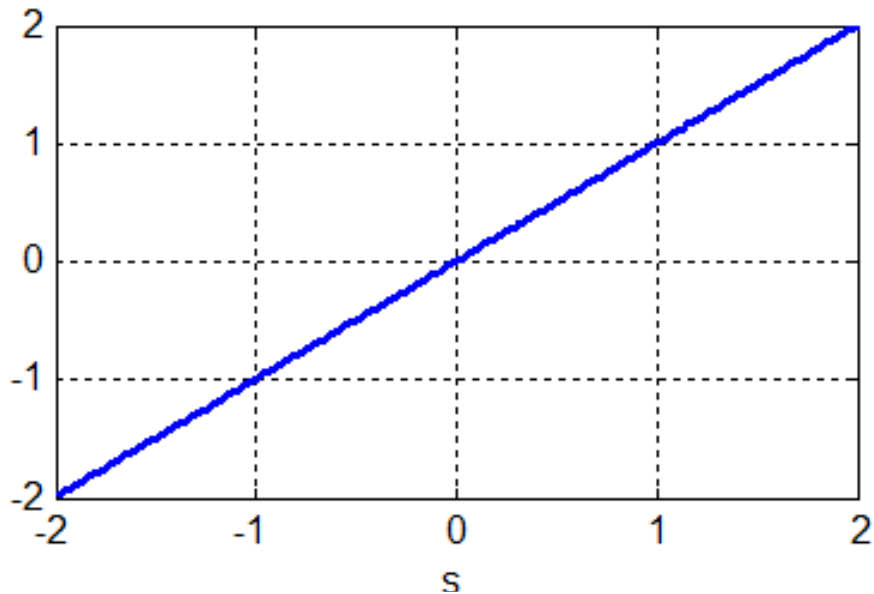


# Activation (Transfer) Functions

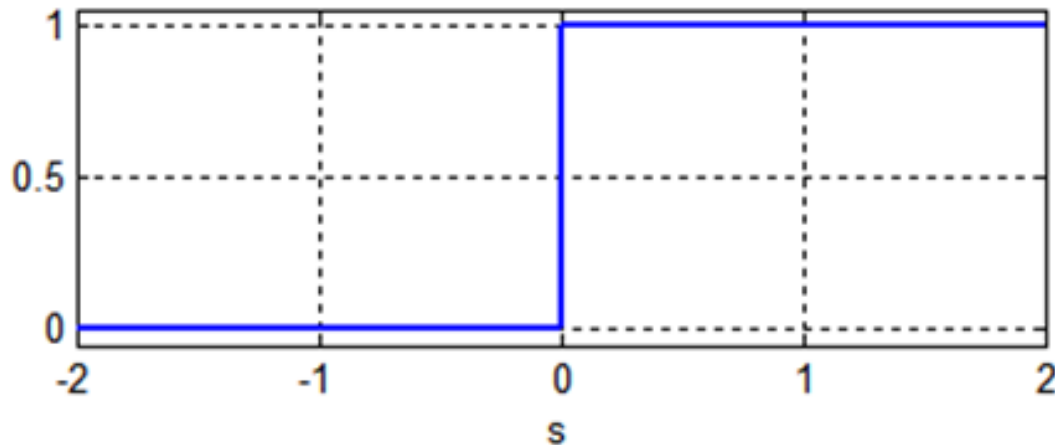




$$f(s) = s$$

Only in the output layer for function fitting problems

*purelin*

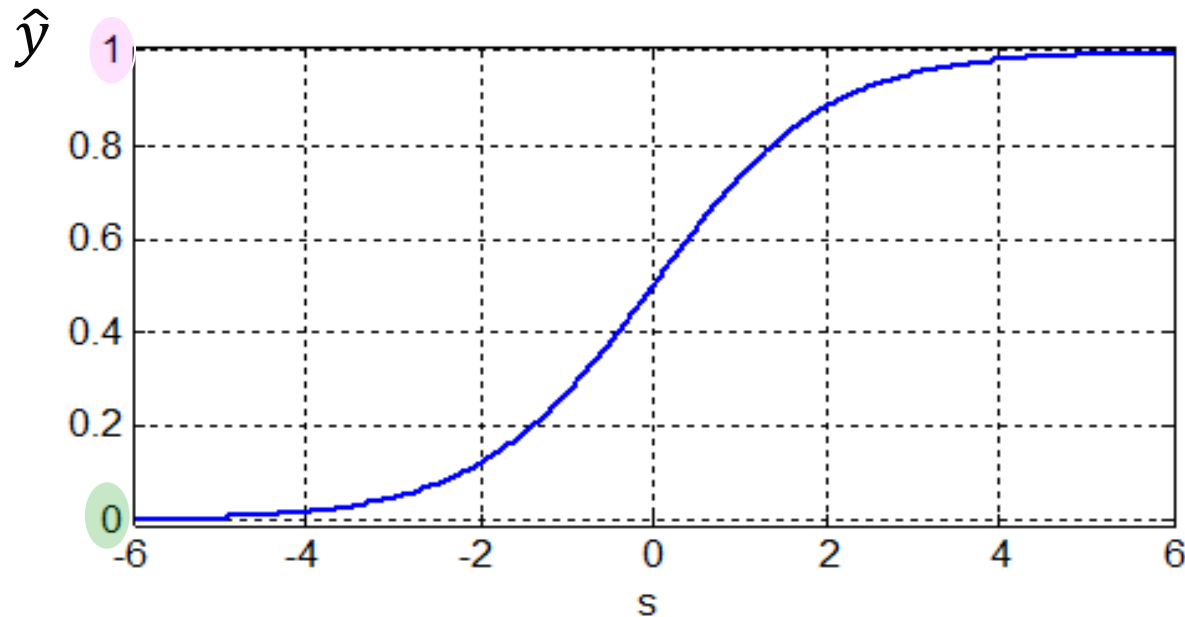


$$f(s) = \begin{cases} 0, & s < 0 \\ 1, & s \geq 0 \end{cases}$$

*hardlim*

*hardlims*  $\hat{y}(s) = \begin{cases} -1, & s < 0 \\ 1, & s \geq 0 \end{cases}$

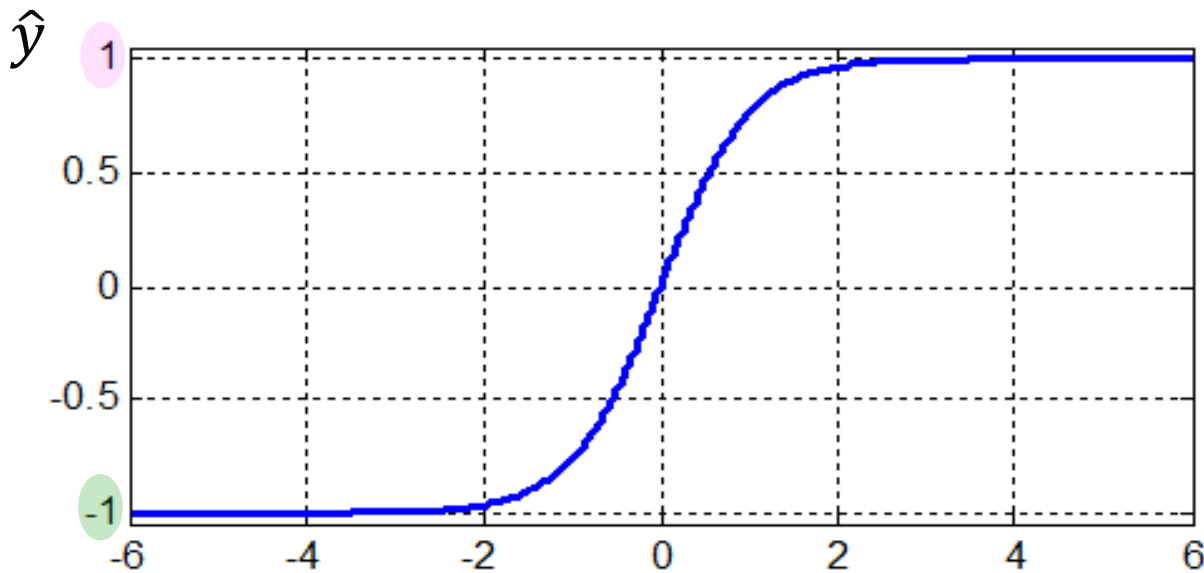




*Only in the output layer  
for binary classification  
problems*

$$\hat{y} = f(s) = \frac{1}{1 + e^{-s}}$$

*logsig* (logistic)



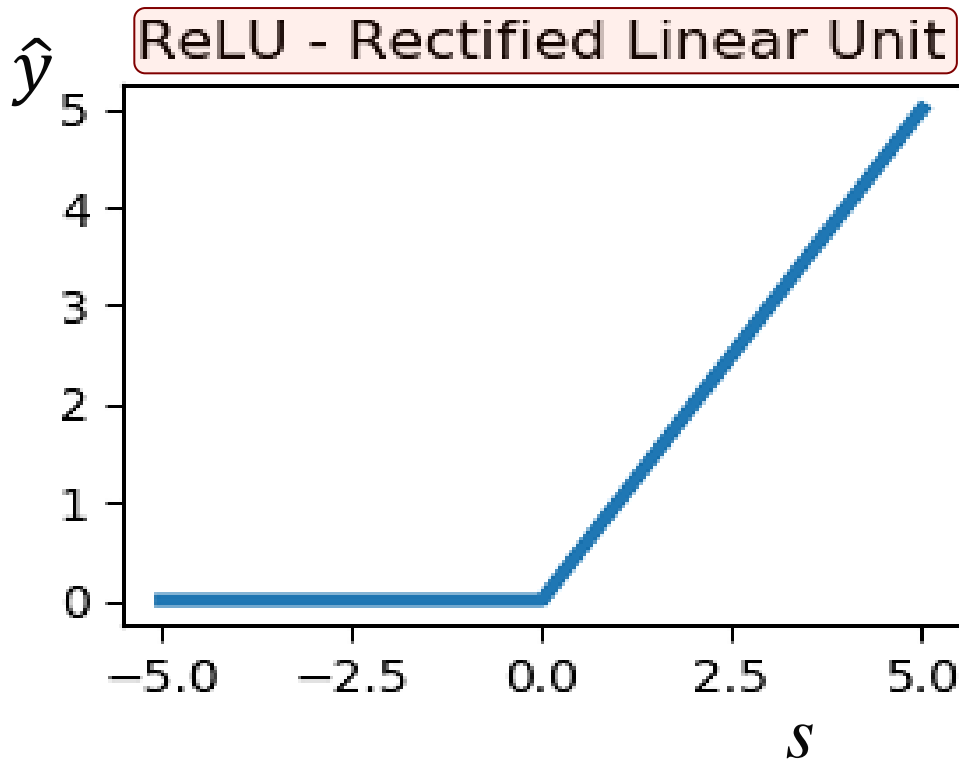
*Recommended in  
hidden layers*

$$\hat{y} = f(s) = \frac{1 - e^{-2s}}{1 + e^{-2s}}$$

$$= \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

*tansig*  
*tanh*





(poslin)

*Highly recommended in hidden layers*

$$\hat{y} = f(s) = \max(0, s)$$

$$\hat{y} = f(s) = \begin{cases} 0, & s < 0 \\ s, & s \geq 0 \end{cases}$$

The **ReLU is now the most used activation function.**

It is used in almost all convolutional neural networks or deep learning.

For optimization of cost function, the differentiation is used in almost every part of Machine Learning and Deep Learning.

$$\hat{y}' = f'(s) = \begin{cases} 0, & s < 0 \\ 1, & s \geq 0 \end{cases}$$

Very simple for ReLU



## Softmax (used for multiclass classification)

In deep learning, the logits layer is popularly used for the last neuron layer of neural network for classification task which produces raw prediction values as real numbers ranging from  $[-\infty, +\infty]$ .

Logits are interpreted as unnormalized (or not-yet normalized) predictions (or outputs) of a model. These can give results, but we don't normally stop with logits, because interpreting their raw values is not easy, **there is no maximum value**

Softmax acts as an activation function, and it turns into a probability distribution by taking the exponents of each output and then normalizing each number by the sum of those exponents.

Logits scores ( $s$ )	Softmax	Output probabilities	Classes
1.0	$f(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}}$	0.1767	Car
2.0		0.4803	Truck
0.2		0.0794	Motorcycle
1.4		0.2636	Bus
		Sum = 1	

Logits scores ( $s$ )	Softmax	Output probabilities	Classes
5.0	$f(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}}$	0.2641	Car
6.0		0.7179	Truck
1.0		0.0048	Motorcycle
2.0		0.0131	Bus
		Sum = 1	

So, **the entire output vector adds up to one — all probabilities should add up to one.**

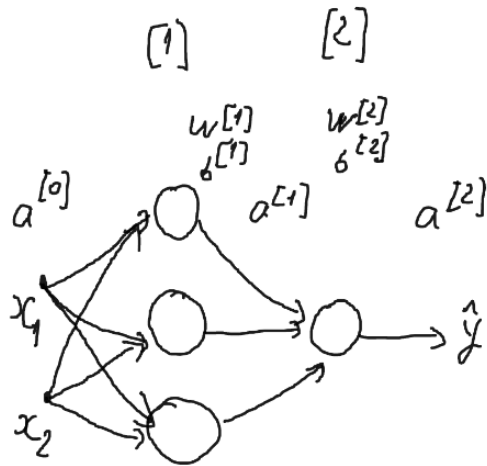
Softmax is frequently appended to the **last layer of a multi-class image classification network** such as those in **CNN** ( Alexnet, VGG16, etc.) used in ImageNet competitions.

[Understand the Softmax Function in Minutes, January 2018, <https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d>]



# Why is it necessary to use nonlinear activation functions?

*Assume:* Getting rid of activation functions (or using purelinear activation function) ...



$$s^{[1]} = w^{[1]}x + b^{[1]}$$

$$a^{[1]} = \cancel{f^{[1]}(s^{[1]})} = s^{[1]}$$

$$a^{[1]} = w^{[1]}x + b^{[1]}$$

$$s^{[2]} = w^{[2]}a^{[1]} + b^{[2]}$$

$$\hat{y} = a^{[2]} = \cancel{f^{[2]}(s^{[2]})} = s^{[2]}$$

$$a^{[2]} = w^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = w^{[2]}a^{[1]} + b^{[2]} = w^{[2]}(w^{[1]}x + b^{[1]}) + b^{[2]}$$

$$= \underbrace{w^{[2]}w^{[1]}}_{w'}x + \underbrace{w^{[2]}b^{[1]} + b^{[2]}}_{b'} = w'x + b'$$

$$a^{[2]} = w'x + b'$$

**Only linear relation can be implemented !!!**

**To implement interesting, nonlinear input-output relation, nonlinear activation functions are mandatory.**

Sometimes the output layer (*only the output layer*) can use a linear activation function, for example for fitting problems – predicting the price of a house



# Pros and cons of activation functions

**Only in the output layer**, if there is a binary classification problem

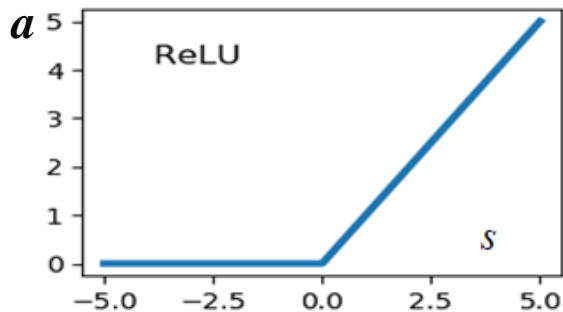
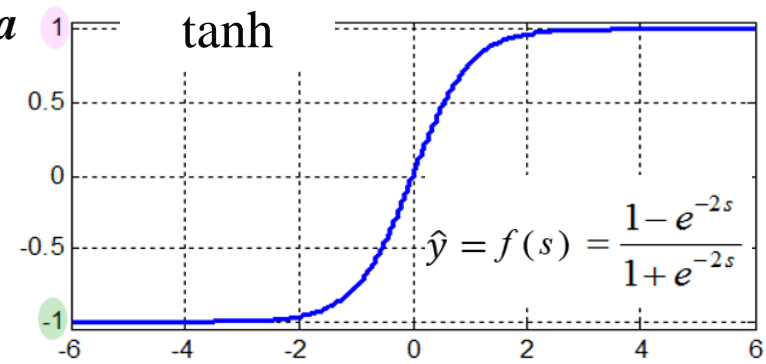
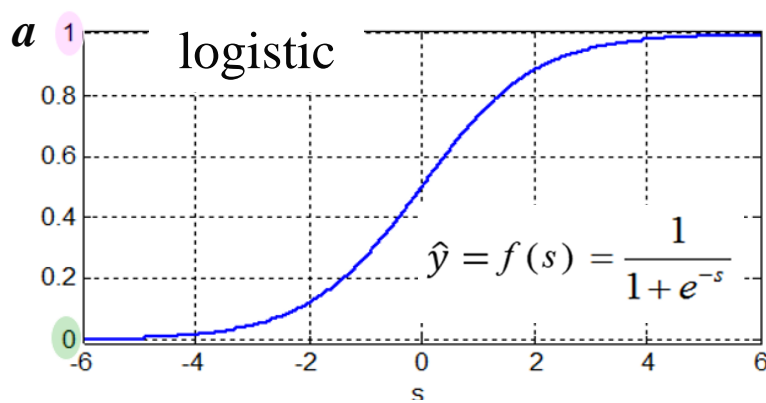
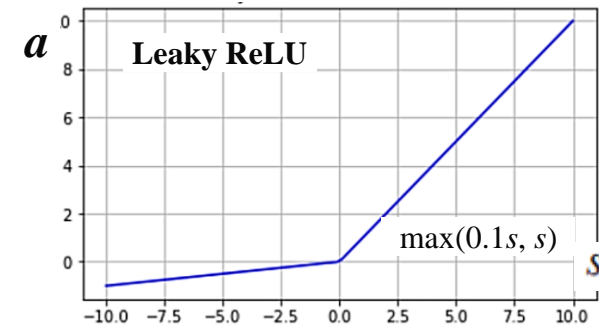
## Recommended in hidden layers

Superior to logistic activation function in all respects. Has the effect of centering data (activation) so that the mean of data is closer to 0 rather than, maybe 0.5 (logistic function). And this actually makes learning for the next layer a little bit easier. The derivatives for large positive and negative values is 0 – disadvantage (slows down gradient descent)

The derivative is very simple: 1 or 0

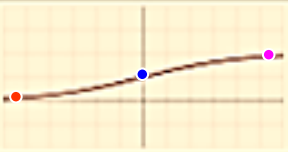
The most commonly used (popular) activation function. Learning is quite fast for most training examples (much faster than when using tanh)

**Highly recommended in hidden layers**



Can solve the problem of zero derivative for negative  $s$ . Usually works better than ReLU although it's just not used as much in practice.

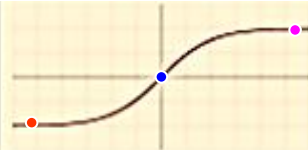
# Derivatives of activation functions

Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
----------------------------	---	-------------------------------	--------------------------

$$x = 0; f(x) = \frac{1}{2}; f'(x) = \frac{1}{2} \left( 1 - \frac{1}{2} \right) = \frac{1}{4}$$

$$x = -10; f(x) \approx 0; f'(x) = 0(1 - 0) = 0$$

$$x = +10; f(x) \approx 1; f'(x) = 0(1 - 0) = 0$$

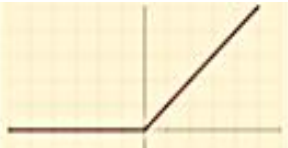
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
------	---	---	----------------------

$$x = 0; f(x) = 0; f'(x) = 1 - 0 = 1$$

$$x = -10; f(x) \approx -1; f'(x) = 1 - 1 = 0$$

$$x = +10; f(x) \approx 1; f'(x) = 1 - 1 = 0$$

The advantage of these functions is that if we've already computed the value for  $f$ , then by using the expressions for the derivatives, we can **very quickly compute the value for the slope** (derivative) in that point

Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
------------------------------	---	--	---

**Very simple and quick slope computation**

**Accelerates iterative training**





# Problem

The diagram for a SNN (shallow neural network) for 3-class classification (C0, C1, C2) is represented in the figure.

The SNN is used in evaluation mode (inference) for two input examples:

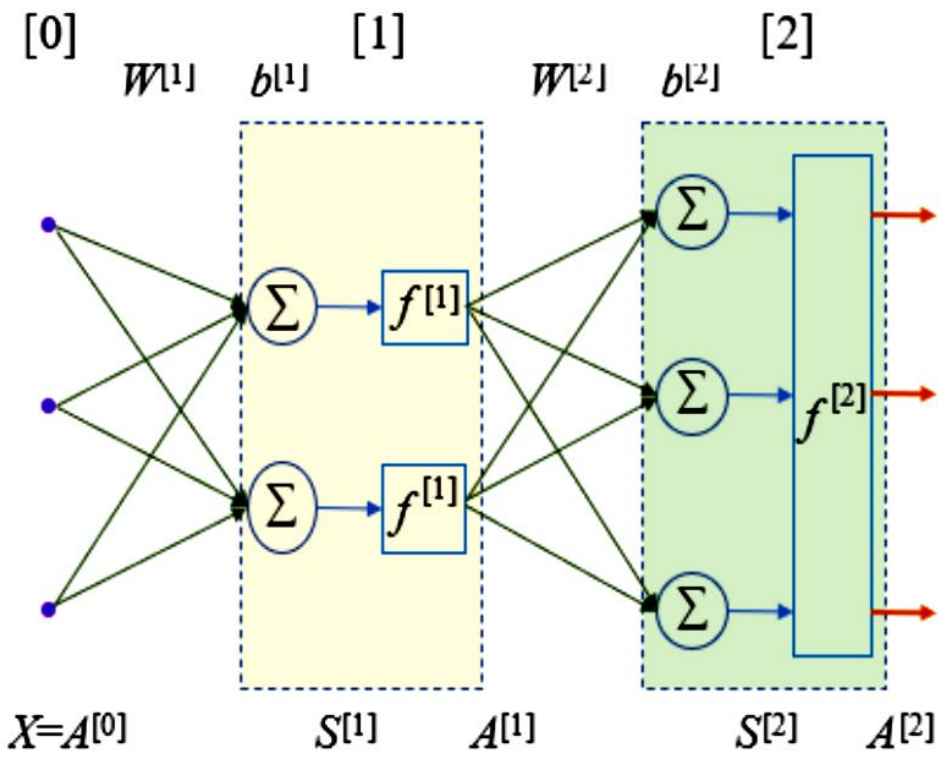
$$X = \begin{bmatrix} -2 & 5 \\ 1 & 2 \\ 2 & -1 \end{bmatrix}$$

For the SNN we know all the parameters:

$$W^{[1]} = \begin{bmatrix} 0.5 & 0.2 & 0.25 \\ -0.2 & 0.1 & -0.2 \end{bmatrix} \quad b^{[1]} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

$f^{[1]}$  - ReLU

$$W^{[2]} = \begin{bmatrix} -0.5 & 0.1 \\ -0.2 & 0.05 \\ 0.5 & -0.1 \end{bmatrix} \quad b^{[2]} = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix} \quad f^{[2]} - \text{softmax}$$



- a) **0.5p** Describe the SNN architecture. What is the role of softmax activation function?
- b) **0.5p** Determine the activation matrix  $A^{[1]}$ .
- c) **0.5p** Determine the predicted output matrix  $A^{[2]} = \hat{Y}$
- d) **0.5p** In the picture, the top output correspond to class C1, the middle output to class C2, and the bottom output to class C3. To which class does each of the examples presented at the entrance belong? Explain your answer.