академия больших данных

mail.ru group

girafe ai

Machine Learning, Lecture 7

# Intro to Deep Learning

Radoslav Neychev, Fall 2021

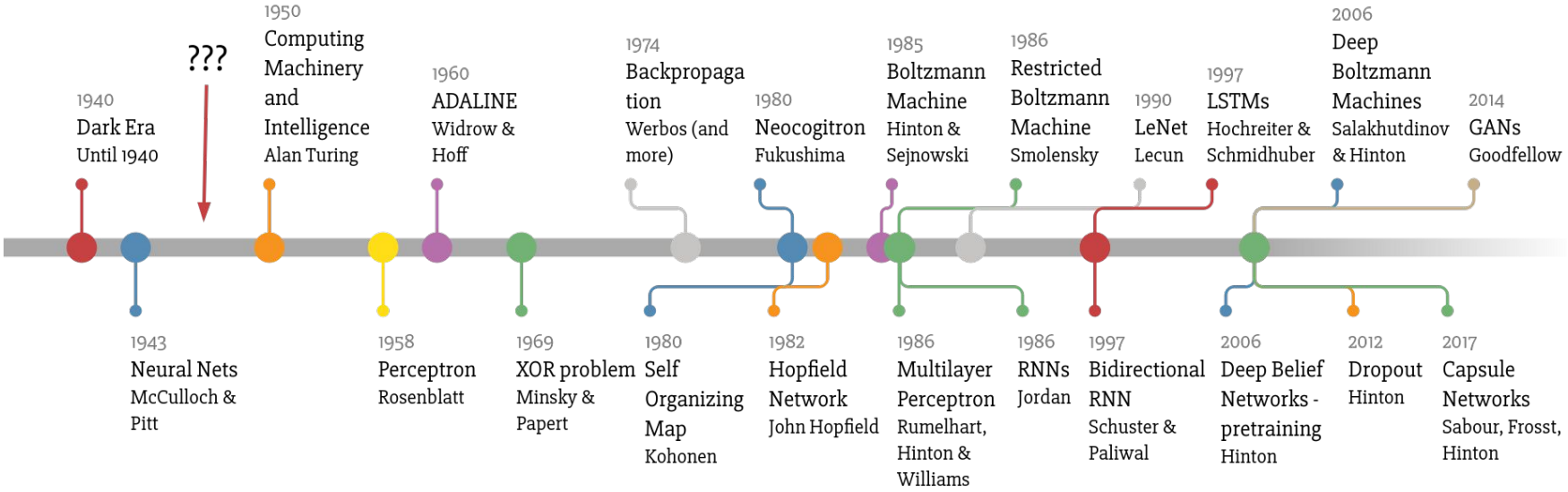
# Outline

1. Neural Networks in different areas. Historical overview.
2. Backpropagation.
3. More on backpropagation.
4. Activation functions.
5. Playground.

# History of Deep Learning

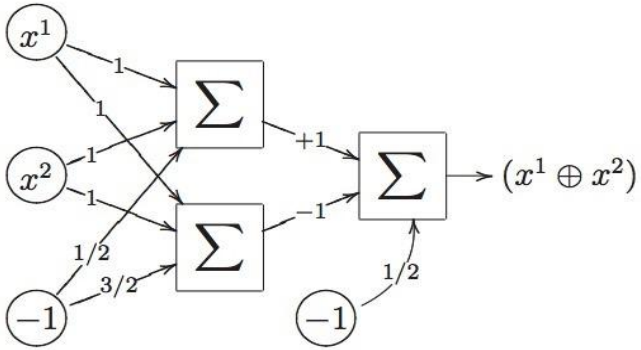
girafe
ai

# Deep Learning Timeline



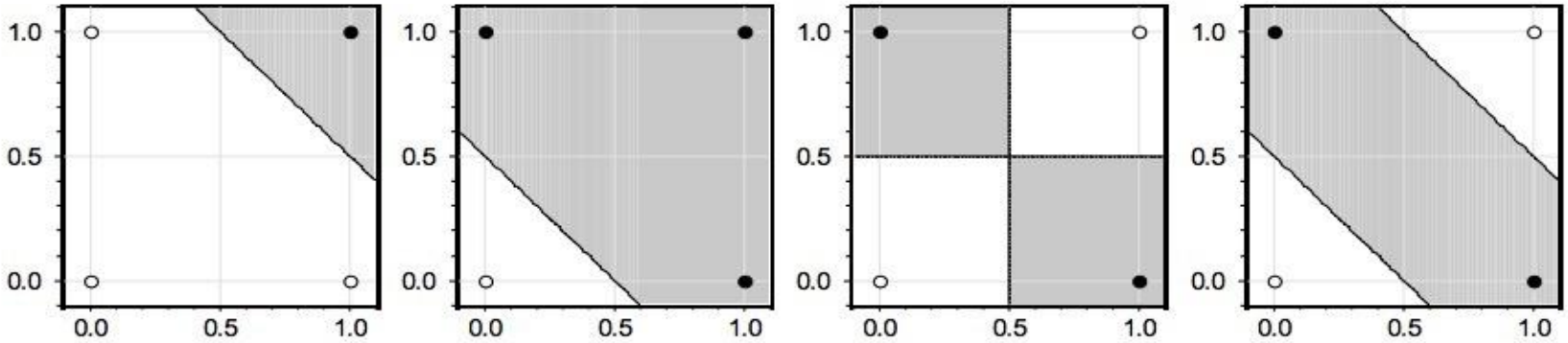

Made by Favio Vázquez

# XOR problem



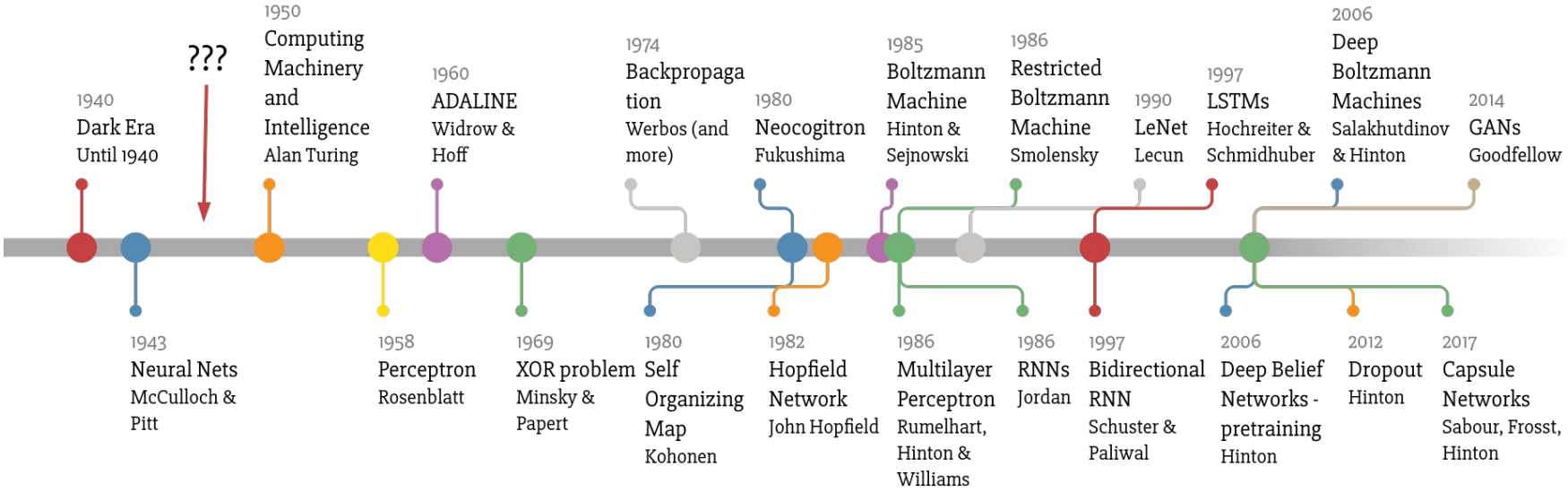

This 2-layer NN (on the left) implements XOR with only x1 and x2 features.

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



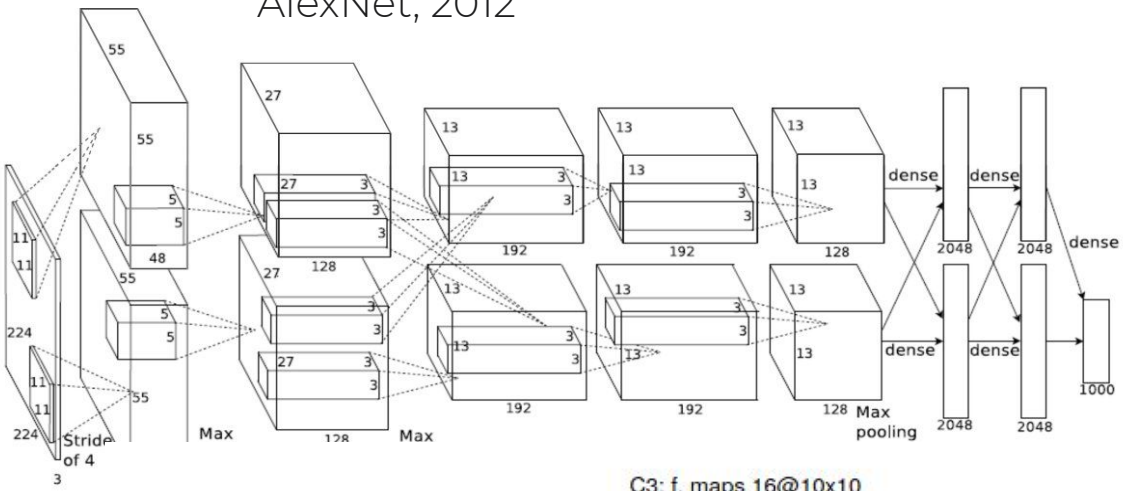

AND OR XOR(with x1*x2) XOR

# Deep Learning Timeline

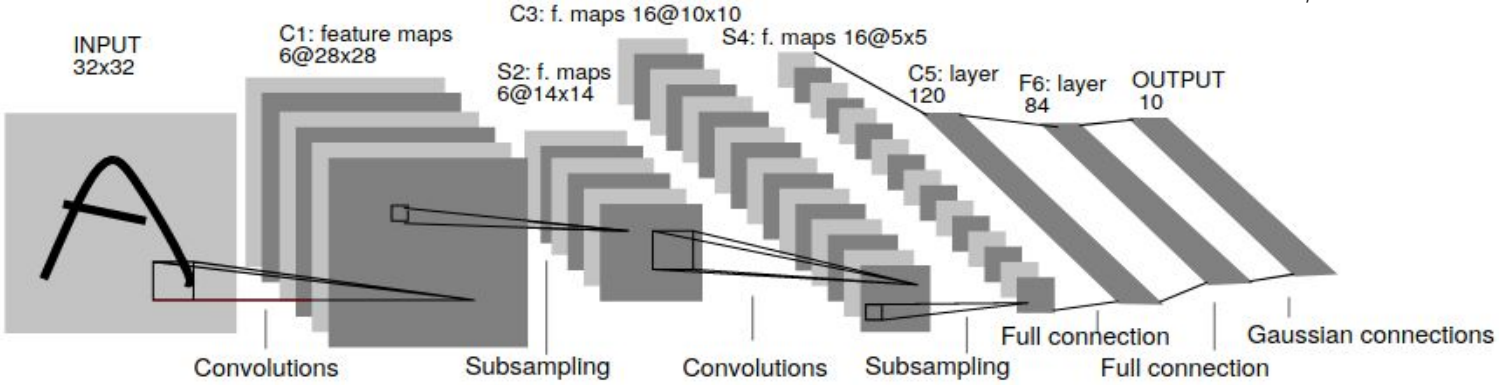



Made by Favio Vázquez

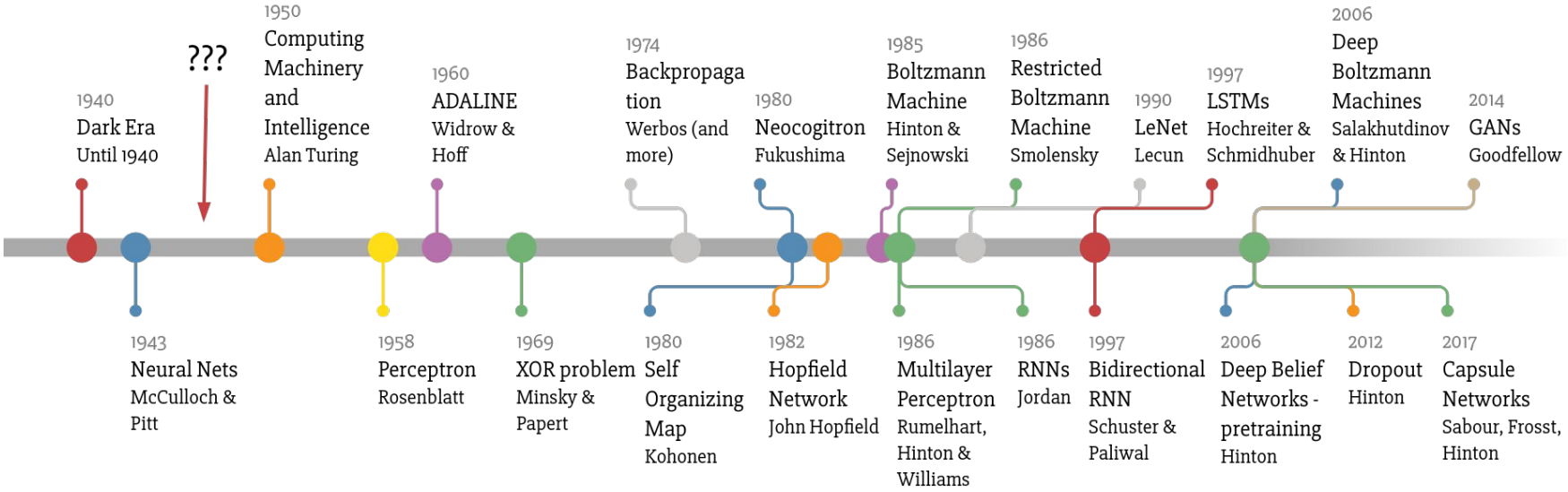
AlexNet, 2012




LeNet-5, 1998



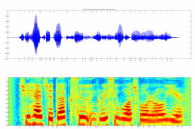
# Deep Learning Timeline



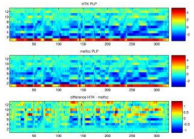

Made by Favio Vázquez

# Real world applications

Audio Features

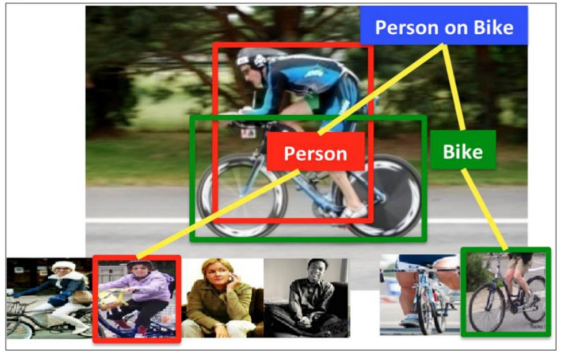


Spectrogram



MFCC

- Object detection
- Action classification
- Image captioning
- …

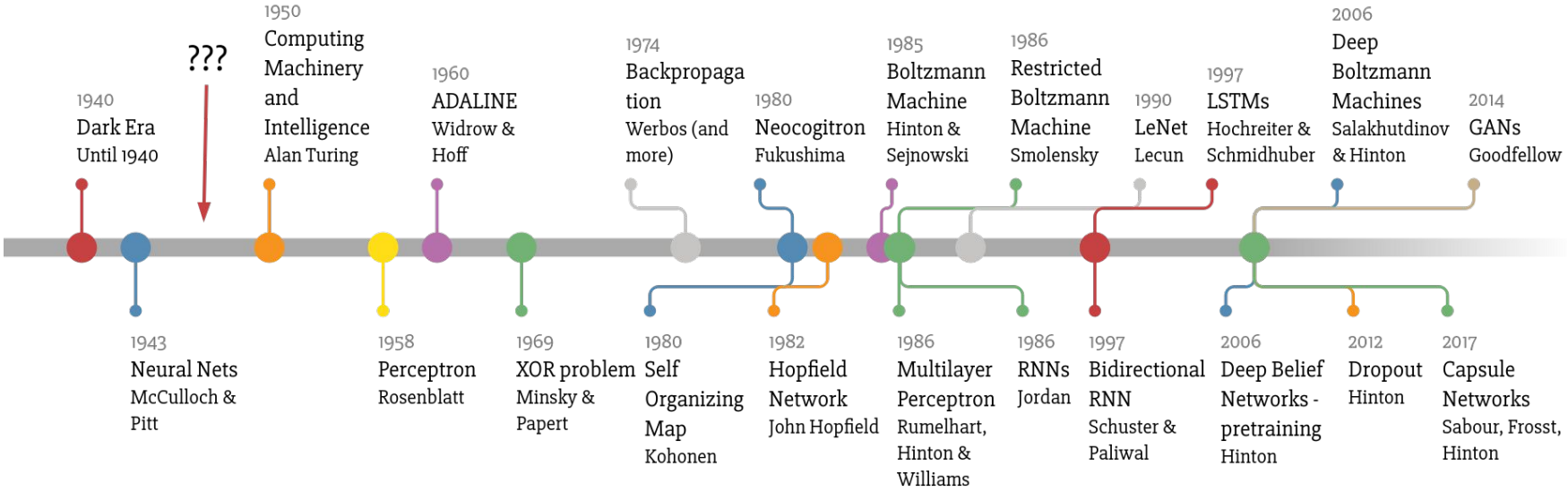


"man in black shirt is playing guitar."

# GANs. 2014+

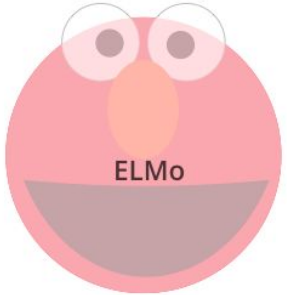
https://thispersondoesnotexist.com/

# Deep Learning Timeline

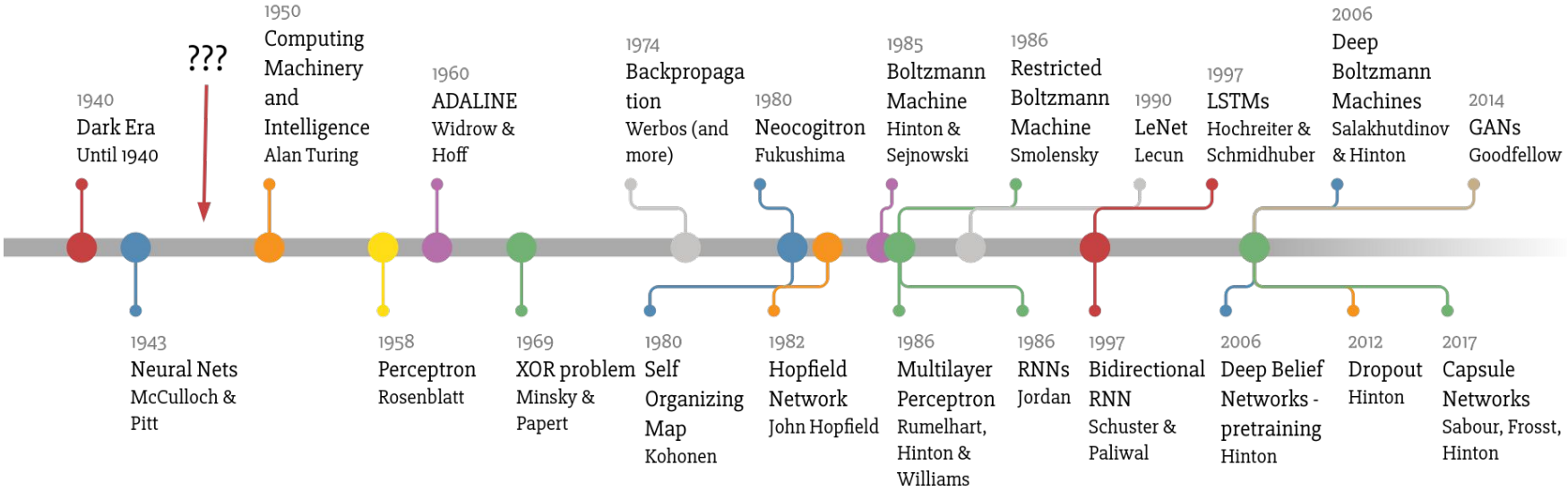



Made by Favio Vázquez

# Transformer, BERT, GPT-2 and more, 2017+



# Deep Learning Timeline



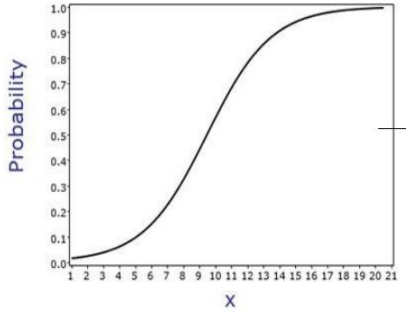

Made by Favio Vázquez

# Deep Learning: intuition

girafe
ai

# Logistic regression

X → Wx +b →



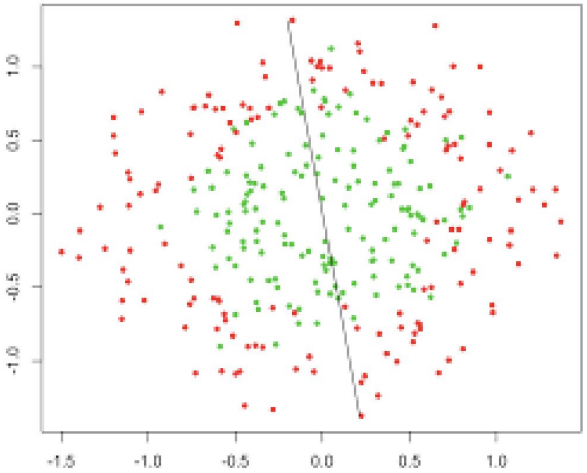

→ P(y)

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

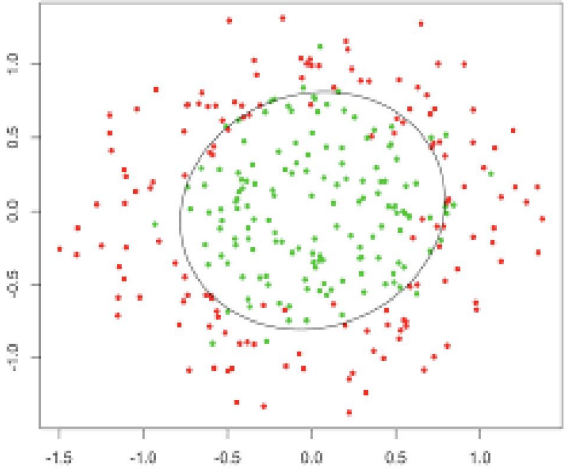
$$L=-\sum_i y_i \log P(y|x_i)+(1-y_i)\log(1-P(y|x_i))$$

# Problem: nonlinear dependencies

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



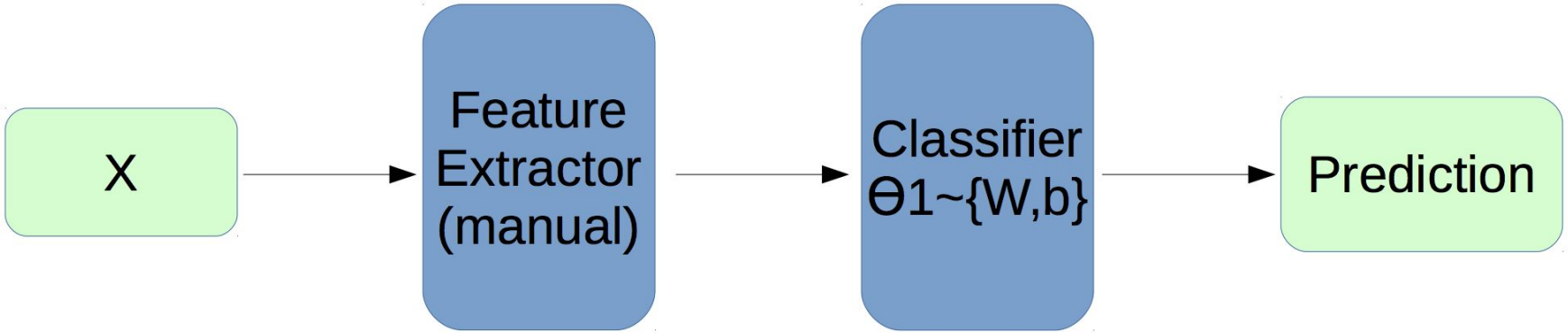

What we have




What we want

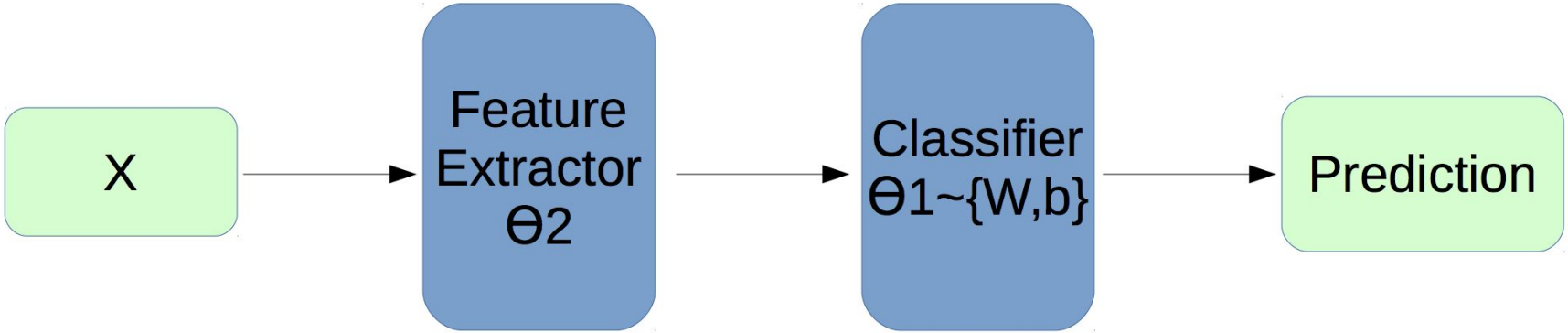
And feature engineering is an art.

# Classic pipeline



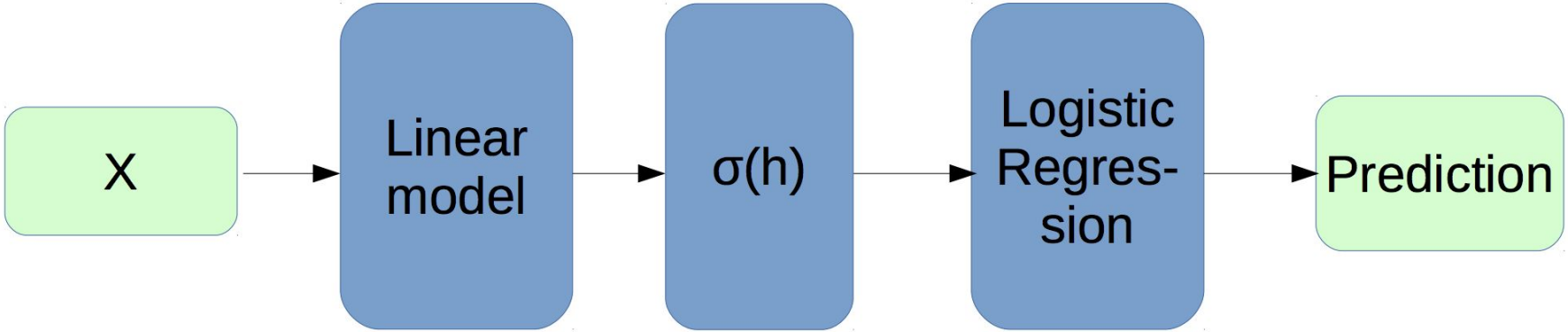

Handcrafted features, generated by experts.

# NN pipeline




Automatically extracted features.

# NN pipeline: example




E.g. two logistic regressions one after another.

Actually, it's a neural network.

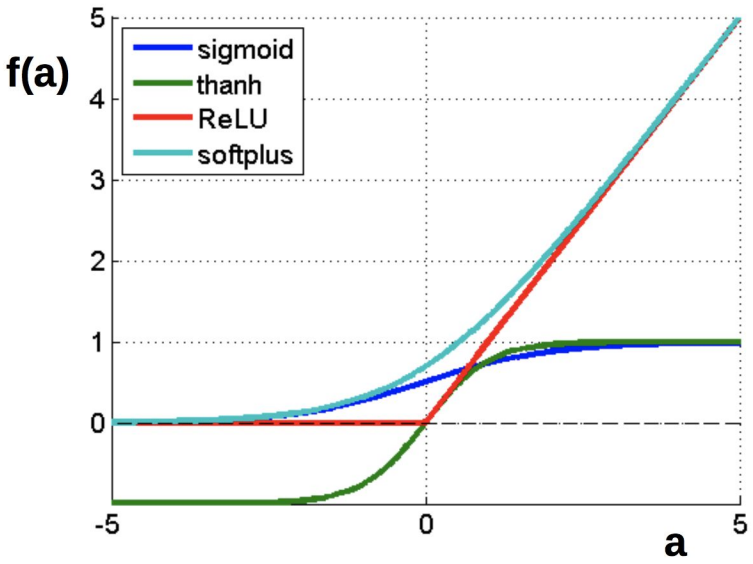
# Activation functions: nonlinearities

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

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

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

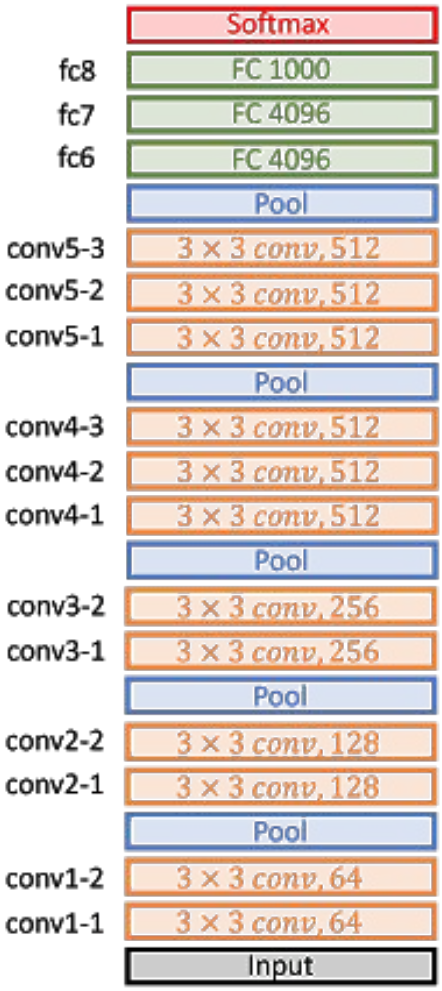
$$f(a) = \log(1 + e^{a})$$



# Some generally accepted terms

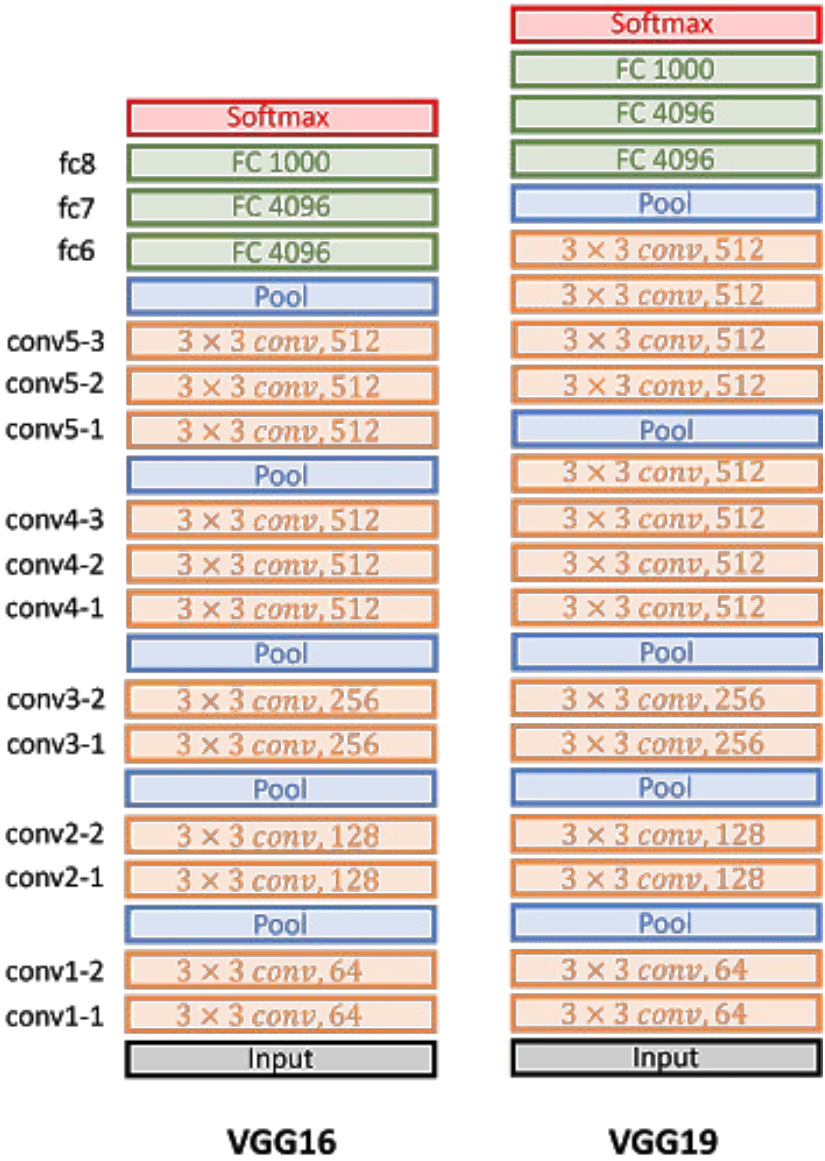
- Layer – a building block for NNs :
  - Dense/Linear/FC layer: $f(x) = Wx+b$
  - Nonlinearity layer: $f(x) = \sigma(x)$
  - Input layer, output layer
  - A few more we will cover later
- Activation function – function applied to layer output
  - Sigmoid
  - tanh
  - ReLU
  - Any other function to get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

Actually, networks can be deep

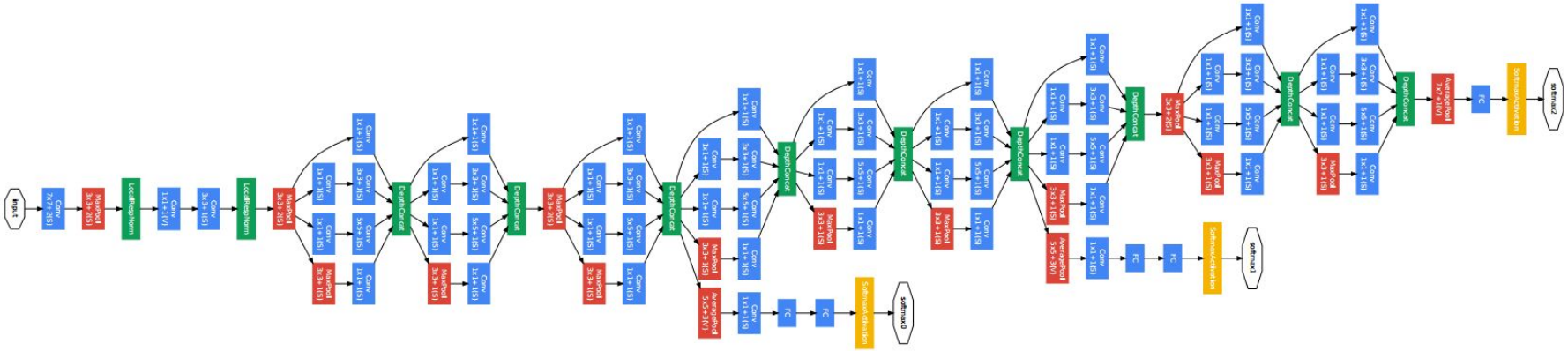



**VGG16**

And deeper...


Softmax
fc8
FC 1000
fc7
FC 4096
fc6
FC 4096
Pool
conv5-3
3 × 3 conv, 512
conv5-2
3 × 3 conv, 512
conv5-1
3 × 3 conv, 512
Pool
conv4-3
3 × 3 conv, 512
conv4-2
3 × 3 conv, 512
conv4-1
3 × 3 conv, 512
Pool
conv3-2
3 × 3 conv, 256
conv3-1
3 × 3 conv, 256
Pool
conv2-2
3 × 3 conv, 128
conv2-1
3 × 3 conv, 128
Pool
conv1-2
3 × 3 conv, 64
conv1-1
3 × 3 conv, 64
Input
VGG16
Softmax
FC 1000
FC 4096
FC 4096
Pool
3 × 3 conv, 512
3 × 3 conv, 512
3 × 3 conv, 512
3 × 3 conv, 512
Pool
3 × 3 conv, 512
3 × 3 conv, 512
3 × 3 conv, 512
3 × 3 conv, 512
Pool
3 × 3 conv, 256
3 × 3 conv, 256
Pool
3 × 3 conv, 128
3 × 3 conv, 128
Pool
3 × 3 conv, 64
3 × 3 conv, 64
Input
VGG19

# Much deeper...



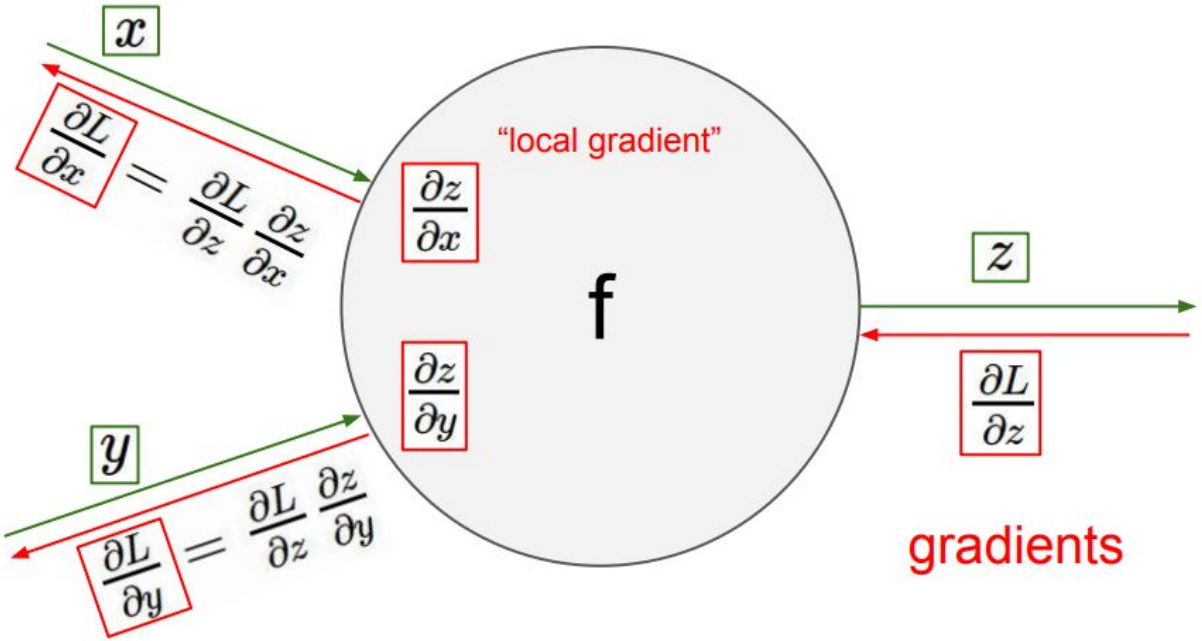

How to train it?

# 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

# Backpropagation

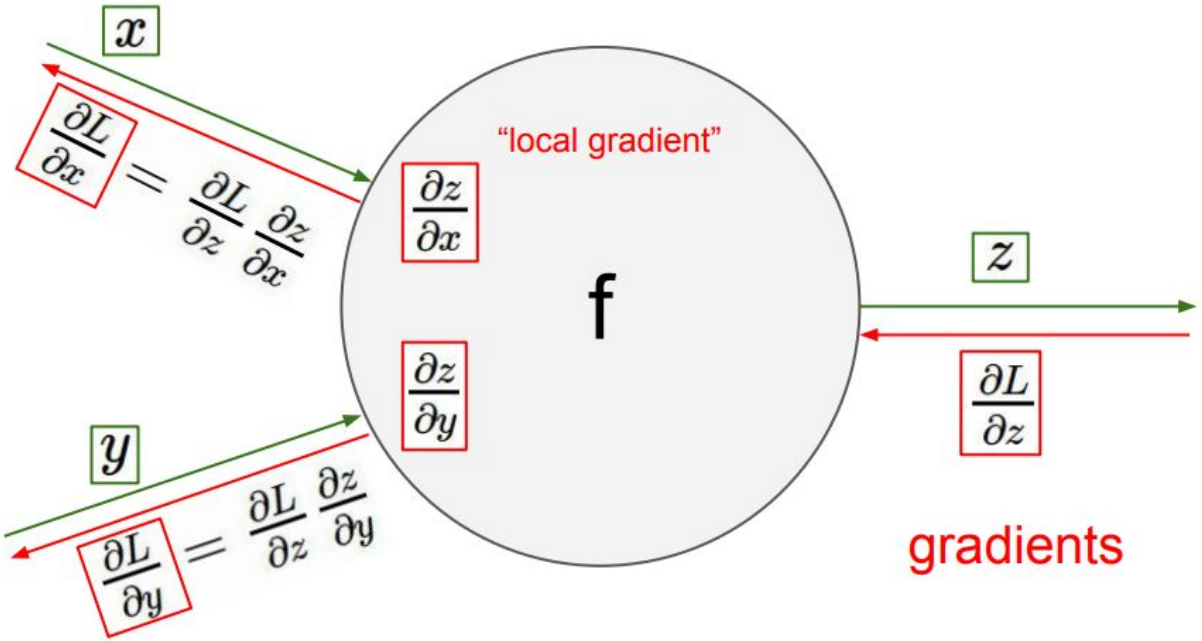
girafe
ai

# 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.

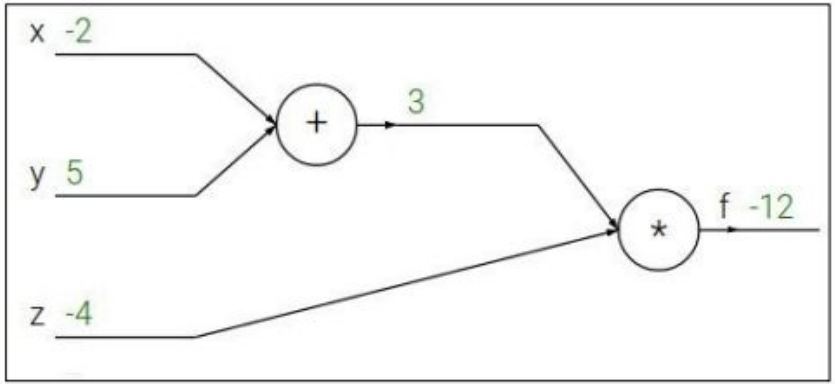



source

# Backpropagation example

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

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

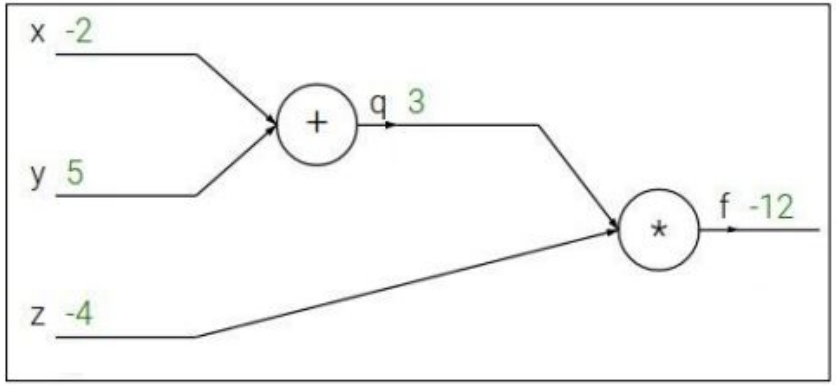



source

# 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

# Backpropagation example

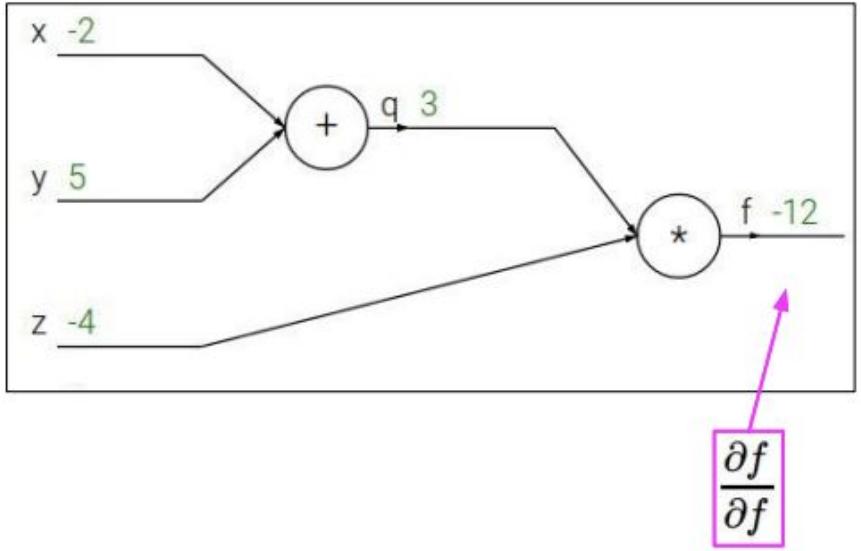
$$f(x, y, z) = (x + y)z$$

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

# Backpropagation example

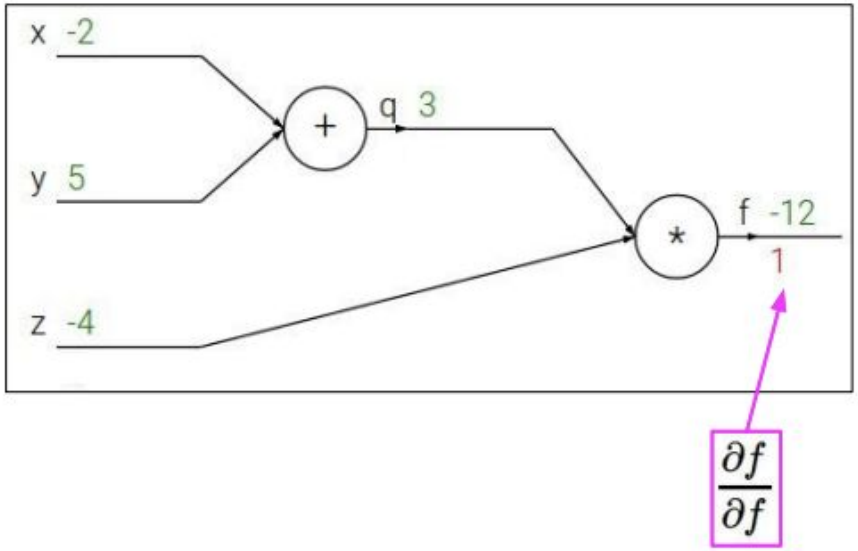
$$f(x, y, z) = (x + y)z$$

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

# Backpropagation example

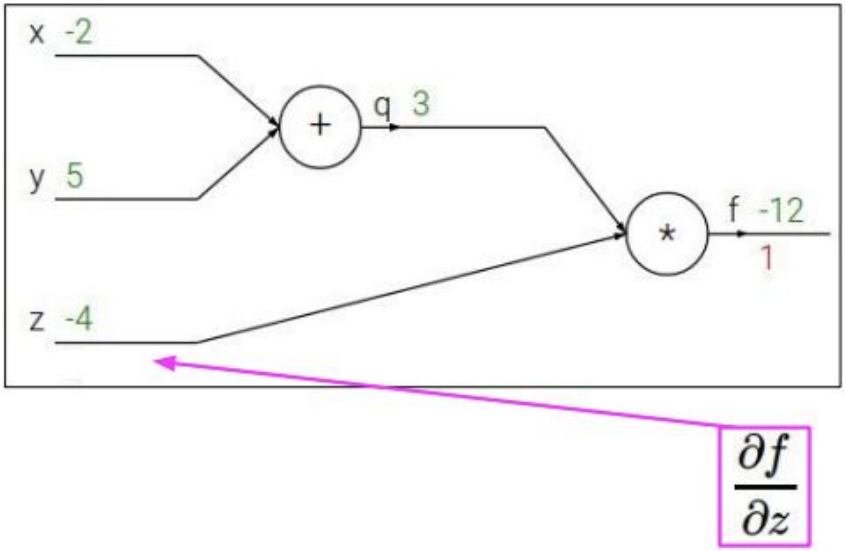
$$f(x, y, z) = (x + y)z$$

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

# Backpropagation example

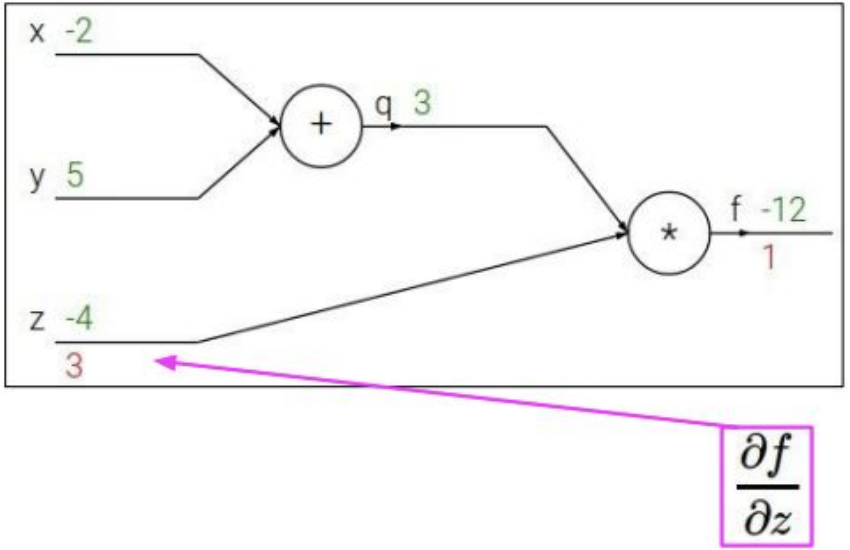
$$f(x, y, z) = (x + y)z$$

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

# Backpropagation example

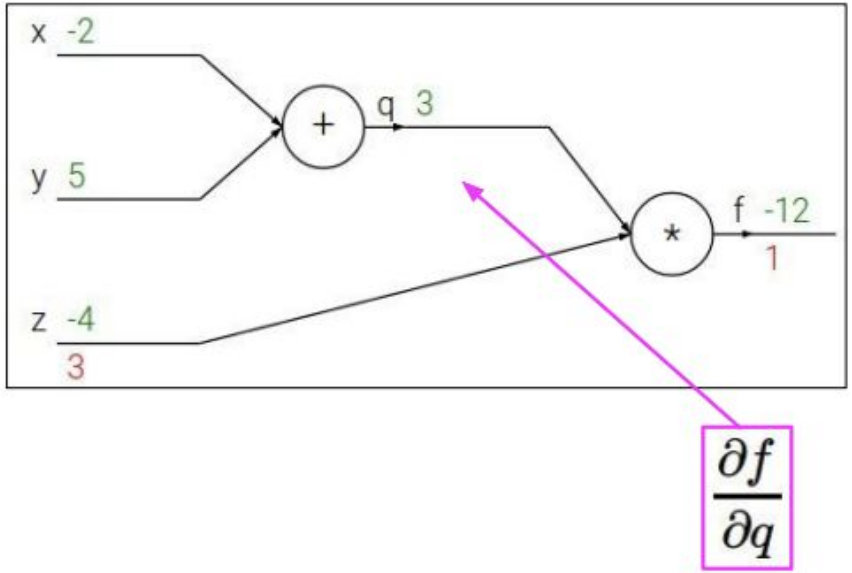
$$f(x, y, z) = (x + y)z$$

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

# Backpropagation example

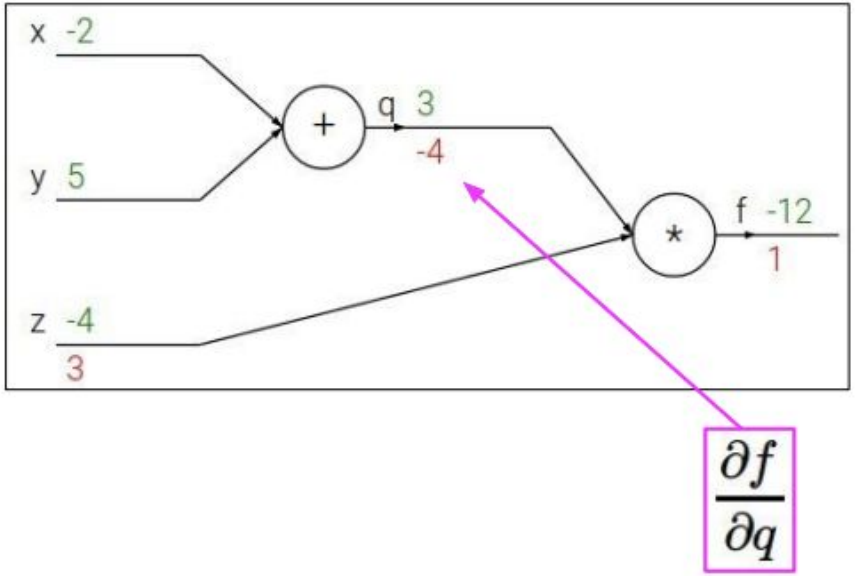
$$f(x, y, z) = (x + y)z$$

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

# Backpropagation example

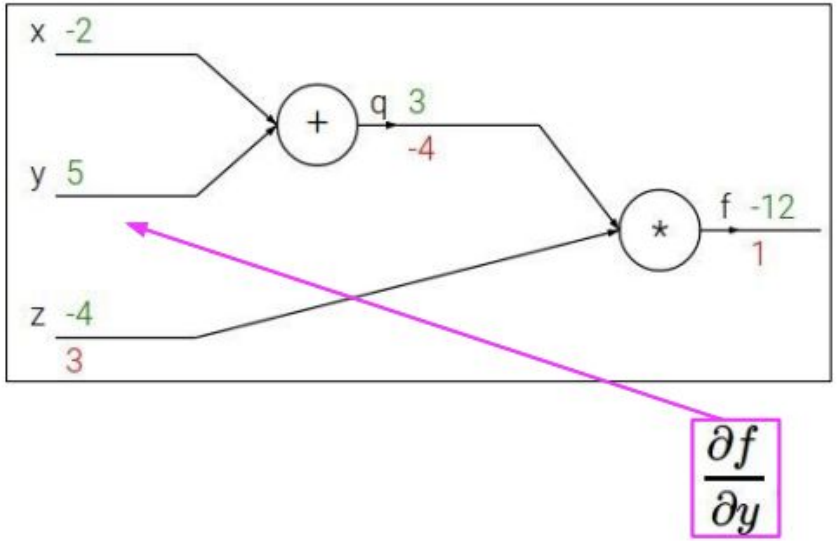
$$f(x, y, z) = (x + y)z$$

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

# Backpropagation example

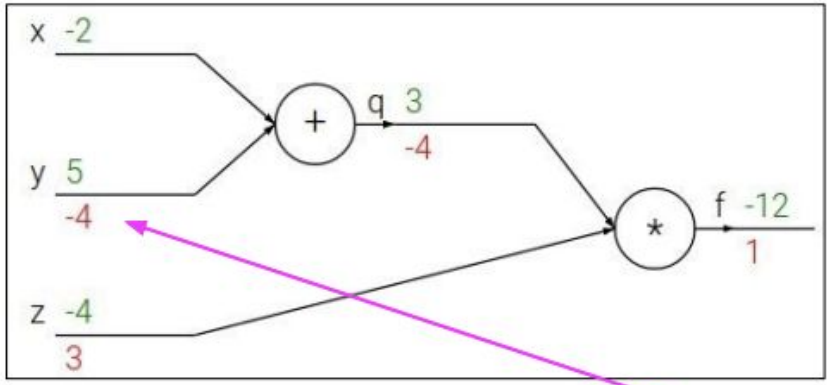
$$f(x, y, z) = (x + y)z$$

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:

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

source

# Backpropagation example

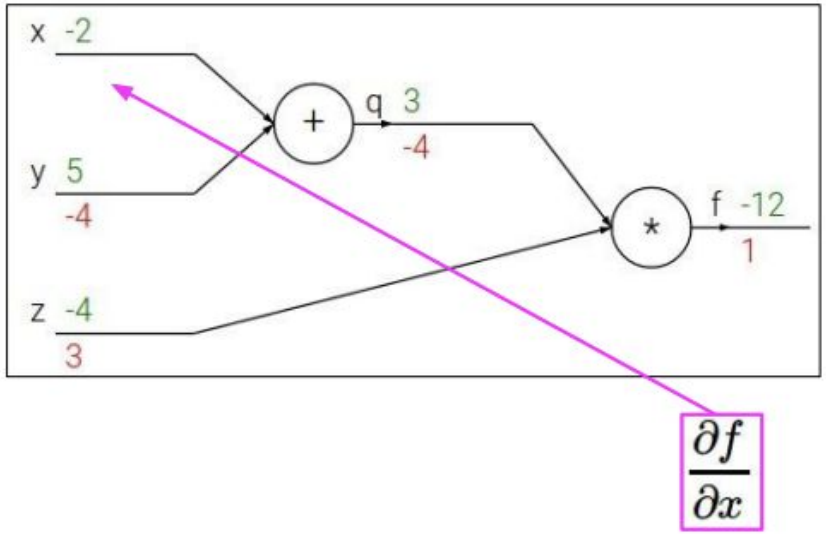
$$f(x, y, z) = (x + y)z$$

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

# Backpropagation example

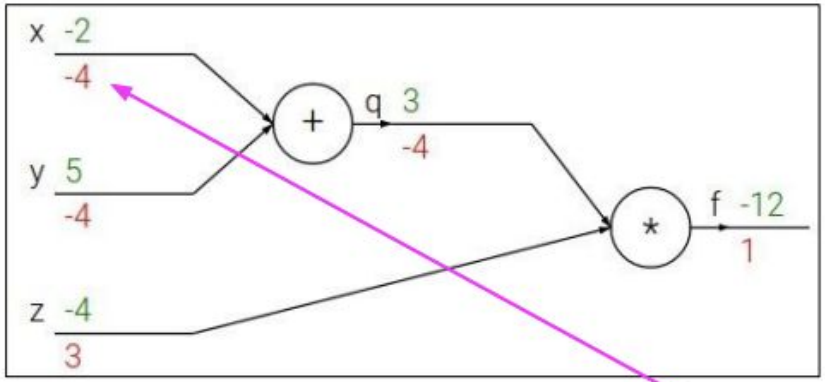
$$f(x, y, z) = (x + y)z$$

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:

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

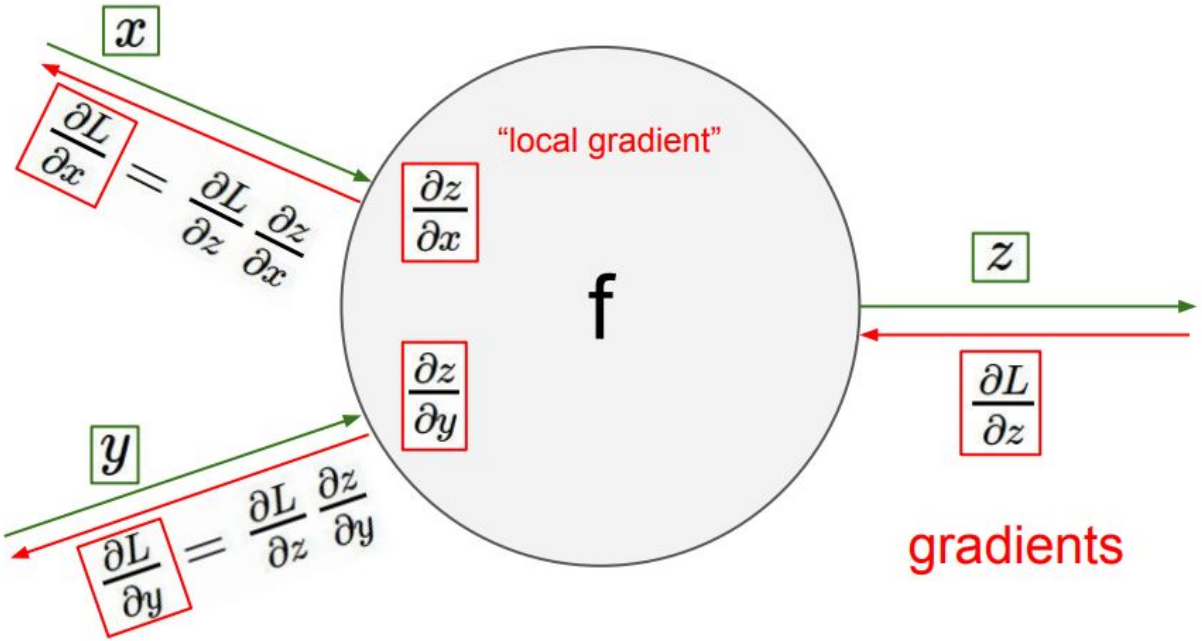
source

# Backpropagation and chain rule

Chain rule is just simple math:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

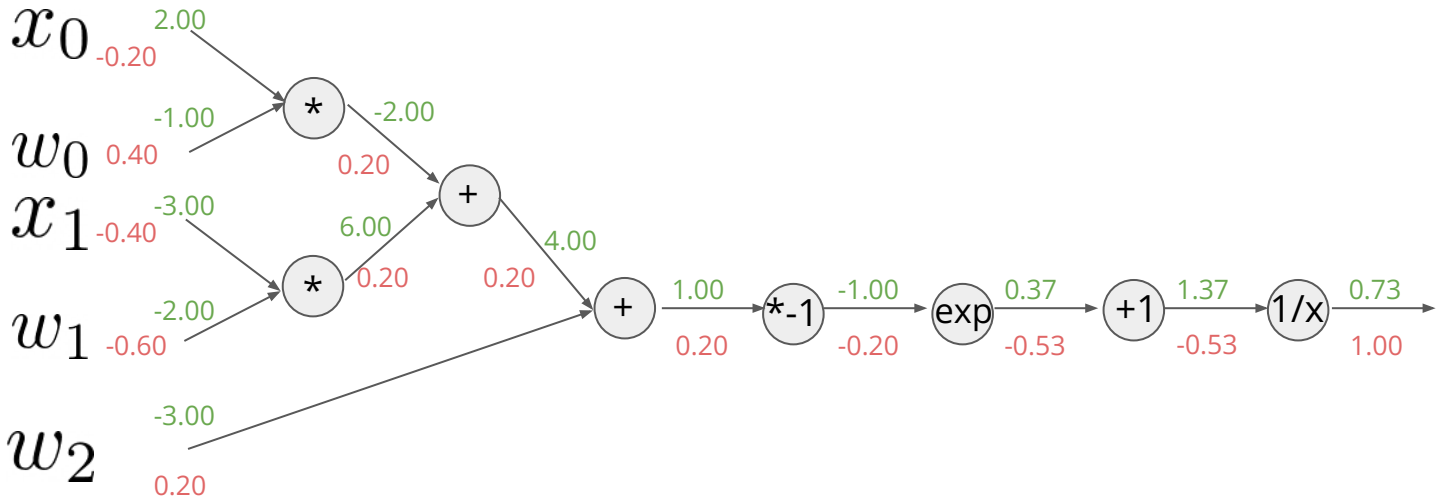
Backprop is just way to use it in NN training.




source

# Backpropagation example

$$L(w,x) = \frac{1}{1+\exp(-(x_0 w_0 + x_1 w_1 + w_2))}$$




source

# Backpropagation: matrix form

$$
\begin{aligned}
y_1 &= f_1(\mathbf{x}) &= x_1 \\
y_2 &= f_2(\mathbf{x}) &= x_2 \\
&\vdots \\
y_n &= f_n(\mathbf{x}) &= x_n
\end{aligned}
$$

$$
\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \nabla f_1(\mathbf{x}) \\ \nabla f_2(\mathbf{x}) \\ \dots \\ \nabla f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}} f_1(\mathbf{x}) \\ \frac{\partial}{\partial \mathbf{x}} f_2(\mathbf{x}) \\ \dots \\ \frac{\partial}{\partial \mathbf{x}} f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\mathbf{x}) & \frac{\partial}{\partial x_2} f_1(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_1(\mathbf{x}) \\ \frac{\partial}{\partial x_1} f_2(\mathbf{x}) & \frac{\partial}{\partial x_2} f_2(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_2(\mathbf{x}) \\ & & \dots & \\ \frac{\partial}{\partial x_1} f_m(\mathbf{x}) & \frac{\partial}{\partial x_2} f_m(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_m(\mathbf{x}) \end{bmatrix}
$$

source

# Backpropagation: matrix form

| | scalar $x$ | vector $\mathbf{x}$ |
|---|---|---|
| scalar $f$ | $\frac{\partial f}{\partial x}$ | $\frac{\partial f}{\partial \mathbf{x}}$ |
| vector $\mathbf{f}$ | $\frac{\partial \mathbf{f}}{\partial x}$ | $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ |

source

# Backpropagation: matrix form

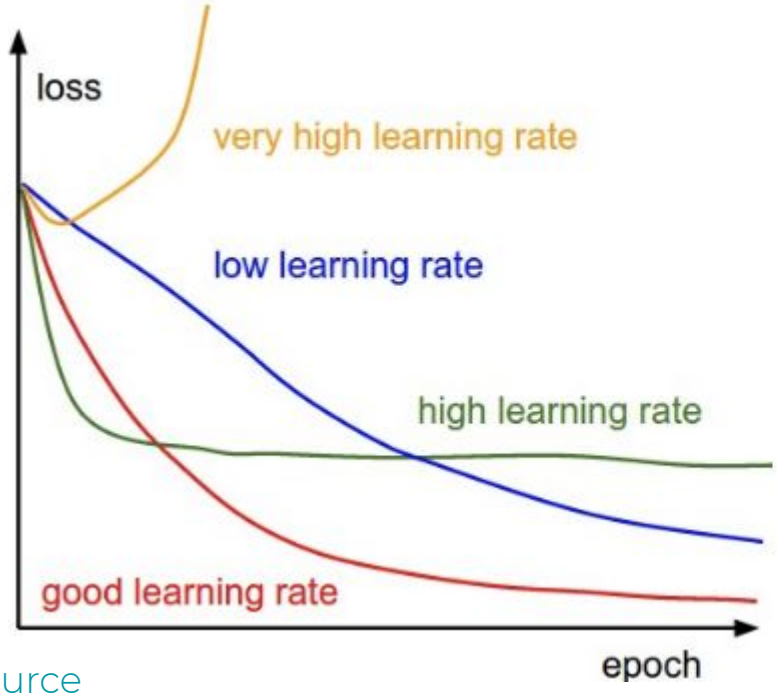
$$
\begin{aligned}
\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}} f_1(\mathbf{x}) \\ \frac{\partial}{\partial \mathbf{x}} f_2(\mathbf{x}) \\ \dots \\ \frac{\partial}{\partial \mathbf{x}} f_m(\mathbf{x}) \end{bmatrix} &= \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\mathbf{x}) & \frac{\partial}{\partial x_2} f_1(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_1(\mathbf{x}) \\ \frac{\partial}{\partial x_1} f_2(\mathbf{x}) & \frac{\partial}{\partial x_2} f_2(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_2(\mathbf{x}) \\ & & \dots & \\ \frac{\partial}{\partial x_1} f_m(\mathbf{x}) & \frac{\partial}{\partial x_2} f_m(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_m(\mathbf{x}) \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & \frac{\partial}{\partial x_2} x_1 & \dots & \frac{\partial}{\partial x_n} x_1 \\ \frac{\partial}{\partial x_1} x_2 & \frac{\partial}{\partial x_2} x_2 & \dots & \frac{\partial}{\partial x_n} x_2 \\ & & \dots & \\ \frac{\partial}{\partial x_1} x_n & \frac{\partial}{\partial x_2} x_n & \dots & \frac{\partial}{\partial x_n} x_n \end{bmatrix} \\
&\quad \text{(and since } \frac{\partial}{\partial x_j} x_i = 0 \text{ for } j \neq i\text{)} \\
&= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & 0 & \dots & 0 \\ 0 & \frac{\partial}{\partial x_2} x_2 & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & \frac{\partial}{\partial x_n} x_n \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & 1 \end{bmatrix} \\
&= I \quad (I \text{ is the identity matrix with ones down the diagonal})
\end{aligned}
$$

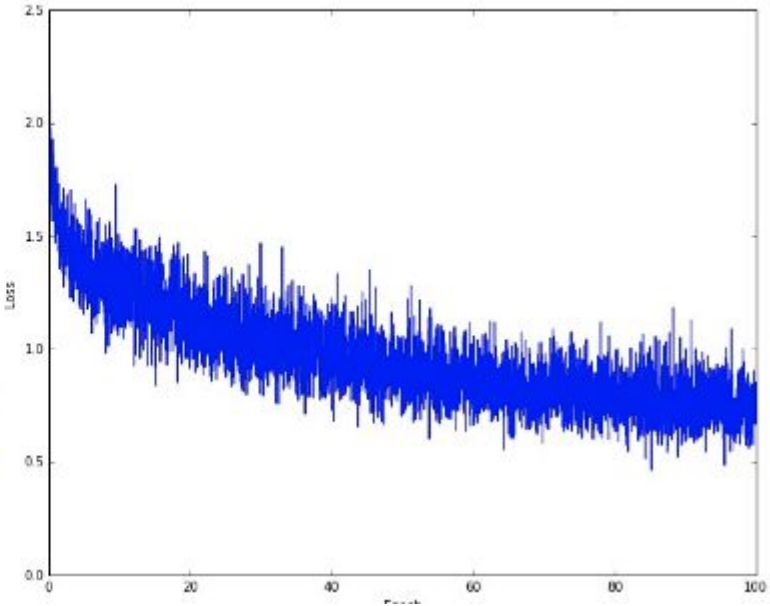
source

# Gradient optimization

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

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




source

# Activation functions

girafe
ai

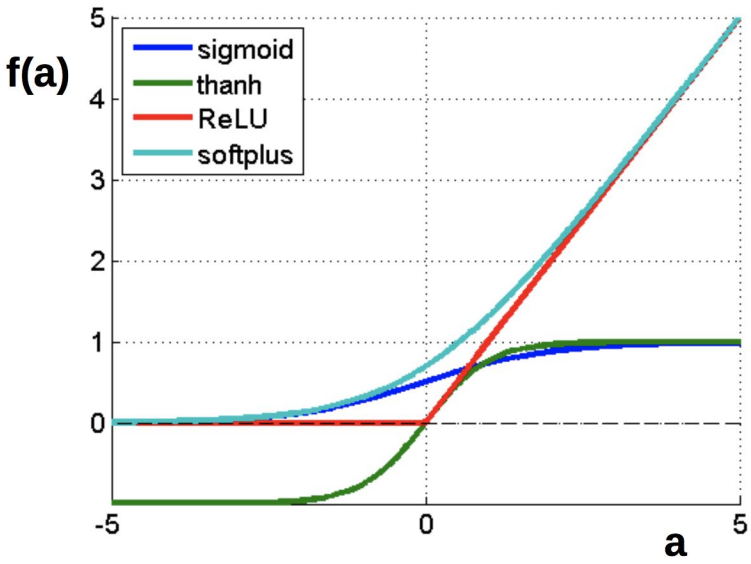
# Once more: nonlinearities

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

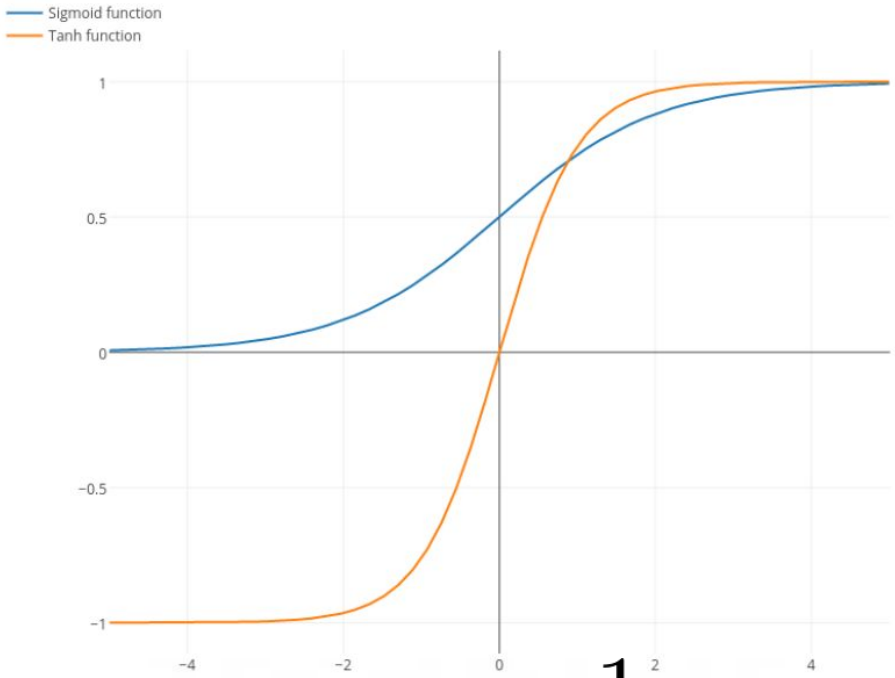
$$f(a) = \tanh(a)$$

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

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



# Activation functions: Sigmoid



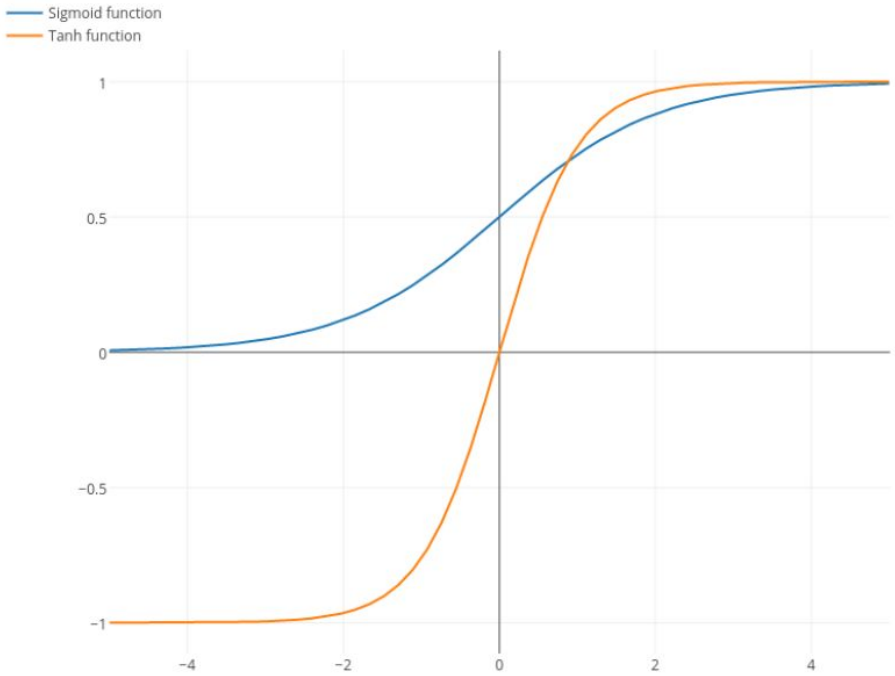

$$f(a) = \frac{1}{1 + e^{-a}}$$

- Maps R to (0,1)
- Historically popular, one of the first approximations of neuron activation

Problems:

- Almost zero gradients on the both sides (saturation)
- Shifted (not zero-centered) output
- Expensive computation of the exponent

# Activation functions: tanh



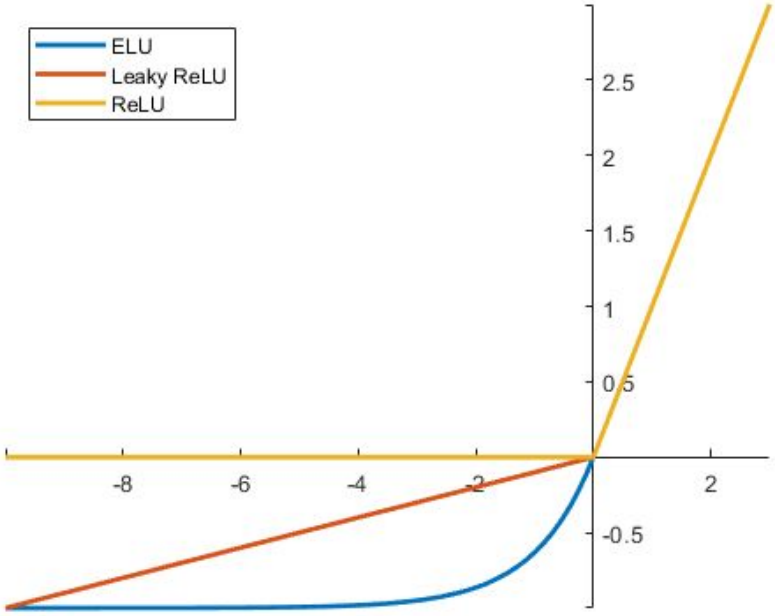

$$f(a) = \tanh(a)$$

- Maps R to (-1,1)
- Similar to the Sigmoid in other ways

Problems:

- Almost zero gradients on the both sides (saturation)
- ~~Shifted (not zero-centered) output~~
- Expensive computation of the exponent

# Activation functions: ELU



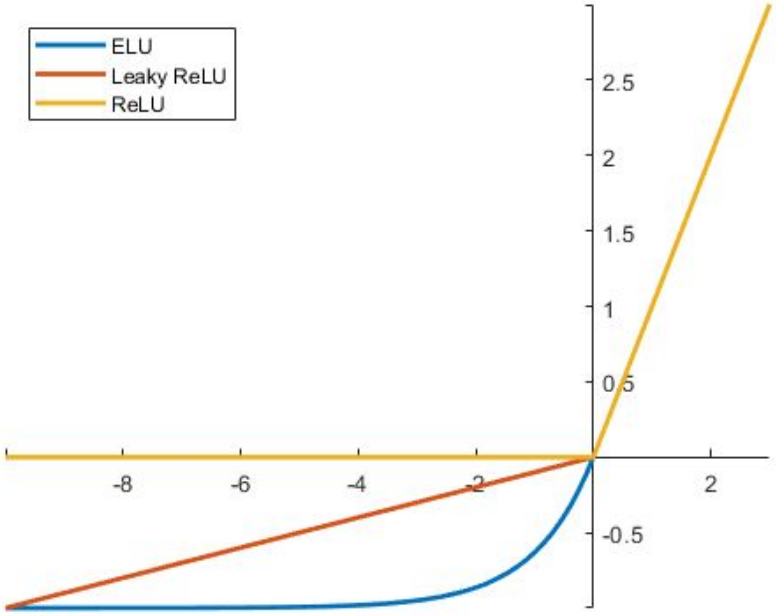

$$f(a) = \max(0, a)$$

- Very simple to compute (both forward and backward)
  - Up to 6 times faster than Sigmoid
- Does not saturate when x > 0
  - So the gradients are not 0

Problems:

- Zero gradients when x < 0
- Shifted (not zero-centered) output

# Activation functions: ELU



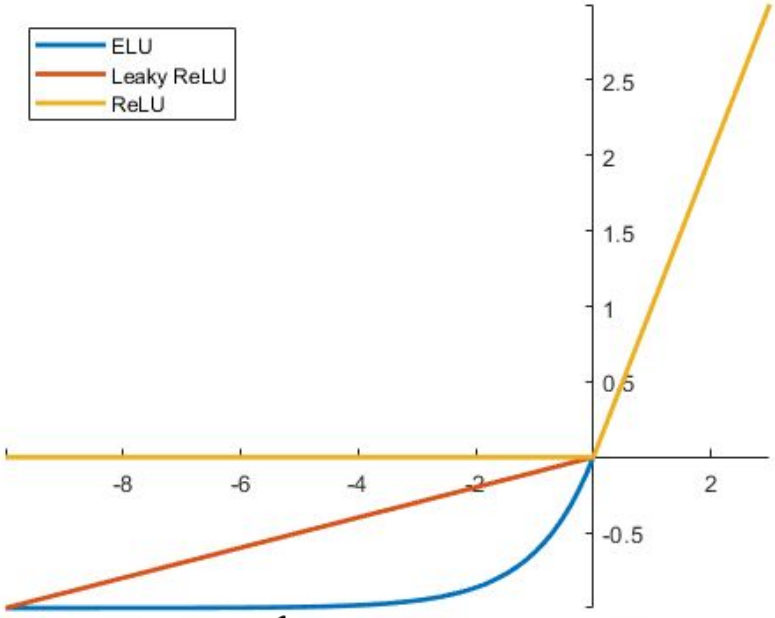

- Very simple to compute (both forward and backward)
  - Up to 6 times faster than Sigmoid
- Does not saturate when

Problems:

- Shifted, but not so much output

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

# Activation functions: ELU




- Similar to ReLU
- Does not saturate
- Close to zero mean outputs

Problems:

- Requires exponent computation

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

# 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

# Fancy neural networks

girafe
ai

# RNN

Shakespeare

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

Algebraic Geometry (Latex)

*Proof.* Omitted. □

**Lemma 0.1.** *Let $\mathcal{C}$ be a set of the construction.*

*Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that*

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

*Proof.* This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

**Lemma 0.2.** *This is an integer $\mathcal{Z}$ is injective.*

*Proof.* See Spaces, Lemma ??. □

**Lemma 0.3.** *Let $S$ be a scheme. Let $X$ be a scheme and $X$ is an affine open covering. Let $\mathcal{U} \subset X$ be a canonical and locally of finite type. Let $X$ be a scheme. Let $X$ be a scheme which is equal to the formal complex.*

*The following to the construction of the lemma follows.*

*Let $X$ be a scheme. Let $X$ be a scheme covering. Let*

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

*be a morphism of algebraic spaces over $S$ and $Y$.*

*Proof.* Let $X$ be a nonzero scheme of $X$. Let $X$ be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over $S$.
(2) If $X$ is an affine open covering.

Consider a common structure on $X$ and $X$ the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

Linux kernel (source code)

```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
  unsigned long flags;
  int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
  buf[0] = 0xFFFFFFFF & (bit << 4);
  min(inc, slist->bytes);
  printk(KERN_WARNING "Memory allocated %02x/%02x, "
    "original MLL instead\n"),
    min(min(multi_run - s->len, max) * num_data_in),
    frame_pos, sz + first_seg);
  div_u64_w(val, inb_p);
  spin_unlock(&disk->queue_lock);
  mutex_unlock(&s->sock->mutex);
  mutex_unlock(&func->mutex);
  return disassemble(info->pending_bh);
}
```

source

*Proof.* Omitted. □

**Lemma 0.1.** *Let $\mathcal{C}$ be a set of the construction.*

*Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that*

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

*Proof.* This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

**Lemma 0.2.** *This is an integer $\mathcal{Z}$ is injective.*

*Proof.* See Spaces, Lemma ??. □

**Lemma 0.3.** *Let $S$ be a scheme. Let $X$ be a scheme and $X$ is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let $X$ be a scheme. Let $X$ be a scheme which is equal to the formal complex.*

*The following to the construction of the lemma follows.*

*Let $X$ be a scheme. Let $X$ be a scheme covering. Let*

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

*be a morphism of algebraic spaces over $S$ and $Y$.*

*Proof.* Let $X$ be a nonzero scheme of $X$. Let $X$ be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over $S$.
(2) If $X$ is an affine open covering.

Consider a common structure on $X$ and $X$ the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{tikzcd}
S \arrow[r] \arrow[d] & & \\
\xi \arrow[r] & \mathcal{O}_{X'} \arrow[dr] & \\
\mathrm{gor}_s \arrow[u] & & \\
 & = \alpha' \arrow[r] \arrow[d, leftrightarrow] & \\
 & = \alpha' \arrow[r] & \alpha \\
 & \mathrm{Spec}(K_\psi) & \mathrm{Mor}_{Sets} \quad \mathrm{d}(\mathcal{O}_{X_{X/k}}, \mathcal{G}) \arrow[u, "X"']
\end{tikzcd}$$

is a limit. Then $\mathcal{G}$ is a finite type and assume $S$ is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type $f_*$. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

*Proof.* We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of $X$ is an open neighbourhood of $U$. □

*Proof.* This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that $U$ is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_t}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_t}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that $X$ is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over $S$.
If $\mathcal{F}$ is a scheme theoretic image points. □

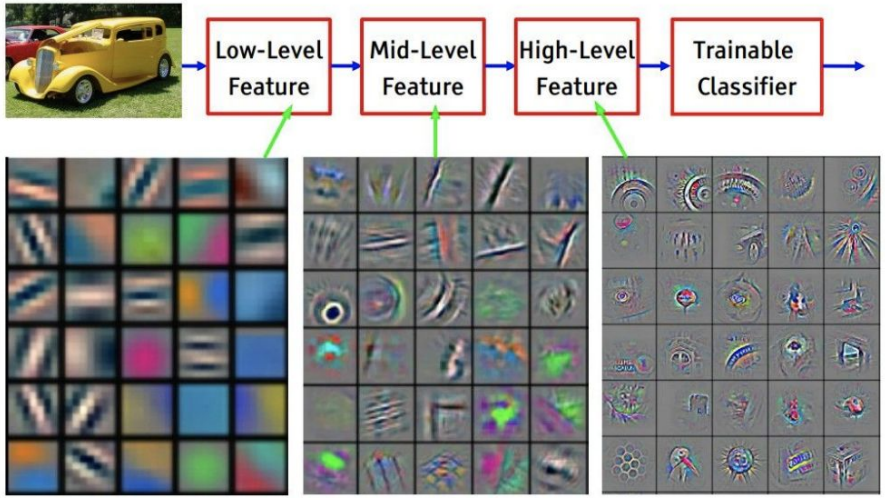
If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

source

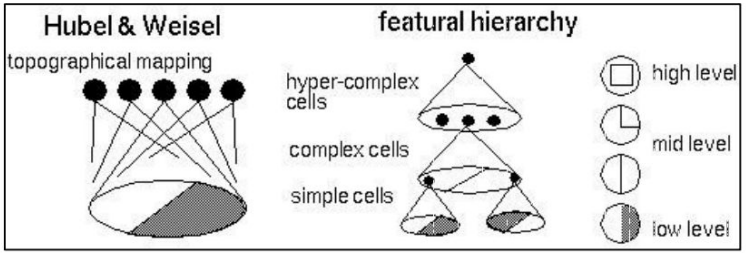
```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>

#define REG_PG    vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)     (func)

#define SWAP_ALLOCATE(nr)     (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %%esp, %0, %3" : : "r" (0));   \
  if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
          pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
  PUT_PARAM_RAID(2, sel) = get_state_state();
  set_pid_sum((unsigned long)state, current_state_str(),
           (unsigned long)-1->lr_full; low;
}
```

source

# CNN:Convolutional layer and visual cortex




Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

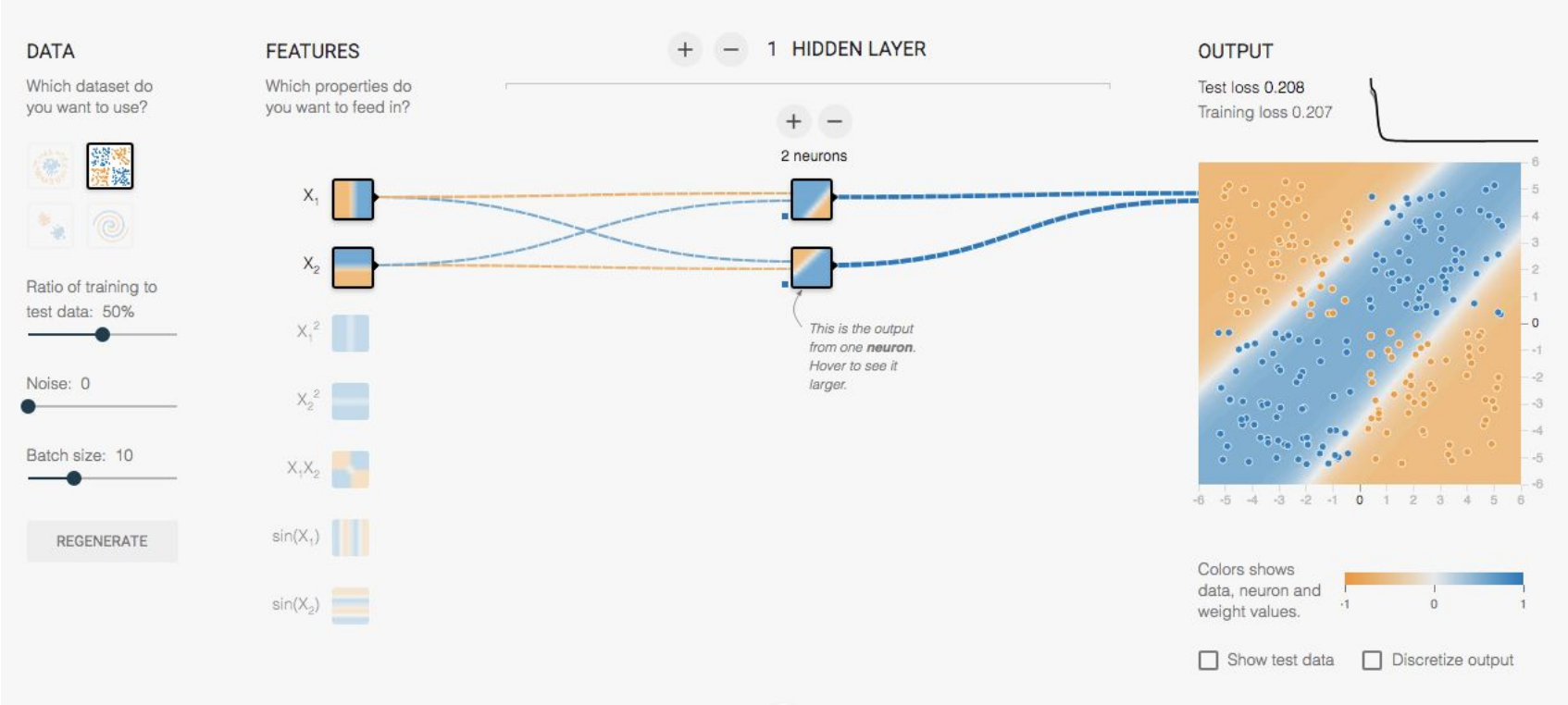



[From Yann LeCun slides]

# CNN:Convolutional layer and visual cortex

source

# Don’t miss the interactive playground




source

WHO'S AWESOME?
You're Awesome

# Outro

- Neural Networks are great
  - Especially for data with specific structure
- All operations should be differentiable to use backpropagation mechanics
  - And still it is just basic differentiation
- Many techniques in Deep Learning are inspired by nature
  - Or general sense
- Do not hesitate to ask questions (and answer them as well)

More materials for self-study: link

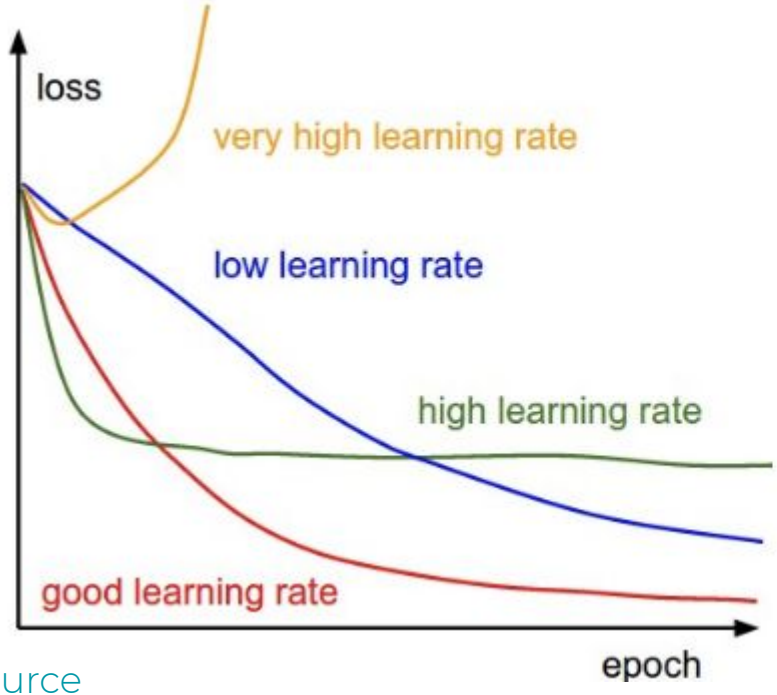
# Backup

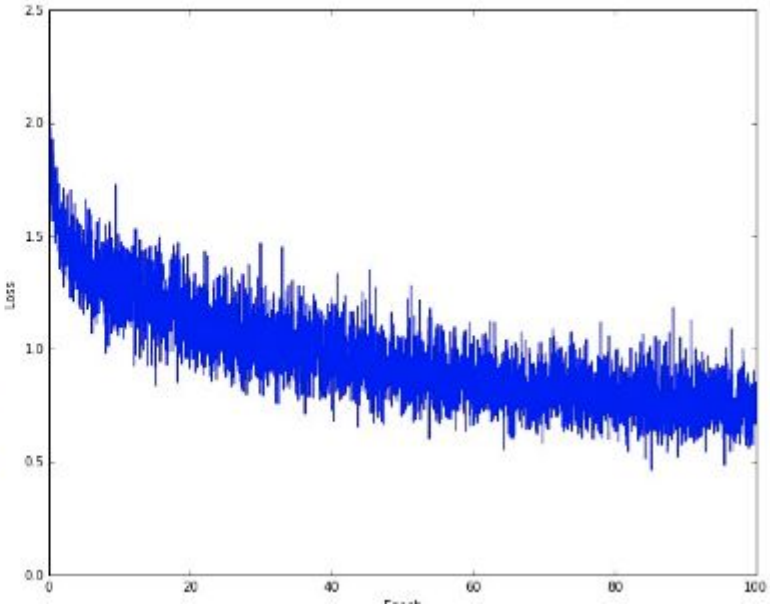
girafe
ai

# Optimizers

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

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



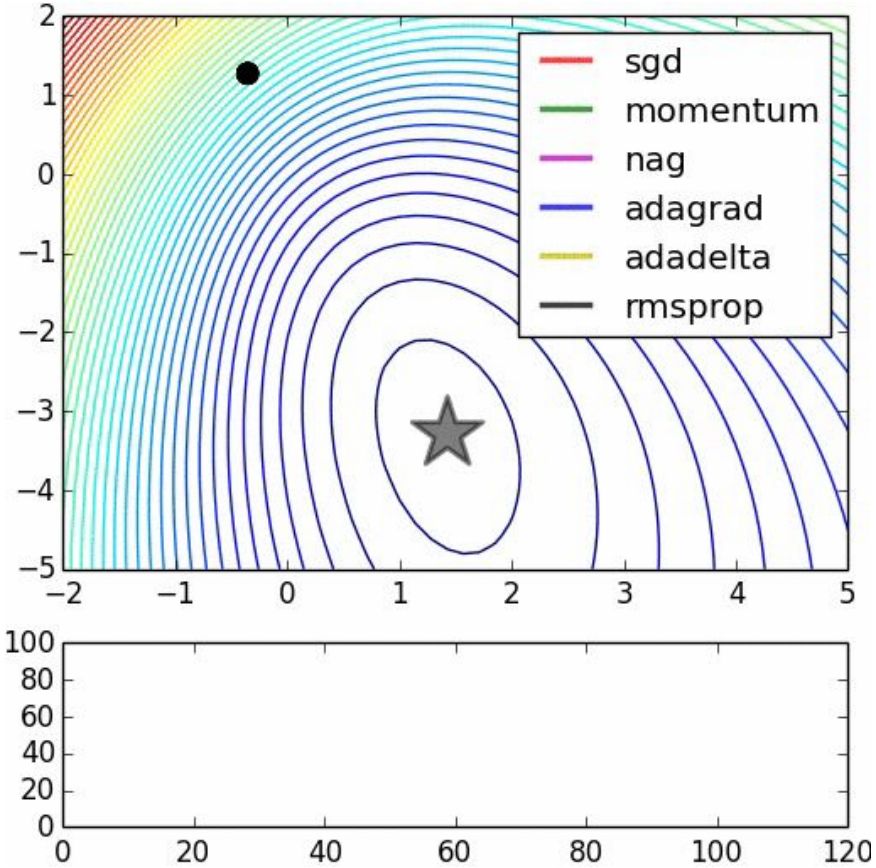




source

# Optimizers

There are much more optimizers:

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

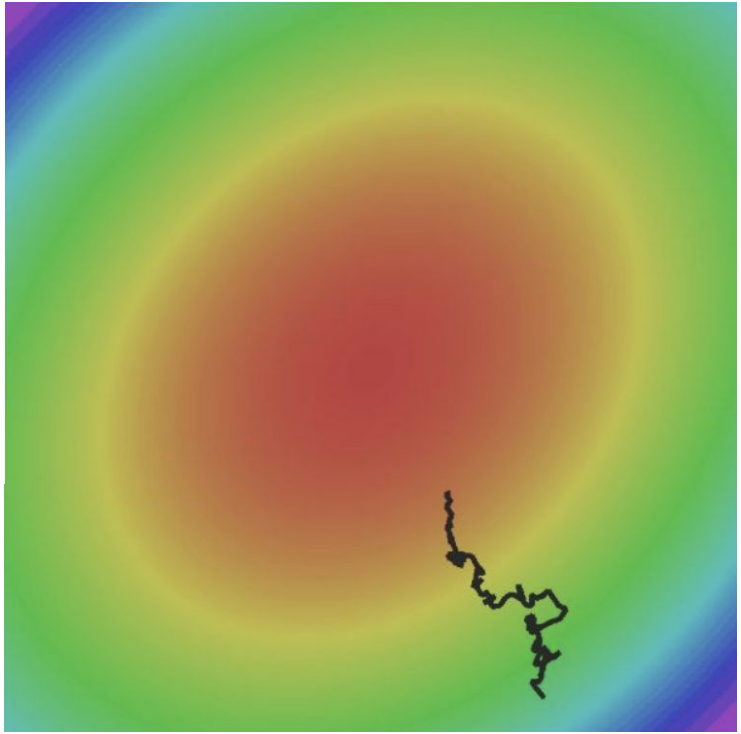


# Optimization: SGD

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

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

Averaging over minibatches ->noisy gradient



source

# First idea: momentum

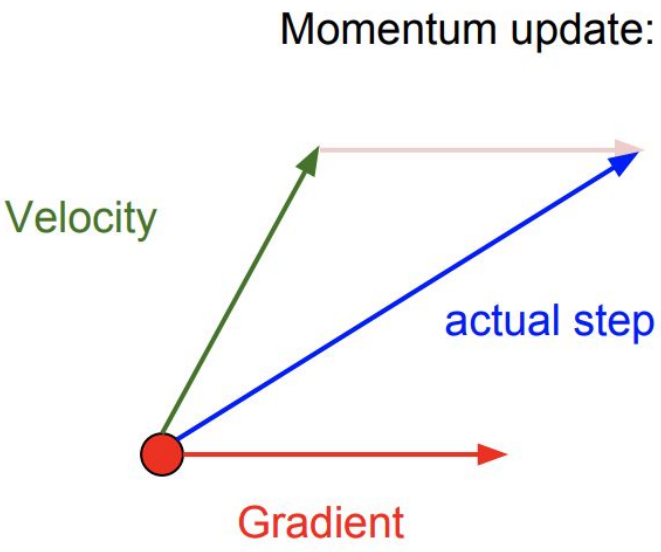
Simple SGD

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

SGD with momentum

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

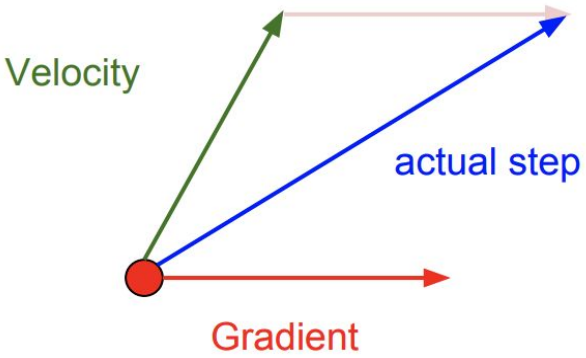
$$x_{t+1} = x_t - \alpha v_{t+1}$$




source

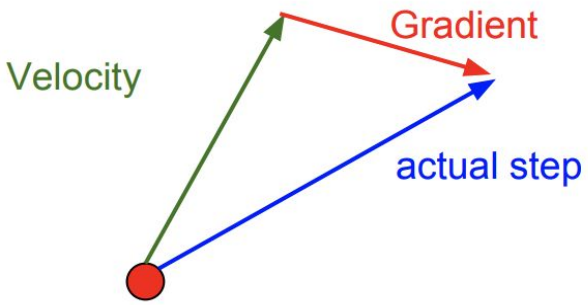
# Nesterov momentum

Momentum update:




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

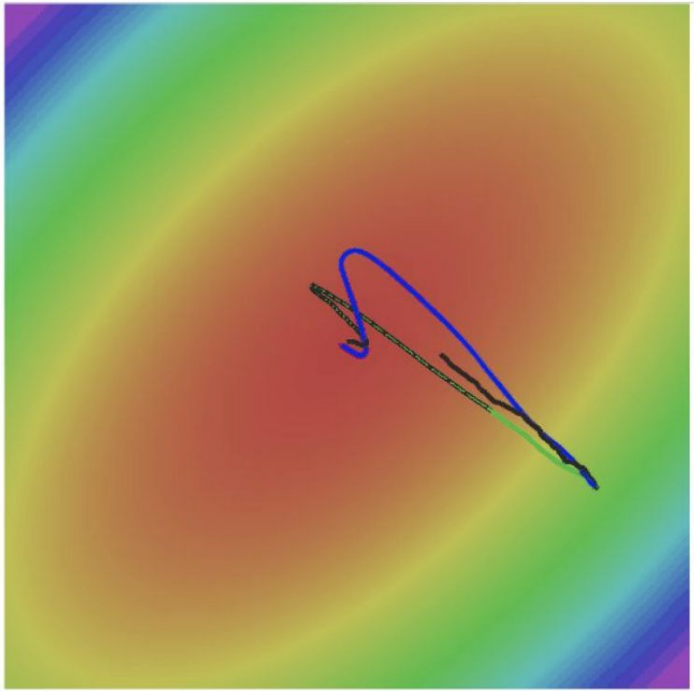
Nesterov Momentum




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

source

# Comparing momentums



SGD

SGD+Momentum

Nesterov

source

# 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

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

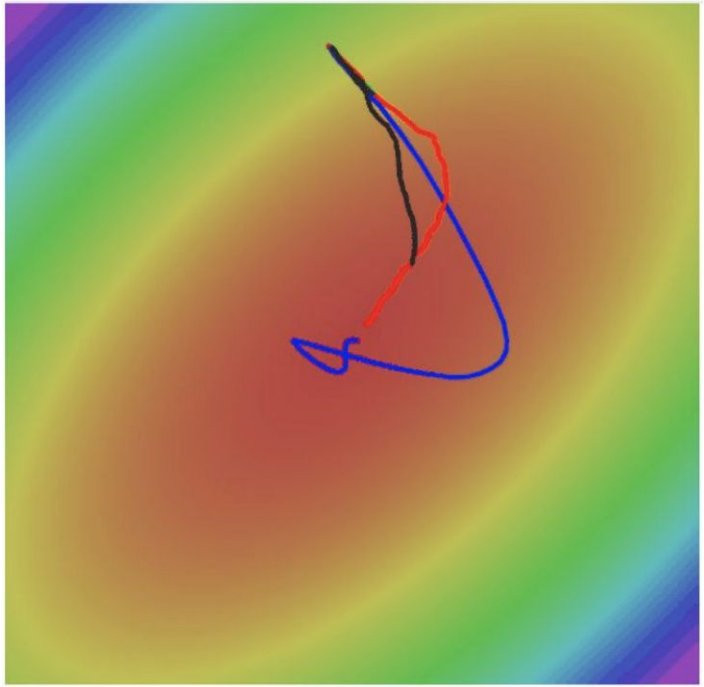
$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$

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



SGD

SGD+Momentum

RMSProp

source

# 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 link for more info.

# Adam

Let's combine the momentum idea and RMSProp normalization:

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

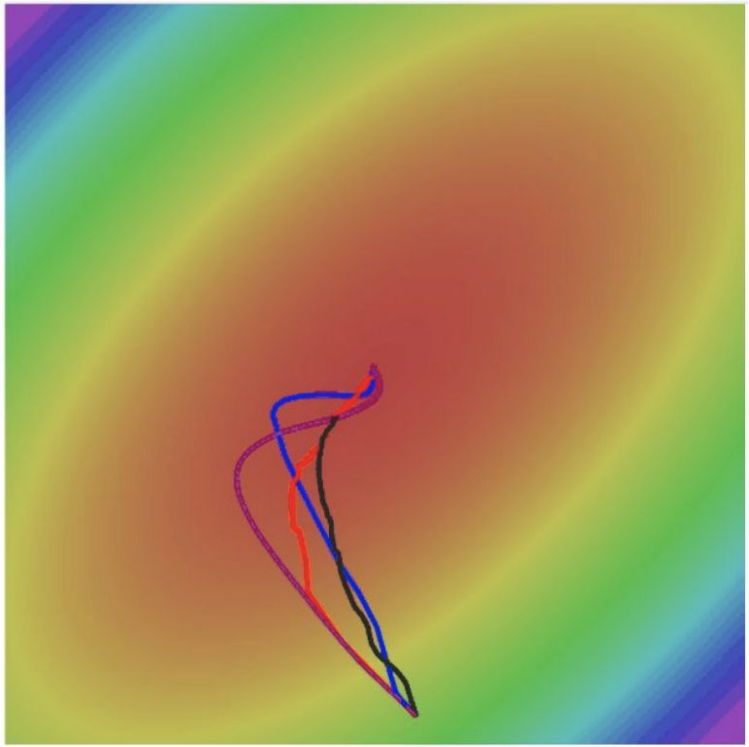
$$\text{cache}_{t+1} = \beta \text{cache}_t + (1-\beta)(\nabla f(x_t))^2$$

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

Actually, that's not quite Adam.

Adam full form involves bias correction term. See link for more info.

# Comparing optimizers



SGD

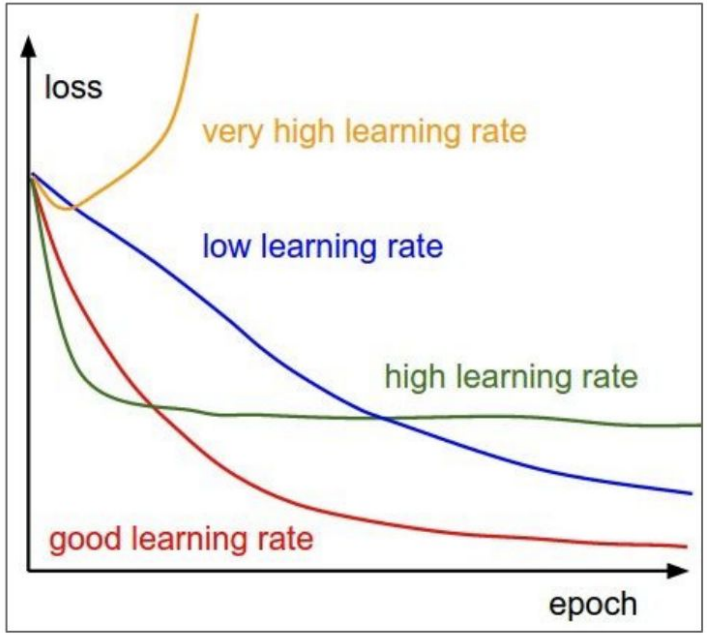
SGD+Momentum

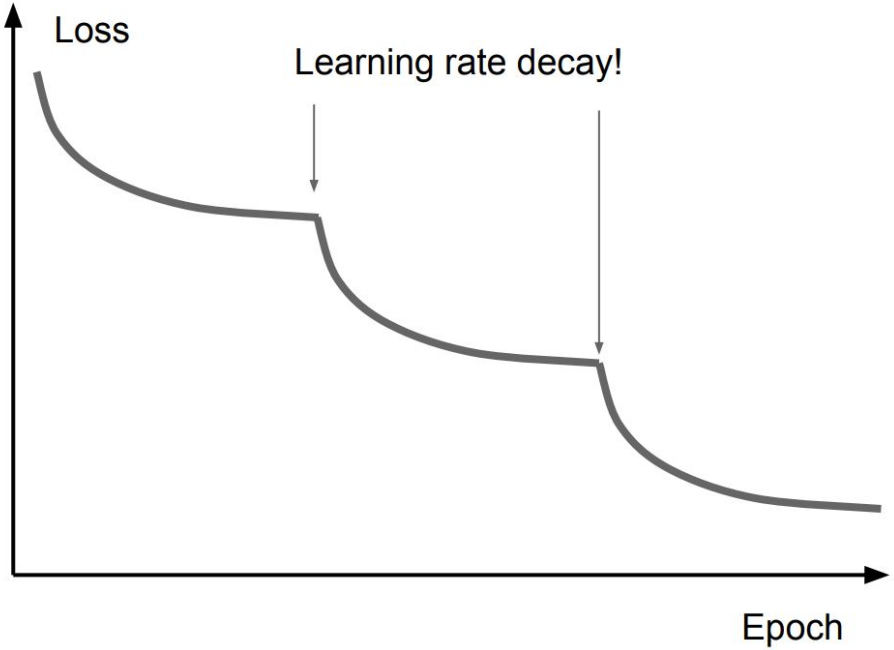
RMSProp

Adam

source

# Once more: learning rate

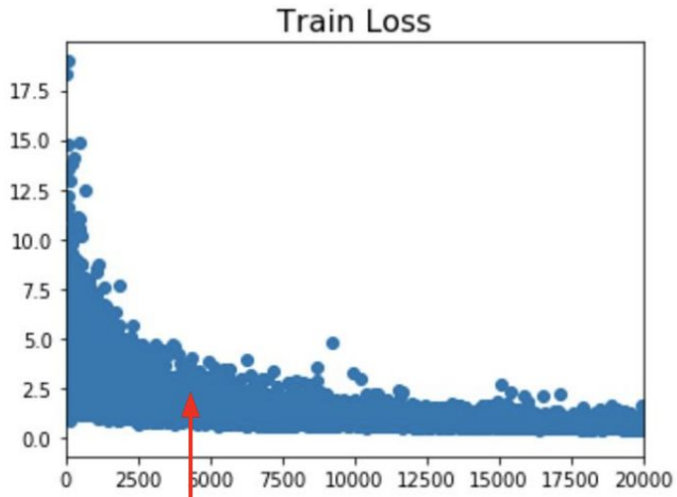


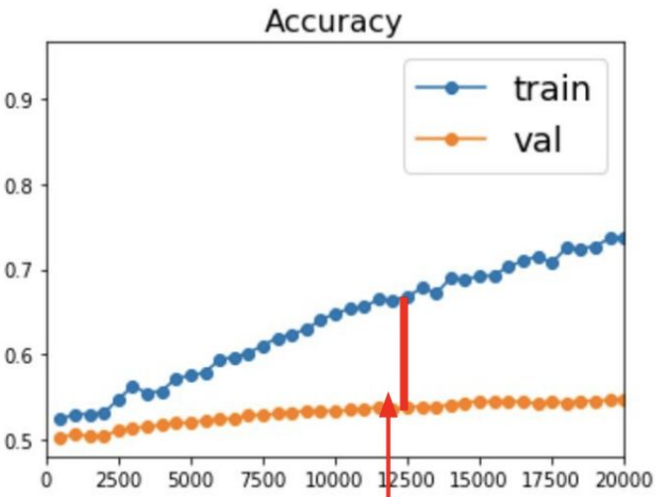




source

# Sum up: optimization

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



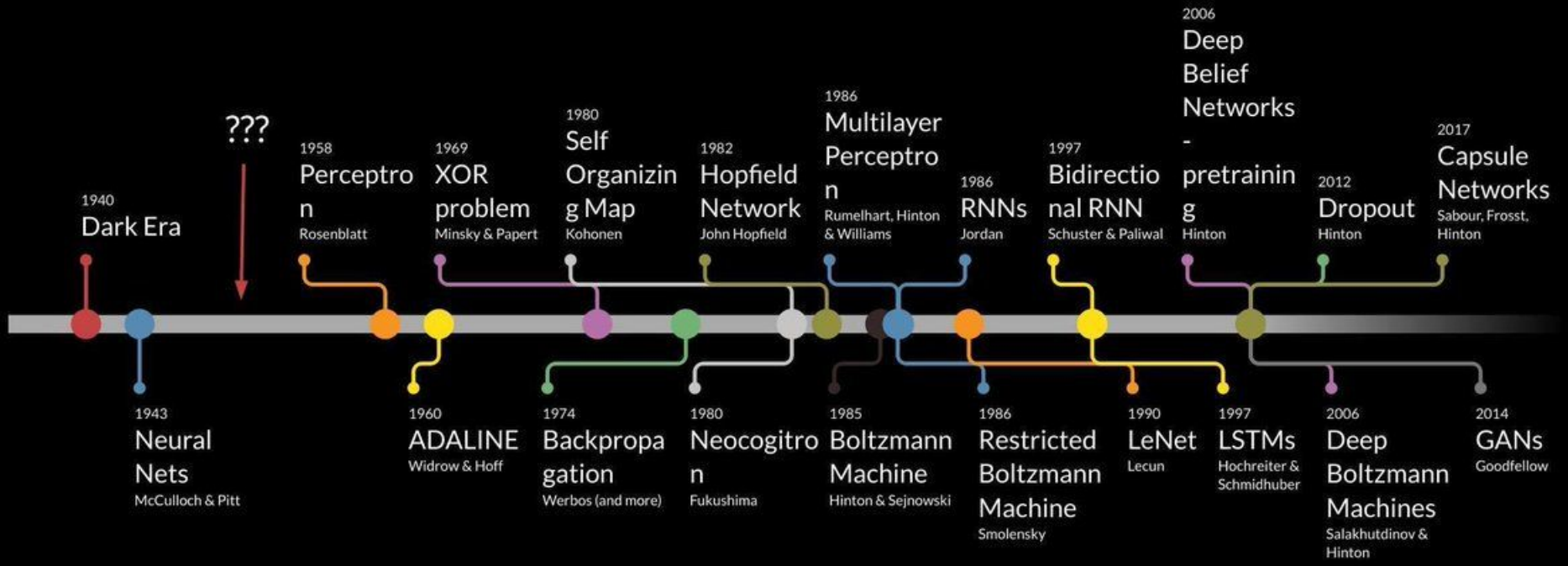




Better optimization algorithms help reduce training loss

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

source

# Deep Learning Timeline




Made by Favio Vázquez

# Revise

1. Neural Networks in different areas. Historical overview.
2. Backpropagation.
3. More on backpropagation.
4. Activation functions.
5. Playground.

# Thanks for attention!

Questions?

girafe ai