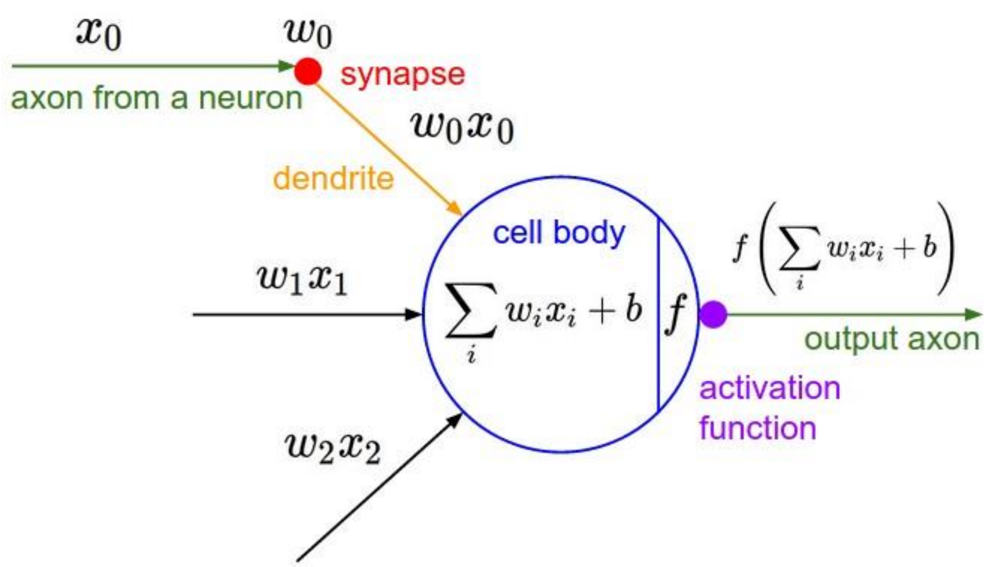
ISE 469- DERİN ÖĞRENMEYE GİRİŞ

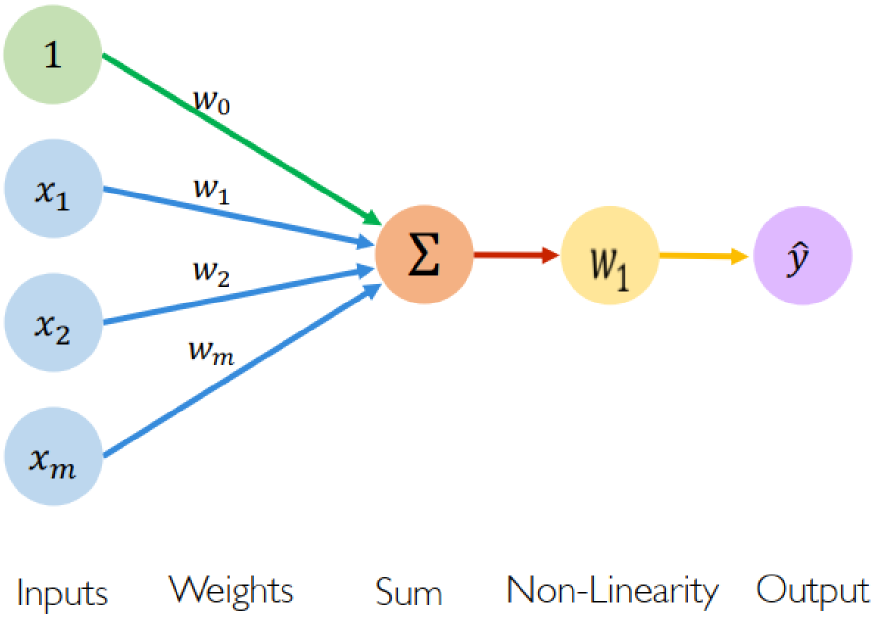
Yapay sinir hücresi



Kaynak: http://cs231n.github.io/convolutional-networks/

Yapay Sinir Ağları

The Perceptron: Forward Propagation



Output

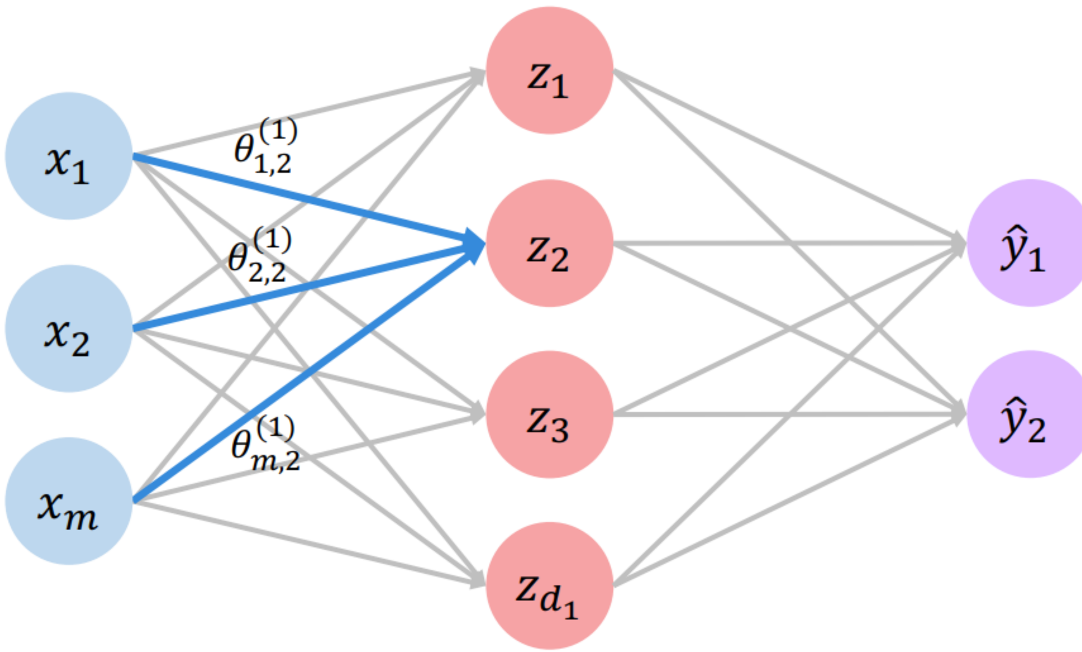
Linear combination of inputs

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i \right)$$

Non-linear activation function

Bias

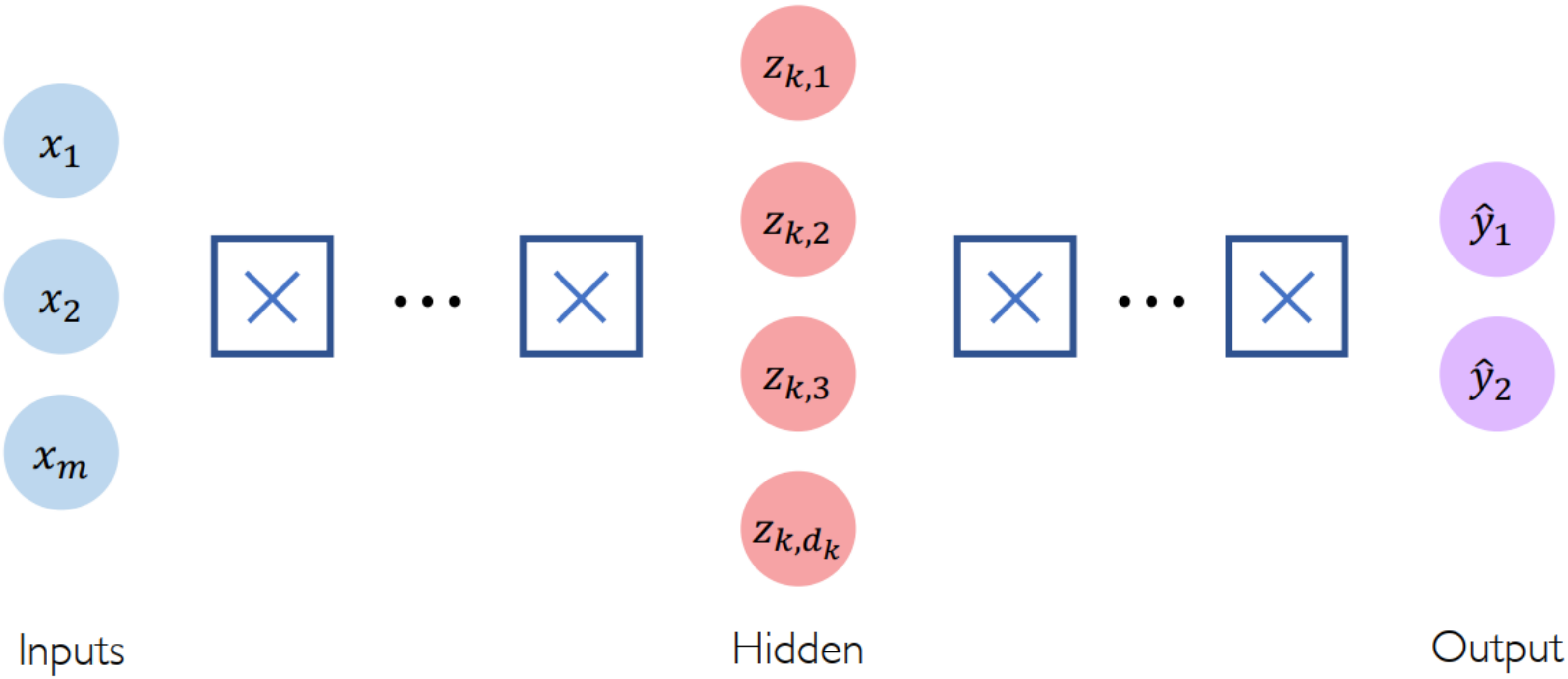
Yapay Sinir Ağları



$$z_2 = w_{0,2}^{(1)} + \sum_{j=1}^{m} x_j\, w_{j,2}^{(1)}$$

$$= w_{0,2}^{(1)} + x_1\, w_{1,2}^{(1)} + x_2\, w_{2,2}^{(1)} + x_m\, w_{m,2}^{(1)}$$

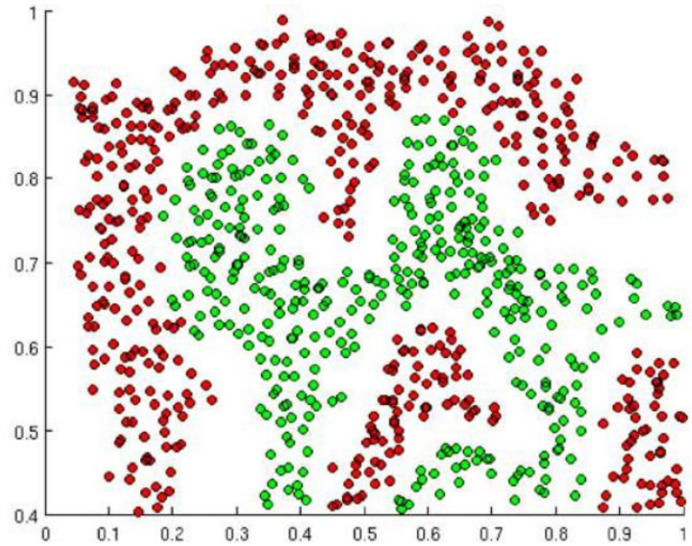
Deep Neural Network



$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{d_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

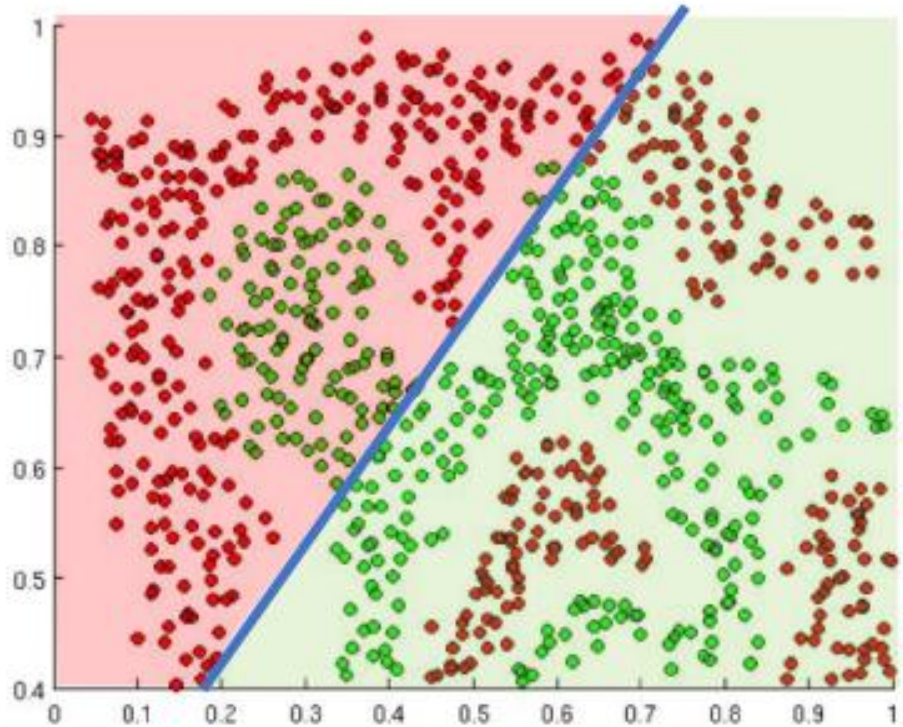
Aktivasyon fonksiyonunun önemi

The purpose of activation functions is to ***introduce non-linearities*** *into the network*



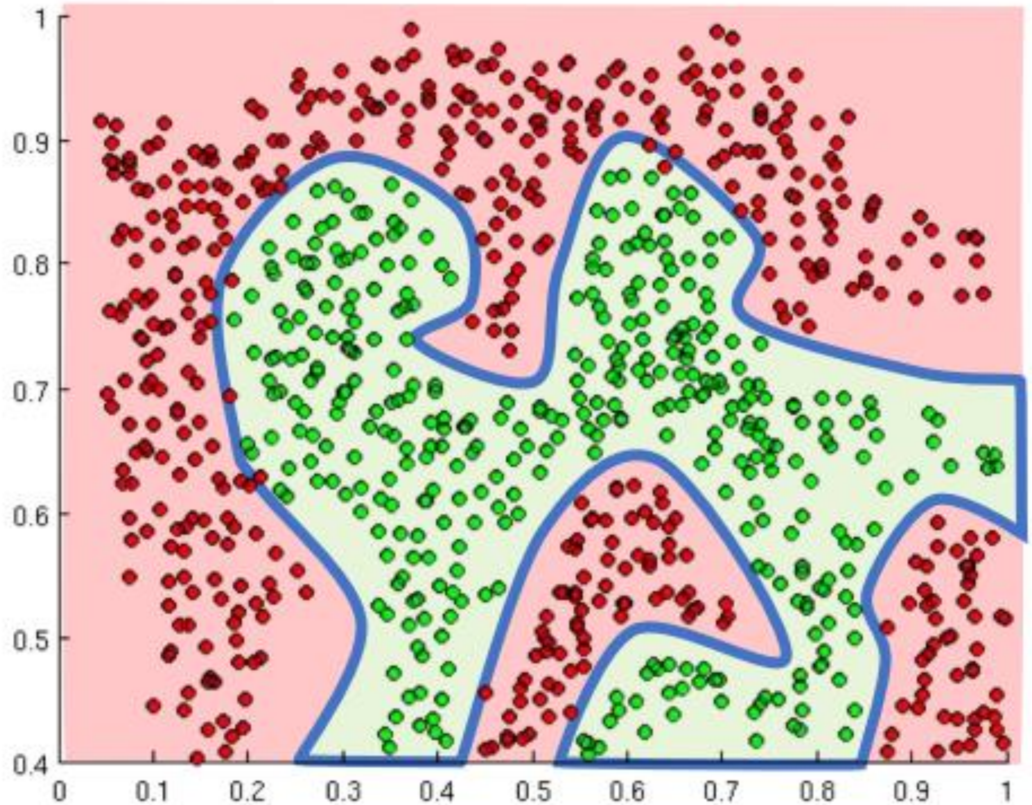
What if we wanted to build a Neural Network to distinguish green vs red points?

Doğrusal (Linear) aktivasyon fonksiyonu



Linear Activation functions produce linear decisions no matter the network size

Doğrusal olmayan (Linear) aktivasyon fonksiyonu



Non-linearities allow us to approximate arbitrarily complex functions

Aktivasyon fonksiyonları

linear

```
keras.activations.linear(x)
```

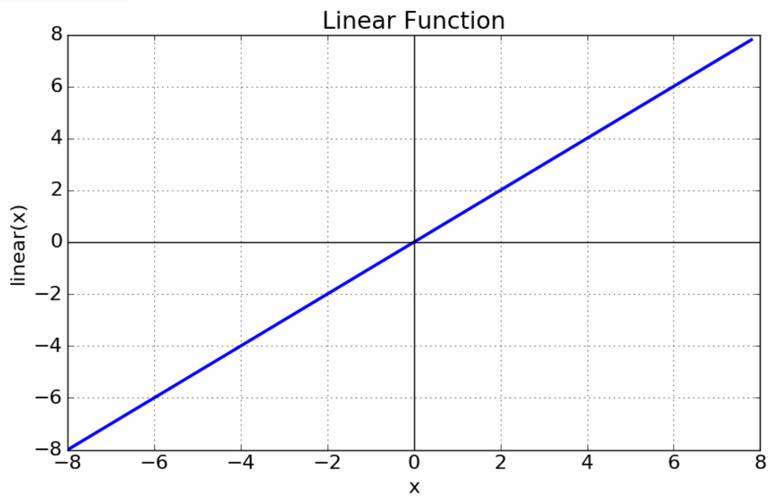
Linear (i.e. identity) activation function.

Arguments

- **x**: Input tensor.

Returns

Input tensor, unchanged.



Kaynak: https://keras.io/activations/
Kaynak: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

Aktivasyon fonksiyonları

relu

```
keras.activations.relu(x, alpha=0.0, max_value=None, threshold=0.0)
```

Rectified Linear Unit.

With default values, it returns element-wise `max(x, 0)`.

Otherwise, it follows: `f(x) = max_value` for `x >= max_value`, `f(x) = x` for `threshold <= x < max_value`, `f(x) = alpha * (x - threshold)` otherwise.

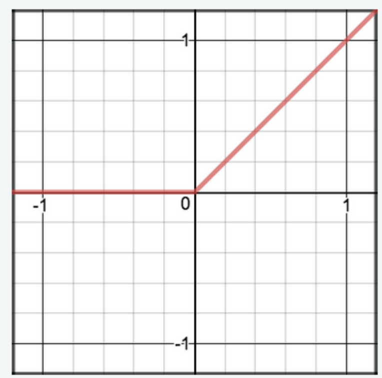
Arguments

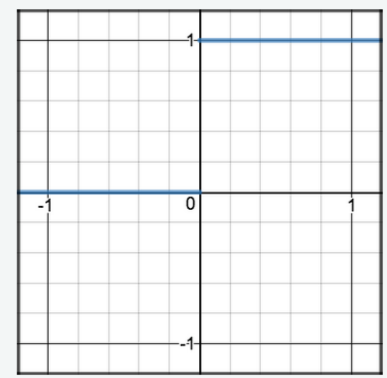
- **x**: Input tensor.
- **alpha**: float. Slope of the negative part. Defaults to zero.
- **max_value**: float. Saturation threshold.
- **threshold**: float. Threshold value for thresholded activation.

Returns

A tensor.

Function	Derivative
$R(z) = \begin{Bmatrix} z & z > 0 \\ 0 & z <= 0 \end{Bmatrix}$	$R'(z) = \begin{Bmatrix} 1 & z > 0 \\ 0 & z < 0 \end{Bmatrix}$





Kaynak: https://keras.io/activations/
Kaynak: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

Aktivasyon fonksiyonları

sigmoid

```
keras.activations.sigmoid(x)
```

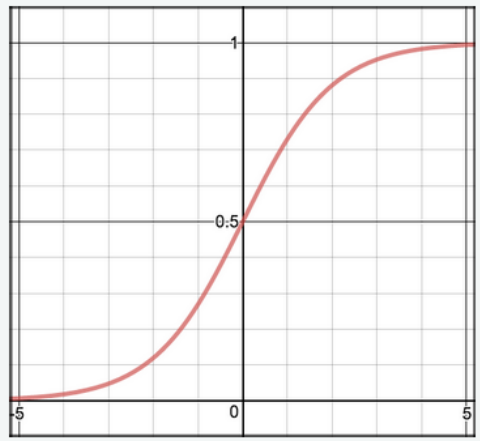
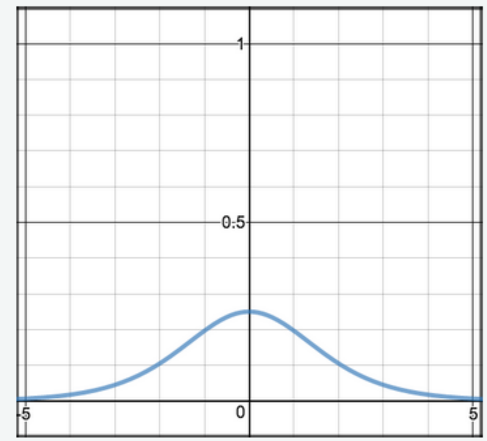
Sigmoid activation function.

Arguments

- **x**: Input tensor.

Returns

The sigmoid activation: `1 / (1 + exp(-x))`.

Function	Derivative
$S(z) = \frac{1}{1+e^{-z}}$	$S'(z) = S(z) \cdot (1 - S(z))$
	

Kaynak: https://keras.io/activations/
Kaynak: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

Aktivasyon fonksiyonları

softmax

```
keras.activations.softmax(x, axis=-1)
```

Softmax activation function.

Arguments

- **x**: Input tensor.
- **axis**: Integer, axis along which the softmax normalization is applied.

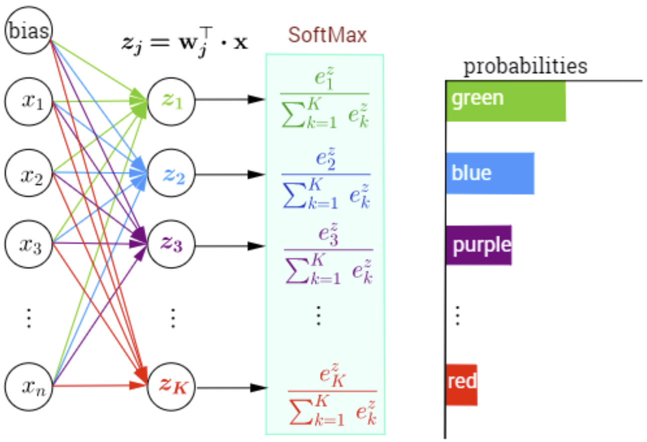
Returns

Tensor, output of softmax transformation.

Raises

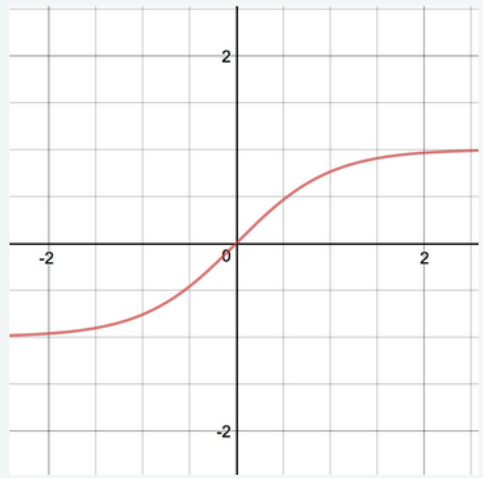
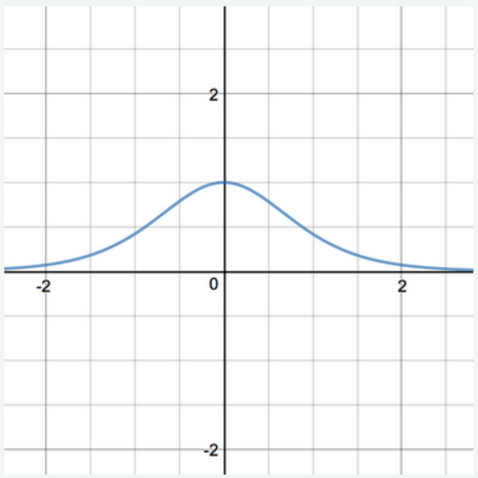
- **ValueError**: In case `dim(x) == 1`.

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



Kaynak: https://keras.io/activations/
Kaynak: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

Aktivasyon fonksiyonları

Function	Derivative
$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$tanh'(z) = 1 - tanh(z)^2$
	

tanh

```
keras.activations.tanh(x)
```

Hyperbolic tangent activation function.

Arguments

- **x**: Input tensor.

Returns

The hyperbolic activation: `tanh(x) = (exp(x) - exp(-x)) / (exp(x) + exp(-x))`

Kaynak: https://keras.io/activations/
Kaynak: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Aktivasyon fonksiyonları

elu

```
keras.activations.elu(x, alpha=1.0)
```

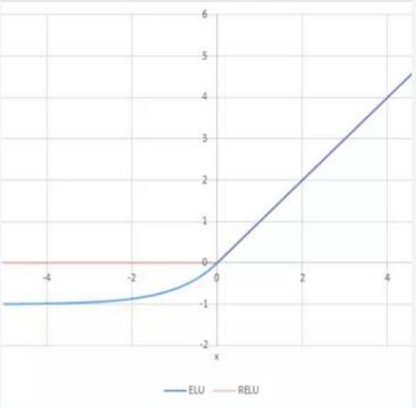
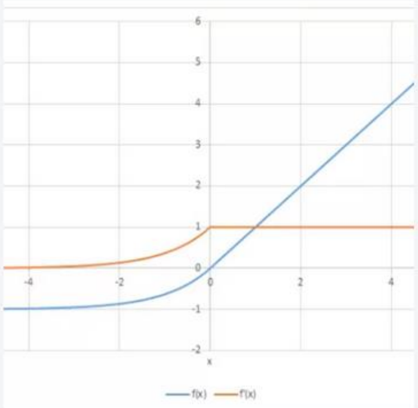
Exponential linear unit.

Arguments

- **x**: Input tensor.
- **alpha**: A scalar, slope of negative section.

Returns

The exponential linear activation: `x` if `x > 0` and `alpha * (exp(x)-1)` if `x < 0`.

Function	Derivative
$R(z) = \begin{Bmatrix} z & z > 0 \\ \alpha.(e^z - 1) & z <= 0 \end{Bmatrix}$	$R'(z) = \begin{Bmatrix} 1 & z > 0 \\ \alpha.e^z & z < 0 \end{Bmatrix}$
	

Kaynak: https://keras.io/activations/
Kaynak: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

Aktivasyon fonksiyonları

selu

```
keras.activations.selu(x)
```

Scaled Exponential Linear Unit (SELU).

SELU is equal to: `scale * elu(x, alpha)`, where alpha and scale are predefined constants. The values of `alpha` and `scale` are chosen so that the mean and variance of the inputs are preserved between two consecutive layers as long as the weights are initialized correctly (see `lecun_normal` initialization) and the number of inputs is "large enough" (see references for more information).

Arguments

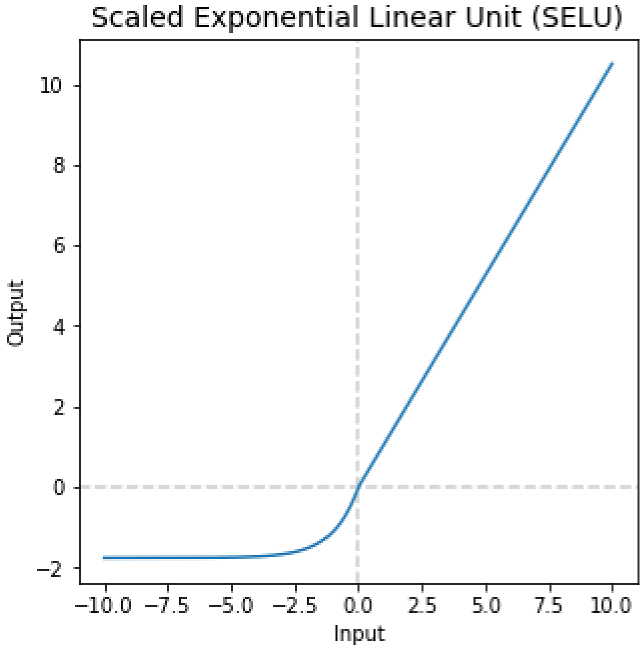
- **x**: A tensor or variable to compute the activation function for.

Returns

The scaled exponential unit activation: `scale * elu(x, alpha)`.

Note

- To be used together with the initialization "lecun_normal".
- To be used together with the dropout variant "AlphaDropout".



Kaynak: https://keras.io/activations/
Kaynak: https://towardsdatascience.com/deep-study-of-a-not-very-deep-neural-network-part-2-activation-functions-fd9bd8d406fc

Aktivasyon fonksiyonları

hard_sigmoid

```
keras.activations.hard_sigmoid(x)
```

Hard sigmoid activation function.

Faster to compute than sigmoid activation.

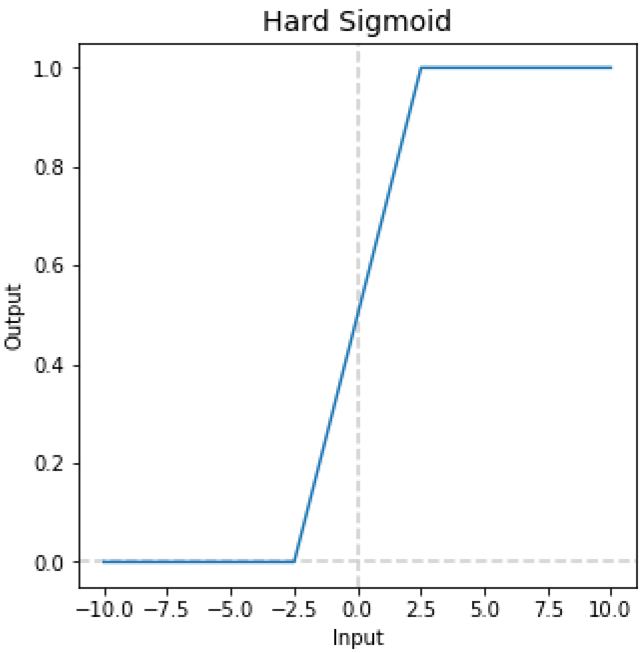
Arguments

- **x**: Input tensor.

Returns

Hard sigmoid activation:

- `0` if `x < -2.5`
- `1` if `x > 2.5`
- `0.2 * x + 0.5` if `-2.5 <= x <= 2.5`.



Kaynak: https://keras.io/activations/
Kaynak: https://towardsdatascience.com/deep-study-of-a-not-very-deep-neural-network-part-2-activation-functions-fd9bd8d406fc

Aktivasyon fonksiyonları

softplus

```
keras.activations.softplus(x)
```

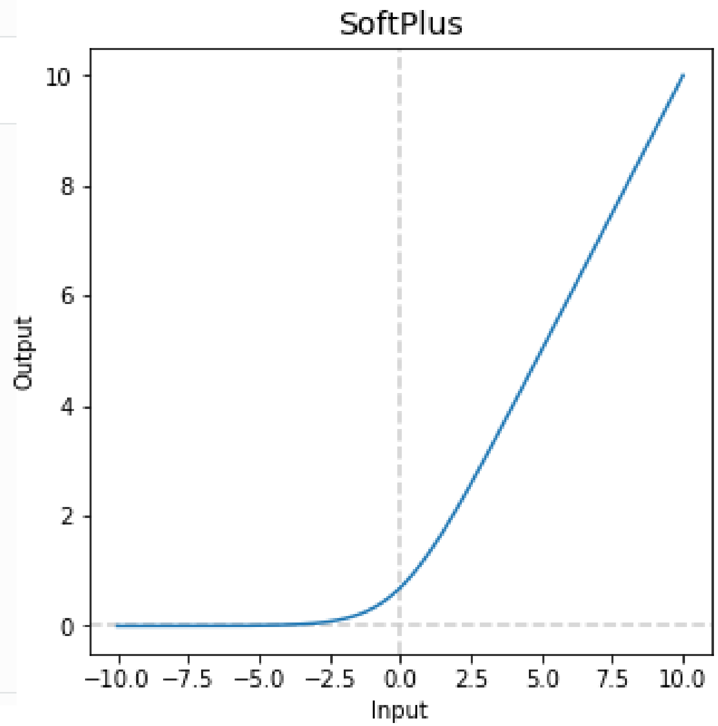
Softplus activation function.

Arguments

- **x**: Input tensor.

Returns

The softplus activation: `log(exp(x) + 1)`.



Kaynak: https://keras.io/activations/
Kaynak: https://towardsdatascience.com/deep-study-of-a-not-very-deep-neural-network-part-2-activation-functions-fd9bd8d406fc

Aktivasyon fonksiyonları

softsign

```
keras.activations.softsign(x)
```

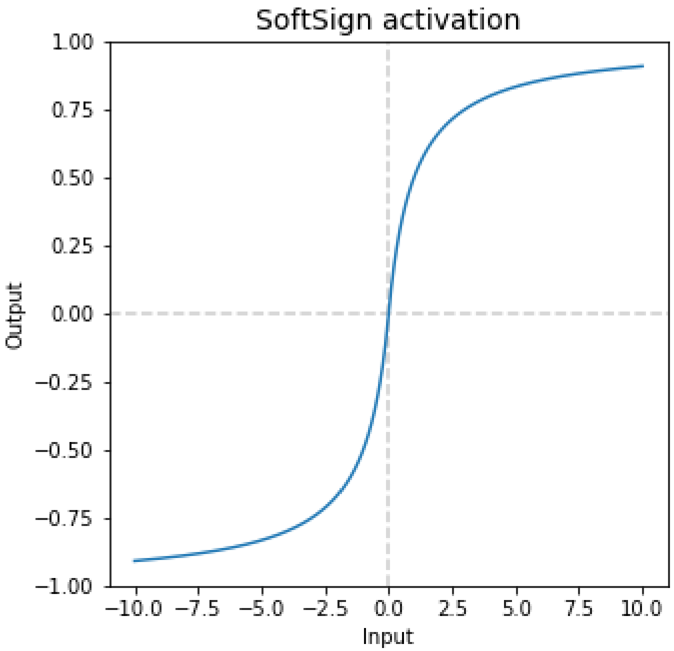
Softsign activation function.

Arguments

- **x**: Input tensor.

Returns

The softsign activation: `x / (abs(x) + 1)`.



Kaynak: https://keras.io/activations/
Kaynak: https://towardsdatascience.com/deep-study-of-a-not-very-deep-neural-network-part-2-activation-functions-fd9bd8d406fc

Kaynak: https://keras.io/activations/

Aktivasyon fonksiyonları

exponential

```
keras.activations.exponential(x)
```

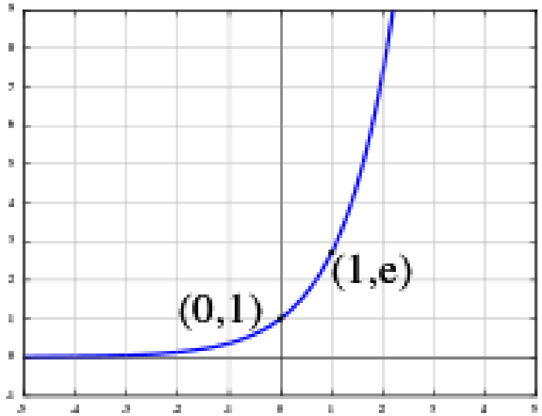
Exponential (base e) activation function.

Arguments

- **x**: Input tensor.

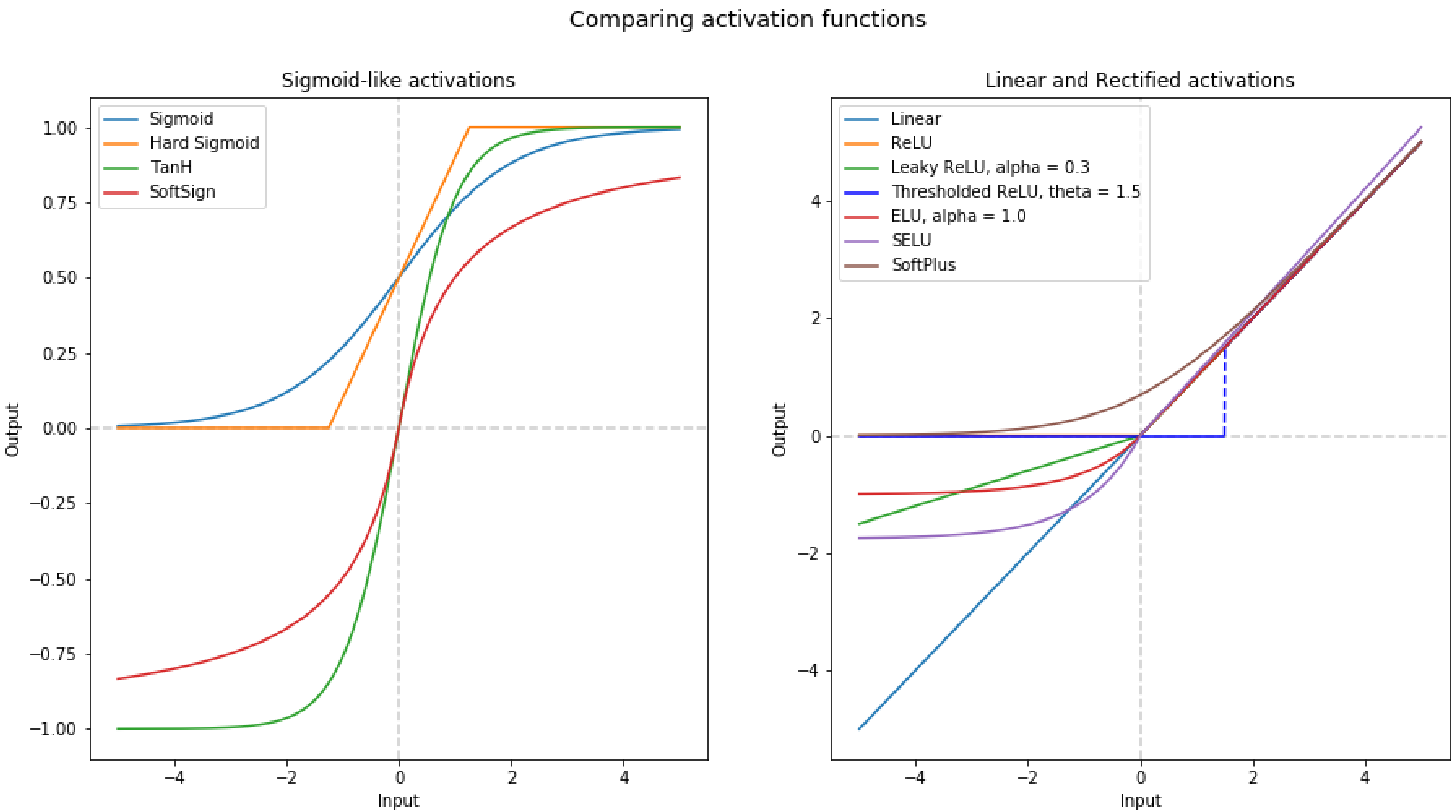
Returns

Exponential activation: `exp(x)`.



Kaynak: https://keras.io/activations/

Aktivasyon fonksiyonları



Kaynak: https://towardsdatascience.com/deep-study-of-a-not-very-deep-neural-network-part-2-activation-functions-fd9bd8d406fc

Applying Neural Networks

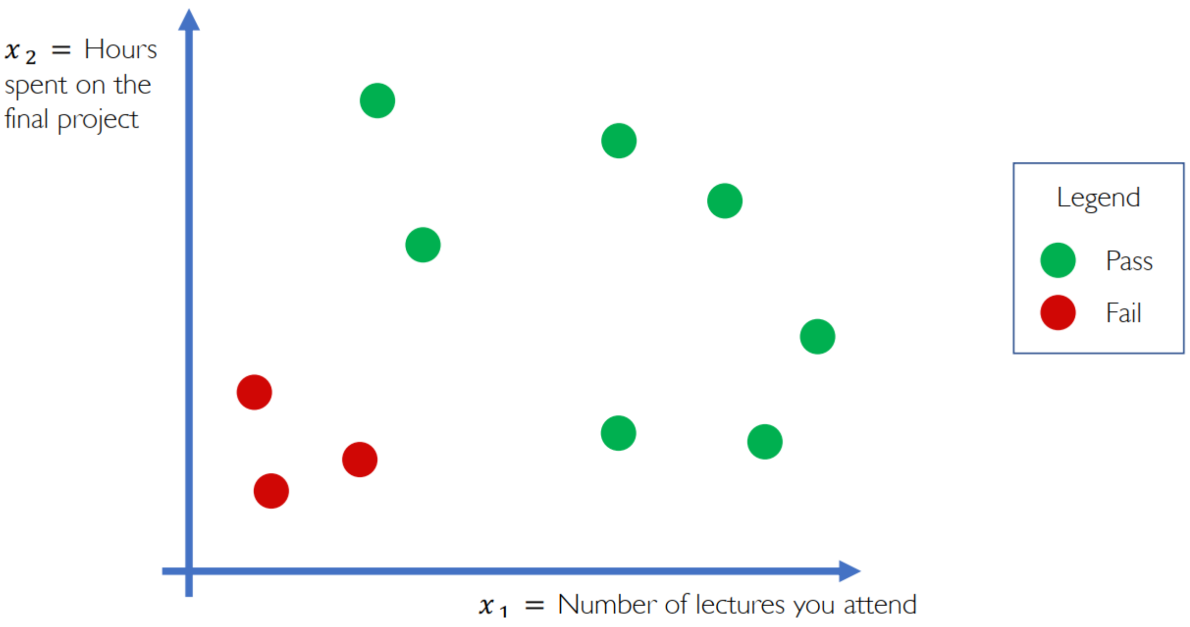
Example Problem

Will I pass this class?

Let's start with a simple two feature model

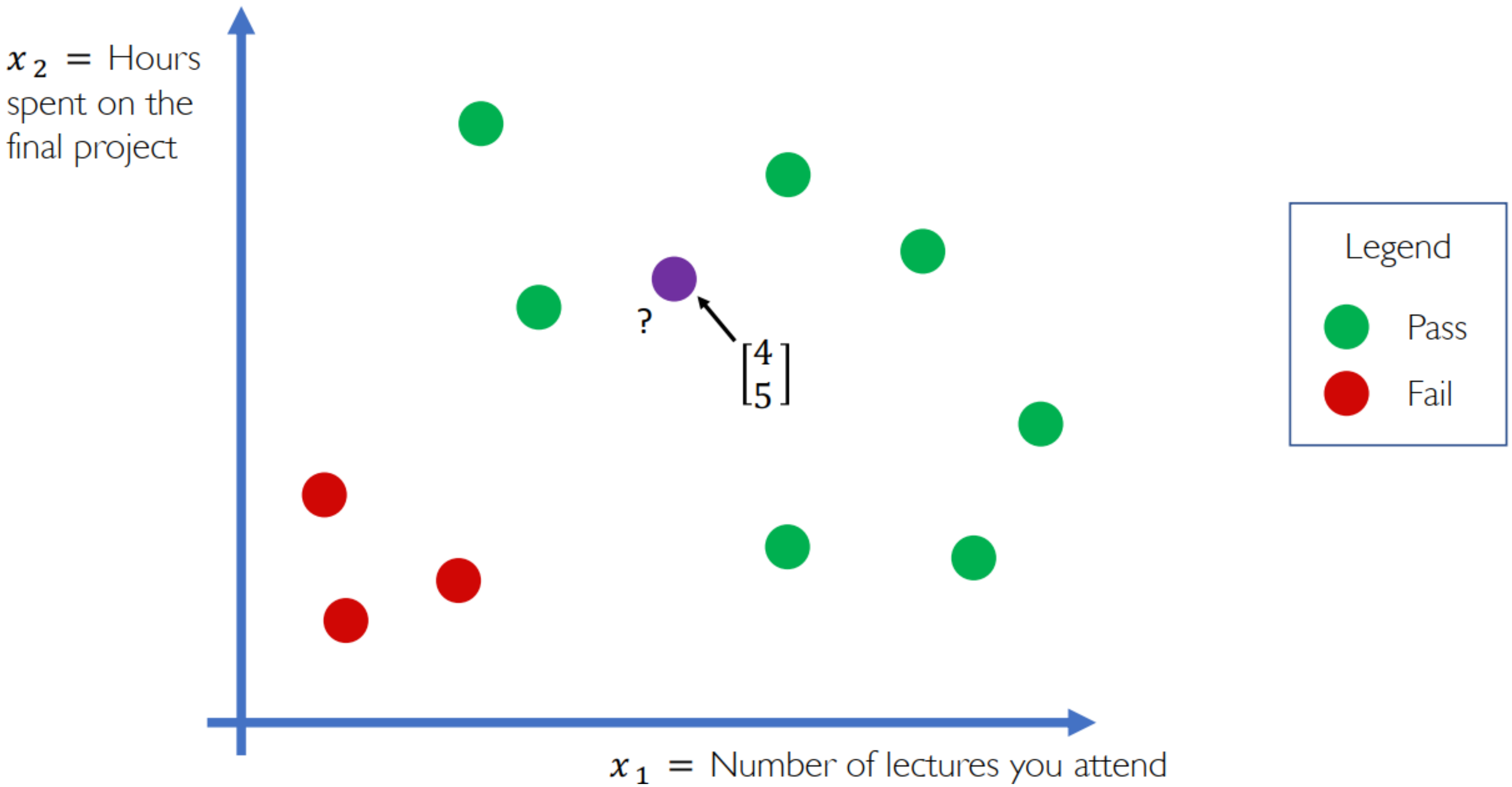
x_1 = Number of lectures you attend

x_2 = Hours spent on the final project

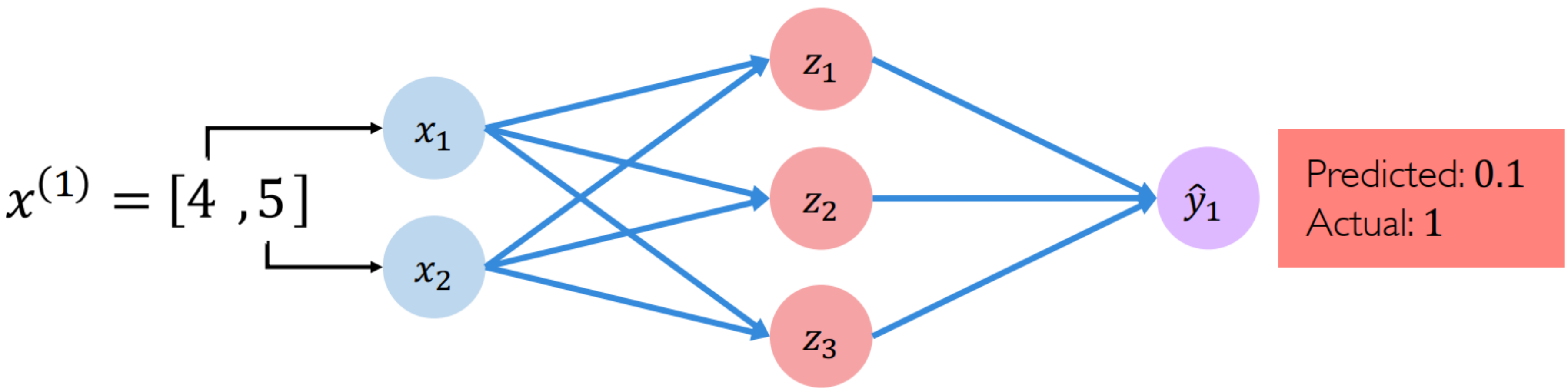


Applying Neural Networks

Example Problem: Will I pass this class?

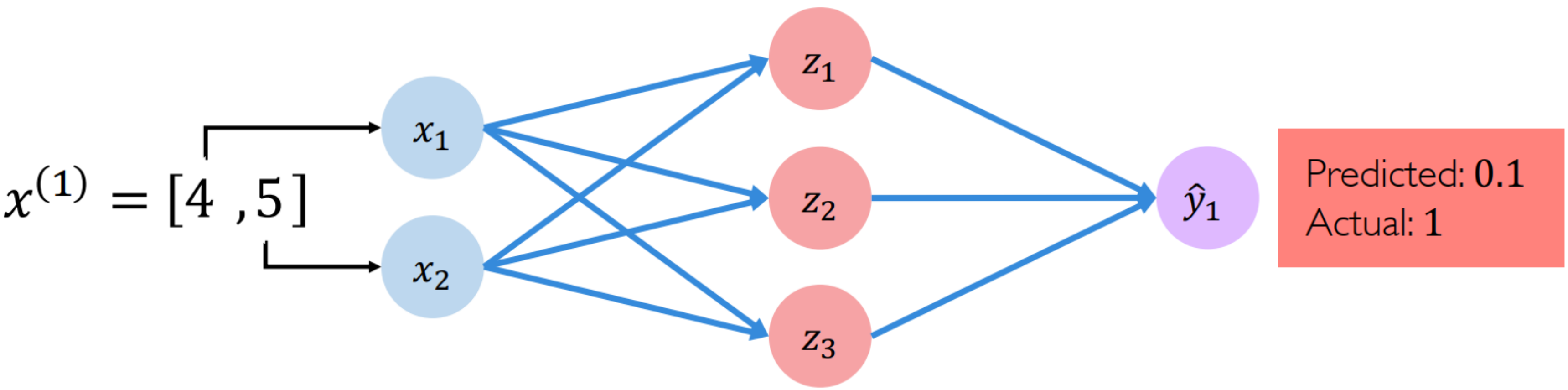


Applying Neural Networks



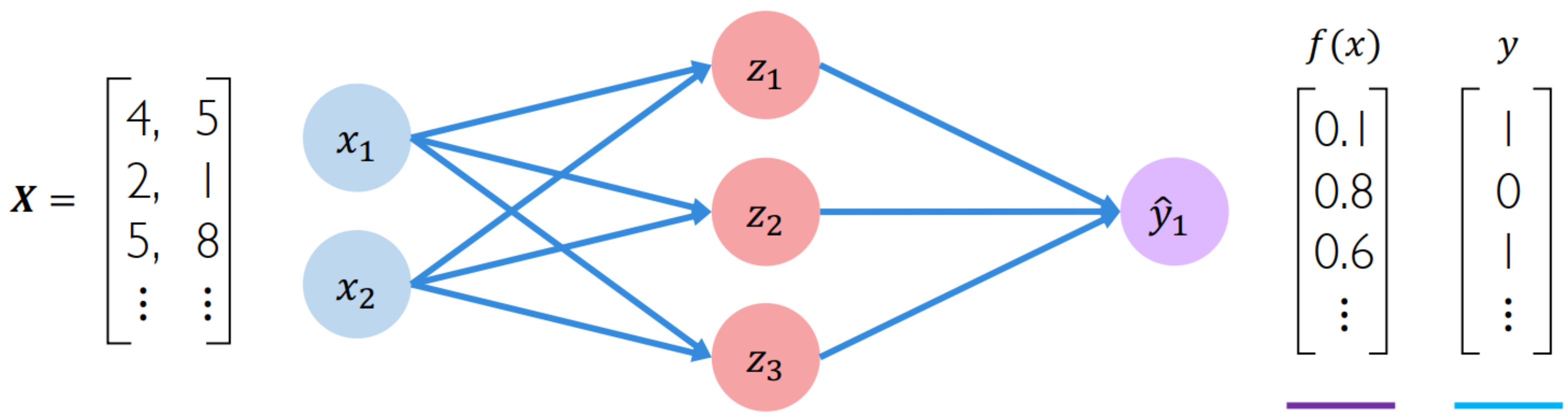
Quantifying Loss

*The **loss** of our network measures the cost incurred from incorrect predictions*



$$\mathcal{L}\big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}},\ \underbrace{y^{(i)}}_{\text{Actual}}\big)$$

*The **empirical loss** measures the total loss over our entire dataset*



$$\boldsymbol{J}(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\Big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\Big)$$

Also known as:

- Objective function
- Cost function
- Empirical Risk