The Joy of *Finally* Learning

www.mldawn.com

OUTLINE OF THE TALK

- What is an Artificial Neural Network (ANN)?
- What is a Perceptron?
- Bayes' Rule and Birth of Error Functions (with Derivations)
- The Birth of Binary-Cross Entropy Function (with Derivations)
- Forward Pass and Backward Pass in a Binary Classifier
- A Demo

WHAT IS A NEURAL NETWORK?

- An ANN is a function approximator!
  - What is a function?
  - Examples of functions: *real-valued, discrete-valued, and vector-valued*

- Applications of ANNs:
  - Recognizing hand-written characters
  - Recognizing spoken words
  - Recognizing faces

WHAT IS A NEURAL NETWORK?

Unknown Function !!!

x → $\bar{f}(\,) \equiv$ **ANN** → $f(x)$

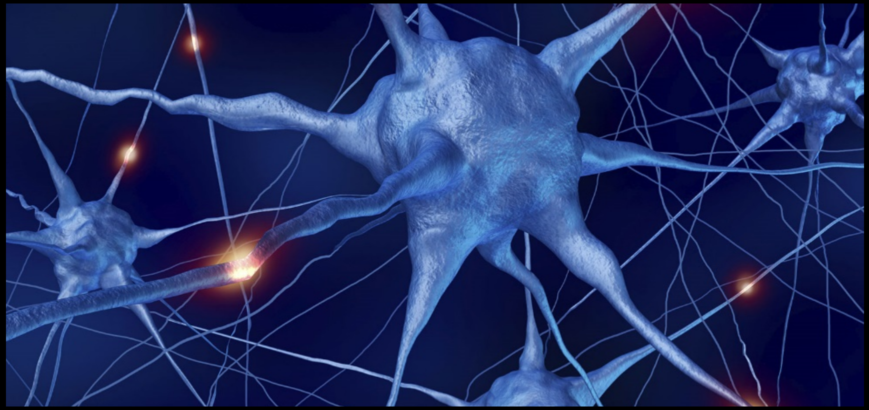
x	$f(x)$
Image	Cat / Dog (Prob. Vector)
Stock Information	Price ($) (real-valued)
Car Information	Number (discrete-valued)

$$\bar{f}(\,) \approx f(\,)$$

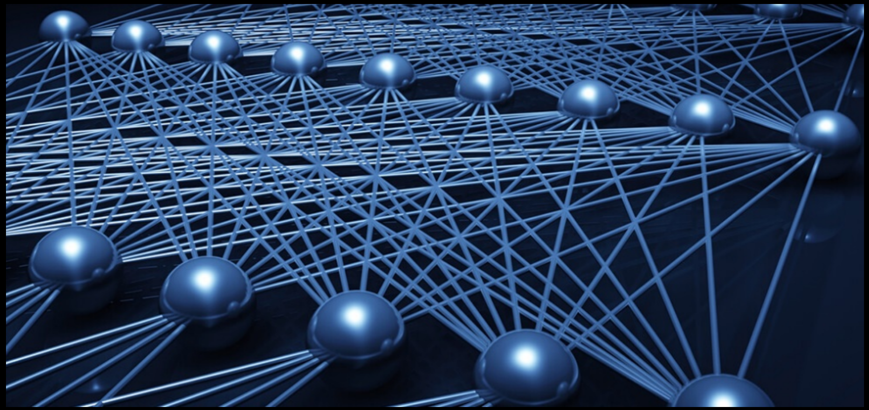
ANN approximates $f(\,)$ *for us*

ANN BIOLOGICAL MOTIVATION

- ANNs are inspired by Biological Learning Systems (BLS)
  - BLS: A massive interconnection of neurons



- ANNs:
  - Densely interconnected set of simple unites



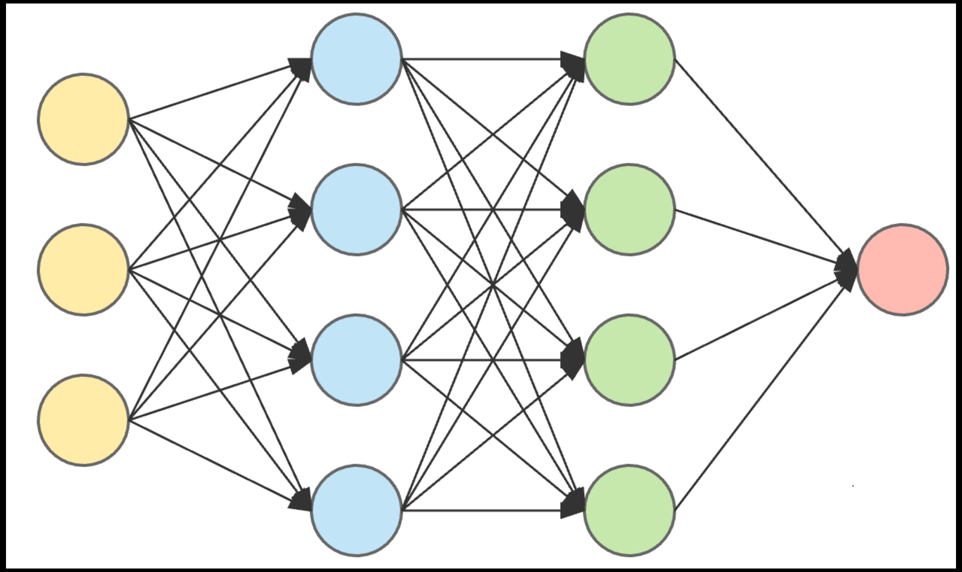
ANN BIOLOGICAL MOTIVATION

- For each unit in ANN:
  - **<u>Possibly</u>** several inputs
  - A Single real-valued output

- BLS processes are highly parallelized
  - Over nearly 10^{11} neurons

- ANNs do NOT fully reflect a BLS
  - Output of neurons are different

- Researchers try to:
  1. Model BLS processes
  2. Develop effective ANNs (no regards to BLS compatibility)



SIMPLEST ANN

A PERCEPTRON

A PERCEPTRON

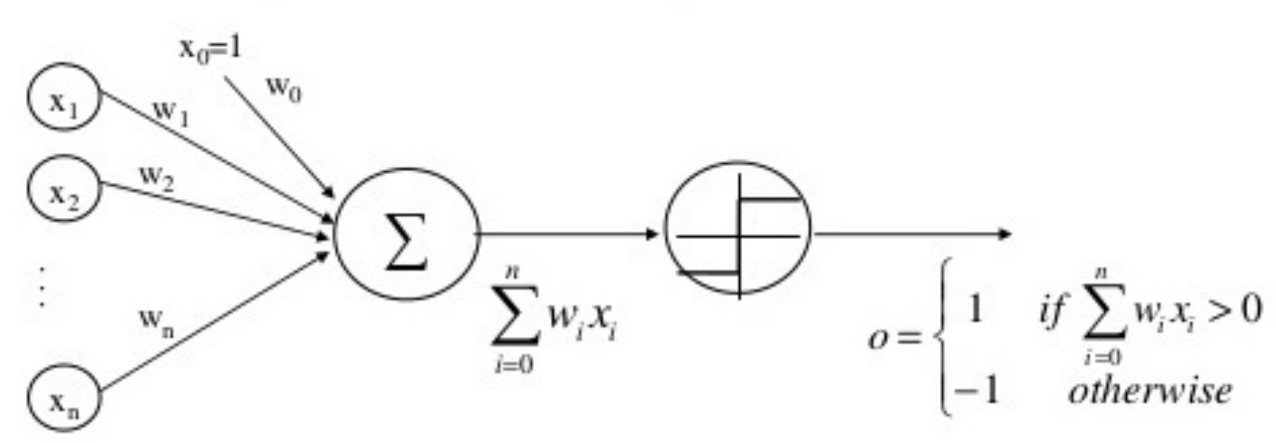
- A type of ANN that is based on a unit called ***perceptron***

- There is a rule about input/output of a perceptron:

  - Input: A vector of real-valued data

  - Linearly combine the components of the input

  - $$o(x_1, \ldots, x_n) = \begin{cases} +1 & if\ w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n > 0 \\ -1 & Otherwise \end{cases}$$



A PERCEPTRON

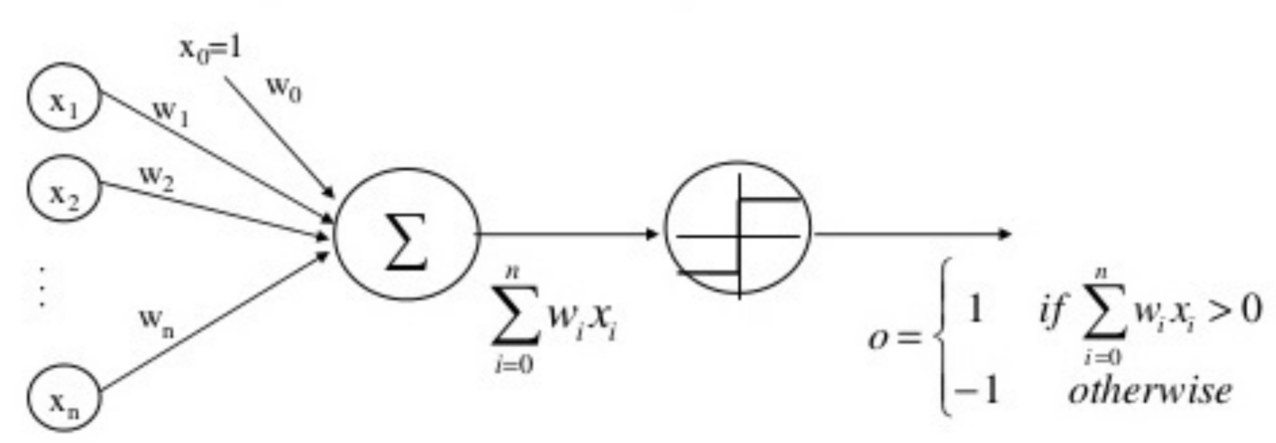
- What are the weights? What is that w_0?

- Let's imagine an additional input
  - $x_0 = 1$

- This simplifies the inequality:
  - $\sum_{i=0}^{n} w_i x_i > 0$
  - Or in vector form: $\vec{w} \,.\, \vec{x} > 0$

- Can we show the perceptron smarter?

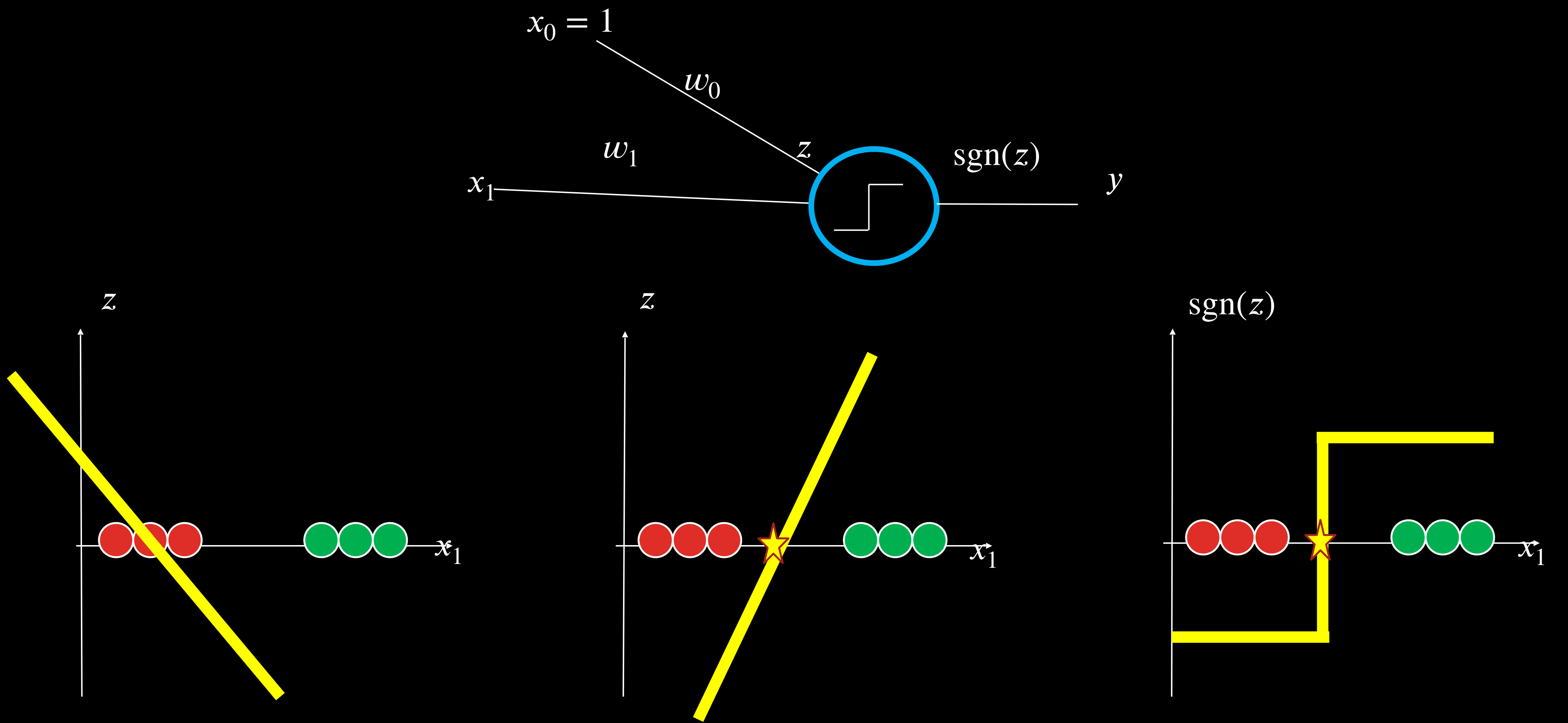
  - $o(\vec{x}) = \mathrm{sgn}(\vec{w} \,.\, \vec{x})$
  - Where: $sgn(z) = \begin{cases} +1 & \text{If } z > 0 \\ -1 & \text{otherwise} \end{cases}$



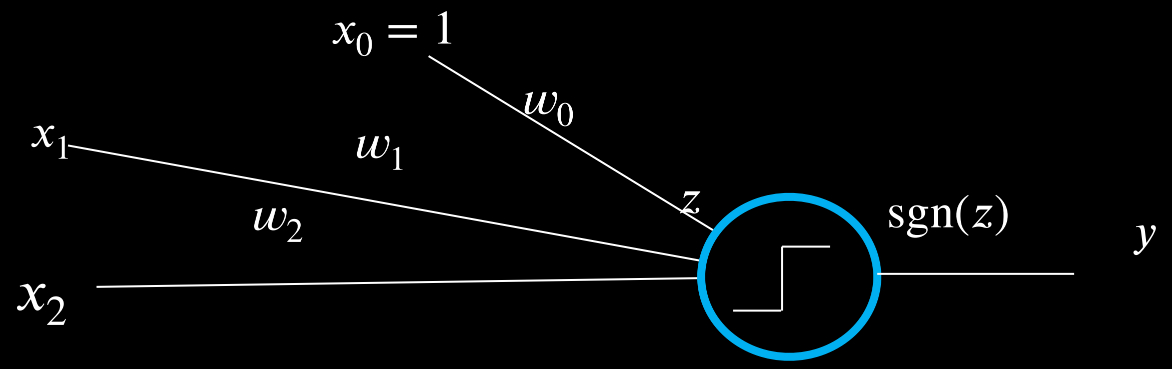
- Learning a perceptron = Choosing the weights

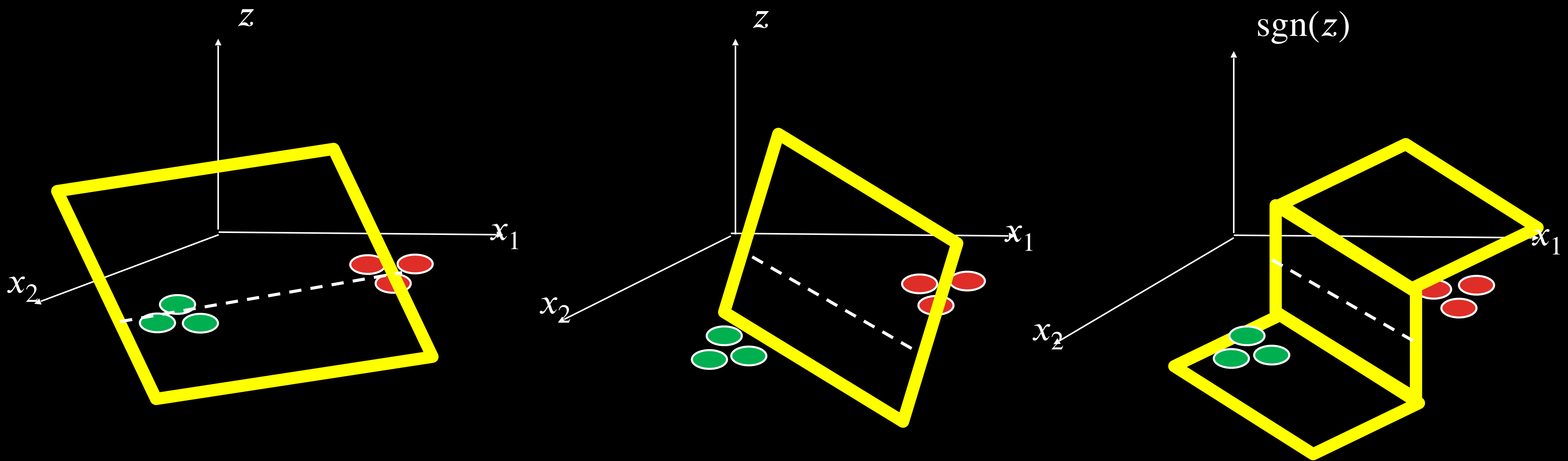
- The hypothesis space: $H = \{\vec{w} \mid \vec{w} \in R^{(n+1)}\}$

A PERCEPTRON: A DEEPER LOOK



A PERCEPTRON: A DEEPER LOOK





THE BIRTH OF ERROR FUNCTIONS

DO I *TRUST* MY ANN?

BAYES' RULE

- Let's assume the following notation:
  - M : Our Model
  - D: Our Evidence

- Then the Bayes' Rule:

$$P\left(M \mid D\right) = \frac{P\left(D \mid M\right) P(M)}{P(D)}$$

TAKING THE LOGS() : A SIMPLIFICATION

$$\log\left(\frac{a}{b}\right) = \log(a) - \log(b)$$

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)} \longrightarrow \ln P(M|D) = \ln\left[P(D|M)P(M)\right] - \ln P(D)$$

$$\log(a \times b) = \log(a) + \log(b)$$

$$\ln P(M|D) = \ln P(D|M) + \ln P(M) - \ln P(D)$$

THE BAYESIAN MACHINERY (MAPE)

$$lnP(M\,|\,D) = ln\;P\big(D\,|\,M\big) + \;ln\;P(M) - lnP(D)$$

$$\boldsymbol{Maximize\;P(M\,|\,D) \equiv Minimize\;\; - P(M\,|\,D)}$$

$$-lnP(M\,|\,D) = -ln\;P\big(D\,|\,M\big) - \;ln\;P(M) + lnP(D)$$

$$E = \;-lnP(M\,|\,D)$$

This is how Bayes' Rule connects to Error Functions

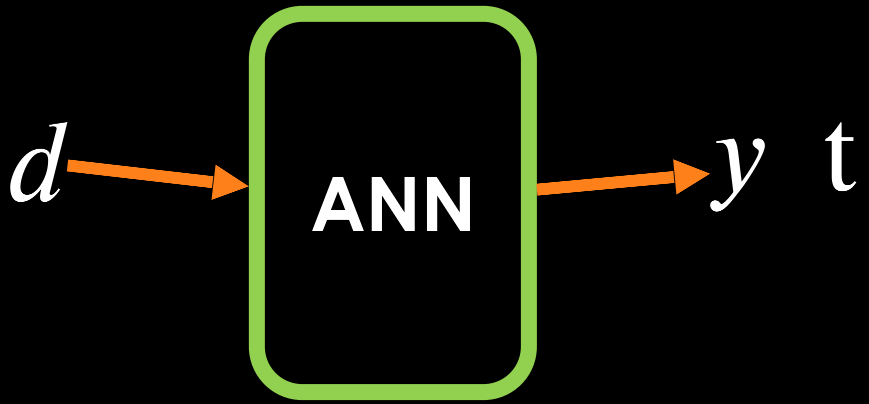
MAPE AND ANNS : SUPERVISED LEARNING

- How can we apply the MAPE to an ANN with weights w?
- Let's assume that we have a set of examples (d_i,t_i) where d_i is the i-th input and t_i the corresponding target.
- Then we can rewrite the likelihood as:

$$P(D|M) \ and \ D \approx (d_i \ , \ t_i)$$

$$P(D|M) = P\left((d_i, t_i)\middle|M\right)$$

$$P\left((d_i, t_i)\middle|M\right) = \mathrm{P}(\mathrm{t_i}|\mathrm{d_i}, M) \times \underbrace{P(d_i|M)}_{P(d_i)}$$



MAPE AND ANNS : SUPERVISED LEARNING

$$-lnP(M|D) = -ln\ P(D|M) - \ ln\ P(M) + lnP(D)$$

Model indeed means our weights W

$$-lnP(W|D) = -ln\ \underbrace{P(D|W)}_{P(\mathrm{t_i}|\mathrm{d_i},\ \mathrm{W}) \times P(d_i)} - \ ln\ P(W) + lnP(D)$$

$P(D)$ and $P(d_i)$ are both independent of our weights W. Ignore them during Maximization

$$-lnP(W|D) = -\ln\prod_{i=1}^{N}\left[\mathrm{P}(\mathrm{t_i}|\mathrm{d_i},\ \mathrm{W}) \times P(d_i)\right] - \ln P(W) + \ln P(D)$$

$$-lnP(W|D) = -\sum_{i=1}^{N}\ln\left[\mathrm{P}(\mathrm{t_i}|\mathrm{d_i},\ \mathrm{W}) \times P(d_i)\right] - \ln P(W) + \ln P(D)$$

$$-lnP(W|D) = -\sum_{i=1}^{N}\ln\left[\mathrm{P}(\mathrm{t_i}|\mathrm{d_i},\ \mathrm{W})\right] - \sum_{i=1}^{N}\ln P(d_i) - \ln P(W) + \ln P(D)$$

MAPE AND ANNS : SUPERVISED LEARNING

$$-lnP(W|D) = -\sum_{i=1}^{N} \ln\left[P(t_i|d_i,\ W)\right] - \underbrace{\cancel{\ln P(W)}}$$

Can be used for regularizing W
to avoid overfitting

$$P(W) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(w-\mu)^2}{2\sigma^2}} \quad \xrightarrow{\ln(\ )} \approx -(w_1^2 + w_1^2 + \ldots + w_N^2)$$

$$E = -lnP(W|D) = -\sum_{i=1}^{N} \ln\left[P(t_i|d_i,\ W)\right]$$

BINARY CROSS-ENTROPY ERROR FUNCTION

THE DERIVATION

BINARY CLASSIFICATION

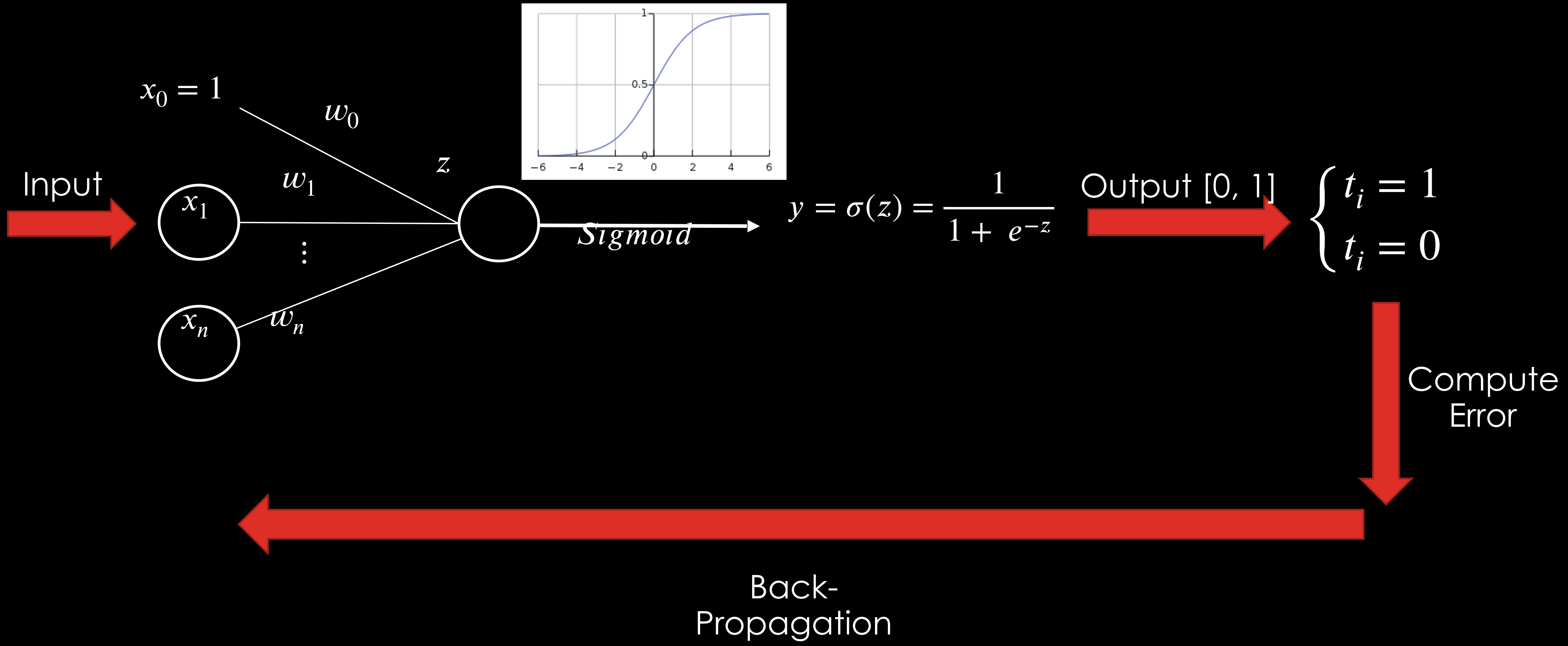
- We have 2 classes: Positive and Negative

- Binary Classification:
  - $t_i = 1$ means that x_i belongs to the **positive** class
  - $t_i = 0$ means that x_i belongs to the **negative** class

- Artificial Neural Networks (ANNs): Probabilistic approach
  - ANN computes the probability of the **Positive Class**: $P(y_i = 1 \mid x_i)$
  - Could a linear model work? Say, $y = w^T x + b$ → **Not Bounded!**

We need a Squasher!

BINARY CLASSIFICATION



BINARY CLASSIFICATION

- The output of our ANN is probability. Let's denote:
  - y_i: Probability x_i belongs to the Positive class (i.e., $t_i = 1$)
  - $(1 - y_i)$: Probability x_i belongs to the Negative class (i.e., $t_i = 0$)

$$P(t_i \mid d_i,\ w) = \begin{cases} y_i & if\ t_i = 1 \\ 1 - y_i & if\ t_i = 0 \end{cases}$$

$$P(t_i \mid d_i,\ w) = y_i^{t_i}(1\ - y_i)^{1-t_i}$$

Bernoulli Distribution

BINARY CLASSIFICATION

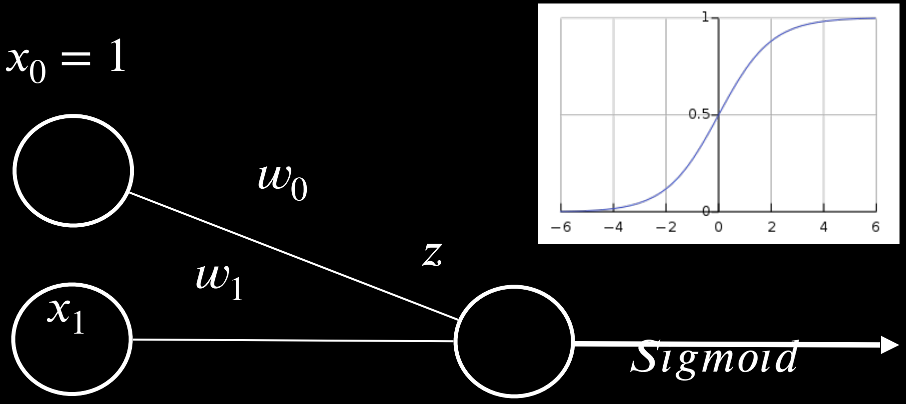
$$P(t_i \mid d_i, w) = y_i^{t_i}(1 - y_i)^{1-t_i}$$

$$E = -\sum_{i=1}^{N} \ln\left[\mathrm{P}\left(\mathrm{t_i} \mid \mathrm{d_i},\ \mathrm{W}\right)\right] = -\sum_{i=1}^{N} \ln[y_i^{t_i}(1 - y_i)^{1-t_i}]$$

$$= -\sum_{i=1}^{N} [\ln y_i^{t_i} + \ln\left(1 - y_i\right)]$$

$$\boldsymbol{E = -\sum_{i=1}^{N} [t_i \ln y_i + \left(1 - t_i\right)\ln(1 - y_i)]}$$

FORWARD PASS



$$y = \sigma(z) = \frac{1}{1 + e^{-z}} \longrightarrow E = -t \times \ln y - (1 - t) \times \ln(1 - y)$$

$$Error = E\big(\underbrace{\sigma(Z)}_{y}\big) = E(\sigma(\underbrace{w_1 x_1 + w_0}_{Z}))$$

Binary Cross-Entropy Error Function

BACKWARD PASS

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial Z} \times \frac{\partial Z}{\partial w_1} = \left[\frac{-t}{y} + \frac{(1-t)}{(1-y)}\right] \times y\left(1-y\right) \times x_1 = (y - t) \times x_1$$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial Z} \times \frac{\partial Z}{\partial w_0} = \left[\frac{-t}{y} + \frac{(1-t)}{(1-y)}\right] \times y\left(1-y\right) \times x_0 = (y - t)$$

- So the learning rule for both our learnable parameters (i.e., weights) is as follows:

$$w_1 = w_1 - \eta \times \frac{\partial E}{\partial w_1} = w_1 - \eta \times [(y-t) \times x_1]$$

$$w_0 = w_0 - \eta \times \frac{\partial E}{\partial w_0} = w_0 - \eta \times (y-t)$$

The Joy of *Finally* Learning

www.mldawn.com

@MLDawn2018

Mehran Bazargani

ML Dawn

Course Available at: https://www.mldawn.com/course/the-birth-of-error-functions/

Demo's Code: https://www.mldawn.com/binary-classification-from-scratch-using-numpy/