Neural Networks basics

Vladislav Goncharenko

girafe ai

ITMO, fall 2024

Questions

1. Какой размер имеет бутстрапированная выборка?
2. Чем Random Forest отличается от Bagging-а над деревьями решений?
3. Что является таргетом для следующего дерева в бустинге общем случае?
4. Что является таргетом для следующего дерева в бустинге в случае MSE?
5. Перечислите три библиотеки для градиентного бустинга
6. Аналитическое решение задачи линейной регрессии с L2 регуляризацией
7. Техники валидации модели: перечислить 3-5 известных способов
8. Что должна предсказывать модель чтобы мы могли подсчитать roc-auc?
9. Функция потерь для логистической регрессии

Outline

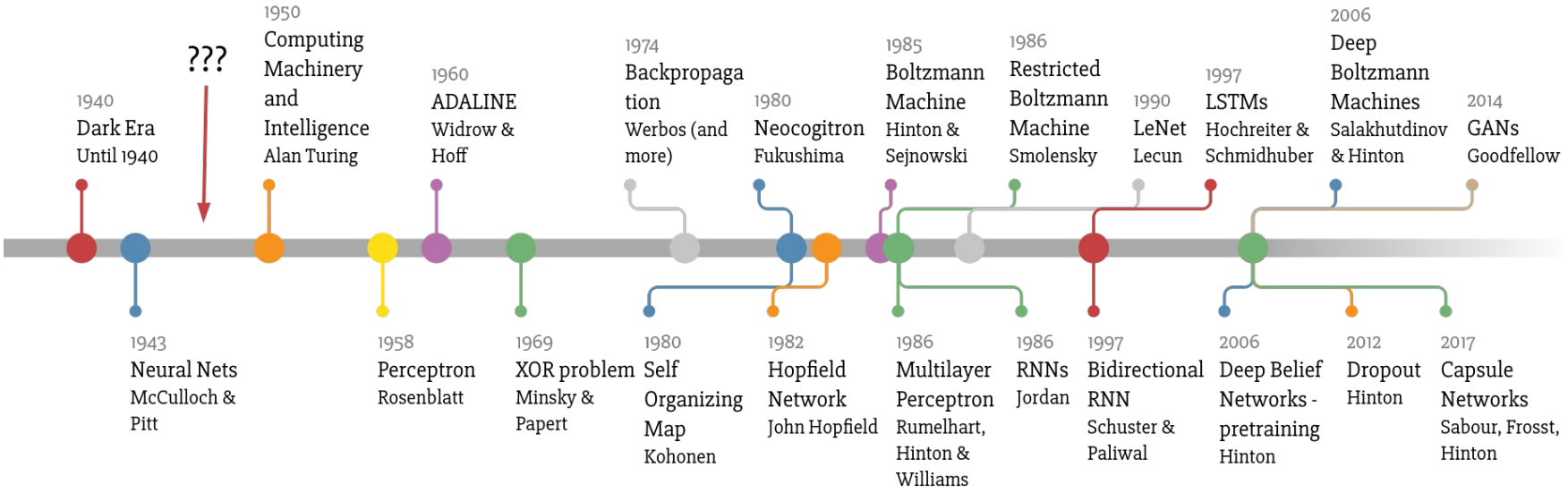
1. History of Deep Learning
2. Neural networks intuition
3. Linear layer
4. Activation functions
5. Cross-entropy
6. Backpropagation
7. Neural network libraries overview

History of Deep Learning

girafe ai

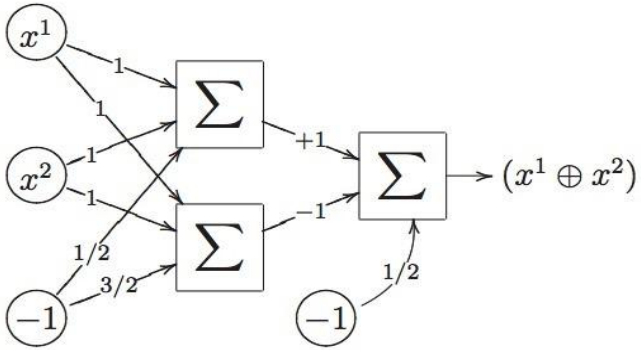
01

Deep Learning Timeline



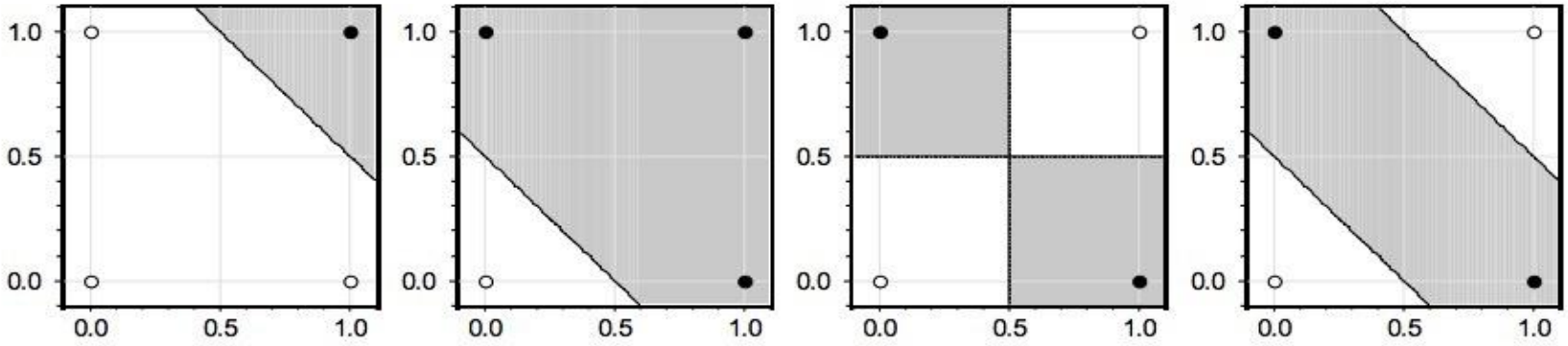
Made by Favio Vázquez

XOR problem



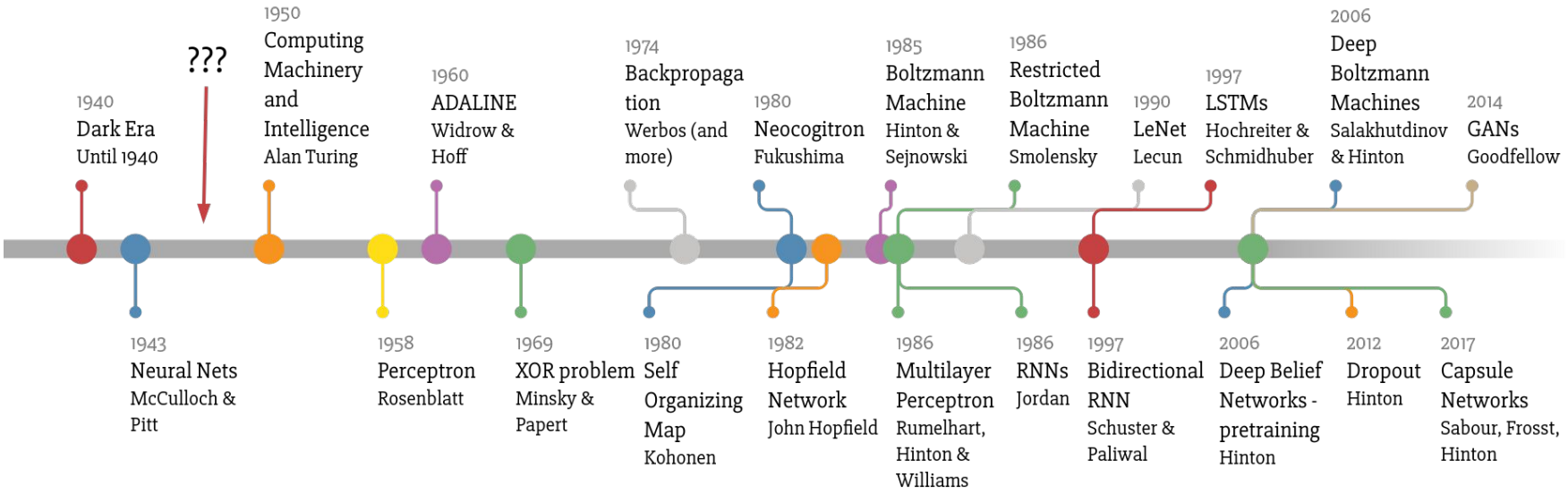
This 2-layer NN (on the left) implements XOR with only x1 and x2 features.

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



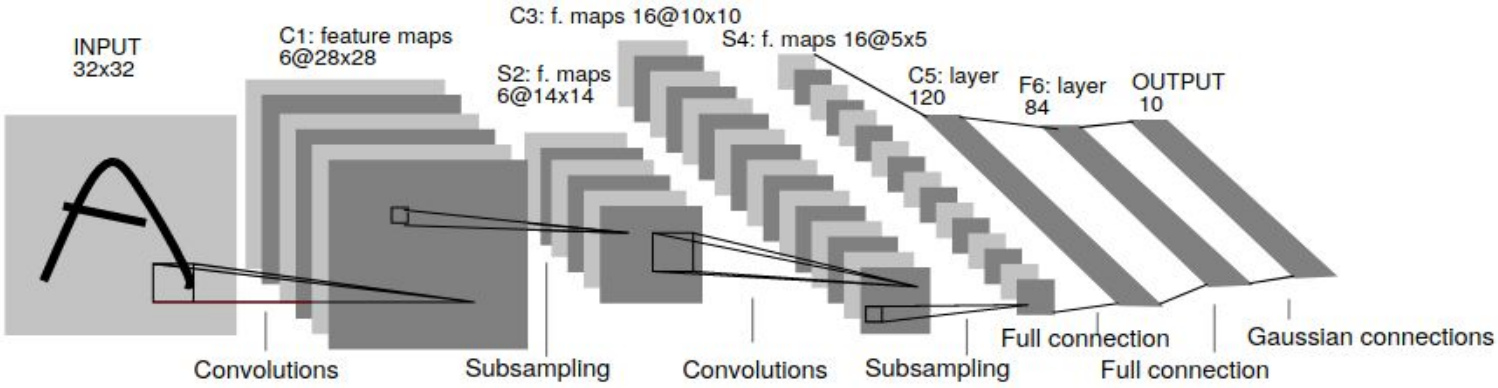
AND OR XOR(with x1*x2) XOR

Deep Learning Timeline

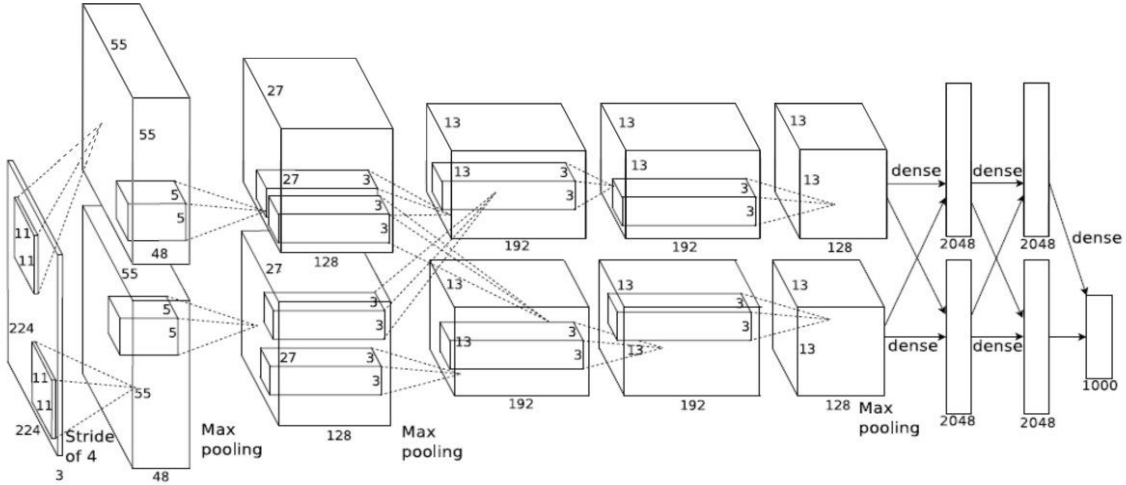


Made by Favio Vázquez

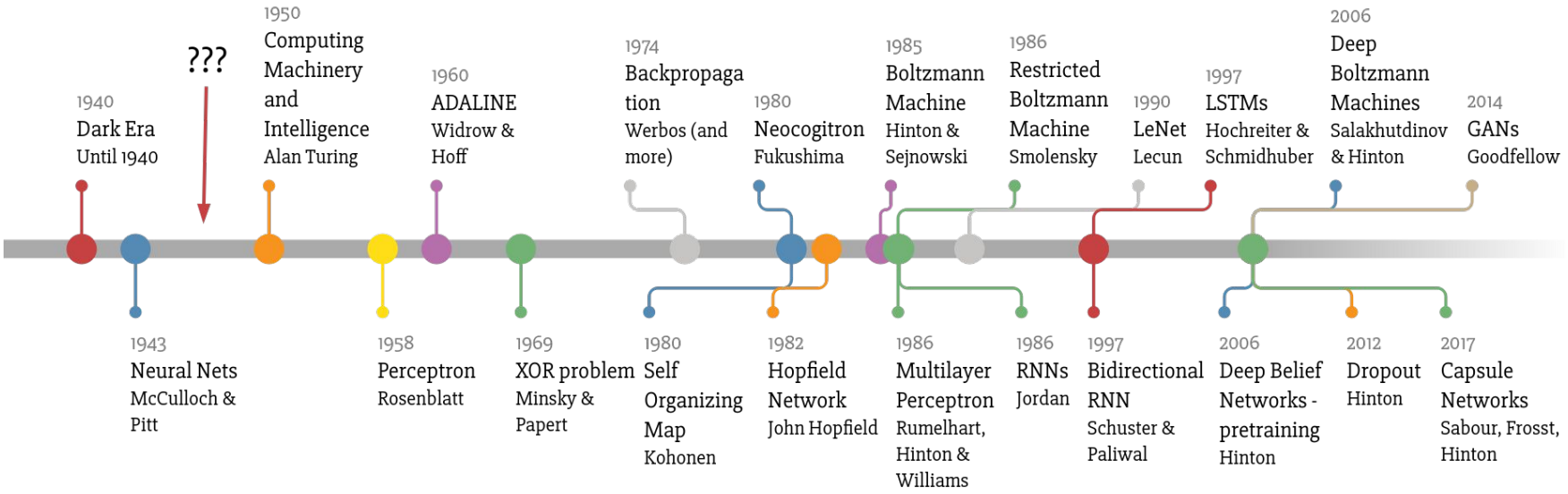
LeNet-5, 1998



AlexNet, 2012



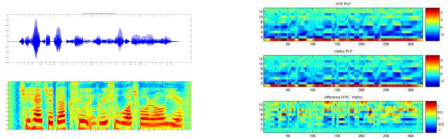
Deep Learning Timeline



Made by Favio Vázquez

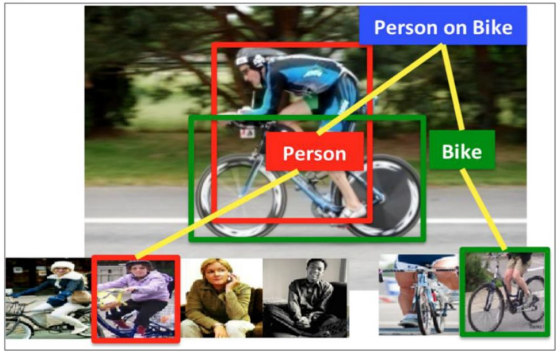
Real world applications

Audio Features



Spectrogram MFCC

- Object detection
- Action classification
- Image captioning
- …

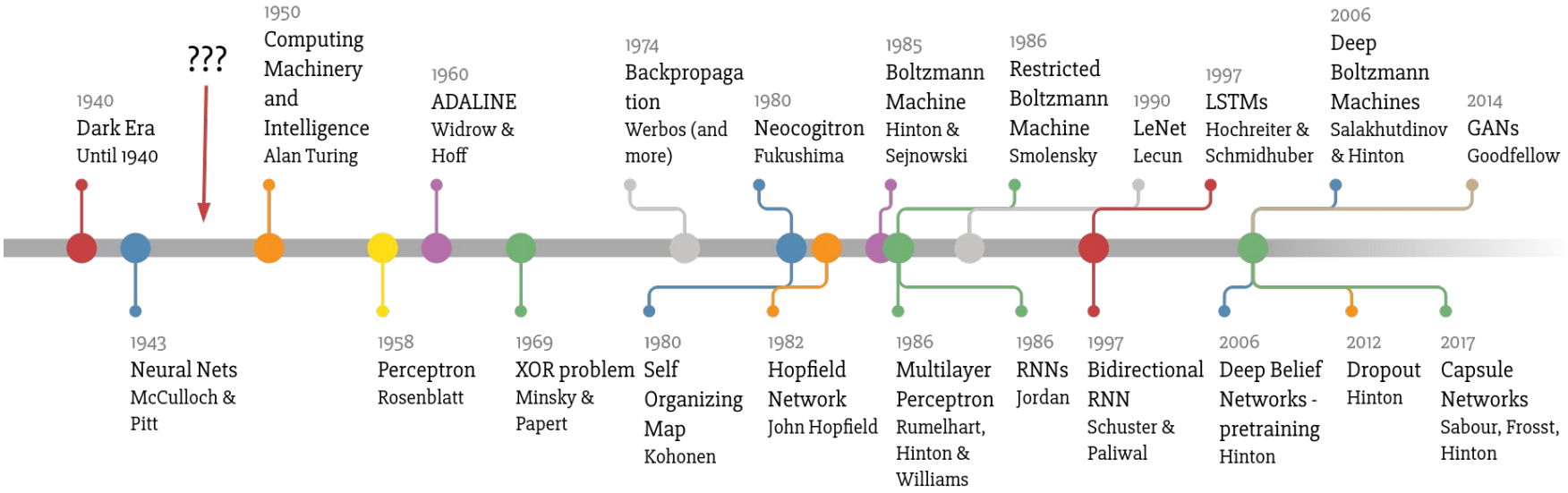


"man in black shirt is playing guitar."

GANs. 2014+

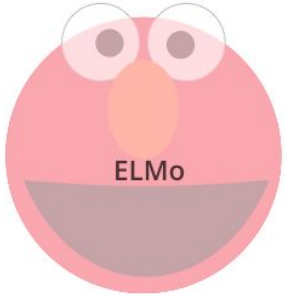
https://thispersondoesnotexist.com/

Deep Learning Timeline

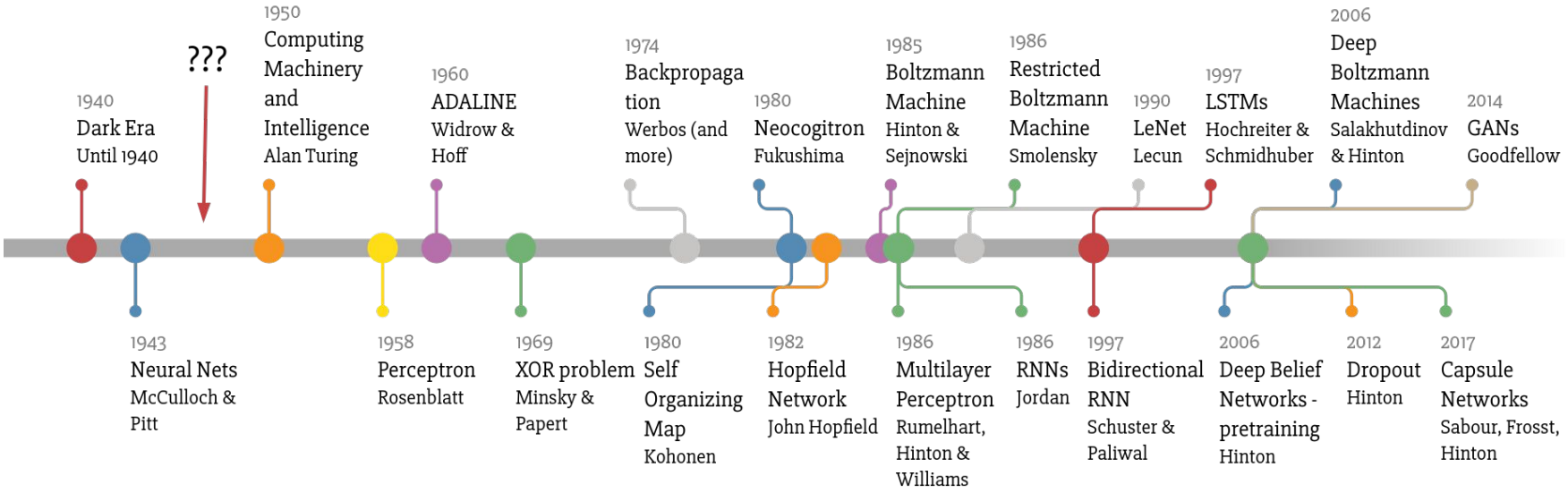


Made by Favio Vázquez

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



Deep Learning Timeline



Made by Favio Vázquez

Term origin

Artificial neural networks were originally used to model biological neural networks starting in the 1930s under the approach of connectionism.

However, starting with the invention of the perceptron, a simple artificial neural network, by Warren McCulloch and Walter Pitts in 1943, followed by the implementation of one in hardware by Frank Rosenblatt in 1957, artificial neural networks became increasingly used for machine learning applications instead, and **increasingly different from their biological counterparts**.

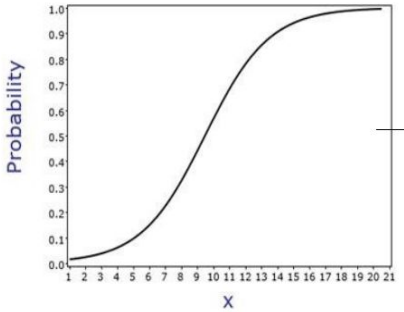
Common knowledge site

Neural networks intuition

girafe ai

02

Logistic regression

X → Wx +b → 

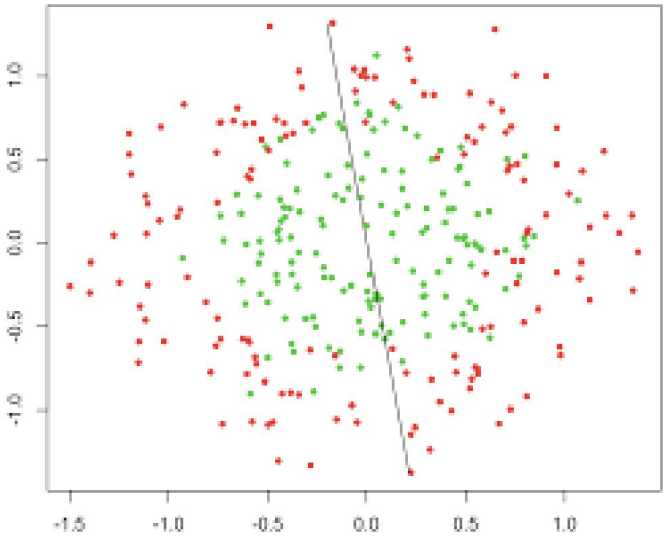
→ P(y)

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

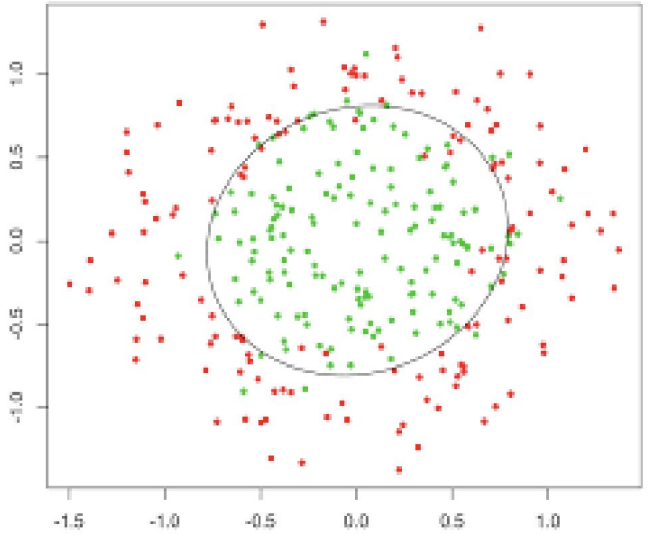
$$L = -\sum_i y_i \log P(y|x_i) + (1 - y_i) \log (1 - P(y|x_i))$$

Problem: nonlinear dependencies

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



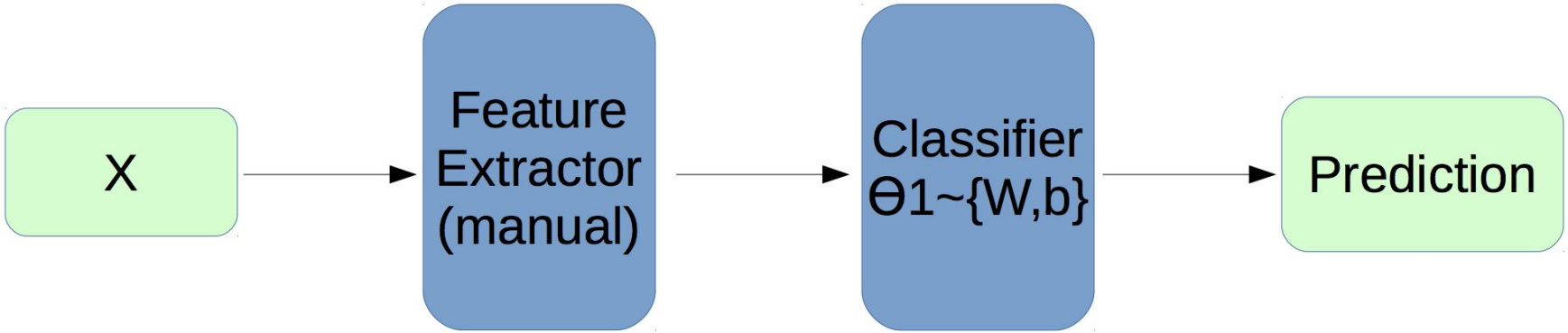
What we have



What we want

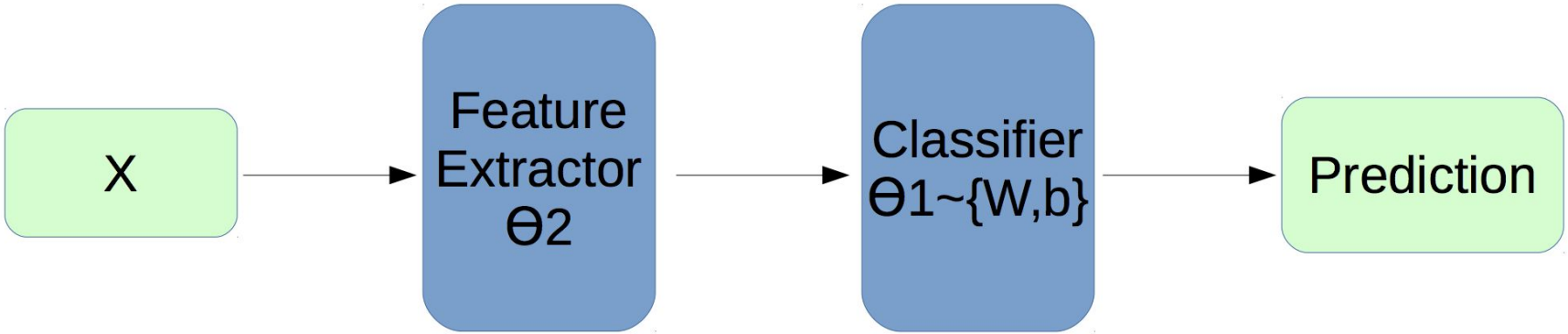
And feature engineering is an art.

Classical pipeline



Handcrafted features generated by experts.

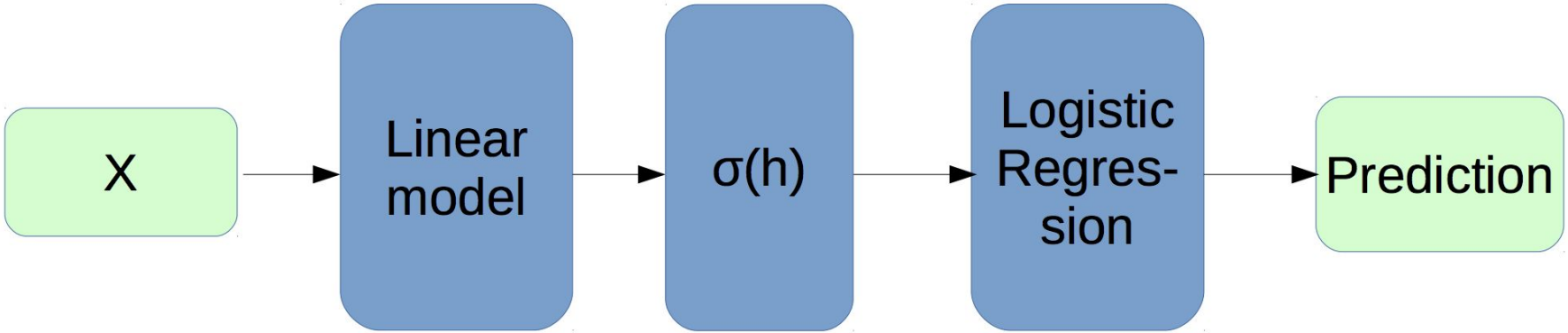
Parametrized pipeline



Automated feature extraction

We fit extractor parameters to data

Neural network pipeline



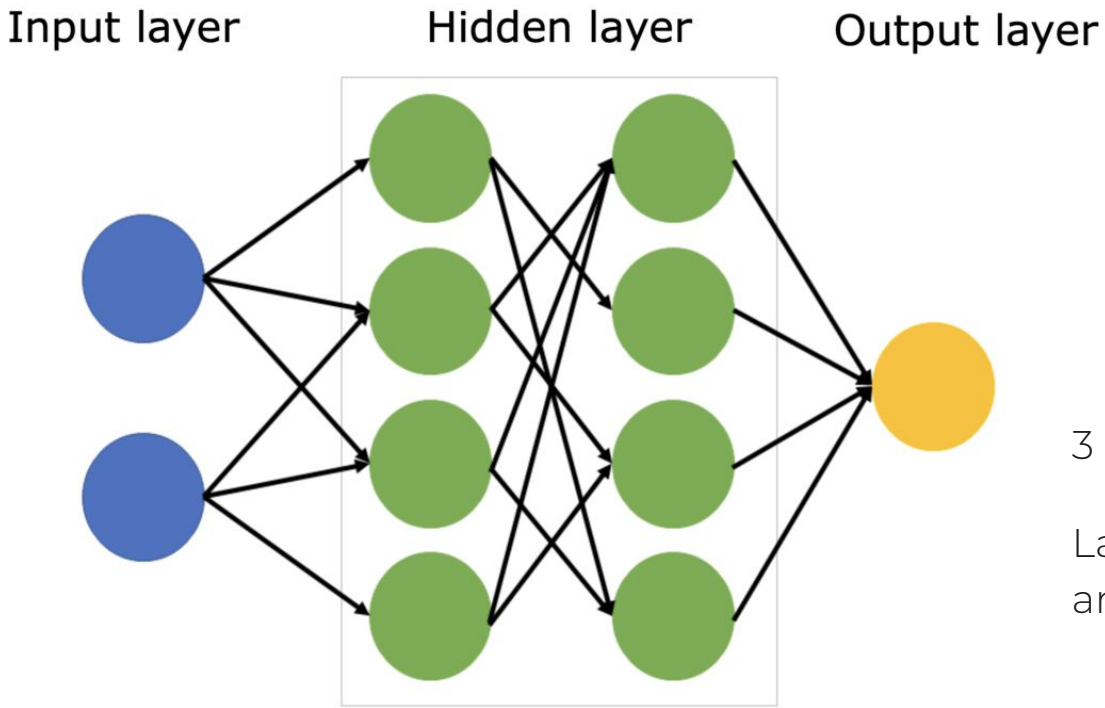
Fully automated feature extraction fitted from data

We can save experts efforts by having big datasets and compute resources

(First linear model has multidimensional output)

Graphical layer notation

How many layers are presented on the picture?



3 layers.... 🤦

Layer is set of arrows 🤦🤦🤦

Layer notations

First notation **counts only trainable transforms**.
In such way LeNet-5 has 5 trainable layers. It is used in academical papers and as consequence in popular architectures such as VGG-19, ResNet-101.
Layers are: Linear, Convolutional, Self-attention, etc.
This is main notation used in everyday life.

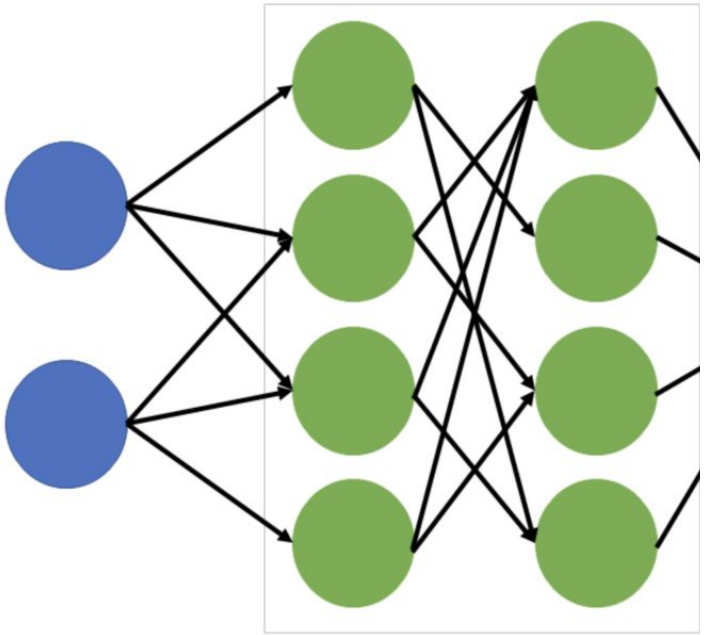
Second notation **counts every transform** made in neural network.
It is derived from engineering practice such as PyTorch code where each Module added to Sequential container is considered as a layer.
Layers are: all from first notation, Sigmoid, Dropout, BatchNorm, etc.

Third notation **counts feature vectors** in between transforms.
Usually such layers are not counted, it is just a notation to indicate parts of neural network.
Layers are: Input, Output and Hidden.

Embedding

in ML is a representation of some object in real valued linear space.

Often embedding are just output of one but last layer of a neural network.



Embedding to R^4

Linear layer

girafe ai

03

Linear layer

It is just a linear transform of input features into output features.

This layer also called **Dense** or **Fully Connected (FC)** because each input feature may affect each output feature.

Also it is sometimes called **Perceptron** or **Multi-layer perceptron (MLP)** in case of multiple linear layers. Another name is **Feedforward neural network**.

Linear layer formula

Implementation:
https://pytorch.org/docs/stable/generated/torch.nn.Linear.html

Number of parameters:

```
in*out + out
```

$$Z = X \cdot W + b$$

$$Z \in R^{n \times \text{out}}$$

$$X \in R^{n \times \text{in}}$$

Activation functions

girafe ai

04

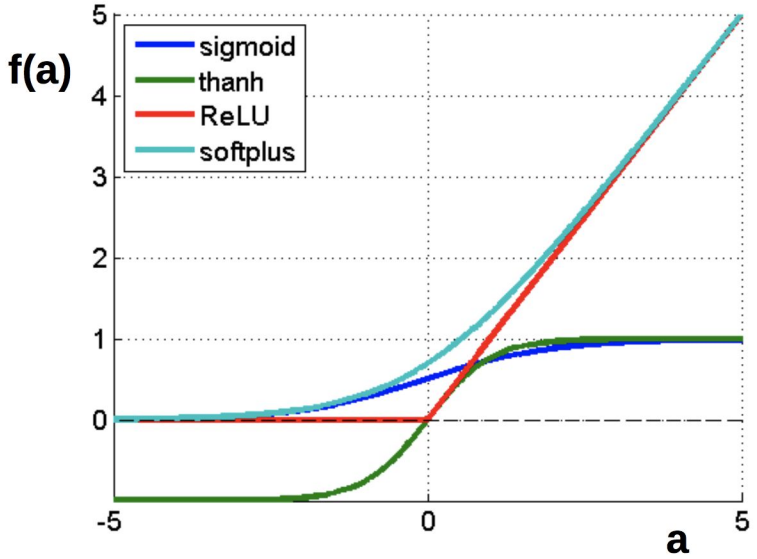
Nonlinearities

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

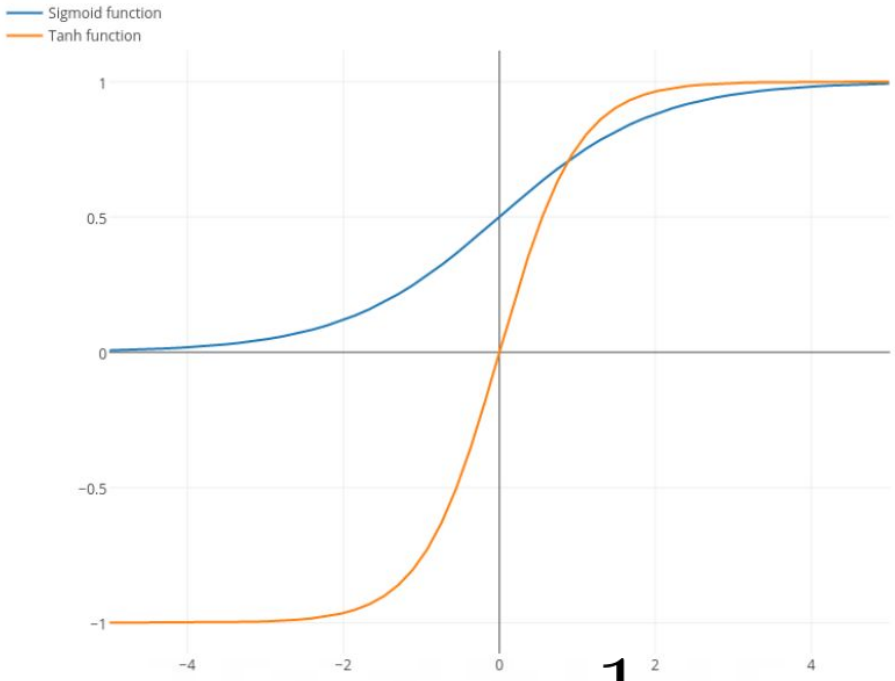
$$f(a) = \tanh(a)$$

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

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



Sigmoid



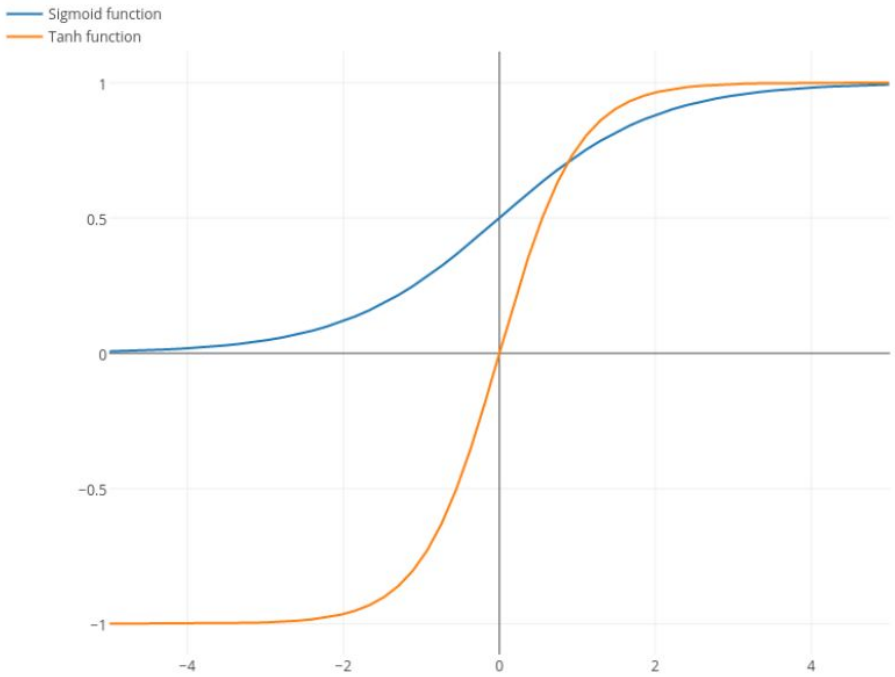
$$f(a) = \frac{1}{1 + e^{-a}}$$

- 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

tanh



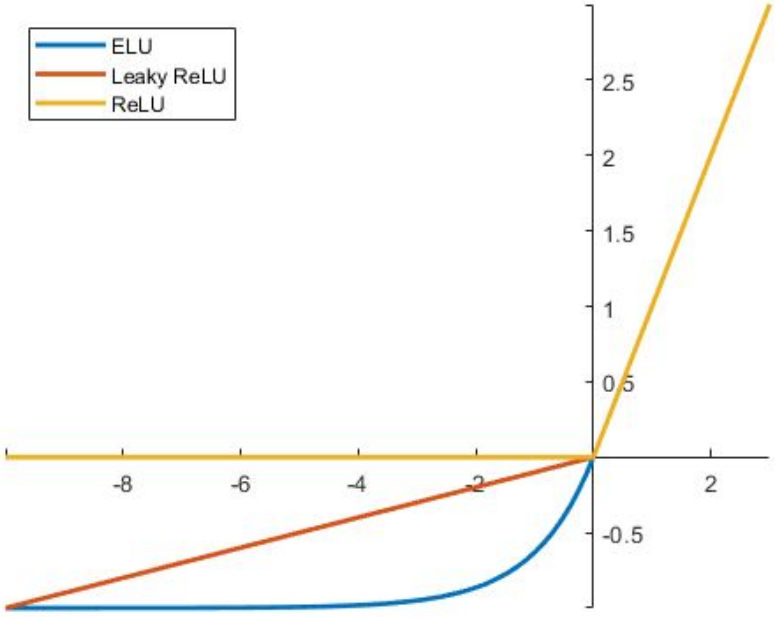
$$f(a) = \tanh(a)$$

- 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

Rectified Linear Unitn (ReLU)



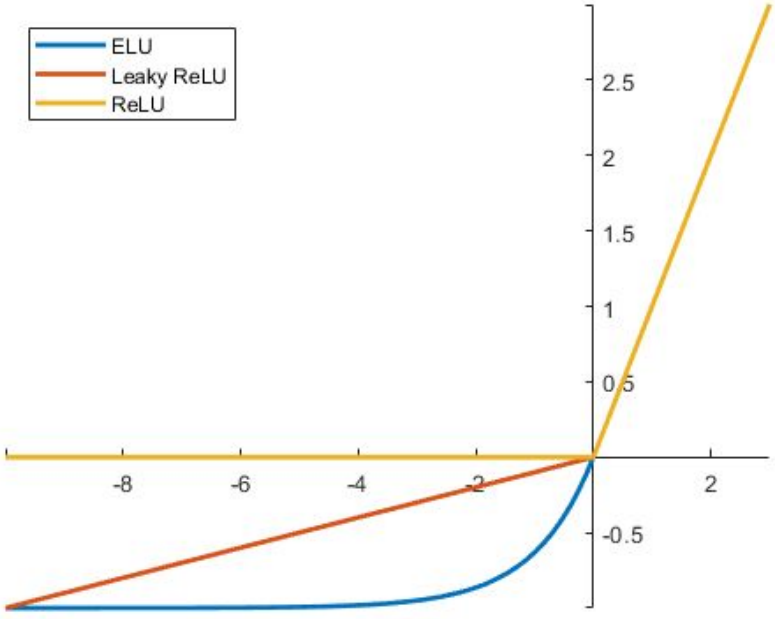
$$f(a) = \max(0, a)$$

- 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

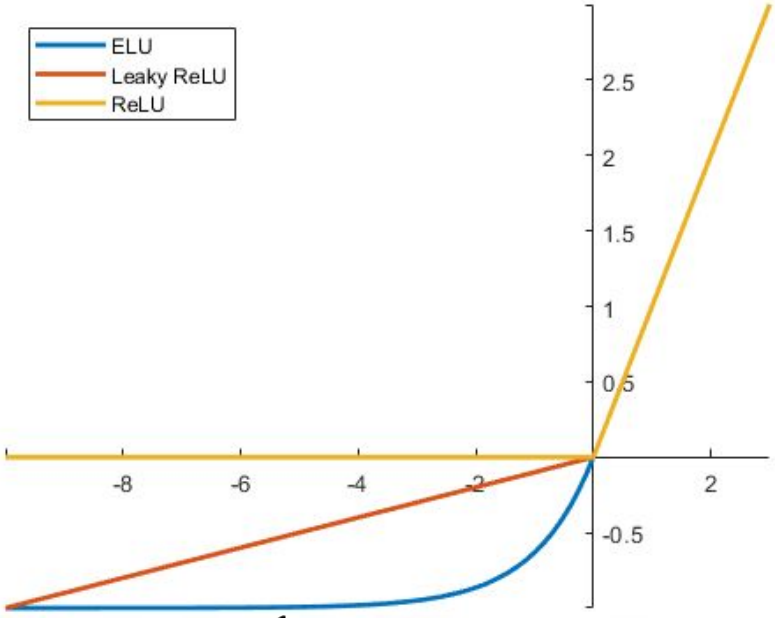
LeakyReLU



- Very simple to compute (both forward and backward)
  Up to 6 times faster than Sigmoid
- Does not saturate
- Shifted, but not so much

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

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
- In case you’re not satisfied try out Leaky ReLU or ELU
- Is used in special cases tanh such as RNN
- Do not use Sigmoid except last layer

Softmax

N-dimensional extension of sigmoid.

It is really differentiable maximum function!

Properties:

- subtraction of constant from input vector dont' change result
- embraces the biggest value
- integrates to 1
- each component is in [0, 1]

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Cross-entropy

girafe ai

05

Information content

was introduced by Claude Shannon in his 1948 article A Mathematical Theory of Communication.

He defined information function `I(p)` which is the amount of information acquired due to the observation of event `i`. It has to meet following properties:

1. `I(p)` is monotonically decreasing in p: an increase in the probability of an event decreases the information from an observed event, and vice versa.
2. `I(1) = 0`: events that always occur do not communicate information.
3. `I(p1 ·p2) = I(p1) + I(p2)`: the information learned from independent events is the sum of the information learned from each event.

Then function `I(p)` is a random variable.

$$\mathrm{I}(p) = \log\left(\frac{1}{p}\right) = -\log(p)$$

Information entropy

quantifies the *average* level of uncertainty or information associated with the variable's potential states or possible outcomes.

This measures the expected amount of information needed to describe the state of the variable, considering the distribution of probabilities across all potential states.

Then entropy is expected value of information content

$$\mathrm{H}(X) = \mathbb{E}[\mathrm{I}(X)] = \mathbb{E}[-\log p(X)]$$

$$-\sum_{x \in \mathcal{X}} p(x) \log_b p(x)$$

Kullback–Leibler divergence

type of statistical distance: a measure of how one reference probability distribution P is different from a second probability distribution Q

$$D_{\mathrm{KL}}(P \parallel Q) = -\sum_{x \in \mathcal{X}} P(x)\ \log\left(\frac{Q(x)}{P(x)}\right)$$

Cross-entropy

measures the average number of bits needed to identify an event drawn from the set when the coding scheme used for the set is optimized for an estimated probability distribution q, rather than the true distribution p

p is reference distribution, q is predicted distribution

$$H(p, q) = H(p) + D_{\mathrm{KL}}(p \parallel q)$$

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x)$$

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 $X_{\acute{e}tale}$ 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 $\mathcal{U} \subset X$ 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.

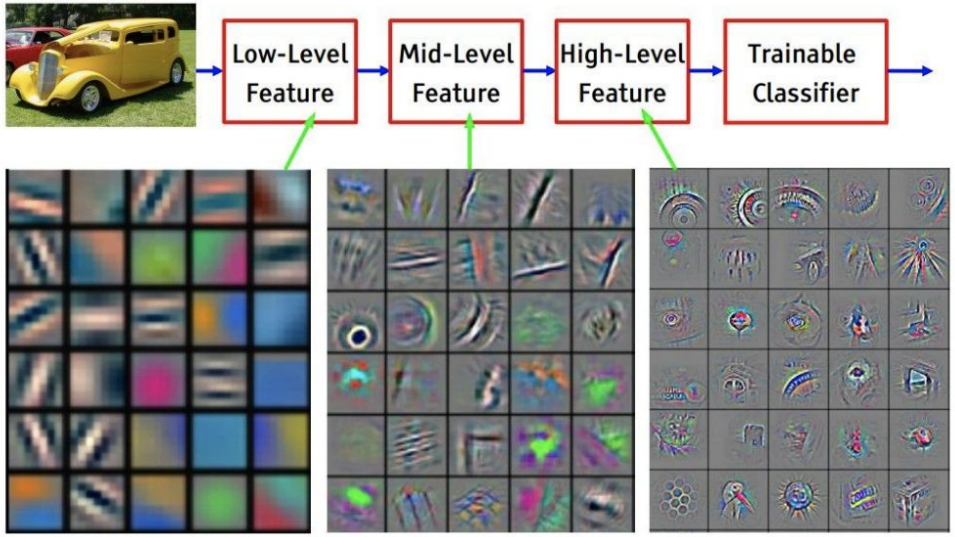
Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

Linux kernel (source code)

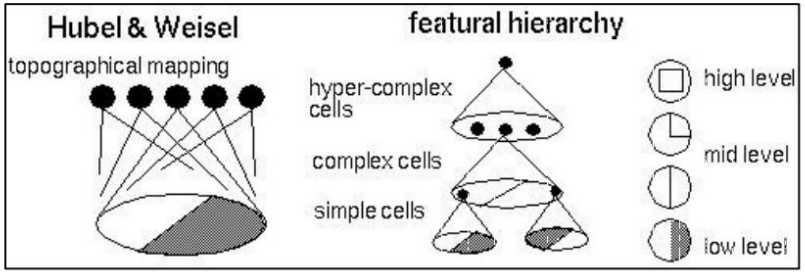
```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
  unsigned long flags;
  int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
  buf[0] = 0xFFFFFFFF & (bit << 4);
  min(inc, slist->bytes);
  printk(KERN_WARNING "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);
}
```


https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Convolutional layer and visual cortex



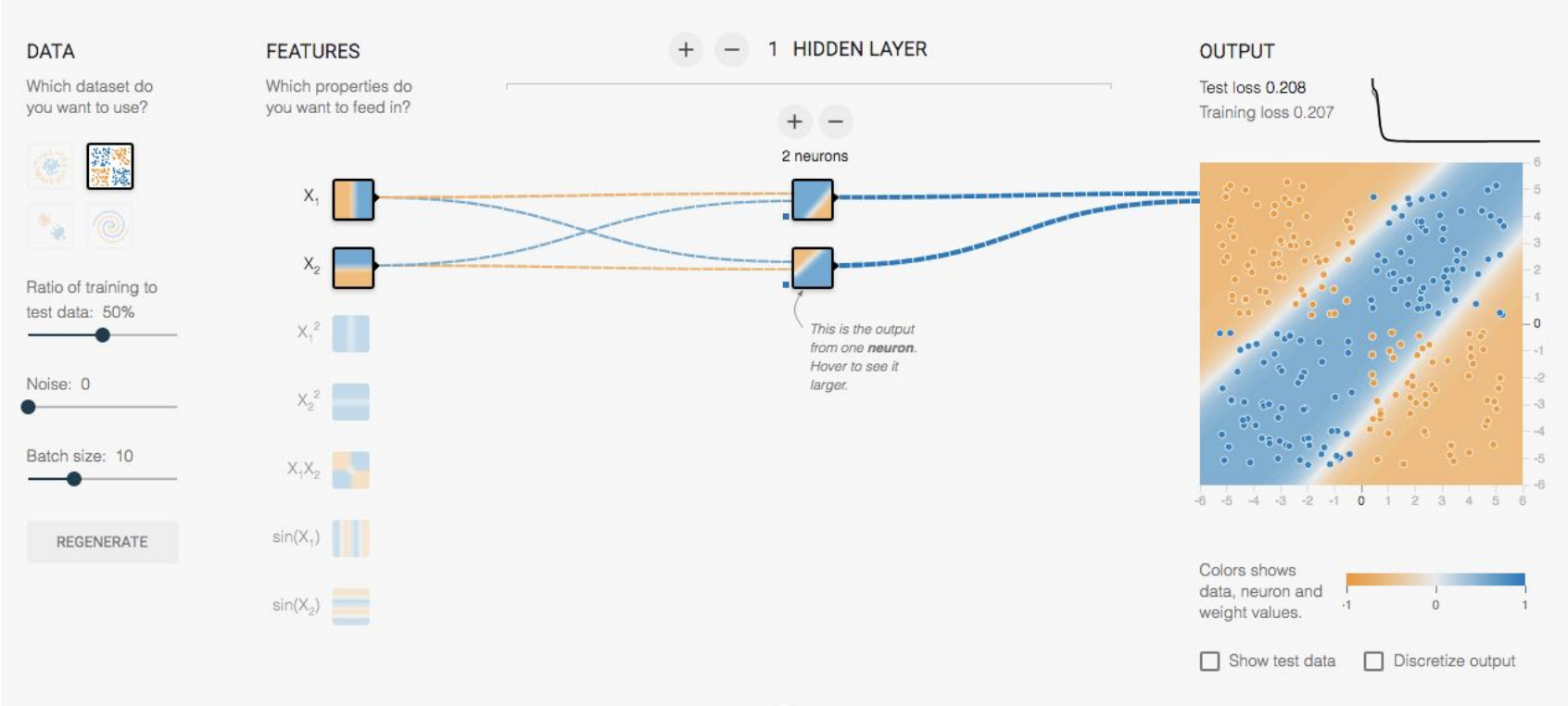
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



Convolutional layer and visual cortex

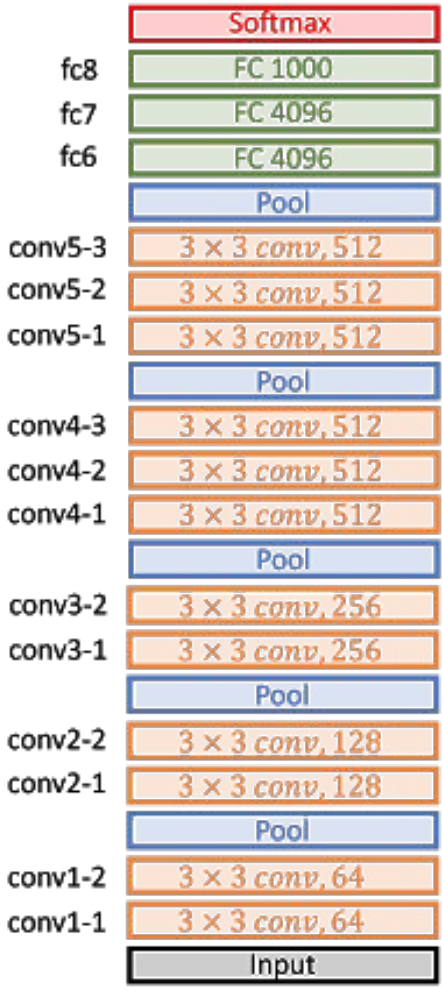
source

Interactive playground



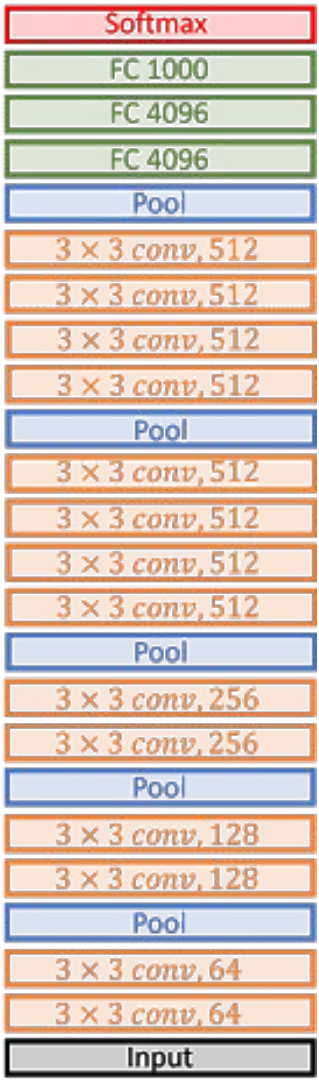
https://playground.tensorflow.org/

Actually, networks can be deep



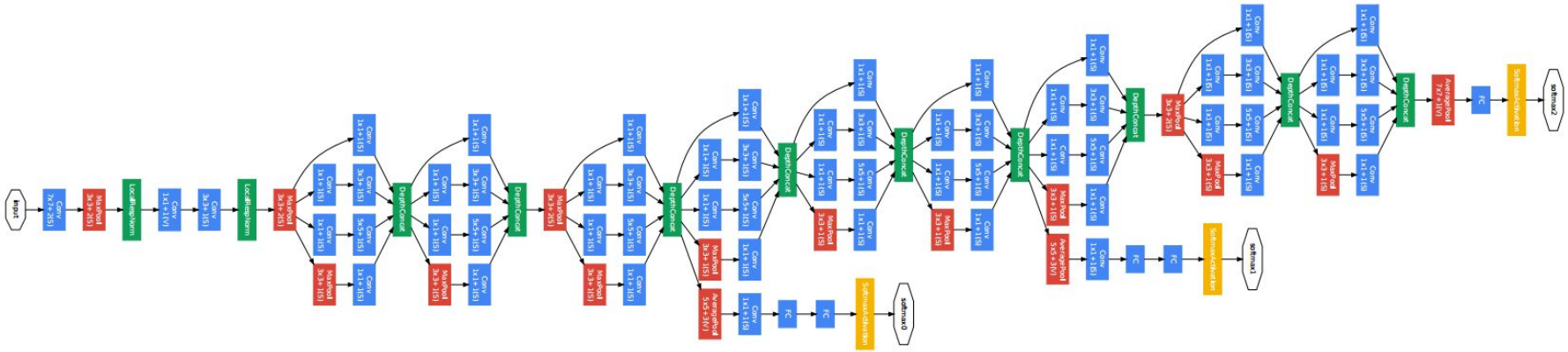
VGG16

And deeper...



VGG19

Much deeper...



How to train it?

Main layers in NNs

- Linear layer
- RNN
- CNN
- Attention

Backpropagation

girafe ai

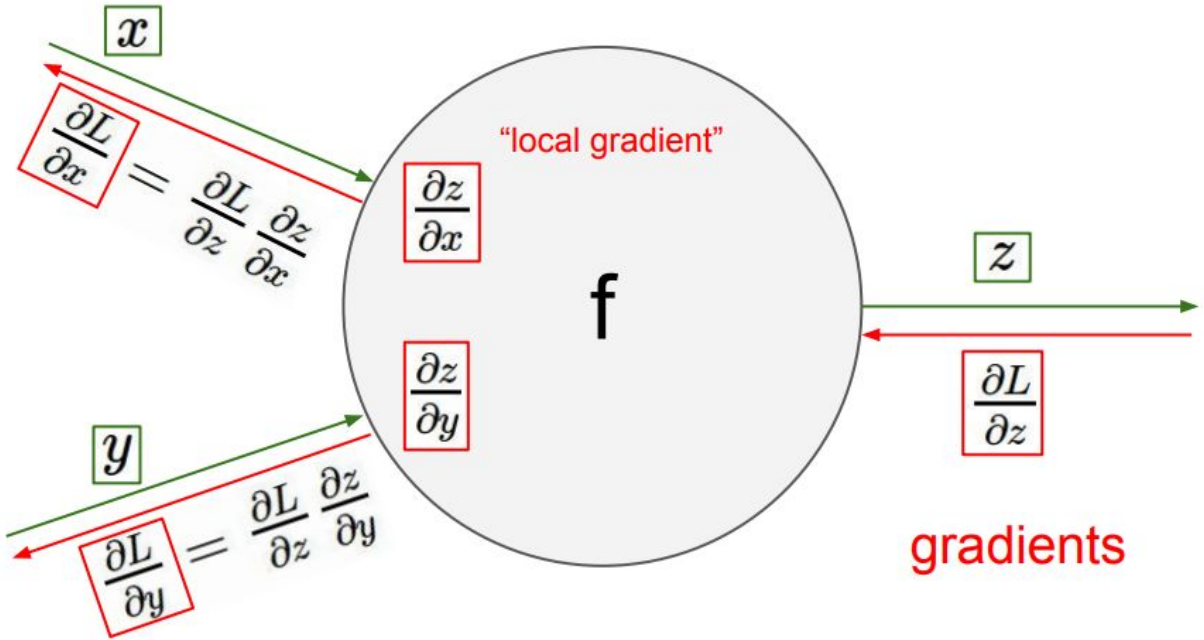
06

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.

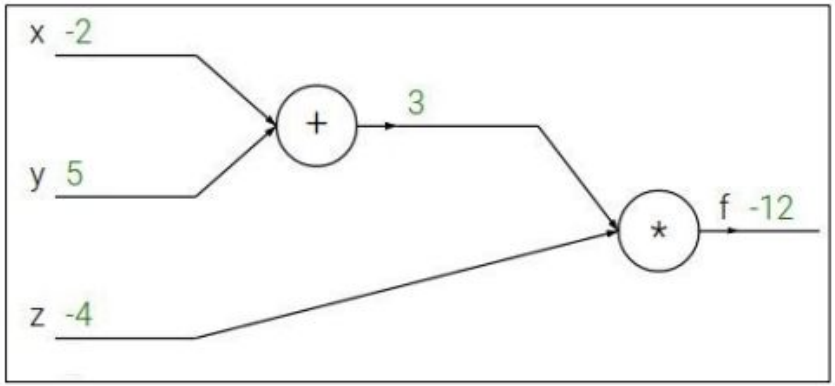
http://cs231n.github.io/neural-networks-3/



Backpropagation example

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

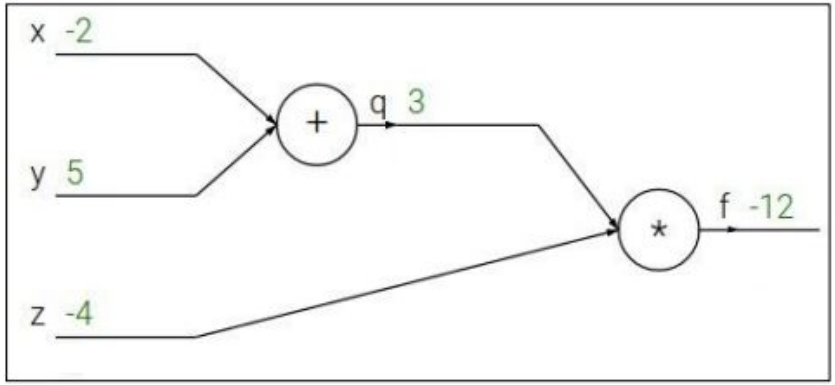
e.g. x = -2, y = 5, z = -4



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

Backpropagation example

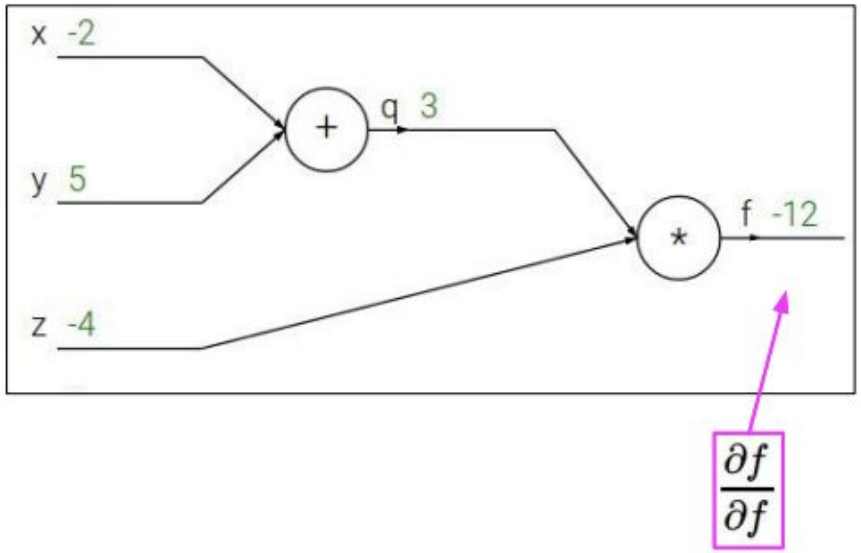
$$f(x, y, z) = (x + y)z$$

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



Backpropagation example

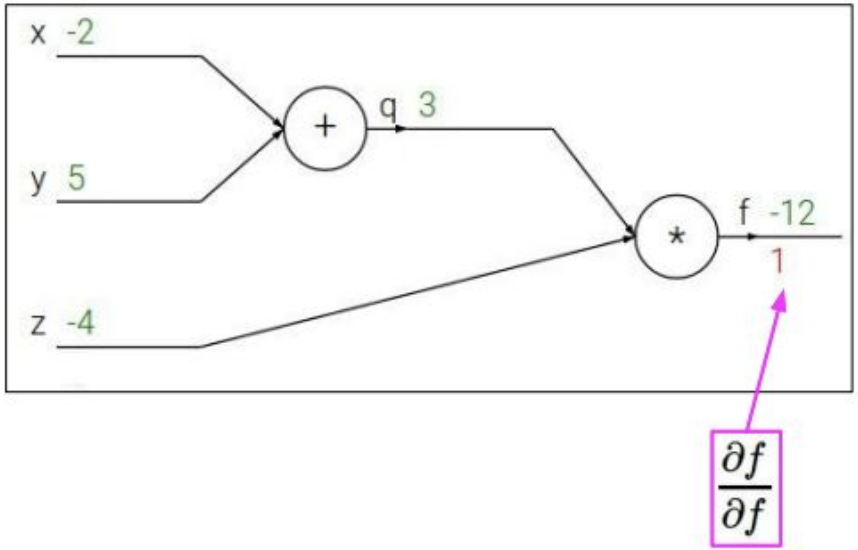
$$f(x, y, z) = (x + y)z$$

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



Backpropagation example

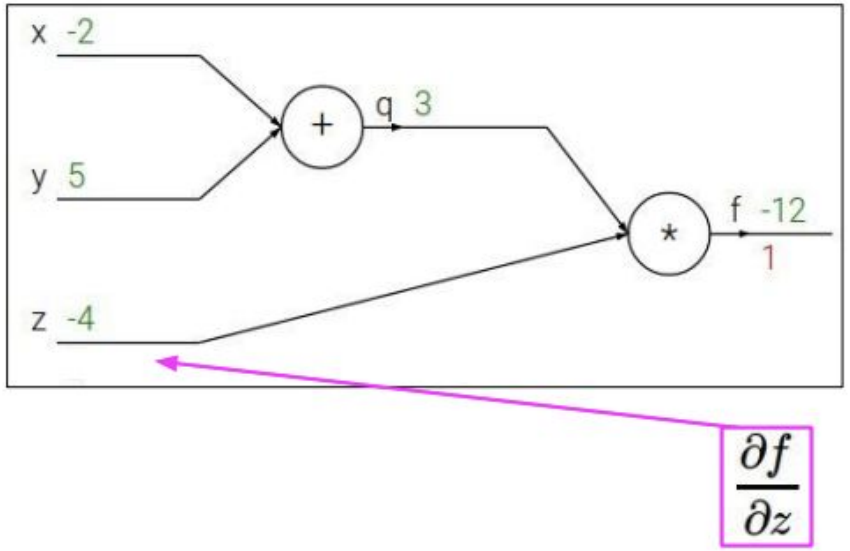
$$f(x, y, z) = (x + y)z$$

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



Backpropagation example

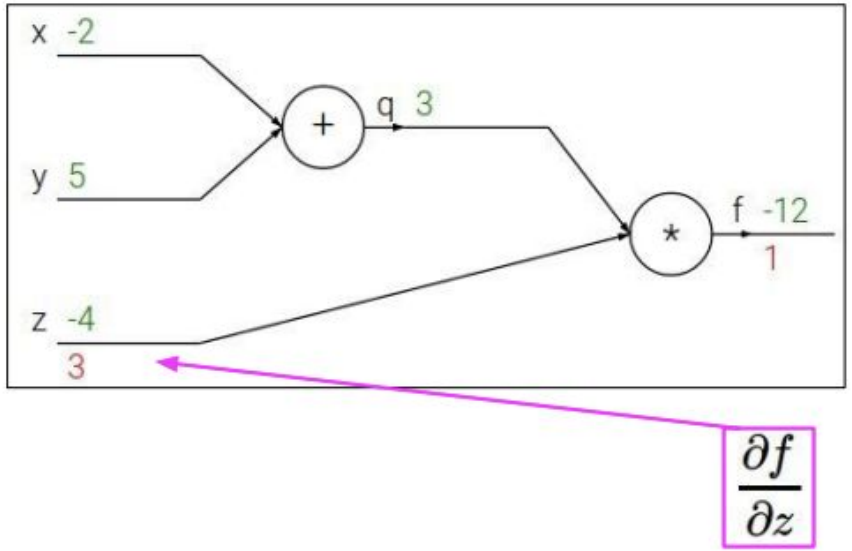
$$f(x, y, z) = (x + y)z$$

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



Backpropagation example

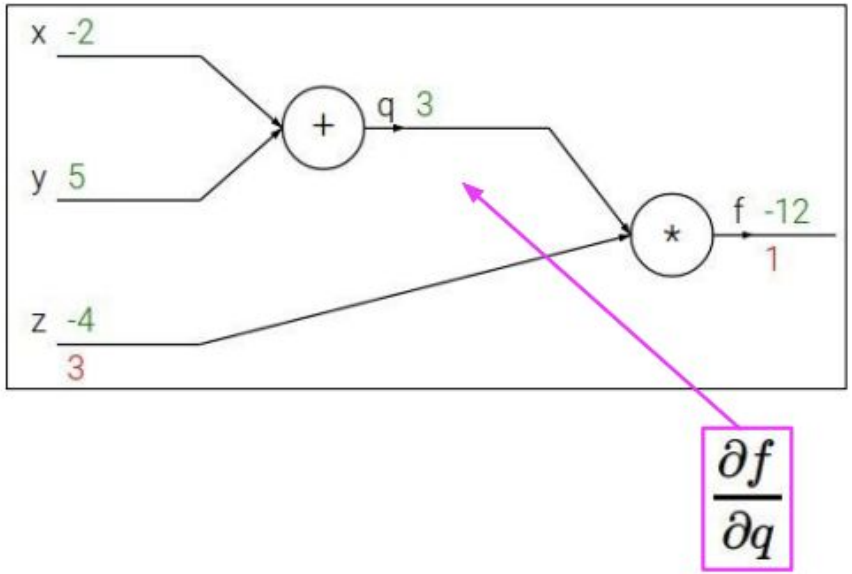
$$f(x, y, z) = (x + y)z$$

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



Backpropagation example

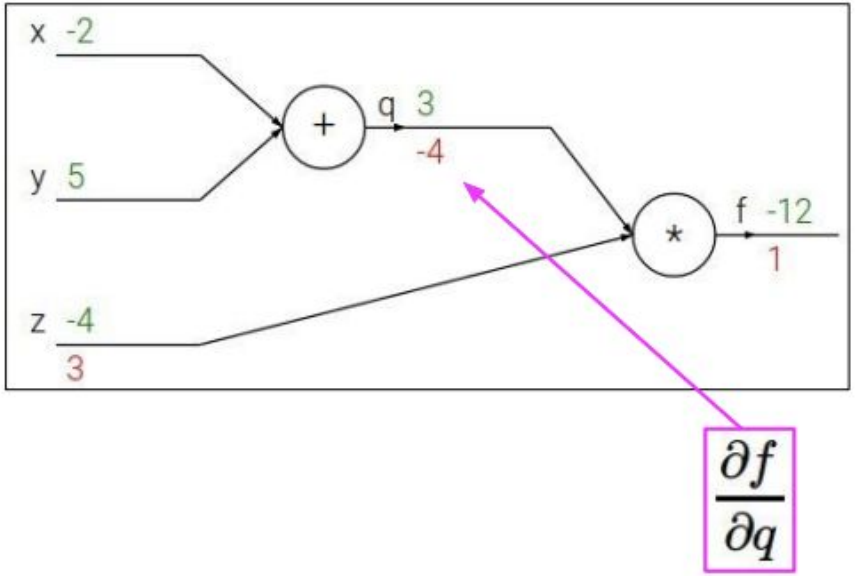
$$f(x, y, z) = (x + y)z$$

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



Backpropagation example

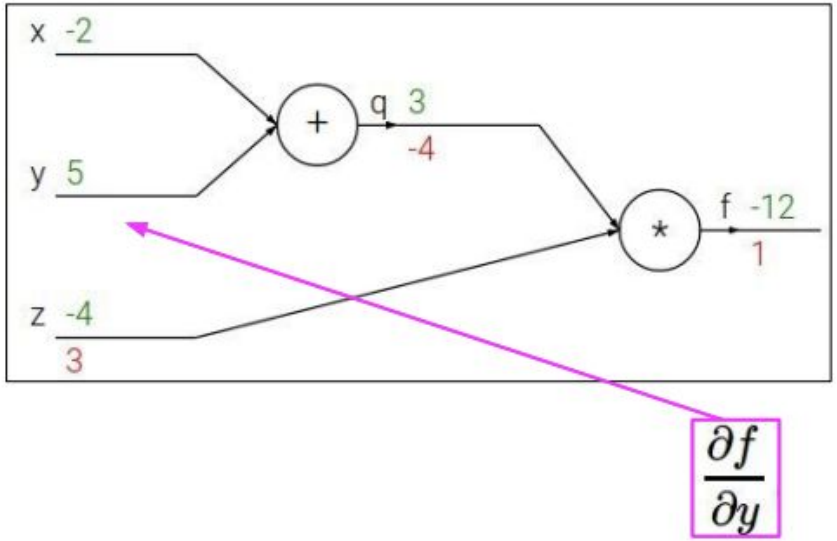
$$f(x, y, z) = (x + y)z$$

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



Backpropagation example

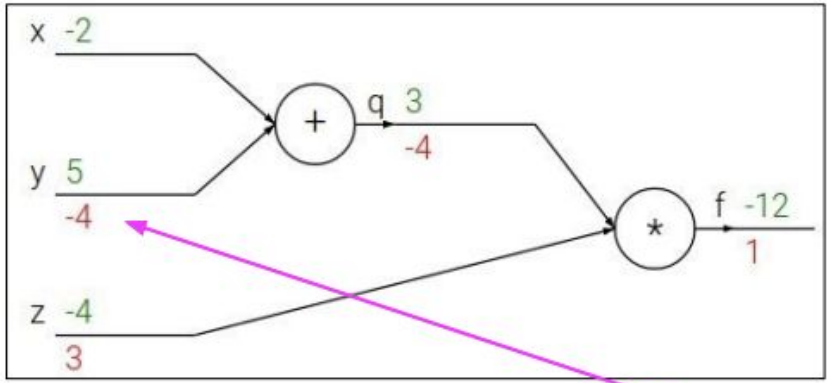
$$f(x, y, z) = (x + y)z$$

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:

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

Backpropagation example

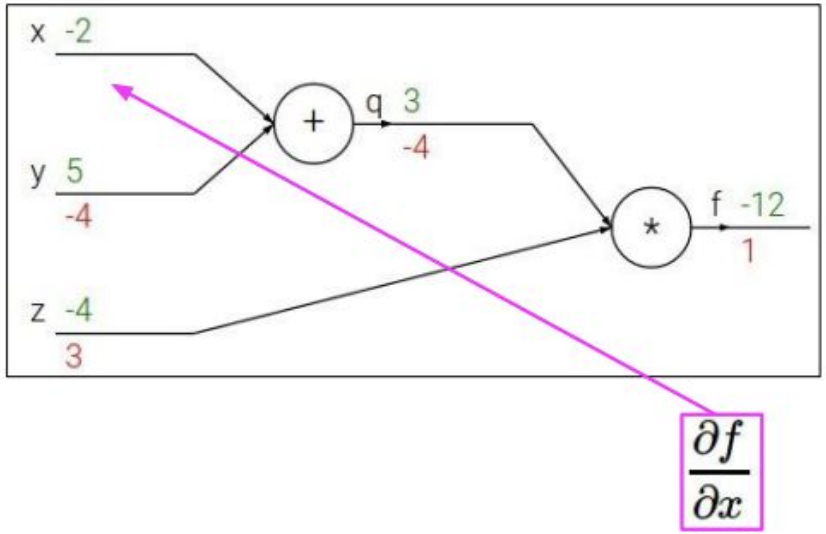
$$f(x, y, z) = (x + y)z$$

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



Backpropagation example

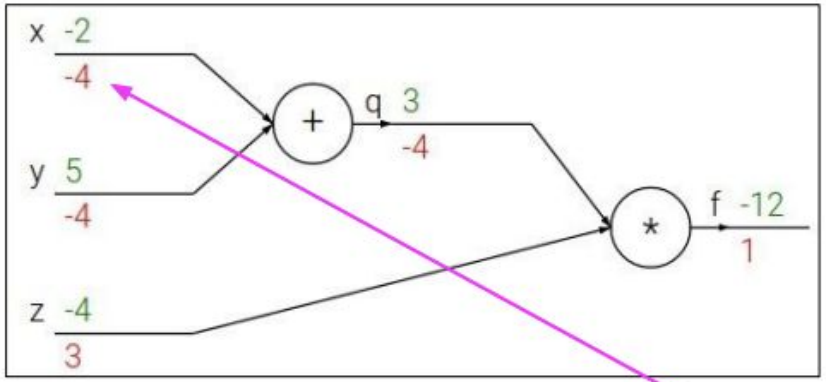
$$f(x, y, z) = (x + y)z$$

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



Chain rule:

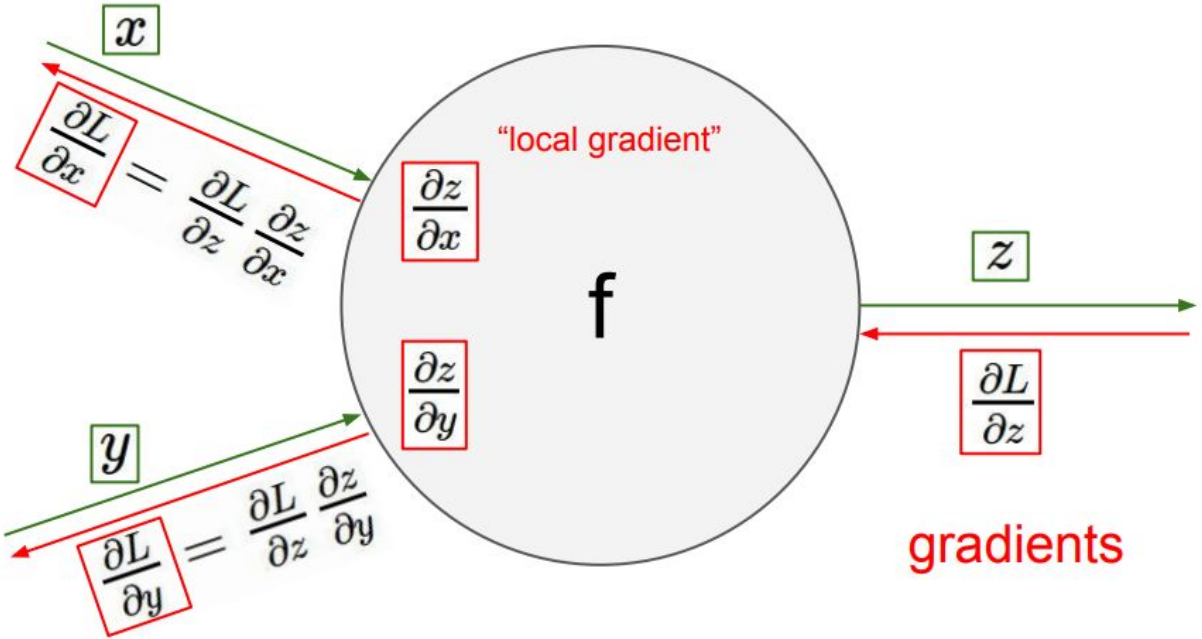
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

Backpropagation and chain rule

Chain rule is just simple math:

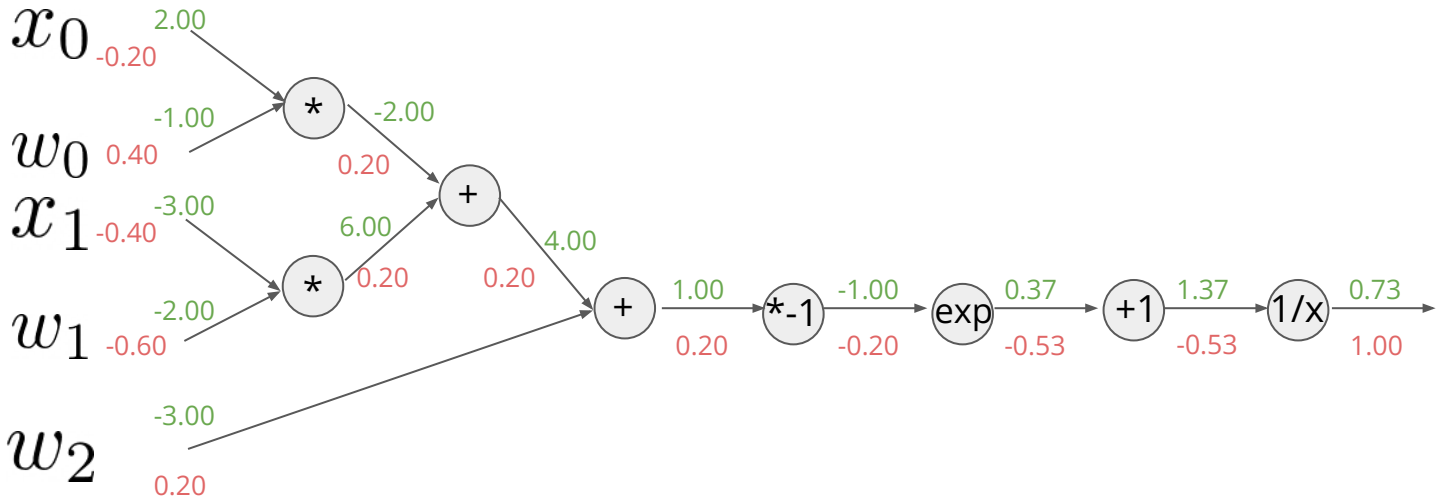
$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

Backprop is just way to use it in NN training.



Backpropagation in LogReg

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



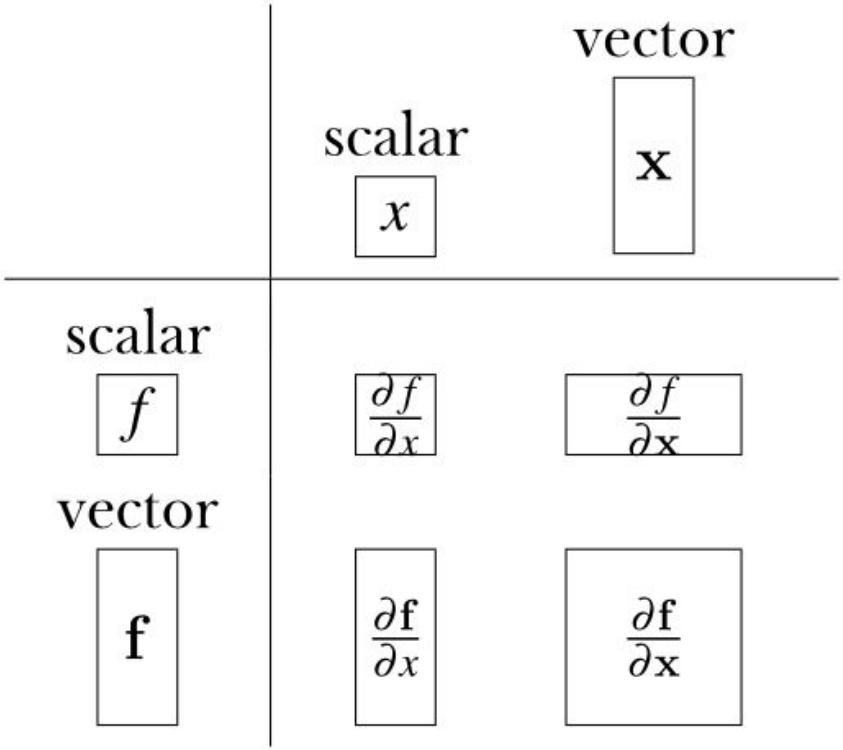
Backpropagation matrix form

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

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

https://explained.ai/matrix-calculus/

Backpropagation matrix form



Derivative of vector by itself

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

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

$$= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & \frac{\partial}{\partial x_2} x_1 & \cdots & \frac{\partial}{\partial x_n} x_1 \\ \frac{\partial}{\partial x_1} x_2 & \frac{\partial}{\partial x_2} x_2 & \cdots & \frac{\partial}{\partial x_n} x_2 \\ & & \cdots & \\ \frac{\partial}{\partial x_1} x_n & \frac{\partial}{\partial x_2} x_n & \cdots & \frac{\partial}{\partial x_n} x_n \end{bmatrix}$$

(and since $\frac{\partial}{\partial x_j} x_i = 0$ for $j \neq i$)

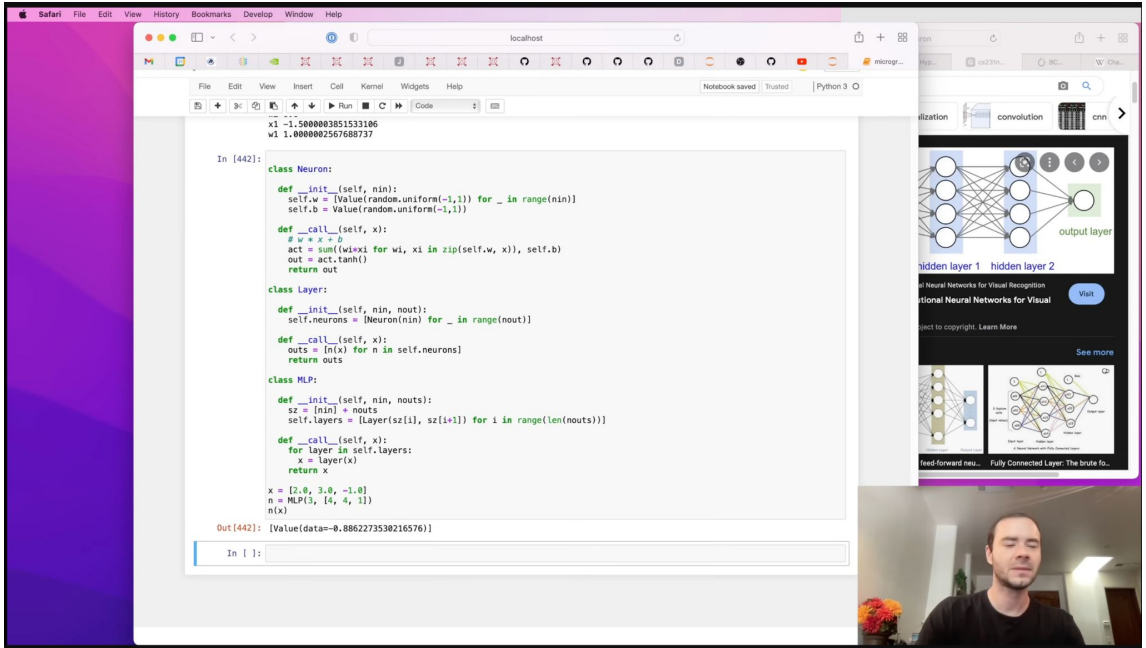
$$= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & 0 & \cdots & 0 \\ 0 & \frac{\partial}{\partial x_2} x_2 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & \frac{\partial}{\partial x_n} x_n \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

$= I$ (I is the identity matrix with ones down the diagonal)

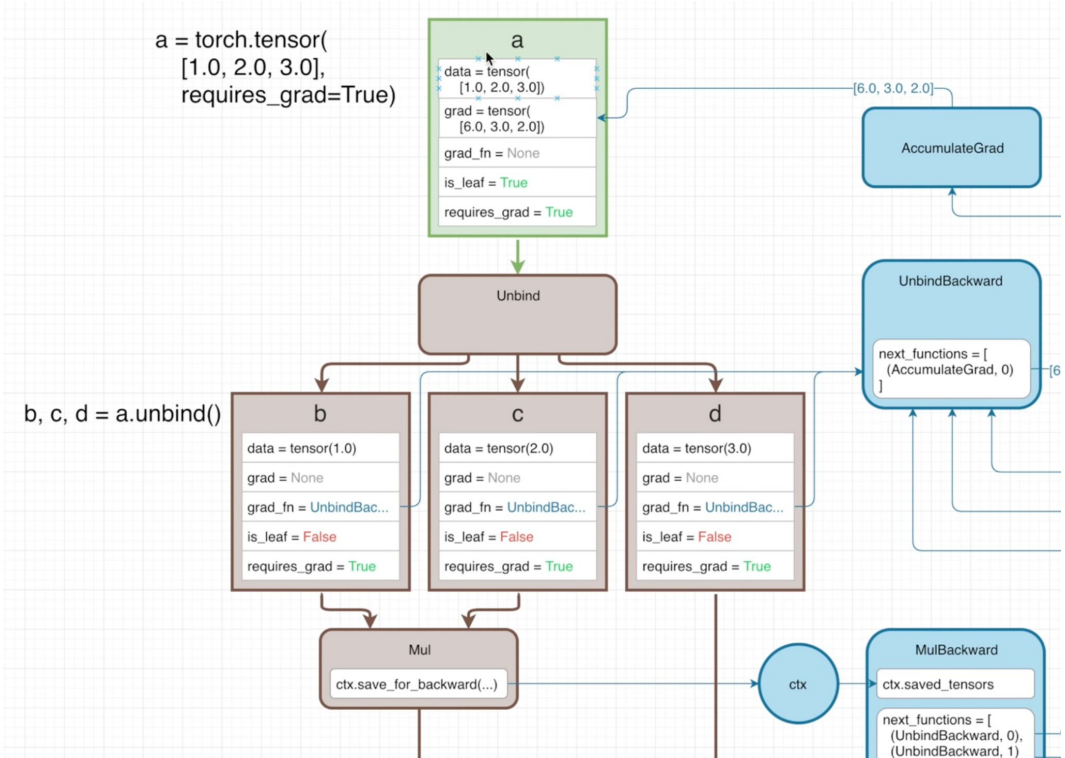
Autograd in Python from the scratch with Andrej Karpathy

https://www.youtube.com/watch?v=VMj-3S1tku0



Autograd in PyTorch

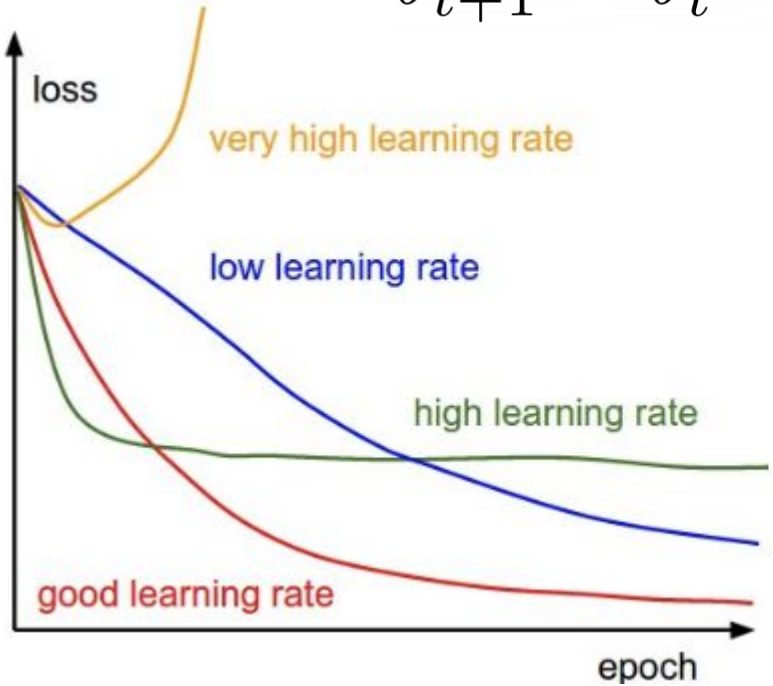
https://www.youtube.com/watch?v=MswxJw-8PvE

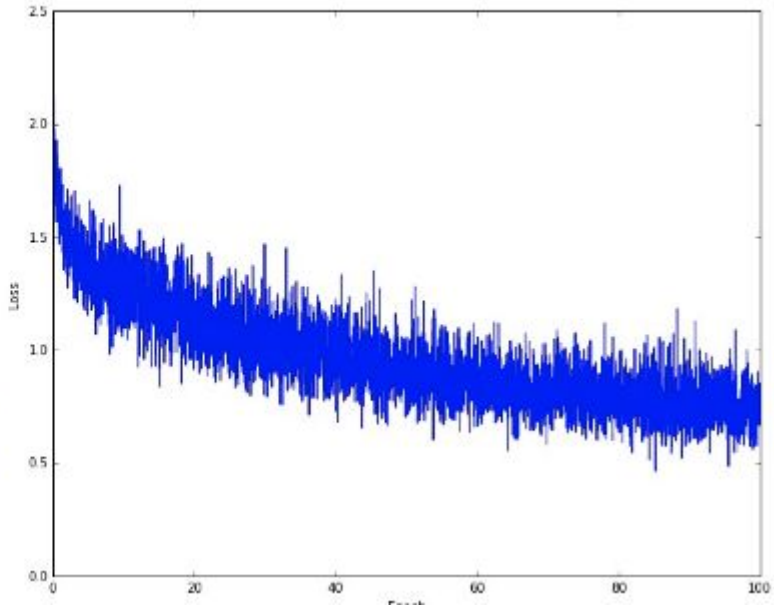


Gradient optimization

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

$$\theta_{t+1} = \theta_t - \text{learning rate} \cdot \frac{\partial}{\partial \theta} Loss$$





Neural networks summary

- Backpropagation is just a fancy usage of chain rule
  - and still it is just basic differentiation
  - used for gradient descent
- All operations should be differentiable to use backpropagation mechanics
  - NN frameworks implement such automatic gradients
- Neural network is like layer cake or lasagna
  - first parameterized linear operation
  - activation function
  - and so on

Neural network libraries overview

girafe ai

06

Pytorch

King of the hill at the moment.

Originally developed by Facebook since 2016.

Written in C++, has 83k stars on Github (2024)

BSD-3 licensed

PyTorch

Tensorflow

Previous king of the hill.

Originally developed by Google since 2015.

Written in C++, has 186k starts on Github (2024)

Apache License 2.0

TensorFlow

Jax

Next king of the hill.

Originally developed by Google since.

Revise

1. History of Deep Learning
2. Neural networks intuition
3. Linear layer
4. Activation functions
5. Crossentropy
6. Backpropagation
7. Neural network libraries overview

Thanks for attention!

Questions?