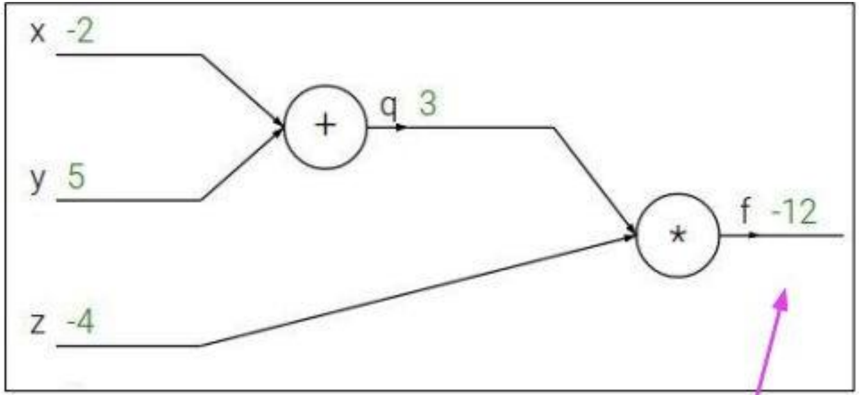
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



$$\frac{\partial f}{\partial f}$$

source: http://cs231n.github.io

Backpropagation example

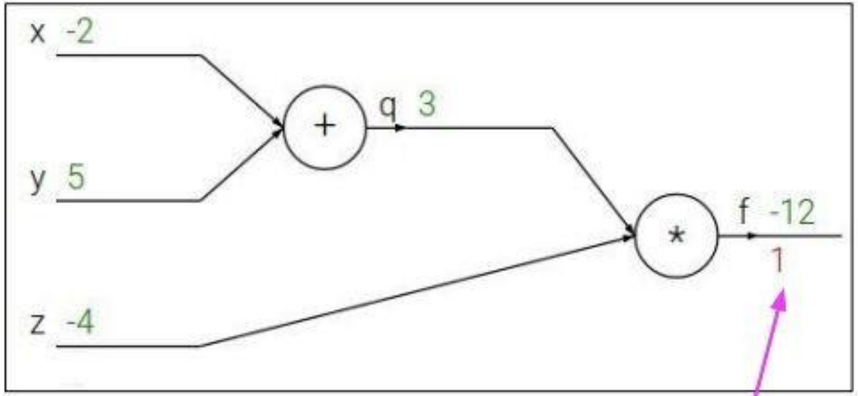
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



$\frac{\partial f}{\partial f}$

source: http://cs231n.github.io

Backpropagation example

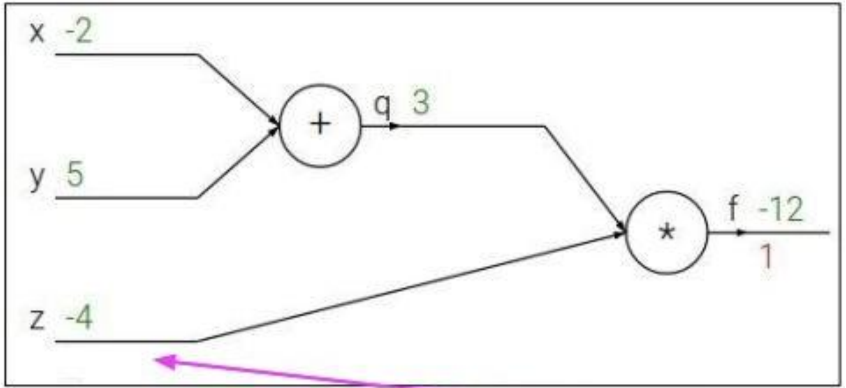
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



$$\frac{\partial f}{\partial z}$$

source: http://cs231n.github.io

Backpropagation example

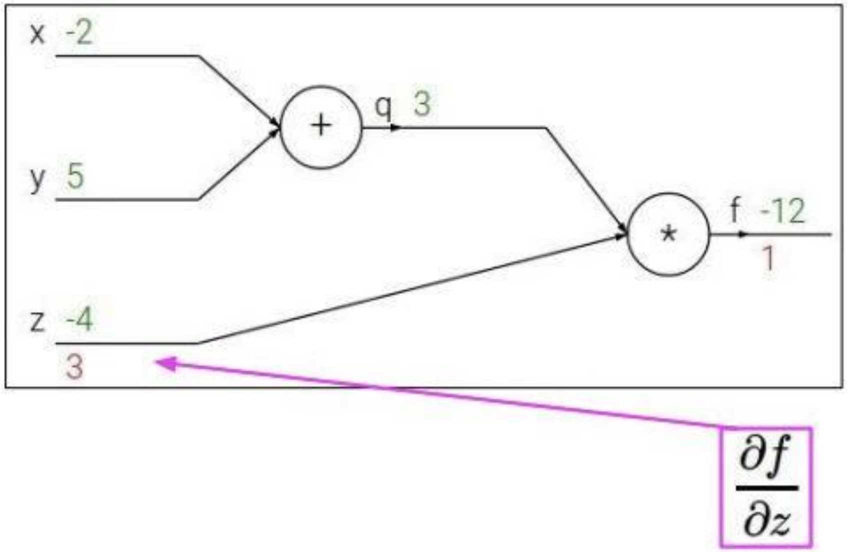
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



source: http://cs231n.github.io

Backpropagation example

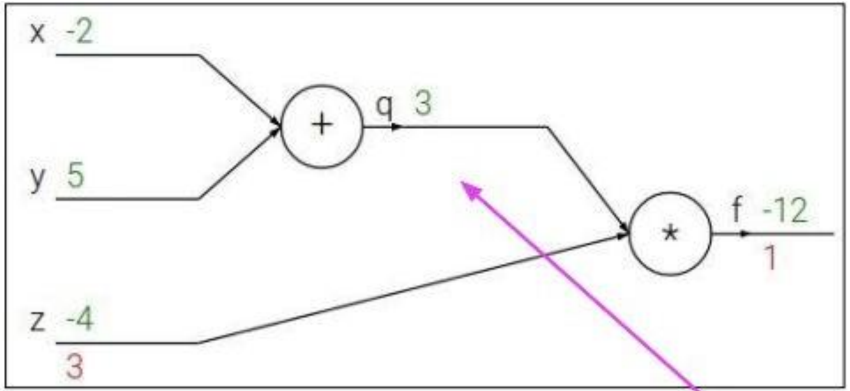
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



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

source: http://cs231n.github.io

Backpropagation example

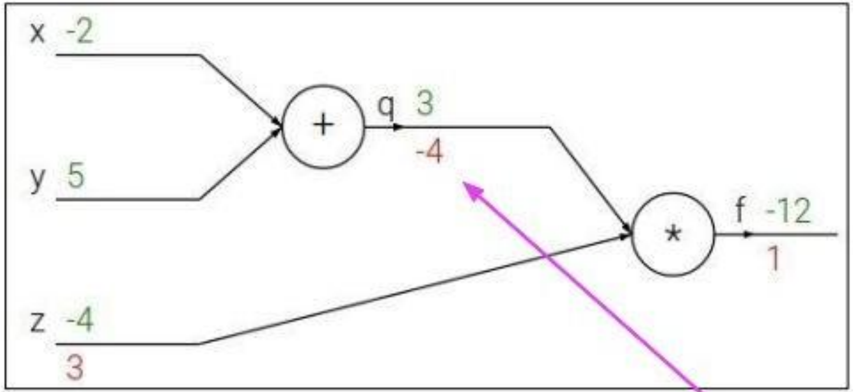
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



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

source: http://cs231n.github.io

Backpropagation example

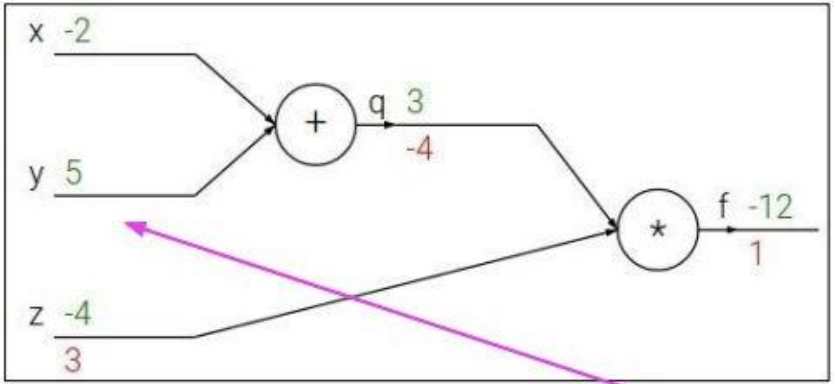
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



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

source: http://cs231n.github.io

Backpropagation example

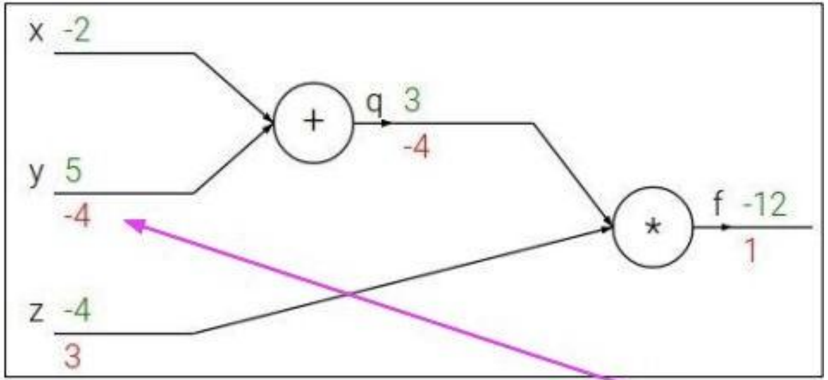
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



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

Chain rule:

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

source: http://cs231n.github.io

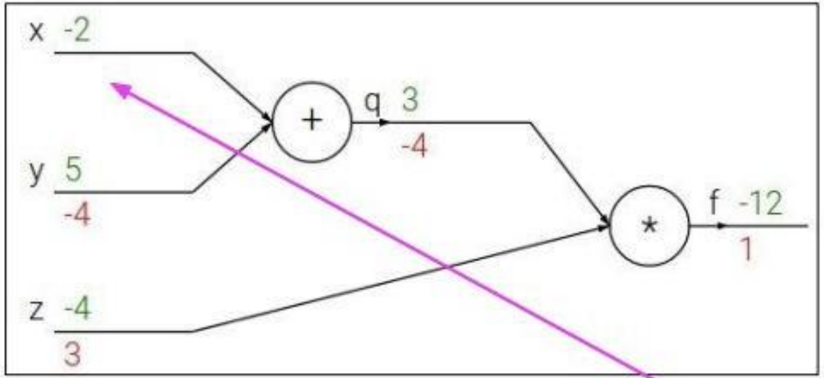
Backpropagation example

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

e.g. x = -2, y = 5, z = -4

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

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



$$\frac{\partial f}{\partial x}$$

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

source: http://cs231n.github.io

Backpropagation example

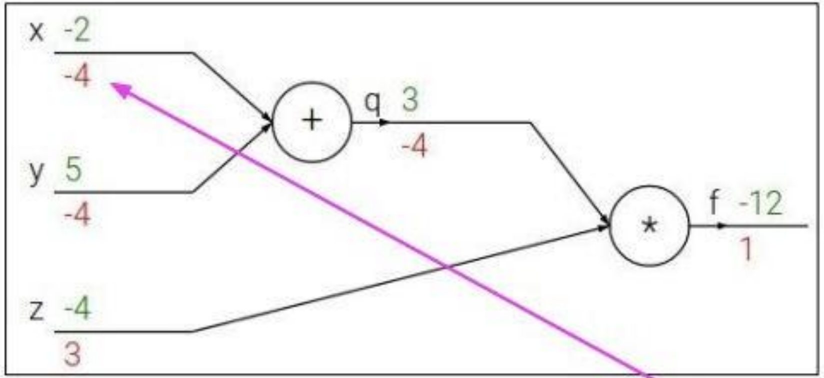
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

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

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

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



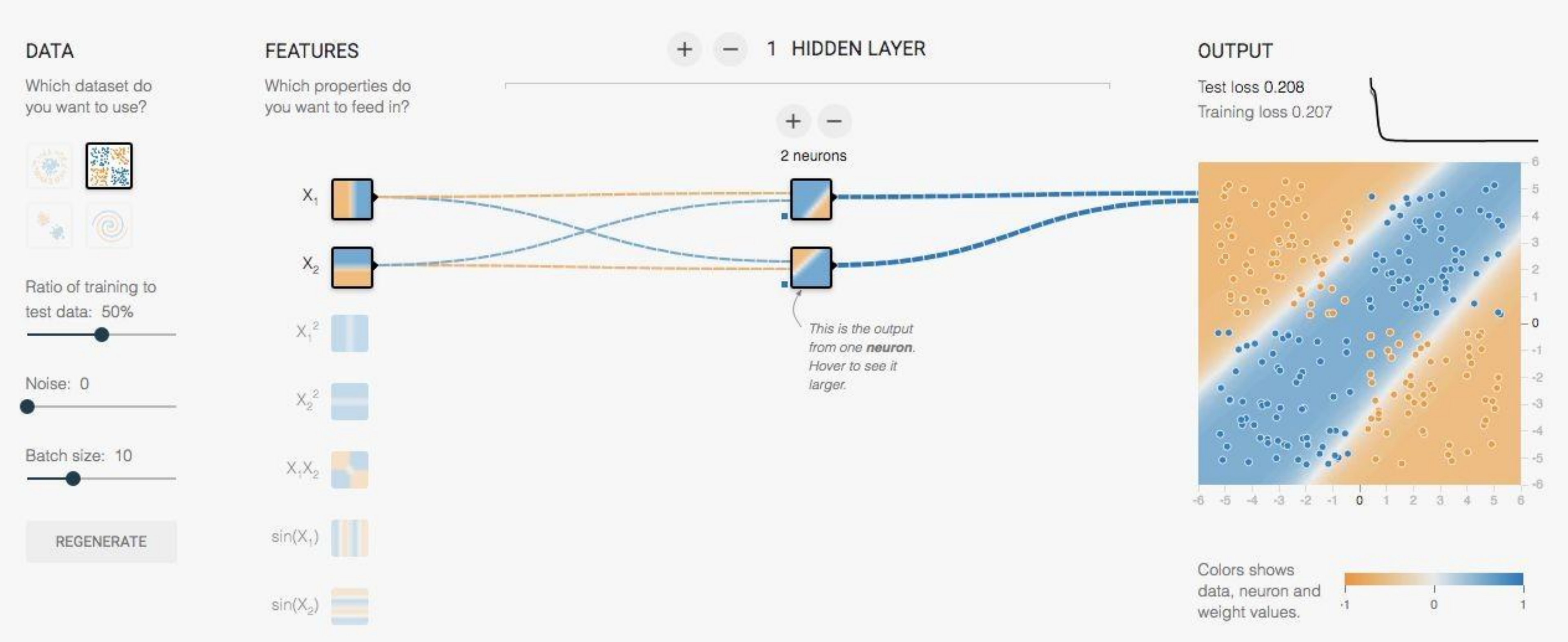
$$\frac{\partial f}{\partial x}$$

Chain rule:

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

source: http://cs231n.github.io

Practice time: interactive playground



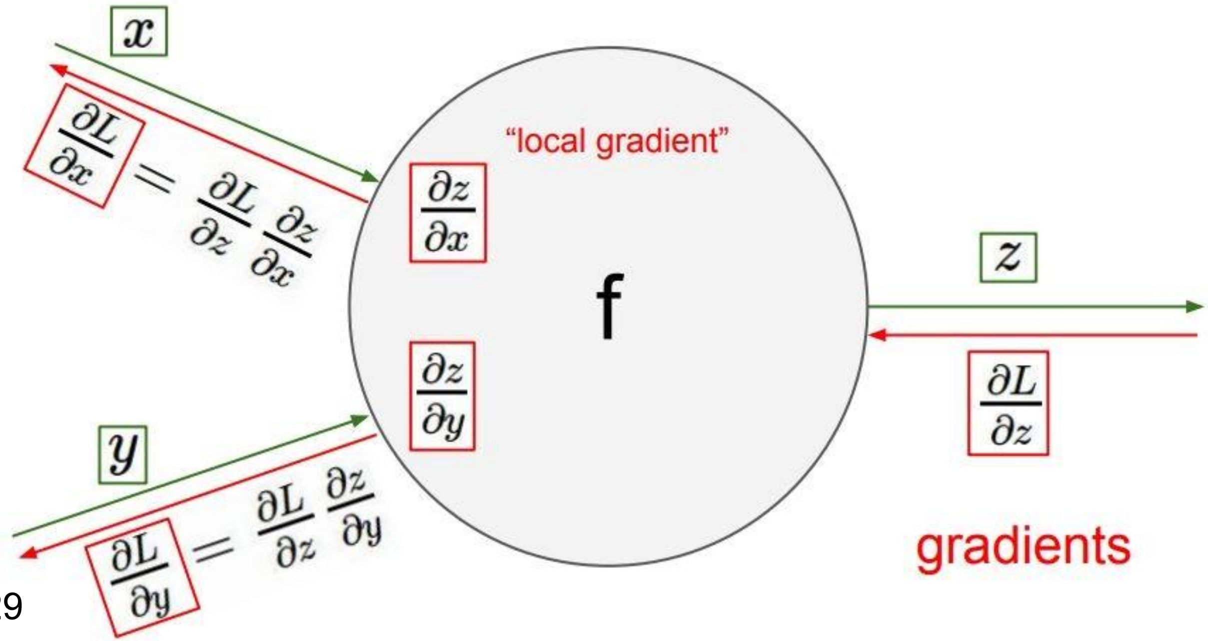
https://playground.tensorflow.org/

0

Backpropagation and chain rule

Chain rule is just simple math: $$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

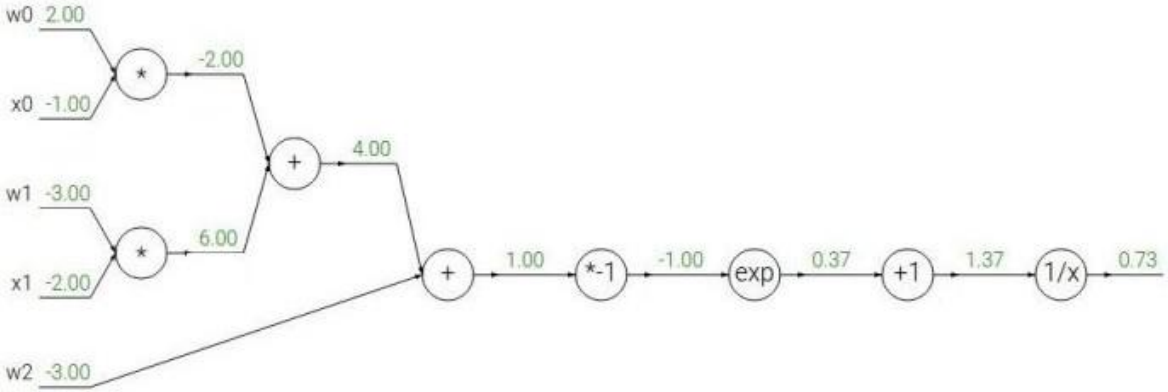
Backprop is just way to use it in NN training.



source: http://cs231n.github.io

Backpropagation example

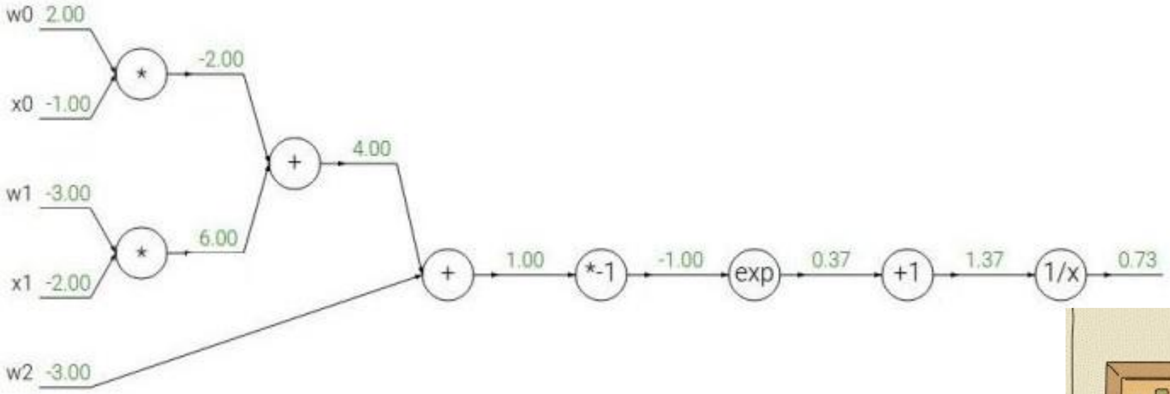
Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

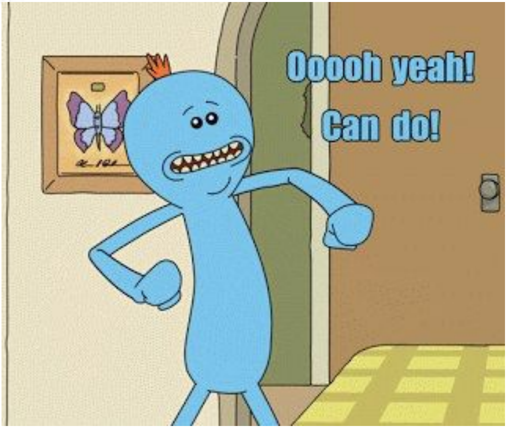


source: http://cs231n.github.io

Backpropagation example

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

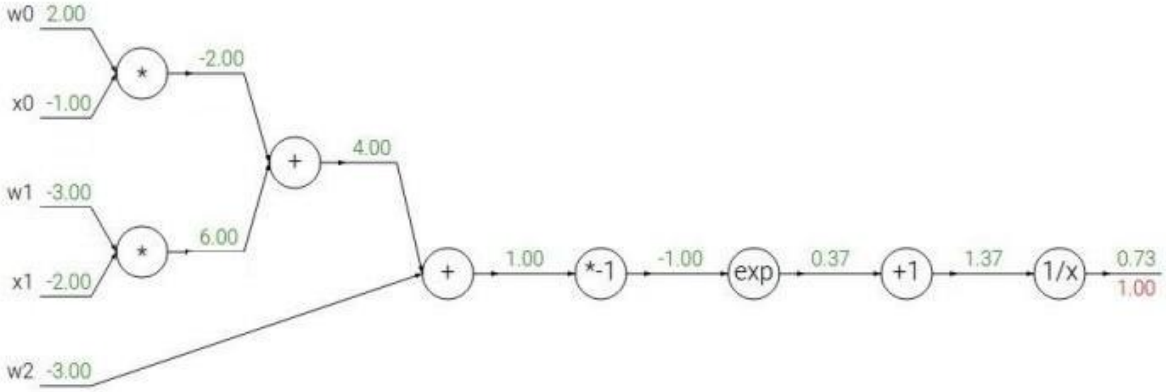




source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

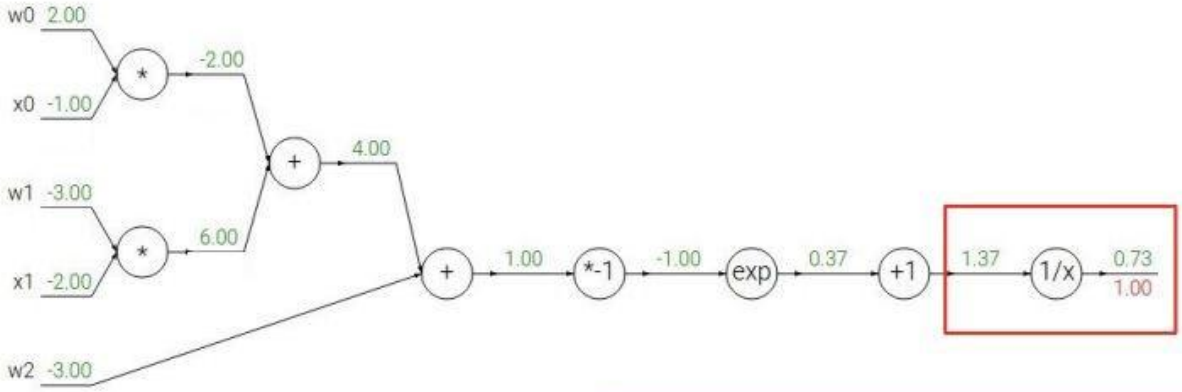


$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$		$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$	
$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$		$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$	

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

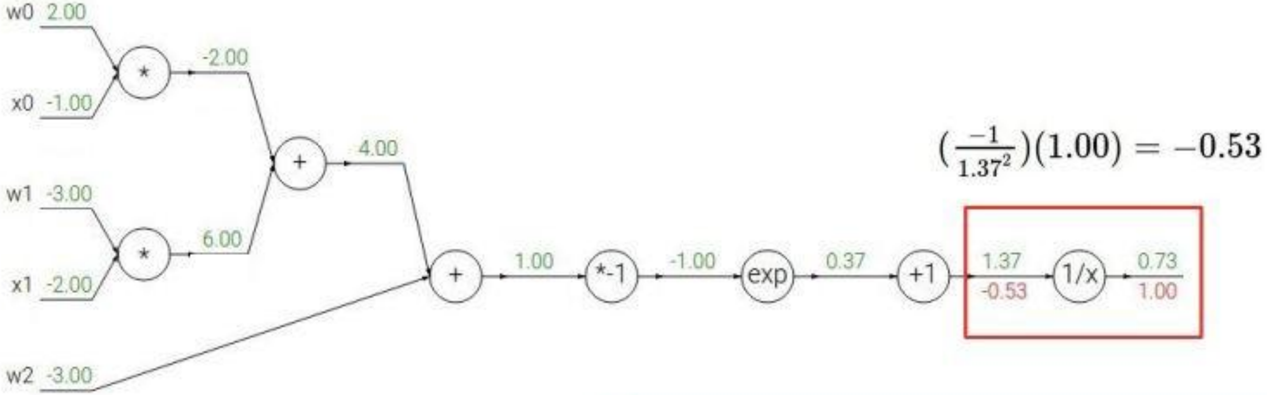
$\boxed{f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2}$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

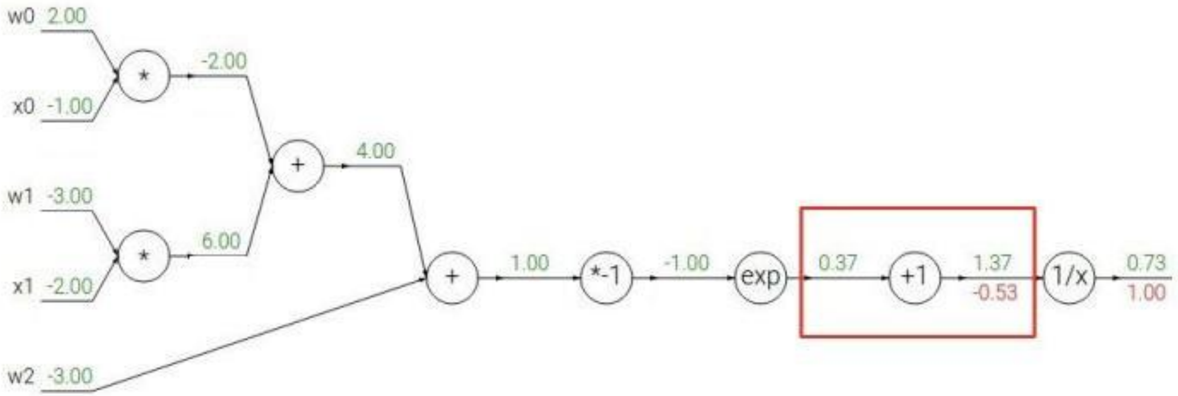


$f(x) = e^x$	$\rightarrow$	$\frac{df}{dx} = e^x$	
$f_a(x) = ax$	$\rightarrow$	$\frac{df}{dx} = a$	
$\boxed{f(x) = \frac{1}{x}}$	$\rightarrow$	$\boxed{\frac{df}{dx} = -1/x^2}$	
$f_c(x) = c + x$	$\rightarrow$	$\frac{df}{dx} = 1$	

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

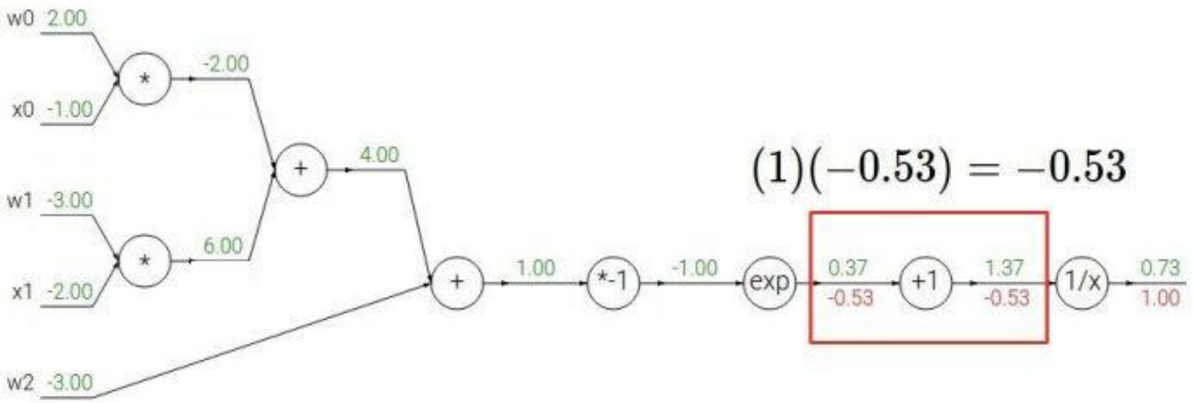
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

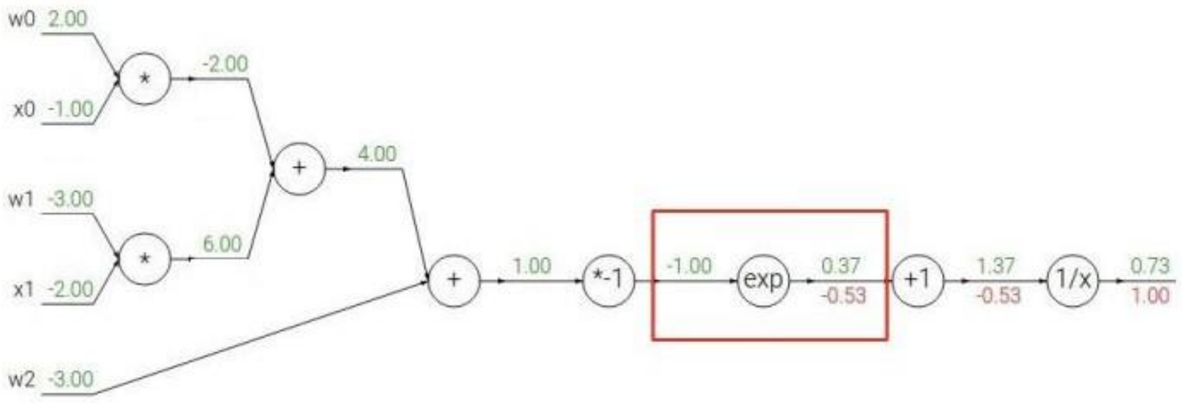
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \dfrac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$$\boxed{f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x}$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

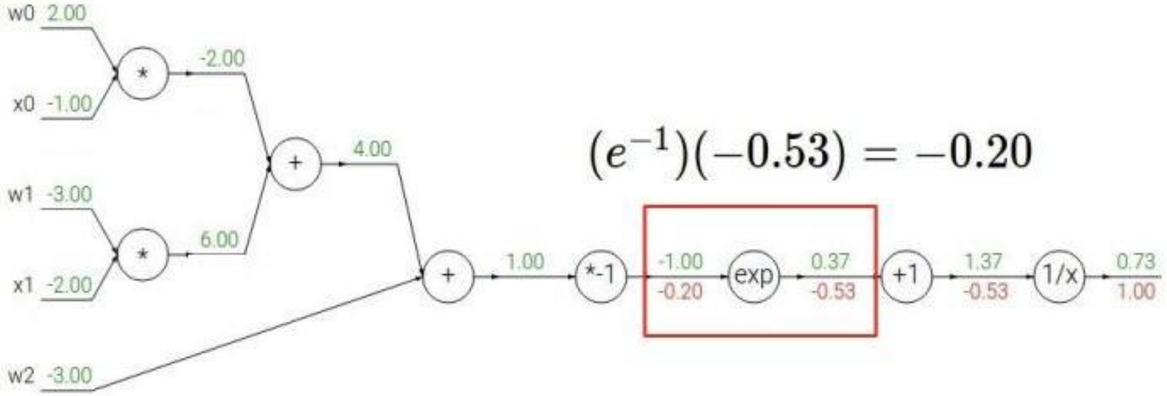
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$(e^{-1})(-0.53) = -0.20$

$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

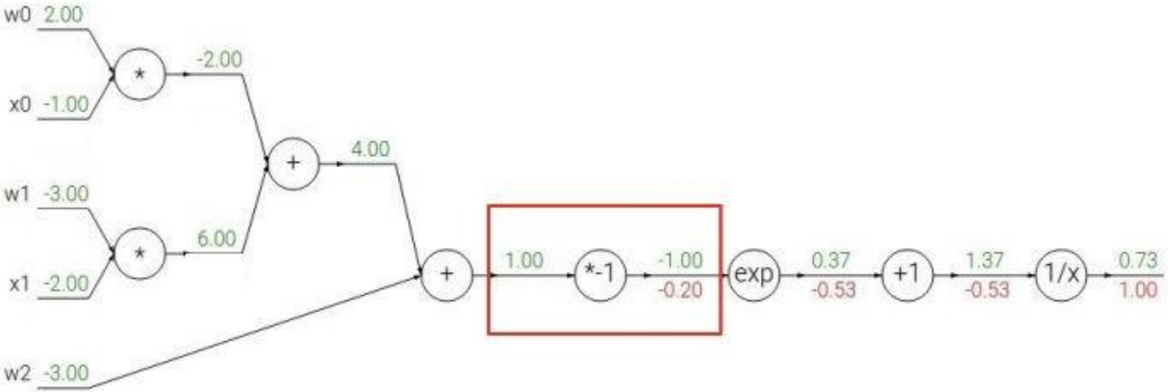
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

source: http://cs231n.github.io

Backpropagation example

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

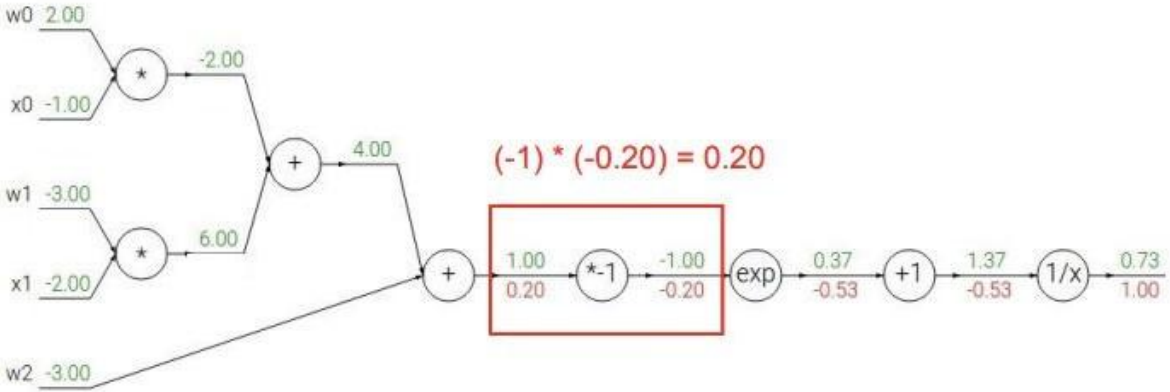
$$\boxed{f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a}$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

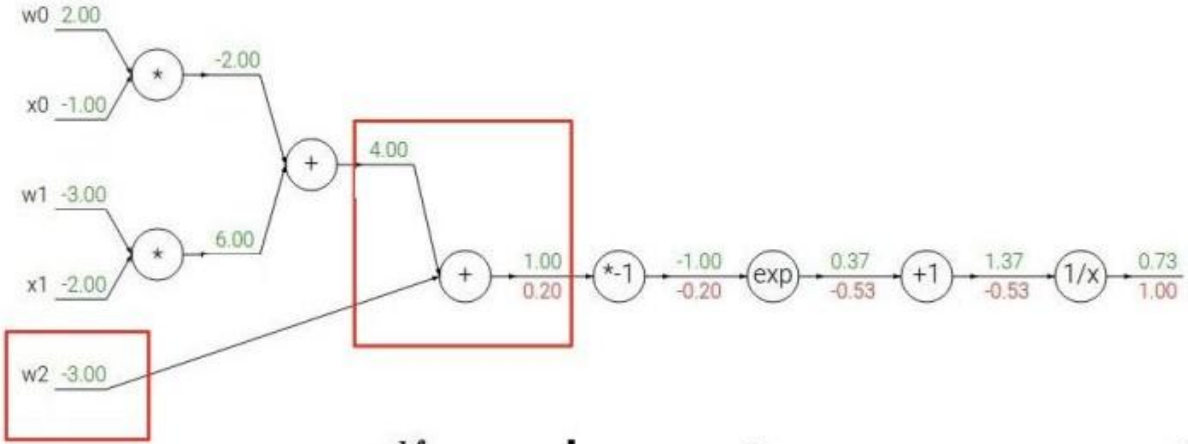
$\boxed{f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a}$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \dfrac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



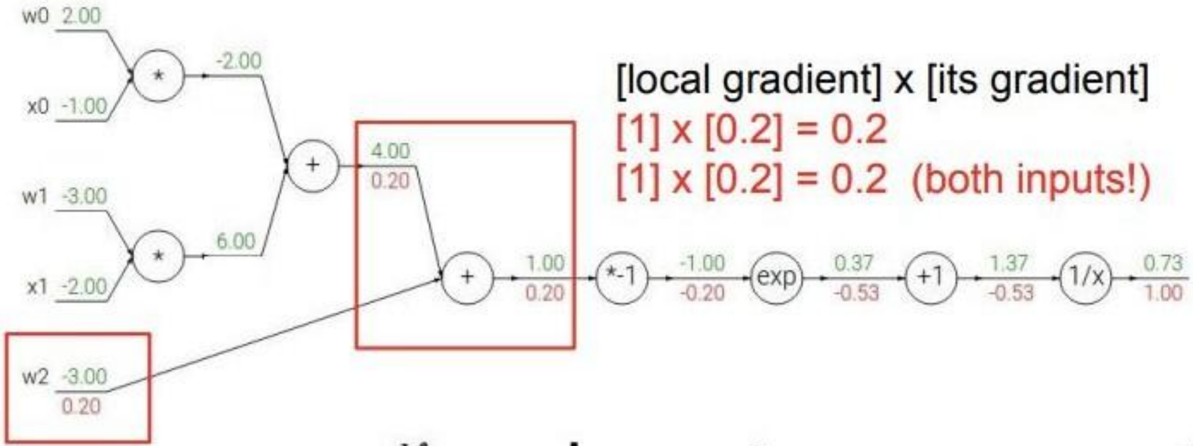
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \qquad\qquad f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \qquad\qquad f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

source: http://cs231n.github.io

Backpropagation example

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



[local gradient] x [its gradient]

[1] x [0.2] = 0.2

[1] x [0.2] = 0.2 (both inputs!)

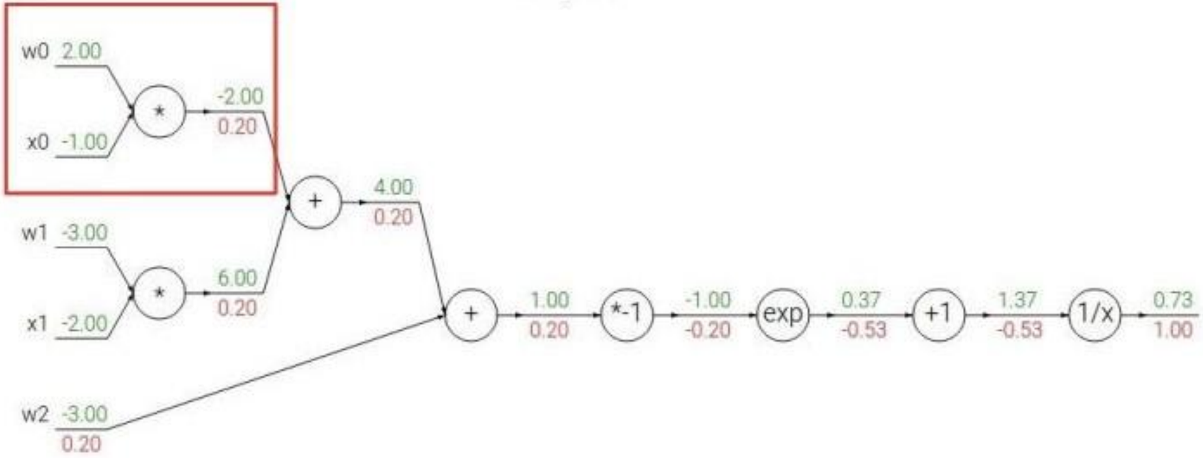
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \qquad\qquad f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \qquad\qquad f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

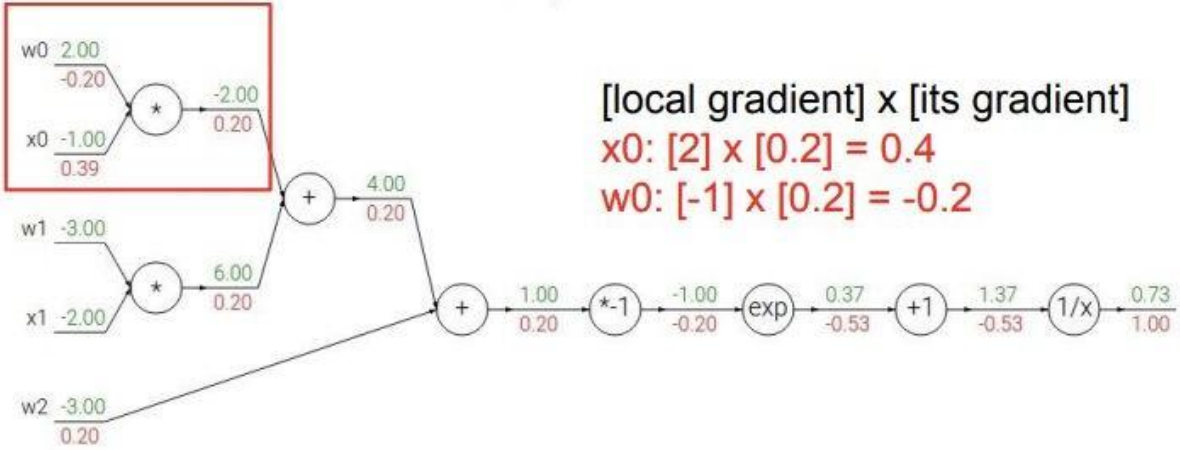
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

source: http://cs231n.github.io

Backpropagation example

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



[local gradient] x [its gradient]
x0: [2] x [0.2] = 0.4
w0: [-1] x [0.2] = -0.2

$f(x) = e^x$	$\rightarrow$	$\frac{df}{dx} = e^x$	$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	$\rightarrow$	$\frac{df}{dx} = a$	$f_c(x) = c + x$	$\rightarrow$	$\frac{df}{dx} = 1$

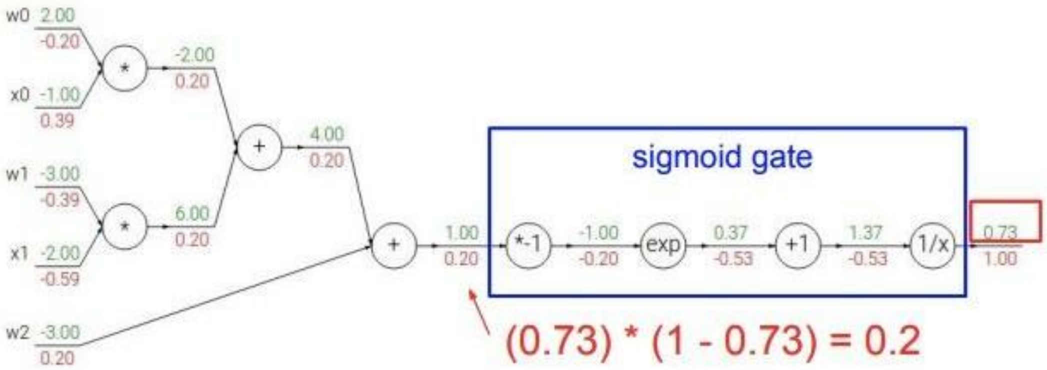
source: http://cs231n.github.io

Backpropagation example

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

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

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\,\sigma(x)$$

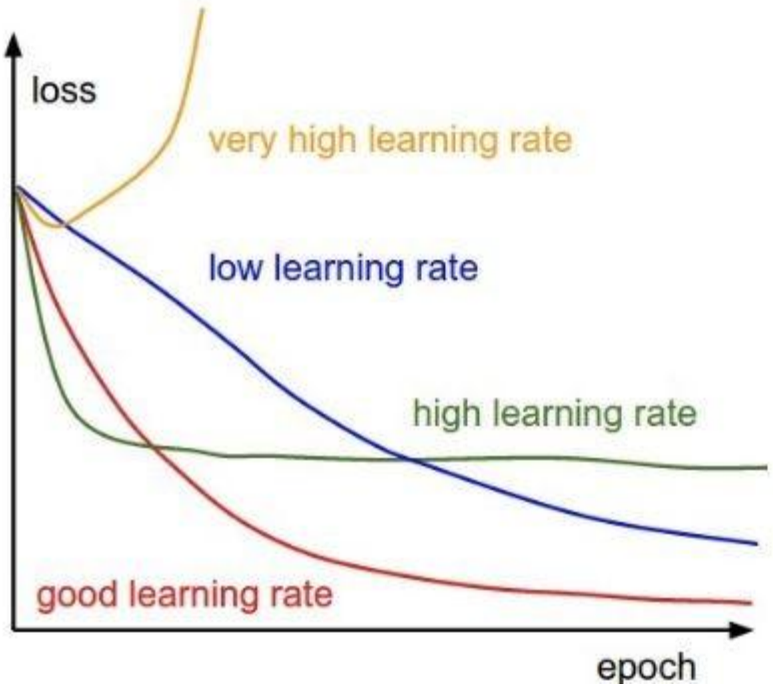


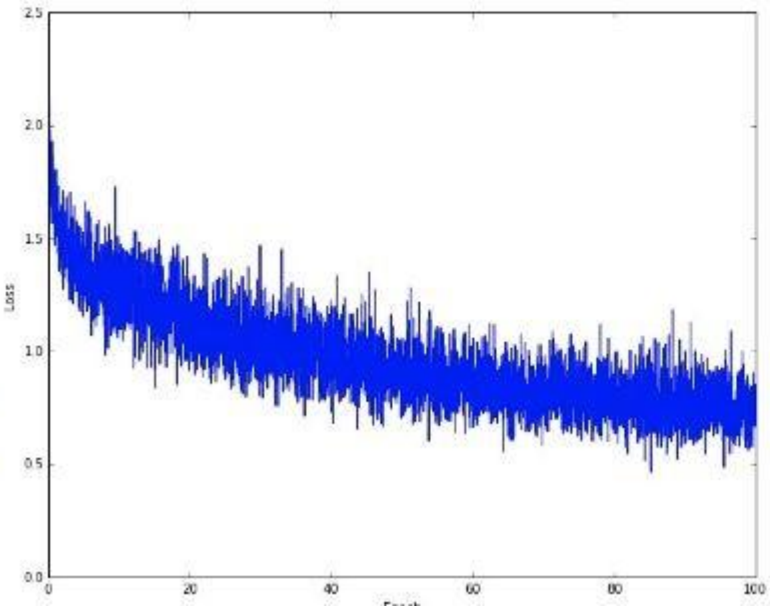
source: http://cs231n.github.io

Gradient optimization

Stochastic gradient descent (and variations) is used to optimize NN parameters.

$$x_{t+1} = x_t - \text{learning rate} \cdot dx$$





source: http://cs231n.github.io/neural-networks-3/

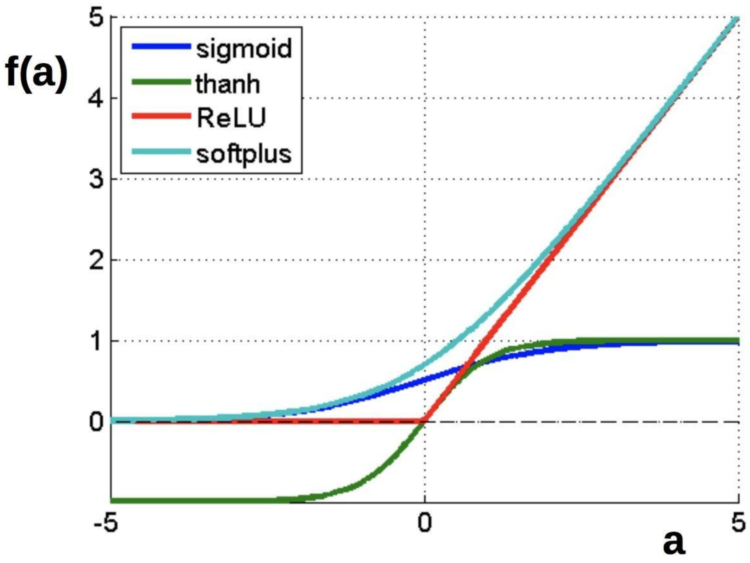
Once more: nonlinearities

$$f(a) = \frac{1}{1+e^a}$$

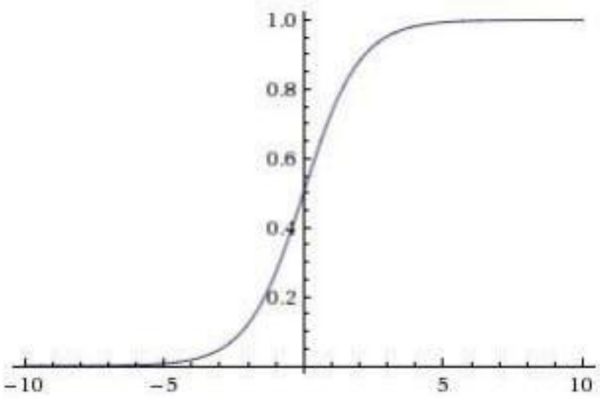
$$f(a) = \tanh(a)$$

$$f(a) = \max(0, a)$$

$$f(a) = \log(1+e^a)$$



Activation functions



Sigmoid

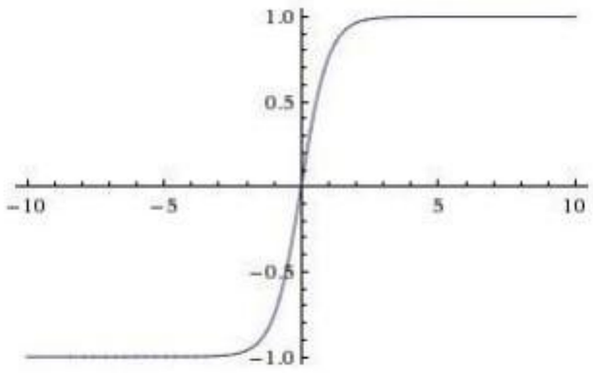
$$f(a) = \frac{1}{1 + e^{a}}$$

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

3 problems:

1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

Activation functions

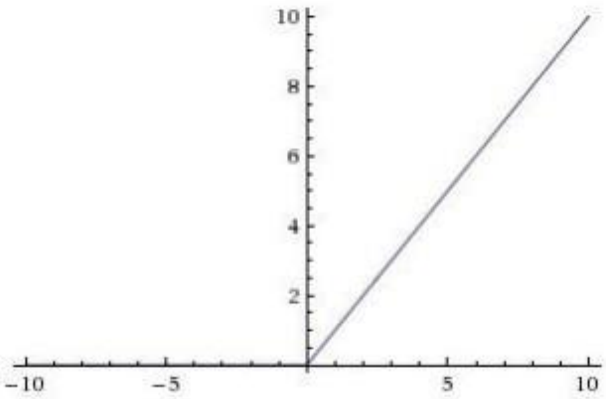


tanh(x)

- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :(

$$f(a) = \tanh(a)$$

Activation functions



ReLU
(Rectified Linear Unit)

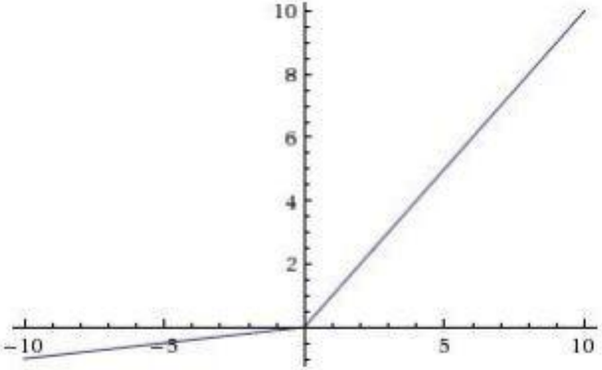
$$f(a) = \max(0, a)$$

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

- Not zero-centered output
- An annoyance:

hint: what is the gradient when x < 0?

Activation functions

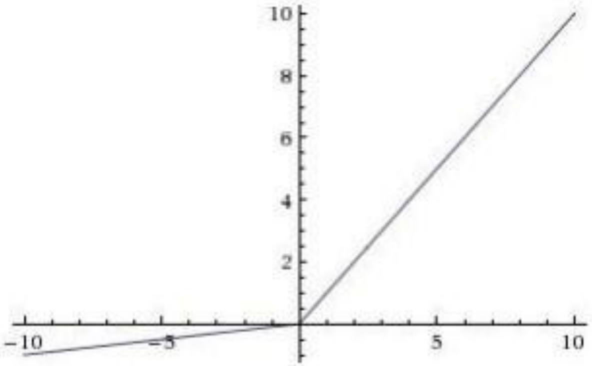


- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not "die".**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

Activation functions



- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not "die".**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

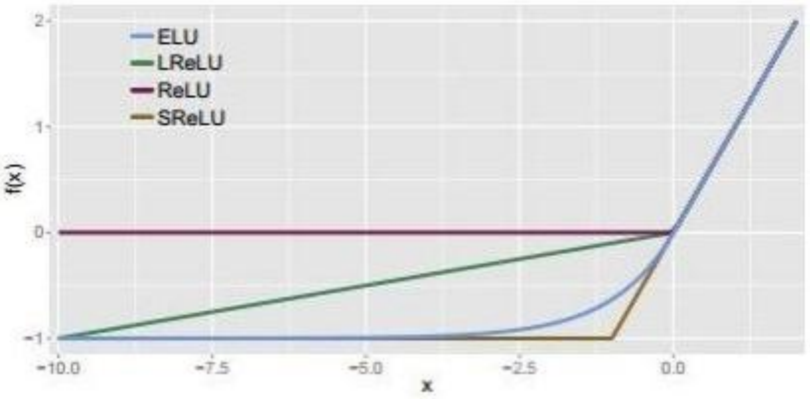
Parametric Rectifier (PReLU)

$$f(x) = \max(\alpha x, x)$$

backprop into \alpha
(parameter)

Activation functions

Exponential Linear Units (ELU)



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

- All benefits of ReLU
- Does not die
- Closer to zero mean outputs

- Computation requires exp()

Activation functions: sum up

- Use ReLU as baseline approach
- Be careful with the learning rates
- Try out Leaky ReLU or ELU
- Try out tanh but do not expect much from it
- Do not use Sigmoid

Weights initialization

Weights initialization

- Pitfall: all zero initialization.

Weights initialization

- Pitfall: all zero initialization.
- Small random numbers.

Weights initialization

- Pitfall: all zero initialization.
- Small random numbers.
- Calibrated random numbers.

$$s = \sum\nolimits_i^n w_i x_i$$

$$\mathrm{Var}(s) = \mathrm{Var}(\sum_i^n w_i x_i)$$

$$= \sum_i^n \mathrm{Var}(w_i x_i)$$

$$= \sum_i^n [E(w_i)]^2 \mathrm{Var}(x_i) + E[(x_i)]^2 \mathrm{Var}(w_i) + \mathrm{Var}(x_i)\mathrm{Var}(w_i)$$

$$= \sum_i^n \mathrm{Var}(x_i)\mathrm{Var}(w_i)$$

$$= (n\mathrm{Var}(w))\ \mathrm{Var}(x)$$

Weights initialization

Xavier initialization (Glorot, Bengio, 2010)

Simple linear neuron

$$y = \mathbf{w}^\top \mathbf{x} + b = \sum_i w_i x_i + b$$

Compute the variance

Weights initialization

Xavier initialization (Glorot, Bengio, 2010)

Simple linear neuron

$$y = \mathbf{w}^\top \mathbf{x} + b = \sum_i w_i x_i + b$$

Compute the variance

$$\begin{aligned} \mathrm{Var}[y_i] &= \mathrm{Var}[w_i x_i] = \mathbb{E}\left[w_i^2 x_i^2\right] - \left(\mathbb{E}[w_i x_i]\right)^2 = \\ &= \mathbb{E}[x_i]^2 \, \mathrm{Var}[w_i] + \mathbb{E}[w_i]^2 \, \mathrm{Var}[x_i] + \mathrm{Var}[w_i] \, \mathrm{Var}[x_i] \end{aligned}$$

Weights initialization

$$\mathrm{Var}[y_i] = \mathrm{Var}[w_i x_i] = \mathbb{E}\left[w_i^2 x_i^2\right] - \left(\mathbb{E}[w_i x_i]\right)^2 =$$
$$= \mathbb{E}[x_i]^2 \mathrm{Var}[w_i] + \mathbb{E}[w_i]^2 \mathrm{Var}[x_i] + \mathrm{Var}[w_i] \mathrm{Var}[x_i]$$

$$\mathrm{Var}[y_i] = \mathrm{Var}[w_i] \mathrm{Var}[x_i]$$

Zero mean for weights and data

$$\mathrm{Var}[y] = \mathrm{Var}\left[\sum_{i=1}^{n_{\text{out}}} y_i\right] = \sum_{i=1}^{n_{\text{out}}} \mathrm{Var}[w_i x_i] = n_{\text{out}} \mathrm{Var}[w_i] \mathrm{Var}[x_i]$$

Weights initialization

$$\operatorname{Var}[y_i] = \operatorname{Var}[w_i x_i] = \mathbb{E}\left[w_i^2 x_i^2\right] - \left(\mathbb{E}[w_i x_i]\right)^2 =$$
$$= \mathbb{E}[x_i]^2 \operatorname{Var}[w_i] + \mathbb{E}[w_i]^2 \operatorname{Var}[x_i] + \operatorname{Var}[w_i] \operatorname{Var}[x_i]$$

$$\operatorname{Var}[y_i] = \operatorname{Var}[w_i] \operatorname{Var}[x_i]$$

Zero mean for weights and data

$$\operatorname{Var}[y] = \operatorname{Var}\left[\sum_{i=1}^{n_{\text{out}}} y_i\right] = \sum_{i=1}^{n_{\text{out}}} \operatorname{Var}[w_i x_i] = \boxed{n_{\text{out}} \operatorname{Var}[w_i]} \operatorname{Var}[x_i]$$

Weights init: Neural Networks: Tricks of the Trade

$$w_i \sim U\left[-\frac{1}{\sqrt{n_{\text{out}}}}, \frac{1}{\sqrt{n_{\text{out}}}}\right]$$

$$\mathbf{Var}[w_i] = \frac{1}{12}\left(\frac{1}{\sqrt{n_{\text{out}}}} + \frac{1}{\sqrt{n_{\text{out}}}}\right)^2 = \frac{1}{3n_{\text{out}}}$$

$$n_{\text{out}}\,\mathbf{Var}[w_i] = \frac{1}{3}$$

Weights init: How to fix it?

$$\mathrm{Var}[w_i] = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$

$$w_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{\text{in}} + n_{\text{out}}}}, \frac{\sqrt{6}}{\sqrt{n_{\text{in}} + n_{\text{out}}}}\right]$$

$$\mathrm{Var}[w_i x_i] = \mathbb{E}[x_i]^2 \,\mathrm{Var}[w_i] + \mathbb{E}[w_i]^2 \,\mathrm{Var}[x_i] + \mathrm{Var}[w_i]\,\mathrm{Var}[x_i]$$

Weights init: relu case

$$\operatorname{Var}[w_i x_i] = \mathbb{E}[x_i]^2 \operatorname{Var}[w_i] + \mathbb{E}[w_i]^2 \operatorname{Var}[x_i] + \operatorname{Var}[w_i] \operatorname{Var}[x_i]$$

$$\operatorname{Var}[w_i x_i] = \mathbb{E}[x_i]^2 \operatorname{Var}[w_i] + \operatorname{Var}[w_i] \operatorname{Var}[x_i] = \operatorname{Var}[w_i] \mathbb{E}\left[x_i^2\right]$$

$$\operatorname{Var}\left[y^{(l)}\right] = n_{\text{in}}^{(l)} \operatorname{Var}\left[w^{(l)}\right] \mathbb{E}\left[\left(x^{(l)}\right)^2\right]$$

Weights init: ReLU case

$$\mathrm{Var}\left[y^{(l)}\right] = n_{\text{in}}^{(l)}\,\mathrm{Var}\left[w^{(l)}\right]\mathbb{E}\left[\left(x^{(l)}\right)^2\right]$$

$$x^{(l)} = \max\left(0, y^{(l-1)}\right)$$

Symmetric distribution across zero for y

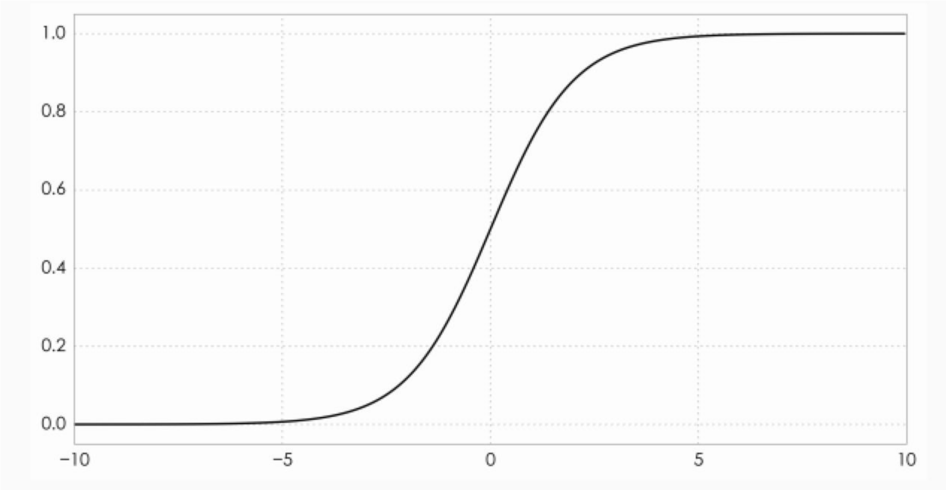
$$\mathbb{E}\left[\left(x^{(l)}\right)^2\right] = \frac{1}{2}\mathrm{Var}\left[y^{(l-1)}\right], \qquad \mathrm{Var}\left[y^{(l)}\right] = \frac{n_{\text{in}}^{(l)}}{2}\mathrm{Var}\left[w^{(l)}\right]\mathrm{Var}\left[y^{(l-1)}\right]$$

Weights init: ReLU case

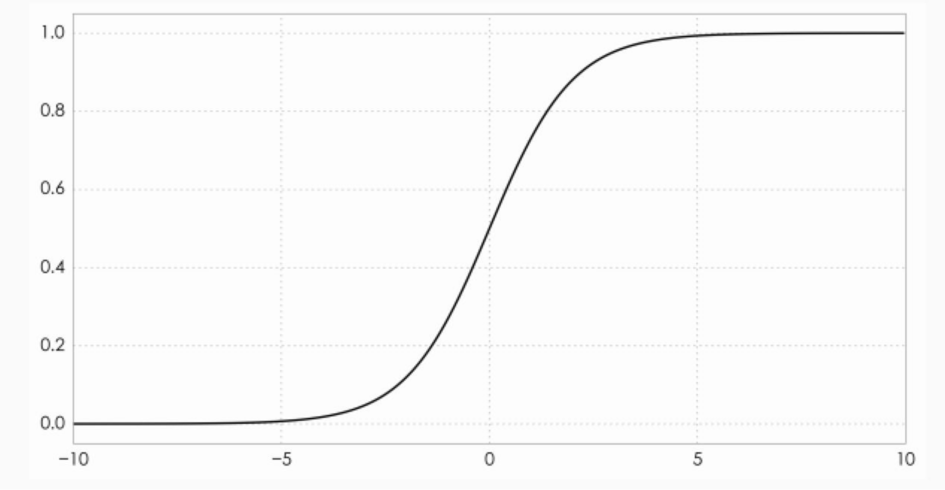
$$\mathrm{Var}\left[y^{(l)}\right] = \frac{n_{\mathrm{in}}^{(l)}}{2}\mathrm{Var}\left[w^{(l)}\right]\mathrm{Var}\left[y^{(l-1)}\right]$$

$$\mathrm{Var}[w_i] = 2/n_{\mathrm{in}}^{(l)} \qquad w_i \sim N(0, \sqrt{2/n_{\mathrm{in}}^{(l)}})$$

Weights init: Sigmoid



Weights init: Task



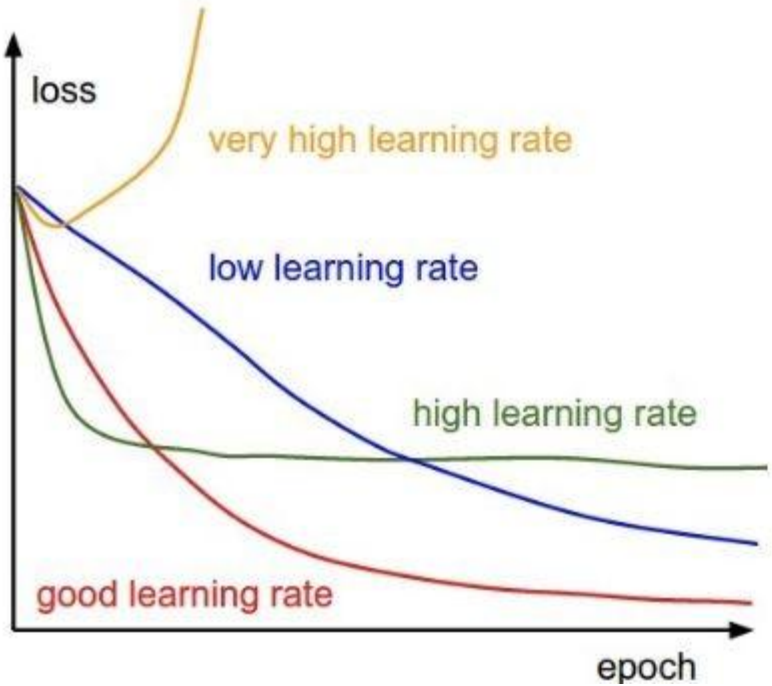
Определим две функции: $\sigma(z) = \frac{1}{1+e^{-z}}$ и $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$. Предположим, что перед обучением вашей нейронной сети состоящей из нескольких полно-связных слоев с одной из функций активаций указанных выше мы делаем следующие предположения: а) Данные центрированы по нулю. б)Все веса инициализируются независимо со средним значением 0 и дисперсией 0.001. с) Все смещения инициализируются до 0. д) Скорость обучения мала и фиксирована.

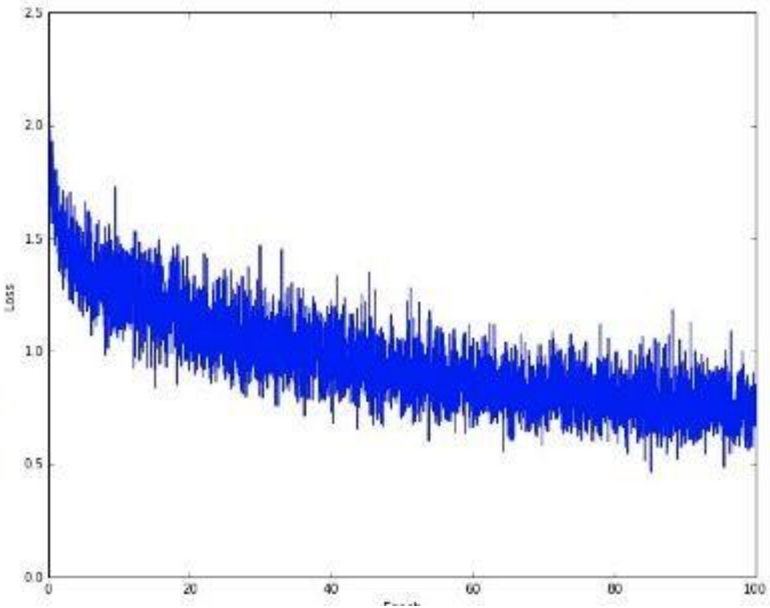
Попробуйте объяснить, какая функция активации между $\tanh$ и σ приведет к более высокому градиенту во время первого обновления.

Optimizers

Stochastic gradient descent is used to optimize NN parameters.

$$x_{t+1} = x_t - \text{learning rate} \cdot dx$$



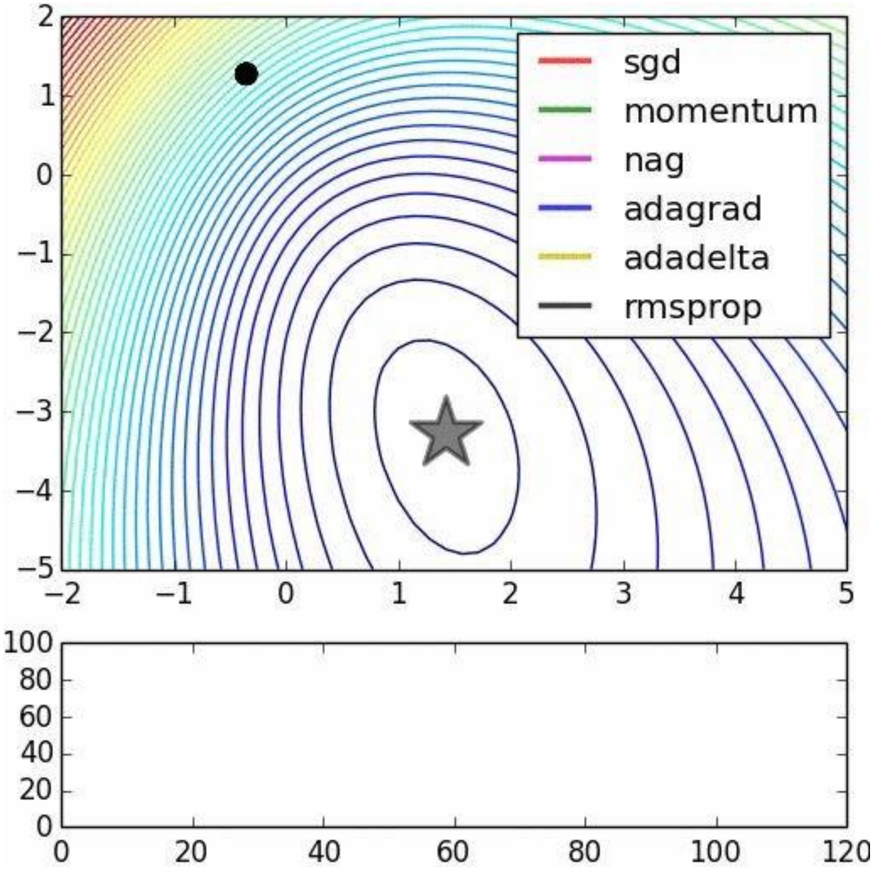


source: http://cs231n.github.io/neural-networks-3/

Optimizers

There are much more optimizers:

- Momentum
- Adagrad
- Adadelta
- RMSprop
- Adam
- …
- even other NNs

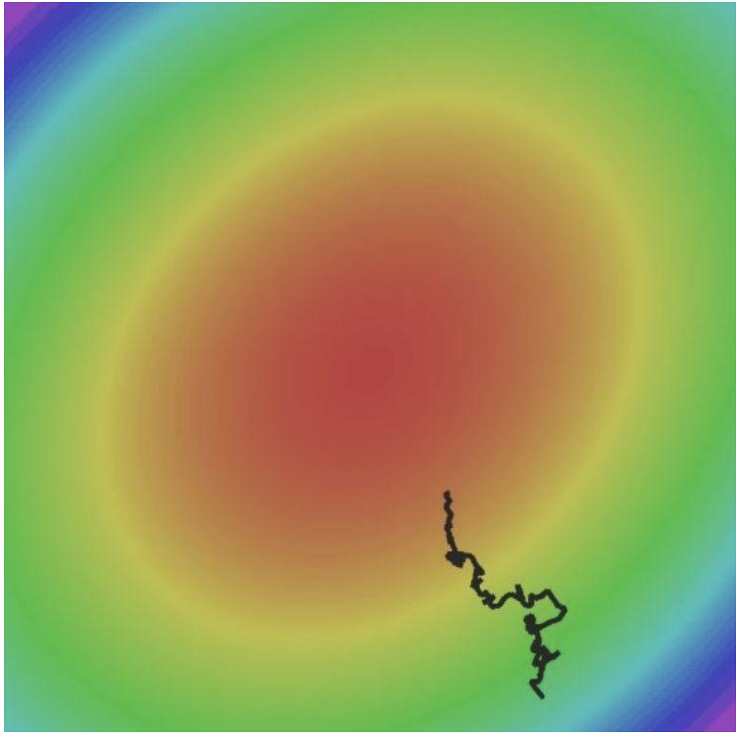


Optimization: SGD

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(x_i, y_i, W)$$

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W)$$

Averaging over minibatches ---> noisy gradient



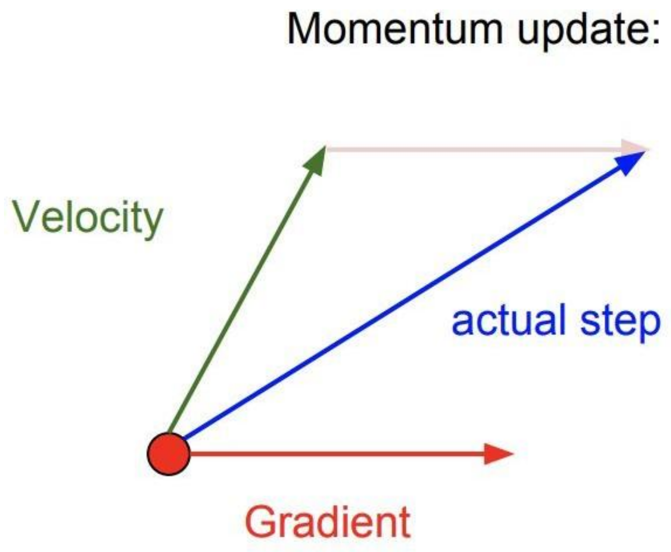
First idea: momentum

Simple SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

SGD with momentum

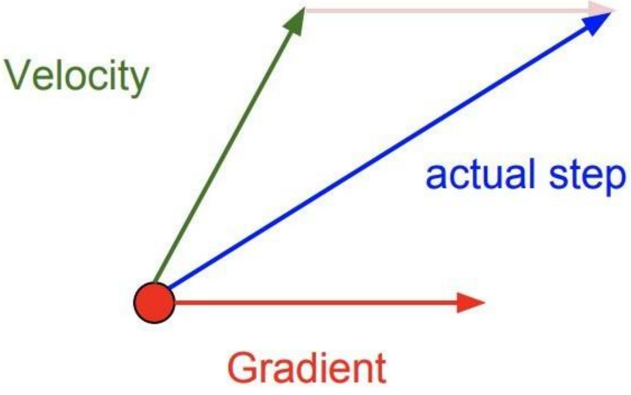
$$v_{t+1} = \rho v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$



source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

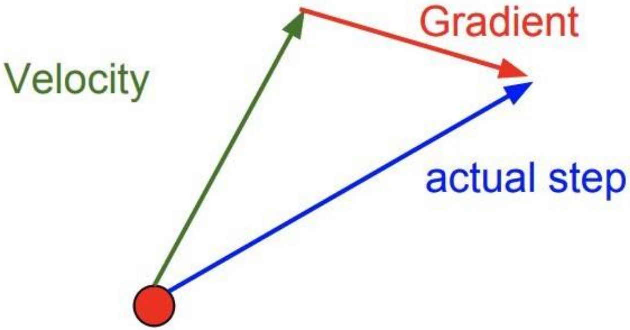
Nesterov momentum

Momentum update:



$$v_{t+1} = \rho v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$

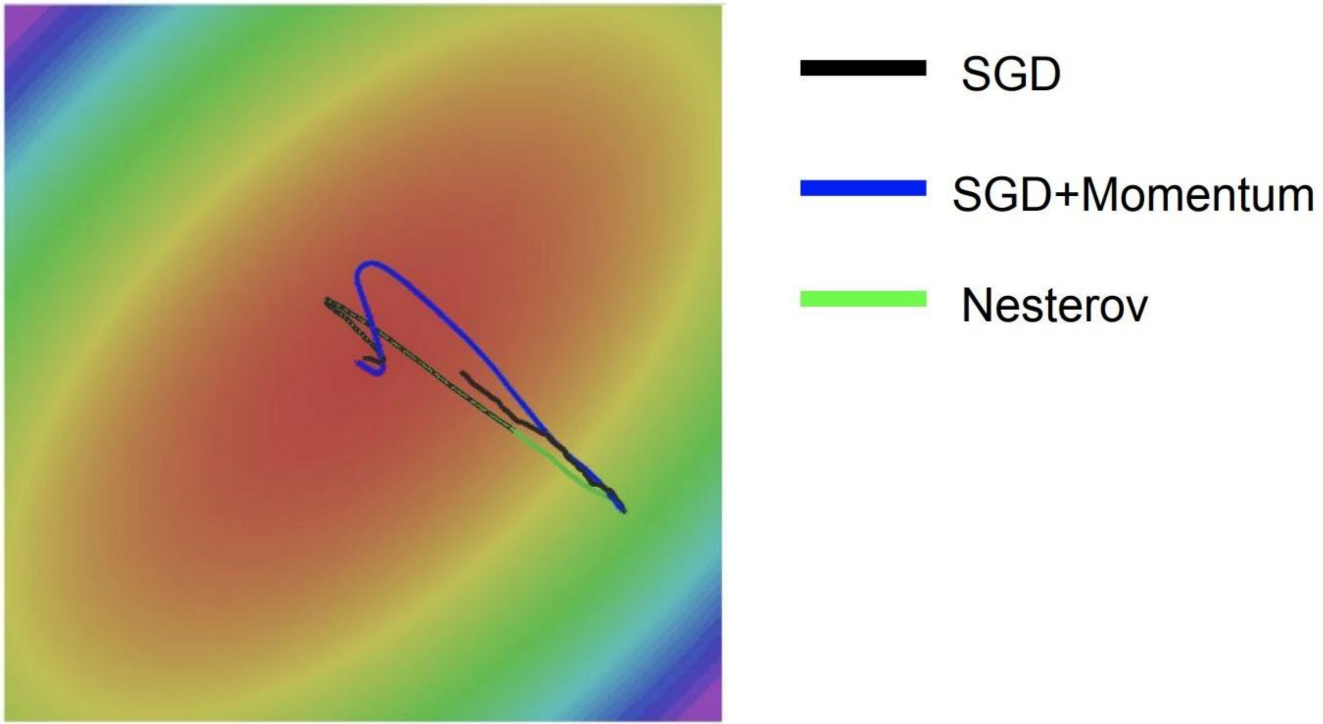
Nesterov Momentum



$$v_{t+1} = \rho v_t - \alpha \nabla f(\boxed{x_t + \rho v_t})$$
$$x_{t+1} = x_t + v_{t+1}$$

source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

Comparing momentums



source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

Second idea: different dimensions are different

Adagrad: SGD with cache

$$\text{cache}_{t+1} = \text{cache}_t + (\nabla f(x_t))^2$$
$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$

Second idea: different dimensions are different

Adagrad: SGD with cache

$$\text{cache}_{t+1} = \text{cache}_t + (\nabla f(x_t))^2$$
$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$

Problem: gradient fades with time

Second idea: different dimensions are different

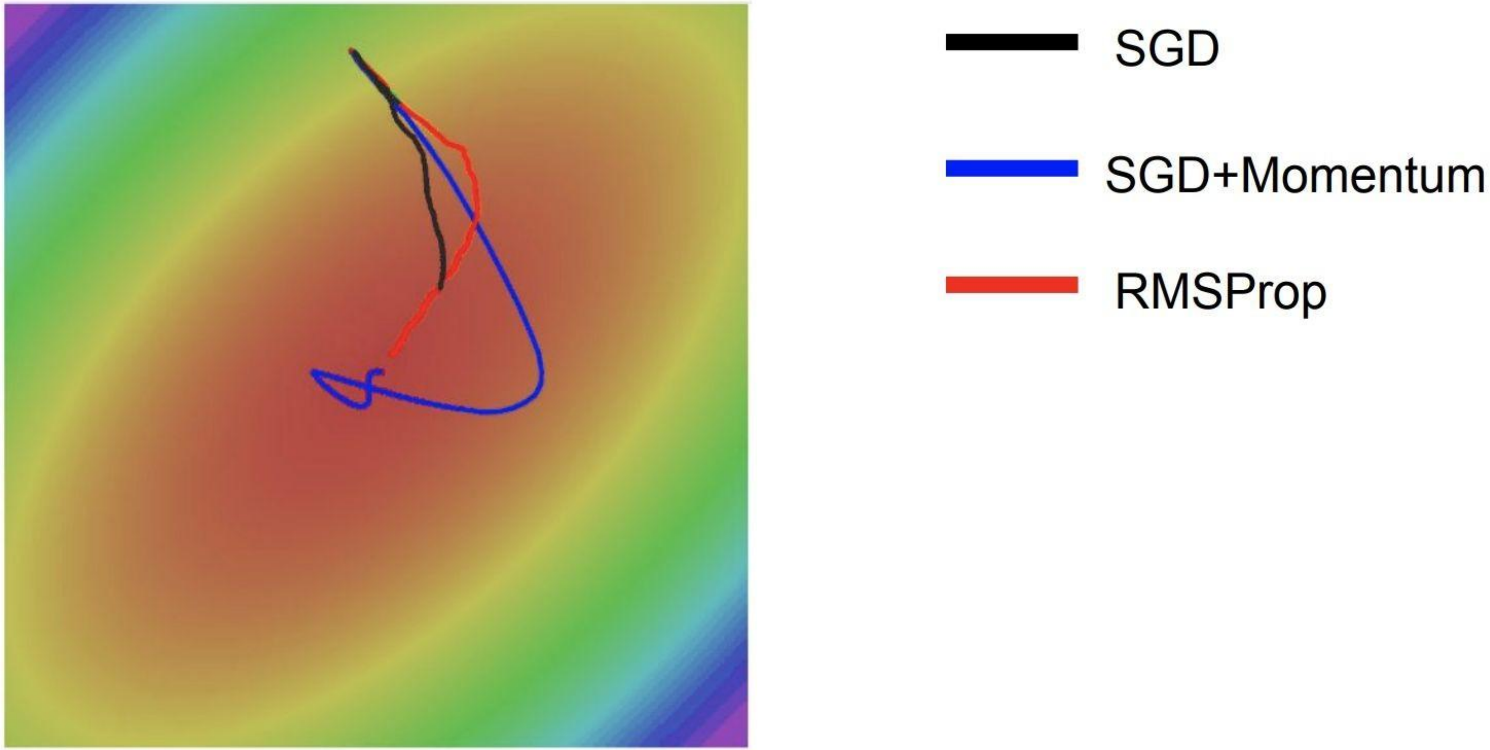
Adagrad: SGD with cache

$$\text{cache}_{t+1} = \text{cache}_t + (\nabla f(x_t))^2$$
$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$

RMSProp: SGD with cache with exp. Smoothing

$$\text{cache}_{t+1} = \beta \text{cache}_t + (1 - \beta)(\nabla f(x_t))^2$$
$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$

Slide 29 Lecture 6 of Geoff Hinton's Coursera class
http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf



source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

Adam

Let's combine the momentum idea and RMSProp normalization:

$$v_{t+1} = \gamma v_t + (1-\gamma)\nabla f(x_t)$$

$$\text{cache}_{t+1} = \beta \text{cache}_t + (1-\beta)(\nabla f(x_t))^2$$

$$x_{t+1} = x_t - \alpha \frac{v_{t+1}}{\text{cache}_{t+1} + \varepsilon}$$

Adam full form involves bias correction term. See http://cs231n.github.io/neural-networks-3/ for more info.

Adam

Let's combine the momentum idea and RMSProp normalization:

$$v_{t+1} = \gamma v_t + (1-\gamma)\nabla f(x_t)$$

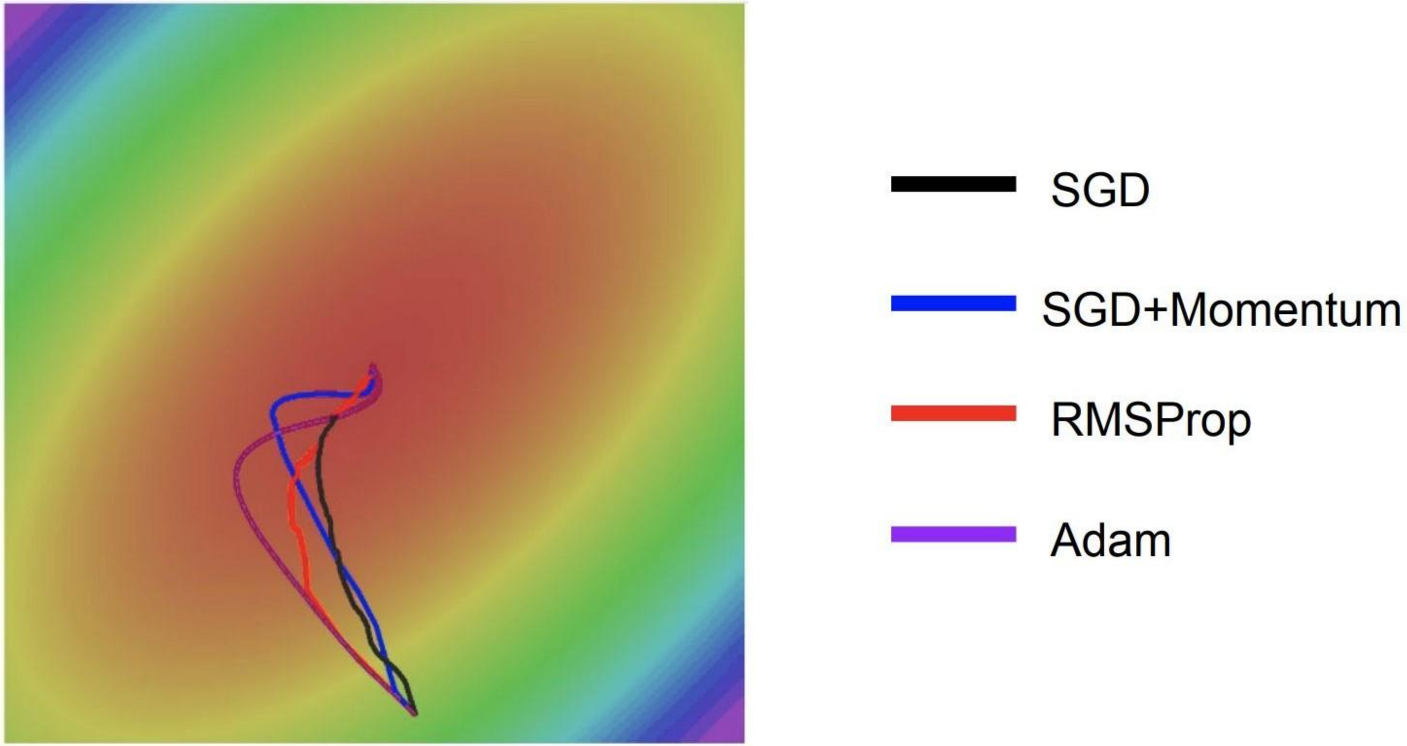
$$\text{cache}_{t+1} = \beta \text{cache}_t + (1-\beta)(\nabla f(x_t))^2$$

$$x_{t+1} = x_t - \alpha \frac{v_{t+1}}{\text{cache}_{t+1} + \varepsilon}$$

Actually, that's not quite Adam.

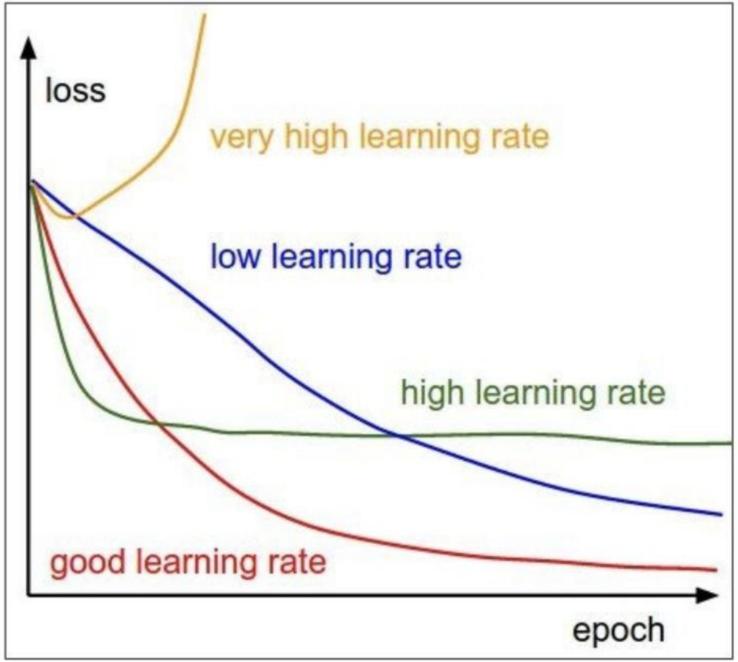
Adam full form involves bias correction term. See http://cs231n.github.io/neural-networks-3/ for more info.

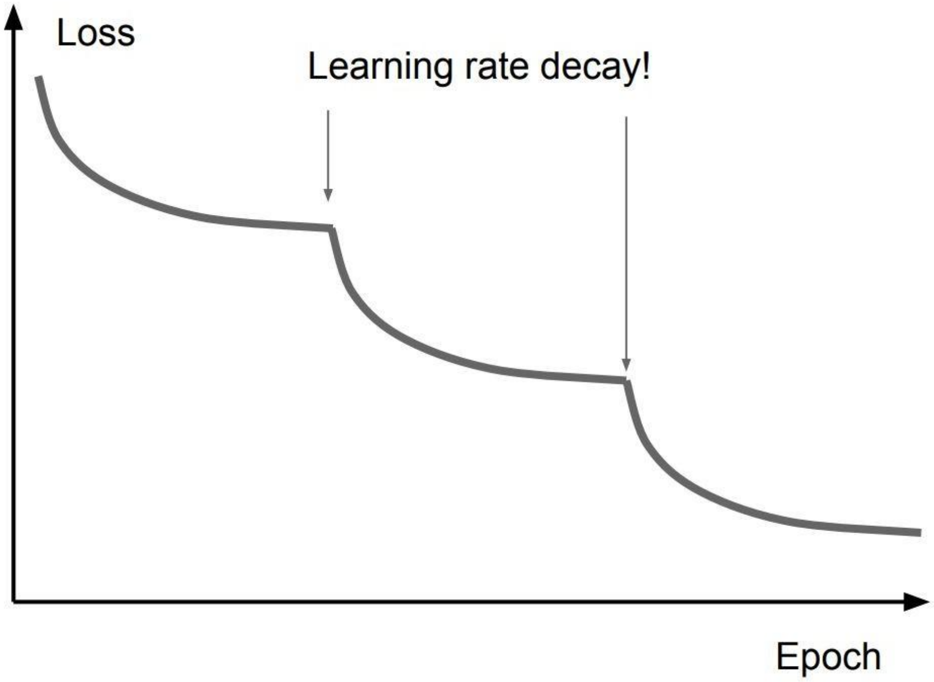
Comparing optimizers



source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

Once more: learning rate

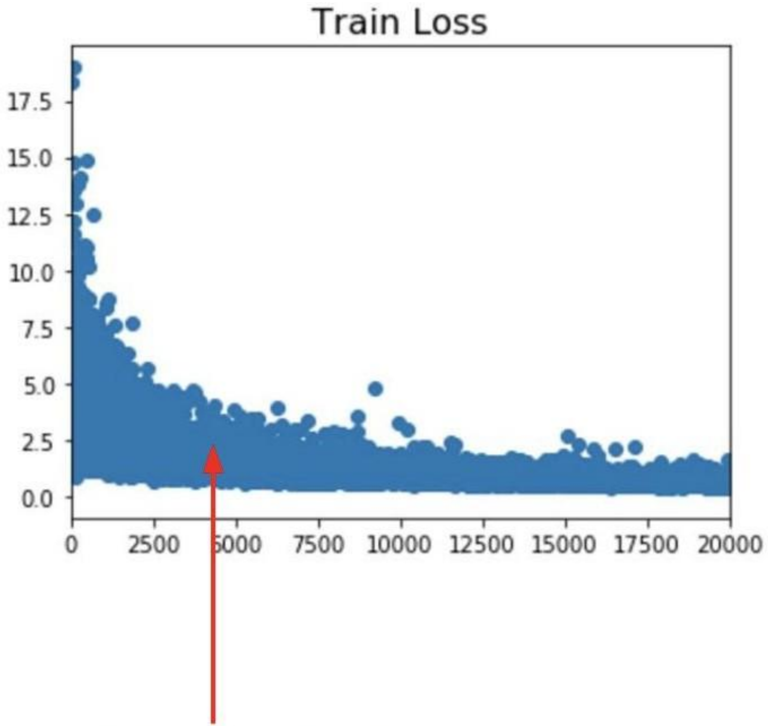


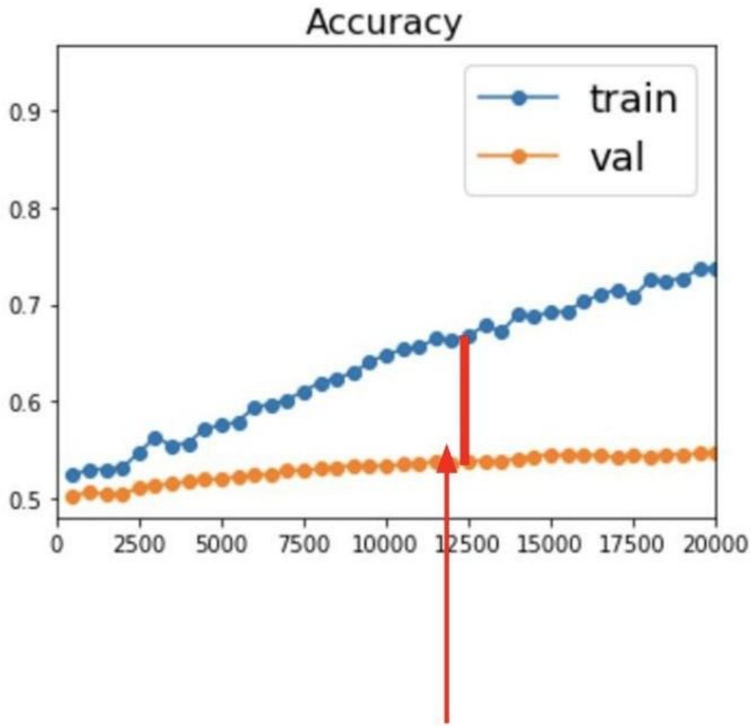


source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

Sum up: optimization

- Adam is great basic choice
- Even for Adam/RMSProp learning rate matters
- Use learning rate decay
- Monitor your model quality





Better optimization algorithms help reduce training loss

But we really care about error on new data - how to reduce the gap?

source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf