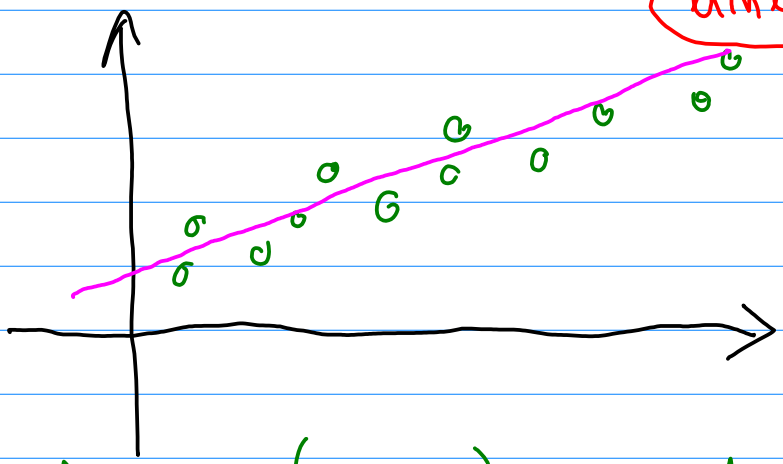
Course 2

Linear regression: fitting data with a linear (affine) function



Data: (x_i, y_i) , $i = 1 \dots N$

Find the line $y = mx + n$ such that

$$mx_i + n \approx y_i$$

DATA: For parameters

x_1 → we get y_1

x_2 → we get y_2

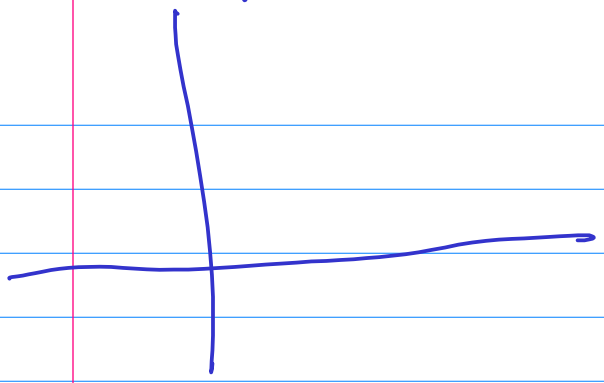
$\vdots$

x_M → we get y_M

} verified output encoded in tensors

Objects → convert to real values/vectors/matrices/tensors

Data as a straight line:



weight → bias

$y = mx + n$

parameters

Samples x_i

Compute the values $y_i = mx_i + n$

$\Rightarrow (x_i, y_i)$

DATA

input	output	
x_1	y_1	training set
x_2	y_2	Train the model
⋮	⋮	using THIS DATA
x_{80}	y_{80}	test set
⋮		NOT USED FOR
x_{100}	y_{100}	TRAINING

80%

20%

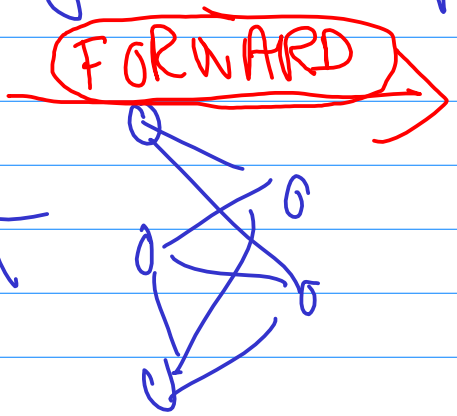
WHY? We need to test the model on data it "hasn't seen before".

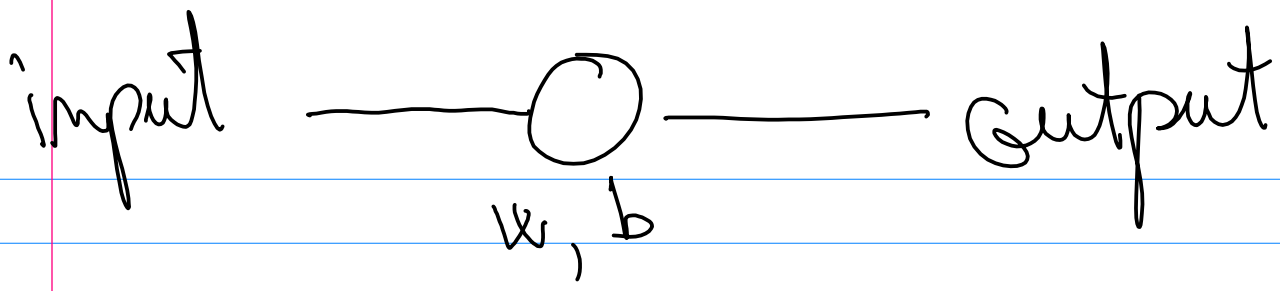
Plotting the data helps us understand better what's happening.

MODEL FOR LINEAR REG:

→ Parameters: weight (slope)
bias (translation)

→ "Compute" the model
"forward" pass through the model



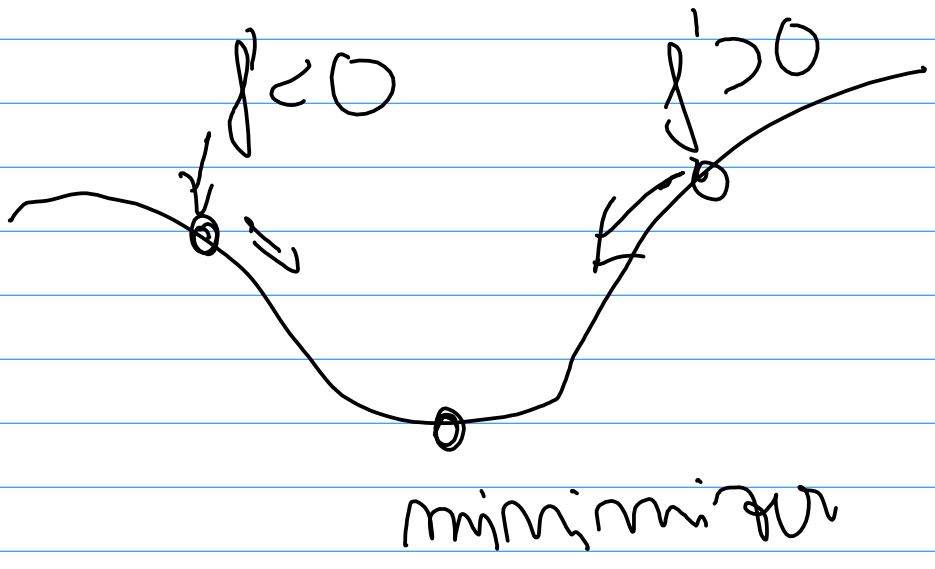


$x \longrightarrow x \cdot w + b \longrightarrow$ output

→ For training we need to differentiate the "formula" that is used in the forward function.

→ automated differentiation

→ requires_grad = True keeps track of all computations such that the derivative can be computed automatically



Loss function: measure the "difference" between model results and expected results.

$$L1\ loss = \sum_{i=1}^{N} |Model(x_i) - y_i|$$

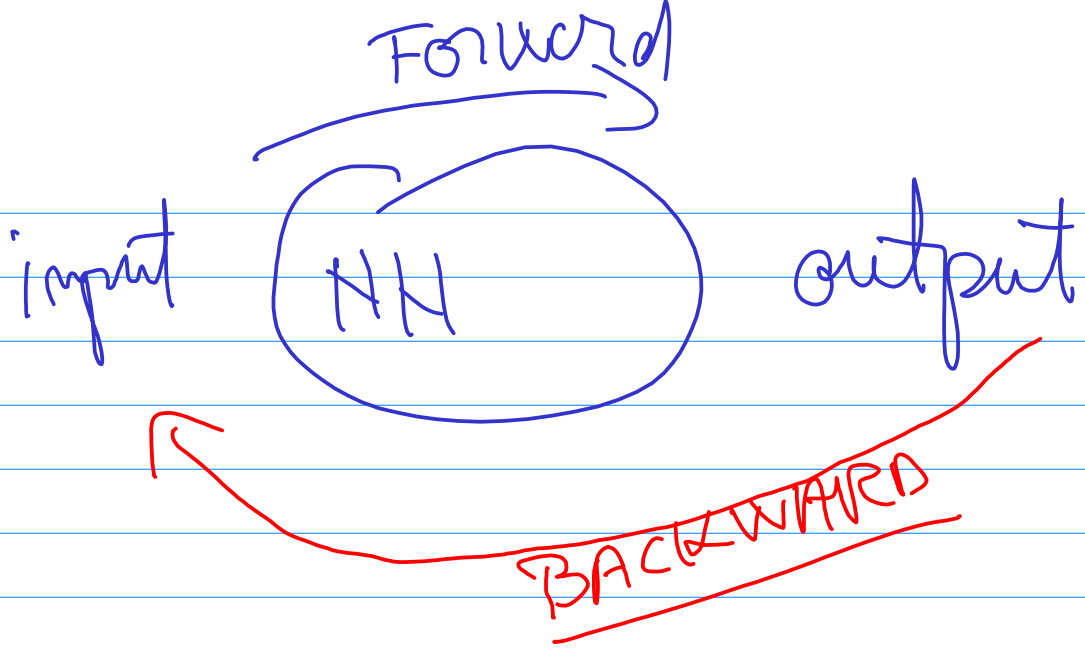
$Model(x_i)$: result of model for x_i

y_i: Data for x_i

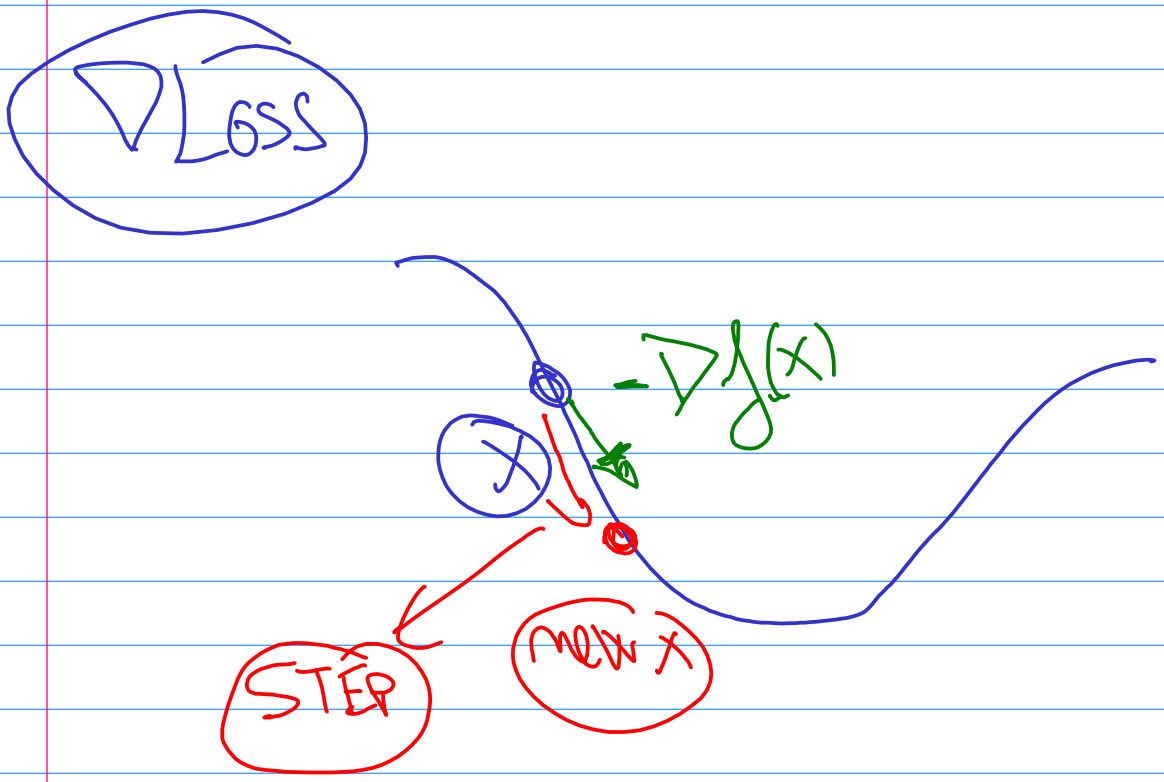
Loss "small" $\Rightarrow$ model predicts "well"

$$MSE\ Loss = \sum_{i=1}^{N} (Model(x_i) - y_i)^2$$

OBJECTIVE: make Loss function small!! Use on optimization alg.



$$\text{Loss} := \sum (\text{output}_i - y_i)^2$$



Observations: -> Do as many epochs as you can (reasonable in time)

-> Decreasing learning rate can

improve results after an initial run.