

Neural Networks basics

Radoslav Neychev



YSDA, Fall 2024





Questions

1. Какой размер имеет бутстрэпированная 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

- 
- A decorative graphic in the bottom-left corner consists of several white-outlined geometric shapes on a teal background. It includes a large irregular polygon on the left, a pentagon in the center, and some curved lines at the bottom.
- 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

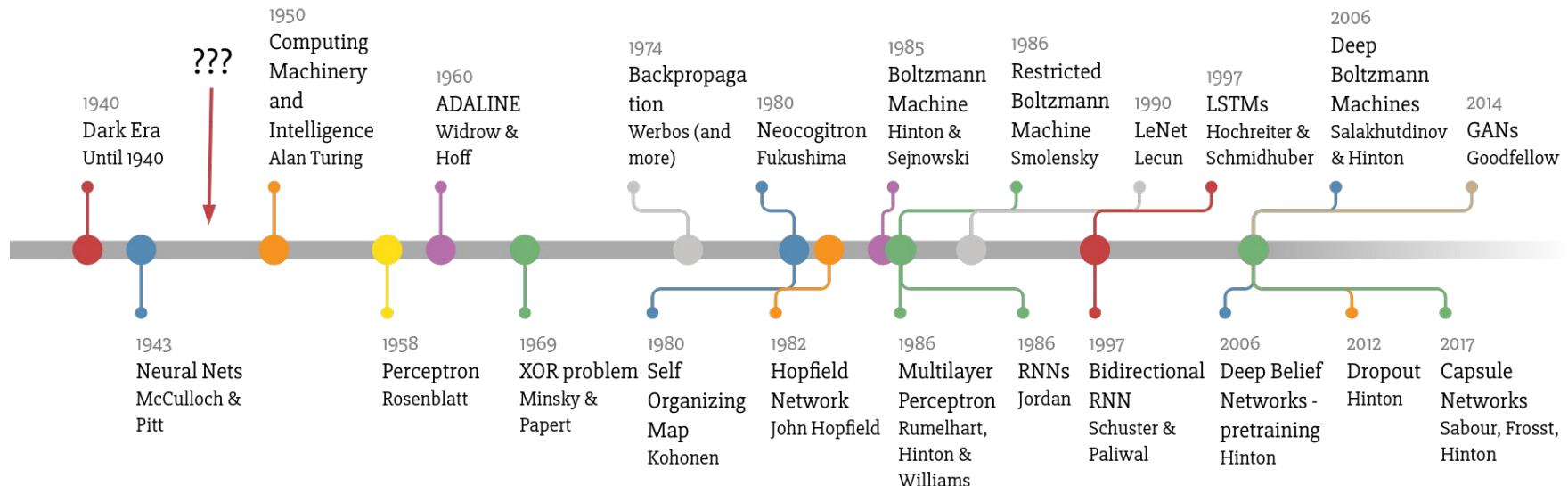
History of Deep Learning

girafe
ai

01

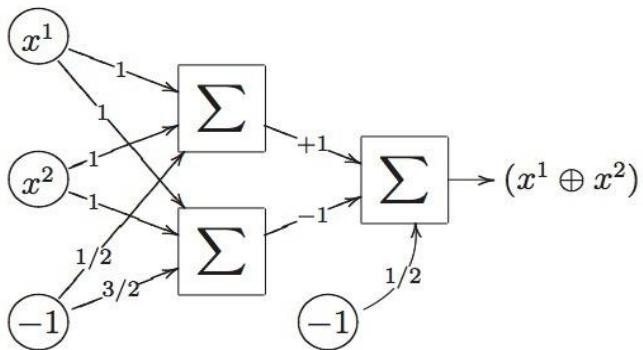


Deep Learning Timeline



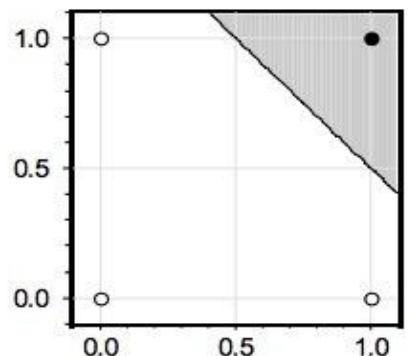


XOR problem

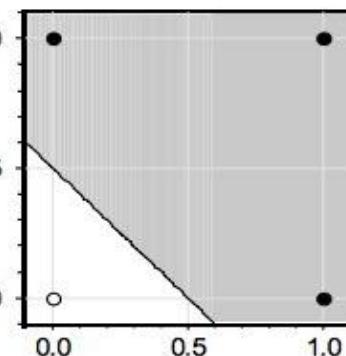


This 2-layer NN (on the left) implements XOR with only x^1 and x^2 features.

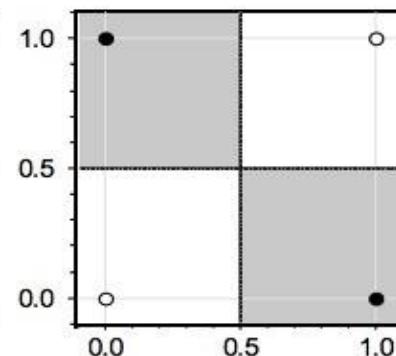
1-layer NN also can succeed, but only with extra feature $x^1 \cdot x^2$.



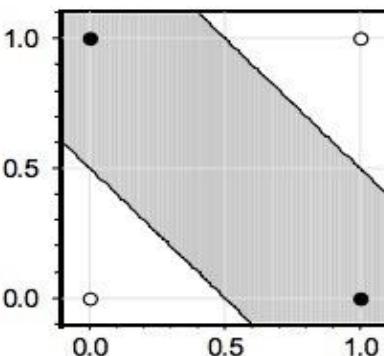
AND



OR



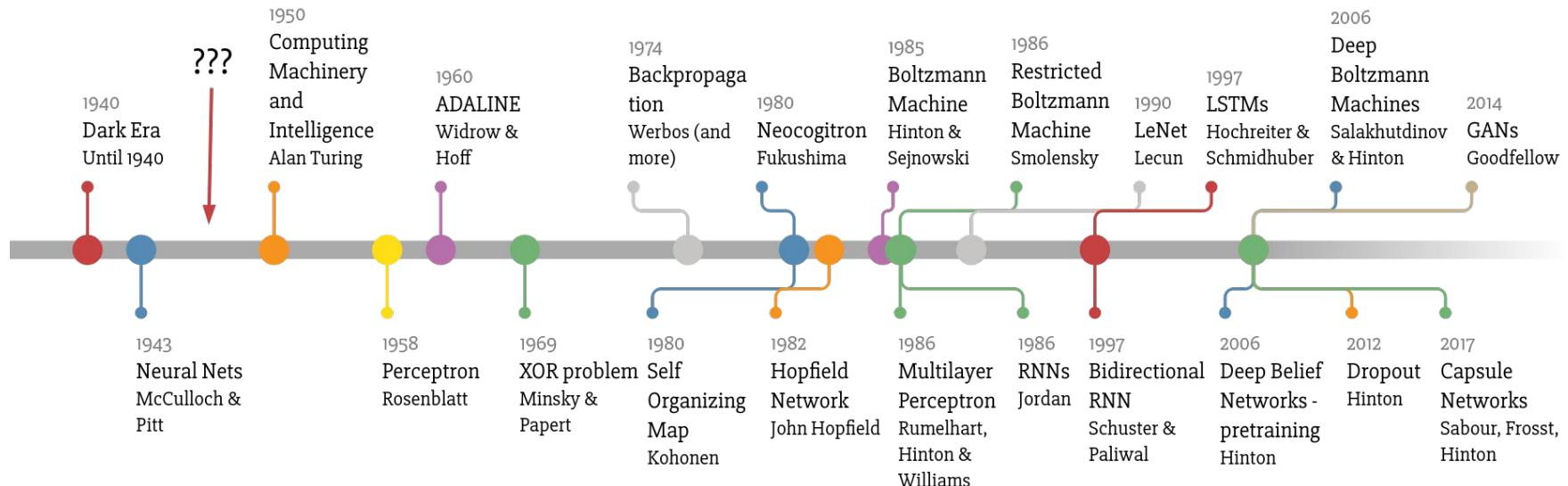
XOR(with $x^1 \cdot x^2$)



XOR

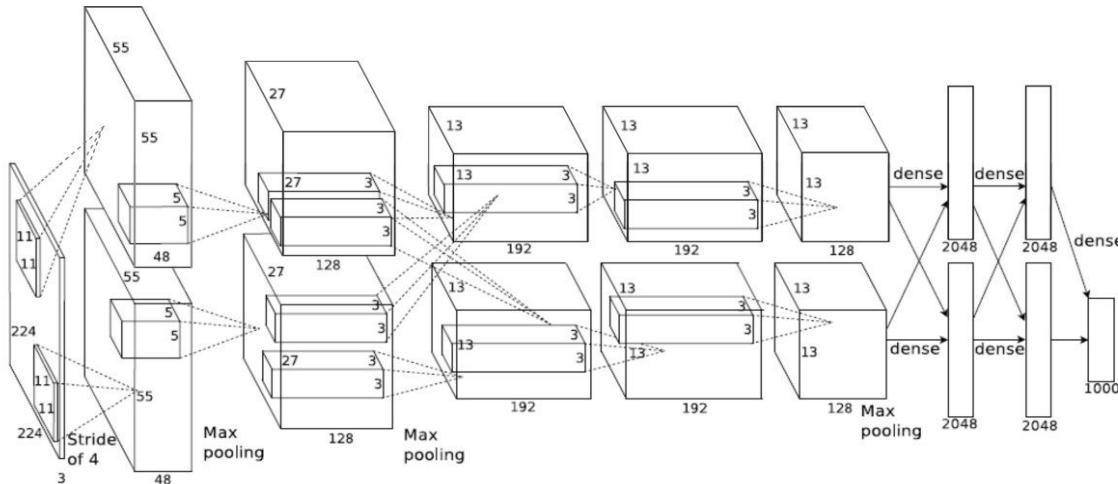
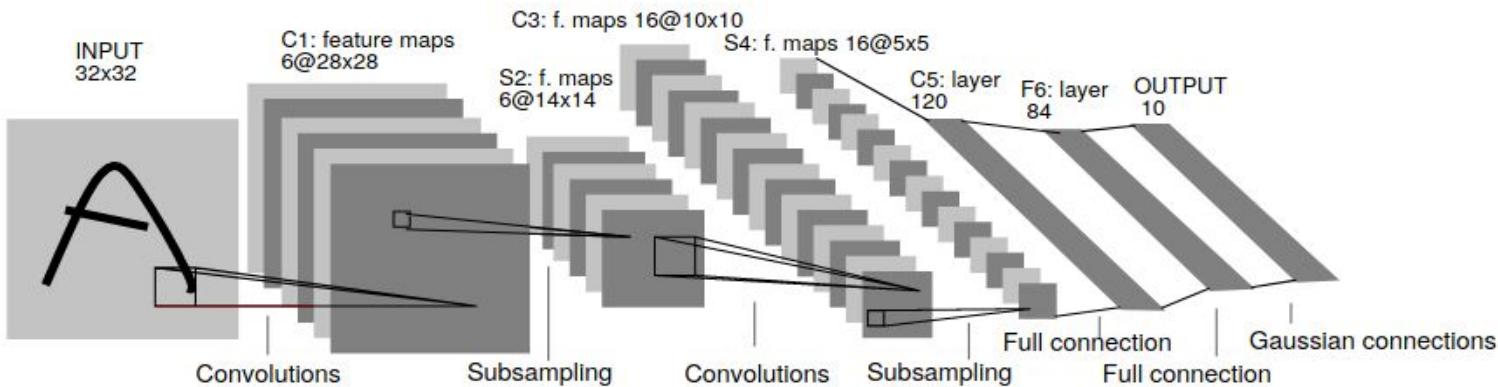


Deep Learning Timeline





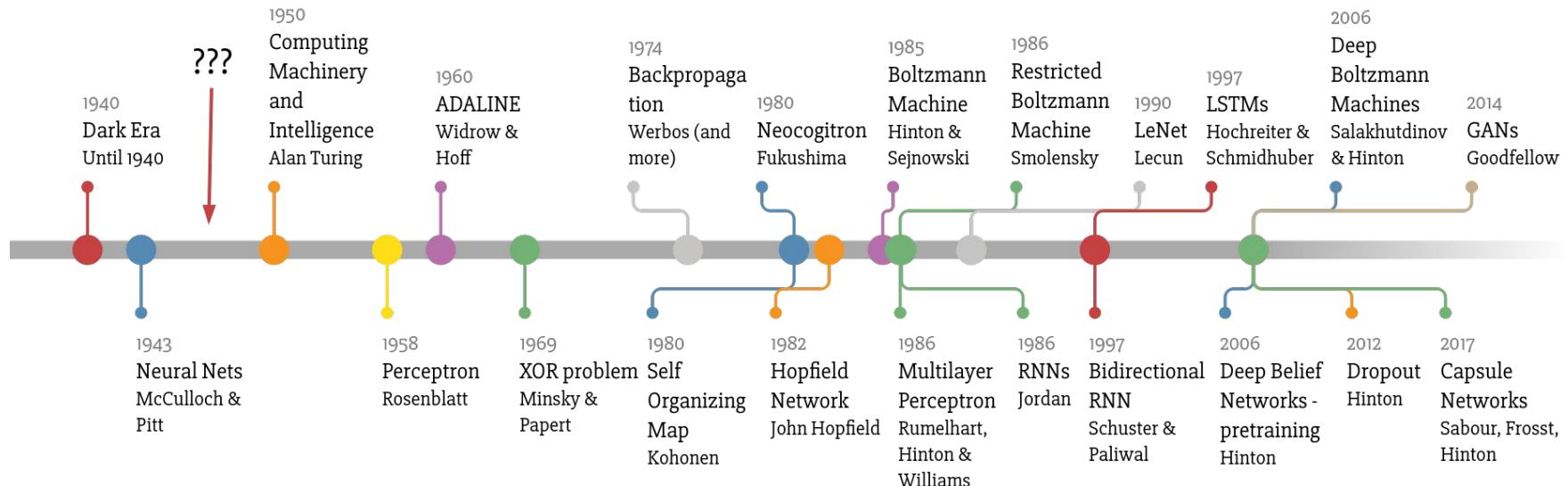
LeNet-5, 1998



AlexNet, 2012



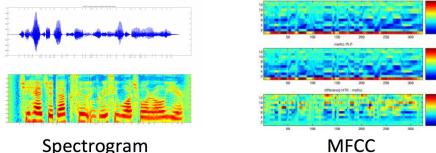
Deep Learning Timeline



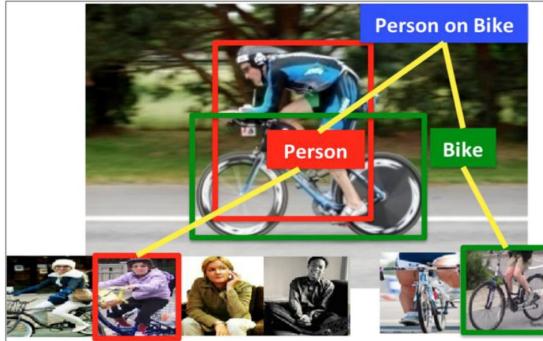
Real world applications



Audio Features



- Object detection
- Action classification
- Image captioning
- ...



"man in black shirt is playing guitar."

GANs. 2014+



1	3	9	3	9	9
1	1	0	6	0	0
0	1	9	1	2	2
6	3	2	0	8	8

a)



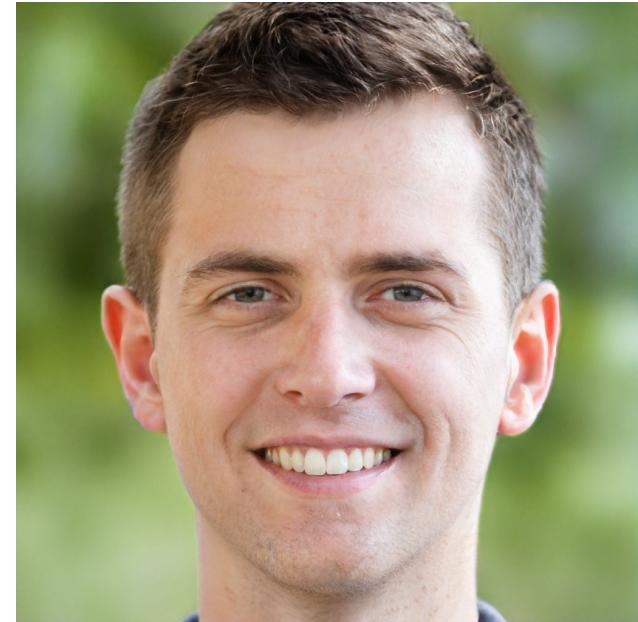
b)



c)



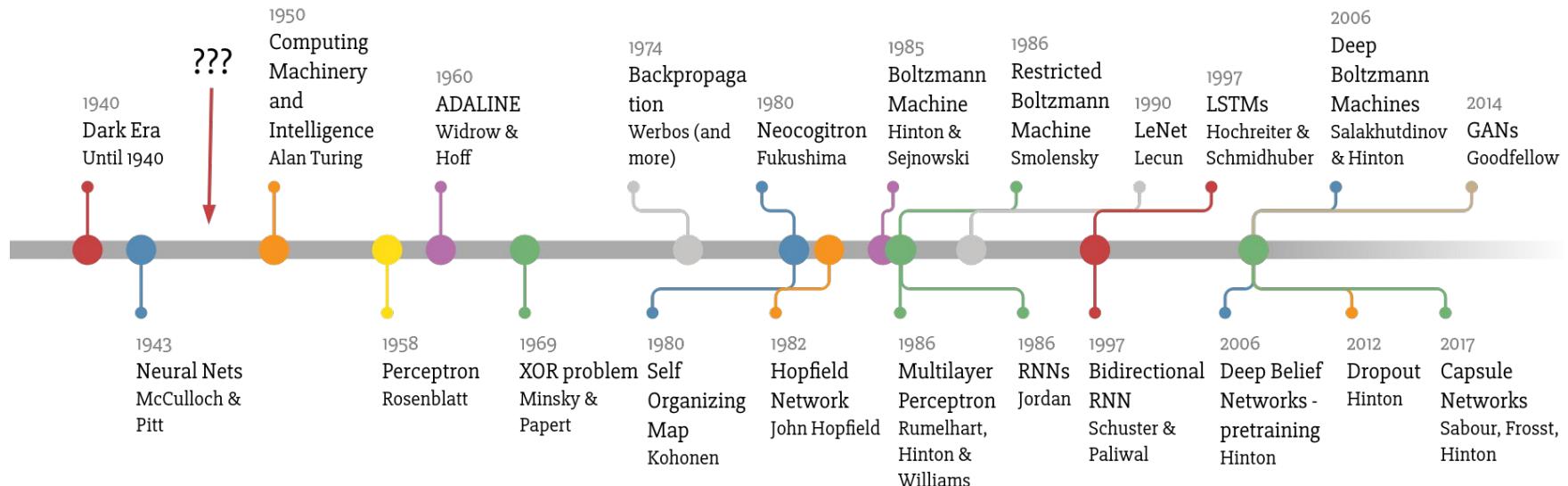
d)



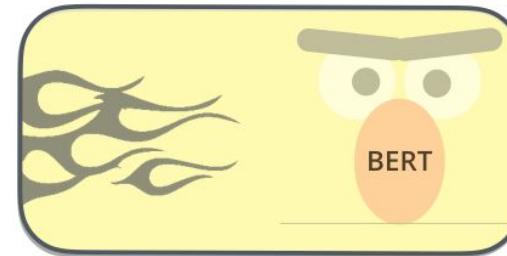
<https://thispersondoesnotexist.com/>



Deep Learning Timeline

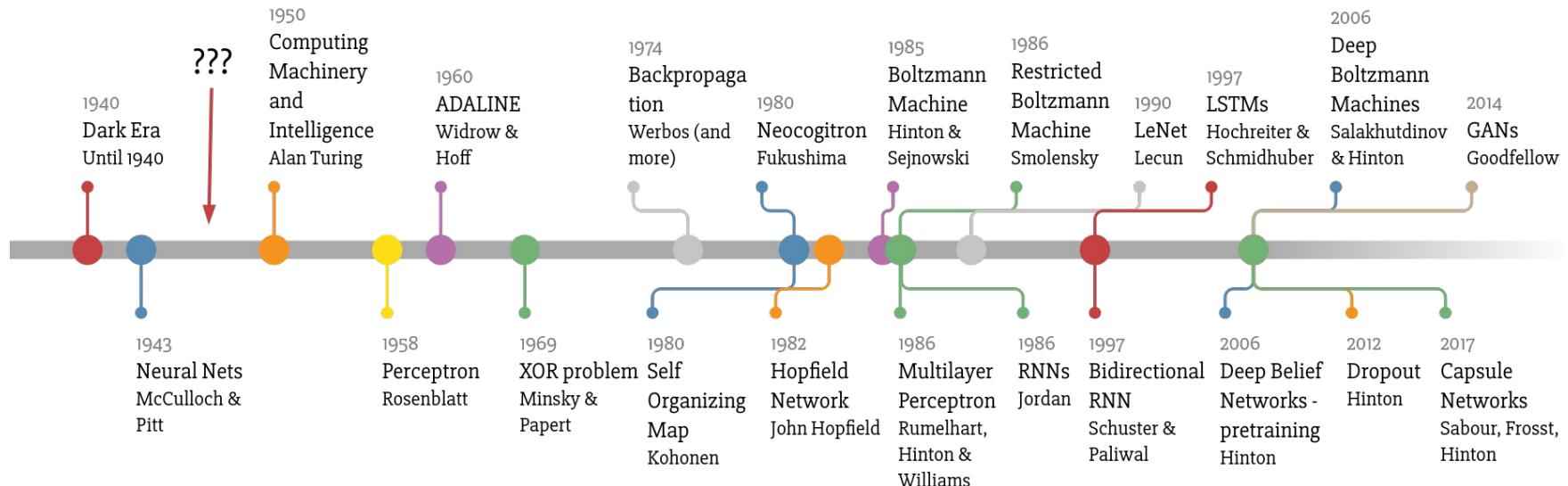


Transformer, BERT, GPT-2 and more, 2017+





Deep Learning Timeline





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**.

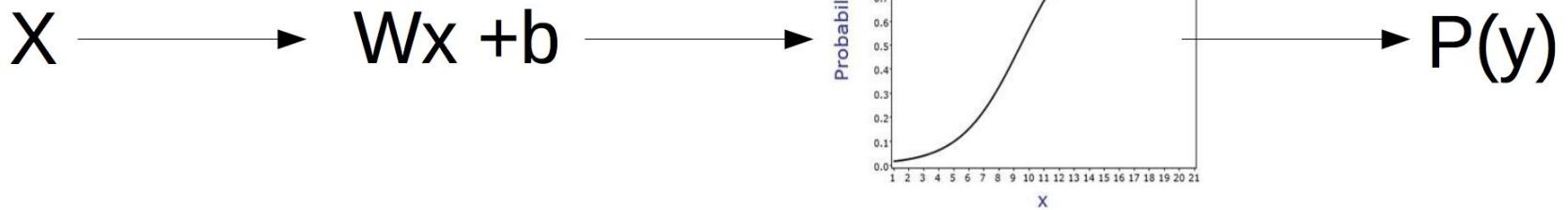
[Common knowledge site](#)

Neural networks intuition

girafe
ai

02

Logistic regression



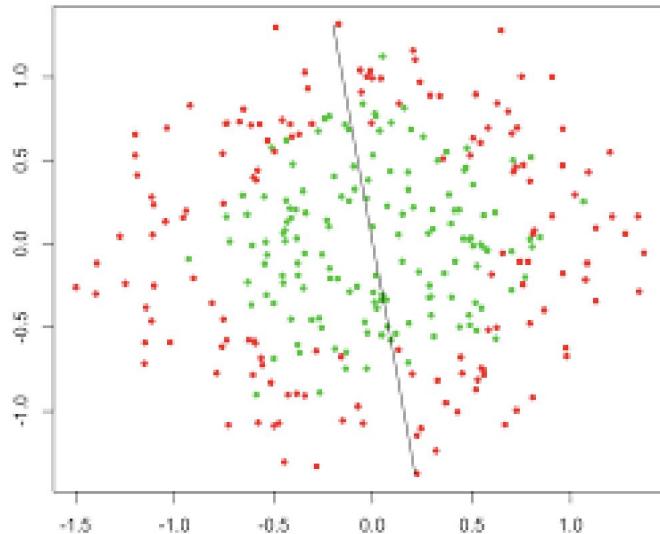
$$P(y|x) = \sigma(w \cdot x + b)$$

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

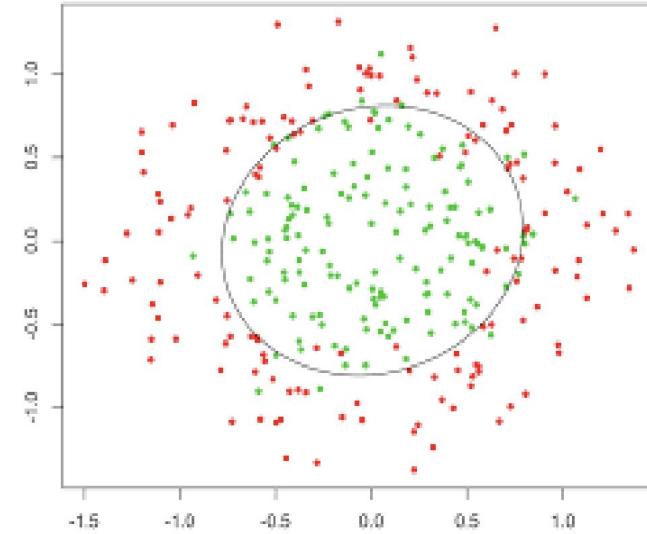
Problem: nonlinear dependencies



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



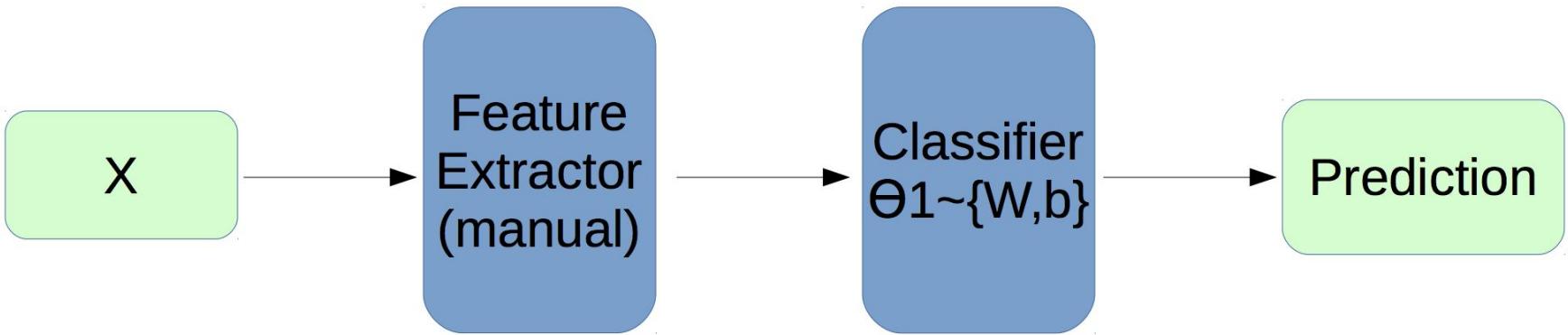
What we have



What we want

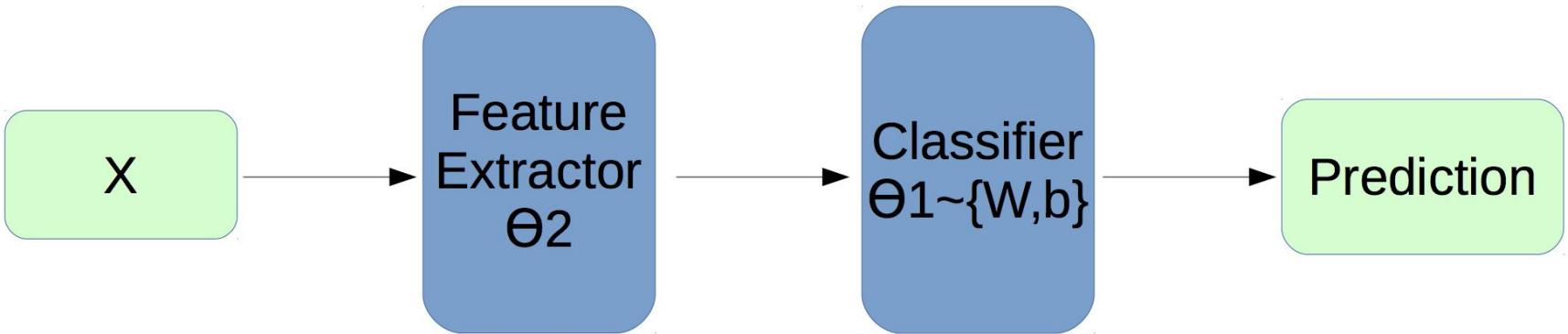
And feature engineering is an art.

Classical pipeline



Handcrafted features generated by experts.

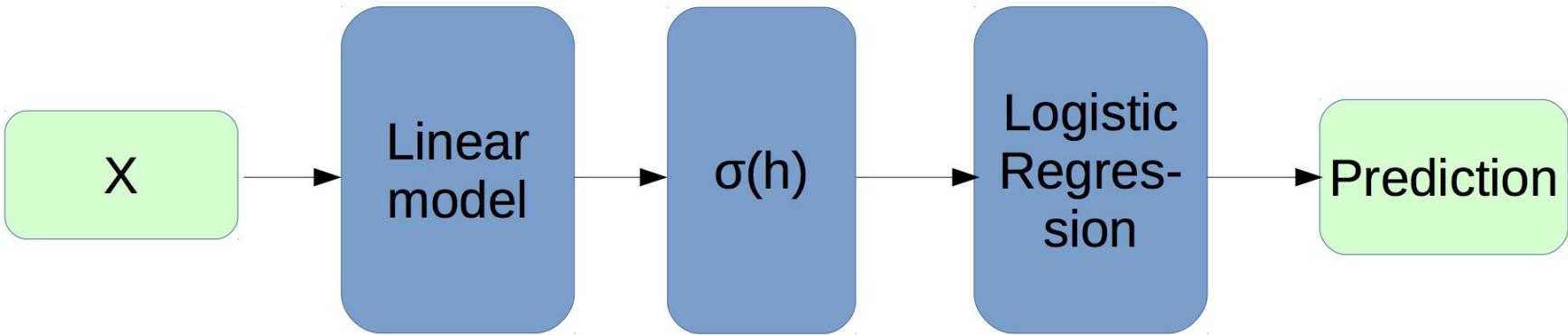
Parametrized pipeline



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)



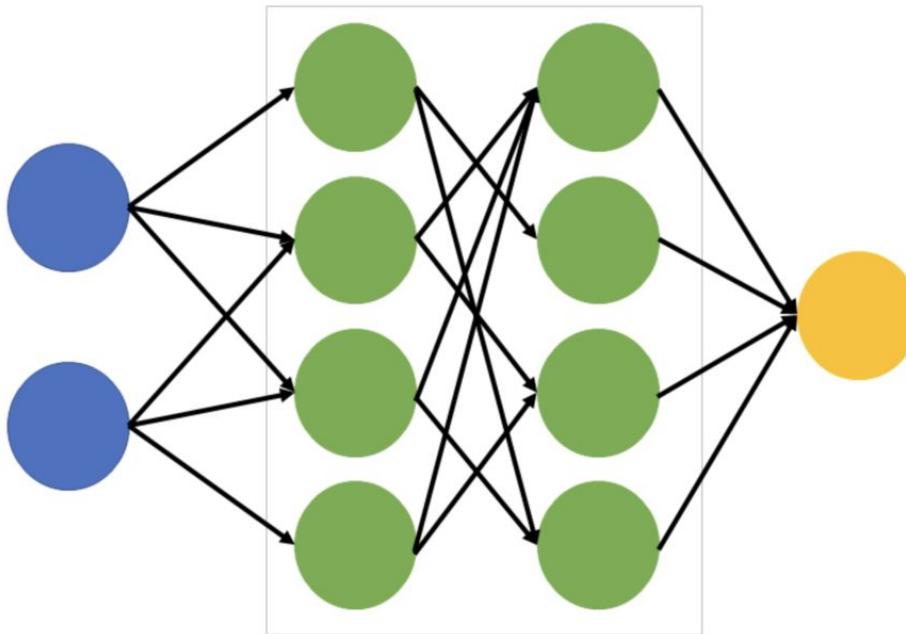
Graphical layer notation

How many layers are presented on the picture?

Input layer

Hidden layer

Output layer



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.

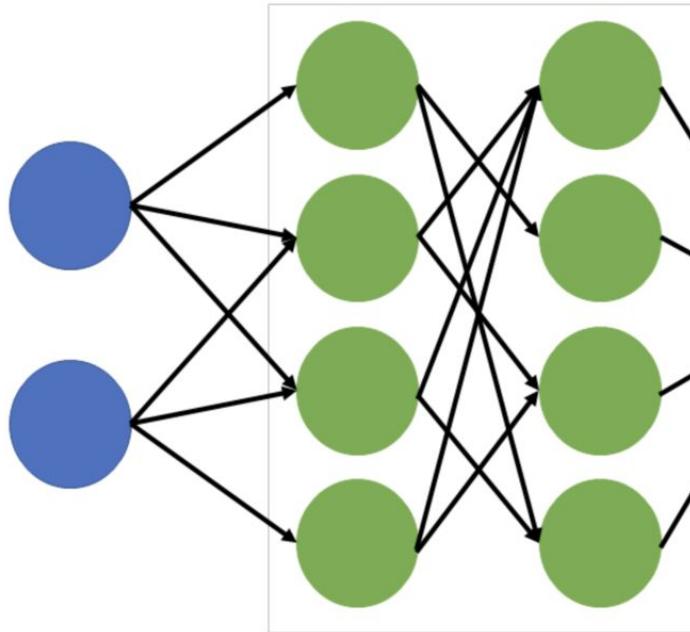
Layers are: Input, Output and Hidden.



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 \mathbb{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.

<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

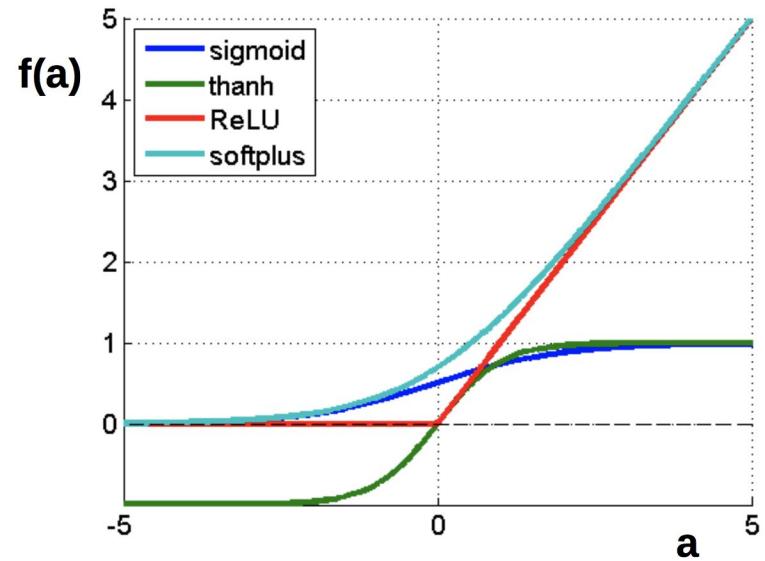


$$f(a) = \frac{1}{1 + e^{-a}}$$

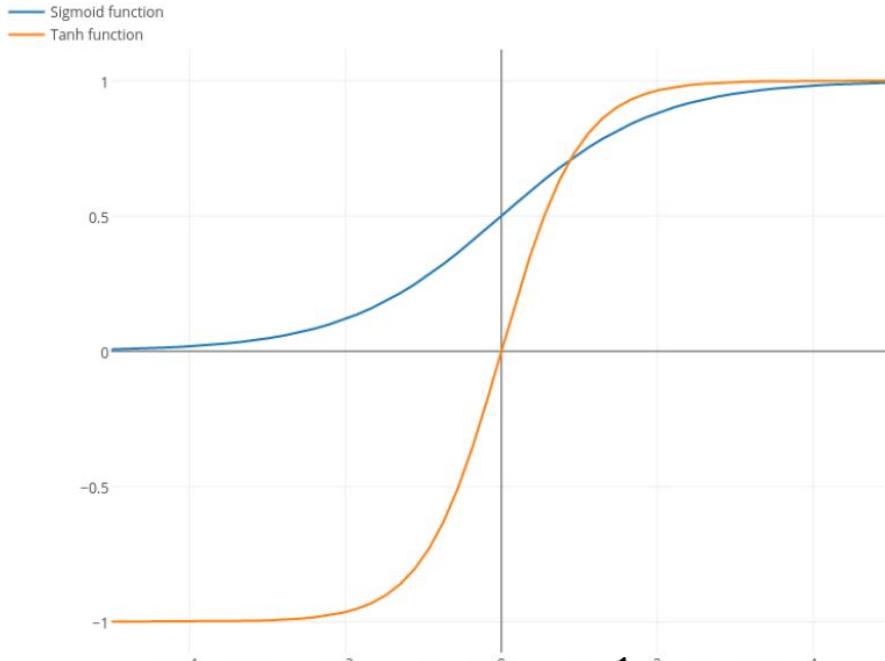
$$f(a) = \tanh(a)$$

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

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



Sigmoid



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

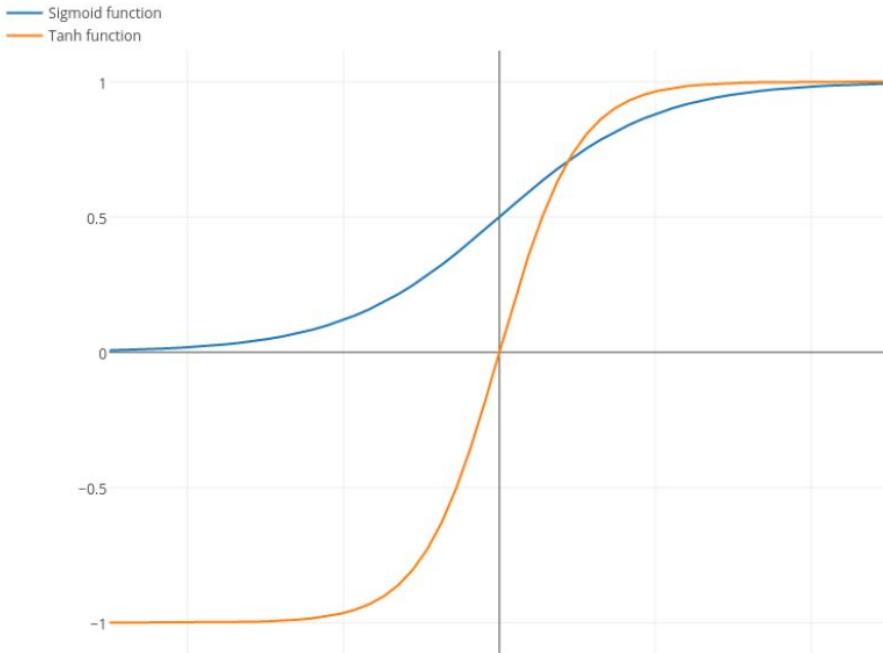
- Maps R to (0,1)
- Historically popular, one of the first approximations of neuron activation

Problems:

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



tanh



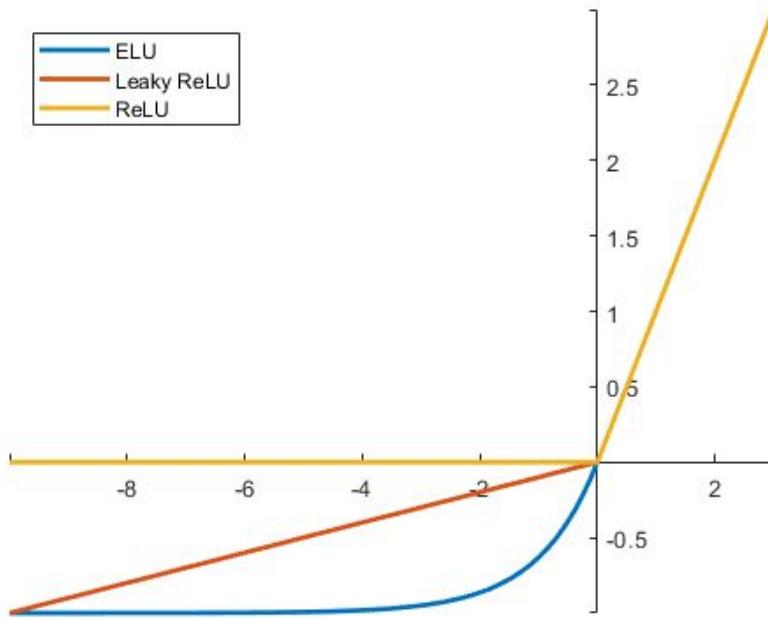
$$f(a) = \tanh(a)$$

- 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

ReLU



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

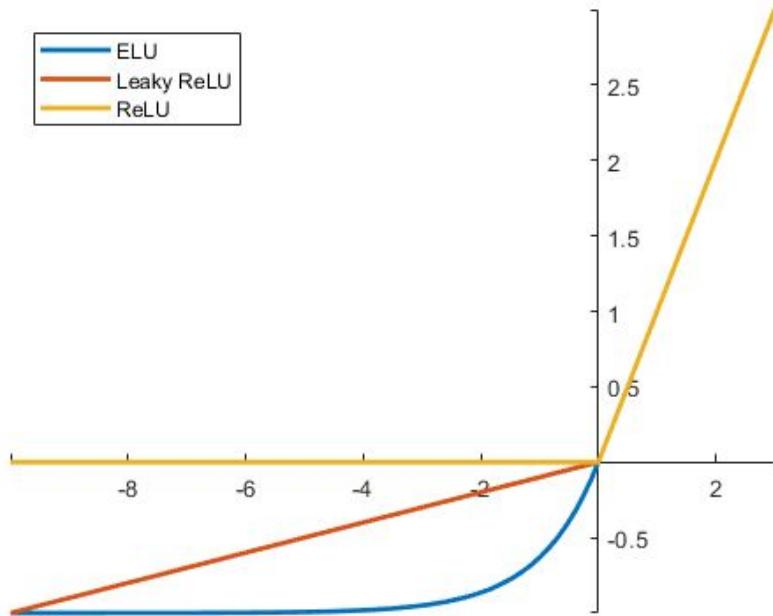
- Very simple to compute (both forward and backward)
 - Up to 6 times faster than Sigmoid
- Does not saturate when $x > 0$
 - So the gradients are not 0

Problems:

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



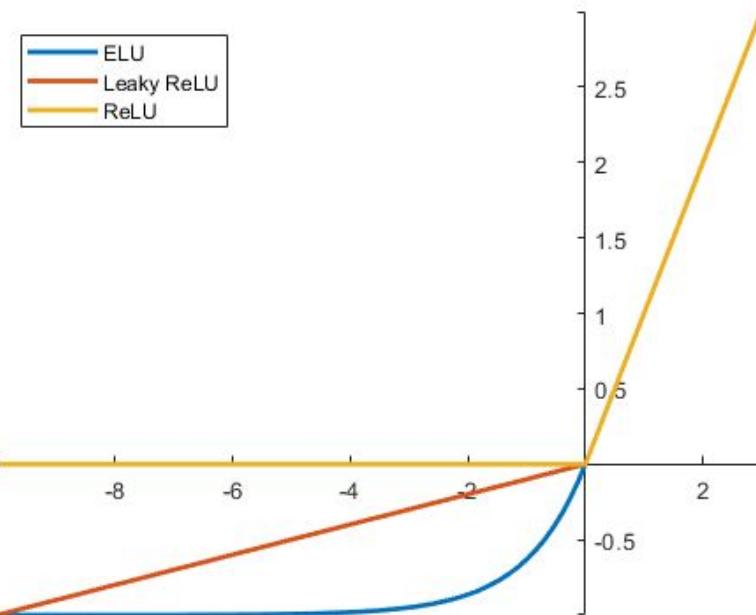
LeakyReLU



- 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



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

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

Problems:

- Requires exponent computation

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

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. $I(p_1 \cdot p_2) = I(p_1) + I(p_2)$: the information learned from independent events is the sum of the information learned from each event.

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

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



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

$$\begin{aligned} H(X) &= \mathbb{E}[I(X)] = \mathbb{E}[-\log p(X)] \\ &\quad - \sum_{x \in \mathcal{X}} p(x) \log_b p(x) \end{aligned}$$



Kullback–Leibler divergence

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

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



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

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

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

Fancy neural networks

girafe
ai

RNN



Shakespeare

PANDARUS:

Alas, I think he shall be come approached and the day
When little strain 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. \square

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

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

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

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

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

Lemma 0.2. This is an integer 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

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X,$$

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

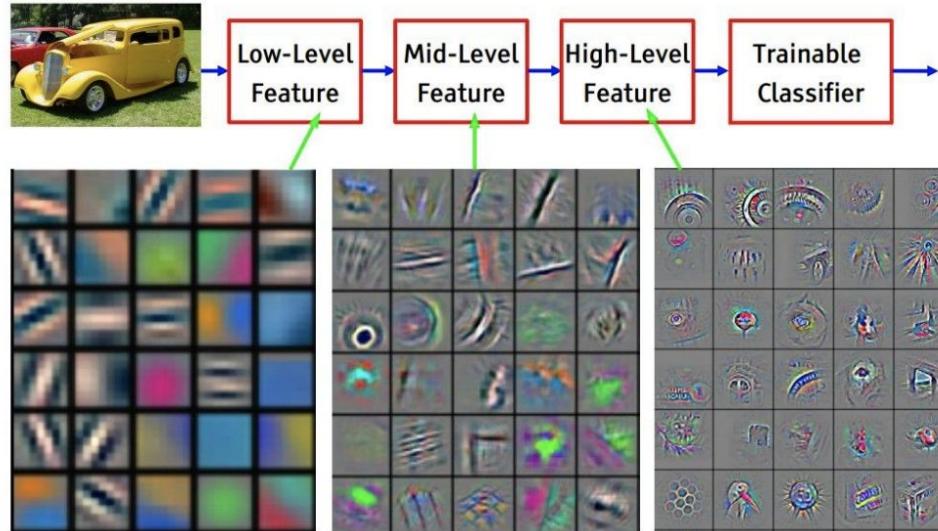
- (1) \mathcal{F} is an algebraic space over S .
- (2) If X is an affine open covering.

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

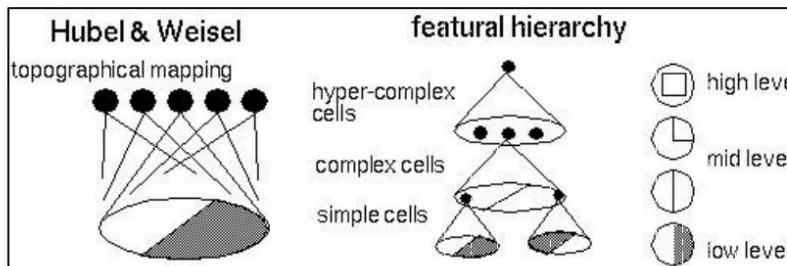
Linux kernel (source code)

```
/*
 * 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->add, *sys &-| (unsigned long) *FIRST_COMPAT);
    buf[0] = 0xFFFFFFFF & (bit << 4);
    min(inc, slist->bytes);
    printk(KERN_WARNING "Memory allocated %02x/%02x, "
           "original NLL instead\n"),
    min(min(multi_run - s->plen, max) * num_data_in),
    frame_pos, sz + first_seg);
    div_u64_w(val, imb_p);
    spin_unlock(&disk->queue_lock);
    mutex_unlock(&s->sock->mutex);
    mutex_unlock(&func->mutex);
    return disassemble(info->pending_bh);
}
```

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



DATA

Which dataset do you want to use?



Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

$X_1 X_2$

$\sin(X_1)$

$\sin(X_2)$

+

-

1 HIDDEN LAYER

+

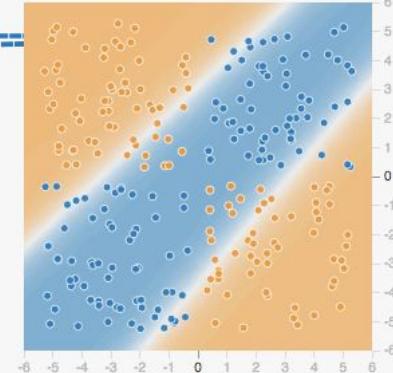
-

2 neurons

This is the output from one neuron.
Hover to see it larger.

OUTPUT

Test loss 0.208
Training loss 0.207



Colors shows data, neuron and weight values.

Show test data

Discretize output

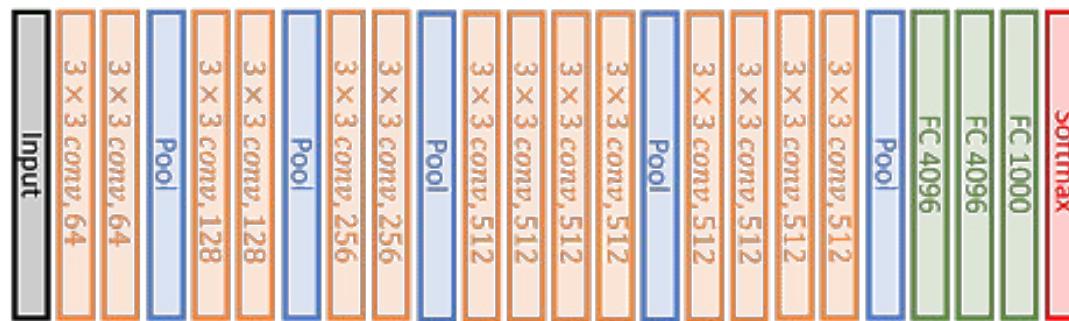
<https://playground.tensorflow.org/>



Actually, networks can be deep



And deeper...

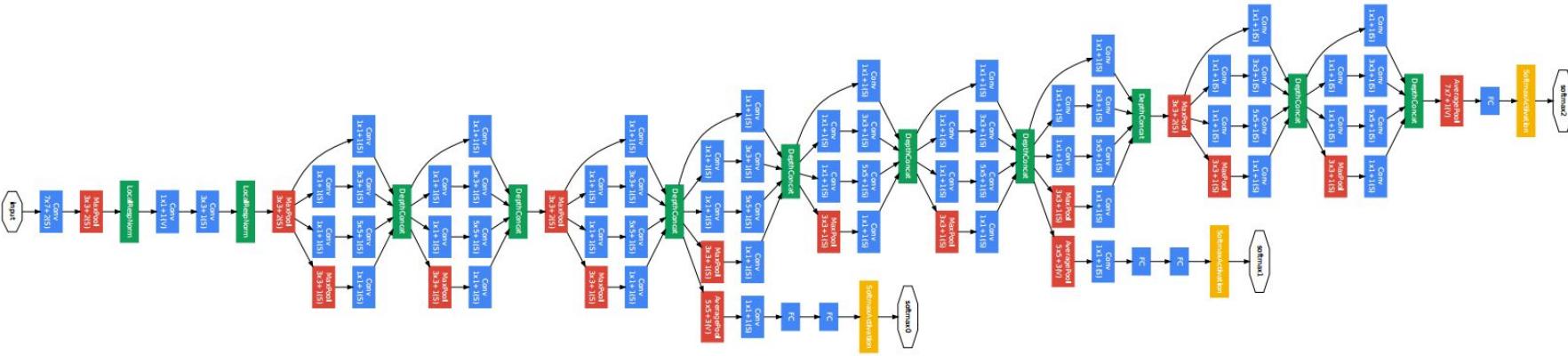


VGG16

VGG19



Much deeper...



How to train it?

Backpropagation

girafe
ai

06

Chain rule

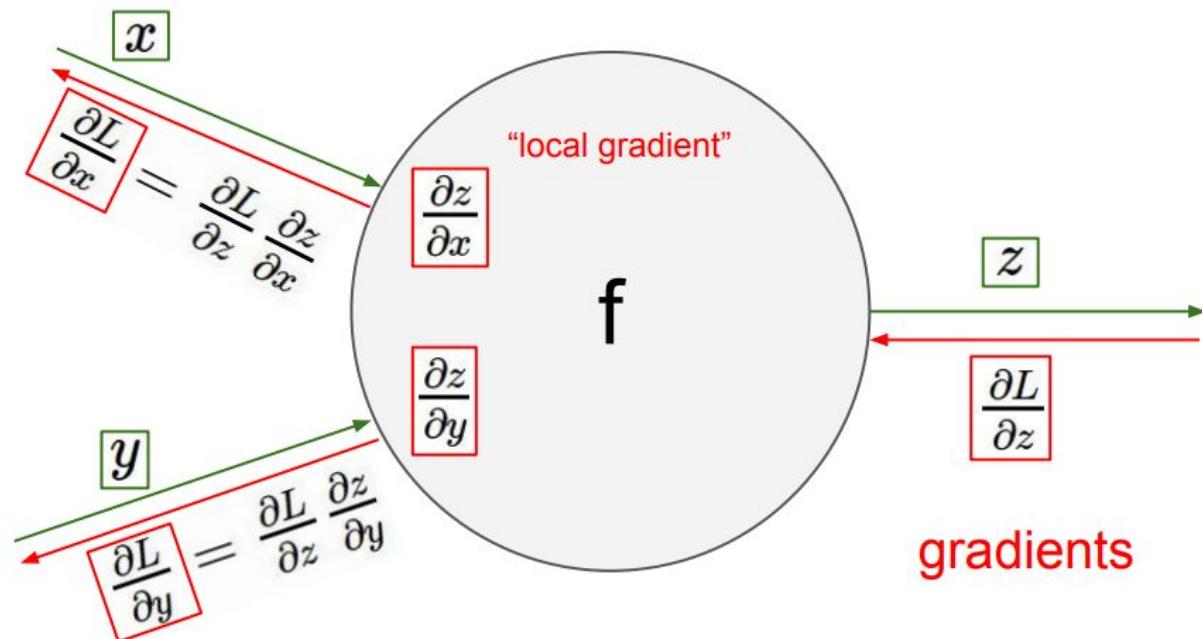


Chain rule is just simple math:

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

Backprop is just way to use it in NN training.

<http://cs231n.github.io/neutral-networks-3/>

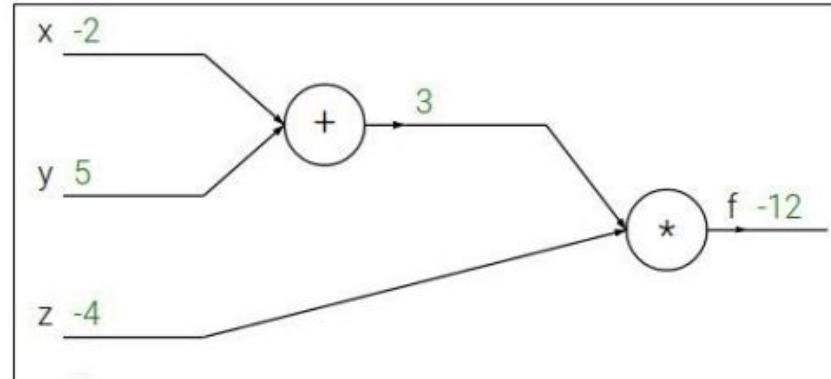


Backpropagation example



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

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





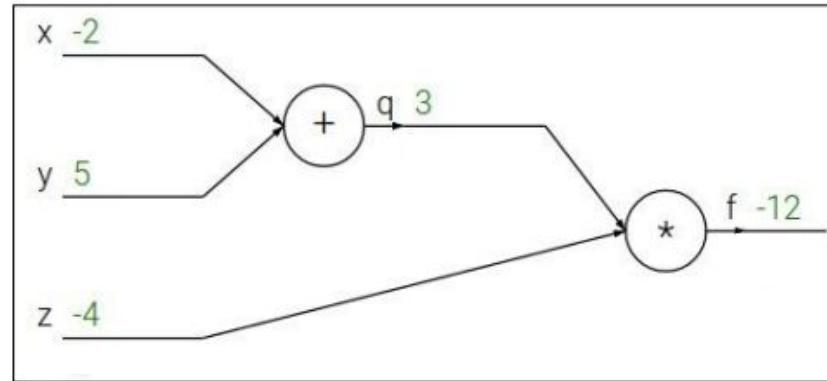
Backpropagation example

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

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

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

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



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

Backpropagation example



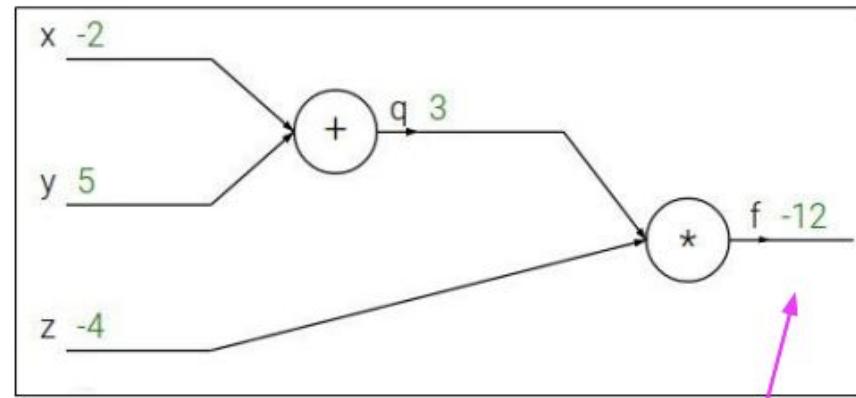
$$f(x, y, z) = (x + y)z$$

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

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

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

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



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

Backpropagation example

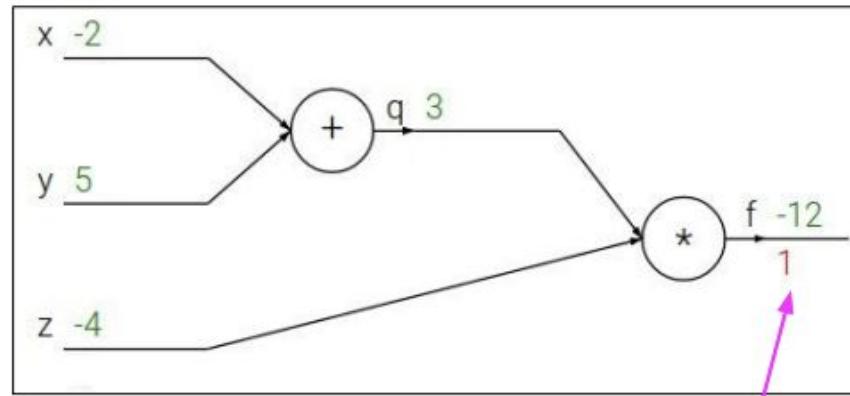


$$f(x, y, z) = (x + y)z$$

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

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

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



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



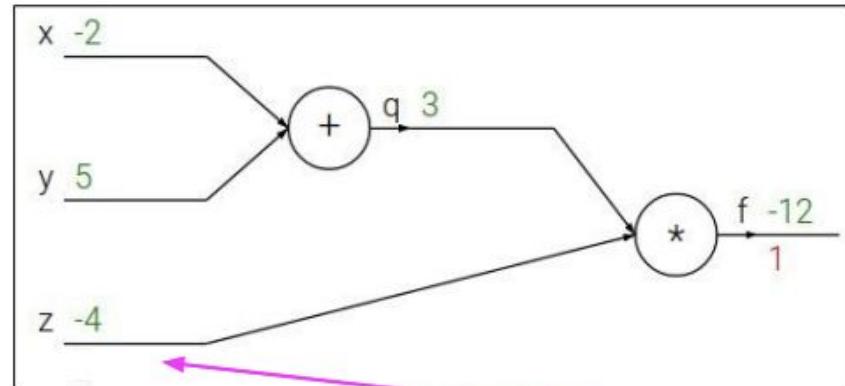
Backpropagation example

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

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

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

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



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

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



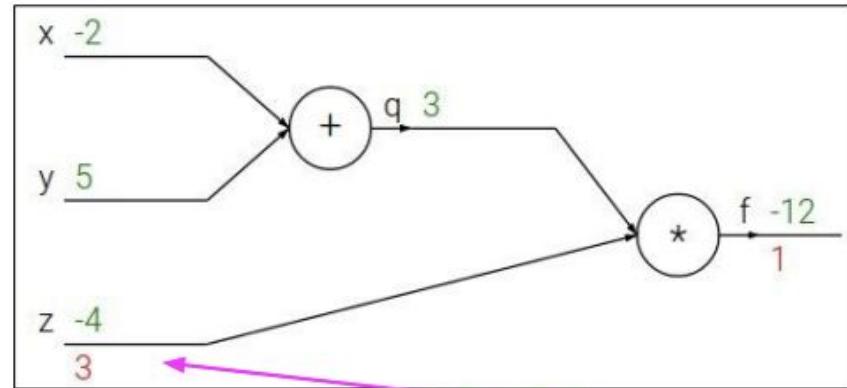
Backpropagation example

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

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

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

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



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

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



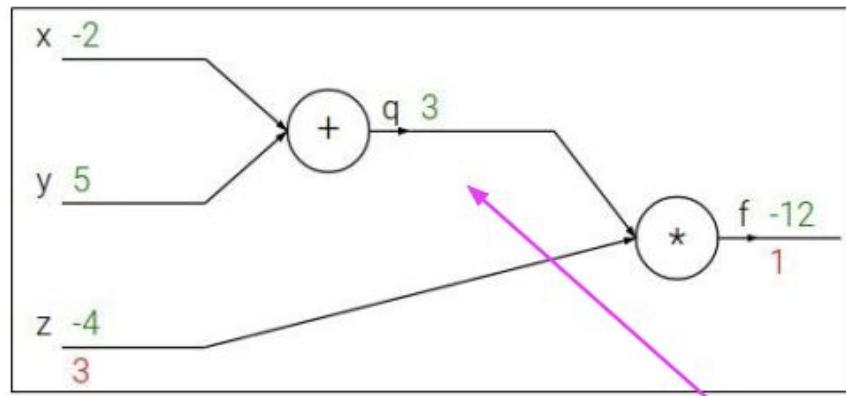
Backpropagation example

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

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

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

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



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

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



Backpropagation example

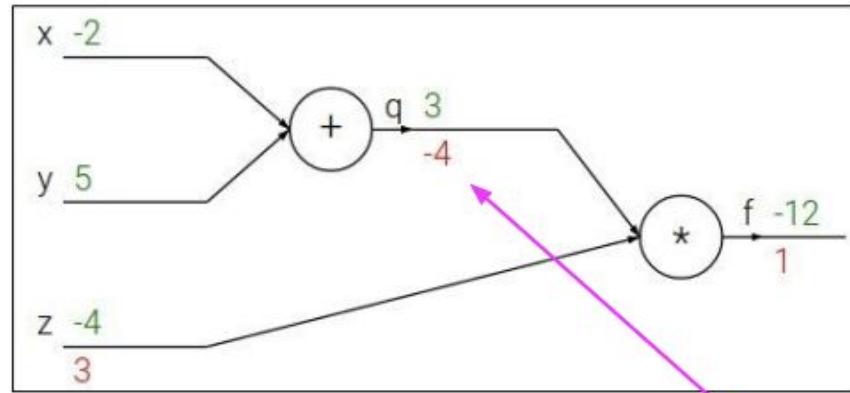
$$f(x, y, z) = (x + y)z$$

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

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

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

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



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

Backpropagation example

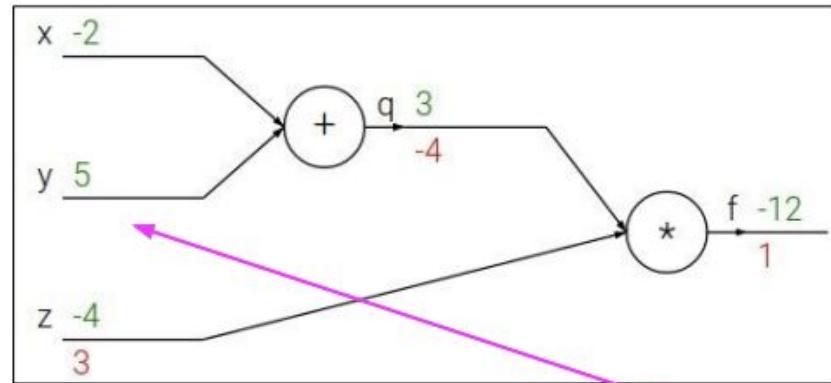


$$f(x, y, z) = (x + y)z$$

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

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

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



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

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



Backpropagation example

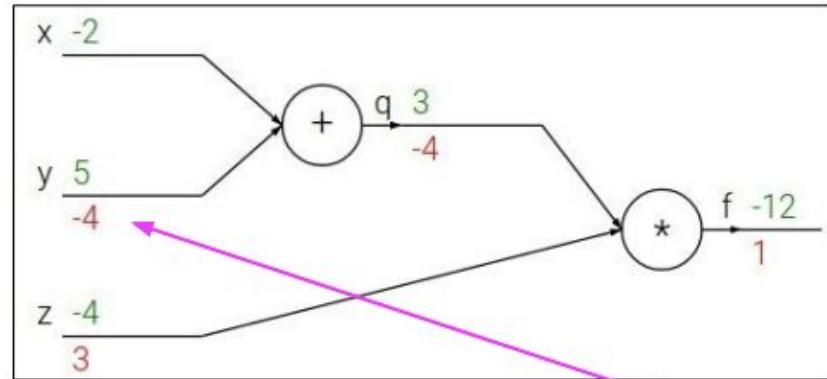
$$f(x, y, z) = (x + y)z$$

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

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

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

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



Chain rule:

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

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



Backpropagation example

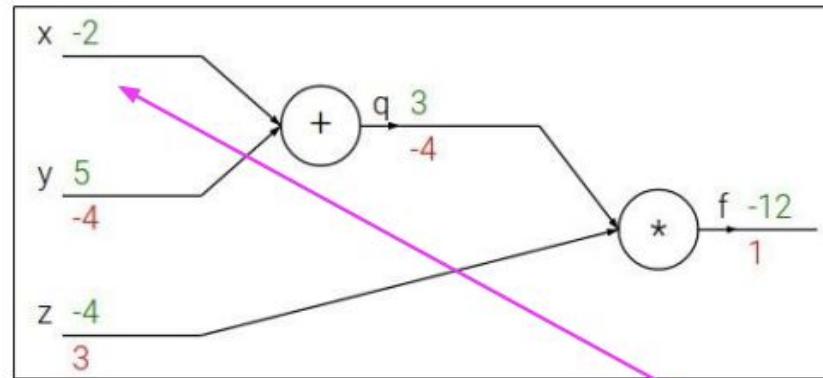
$$f(x, y, z) = (x + y)z$$

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

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

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

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



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

Backpropagation example



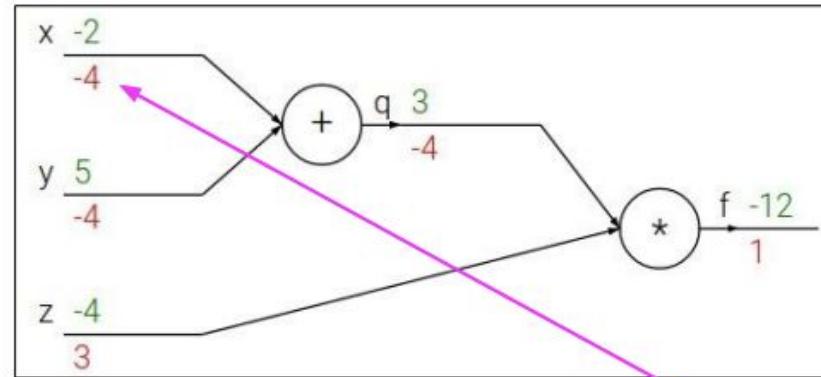
$$f(x, y, z) = (x + y)z$$

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

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

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

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



Chain rule:

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

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

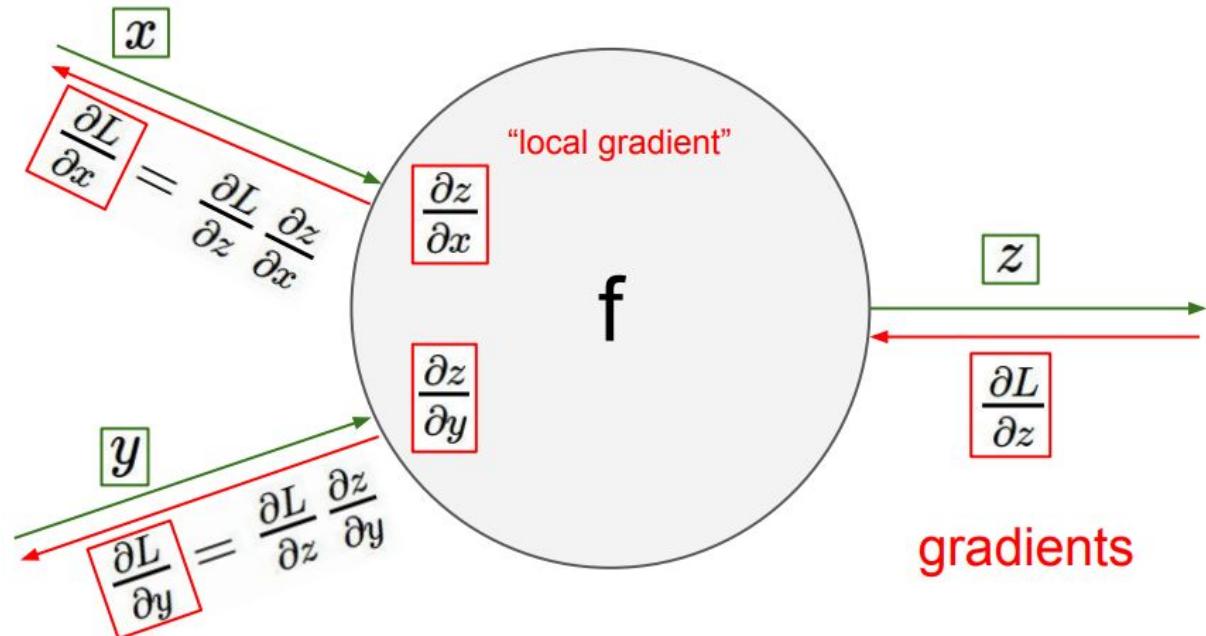
Backpropagation and chain rule



Chain rule is just simple math:

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

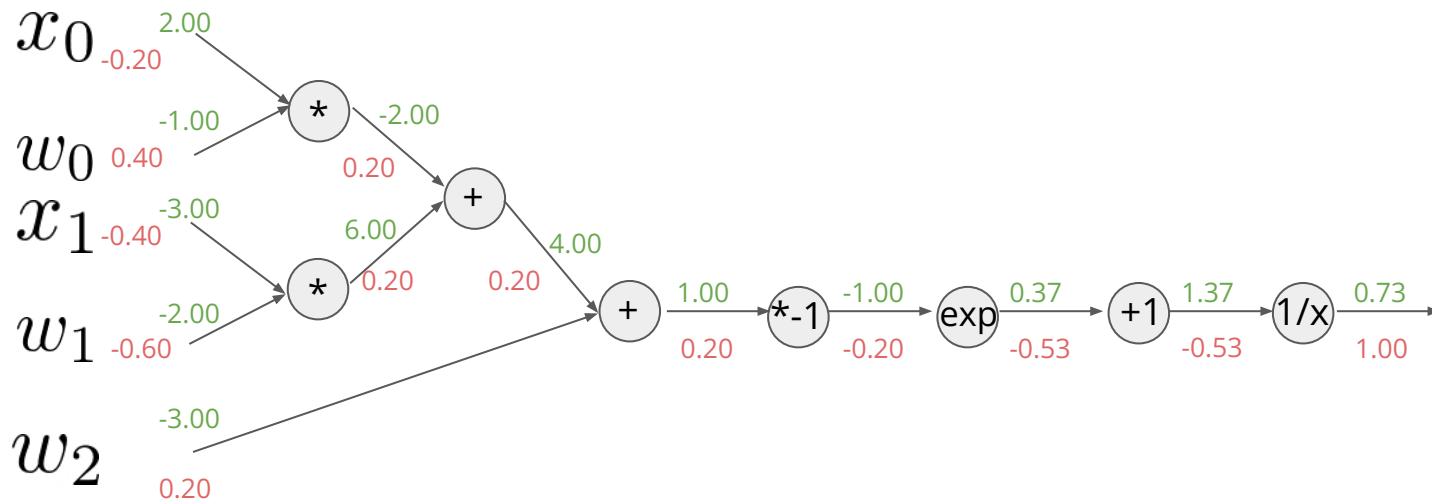
Backprop is just way to use it in NN training.





Backpropagation in LogReg

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





Backpropagation matrix form

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

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

Backpropagation matrix form



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



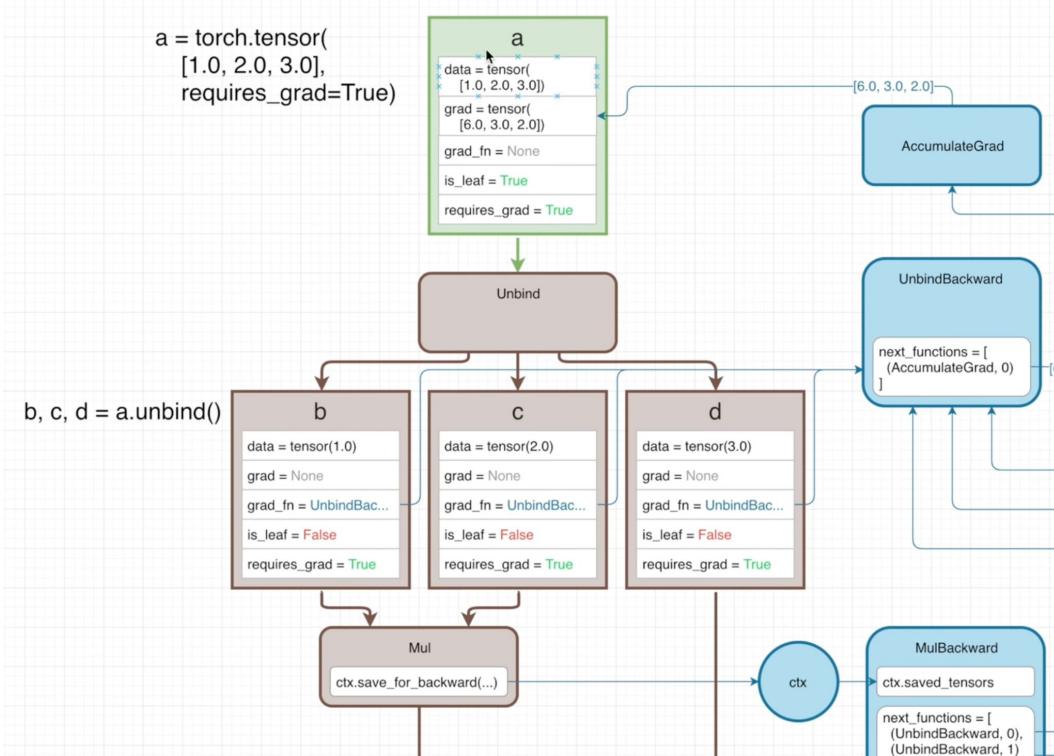
Derivative of vector by itself

$$\begin{aligned}
 y_1 &= f_1(\mathbf{x}) = x_1 & \frac{\partial \mathbf{y}}{\partial \mathbf{x}} &= \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}} f_1(\mathbf{x}) \\ \frac{\partial}{\partial \mathbf{x}} f_2(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial \mathbf{x}} f_m(\mathbf{x}) \end{bmatrix} &= \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\mathbf{x}) & \frac{\partial}{\partial x_2} f_1(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_1(\mathbf{x}) \\ \frac{\partial}{\partial x_1} f_2(\mathbf{x}) & \frac{\partial}{\partial x_2} f_2(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_2(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} f_m(\mathbf{x}) & \frac{\partial}{\partial x_2} f_m(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_m(\mathbf{x}) \end{bmatrix} \\
 y_2 &= f_2(\mathbf{x}) = x_2 \\
 \vdots \\
 y_n &= f_n(\mathbf{x}) = x_n \\
 && &= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & \frac{\partial}{\partial x_2} x_1 & \dots & \frac{\partial}{\partial x_n} x_1 \\ \frac{\partial}{\partial x_1} x_2 & \frac{\partial}{\partial x_2} x_2 & \dots & \frac{\partial}{\partial x_n} x_2 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} x_n & \frac{\partial}{\partial x_2} x_n & \dots & \frac{\partial}{\partial x_n} x_n \end{bmatrix} \\
 && & \text{(and since } \frac{\partial}{\partial x_j} x_i = 0 \text{ for } j \neq i) \\
 && &= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & 0 & \dots & 0 \\ 0 & \frac{\partial}{\partial x_2} x_2 & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & \frac{\partial}{\partial x_n} x_n \end{bmatrix} \\
 && &= \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & 1 \end{bmatrix} \\
 && &= I \quad (I \text{ is the identity matrix with ones down the diagonal})
 \end{aligned}$$



Autograd in Pytorch

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

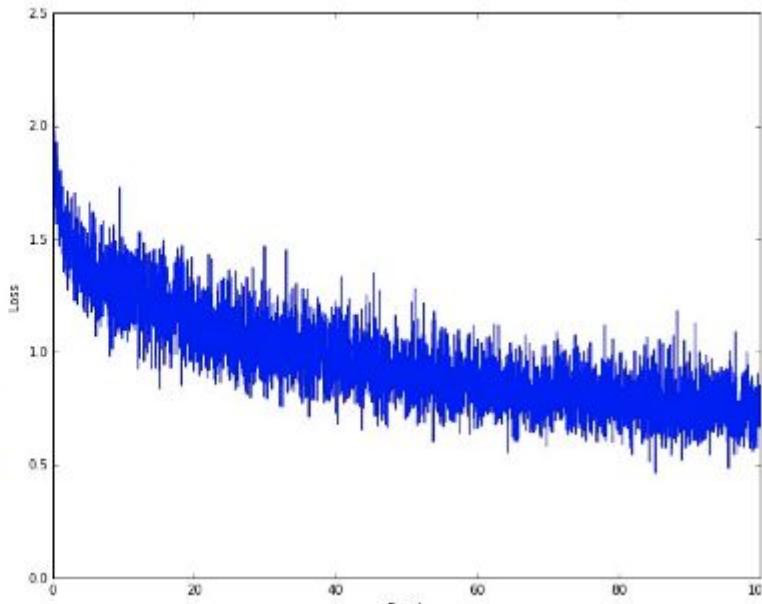
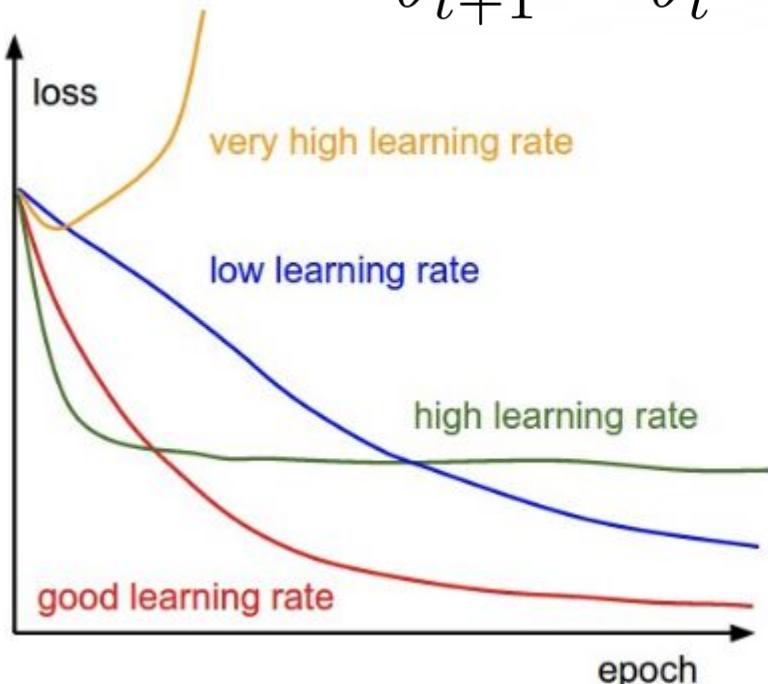




Gradient optimization

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





Tensorflow

Previous king of the hill.

Originally developed by Google since 2015.

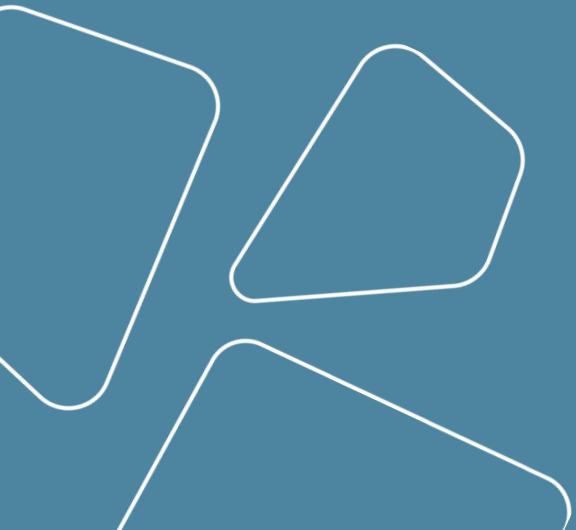
Written in C++, has 186k starts on Github (2024)

Apache License 2.0



TensorFlow

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?



girafe
ai

