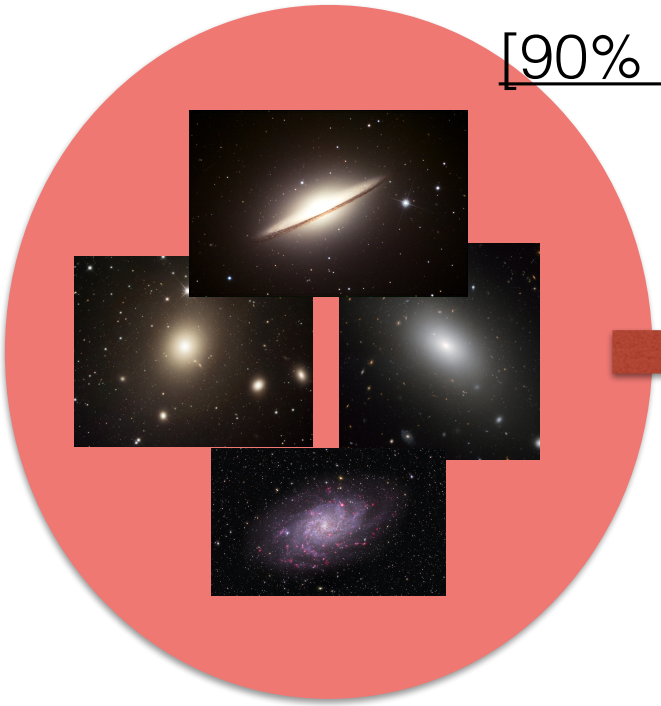
IMAGINE AN EXTREME CASE WITH VERY UNBALANCED DATA…

[90% E, 10% S]



IF I SAY THAT ALL GALAXIES IN THE UNIVERSE ARE ELLIPTICALS I WILL BE RIGHT 90% OF THE TIMES

Evaluation of results [binary class.]

THE ROC CURVE (Receiver Operating Characteristic)

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1. $$TPR = \frac{TP}{TP + FN}$$ [Also called Sensitivity, Completeness]

"Fraction of positive examples classified correctly"

Evaluation of results [binary class.]

THE ROC CURVE (Receiver Operating Characteristic)

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1. $$TPR = \frac{TP}{TP + FN}$$ TRUE POSITIVE RATE [Also called Sensitivity, Completeness]

"Fraction of positive examples classified correctly"

2. $$FPR = \frac{FP}{FP + TN}$$ FALSE POSITIVE RATE [Also called Specificity, Contamination]

"Fraction of negative examples classified as positive"

Evaluation of results [binary class.]

ver Operating Characteristic)

YOU WANT THIS TO BE AS BIG AS POSSIBLE

IT IS BASED ON TWO VERY SIMPLE PARAMETERS

1. $TPR = \frac{TP}{TP +$ TRUE POSITIVE RATE Completeness]

YOU WANT THIS TO BE AS SMALL AS POSSIBLE

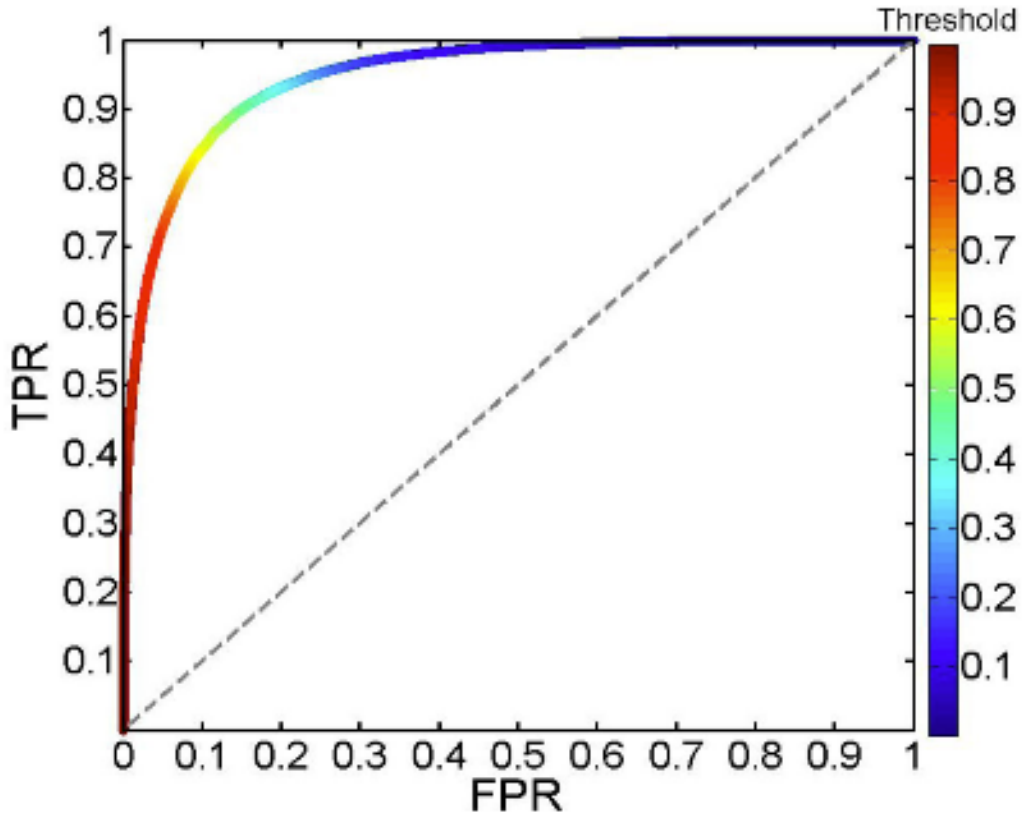
"Fraction of positive examples classified correctly"

2. $FPR = \frac{FP}{FP + TN}$ FALSE POSITIVE RATE [Also called Specificity, Contamination]

"Fraction of negative examples classified as positive"

IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY, TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

ROC CURVE

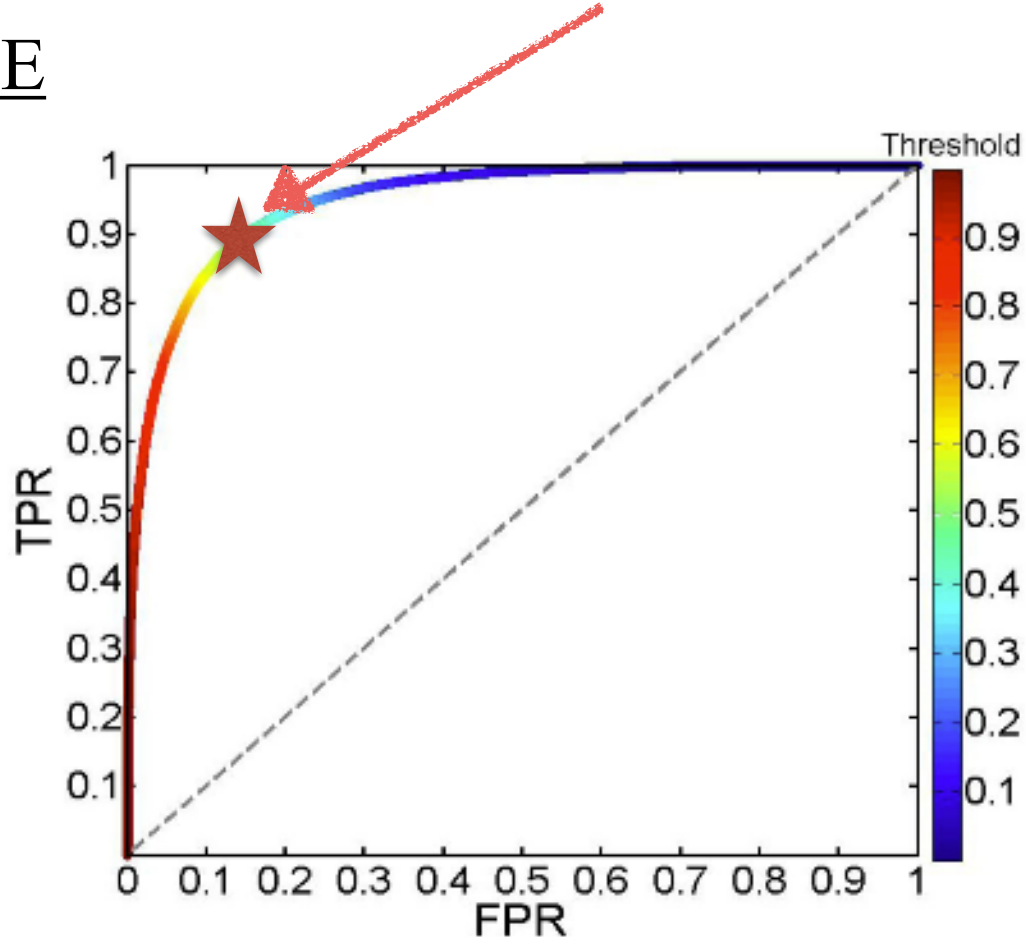


Teimiroonia+16

IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY, TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

EACH POINT HERE SHOWS THE VALUES OF TPR AND FOR FOR A GIVEN THRESHOLD

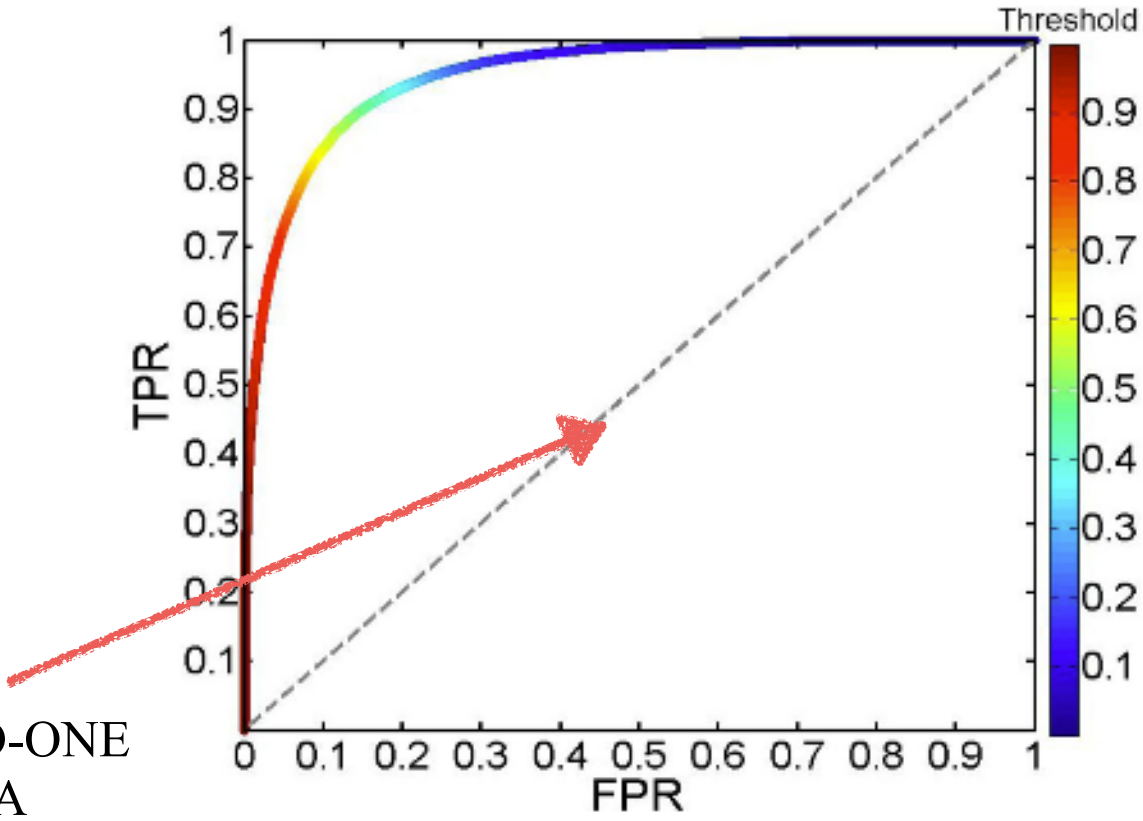
ROC CURVE



Teimiroonia+16

IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY, TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

ROC CURVE



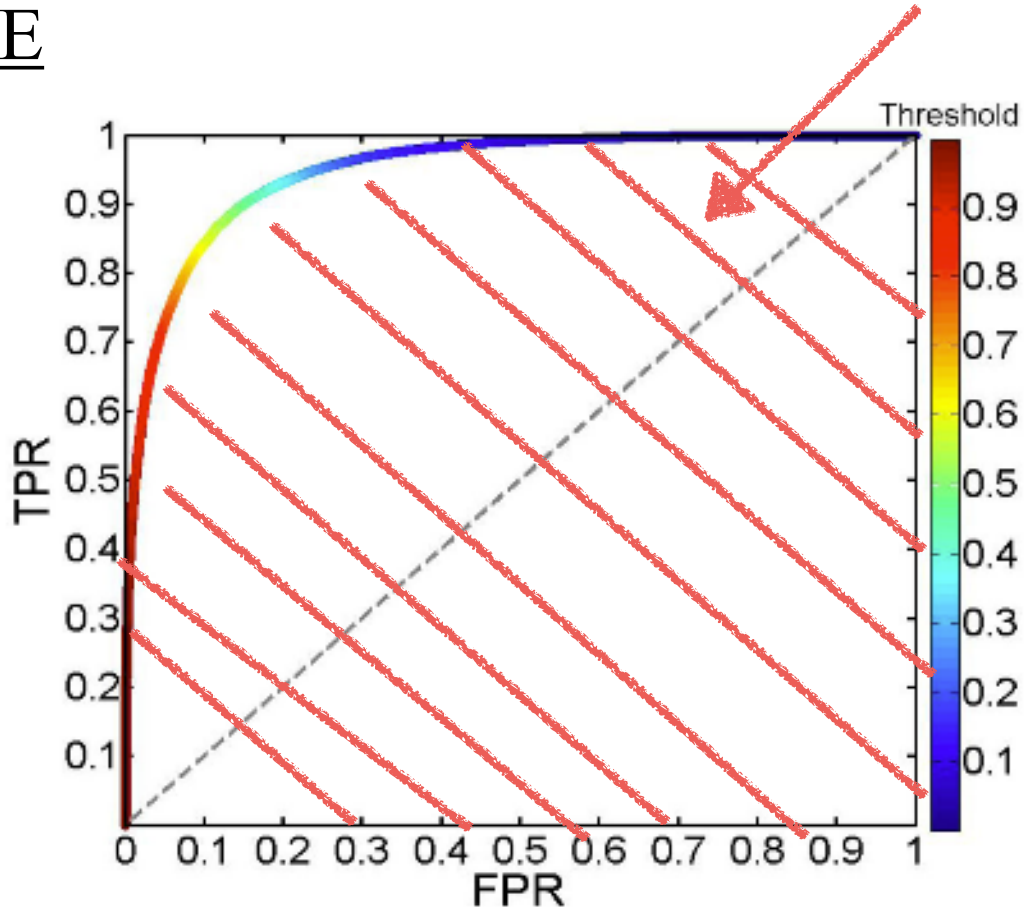
THE ONE-TO-ONE LINE IS A RANDOM CLASSIFICATION

Teimiroonia+16

IF YOUR CLASSIFIER OUTPUTS A SORT OF PROBABILITY, TPR AND FPR CAN BE PLOTTED ONE AGAINST THE OTHER

THE AREA UNDER THE CURVE **[AUC]** ALSO MEASURES THE GLOBAL ACCURACY

ROC CURVE



Teimiroonia+16

Evaluation of results

THE P-R CURVE (Precision - Recall)

$$Recall = \frac{TP}{TP + FN} = TPR \qquad \text{[completeness]}$$

$$Precision = \frac{TP}{TP + FP} \qquad \text{[purity]}$$

Evaluation of results

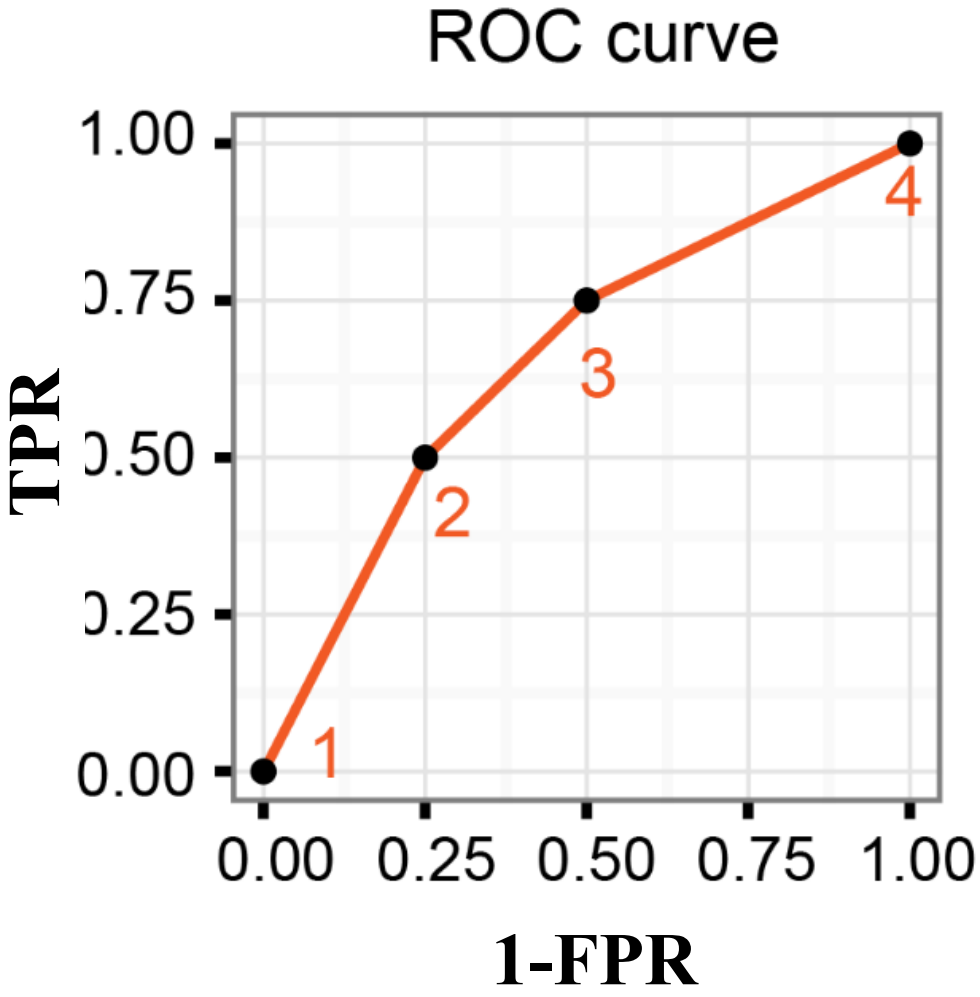
THE P-R CURVE (Precision - Recall)

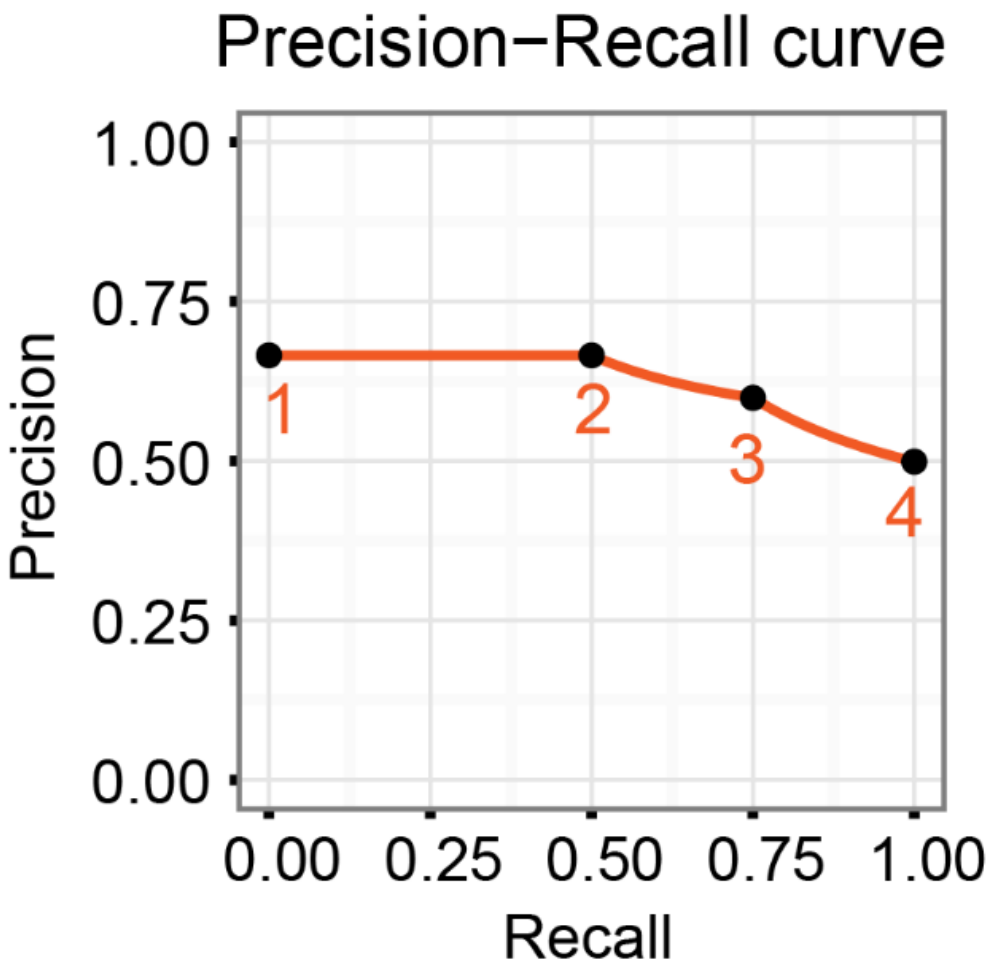
$$Recall = \frac{TP}{TP + FN} = TPR \qquad \text{[completeness]}$$

$$Precision = \frac{TP}{TP + FP} \qquad \text{[purity]}$$

FOR BALANCED DATA: $Precision \sim 1 - FPR$

Evaluation of results





SUMMARY OF DIFFERENT ACCURACY TRACERS

		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

SOURCE

SUMMARY OF DIFFERENT ACCURACY TRACERS

		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	

F_1 score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$

THE F1 SCORE
COMBINES BOTH INFORMATIONS IN ONE VALUE

SOURCE

ALL THESE ARE INCLUDED IN SKLEARN

AND ARE VERY EASY TO USE. NO NEED OF CODING THEM AGAIN!

`sklearn.metrics.` **`precision_recall_curve`** (*y_true*, *probas_pred*, *pos_label=None*, *sample_weight=None*) ¶ [source]

Compute precision-recall pairs for different probability thresholds

Note: this implementation is restricted to the binary classification task.

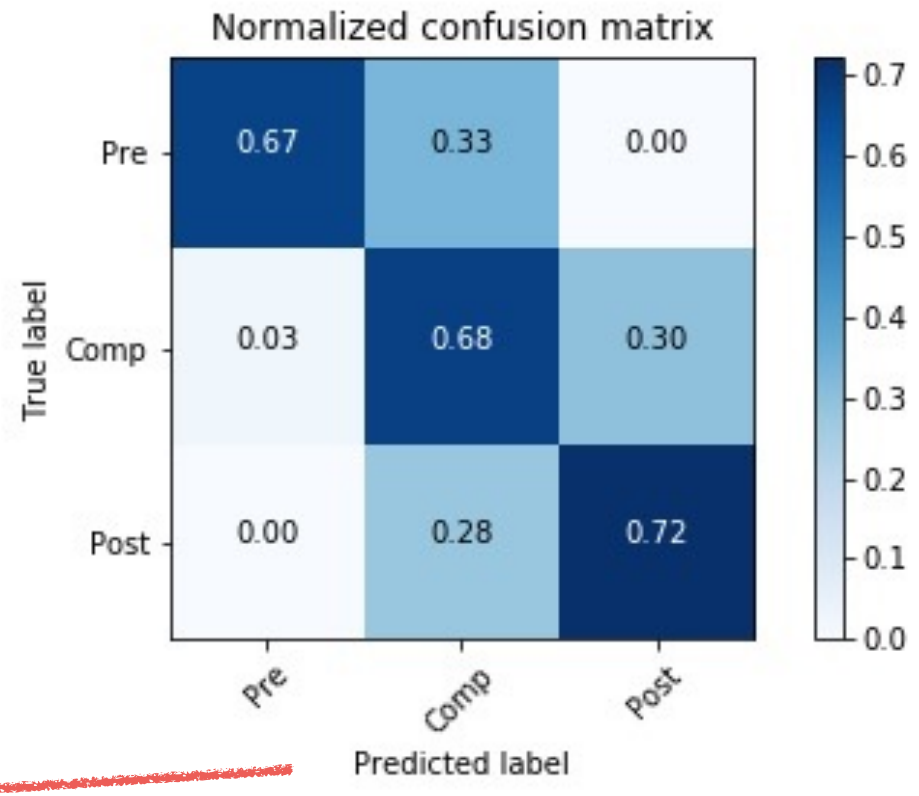
The precision is the ratio `tp / (tp + fp)` where `tp` is the number of true positives and `fp` the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The recall is the ratio `tp / (tp + fn)` where `tp` is the number of true positives and `fn` the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The last precision and recall values are 1. and 0. respectively and do not have a corresponding threshold. This ensures that the graph starts on the x axis.

Evaluation of results [multi-class]

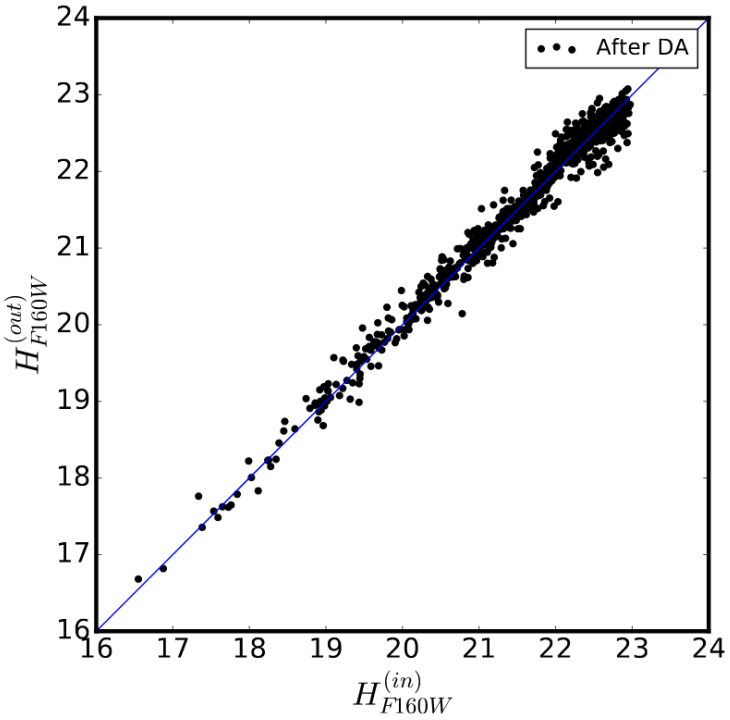
CONFUSION MATRIX



CORRESPONDANCE BETWEEN THE PREDICTED AND TRUE LABEL

Huertas-Company+18

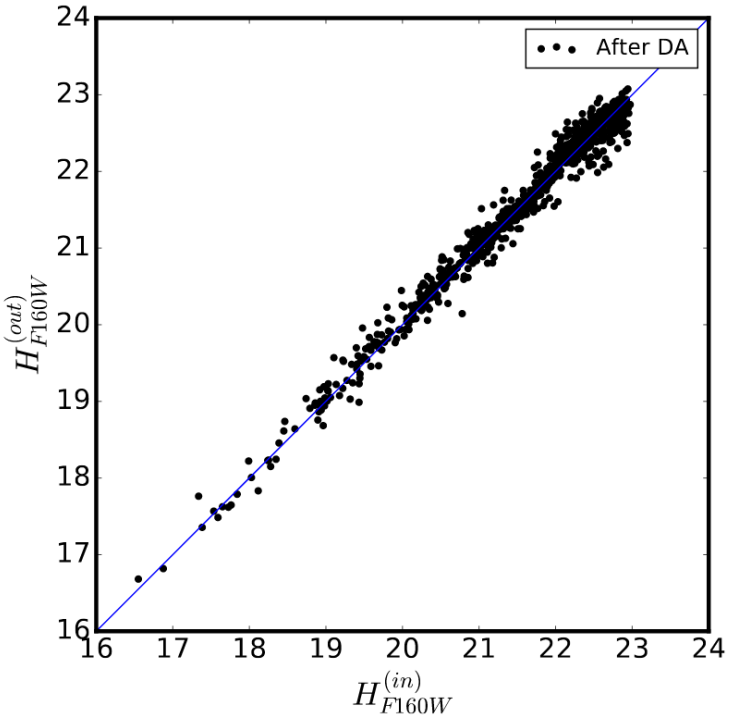
Evaluation of results [regression]



FOR REGRESSIONS, SIMPLY USE
“STANDARD ACCURACY MEASUREMENTS

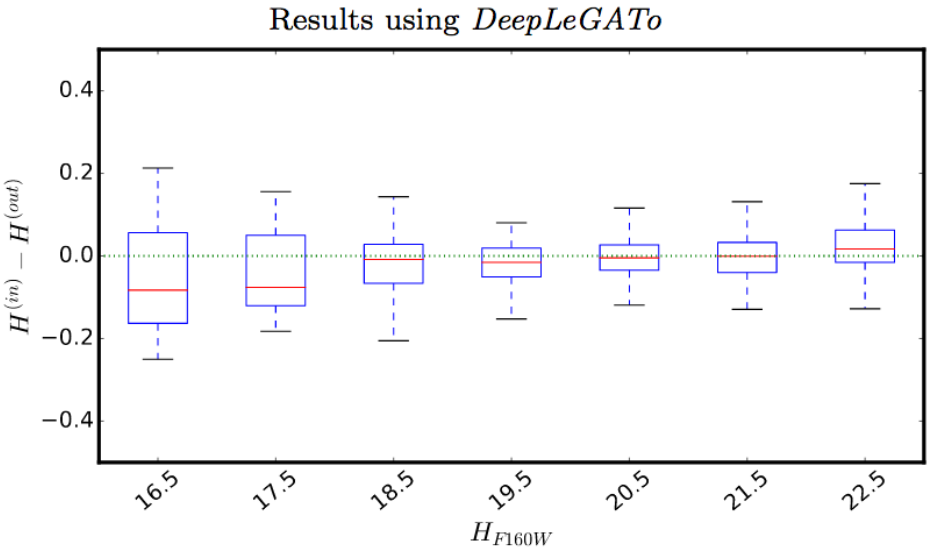
Tuccillo+18

Evaluation of results [regression]



FOR REGRESSIONS, SIMPLY USE
“STANDARD ACCURACY MEASUREMENTS

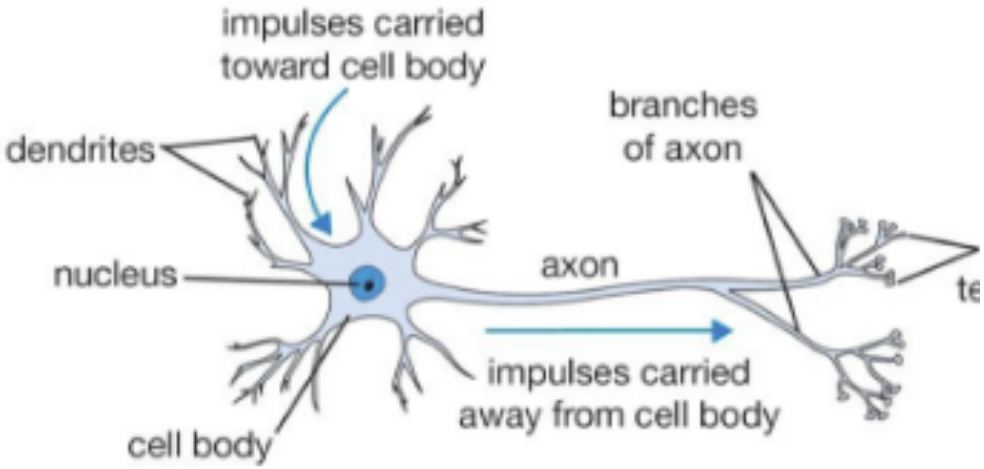
BIAS, SCATTER …YOU KNOW!



Tuccillo+18

PART II: A FOCUS ON "SHALLOW" NEURAL NETWORKS

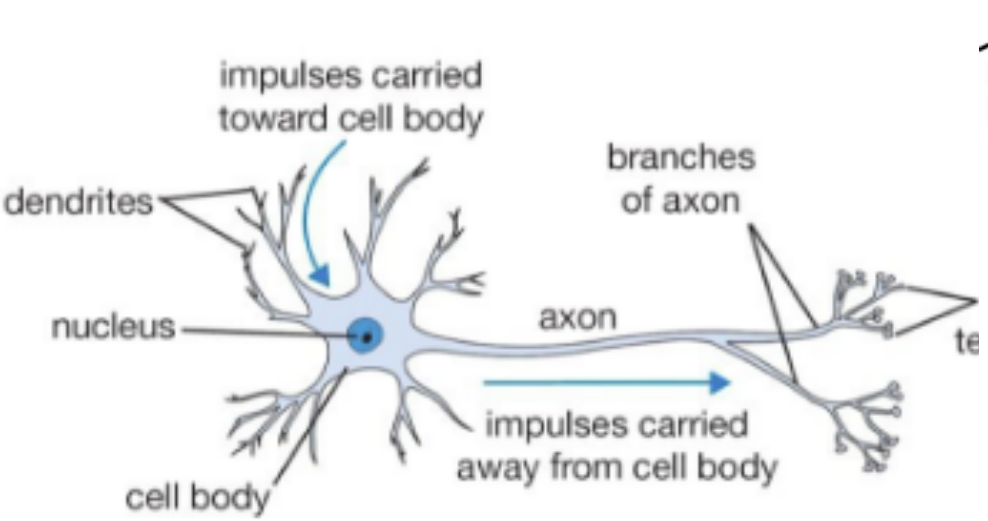
THE NEURON

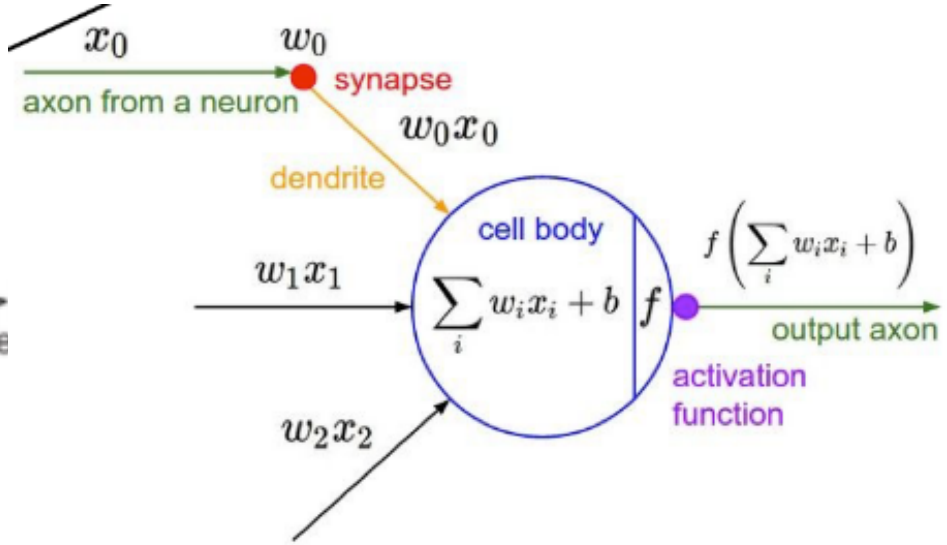


INSPIRED BY NEURO - SCIENCE?

Credit: Karpathy

THE NEURON





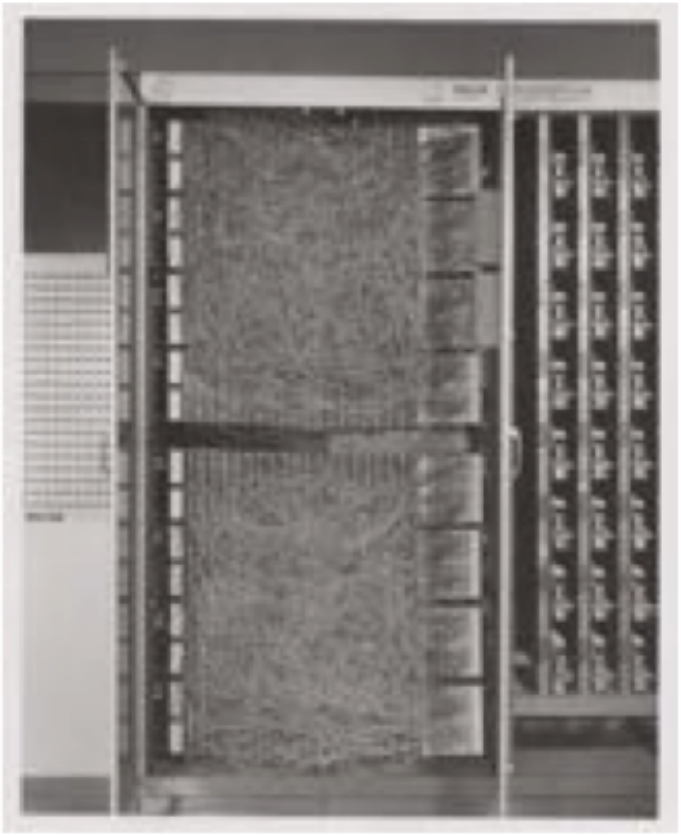
INSPIRED BY NEURO - SCIENCE?

Credit: Karpathy

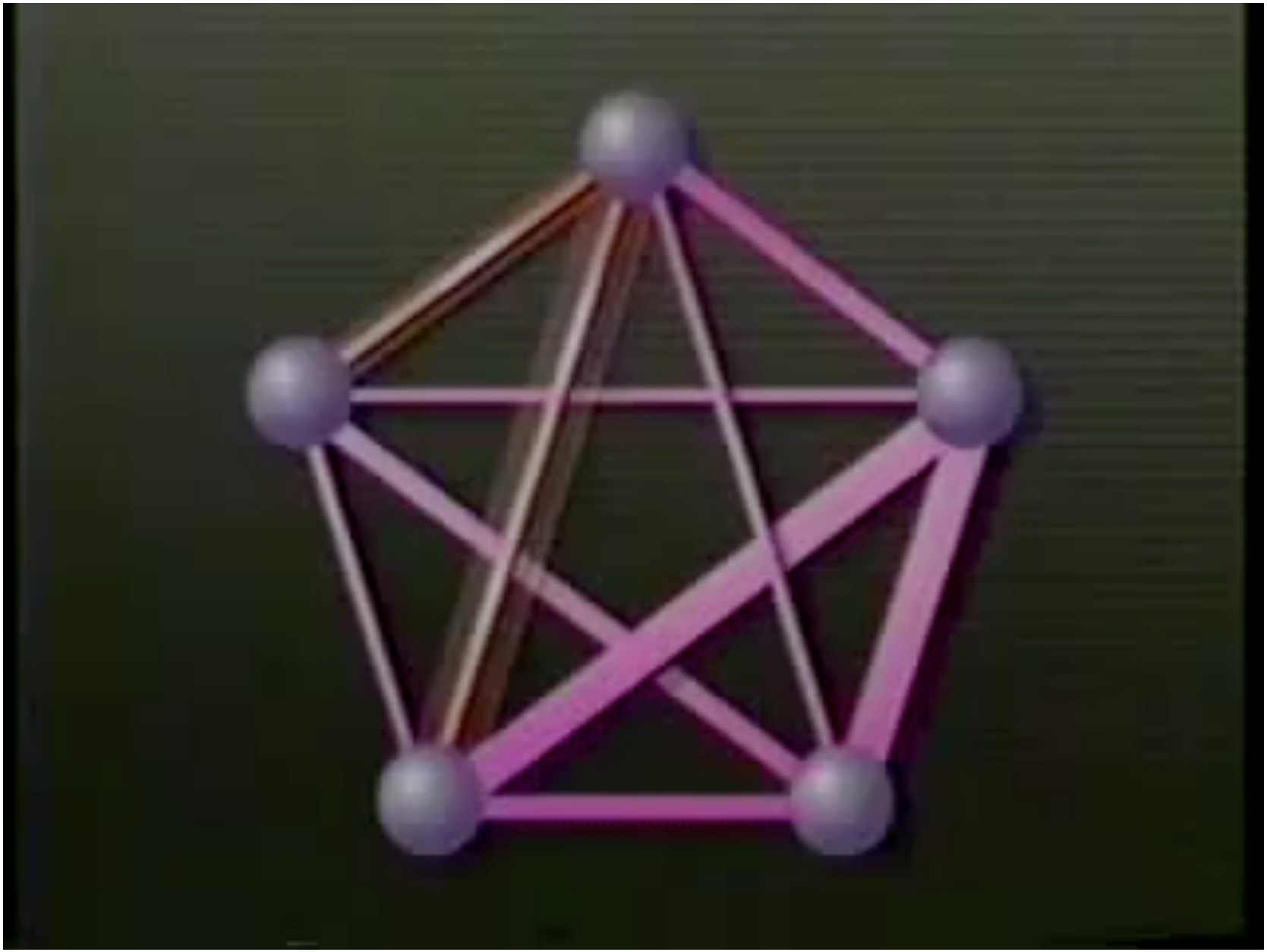
Rosenblatt Perceptron

FIRST IMPLEMENTATION OF NEURAL NETWORK [Rosenblatt, **1957!**]

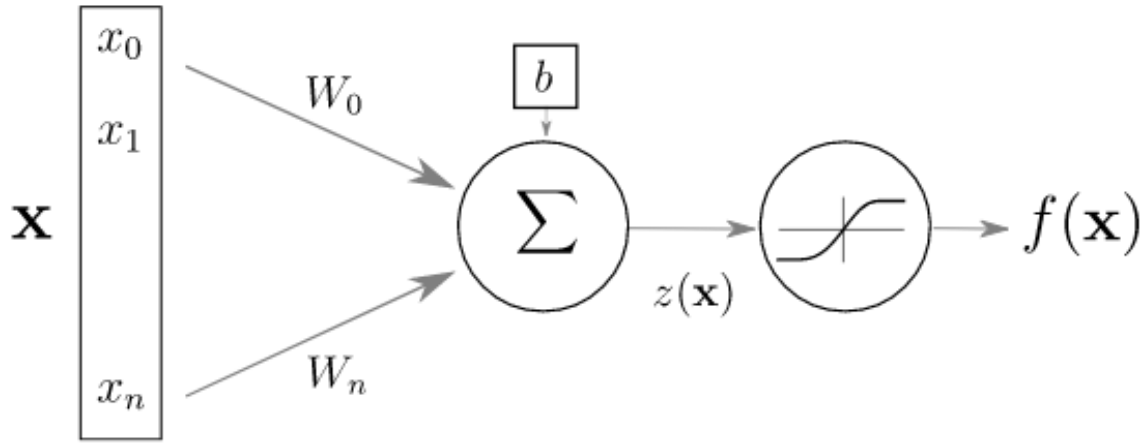
INTENDED TO BE A MACHINE (NOT AN ALGORITHM)



it had an array of 400 photocells, randomly connected to the "neurons". Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors



TODAY'S ARTIFICIAL NEURON



$$z(\vec{x}) = \vec{W}.\vec{x} + b \qquad f(\vec{x}) = g(\vec{W}.\vec{x} + b)$$

Pre-Activation

Weights

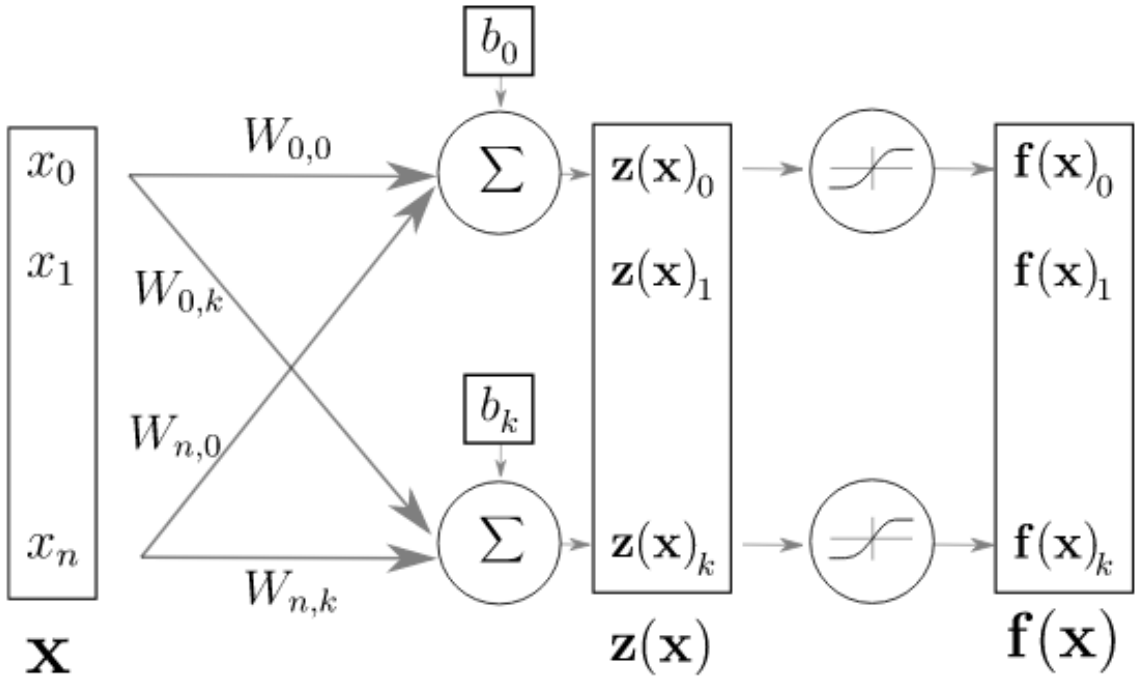
Input

Bias

Output

Activation Function

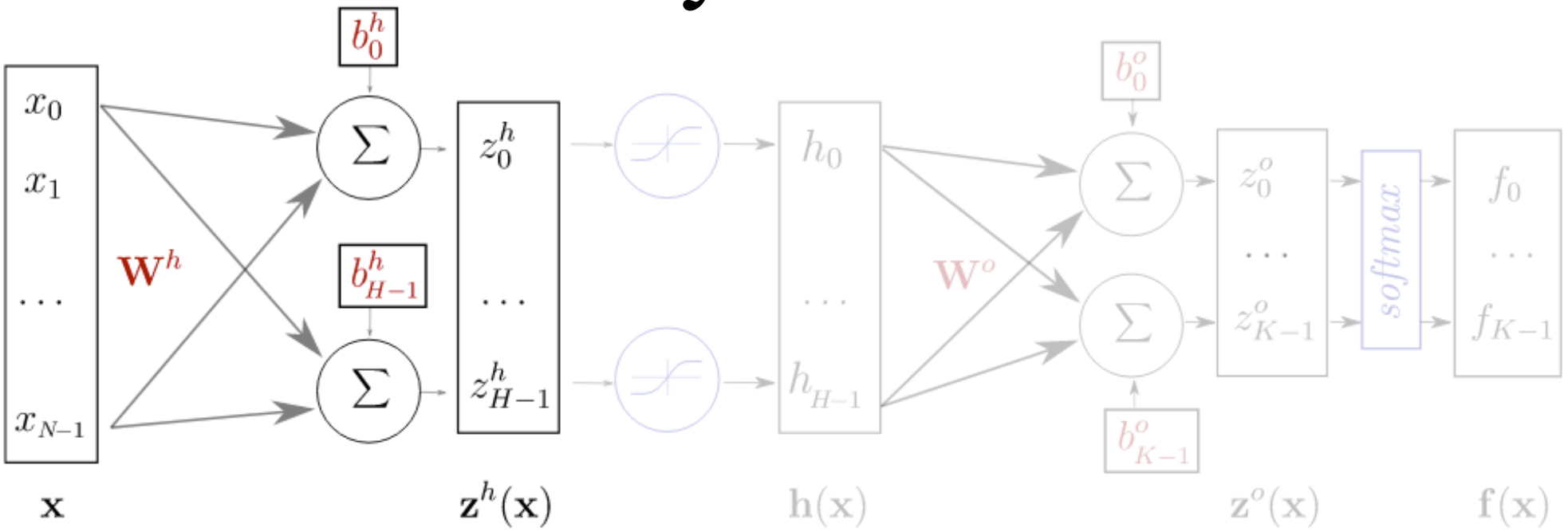
LAYER OF NEURONS



$$f(\vec{x}) = g(\mathbf{W}.\vec{x} + \vec{b})$$

SAME IDEA. NOW **W** becomes a matrix and **b** a vector

Hidden Layers of Neurons

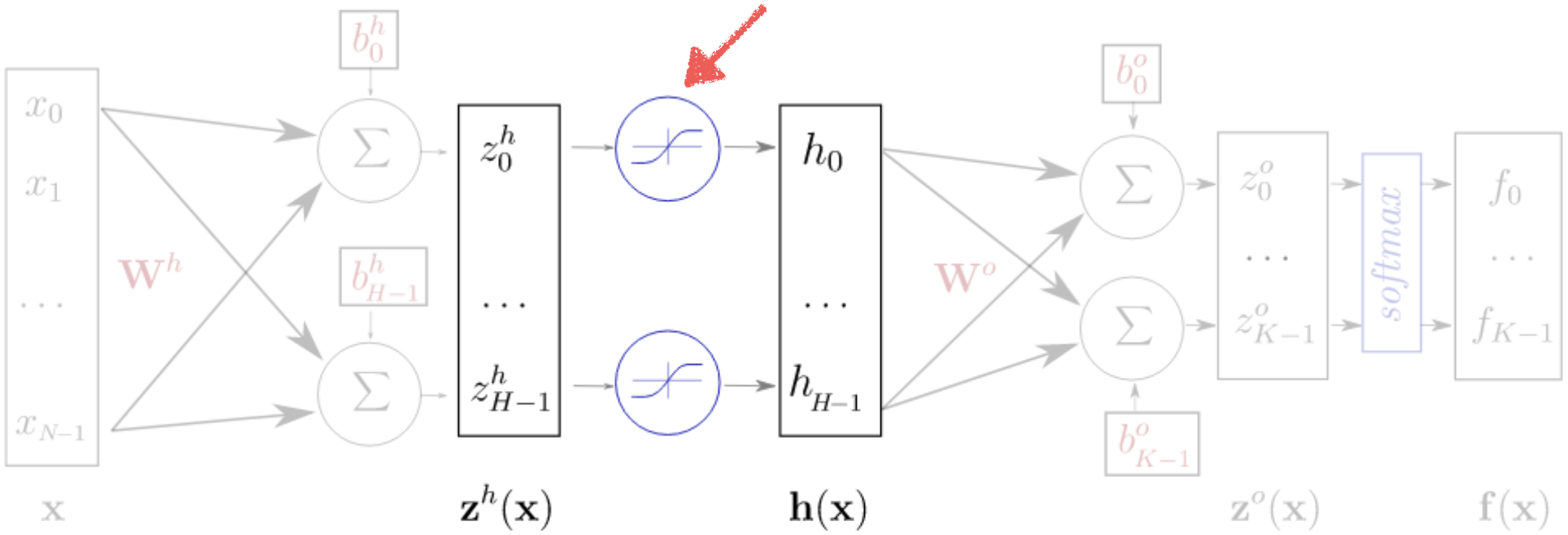


FIRST LAYER

INPUT

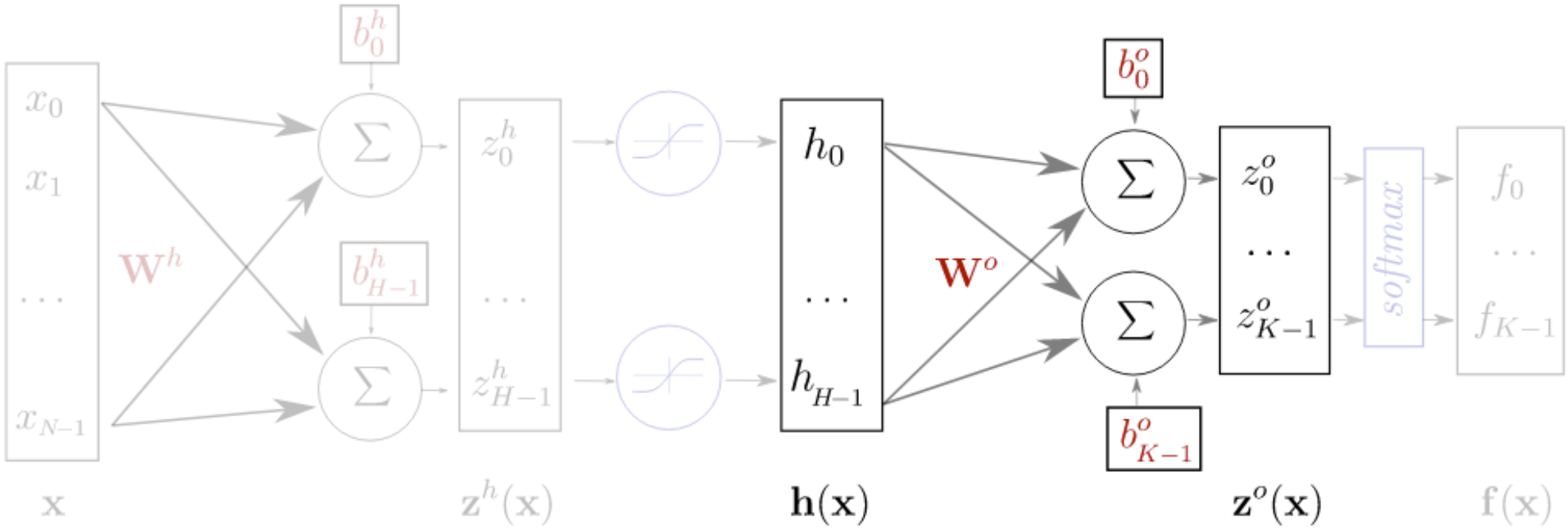
$$z^h(x) = W^h x + b^h$$

ACTIVATION FUNCTION



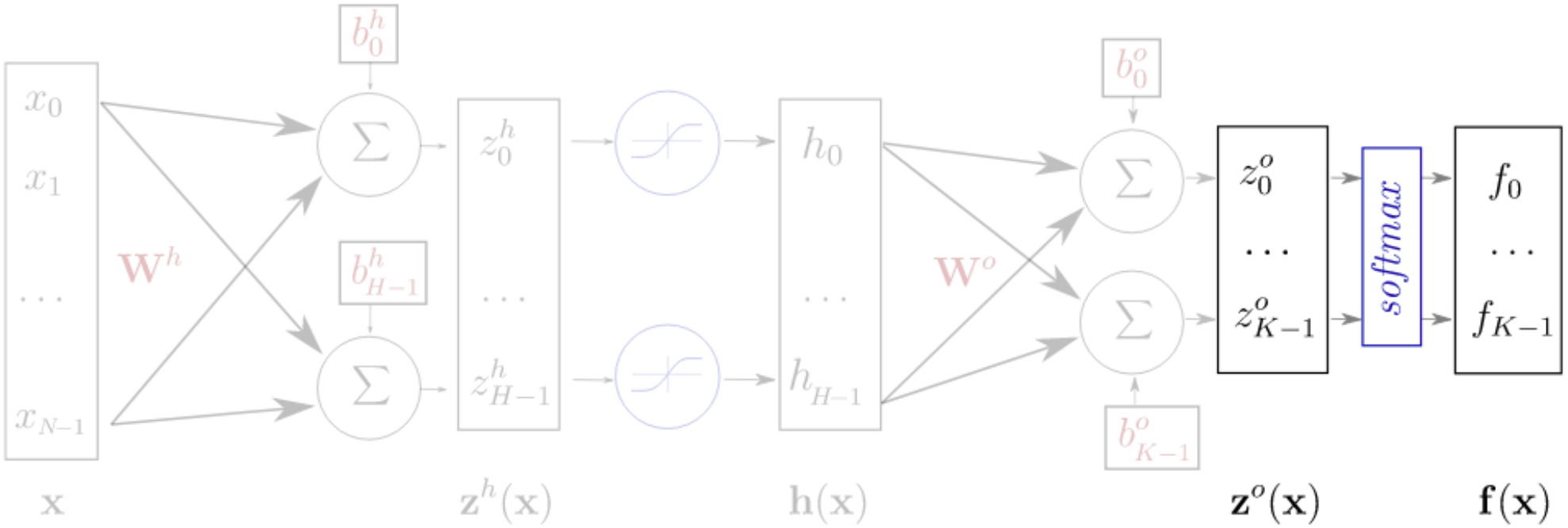
HIDDEN LAYER

$$h(x) = g(z^h(x)) = g(W^h x + b^h)$$



OUTPUT LAYER

$$z^0(\mathbf{x}) = W^0 h(\mathbf{x}) + b^0$$



PREDICTION LAYER

$$f(\mathbf{x}) = softmax(\mathbf{z^0})$$

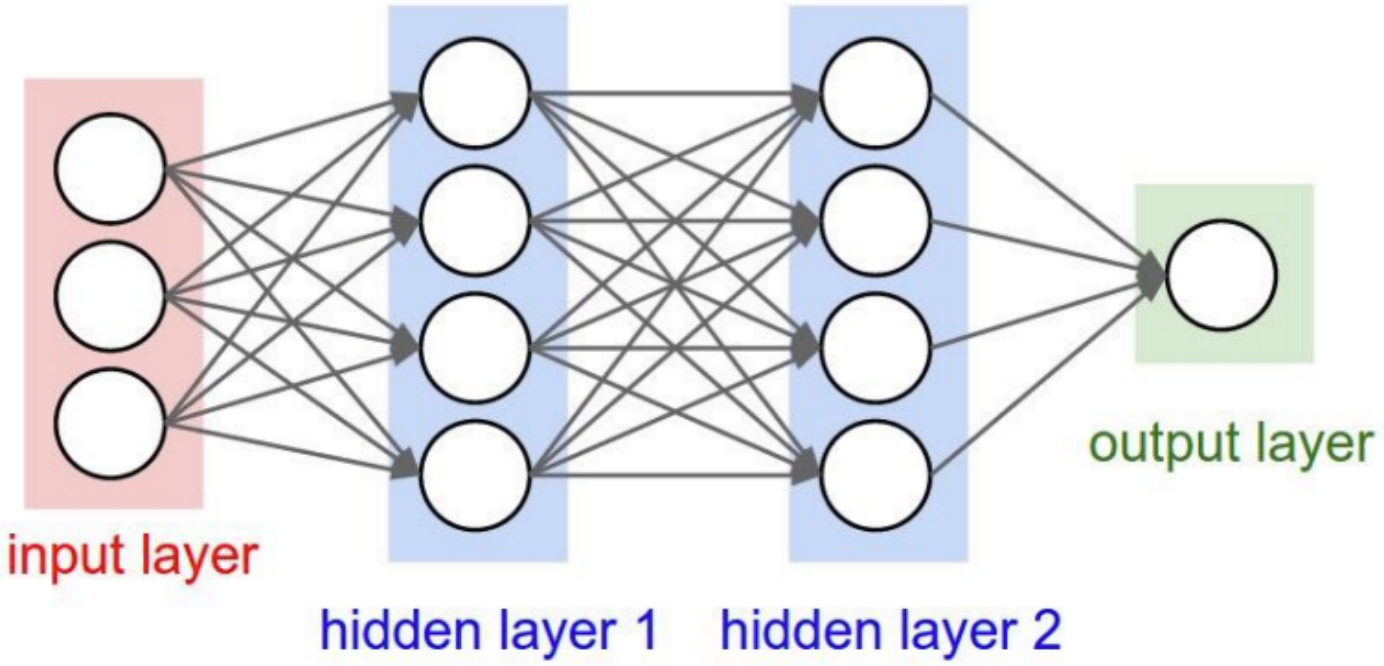
“CLASSICAL” MACHINE LEARNING

$$f_{\vec{W}}(\vec{x}) = \vec{y}$$

LABEL

Q , SF

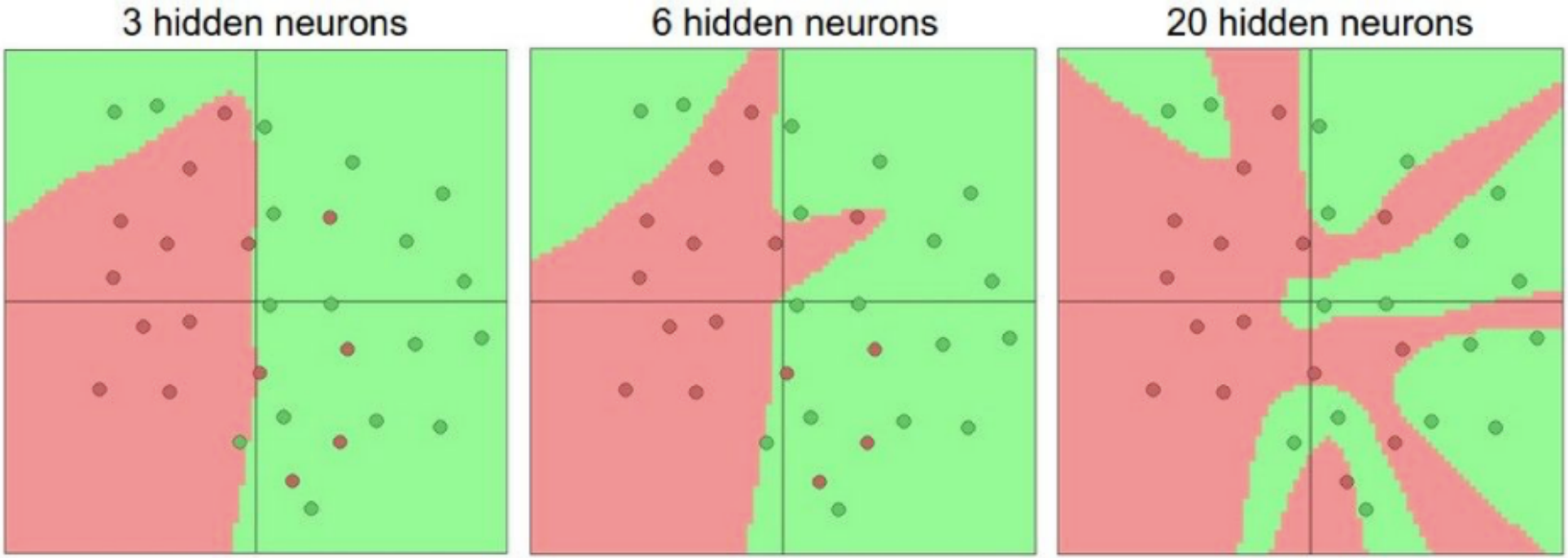
REPLACE THIS BY A GENERAL NON LINEAR FUNCTION WITH SOME PARAMETERS W



$$p = g_3(W_3 g_2(W_2 g_1(W_1 \vec{x_0})))$$

NETWORK FUNCTION

WHY HIDDEN LAYERS?



More complex functions allow increasing complexity

Credit: Karpathy

SO LET’S GO DEEPER AND DEEPER!

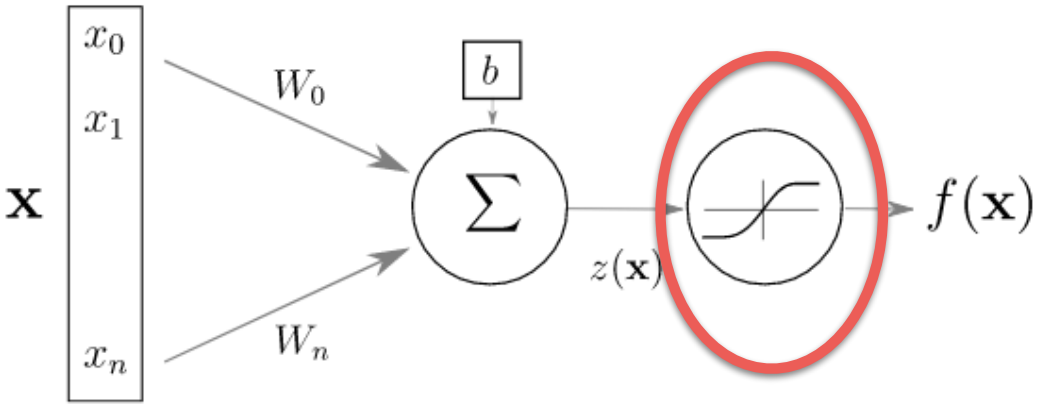
SO LET’S GO DEEPER AND DEEPER!

YES BUT…

NOT SO STRAIGHTFORWARD, DEEPER MEANS MORE WEIGHTS, MORE DIFFICULT OPTIMIZATION, RISK OF OVERFITTING…

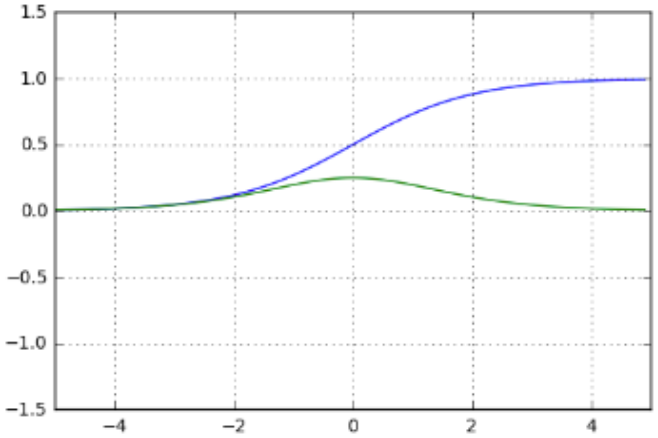
LET’S FIRST EXAMINE IN MORE DETAIL HOW SIMPLE “SHALLOW” NETWORKS WORK

ACTIVATION FUNCTIONS?



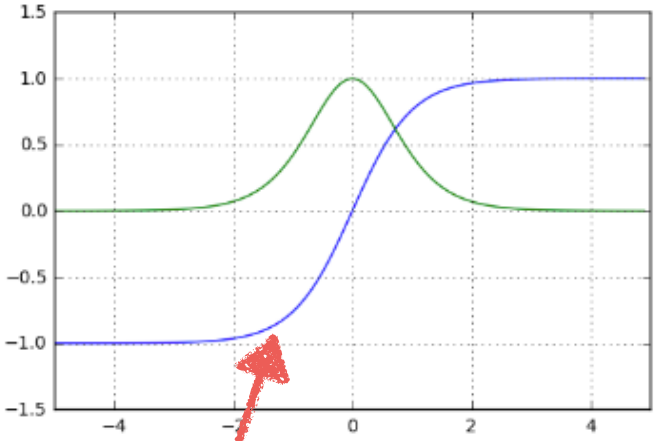
ADD NON LINEARITIES TO THE PROCESS

ACTIVATION FUNCTIONS



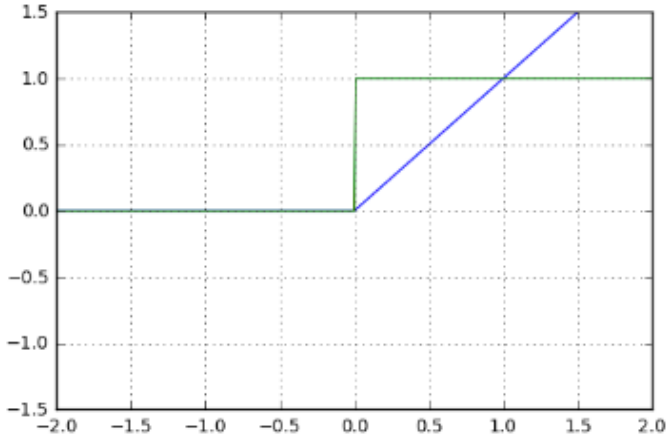
$$\text{sigm}(x) = \frac{1}{1+e^{-x}}$$

$$\text{sigm}'(x) = \text{sigm}(x)(1-\text{sigm}(x))$$



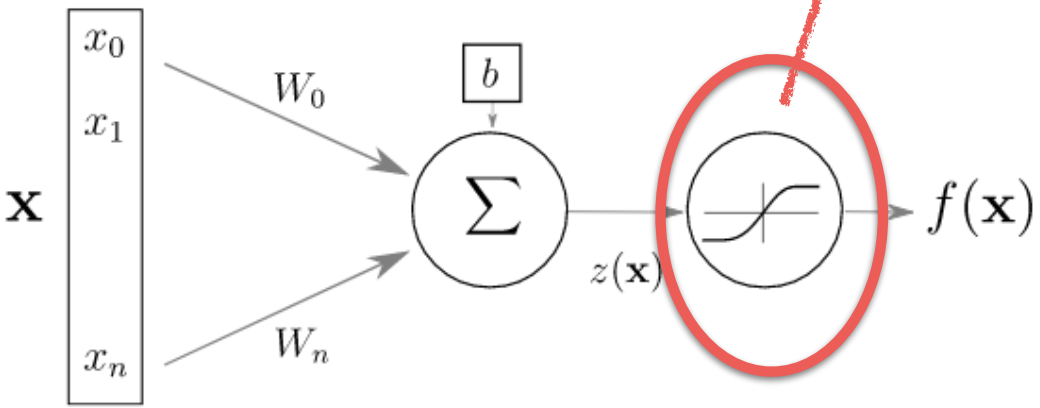
$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

$$\tanh'(x) = 1-\tanh(x)^2$$



$$\text{relu}(x) = \max(0,x)$$

$$\text{relu}'(x) = 1_{x>0}$$



SOFTMAX

A generalization of the SIGMOID ACTIVATION

$$softmax(\mathbf{x}) = \frac{e^{\mathbf{x}}}{\sum_{i=1}^{n} e^{x_i}}$$

THE OUTPUT IS NORMALIZED BETWEEN 0 AND 1

THE COMPONENTS ADD TO 1

CAN BE INTERPRETED AS A PROBABILITY

$$p(Y = c|X = \mathbf{x}) = softmax(z(\mathbf{x}))_c$$

SOFTMAX

A generalization of the SIGMOID ACTIVATION

GENERALLY
USED AS ACTIVATION
OF LAST LAYER

(will come back later)

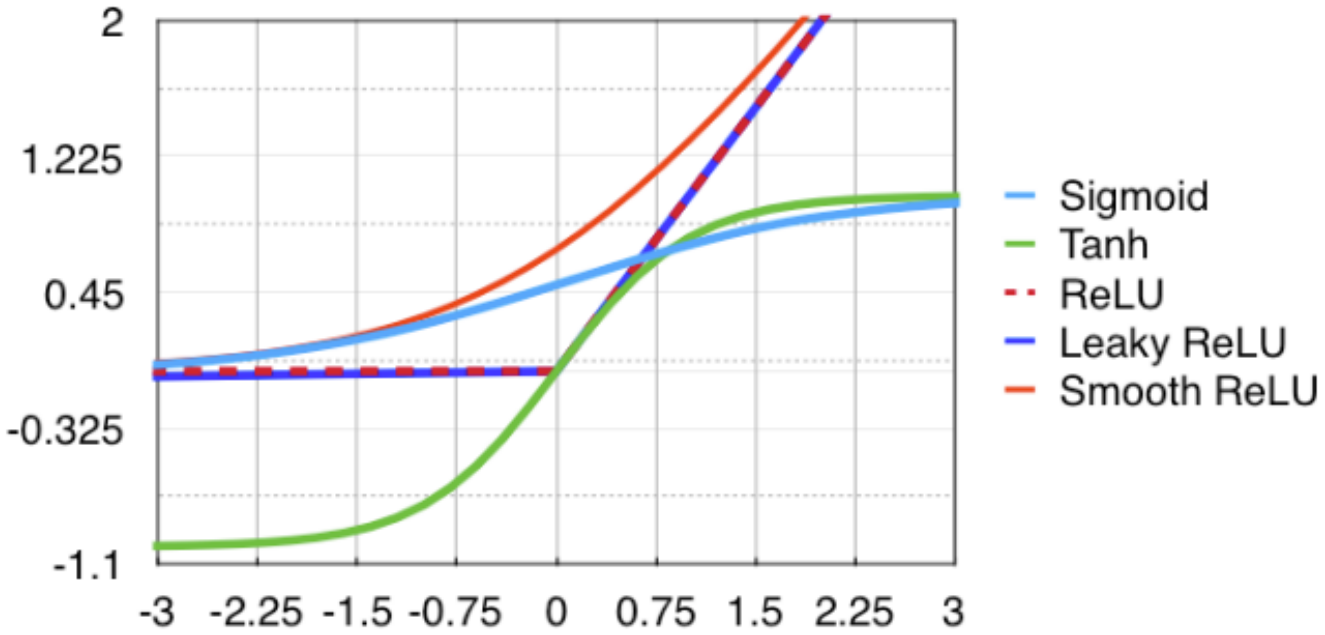
THE OUTP AND 1

THE COMPONENTS ADD TO 1

CAN BE INTERPRETED AS A PROBABILITY

$$p(Y = c|X = \mathbf{x}) = softmax(z(\mathbf{x}))_c$$

ACTIVATION FUNCTIONS



Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$

Tanh: $f(x) = tanh(x)$

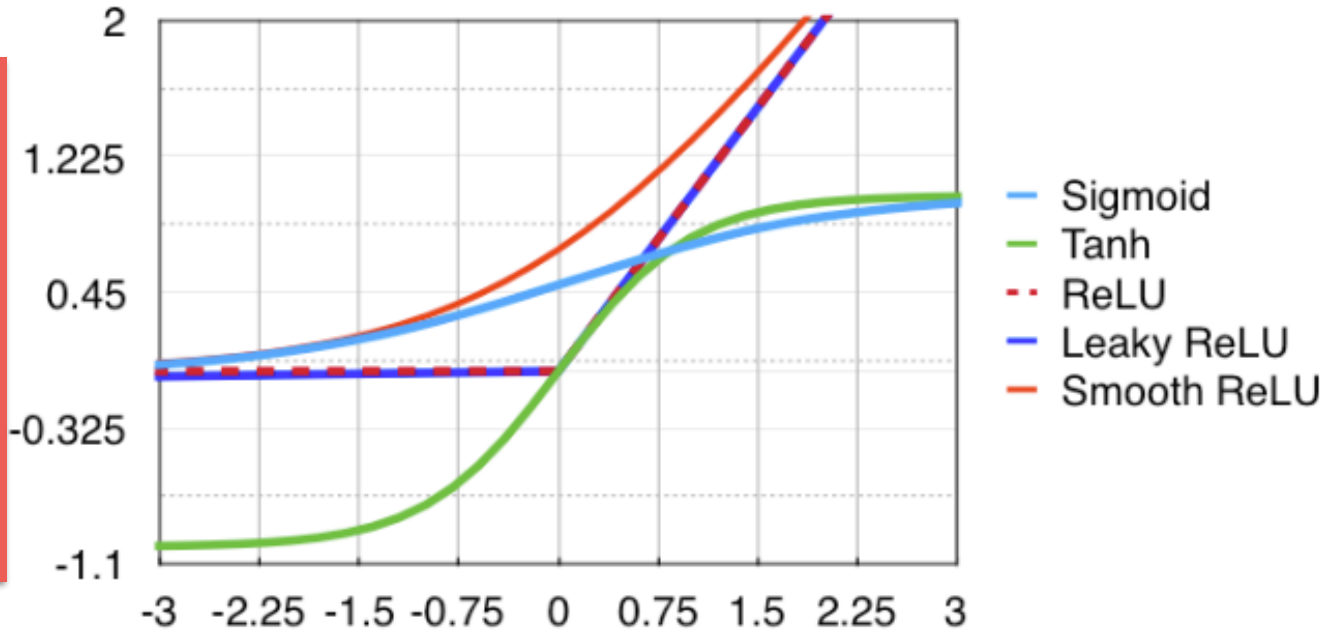
ReLu: $f(x) = max(0, x)$

Soft ReLu: $f(x) = log(1+e^{x})$

Leaky ReLu: $f(x) = \epsilon x + (1-\epsilon)max(0, x)$

ACTIVATION FUNCTIONS

+ MANY OTHERS!



Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$

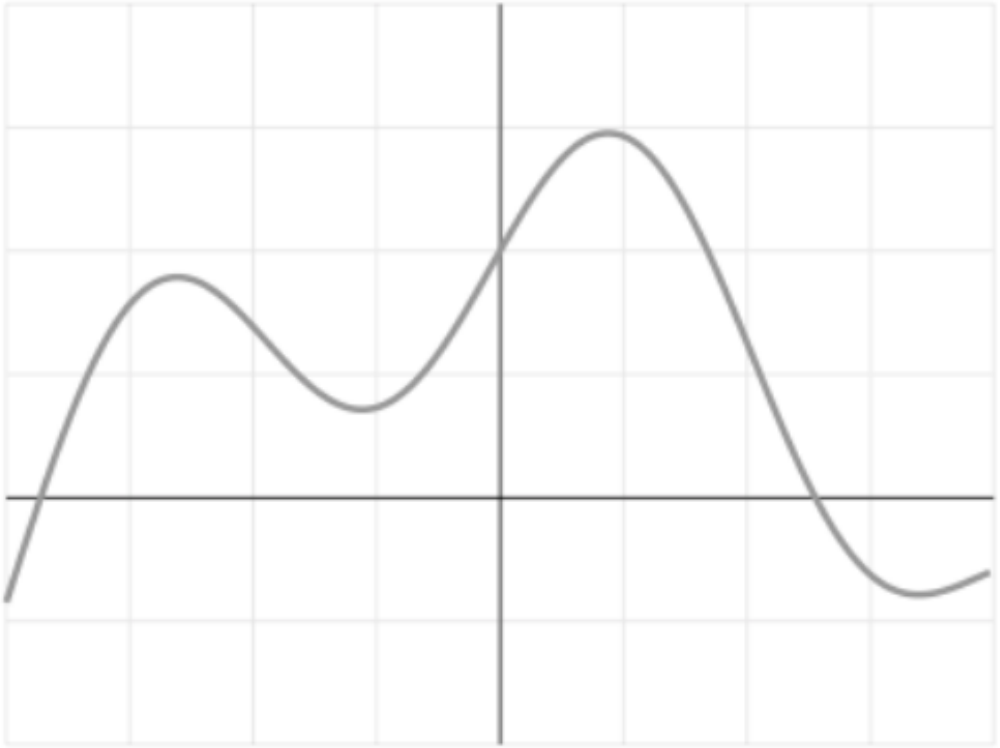
Tanh: $f(x) = tanh(x)$

ReLu: $f(x) = max(0, x)$

Soft ReLu: $f(x) = log(1+e^{x})$

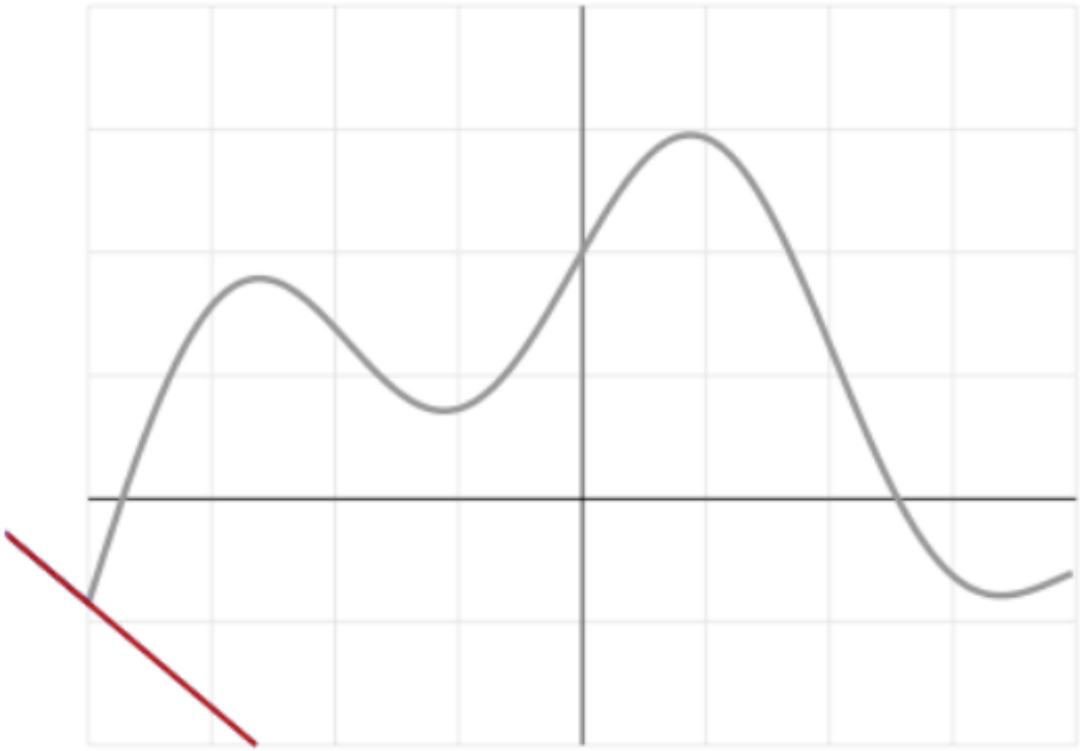
Leaky ReLu: $f(x) = \epsilon x + (1-\epsilon)max(0, x)$

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



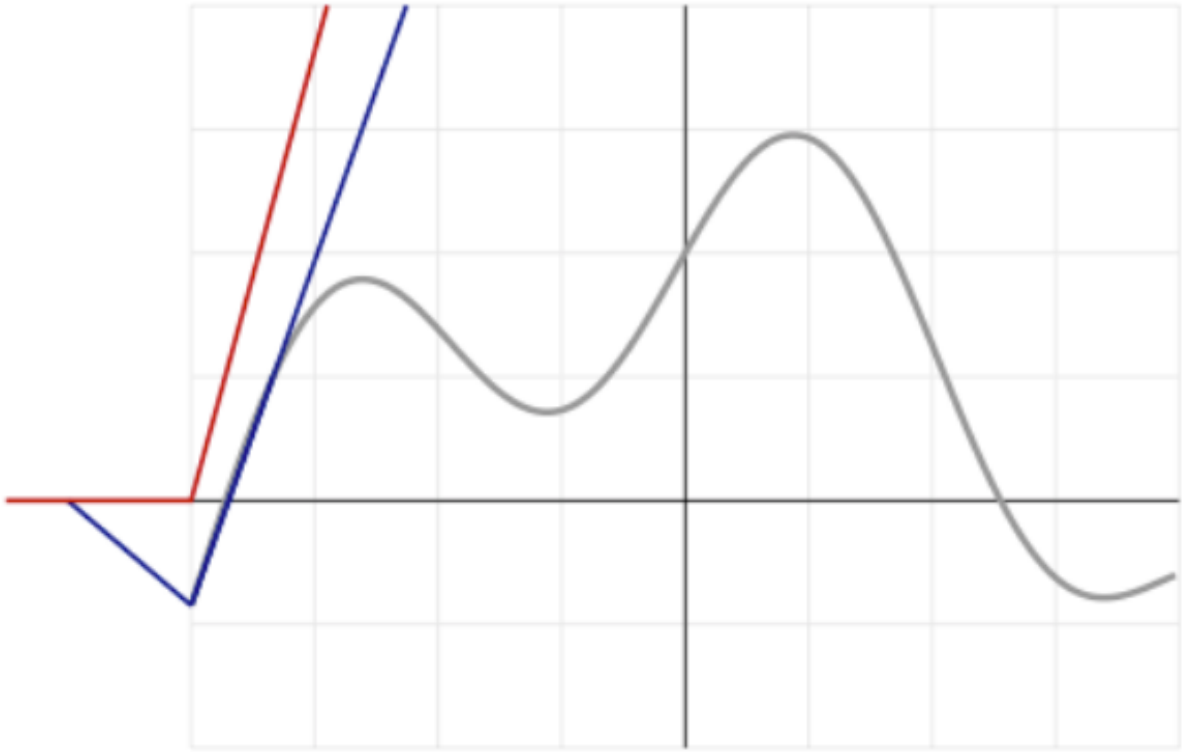
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



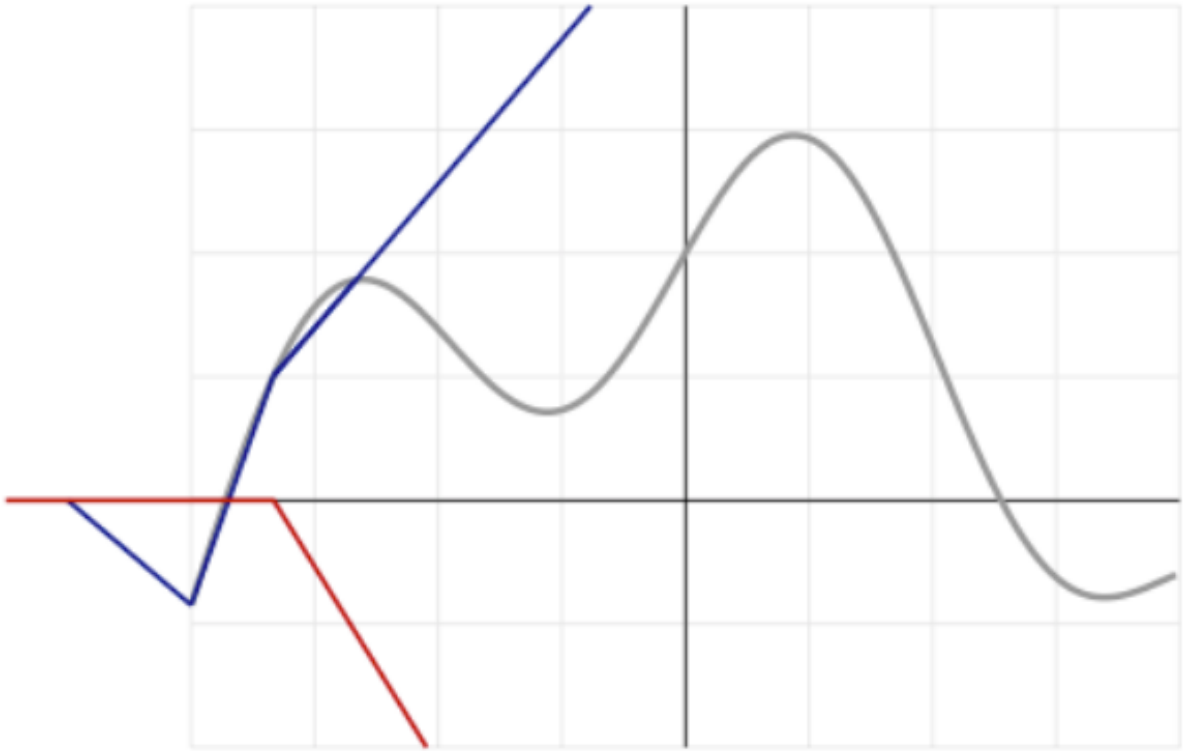
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



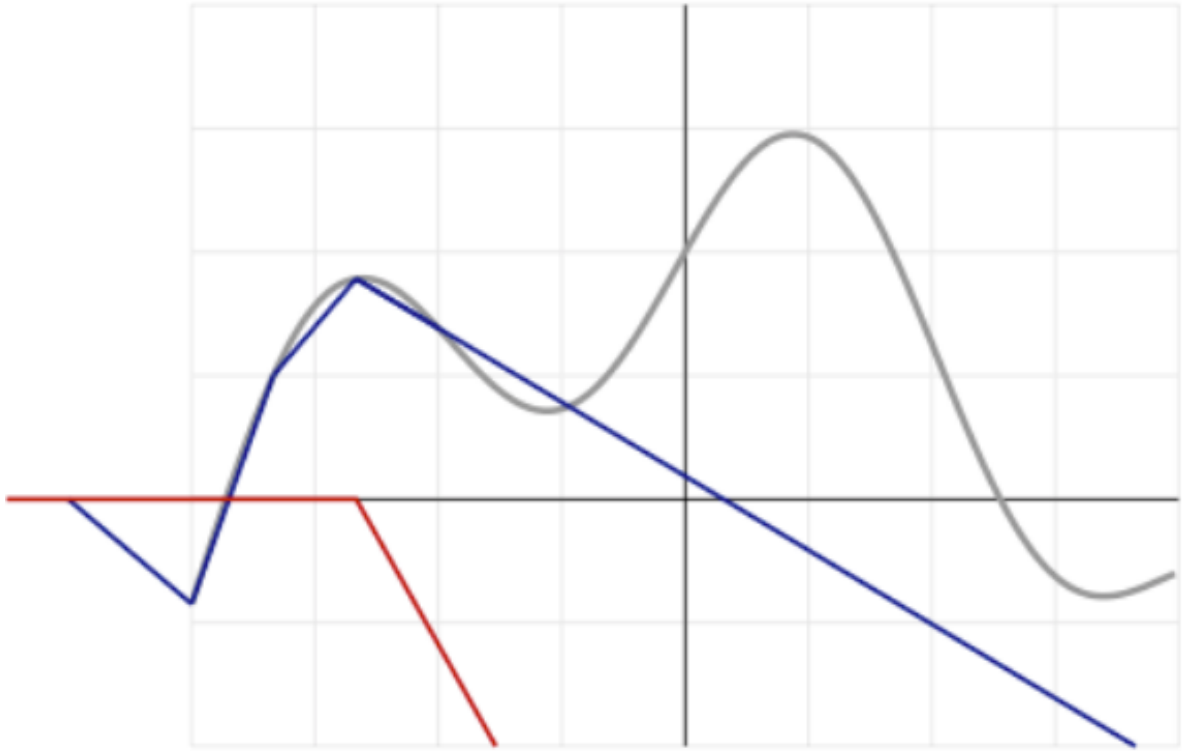
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



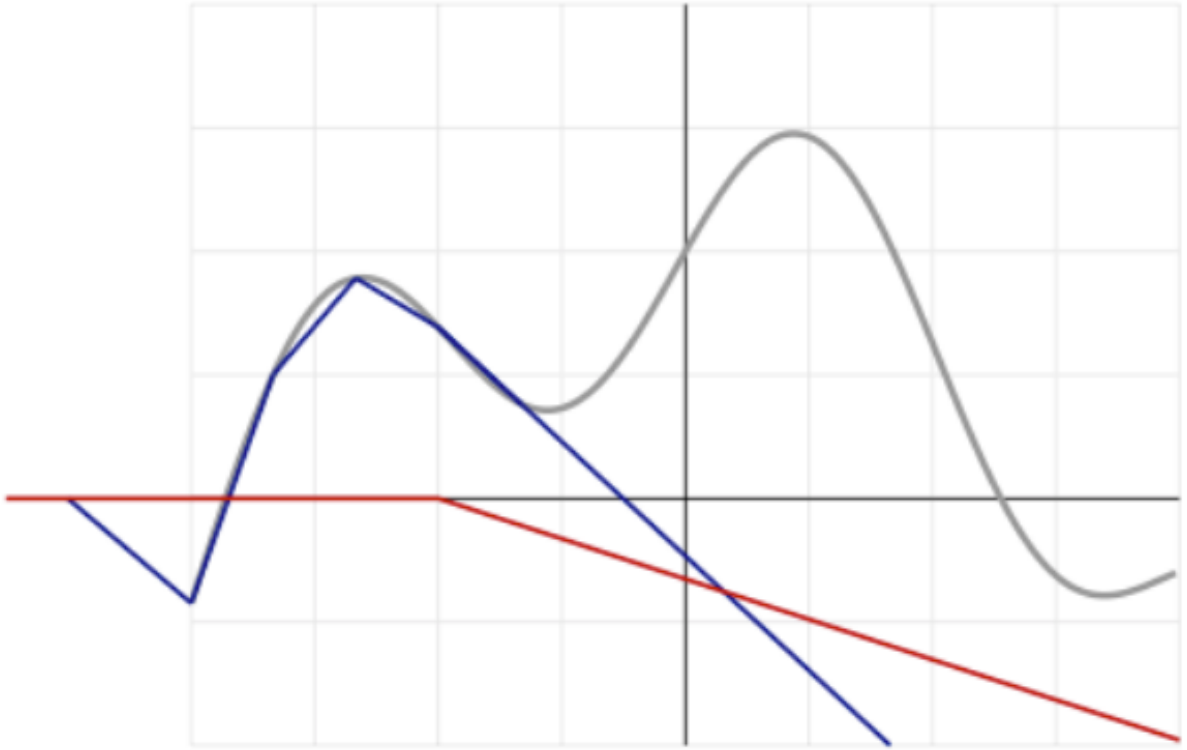
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



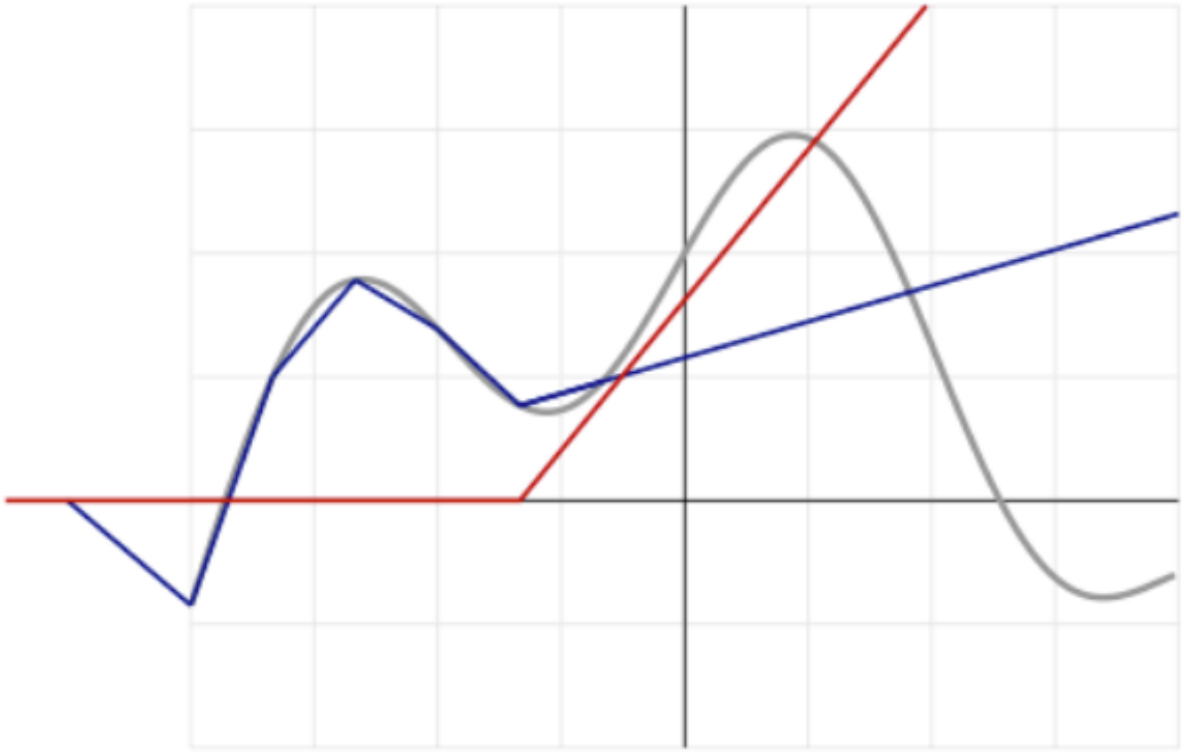
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



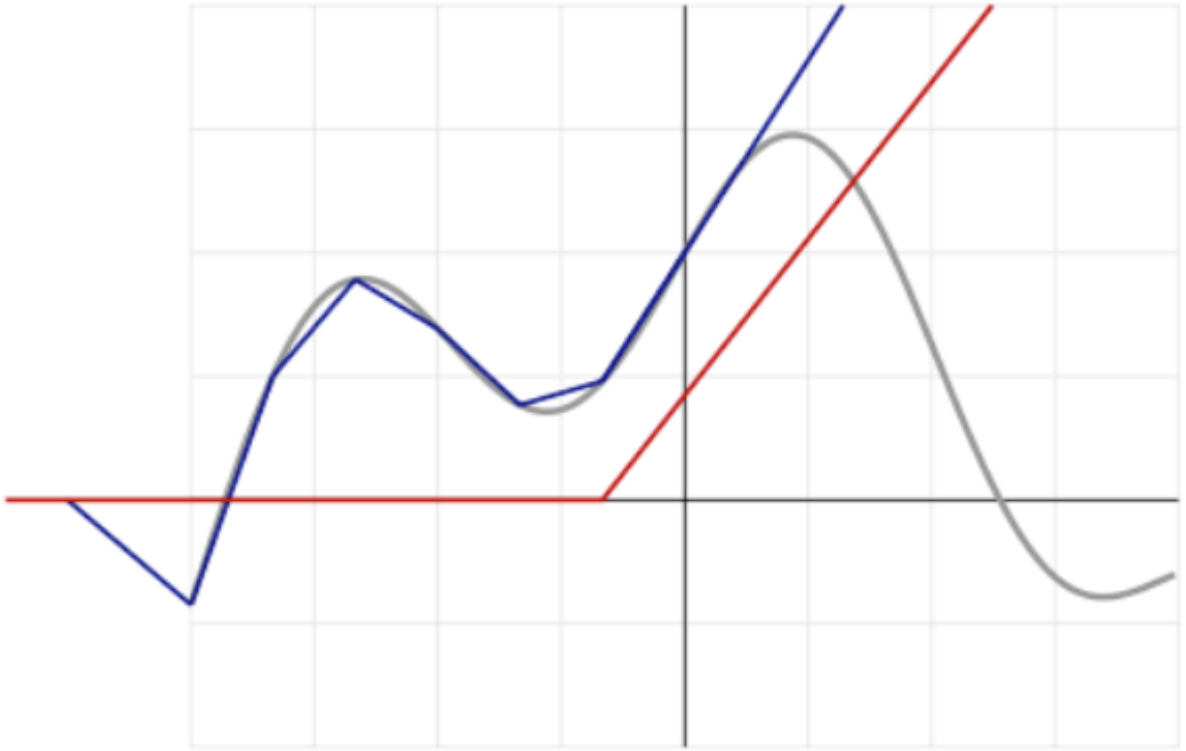
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



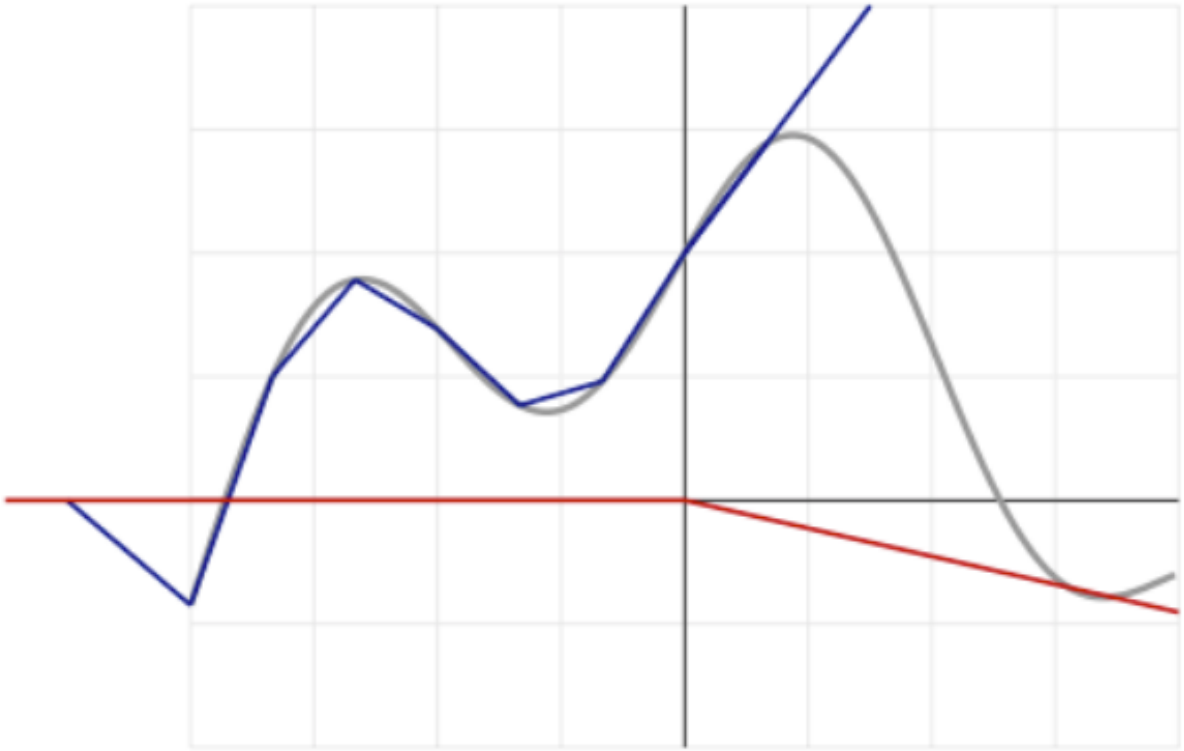
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



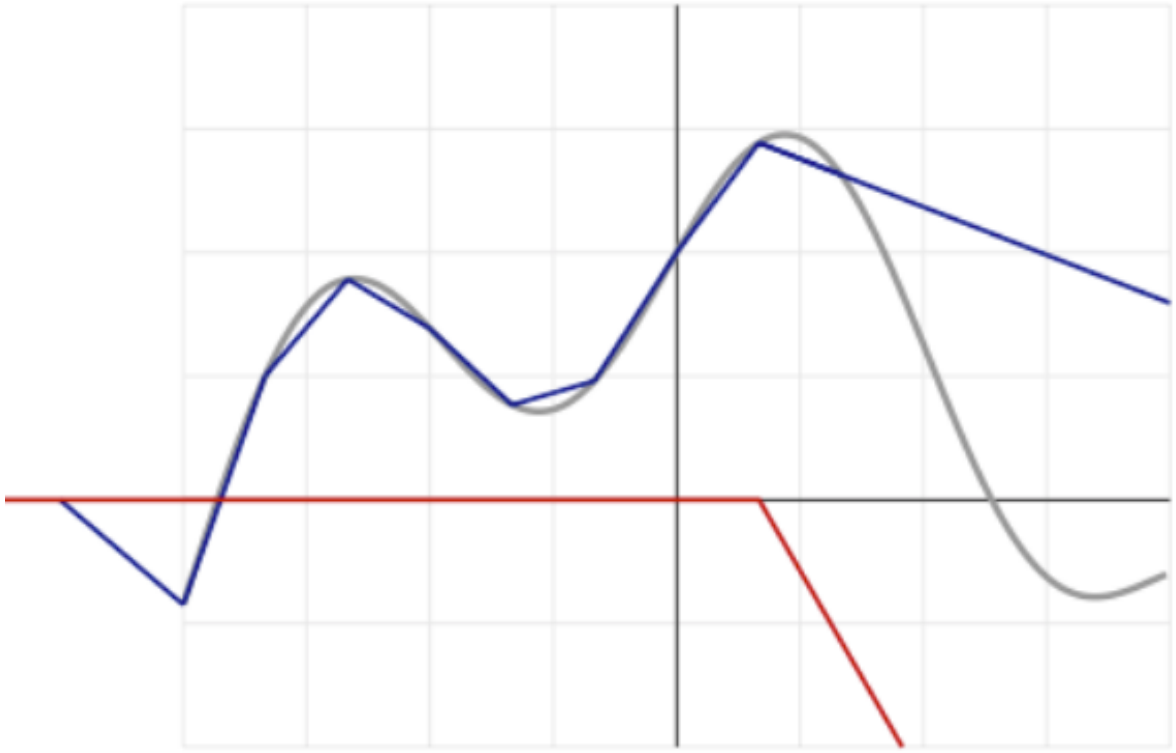
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



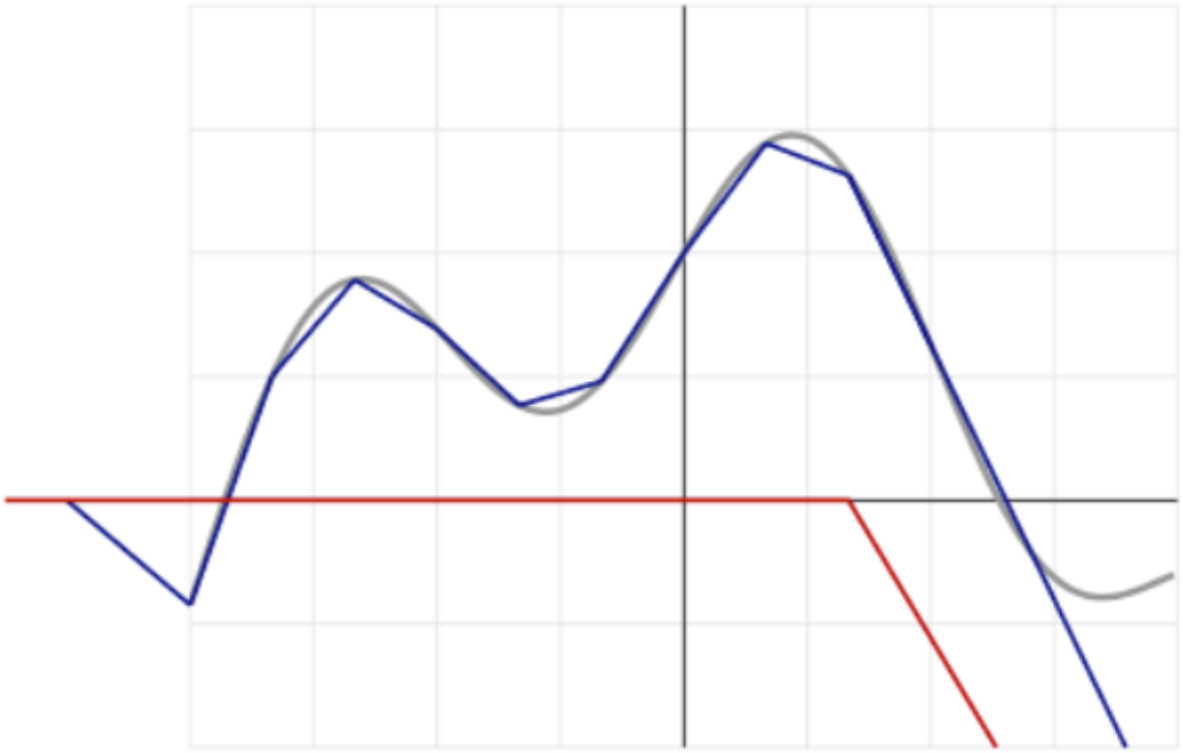
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



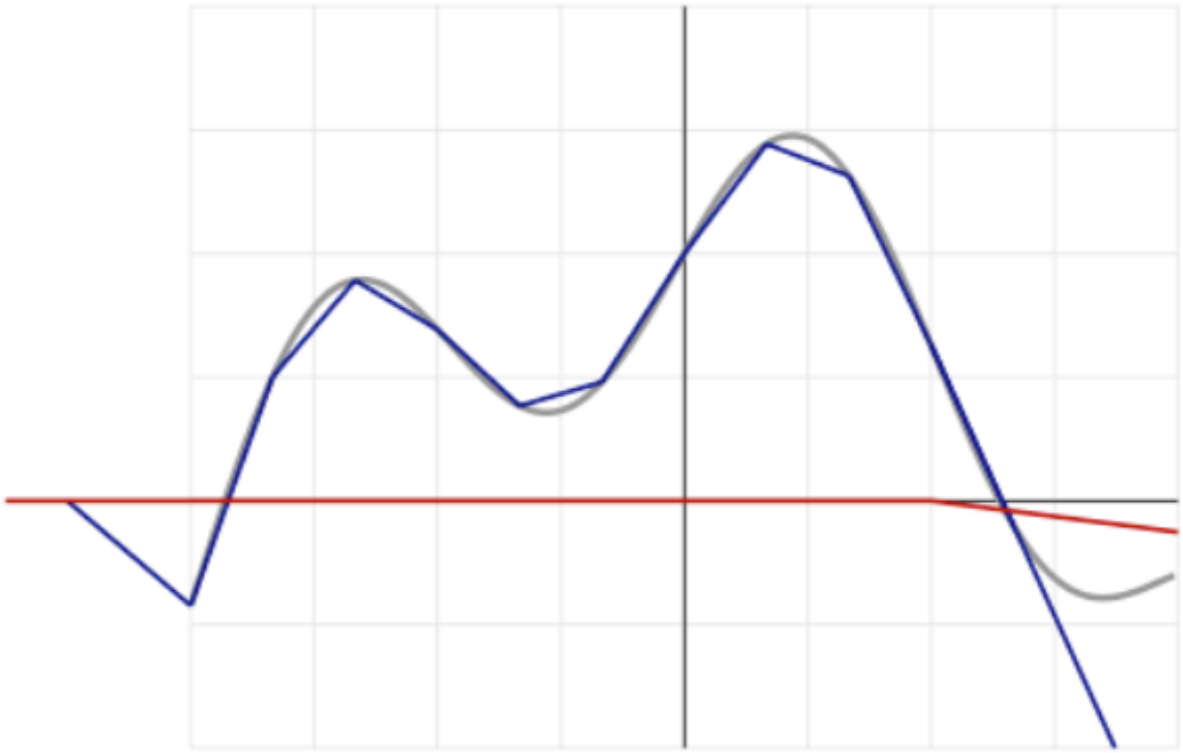
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



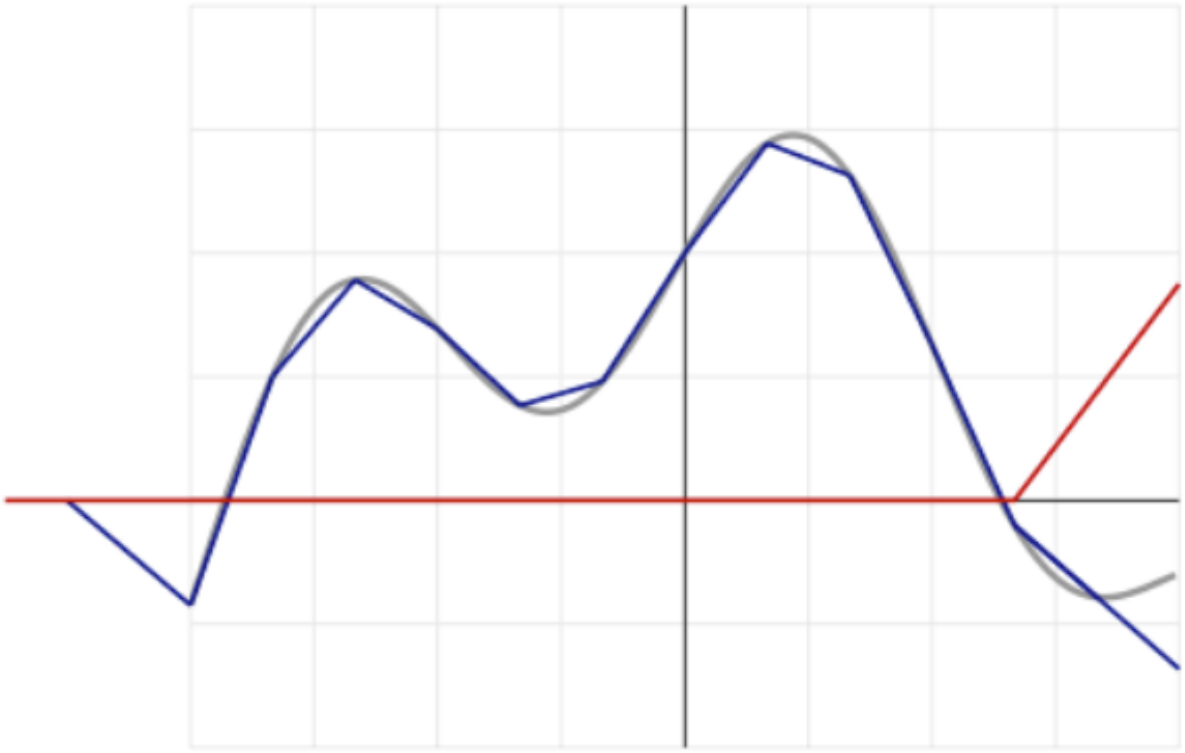
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



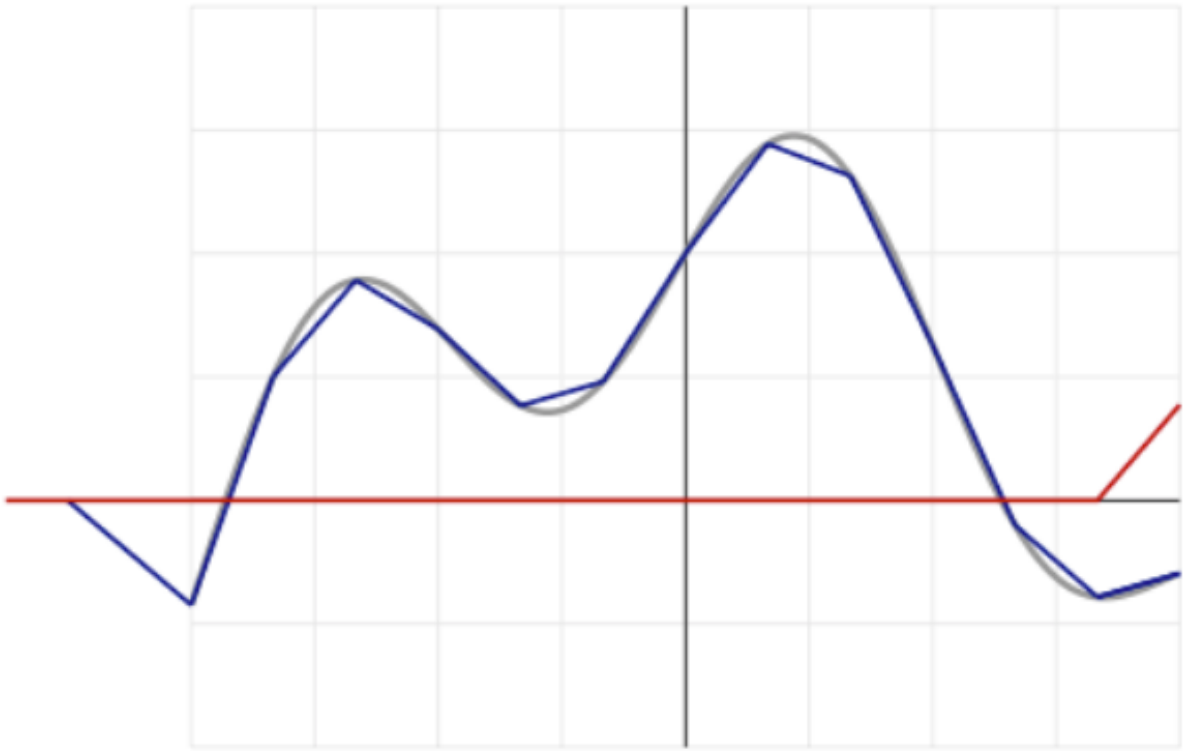
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



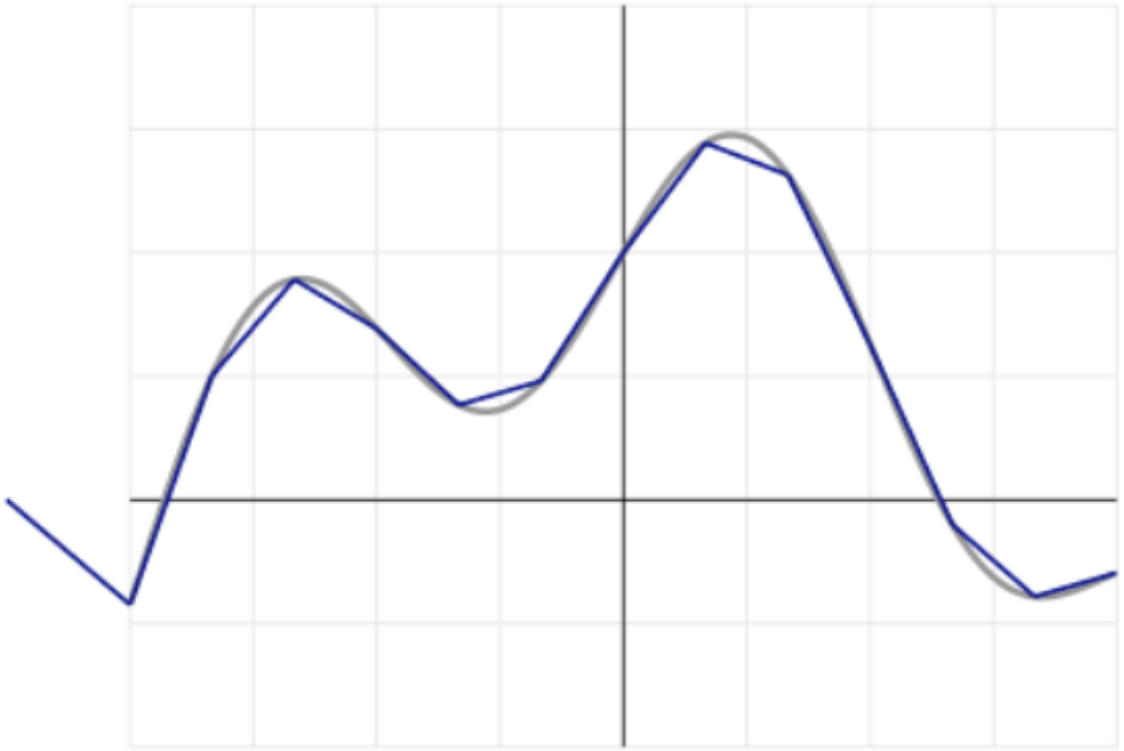
Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?



Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

WHAT IS THE MEANING OF THE ACTIVATION FUNCTION?

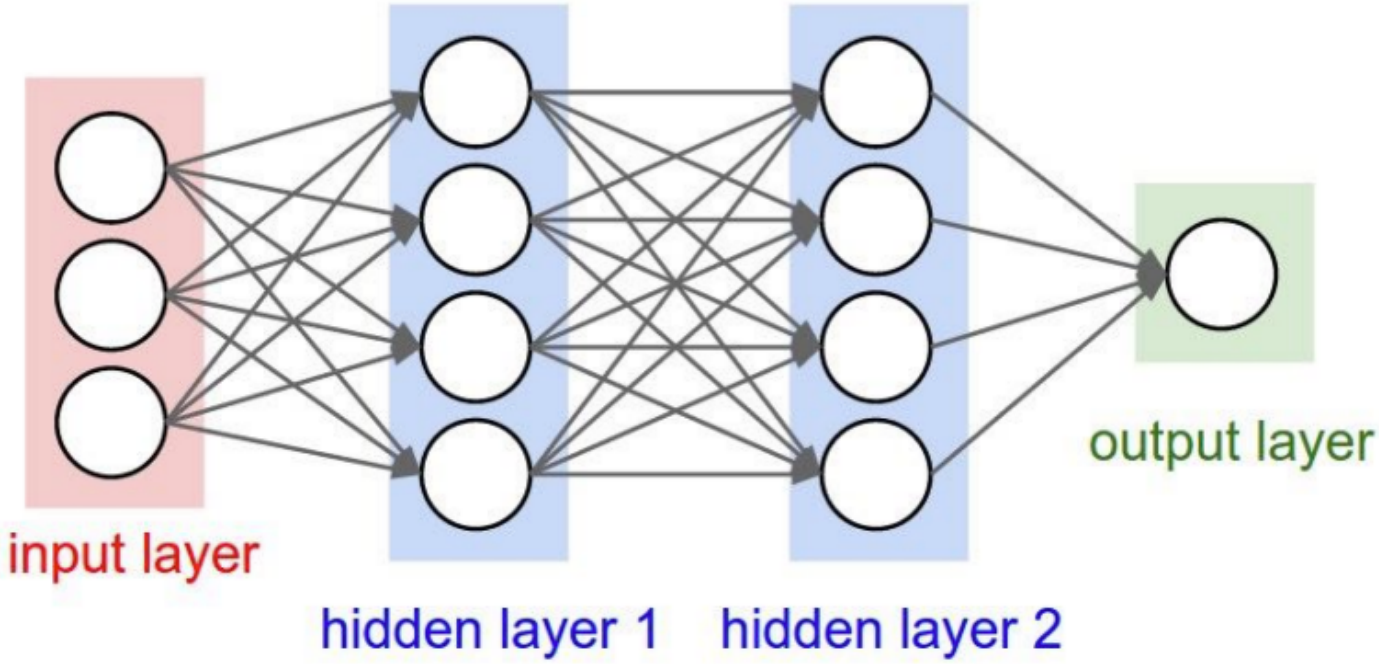


Any real function in a interval (a,b) can be approximated with a linear combination of translated and scaled ReLu functions

OK, SO NOW LET'S FIND THE WEIGHTS

OPTIMIZATION

[OR HOW TO FIND THE WEIGHTS?]

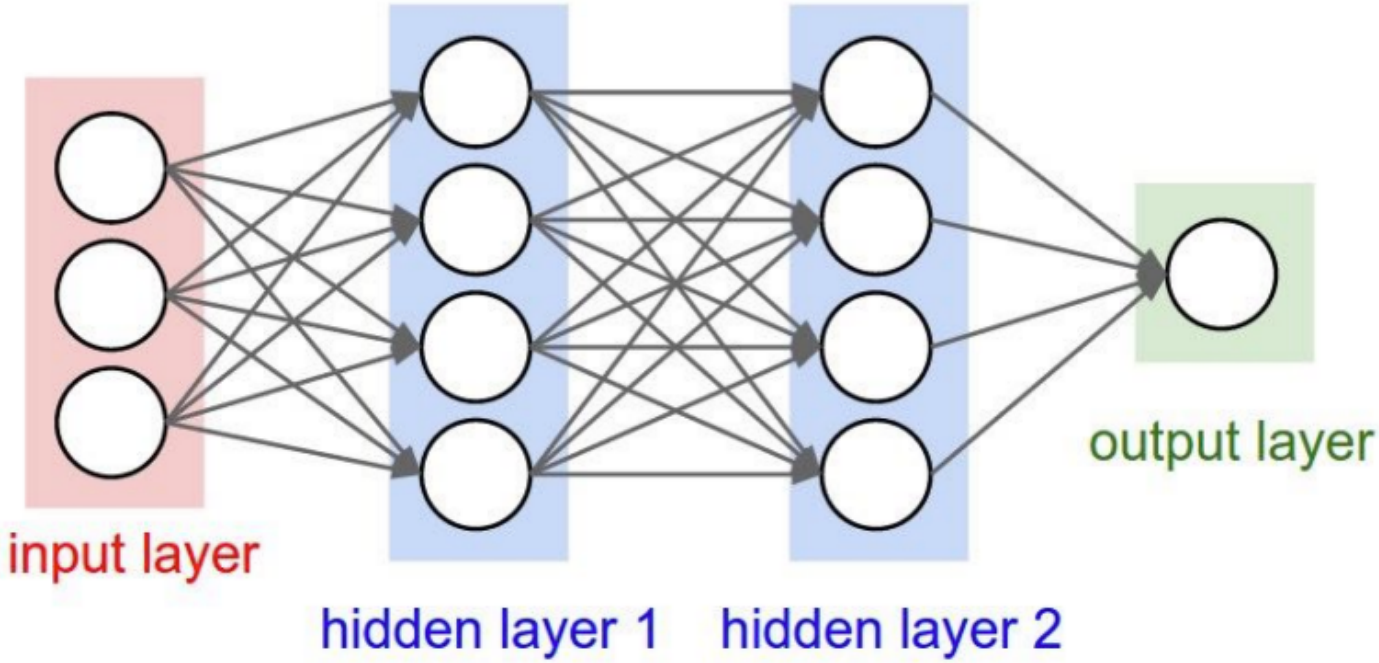


$$p = g_3(W_3 g_2(W_2 g_1(W_1 \vec{x_0})))$$

NETWORK FUNCTION

OPTIMIZATION

[OR HOW TO FIND THE WEIGHTS?]



$$p = g_3(W_3 g_2(W_2 g_1(W_1 \vec{x_0})))$$

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$$

LOSS FUNCTION

WE SIMPLY WANT TO MINIMIZE THE LOSS FUNCTION WITH RESPECT TO THE WEIGHTS, i.e. FIND THE WEIGHTS THAT GENERATE THE MINIMUM LOSS

WE SIMPLY WANT TO MINIMIZE THE LOSS FUNCTION WITH RESPECT TO THE WEIGHTS, i.e. FIND THE WEIGHTS THAT GENERATE THE MINIMUM LOSS

WE THEN USE STANDARD MINIMIZATION ALGORITHMS THAT YOU ALL KNOW…

FOR EXAMPLE….

Gradient Descent

$$W_{t+1} = W_t - \lambda_t \nabla f(W_t)$$

[gradient]

Newton

$$W_{t+1} = W_t - \lambda [Hf(W_t)]^{-1} \nabla f(W_t)$$

[hessian]

NEWTON CONVERGES FASTER…

FOR EXAMPLE….

Gradient Descent

$$W_{t+1} = W_t - \lambda_t \nabla f(W_t)$$

[gradient]

Newton

$$W_{t+1} = W_t - \lambda [Hf(W_t)]^{-1} \nabla f(W_t)$$

[hessian]

NEWTON CONVERGES FASTER…

BUT NEEDS THE HESSIAN

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial W_1^2} & \frac{\partial^2 f}{\partial W_1 \, \partial W_2} & \cdots & \frac{\partial^2 f}{\partial W_1 \, \partial W_n} \\ \frac{\partial^2 f}{\partial W_2 \, \partial W_1} & \frac{\partial^2 f}{\partial W_2^2} & \cdots & \frac{\partial^2 f}{\partial W_2 \, \partial W_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial W_n \, \partial W_1} & \frac{\partial^2 f}{\partial W_n \, \partial W_2} & \cdots & \frac{\partial^2 f}{\partial W_n^2} \end{bmatrix}$$

FOR EXAMPLE….

Gradient Descent

$$W_{t+1} = W_t - \lambda_t \nabla f(W_t)$$

[gradient]

Newton

$$W_{t+1} = W_t - \lambda [Hf(W_t)]^{-1} \nabla f(W_t)$$

[hessian]

NEWTON CONVERGES FASTER…

BUT NEEDS THE HESSIAN

MOST USED BY FAR….

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial W_1^2} & \frac{\partial^2 f}{\partial W_1 \partial W_2} & \cdots & \frac{\partial^2 f}{\partial W_1 \partial W_n} \\ \frac{\partial^2 f}{\partial W_2 \partial W_1} & \frac{\partial^2 f}{\partial W_2^2} & \cdots & \frac{\partial^2 f}{\partial W_2 \partial W_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial W_n \partial W_1} & \frac{\partial^2 f}{\partial W_n \partial W_2} & \cdots & \frac{\partial^2 f}{\partial W_n^2} \end{bmatrix}$$

FOR EXAMPLE….

Gradient Descent

$$W_{t+1} = W_t - \lambda_t \nabla f(W_t)$$

[gradient]

EVERYTHING RELIES
ON COMPUTING THE GRADIENT

MOST USED BY FAR….

Newton

$$W_{t+1} = W_t - \lambda [Hf(W_t)]^{-1} \nabla f(W_t)$$

[hessian]

NEWTON CONVERGES FASTER…

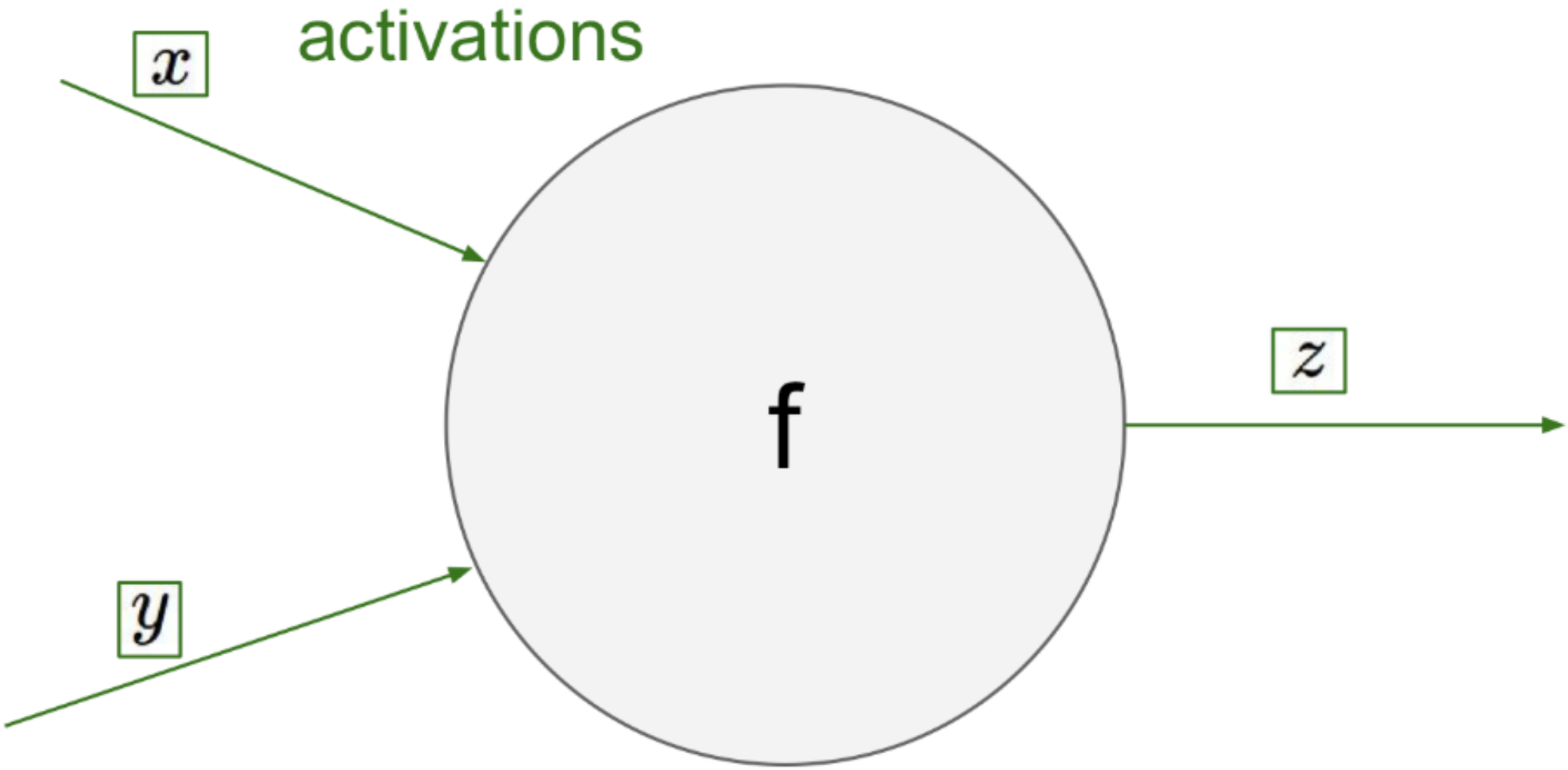
BUT NEEDS THE HESSIAN

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial W_1^2} & \frac{\partial^2 f}{\partial W_1 \partial W_2} & \cdots & \frac{\partial^2 f}{\partial W_1 \partial W_n} \\ \frac{\partial^2 f}{\partial W_2 \partial W_1} & \frac{\partial^2 f}{\partial W_2^2} & \cdots & \frac{\partial^2 f}{\partial W_2 \partial W_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial W_n \partial W_1} & \frac{\partial^2 f}{\partial W_n \partial W_2} & \cdots & \frac{\partial^2 f}{\partial W_n^2} \end{bmatrix}$$

NICE, BUT I NEED TO COMPUTE TE GRADIENT AT EVERY ITERATION OF AN ARBITRARY COMPLEX FUNCTION!

BACKPROPAGATION

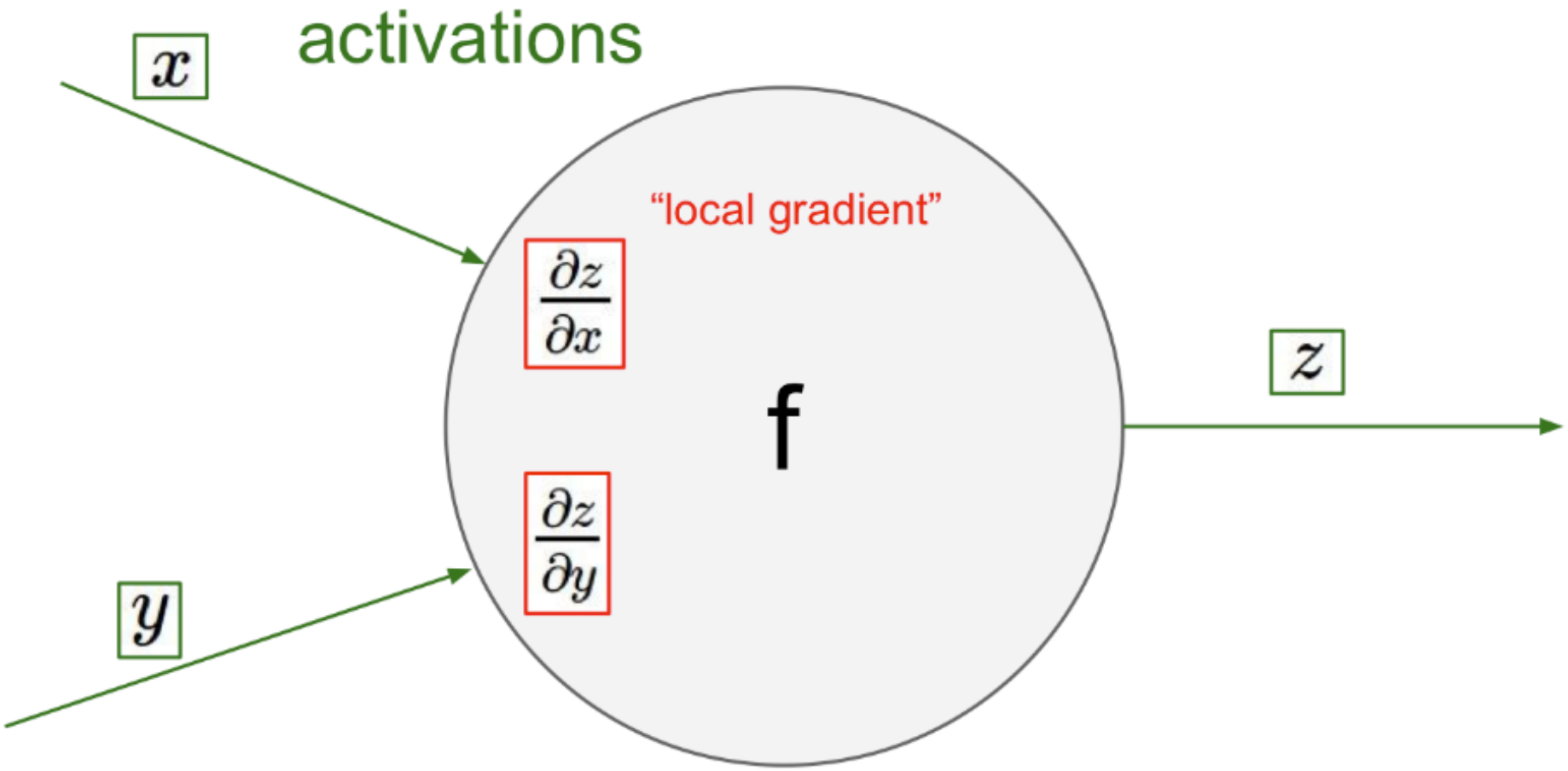
[AT THE NEURON LEVEL]



Credit: A. Karpathy

BACKPROPAGATION

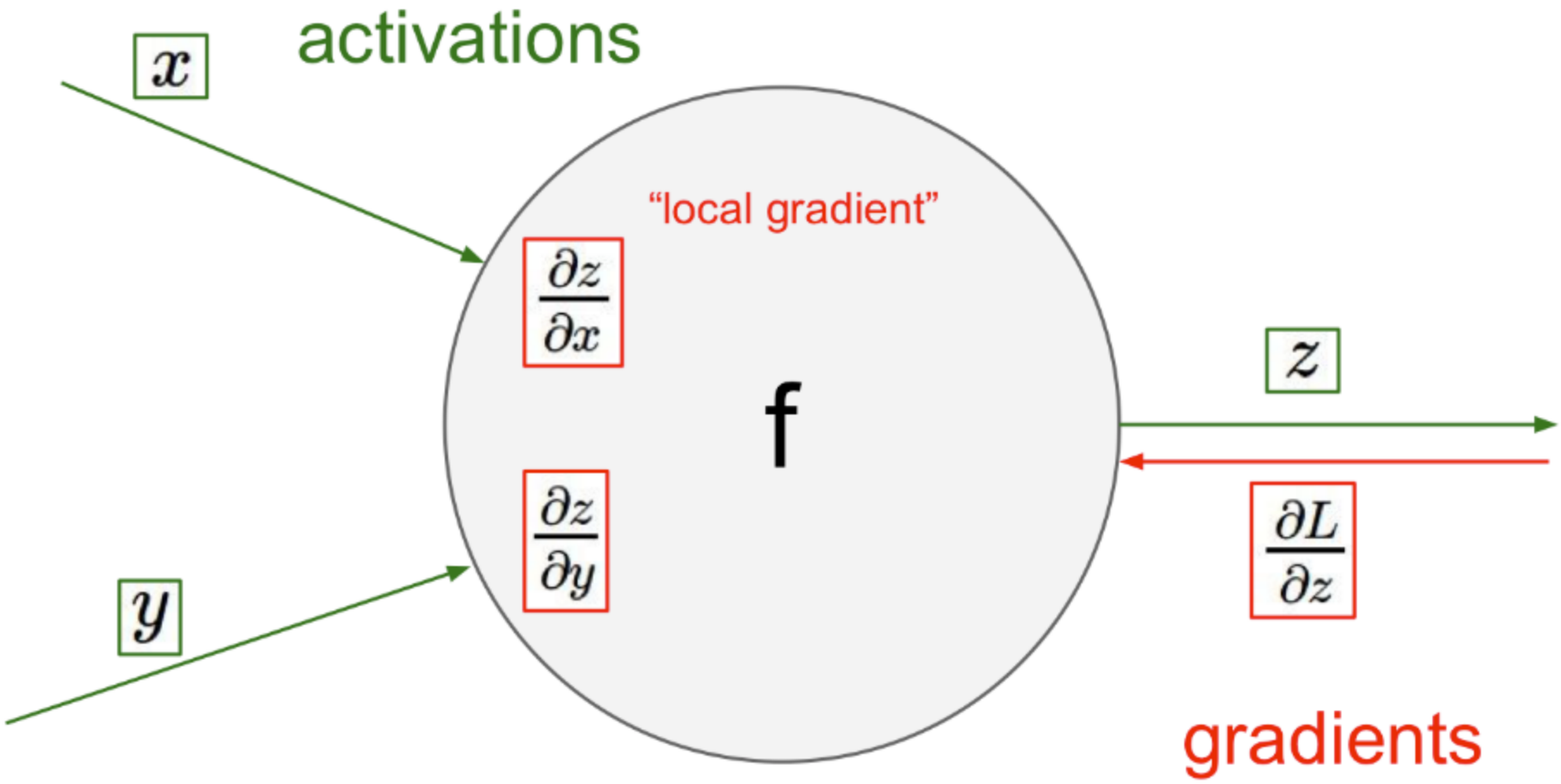
[AT THE NEURON LEVEL]



Credit: A. Karpathy

BACKPROPAGATION

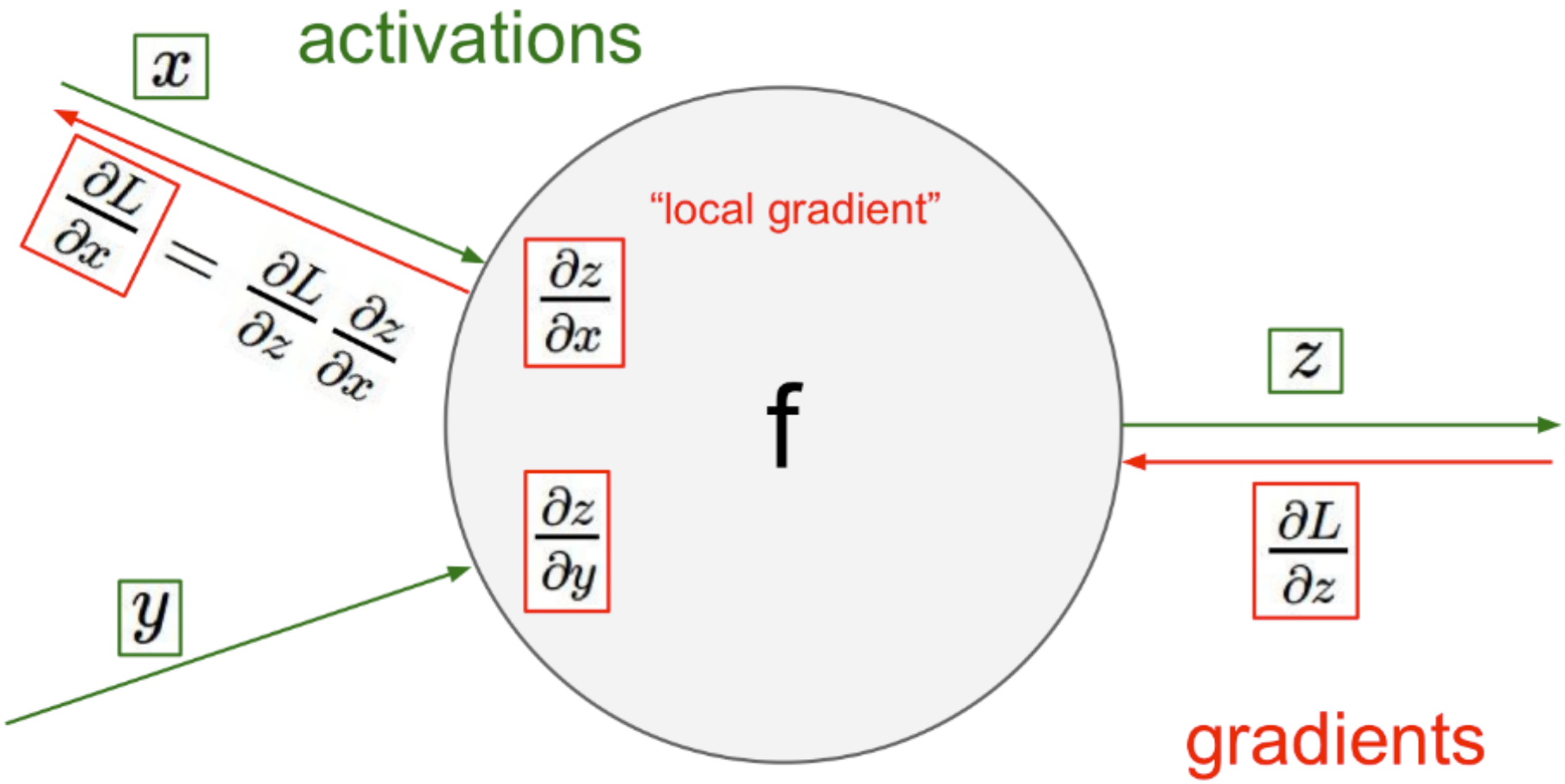
[AT THE NEURON LEVEL]



Credit: A. Karpathy

BACKPROPAGATION

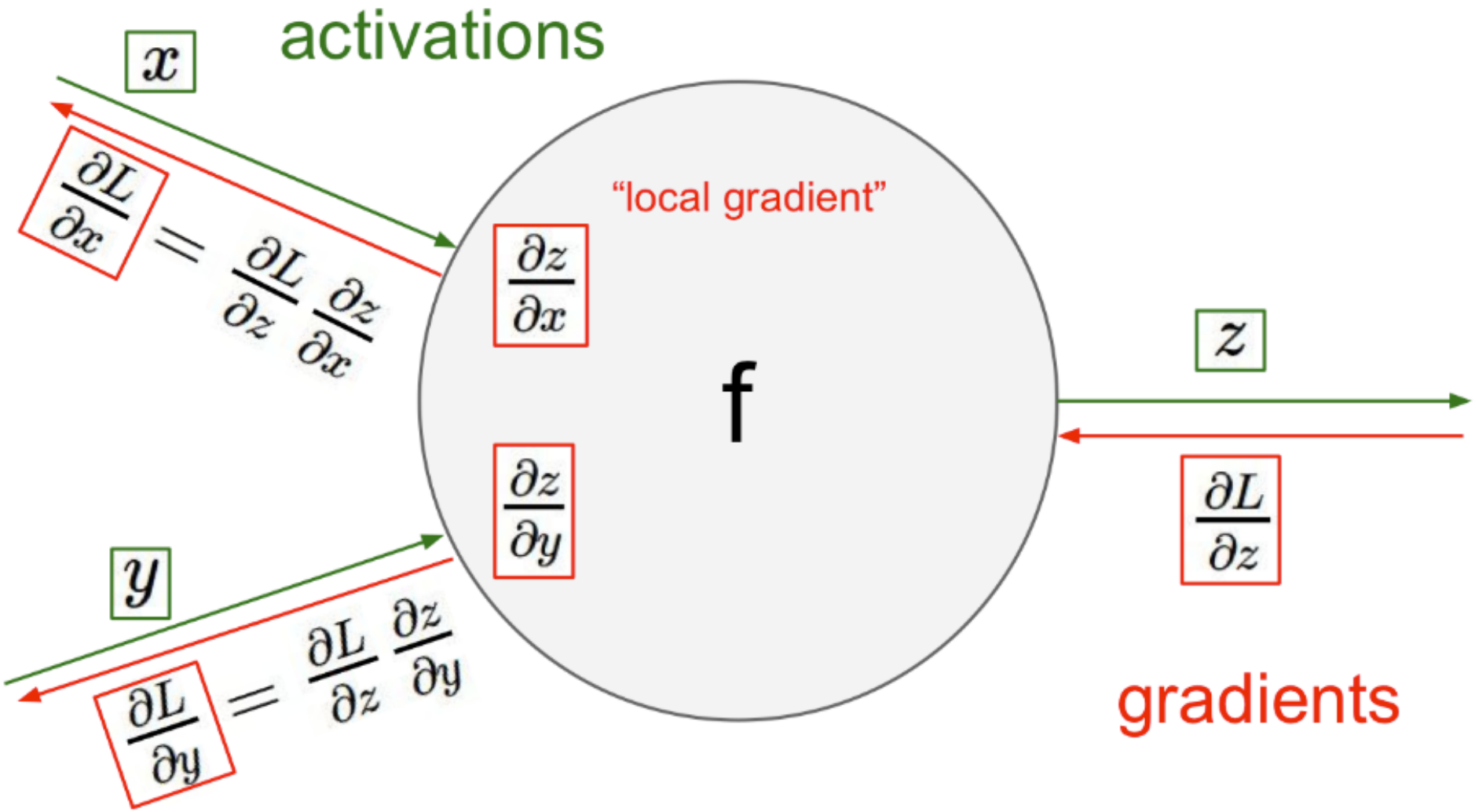
[AT THE NEURON LEVEL]



Credit: A. Karpathy

BACKPROPAGATION

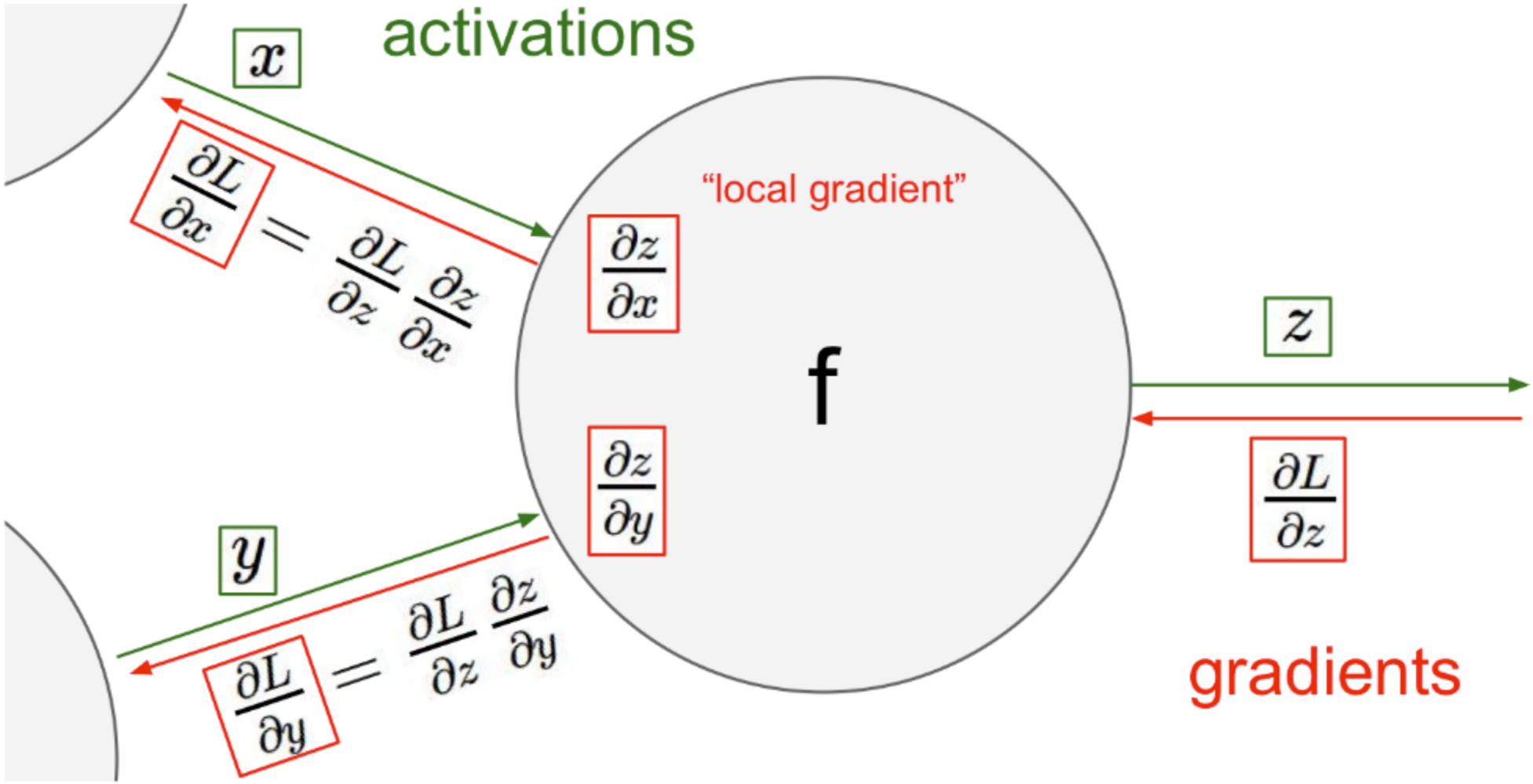
[AT THE NEURON LEVEL]



Credit: A. Karpathy

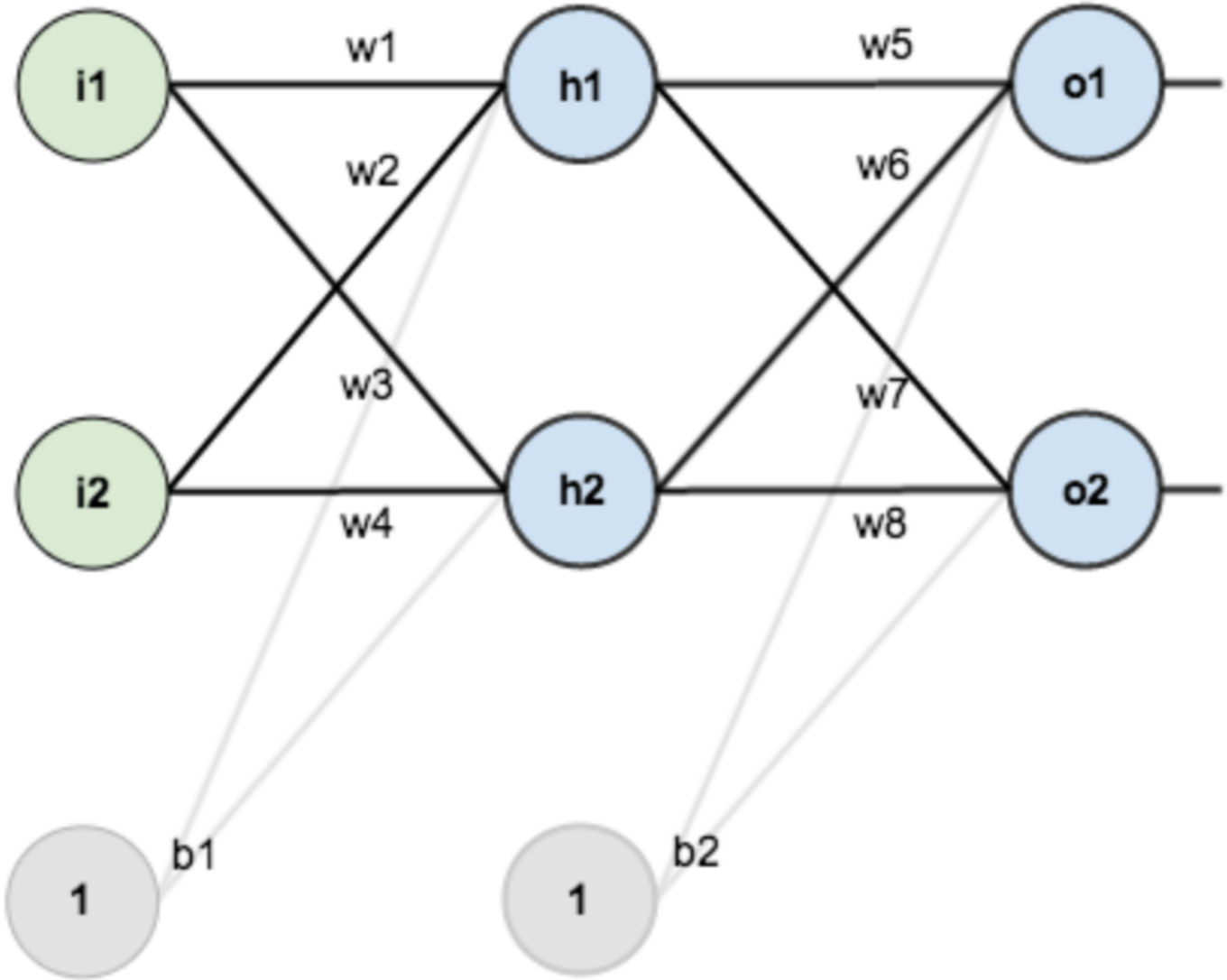
BACKPROPAGATION

[AT THE NEURON LEVEL]

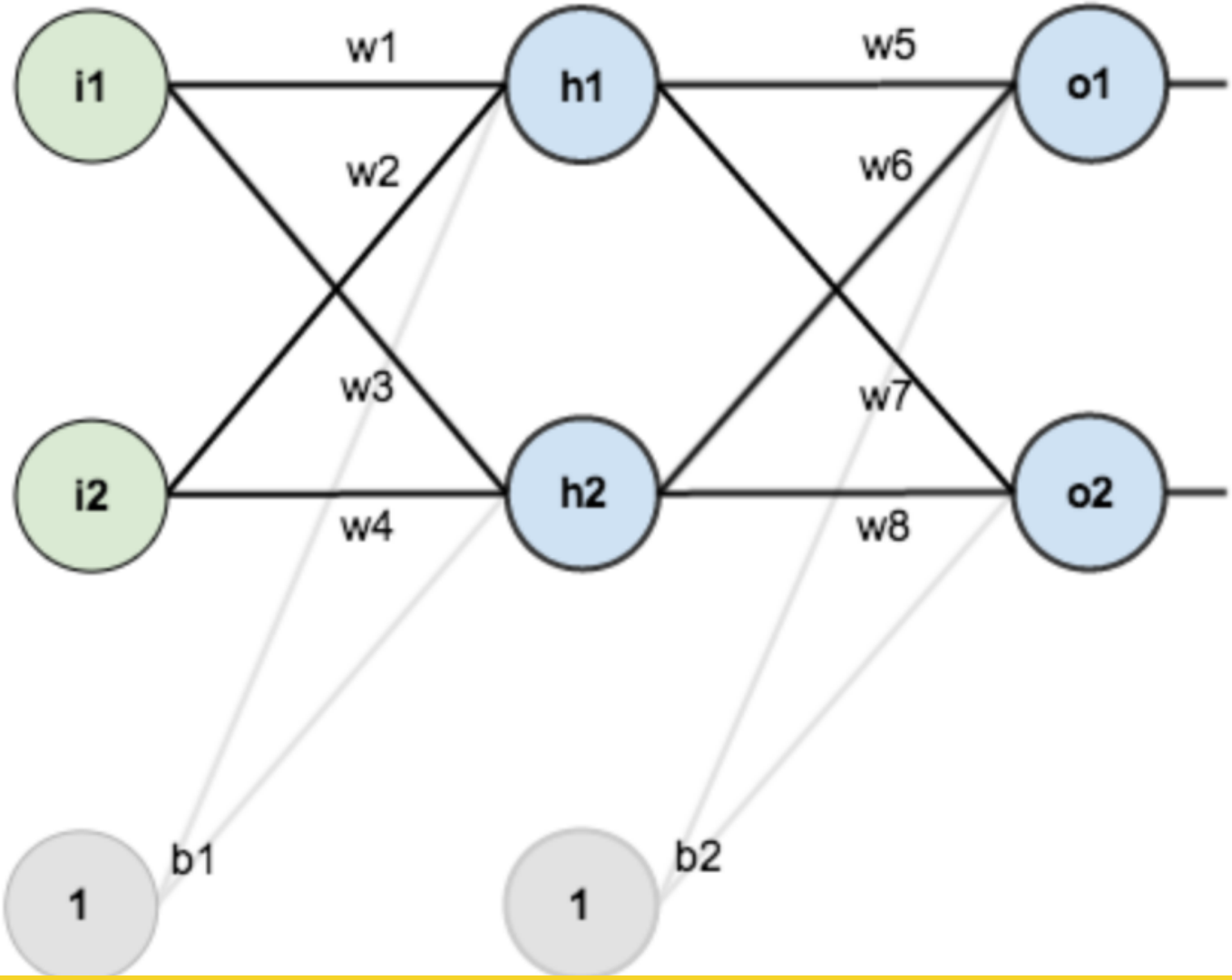


Credit: A. Karpathy

LET’S FOLLOW A NETWORK WHILE IT LEARNS…

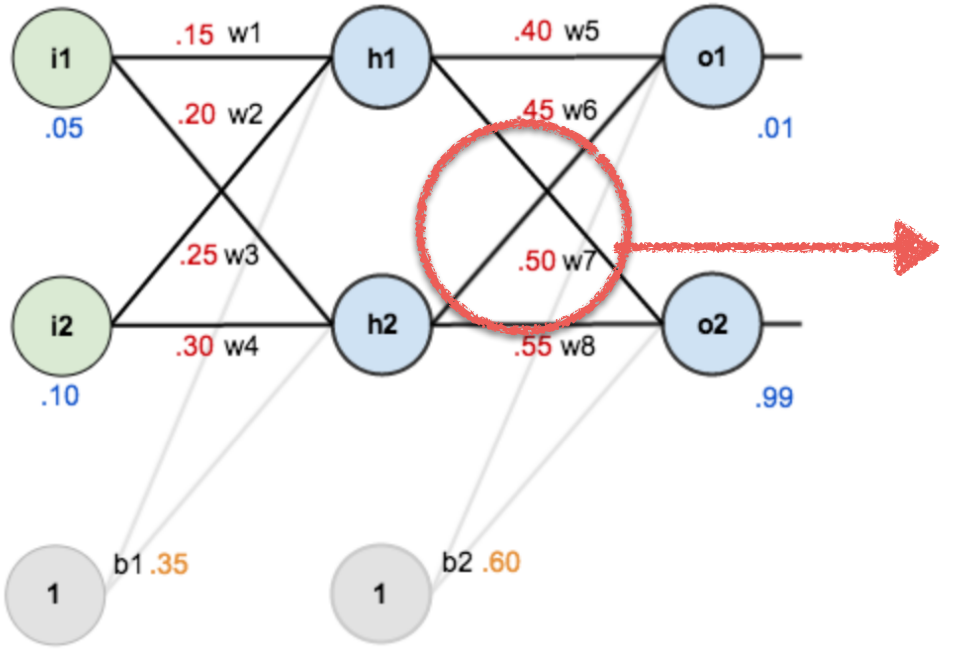


EXAMPLE TAKEN FROM HERE



LET'S ASSUME A VERY SIMPLE TRAINING SET:
X=(0.05, 0.10) —> Y=(0.01,0.99)

EXAMPLE TAKEN FROM HERE



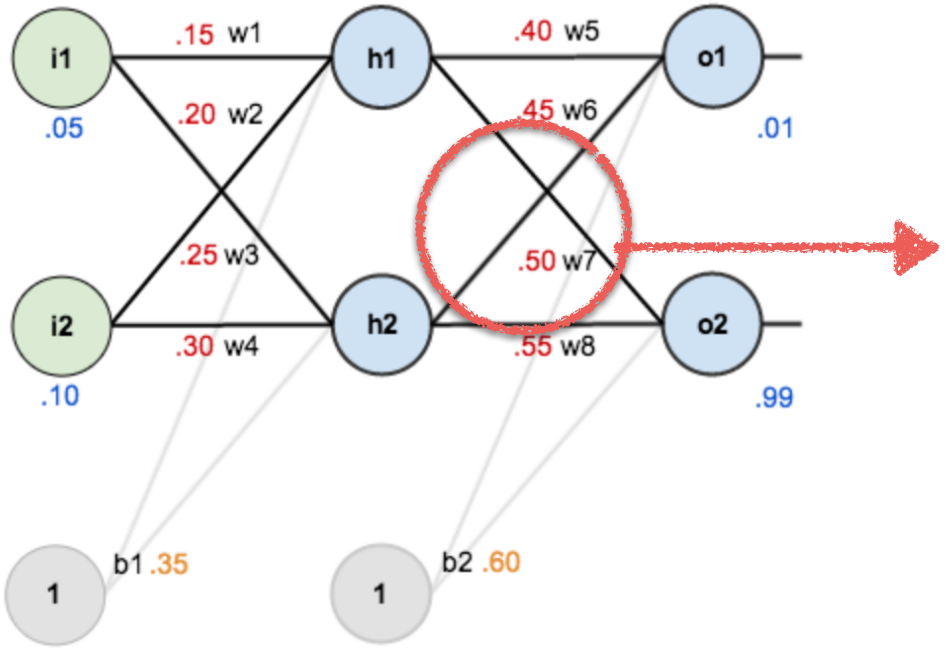
initial weights

1. THE FORWARD PASS

$$in_{h1} = w_1 i_1 + w_2 i_2 + b_1$$

$$in_{h1} = 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35 = 0.3775$$

[with initial weights]



initial weights

1. THE FORWARD PASS

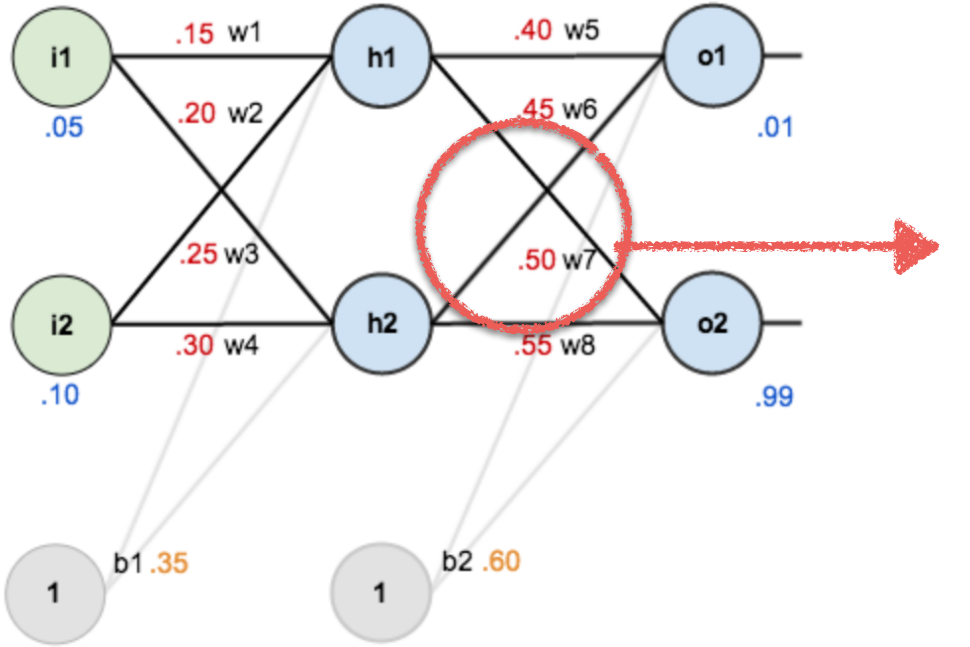
$$in_{h1} = w_1 i_1 + w_2 i_2 + b_1$$

$$in_{h1} = 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35 = 0.3775$$

[with initial weights]

$$out_{h1} = \frac{1}{1 + e^{-in_{h1}}} = 0.5932$$

[after the activation function]



initial weights

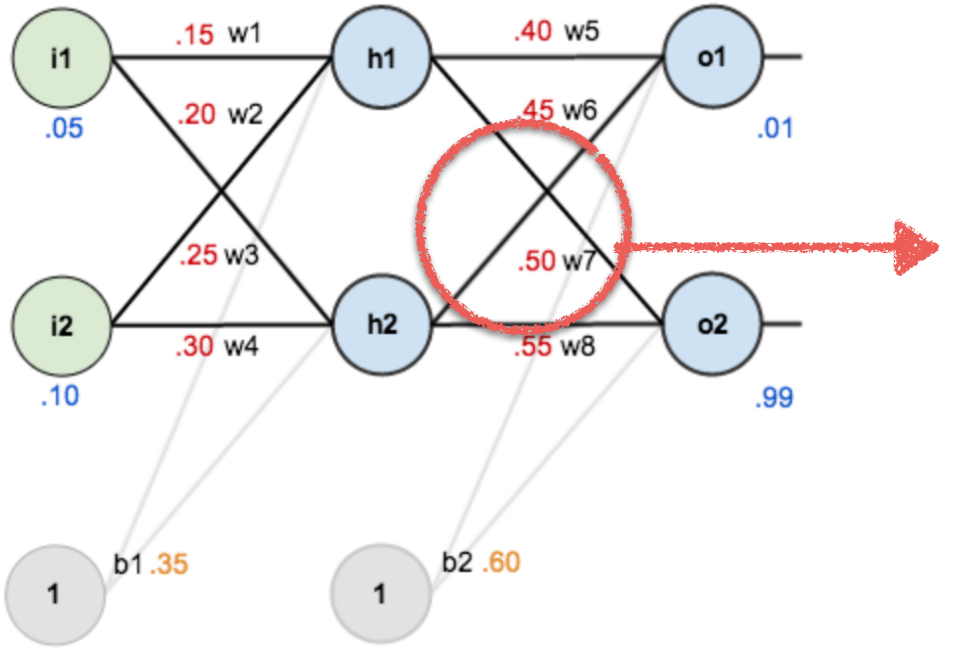
1. THE FORWARD PASS

WE CONTINUE TO o1

$$in_{o1} = w_5 out_{h1} + w_6 out_{h2} + b_2$$

$$in_{o1} = 0.4 \times 0.593 + 0.45 \times 0.596 + 0.6 = 1.105$$

$$out_{o1} = \frac{1}{1 + e^{-1.105}} = 0.751$$

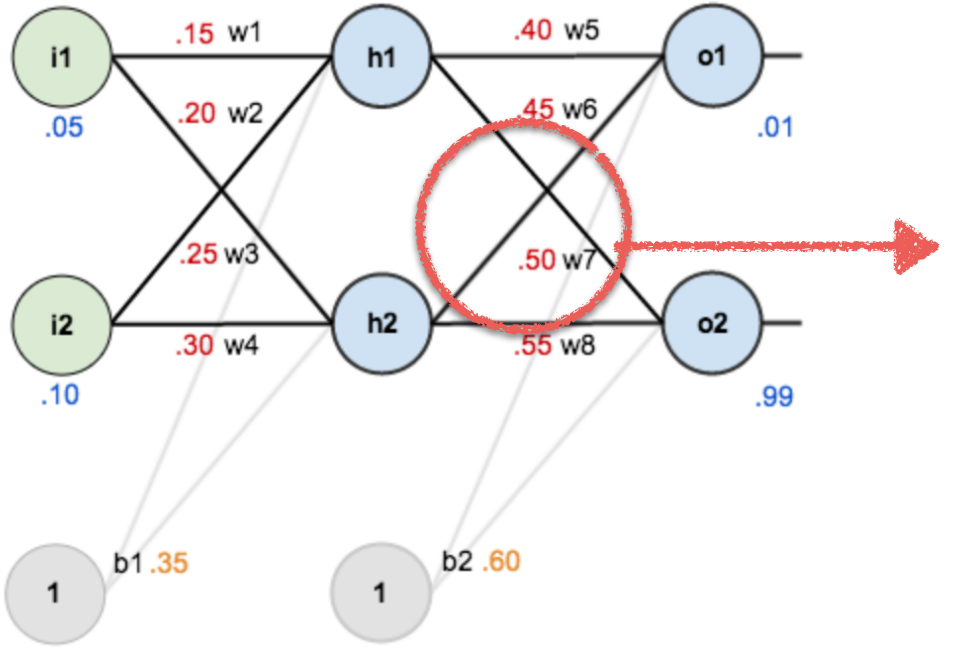


initial weights

1. THE FORWARD PASS

AND THE SAME FOR o2

$$out_{o2} = 0.7729$$



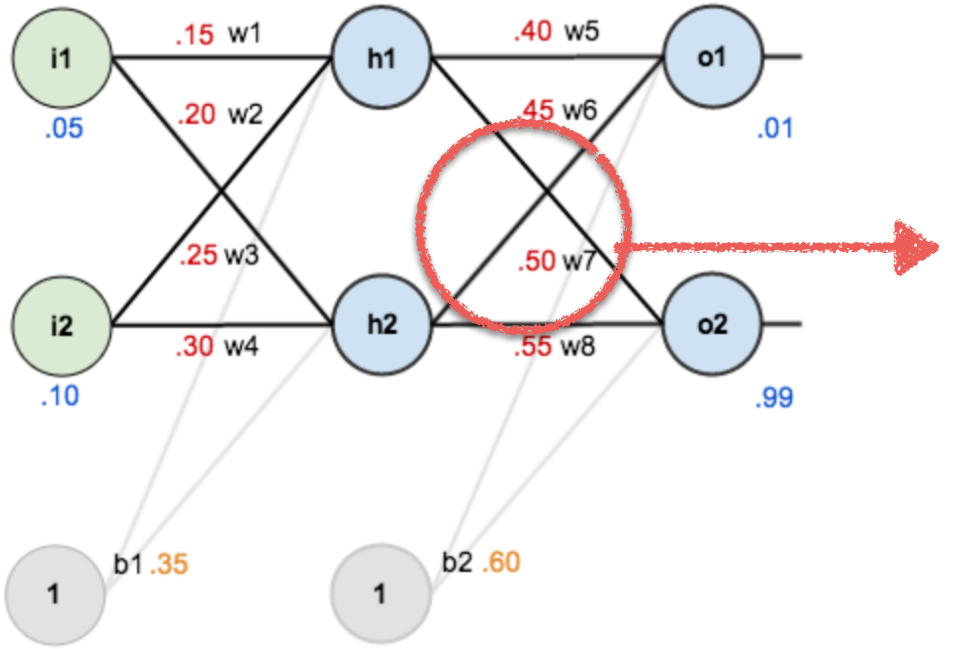
initial weights

2. THE LOSS FUNCTION

$$L_{total} = \sum 0.5(target - output)^2$$

$$L_{o1} = 0.5(target_{o1} - output_{01})^2 = 0.5 \times (0.01 - 0.751)^2 = 0.274$$

$$L_{o2} = 0.023$$



initial weights

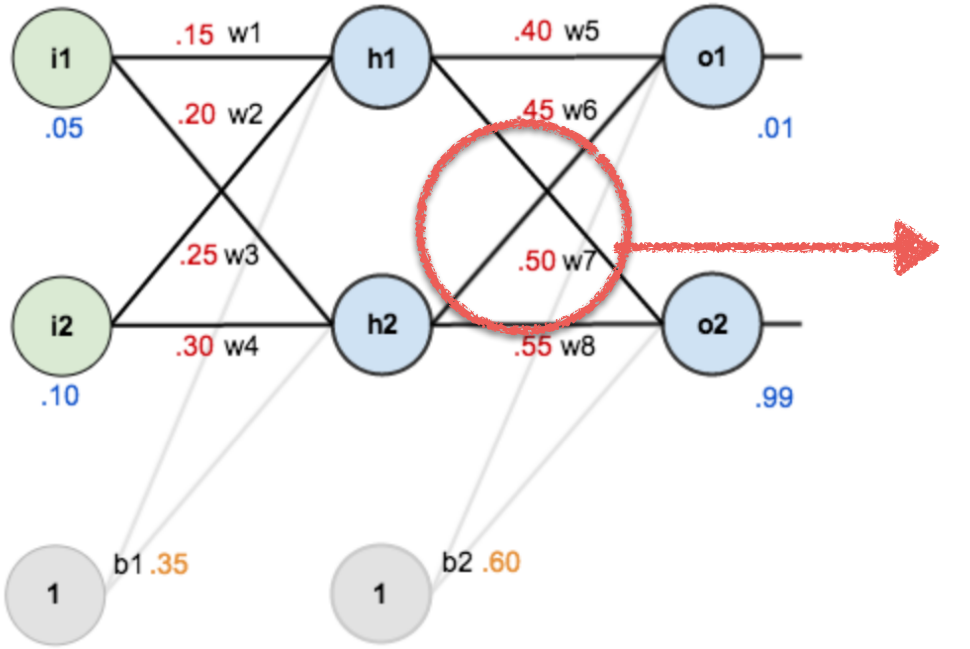
2. THE LOSS FUNCTION

$$L_{total} = \sum 0.5(target - output)^2$$

$$L_{o1} = 0.5(target_{o1} - output_{01})^2 = 0.5 \times (0.01 - 0.751)^2 = 0.274$$

$$L_{o2} = 0.023$$

$$L_{total} = L_{o1} + L_{o2} = 0.298$$



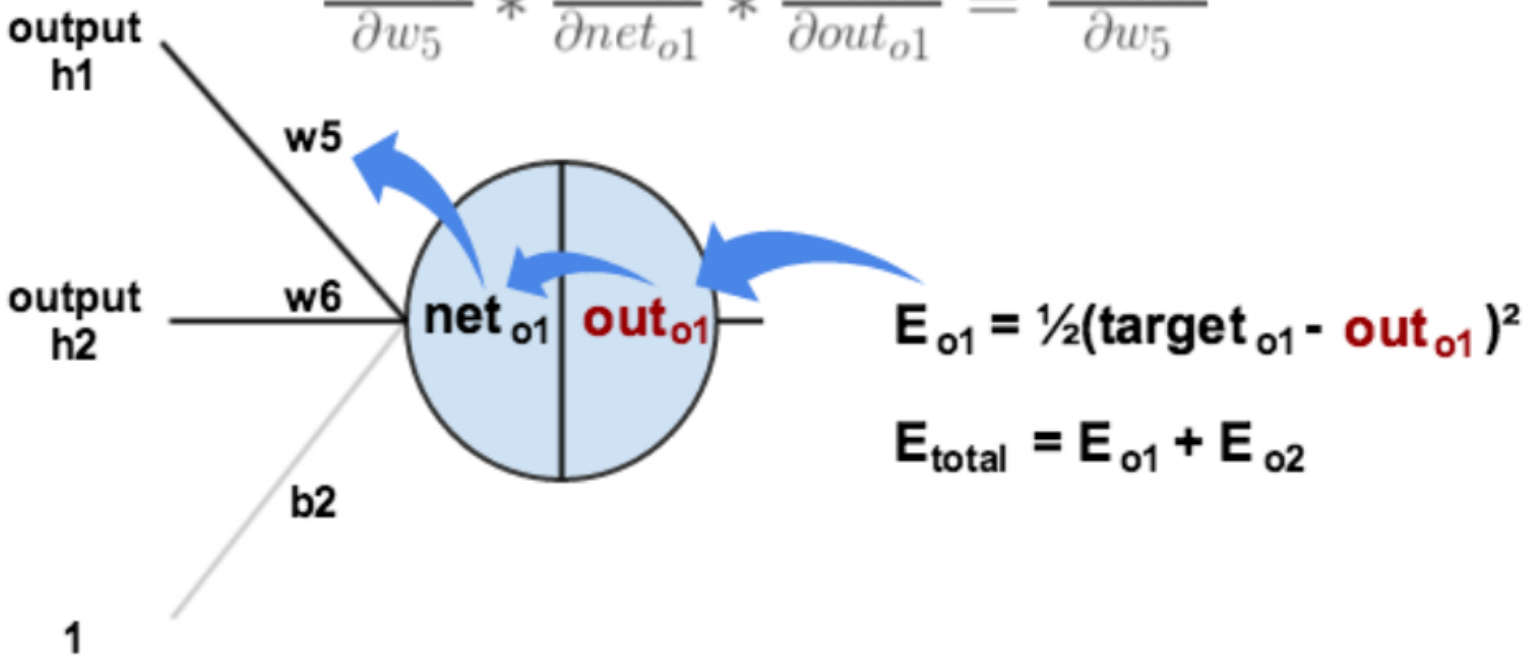
initial weights

3. THE BACKWARD PASS

FOR W5

WE WANT: $\frac{\partial L_{total}}{\partial w_5}$ [gradient of loss function]

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



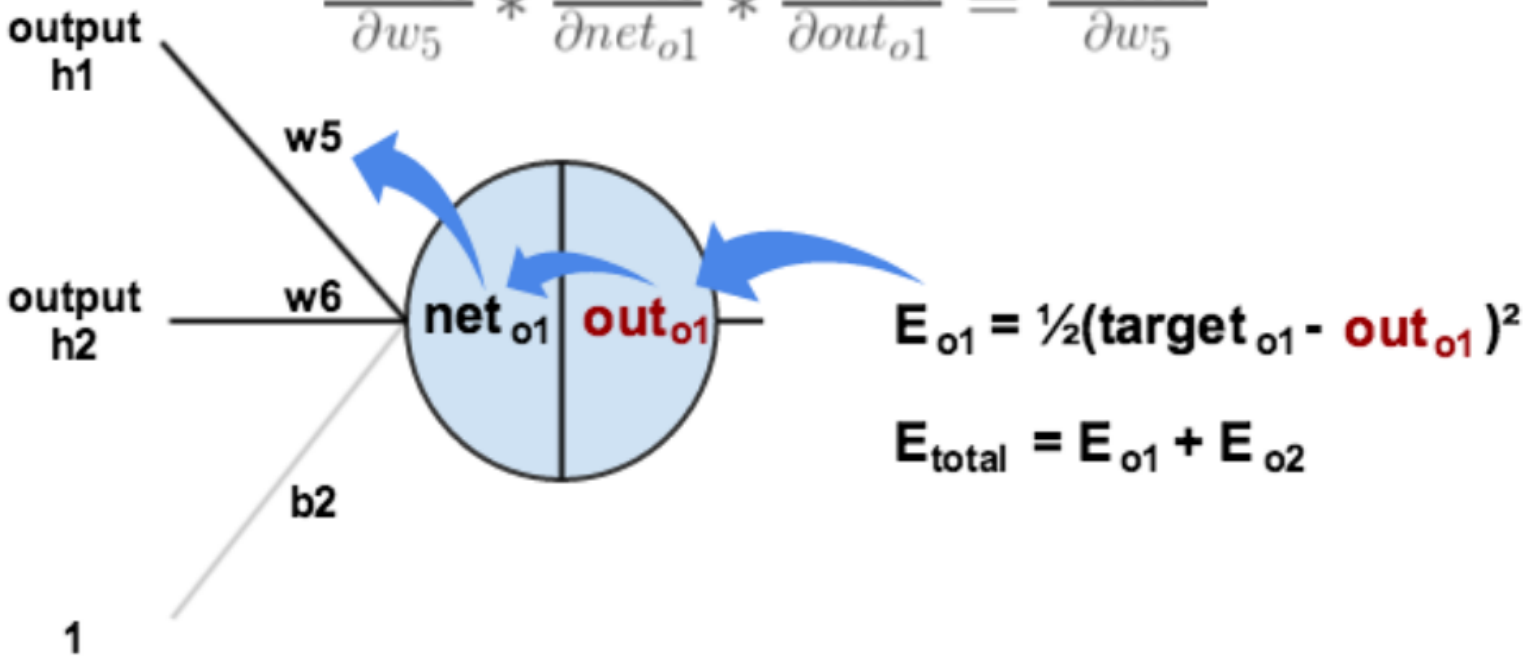
3. THE BACKWARD PASS

FOR W5 WE WANT: $\frac{\partial L_{total}}{\partial w_5}$ [gradient of loss function]

WE APPLY THE CHAIN RULE:

$$\frac{\partial L_{total}}{\partial w_5} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



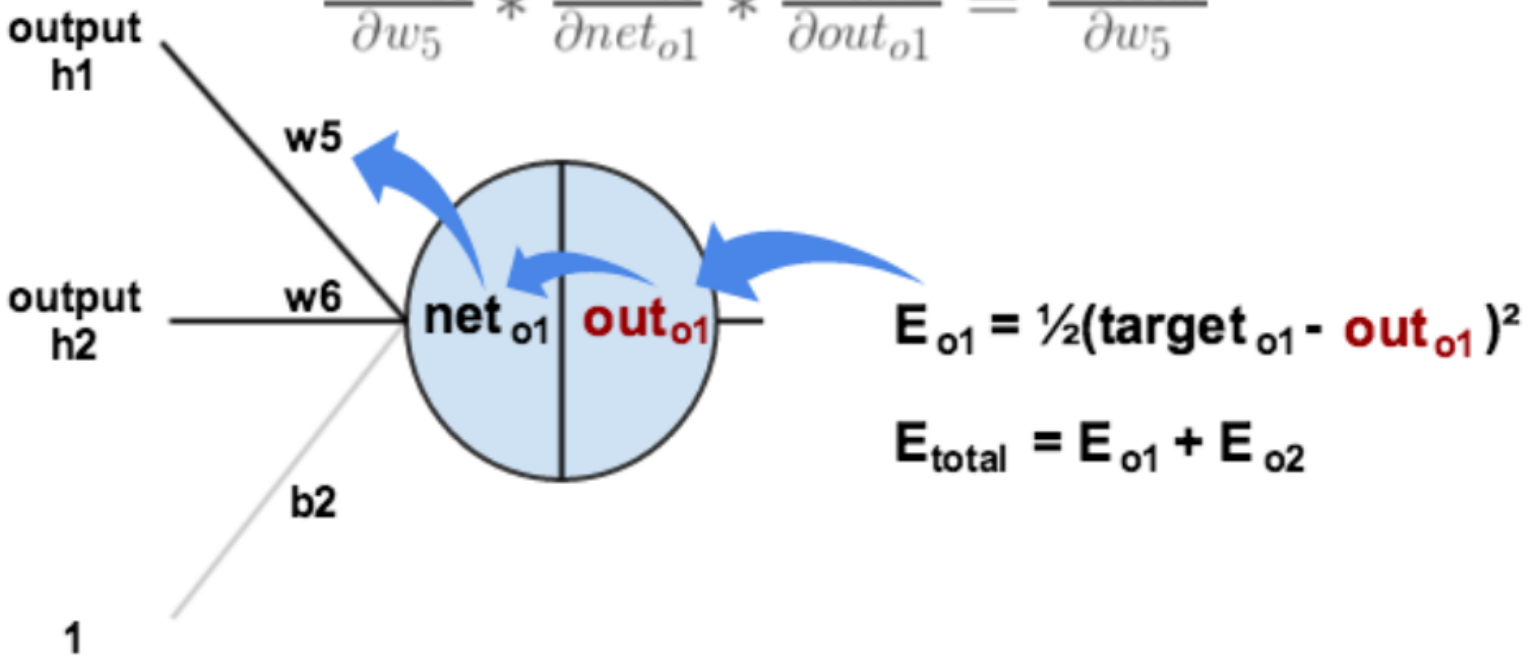
3. THE BACKWARD PASS

$$\frac{\partial L_{total}}{\partial w_5} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

$$L_{total} = 0.5(target_{o1} - out_{o1})^2 + 0.5(target_{o2} - out_{o2})^2$$

$$\frac{\partial L_{total}}{\partial out_{o1}} = 2 \times 0.5(target_{o1} - out_{o1}) \times (-1) = 0.741$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



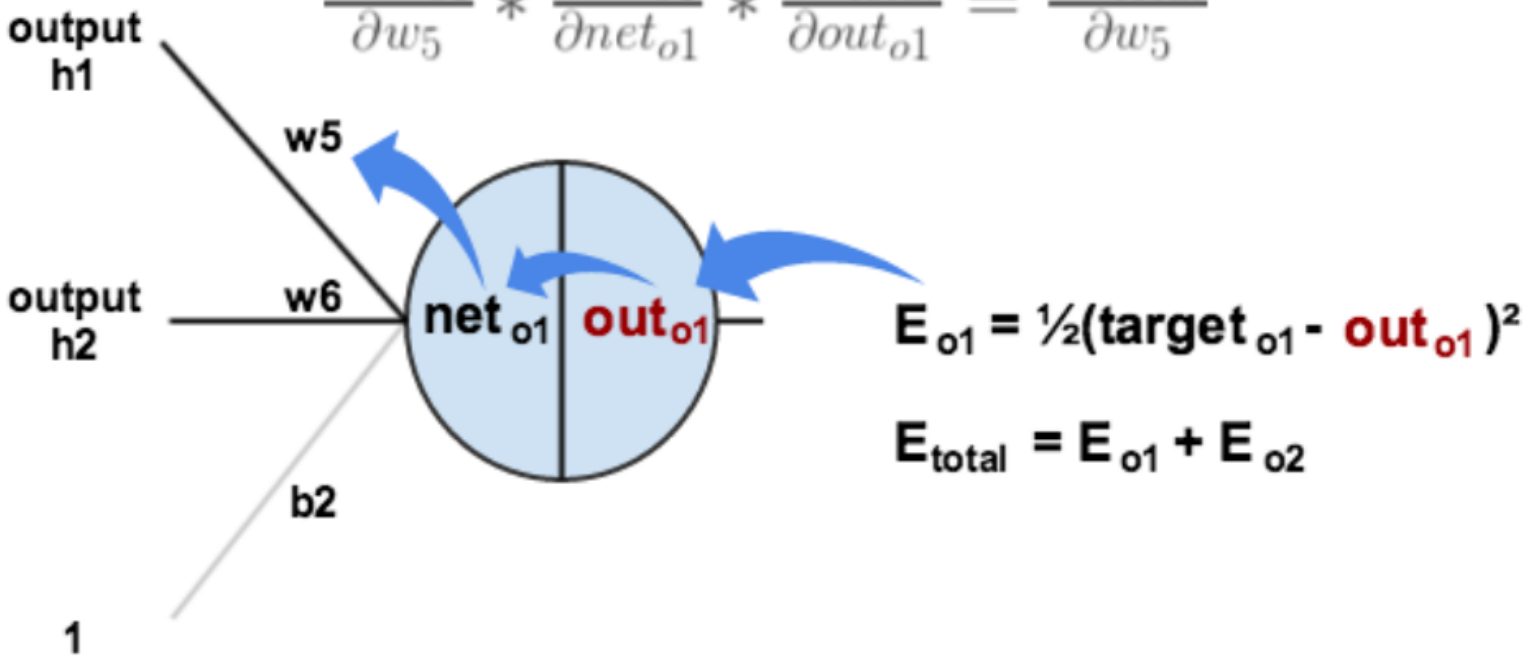
3. THE BACKWARD PASS

$$\frac{\partial L_{total}}{\partial w_5} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

$$out_{o1} = \frac{1}{1 + e^{-in_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial in_{o1}} = out_{o1} \times (1 - out_{o1}) = 0.186$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



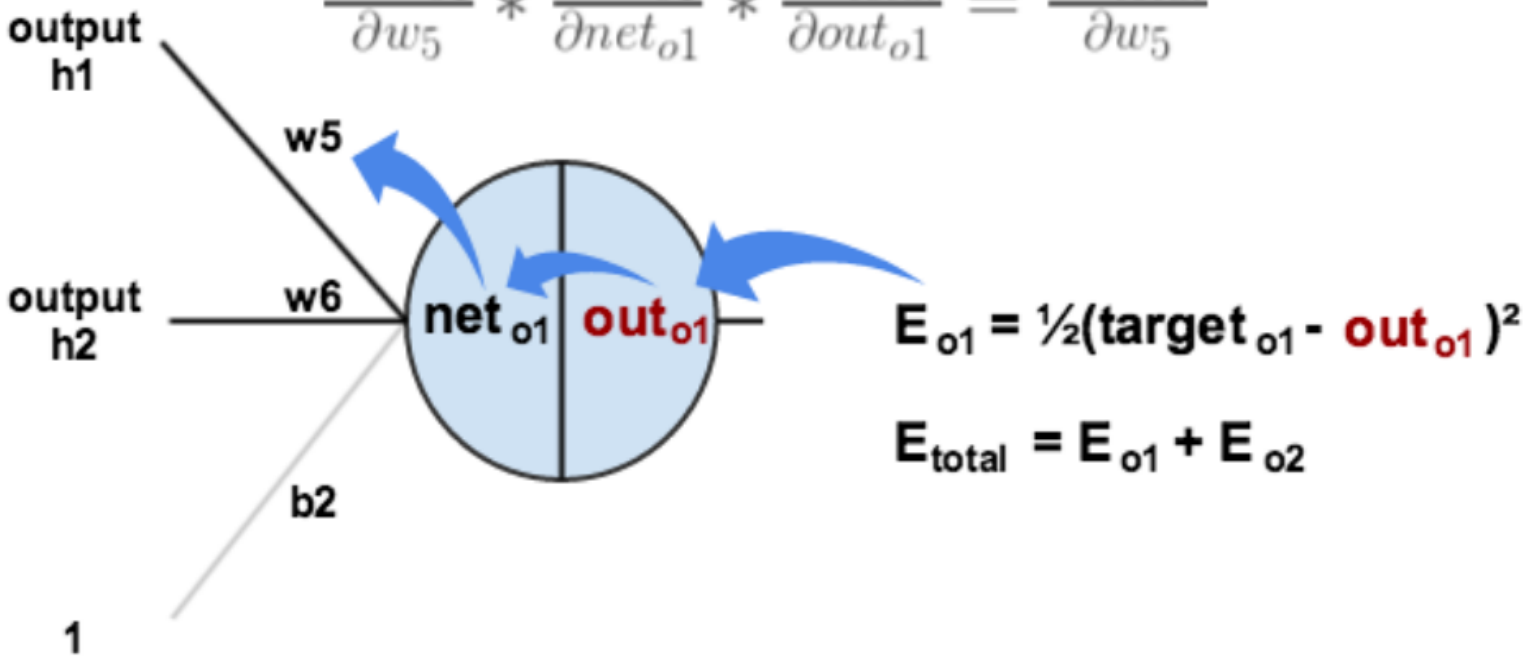
3. THE BACKWARD PASS

$$\frac{\partial L_{total}}{\partial w_5} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

$$in_{o1} = w_5 \times out_{h1} + w_6 \times out_{h2} + b_2$$

$$\frac{\partial in_{o1}}{\partial w_5} = out_{h1} \times w_5^{1-1} = out_{h1} = 0.593$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



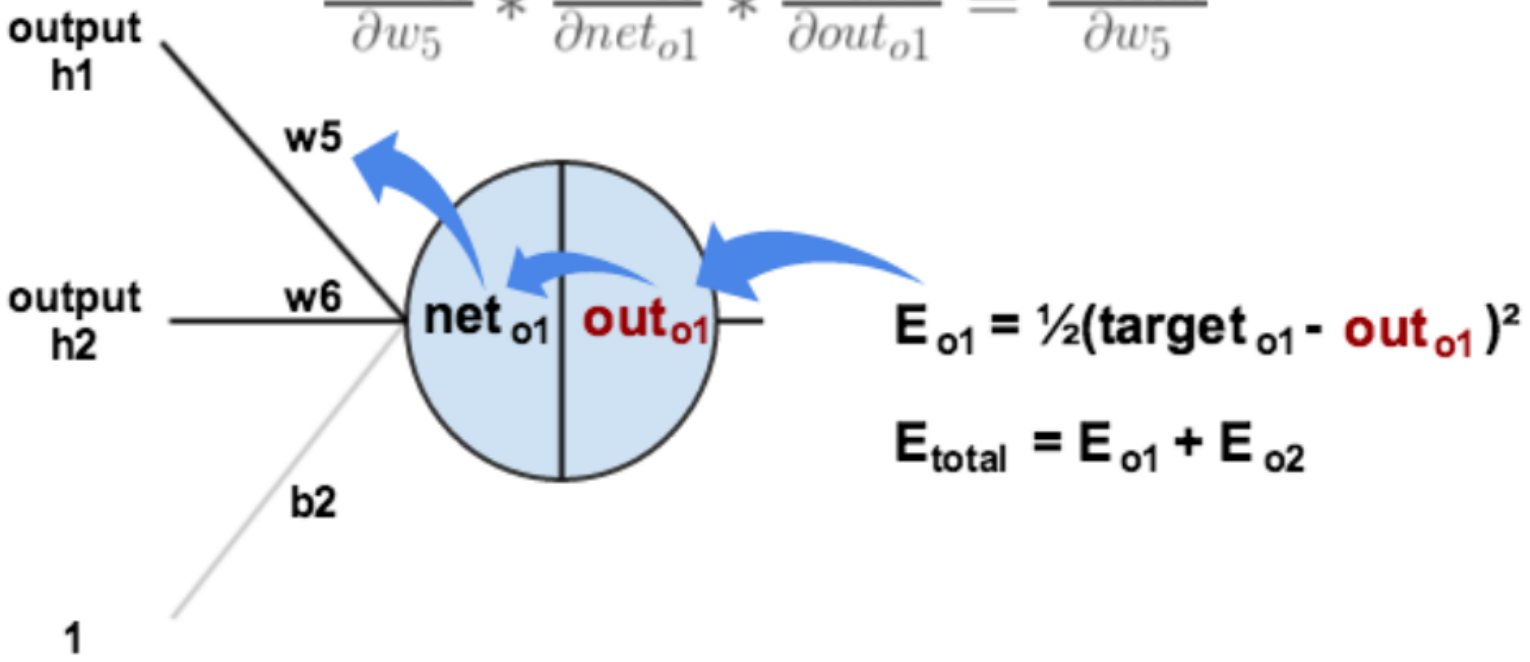
3. THE BACKWARD PASS

ALL TOGETHER:

$$\frac{\partial L_{total}}{\partial w_5} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

$$\frac{\partial L_{total}}{\partial w_5} = 0.741 \times 0.186 \times 0.593 = 0.082$$

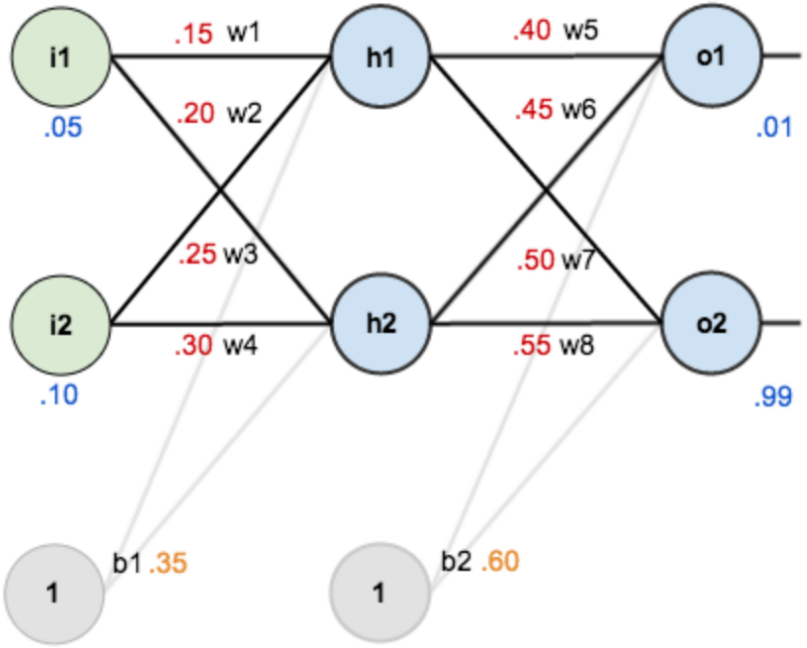
$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



4. UPDATE WEIGHTS WITH GRADIENT AND LEARNING RATE

$$w_5^{t+1} = w_5 - \lambda \times \frac{\partial L_{total}}{\partial w_5}$$

$$w_5^{t+1} = 0.4 - 0.5 \times 0.082 = 0.358$$

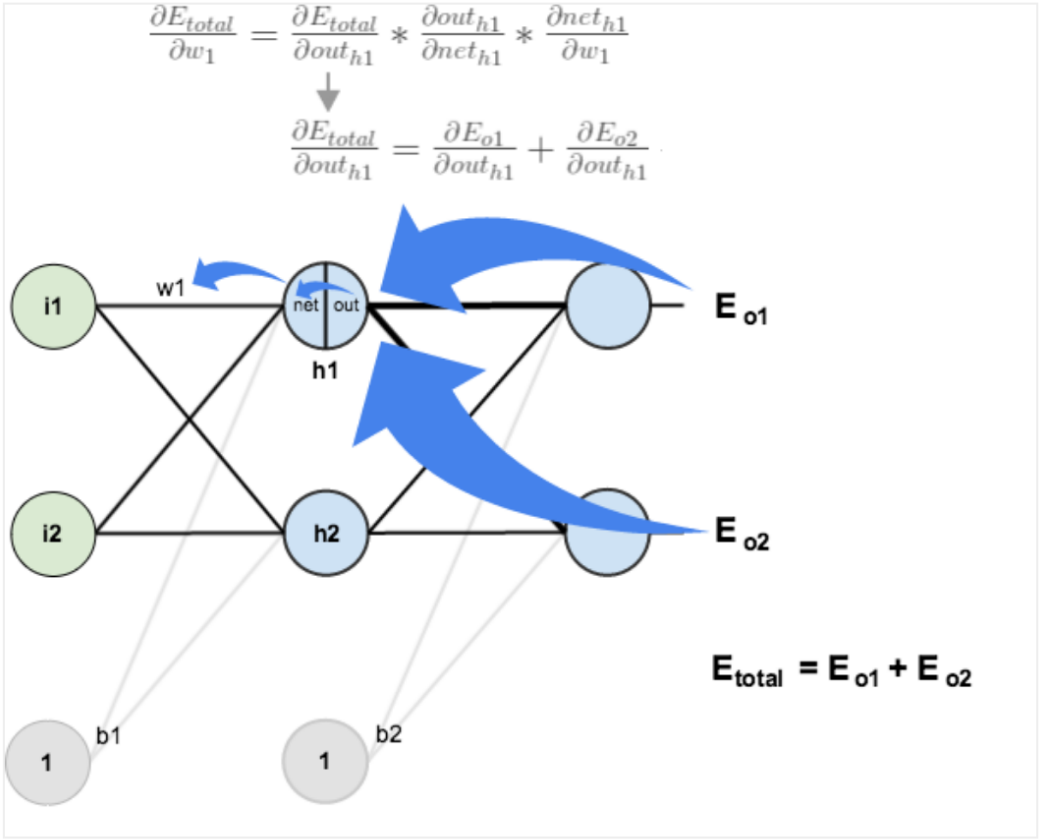


THIS IS REPEATED FOR THE OTHER WEIGHTS OF THE OUTPUT LAYER

$$w_6^{t+1} = 0.408$$

$$w_7^{t+1} = 0.511$$

$$w_8^{t+1} = 0.561$$



AND BACK-PROPAGATED TO THE HIDDEN LAYERS

VISUALIZE SIMPLE NETWORK LEARNING