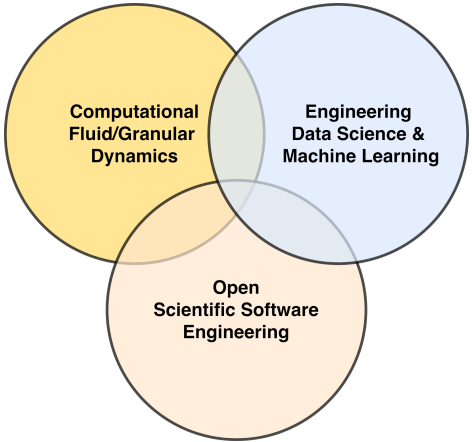
Hands-On Deep Learning (DL)

d.r.tunuguntla@saxion.nl (Deepak)

8^{th} January, 2026

Who am I?

- **2011-2018** PhD, PostDoc in Modelling Granular Materials
- **2015-2023** $\approx$ 7.5 yrs in Scientific Software, Data and AI industry
- **2023-Now** Associate Professor Applied Data and AI at Saxion UAS, Enschede



Acknowledgements

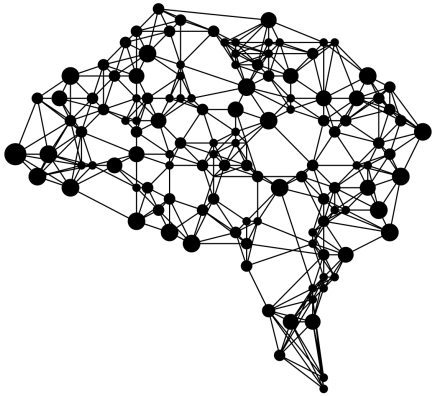
- Special thanks to CCC-ParaSolS
- Developers of DataOps module in Masters Software Engineering at Saxion UAS
- Open Course Wares (OCWs) CS129, CS229 and CS230 from Stanford University
- AI by Hand workbooks by Prof. Tom Yeh
- OCW Learn PyTorch
- Google Colab
- OCW AI for Engineering and Science (AISE) from ETH Zurich
- Dutch KIEM grant: GranML
- Ambient Intelligence Group at Saxion UAS

Learning materials

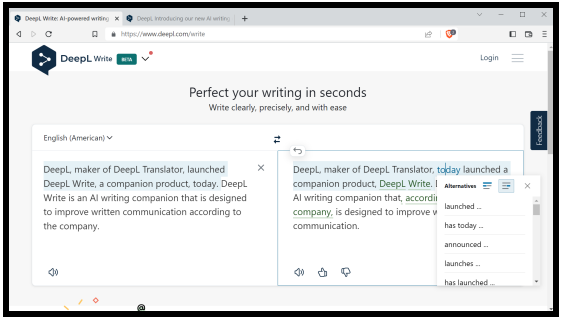
- Open Course Wares (OCWs) CS129, CS229 and CS230 from Stanford University
- Understanding Deep Learning by Simon Prince
- Dive into Deep Learning by Zhang, Lipton, Li and Zmola
- AI by Hand workbooks by Prof. Tom Yeh
- OCW Learn PyTorch
- OCW AI for Engineering and Science (AISE) from ETH Zurich
- ...

Learning objectives

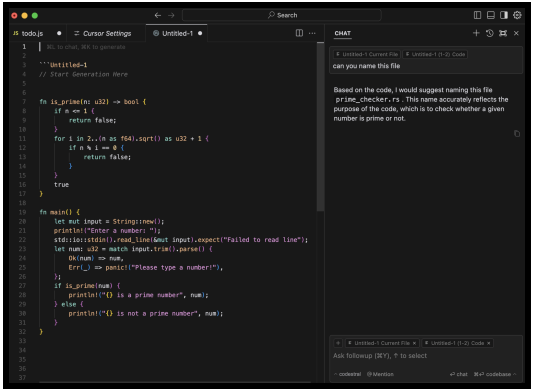
- Understanding the concept of Deep Learning (DL)
- Familiarising with the DL jargon
- Getting to know world renowned DL frameworks



Rise of AI/DL



Source: bureauworks



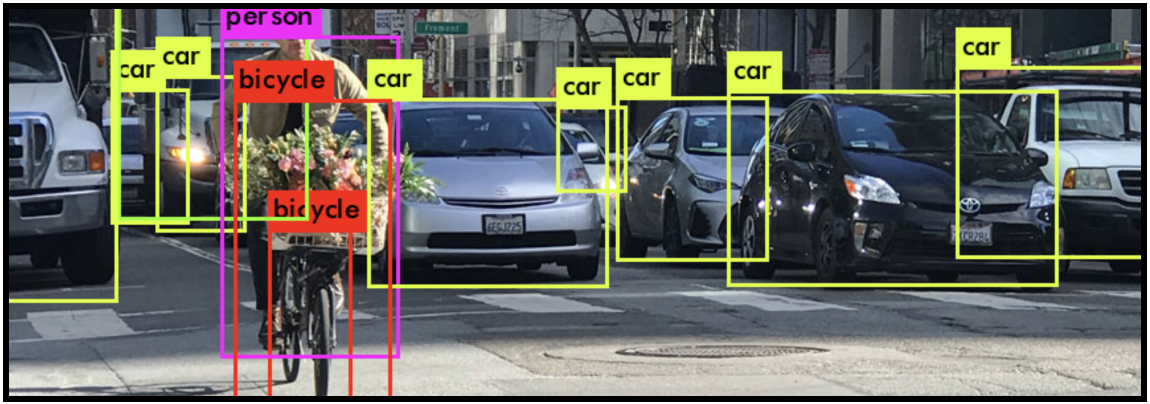
Source: Alexander Young

Rise of AI/DL

Prompt: Photorealistic closeup video of two pirate ships battling each other as they sail inside a cup of coffee

Source: SORA OpenAI

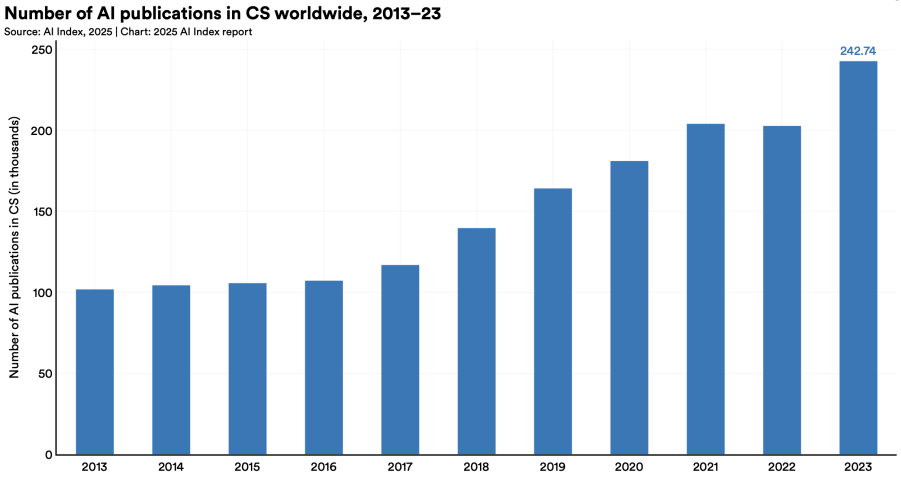
Rise of AI/DL



Source: dagshub

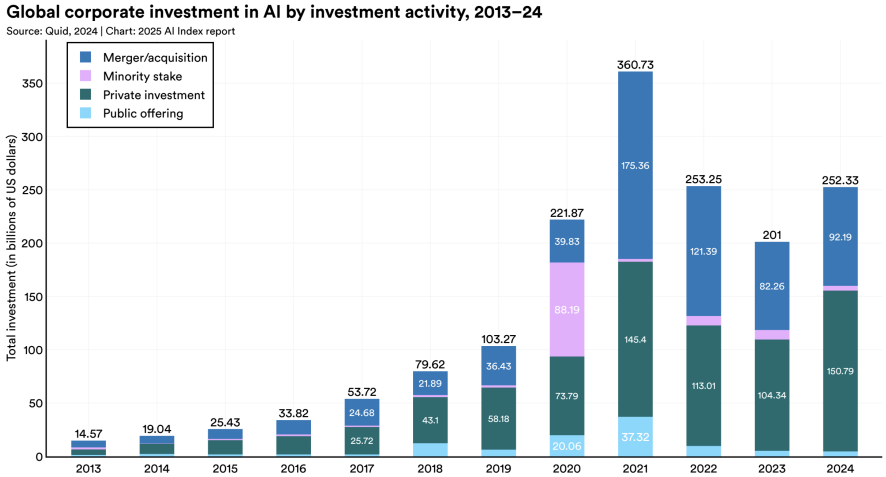
Rise of AI/DL

Stanford AI Report, 2025



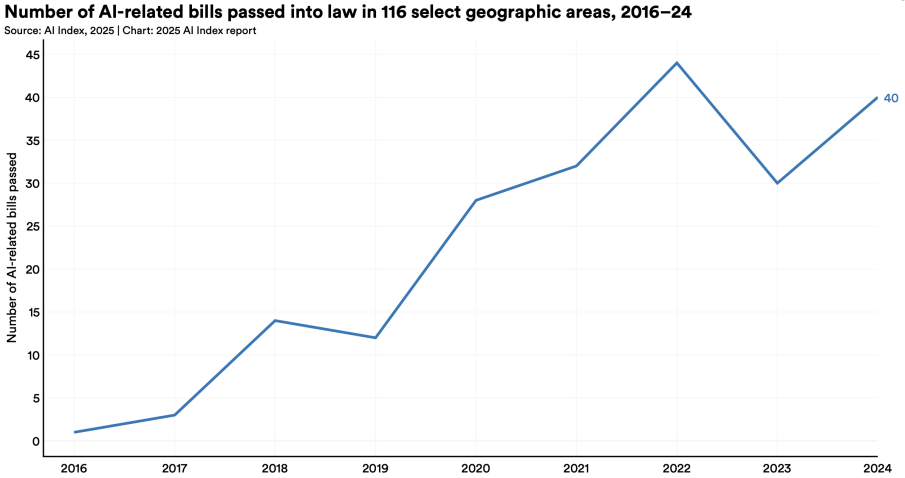
Rise of AI/DL

Stanford AI Report, 2025



Rise of AI/DL

Stanford AI Report, 2025



Rise of AI/DL in Science

Rise of AI/DL in Science

- Climate, Space, Biology, Chemistry, Particle Mechanics, ...

Rise of AI/DL in Science

- Climate, Space, Biology, Chemistry, Particle Mechanics, ...
- Workshops at reputed conferences

Rise of AI/DL in Science

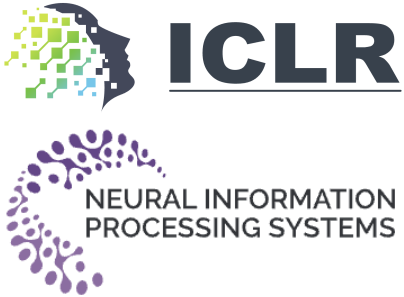
- Climate, Space, Biology, Chemistry, Particle Mechanics, ...
- Workshops at reputed conferences

Select ICLR workshops

- XAI4Science, AI4Mat
- AI4Differential Equations in Science
- Physics for Machine Learning, Neural Fields across Fields, ML4Materials

Rise of AI/DL in Science

- Climate, Space, Biology, Chemistry, Particle Mechanics, ...
- Workshops at reputed conferences



Select ICLR workshops

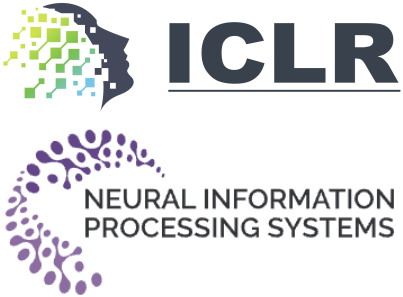
- XAI4Science, AI4Mat
- AI4Differential Equations in Science
- Physics for Machine Learning, Neural Fields across Fields, ML4Materials

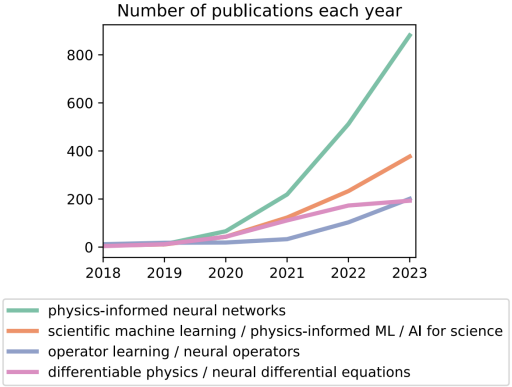
Select NeurIPS workshops

- Machine Learning and the Physical Sciences (ML4PS), AI for Science
- Foundation Models for Science, AI4Mat, Machine Learning and Physical Sciences

Rise of AI/DL in Science

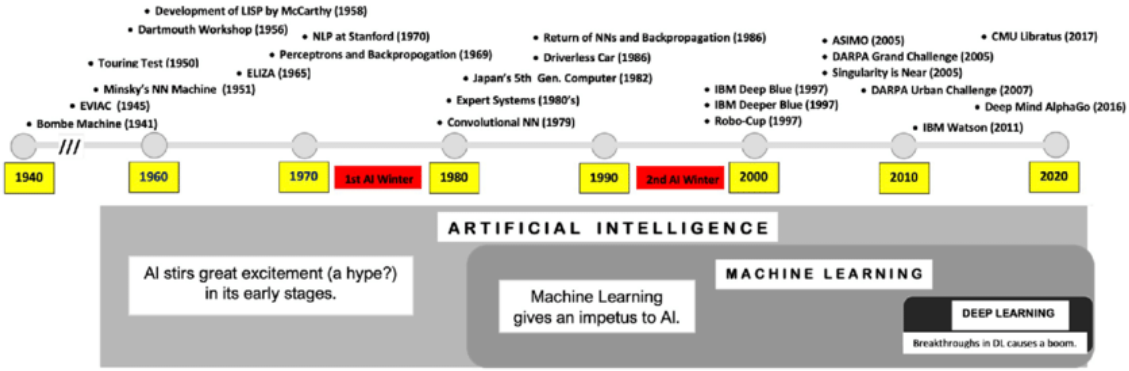
- Climate, Space, Biology, Chemistry, Particle Mechanics, ...
- Workshops at reputed conferences





Screenshot: AISE 2024 course

Why now? (AI Timeline)

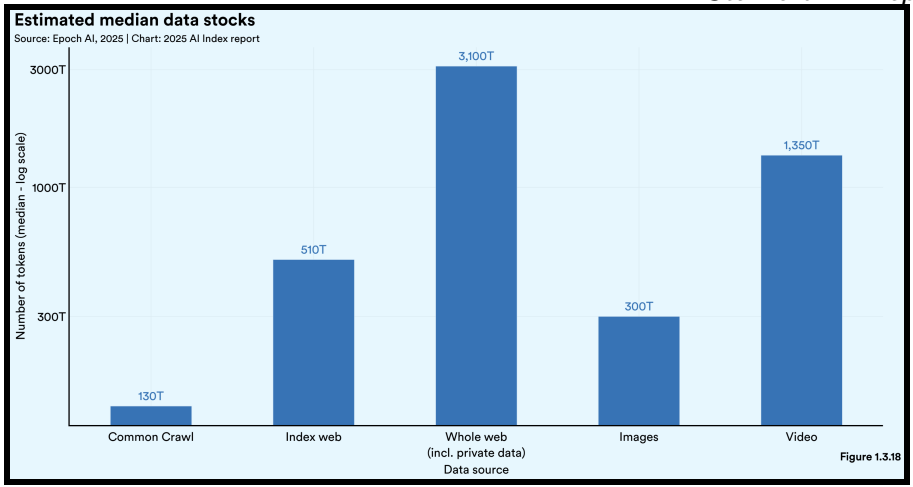


- Neural Networks vs Deep Learning

Kaynak, 2021

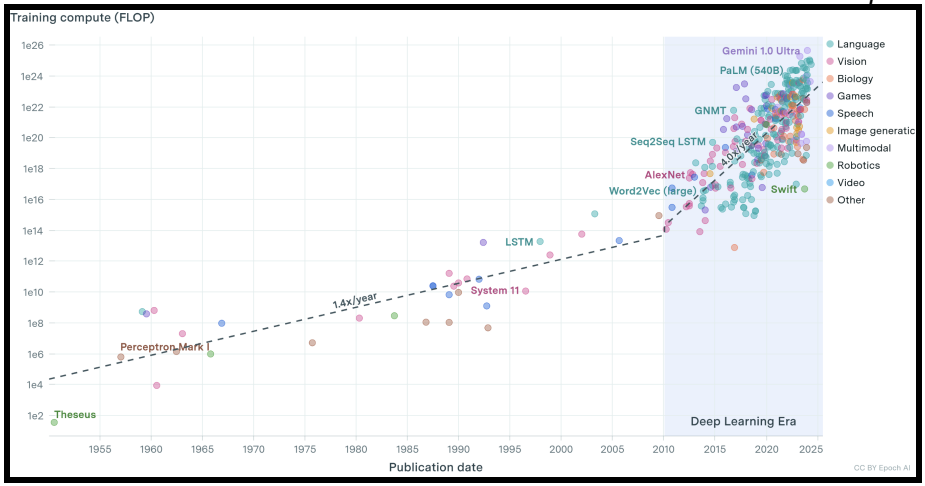
Why now? (Lots of Data)

Stanford AI Report, 2025

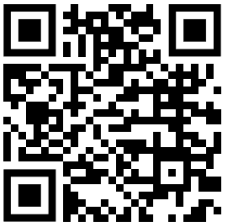


Why now? (Growing compute power)

EpochAI, 2022



Why now? (Forest of frameworks)

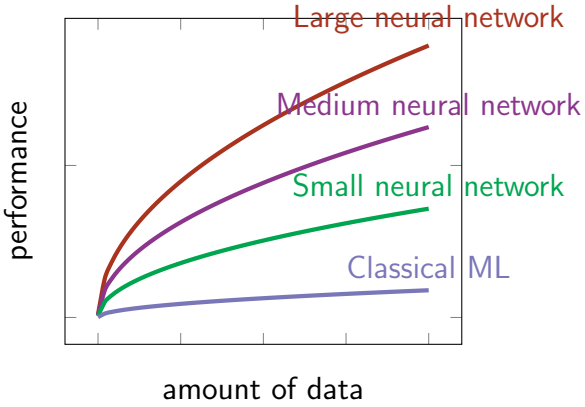


- The MAD 2025 landscape

- Linux Foundation AI and Data landscape

But, really, why now?

The real magic lies in the algorithmic advances.



DL building blocks

DL building blocks

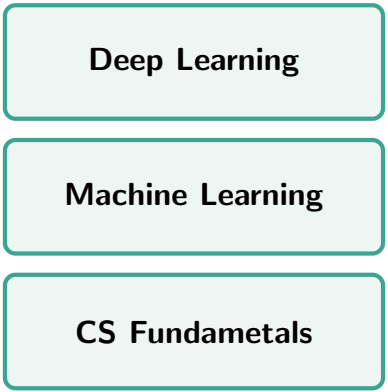
CS Fundametals

DL building blocks

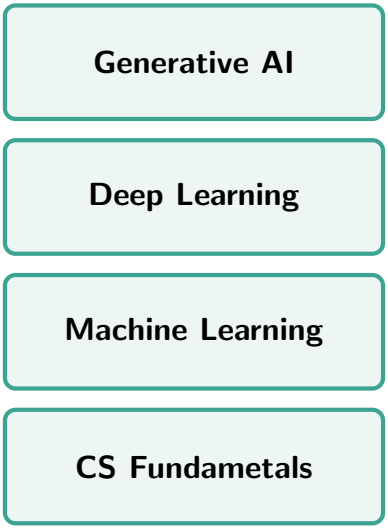
Machine Learning

CS Fundametals

DL building blocks



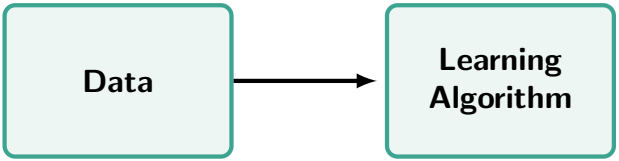
DL building blocks



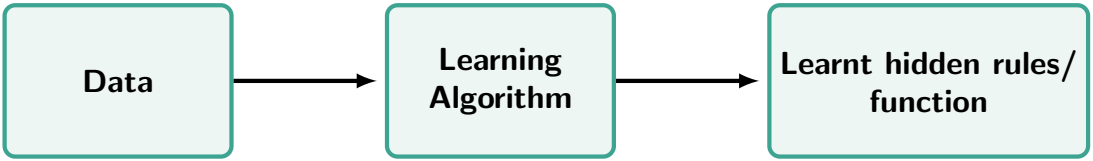
What is Machine Learning?

What is Machine Learning?

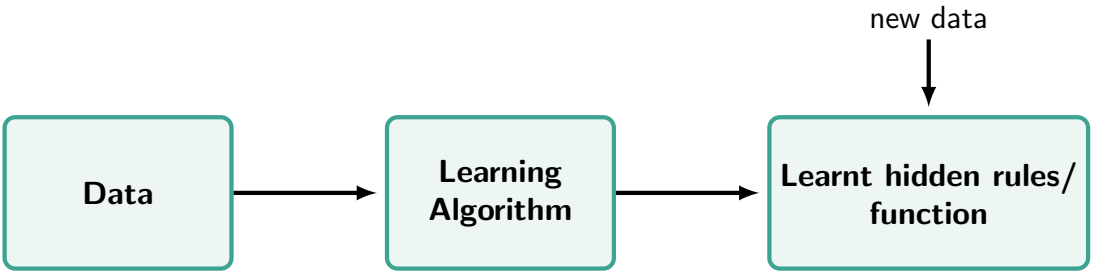
What is Machine Learning?



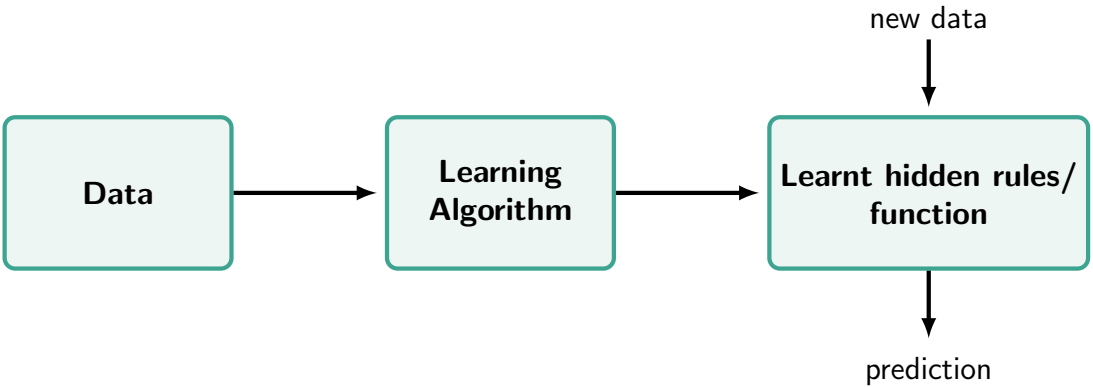
What is Machine Learning?



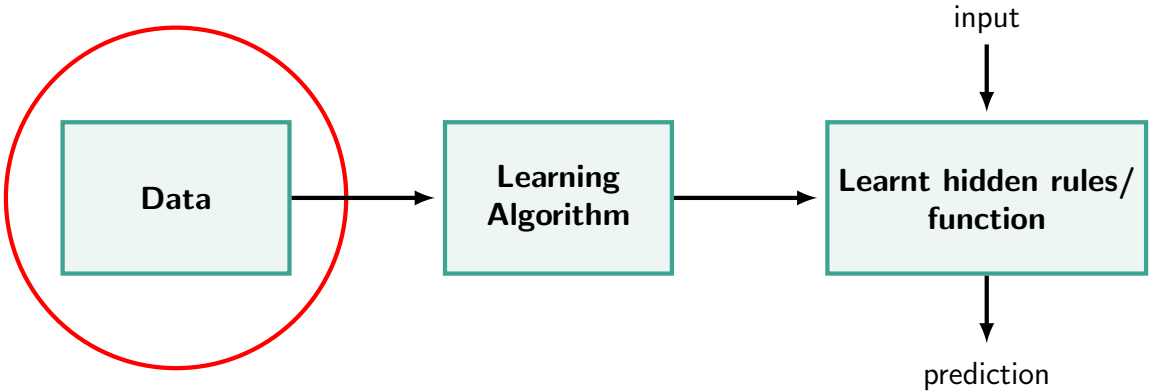
What is Machine Learning?



What is Machine Learning?



Data in ML


input
Data
Learning Algorithm
Learnt hidden rules/ function
prediction

Different data forms

Unstructured data

The university has 5600 students.
John's ID is number 1, he is 18 years old and already holds a B.Sc. degree.
David's ID is number 2, he is 31 years old and holds a Ph.D. degree. Robert's ID is number 3, he is 51 years old and also holds the same degree as David, a Ph.D. degree.

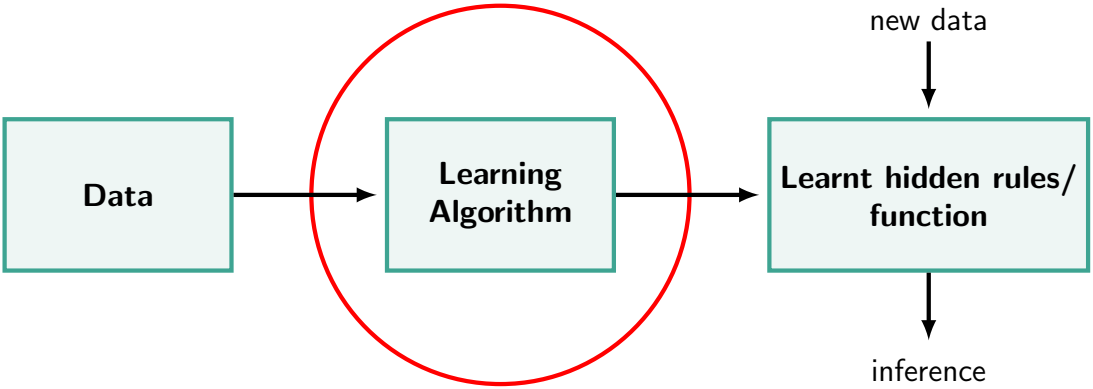
Semi-structured data

```
<University>
 <Student ID="1">
   <Name>John</Name>
   <Age>18</Age>
   <Degree>B.Sc.</Degree>
 </Student>
 <Student ID="2">
   <Name>David</Name>
   <Age>31</Age>
   <Degree>Ph.D. </Degree>
 </Student>
 ....
</University>
```

Structured data

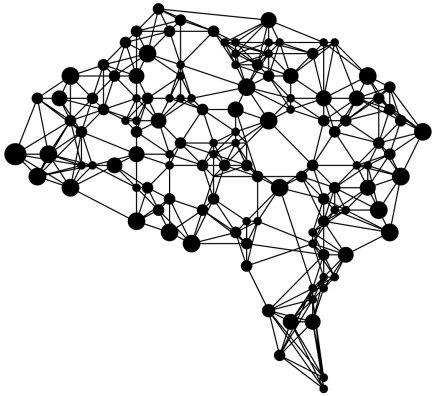
ID	Name	Age	Degree
1	John	18	B.Sc.
2	David	31	Ph.D.
3	Robert	51	Ph.D.
4	Rick	26	M.Sc.
5	Michael	19	B.Sc.

ML learning algorithms



ML learning algorithms

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Recommender systems
- Reinforcement learning
- Self-supervised learning
- Contrastive learning
- Transfer learning



Supervised learning algorithm

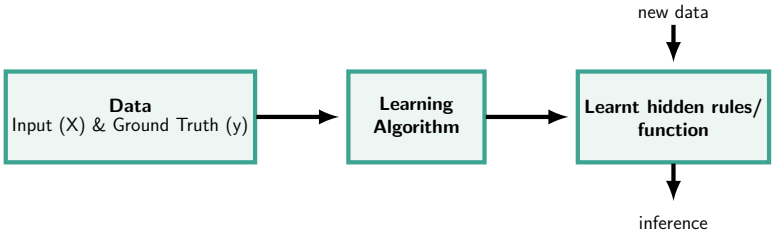
Learns from being provided with **true answers/ground truth**

Supervised learning

Input (X)	**Ground Truth (y)**	**Application**
Email	Spam (0/1)	Spam filter
Audio (speech)	Transcripts	Speech2Text
DEM contact parameters	Bulk characteristics	Paramter calibration
...	...	...

Supervised learning

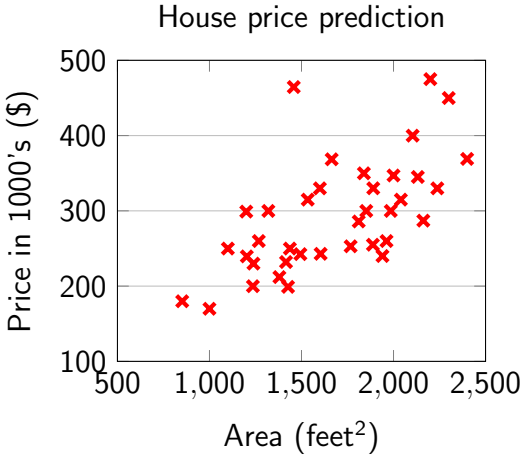
Input (X)	**Ground Truth (y)**	**Application**
Email	Spam (0/1)	Spam filter
Audio (speech)	Transcripts	Speech2Text
DEM contact parameters	Bulk characteristics	Paramter calibration
...	...	...



Single variable linear regression

- Data: [X,y] pairs

Area (X)	**Price (y)**
2104	399.900
1600	329.900
2400	369.000
...	...

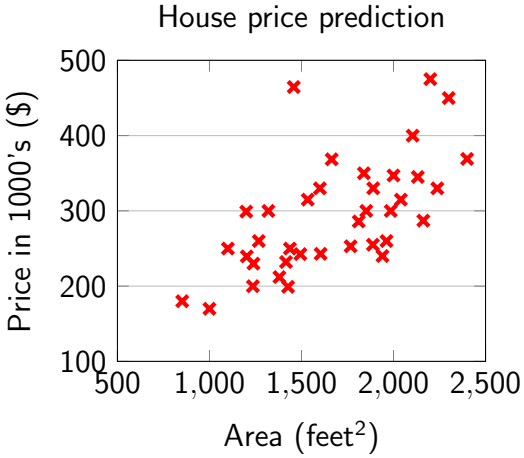


Single variable linear regression

- Data: [X,y] pairs

Area (X)	**Price (y)**
2104	399.900
1600	329.900
2400	369.000
...	...

$X :=$ feature, $y :=$ target values/labels



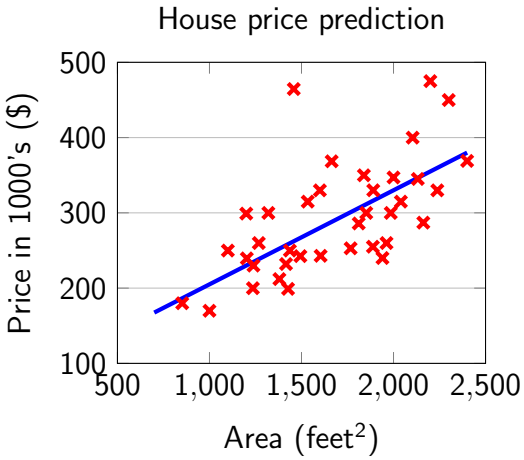
Single variable linear regression

- Data: [X,y] pairs

Area (X)	**Price (y)**
2104	399.900
1600	329.900
2400	369.000
...	...

$X :=$ feature, $y :=$ target values/labels

- Learning $:= \hat{y} = f(X)$, $\hat{y} \neq y$



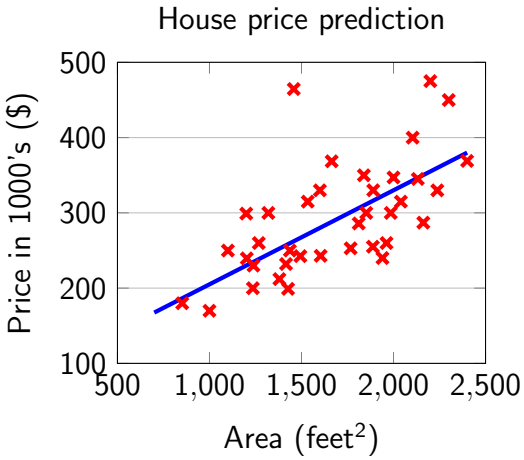
Single variable linear regression

- Data: [X,y] pairs

Area (X)	**Price (y)**
2104	399.900
1600	329.900
2400	369.000
...	...

$X :=$ feature, $y :=$ target values/labels

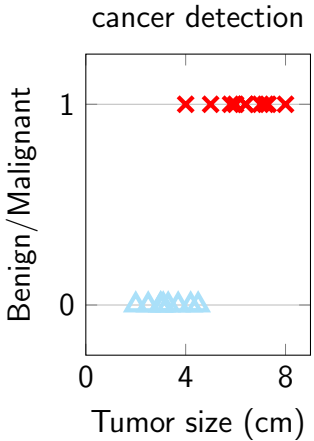
- Learning $:= \hat{y} = f(X)$, $\hat{y} \neq y$
- Fit is continuous function, i.e., can have infinite number of possible outcomes



Single variable classification

- Data: [X,y] pairs

Tumor size (X)	**Ben./Mal. (y)**
2	0
2.5	0
...	...
4	1
...	...

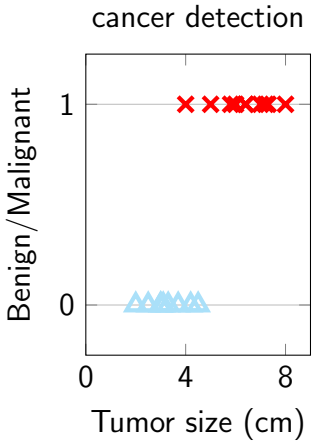


Single variable classification

- Data: [X,y] pairs

Tumor size (X)	**Ben./Mal. (y)**
2	0
2.5	0
...	...
4	1
...	...

X := feature, y := target values/labels



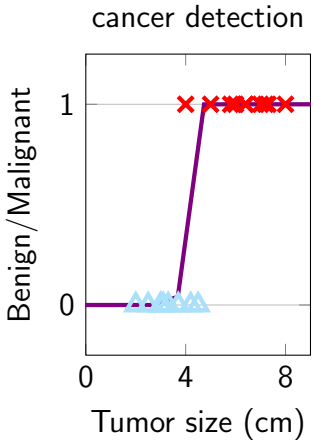
Single variable classification

- Data: [X,y] pairs

Tumor size (X)	**Ben./Mal. (y)**
2	0
2.5	0
...	...
4	1
...	...

$X :=$ feature, $y :=$ target values/labels

- Learning $:= \hat{y} = f(X)$, $\hat{y} \neq y$



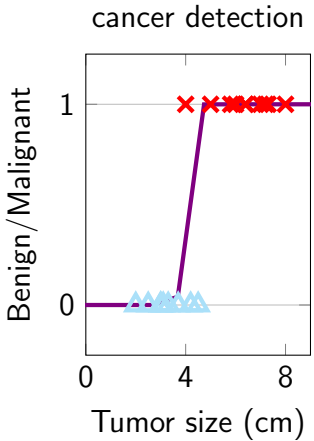
Single variable classification

- Data: [X,y] pairs

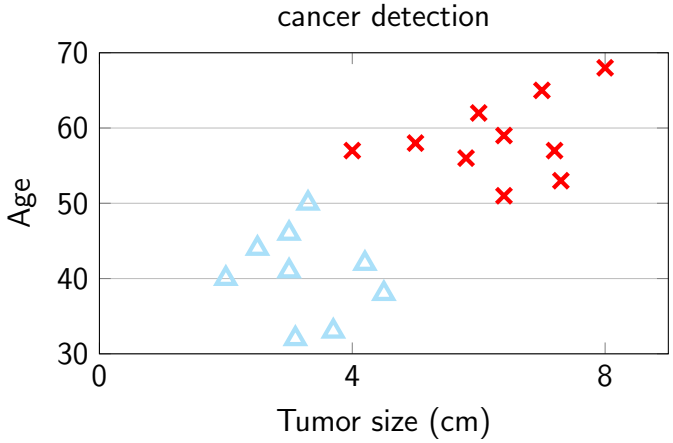
Tumor size (X)	**Ben./Mal. (y)**
2	0
2.5	0
...	...
4	1
...	...

$X :=$ feature, $y :=$ target values/labels

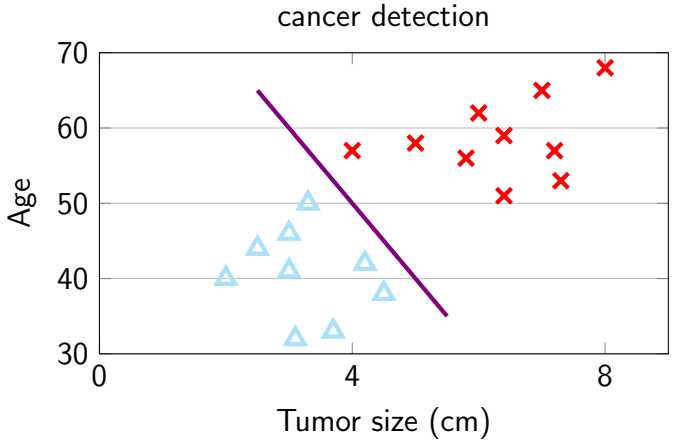
- Learning $:= \hat{y} = f(X)$, $\hat{y} \neq y$
- Fit is a discrete function, i.e., has limited possible outcomes; $y \in 0{,}1$



Multiple variable classification



Multiple variable classification

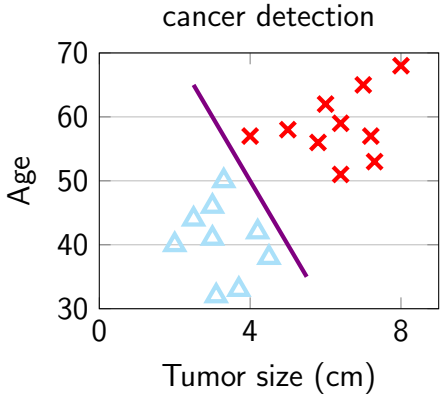


Unsupervised learning algorithm

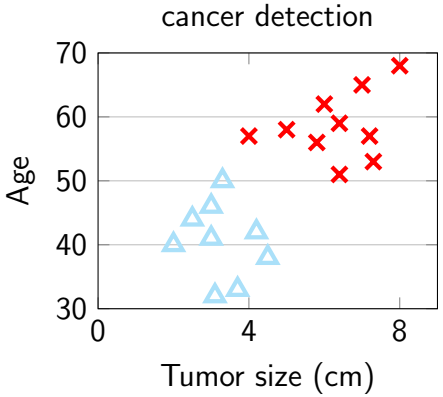
Learns from **unlabelled data alone**, i.e., no target values/labels (y)

Supervised vs Unsupervised

Data: [X,y] pairs

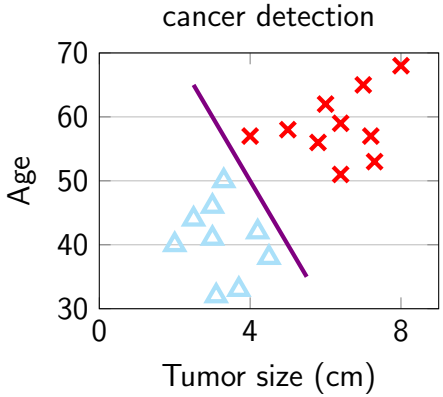


Data: [X]

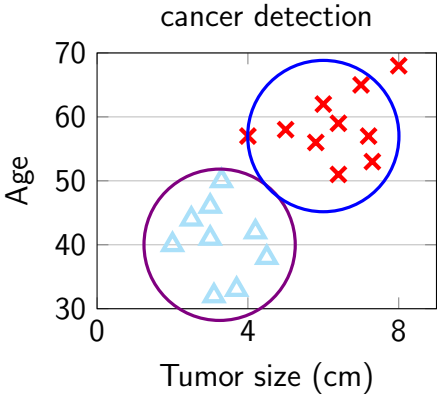


Supervised vs Unsupervised

Data: [X,y] pairs

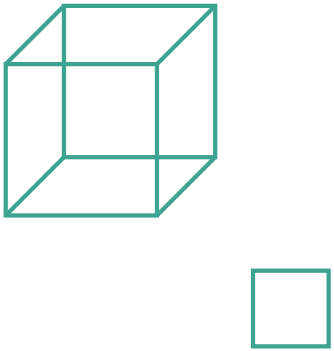


Data: [X]

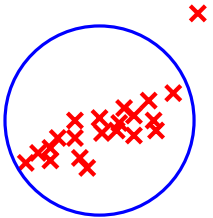


Unsupervised learning algorithms

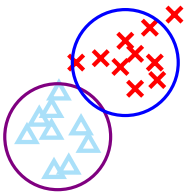
Dimensionality Reduction



Anomaly Detection

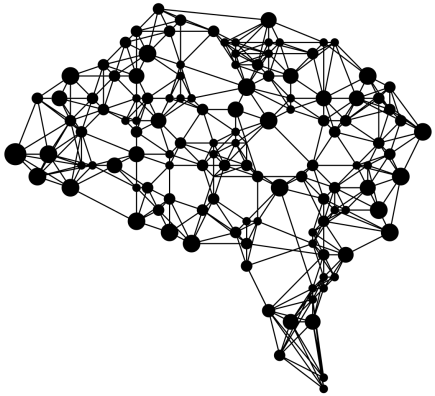


Clustering



So far

- Rise of Deep Learning
- Building blocks of DL
- What is ML?, data forms and classes of learning algorithms
- Terms: Feature, target/label

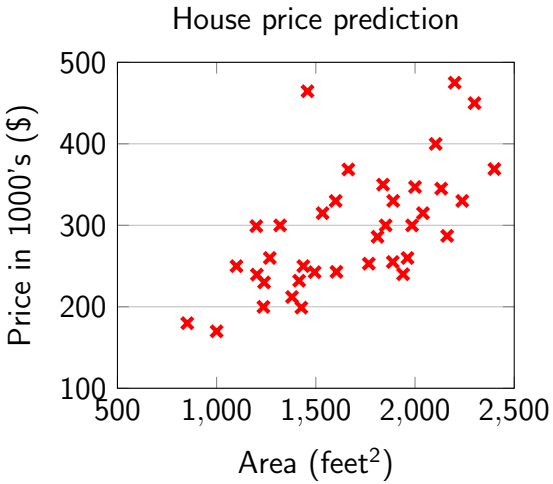


Single variable linear regression

- Data: [X,y] pairs
- Learning

$$\hat{y} = f(X), \hat{y} \neq y$$
$$f(X) = w * \text{Area} + b$$

- $w :=$ weight, $b :=$ bias

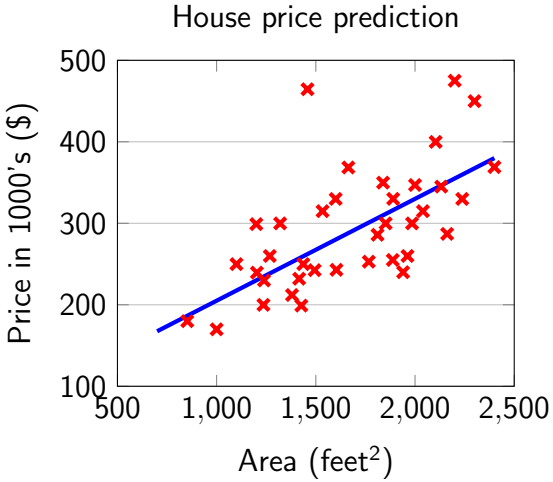


Single variable linear regression

- Data: [X,y] pairs
- Learning

$$\hat{y} = f(X), \hat{y} \neq y$$
$$f(X) = w * \text{Area} + b$$

- $w :=$ weight, $b :=$ bias

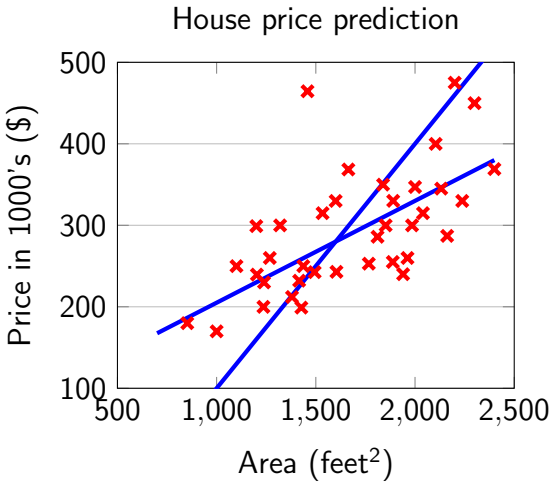


Single variable linear regression

- Data: [X,y] pairs
- Learning

$$\hat{y} = f(X), \hat{y} \neq y$$
$$f(X) = w * \text{Area} + b$$

- $w :=$ weight, $b :=$ bias

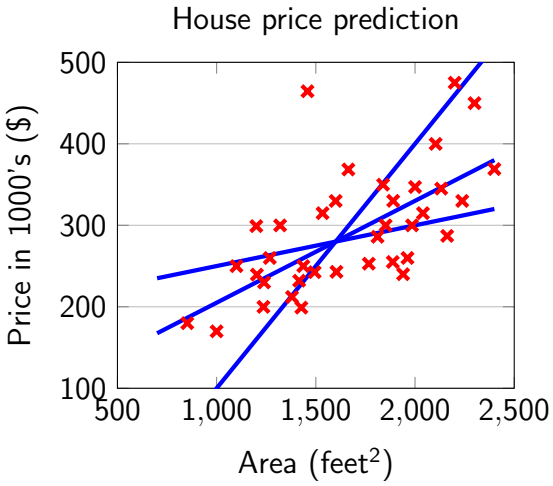


Single variable linear regression

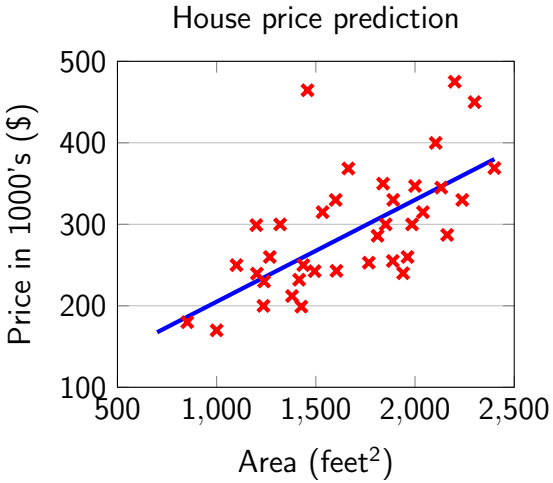
- Data: [X,y] pairs
- Learning

$$\hat{y} = f(X), \hat{y} \neq y$$
$$f(X) = w * \text{Area} + b$$

- $w :=$ weight, $b :=$ bias
- How do you find the best w and b to get a good model?



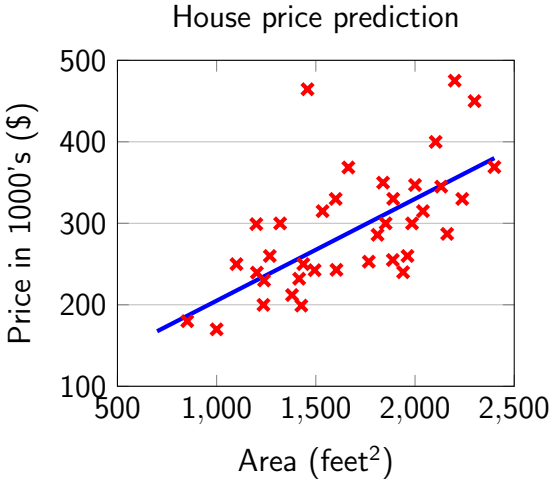
Problem statement



Problem statement

Given

Data := [X,y] pairs



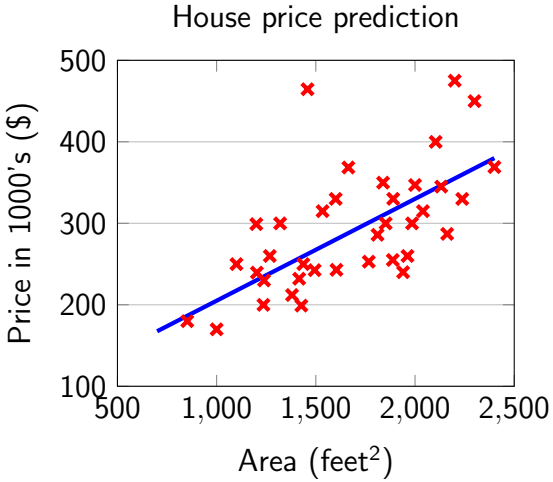
Problem statement

Given

Data := [X,y] pairs

Model type

$$\hat{y} = f_{w,b}(X) = w * X + b$$



Problem statement

Given

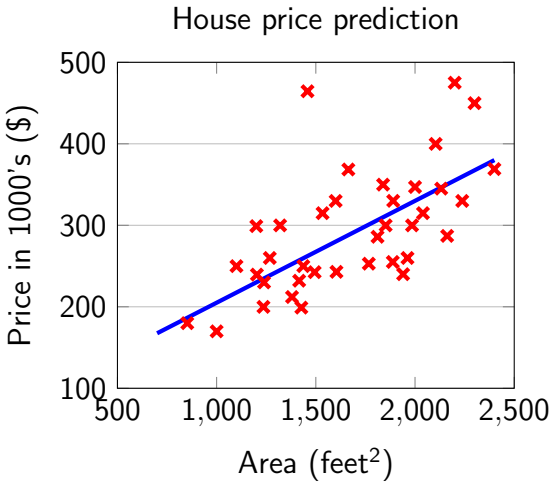
Data := [X,y] pairs

Model type

$$\hat{y} = f_{w,b}(X) = w * X + b$$

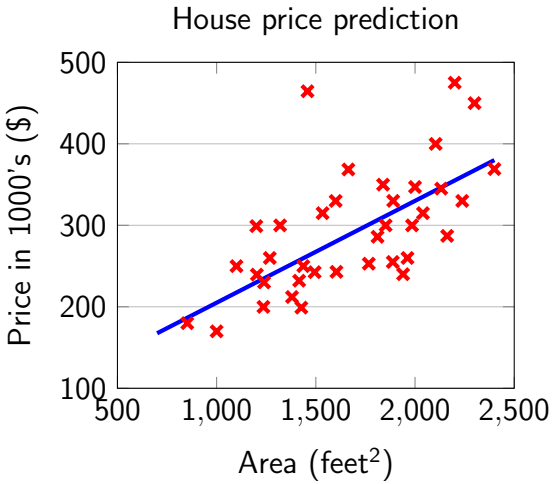
Find

w, b such that $\hat{y}$ is close to y



Loss function

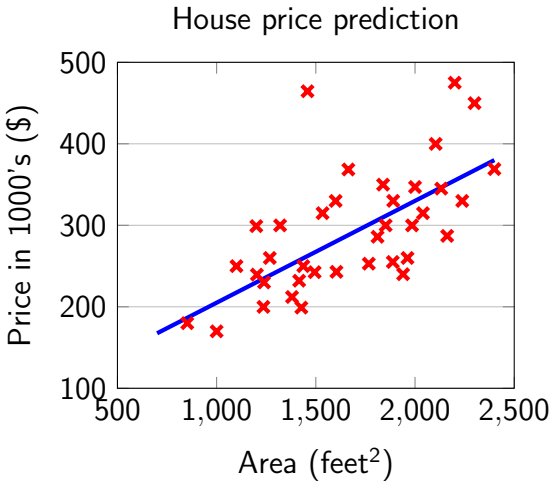
$$\underbrace{(\hat{y} - y)}_{error}$$



Loss function

$$\underbrace{(\hat{y} - y)}_{error}$$

$$(\hat{y} - y)^2$$

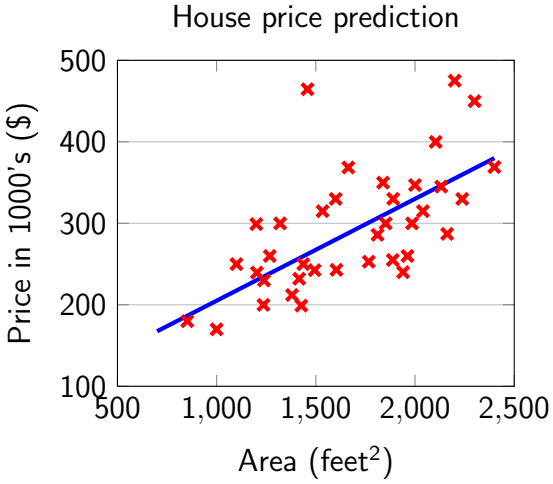


Loss function

$$\underbrace{(\hat{y} - y)}_{error}$$

$$(\hat{y} - y)^2$$

$$\frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(X^{(i)}) - y^{(i)} \right)^2$$

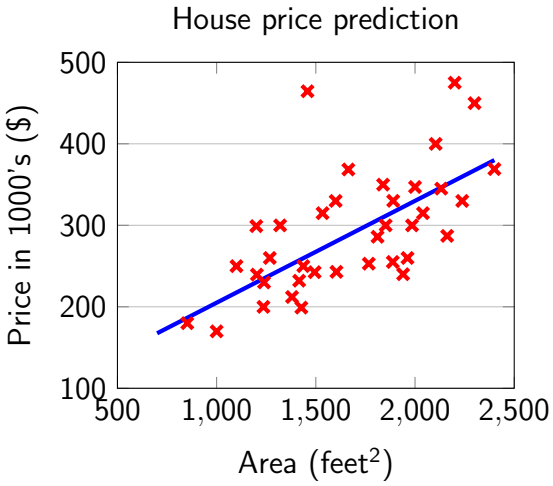


Loss function

$$\underbrace{(\hat{y} - y)}_{error}$$

$$(\hat{y} - y)^2$$

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(X^{(i)}) - y^{(i)} \right)^2$$



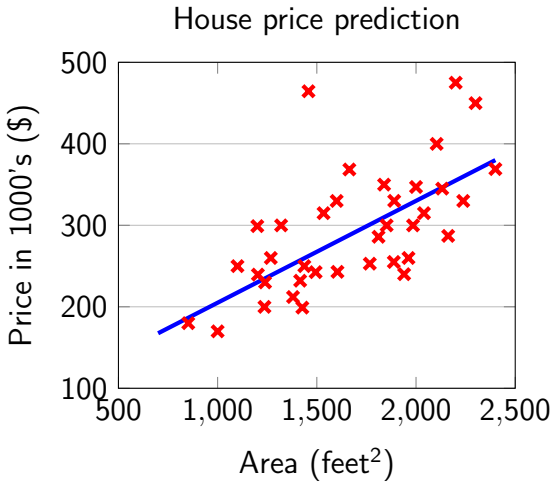
Loss function

$$\underbrace{(\hat{y} - y)}_{error}$$

$$(\hat{y} - y)^2$$

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(X^{(i)}) - y^{(i)} \right)^2$$

Least square error!



Minimisation problem

Data := [X,y] pairs

Model

$$\hat{y} = f_{w,b}(X) = w * X + b$$

Loss function

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(X^{(i)}) - y^{(i)} \right)^2$$

Minimise

$$\min_{w,b} J(w,b)$$

Minimisation problem

Data := [X,y] pairs

Model

$$\hat{y} = f_{w,b}(X) = w * X + b$$

Loss function

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(X^{(i)}) - y^{(i)} \right)^2$$

Minimise

$$\min_{w,b} J(w,b)$$

Simplified

Model

$$\hat{y} = f_{w}(X) = w * X$$

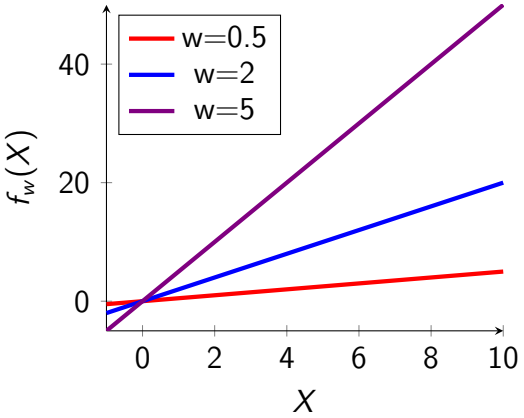
Loss function

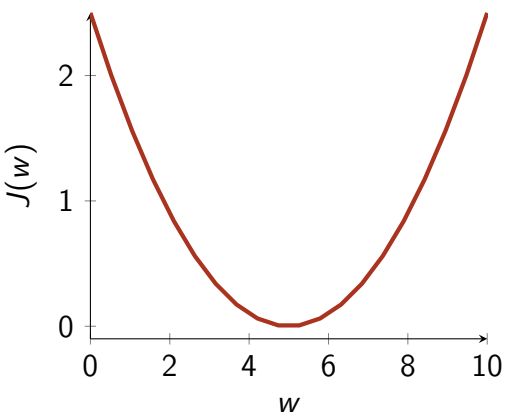
$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w}(X^{(i)}) - y^{(i)} \right)^2$$

Minimise

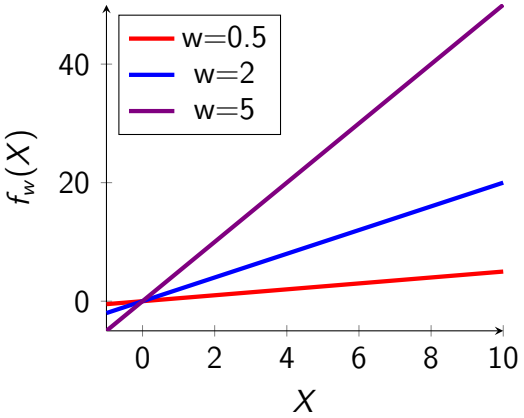
$$\min_{w} J(w)$$

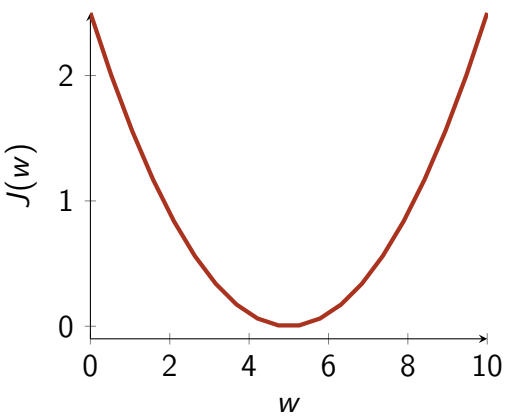
Minimisation problem



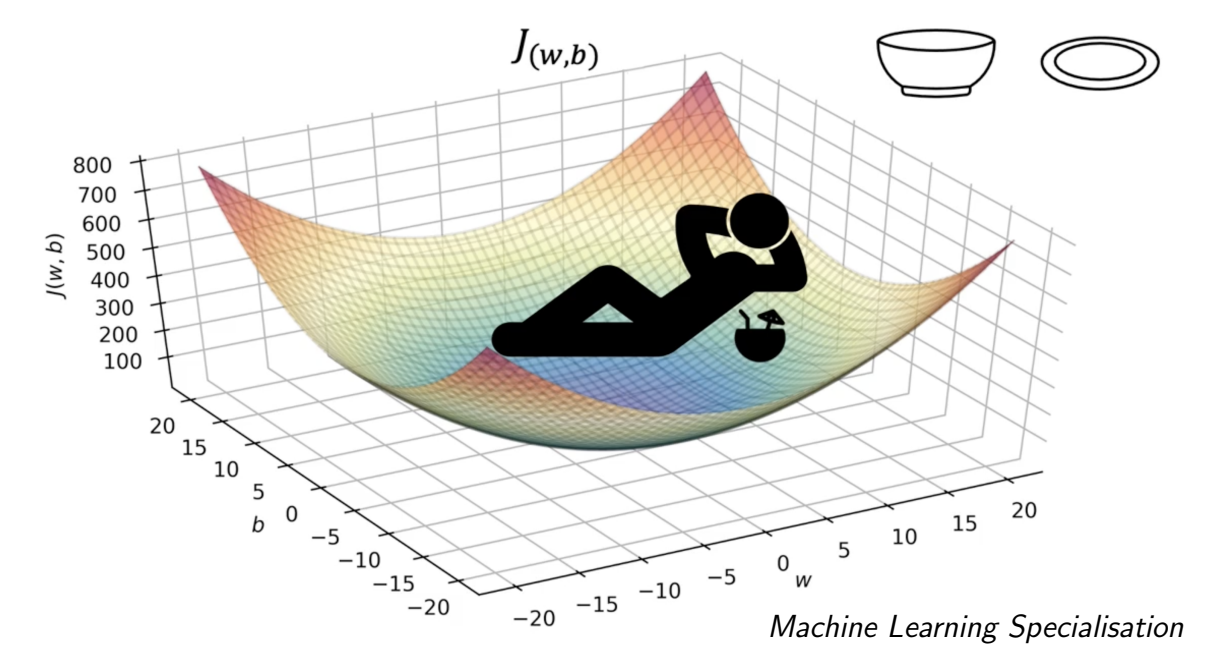


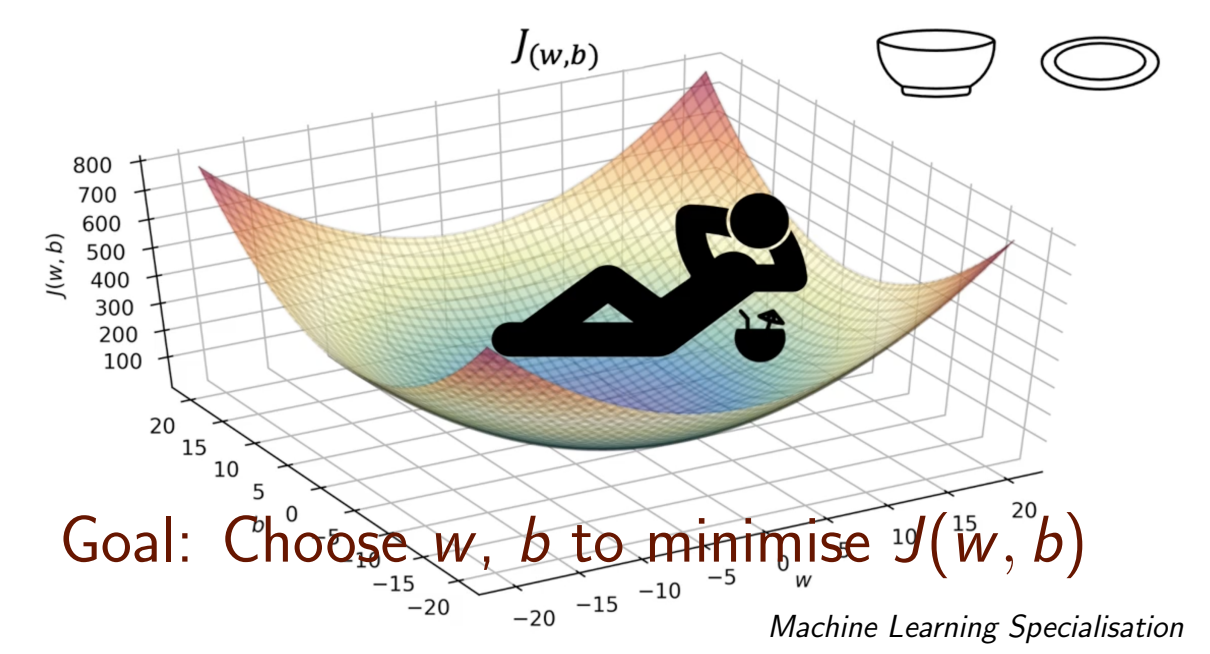
Minimisation problem





Goal: Choose w to minimise $J(w)$


$J_{(w,b)}$
800
700
600
500
400
300
200
100
$J(w,b)$
20
15
10
5
0
−5
−10
−15
−20
b
−20
−15
−10
−5
0
5
10
15
20
w
Machine Learning Specialisation



Goal: Choose w, b to minimise $J(w, b)$

Machine Learning Specialisation

Gradient Descent (Optimiser)

Loss function

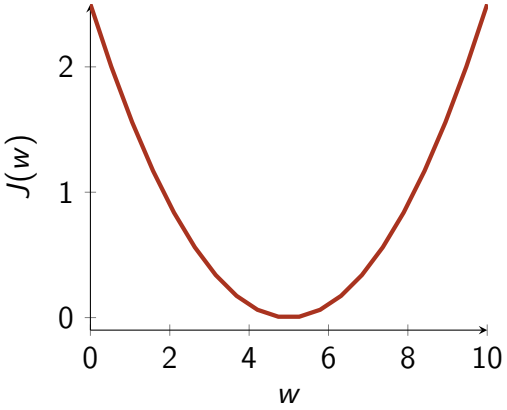
$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_w(X^{(i)}) - y^{(i)} \right)^2$$

Minimise

$$\min_w J(w)$$

Steps

- Start with some w, say $w = 0$
- Update w to decrease $J(w)$
- Till we settle at a minimum



Gradient descent algorithm

- Presuming $J(w, b)$

$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$

Gradient descent algorithm

- Presuming $J(w, b)$

$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$
$$b = b - \alpha \frac{\partial J(w, b)}{\partial b}$$

Gradient descent algorithm

- Presuming $J(w, b)$

$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$
$$b = b - \alpha \frac{\partial J(w, b)}{\partial b}$$

- $\alpha :=$ learning rate

Gradient descent algorithm

- Presuming $J(w, b)$

$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$
$$b = b - \alpha \frac{\partial J(w, b)}{\partial b}$$

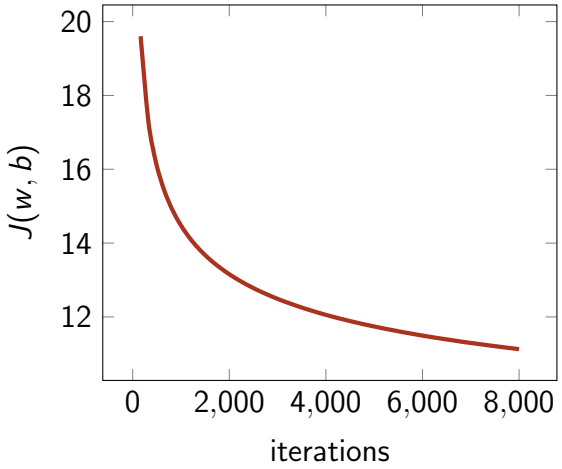
- $\alpha :=$ learning rate

- Algorithm

$$tmp\_w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$
$$tmp\_b = b - \alpha \frac{\partial J(w, b)}{\partial b}$$
$$w = tmp\_w$$
$$b = tmp\_b$$

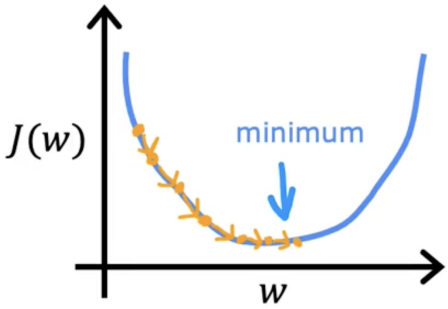
simultaneous update!

Gradient descent convergence

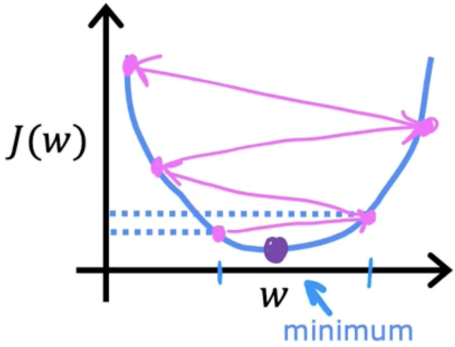


Effect of learning rate (α)

Small α

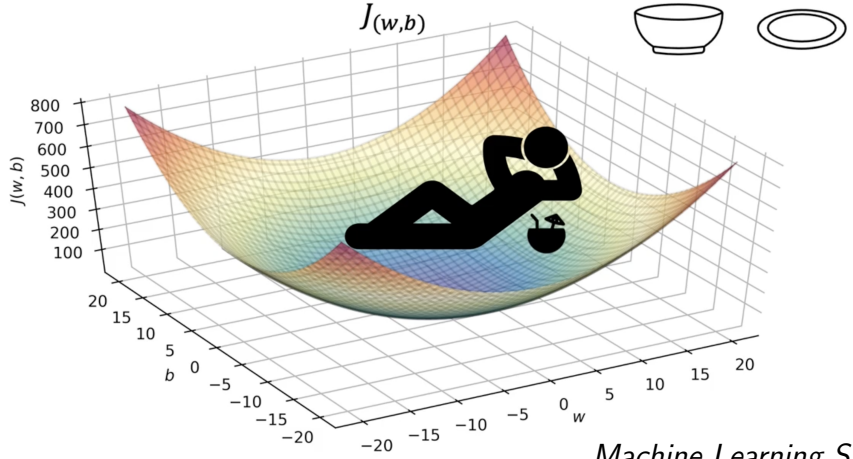


Large α



Machine Learning Specialisation

Talking about loss functions



Machine Learning Specialisation

Binary logistic regression (classification)

- Data: [X,y] pairs, e.g., $y \in [0, 1]$
- Learning

$$\hat{y} = f(X), \hat{y} \neq y$$

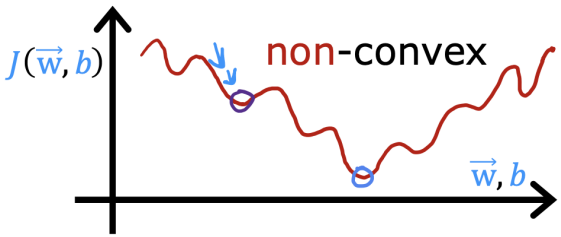
$$f_{w,b}(X) = \frac{1}{1 + \exp^{-(w*X+b)}}$$

Binary logistic regression (classification)

- Data: [X,y] pairs, e.g., $y \in [0, 1]$
- Learning

$$\hat{y} = f(X), \hat{y} \neq y$$

$$f_{w,b}(X) = \frac{1}{1 + \exp^{-(w*X+b)}}$$



- When using least squares error loss

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{w,b}(X^{(i)}) - y^{(i)} \right)^2$$

Logistic loss function

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

Logistic loss function

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(X^{(i)}) - y^{(i)}\right)^2$$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\begin{cases} -log(f_{w,b}(X^{(i)})) & \text{if } y^{(i)} = 1 \\ -log(1 - f_{w,b}(X^{(i)})) & \text{if } y^{(i)} = 0 \end{cases}$$

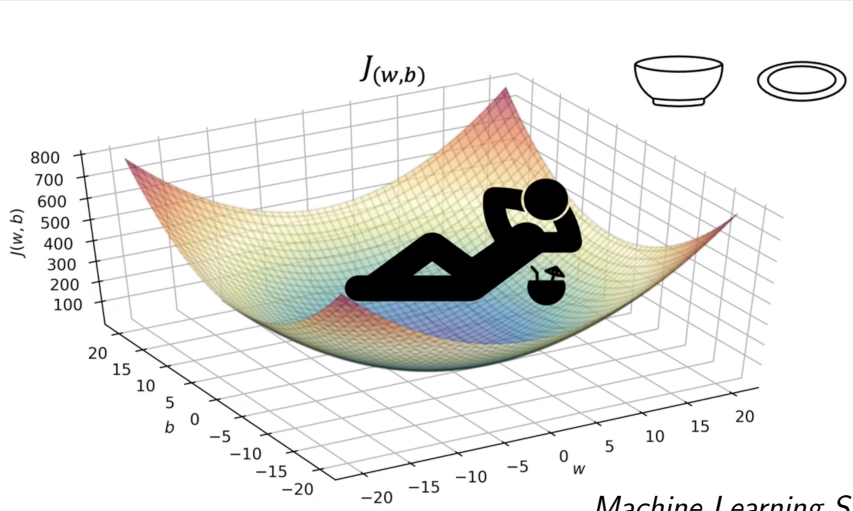
Logistic loss function

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(X^{(i)}) - y^{(i)}\right)^2$$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\begin{cases} -log(f_{w,b}(X^{(i)})) & \text{if } y^{(i)} = 1 \\ -log(1 - f_{w,b}(X^{(i)})) & \text{if } y^{(i)} = 0 \end{cases}$$

cross entropy loss

Brings us back to happy convex scenario



Machine Learning Specialisation

Life's more than univariate regression/classification

Data := $[[X_1, X_2, ..., X_n], y]$

Model

$$\hat{\vec{y}} = f_{\vec{w},b}(\vec{X}) = w_1 X_1 + ... + w_n X_n + b$$

Loss function

$$J(w_1, ..., w_n, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{X}^{(i)}) - y^{(i)} \right)^2$$

Minimise

$$\min_{\vec{w},b} J(\vec{w}, b)$$

Life's more than univariate regression/classification

Data := $[[X_1, X_2, ..., X_n], y]$

Model

$$\hat{\vec{y}} = f_{\vec{w},b}(\vec{X}) = w_1 X_1 + ... + w_n X_n + b$$

Loss function

$$J(w_1, ..., w_n, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{X}^{(i)}) - y^{(i)} \right)^2$$

Minimise

$$\min_{\vec{w},b} J(\vec{w}, b)$$

- Update weights and bias

$$w_j = w_j - \alpha \frac{\partial J(w_1, ..., w_n, b)}{\partial w_j}$$

$$b = b - \alpha \frac{\partial J(w_1, ..., w_n, b)}{\partial b}$$

Life's more than univariate regression/classification

Data := $[[X_1, X_2, ..., X_n], y]$

Model

$$\hat{\vec{y}} = f_{\vec{w},b}(\vec{X}) = w_1 X_1 + ... + w_n X_n + b$$

Loss function

$$J(w_1, ..., w_n, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{X}^{(i)}) - y^{(i)} \right)^2$$

Minimise

$$\min_{\vec{w},b} J(\vec{w}, b)$$

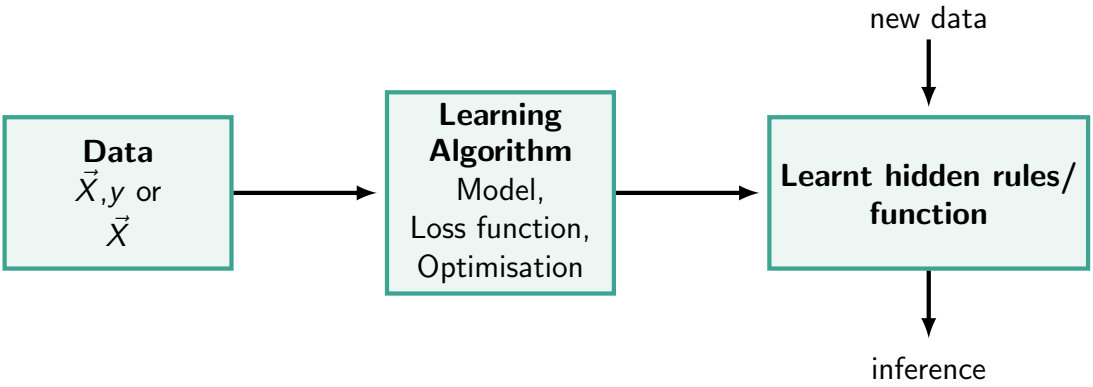
- Update weights and bias

$$w_j = w_j - \alpha \frac{\partial J(w_1, ..., w_n, b)}{\partial w_j}$$

$$b = b - \alpha \frac{\partial J(w_1, ..., w_n, b)}{\partial b}$$

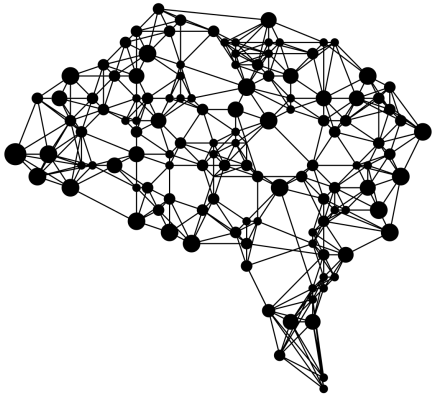
vectorised
$$\begin{cases} f_{\vec{w},b}(\vec{X}) = \vec{w} \cdot \vec{X} + b \\ w_j = w_j - \alpha \dfrac{\partial J(\vec{w}, b)}{\partial w_j} \\ b = b - \alpha \dfrac{\partial J(\vec{w}, b)}{\partial b} \end{cases}$$

And, that's machine learning



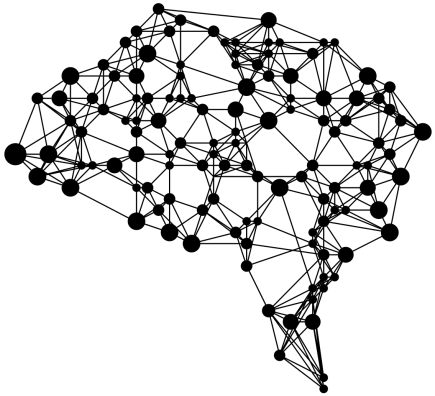
So far

- Rise of Deep Learning
- Building blocks of DL
- What is ML?, data forms and classes of learning algorithms
- Terms: Feature, target/label, weights, bias, loss function, gradient descent optimiser

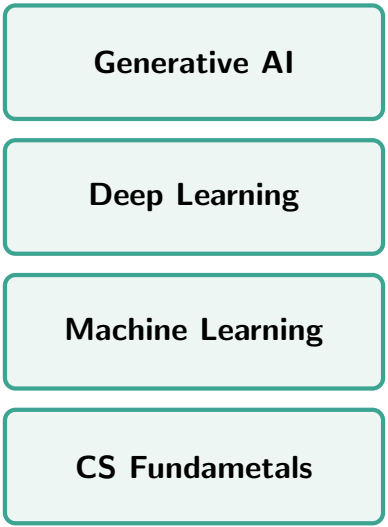


So far

- Rise of Deep Learning
- Building blocks of DL
- What is ML?, data forms and classes of learning algorithms
- Terms: Feature, target/label, weights, bias, loss function, gradient descent optimiser
- Feature scaling and engineering
- Different types of optimisers
- Bias-variance trade off, i.e, overfitting/underfitting, regularisation

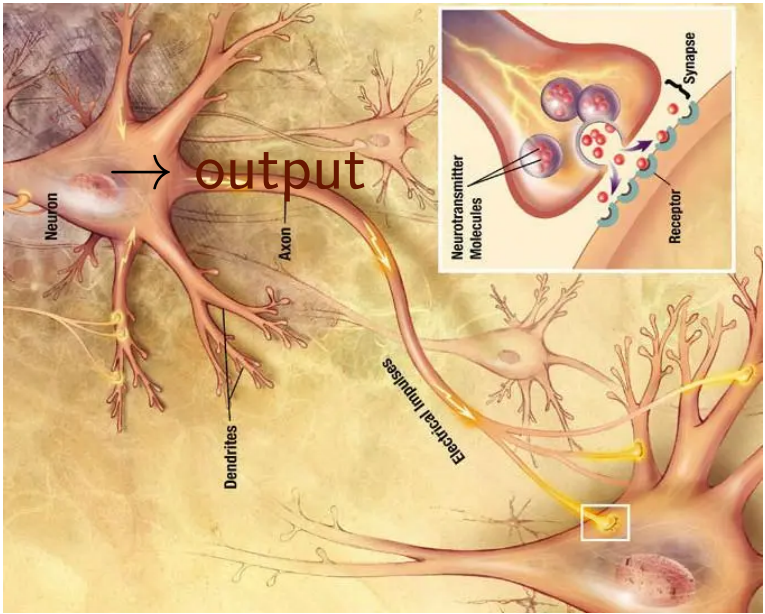


DL building blocks

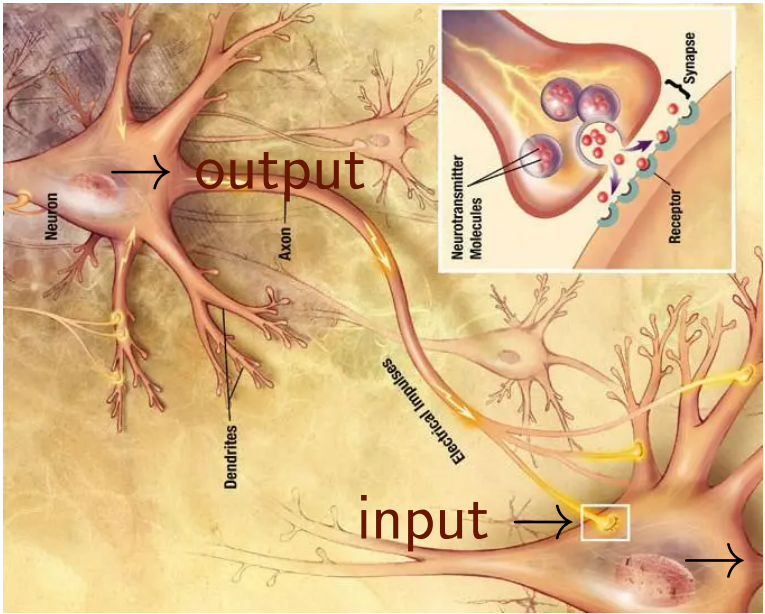


Neuron
Axon
Dendrites
Electrical Impulses
Neurotransmitter Molecules
Receptor
Synapse

input → → output

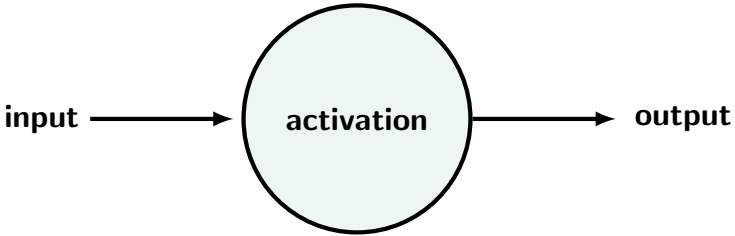


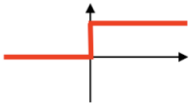
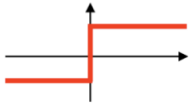
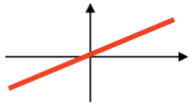
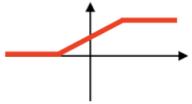
input → → output

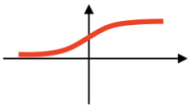
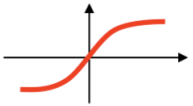
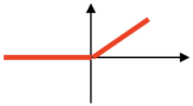
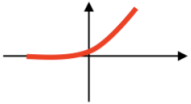


input → → output

Artificial neuron

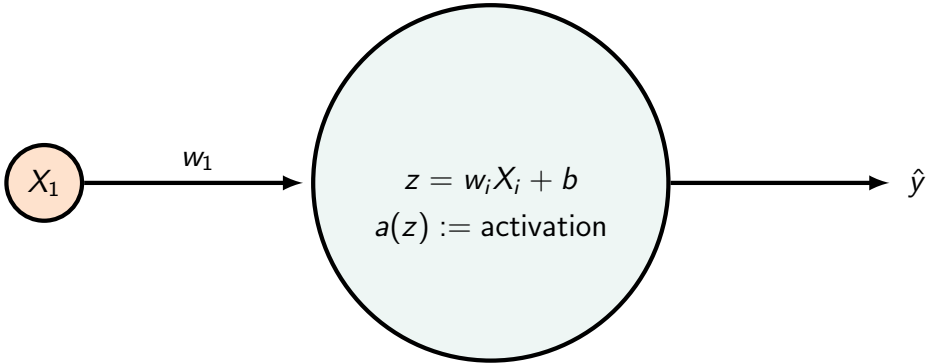


Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	

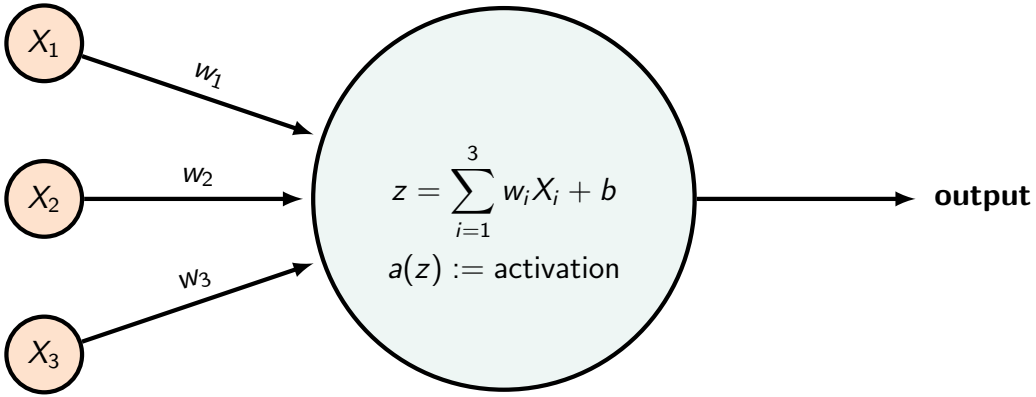
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016
(http://sebastianraschka.com)

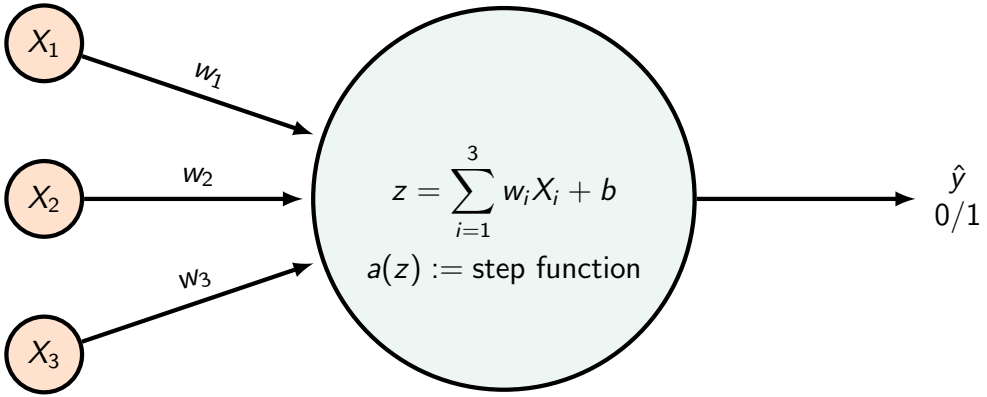
Artificial neuron network, univariate example



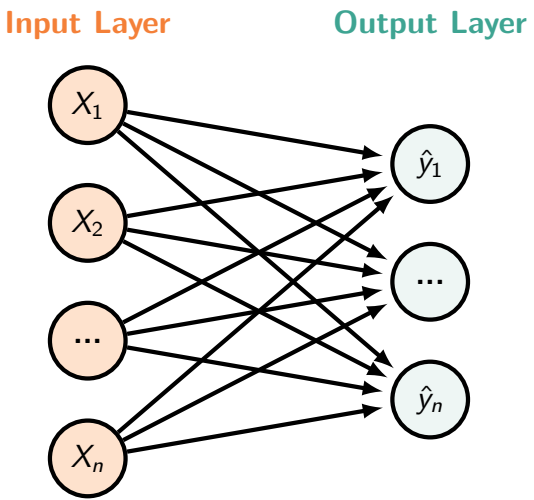
Artificial neuron network, three variables example



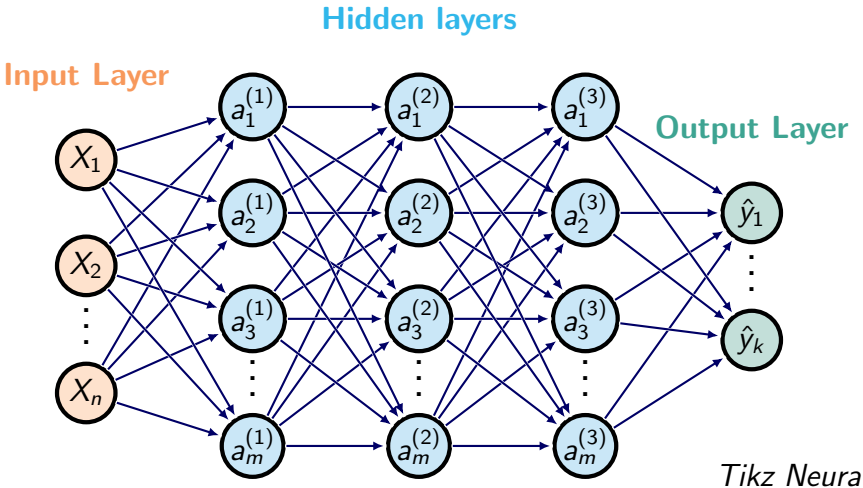
Perceptron (1958)



Single layer perceptron (SLP)



Due to limitations of SLP, Multi layer perceptron (MLP)



Tikz Neural Networks

Aspect	Single-Layer Perceptron (SLP)	Multi-Layer Perceptron (MLP)
Architecture	One input layer, one output layer (no hidden layers)	Input, hidden, and output layers
Problem Solvability	Solves only linearly separable problems (e.g., AND)	Solves both linear and non-linear problems (e.g., XOR)
Activation Function	Step function (binary output)	Non-linear functions (e.g., sigmoid, ReLU, tanh)
Learning Algorithm	Perceptron learning rule (no backpropagation)	Trained using backpropagation and gradient descent
Output	Binary (0 or 1)	Continuous (for regression) or multi-class (for classification)

Abhishek Jain, medium

SLP vs MLP

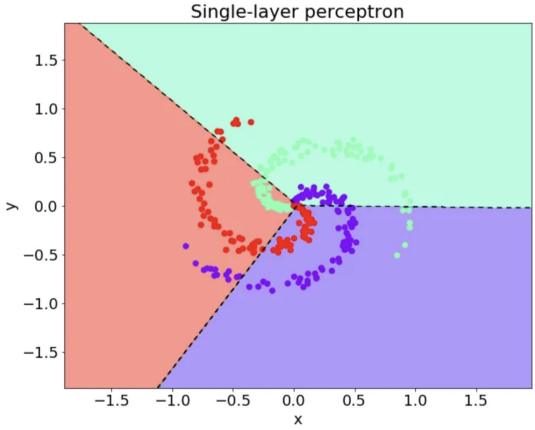
- **Single-layer perceptron**:

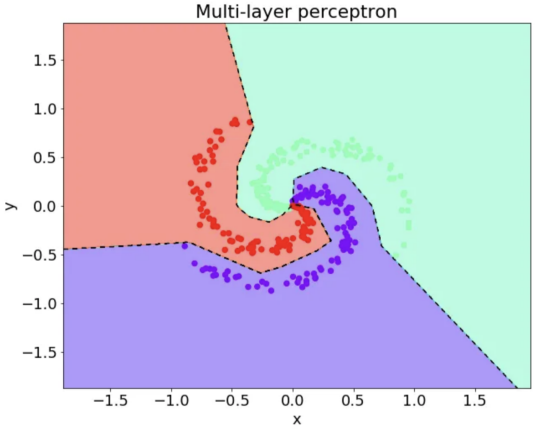
  2 inputs/features (X_1-X_2 coordinates) and 3 outputs (3 colors)
- **Multi-layer perceptron**:

  2 inputs/features (X_1-X_2 coordinates), one hidden layer with 50 neurons, and 3 outputs (3 colors)

Linh Tran, medium

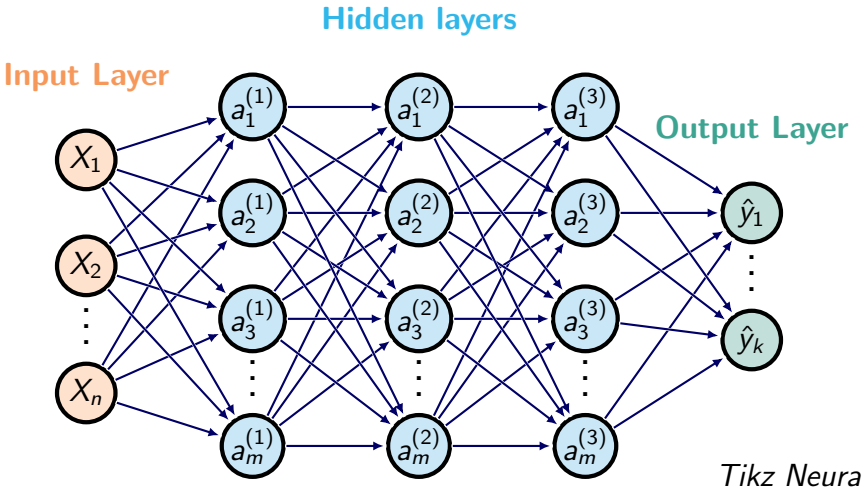
SLP vs MLP



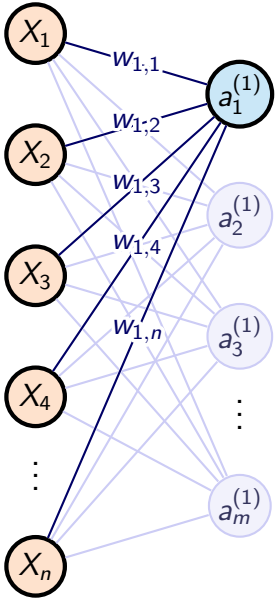


Linh Tran, medium

Closer look at MLP



Tikz Neural Networks



$$= \sigma\left(w_{1,1}X_1 + w_{1,2}X_2 + \ldots + w_{1,n}X_n + b_1^{(0)}\right)$$

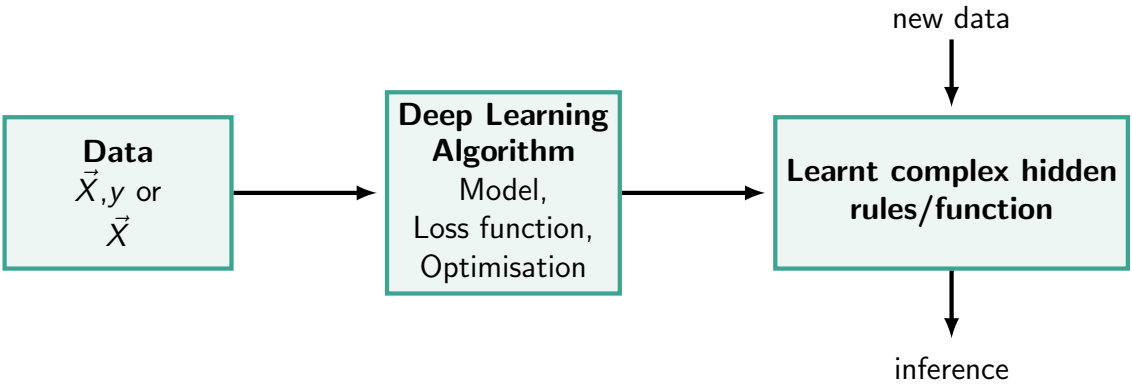
$$= \sigma\left(\sum_{i=1}^{n} w_{1,i}X_i + b_1\right)$$

$$\begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_m^{(1)} \end{pmatrix} = \sigma\left[\begin{pmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,n} \\ w_{2,1} & w_{2,2} & \ldots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \ldots & w_{m,n} \end{pmatrix}\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} + \begin{pmatrix} b_1^{(0)} \\ b_2^{(0)} \\ \vdots \\ b_m^{(0)} \end{pmatrix}\right]$$

$$\mathbf{a}^{(1)} = \sigma\left(\mathbf{W}^{(0)}\mathbf{X} + \mathbf{b}^{(0)}\right)$$

Tikz Neural Networks

Deep learning in a nutshell



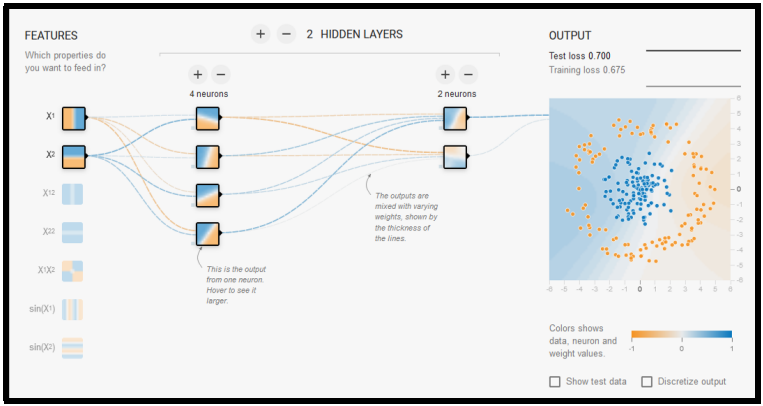
Exercise #1

- Get a Google account. Note that having a gmail address implies you already have a Google account.
- Check out the Multi-layer Perceptron spreadsheet. Follow the steps as described in the description in the following linked article (click here).

Exercise #2

- Go ahead only if you have MS Excel installed on your machine. If not skip this exercise and move to the next one.
- Check out how forward and backward propogation works in a neural network. Follow the steps as described in the description in the following linked article (click here).

Exercise #3



Check out the graphical DL tool from Deep Playground (click here).