nodeflux

Extending Vision Beyond Imagination

Machine Learning and Neural Network

Outline

Introduction

Outline This Camp

Outline Camp DSW Basic and Introduction Nodeflux

Machine Learning

Introduction Machine Learning

Basic explanation about Machine Learning

Neural Network

Introduction Neural Network

Where else we use Feed Forward and Backpropagation, why is it still impactful

MNIST Classification with Tensorflow

Coding

What we compute and how

Nodeflux Introduction

nodeflux
Extending Vision Beyond Imagination

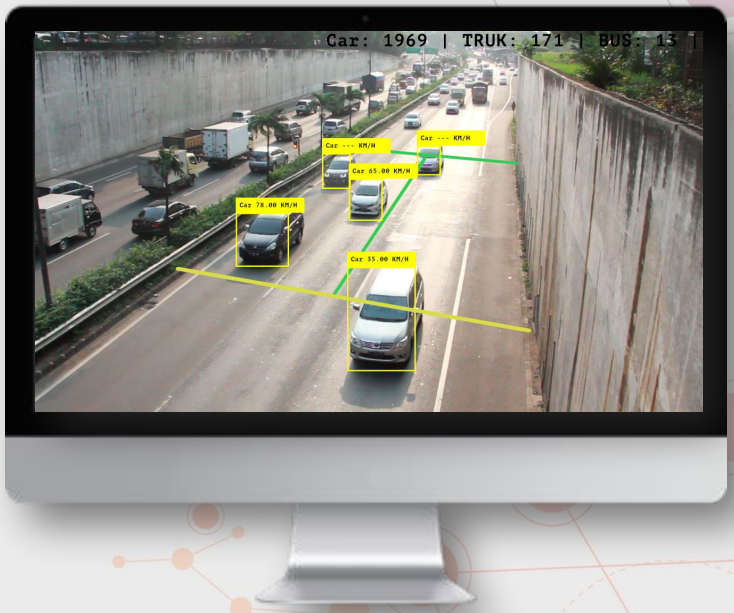
Our Company

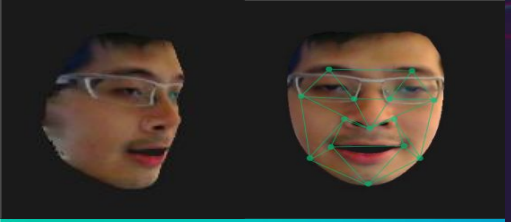
Nodeflux is a technology startup company from Jakarta, Indonesia, founded on 2016 by Meidy Fitranto (CEO) and Faris Rahman (CTO), two engineers from Bandung Institute of Technology.

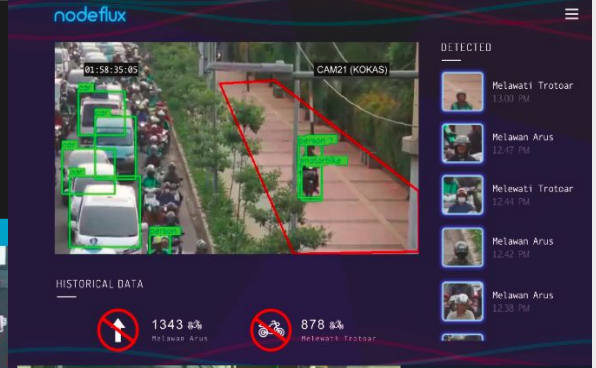
Nodeflux provide computer vision with deep learning to solve many challenging problems by extending vision using Artificial Intelligence.

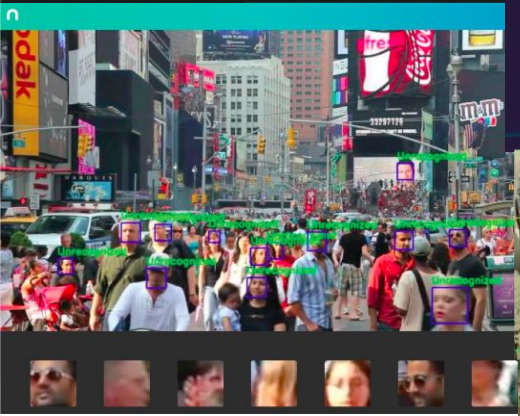
Nodeflux Intelligent Video Analytics can be deployed into any kind of source, whether its CCTV, webcam, phone, camera, or others. Many kinds of logic rules can be applied, even can be customized specifically only for client's business process or needs.

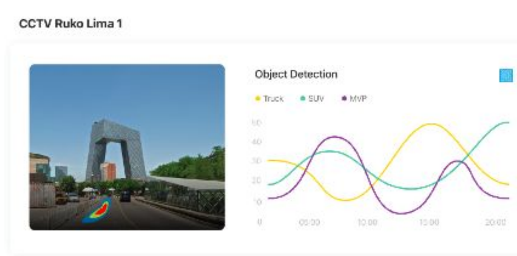
Nodeflux's products and services cover wide range of sector including but not limited to smart city, defense and security, traffic management, toll management, store analytic (wholesale and retail), asset and facilities management, advertising, and transportation.

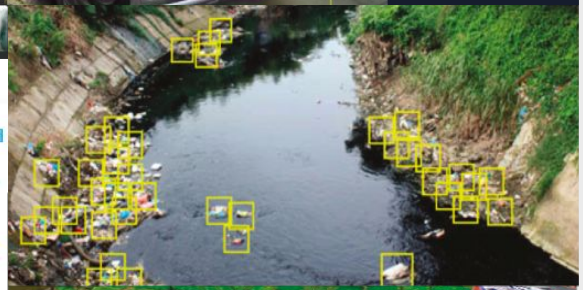


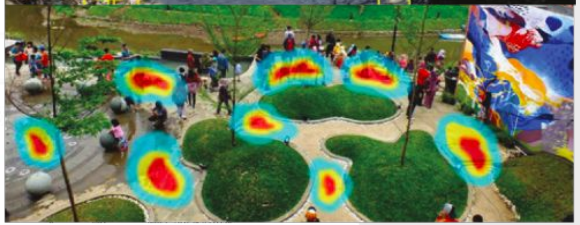








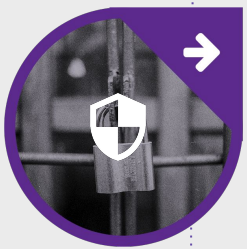




Product Overview

Nodeflux's products and services cover wide range of sector including but not limited to smart city, defense and security, traffic management, toll management, store analytic (wholesale and retail), asset and facilities management, advertising, and transportation.

INDUSTRY SOLUTION

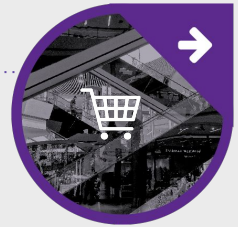


Defense and Security

- Face Recognition
- 2D to 3D Face Reconstruction
- Time Compression Analysis
- Pixel Enhancement
- License Plate Recognition
- Crowd Behaviour Analysis

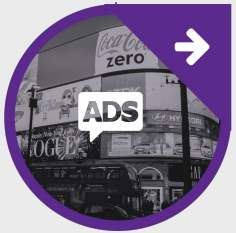
Smart City

- Traffic Monitoring
- Road and River Monitoring
- Flood Monitoring
- Vehicle Detection
- Illegal Parking Detection
- Dynamic Traffic Lights



Store Analytics

- Visitor Counting
- Visitor Trajectory Flow
- Visitor Heat Map
- Product View Rank
- Queue Analysis



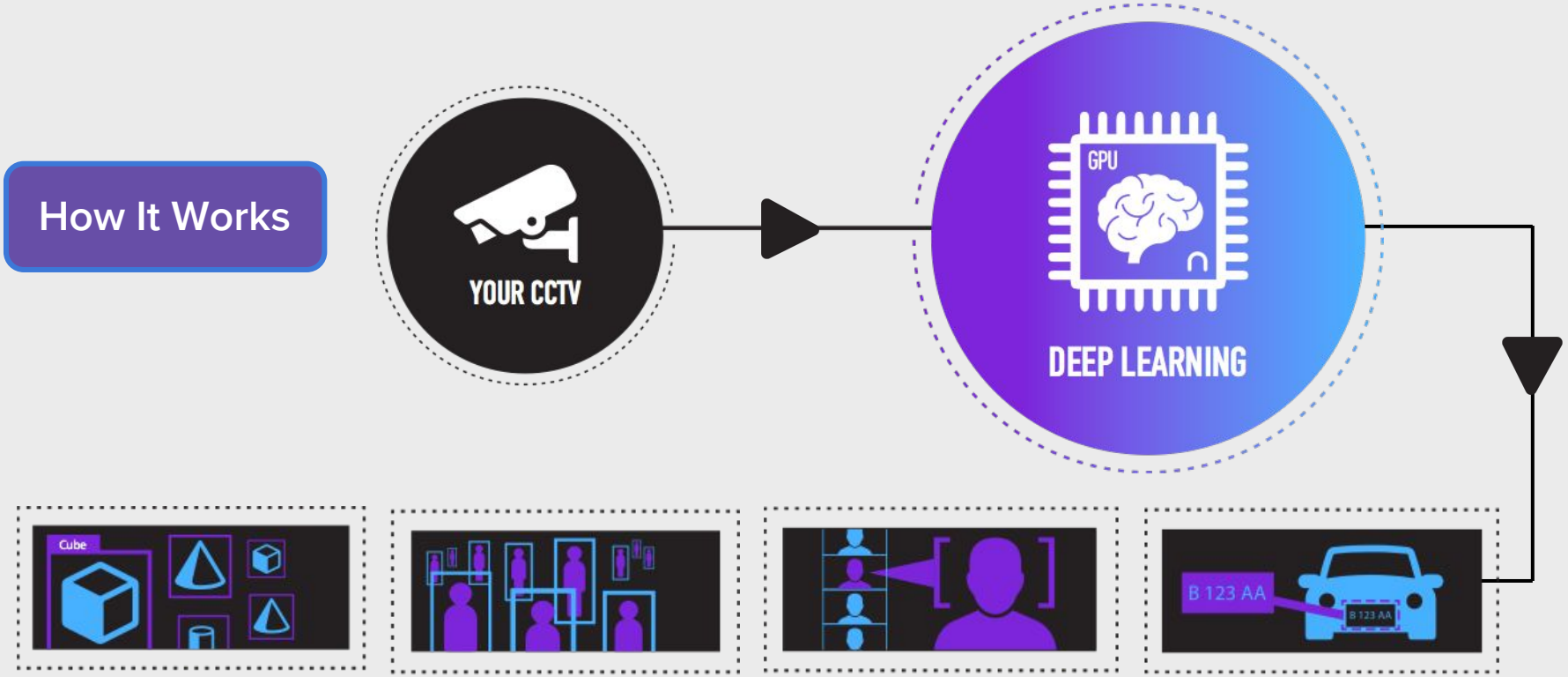
Advertising

- Audience Gender, Age, Emotion
- Audience View Duration

Toll Management

- Vehicle Counting
- Vehicle Classification
- Vehicle Flow
- Vehicle Speed
- Incident Detection

How It Works

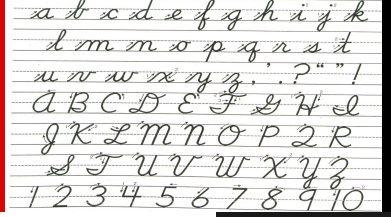


Actionable Insight Advanced Analytic Automation & Alert

NVIDIA | nodeflux
Extending Vision Beyond Imagination

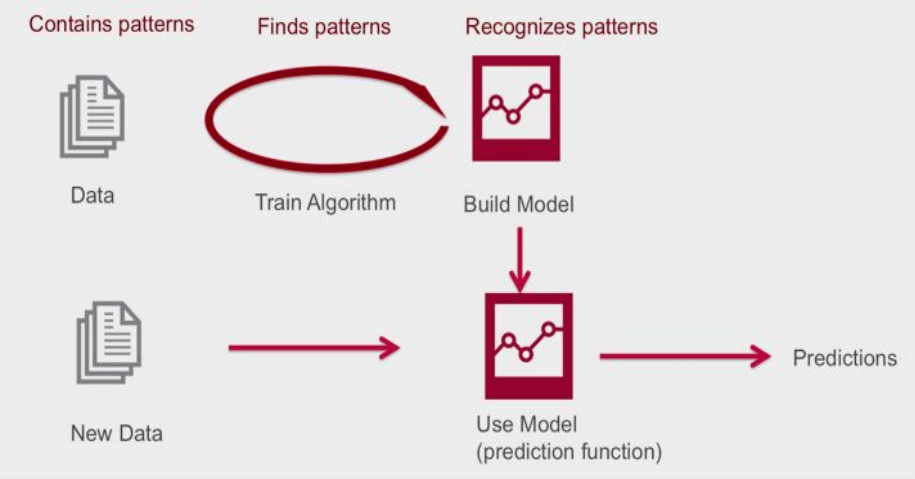
Introduction
Machine Learning

NETFLIX



Spotify®

What is Machine Learning?



Machine Learning is field of study that gives computers the ability to learn without being explicitly programmed

Arthur Samuel (1959)

Image by: https://mapr.com/blog/demystifying-ai-ml-dl/

What is Machine Learning?

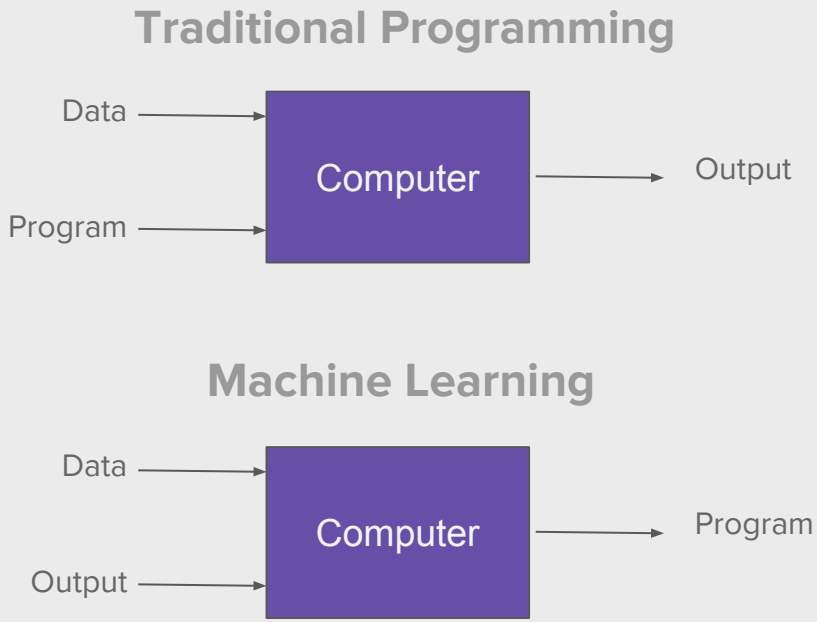
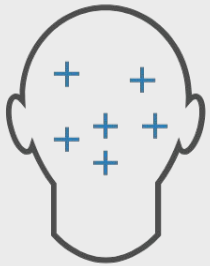


Image by: https://www.youtube.com/watch?v=h0e2HAPTGF4

Why we use **Machine Learning?**

Consider using **machine learning** when you have a complex task or problem involving a large amount of data and lots of variables, ut no existing formula or equation.



Hand-written rules and equations are too complex as in face recognition and speech recognition.

The rules of a task are constantly changing as in fraud detection from transaction records.

The nature of the data keeps changing, and the program needs to adapt—as in automated trading, energy demand forecasting, and predicting shopping trends.

How are things Learn?

Memorization

Accumulation of individual facts

Limited by

- Time to observe facts
- Memory to store facts

Declarative Knowledge

Generalization

Deduce new facts from old facts

Limited by accuracy of deduction process

- Essentially a predictive activity
- Assumes that the past predicts the future

Imperative Knowledge

Interested in extending to programs that can infer useful information from **implicit** patterns in data

Image by: https://www.youtube.com/watch?v=h0e2HAPTGF4

Basic Paradigm

Observe set of set examples : **training data**

Infer something about process that generated that data

Use inference to make predictions about previously unseen data:

test data

Image by: https://www.youtube.com/watch?v=h0e2HAPTGF4

Type of Machine Learning

Supervised Learning

This algorithm consist of a target / outcome variable (or dependent variable) which is to be predicted from a given set of predictors (independent variables).

Unsupervised Learning

In this algorithm, we do not have any target or outcome variable to predict / estimate.

Reinforcement Learning

Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it trains itself continually using trial and error.

Source : https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

Technique of **Machine Learning**

Supervised Learning

Classification:

Based on training data having observations with known categories.

Example :

- Fraud detection (fraud, not fraud)
- Credit card application (good credit, bad credit)
- Email spam detection (spam, not spam)

Regression:

Predicting a value from a continuous data set.

Example :

- Predicting the price of a house

Unsupervised Learning

Clustering :

An algorithm classifies inputs into categories by analyzing similarities between input examples.

Example :

- Grouping similar customers
- Text categorization

Dimensionality Reduction :

In machine learning, “dimensionality” simply refers to the number of features (i.e. input variables) in your dataset.

Example :

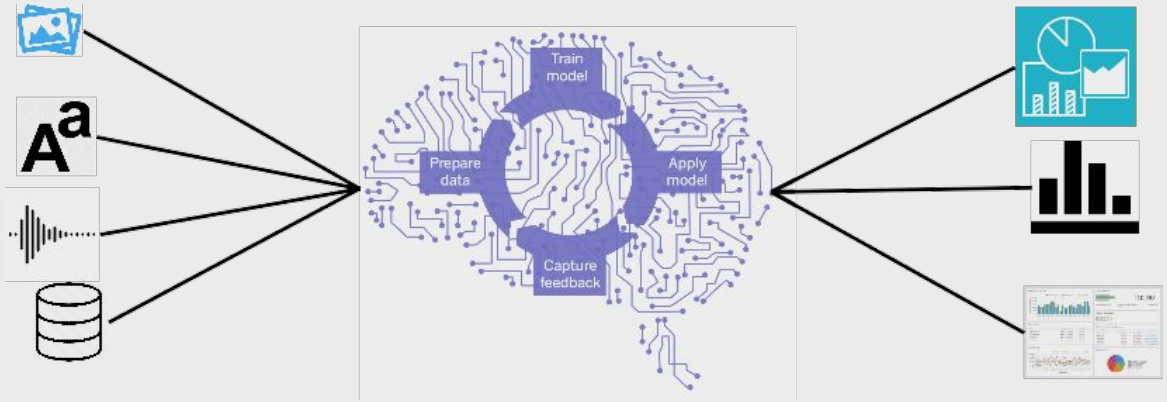
- Noise reducing

How Machine Learning works?

Input Data Source

Build Model

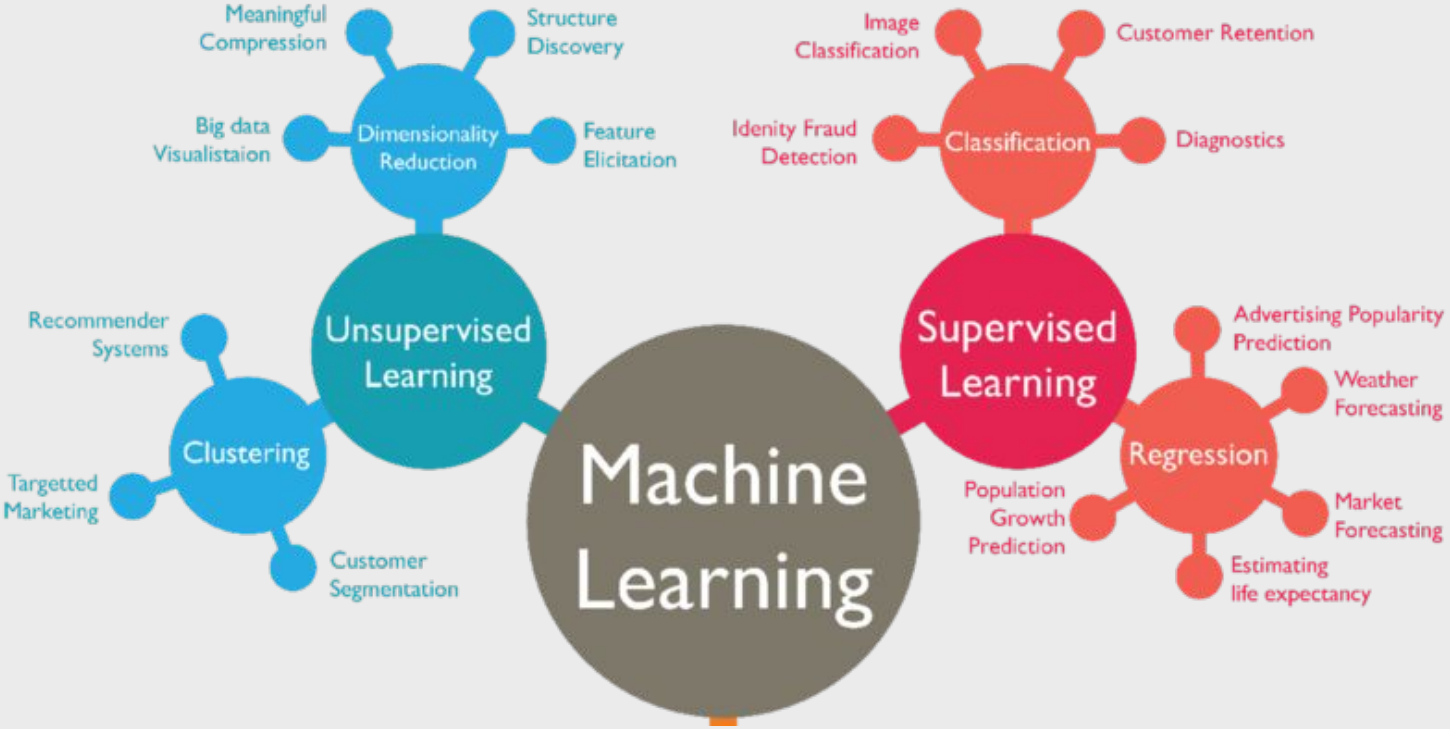
Predictive and visualization



Machine Learning Work-flow breakdown

1. Identifying the problem
2. Data Explorations
3. Data Pre-Processing
4. Model Development
5. Model Improvement

Example case: **Based on Technique**

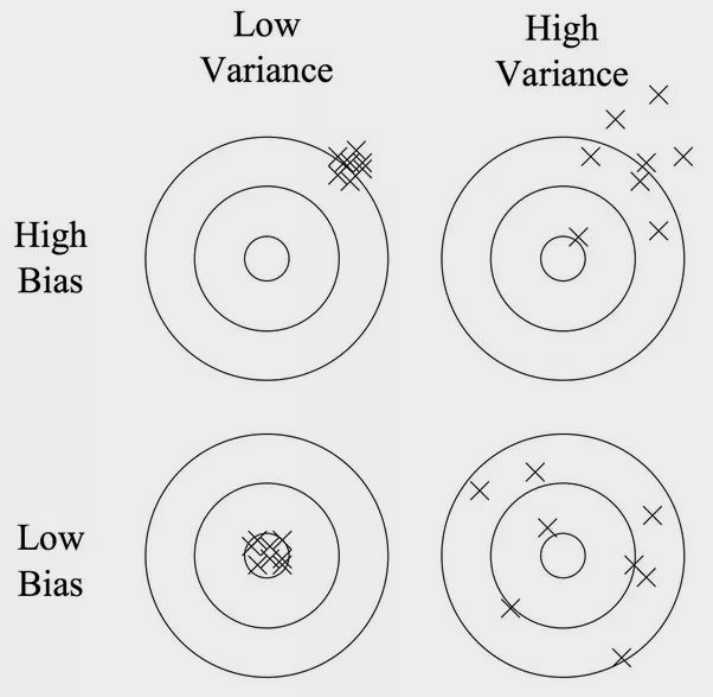


How do we decide Which Machine Learning Algoritm to Use?

There is no best method or one size fits all. Finding the right algorithm is partly just trial and error. But, algorithm selection also depends on the size and type of data you're working with, the insights you want to get from the data, and how those insights will be used.

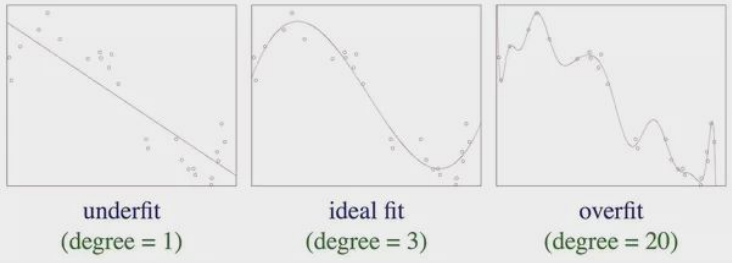
Choosing the right algorithm can seem overwhelming—there are dozens of supervised and unsupervised machine learning algorithms, and each takes a different approach to learning.

Bias Variance Trade-Off



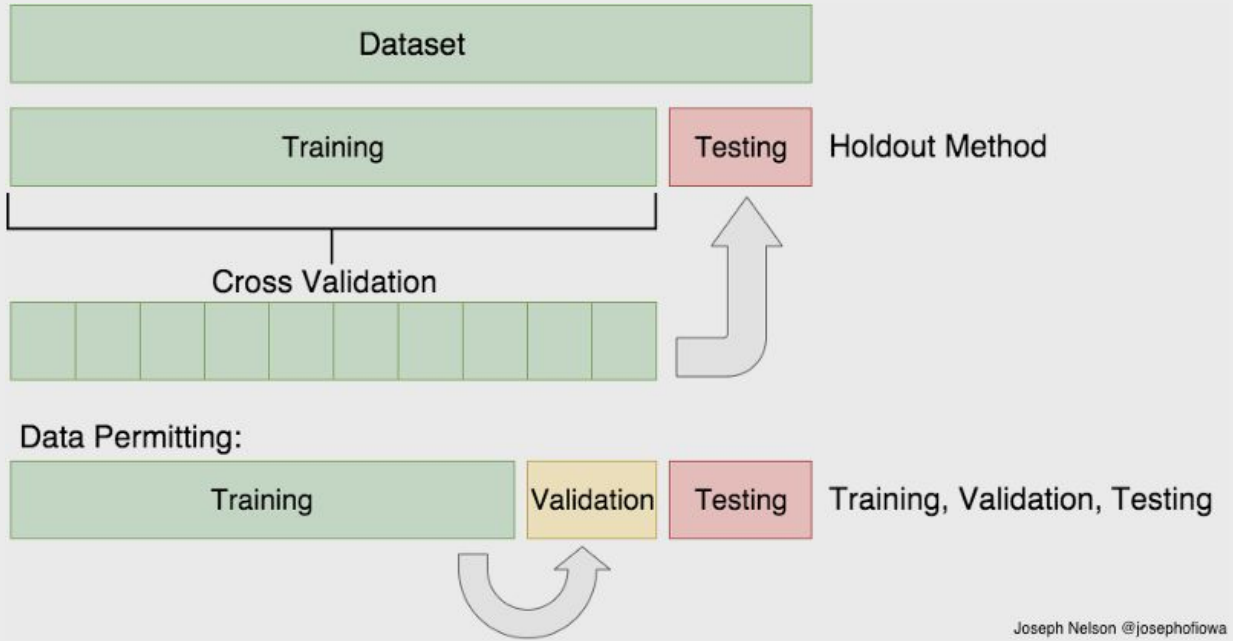
Bias is the algorithm's tendency to consistently learn the wrong thing by not taking into account all the information in the data (**underfitting**).

Variance is the algorithm's tendency to learn random things irrespective of the real signal by fitting highly flexible models that follow the error/noise in the data too closely (**overfitting**).



Source : https://www.quora.com/What-is-the-best-way-to-explain-the-bias-variance-trade-off-in-layman%E2%80%99s-terms

Train/Test Split and Cross Validation



Source : https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

Accuracy: Overall, how often is the classifier correct?

(TP+TN)/total = (100+50)/165 = 0.91

Misclassification Rate: Overall, how often is it wrong?

(FP+FN)/total = (10+5)/165 = 0.09

True Positive Rate: When it's actually yes, how often does it predict yes?

TP/actual yes = 100/105 = 0.95

False Positive Rate: When it's actually no, how often does it predict yes?

FP/actual no = 10/60 = 0.17

Specificity: When it's actually no, how often does it predict no?

TN/actual no = 50/60 = 0.83

Precision: When it predicts yes, how often is it correct?

TP/predicted yes = 100/110 = 0.91

Prevalence: How often does the yes condition actually occur in our sample?

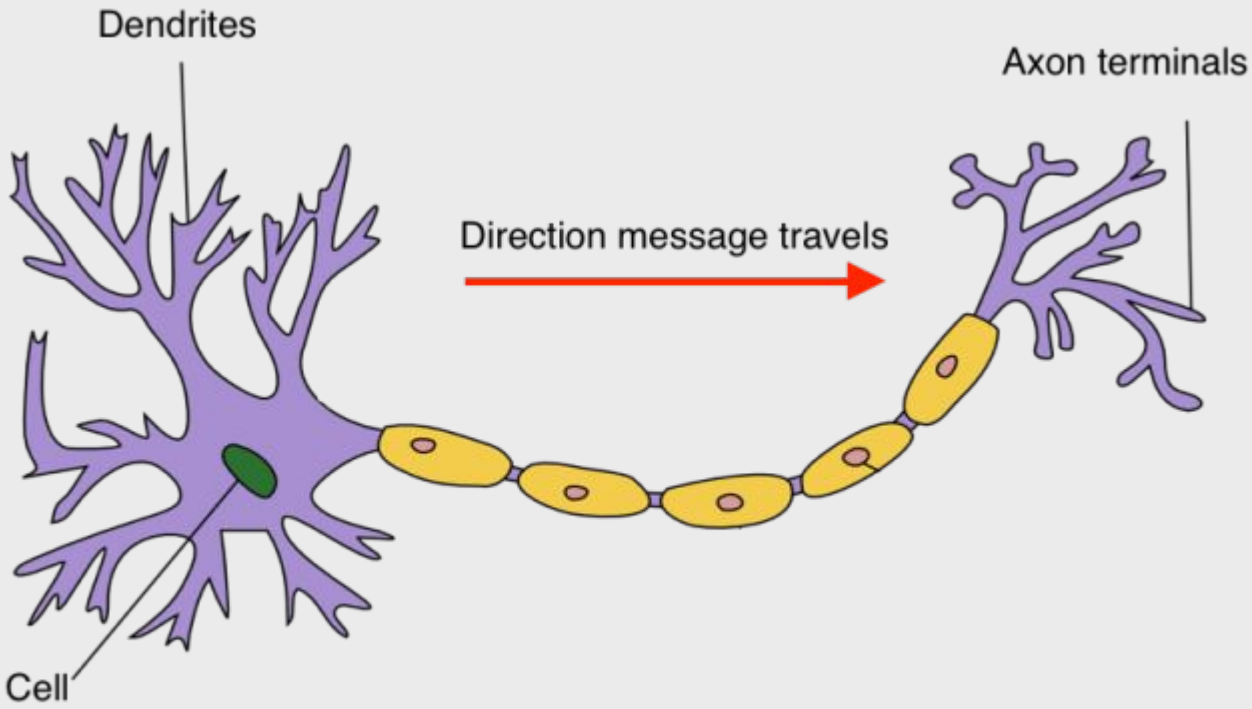
actual yes/total = 105/165 = 0.64

Source : https://classeval.wordpress.com/introduction/basic-evaluation-measures/, http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/,

Application of Machine Learning

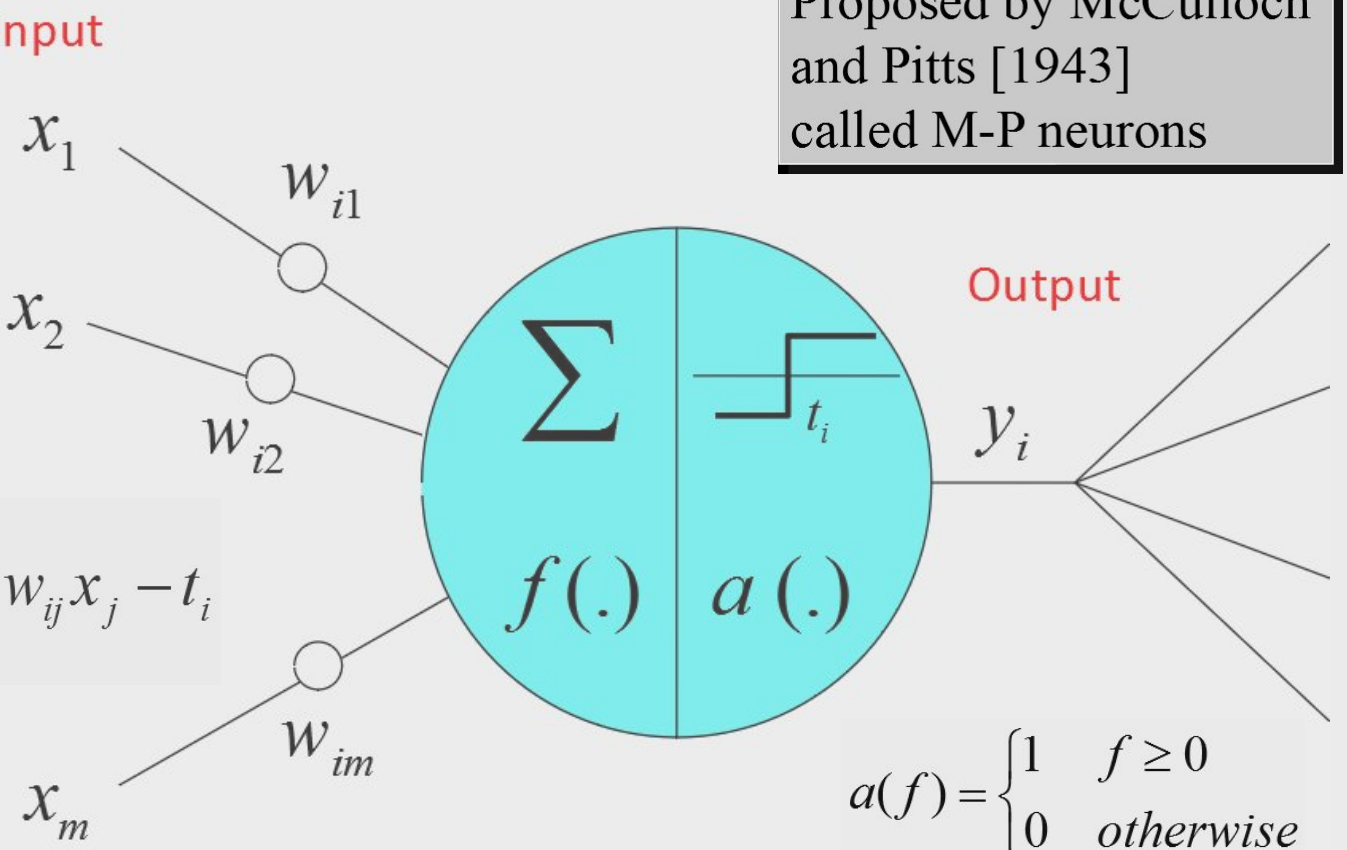
Neural Network

Neuron in the Brain



The Artificial Neuron

Proposed by McCulloch and Pitts [1943] called M-P neurons



$$f(\mu_i) = \sum_{j=1}^{m} w_{ij} x_j - t_i$$

$$a(f) = \begin{cases} 1 & f \geq 0 \\ 0 & otherwise \end{cases}$$

Neuron

The neuron is the basic information processing unit of a Neural Network. It consists of:

1. A set of links, describing the neuron inputs, with weights w1, w2, ..., wm
2. An **adder functio**n (linear combiner) for computing the weighted sum of the inputs:

   (real numbers) $$f = \sum_{i=1}^{i=m} w_i x_i$$

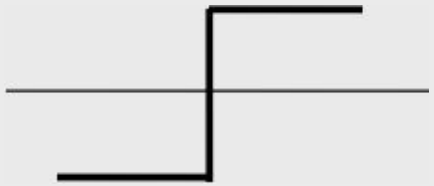
3. Activation function a(.) for limiting the amplitude of the neuron output.

$$y = a(f)$$

Activation Function a(.)

- Step function

$$a(f) = \begin{cases} 1 \text{ if } f \geq t \\ 0 \text{ otherwise} \end{cases}$$

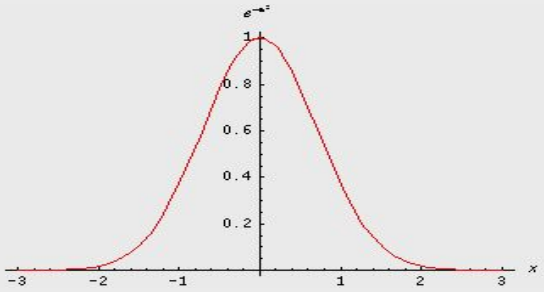


- Sigmoid function

$$a(f) = \frac{1}{1+\exp(-f)}$$

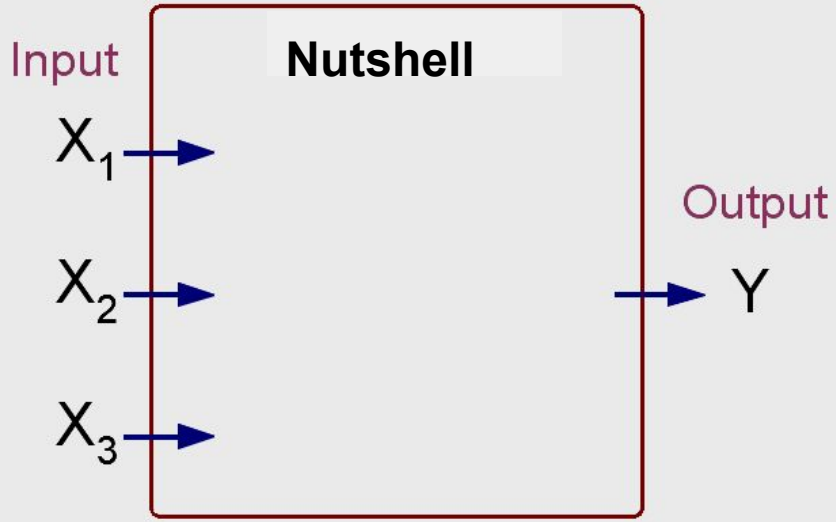
- Gaussian function

$$a(f) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{f-\mu}{\sigma}\right)^2\right)$$



NN in Nutshell

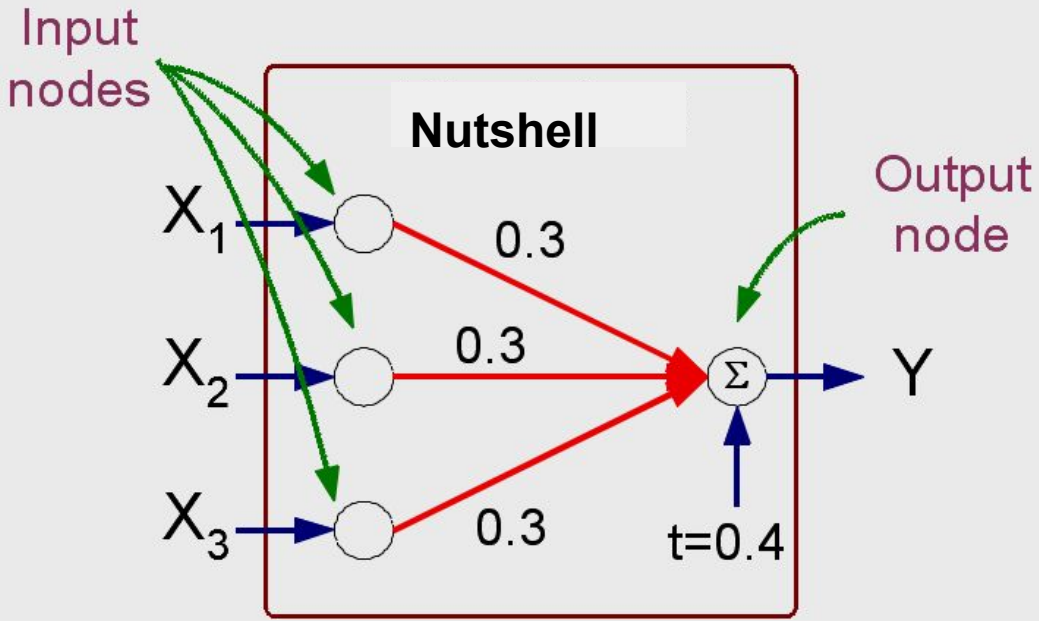
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

NN in Nutshell

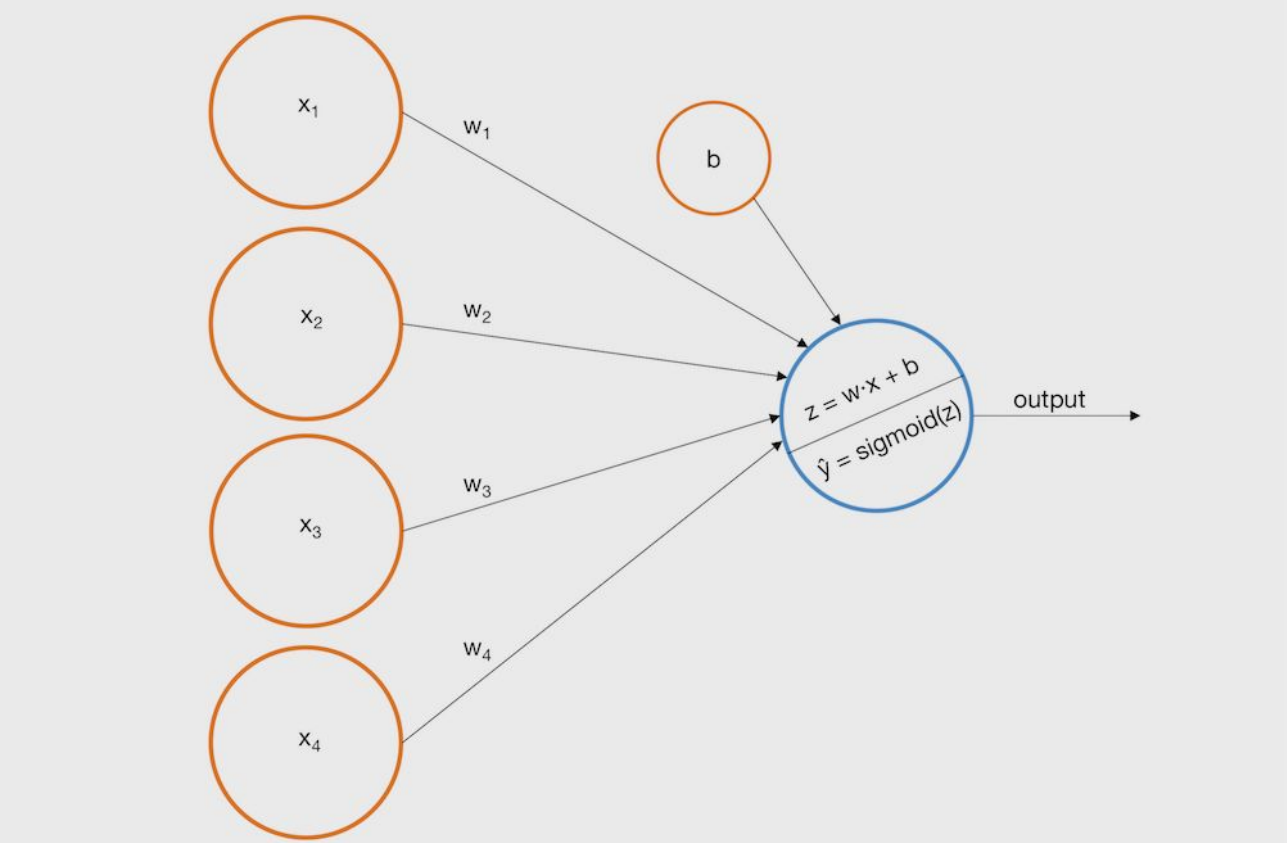
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



$$Y = a(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

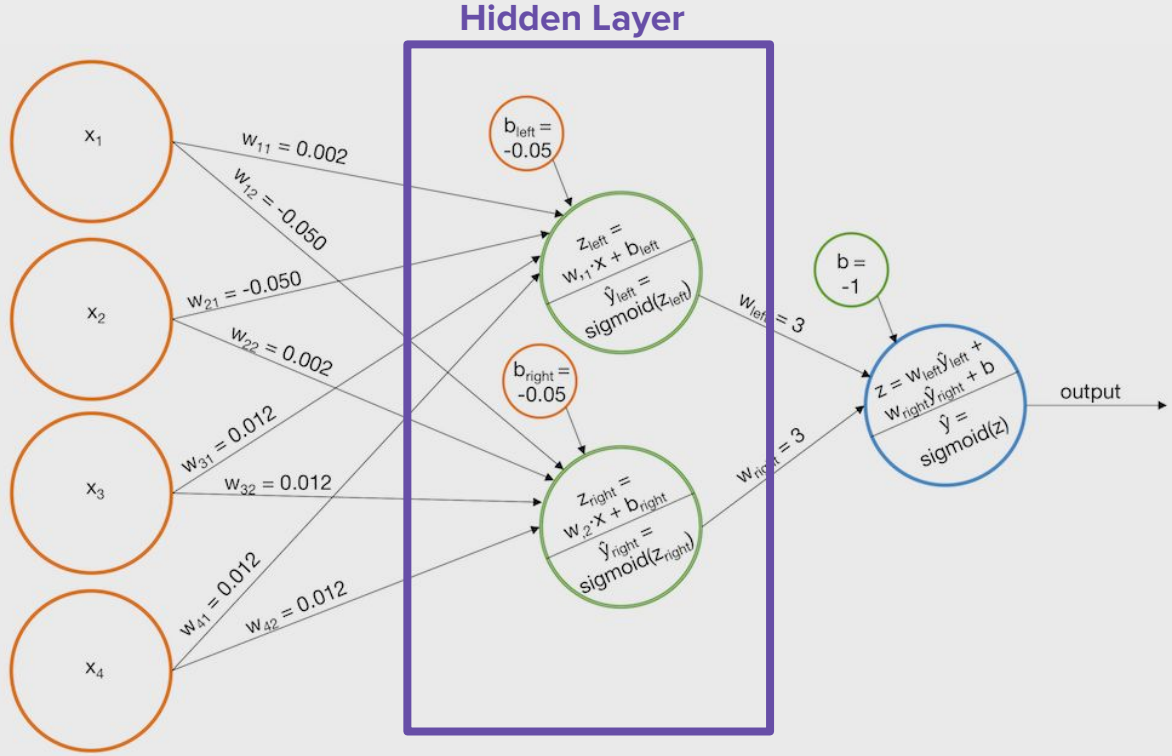
$$\text{where } a(f) = \begin{cases} 1 & \text{if } f \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

NN with bias in Nutshell



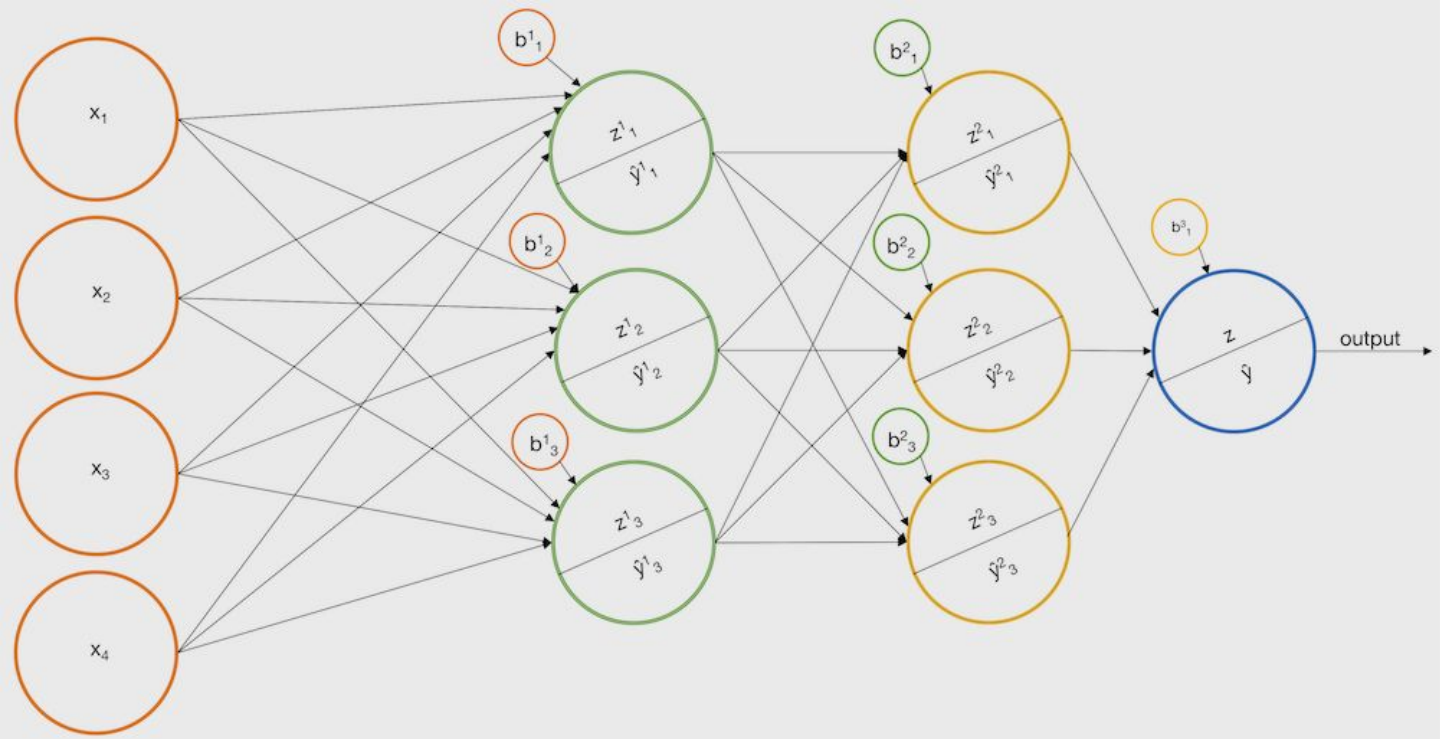
Source: http://blog.kaggle.com/2017/11/27/introduction-to-neural-networks/

NN in bigger Nutshell
“Multi Layer Perceptron (MLP)”



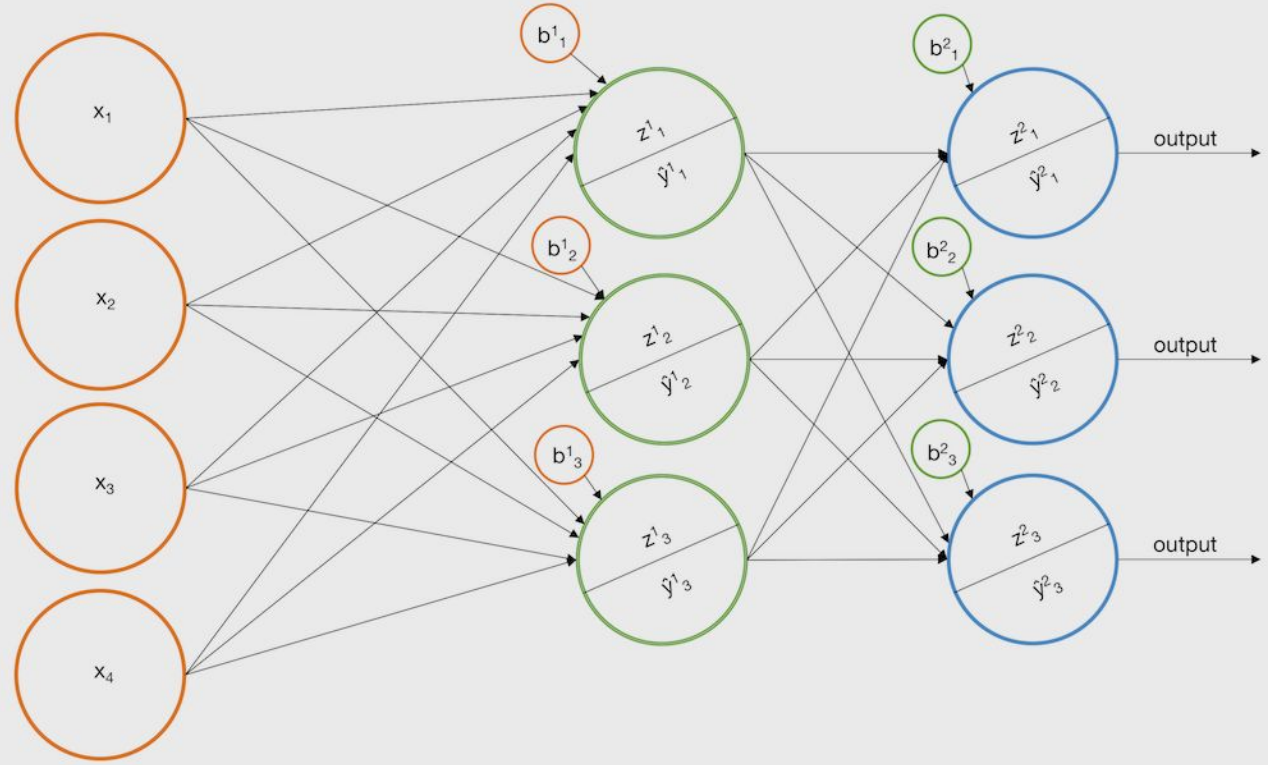
Source: http://blog.kaggle.com/2017/11/27/introduction-to-neural-netwoks/

NN in bigger Nutshell
“Multi Layer Perceptron (MLP)”



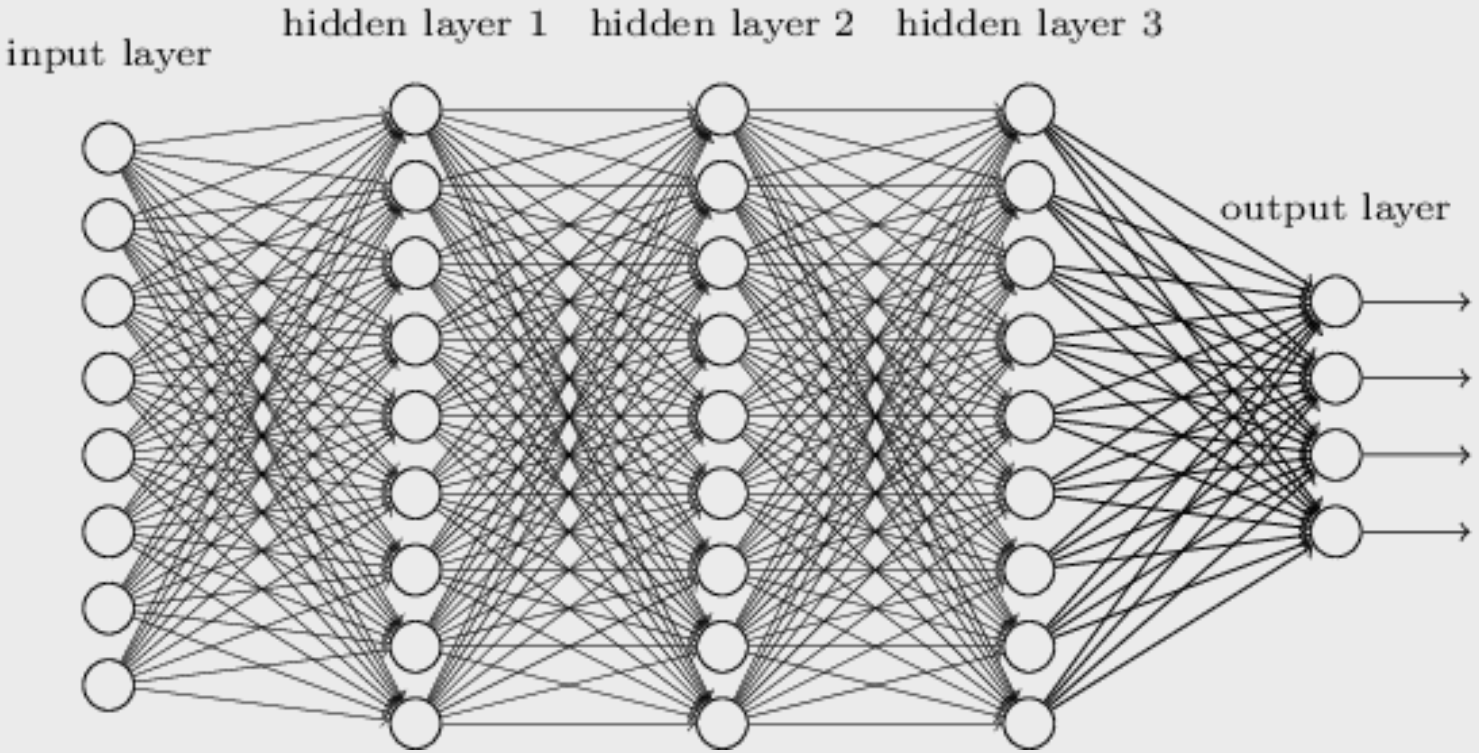
Source: http://blog.kaggle.com/2017/11/27/introduction-to-neural-netwoks/

MLP Multi Class



Source: http://blog.kaggle.com/2017/11/27/introduction-to-neural-netwoks/

Go Deep



Source: https://www.mathworks.com/matlabcentral/fileexchange/64247-simple-neural-network

Optimization

“Gradient Descent”

$$\theta = \theta - \eta \cdot \nabla J(\theta)$$

is the formula of the parameter updates, where **‘η’** is the learning rate ,**’∇J(θ)’** is the **Gradient** of **Loss function**-**J(θ)** w.r.t parameters-‘θ’

We use this to optimize weight.

Lost Function

MAE

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

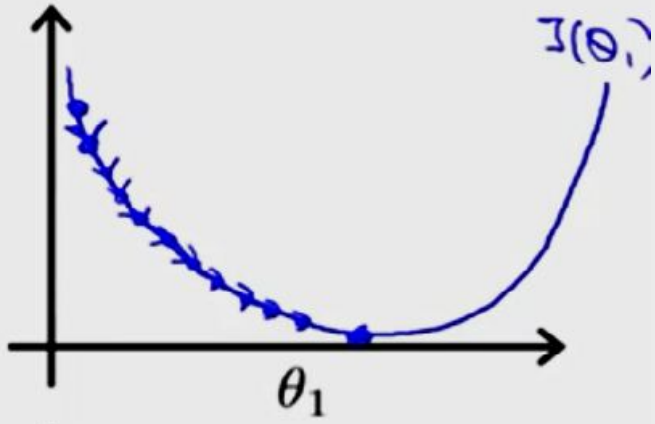
Cross Entropy

$$\mathcal{L}(X,Y) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \ln a(x^{(i)}) + \left(1 - y^{(i)}\right) \ln\left(1 - a(x^{(i)})\right)$$

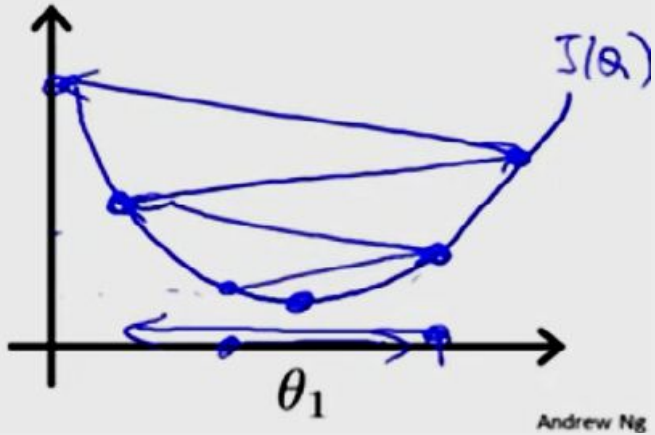
Learning Rate

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



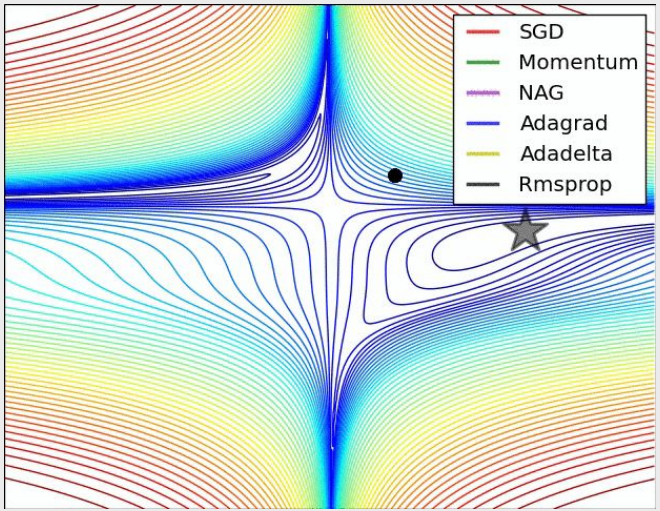
If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Andrew Ng

More Gradient Optimization

- Sochastic Gradient Descent (SGD)
- Momentum
- Nesterov accelerated gradient
- Adagrad
- AdaDelta
- Adam



Feed Forward algorithm

Step 0 - Initialize weights (w0, w1, …, wn), m = 0, learning rate , and threshold t

Step 1 – Do m = m + 1

Step 2 – Select input Xm

Step 3 – Calculate output $o = a(f) \quad f(w_i, X_i) = \sum_i w_i X_i - t$

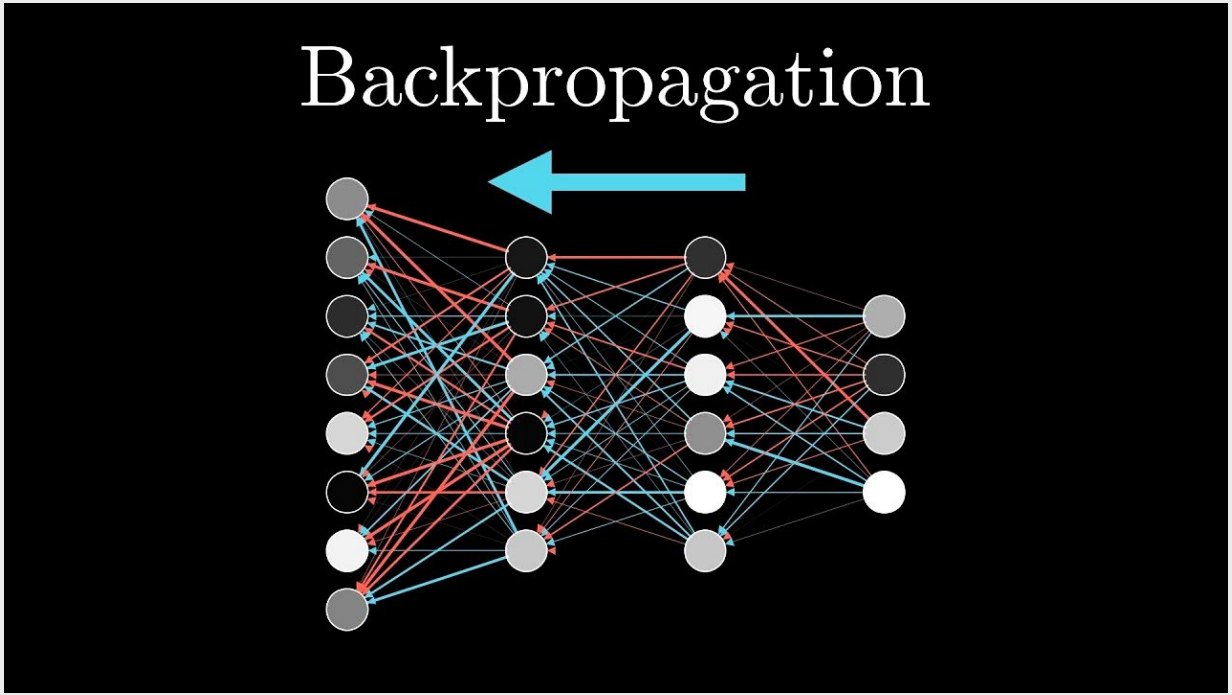
Step 3 – Calculate delta = *y* – *o*

Step 4 – Update weight $w_{(new)} = w_{(old)} + \eta \sum \delta_i X_i$

Step 5 – Repeat until w convergent

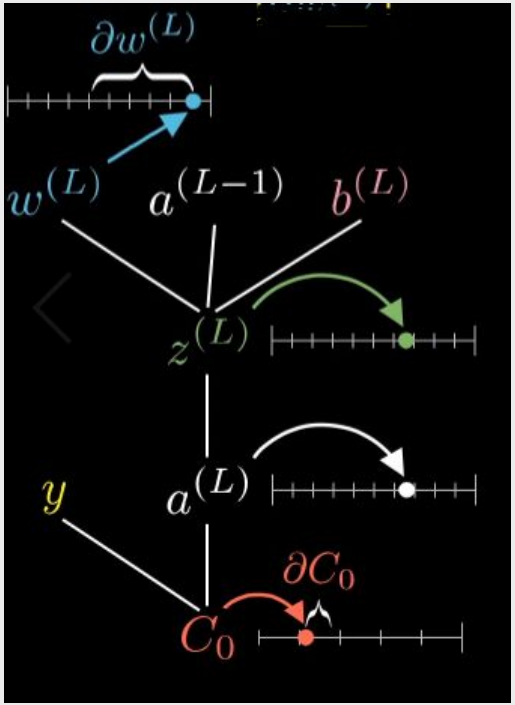
Step 6 – Return w

Backpropagation



Source: https://i.ytimg.com/vi/Ilg3gGewQ5U/maxresdefault.jpg

Backpropagation



$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C0}{\partial a^{(L)}}$$

$$\frac{\partial C_0}{\partial b^{(L)}} = \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C0}{\partial a^{(L)}}$$

$$\frac{\partial C_0}{\partial a^{(L-1)}} = \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C0}{\partial a^{(L)}}$$

Source: https://www.youtube.com/watch?v=tIeHLnjs5U8

Let's code

Learning Factors

- Initial Weights
- Learning Rate
- Cost Functions
- Update Rules
- Training Data and Generalization
- Number of Layers
- Number of Hidden Nodes

indra@nodeflux.io
rizqi.okta@nodeflux.io