Deep Learning
Foundations and Algorithms

ADMSI -- Talent Sprint – Module 4.1A

Balaji Srinivasan
Mechanical Engineering
IIT-Madras

List of topics to be covered

Series of 4 classes on Deep Learning foundations

1. Artificial Neural Networks
   - Deep neural Networks
   - Backpropagation

2. Optimization in Neural Networks
   - Variants of Gradient Descent
   - Overfitting and Dropout

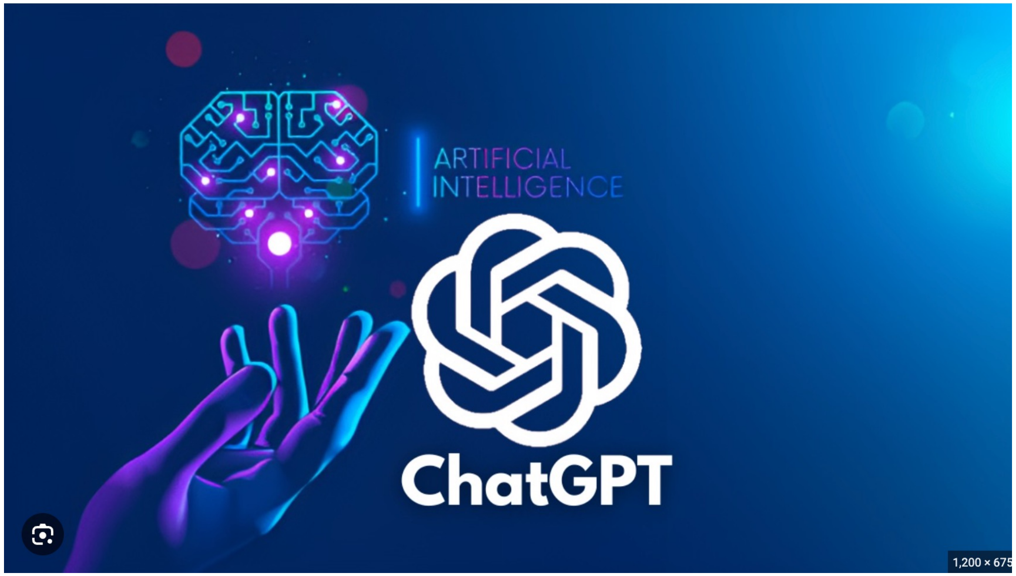
Topics for Today's talk

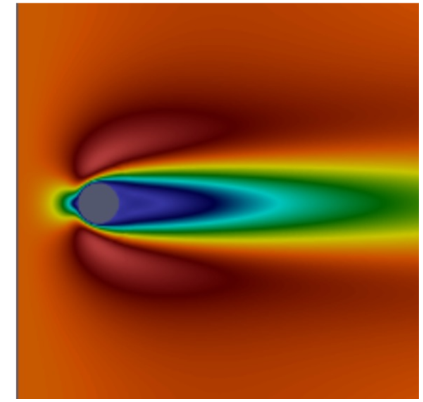
- What makes Machine Learning in general and Deep Learning in specific work?

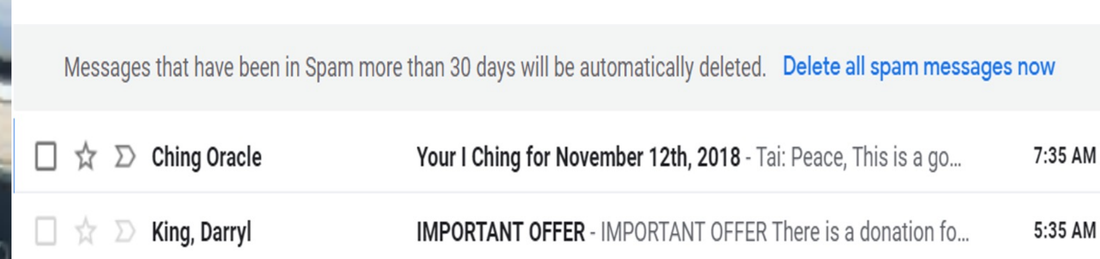
- An introduction to Deep Learning
  - Neural Networks and Linear Models
  - Linear regression and Gradient Descent

Overview of AI and ML

What makes these possible?





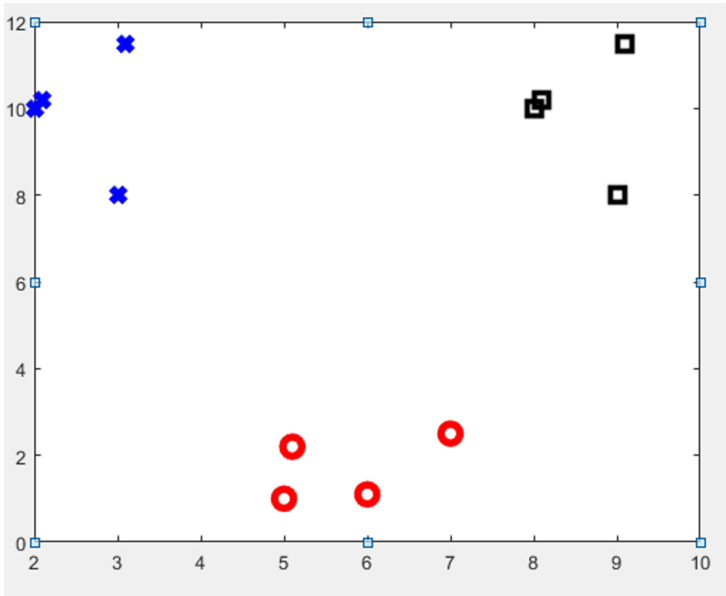


Simplistic Definition – Artificial Intelligence and Machine Learning aim to replicate activities requiring human cognition

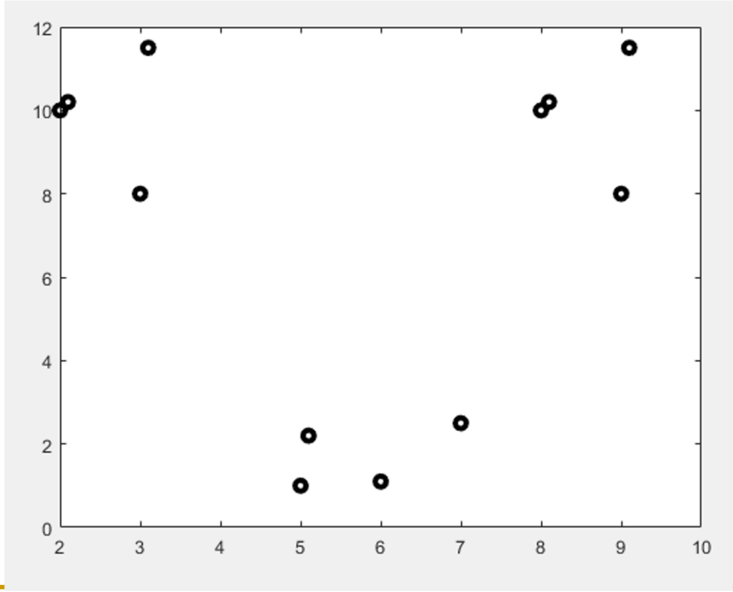
Types of learning approaches

AI
Rules
Learning
DL

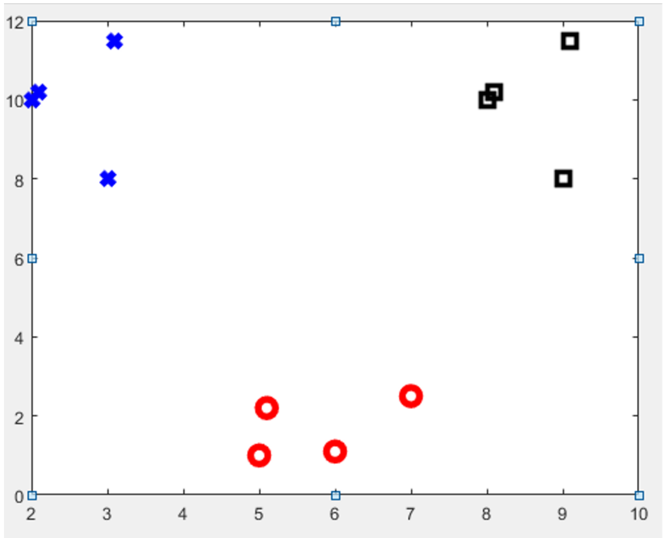
- Supervised Learning
  - Data labeled by human experts
  - Labeling images
  - Speech recognition
  - OCR

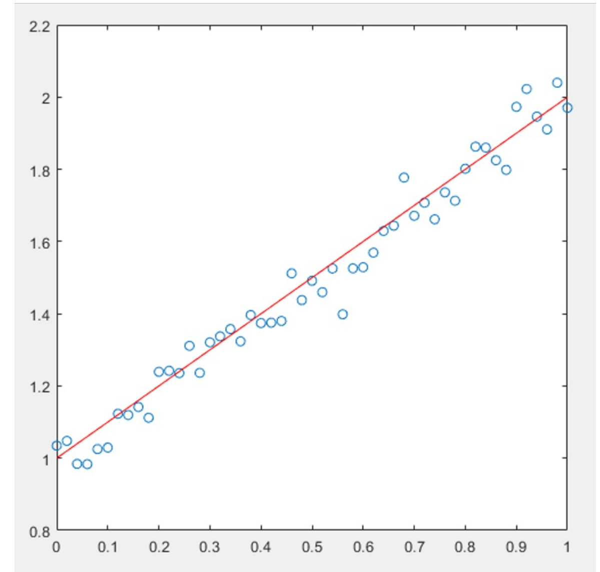


- Unsupervised Learning
  - Unlabeled data
  - Grouping customers
  - Detecting new diseases
  - Anomaly detection



Two problems in Supervised Learning





Classification	Regression
Split it	Fit it
Discrete or Categorical data.	Real number data
Has category associated	Has associated number
Example : Tumour classification	Example : Prediction of stock market

Other types of learning approaches

- Generative approaches
  - Creating new data that is “like” given data
  - Generally included in unsupervised learning

- Semi-supervised learning
  - Small amount of labeled data available along with unlabeled data

- Self-supervised learning
  - Implicit labels are extracted from data using heuristics

- Reinforcement learning
  - Action strategy chosen based on temporally delayed rewards.
  - Useful in strategic decisions . Example : Chess, Games, Investment, etc

The distinction between the various types of learning is often blurred

The Learning Paradigm

A fundamental "trick" in DL

↳ Deep Learning

- All problems are data, all solutions are functions/maps

- Cognitive tasks -- Humans get sensory inputs as qualia
  - We must convert these qualitative inputs into **numbers** – Input Vectors
  - Similarly, outputs that humans give must also be converted into numbers – Output/Target vectors

Input → Output

- Determining appropriate inputs and outputs for a machine learning task is an essential part of the process

- Often the "Learning Task" is learning the mapping from input to output.

Clean-Input "Features" → ML

Unclean → DL

Physical Mathematical Modeling

Inverse Problems

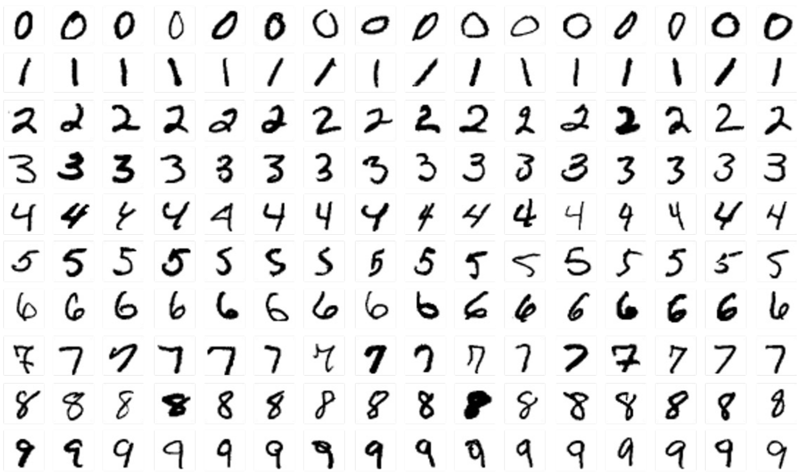
DL

Input → Output

Hello Physics vs Data

(Domain Knowledge)

Example – From image to vector



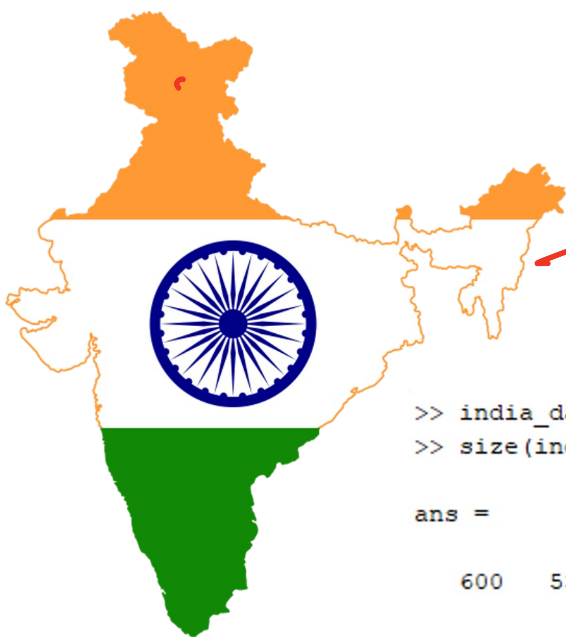
1 0

3x3

Pixel Intensity 0.5 0 0.5

1 0 1
0 0 0
1 1 1

Mathematical



```
>> india_data=imread('india.png');
>> size(india_data)

ans =

   600   538     3
```

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	240
255	255	255	255	255	255	255	255	255	255	231	80
255	255	255	255	255	255	255	255	255	242	78	51
255	255	255	255	255	255	255	255	255	131	51	51
255	255	255	255	255	255	255	255	225	54	51	51
82	140	209	254	255	255	255	255	108	51	51	51
51	51	51	89	180	227	209	120	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51
51	51	51	51	51	51	51	51	51	51	51	51

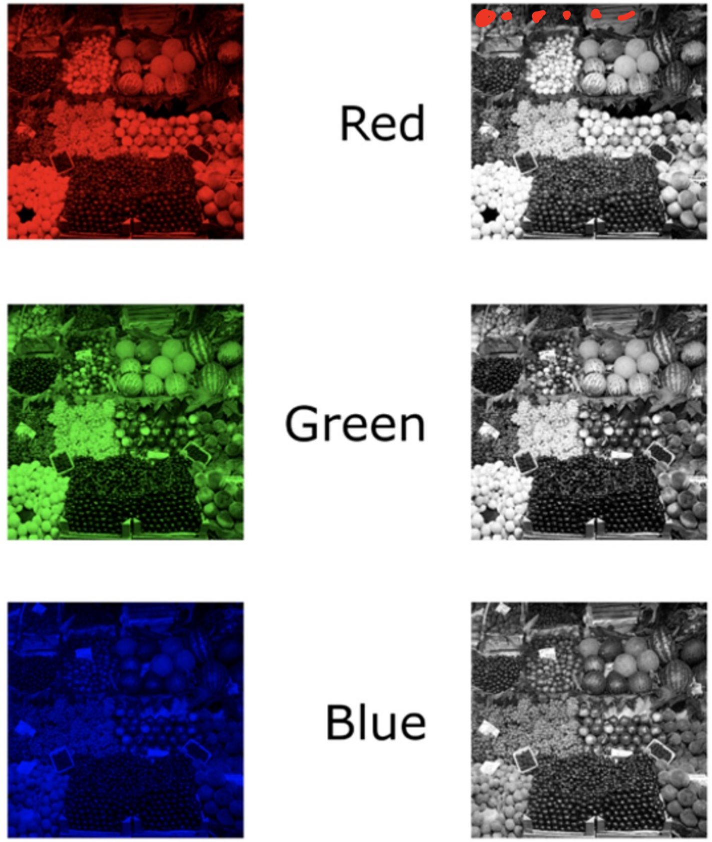
9

https://upload.wikimedia.org/wikipedia/commons/2/27/MnistExamples.png

https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/India_geo_stub.svg/538px-India_geo_stub.svg.png

Image parametrization

- RGB images



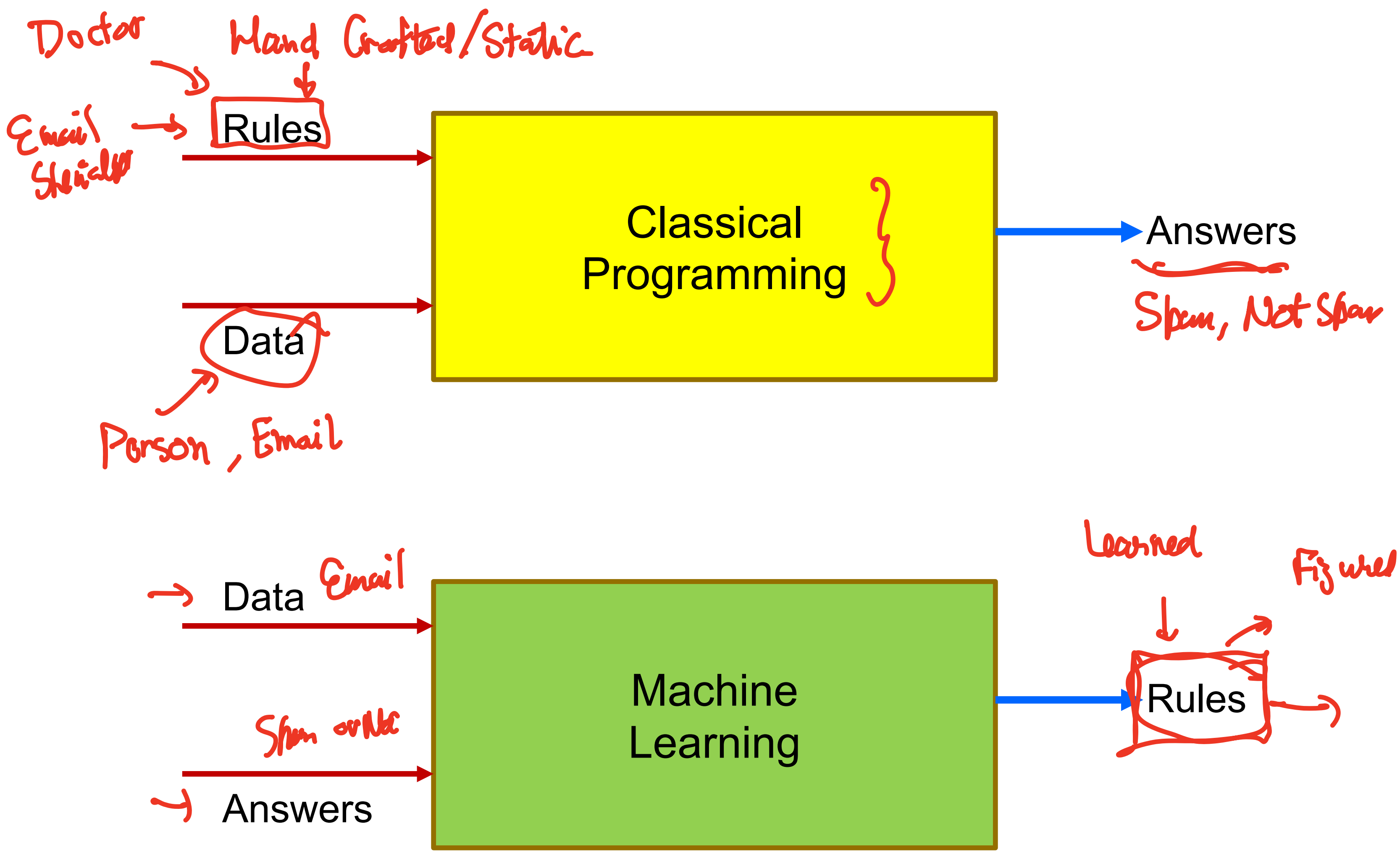
0.
255

10^6

10^6

10^6

Recall -- The Machine Learning Paradigm

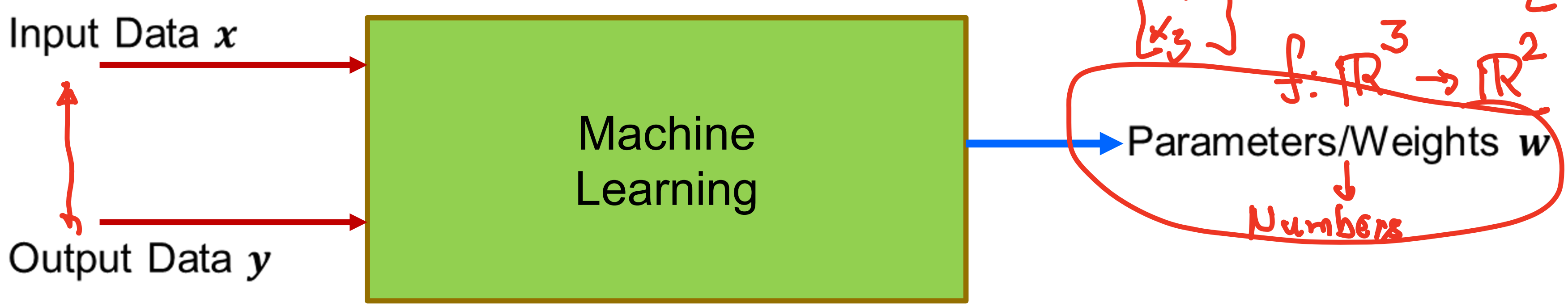


Let us now look at the Machine Learning Paradigm in some more detail

General Paradigm

Training Process

$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ $Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$



- We wish to learn the relationship between the input and the output data.

  $f(x) \to y$

- For now, we will think of this relationship as a function (Mathematical)
  - We can call this function the model or the hypothesis function

  Neural Network is a mathematical function

  Architecture (NN)

- The function has two parts
  - Form of the function → Fixed
  - Parameters of the function ↳ Adjustable

(x_1, x_2)

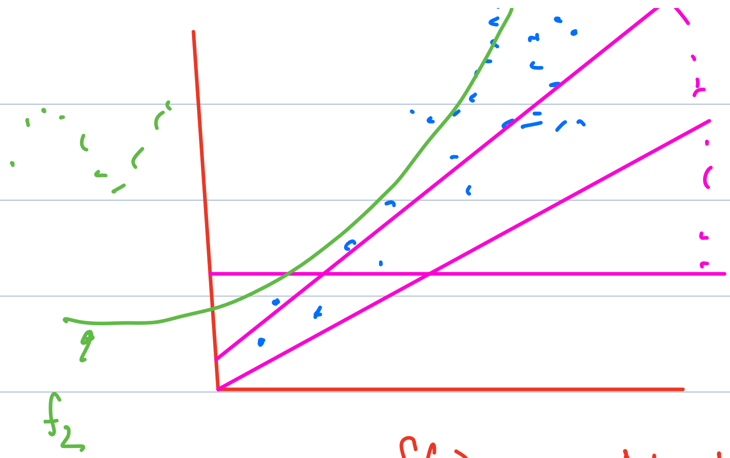
$$e.g\ y = h(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2$$

$$i.e\ w_0, w_1, w_2, w_3$$

- Typical machine learning learns only the parameters
  - The form is provided by the ML engineer. Requires domain knowledge

Learning involves two processes – Feedforward and feedback

Linear



$f_1 = \boxed{a} + \boxed{b}x$

$f_2 = c + dx + ex^2$

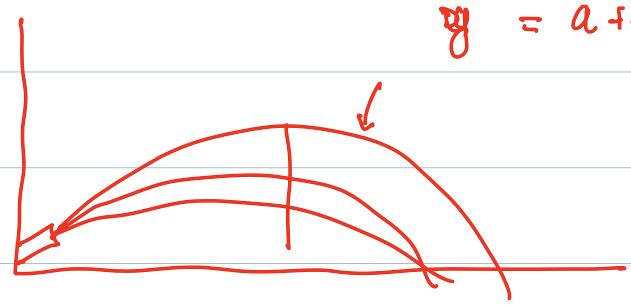
Quadratic.

$f(x) = w_0 + w_1 x$

$\tilde{h}(x) = 5 + 3x_1$ Is this part of $h(x)$ family?

$\tilde{h}(x) = 5 + 3x_1^2$

$y = a + bx + cx^2$

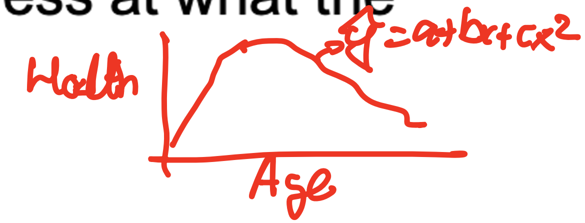


Forward Modeling

$h(x; w) = \underline{w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2}$

Length Depth Area

- A model or hypothesis is simply an educated guess at what the relationship between input and output is.



- As mentioned before, it has two pieces
  - Form of the function – Linear, Quadratic, Exponential, etc
  - Parameters of the function

Fixed variable / coeffi

$f(x, w)$

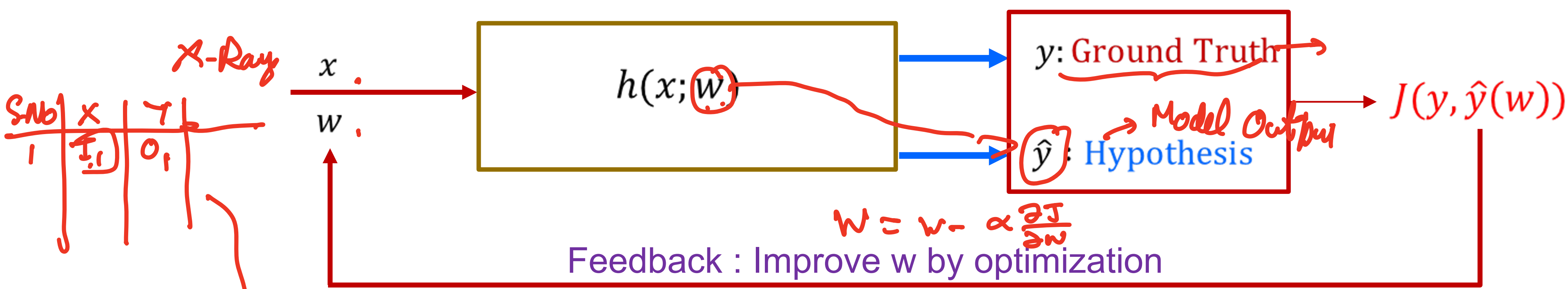
- We sometimes use the notation $y = f(x; w)$
  - Given x and a choice of w, we can find a corresponding y

Given input data & parameters <u>Compute</u> output

- The function *f* going from x to y is called the *forward model*
  - The process is called the forward pass or inference-> Given x, w finding y

Learning the parameters via feedback

Training



To learn the parameters (given x,y find w), we follow this paradigm

- Collect lots of data pairs (Input Vector, Output Vector) $= \ (\boldsymbol{x}, \boldsymbol{y})$
- Guess for the form of the hypothesis function $h(x; w)$
  - Example : $h(x; w) = w_0 + w_1 x_1 + w_2 x_2$
- For an arbitrary guess for w
  - We will get some $\hat{\boldsymbol{y}} = h(\boldsymbol{x}; \boldsymbol{w})$ which will not match the ground truth $\boldsymbol{y}$
- Define a cost function $J(y, \hat{y}(w))$ depending on the difference
- Find optimal w by minimizing $J(\mathrm{w})$ → Feedback process
  - By using some optimization procedure such as Gradient Descent

$(y-\hat{y})^4$ $\quad J = \frac{1}{2}(y-\hat{y})^2$ $\quad |y-\hat{y}|$

At the end of optimization → Trained Model. } Learning / Training Process

Summary

Two main ideas in this session

1. What enables breadth of application?

   By converting even qualitative problems into data problems and treating every rule which we wish to learn as a mathematical function.

   $$\text{Input} \xrightarrow{\text{Function}} \text{Output}$$

2. How does (supervised) learning happen?
   1. We collect data as (Input, Output) pairs.
   2. We provide a hypothesis as a form of the function connecting the inputs to the outputs
   3. Learn (improve) the parameters based on mathematical feedback from data via loss function.

Machine Learning Models

Main ideas

Two main ideas in this section

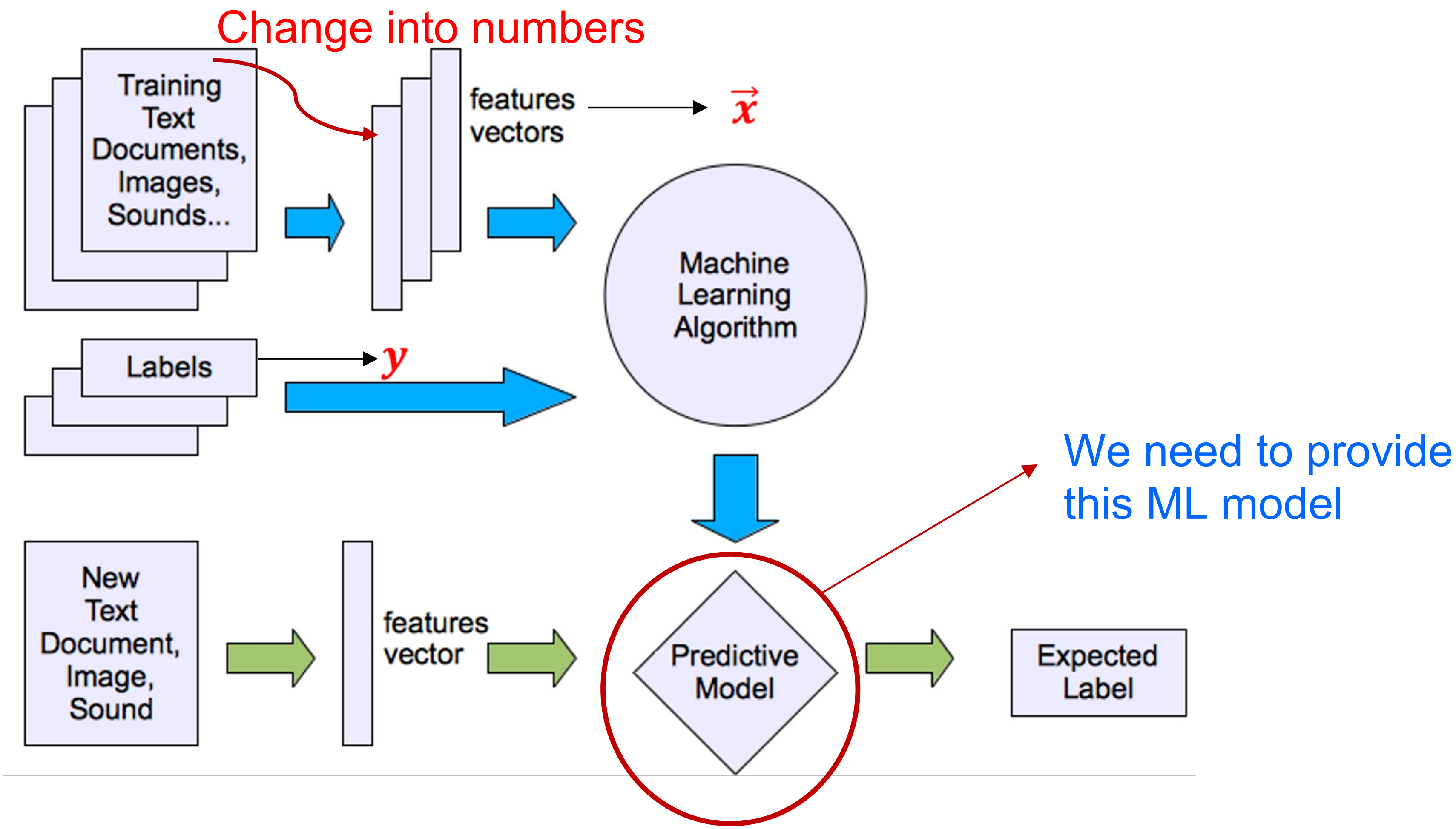
1. Demonstration of the learning paradigm.

We will see how to apply the learning paradigm on a very simple model – the linear model. This will give us the chance to see some details of the training process.

2. Neural Networks

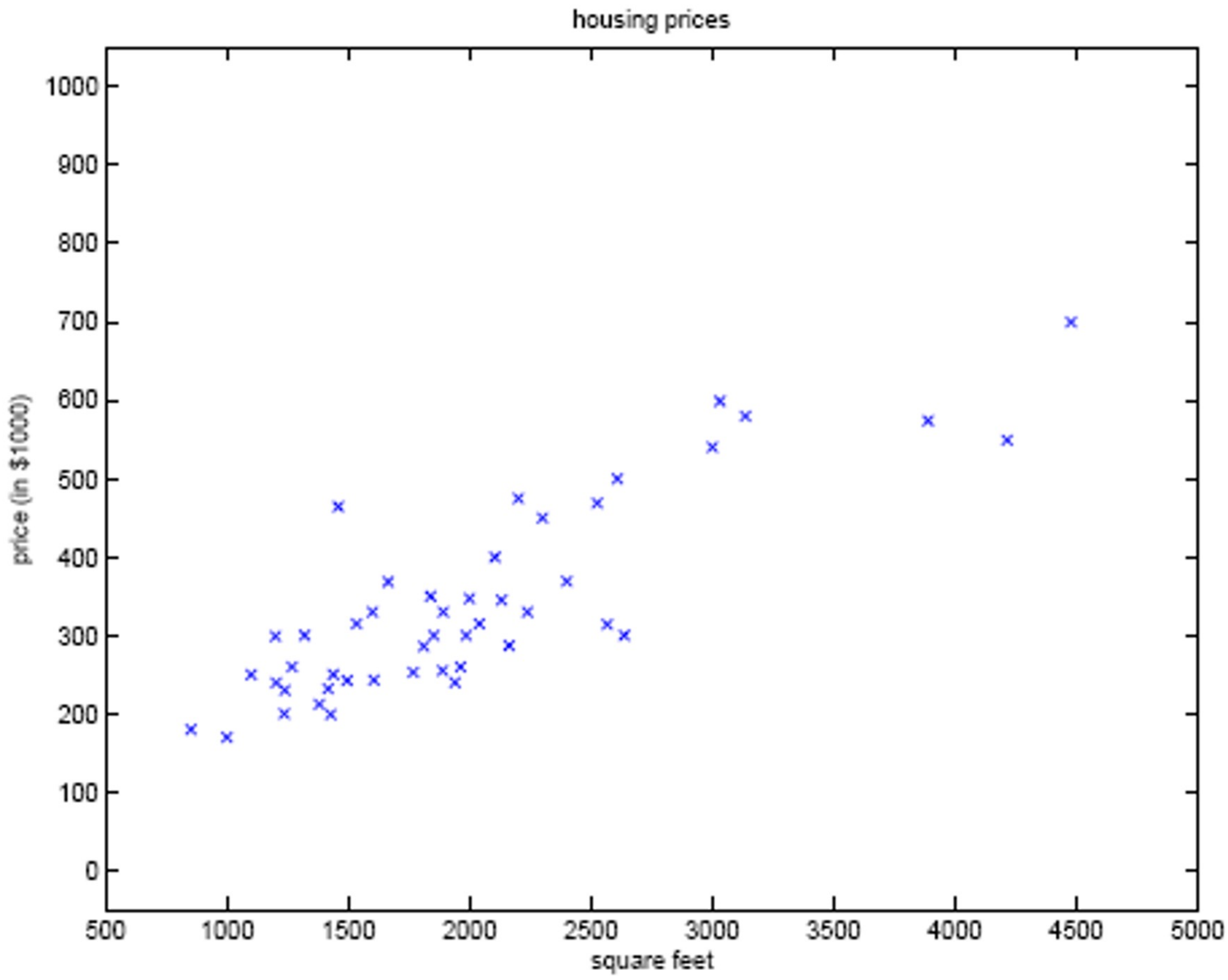
We will see how, with a small twist on linear models, you can obtain a very sophisticated model – the neural network. We will also discuss why Neural Networks are so popular.

General structure of supervised learning



Philosophical underpinning of ML/NN: All problems are data, all solutions are functions

Regression Example – House price prediction

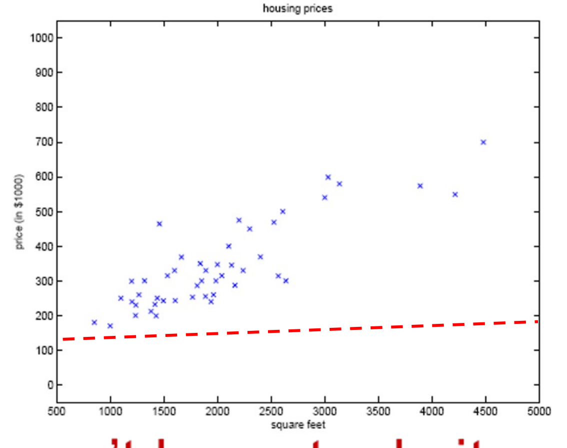


Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

Regression question : Given the living area, can you predict the price?

Learning a model or map for house price

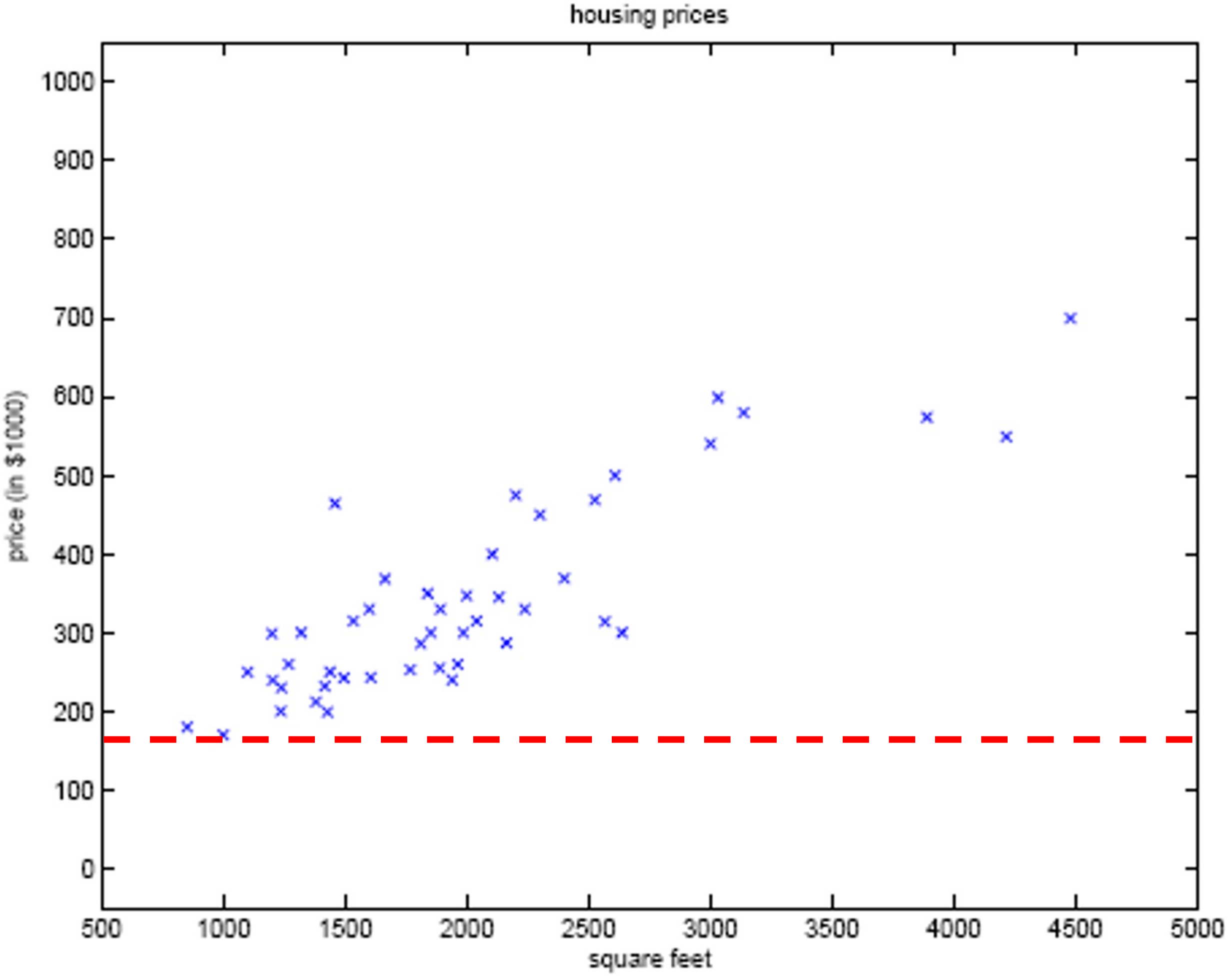
- Machine learning may be viewed as learning an optimal map between
  - Input vectors **x** → Area, Images, Email, Geometry
  - Output vectors **y** → Price, Class, Spam label, Heat Transfer
  - by training over many example sets → Given data points



- The map/function consists of two parts
  - Form of the function –
    - E.g. Linear: $\hat{y} = h^{lin}(x) = w_0 + w_1 x$

  We provide this by ourselves. The machine doesn't know to do it.
  - Parameters of the map
    - E.g Find w_0, w_1 through "training" by minimizing least squares

Regression Example – Housing price prediction

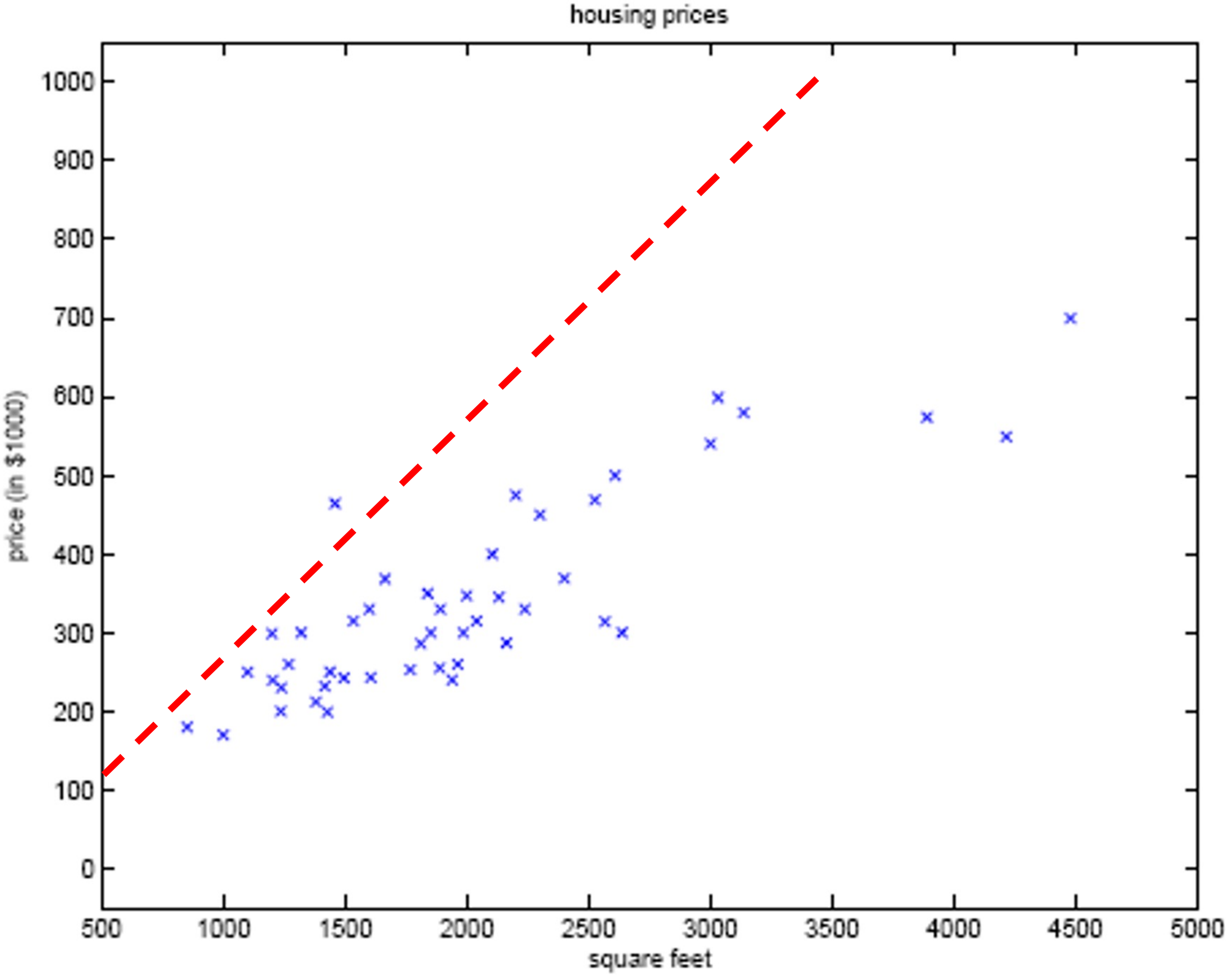


Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
$\vdots$	$\vdots$

$w_0 = 150, w_1 = 0$

Let us guess for parameters and improve

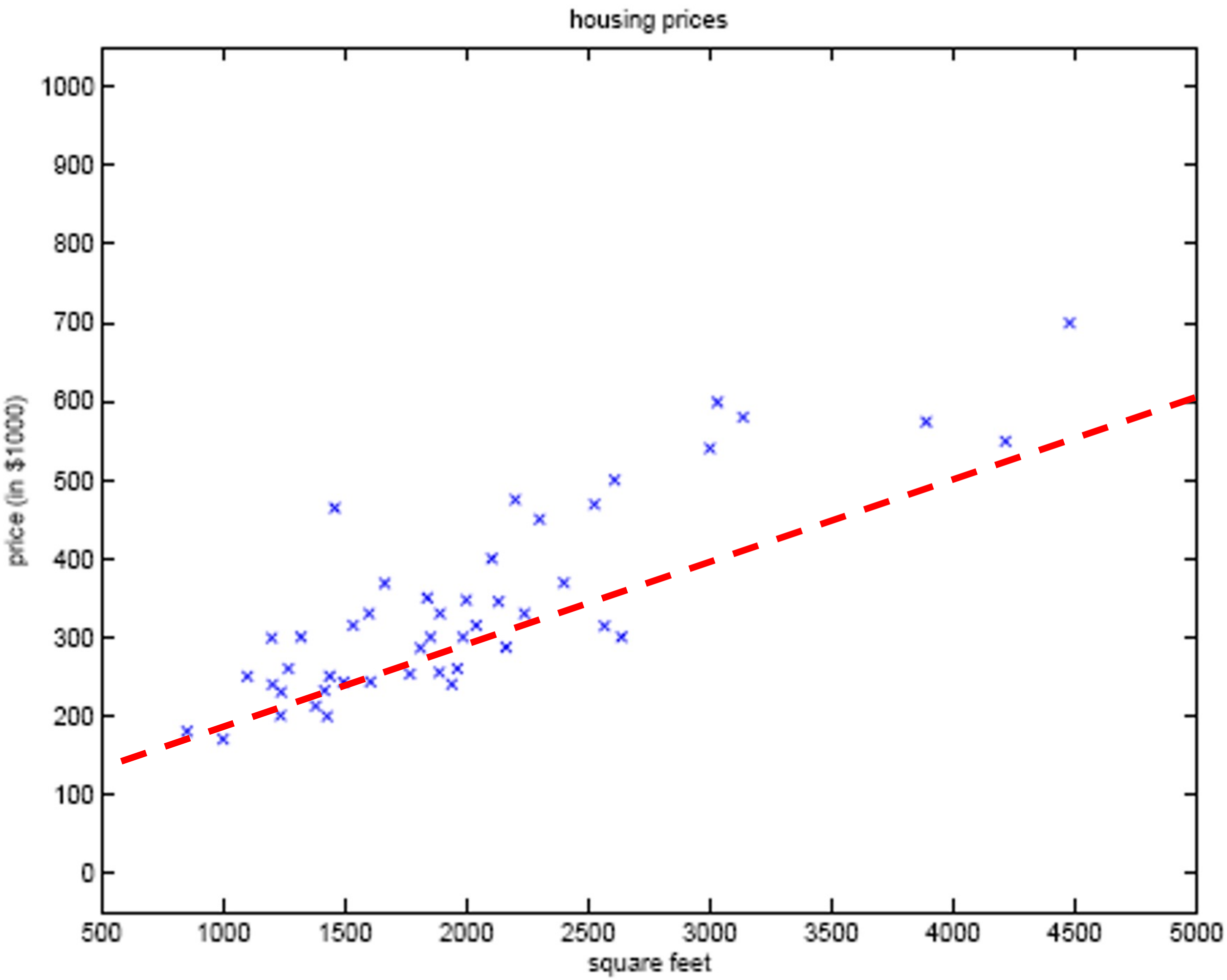
Regression Example – Housing price prediction



Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

Let us guess for parameters and improve

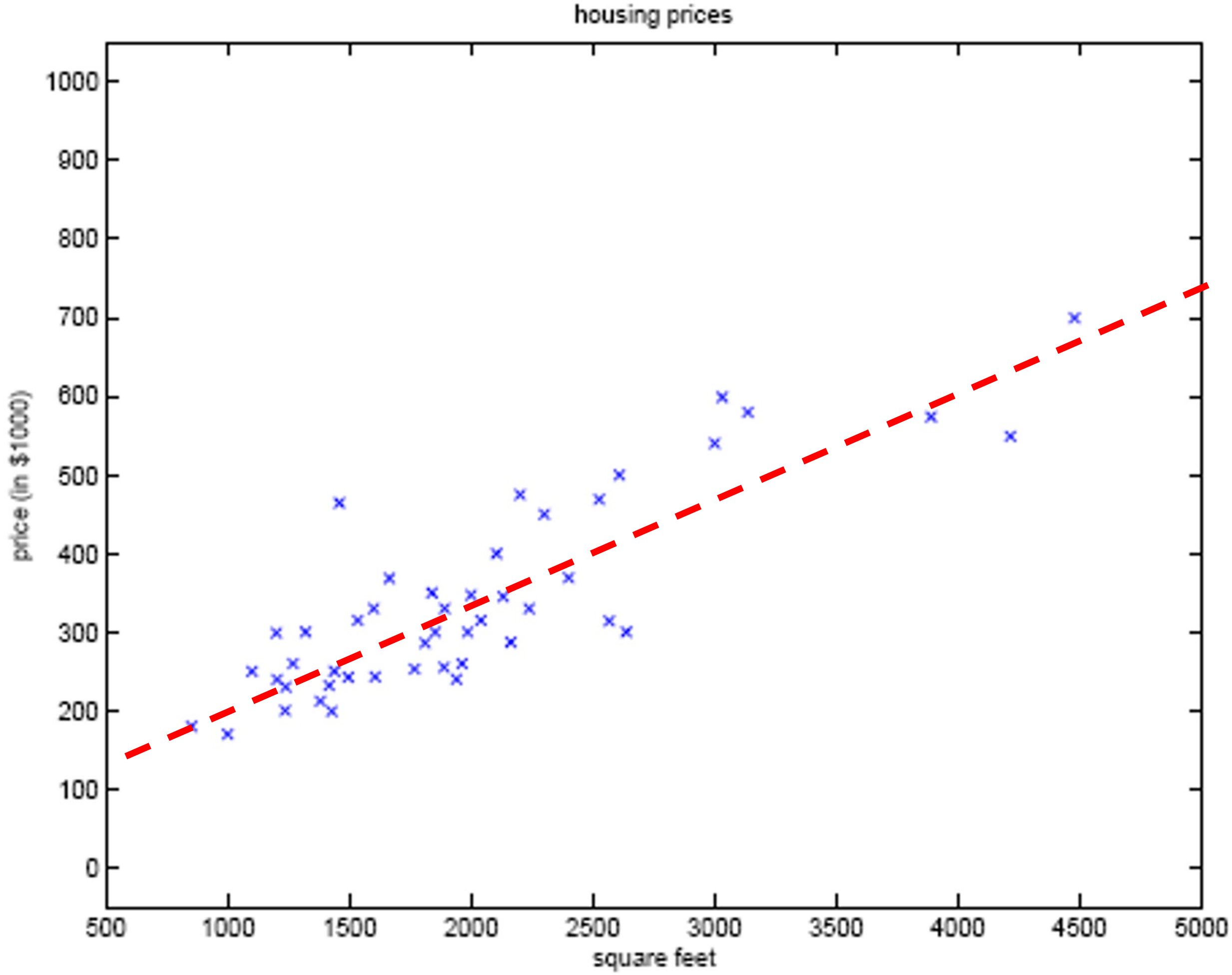
Regression Example – Housing price prediction



Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

Let us guess for parameters and improve

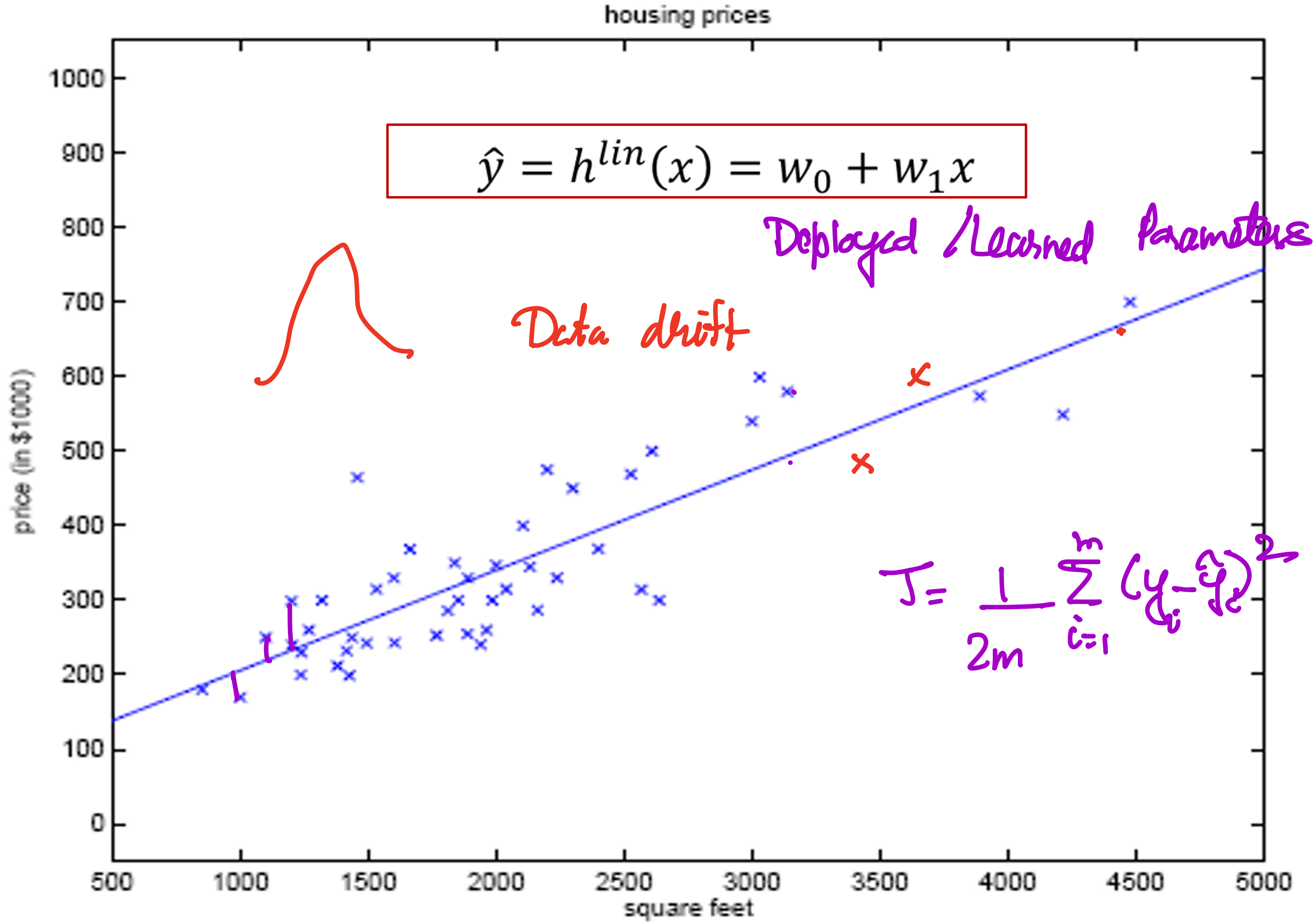
Regression Example – Housing price prediction



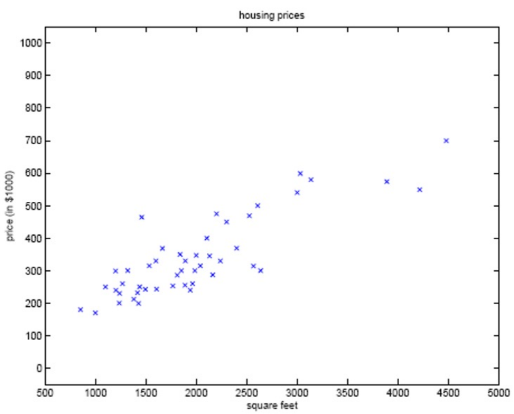
Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
$\vdots$	$\vdots$

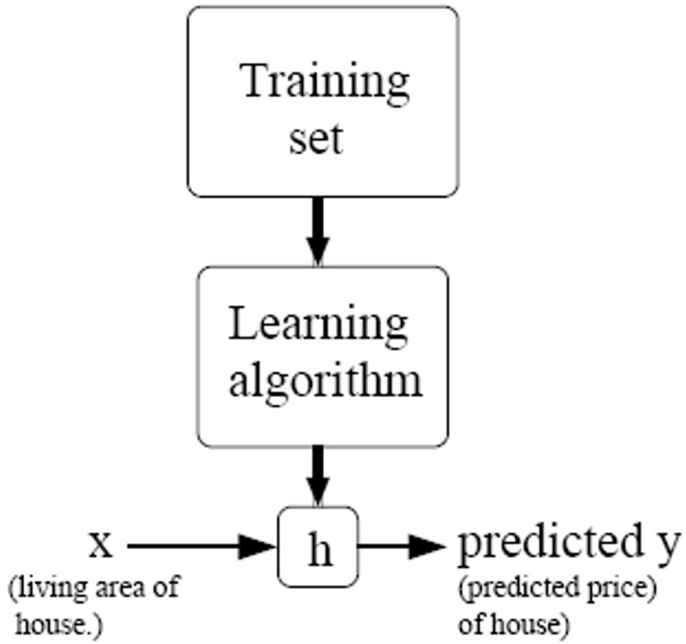
Let us guess for parameters and improve

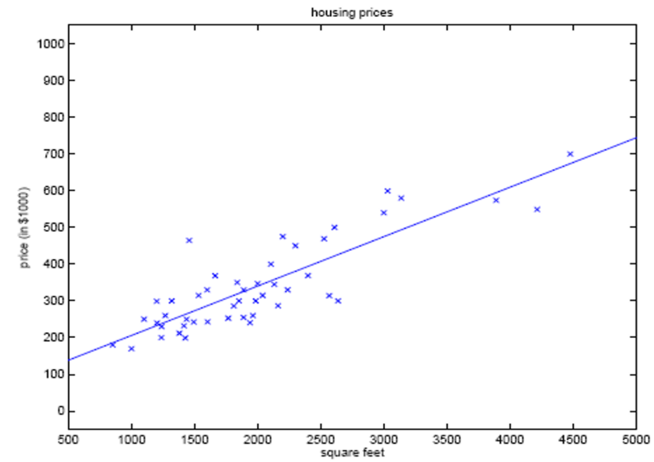
Example : Linear Regression



Workflow







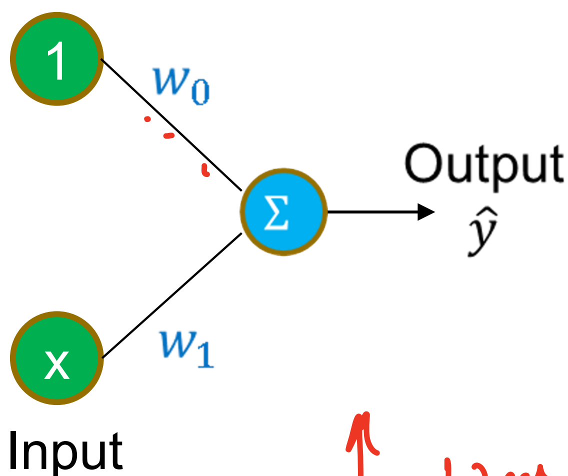
- Determine relevant input, output vectors and features
- Determine form of model (say, linear)
- Decide on cost function – e.g $J(y, \hat{y}) = (y - \hat{y})^2$
- Find the parameters that minimize the cost function
  - Can either solve this problem directly or iteratively
  - Simplest iterative algorithm – Gradient Descent

Is there ONE general model which can approximate any function?

Neural Networks → Family of functions

Pictorial representation

Linear Model



Linear combination

Algebraic notation

$$\hat{y} = w_0 + w_1 x = \Sigma w_i x_i$$

Matrix $\rightarrow W^T x \rightarrow x = \begin{bmatrix} 1 \\ x \end{bmatrix}$

$W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$

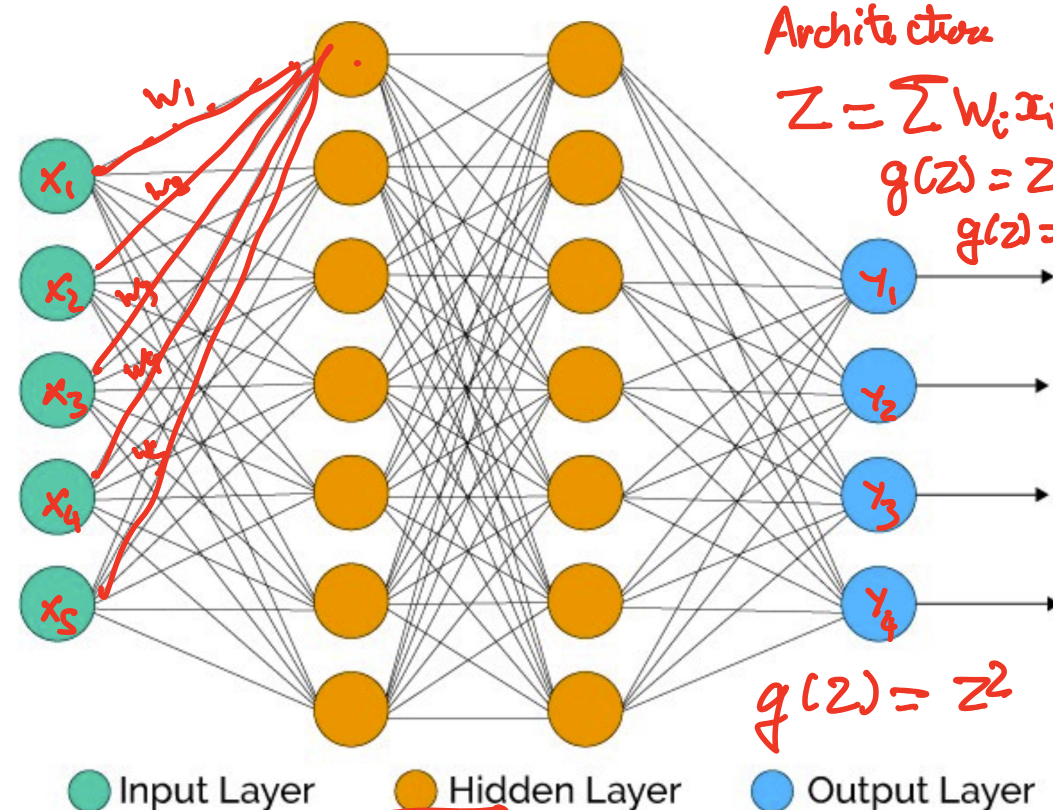
Neural Network Model

Architecture

$Z = \Sigma w_i x_i$

$g(z) = z^2$

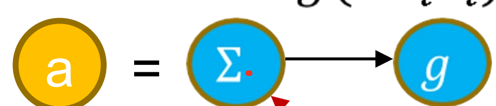
$g(z) = z^3$

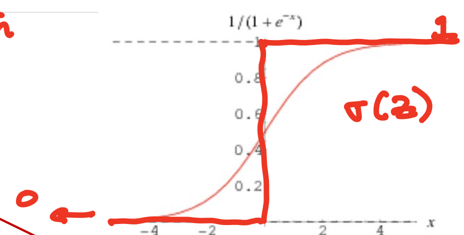


$g(z) = z^2$

$$a = g(\Sigma w_i x_i)$$

Activation Function

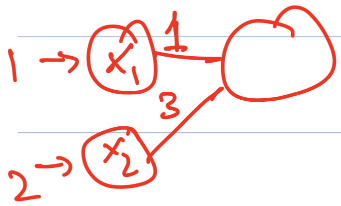




A Neuron has a linear and a nonlinear operation

Example nonlinear function is Sigmoid(x) = $\frac{1}{1+\exp(-x)}$

$\sigma(x)$



$$z = \overset{1}{x_1} + 3\overset{2}{x_2} = 7$$

$$g(z) = z$$

$$g(z) = 3z$$

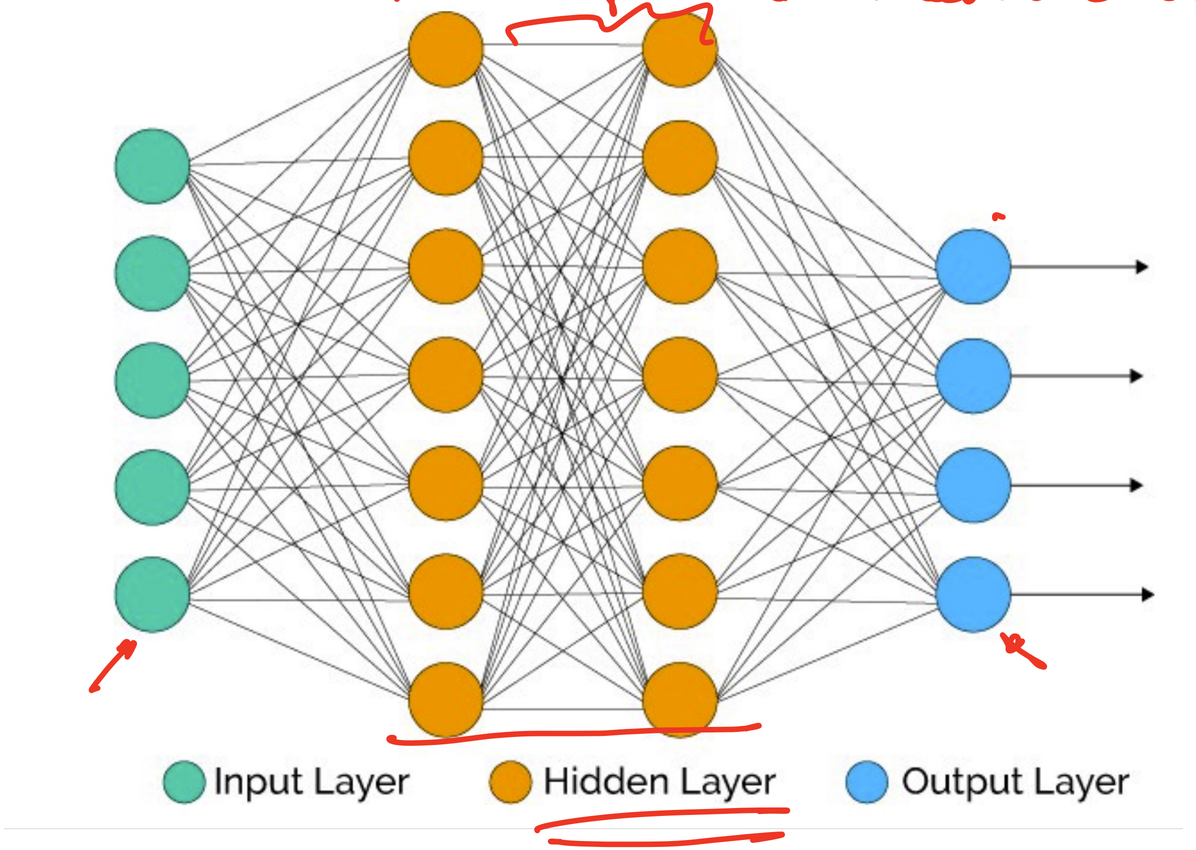
$$g(z) = z^2 \rightarrow 7^2 = 49$$

$\rightarrow$ Non linear model

$$g(z) = z \rightarrow \text{Linear Model}$$

Universal Approximation Theorem

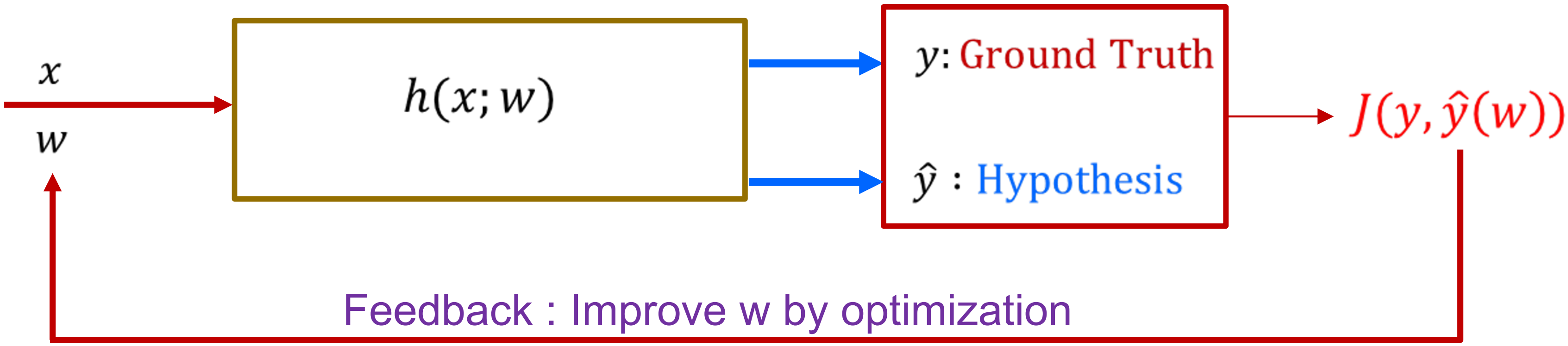
Arbitrary Intermediate calculate



Theorem -- Given sufficient data and neurons, a Neural Network can approximate any function to any desired accuracy

Heart of modern Deep Learning – Lots of data, lots of layers => Lots of learning

Recall : Learning the parameters via feedback



To learn the parameters, we follow this paradigm

- Guess for the form of the hypothesis function $h(x; w)$

- For an arbitrary guess for w
  - We will get some $\hat{\boldsymbol{y}} = h(\boldsymbol{x}; \boldsymbol{w})$ which will not match the ground truth $\boldsymbol{y}$
  - Define a cost function $J(y, \hat{y}(w))$ depending on the difference

- Find optimal w by minimizing $J(\mathrm{w})$ ⟶ Feedback process
  - Requires **Backpropagation** for $\frac{\partial J}{\partial w}$

Even complicated Neural Networks work in exactly the same way!