

NEURAL NETWORKS - REGRESSION

let $\{x_i, y_i\}_{i=1,\dots,M}$ a set of input-output pairs

we want to find a model \hat{y} that predicts new outputs given new inputs

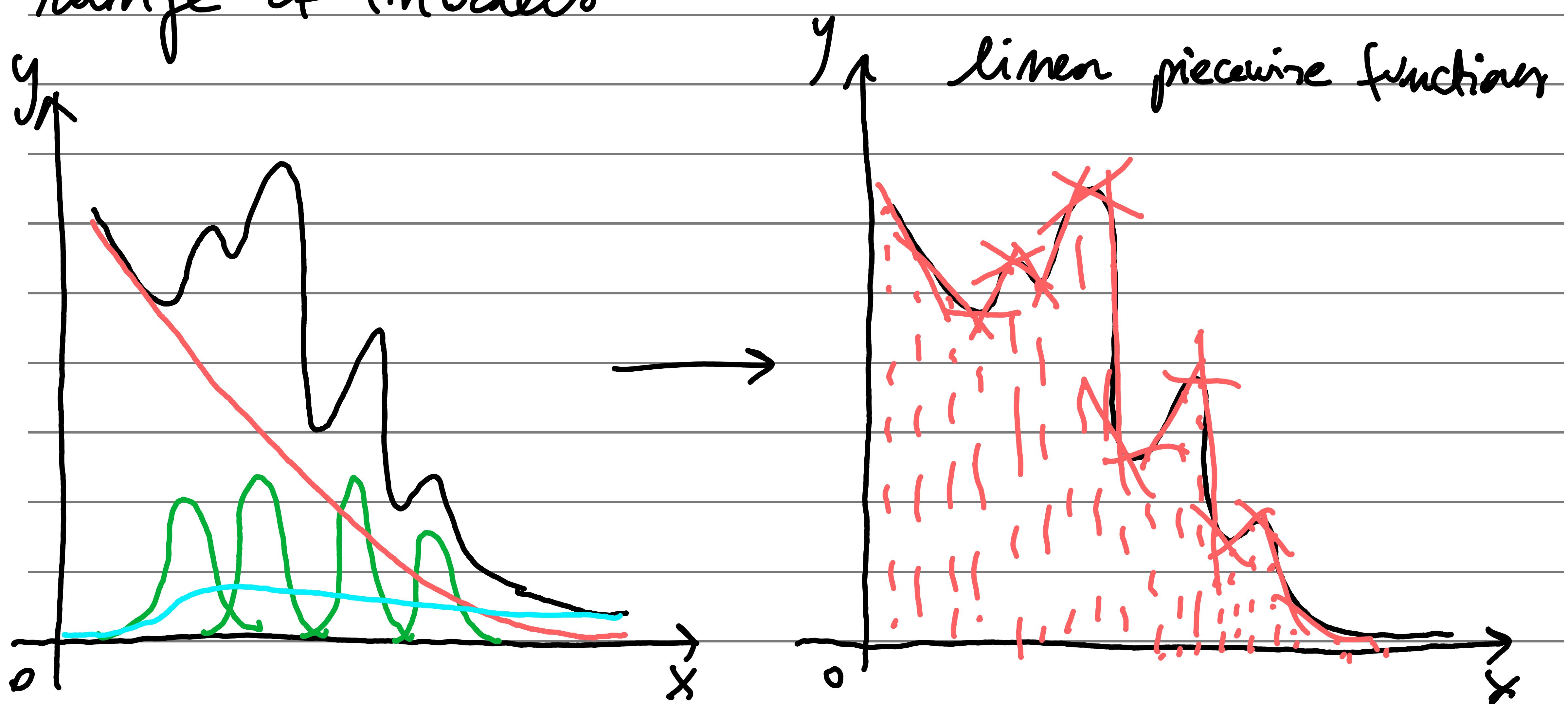
We have seen linear regression

$$f(x, w) = \sum_{j=1}^M w_j \phi_j(x)$$

i.e. models that are linear in the parameters

It might be necessary to consider a wider

range of models



sacrifice interpretability to performance

$$f(x, \underline{w}, \underline{\theta}) = \sum_{i=0}^n w_i h_i(x, \underline{\theta})$$

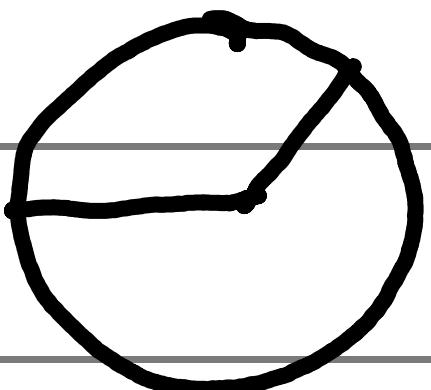
where $h_i = a[\theta_{0i} + \theta_{1i} x]$

↓
linear
function

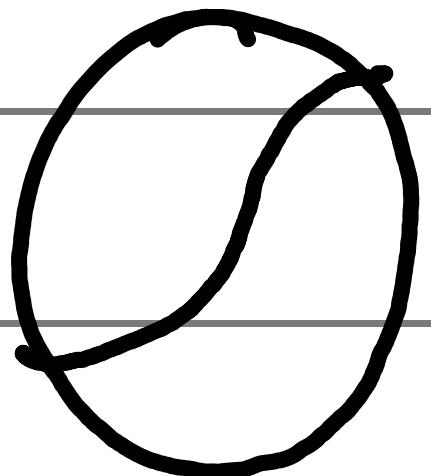
non linear
function (activation)

activation:

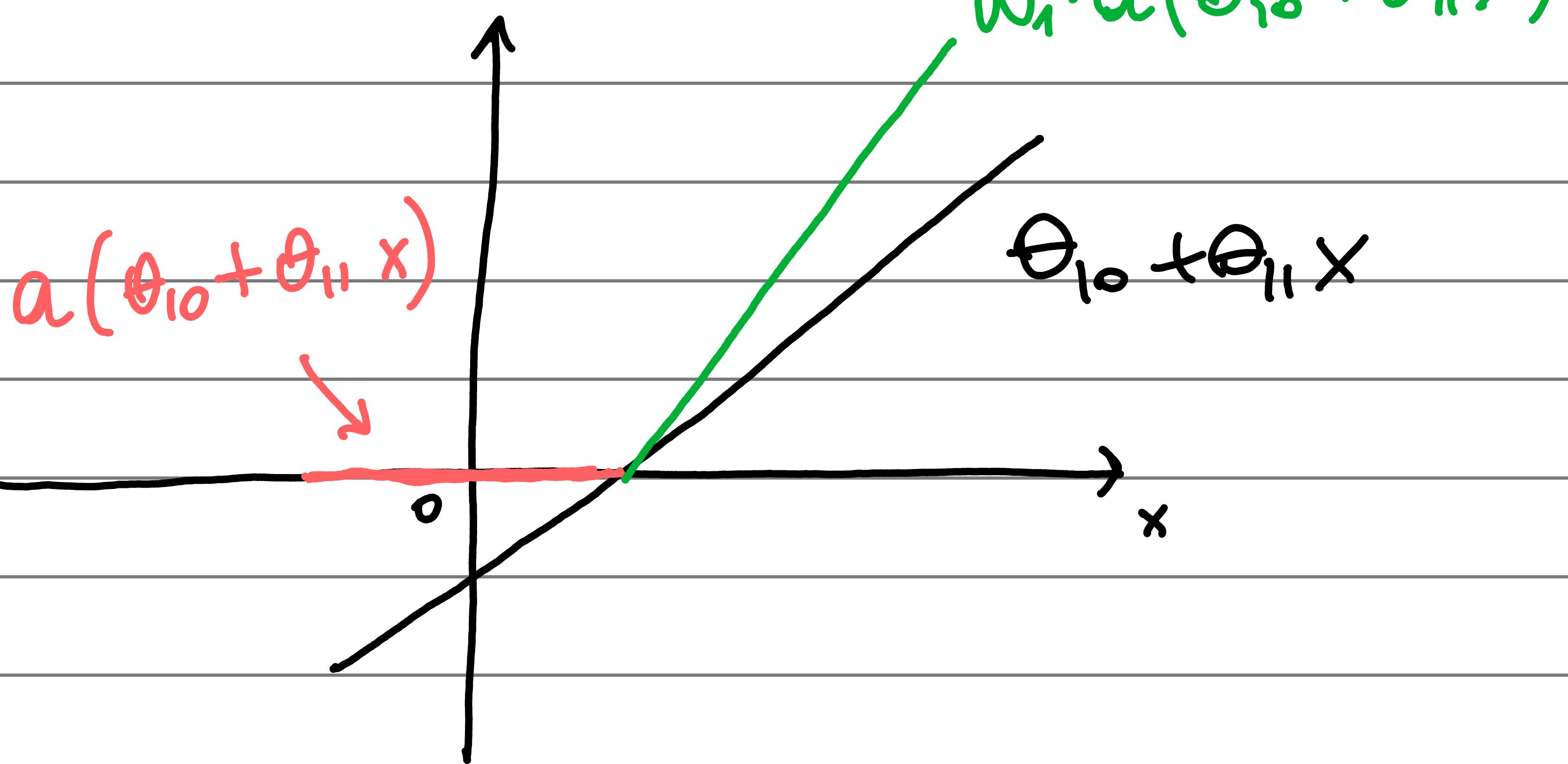
$$\text{ReLU} \rightarrow a(z) = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$$



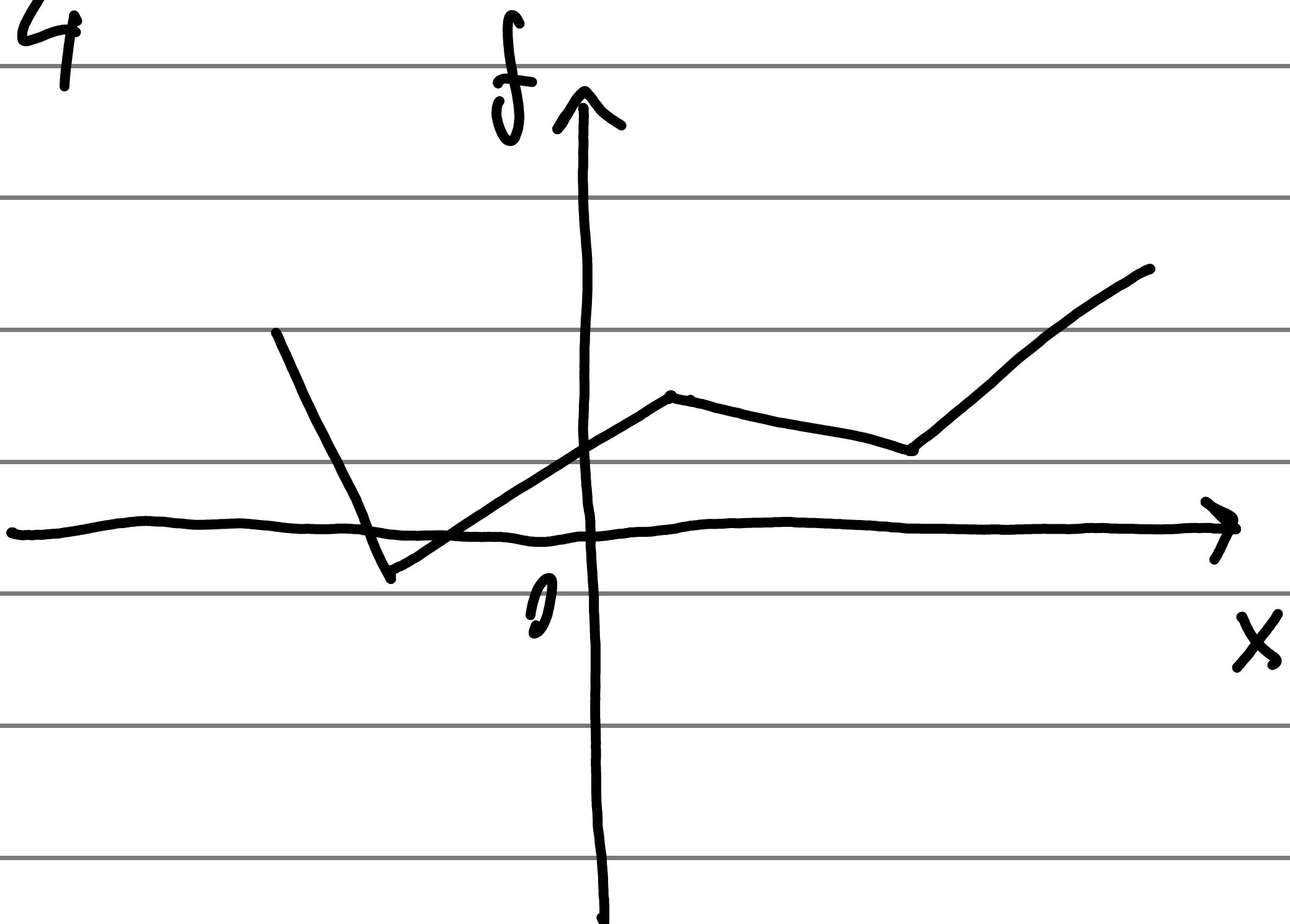
$$\text{SIGMOID} \rightarrow a(z) = \frac{1}{1 + \exp(-z)}$$



e.g.

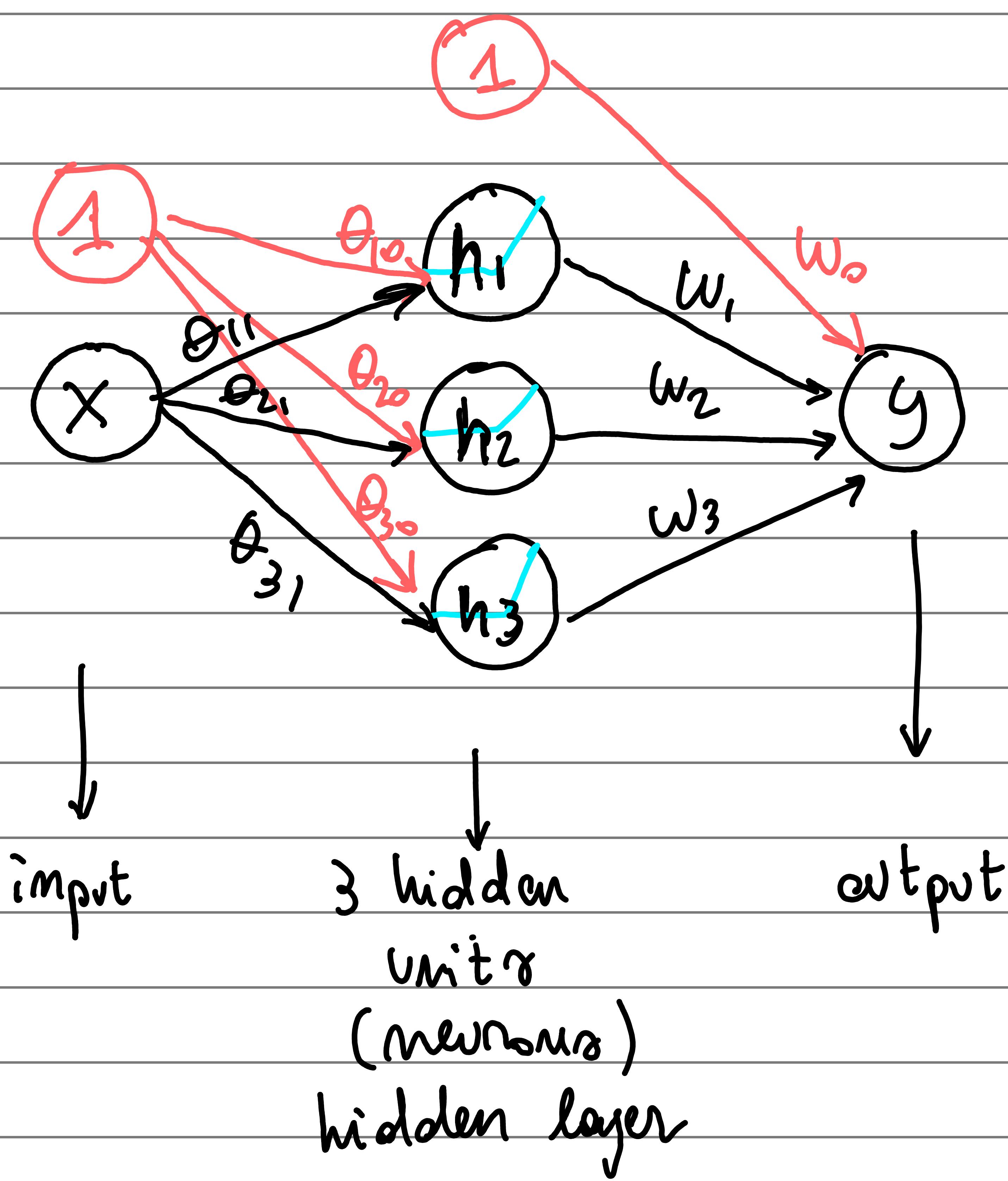


for $M = 4$



$$w_0 + w_1 h_1 + w_2 h_2 + w_3 h_3$$

the corresponding neural network can be represented as



UNIVERSAL APPROXIMATION THEOREM

Suppose we have D hidden units

$$\hat{y} = \sum_{i=0}^D w_i h_i \quad (h_0 = 1)$$

D is the network capacity

TH: with enough capacity, a "shallow" network

can describe any continuous 1D function

defined on a compact subset of the real line

to arbitrary precision

Note: I can have $x \in \mathbb{R}^d$, so each d

has now D parameters and the

space is divided into 2^d regions

→ dimension of the network grows very

quickly with complexity of input

HOW TO BUILD A NEURAL NETWORK (FOR REGRESSION)

The setup of a regression analysis using a

neural network follows the same basic steps

of the linear regression approach (with a few
key differences)

- TRAINING AND TEST DATASETS (AND VALIDATION)
- DEFINE MODEL (CAPACITY, DEPTH, ...)
- LOSS FUNCTION (e.g. $\epsilon(w)$ error function in LR)
- OPTIMIZATION METHOD AND INITIALIZATION
- PERFORMANCE ESTIMATION
- REGULARIZATION

OPTIMIZATION METHOD AND INITIALIZATION

Given a training set $\{x_i, y_i\}_{i=1, \dots, n}$

and a model $\hat{y}(\underline{x}, \underline{w})$ I want to find

the best set of parameters \underline{w} so that \hat{y} is a good model. The loss function $L(\underline{w})$

gives a measure of how good is a given set of parameters.

In linear regression, we could analytically minimize with respect to \underline{w} and with MCMC we just explored parameter space.

Both are not feasible in neural network since

- 1) They are non-linear models \rightarrow analytical solution doesn't exist
- 2) they have a large parameter space \rightarrow impossible to explore

The process of minimization of $L(\underline{w})$ is ITERATIVE

let's consider $\{w_i\}_{i=1,\dots,M}$ initial parameters

STEP 1: Compute derivative of the loss

$$\frac{\partial L}{\partial w^0} = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \vdots \\ \frac{\partial L}{\partial w_M} \end{bmatrix}$$

STEP 2: Update parameters

$$w^1 = w^0 - \alpha \frac{\partial L}{\partial w^0}$$

$\downarrow \alpha > 0$ magnitude of change
(learning rate)

$$w^2 = w^1 - \alpha \frac{\partial L}{\partial w^1}$$

.

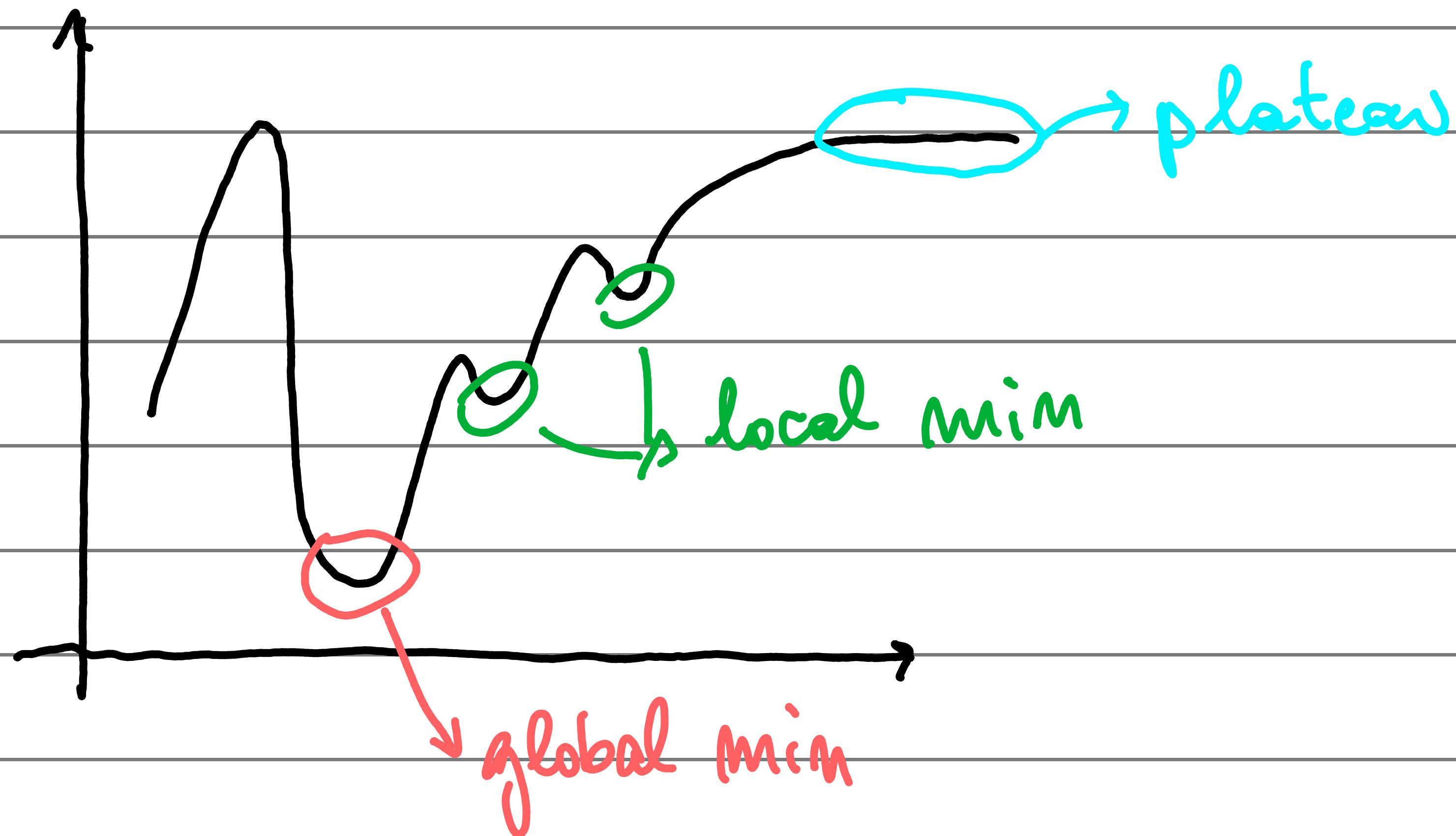
$$w^{M+1} = w^M - \alpha \frac{\partial L}{\partial w^M}$$

Theoretically we should find a $\{w_i^m\}_{i=1,\dots,M}$ such

that $\frac{\partial L}{\partial w^m} = 0$ but in practice this is numerically

unfeasible \rightarrow we monitor and terminate when $\frac{\partial L}{\partial w} \ll 0$

PROBLEM: In complex (and higher-dimensional) cases $L(w)$ has local minima and saddle points



→ hard to converge to global minimum

- searching entire parameter space is unfeasible
- trying different starting points also

~~STOCHASTIC GRADIENT DESCENT~~

IDEA: compute iterations for a subset of the

training set (BATCH)

$$w^{t+1} = w^t - \alpha \sum_{i \in B_t} \frac{\partial L}{\partial w^t}$$

$B_t \subset \text{TRAINING SET}$

once all training data has been seen → EPOCH