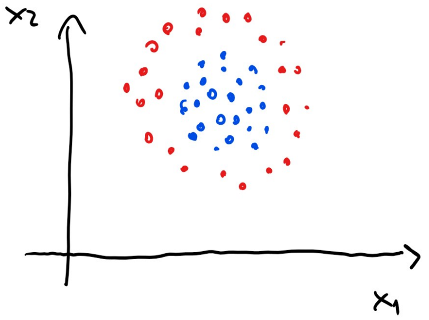
NEURAL NETWORKS

ASSUME I'VE THE FOLLOWING CLASSIFICATION PROBLEM



IT IS AN EASY PROBLEM, THE CLASS DEPENDS ON THE DISTANCE FROM THE MEAN OF THE DATASET, HOWEVER THE DATA IS NOT LINEARLY SEPARABLE (AT LEAST NOT IN THIS INPUT SPACE). SO I'VE NO HOPE TO SOLVE THE PROBLEM WITH A LINEAR MODEL (WITHOUT USING THE KERNEL TRICK)

WE'VE SEEN THAT KERNEL TRICK ALLOWS TO TRANSFOR THE INPUT SPACE INTO A FEATURE SPACE IN WHICH THE DATA MAY BE LINEARLY SEPARABLE, HOWEVER THIS TRANSFORMATION IS HAND CRAFTED, SO I MUST CHOOSE THE KERNEL THAT TRANSFORMS THE DATA INTO THE RIGHT FEATURE SPACE.

NEURAL NETWORKS ALLOW TO DO SOMETHING AMAZING: THEY LEARN AN INTERMEDIATE REPRESENTATION OF DATA (CALLED LATENT REPRESENTATION) THAT ALLOWS TO SOLVE THE SPECIFIC TASK.

THIS LATENT REPRESANTATION CAN BE VERY COMPLEX (I.E. A VERY NONLINEAR TRANSFORMATION) BUT IT IS OBTAINED STACKING MULTIPLE SIMPLE TRANSFORMATIONS THAT CONSIST IN A LINEAR MAP AND A NONLINEAR FUNCTION APPLIED ELEMENTWISE.

LETS DESIGN A NEURAL NETWORK WITH 3 LAYERS THAT SOLVES THE CLASSIFICATION PROBLEM DESCRIBED ABOVE.

WE HAVE OUR DATA MATRIX

$\bar{X} \in \mathbb{R}^{N \times 2}$

WE FIRST APPLY A LINEAR MAP THAT MAPS THE DATA INTO A HIGHER DIMENSIONAL SPACE, FOR EXAMPLE $\mathbb{R}^5$, WE ALSO APPLY A NONLINEARITY ELEMENTWISE, SUCH THAT THE MAPPING IS NONLINEAR, THEN WE APPLY A LINEAR MODEL (LOGISTIC REGRESSION) TO THE POINTS IN THE HIDDEN SPACE

$$\underbrace{X \in \mathbb{R}^{N\times 2}}_{\text{DATA MATRIX}} \quad \underbrace{W^1 \in \mathbb{R}^{2\times S}}_{\text{LINEAR MAP}} \quad \underbrace{Q: \mathbb{R} \to \mathbb{R}}_{\text{NONLINEAR FUNCTION}} \quad \underbrace{W^2 \in \mathbb{R}^{S\times 1}}_{\text{LINEAR MAP}} \quad \underbrace{\sigma: \mathbb{R} - \mathbb{R}}_{\text{SIGMOID FUNCTION}}$$

- FIRST I COMPUTE THE HIDDEN REPRESENTATION

$$H = Q\left(X W^1\right) \qquad H \in \mathbb{R}^{N\times S}$$

- THEN I APPLY LOGISTIC REGRESSION TO THE DATA IN THE HIDDEN SPACE

$$O = \sigma\left(H W^2\right) \qquad O \in \mathbb{R}^{N\times 1}$$

- THEN I CAN COMPUTE THE LOSS

$$L = -\sum_{i=1}^{N} Y[i] \log\left(O[i]\right) + \left(1 - Y[i]\right) \log\left(1 - O[i]\right)$$

- I CAN ALSO WRITE THE OUTPUT AS A FUNCTION OF THE INPUT AS

$$O = \sigma\left(Q\left(X W^1\right) W^2\right)$$

$\boxed{\text{BACKPROPAGATION}}$

NOW THE TRICKY PART, I'VE TO COMPUTE THE GRADIENT OF THE LOSS WITH RESPECT TO <u>ALL</u> THE MODELS PARAMETERS

$$\nabla_\theta L = ?$$

I CAN DO IT APPLYING THE <u>CHAIN RULE</u>

ASSUME I'VE $f: \mathbb{R}^2 \to \mathbb{R}$, $(x,y) \to z = f(x,y)$ AND $g: \mathbb{R}^2 \to \mathbb{R}^2$, $(s,t) \to (x = g_1(s,t), y = g_2(s,t))$

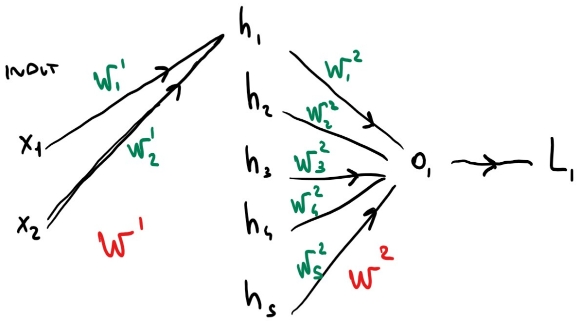
AND LET

$$F(s,t) = (f \circ g)(s,t) = f\left(g_1(s,t), g_2(s,t)\right)$$

THEN

$$\frac{\partial F}{\partial s} = \frac{\partial f}{\partial x} \cdot \frac{\partial g_1(s,t)}{\partial s} + \frac{\partial f}{\partial y} \cdot \frac{\partial g_2(s,t)}{\partial s}$$

$$\frac{\partial F}{\partial t} = \frac{\partial f}{\partial x} \cdot \frac{\partial g_1(s,t)}{\partial t} + \frac{\partial f}{\partial y} \cdot \frac{\partial g_2(s,t)}{\partial t}$$



FOR EXAMPLE LET US COMPUTE $\frac{\partial L}{\partial w_1^2}$ APPLYING THE CHAIN RULE

$$\frac{\partial L}{\partial w_1^2} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial w_1^2}$$

NOW I CAN COMPUTE $\frac{\partial L}{\partial w_1^1}$

$$\frac{\partial L}{\partial w_1^1} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1^1} + \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_1^1} + \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_3} \cdot \frac{\partial h_3}{\partial w_1^1} + \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_4} \cdot \frac{\partial h_4}{\partial w_1^1} + \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_5} \cdot \frac{\partial h_5}{\partial w_1^1}$$

NOTE THAT THIS TERM IS REPEATED OVER AND OVER, I CAN COMPUTE IT JUST ONCE !!

IN THE SAME WAY I CAN COMPUTE $\frac{\partial L}{\partial w_2^1}$

$$\frac{\partial L}{\partial w_2^1} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_2^1} + \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2^1} + \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_3} \cdot \frac{\partial h_3}{\partial w_2^1} + \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_4} \cdot \frac{\partial h_4}{\partial w_2^1} + \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h_5} \cdot \frac{\partial h_5}{\partial w_2^1}$$

I'VE A LOT OF REPEATED TERMS !!

IDEA

- COMPUTE THE OUTPUT FROM THE INPUT GOING FORWARD (FOLLOWING THE ARROWS OF THE COMPUTATIONAL GRAPH)
- COMPUTE THE GRADIENTS GOING BACKWARD

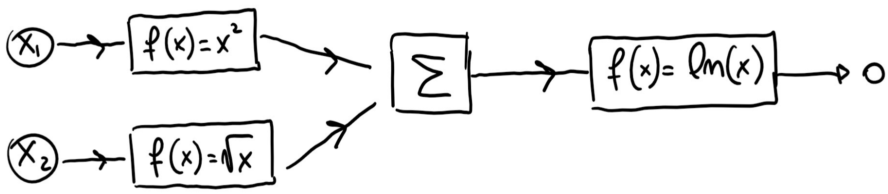
→ IN PROGRAMMING ENVIRONMENTS THAT SUPPORT AUTOMATIC DIFFERENTIATION (AUTOGRAD) EACH OPERATION HAS A FORWARD AND BACKWARD FUNCTIONS, THE FORWARD OPERATION COMPUTES THE OUTPUT W.R.T. THE INPUT, WHILE THE BACKWARD FUNCTION COMPUTES THE GRADIENT OF THE OUTPUT W.R.T. THE INPUT

EXAMPLE

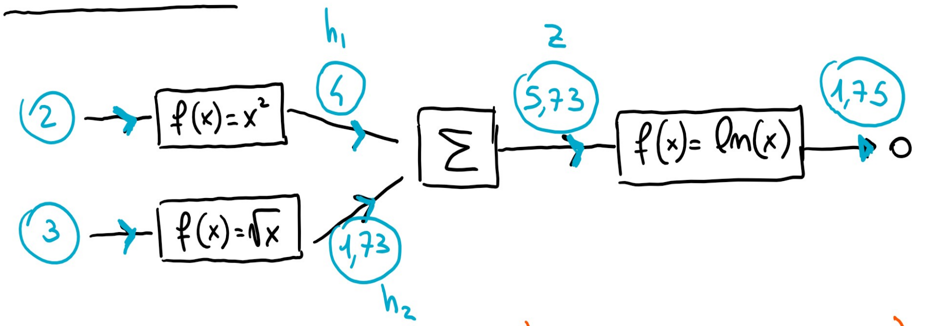
COMPUTATIONAL GRAPH

○ DATA

□ OPERATION



FORWARD PASS



BACKWARD PASS