ANNs(Artificial Neural network)

ANNs(Artificial Neural network)

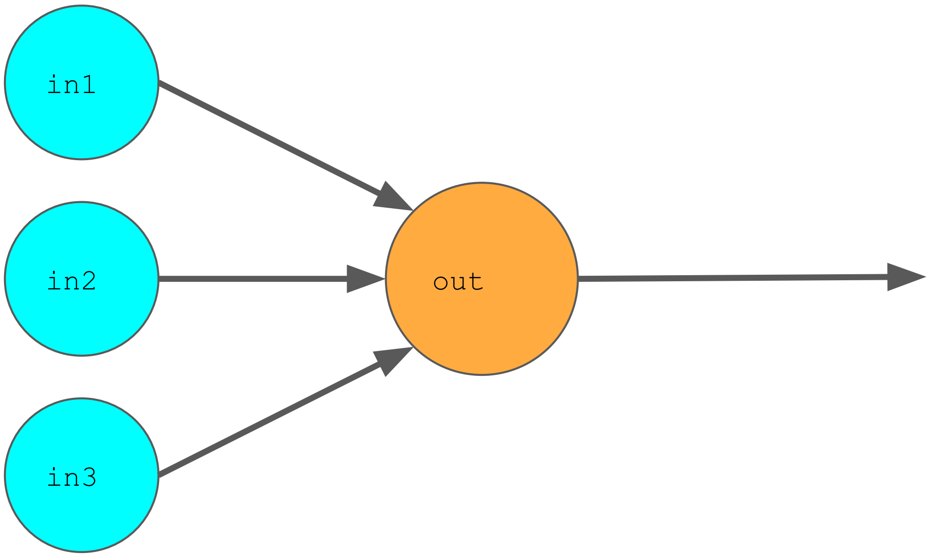
In this Lecture, you will learn about

- The basic architecture of an ANN (artificial neural network).
- The linear and nonlinear components of an artificial neural network.
- What the terms “feature space” and “separating hyperplane” mean.
- More about biases, weights, and activation functions.
- Different categories of errors, and their corresponding loss functions.
- The difference between loss and cost.
- How the gradient descent algorithm is extended to DL

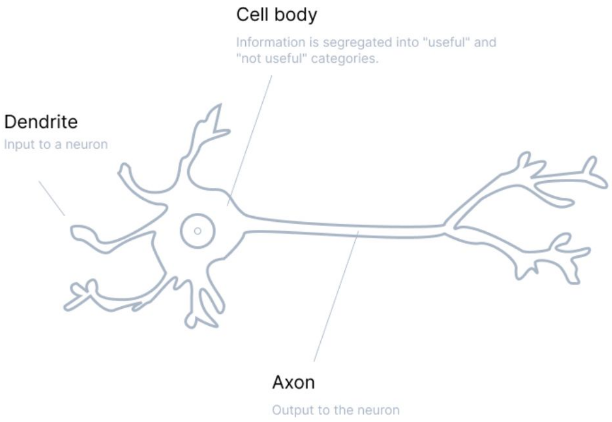
Topic-1

The perceptron and ANN architecture

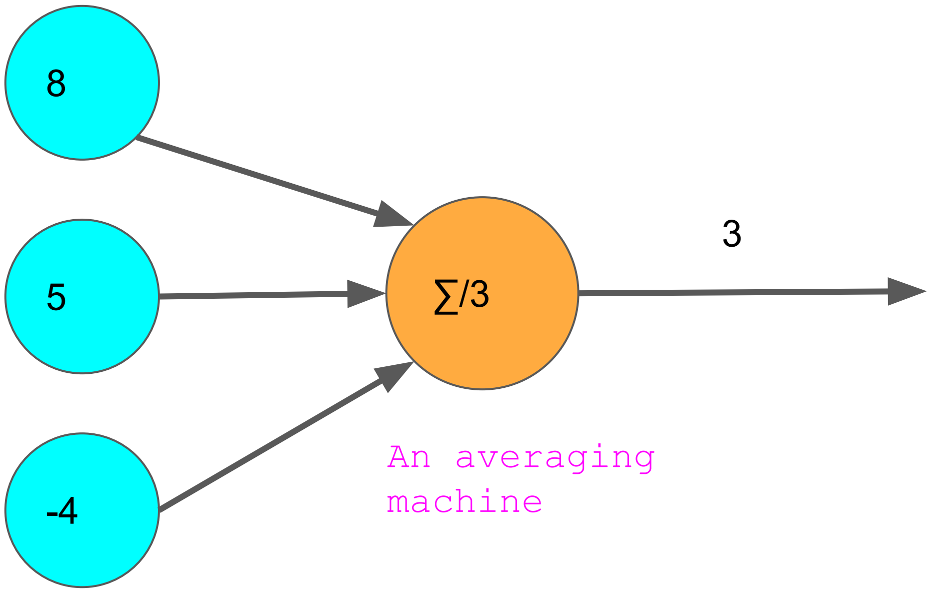
The perceptron



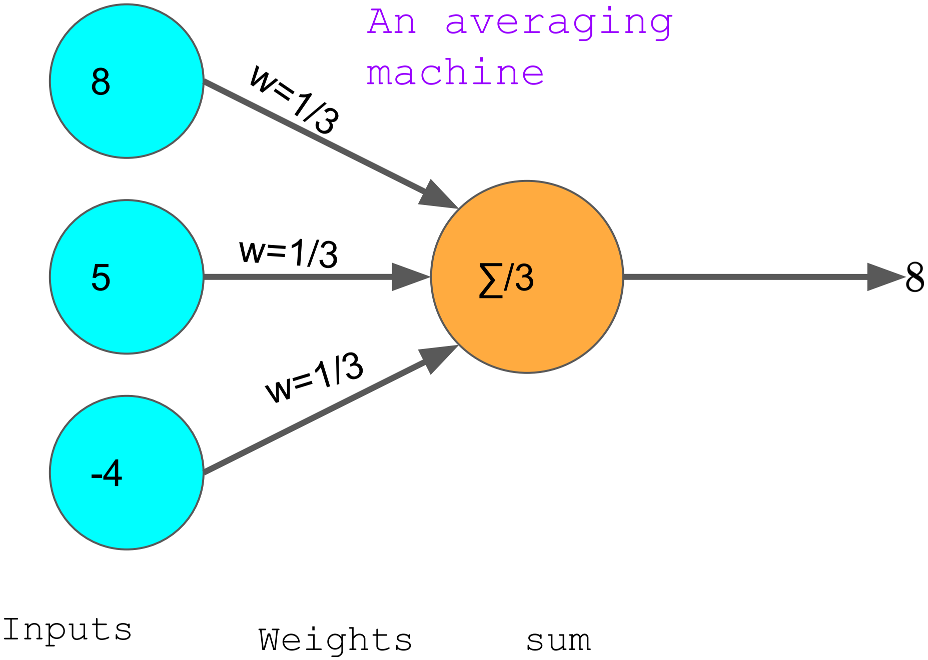
Inputs



The perceptron

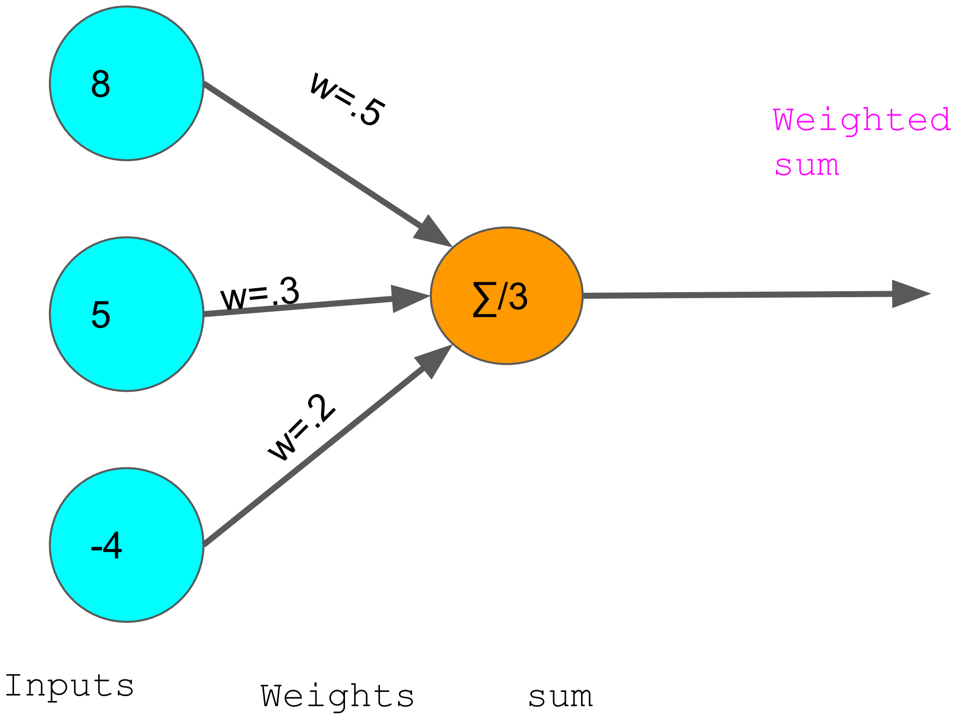


The perceptron



$$8 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} \pm 4 \cdot \frac{1}{3} = 3$$

The perceptron

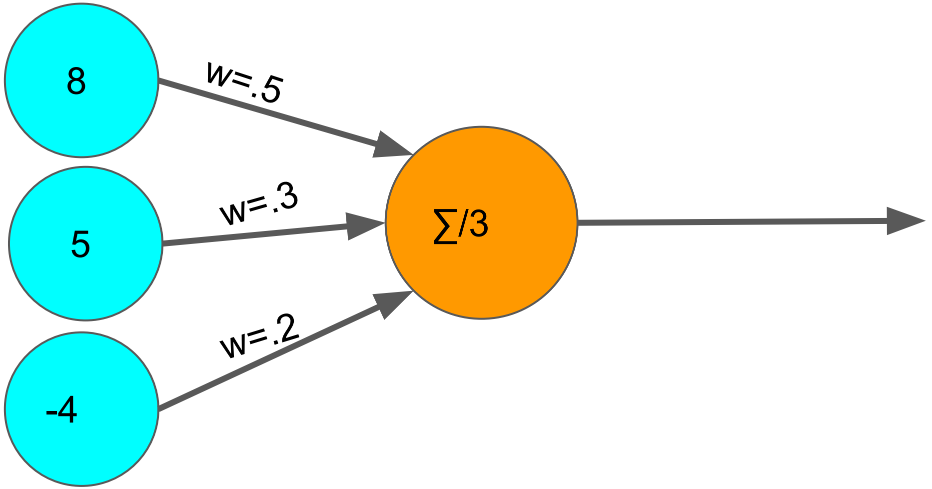


Weighted average=weighted sum

$$8 \cdot .5 + 5 \cdot .3 \pm 4 \cdot .2 = 4.7$$

The perceptron

A weighted averaging machine



$$x = \begin{bmatrix} 8 \\ 5 \\ -4 \end{bmatrix}, w = \begin{bmatrix} .5 \\ .3 \\ .2 \end{bmatrix}$$

$$y = x^T w$$

Inputs Weights sum

Linear Vs. NonLinear Operations

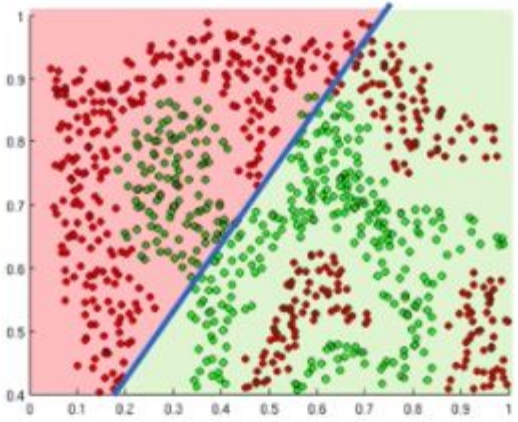
- Linear:Addition and Multiplication
- NonLinear:Anything else

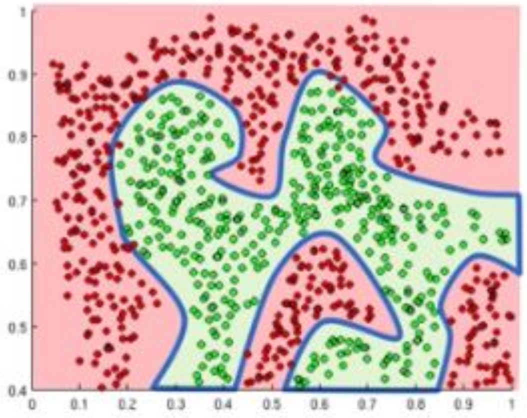
  - Linear models only solve linearly separable problems.
  - Nonlinear models can solve more complex problems.

Note:Never use a linear model for a nonlinear problem, and never use a nonlinear model for a linear problem!

Importance of Activation Functions

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

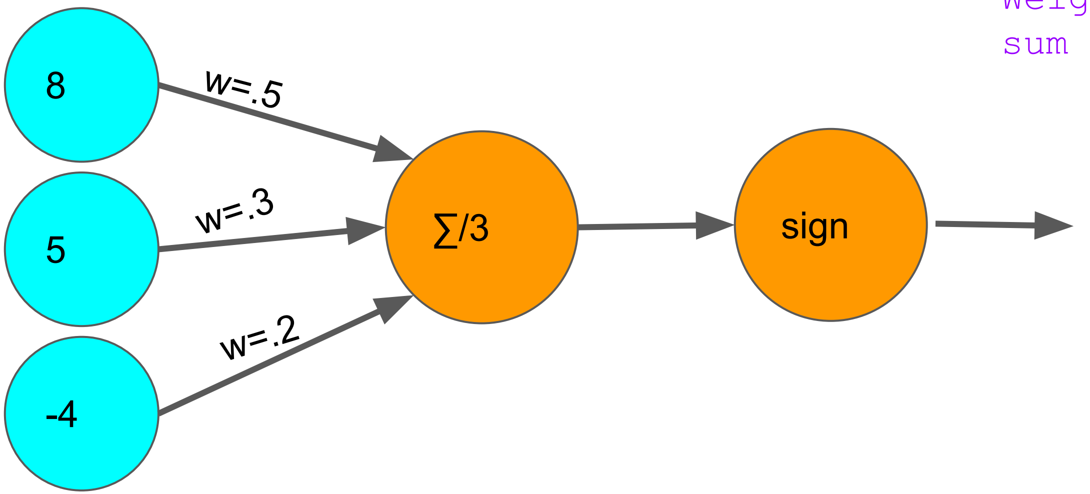




Linear activation function produce **linear decision** no matter the network size

Non Linearities allow us to approximate arbitrary **complex function**

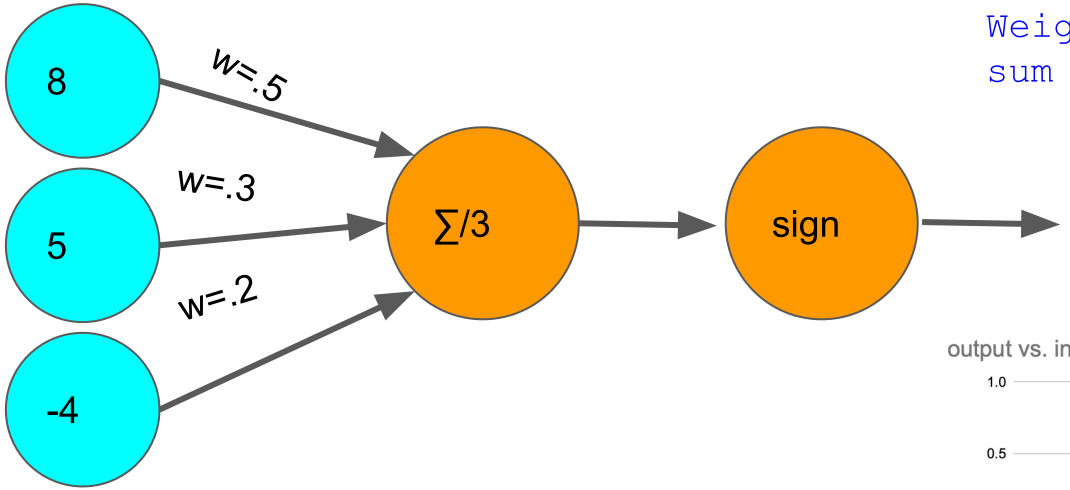
The perceptron



Weighted average=weighted sum

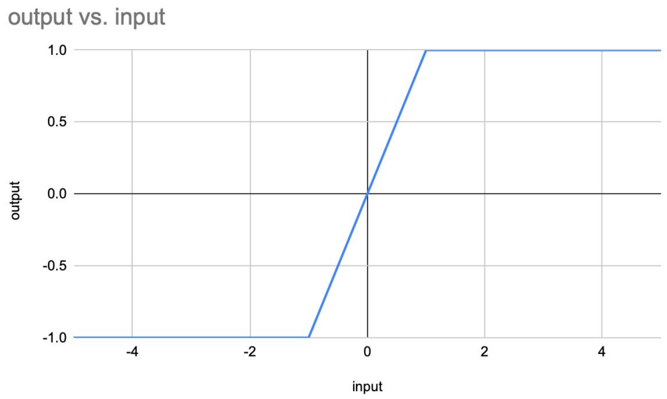
sgn$(8 \cdot .5 + 5 \cdot .3 \pm 4 \cdot .2 = 4.7)$=1

The perceptron

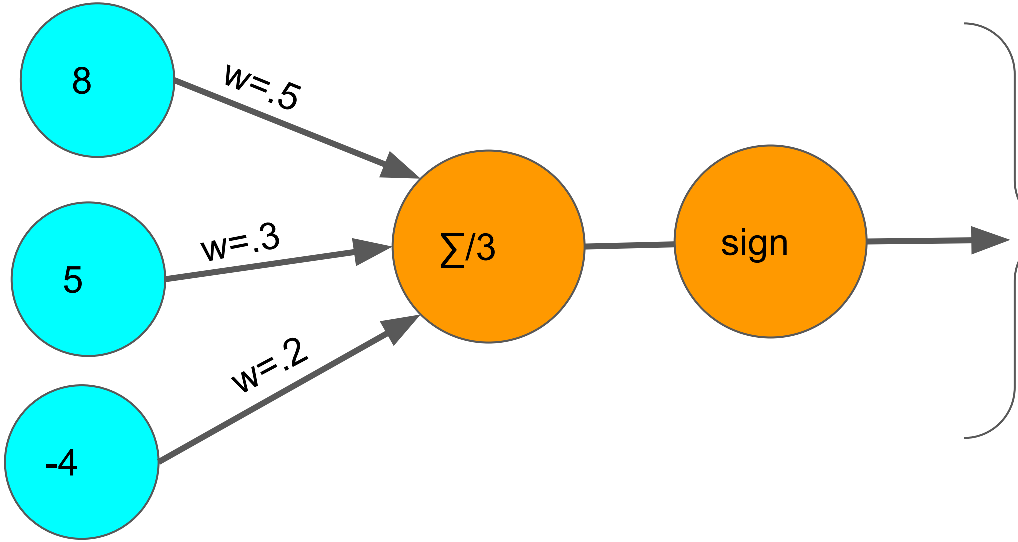


Weighted average=weighted sum

sgn($8 \cdot .5 + 5 \cdot .3 \pm 4 \cdot .2 = 4.7$)=1



The perceptron



$$x = \begin{bmatrix} 8 \\ 5 \\ -4 \end{bmatrix}, \; w = \begin{bmatrix} .5 \\ .3 \\ .2 \end{bmatrix}$$

$$\hat{y} = \sigma\left(x^T w\right)$$

This is a nonlinear function

The math of deep learning

$$\hat{y} = \sigma\left(x^T w\right)$$

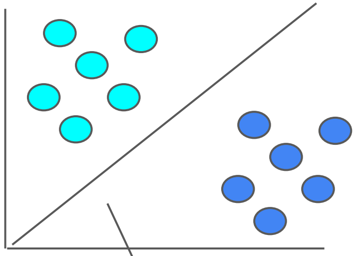
Output

Dot product
(linear weighted
sum)

The bias term

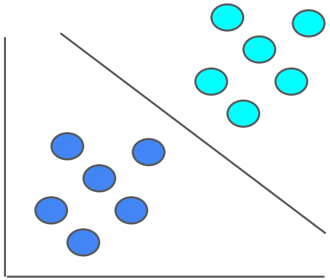
Goal:separate the two colors

y=mx



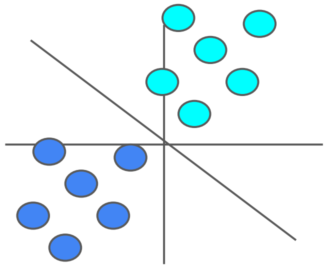
No bias term!

y=mx+b



With bias term!

y=mx

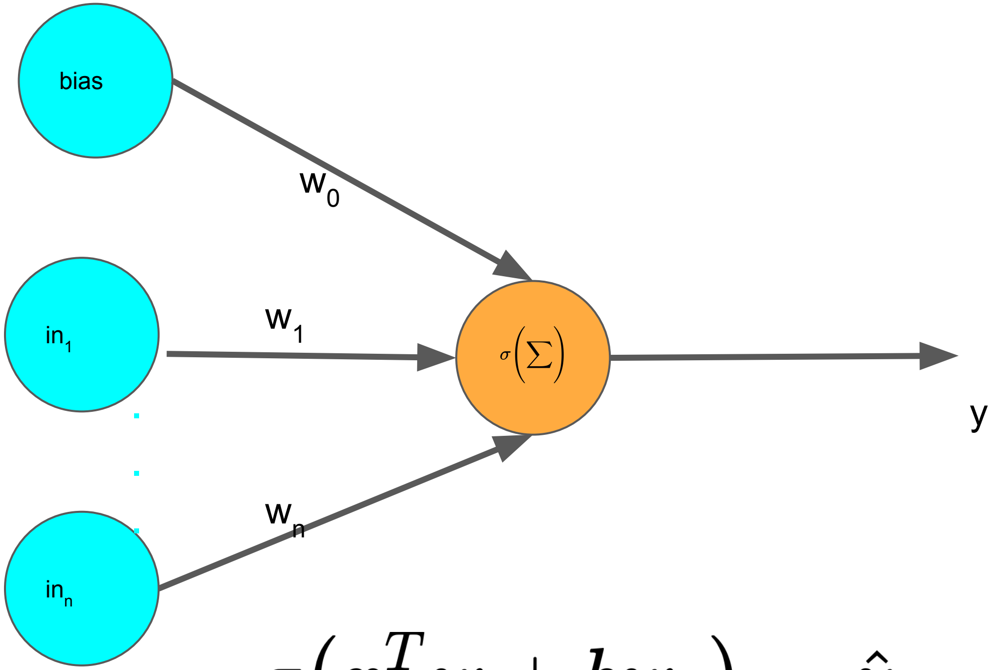


No bias term
Mean-centered
data

This is a linearly separable problem

Note: in general, always include a bias term.

The full perceptron model

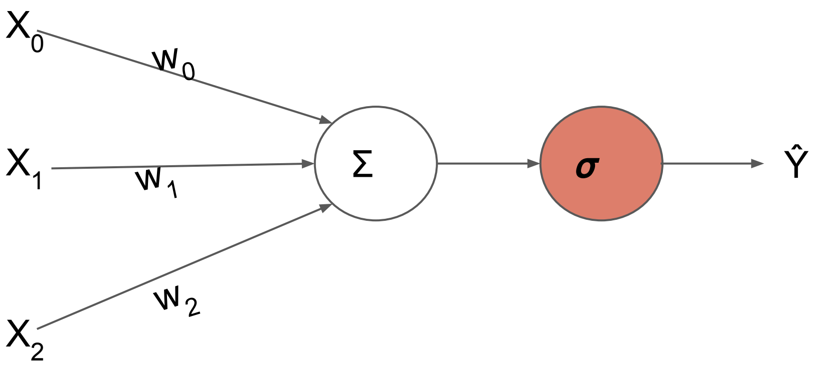


$$\sigma\left(x^T w + b w_0\right) = \hat{y}$$

Topic-2

A geometric view of ANNs

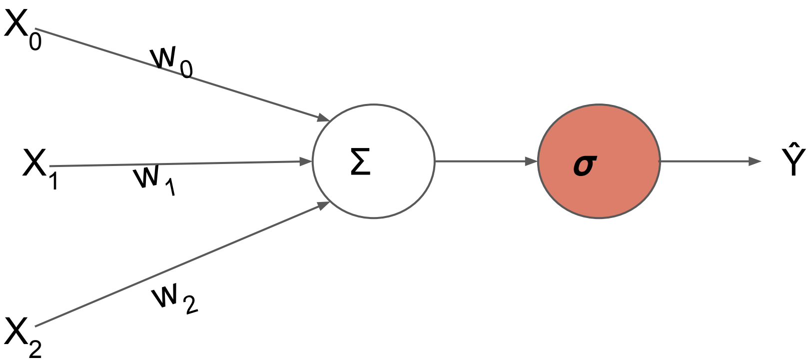
Feature space

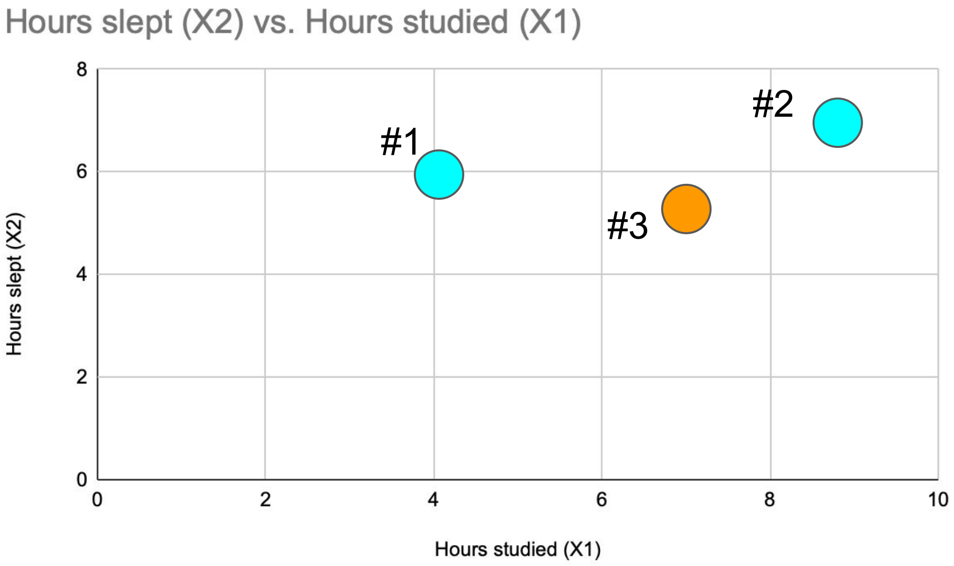


	X_1	X_2	y
ID#	Studie d	Slept	Result s
1	5	6	Pass
2	10	7	Pass
N	7	5	Fail

Question:Can we predict whether students pass or fail based on how much they slept and how much they studied?

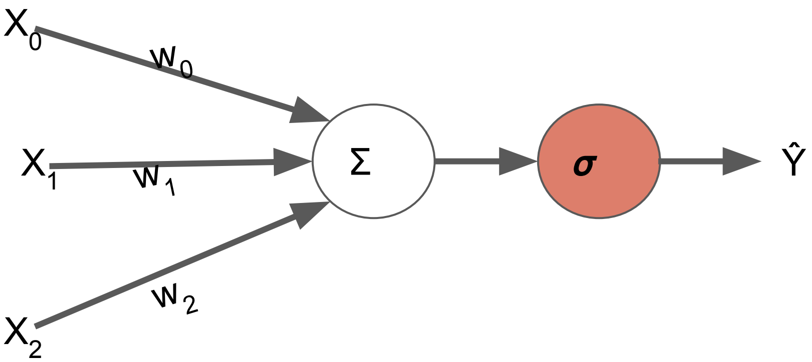
Feature space

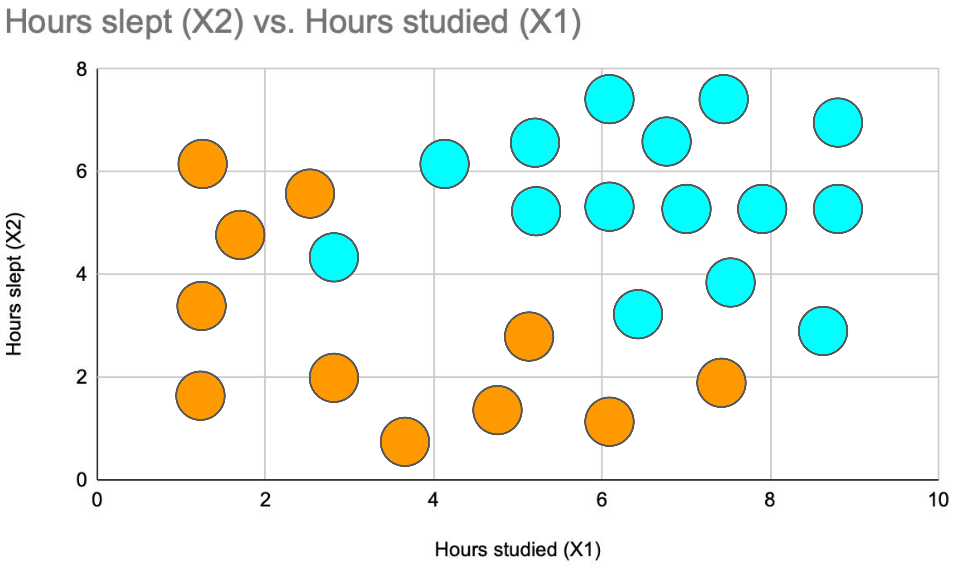




	X_1	X_2	y
ID#	Studied	Slept	Results
1	5	6	Pass
2	10	7	Pass
N	7	5	Fail

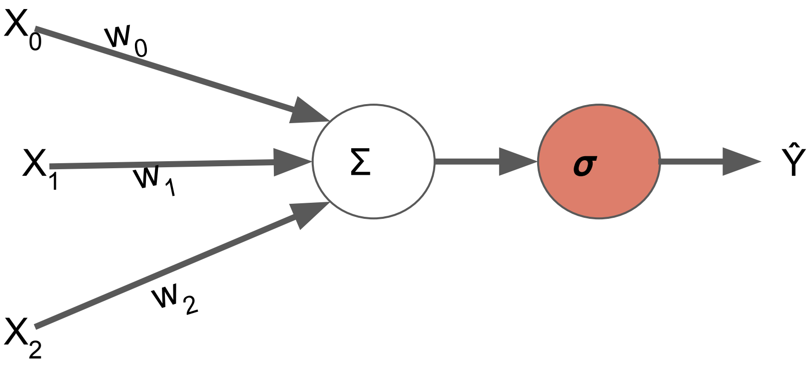
Feature space

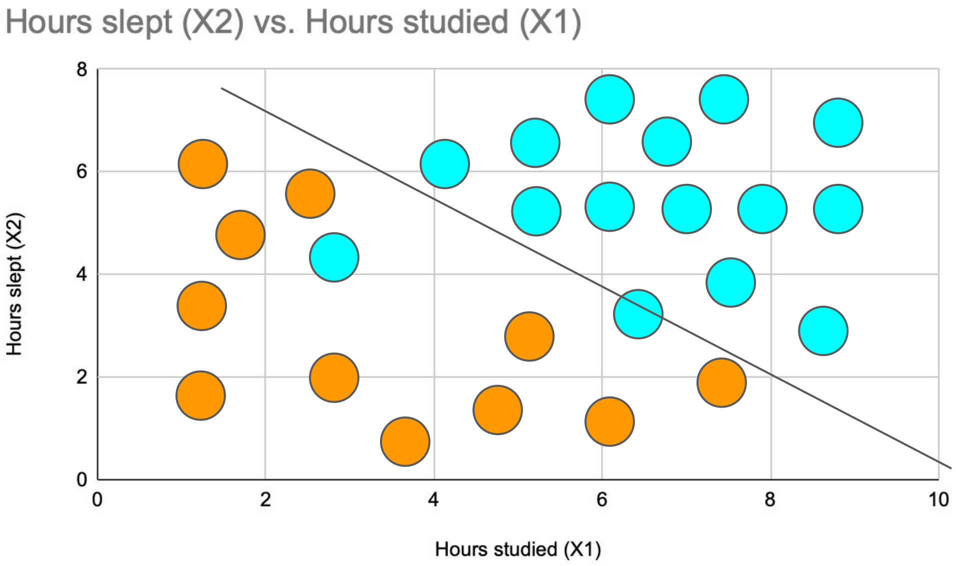




Feature Space: A geometric representation of the data, where each feature is an axis, and each observations a coordinate.

Feature space



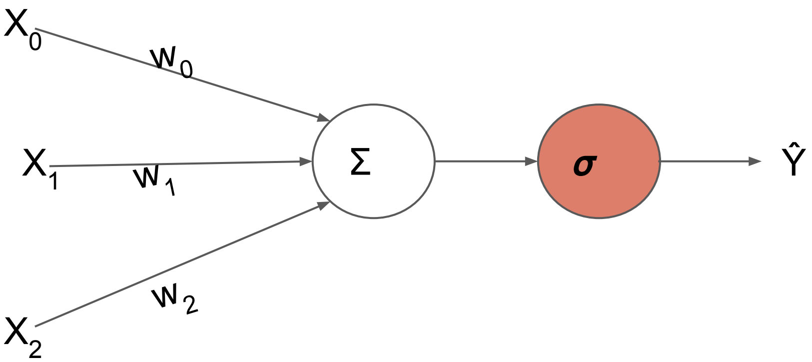


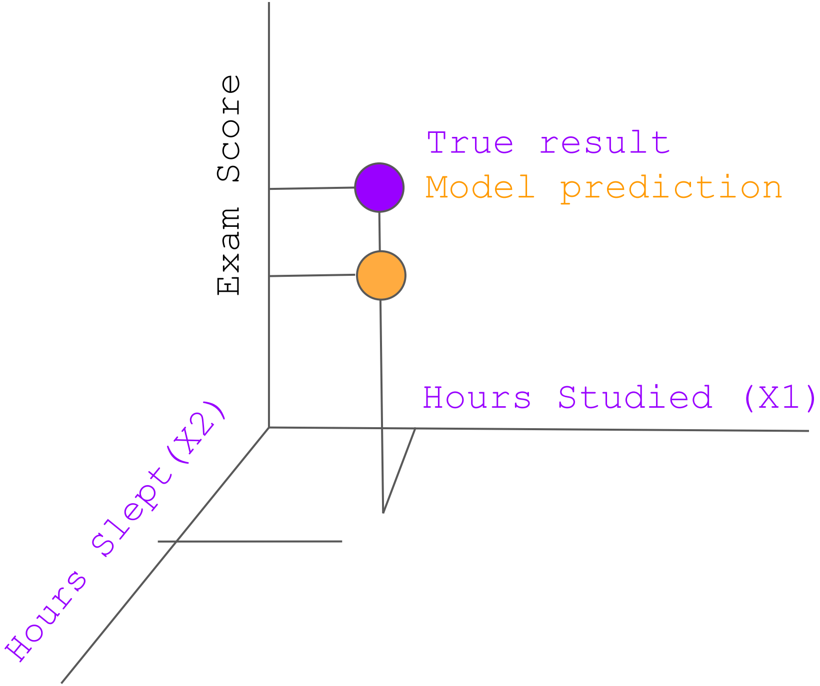
Separating hyperplane: A boundary that binarizes and categorizes data. It is used as a "decision boundary."

Categories of model output

Discrete/Categorical/ binary/boolean	Numeric/continuous
Pass/fail	Grade(exam score)
Text sentiment (positive/negative)	Language translation
Race(white, Asian, Black)	Attractiveness

Feature space

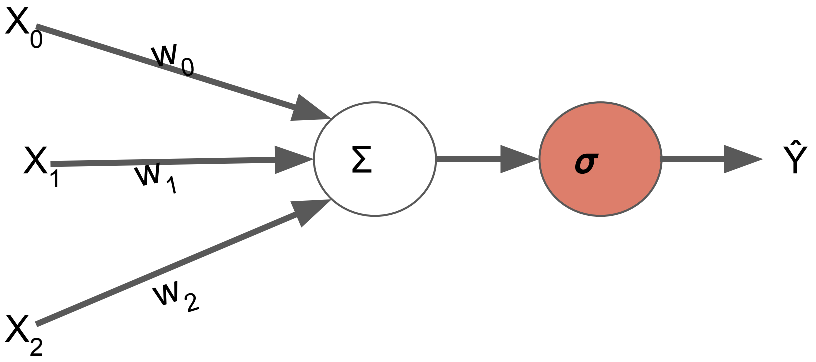




Topic-3

ANN math part 1(forward prop)

The model and the math



$$\hat{y} = \sigma\left(x_0 w_0 + \sum_{i=1}^{m} x_i w_i\right)$$

$$= \sigma\left(w_0 + x^T w\right)$$

$$= \sigma\left(x^T w\right)$$

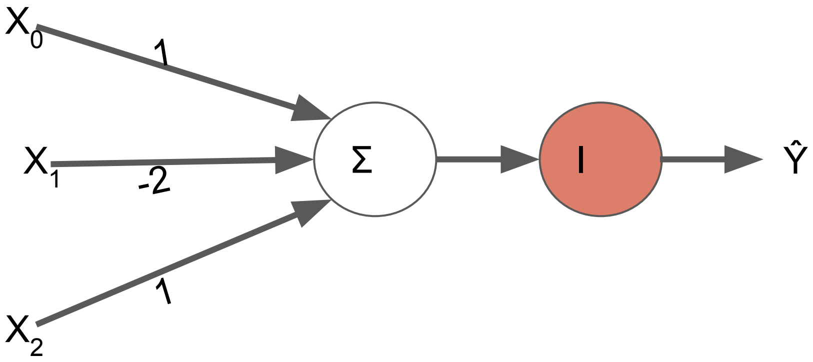
$$x_0 w_0 + \sum_{i=1}^{m} x_i w_i = x_0 w_0 + x_1 w_1 + x_2 w_2$$

$$x_0 w_0 + \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = x_0 w_0 + x_1 w_1 + x_2 w_2$$

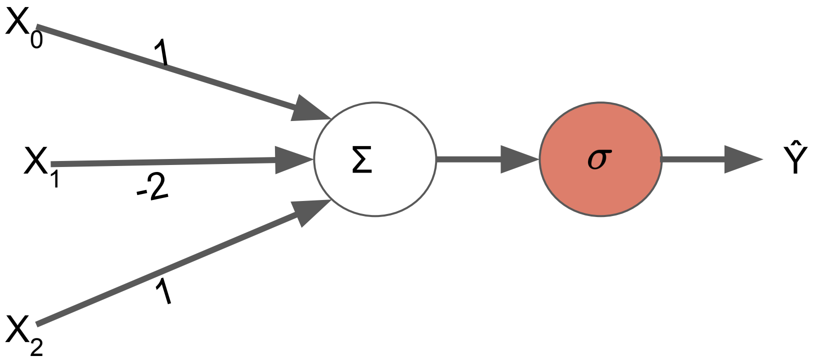
$$\begin{bmatrix} x_0 & x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = x_0 w_0 + x_1 w_1 + x_2 w_2$$

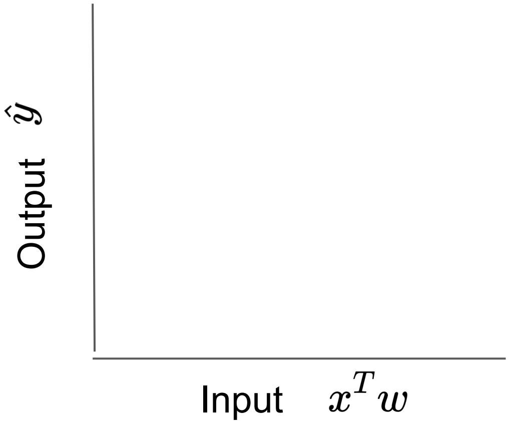
Numerical example

$$\hat{y} = 1 \pm 2x_1 + 1x_2$$

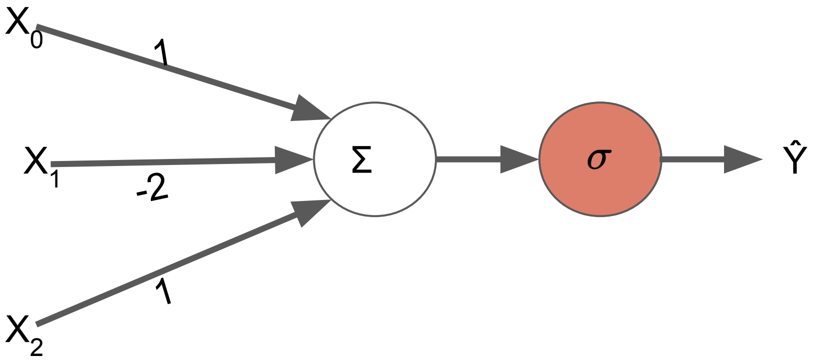


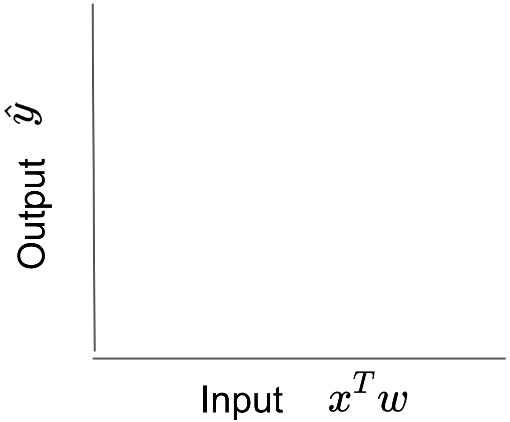
Activation functions

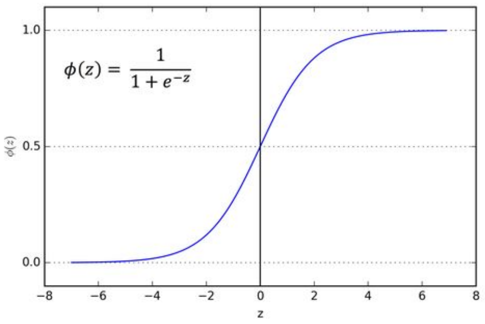




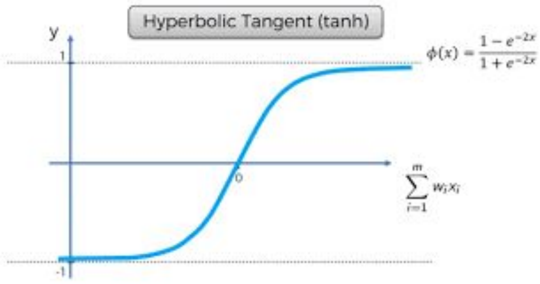
Activation functions



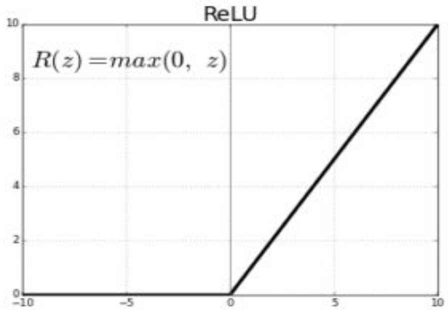




Sigmoi
d

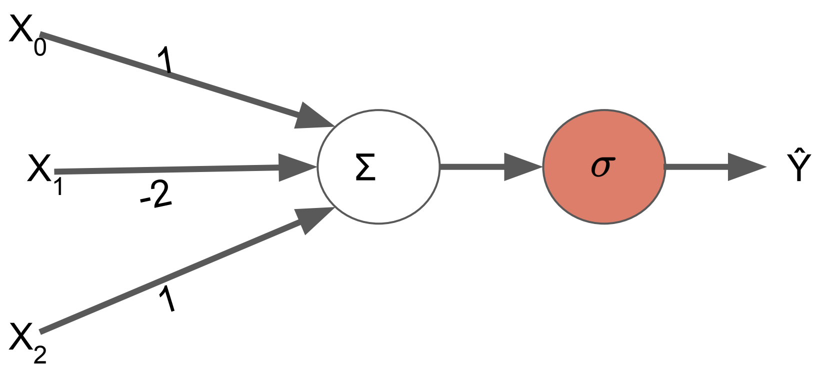


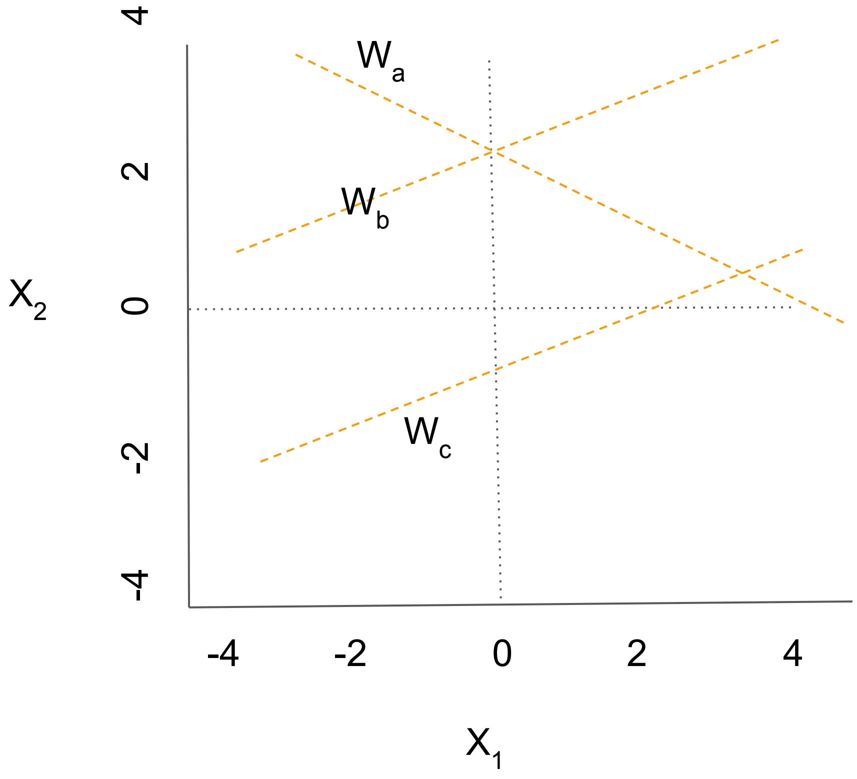
Hyperbolic Tangent



Relu

ANNs: All about the weights





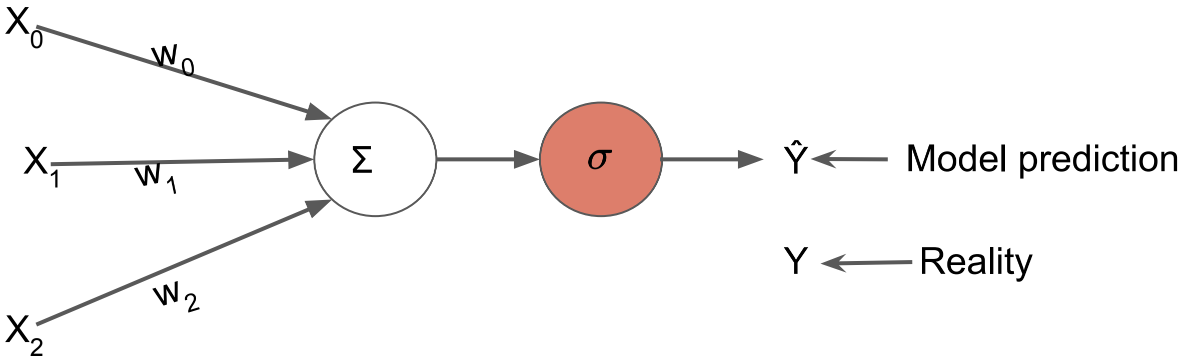
Problem: How to pick the right weights?

Solution: Learn from data! Via back-propagation

Topic-4

ANN math part 2(errors, loss, cost)

Expectation Vs. Reality



Sample	$\hat{Y}$	Y	error	bin.error
X_1	0.9	1	-.1	0
X_2	.2	0	+.2	0
X_3	.1	1	-.9	1
X4	.51	0	+.51	1

Binarized error is easier to interpret, but less sensitive.

Continuous error is more sensitive, but is signed

Loss Functions

Mean-squared error (MSE)

Use for continuous data when the output is a numerical prediction.

E.g., height, house price, temperature

$$L = \frac{1}{2}(\hat{y} - y)^2$$

Cross-entropy (logistic)

Use for categorical data when the output is a probability.

E.g., presence of disease, animal in picture, text sentiment

$$L = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

From loss to cost

$$J = \frac{1}{n} \sum_{i=1}^{n} L(\hat{y}_i, y_i)$$

Cost function

Lossfunction

The goal of DL optimization

Goal: find the set of weights that minimizes the losses.

$$W = \arg\ \min(w)\, J$$

$$J = \frac{1}{n} \sum_{i=1}^{n} L(\hat{y_i}, y_i)$$

$$= \frac{1}{n} \sum_{i=1}^{n} L(f(x, W)_i, y_i)$$

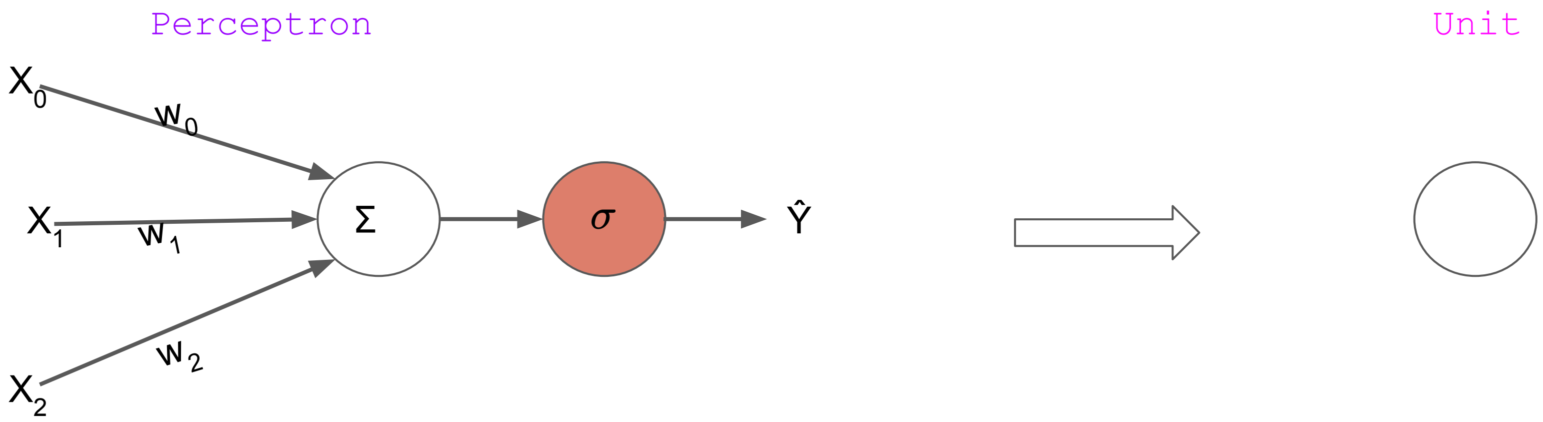
Is anything lost in the cost?

$$J = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y_i}, y_i)$$

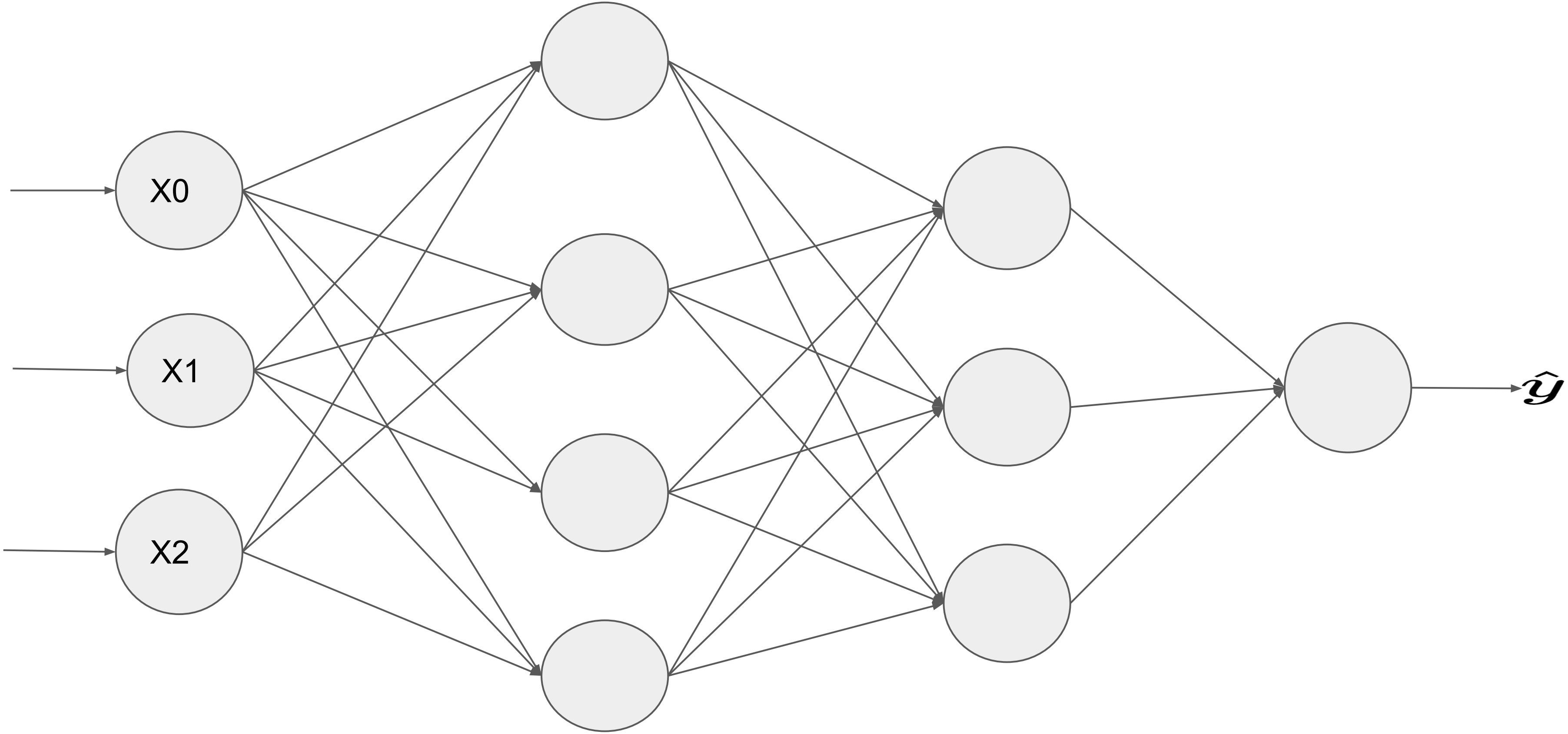
- Why train on cost and not loss?
- Training on each sample is time-consuming and may lead to overfitting.
- But averaging over too many samples may decrease sensitivity
- A good solution is to train the model in "batches" of samples

Topic-5
ANN math part 3(backprop)

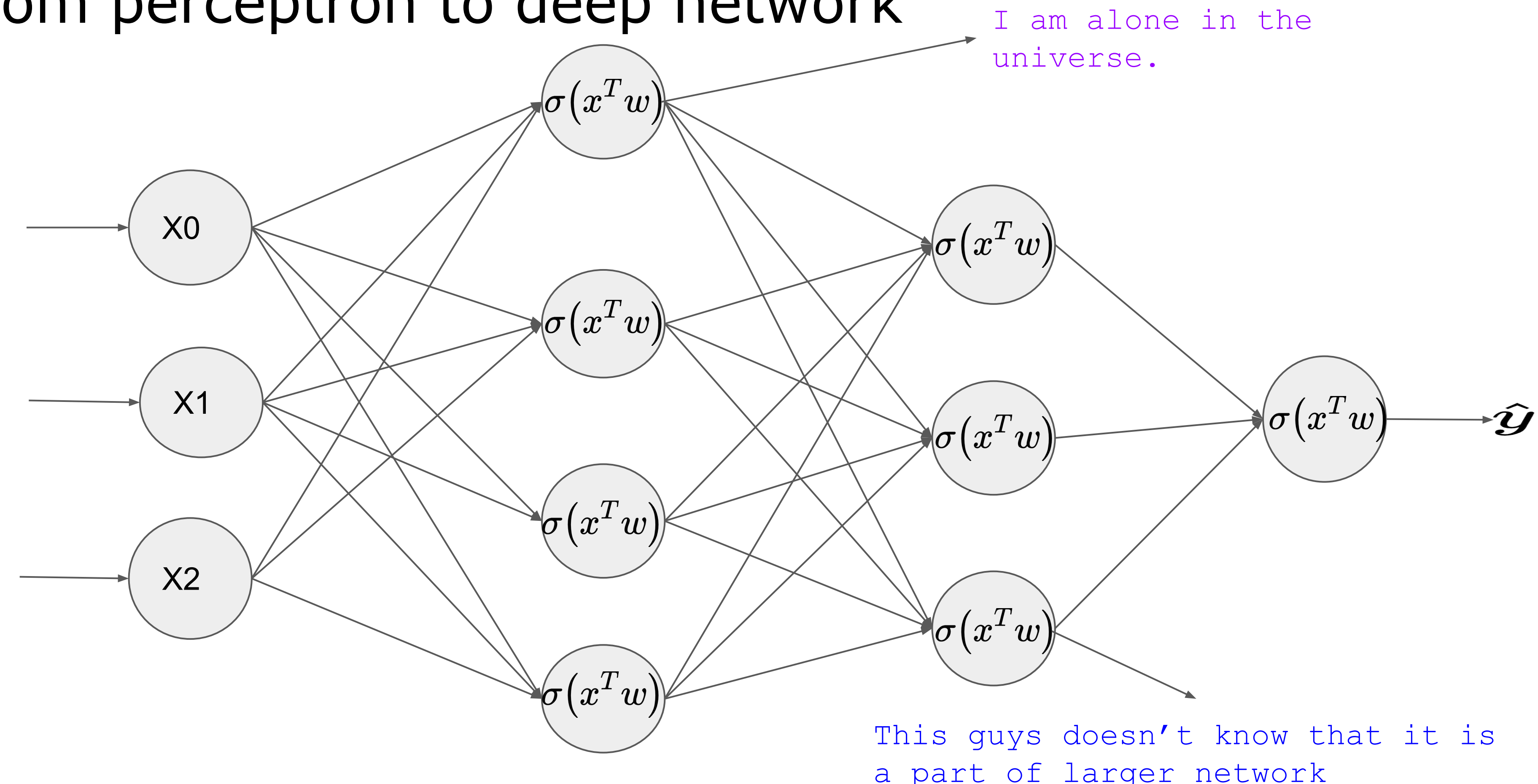
The shortening



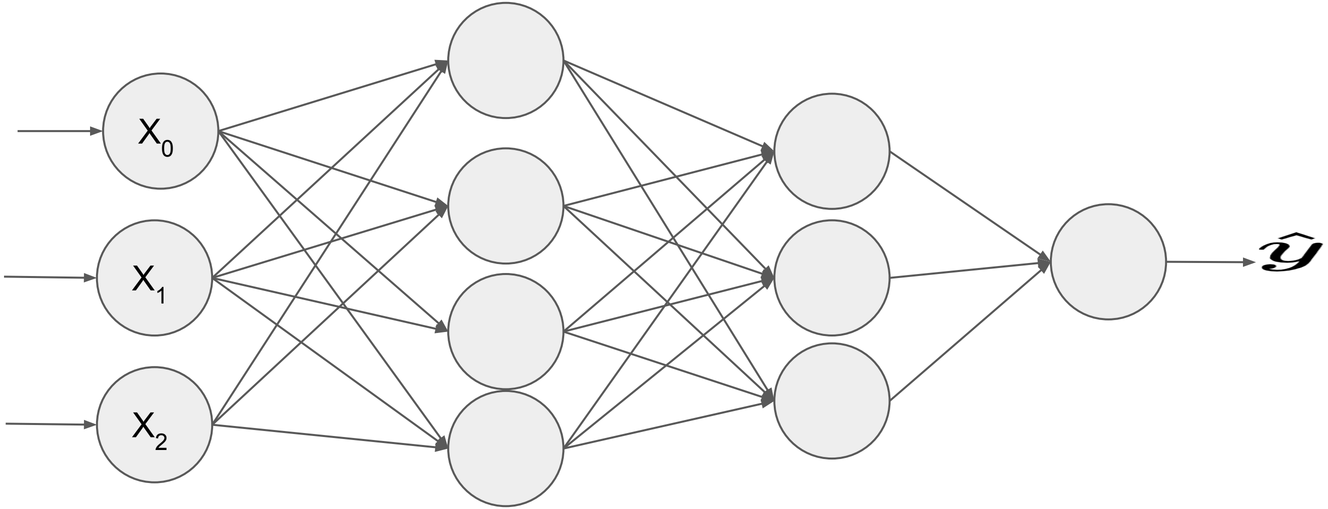
From perceptron to deep network



From perceptron to deep network



Forward prop and backprop



"Forward Propagation": Compute output based on input.

"Backwards propagation (backprop)":Adjust the weights based on loss/cost.

Backprop is g.d. super-charged

Gradient descent algorithm

Initialize random local min starting point

Loop over training iterations

- Compute derivative at local min
- Updated local min is itself minus derivative scaled by learning rate

Backprop and the chain rule

$$\sigma\left(x^T w\right) \longrightarrow \hat{y}$$

$$w \Leftarrow w - \eta \partial L$$

Learning rate(“eta”)

Derivative of loss

$$u = \sigma\left(x^T w\right) - y$$

$$\frac{\partial L(\hat{y}, y)}{\partial w} = \frac{\partial L\left(\sigma\left(x^T w\right), y\right)}{\partial w}$$

$$L(\hat{y}, y) = \frac{1}{2}\left(\sigma\left(x^T w\right) - y\right)^2$$

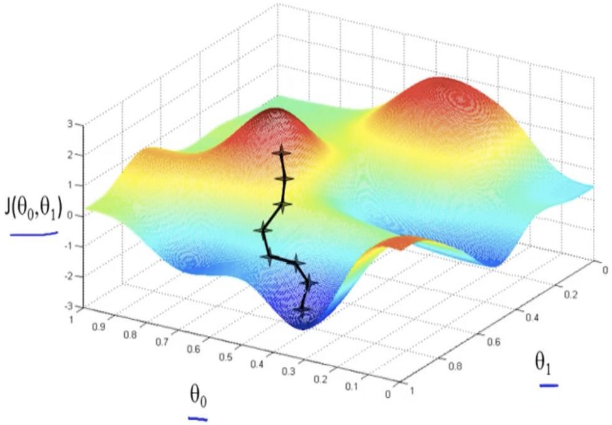
$$\frac{\partial L(u)}{\partial w} = \frac{\partial L(u)}{\partial u} \frac{\partial u}{\partial w}$$

$$\frac{\partial L(u)}{\partial w} = \frac{\partial L(u)}{\partial u} \frac{\partial\left(\sigma\left(x^T w\right) - y\right)}{\partial w}$$

Gradient Descent

Algorithm

1. Initialize weights randomly
2. Loop unit convergence:
3. Compute gradient, $\frac{\partial L(w)}{\partial w}$
4. Update weights, $w \Leftarrow w - \eta \partial L$
5. Return weights



ANNs(Artificial Neural network)

In this Lecture, you have learned about

- The basic architecture of an ANN (artificial neural network).
- The linear and nonlinear components of an artificial neural network.
- What the terms “feature space” and “separating hyperplane” mean.
- More about biases, weights, and activation functions.
- Different categories of errors, and their corresponding loss functions.
- The difference between loss and cost.
- How the gradient descent algorithm is extended to DL