Training Camp 2024

First Day – 4th September 2024

Alessandro Nicolosi

https://github.com/alenic

https://www.linkedin.com/in/alessandro-nicolosi/

LEONARDO

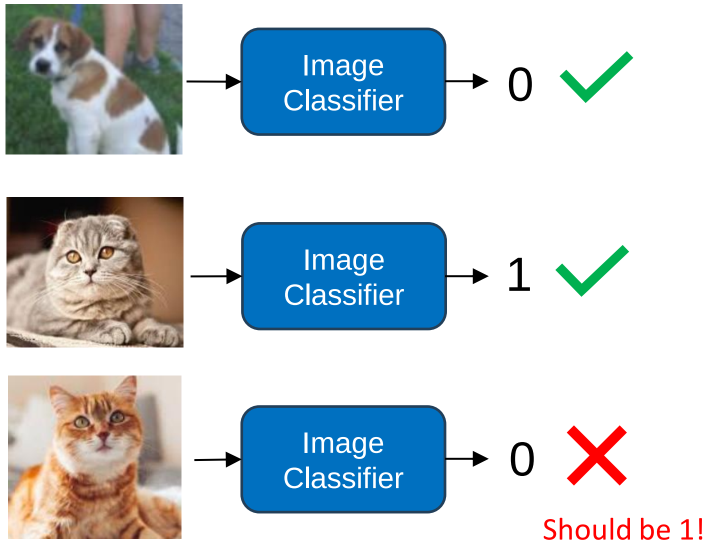
https://www.leonardo.com/it/

Single Modality – Classification Problem

Computer Vision

- **Task**: Image Classification
- **Input Modality**: Image

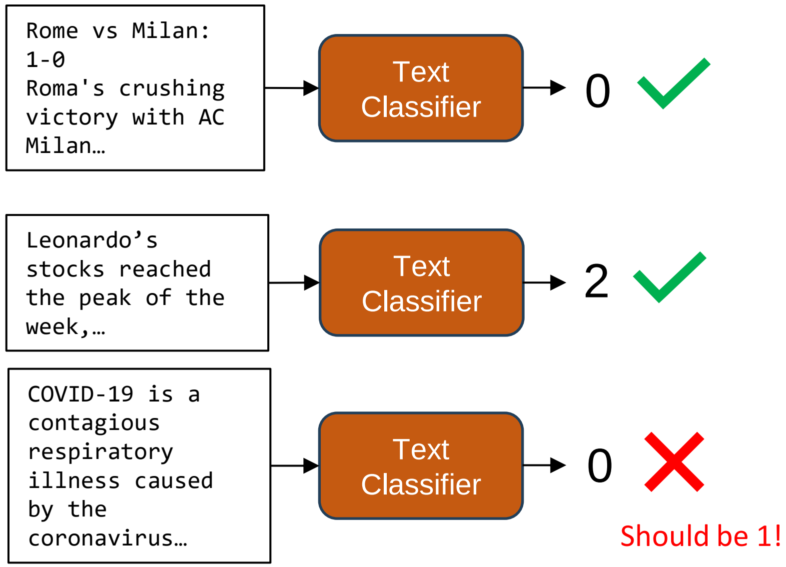
```
idx2label = {
  0: "dog"
  1: "cat"
}
```



Natural Language Processing

- **Task**: Text Classification
- **Input Modality**: Text

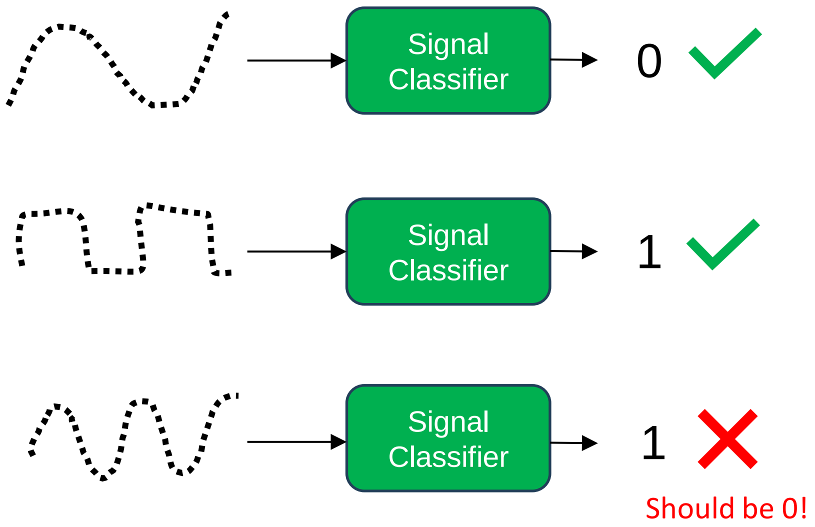
```
idx2label = {
  0: "sport"
  1: "health"
  2: "economy"
}
```



Time Series Analysis

- **Task**: Time Series Classification
- **Input Modality**: Time series

```
idx2label = {
  0: "sinusoid"
  1: "pulse"
}
```



Multimodal Classification Problem

A single example $x_i = (\text{Image}_i, \text{Text}_i, \text{Audio}_i)$

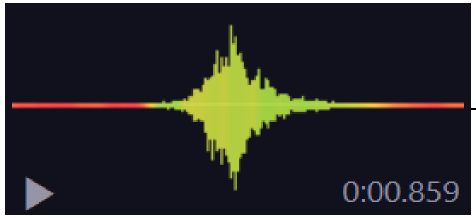
Input Modality

Image

Text description of the image

```
An image of a white
and brown dog, with
a short, pushed-in
nose. The dog is
standing in a garden
```

Audio example of the entity in the image



Multimodal Classifier

50 ✓

Dataset

$$D_{train} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

$$x_i = (\text{Image}_i, \text{Text}_i, \text{Audio}_i)$$

$$y_i \in \{0, 1, \dots, 99\}$$

```
idx2label = {
  0: "cat - persian"
  1: "cat - siamese"
  ...
  49: "dog - pug"
  50: "dog - bulldog"
  ...
  99: "dog - rottweiler"
}
```

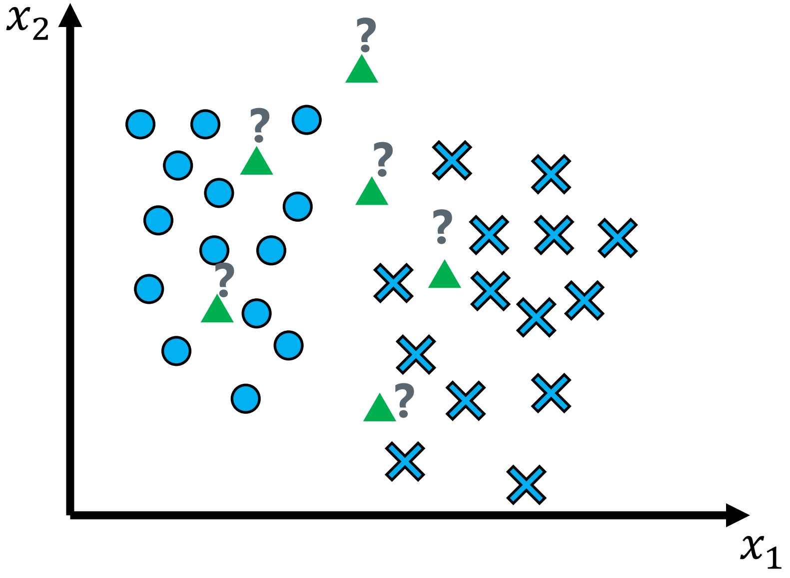
Binary Classification Problem

Given a training dataset

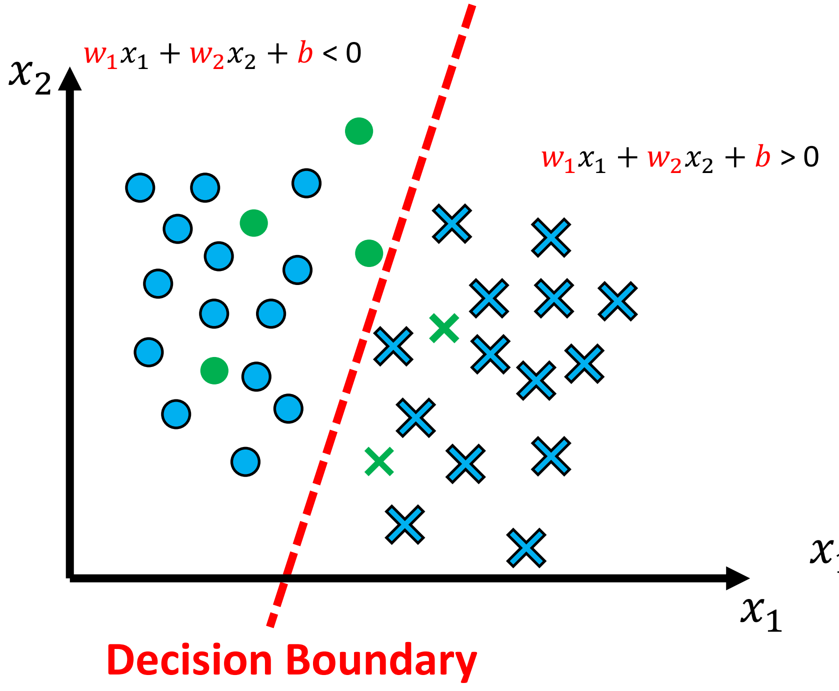
$$D_{train} = \{(x^{(i)}, y^{(i)})_{i=1}^{N_{train}} : \quad x^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{0, 1\}\}$$

$$D_{test} = \{x_{test}^{(1)}, x_{test}^{(2)}, \dots, x_{test}^{(N_t)}\}$$

What is the probability that a point $x_{test}^{(i)}$ belong to the class 1 ? $P(Y = 1 \mid X = x_{test}^{(i)})$?

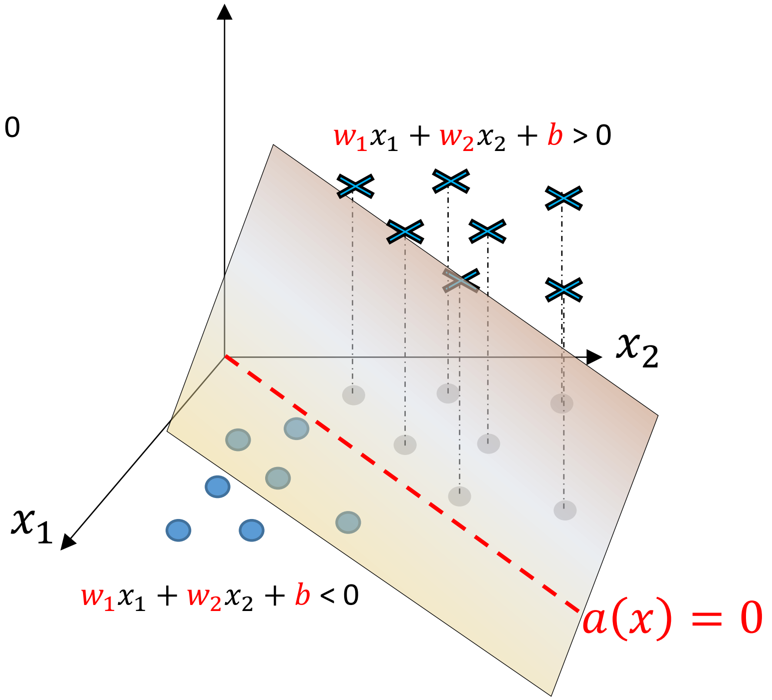


$$a(x) = w_1 x_1 + w_2 x_2 + b = 0$$

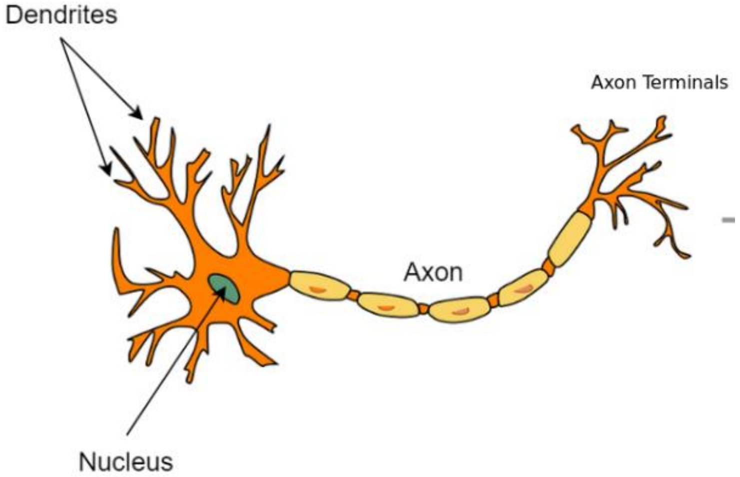


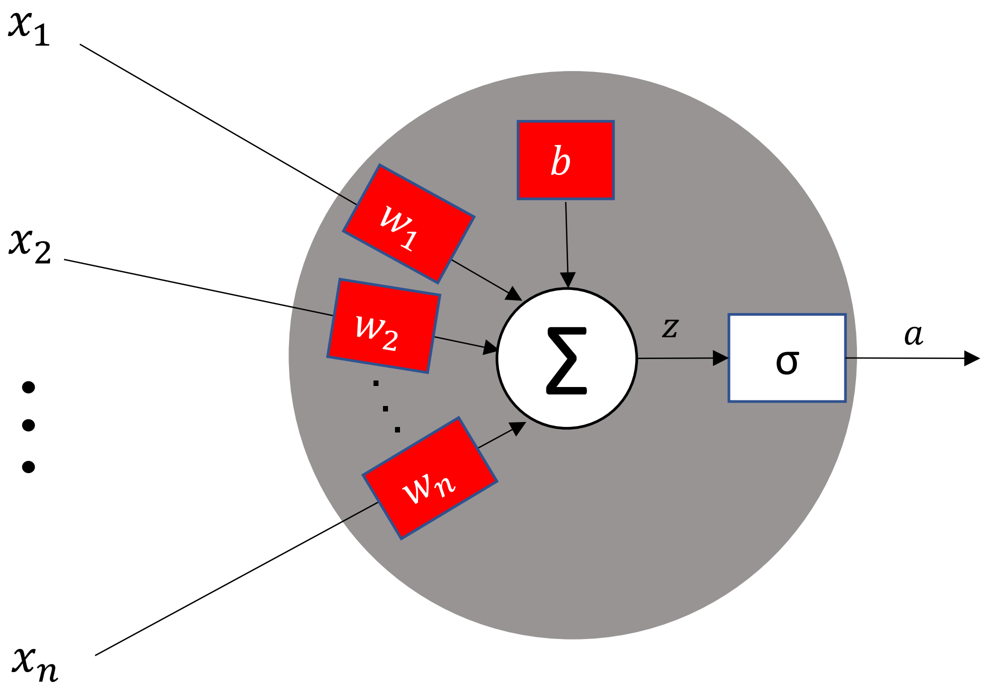
Hyperplane

$$a(x) = w_1 x_1 + w_2 x_2 + b$$



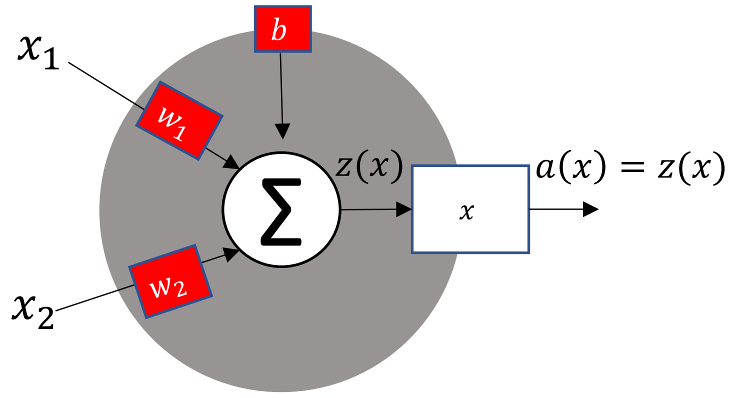
Neural Networks – Single Neuron





Input	$x = [x_1 \ \ x_2 \ \ \dots \ \ x_n] \in \mathbb{R}^n$
Output	$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$ $a(x) = \sigma(\ z(x)\ ) = \sigma(w^T x + b)$
Weights or Parameters	$w = [w_1 \ \ w_2 \ \ \dots \ \ w_n] \in \mathbb{R}^n$ $b \in \mathbb{R}$
Some Activation Functions: $\boldsymbol{\sigma(x)}$	
Linear	x
Hyperbolic Tangent	$\tanh(x)$
ReLU	$\begin{cases} x, & x \geq 0 \\ 0, & \mathrm{x} < 0 \end{cases}$
LeakyReLU	$\begin{cases} x, & x \geq 0 \\ \alpha x, & \mathrm{x} < 0 \end{cases}$

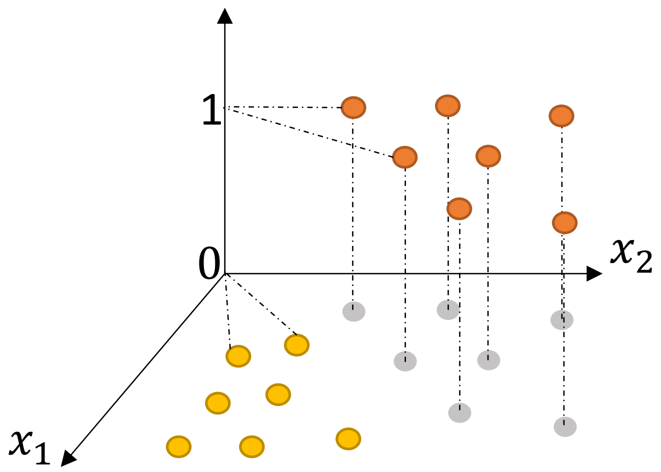
Single Neuron for classification



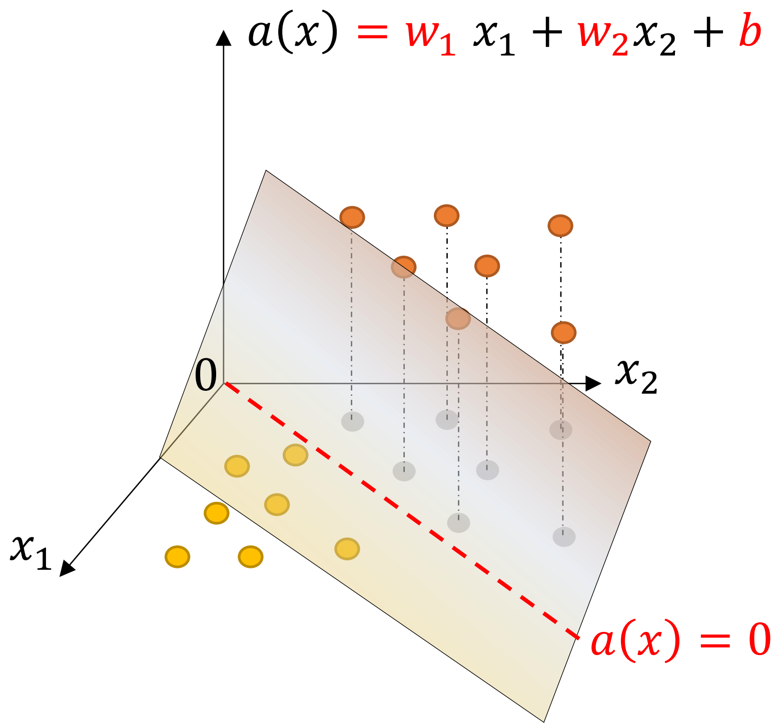
Activation Functions: $\sigma(x) = 0$

Training Dataset

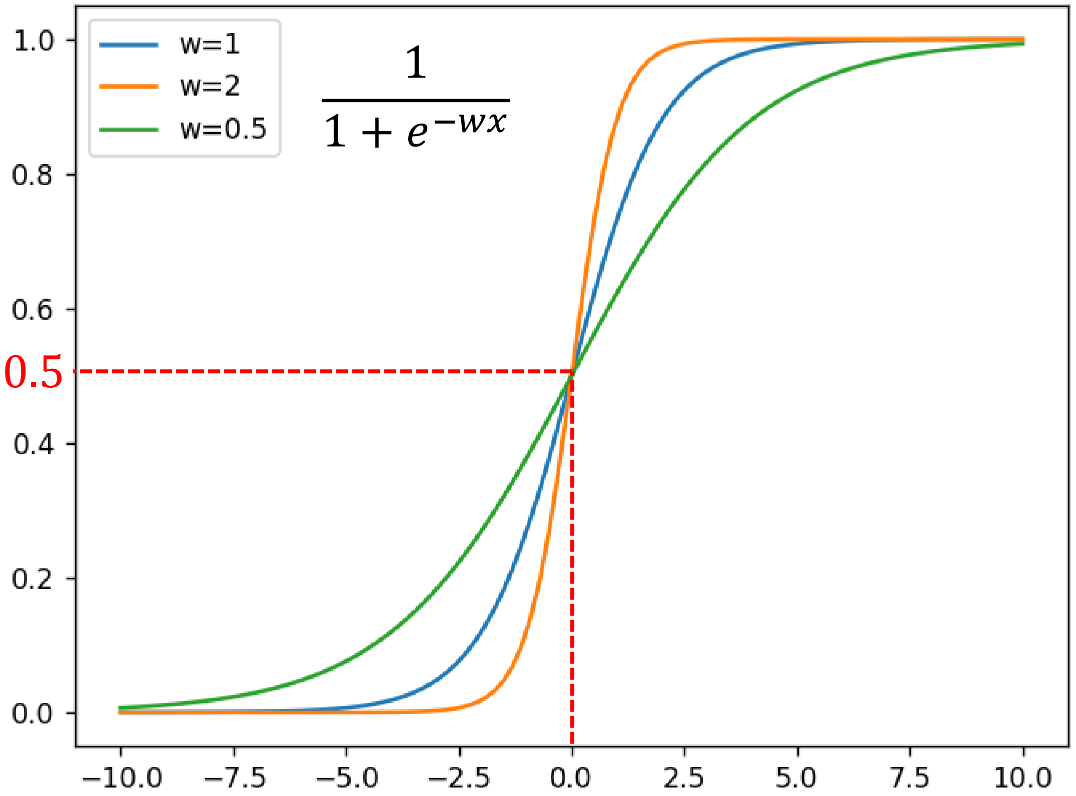
$$\{(x^{(i)}, y^{(i)})_{i=1}^{N}: \quad x^{(i)} \in \mathbb{R}^2, y^{(i)} \in \{0, 1\}\}$$

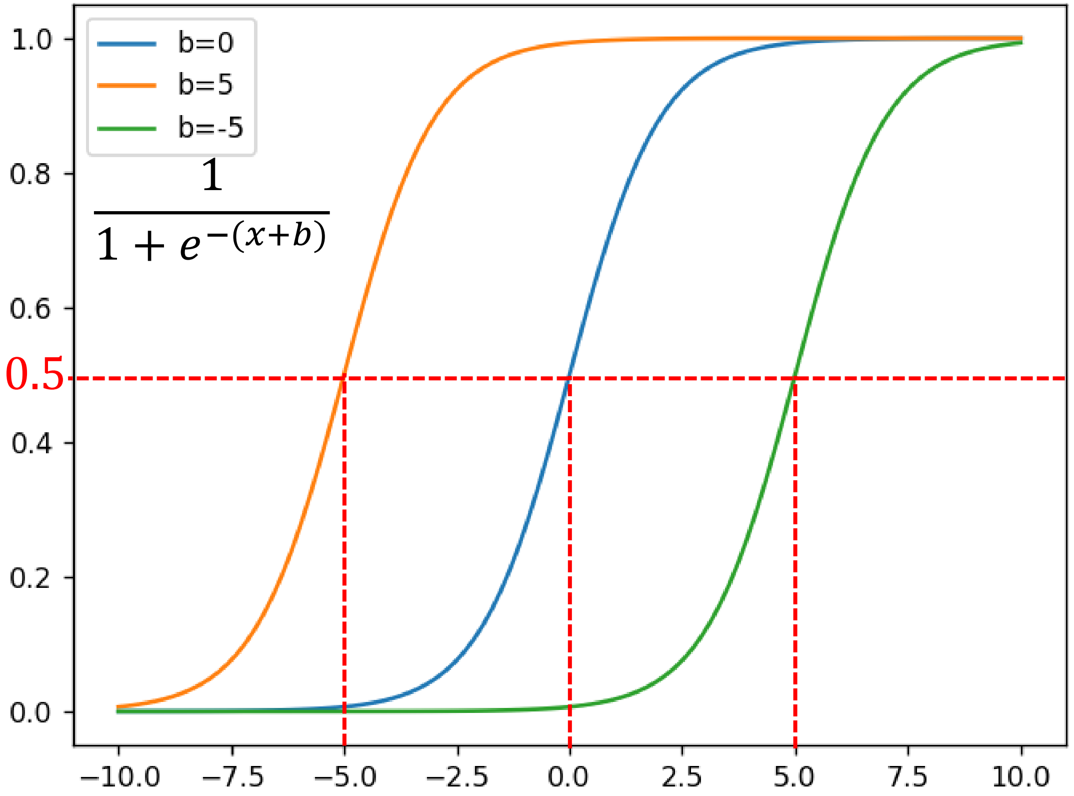


Hyperplane



Single Neuron for classification – Logistic function

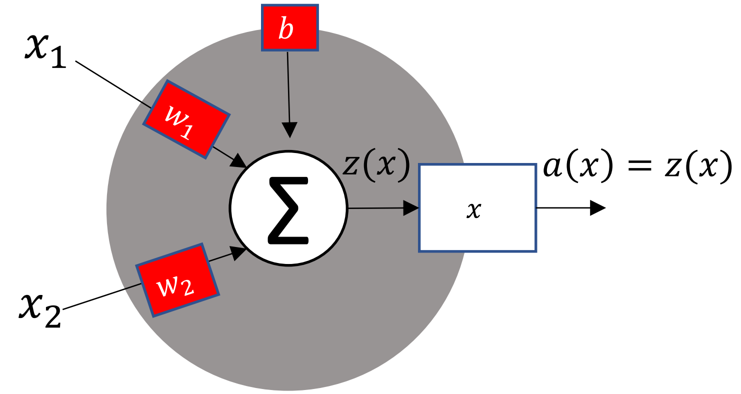




$$y(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

$$y'(x) = y(x)(1 - y(x))$$

Single Neuron for classification

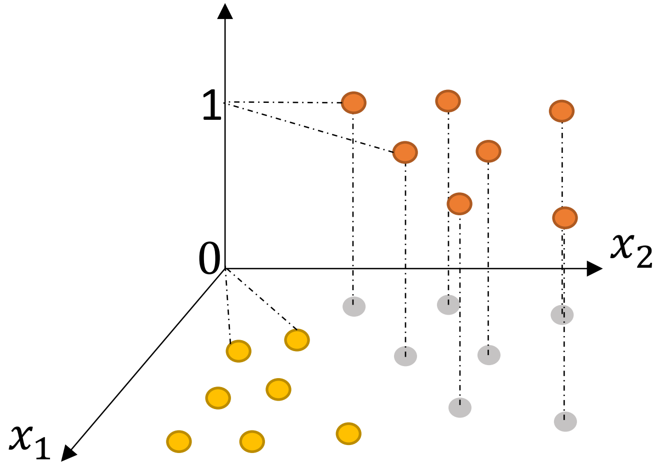


Apply Logistc Function to squash in 0-1

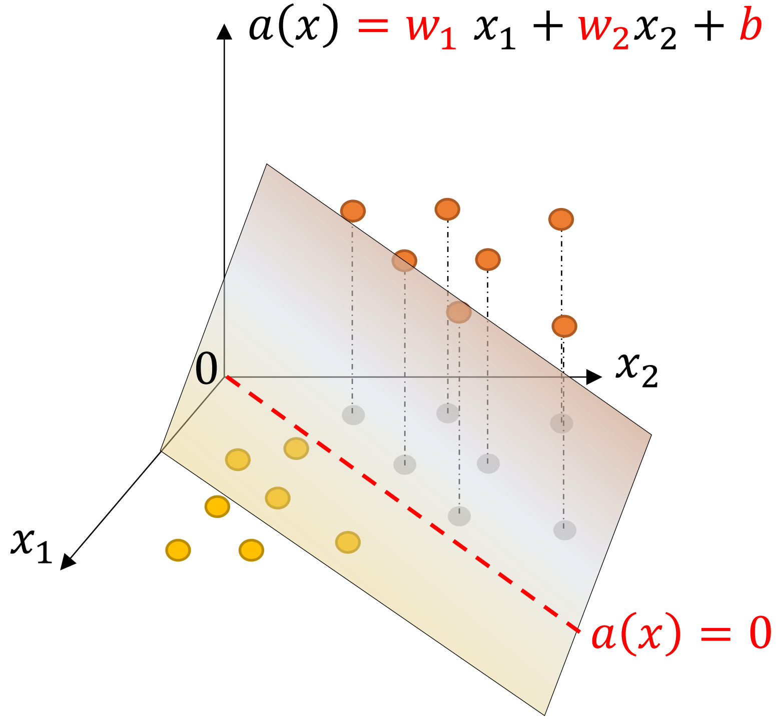
$$\hat{y}\left(x^{(i)}\right) = \frac{e^{a\left(x^{(i)}\right)}}{1 + e^{a\left(x^{(i)}\right)}} = P\left(Y = 1 \mid X = x^{(i)}\right)$$

Training Dataset

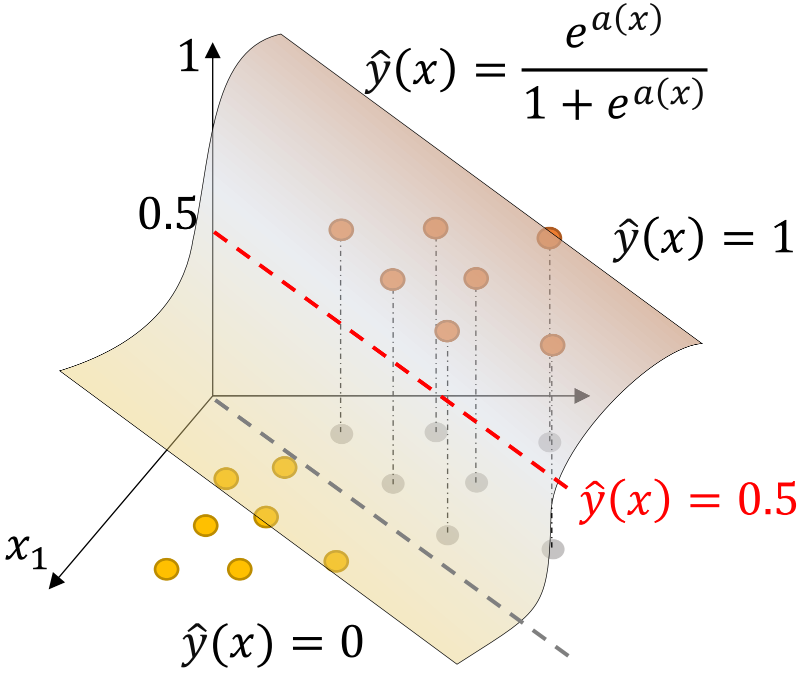
$$\left\{\left(x^{(i)}, y^{(i)}\right)_{i=1}^{N} : \quad x^{(i)} \in \mathbb{R}^2, y^{(i)} \in \{0, 1\}\right\}$$



Hyperplane



Squash in 0-1



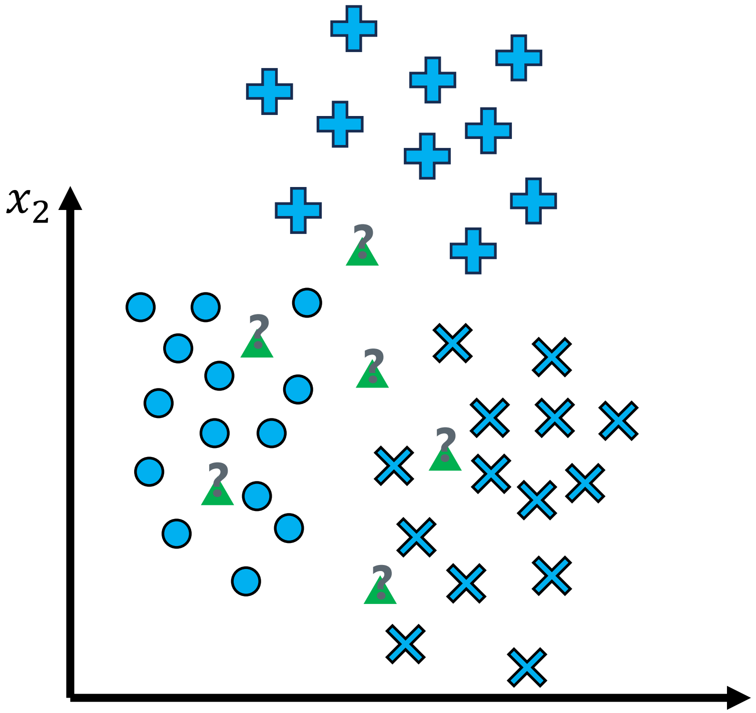
Multiclass Classification Problem

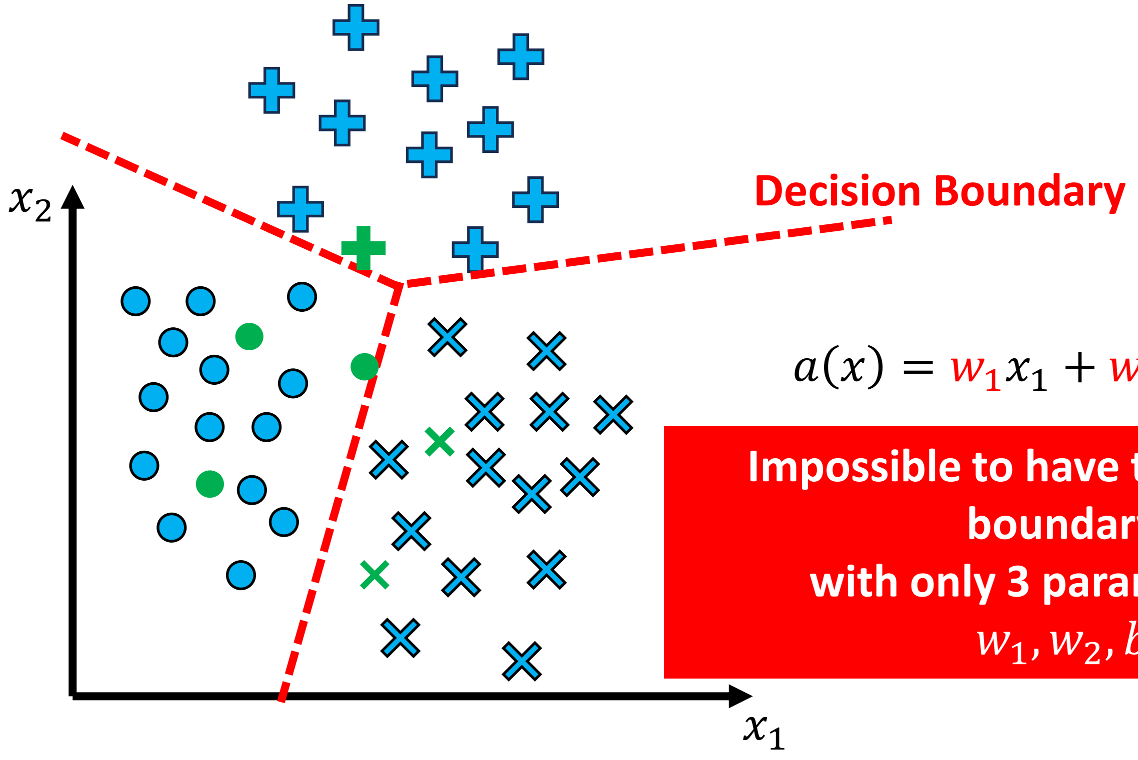
Given a training dataset

$$D_{train} = \{(x^{(i)}, y^{(i)})_{i=1}^{N_{train}} : \quad x^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{0, 1, \dots, C-1\}\}$$

$$D_{test} = \{x_{test}^{(1)}, x_{test}^{(2)}, \dots, x_{test}^{(N_t)}\}$$

What is the probability that a point $x_{test}^{(i)}$ belong to the class 1 ? $P(Y = 1 \mid X = x_{test}^{(i)})$?

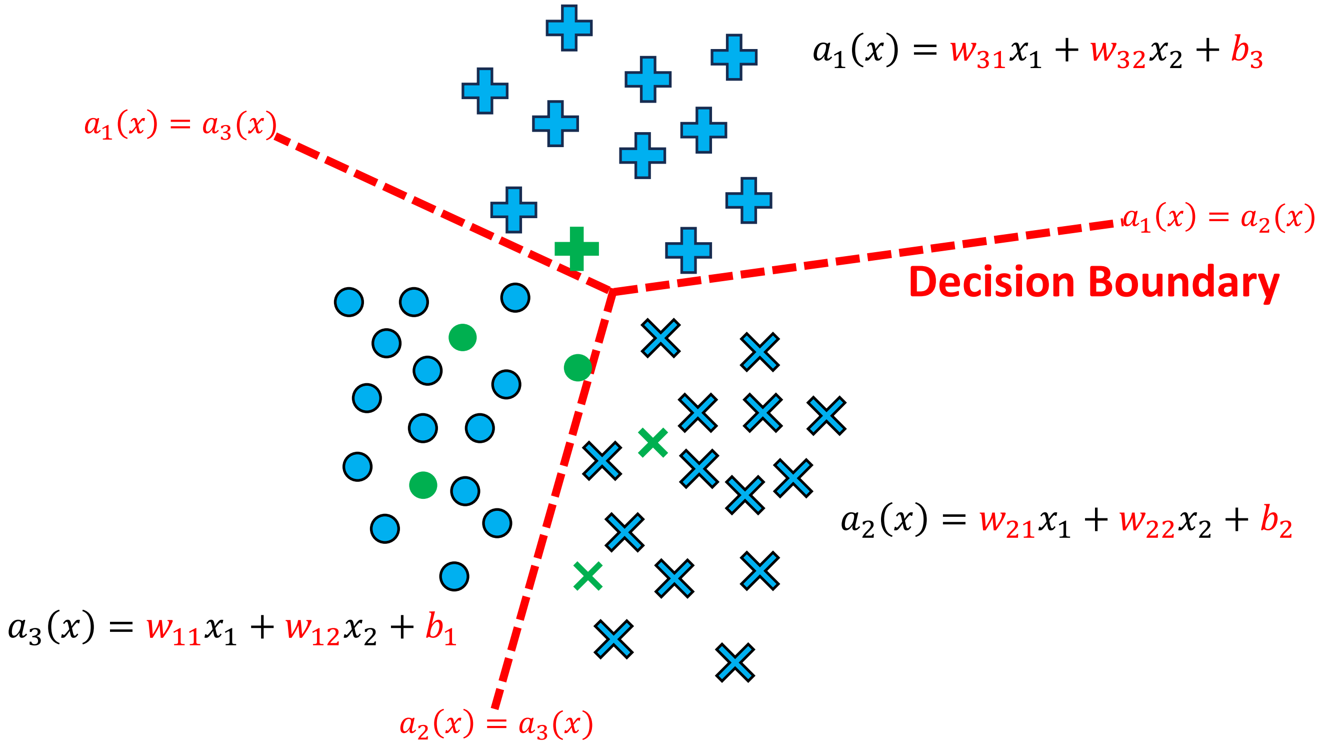


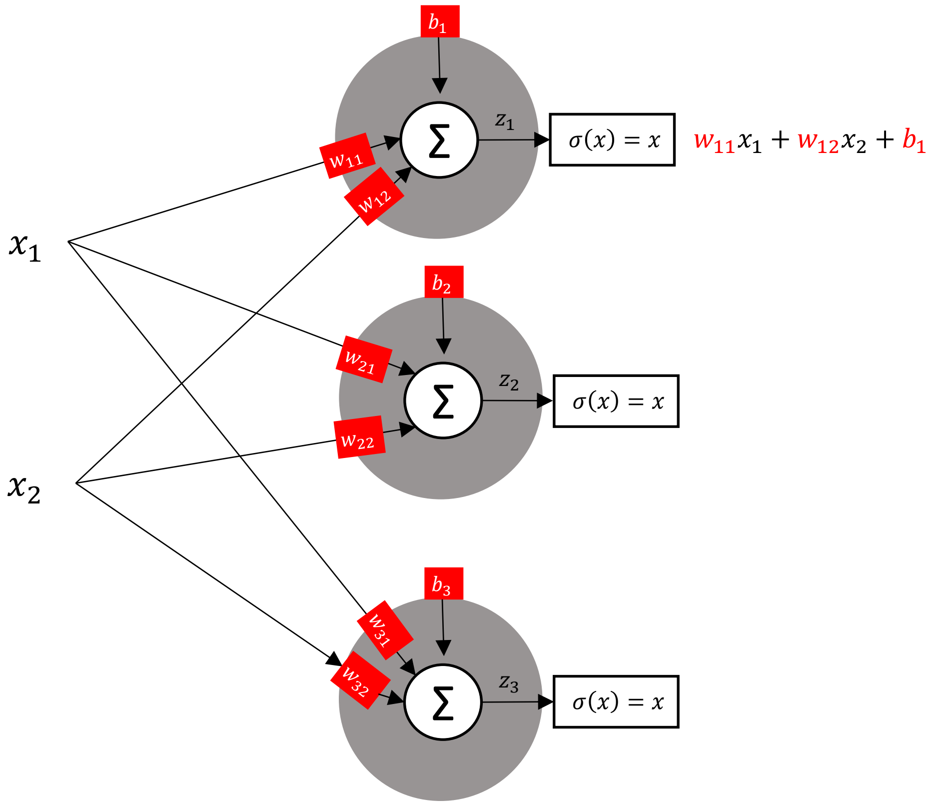


$a(x) = w_1 x_1 + w_2 x_2 + b = 0$

Impossible to have this decision boundary with only 3 parameters!!!
w_1, w_2, b

Multiclass Classification Problem

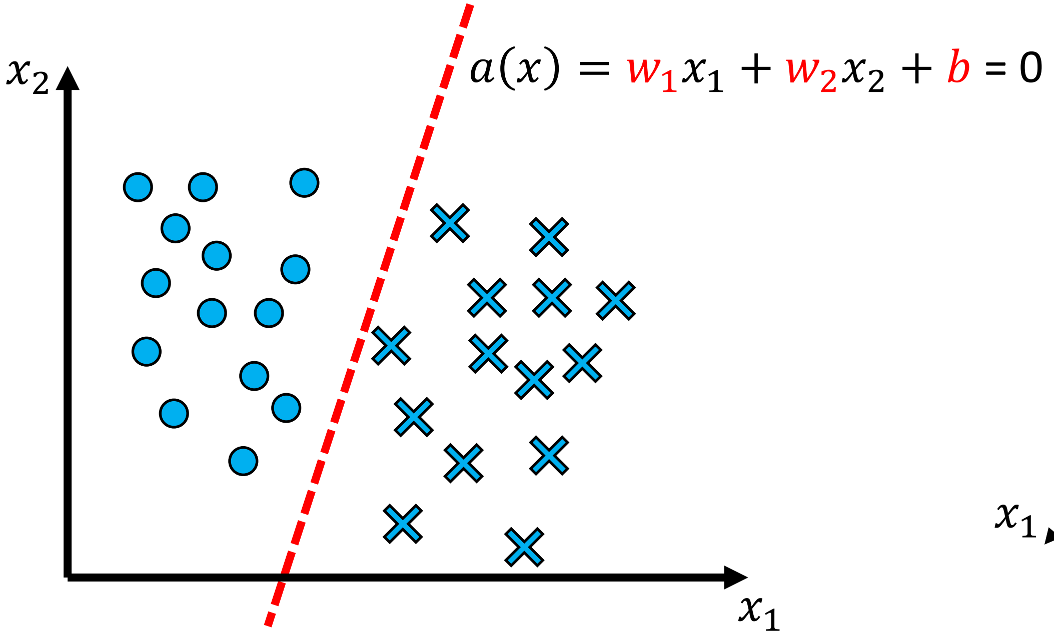


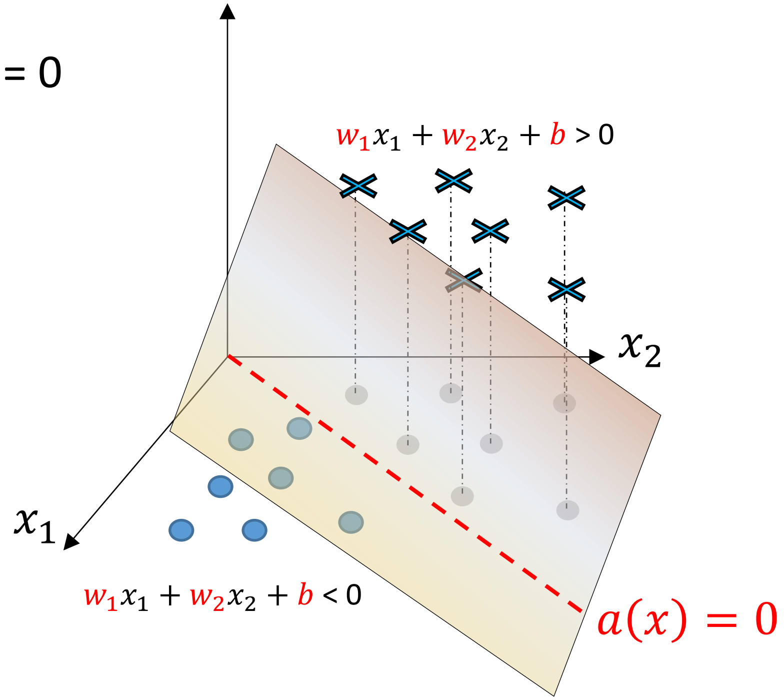


$$\hat{y}(x^{(i)}, W, b) = \text{softmax}(x^{(i)}) = \begin{bmatrix} \dfrac{e^{a_1(x^{(i)})}}{\sum_{j=1}^{3} e^{a_j(x^{(i)})}} \\ \dfrac{e^{a_2(x^{(i)})}}{\sum_{j=1}^{3} e^{a_j(x^{(i)})}} \\ \dfrac{e^{a_3(x^{(i)})}}{\sum_{j=1}^{3} e^{a_j(x^{(i)})}} \end{bmatrix} = \begin{bmatrix} P(Y = 0 | X = x^{(i)}; W, b) \\ P(Y = 1 | X = x^{(i)}; W, b) \\ P(Y = 2 | X = x^{(i)}; W, b) \end{bmatrix}$$

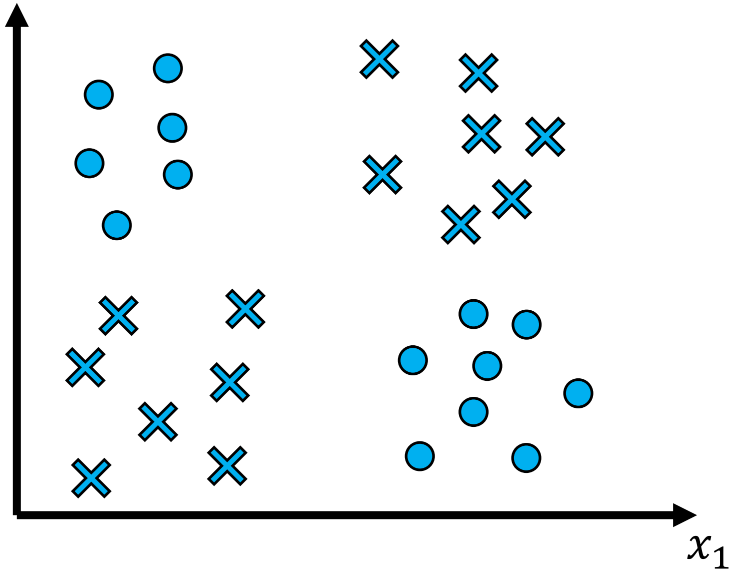
$$y_{pred}(x^{(i)}) = \text{argmax}\ \hat{y}(x^{(i)}, W, b) \in \{0,1,2\}$$

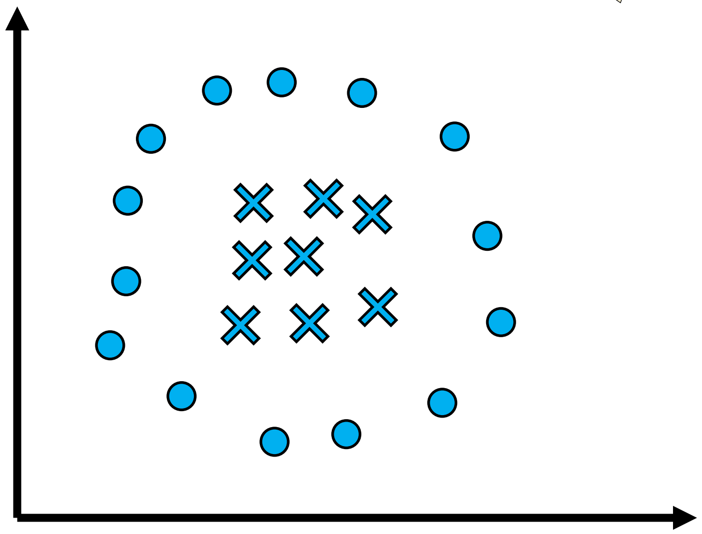
Linear Separability





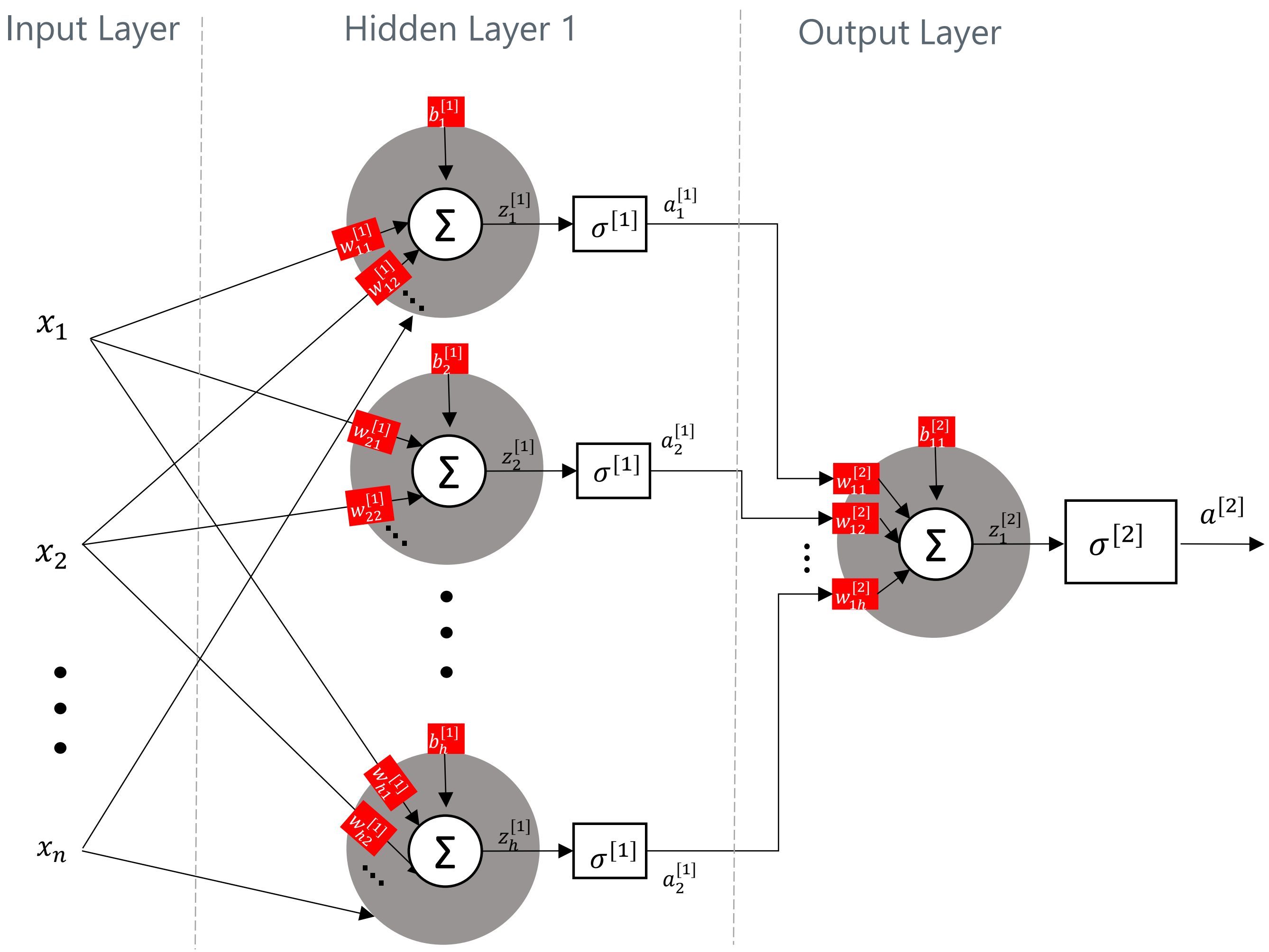
Linearly Separable data





NOT linearly Separable data!

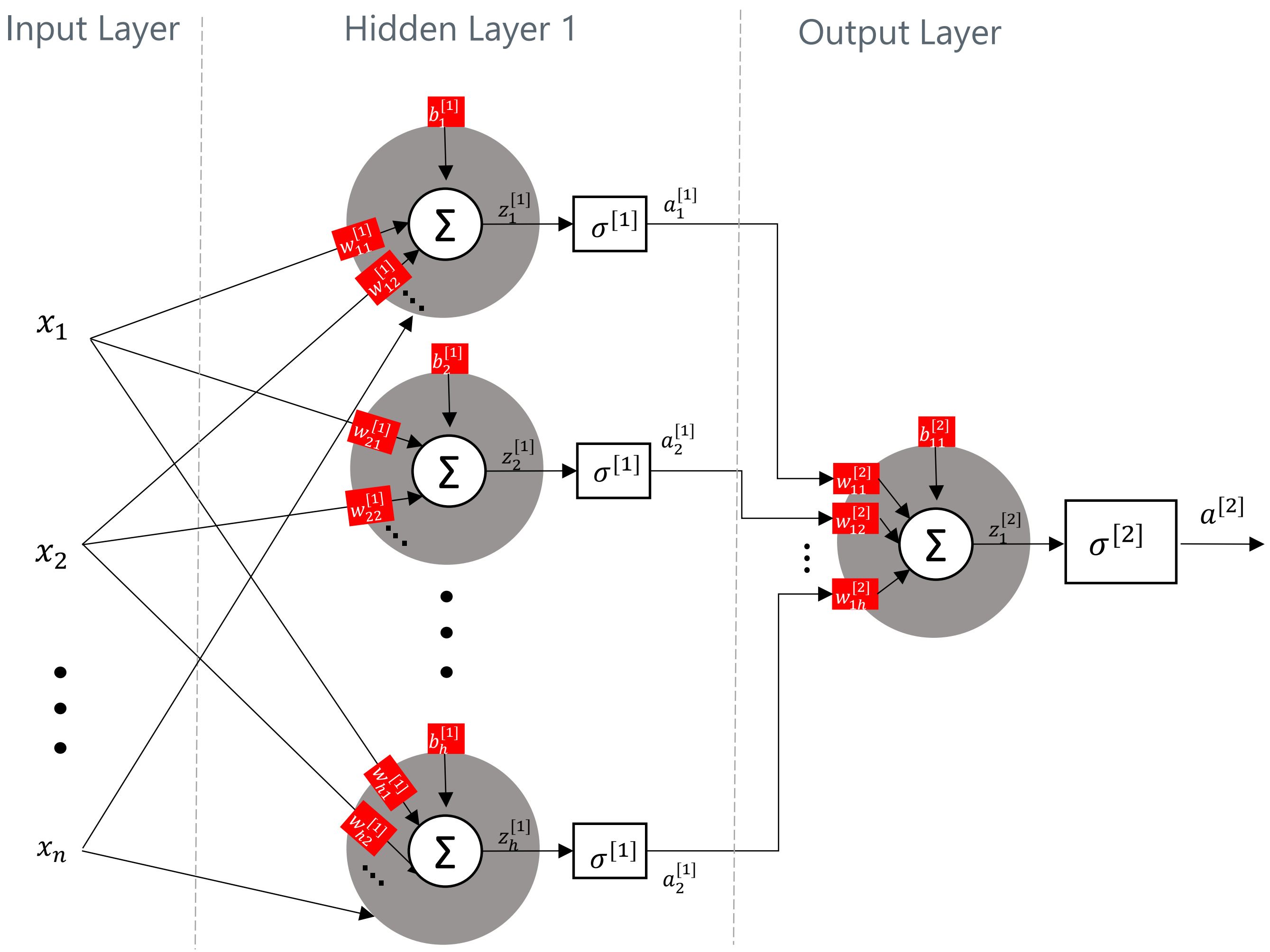
Single-Layer Perceptron



$$z_1^{[2]} = \left(\sum_{i=1}^{h} w_{1i}^{[2]} a_i^{[1]} \right) + b_1^{[2]}$$

- ✓ $a_i^{[1]}$ are Non-linear functions!
- ✓ $z_1^{[2]}$ is a linear combination of non-linear functions
- ✓ If h increases, then $z_1^{[2]}$ have more approximation power
- ✓ In the ideal case, $a^{[1]}(x) = [a_1^{[1]}\ a_2^{[1]}, \dots, a_h^{[1]}]$ is forced to be a new space, where the features are linearly separable!

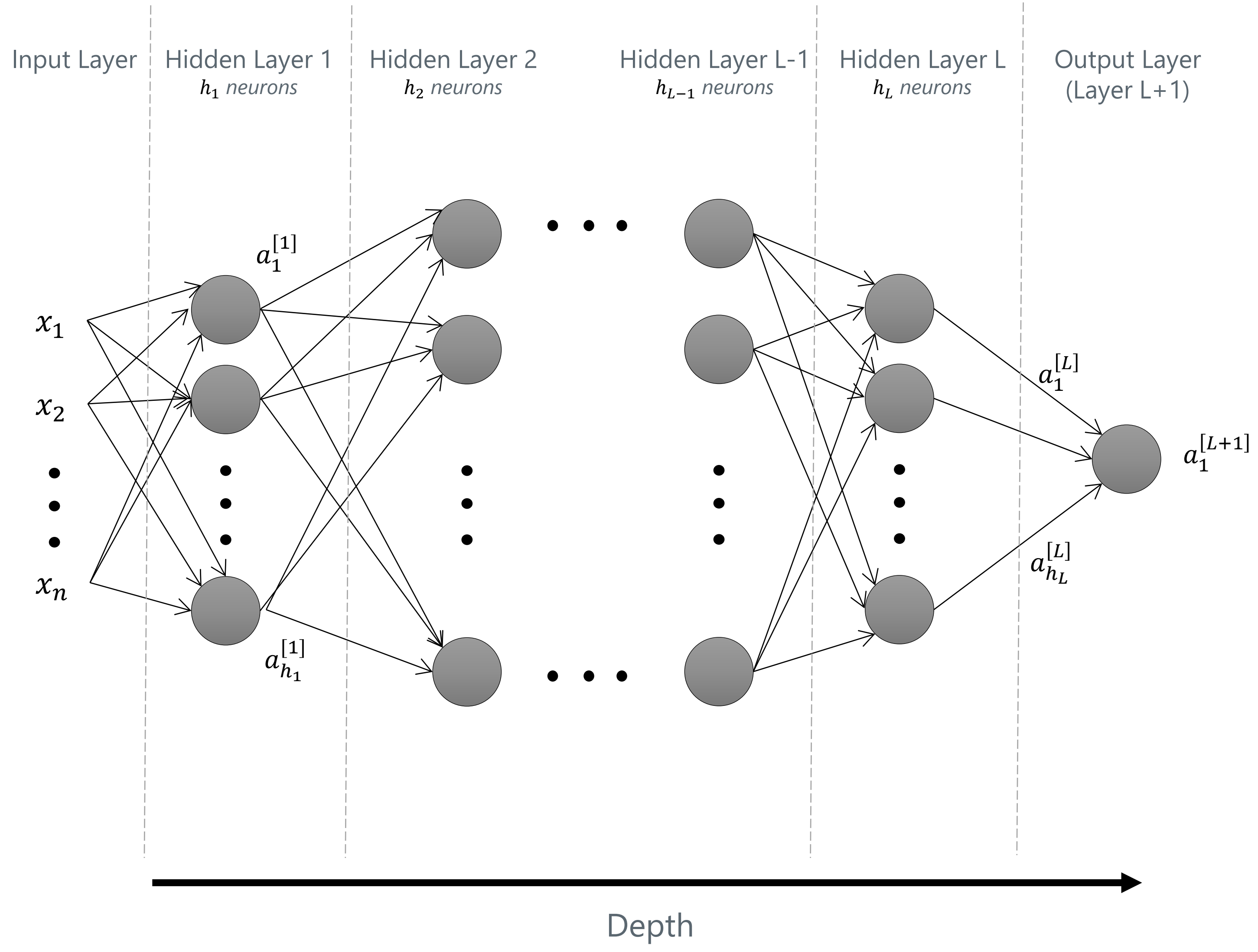
Single-Layer Perceptron



Input	$x = [x_1 \ x_2 \ \dots \ x_n] \in \mathbb{R}^n$
Output	$z_1^{[2]} = \left(\sum_{j=1}^{h} w_{1j}^{[2]} a_j^{[1]}\right) + b_1^{[2]} \quad a^{[2]} = \sigma^{[2]}(z_1^{[2]})$ $z_i^{[1]} = \left(\sum_{j=1}^{n} w_{ij}^{[1]} x_j\right) + b_i^{[1]} \quad a_i^{[1]} = \sigma^{[1]}\left(z_i^{[1]}\right)$
Weights or Parameters	$\begin{cases} W^{[1]} \in \mathbb{R}^{h \times n} & b^{[1]} \in \mathbb{R}^h \\ W^{[2]} \in \mathbb{R}^h & b^{[2]} \in \mathbb{R} \end{cases}$

Some Activation Functions: $\sigma(x)$	
Sigmoid	x
Hyperbolic Tangent	$\tanh(x)$
ReLU	$\begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$
LeakyReLU	$\begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$

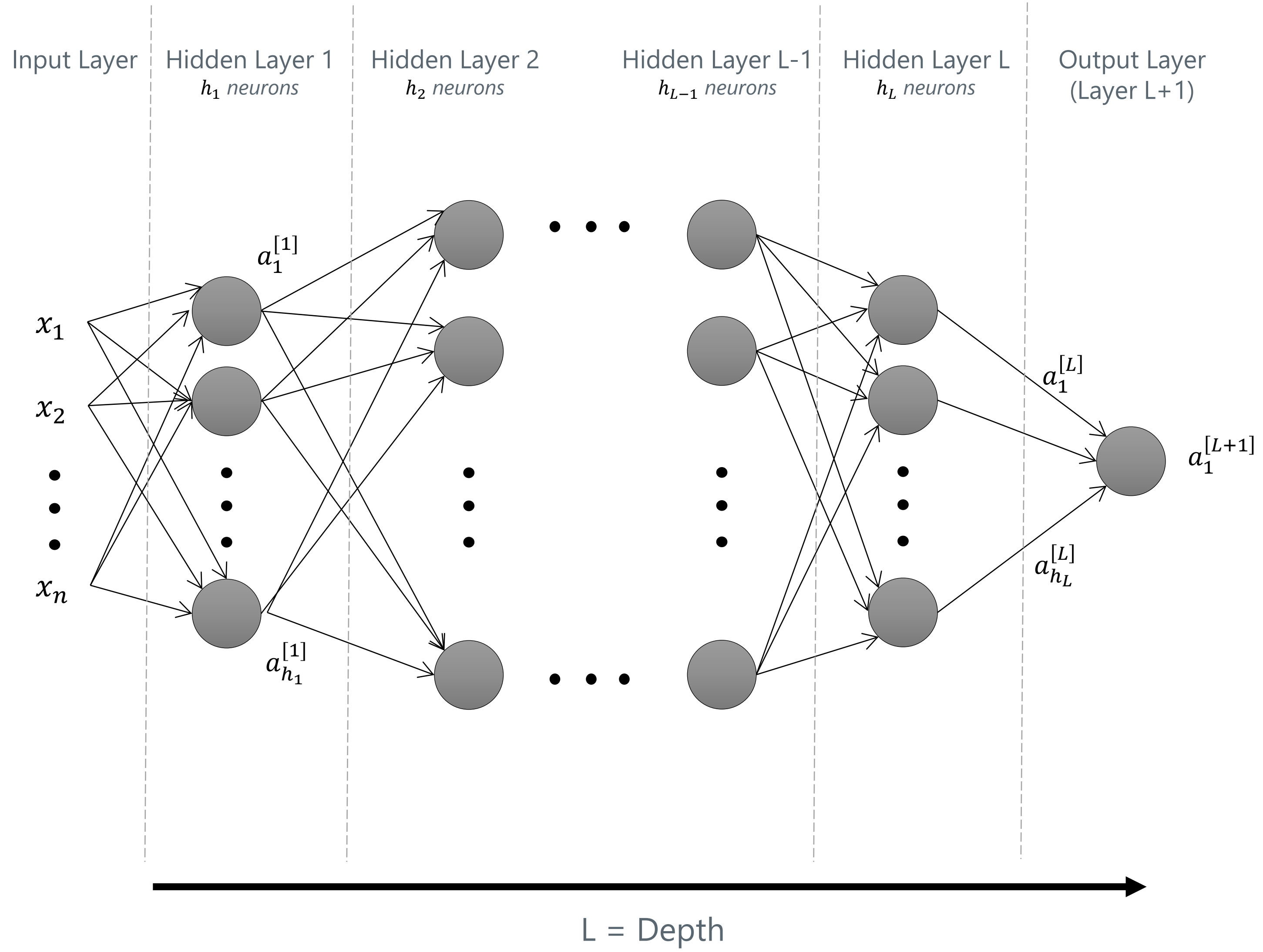
Multi-layer Perceptron (MLP)

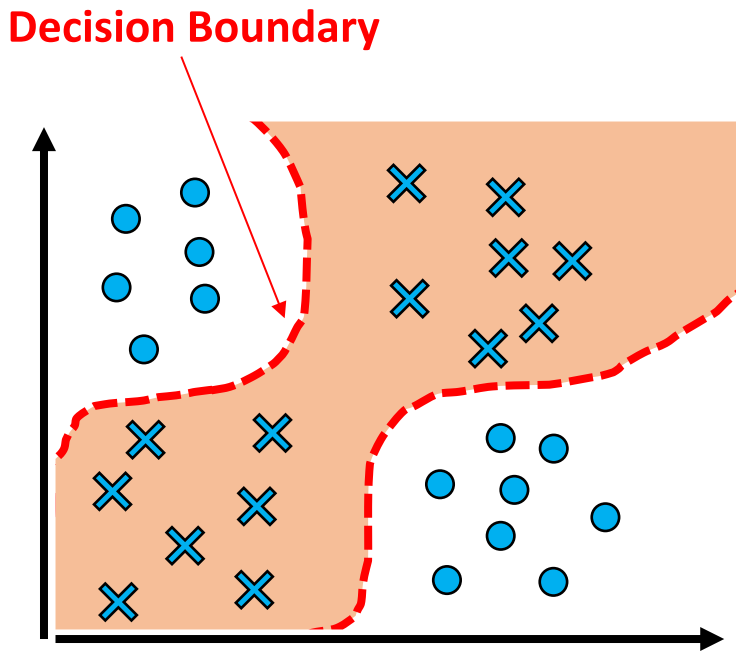


Input	$x = [x_1 \;\; x_2 \;\; \dots \;\; x_n] \in \mathbb{R}^n$
Output	$z_i^{[l]} = \left(\sum_{j=1}^{h_{l-1}} w_{ij}^{[l]} a_i^{[l-1]}\right) + b_i^{[l]} \quad a_i^{[l]} = \sigma^{[l]}(z_i^{[l]})$ $a_i^{[0]} = x_i$
Weights or Parameters	$W^{[L]} \in \mathbb{R}^{h_l \times h_{[l-1]}} \quad \boldsymbol{b}^{[l]} \in \mathbb{R}^{h_l}$ $l = 1, \dots, L$

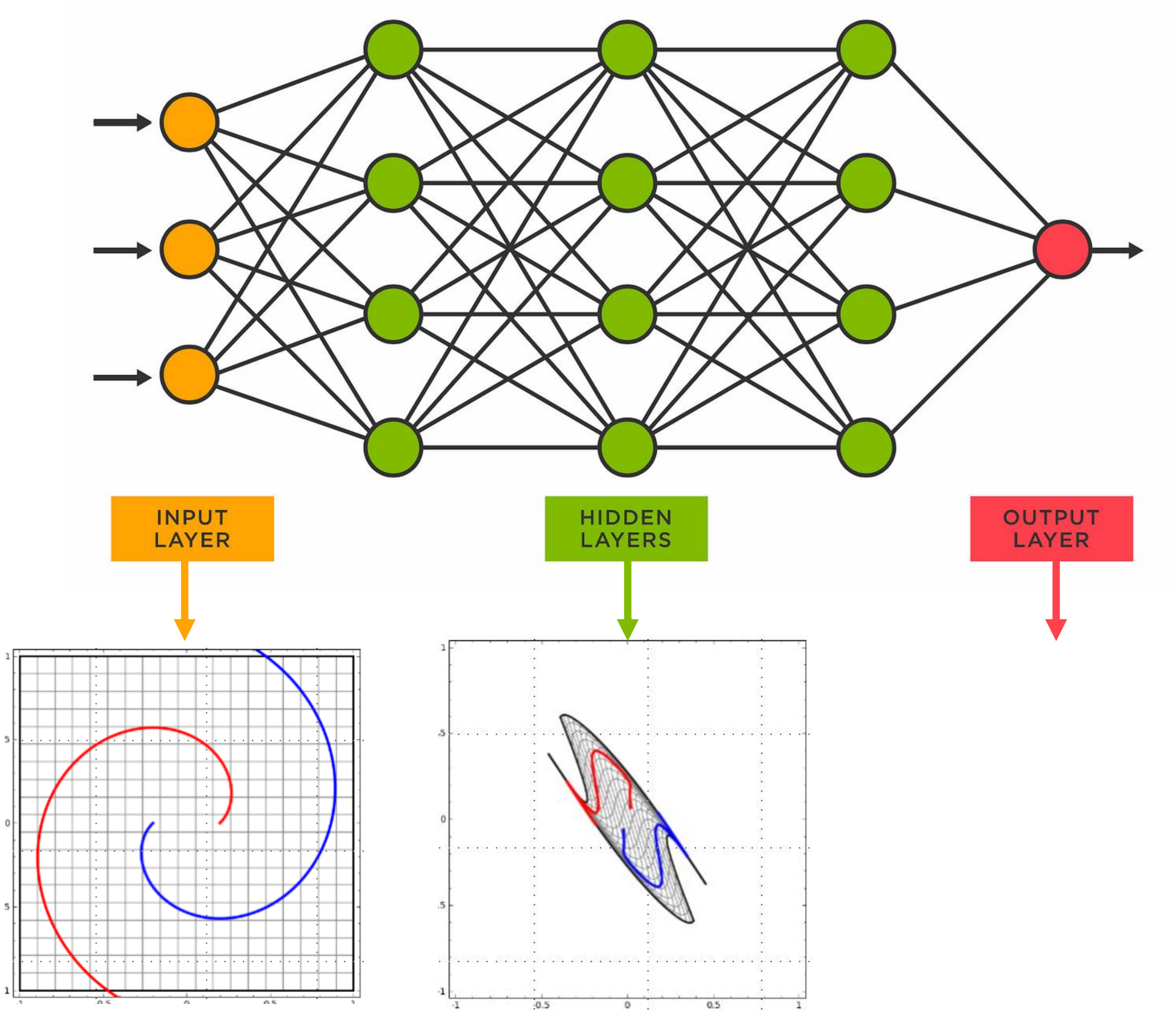
Some Activation Functions: $\sigma(x)$	
Sigmoid	$\frac{e^x}{1+e^x}$
Hyperbolic Tangent	$\tanh(x)$
ReLU	$\begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$
LeakyReLU	$\begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$

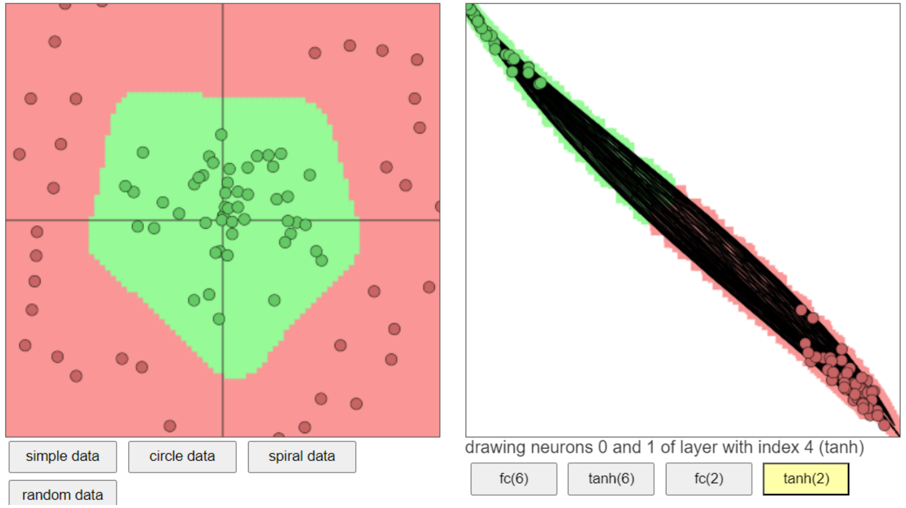
Multi-layer Perceptron (MLP)



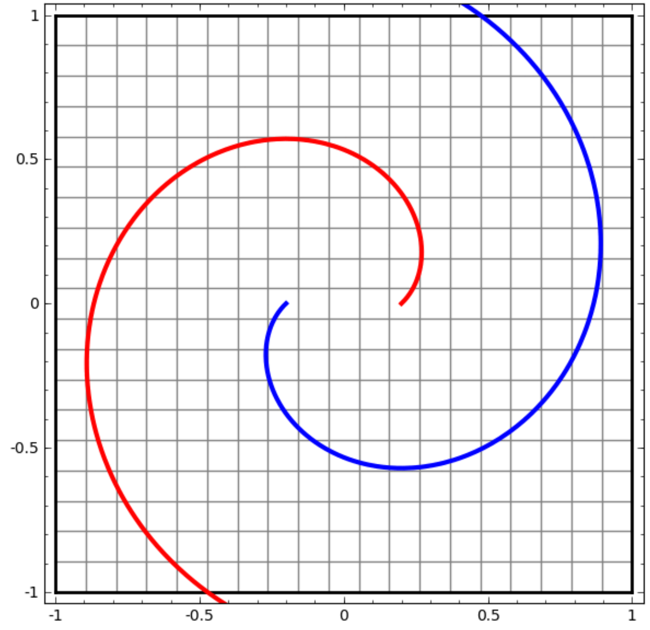


Multi-layer Perceptron (MLP)

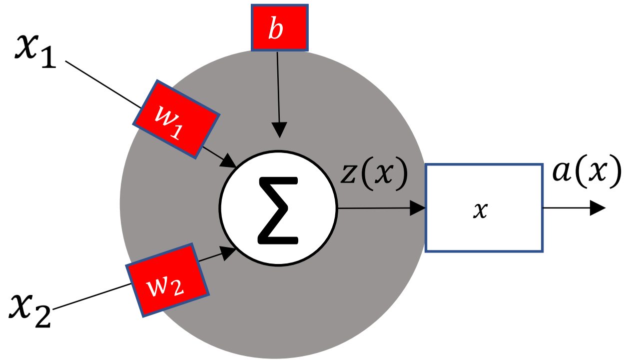




https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html



Single Neuron for classification – Loss



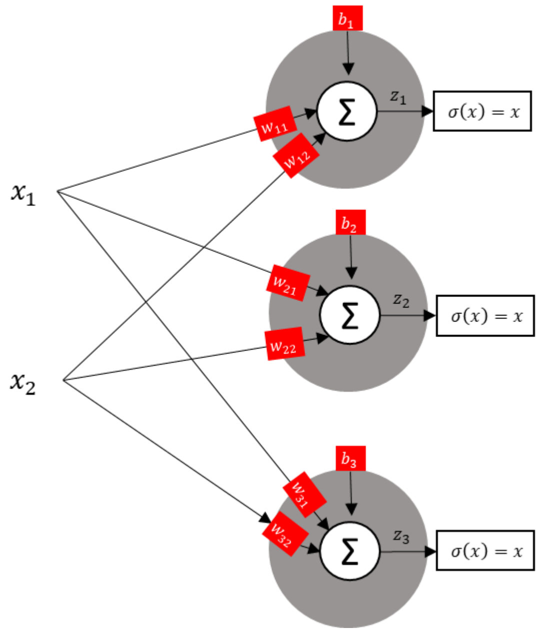
$$\hat{y}(x^{(i)}) = \frac{e^{a(x^{(i)})}}{1 + e^{a(x^{(i)})}} = \hat{y}(x^{(i)}, w, b) = \frac{e^{(w^T x^{(i)} + b)}}{1 + e^{(w^T x^{(i)} + b)}} = P(Y = 1 | X = x^{(i)}; w, b)$$

Binary Cross Entropy minimization

$$(\boldsymbol{w}, \boldsymbol{b})^* = \underset{\boldsymbol{w},\boldsymbol{b}}{\operatorname{argmin}}\, \boldsymbol{C}(\boldsymbol{w}, \boldsymbol{b}) = \underset{\boldsymbol{w},\boldsymbol{b}}{\operatorname{argmin}} \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{N}} \boldsymbol{\mathcal{L}}(\hat{\boldsymbol{y}}(\boldsymbol{x}^{(\boldsymbol{i})}, \boldsymbol{w}, \boldsymbol{b}), \boldsymbol{y}^{(\boldsymbol{i})})$$

$$\mathcal{L}(\hat{y}(x^{(i)}, w, b), y^{(i)}) = -y^{(i)} \log(\hat{y}(x^{(i)}, w, b) - (1 - y^{(i)}) \log(1 - \hat{y}(x^{(i)}, w, b))$$

Multi-class classification – Loss



$$\hat{y}\left(x^{(i)}, W, b\right) = softmax\left(x^{(i)}\right) = \begin{bmatrix} \dfrac{e^{a_1\left(x^{(i)}\right)}}{\sum_{j=1}^{3} e^{a_j\left(x^{(i)}\right)}} \\ \dfrac{e^{a_2\left(x^{(i)}\right)}}{\sum_{j=1}^{3} e^{a_j\left(x^{(i)}\right)}} \\ \dfrac{e^{a_3\left(x^{(i)}\right)}}{\sum_{j=1}^{3} e^{a_j\left(x^{(i)}\right)}} \end{bmatrix} = \begin{bmatrix} P\left(Y = 0 \middle| X = x^{(i)}; W, b\right) \\ P\left(Y = 1 \middle| X = x^{(i)}; W, b\right) \\ P\left(Y = 2 \middle| X = x^{(i)}; W, b\right) \end{bmatrix}$$

If $x^{(i)}$ belong to class 0:

$$y^{(i)} = [1,0,0]$$

If $x^{(i)}$ belong to class 1:

$$y^{(i)} = [0,1,0]$$

If $x^{(i)}$ belong to class 2:

$$y^{(i)} = [0,0,1]$$

Cross Entropy minimization

$$(\boldsymbol{W}, \boldsymbol{b})^* = \underset{\boldsymbol{w},\boldsymbol{b}}{\operatorname{argmin}}\, \boldsymbol{C}(\boldsymbol{W}, \boldsymbol{b}) = \underset{\boldsymbol{w},\boldsymbol{b}}{\operatorname{argmin}} \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{N}} \boldsymbol{\mathcal{L}}\left(\hat{\boldsymbol{y}}\left(\boldsymbol{x}^{(\boldsymbol{i})}, \boldsymbol{W}, \boldsymbol{b}\right), \boldsymbol{y}^{(\boldsymbol{i})}\right)$$

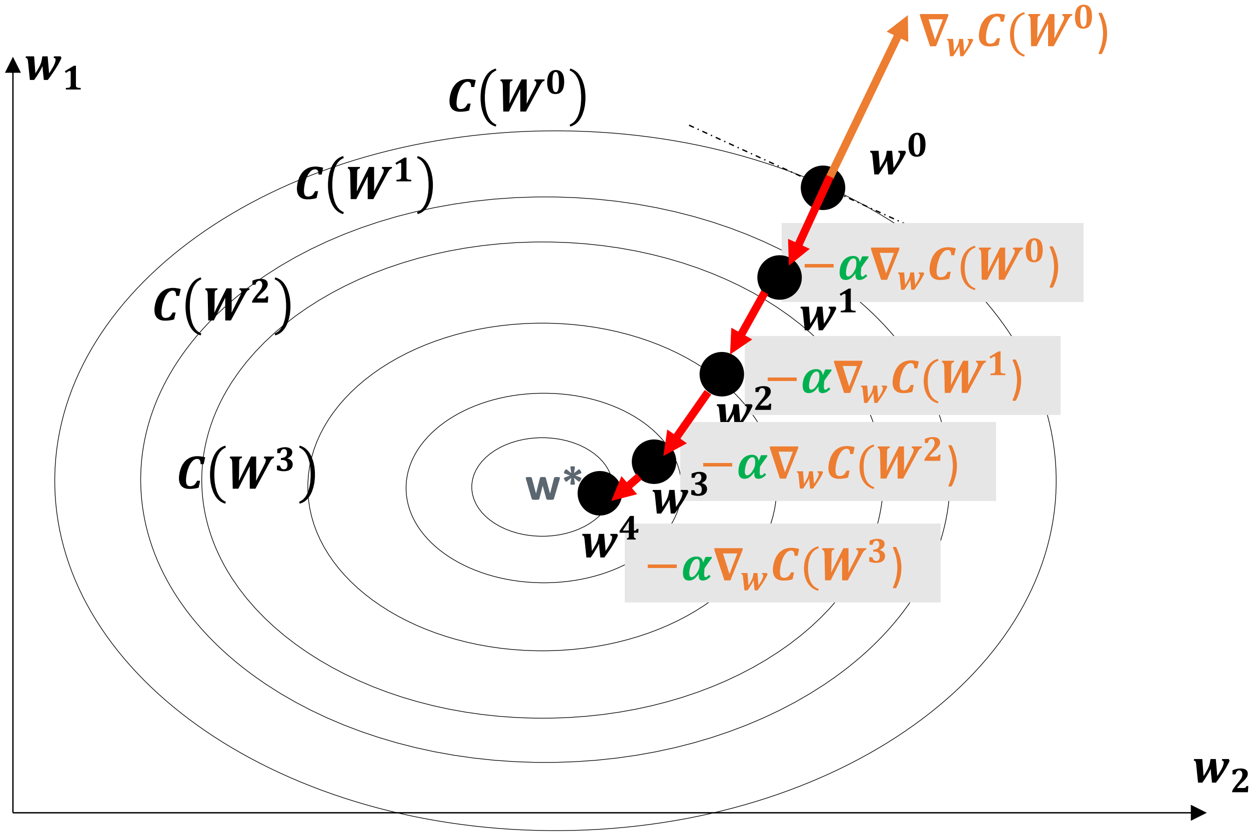
$$\mathcal{L}\left(\hat{y}\left(x^{(i)}, W, b\right), y^{(i)}\right) = -\sum_{j=1}^{C} y_j^{(i)} \log\left(\hat{y}\left(x^{(i)}, W, b\right)\right)$$

Optimizers

$$W = (w, b)$$

Without loss of generality suppose $b^* = 0$

$$W^* = \underset{w}{\operatorname{argmin}}\, C(W) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\big(\hat{y}(x^{(i)}, W), y^{(i)}\big)$$



Gradient Descent Algorithm

0.1 Choose learning rate α
0.2 Initilize weights $W = W^0$

For epoch=1..num_epochs do
1. Compute $\nabla_W C(W^k)$
2. Update weights

$$W = W - \alpha\nabla_W\, C(W^k)$$

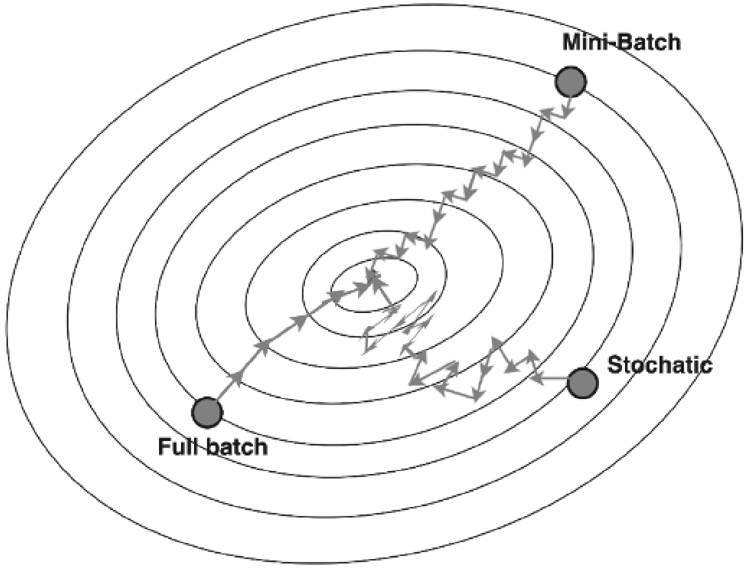
Optimizers – Batch Size

batch_size = N

Gradient Descent Algorithm

0.1. Choose learning rate α
0.2. Initilize weights $\boldsymbol{W}=\boldsymbol{W}^0$

for epoch=1..num_epochs do
  1. Compute $\nabla_{\boldsymbol{W}}\boldsymbol{C}(\boldsymbol{W})$ on the
  2. Update the weights
    $\boldsymbol{W} = \boldsymbol{W} - \alpha\nabla_{\boldsymbol{W}}\boldsymbol{C}(\boldsymbol{W})$



batch_size = 1

Stochastic Gradient Descent

0.1. Choose learning rate α
0.2. Initilize weights $\boldsymbol{W}=\boldsymbol{W}^0$

for epoch=1..num_epochs do
  1. shuffle the dataset
  for i=1..N do
    2. Compute $\nabla_{\boldsymbol{W}}\mathcal{L}_{\boldsymbol{W}}\left(\boldsymbol{y}^{(i)}, \boldsymbol{a}(\boldsymbol{x}^{(i)}, \boldsymbol{W})\right)$
    3. Update the weights
      $\boldsymbol{W} = \boldsymbol{W} - \alpha\nabla_{\boldsymbol{W}}\mathcal{L}_{\boldsymbol{W}}\left(\boldsymbol{y}^{(i)}, \boldsymbol{a}(\boldsymbol{x}^{(i)}, \boldsymbol{W})\right)$

batch_size = b, 1<b<N

Mini-batch Gradient Descent

0.1. Choose learning rate α
0.2. Choose the batch size b
0.3. Initilize weights $\boldsymbol{W}=\boldsymbol{W}^0$

for epoch=1..num_epochs do
  1. shuffle the dataset
  for k=0,…,floor(N/b) - 1
    2. Compute

$$\nabla_{\boldsymbol{W}}\boldsymbol{B}(\boldsymbol{W}) = \frac{\boldsymbol{1}}{\boldsymbol{b}}\sum_{\boldsymbol{i}=\boldsymbol{kb}}^{(\boldsymbol{k}+\boldsymbol{1})\boldsymbol{b}} \nabla_{\boldsymbol{W}}\mathcal{L}_{\boldsymbol{W}}\left(\boldsymbol{y}^{(i)}, \boldsymbol{a}(\boldsymbol{x}^{(i)}, \boldsymbol{W})\right)$$

    3. Update the weights
      $\boldsymbol{W} = \boldsymbol{W} - \alpha\nabla_{\boldsymbol{W}}\boldsymbol{B}(\boldsymbol{W})$

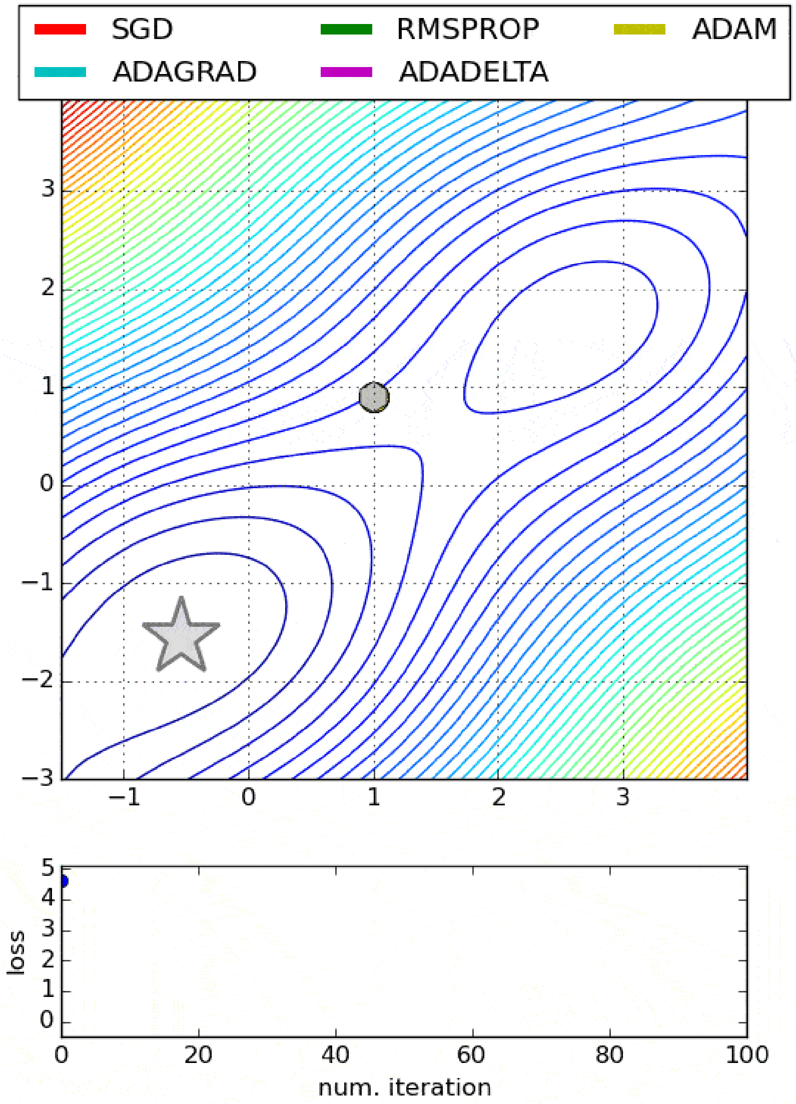
Optimizers – Algorithms

Basic Mini-Batch algorithms

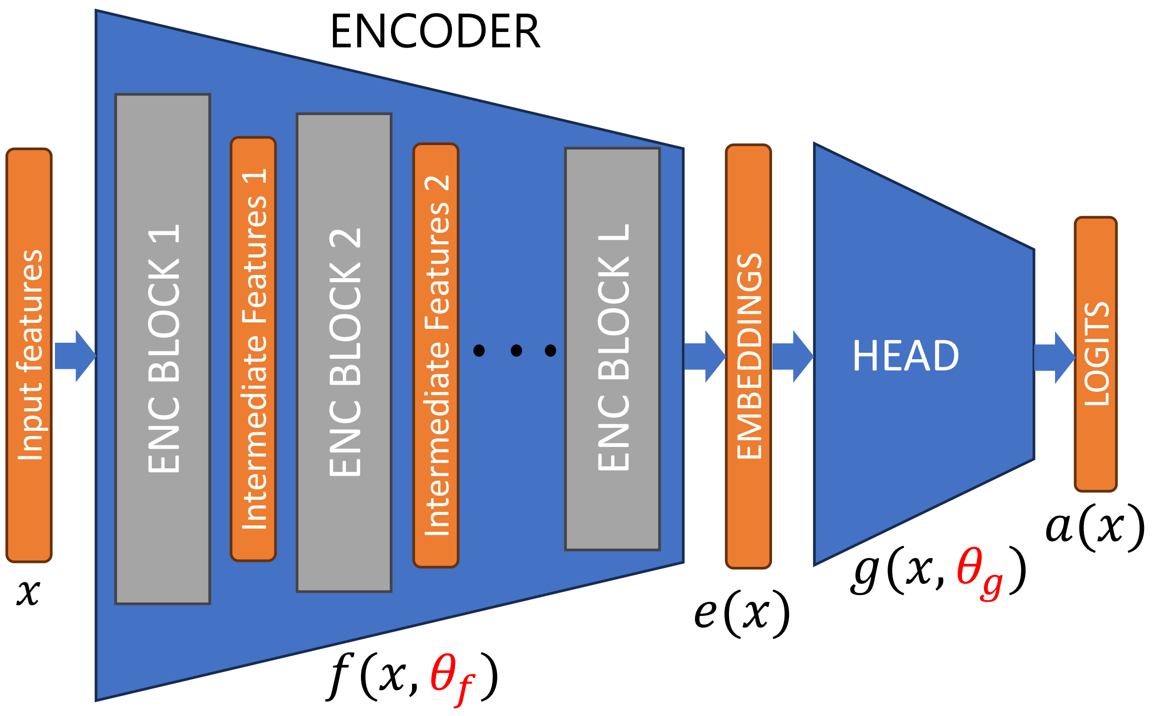
SGD: Stochastic Gradient Descent
SGD is a foundational algorithm that can sometimes outperform others, especially in certain problem domains. However, it can be slow to converge and may struggle with local minima. The momentum parameter can help improve convergence by smoothing out the updates

Adam
Adam is a popular optimization algorithm that combines the benefits of adaptive learning rates and momentum, making it an excellent starting choice for many applications

AdamW
AdamW is similar to Adam but decouples weight decay from the gradient updates, providing more effective regularization. This makes AdamW more suitable when weight regularization is desired, helping to improve generalization

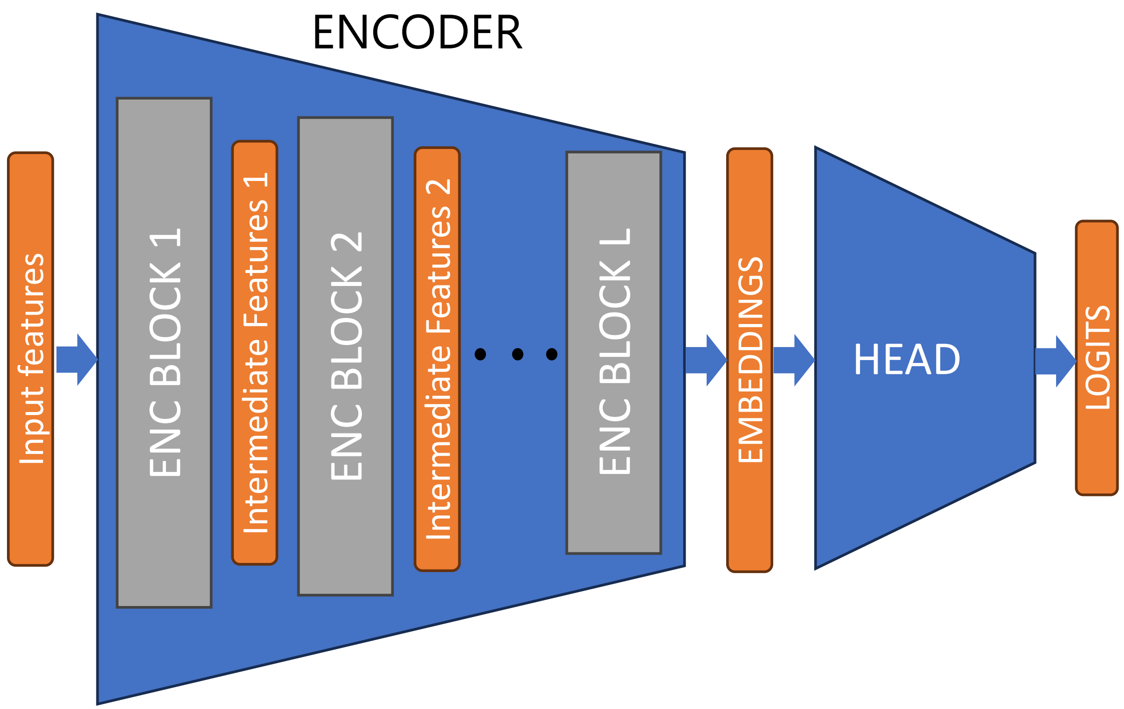


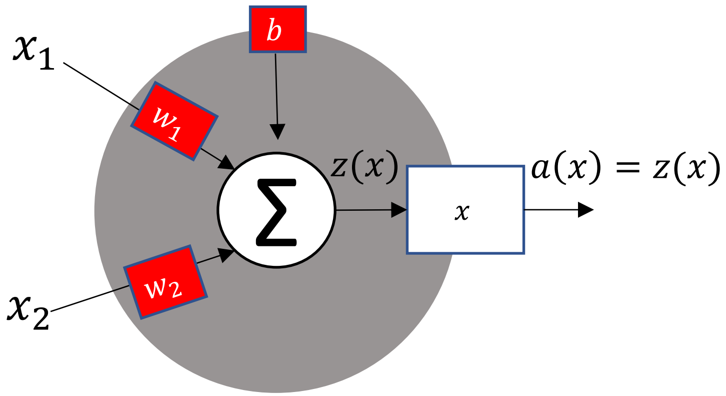
Anatomy of a Neural Network

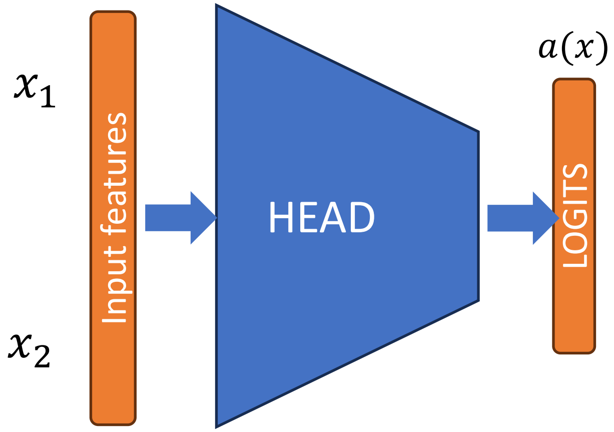


Embeddings (or high level features)	$e(x^{(i)}) = f(x^{(i)}, \theta_f)$
Logits	$a(x^{(i)}) = g(e(x^{(i)}), \theta_g)$
Output	$\hat{y}(x^{(i)}) = \sigma(a(x^{(i)}))$
Parameters/Weights	θ_f, θ_g θ_f:All weights and biases of the encoder θ_g:All weights and biases of the head

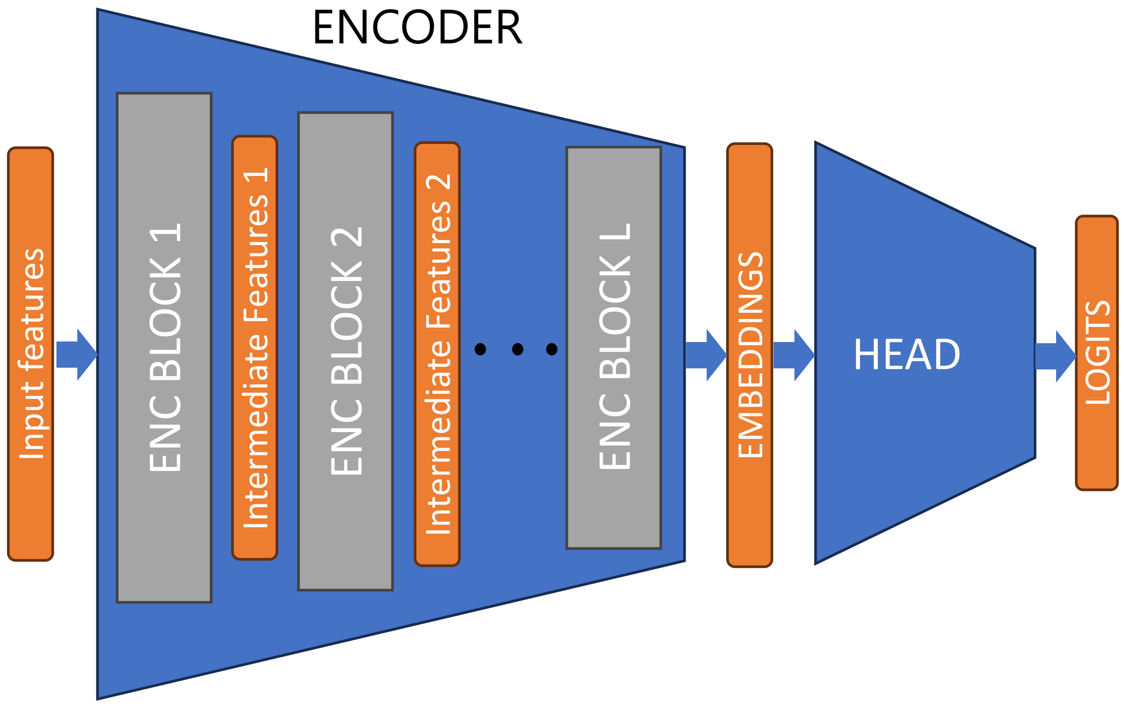
Anatomy of a Neural Network – A single neuron

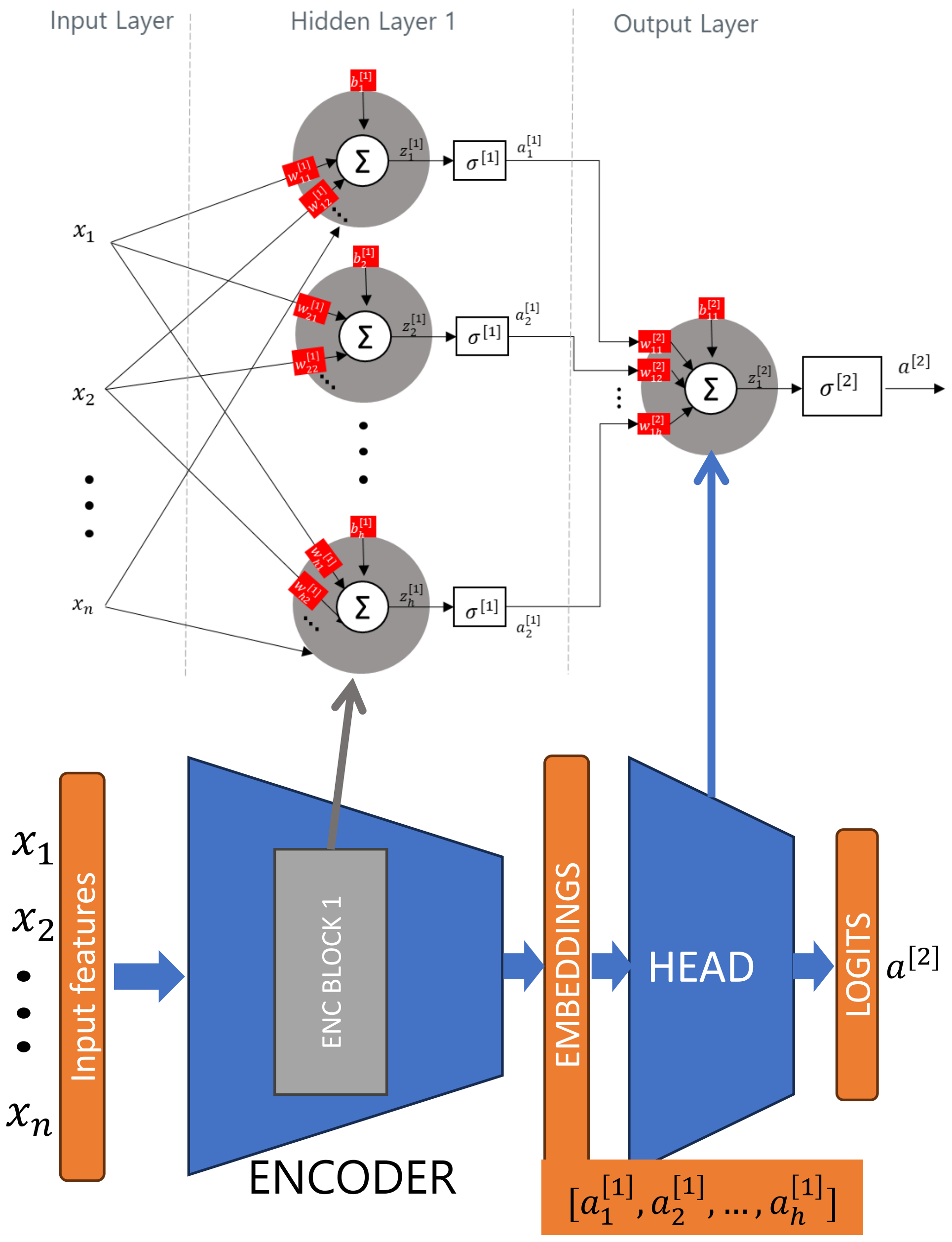




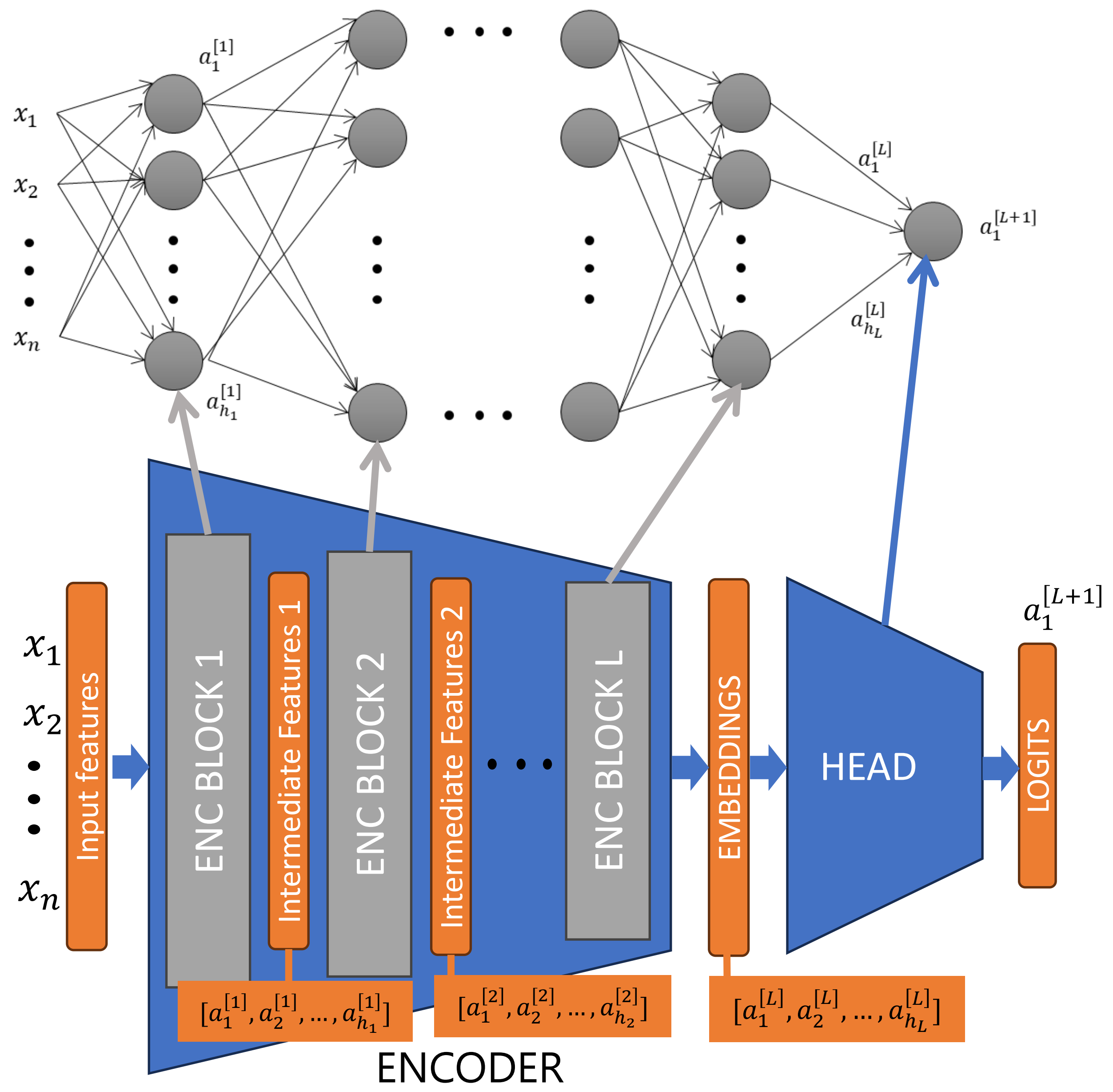


Anatomy of a Neural Network – Single-Layer Perceptron

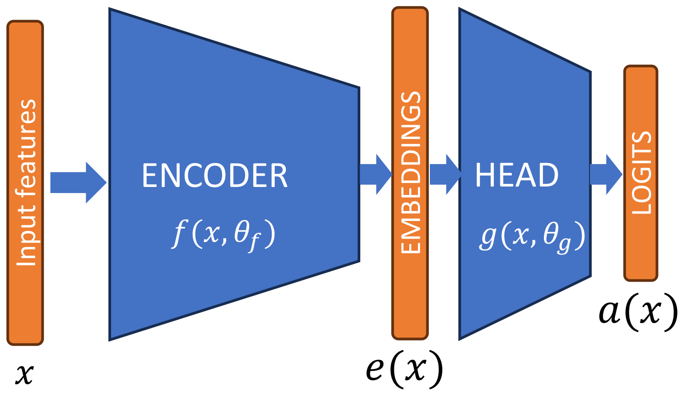




Anatomy of a Neural Network – Multi-Layer Perceptron



Classification Summary



Number of classes C	Dimensions	Output function $\hat{y}(x)$	Loss function
Binary Classification (C=2)	$a(x) \in \mathbb{R}$ $\hat{y}(x) \in [0,1]$ $y(x) \in \{0,1\}$	$\hat{y}$(x) = logistic(x)	Binary Cross Entropy
Multi-class classification (C>2)	$a(x) \in \mathbb{R}^C$ $\hat{y}(x) \in \mathbb{S}_1^C(0)$ (unit sphere) $y(x) \in \{\mathbf{1}^1, \mathbf{1}^2, \dots, \mathbf{1}^C\}$ $\mathbf{1}^i = 1 \text{ in i−th, 0 otherwise}$	$\hat{y}$(x) = softmax(x)	Cross Entropy

Logistic function / sigmoid

$$y(x^{(i)}) = \frac{e^{x^{(i)}}}{1+e^{x^{(i)}}} = \frac{1}{1+e^{-x^{(i)}}}$$

Binary Cross Entropy (BCE) Loss:

$$\mathcal{L}(\hat{y}(x^{(i)}), y^{(i)}) = -y^{(i)} \log(\hat{y}(x^{(i)}) - (1-y^{(i)}) \log(1-\hat{y}(x^{(i)}))$$

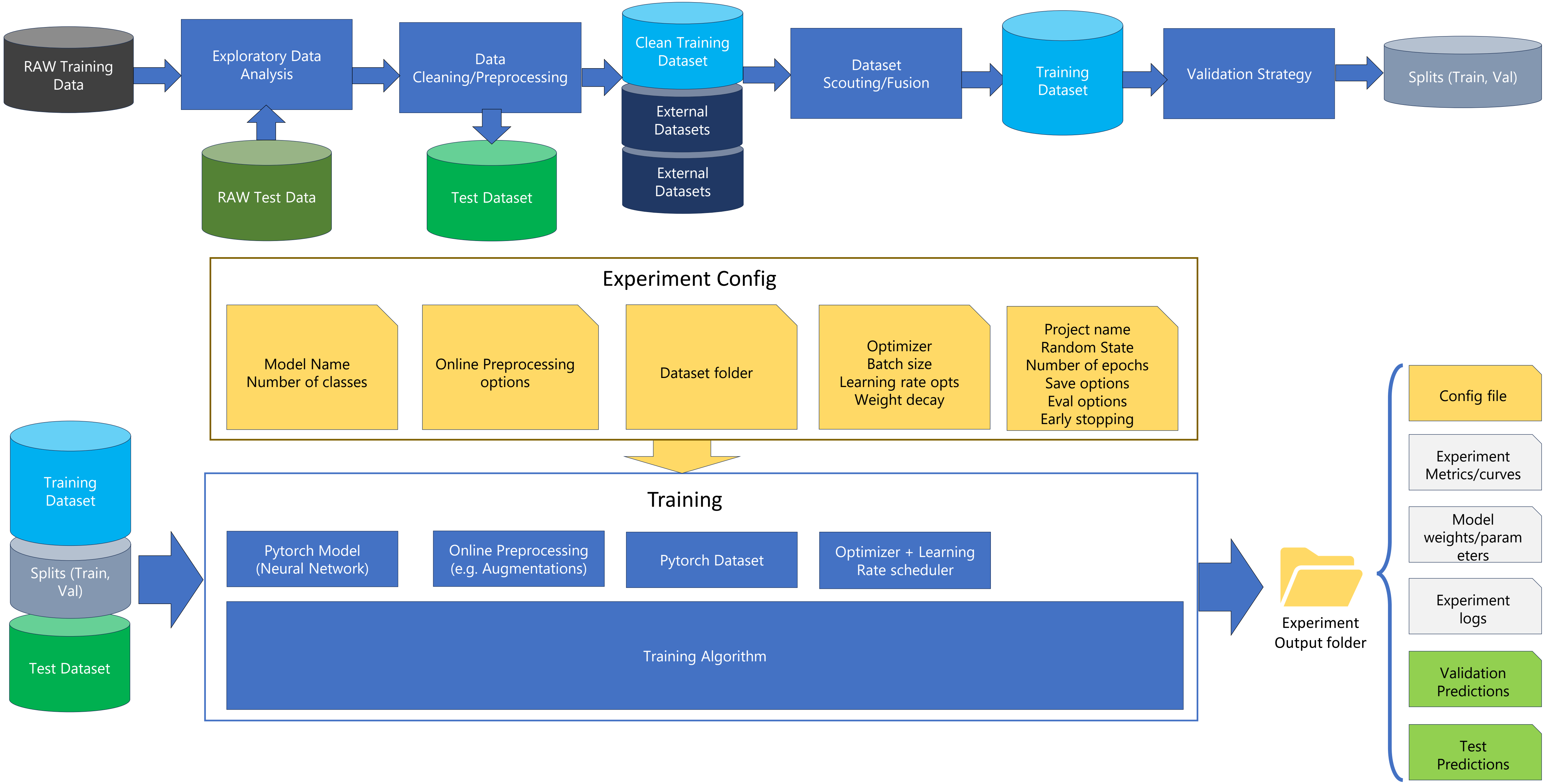
Softmax

$$y(x^{(i)}) = \text{softmax}(\mathrm{x}^{(i)}) = \left[\frac{e^{a_1(x^{(i)})}}{\sum_{i=1}^{C} e^{a_j(x^{(i)})}}, \dots, \frac{e^{a_C(x^{(i)})}}{\sum_{i=1}^{C} e^{a_j(x^{(i)})}}\right]$$

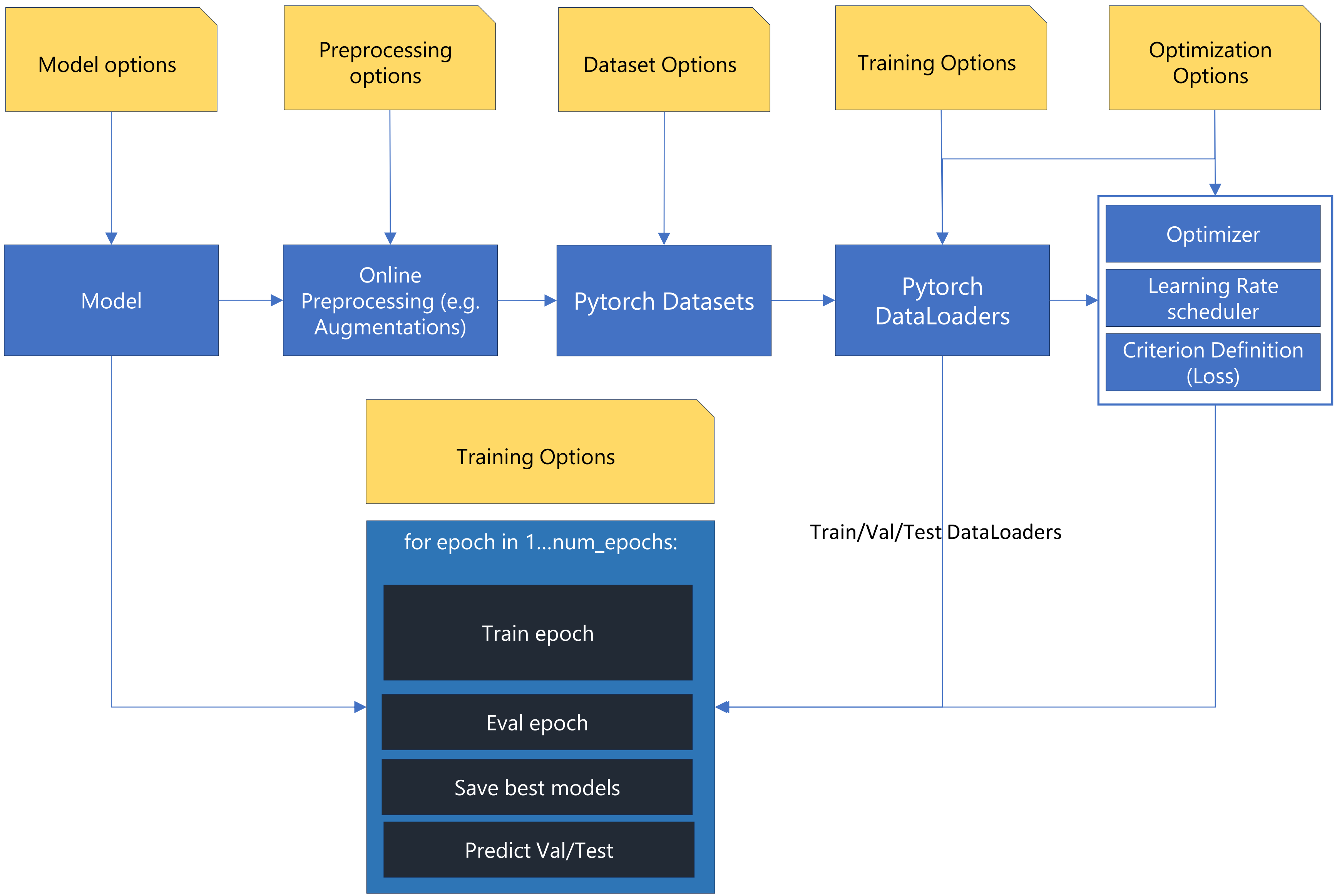
Cross Entropy (CE) Loss:

$$\mathcal{L}(\hat{y}(x^{(i)}), y^{(i)}) = -\sum_{j=1}^{C} y_j^{(i)} \log(\hat{y}(x^{(i)}))$$

Training recipe



Classification Summary



The challenge!

cat → Prompt Generation →

```
Though blind might only
know a presence without
the sighted world,
there, delicate, nimble
whiskered sentinella
trodd gaily along
woodland peripheries
shadowed gregariously
verdulous, its
tessellated pattern like
the underbrush's
scattered ponderosa
```

→ 

Description of the training features

- **image_name :** the file name of the image generated from an unknown text-2-image generator from the relative **description**
- **object**: categorical value in {**cat, dog, bus, car, person, house}**
- **description:** the "scene description", generated from an unknown LLM

- **target**: the unknown context to be predicted, there are only 4 different contexts (from 0 to 3)

object → Prompt Generation
target → Prompt Generation

Prompt Generation → prompt to get the description of the object in a unknown context (target) → Text Generation → description → Image Generation → 

image_name

The challenge!

- **End Date:** Friday 06th at 15:00
- **Private Leaderboard:** Friday 06th at 15:00!
- **Award and Presentations**: Friday 06th at 16:00

- **Maximum Daily submission limit:** 12
- **Scored private submissions:** 2

- **Team:** Groups should be of 4 people.
  Groups of 3 or 5 are also accepted, but no exception will be granted for other sizes

Thank you for your attention!