

# Machine Learning: short introduction (trees and ANN), part 3

Sergey Korpachev on behalf of Dépôt

# Outline

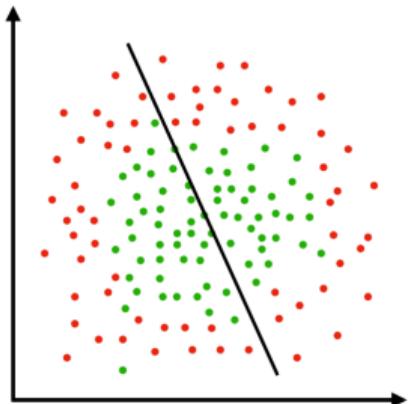
- Machine learning (ML)
  - Data
  - Features in ML
    - ML pipeline
    - Linear model
    - Decision tree
  - Neural network
  - Summary

# Modelling nonlinearities

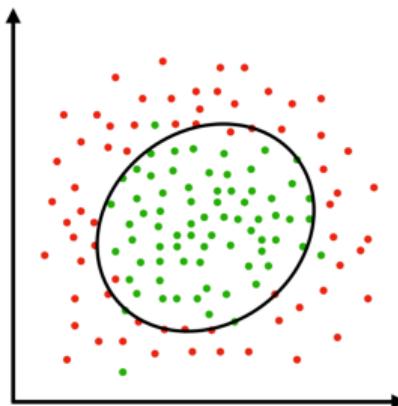
# Modelling nonlinearities

linear models

What we have



What we want

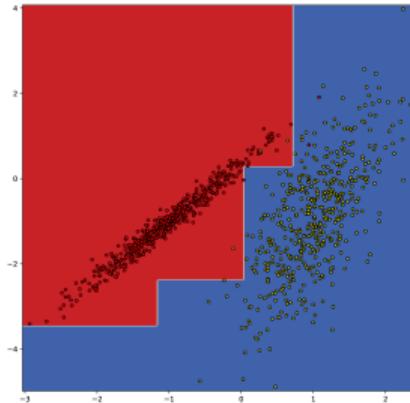
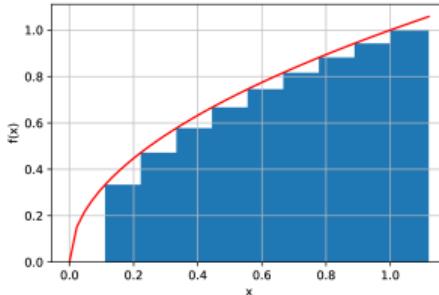


Linear models can't simply describe complex nonlinear data

# Modelling nonlinearities

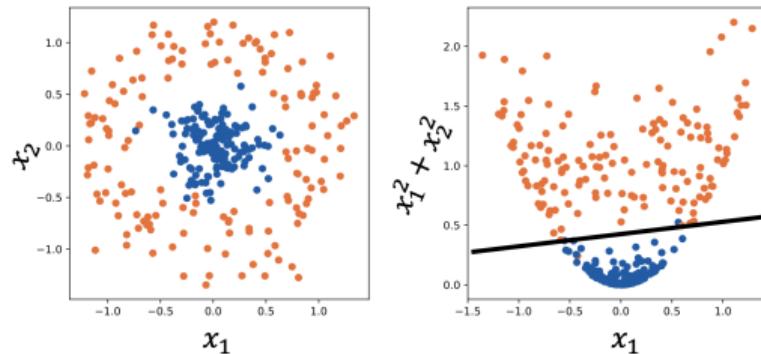
—○ trees

- (Ensembles of) Trees were designed to approximate nonlinearities and are **pretty good** in it + they are **fast and interpretable**
- But they are just “brute-force” algorithms – **don’t infer symmetries** in data by design
- **Ad-hoc, cut-based and piecewise approximations** of data at hand + not differentiable and smooth



# Modelling nonlinearities

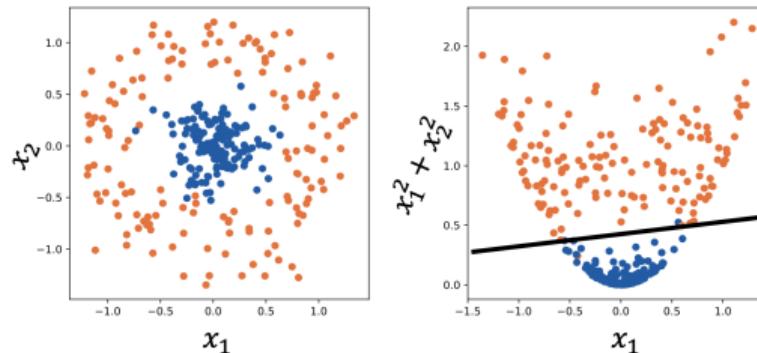
—○ feature engineering



- But sometimes we know *a priori* that there are transformations simplifying the problem → even linear model can do the job
- However, this **feature engineering** is non-trivial, requires domain knowledge and is time-consuming

# Modelling nonlinearities

—○ feature engineering



- But sometimes we know *a priori* that there are transformations simplifying the problem → even linear model can do the job
- However, this **feature engineering** is non-trivial, requires domain knowledge and is time-consuming

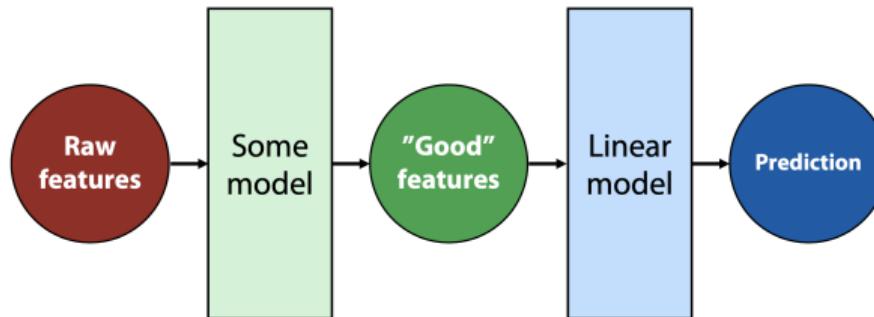
What if we design a model which could **automatically** feature-engineer itself?

# Neural Network

# Neural Network

—○ automating FE

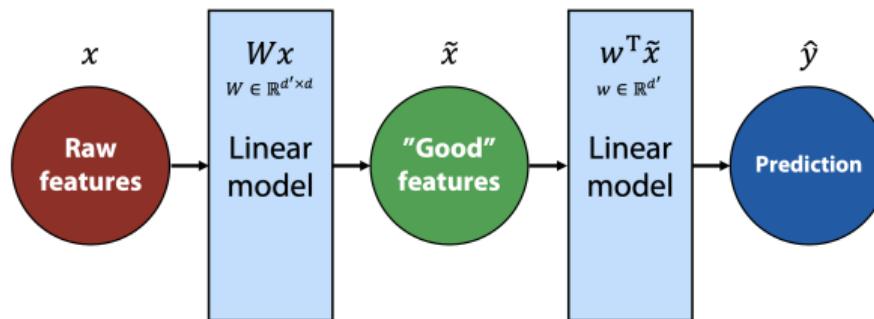
NN illustrations from  
ML in HEP 2020



- Let's use a **simple linear model** to solve our supervised problem
- Add a block to a linear model which will automatically **generate new features** for it
- Two blocks would work together as a **single model** ⇒ their parameters are updated simultaneously
- And **automatically**, by e.g. gradient descent (given their differentiability)

# Neural Network

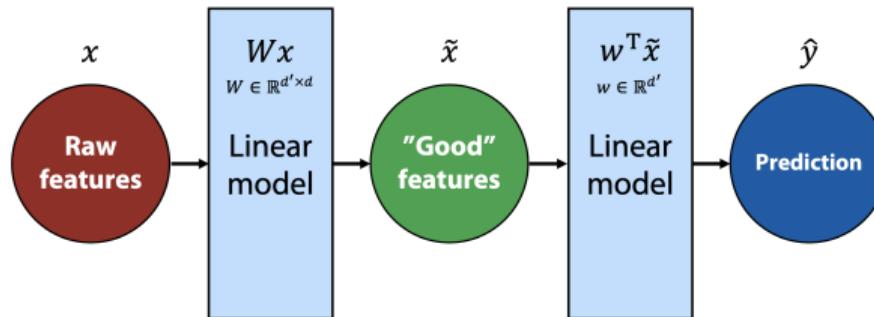
—○ automating FE



- Would a linear model work as a feature generating model?

# Neural Network

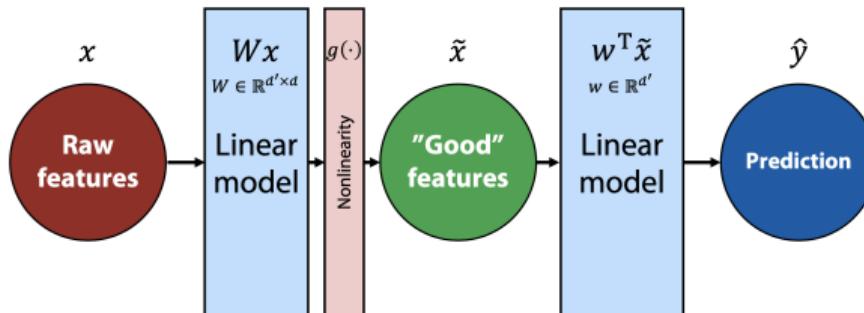
—○ automating FE



- Would a linear model work as a feature generating model? **No**
- $\hat{y} = w^T \tilde{x} = w^T (Wx) = (w^T W)x = w'^T x \Rightarrow$  it is still a linear model
- Input feature space has not changed, only the model weights

# Neural Network

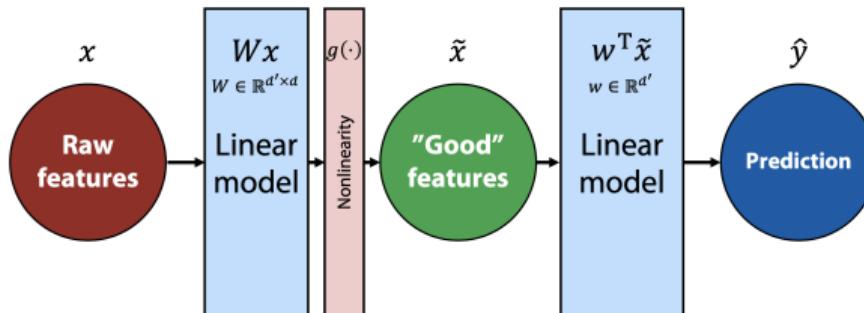
—○ automating FE



- Let's then introduce **nonlinearity** to our model  
→  $\hat{y} = w^T \tilde{x} = w^T g(Wx)$ ,  
where  $g(\cdot)$  – some nonlinear scalar function (applied elementwise)

# Neural Network

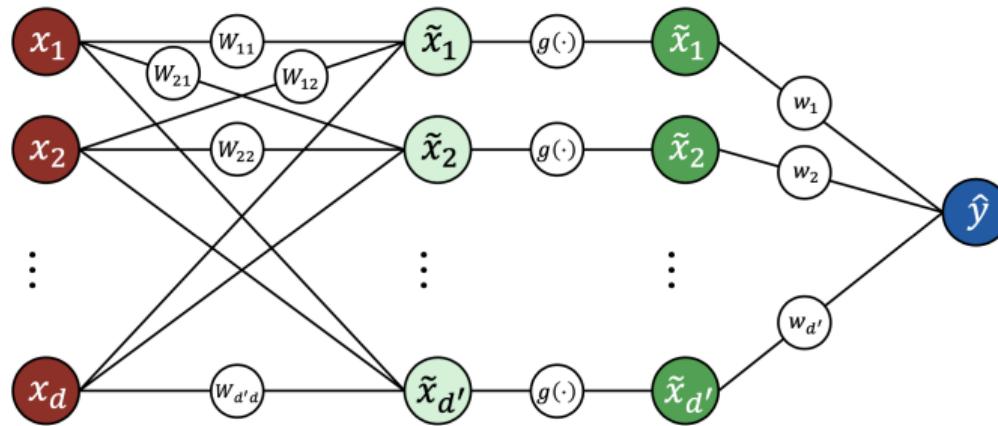
—○ automating FE



- Let's then introduce **nonlinearity** to our model
  - $\hat{y} = w^T \tilde{x} = w^T g(Wx)$ ,  
where  $g(\cdot)$  – some nonlinear scalar function (applied elementwise)
  - This is the simplest example of a **neural network**

# Neural Network

—○ architecture



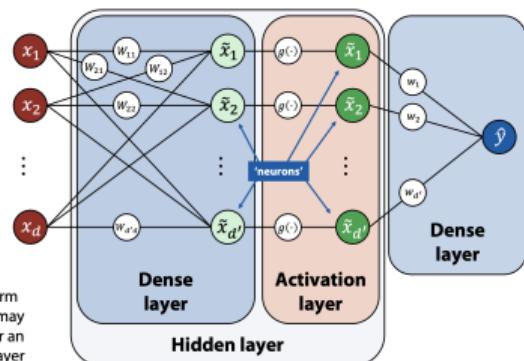
Essentially, NN is just a **composite function** that maps a set of  $X$  to a set of  $Y$

$$\hat{y} = w^T \tilde{x} = w^T g(Wx)$$

# Neural Network

—○ terminology

## Feed-forward network:



Note: the term "activation" may also stand for an output of a layer

- Brown nodes  $x_1, x_2, \dots, x_d$  – features from an **input layer**
- Green nodes  $\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_{d'}$  – **neurons** from a **hidden layer**
- Blue node  $\hat{y}$  – neuron from an **output layer**
- Straight lines (edges) between neurons – **weights**  $w_{ij}$
- $g(\cdot)$  – nonlinear **activation** function, e.g. sigmoid  $\sigma(\cdot)$
- **Important:** each neuron has additional **bias**  $b$  associated to it and added to other inputs (not illustrated)

# Training

# Training —— o how to train?

- Since NN fundamentally is a parametrized differentiable model we can optimise the loss function and use **gradient descent** to train it:

$$\omega_{k+1} \leftarrow \omega_k - \eta \cdot \nabla Q(\omega_k)$$

- But **gradients** are hard to derive analytically – writing down all the derivatives is tough and tedious (especially for large NN)
- Note that NN is just a **composite model**  $\Rightarrow$  can use **chain rule** for differentiating it

# Training —— o chain rule

- Computing derivative of a "base" function is simple  $\Rightarrow$  decompose composite function into a set of base ones and differentiate them one by one
- Let's recall the rule:

$$\frac{\partial f(t(x))}{\partial x} = \frac{\partial f(t)}{\partial t} \cdot \frac{\partial t(x)}{\partial x}$$

**Exercise:** can you compute derivative of sigmoid function by decomposing it into base functions?

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

# Training —— o chain rule

- Computing derivative of a "base" function is simple  $\Rightarrow$  decompose composite function into a set of base ones and differentiate them one by one
- Let's recall the rule:

$$\frac{\partial f(t(x))}{\partial x} = \frac{\partial f(t)}{\partial t} \cdot \frac{\partial t(x)}{\partial x}$$

**Exercise:** can you compute derivative of sigmoid function by decomposing it into base functions?

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

$$\sigma(x) = f(g(h(p(x)))), \quad f(z) = \frac{1}{z}, \quad g(z) = 1 + z, \quad h(z) = e^z, \quad p(z) = -z$$

# Training —— o chain rule

- Computing derivative of a "base" function is simple  $\Rightarrow$  decompose composite function into a set of base ones and differentiate them one by one
- Let's recall the rule:

$$\frac{\partial f(t(x))}{\partial x} = \frac{\partial f(t)}{\partial t} \cdot \frac{\partial t(x)}{\partial x}$$

**Exercise:** can you compute derivative of sigmoid function by decomposing it into base functions?

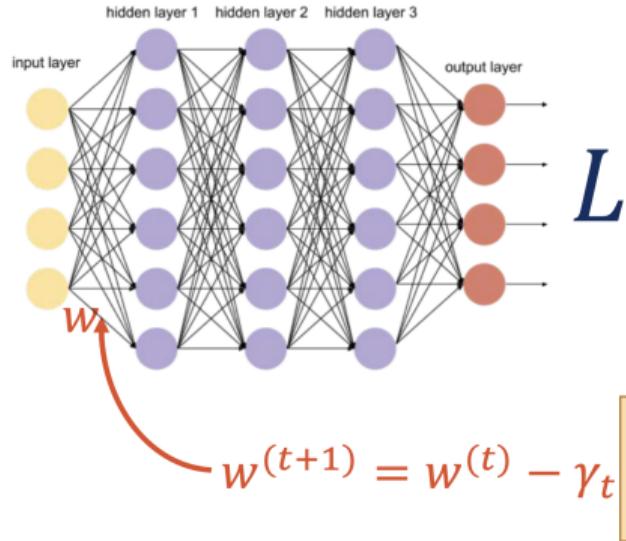
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\sigma(x) = f(g(h(p(x)))), \quad f(z) = \frac{1}{z}, \quad g(z) = 1 + z, \quad h(z) = e^z, \quad p(z) = -z$$

$$\sigma'(x) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial p} \cdot \frac{\partial p}{\partial x} = -\frac{1}{g^2} \cdot 1 \cdot e^p \cdot (-1) = \dots = \sigma(x) \cdot (1 - \sigma(x))$$

# Training — o backpropagation

backprop illustrations from [DMIA](#)

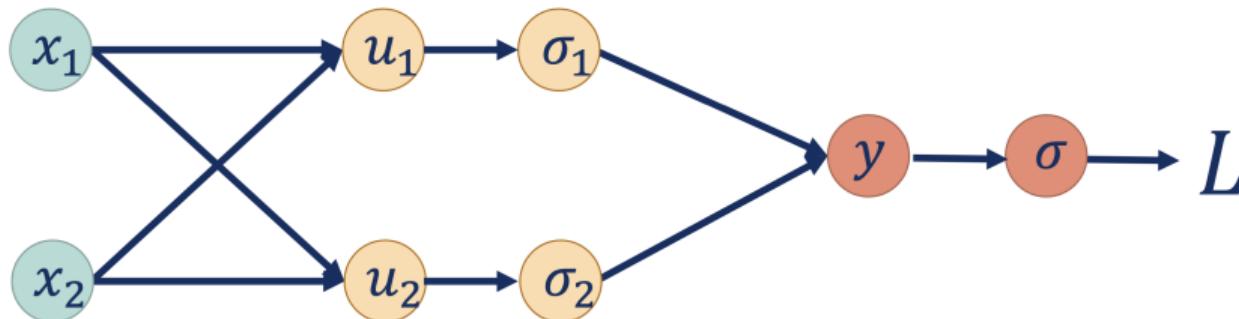


- So how to find partial derivatives of loss function with respect to some weight?

# Training

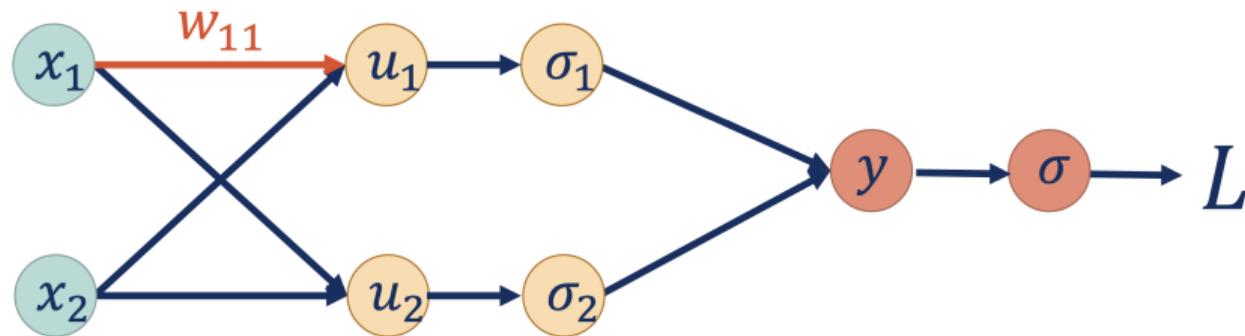
—○ backpropagation

backprop illustrations from [DMIA](#)



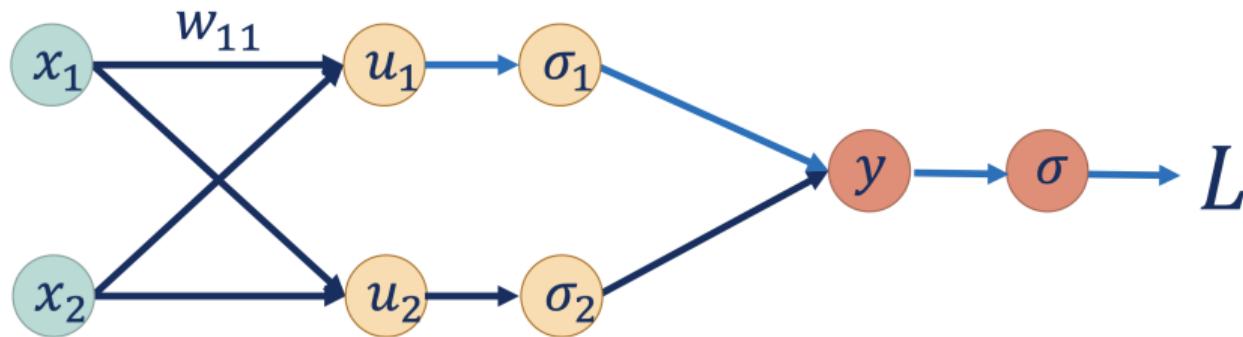
- Let's consider a simplified neural network
- Represent it in the form of a **computational graph**

Training —○ backpropagation



$$w_{11}^{(t+1)} = w_{11}^{(t)} - \gamma_t \frac{\partial L}{\partial w_{11}}$$

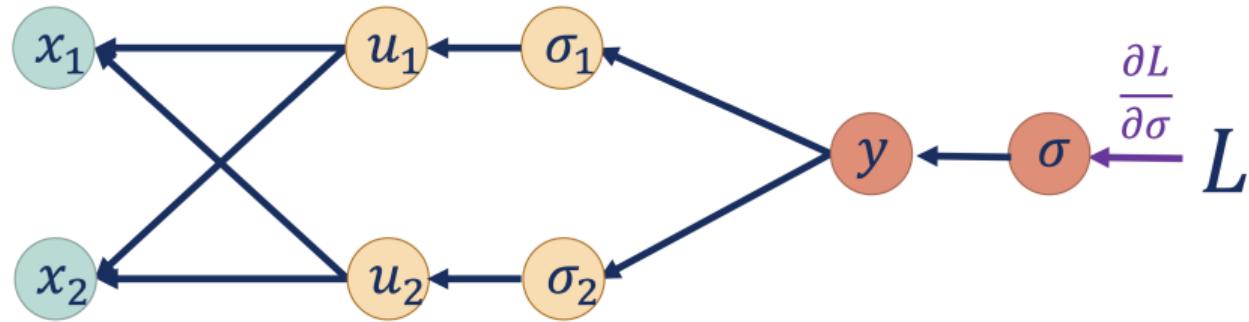
Training —○ backpropagation



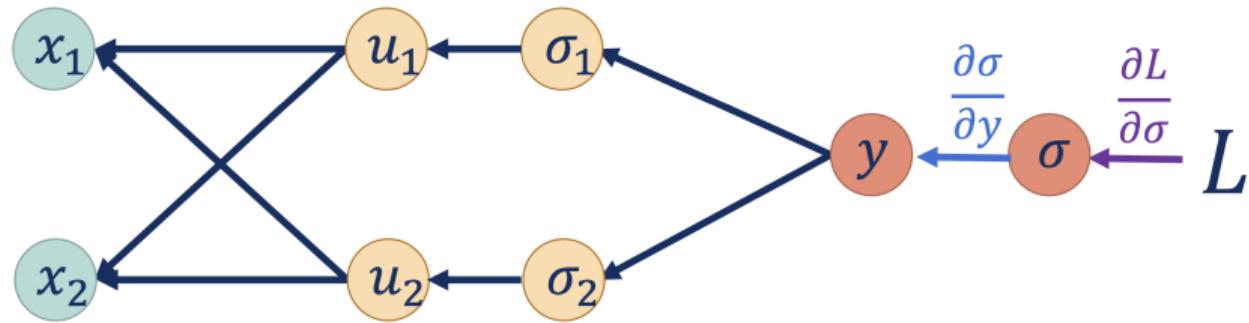
$$w_{11}^{(t+1)} = w_{11}^{(t)} - \gamma_t \frac{\partial L}{\partial w_{11}}$$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial u_1} \frac{\partial u_1}{\partial w_{11}} = \frac{\partial L}{\partial u_1} x_1$$

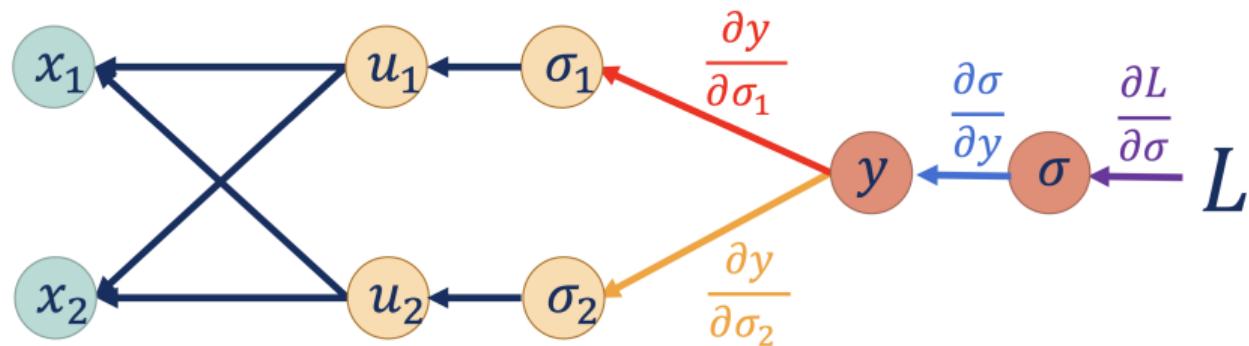
Training —○ backpropagation



Training —○ backpropagation

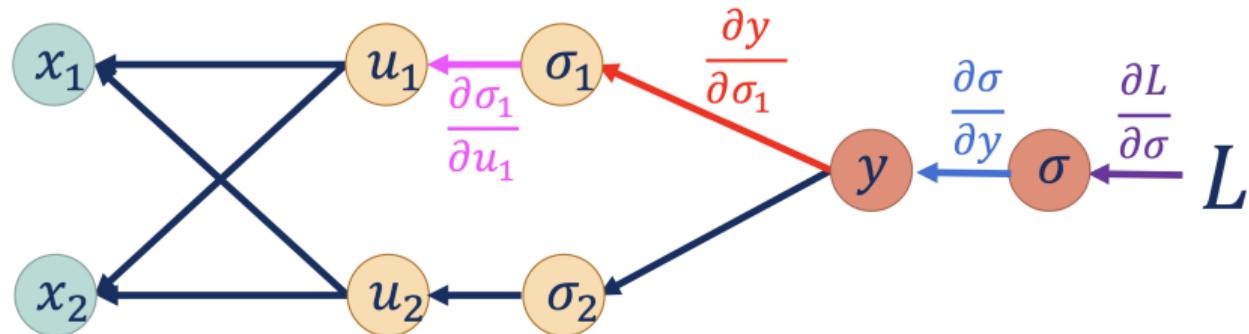


Training —○ backpropagation



# Training

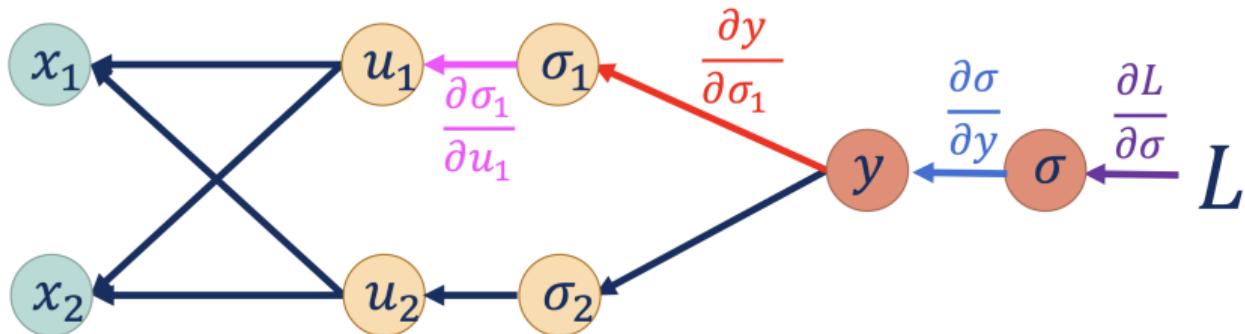
—○ backpropagation



$$\frac{\partial L}{\partial u_1} = \frac{\partial L}{\partial \sigma} \frac{\partial \sigma}{\partial y} \frac{\partial y}{\partial \sigma_1} \frac{\partial \sigma_1}{\partial u_1}$$

# Training

—○ backpropagation



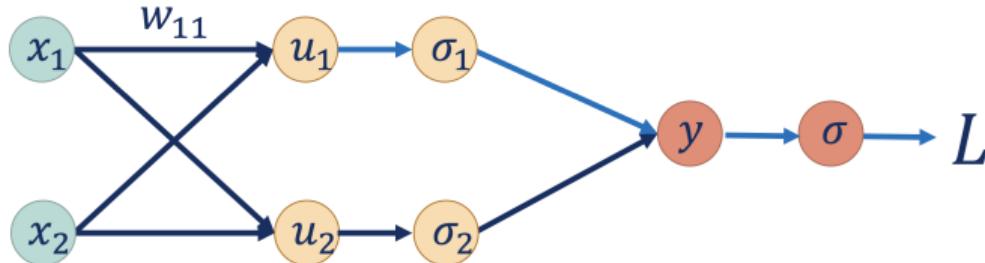
$$\frac{\partial L}{\partial u_1} = \frac{\partial L}{\partial \sigma} \frac{\partial \sigma}{\partial y} \frac{\partial y}{\partial \sigma_1} \frac{\partial \sigma_1}{\partial u_1}$$

- this procedure is called **backpropagation**
- its idea is to **collect derivatives** at each step in the computational graph w/o recalculating them every single time for every weight

# Training

—○ wrap it up

train NN in your browser!

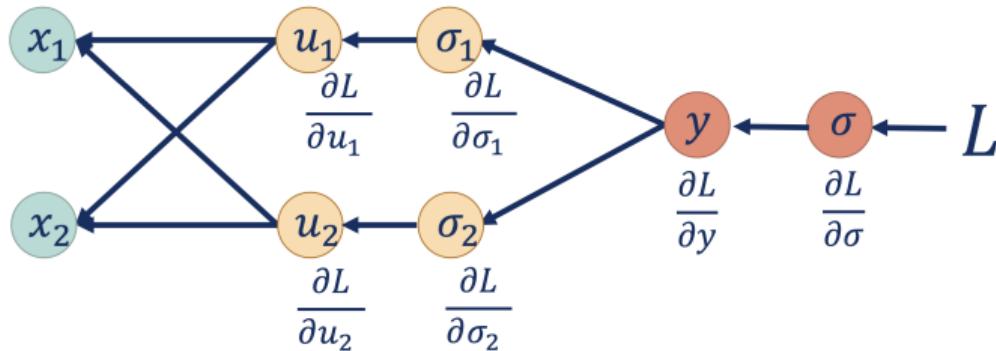


- 1 make **forward pass** through NN to calculate the output of each neuron and value of loss function

# Training

—○ wrap it up

[train NN in your browser!](#)

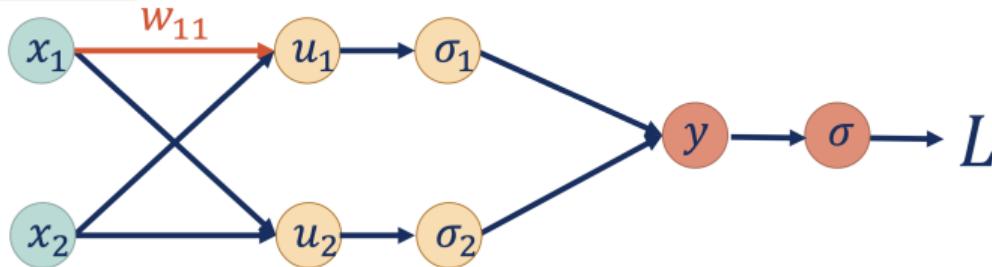


- 1 make **forward pass** through NN to calculate the output of each neuron and value of loss function
- 2 with **backward pass** go back through NN to calculate gradients for all weights

# Training

—○ wrap it up

train NN in your browser!



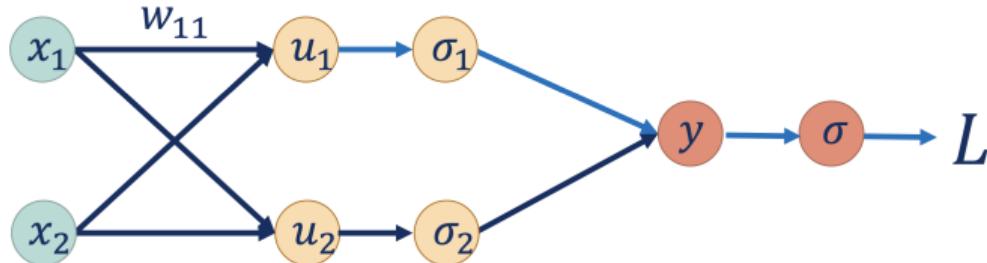
$$w_{11}^{(t+1)} = w_{11}^{(t)} - \gamma_t \frac{\partial L}{\partial w_{11}}$$

- 1 make **forward pass** through NN to calculate the output of each neuron and value of loss function
- 2 with **backward pass** go back through NN to calculate gradients for all weights
- 3 update weights with their gradients

# Training

—○ wrap it up

train NN in your browser!



- 1 make **forward pass** through NN to calculate the output of each neuron and value of loss function
- 2 with **backward pass** go back through NN to calculate gradients for all weights
- 3 update weights with their gradients
- 4 repeat until convergence

\*one iteration (**epoch**) = forward and backward pass

# Summary

## Summary — o Socratic method

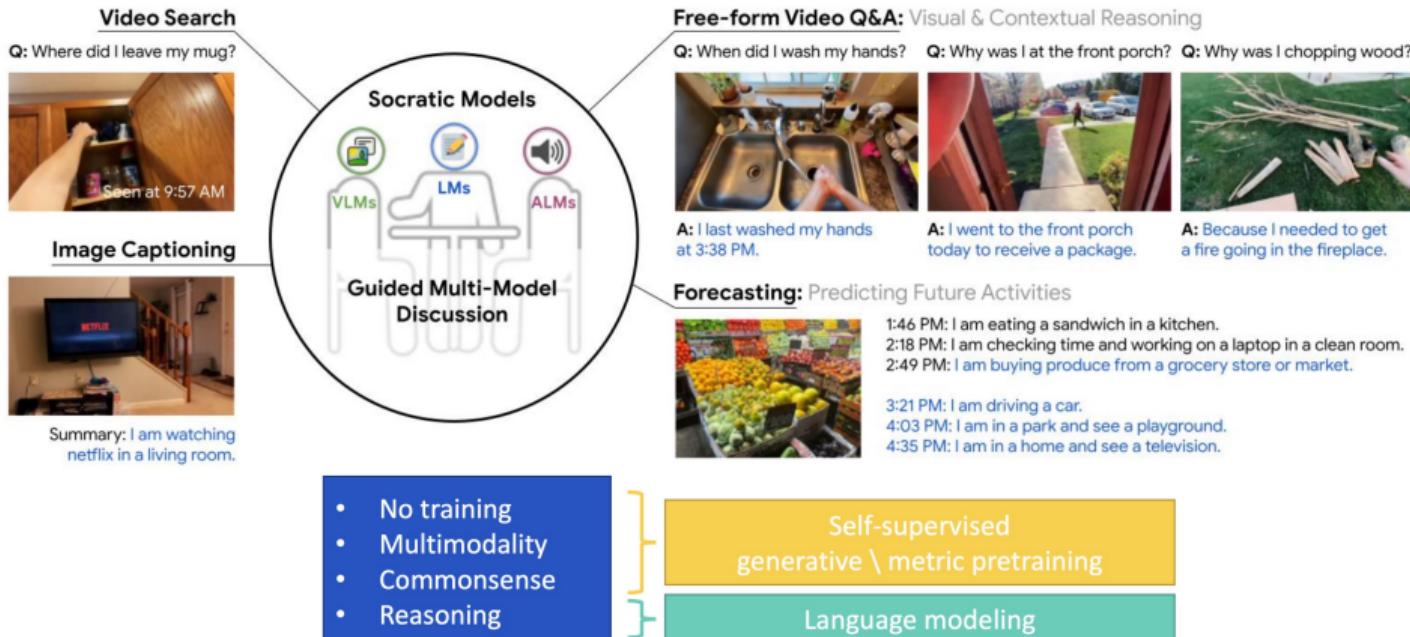
From Wikipedia, the free encyclopedia

The **Socratic method** (also known as **method of Elenchus** or **Socratic debate**) is a form of [argumentative dialogue](#) between individuals, based on asking and answering questions.

In [Plato's](#) dialogue "[Theaetetus](#)", [Socrates](#) describes his method as a form of "midwifery" because it is employed to help his interlocutors develop their understanding in a way analogous to a child developing in the womb. The Socratic method begins with commonly held beliefs and scrutinizes them by way of questioning to determine their internal consistency and their coherence with other beliefs and so to bring everyone closer to the truth.

In modified forms, it is employed today in a variety of pedagogical contexts.

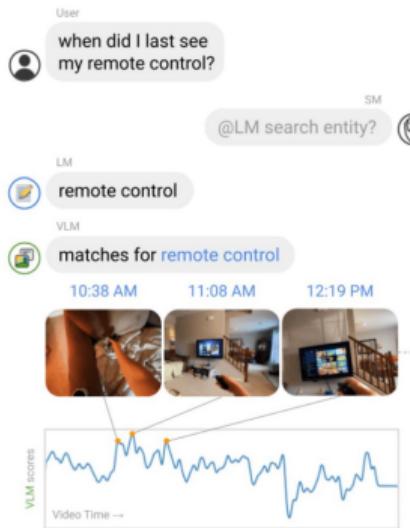
# Summary —○ ML today



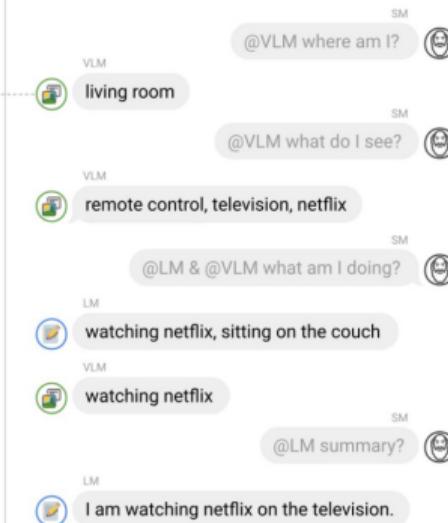
Zeng, Andy, et al. "Socratic models: Composing zero-shot multimodal reasoning with language." *arXiv preprint arXiv:2204.00598* (2022).

# Summary —○ ML today

## Socratic Question to Video Search



## Socratic Egocentric Image Summaries



## Socratic Video Q&A with Linguistic World-State History

### Generated Linguistic World-State History from Video

10:38 AM: Places: living room. Objects: sofa bed, loveseat, coffee table.  
Activities: watching TV. I was watching TV in a living room.  
11:08 AM: Places: living room. Objects: remote control, television, netflix.  
Activities: watching netflix. I was watching netflix on the television.  
12:19 PM: Places: television room. Objects: television, remote control, netflix. Activities: watching netflix. I was watching netflix on a television.

\*\*\*

User  
where did I leave my remote control?

LM  
I left my remote control in the living room.

SM  
@LM explain? (User)

LM  
I was watching TV in the living room and I needed it to change the channel.

Zeng, Andy, et al. "Socratic models: Composing zero-shot multimodal reasoning with language." *arXiv preprint arXiv:2204.00598* (2022).

# Summary —○ ChatGPT3 problems

## ChatGPT3 on fractions :

Me: what is larger, two fourths or one half?

ChatGPT :

One half is larger than two fourths.

Two fourths can be written as  $2/4$ , which is equal to  $1/2$ .  
Therefore, one half is equal to two fourths.

Me: didn't you just contradict yourself?

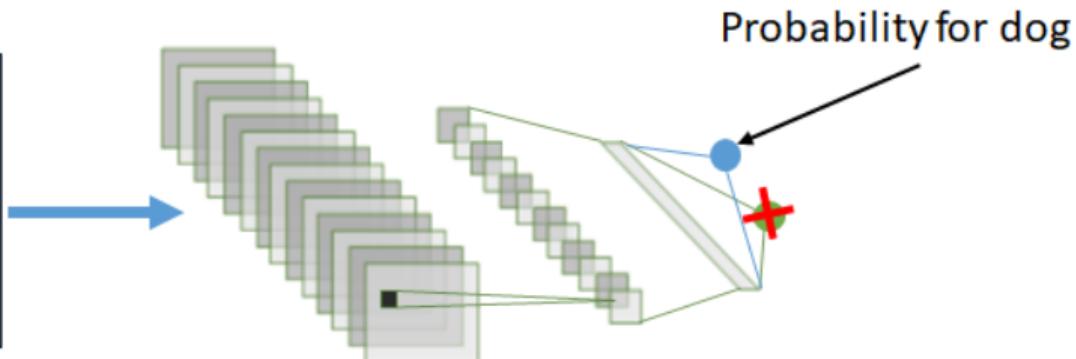
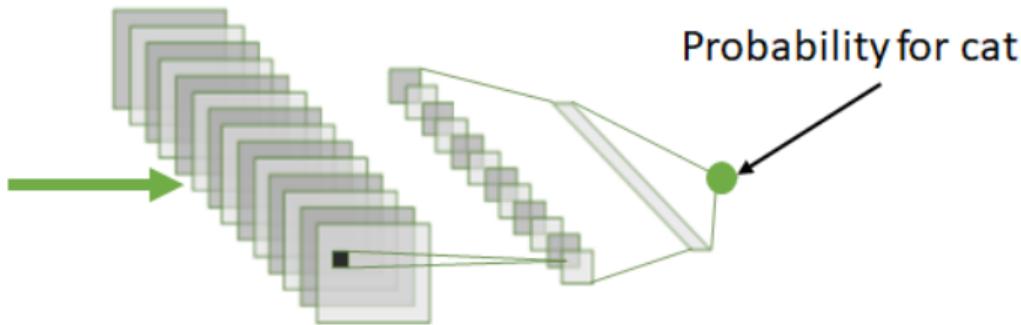
ChatGPT: I apologize, I made a mistake in my previous answer.  
Two fourths ( $2/4$ ) is smaller than one half ( $1/2$ ).

Me : Are you sure?

ChatGPT: Yes, I am sure. Two fourths ( $2/4$ ) is equal to  $1/2$ . Since  $1/2$  is larger than  $2/4$ , it follows that one half is larger than two fourths.



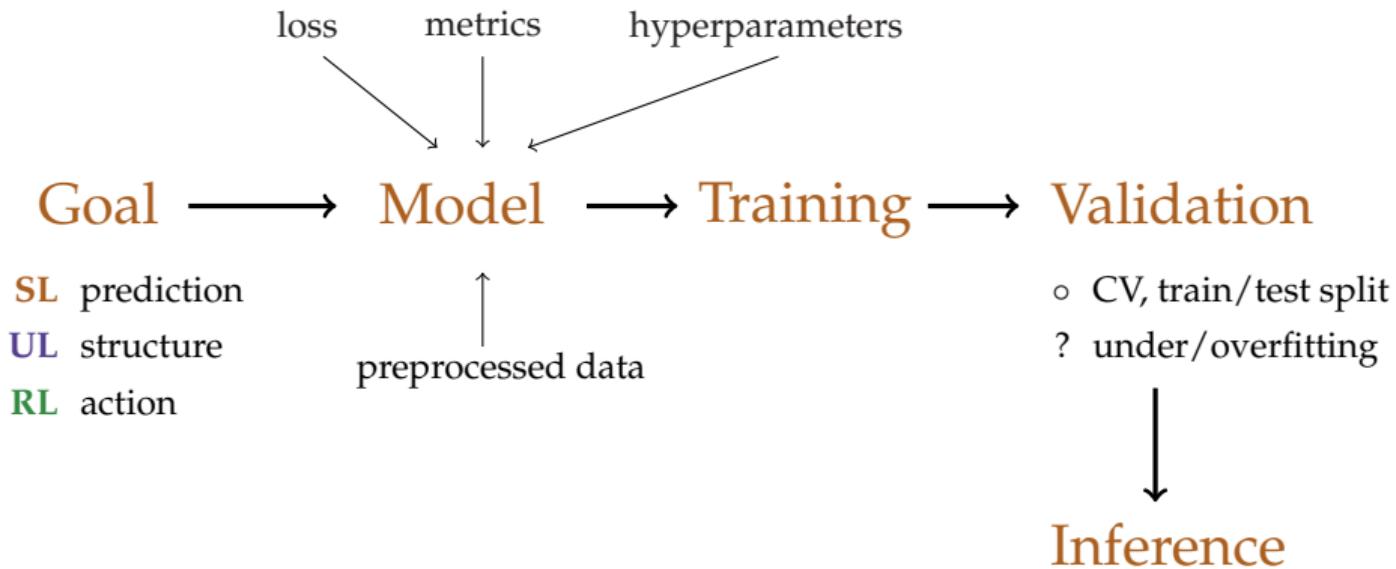
## Summary —○ Transfer learning



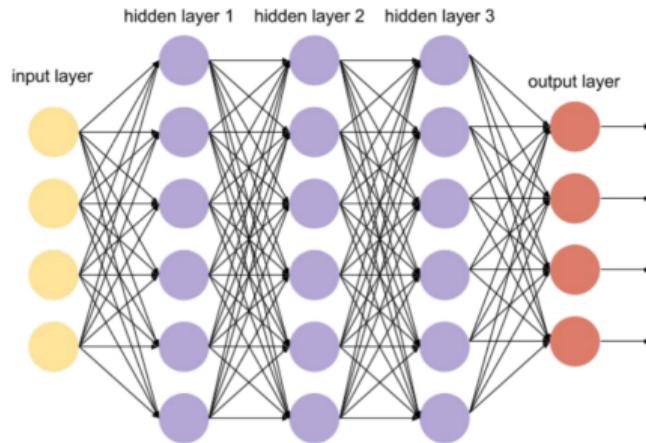
# Backup slides

# Backup slides

—○ what is ML pipeline?



Backup slides —○ NN formula



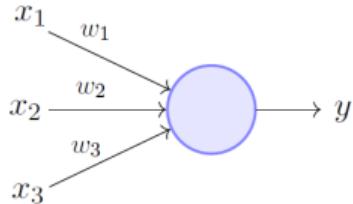
$$h_t = f(W h_{t-1} + b)$$

$$a(x) = f_3(W_3 f_2(W_2 f_1(W_1 f_0(W_0 x + b_0) + b_1) + b_2) + b_3)$$



# Backup slides

—○ calculating weights: slide 1



Perceptron Model (Minsky-Papert in 1969)

## Source

$$Q(y_{model}, y_{target}) = Q(y(x, w, b), y_{target})$$

$$out = y = \sigma(sum) = \sigma\left(\sum_{i=1}^3 (x_i \cdot w_i) + b\right)$$

$$\frac{\partial Q}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial sum} \cdot \frac{\partial sum}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot \mathbf{1} \cdot \frac{\partial \sigma}{\partial sum} \cdot \frac{\partial sum}{\partial w_i}$$

$$\frac{\partial \sigma}{\partial sum} = \sigma(sum) \cdot (1 - \sigma(sum))$$

# Backup slides —○ calculating weights: slide 2

$$Q(y(x, w, b), y_{target})$$

$$Q = (y_{target} - y(x_1, w_1, b))^2$$

$$out = y = (x \cdot w) + b$$

$$\frac{\partial Q}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w_i}$$

Input:  $x = 1$ ,  $w = 0.1$  and  $b = 1$

$y_{target} = 2$  and learning rate:  $\eta = 0.1$

# Backup slides —○ calculating weights: slide 2

$$\begin{aligned} Q(y(x, w, b), y_{target}) \\ Q = (y_{target} - y(x_1, w_1, b))^2 \\ out = y = (x \cdot w) + b \\ \frac{\partial Q}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w_i} \end{aligned}$$

Input:  $x = 1$ ,  $w = 0.1$  and  $b = 1$

$y_{target} = 2$  and learning rate:  $\eta = 0.1$

How to find  $w_{new}$  and  $b_{new}$ ?

$$\begin{aligned} \frac{\partial Q}{\partial y} &= -2 \cdot (y_{target} - y) \\ \frac{\partial Q}{\partial w} &= \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w} = -2 \cdot x \cdot (y_{target} - y) \\ \frac{\partial Q}{\partial b} &= \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial b} = -2 \cdot (y_{target} - y) \\ [w_{new}, b_{new}] \Rightarrow \theta_{i+1} &= \theta_i - \eta \cdot \frac{\partial Q(\theta_i)}{\partial \theta_i} \end{aligned}$$

# Backup slides

—○ calculating weights: slide 2

$$Q(y(x, w, b), y_{target})$$

$$Q = (y_{target} - y(x_1, w_1, b))^2$$

$$out = y = (x \cdot w) + b$$

$$\frac{\partial Q}{\partial w_i} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w_i}$$

Input:  $x = 1$ ,  $w = 0.1$  and  $b = 1$

$y_{target} = 2$  and learning rate:  $\eta = 0.1$

How to find  $w_{new}$  and  $b_{new}$ ?

$$\frac{\partial Q}{\partial y} = -2 \cdot (y_{target} - y)$$

$$\frac{\partial Q}{\partial w} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial w} = -2 \cdot x \cdot (y_{target} - y)$$

$$\frac{\partial Q}{\partial b} = \frac{\partial Q}{\partial y} \cdot \frac{\partial y}{\partial b} = -2 \cdot (y_{target} - y)$$

$$[w_{new}, b_{new}] \Rightarrow \theta_{i+1} = \theta_i - \eta \cdot \frac{\partial Q(\theta_i)}{\partial \theta_i}$$

y	Q	$\frac{\partial Q}{\partial y}$	$\frac{\partial Q}{\partial w}$	$\frac{\partial Q}{\partial b}$	$w_{new}$	$b_{new}$
1.1	0.81	-1.8	-1.8	-1.8	0.28	1.18

# Backup slides

—○ human brain

