ECE 09495/09595

Advanced Emerging Topics in Computational Intelligence, Machine Learning and Data Mining
Emerging Topics in Computational Intelligence, Machine Learning and Data Mining

Ghulam Rasool, PhD

30 January 2019

Lecture 3 and 4

Housekeeping

- Google ML Crash Course
- Projects
- Google Colab

Data Preparation

May take up to 70-80% of the teams' effort initially

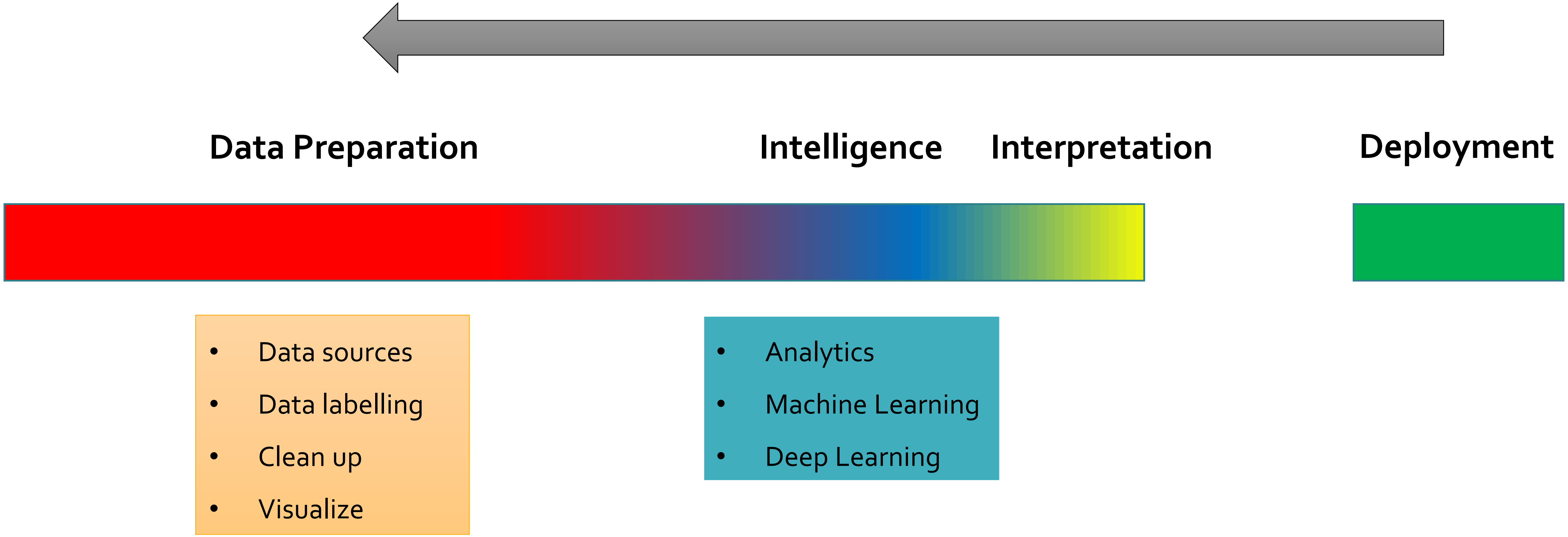
Data Preparation **Intelligence** **Interpretation** **Deployment**

- Data sources
- Data labelling
- Clean up
- Visualize

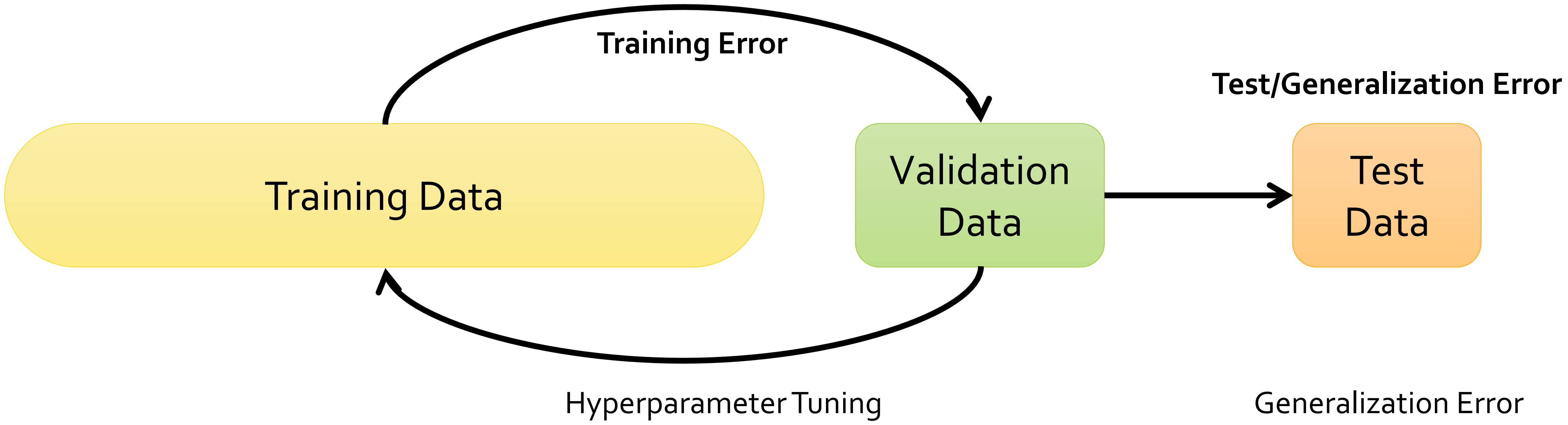
- Analytics
- Machine Learning
- Deep Learning

Data Preparation

Evaluation and design iteration



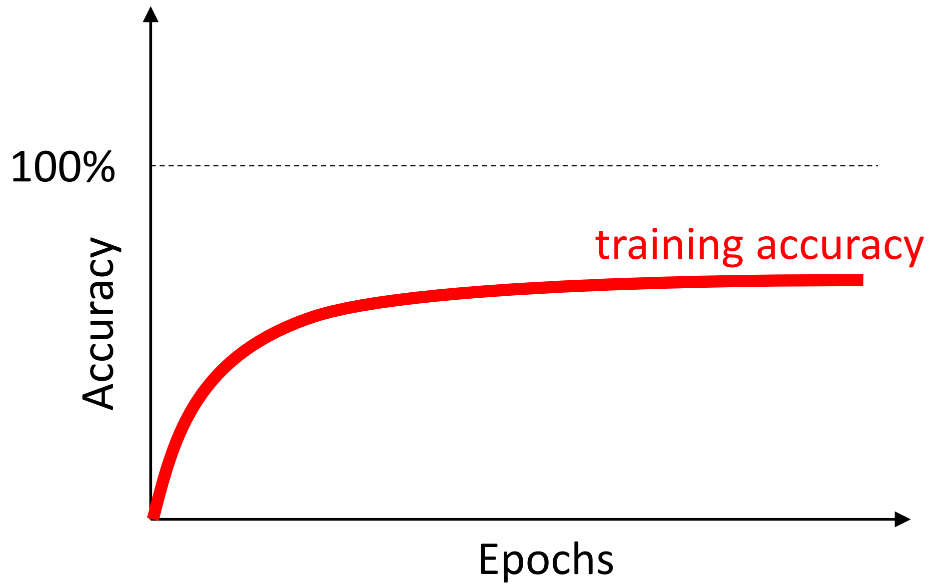
Data Distribution



- **Make the training error small ->** **Model performs well**
- **Make the gap between training and test error small ->** **Model generalizes well**

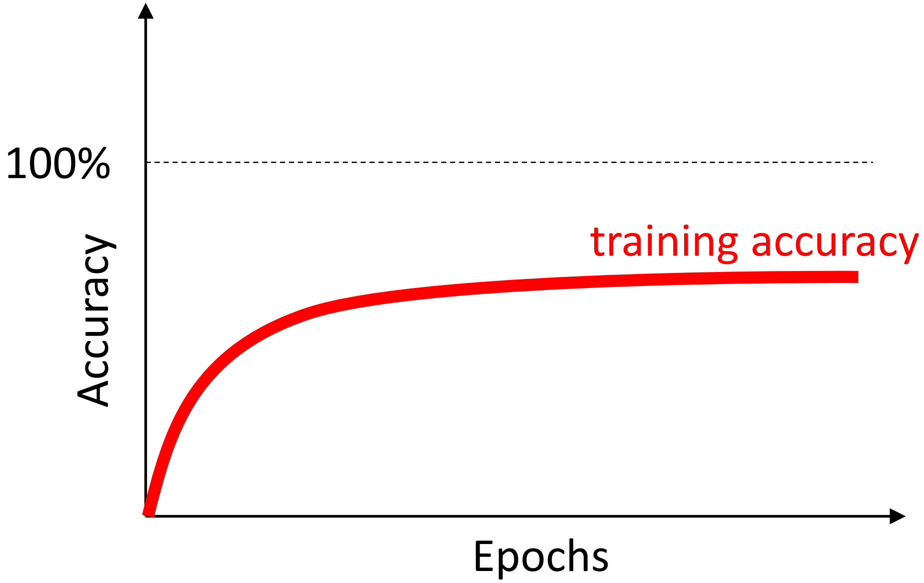
Underfitting

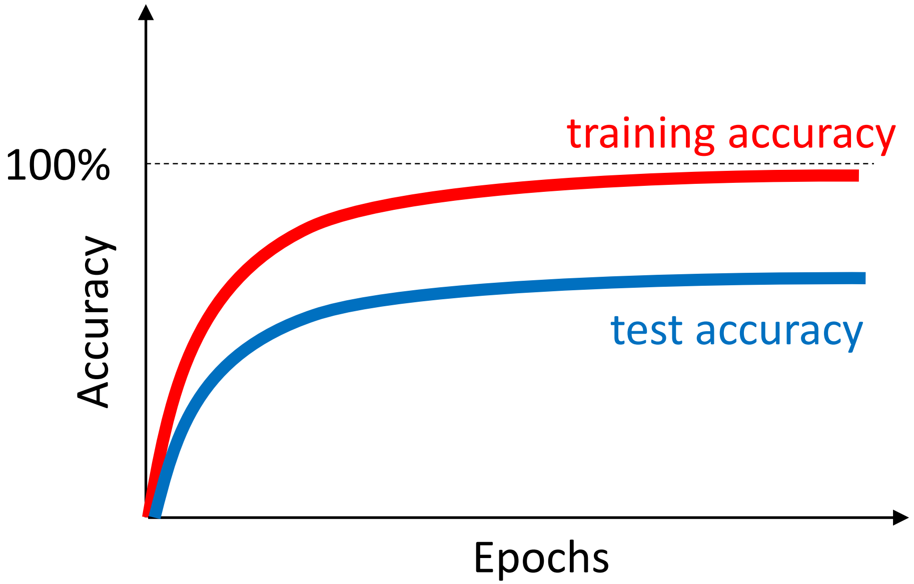
Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set.



Underfitting and Overfitting

Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set.





Overfitting occurs when the gap between the training accuracy and test accuracy is too large.

Regularization – Dropout, L1, L2, Weight Decay

Linear Models - Why?

Supervised Learning

Linear Regression → Logistic Regression → Neuron

- Labels are real numbers
- Interpretability
- Closed-form solution
- Gradient descent

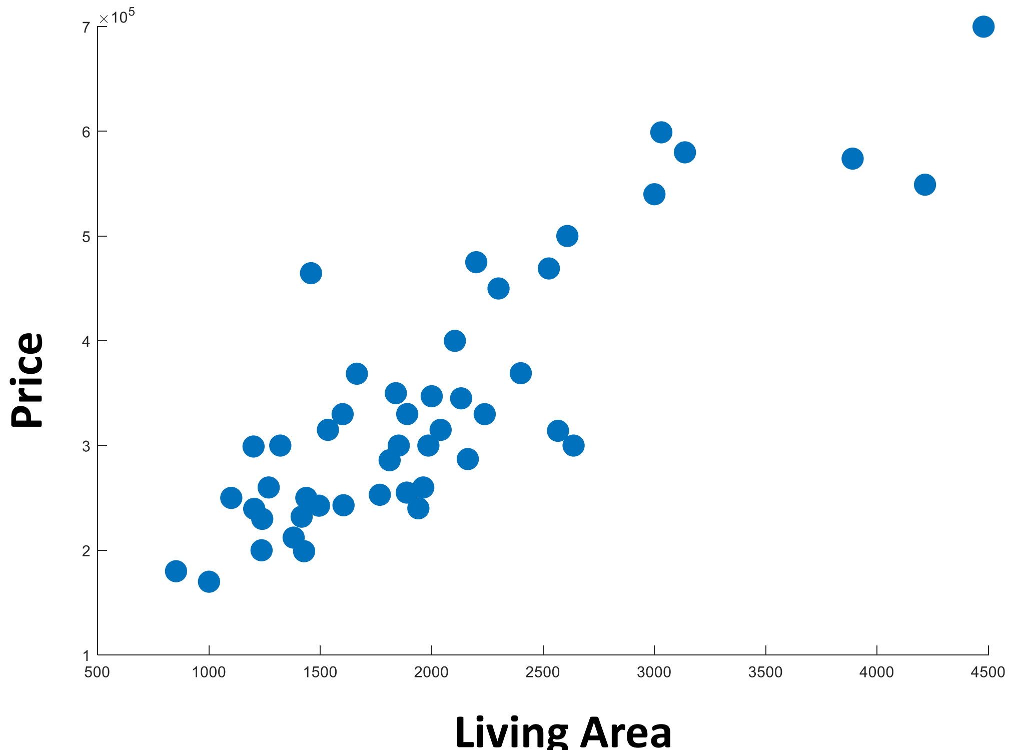
- Labels are categories
- Classification problem
- Nonlinearity
- Gradient descent

Neural networks are stacked layers of neurons

Linear Regression

- We have a dataset giving the living areas and prices of houses

Living Area	Price $K
2104	400
1600	330
2400	369
⋮	⋮

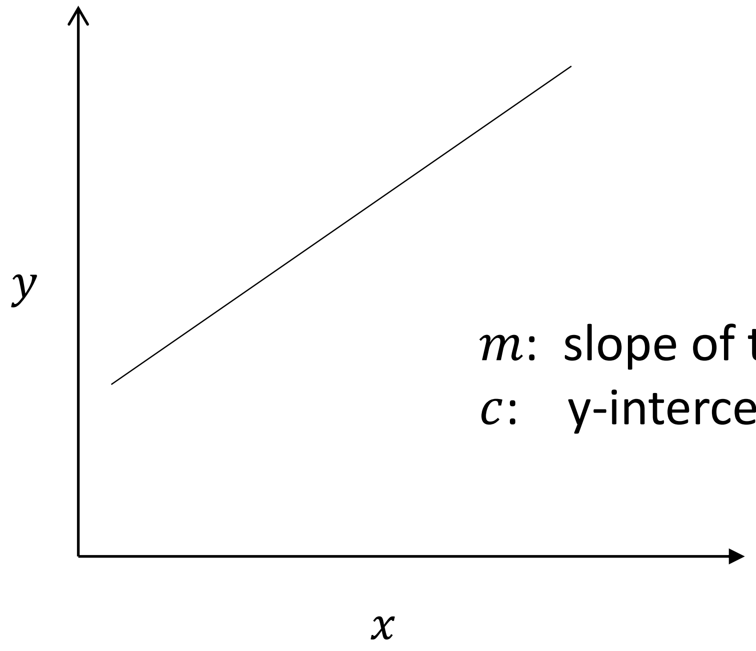


Linear Model

- Lets try to make a linear model

$$h_w(x) = w_0 + w_1 x$$

$$y = mx + c$$



m: slope of the line
c: y-intercept

Notations

- $x^{(i)}$ is the i-th input example (feature) with $y^{(i)}$ as the target or the label.
- Together $(x^{(i)}, y^{(i)})$ are referred to as the i-th training example.
- $\{(x^{(i)}, y^{(i)}); i = 1, \cdots, m\}$ is the training data set.
- $\mathcal{X}$ is the space of input values, and $\mathcal{Y}$ the space of labels/targets, $\mathcal{X} = \mathcal{Y} = \mathbb{R}$

Our goal

$$h : \mathcal{X} \mapsto \mathcal{Y}$$

so that $h(x)$ is a good predictor of y

More features

- We may have more features, living area and number of bedrooms, so we have $x_1^{(i)}$, and $x_2^{(i)}$. Feature space is now two-dimensional, $x \in \mathbb{R}^2$.
- Who decides how many features we have?

Lets try to make a linear model

$$h_w(x) = w_0 + w_1 x_1 + w_2 x_2$$

$$h(x) = \sum_{i=0}^{n} w_i x_i = w^T x$$

x_0 is set to 1.

w_i are the parameters or weights and w is the vector of parameters/weights. x is the vector on inputs.

What are known and what are unknowns?

The unknown w and the Least-Squares

Given training set $\{(x^{(i)}, y^{(i)}), i = 1, \cdots, m\}$, how do we learn the parameters w such that $h(x)$ is close to the ground truth y.

Choose some w randomly and test all examples and compare $h(x)$ and y.

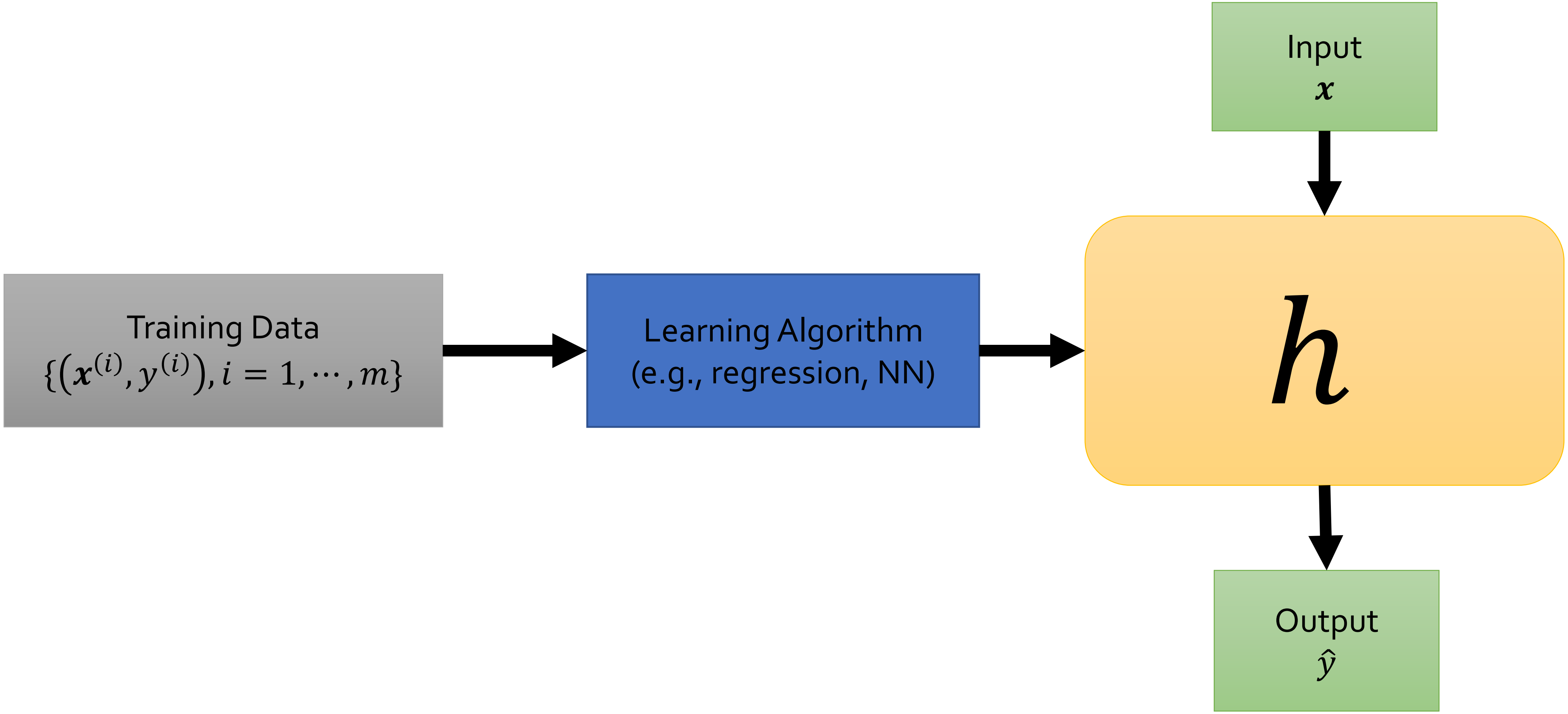
Lets define cost function

$$J(w) = \frac{1}{2}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{i}\right)^2$$

This is a least-squares cost function and referred to as the ordinary least-squares (OLS) regression.

Rowan University

Machine Learning - Supervised Learning



Linear Regression - cost function

$$J(w) = \frac{1}{2}\boxed{\sum_{i=1}^{m}\boxed{\left(\boxed{h\left(x^{(i)}\right)} - \boxed{y^{i}}\right)^{2}}}$$

$$h(x) = \sum_{i=0}^{n} w_i x_i = w^T x$$

Square Error

Ground truth

Sum over all given examples

Goal : Minimize $J(w)$

Gradient Descent

We want to choose w so as to minimize $J(w)$

Starts with some “guess” for w, and then repeatedly change w to make $J(w)$ smaller.

The gradient descent algorithm starts with some initial w, and repeatedly performs the update:

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w) \qquad j = 1, \cdots, n$$

α is the learning rate (more on this later)

For a single training example, we may solve and get:

$$w_j := w_j - \alpha\left(h_w\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

LMS (Least Mean Squares) Update Rule

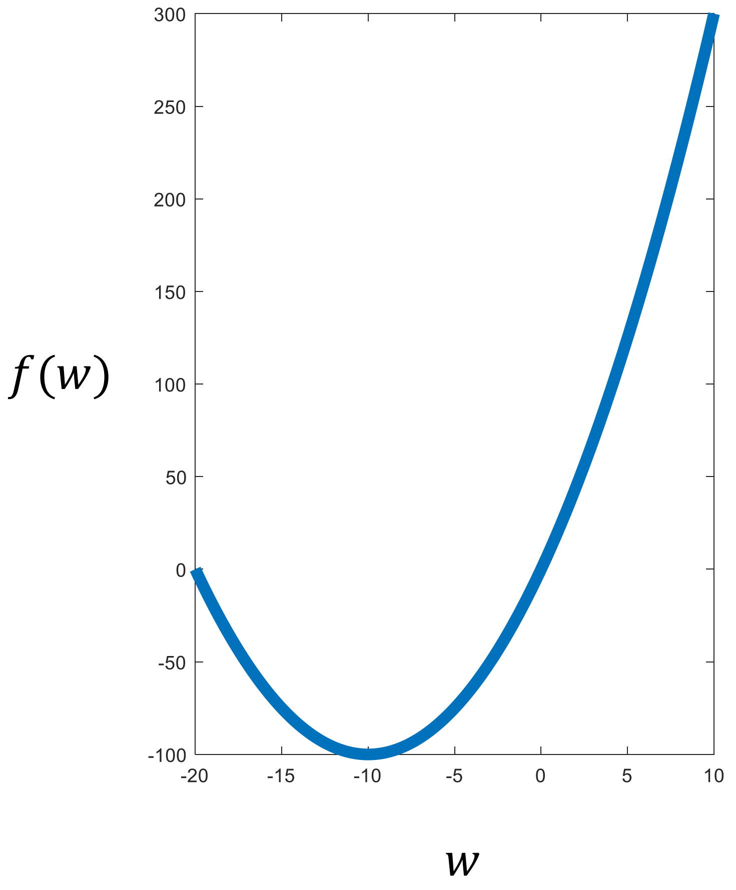
Gradient descent: But how?

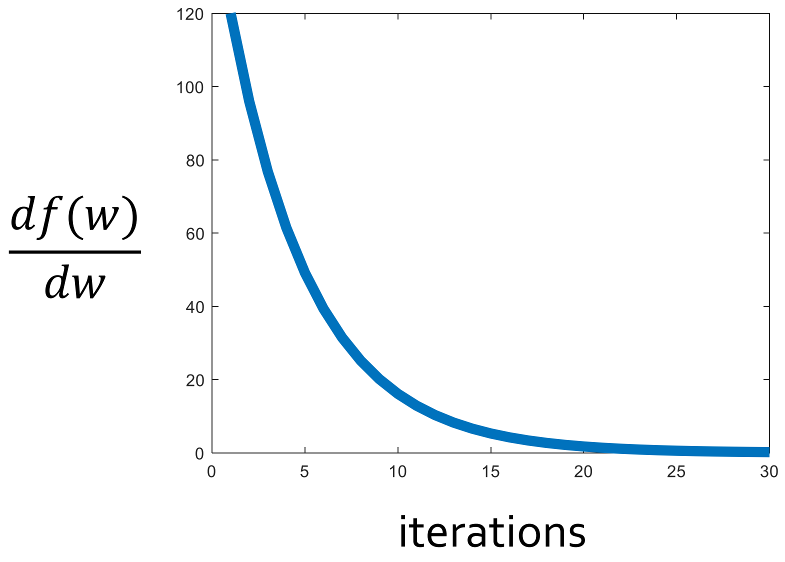
$$f(w) = w^2 + 20w$$

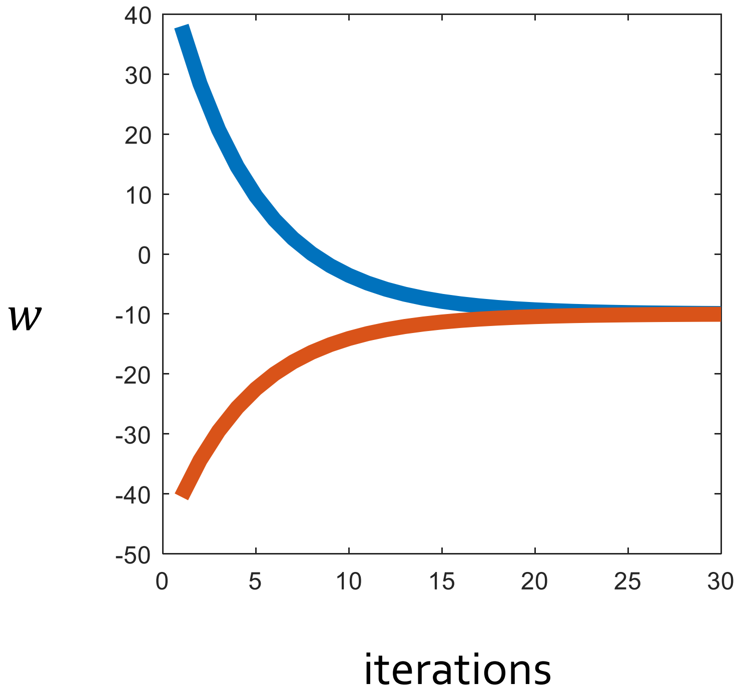
$$\frac{df(w)}{dw} = 2w + 20$$

$$\alpha = 0.1$$

$$w := w + \alpha\left(-\frac{df(w)}{dw}\right)$$







Batch gradient descent

Loop till convergence:

$$w_j := w_j - \alpha \sum_{i=1}^{m} \boxed{\left(h_w\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}}$$

$$\frac{\partial}{\partial w_j} J(w)$$

This method looks at every example in the entire training set on every step

Rowan University

Stochastic gradient descent

Loop:{

for $i = 1{:}m$ {

$$w_j := w_j - \alpha\left(h_w\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \quad \text{for all } j$$

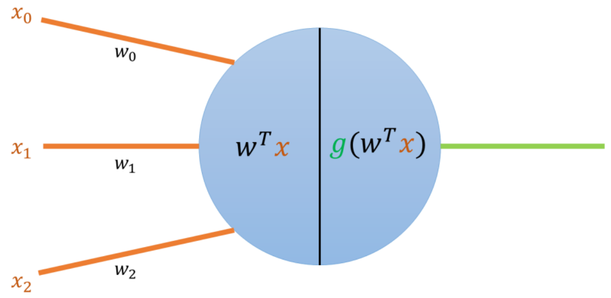
}

}

Batch gradient descent has to scan through the entire training set before taking a single step of updating w – a costly operation if m is large

Stochastic gradient descent can start making progress right away, and continues to make progress with each example it looks at.
Stochastic gradient descent may get w close to the minimum much faster than batch gradient descent.
It may never converge to the minimum, that is, w may keep oscillating around the minimum of $J(w)$

Logistic Regression



Labels are now discrete categories, e.g.

$y \in \{0, 1\}$ – a binary classification problem

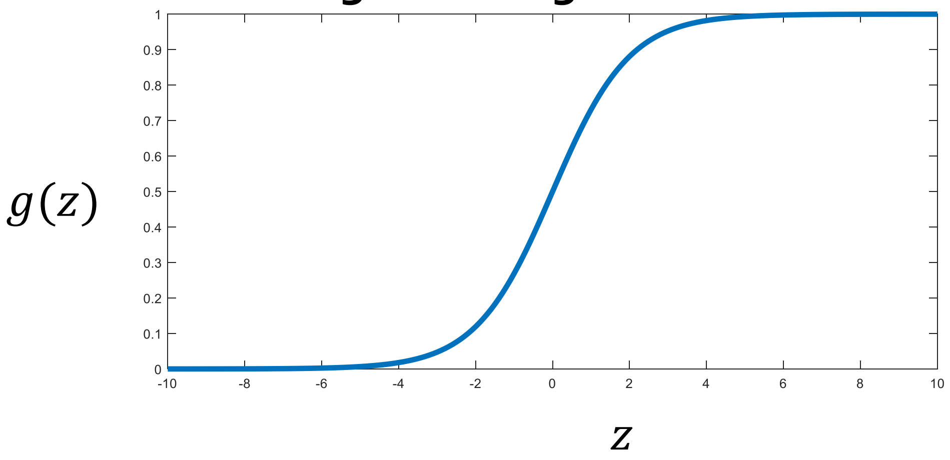
Previously : $h_w(x) = w^T x$

Now: $h_w(x) = g(w^T x)$ such that

$h_w(x) \in \{0,1\}$

$$g(z) = \frac{1}{1 + \exp(-z)}$$

Logistic or Sigmoid Function



Logistic regression: How to estimate w?

Lets start with probability assignment:

$$P(y = 1|x; w) = h_w(x) \text{ and thus } P(y = 0|x; w) = 1 - h_w(x)$$

$$p(y|x; w) = [h_w(x)]^y [1 - h_w(x)]^{1-y}$$

For m independent training examples:

$$L(w) = \prod_{i=1}^{m} \left[h_w\left(x^{(i)}\right)\right]^{y^{(i)}} \left[1 - h_w\left(x^{(i)}\right)\right]^{1-y^{(i)}}$$

$$l(w) = \log L(w) = \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + \left(1 - y^{(i)}\right) \log(1 - h(x^{(i)}))$$

Rowan University

How do we maximize it? Gradient Ascent

Logistic Regression Loss Function

Inputs: x, y
Parameters: w and b
Output : $\hat{y}$

$$J(w,b) = -\sum_{i=1}^{m} y^{(i)} \log(\hat{y}^{(i)}) + \left(1 - y^{(i)}\right) \log(1 - \hat{y}^{(i)})$$

$$\hat{y}^{(i)} = \sigma\left(w^T x^{(i)} + b\right) = \frac{1}{1+\exp(-w^T x^{(i)}+b)}$$

w and b can be combined

$$w^T x + b = [w_1, w_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b$$

$$= [b, w_1, w_2]\begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

$y^{(i)} = 1$ and $h\left(x^{(i)}\right) = 0.999$ Loss ≈ 0

$y^{(i)} = 0$ and $h\left(x^{(i)}\right) = 0.001$ Loss ≈ 0

$y^{(i)} = 1$ and $h\left(x^{(i)}\right) = 0.001$ Loss ≈ 3

$y^{(i)} = 0$ and $h\left(x^{(i)}\right) = 0.999$ Loss ≈ 3

Loss function / Cost function

- Function of the unknown weights, w, and b
- Differentiable
- Must output a scalar

Rowan University

Linear Regression and Logistic Regression

When the target variable (y) we are trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. We can solve this problem using linear models and this is referred to as Linear Regression.

Examples???

When y can take on only a small number of discrete values or categories, (e.g., low-range, mid-range, expensive, crazy-expensive), we call it a classification problem. We can solve this problem using linear models, this is referred to as Logistic Regression.

Examples???

Logistic Regression – 1 example case

b

$$\frac{dJ}{db} = \frac{dJ}{dz}$$

w_1

x_1

$$\frac{dJ}{dw_1} = x_1 \frac{dJ}{dz}$$

w_2

x_2

$$\frac{dJ}{dw_2} = x_2 \frac{dJ}{dz}$$

$$z = w_1 x_1 + w_2 x_2 + b$$

$$\frac{dJ}{dz} = \frac{dJ}{da}\frac{da}{dz}$$

$$\frac{dJ}{dz} = \left(-\frac{y}{a} + \frac{1-y}{1-a}\right) * a(1-a)$$

$$\frac{dJ}{dz} = a - y$$

$$a = \sigma(z)$$

$$\frac{dJ}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

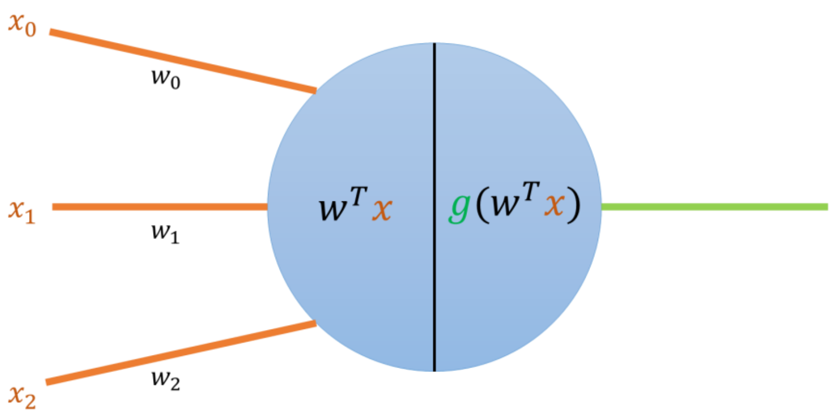
$$J(w,b) = -y\log(a) - (1-y)\log(1-a)$$

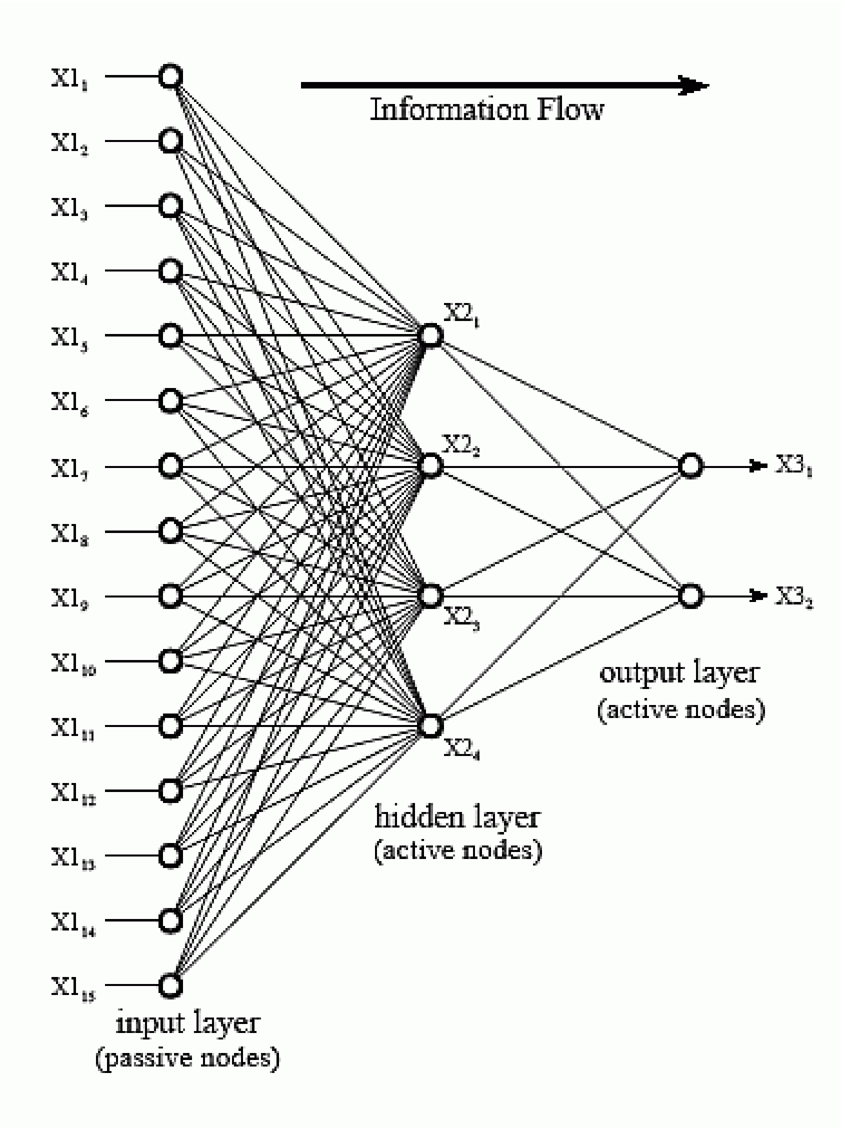
$$w_1 := w_1 - \alpha \frac{dJ}{dw_1}$$

$$w_2 := w_2 - \alpha \frac{dJ}{dw_2}$$

$$b := b - \alpha \frac{dJ}{db}$$

Logistic Regression to Neuron

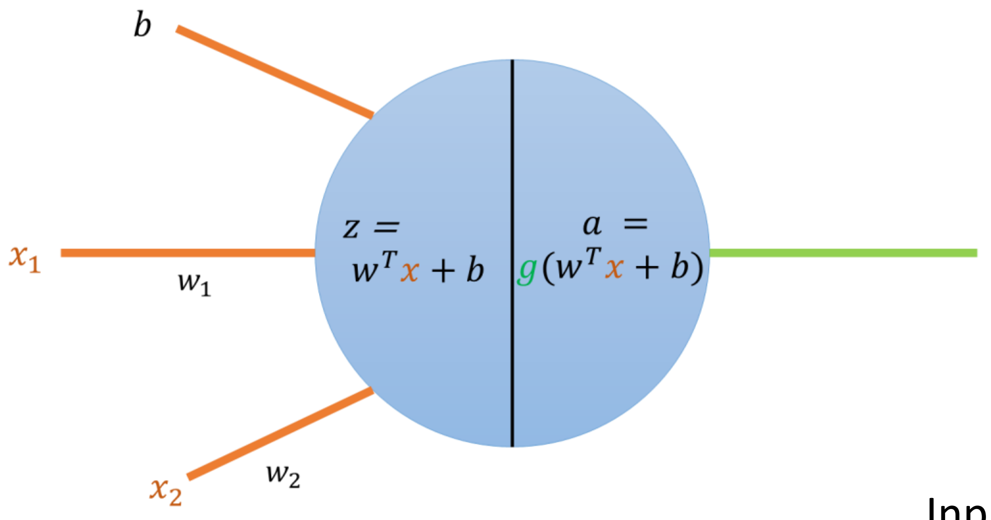


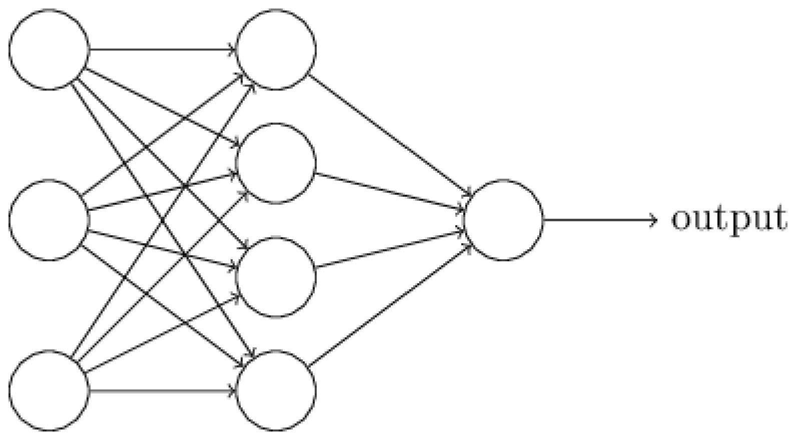


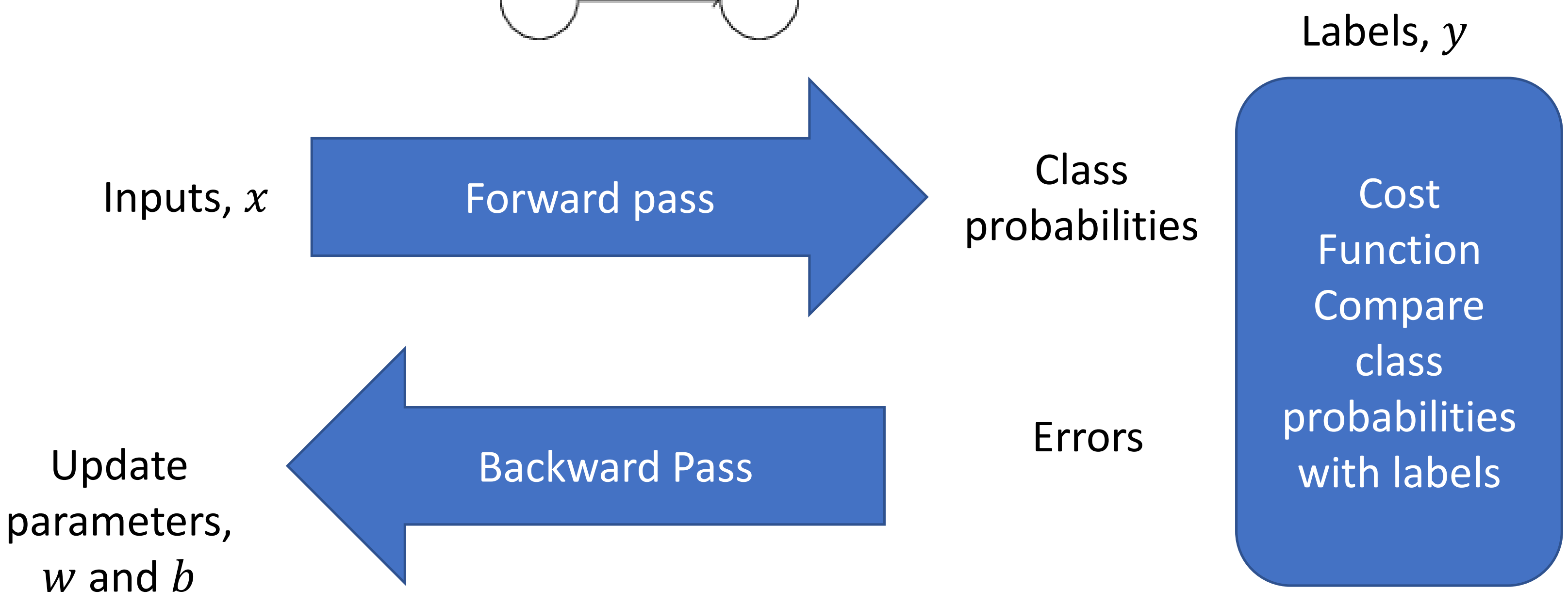
Neural Network Examples

Applications	Input	Output	Neural Network
Housing Example			Simple NN
Iris Classification			Simple NN
Image classification			CNN
Tumor Segmentation			CNN
Speech translation			RNN / LSTM
Sentiment analysis			RNN / LSTM
Autonomous cars			Customized Architectures

Neuron to Artificial neural network







http://www.dspguide.com/ch26/2.htm

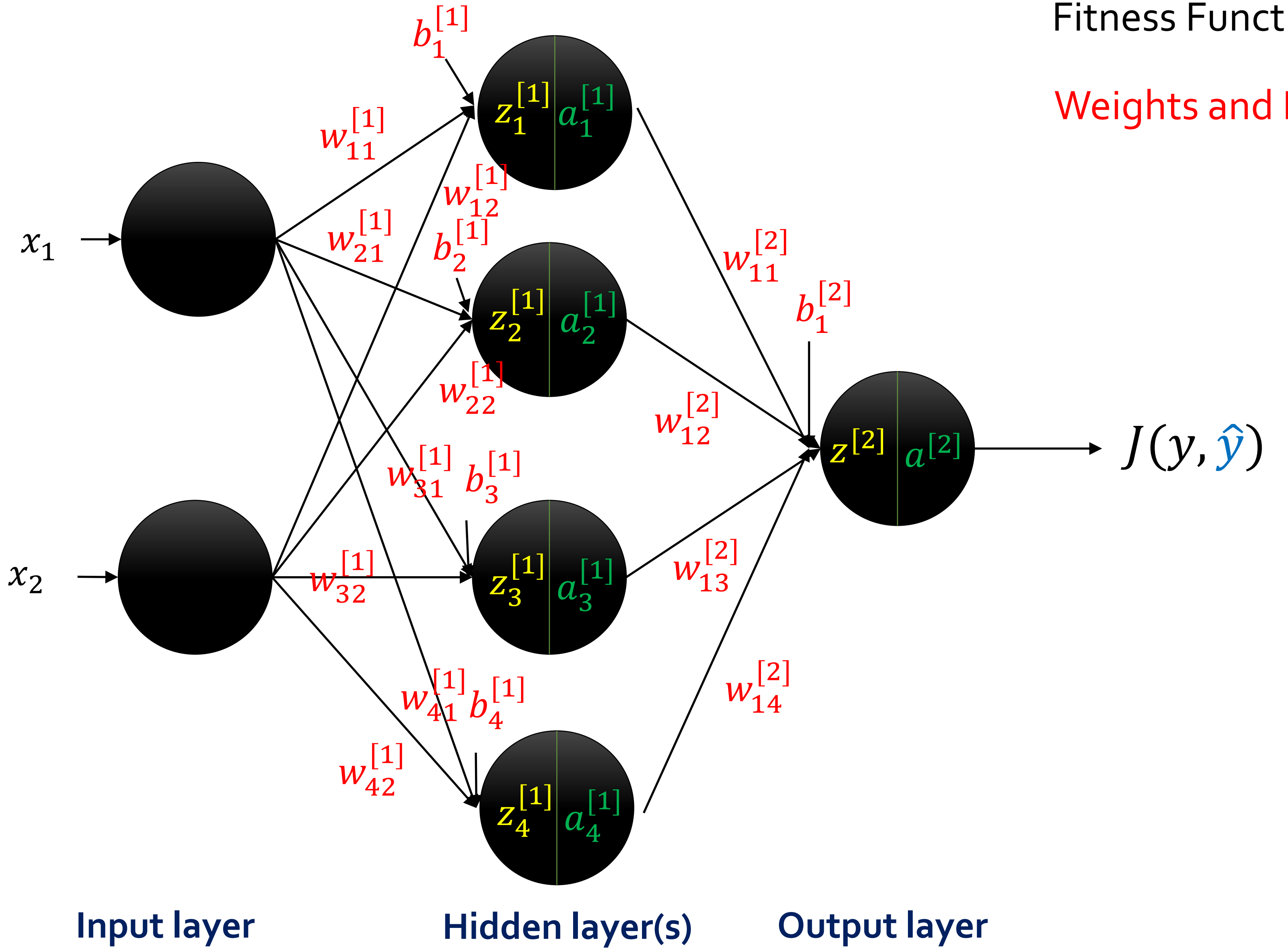
Artificial Neural Networks

Input Data = $\boldsymbol{x}, y$

Out Data = $\hat{y}$

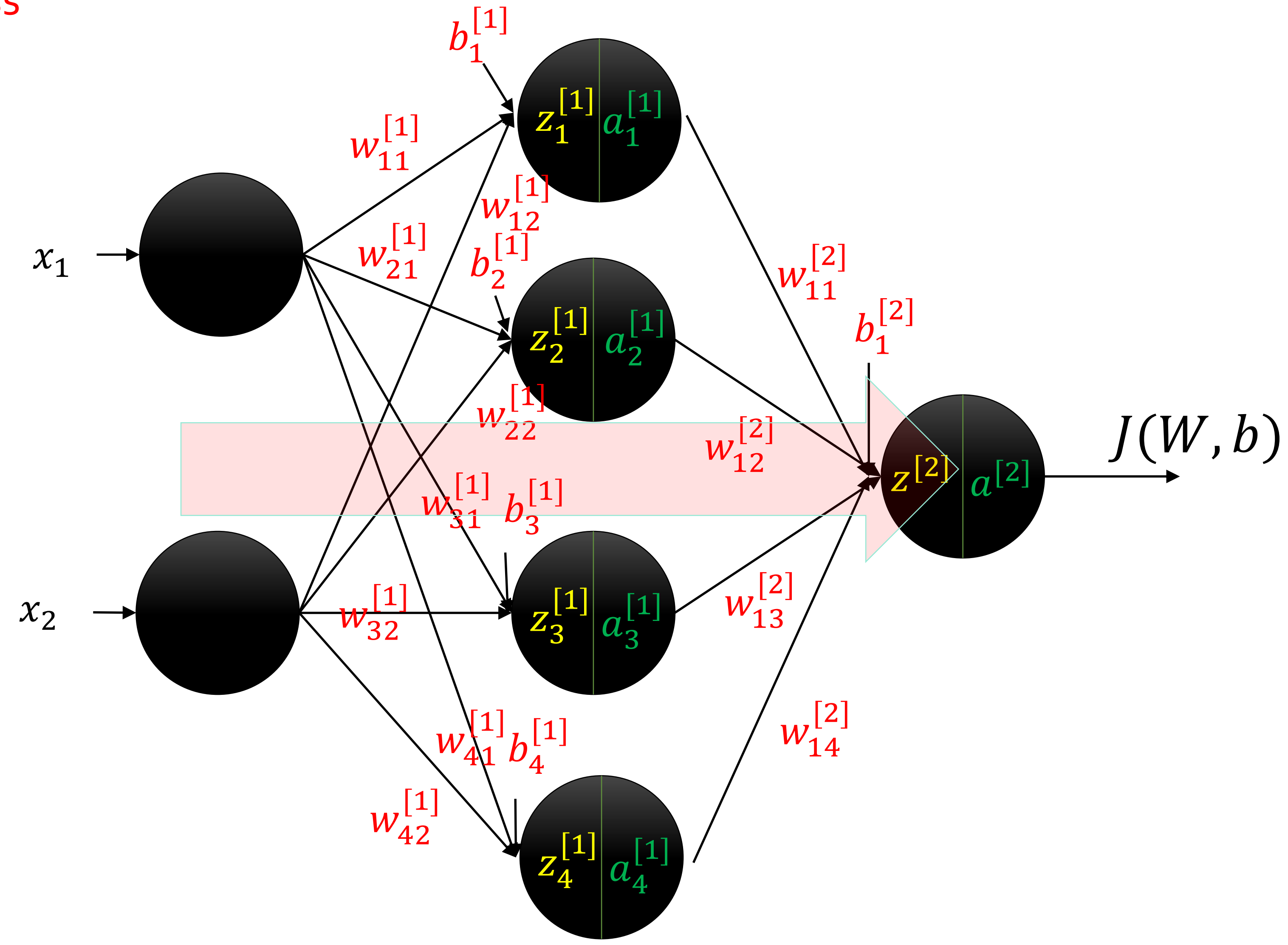
Fitness Function = $J(y, \hat{y})$

Weights and Biases = ? ? ?



Artificial Neural Networks

Forward Pass



$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$

$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(Z^{[2]})$$

$$X = \begin{bmatrix} | & | & | \\ x^{(1)} & x^{(2)} & x^{(m)} \\ | & | & | \end{bmatrix}$$

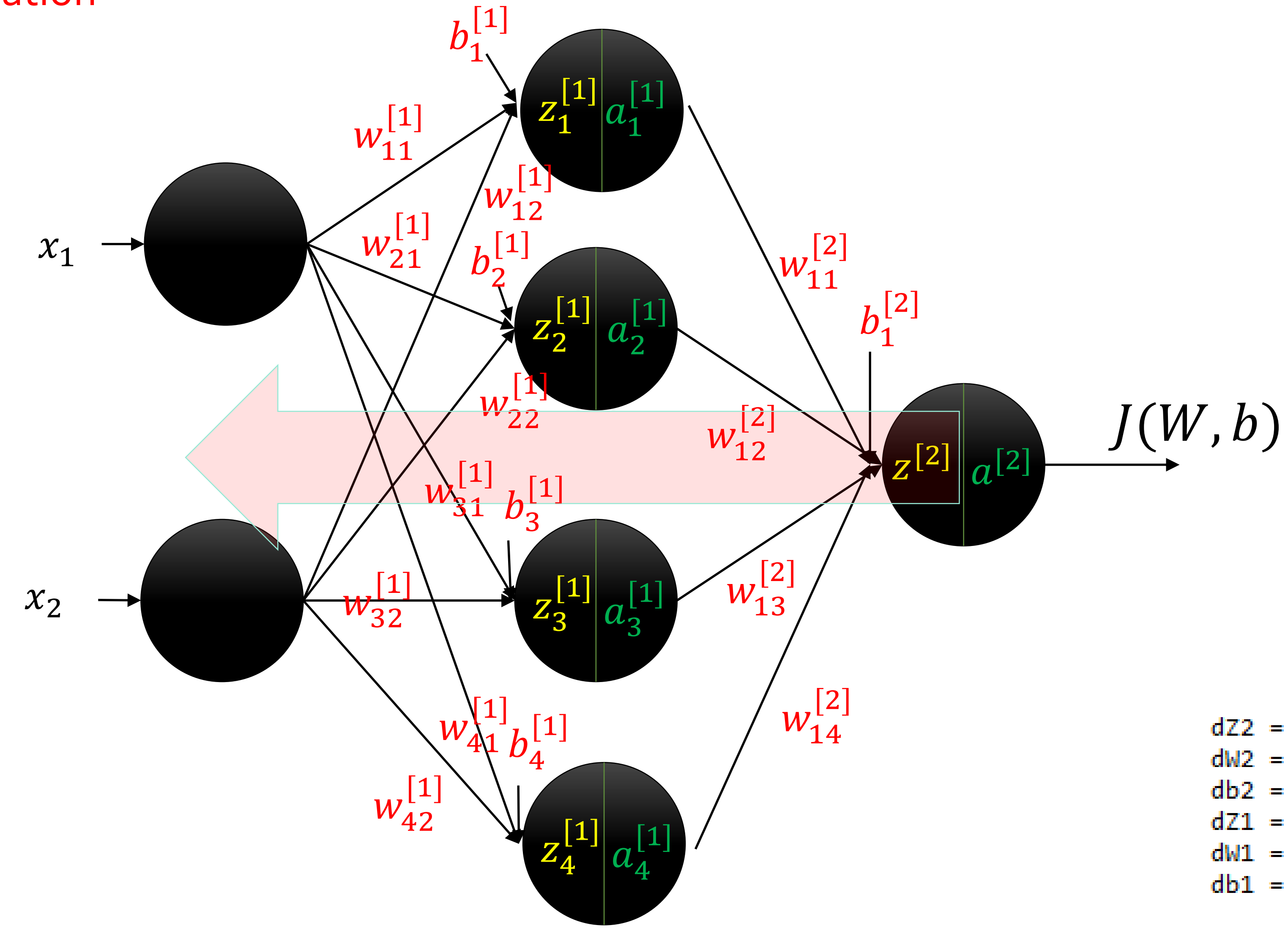
$$Z^{[1]} = \begin{bmatrix} | & | & | \\ z^{[1](1)} & z^{[1](2)} & z^{[1](m)} \\ | & | & | \end{bmatrix}$$

$$A^{[1]} = \begin{bmatrix} | & | & | \\ a^{[1](1)} & a^{[1](2)} & a^{[1](m)} \\ | & | & | \end{bmatrix}$$

```
Z1 = np.dot(W1, X) + b1
A1 = sigmoid(Z1)
Z2 = np.dot(W2, A1) + b2
A2 = sigmoid(Z2)
```

Artificial Neural Networks

Backpropagation

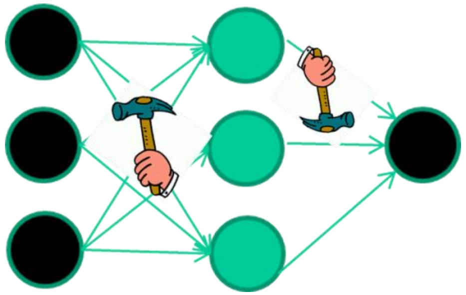


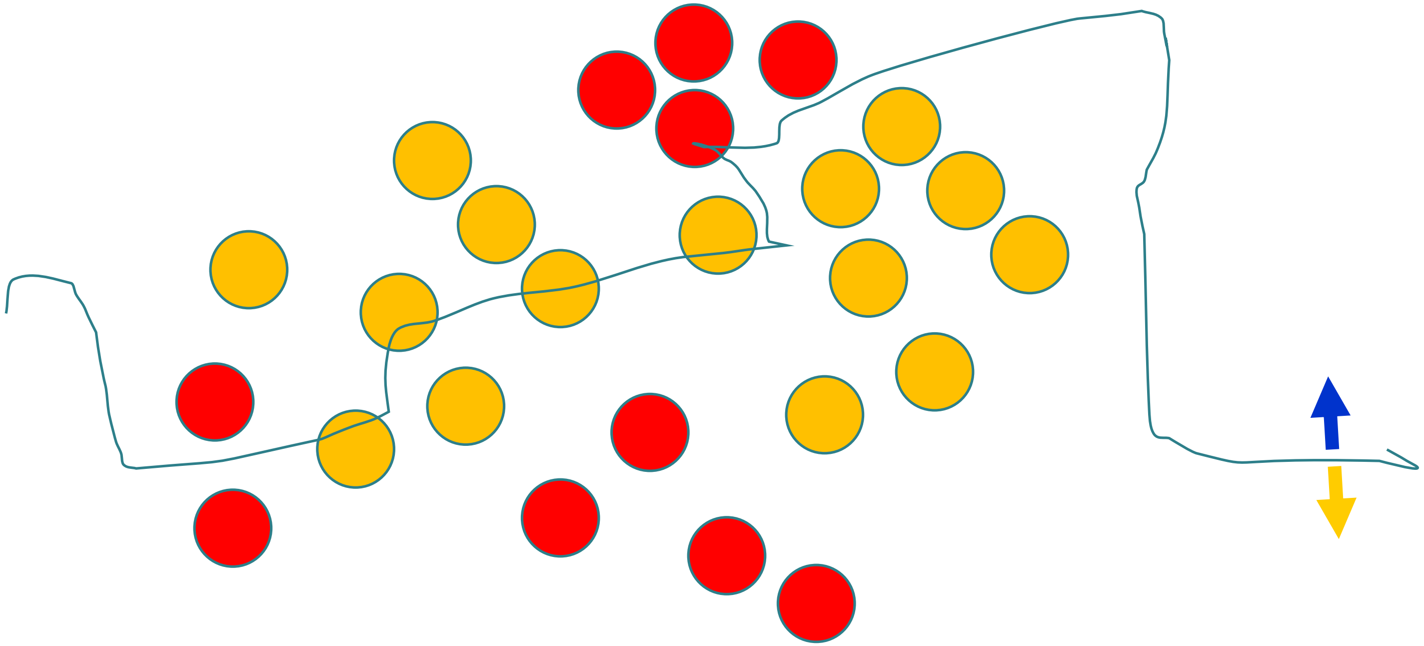
$$dZ^{[2]} = A^{[2]} - Y$$
$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$
$$db^{[2]} = \frac{1}{m} \sum dZ^{[2]}$$
$$dZ^{[1]} = W^{[2]T} dZ^{[2]} * \acute{\sigma}(Z^{[1]})$$
$$dW^{[1]} = dZ^{[1]} X^T$$
$$db^{[1]} = \frac{1}{m} \sum dZ^{[1]}$$

```
dZ2 = A2 - Y
dW2 = 1/m*(np.dot(dZ2,A1.T))
db2 = 1/m*(np.sum(dZ2, axis=1, keepdims=True))
dZ1 = np.dot(W2.T, dZ2)* (A1*(1-A1))
dW1 = 1/m*np.dot(dZ1, X.T)
db1 = (1/m) * np.sum(dZ1, axis=1, keepdims=True)
```

Artificial Neural Networks – Training

Classification Task

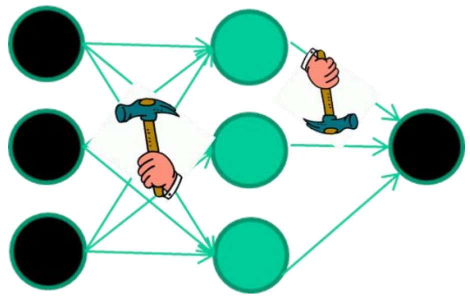


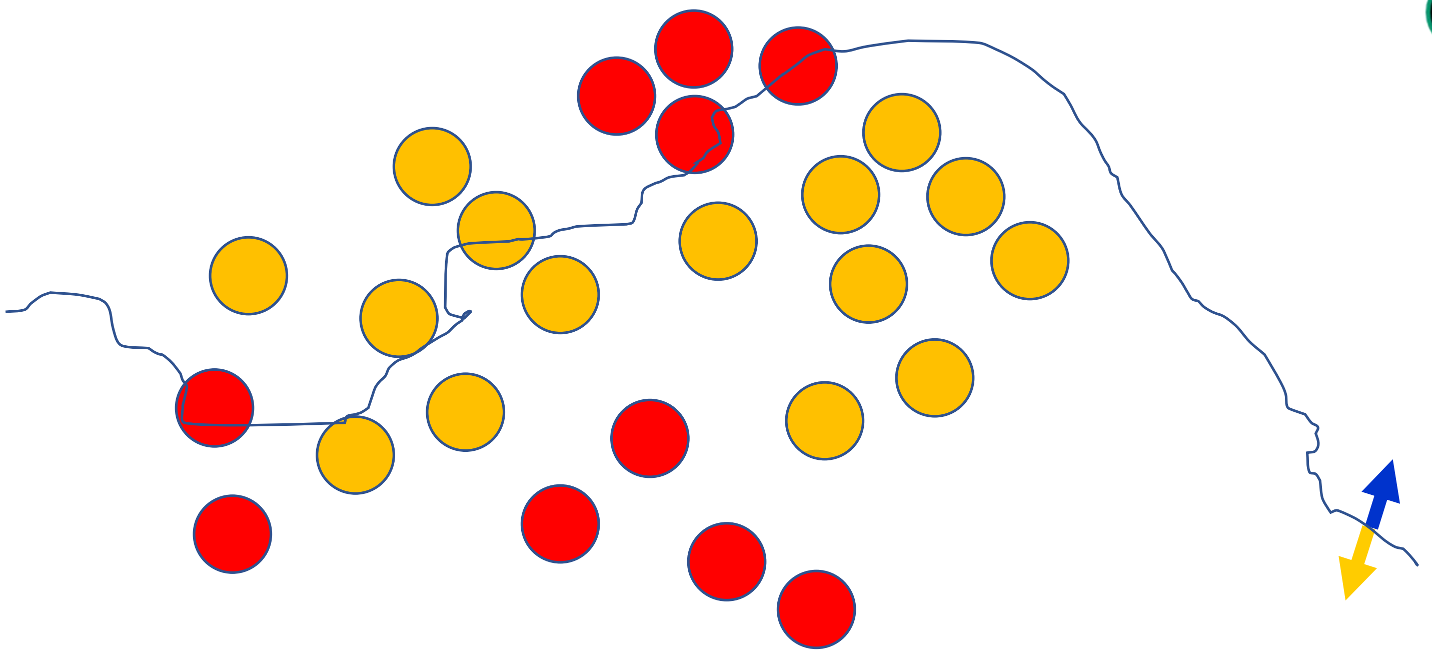


Initial random weights

Artificial Neural Networks – Training

Classification Task

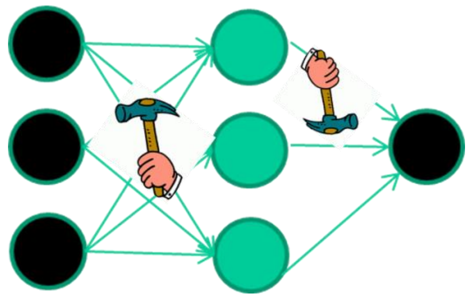


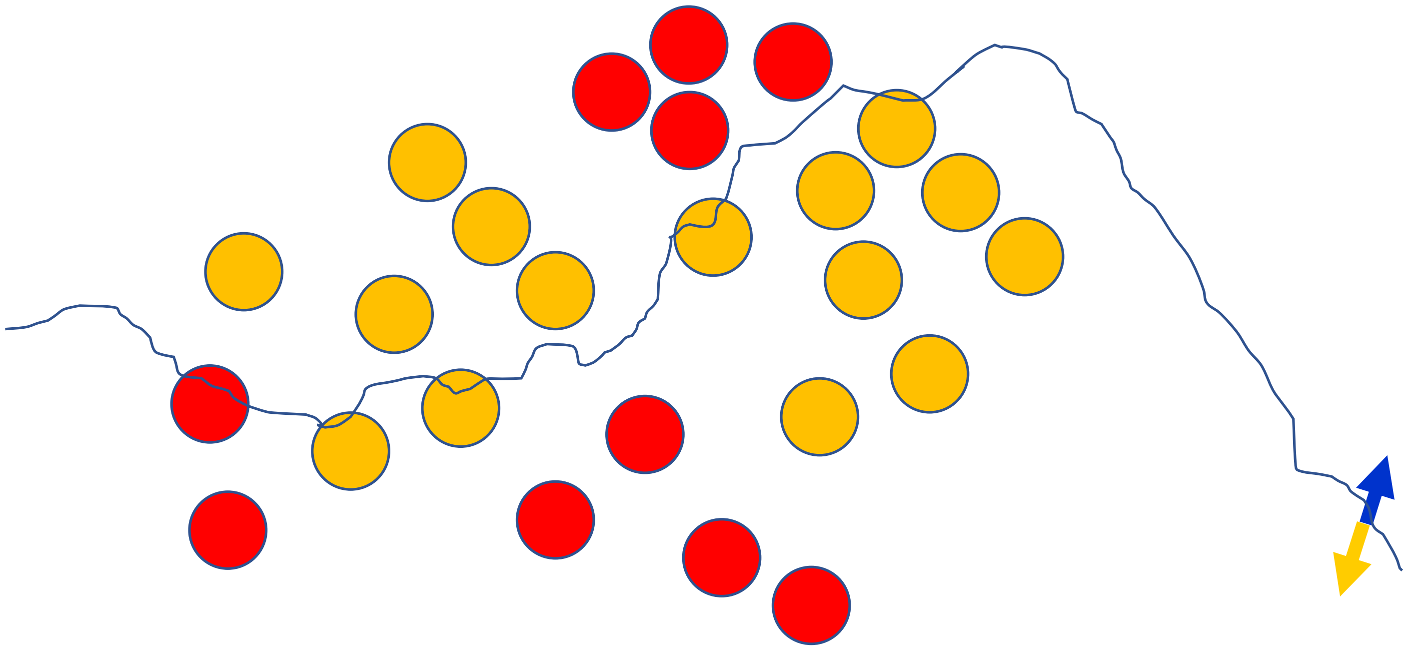


Present a training instance / adjust the weights

Artificial Neural Networks – Training

Classification Task

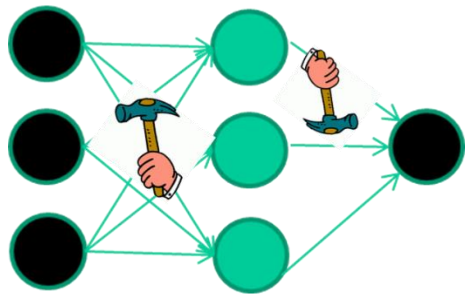


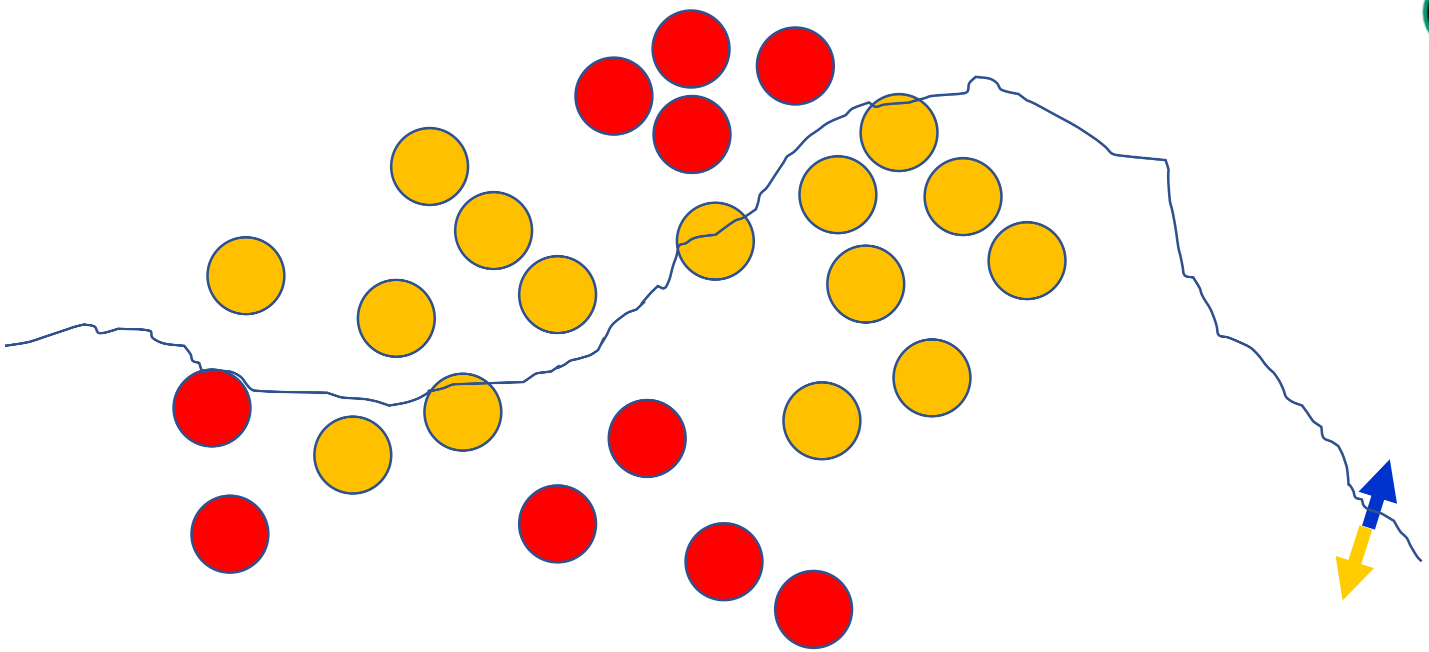


Present a training instance / adjust the weights

Artificial Neural Networks – Training

Classification Task

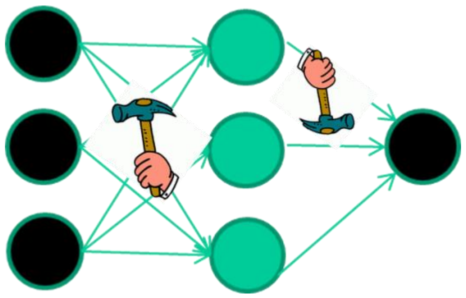


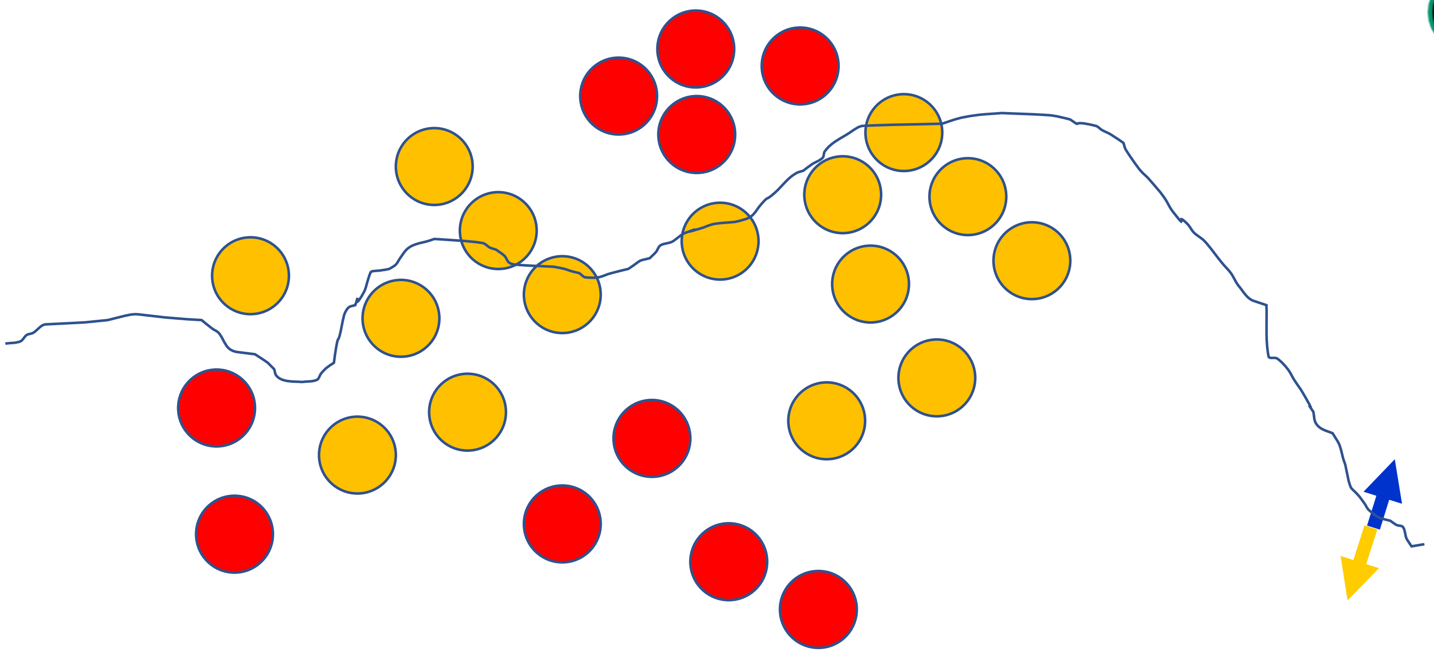


Present a training instance / adjust the weights

Artificial Neural Networks – Training

Classification Task

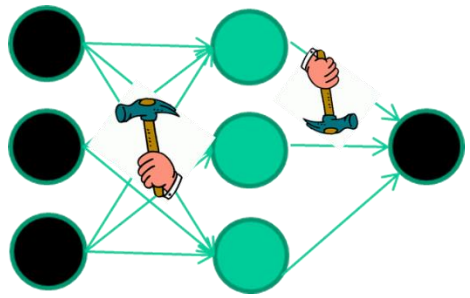


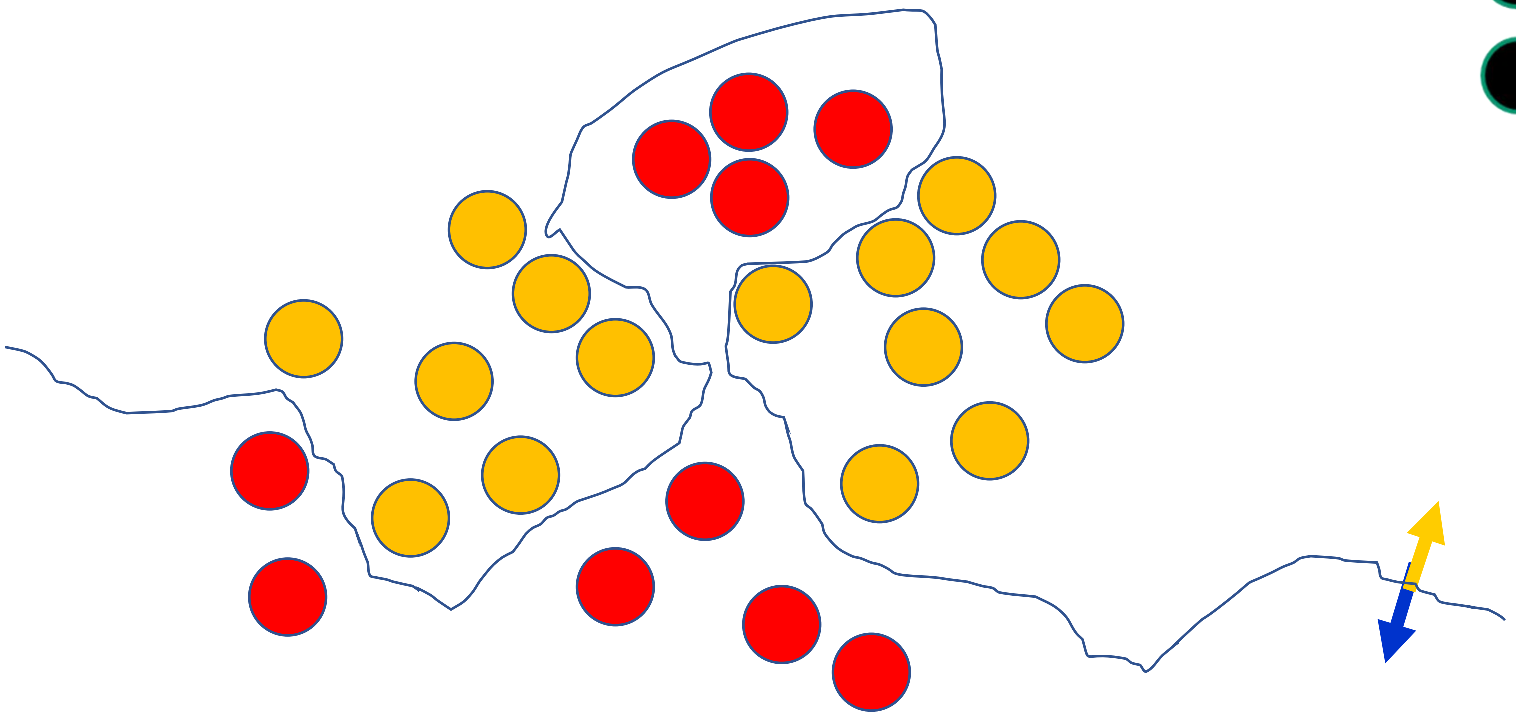


Present a training instance / adjust the weights

Artificial Neural Networks – Training

Classification Task

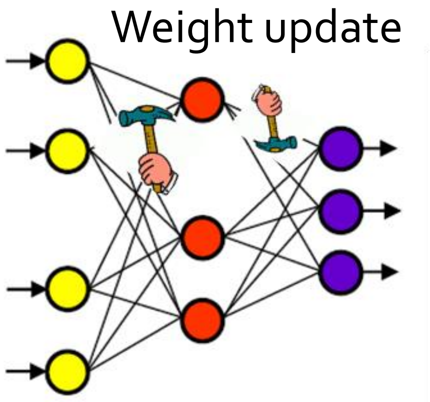




Eventually

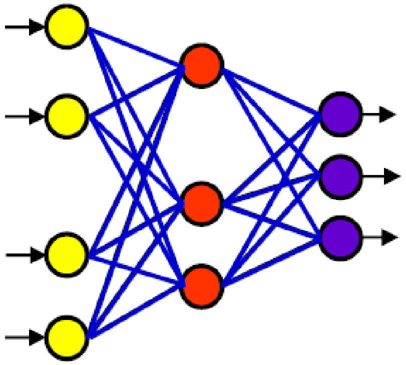
Artificial Neural Networks – Training and Testing

Train



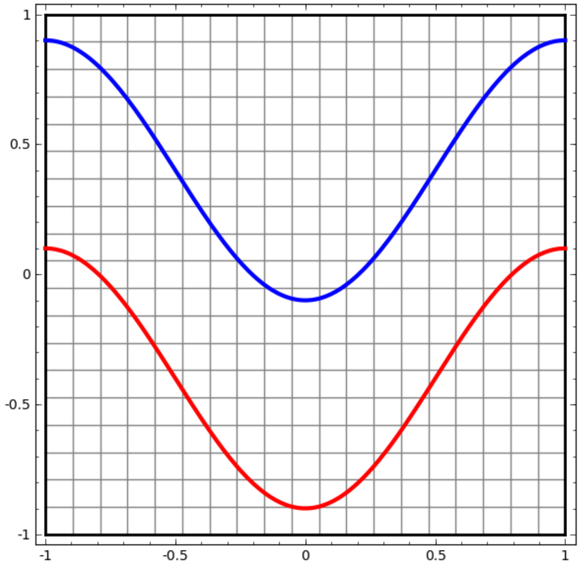
73	73	101
101	134	101
134	73	134

Test

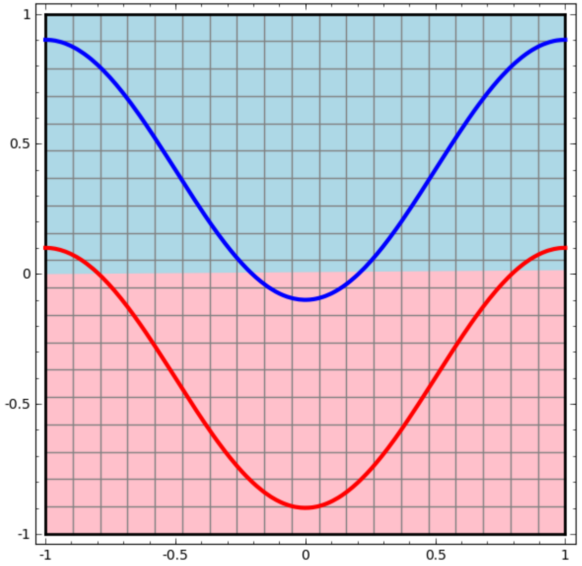


?

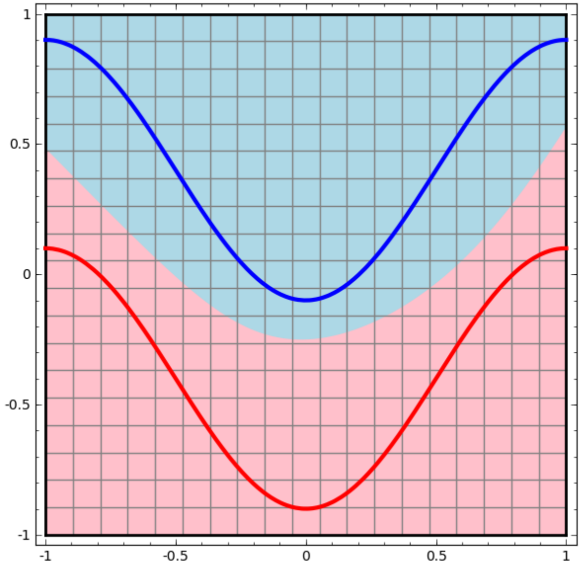
Artificial Neural Networks - Nonlinearity



The network will learn to classify points as belonging to one or the other.

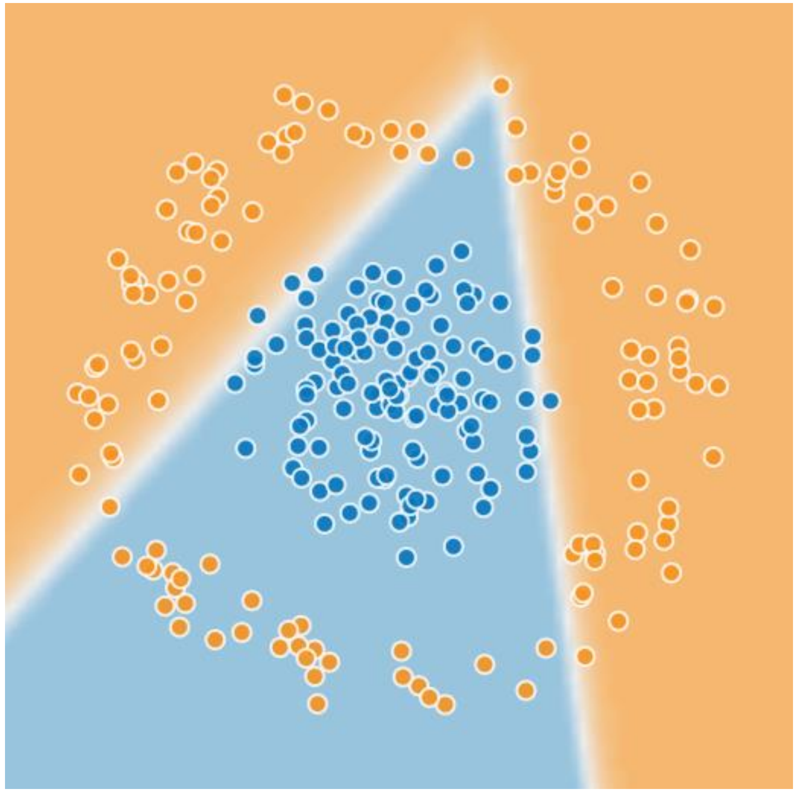


ANN with 1 input layer and 1 output layer. The network simply tries to separate the two classes of data by dividing them with a line.

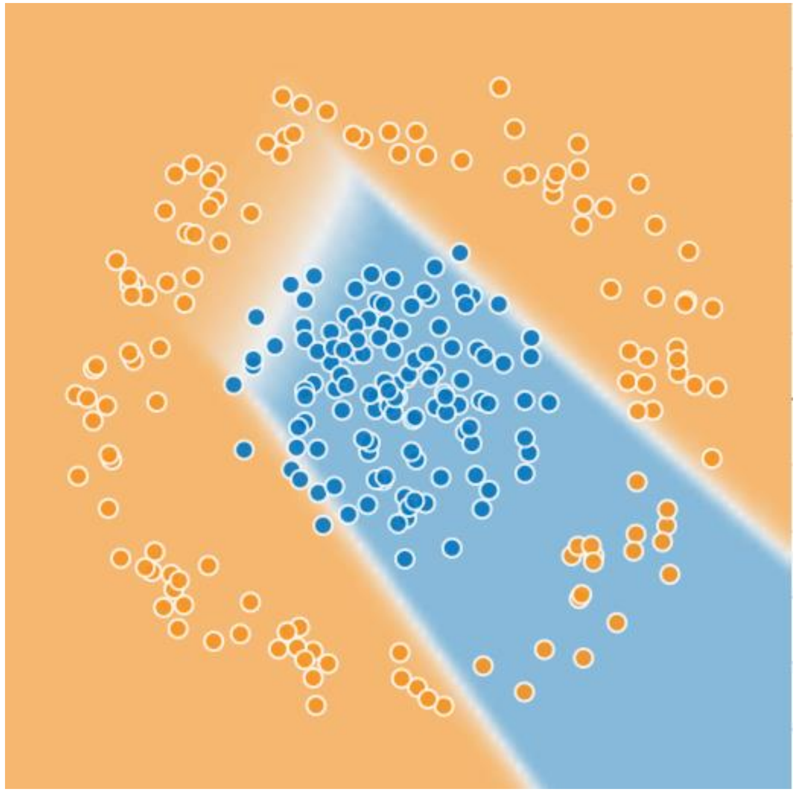


ANN with 1 hidden layer. It separates the data with a more complicated curve than a line.

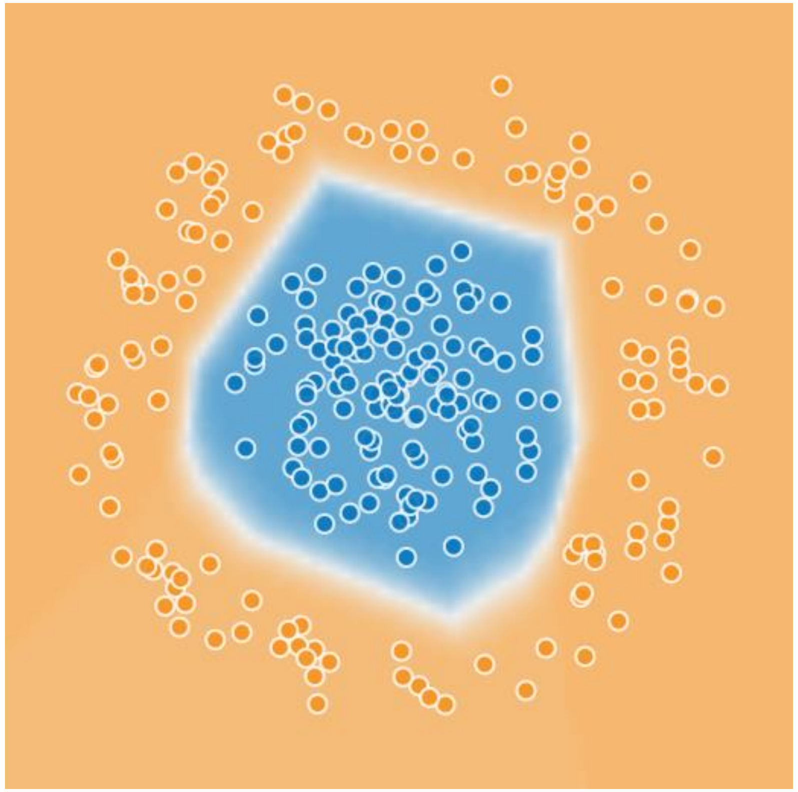
Artificial Neural Networks – Why Go Deep?



1 hidden layer – 2 neurons



2 hidden layers – 4 neurons



3 hidden layers – 9 neurons

Background Math

- The Matrix Calculus You Need For Deep Learning

https://arxiv.org/pdf/1802.01528.pdf

Coding Assignment

- Logistic Regression
- Neural Network
  - Without using any libraries
  - Keras
  - TensorFlow
- Google co-lab

https://colab.research.google.com/drive/1H_F3Fgab3aTH1gBmXHTjlvarBheAJGvy