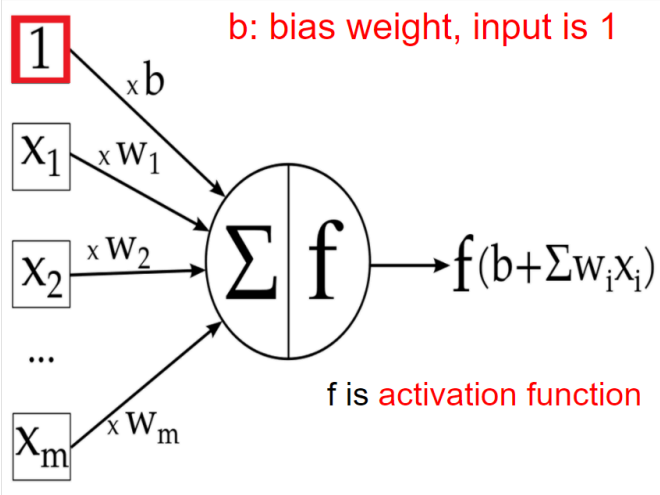
Notes| Neural Networks

Neural networks resemble the human brain in the following two ways:

1. Neural networks possess the ability to acquire and store knowledge.

2. It acquires knowledge through learning.

3. It stores knowledge within inter-neuron connection strengths known as synaptic weights.



The number of neurons in the input and output layer depends on the training set.

Example : 2 input OR gate

2 inputs, 1 output

Ex. Identifying 5x7 representation of digits 0-9

45 inputs, 10 outputs

Plasticity of brain/neuroplasticity—the ability to morph and change.

✔ *New synaptic connections are made, old ones go away, and existing connections become stronger or weaker, based on experience.*

✔ The idea of plasticity is the key principle in the training of ANNs — iteratively modifying the weights of the networks.

✔ The field of meta-learning (*learning how to learn)has* expanded the use of plasticity in ANNs beyond parameters and is applied to modify hyperparameters or even a whole model.

Notes| Neural Networks

EPANNs (Evolved Plastic ANNs)

- ✔ most advanced AI and machine learning systems rely on
- ✔ Evolution
- ✔ Learning

Meta-Learning Evolutionary Artificial Neural Networks (MLEANN)

evolution searches for *initial weights, neural architectures, and transfer functions, and finally optimizes existing learning rules*.

ANNs works in two phases:

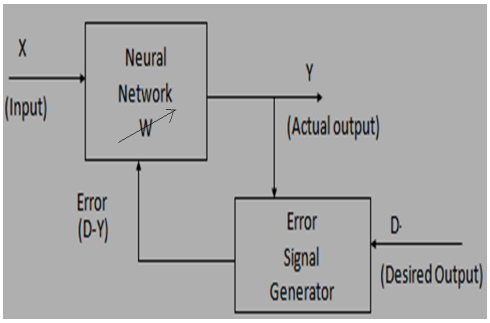
– Learning phase – the adaptation of ANN's internal parameters *(weights and biases).*

– Evaluation phase (recall) – use/test what was learned.

Classification of Learning

- Supervised learning
- Unsupervised learning
- Reinforcement learning
- others

Supervised Learning



The learning algorithm receives a set of inputs and corresponding correct outputs/labels.
The algorithm learns by comparing its actual output with the "correct/desired/true" output, then, it modifies the network parameters.
The goal is to learn a general rule that maps inputs to outputs by minimization of error.

Notes| Neural Networks

Unsupervised Learning

✔No label or target given for the examples.
✔Uncovers patterns from the *unclassified and unlabeled data*
✔ One common task is to group *similar* examples together called clustering.
✔ In unsupervised learning, there is no feedback. The learning algorithm will group the data into certain groups based on the common aspects they share.
✔The network must discover patterns, features for the input data and change parameters
✔This process is called Self-organizing

Semi-Supervised /Hybrid Learning

Supervised: All labeled data
Unsupervised: All un-labeled data

Semi-supervised: some labeled data + lots of unlabeled data

This method is particularly useful when extracting relevant features from the data is difficult, and labeling examples is a time-intensive task for experts.

✔For medical images like CT scans or MRIs, a trained radiologist can go through and label a small subset of scans for tumors or diseases.
✔It would be too time-consuming and costly to manually label all the scans.

Self-training is the simplest form of semi-supervised classification.

Train a classifier on the small amount of labeled data, and then use the classifier to make predictions on the unlabeled data.

Notes| Neural Networks

Reinforcement Learning

- The ANN makes a decision by observing its environment.
- The machines are exposed to an environment where they can learn autonomously by hit and trial /experiential methods.

Example of RL: learning motor skills like riding a bike.

You try and see what makes you fall over and what keeps you upright.
An agent interacting with the world makes observations, takes action, and is rewarded or punished.
It should learn to choose actions in such a way as to maximize rewards.

The New Dawn of AI: Federated Learning(FL)

A new way of training a machine learning *using distributed data that is not centralized in a server*

✔FL was introduced by Google in 2016
✔It's a new ML paradigm that allows us to build ML models from private data, without sharing such data with a data center.
✔Equipped with AI chips and significant computing power, starting with Samsung S9, or Apple X series, many of the ML models will be able to run locally on these mobile devices.

In the next few years, model building and computation on the edge, based on Federated Learning and secured with *Homomorphic Encryption* will make significant progress.

Homomorphic encryption is a form of encryption allowing one to perform calculations on encrypted data without decrypting it first.

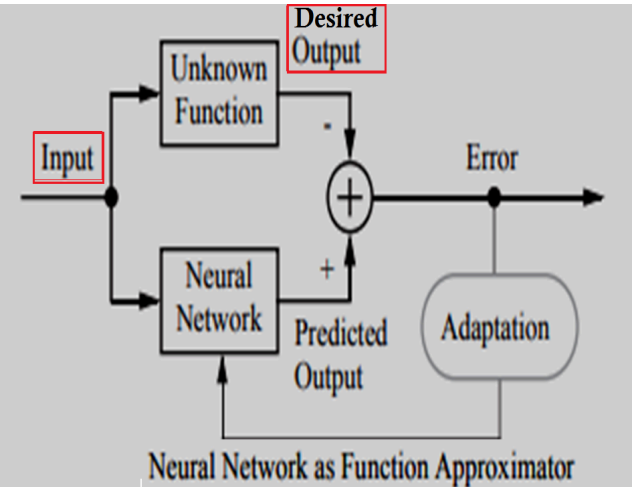
Notes| Neural Networks

Application Fields of ANNs

ANNs are used in a variety of domains like Medical, Forecasting, Industrial Measurement, and Control, Image processing, Finance & Banking, Voice recognition/synthesis, Targeted marketing, Intelligent searching, etc

Applications can be classified into the following BROAD categories:

Function Approximation/Predictive modeling:



Predictive modeling is a process that uses data and statistics to predict outcomes.
Predictive modeling for learning a mapping function from inputs to outputs called function approximation.

Neural nets are highly capable of approximating almost any function in arbitrary dimensionality.

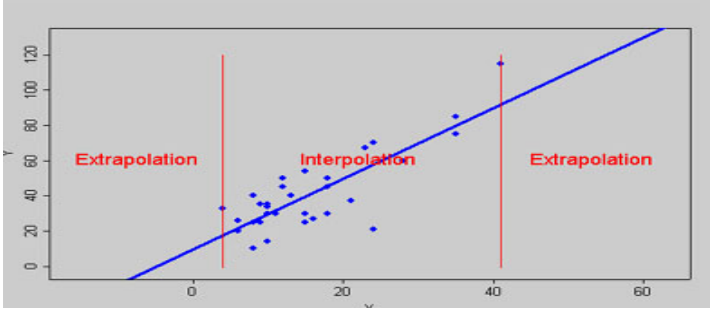
Given input/output pairs $(x_1,d_1),(x_2,d_2), ...(x_n,d_n)$ approximate the function g(.) such that d = g(x).

FUNCTION FITTING/APPROXIMATION AND CURVE FITTING.

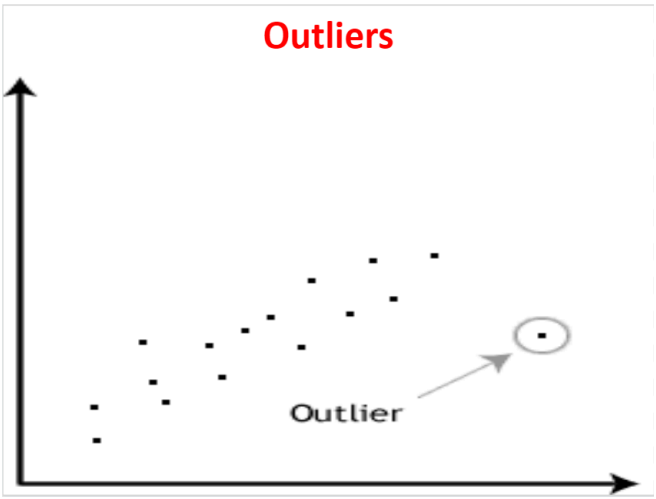
Function fitting: the process of training a neural network on a set of inputs to produce an associated set of target outputs.

Once the neural network has fit the data, it forms a generalization of the input-output relationship and can be used to generate outputs for inputs it was not trained on.

Notes| Neural Networks



✔More examples better would be the mapping function.
✔The less noise in observations, the better the approximation of the mapping function.
✔Estimation/Prediction of values between training patterns is called interpolation.
✔Interpolation is useful where the data surrounding the missing data is available and its trend, seasonality, and longer-term cycles are known.
✔Estimation/prediction of values outside training patterns is called Extrapolation.
✔In extrapolation, the underlying assumption is *that our observed trend continues for values of x outside the range we used to form our model, which may not be the case*



- An outlier is an observation that appears to deviate markedly from other observations
- Outliers may be due to random variation or may indicate something scientifically interesting.
- If it can be determined that an outlying point is in fact erroneous, then the outlying value should be deleted from the analysis (or corrected if possible).

✔ In any event, do not simply delete the outlying observation.

Notes| Neural Networks

Black Swan Event is an incident that occurs randomly and unexpectedly

- ✔ If you saw a thousand swans and they were all white, you might logically conclude that all swans are white.
- ✔ In other words, just because you did not see a black swan, it does not mean they do not exist

Nassim Nicholas Taleb is a Lebanese-American essayist, scholar, statistician, and former trader and risk analyst who works on problems of randomness, probability, and uncertainty.

Classification Predictive vs Regression Predictive Modeling

Classification is about predicting a label and Regression is about predicting a quantity.

In classification, predictive problems target variable can take on only a small number of discrete values. The algorithm must predict the most probable category, class, or label for new examples.

Examples: Binary/Two class problems

Y {0,1}	1 or Positive class	0 or Negative class
Email	Spam	Not Spam
Tumor	Malignant	Benign
Transaction	Fraudulent	Not Fraudulent
Stock Price	Rise	Fall
News-article	Politics	Leisure

Notes| Neural Networks

Multiclass Classification:

House	Tree	Beach	Cloud	Mountain	Animal
Yes	Yes	no	Yes	no	no

✔ Put a tennis ball into the Green, Orange, or White ball bin (color)

✔ Decide if an email is an advertisement, newsletter, phishing, hack, or personal.

✔(Optical Recognition) Classify a scanned character into digit (0..9)

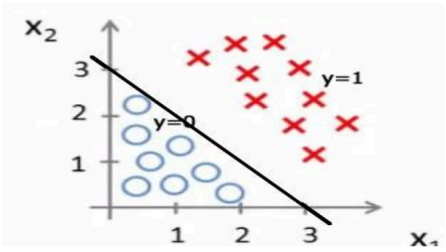
✔*Multi-class classification makes the assumption that each sample is assigned to one and only one label.*

MULTI-LABEL PROBLEM

✔Classification task of labeling each sample with x labels from n_classes, where x can be 0[no class] to n_classes inclusive[all classes]

✔Example: Prediction of the topics relevant to a text document/video to one of ‘religion’, ‘politics’, ‘finance’ or ‘education’, several of the topic classes or all of the topic classes.

Binary Classification :
Linear Decision Boundary

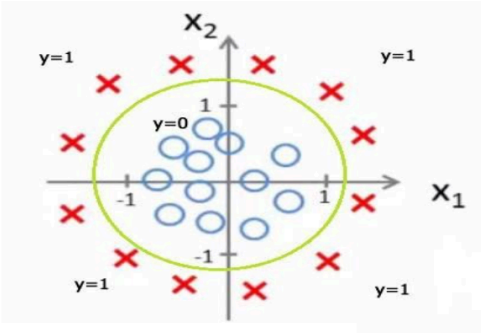


For y = 1, Equation of line would be $x_1 + x_2 >= 3$

For y = 0, Equation of line would be $x_1 + x_2 < 3$

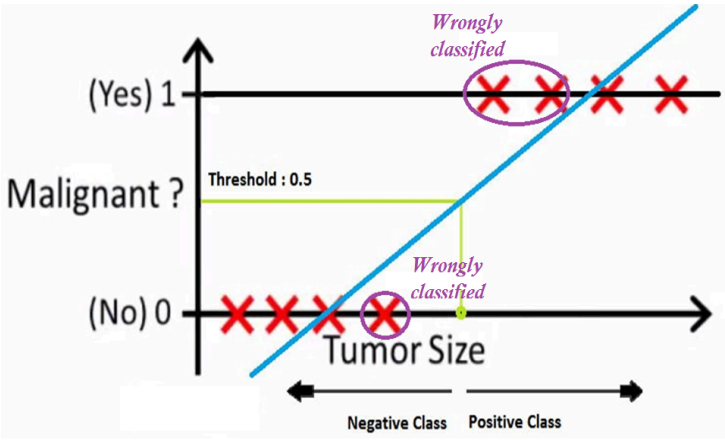
Notes| Neural Networks

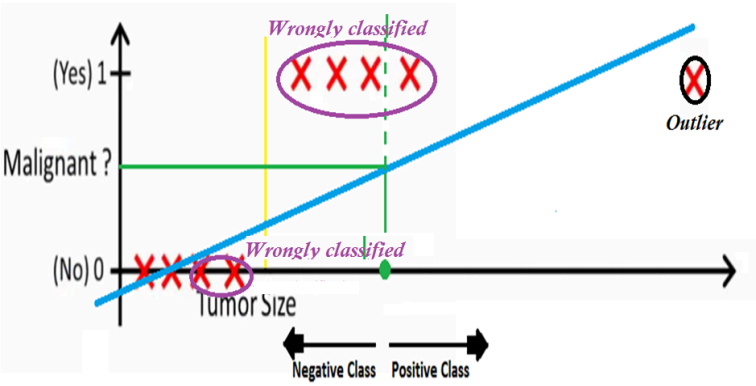
Non-linear Decision Boundary



For y=1, equation would be $x_1^2+x_2^2 >= 1$
For y=0, equation would be $x_1^2+x_2^2 < 1$

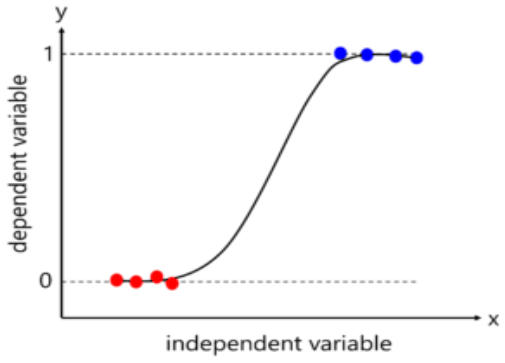
Binary Classification Problem:
Linear Decision Boundary



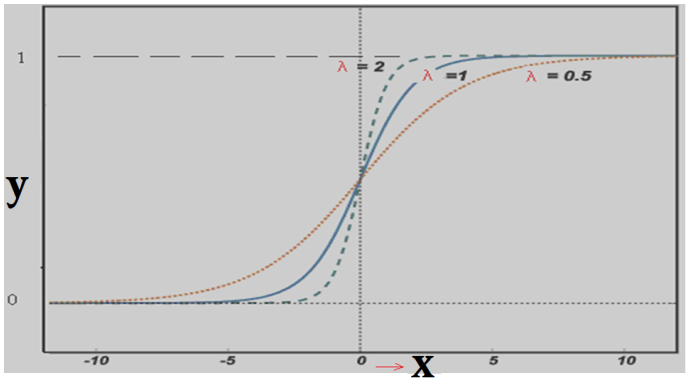


Just a single outlier is disturbing the whole linear regression predictions.
Let's use logistic regression

Classification Problem also called
[Logistic regression]- uses a Sigmoid function,
which takes any real value between 0 and 1



Notes| Neural Networks



$$y = \frac{1}{1 + \exp(-\lambda . x)}$$

$$\lambda - Slope$$

Perceptron, Logistic Regression, and SVM(Support Vector Machines) designed for binary classification *do not support multi-classification tasks*

Two approaches for using binary classification algorithms for multi-class classification by changing the multi-class problem to binary class:

1. One-vs-Rest / One vs All
2. One-vs-One

One-Vs-Rest for Multi-Class Classification

A multi-class classification problem with examples for '*red*,' '*blue*,' and '*green*' class.

Divide into three binary classification datasets :
Binary Classification Problem 1:
red vs [blue, green]

Binary Classification Problem 2:
blue vs [red, green]

Binary Classification Problem 3:
green vs [red, blue]

One-Vs-One for Multi-Class Classification

Notes| Neural Networks

Given: a multi-class classification problem with four classes: *'red,' 'blue,' 'green'* and *'yellow.'*

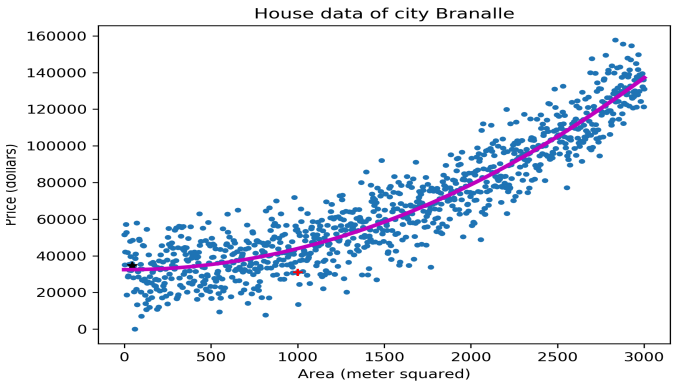
Divide into six binary classification datasets as follows:

Binary Classification Problem 1: red vs. blue
Binary Classification Problem 2: red vs. green
Binary Classification Problem 3: red vs. yellow
Binary Classification Problem 4: blue vs. green
Binary Classification Problem 5: blue vs. yellow
Binary Classification Problem 6: green vs. yellow

REGRESSION Predictive Model

✔When the target variable to predict is continuous, the learning problem is called a REGRESSION problem.

Examples:
✔*"What is the sale price for a house based on its features?"*
✔*"How many products will a customer order on this website?"*
✔*"What is the fat percentage for a BMI of 28?"*



A problem with multiple input variables is often called a multivariate regression problem.

✔A regression problem where input variables are ordered by time is called a time series forecasting problem.

✔Ex. Forecast population in 2030 based on historical data, forecasting stock prices, rainfall, climate

Notes| Neural Networks

Branches of Business Analytics (BA)
Descriptive/Diagnostic/predictive/prescriptive

Descriptive Analytics tells you what happened in the past.

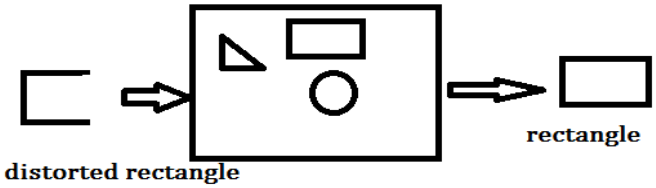
Diagnostic Analytics helps you understand why something happened in the past.

Predictive Analytics predicts what is most likely to happen in the future based on historical data.

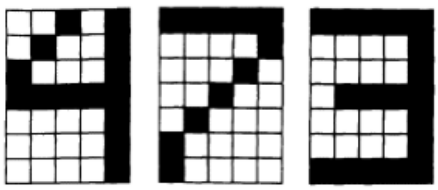
Prescriptive Analytics recommends actions you can take to affect those outcomes.

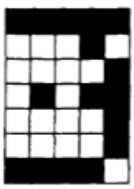
ASSOCIATION:

✔ It is the task of associating/mapping input patterns to target patterns (attractors).
✔Learning consists of encoding the desired patterns as a weight matrix.
✔Network stores different patterns and can produce one of the stored patterns by matching them with the given input pattern.



✔Storing information is called ENCODING
✔A network stores data as a result of LEARNING
✔A degraded input pattern serves as a CUE for retrieval of its original


(For training)

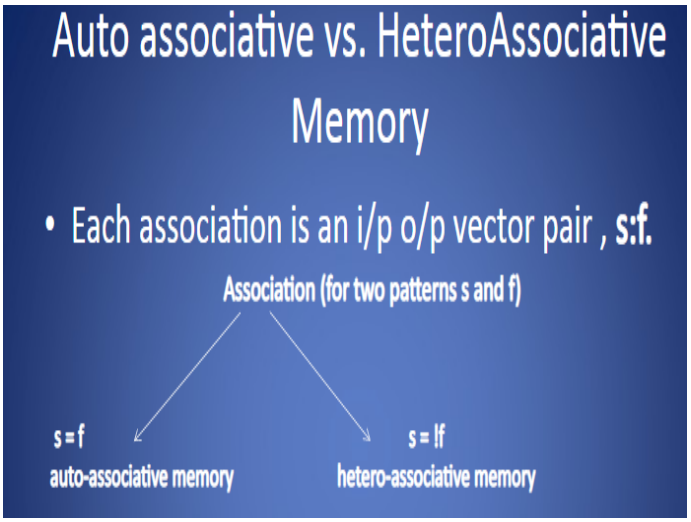

(New)

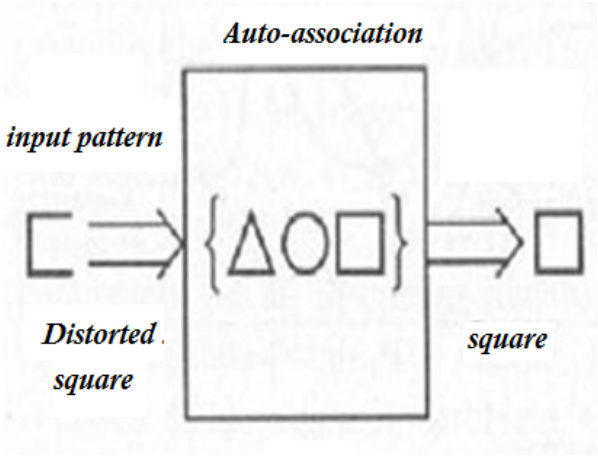
Three class patterns stored while training and a new one (corrupted) is to be associated with one from the class patterns that are most similar.

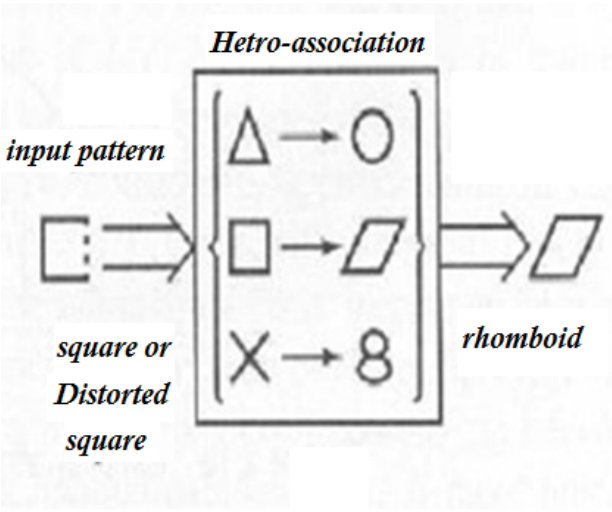
Notes| Neural Networks

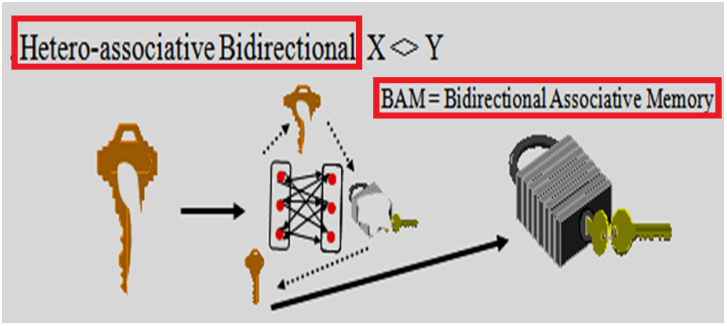
✔An associative memory can be considered as a memory unit whose stored data can be identified for access by the content of the input data itself rather than by an address of memory location.

✔Associative memory is often referred to as Content Addressable Memory (CAM).





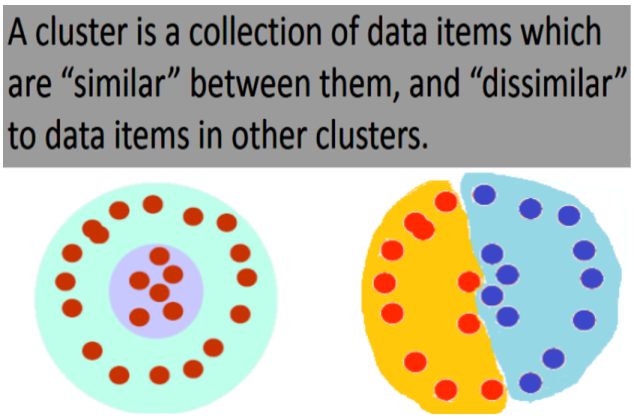


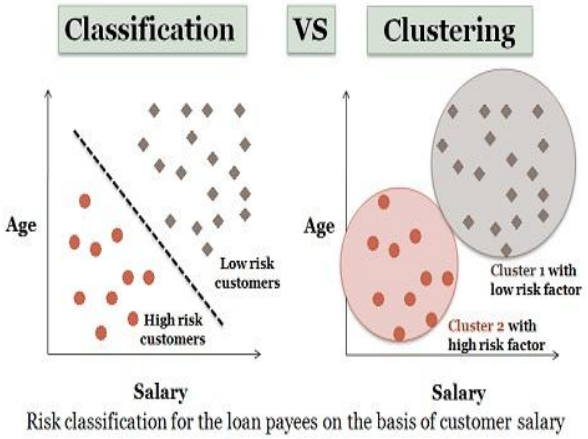


Notes| Neural Networks

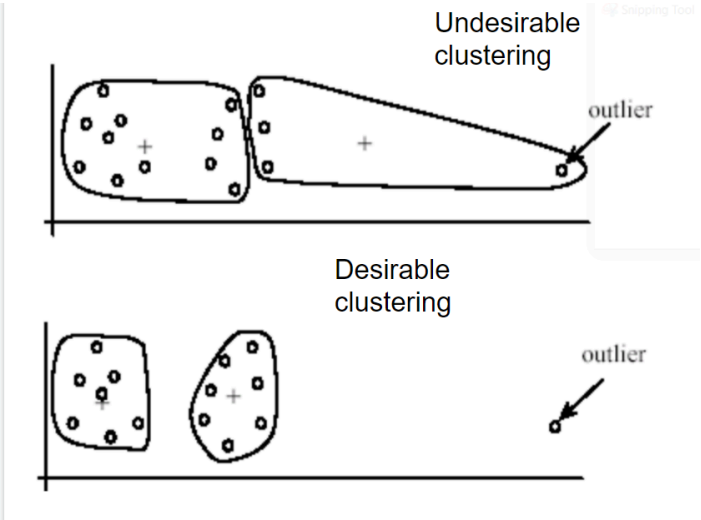
CLUSTERING

✔Clustering can be said as the identification of similar classes of objects.





Risk classification for the loan payees on the basis of customer salary



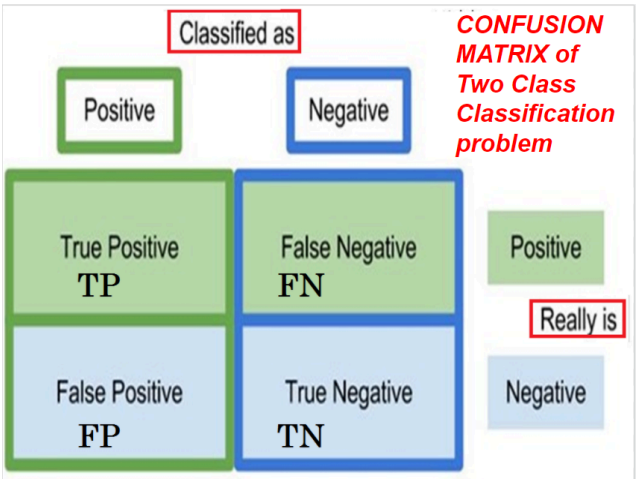
PERFORMANCE MEASURE OF CLASSIFICATION/CLASSIFIER

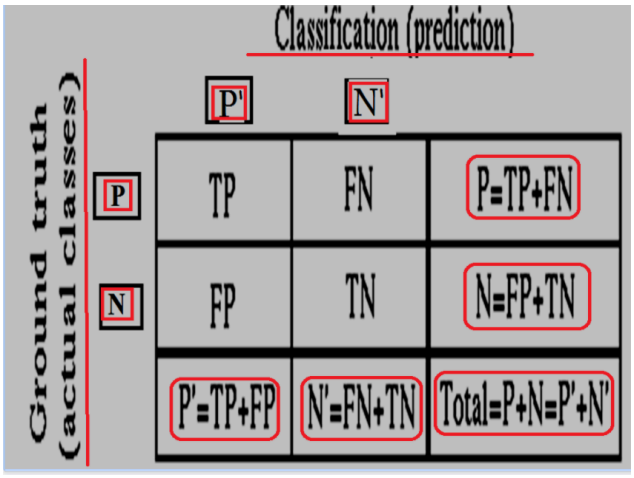
✔The classification model produces continuous or discrete outputs.

✔The discrete output from a classification model represents the predicted discrete class label of the unknown/test sample

✔ The continuous output represents the estimation of the sample's class *membership probability*

Notes| Neural Networks





Ex. Cricket Umpire for identifying
OUT(POSITIVE), NOT-OUT (NEGATIVE)
Umpire gives a Batsman NOT OUT when he is NOT-OUT.
True Negative
Umpire gives a Batsman OUT when he is OUT.
True Positive
Umpire gives a Batsman OUT when he is NOT- OUT.
False Positive

Ex. Diagnosing for disease (sickness)
Sick people correctly identified as sick
True positive
Healthy people incorrectly identified as sick
False-positive
Healthy people correctly identified as healthy
True negative
Sick people incorrectly identified as healthy
False-negative

Notes| Neural Networks

TYPE I and TYPE II ERRORS

Ex: smog prediction system
A FALSE POSITIVE TYPE I ERROR
The public would take precautionary measures when they didn't need to.
A FALSE NEGATIVE TYPE II ERROR
not warning about a smog day when in fact it is a high smog day, leading to health issues in the public that is unable to take precautions.
Which type I or II is risky?
Type II is riskier
FALSE NEGATIVE Type II error

Ex.: Testing whether the Construction Model of a bridge is correct.
Type I error (FP)
Predicting that the model is correct when it is not.
Type II error (FN): Predicting that a model is not correct when it is correct.
Which error is riskier?
Type I error (FP) is a huge risk.
It could mean building a bridge that is faulty and can lead to a bridge collapse.
Type II error (FN) is less serious
as we can go through more models and still find the correct one.

Biometrics for surveillance/security for identifying authorized persons

✔If an Authorized person is recognized correctly, it counts to the number of?
True Positives (TP)

✔if a test image of an Authorized person is rejected and labeled as impostor due to the classifier's limited ability to recognize correctly, the count adds to the number of?
False Negatives (FN).

Notes| Neural Networks

✔If a test image of a person from the impostor database is recognized correctly as an impostor, then it counts too?

True Negatives (TN)

✔if an impostor is recognized as an authorized person, it counts?

False Positives (FP) – a risk for the high-security areas.

Term	Meaning
True Positive (TP)	Correct Identification
True Negative (TN)	Correct Rejection
False Negative (FN)	Incorrect Rejection
False Positive (FP)	Incorrect Identification

A classifier is said to be efficient when not only does it recognize the authorized test image correctly, but also rejects the impostors (unauthorized persons) in the test samples.

A perfect classifier will have no entries for
FP and FN
(i.e., number of FP = number of FN = 0).

Performance Measurement TERMS in classification

Sensitivity/Recall/TPR (True Positive Rate)

How good a test is at detecting positives out of a total number of really positive samples?

$$Sensitivity(TPR = \mathrm{Re}\,call) = \frac{TP}{TAP} = \frac{TP}{TP + FN}; TAP = TotalActualPositive$$

Specificity/TNR(True Negative Rate)

When it's actually NO, how often does it predict NO?

Specificity = TN/Actual NO=TN/N

$$Specificity(TNR) = \frac{TN}{TAN} = \frac{TN}{TN + FP}; TAN = Total\ Actually\ Negative$$

FP is when you receive a positive result for a negative result- "false alarm"

High Specificity is avoiding False Alarms FP=0, Specificity is 100%

Notes| Neural Networks

- ✔If the analysis is being carried out for a financial lending firm, which is interested in determining whether or not a loan applicant income is greater than Rs 50,000/pm
- ✔And a person having income greater than Rs. 50,000/pm, classified as positive would be TP
- ✔If the purpose of the test is to detect disease, then what is TP?
- ✔Sick Person identified as sick is TP
- ✔If the purpose is to catch imposters, then what is TP?
- ✔An imposter identified as an imposter would be TP
- ✔If the purpose is to identify an authorized person, then what is TP?
- ✔An authorized person identified as authorized would be considered TP

SENSITIVITY/*RECALL/TPR*, measures, *True Positive Rate* (TPR)=TP/TAP
TAP: Total actually positive= TP+FN

SPECIFICITY measures, *True Negative Rate*(TNR)=TN/TAN
TAN: Total Actually negative = TN+FP.

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

When it's actually NO, how often does it predict YES?

False Positive rate (FPR) / False Acceptance rate(FAR) or Fallout
complements Specificity (TNR)

$$FPR = 1 - TNR = \frac{FP}{FP + TN} = \frac{FP}{N}$$

$$FNR = 1 - TPR = \frac{FN}{FN + TP} = \frac{FN}{P}$$

FPR = 1 - TNR = FP/ TAN= FP/(FP+TN)

False-negative rate (FNR) or MISS RATE is the proportion of positive samples that were incorrectly classified.
It complements Sensitivity(TPR)
FNR = 1 - TPR = FN/ TAP = FN/ (FN+TP)

- Both FPR and FNR are not sensitive to changes in data distributions and hence both metrics can be used with imbalanced data

Notes| Neural Networks

RECISION/PPV (Positive Predictive Value)

✔The ratio of TP to all those who Tested/Classified As Positive

$$\Pr ecision(PPV) = \frac{TP}{TPP} = \frac{TP}{TP + FP}$$

$$TPP = Total\ \Pr edicted\ Positive$$

✔While Sensitivity/Recall/TPR is the Ratio of TP to those who are TRULY POSITIVE

✔Low precision indicates a high number of false positives.

✔Sensitivity= TP/Actual yes

✔Low recall indicates a high number of false negatives.

Ex: In cancer test of 100 people, 5 people actually have cancer. If the model predicts every case as cancer.
Calculate Recall, Specificity, and Precision
Find TP, FP, TN, FN

The sick person identified as sick=?
TP=5,
The healthy person identified as sick=?
FP=95,
The healthy person identified as healthy=?
TN = 0,
The sick person identified as healthy=?
FN = 0,
TP = 5, FP = 95, FN = 0, TN = 0

Sensitivity= Recall Rate = TP/ (TP+FN)
Since we predicted 5 cancer cases correctly,

The sensitivity/Recall of such a model is 100%.

What about Specificity ?? Are all negative cases identified?
NO

Notes| Neural Networks

The specificity of such a model is 0%.

The precision of such a model is ??
Out of tested positive, how many are (+) 5%

When to use Precision and When to use Recall?

✔ *Recall = TP/ (TP+FN)*
✔ *Precision = TP/ (TP+FP)*

Recall gives us information about a classifier's performance with respect to false negatives (how many did we miss), while

Precision gives us information about its performance with respect to false positives(how many did we catch).

✔ If the focus is on minimizing False Negatives, Recall should be close to 100% without precision being too bad

✔ If the focus is on minimizing False positives, Precision should be close to 100%

The denominator of precision is TP + FP. It does not consider FN.
If you predict that ONLY one email is spam[because of very high decision threshold), and you get that right, then you're precision would be 1.

Why?
Because there are no false positives(FP=0). However, you will be missing a lot of predictions (FN).

For rare cancer data modeling, What is more, important Recall or Precision?

Minimizing FN means (increase Recall),i.e. *Avoid diagnosing a person as healthy when he/she is actually sick.*

Notes| Neural Networks

No treatment. if you diagnose a sick person as healthy, they might not get additional tests done.

Minimizing FP means(increase Precision)

avoid diagnosing a person as sick when he/she is actually healthy
If you diagnose a healthy person as sick, additional tests can help towards the right diagnosis.
Which figure of merit is more important, is to be kept high?
The recall is a better measure than precision in disease detection

YouTube recommendations,

What is important Recall or Precision?
Which is to be minimized FP or FN
FP: irrelevant recommendations
FN: relevant being missed out

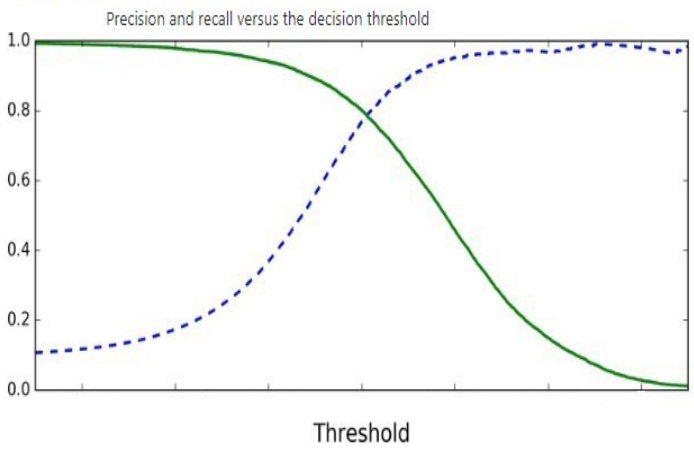
FP is more serious, should be minimized,
Precision is better to measure

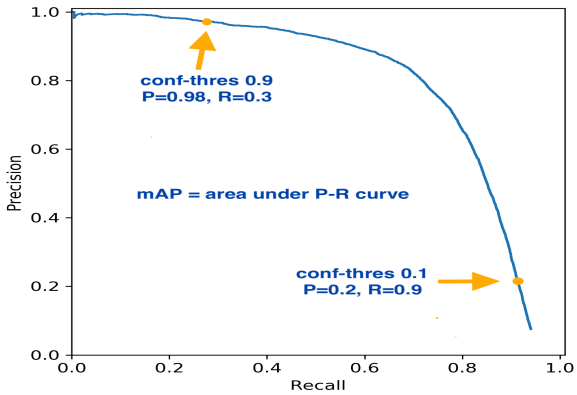
In spam email detection

✔ *What is more important Recall or Precision?*
✔*Should FP (imp mail considered as spam)*
✔*or FN (spam be taken as non-spam) be avoided?*
✔ it is OK if a spam email (positive case) is left undetected and doesn't go to the spam folder *but*, if an email is good (negative), then it must not go to the spam folder.

✔Precision is more important. (If the model predicts something positive (i.e. spam), it better be spam, else, you may miss important emails).

Notes| Neural Networks





mAP: Mean Average Precision

The F-β measure is a combined representation of Precision (PPV) and Sensitivity/Recall/TPR

$$F-\beta = \frac{(1+\beta^2) * \mathrm{Pr}\,ecision * \mathrm{Re}\,call}{\beta^2 * \mathrm{Pr}\,ecision + \mathrm{Re}\,call}$$

3 common values for the beta parameter

F0.5-Measure (β=0.5):
More weight on precision, less weight on recall.

F1-Measure (β =1.0):
Balance the weight on precision and recall.

F2-Measure (β =2.0):
Less weight on precision, more weight on recall

Notes| Neural Networks

$$F1 = 2 \times \frac{\Pr ecision \times \operatorname{Re} call}{\Pr ecision + \operatorname{Re} call} \ldots\ldots\ldots Harmonic \;\; mean$$

A classifier with a precision of 1.0 and a recall of 0.0 has a simple average of 0.5 but an F1 score of 0.

Example: Fingerprint recognition has a Precision of 1.0 and Recall of 0.2.

Intuitively, the total performance of the system should be very low because the system covers correctly only 20% of the registered fingerprints, which is almost useless.

Arithmetic Mean of 1.0 and 0.2 is 0.6

Harmonic Mean: 2x(1x0.2)/ (1+0.2) = 1/3= 0.333 -more reasonable score

Classification Accuracy
is the answer to the question - Overall, how often is the classifier correct?

Accuracy = Correct predictions / Total predictions

Write the Expression $$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Misclassification Rate/Error Rate

Overall, how often is it wrong?

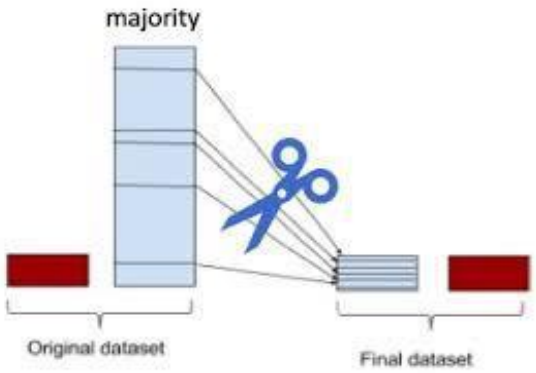
Error = Incorrect predictions/Total predictions

Write the expression $$Error = 1 - Accuracy = \frac{FP + FN}{TP + FP + TN + FN}$$

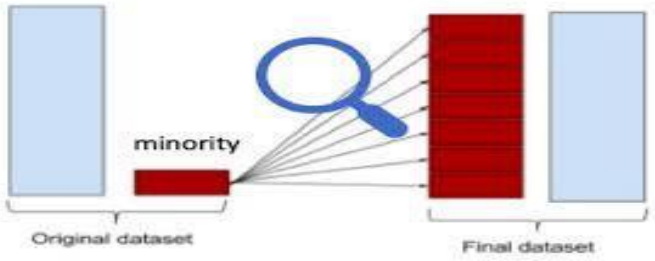
Techniques for unbalanced data :

Notes| Neural Networks

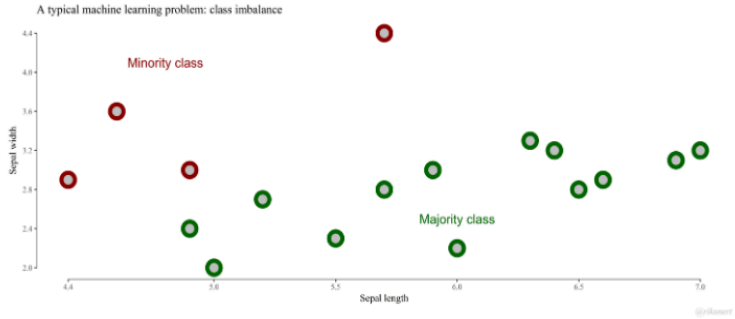
Under-sampling: Remove samples from over-represented classes; use this if you have a huge dataset



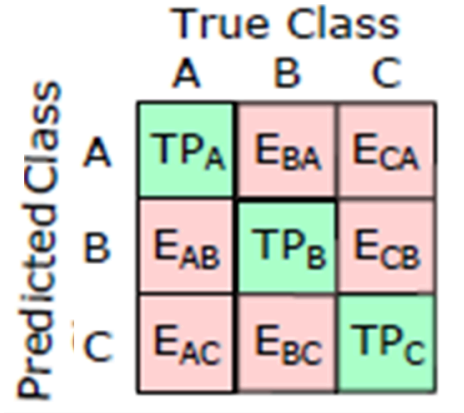
Over-sampling: Add more samples from under-represented classes; use this if you have a small dataset



SMOTE: Synthetic Minority Over-sampling Technique
designed to generate new samples that are coherent with the minor class distribution.



Confusion matrix for a multi-class classification problem with three classes (A, B, and C), interested in the performance of all classes



TP_A: A correctly classified as A
TP_B: B correctly classified as B
TP_C: C correctly classified as C

Notes| Neural Networks

E_{AB}: True class A, incorrectly classified as class B

E_{CB}: True class C, incorrectly classified as B

E_{CA}: True class C, incorrectly classified as A

FN_A = A NOT Classified as A , but classified as B &C
$= E_{AB} + E_{AC}$

FN_B = B NOT Classified as B, but classified as A &C
$= E_{BA} + E_{BC}$

FN_C = C NOT Classified as C, but classified as A &B
$= E_{CA} + E_{CB}$

FP_A = B &C classified as A
$= E_{BA} + E_{CA}$

FP_B = A &C classified as B
$= E_{AB} + E_{CB}$

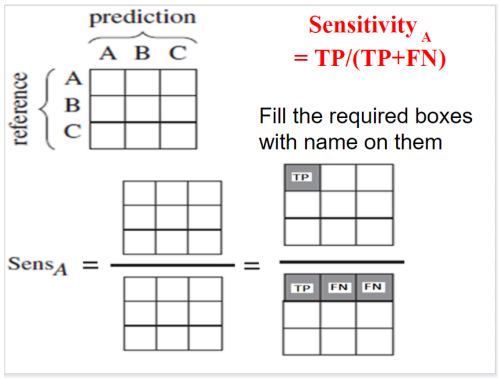
FP_C = A &B classified as C
$= E_{AC} + E_{BC}$

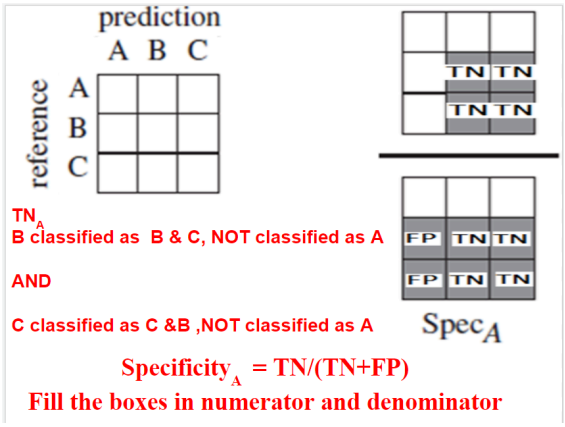
TN_A : B & C classified NOT classified as A,
i.e. B classified as B and C, and C classified as C & B
$TN_A = TP_B + TP_C + E_{BC} + E_{CB}$

TN_B : A & C classified NOT classified as B
i.e. A classified as A & C, and C classified as C & A
$TN_B = TP_A + TP_C + E_{AC} + E_{CA}$

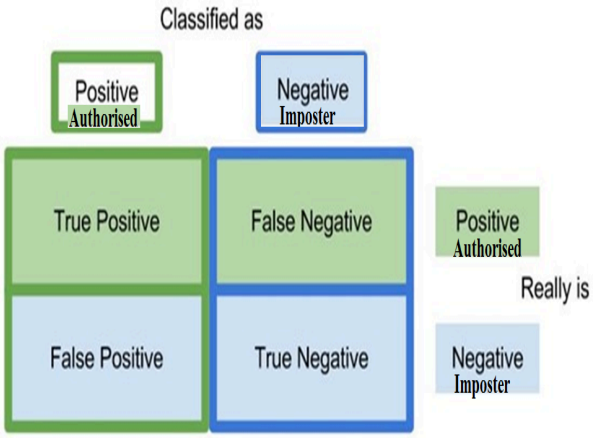
TN_C : A & B NOT classified as C
i.e. A classified as A & B, and B classified as B &A
$TN_C = TP_A + TP_B + E_{AB} + E_{BA}$

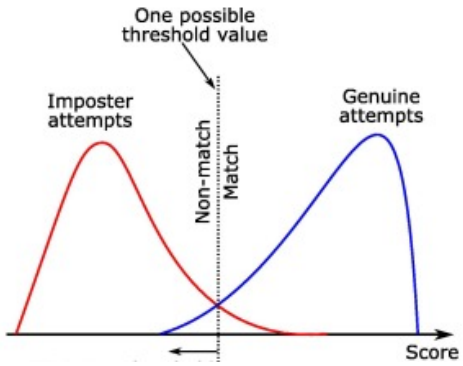
Notes| Neural Networks





Assessing the Performance of Biometric Systems





Two Important Specifications in Biometric system

1. FRR measures the percent of valid users who are rejected as impostors.

FRR = FNR(False negative rate) **FRR = FN/(FN + TP)**

2. FAR measures the percent of invalid users who are incorrectly accepted as genuine users

FAR(false acceptance rate)=
FPR (false positive rate)
=1-Specificty

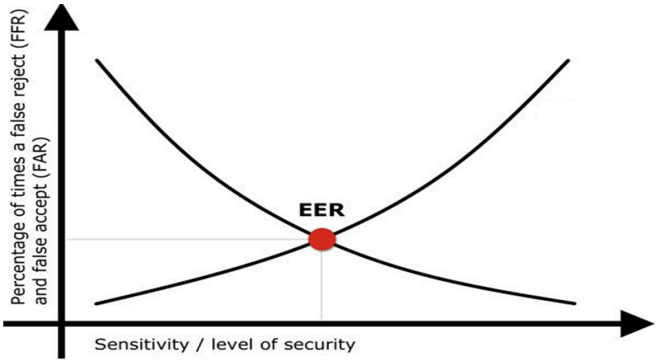
Notes| Neural Networks

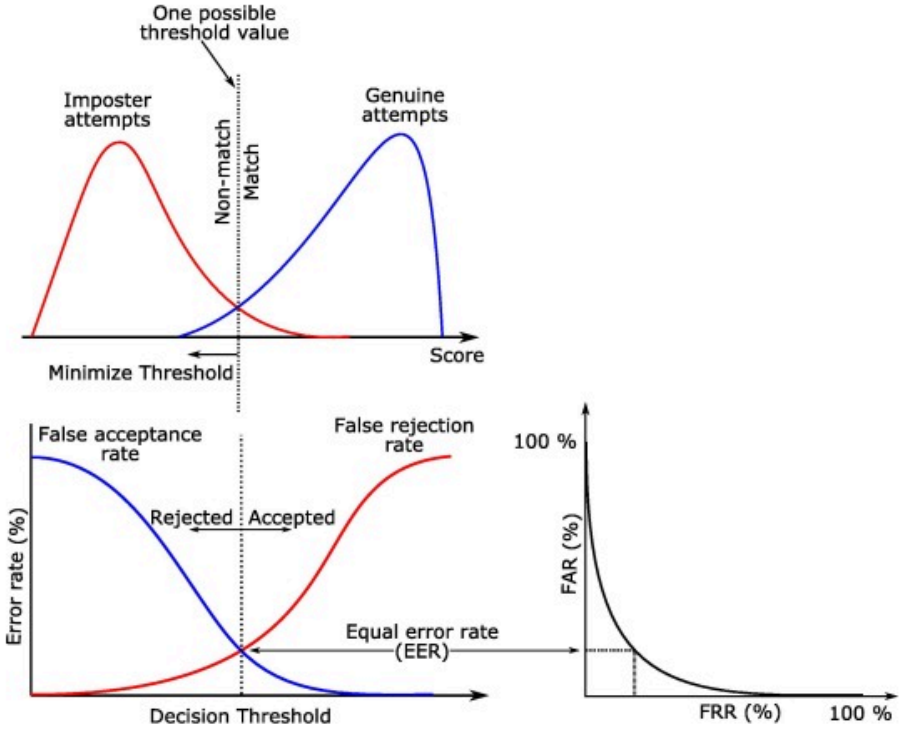
- *If FRR is 0.05%*
  - *One out of every 2000 authorized persons attempting to access the system will not be recognized by that system.*

- If FAR is 0.1%
  - One out of every 1000 impostors attempting to breach the system will be successful.

for Security level FAR (False Acceptance Rate) should be LOW and
For User Convenience FRR (False rejection rate) should be LOW

The accuracy of biometric authentication is calculated based on
FAR as a security index and FRR as a usability index
The lower the equal error rate value, the higher the accuracy of the biometric system





ROC Plot and

Notes| Neural Networks

AUC (Area under ROC curve)

Receiver Operator Characteristic

Developed in WW-II to statistically model FP and FN detections of radar operators

Standard measure in medicine and biology

Becoming more popular in Machine Learning

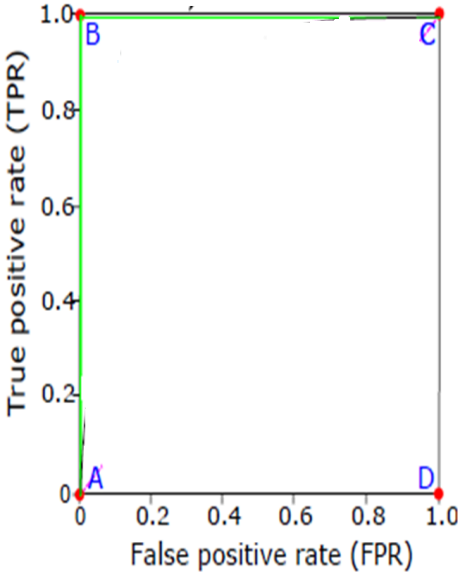
Plot of the

TPR (= TP/TP+FN)

vs

FPR (= FP/TN+FP)

as a function of the model's threshold For displaying the performance of a binary classification algorithm.



Point A (0,0)

TPR = 0 and FPR = 0

all positive samples misclassified

all negative samples correctly classified [threshold very very high]

Point C (1,1)

TPR = 1, FPR =1

all positive samples correctly classified, all the negative samples misclassified [threshold very very low]

The point D (1,0) *TPR = 0 and FPR = 1*

All positive and negative samples are misclassified

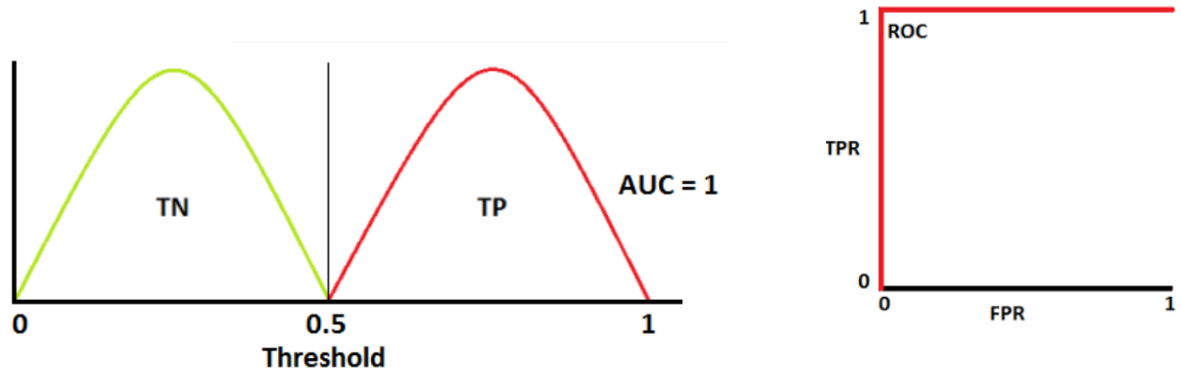
The point B (0,1) *TPR = 1 and FPR = 0*

All positive and negative samples are correctly classified;

represents the perfect classification or the Ideal operating point.

The ROC curve must pass through the point (0,0) where the threshold value is plus infinity (in which all samples are classified as negative samples) and the point (1,1) where the threshold is minus infinity (in which all samples are classified as positive samples)

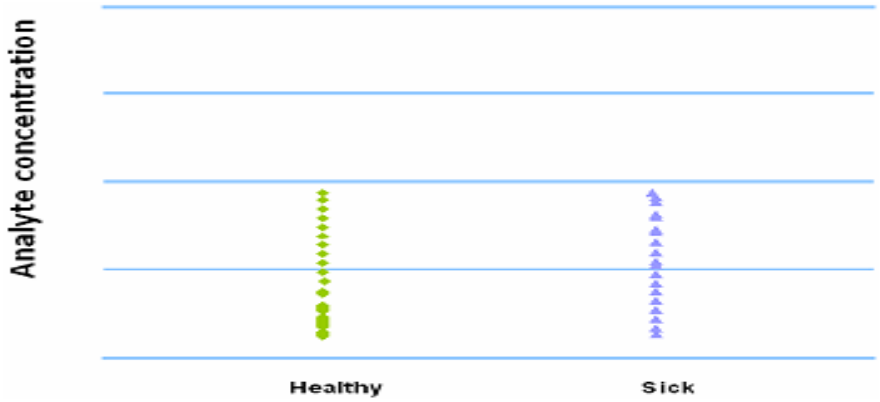
Notes| Neural Networks



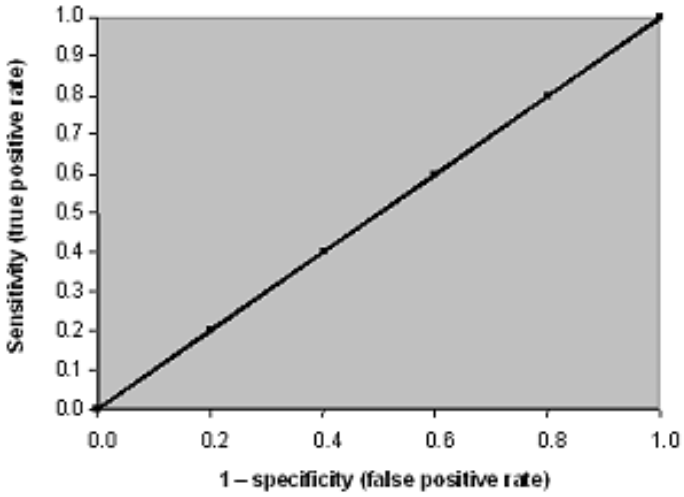
This is an ideal situation. When two curves don't overlap at all means model has an ideal measure of separability.
It is perfectly able to distinguish between positive class and negative class.

A perfect test is able to discriminate between the healthy and sick with 100 % sensitivity and 100 % specificity. **No overlap between healthy and sick**
It will have a ROC curve that passes through the upper left corner (~100 % sensitivity and 100 % specificity). The area under the ROC curve is 1.

worthless test gives complete overlap between the results from the healthy and the results from the sick population
A worthless test has a discriminating ability equal to flipping a coin.



Notes| Neural Networks



The ROC curve of the worthless test falls on the diagonal line. It includes the point with 50 % sensitivity and 50 % specificity.

The area under the ROC curve of the worthless test is 0.5.

ANNs

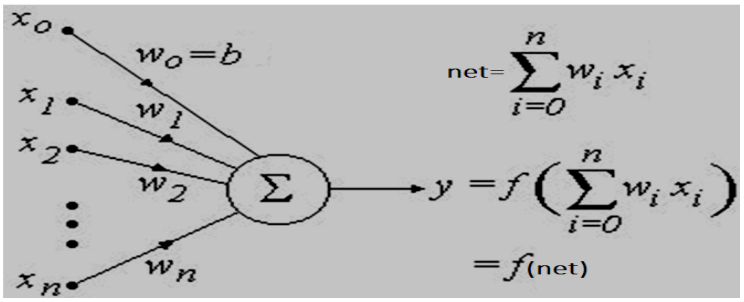
History
Activation Functions
Architectures
Perceptron Algorithm

Designers of the first neural network.

McCulloch (neuroscientist) and Pitts (logician) tried to understand how the brain could produce highly complex patterns by using many basic brain cells called neurons that are connected together.

- Rosenblatt known as the founder of Neurocomputing designed the first neural network called a perceptron.
- In 1977 Associative memory model was developed by Finnish scientist Kohonen.
- John Hopfield introduced Hopfield Net in his 1982 paper introduced the concept of energy.
- In 1986 the multi-layer perceptron was proposed.

Notes| Neural Networks



An artificial neuron is characterized by the parameters
$\theta = (w_1, w_2, \ldots, w_n, b, f)$
f is the activation function.
The bias b can be treated as another "weight" by adding an input node x_0 that always takes the input value $x_0 = +1$ and setting $w_0 = b$.

The structure of ANN is composed of:

- Activation Function (*how neurons respond*)
- Architecture (*topology of how the network is structured*)
- Learning Rule (*how the weights of the connections change over time, w.r.t input, output, and error*).

Activation Function of ANN

A Neural Network without Activation function would simply be a Linear regression model.
Activation functions add to NN :

1. *the ability to learn and model images, videos, audio, speech, etc*
2. *the ability to Represent non-linear complex arbitrary functional mapping between inputs and outputs.*

Properties of Activation Functions

Differentiable: all the activation functions in BPA (Backpropagation algorithm) need to be differentiable

Computational Expense: Activation functions are applied after every layer and need to be calculated millions of times in deep networks. Hence, they should be computationally inexpensive to calculate.

Vanishing Gradient problem: If gradients shift towards zero (vanishing gradient), it takes much time to converge into the global optimum or it even can't reach that

Notes| Neural Networks

point.

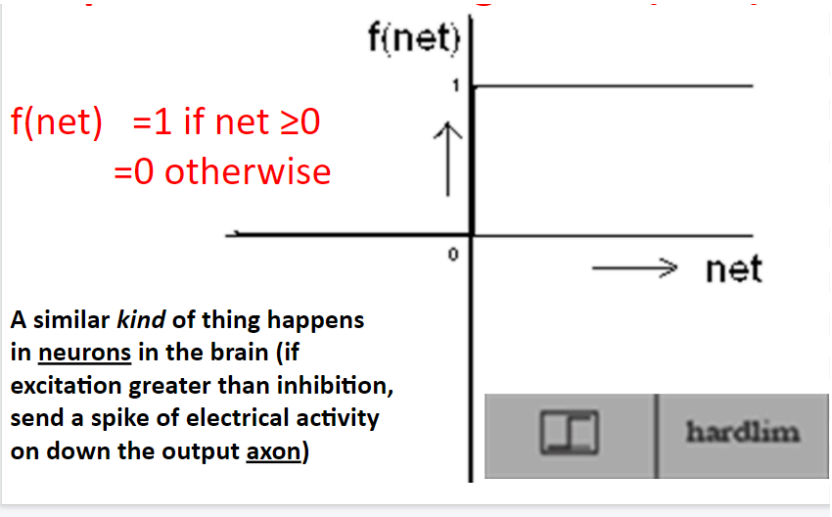
Monotonicity: Most of the activation functions are monotonic i.e either entirely non-decreasing or entirely non-increasing. A derivative of activation may not be monotonic

Two types of Activation Functions

1. Linear Activation Function
2. Non-linear Activation Functions

Unipolar Binary Step Function

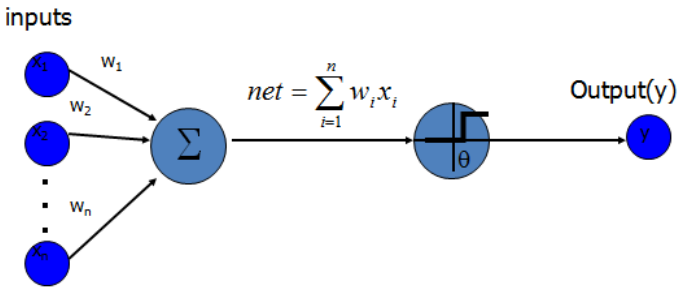
Unipolar Threshold Logic Unit (TLU)



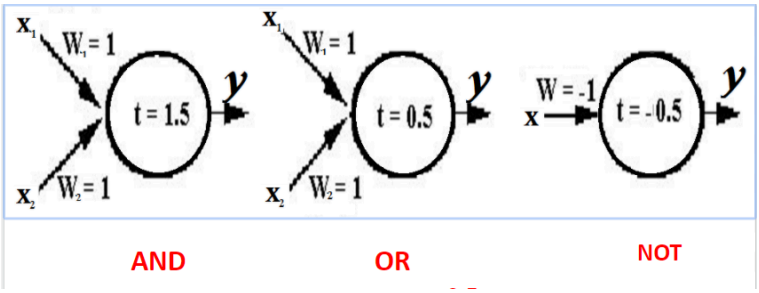
Unipolar Threshold Logic Unit (TLU)

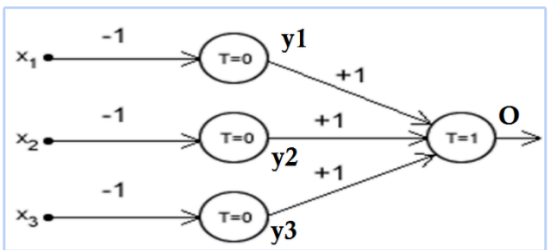
y= 1 if net≥ θ
0 if net < θ

This activation function can be used in binary classifications it can not be used in a multi-classes situation



IDENTIFY GATES with binary input/output **activation is unipolar TLU, SINGLE NEURON**

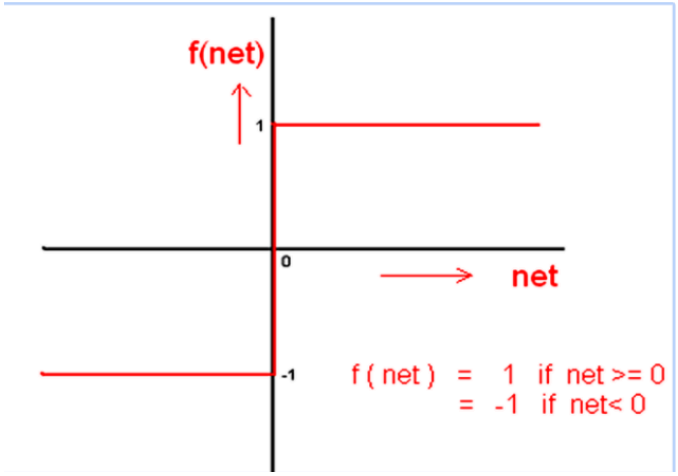




NAND GATE [only 1,1,1, will give output as 0

Notes| Neural Networks

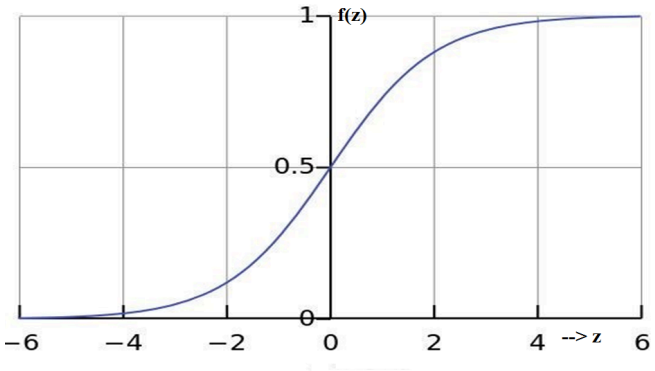
Bipolar Binary Step Function

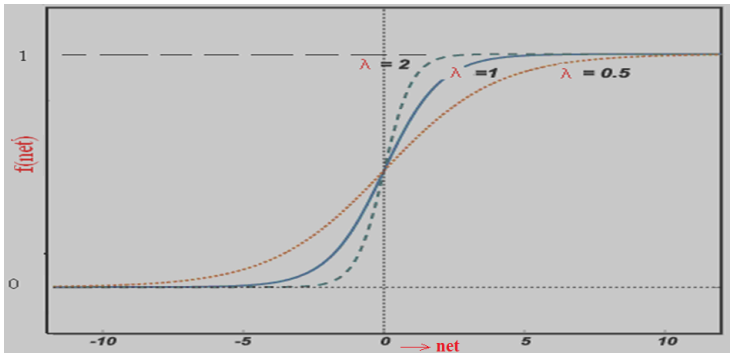


slope is zero except at origin Not differentiable,

Log sigmoid Activation function: output varies between 0 and 1, sigmoid function s makes it unique for the binary classification problems

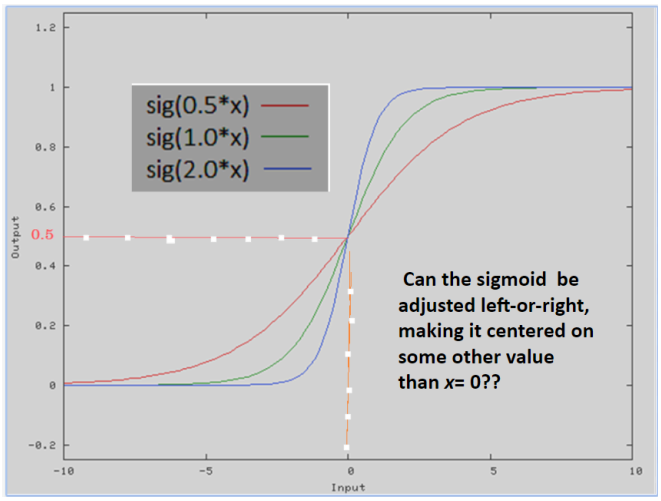
S-shaped curve

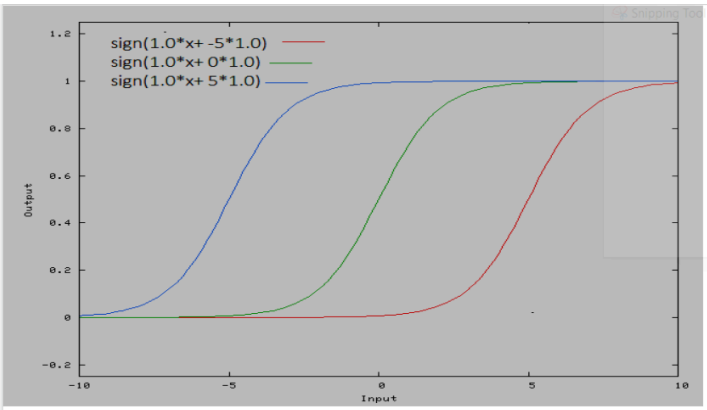




$$\log sigmoid = y = f(net) = \frac{1}{1+\exp(-\lambda.net)}$$

$$\lambda - Slope$$





a bias value (-5/0/5)to sig with slope of 1 allows you to shift the activation function to the left or right, which may be critical for successful learning.

Notes| Neural Networks

- net=-10, f(net)=0.000045,
- net=0, f(net)=0.5,
- net=10, f(net)=0.99995

when the neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero.

Sigmoids saturate and kill gradients.

If the initial weights are too large then most neurons would become saturated and the network will barely learn.

Pay extra caution when initializing the weights of sigmoid neurons to prevent saturation.

Sigmoid is especially used for models where we have to predict the probability as an output.

Since the probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

Log Sigmoid - Not a zero-centered (output can be either negative or positive) function (T/F)

✔ Logsigmoid- the output is always positive so it is not a zero-centered function similar to a situation where you are only allowed to move left and forward, not allowed to move right and backward then it is very hard to reach the desired destination.

✔ Because of e^x, it is *computationally-intensive* which makes convergence slower.

$$Calculate\ y' = f'(net)\ wrt\ net\ in\ terms\ of\ y$$

$$where\quad y = f(net) = \frac{1}{1+e^{-\lambda net}}$$

$$y' = f'(net) = \lambda y(1-y)$$

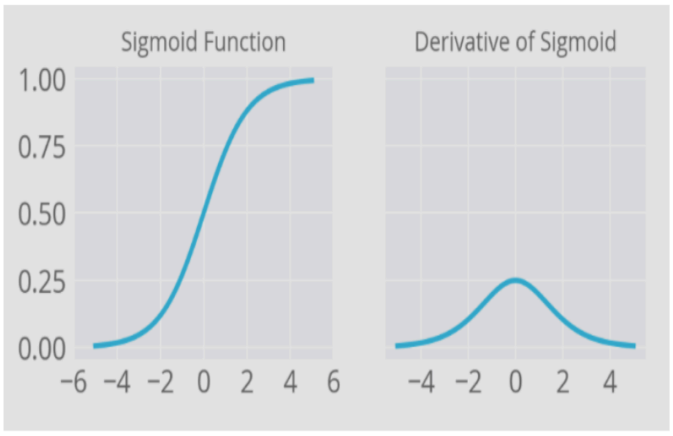
The derivative of sigmoid can be expressed in terms of function itself - so computationally easy to perform.

Notes| Neural Networks

At what value of net, Derivative of the log-sigmoid activation function is maximum and what is its value

dy'/y= 0 implies ⇒1 – 2y = 0 , i.e. y = 1/2
derivative is maximum at y= =1/2= 1/1+e^{-net} ⇒ net =0
and maximum value of slope is
y'= y(1-y)= ½(1-1/2)=1/4=0.25

If λ=2, a maximum value of derivative y' will be 2(1/4)=1/2



The function is **monotonic** while its **derivative is not**

- The maximum value of the derivative of the sigmoid is 0.25 and the minimum is 0.
- The maximum value is less than 1. Many times multiplication of smaller numbers results in a very small number (close to *0*).
- So this is how vanishing gradient is happening in a neural network (NN) with sigmoid activations.

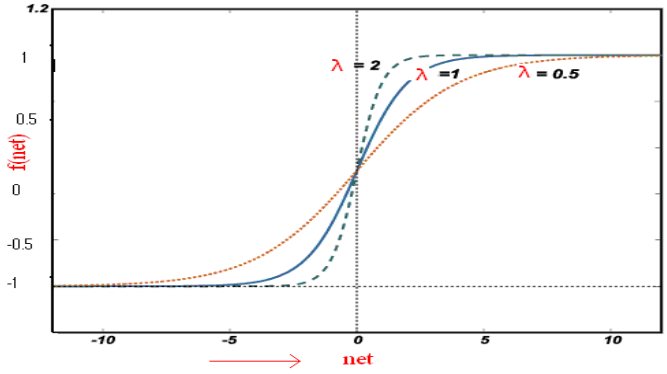
✔ If NN is deep and all activations are sigmoid, then there is a very high chance of vanishing gradient.

For the BPA in a neural network, it means
Errors multiplied by the derivatives of activation function will be squeezed by (at least) a quarter at each layer.
SO deeper your network is, more knowledge from the data will be "lost".
Some "big" errors from the output layer might not be able to affect the weight of the synapse of a neuron in a relatively shallow layer much ("shallow" means it's close to the input layer).

Notes| Neural Networks

Due to this, sigmoids have fallen out of favor as activations on hidden units in DNNs.

Tan Sigmoid activation function



The output is zero centered because its range is between -1 to 1.
The tan sigmoid/tanh function is mainly used for classification between two classes.

$$\tan sigmoid \quad y = f(net) = 2\log sigmoid - 1$$

$$= \frac{2}{(1+e^{-net})} - 1$$

$$= \frac{1-e^{-net}}{1+e^{-net}}$$

$$\tan sigmoid \quad y = f(net) = \frac{2}{(1+e^{-net})} - 1$$

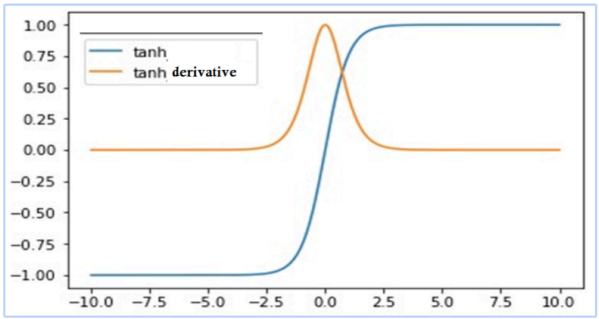
$$= \frac{1-e^{-net}}{1+e^{-net}}$$

$$\tanh : y = f(net) = \frac{2}{(1+e^{-2net})} - 1$$

$$= \frac{1-e^{-2net}}{1+e^{-2net}}$$

$$f'(net) = 0.5(1-y^2)$$

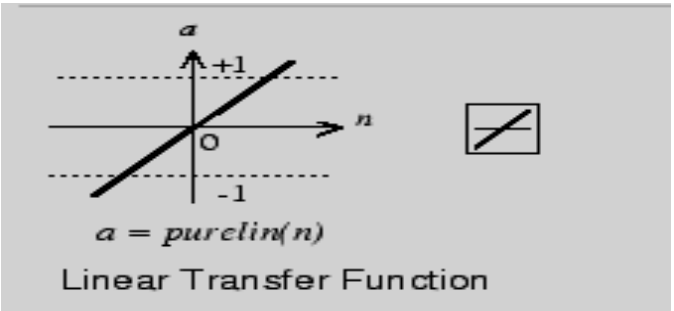
For tanh
derivative is $(1-y^2)$



The gradient is stronger for tanh than sigmoid (derivatives are steeper).
Tanh also has the vanishing gradient problem.

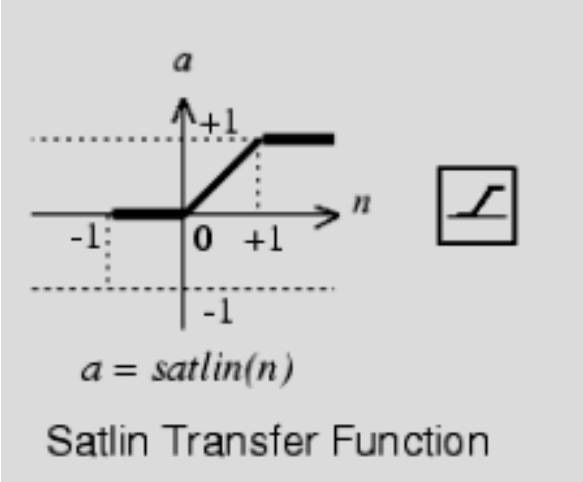
Notes| Neural Networks

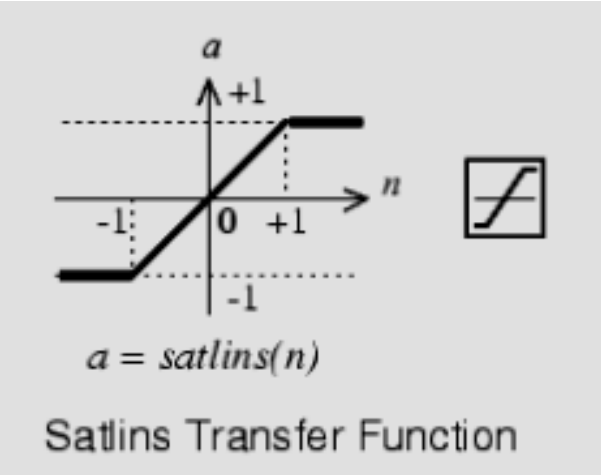
Linear Activation function

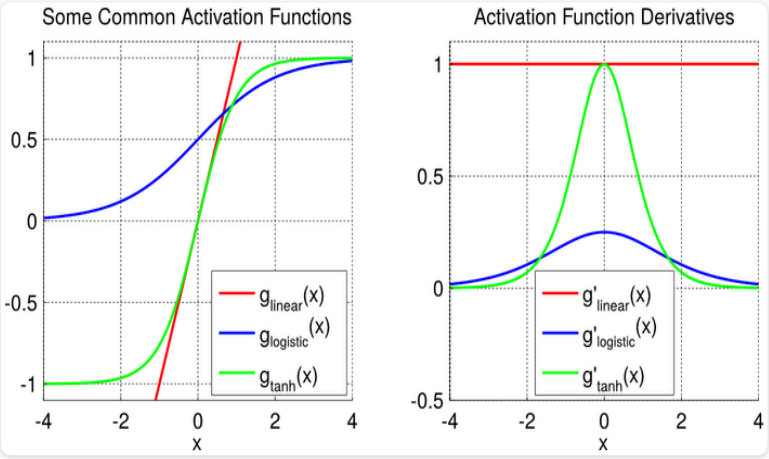


y = f(net)= c*net, y'=f'(net) = c,

if c=1, f'(net)=1







With linear activation function changes made in back-propagation will be constant which is not good for learning

Notes| Neural Networks

Ex: The input to a SINGLE NEURON is 2.0, its weight is 2.3, and bias is -3.0

1. What is the net input to the activation/transfer function?
2. What is neuron output if activation function is

(i) Hard limit[0/1] (ii) linear (iii) logsigmoid

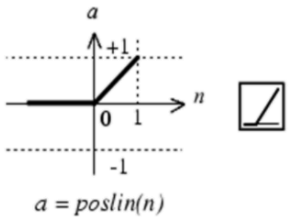
Net input is : 2x(2.3) – 3 = 1.6

$$output = hard\lim(1.6) = 1$$

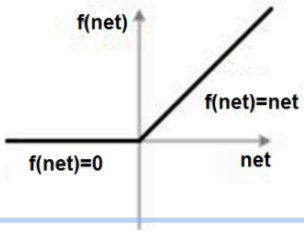
$$= lin(1.6) = 1.6$$

$$= \log sig(1.6) = \frac{1}{1+e^{-1.6}} = 0.8320$$

Currently, the most popular activation function for neural networks is the rectified linear unit (ReLU), which was first proposed for restricted Boltzmann machines (**Nair & Hinton, 2010)**



Positive Linear Transfer Funct



f(net) = max(0,net) i.e
If net < 0 , f(net) = 0 and
if net >= 0 , f(net) = net.

ReLu Activation Function

✔Easy to *compute*.

✔Does not *saturate* for the positive value of the weighted sum of inputs.

✔ *ReLu* is a zero-centered function. (T? F)
False

✔defined as $max(0, w_1 x_1 + w_2 x_2 + \ldots + b)$

✔non-differentiable at zero; and the value of the derivative at zero can be arbitrarily chosen to be 0 or 1

✔may suffer from the "Dying ReLU problem".

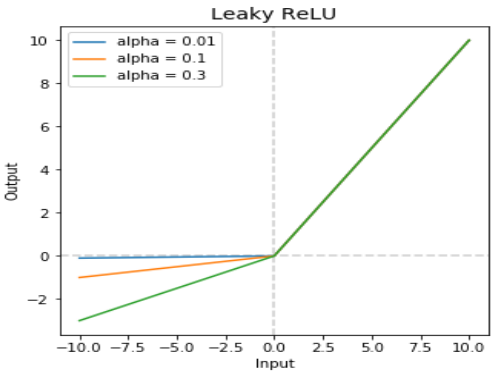
Notes| Neural Networks

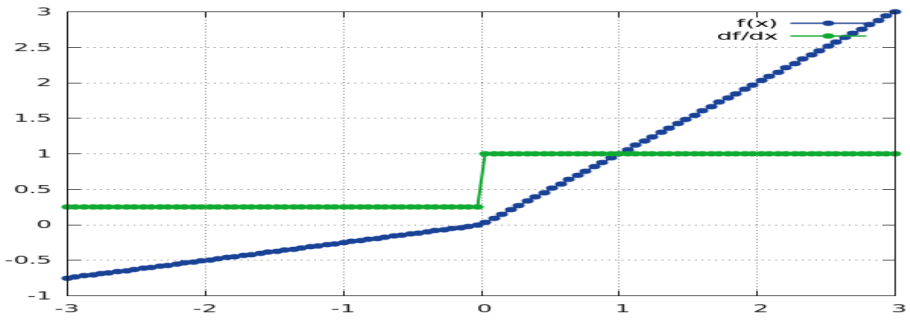
Dying ReLU problem: Any negative input given to the ReLU activation function turns the value into zero, not mapping the negative values appropriately.

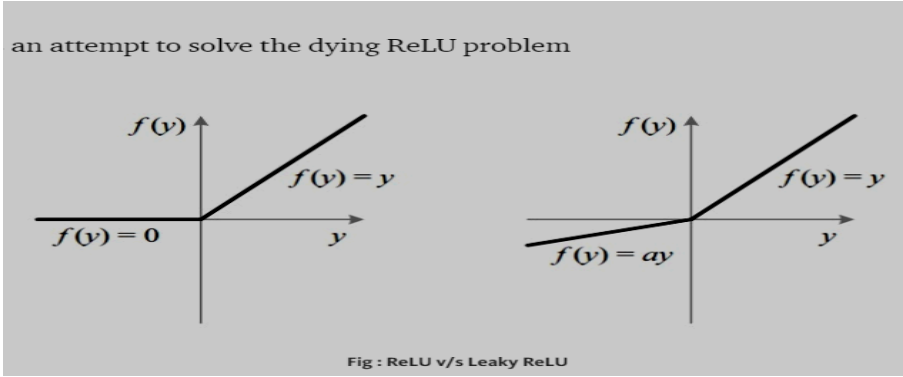
ReLU neurons can sometimes be pushed into states in which they become inactive for essentially all inputs.

In this state, no gradients flow backward through the neuron, and so the neuron becomes stuck in a perpetually inactive state and "dies".

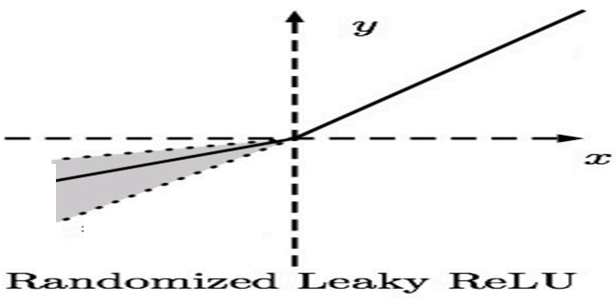
This is a form of the vanishing gradient problem.







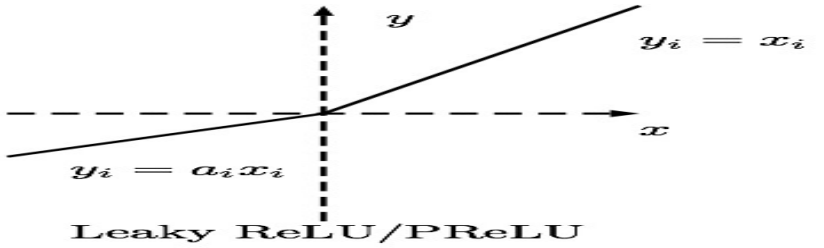
Notes| Neural Networks



Randomized Leaky ReLU

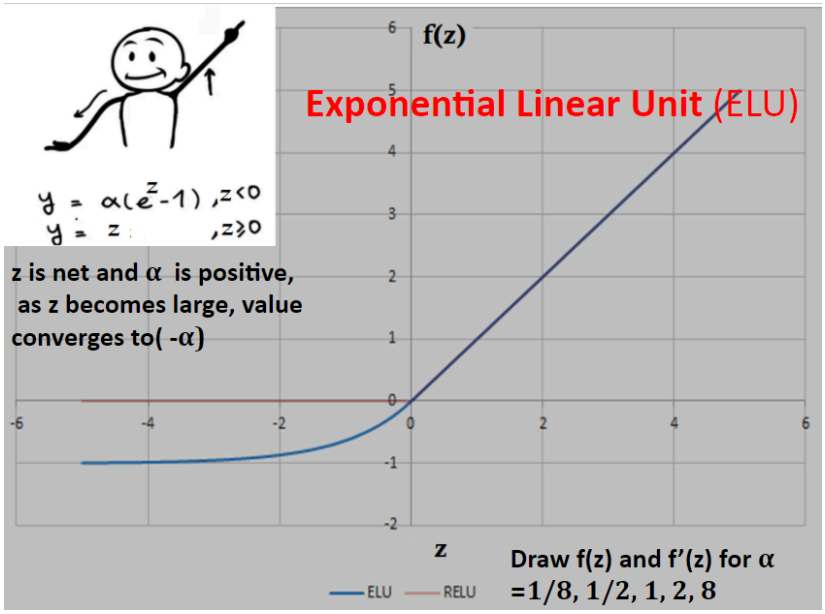
When **α is not 0.01** then it is called **Randomized ReLU**

In Keras, Leaky ReLu has default value of **α=** 0.3



Leaky ReLU/PReLU

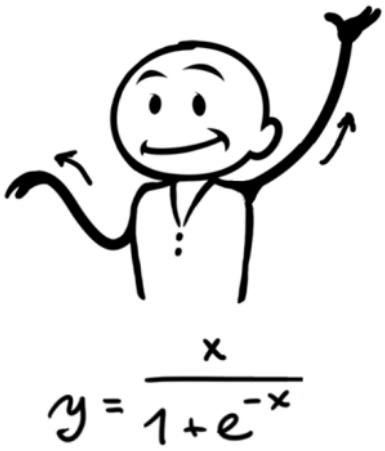
Parametric ReLU has the same advantage with the only difference that the slope of the output for negative inputs is a learnable parameter while In Leaky ReLU, it's a fixed hyperparameter.

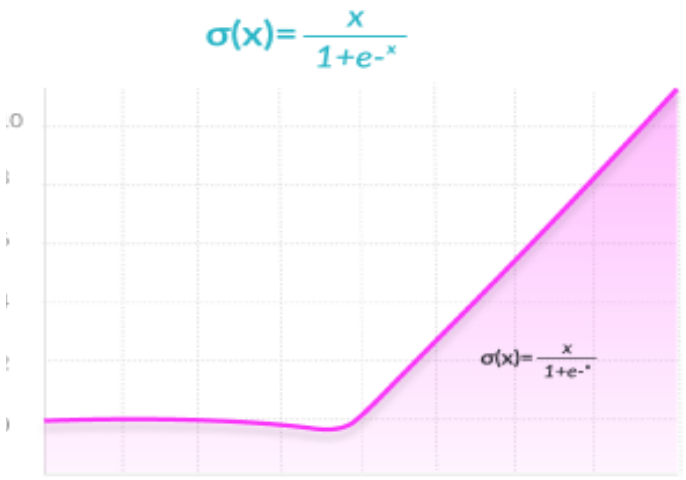


Continuously Differentiable Exponential Linear Units(CELU)

$$\mathrm{CELU}(x,\alpha) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha\left(\exp\left(\frac{x}{\alpha}\right) - 1\right) & \text{otherwise} \end{cases} \qquad \frac{d}{dx}\mathrm{CELU}(x,\alpha) = \begin{cases} 1 & \text{if } x \geq 0 \\ \exp\left(\frac{x}{\alpha}\right) & \text{otherwise} \end{cases}$$

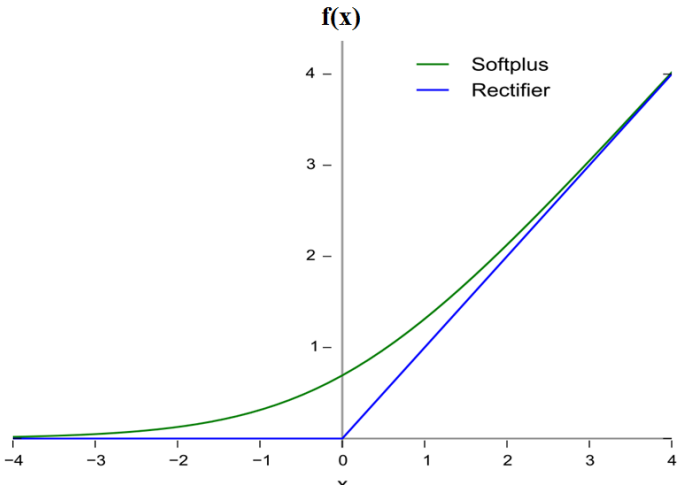
Notes| Neural Networks





Swish is a new activation function discovered by researchers at Google.
In experiments on ImageNet with identical models running ReLU and Swish, the new function achieved top -1 classification accuracy 0.6-0.9% higher.

A smooth approximation to the rectifier is the analytic function called **softplus** function



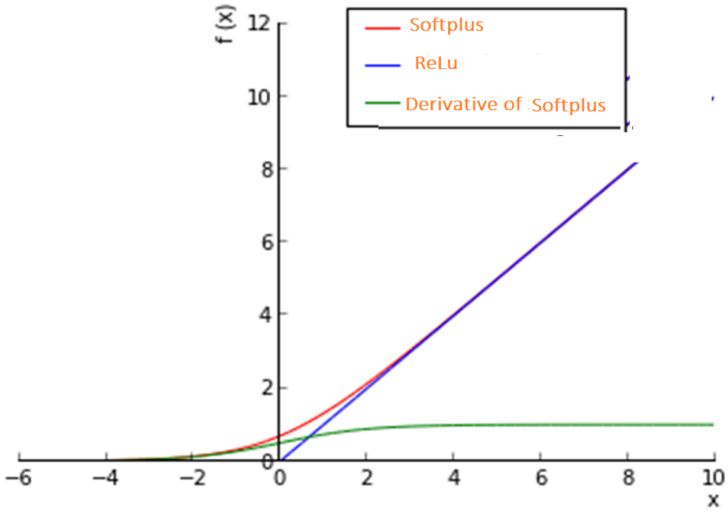
Both the ReLU and Softplus are largely similar, except near 0 where the softplus is enticingly smooth and differentiable.

Both function and its derivative are monotonic

Notes| Neural Networks

$$f(net) = \ln[1 + \exp(net)]$$

$$f'(net) = \frac{\exp(net)}{[1 + \exp(net)]} = \frac{1}{1 + \exp(-net)} = \log sigmoid$$



ReLu- Rectified Linear units : it had *6 times improvement in convergence* from tanh function.

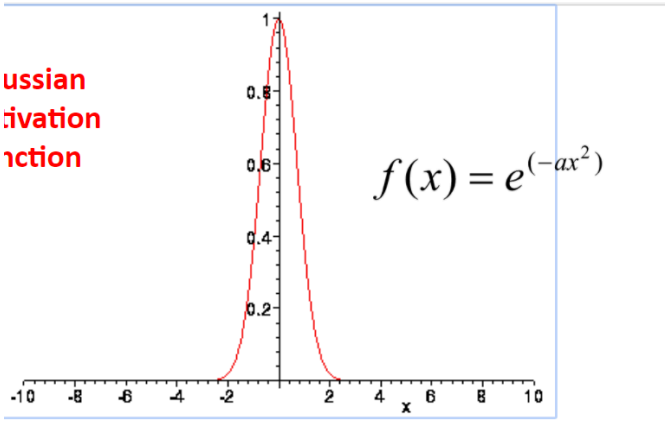
it is very simple and efficient .

it avoids and rectifies vanishing gradient problem.

Almost all deep learning Models use ReLu

In Classification problems, it should only be used within Hidden layers of a Neural Network Model.

Gaussian Activation Function



it is essentially zero everywhere except in a small region around zero.

Notes| Neural Networks

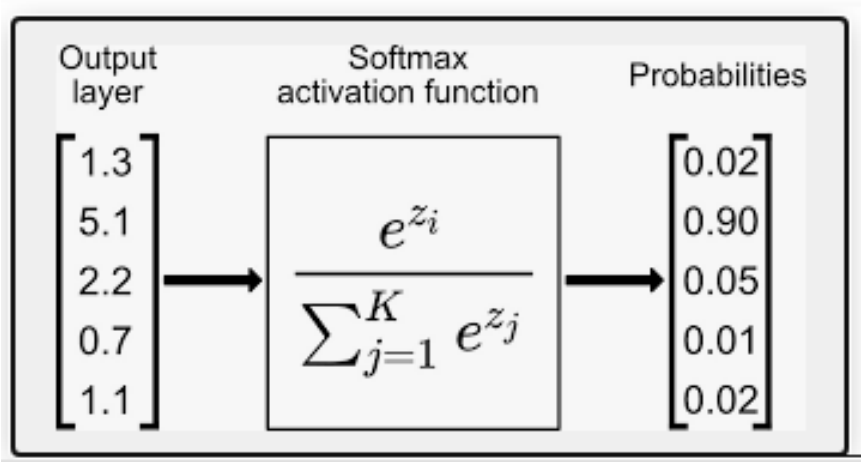
Multiclass classification –

✔classification task with more than two classes.
✔model is to predict one possible class output every time.

Multiclass classification assumption :
each image/sample is assigned to one and only one label - one sample cannot, for example, be both an orange and an apple.

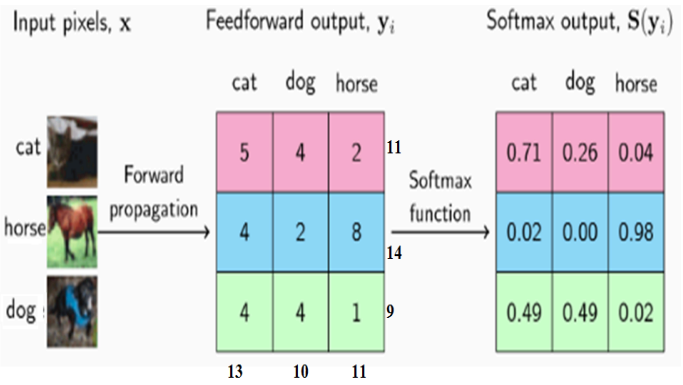
SOFTMAX FUNCTION

Softmax is applied to the last layer in a neural network, to get probabilities for different classes in the output.



✔The Softmax is meant to be a *"softer",* less binary (less all-or-nothing- Hardmax function).
✔ Softmax makes sure that one value is very high (close to 1) and all other values are very low (close to 0).
✔Sum of all the individual probabilities must be equal to one.

- ✔ *Softmax can be used for any number of classes.*
- ✔ *It's also used for hundreds and thousands of classes, in object recognition problems where there are hundreds of different possible objects.*
- ✔ This function ensures that all the output nodes have values between 0–1 and the sum of all output node values equals to 1 always.



Output of Softmax describes the probability/confidence of a neural network that a particular sample belongs to a certain class.

Notes| Neural Networks

Multiple Class, Multiple Label Classification

Classify tweets in natural disaster related classes like

1. *Emergency*
2. *Food Shortage*
3. *Medical Emergency*
4. *Water Shortage*
5. *People Stuck*

Single tweet can belong to multiple classes

Softmax OR Sigmoid in multi-label ?

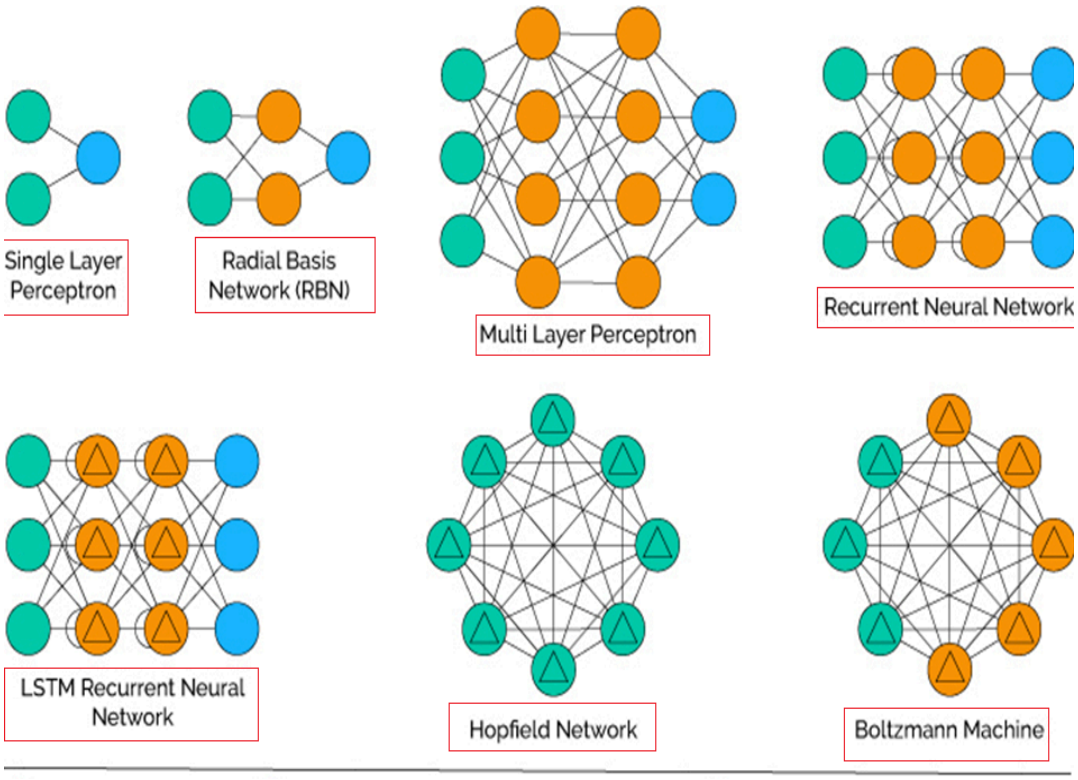
Softmax can't be used as the sum of the probabilities in multi-label case doesn't add to one, each class probability is between 0 and 1

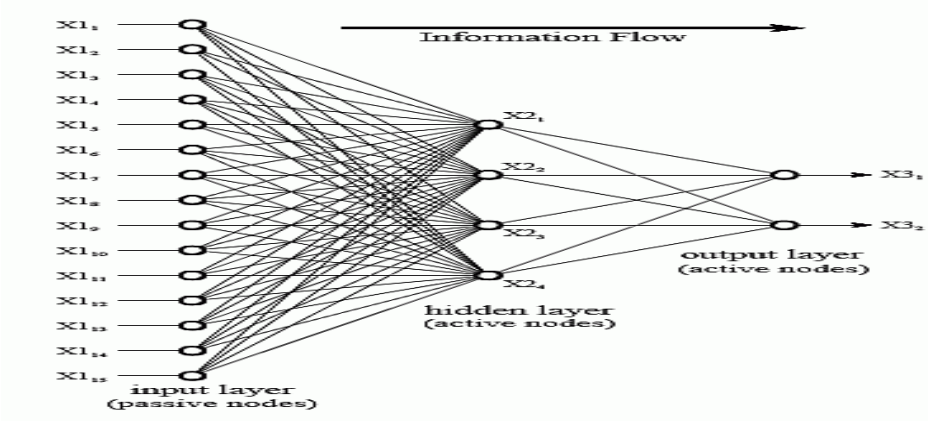
A sigmoid function is used as there may be multiple labels that have high probabilities for a given data item.

This is a common mistake even with experienced practitioners.

Notes| Neural Networks

NEURAL NETWORK ARCHITECTURES





The nodes of the input layer are passive, meaning they do not modify the data.
The nodes of the hidden and output

layer are active.

Notes| Neural Networks

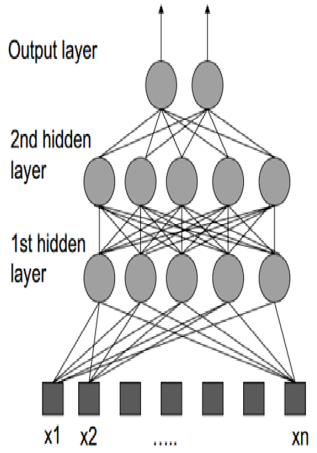
Setting number of layers and their sizes

- ✔*Should we use no hidden layers?*
- ✔*One hidden layer?*
- ✔*Two hidden layers?*

✔How large should each layer be?

✔As the size and number of layers in a Neural Network are increased, space of represent able functions grows since the neurons can collaborate to express many different functions.

Feed Forward Neural Networks



The feedforward neural network was the first and simplest type of ANN.

In this network, the information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any) and to the output nodes.

There are no cycles or loops in the network.

MNIST

Each pixel takes a value between 0 and 255 (RGB color code).

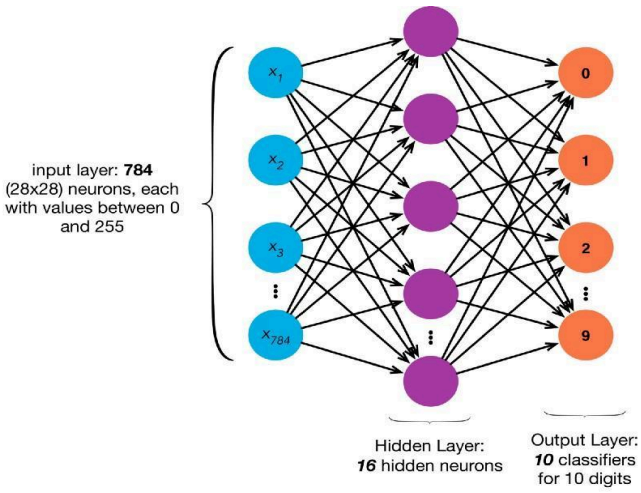
0 means the color is white and 255 means the color black.

dissect the image (28x28) into an array of 784 numbers like [0, 0, 180, 16, 230, …, 4, 77, 0, 0, 0], and feed this array into neural network

B&W image means pixel value can be 0/1

Notes| Neural Networks

total learnable parameters with biases.
[786x16] + [16x10] + [16] +[10] =12,762

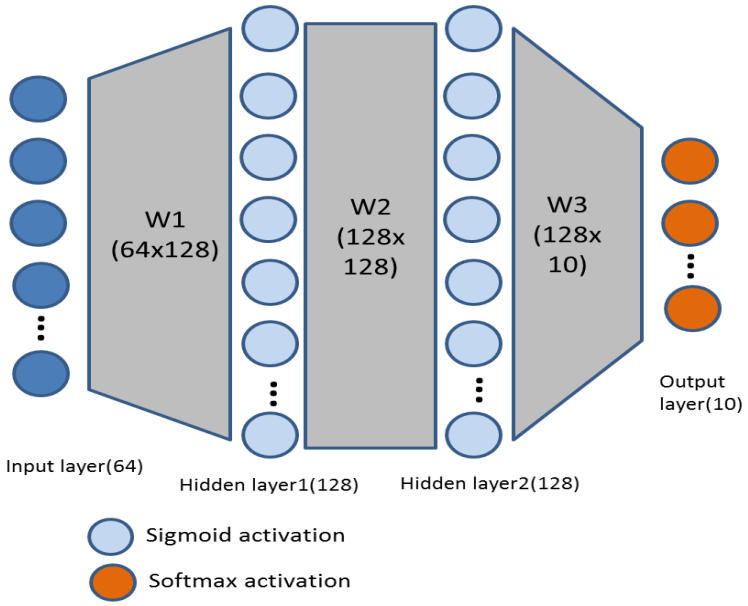


The 10 output neurons,each represents yes/no for a digit from 0 to 9.

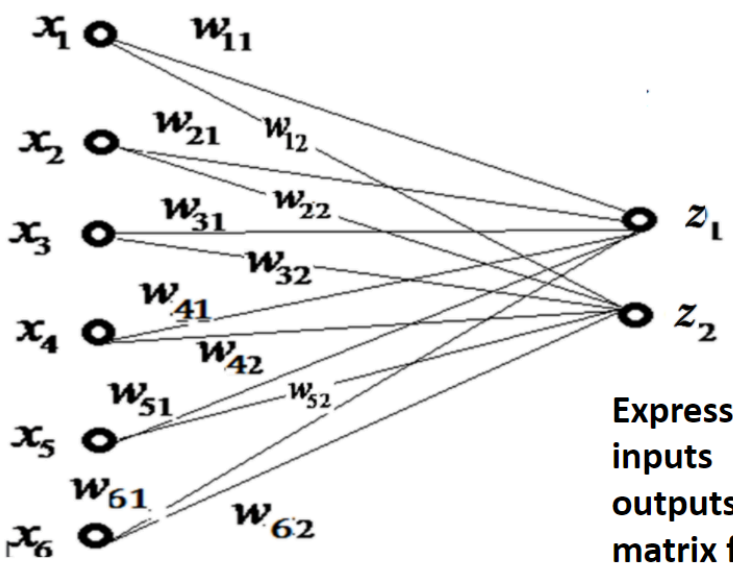
If the neural network thinks the handwritten digit is a zero, then we should get an output array of
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],

If the neural network thinks the handwritten digit is a 5, then we should get
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0].

Single high (1) bit and all the others low (0) ONE HOT REPRESENTATION



Notes| Neural Networks



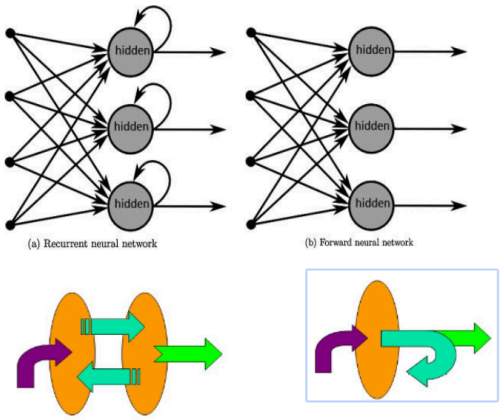
Express inputs outputs in matrix form

WX = Z

$$\begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} & w_{51} & w_{61} \\ w_{12} & w_{22} & w_{32} & w_{42} & w_{52} & w_{w_{62}} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

$$= \begin{bmatrix} w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + w_{41}x_4 + w_{51}x_5 + w_{61}x_6 \\ w_{12}x_1 + w_{22}x_2 + w_{32}x_3 + w_{42}x_4 + w_{52}x_5 + w_{62}x_6 \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}$$

FEEDBACK/RECURRENT NETWORKS



Notes| Neural Networks

Back-Propagation algorithm

- the workhorse of learning in neural networks.
- The BPA was originally introduced in the 1970s, but its importance wasn't fully appreciated until a famous 1986 paper
- *Backpropagation is a very popular neural network learning algorithm because it is conceptually simple, computationally efficient, and because it often works.*
- *Simplest control system that does its work is the best]*
- *However, getting it to work well, and sometimes to work at all, seem more of an art than a science.*
- there are no rules to follow to "*best*" configure a model and training process.
- BPA requires the careful configuration of model and choice of hyperparameters.
- *These choices can be critical, yet there is no foolproof recipe for deciding them because they are largely problem and data dependent.*
- A hyperparameter is a parameter/ variable need to set before applying a machine learning algorithm onto a dataset for training.

Hyperparameters are divided into two categories:

1.Optimizer Hyperparameters

variables/ parameters related more to the optimization and training process

2. Model Hyperparameters

variables/parameters involved in the architecture or structure of the model.

Optimizer Hyperparameters

- Learning Rate: It is a hyperparameter that controls how much we are adjusting the weights of our neural network with respect to the gradient.
- Mini-Batch Size: It is a hyperparameter that has an effect on the resource requirements of the training and also impacts training speed and number of iterations.
- Number of Training Iteration / Epochs, Acceptable error
- Momentum rate, regularization parameter

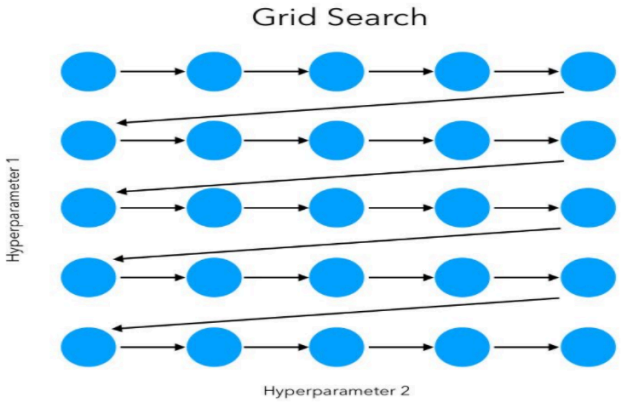
Notes| Neural Networks

Model Hyperparameters

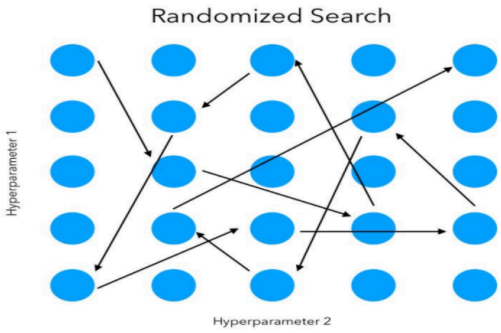
- Variables/parameters involved in the architecture or structure of the model.
- Hidden layers & Number of Hidden Units: Number of hidden layers & neurons in hidden layers in a neural network.
- Activation function
- Model Specific parameters for the architectures like in CNN Kernel width, in RNN: Choosing a cell type like LSTM Cells, Vanilla RNN cells or GRU Cells and how deep the model is.

Approaches to hyperparameter tuning

- Manual: select hyperparameters based on intuition/experience/guessing, train the model with the hyperparameters - an exhausting endeavour.
- Grid Search
- Random Search
- Automated Hyperparameter Tuning: More advanced methods such as Bayesian Optimization and Evolutionary Optimization.

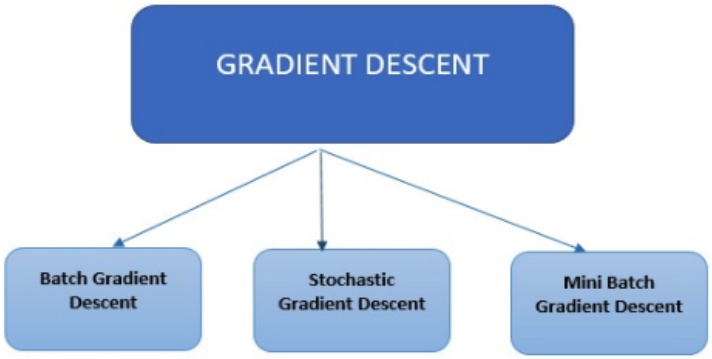


Grid Search: Set up a grid of hyperparameter values and for each combination to train the model and score on the validation data.



Random search: set up a grid of hyperparameter values and select *random* combinations to train the model and score.

Notes| Neural Networks



Full Batch/ Batch Gradient Descent:

- ANN weights are updated after each Epoch, i.e. weight update is based on training error from all N samples in training data set

✔ Also called per Epoch learning - involves *less frequent weight updates,* which makes training rather slow[especially for large data set]

✔ per-epoch learning computes the actual network error of all samples and makes more informed decisions about weight updates.

If there are a million samples in a dataset, with Batch/FB Gradient Descent optimization technique all the one million samples will be fed before making a set of changes (one epoch) , and it has to be done for every epoch until the minima is reached. So computationally very expensive

Stochastic Gradient Descent(SGD) uses only a single sample, i.e. a batch size of ONE, randomly selected, to perform each iteration.

The sample is randomly shuffled and selected for performing the iteration.

Notes| Neural Networks

SGD(Stochastic gradient Descent) also called per-Pattern mode/Sample-by-sample/Online

the whole sequence of forward and backward computation is performed resulting in weight adjustment for each pattern

ANN weights are updated every time a training sample is presented to the network

During per-pattern learning, it is important to present the examples in random order.

- only one data point for updating parameters, SGD is heavily affected by Outliers.
- while randomly choosing data points, if we encountered an Outlier, then it will take some more extra updates to get back on its track of convergence.
- imagine if it is again encountered an outlier which will throw it off the track.
- SGD is sensitive to outliers.
- Chances of overfitting in SGD is less
- Overfitting happens when the model instead of learning how to answer the question, remembers the answer to the question, so when a different question of the same concept is asked, it does not perform well.
- Overfitting will happen when the model considers the same data points repeatedly, such that its parameters are adjusted according to only those parameters (instead of learning the pattern).
- For new data points (test dataset), the error is large.
- By choosing different data point for each update we are restricting the model from remembering the answer.
- SGD is more generalized and less prone to overfitting.

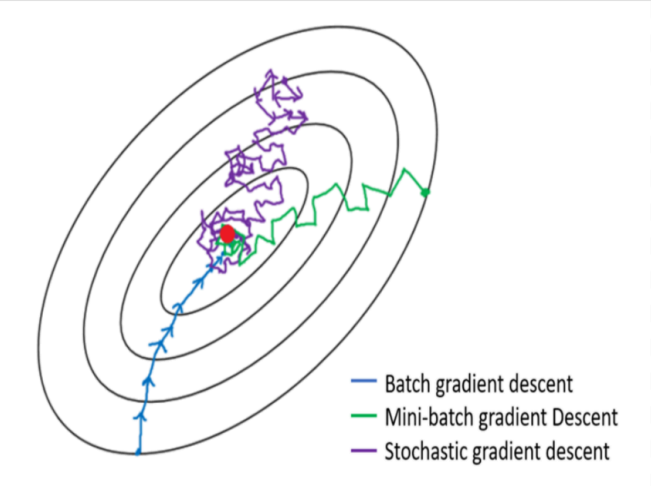
Notes| Neural Networks

If SUBSET of the full dataset is used for calculating weight updates, it is called Minibatch gradient Descent.

- In Mini batch Gradient descent, training is fast and also has smoother gradient updates then SGD.

- Mini batch gradient descent is used in all deep learning applications.

Note: In many books or articles people use the name Stochastic Gradient Descent for the Mini batch gradient descent too.

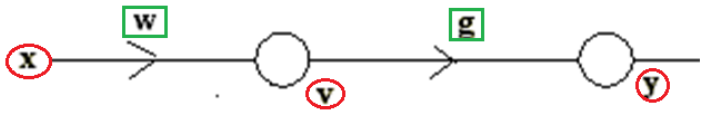


BACK-PROPAGATION LEARNING ALGORITHM

The training involves THREE stages:
1)Feed forward of the input training pattern
2)Calculation and Backpropagation of the associated error
3) Adjustment of weights/biases

Notes| Neural Networks

INSTANTANEOUS BACK PROPAGATION ALGORITHM



$$net_h = wx, v = f(net_h) = \frac{1}{[1 + \exp(-wx)]}$$

$$net_o = vg, y = f(net_o) = \frac{1}{[1 + \exp(-gv)]}$$

$$w(t+1) = w(t) - \eta \frac{\partial E}{\partial w(t)};\ w_{new} = w_{old} - \eta \frac{\partial E}{\partial w_{old}};\ \Delta w = -\eta \frac{\partial E}{\partial w_{old}}$$

$$g(t+1) = g(t) - \eta \frac{\partial E}{\partial g(t)};\ g_{new} = g_{old} - \eta \frac{\partial E}{\partial g_{old}},\ \Delta g = -\eta \frac{\partial E}{\partial g_{old}}$$

$$E = \frac{1}{2}(y_d - y)^2 \qquad Find\ \frac{\partial E}{\partial g(t)}\ \&\ \frac{\partial E}{\partial w(t)}$$

$$v = \frac{1}{[1 + \exp(-wx)]} \qquad y = \frac{1}{[1 + \exp(-gv)]}$$

$$\frac{\partial E}{\partial g(t)} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial g(t)} \qquad \frac{\partial E}{\partial w(t)} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial w(t)}$$

$$\frac{\partial y}{\partial w(t)} = \frac{\partial y}{\partial v(t)} \times \frac{\partial v}{\partial w(t)}$$

$$E = \frac{1}{2}(y_d - y)^2 \qquad \frac{\partial E}{\partial y} = -(y_d - y)$$

$$\frac{\partial E}{\partial g(t)} = \frac{\partial E}{\partial y} \bullet \frac{\partial y}{\partial g} = -(y_d - y) \bullet \frac{\partial y}{\partial g}$$

$$y = \frac{1}{1+\exp(-net_o)}, net_o = gv, y = \frac{1}{1+\exp(-gv)};$$

$$\frac{\partial y}{\partial g(t)} = \frac{\partial y}{\partial net_o} \times \frac{\partial net_o}{\partial g} = y(1-y)v$$

$$\frac{\partial E}{\partial g(t)} = -(y_d - y)y(1-y)v$$

$$E = \frac{1}{2}(y_d - y)^2 \qquad g(t+1) = g(t) - \eta \frac{\partial E}{\partial g(t)}$$

$$\frac{\partial E}{\partial g(t)} = -(y_d - y)y(1-y)v$$

$$g(t+1) = g(t) - \eta \frac{\partial E}{\partial g(t)}$$

$$= g(t) + \eta(y_d - y)y(1-y)v$$

$$E_2 = (y_d - y)y(1-y)$$

$$g(t+1) = g(t) + \eta E_2 v$$

$$\Delta g = \eta E_2 v$$

$$v = \frac{1}{[1+\exp(-wx)]} \quad y = \frac{1}{[1+\exp(-gv)]} \quad \frac{\partial E}{\partial w(t)} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial w(t)} \quad \frac{\partial E}{\partial y} = -(y_d - y)$$

$$\frac{\partial y}{\partial w(t)} = \frac{\partial y}{\partial v(t)} \times \frac{\partial v}{\partial w(t)}$$

$$\frac{\partial y}{\partial v} = \frac{\partial y}{\partial (net_o)} \times \frac{\partial (net_o)}{\partial v};\ gv = net_o \qquad \frac{\partial v}{\partial w} = \frac{\partial v}{\partial (net_h)} \times \frac{\partial (net_h)}{\partial w};\ wx = net_h$$

$$= y(1-y)(g) \qquad = v(1-v)(x)$$

$$\frac{\partial y}{\partial w(t)} = \frac{\partial y}{\partial v(t)} \times \frac{\partial v}{\partial w(t)} = y(1-y)g \times v(1-v)x$$

$$\frac{\partial E}{\partial w(t)} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial w} = -(y_d - y)y(1-y)g \times v(1-v)x$$

Calculating new weights from input to hidden layer

$$w(t+1) = w(t) - \eta \frac{\partial E}{\partial w(t)}$$

$$\frac{\partial E}{\partial w(t)} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial w} = -(y_d - y)y(1-y)g \times v(1-v)x$$

$$w(t+1) = w(t) - \eta \frac{\partial E}{\partial w(t)}$$

$$w(t+1) = w(t) + \eta(y_d - y)y(1-y)g \times v(1-v)x$$

$$w(t+1) = w(t) + \eta E_1 x$$

$$E_1 = v(1-v)E_2 g$$

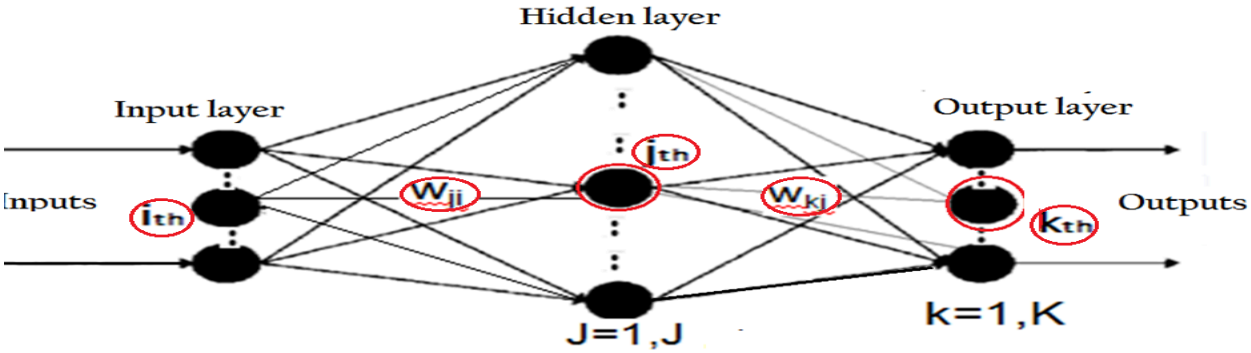
$$E_2 = (y_d - y)y(1-y)$$

Notes| Neural Networks

$$E_2 = (y_d - y)y(1-y) = (y_d - y)f'(net_o);$$
$$net_0 = gv; y = f(net_0)$$
$$E_1 = v(1-v)E_2 g = f'(net_h)E_2 g;$$
$$net_h = wx, v = f(net_h)$$

$$g(t+1) = g(t) + \eta E_2 v = g(t) + \eta E_2 f(net_h)$$
$$w(t+1) = w(t) + \eta E_1 x$$

GENERAL BPA



After inputting p[th] pattern, calculate outputs at all hidden and output nodes. Compute error for the pattern 'p' across all nodes at the output layer .

Write Error component at k [th] o/p node for p[th] pattern e^0_{pk}

$$e^0_{pk} = (y_{kp} - o_{kp})f'(net^o_{kp})$$

Write expression for error component at j[th] hidden node for pattern p.

$$e^h_{pj} = f'(net^h_j)\sum_{k=1}^{K} e^o_{pk} w_{kj}$$

Update the connection-weight values to the output layer

$$\Delta w_{kj} = \eta e^o_{pk} f(net^h_j)$$

Notes| Neural Networks

Update the connection-weight values from input to the hidden layer

$$\Delta w_{ji} = \eta e^{h}_{pj} x_i$$

•Repeat steps 1 to 6 for all vector pairs in the training set; this is called an EPOCH

•Run as many epochs as required to reduce the network error E to fall below a Threshold ε

•Write Expression of SSE

$$E = \sum_{p=1}^{P} \sum_{k=1}^{K} (y_{kp} - o_{kp})^2$$

REMARKS ON THE BACKPROPAGATION ALGORITHM

- Convergence and Local Minima
  - May converge to some local minimum
    - Perhaps not global minimum...
  - Heuristics to alleviate the problem of local minima
    - Add momentum
    - Use adaptive learning rate
    - Train multiple nets with different initial weights using the same data.

If the change of the cost function gradient between successive iteration steps is not smooth but oscillatory, leading to slow convergence.

One way to overcome this problem is to use a *momentum term* that smoothes out the oscillatory behavior and speeds up the convergence.

Notes| Neural Networks

Momentum is a heavy ball rolling down the same hill. The added inertia acts both as a smoother and an accelerator, damping oscillations and causing us to barrel through narrow valleys, small humps and local minima.

Momentum in Backpropagation

- In BPA with momentum, the weight change is in a direction that is a *combination of the current gradient and previous gradient.*

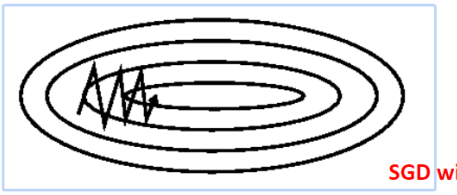
$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

weight increment | learning rate | weight gradient

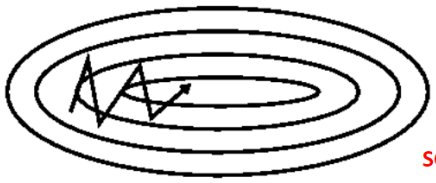
$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor | weight increment, previous iteration

- For each weight
  - *Remember what was added in the previous epoch*
- In the current epoch
  - *Add on a small amount of the previous Δw*



SGD without momentum



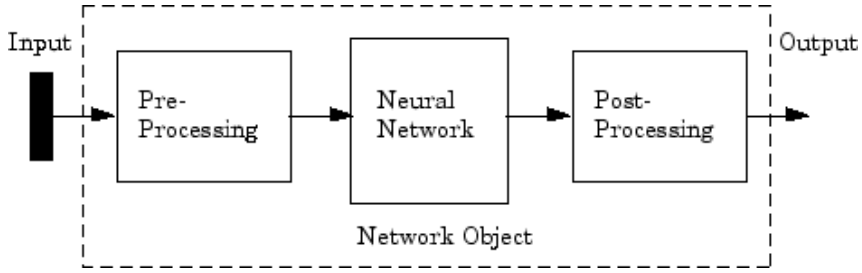
SGD with momentum

n^{th} iteration update depend on $(n\text{-}1)^{th}$ iteration

γ: constant between 0 and 1 is called the *momentum (typical value 0.6 to 0.9).*

Notes| Neural Networks

DATA PRE-PROCESSING/POST- PROCESSING



Data Representation

- Generally ANNs work on continuous (real valued) attributes.
- Attributes of different types[like age, weight, BP, heart rate] may have different ranges of values which affect the training process.
- A data set containing two features, age, and income where age ranges from 0–100, while income ranges from 0–100,000 and higher
- put all features into the same scale [PRE-PROCESS] so that none are dominated by another.

Feature Engineering

✔If data is the crude oil of the 21st century, then feature engineering refines it , and gets a boost in its value.

✔Feature engineering deduces some hidden insights from the crude data, and makes some meaningful features out of it.

✔ Feature engineering requires some good amount of domain knowledge, if it is to be done effectively.

✔to engineer new features out of Forex data [Forex (FX) is the marketplace where various national currencies are traded], you need some good understanding about how Forex, global economies, and currencies actually work.

Notes| Neural Networks

Raw data: pixel grid 256x256 grid		
Better features: clock hands' coordinates	{x1: 0.7, y1: 0.7} {x2: 0.5, y2: 0.0}	{x1: 0.0, y2: 1.0} {x2: -0.38, 2: 0.32}
Even better features: angles of clock hands	theta1: 45 theta2: 0	theta1: 90 theta2: 140

Good features will let you solve a problem more elegantly, using fewer computational resources. Good features will let you train an effective model with little data.

General Guidelines for Pre-processing the data :

- Smaller values: Try to have all the values between 0 and 1, or -1 to 1.
- Homogeneity: All the columns should have values in roughly the same range.
- Mean: Normalize in a way that you have a mean of 0 for each column independently.
- Standard deviation: Normalize in a way that you have a standard deviation of 1 for each column independently.

Feature Scaling

The goal of applying Feature Scaling is to make sure features are on almost the same scale so that each feature is equally important and make it easier to process by most ML algorithms.

Normalizing Data to unit length

vector [3 1 2]

Length of vector = $\text{sqrt}(3^2 + 1^2 + 2^2) = 3.742$

Normalized copy of the vector having unit length

```
x = 3.0 / 3.742 = 0.802
y = 1.0 / 3.742 = 0.267
z = 2.0 / 3.742 = 0.534
```

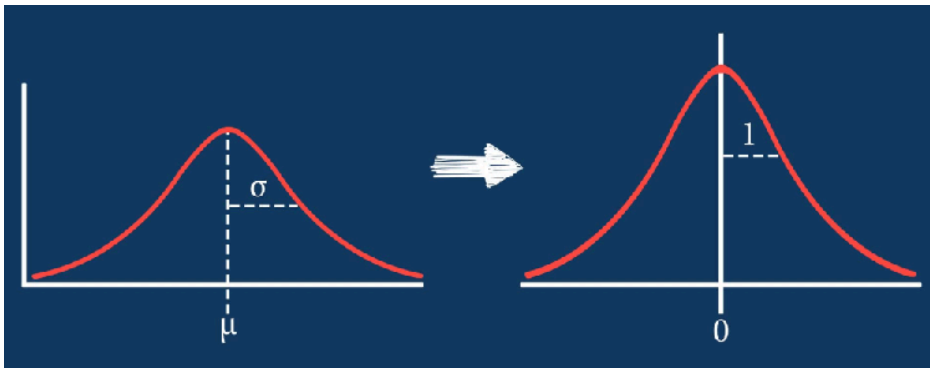
Notes| Neural Networks

Standardization

It will transform the data to have mean 0 and variance 1, having no units.

$$X_{normalised} = \frac{X_{current} - \bar{X}}{\sigma_X}$$

Standardization is useful for comparing variables expressed in different units.



Normalizing both input & target data set during the pre-processing stage ensures that the network output always falls into a normalized range.
During the post-processing stage, the network output is transformed back into the units of the original target data.

NORMALIZATION

Normalizing in the interval [$X_{max\text{-}new}$, $X_{min\text{-}new}$]

$$X_{normalized} = \frac{X_{current} - X_{\min}}{X_{\max} - X_{\min}}(X_{\max-new} - X_{\min-new}) + X_{\min-new}$$

$$X_{normalized}[0,1] = \frac{X_{current} - X_{\min}}{X_{\max} - X_{\min}}$$

$$X_{normalized}[-1,1] = \frac{2(X_{current} - X_{\min})}{X_{\max} - X_{\min}} - 1 = \frac{X_{current} - (X_{\max} + X_{\min})/2}{(X_{\max} - X_{\min})/2}$$

Notes| Neural Networks

Post Processing
(Denormalization of output data)

$$Y_{de-norm} = \frac{Y_{current} - Y_{\min}}{Y_{\max} - Y_{\min}} (Y_{denorm-\max} - Y_{denorm-\min}) + Y_{denorm-\min}$$

Y_{max} and Y_{min} are max and min activation values of the output neurons
$Y_{denorm-max}$ and $Y_{denorm-min}$ are the maximum and minimum values of denormalised Y

$$X_{normalized} = \frac{X_{current} - X_{\min}}{X_{\max} - X_{\min}} (X_{\max-new} - X_{\min-new}) + X_{\min-new}$$

✔Normalization is a good technique to use when you do not know the distribution of your data or when you know the distribution is not Gaussian (a bell curve)

✔Standardization assumes that your data has a Gaussian (bell curve) distribution. This does not strictly have to be true, but the technique is more effective if your attribute distribution is Gaussian.

✔if you have outliers in your feature (column), normalizing your data will scale most of the data to a small/large interval ?

✔Standardisation is more robust to outliers, and in many cases, it is preferable over Max-Min Normalisation.

Issues associated with arriving at global minimum

What is a reasonable learning rate to use?

Notes| Neural Networks

How do we avoid getting stuck in local optima?

What if the loss surface morphology changes?

Even if we can find the global minimum, there is no guarantee that it will remain the global minimum indefinitely.

Examples of loss surface morphology changes

1. Test and training data sets have different distributions , So, when applied to new data, the loss surface will be different.

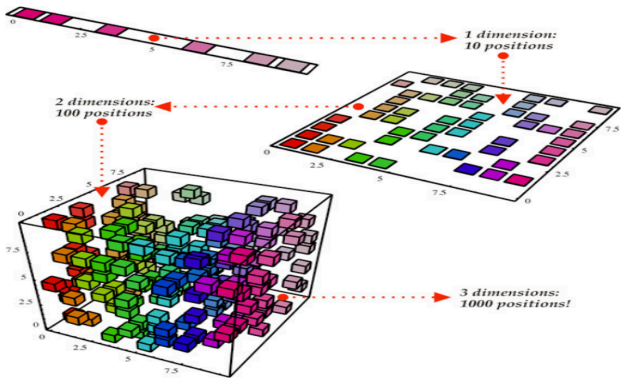
Therefore, making the training and test datasets representative of the total data distribution is important.

2. Dynamic nature of data which habitually changes in distribution— like user preferences for popular music or movies, which changes day-to-day and month-to-month.

If learning algorithm is too slow because the input dimension is too high, then use PCA to reduce dimension space to speed up

Consider a data set that consists of 100k records. Each records contains information about a person(*name, age, height, zodiac sign,country,hair color,weight,obese/not obese*).The data set has 8 features describing each person which means it is a 8 dimension data.The feature *obese* is a categorical variable

that separates the data points into two classes(*obese or not obese)*.

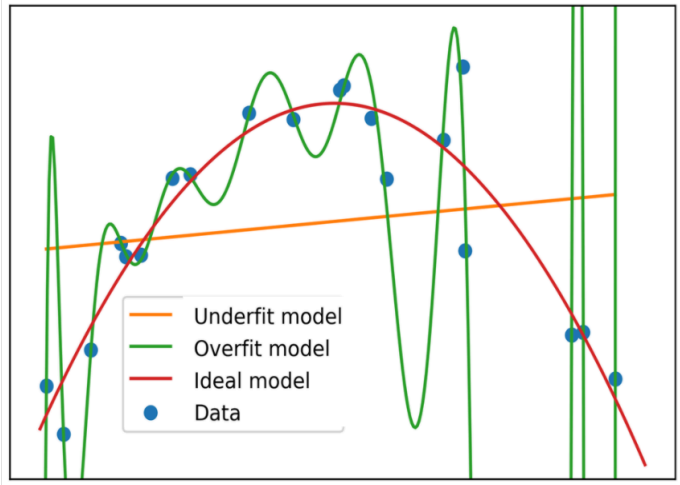


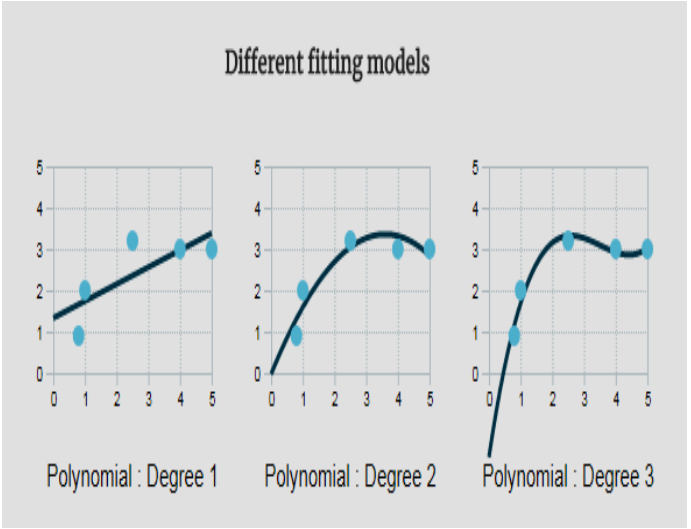
Notes| Neural Networks

✔Overfitting

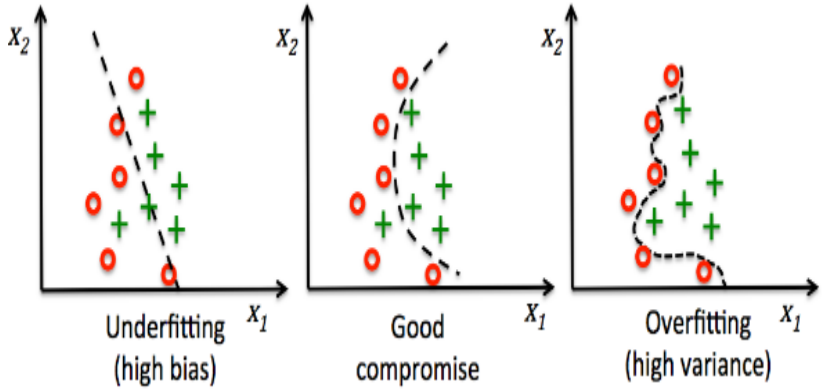
✔Underfitting

✔Regularisation





Underfitting/overfitting in classification problems



Over-fitting results in poor generalization.

- Network fails to respond well when tested and simulated with an unseen data set.

Notes| Neural Networks

- The network actually memorizes the samples it is trained with, instead of learning to generalize the process to respond to unknown conditions.

A model is overfit when :

- ✔Performance on the training data, used to fit the model, is substantially better than performance on a test set
- ✔Model cannot anticipate the fluctuations in the new, unseen data of the testing set.
- ✔There are a large number of features available, relative to the number of samples
- ✔More features there are, greater the chance of discovering a spurious relationship between the features and the response.
- ✔At the opposite end of the spectrum - if a model is not fitting the training data very well, this is known as underfitting, and the model is said to have high bias.
- ✔In this case, the model may not be complex enough, in terms of the features

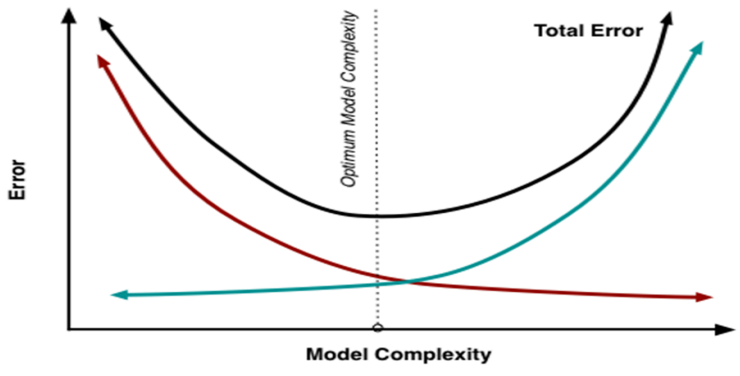
or

the type of model being used.

Bias—high bias means the model is not "fitting" well on the training set.

This means the training error will be large. Low bias means the model is fitting well, and training error will be low.

Variance—high variance means that the model is not able to make accurate predictions on the test set.

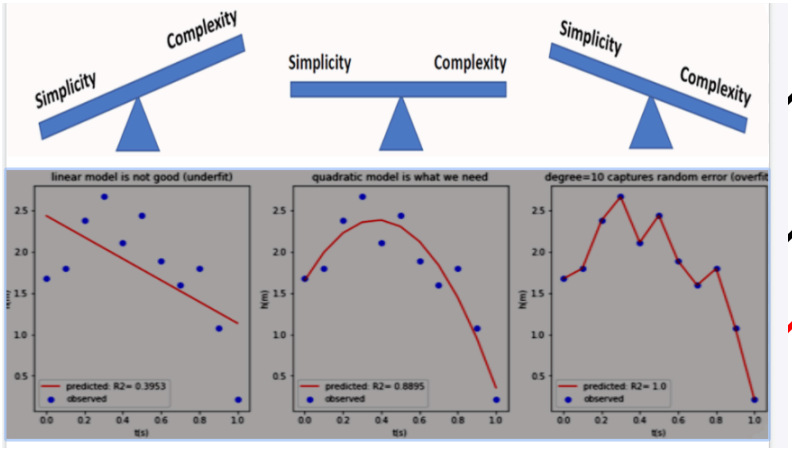


Specifying a network which is just large enough to provide an adequate fit is highly recommended.

Not only will it improve

Notes| Neural Networks

generalization but it will speed up training.



✔ *model may be too complex and overfit*

or

✔ *Model may be too simple and underfit,*

✔ *either way giving poor predictions.*

Regularization-
a technique to avoid over-fitting

✔It is tuning or selecting the preferred level of model complexity so that ANN models are better at predicting (generalizing).

Methods of Regularization

1) Modification of the performance function(MSE) which is, by default mse
2) Early stopping
3) Dropout
4) Data augmentation

The general rule of weights in neural networks is that the higher the weights in the neural network, the more complex the neural network. Because of this, neural networks having higher weights generally tend to overfit.

Smaller weights are, in some sense, lower complexity, and so provide a

simpler and more powerful explanation for the data, and should thus be preferred.

✔In loss function, include a part of the weights , so that weights are also minimized while training

✔There are two methods of doing this, they are called the L1 and L2 regularization

Loss function with L1 regularisation

$$L_1(X, w) = L(X, w) + \lambda \sum |w|$$

Loss function with L2 regularisation

$$L_2(X, \omega) = L(X, \omega) + \lambda \sum \omega_i^2$$

Regularisation parameter $\lambda > 0$ is manually tuned.

Early Stopping

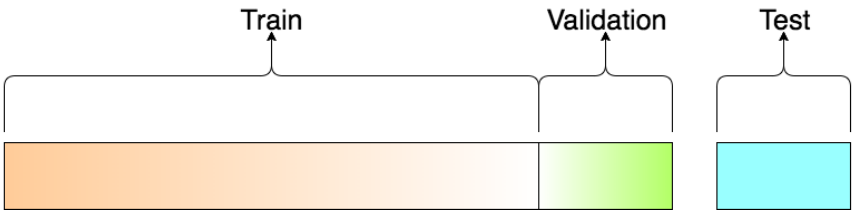
- The default method for improving generalization is called *early stopping*.
- This technique is automatically provided for all of the supervised network creation functions
- In this technique the available data is divided/split into THREE subsets.

Training Set: used to adjust the weights on the neural network.

Testing Set: used only for testing the final solution to confirm the actual predictive power of the network.

Validation Set: used to minimize overfitting.

Notes| Neural Networks



✔Validation data Set is used to minimize overfitting.

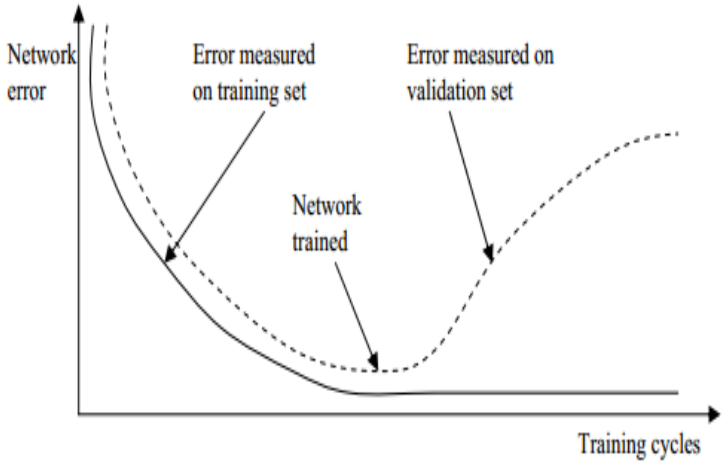
✔Compute the classification accuracy on the validation_data at the end of each epoch.

✔Once the classification accuracy on the validation_data has saturated/starts increasing, we stop training. This strategy is called *early stopping*.

✔Weights of the network are NOT Adjusted with Validation data Set

✔If the accuracy over the training data set increases, but the accuracy over then validation data set stays the same or decreases, then you're overfitting your neural network and you should stop training.

✔When the validation error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned.



✔Why use the validation_data to prevent overfitting, rather than the test_data?

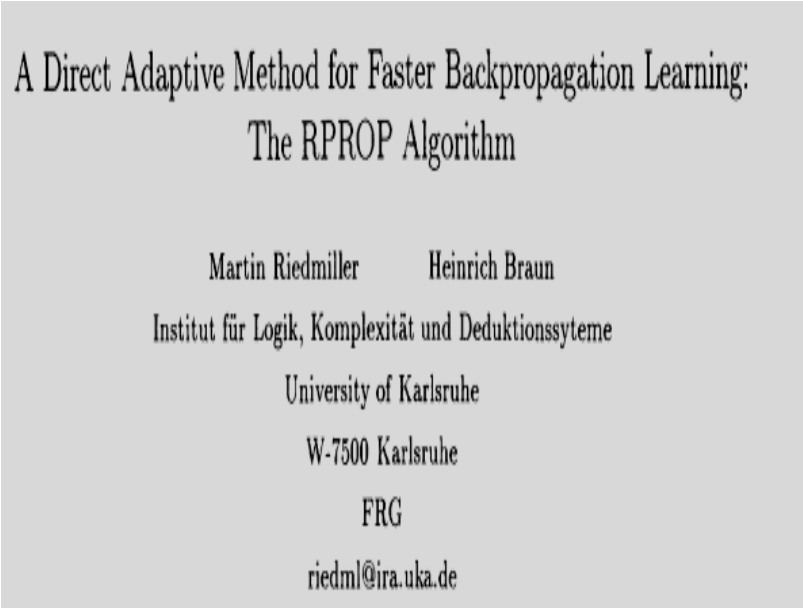
✔ validation_data is used to evaluate different choices of hyper-parameters such as the number of epochs to train for, the learning rate, the best network architecture, and so on.

Notes| Neural Networks

Why not set hyperparameters based on test data ?

✔If we set the hyper-parameters based on evaluations of the test-data..it's possible we'll end up overfitting our hyper-parameters to the test-data, but where the performance of the network won't generalize to other data sets.

✔We guard against that by figuring out the hyper-parameters using the validation data.
✔Then, once we've got the hyper-parameters we want, we do a final evaluation of accuracy using the test data.
✔That gives us confidence that our results on the test data are a true measure of how well our neural network generalizes.
✔ validation data is like a type of training data that helps us learn good hyper-parameters.
✔This approach to finding good hyper-parameters is also known as the *hold out* method, since the validation data is kept apart or "held out" from the training data.



A Direct Adaptive Method for Faster Backpropagation Learning:
The RPROP Algorithm

Martin Riedmiller Heinrich Braun
Institut für Logik, Komplexität und Deduktionssysteme
University of Karlsruhe
W-7500 Karlsruhe
FRG
riedml@ira.uka.de

Most gradient descent variants use the sign and the magnitude of the gradient.
In the Rprop learning algorithm the step-sizes are independent of the absolute value of the partial derivative, depending on the sign of the partial derivative.
A step-size, i.e. the update amount of a weight, is adapted for each weight

individually.

✔After calculating update-value for each weight finally each weight is changed by its own update value, in the opposite direction of that weight's partial derivative, so as to minimise the total error function.

✔if the derivative is positive (increasing error), the weight is decreased by its update-value,

✔if the derivative is negative, the update-value is added, both represented by the following equation:

✔ The Rprop algorithm makes two significant changes to the back-propagation algorithm.

$$\Delta w_{ij}^{(t)} := -\operatorname{sign}\left(\frac{\partial E}{\partial w_{ij}}^{(t)}\right) \cdot \Delta_{ij}^{(t)}$$
$$w_{ij}^{(t+1)} := w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$$

✔ First, Rprop doesn't use the magnitude of the gradient to determine a weight delta; instead, it uses only the sign of the gradient.

✔ Second, instead of using a single learning rate for all weights and biases, Rprop maintains separate weight deltas for each weight and bias, and adapts these deltas during training.

Two main advantages over back propagation:

1. Training with Rprop is often faster than training with back propagation.

2. RPROP doesn't require you to specify any free parameter values, as opposed to back propagation which needs values for the learning rate (and usually an optional momentum term).

✔The main disadvantage of Rprop is that it's a more complex algorithm to implement than back propagation.

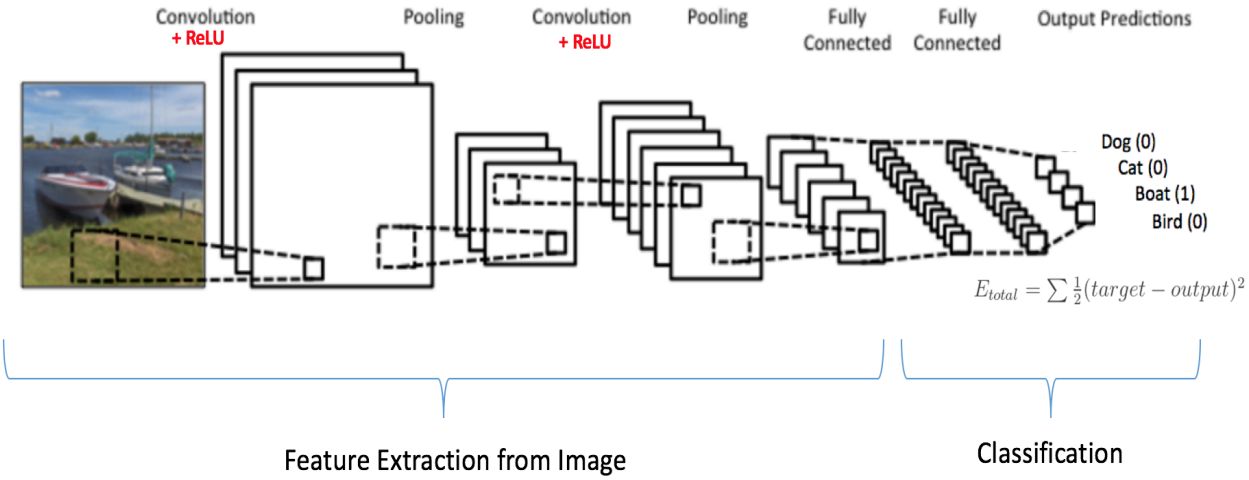
Notes| Neural Networks

Convolution Neural Networks

Convolutional Neural Network (CNN) is a well-known deep learning architecture inspired by the natural visual perception mechanism of the living creatures.
The name "Convolutional Neural Network" comes from the use of the mathematical operation convolution in the network.

✔In Machine Learning, CNN, or ConvNet is a class of deep, Feedforward ANNs, most commonly applied to analyzing visual imagery.

✔ CNNsare trained with a version of the back-propagation algorithm(BPA).



✔The initial layers find edges in an image, or identify the color of the image.

✔The later layers build upon this for complex structures such as finding intersection of edges or shades of colors.

Notes| Neural Networks

✔The final layer then brings all of this together to identify the object in the image.

- Apart from Image Classification, CNNs are now also used for many other applications like:
  - Object Localization and Object Detection
  - Image Segmentation
  - Image Captioning
  - Style Transfer
  - Video Analysis
  - Medical Diagnosis
  - Image Coloring
  - Super Resolution
  - Speech Recognition
  - Document Classification
  - Self Driving Cars
  - Drug discovery to prediction of disease
  - NLP
  - Games….

Comparison of CNN with MLP

Problems with traditional Neural Network Models

Fully connected Networks or Multi Layer Perceptron Model mainly suffers from two big limitations:

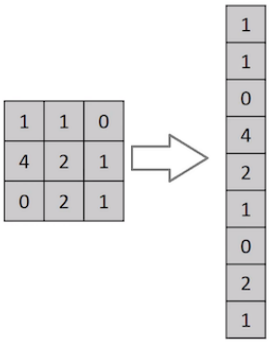
- *Huge number of parameters (weights) are required when inputs are high dimensional.*

Notes| Neural Networks

- *Does not handle the spatial and temporal aspect of the data.*

Spatial Aspect of data

Both are the same images, the second image is obtained by flipping the first image horizontally. Although the subject of the images is same but the **order of pixels** have completely changed now



in fully connected neural networks the weights corresponding to each pixel in the image are learnt, the order in which the pixels are present in the image matters.

If we trained our neural network with images of cats with its face on the left hand side, then neurons of the network will fire when they see the cat face on the left.

- If an inverted image is inputted, the pixels on the left won't correspond to a cat face, instead will correspond to the background and hence we might get different results.

What other operations can change the order of the pixels ???

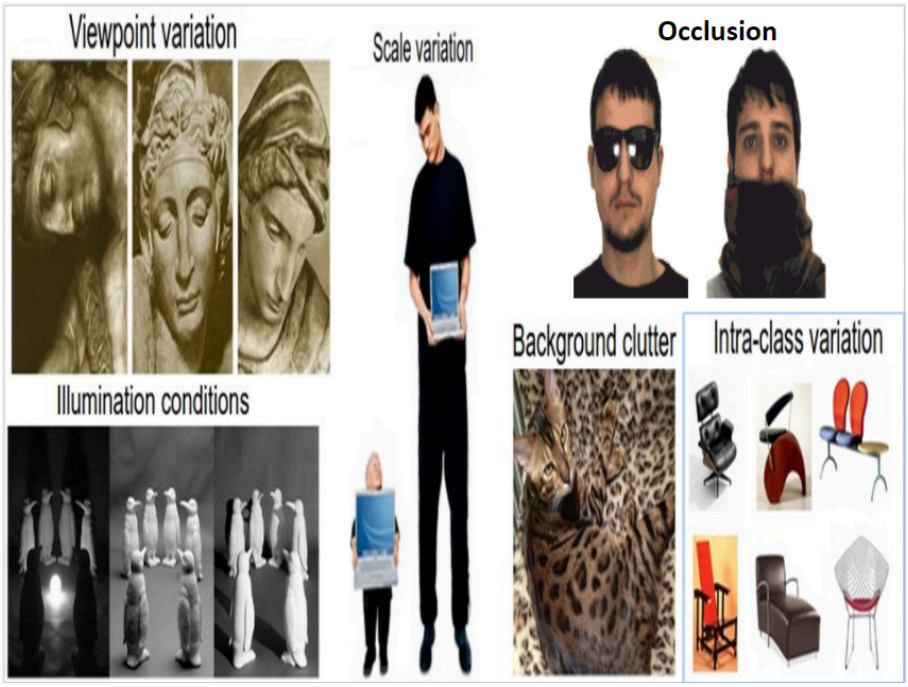
- shifting the image by some amount,
- cropping the image or
- any other operations which change the order of the pixels would result in the same problem.
- Need a neural network model which is not affected by the order in which the pixels are processed, but instead looks for the presence of certain pixels in the image.

Notes| Neural Networks

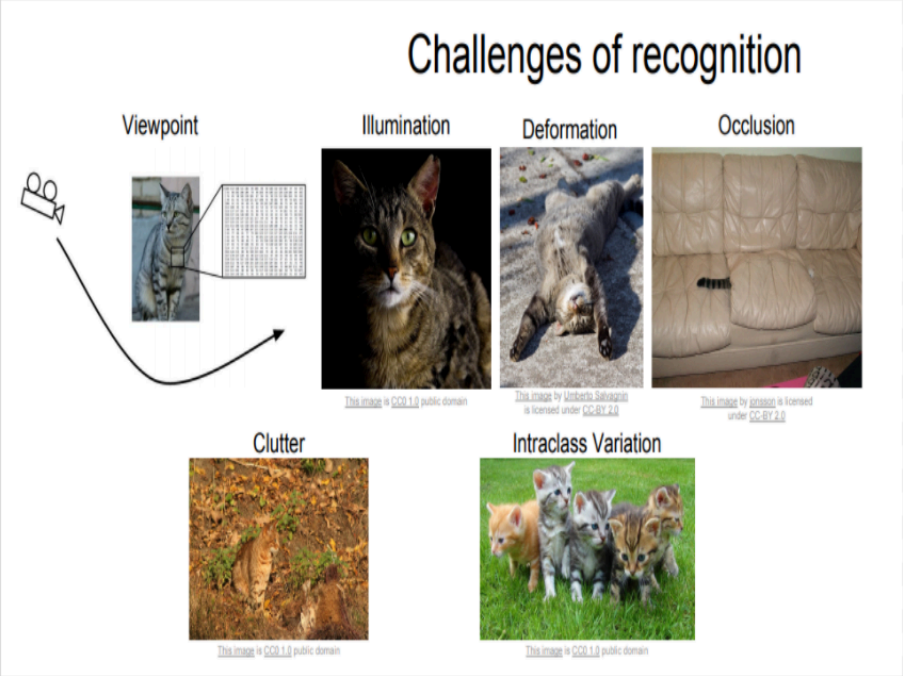
CNNs differ from MLP in:

1. SHARED-WEIGHTS ARCHITECTURE CNNs require Fewer parameters (weights and biases)
2. IMAGE INVARIANCE Characteristics.

✔Even when an image is rotated, sized differently or viewed in different illumination an object will be recognized as the same object –Image Invariance, which
✔do not depend on where the feature occurs in the image.
✔Can tolerate some distortion in the images.
✔Also known as shift invariant or space invariant ANNs (SIANN



Notes| Neural Networks



Historical Context and Motivation

CNNs were developed in the late 1980s and then forgotten about due to the lack of processing power.

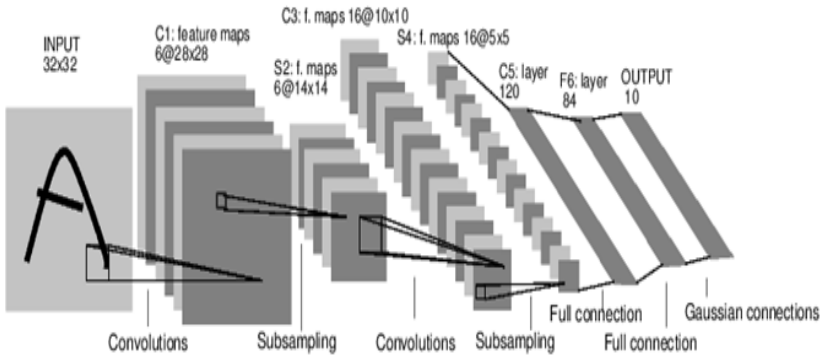
With advent of cheap, but powerful GPUs (graphics cards), research on CNNs and Deep Learning was given new life.

In 1995, Yann LeCun and Yoshua Bengio introduced the concept of CNNs.
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner.

Gradient-based learning applied to document recognition .*Proceedings of the*

IEEE, November 1998.

Notes| Neural Networks



A Full Convolutional Neural Network (LeNet)

1998 paper introduced famous LeNet-5 architecture widely used by banks to recognize handwritten check numbers was one of the very first convolution neural networks which helped propel the field of Deep Learning.

- Publication of the paper, "ImageNet Classification with Deep Convolutional Neural Networks" in 2012 by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton introducing AlexNet was a seismic shift that broke the Richter scale! As of 2018 it has been cited over 25,000 times.

- *The paper forged a new landscape in Computer Vision by demolishing old ideas in one masterful stroke.*

- The original paper's primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of GPUs during training

✔Some of the most important innovations have sprung from submissions by academics and industry leaders to the *ImageNet Large Scale Visual Recognition Challenge*, or *ILSVRC*--- called the Olympics for "seeing computers"

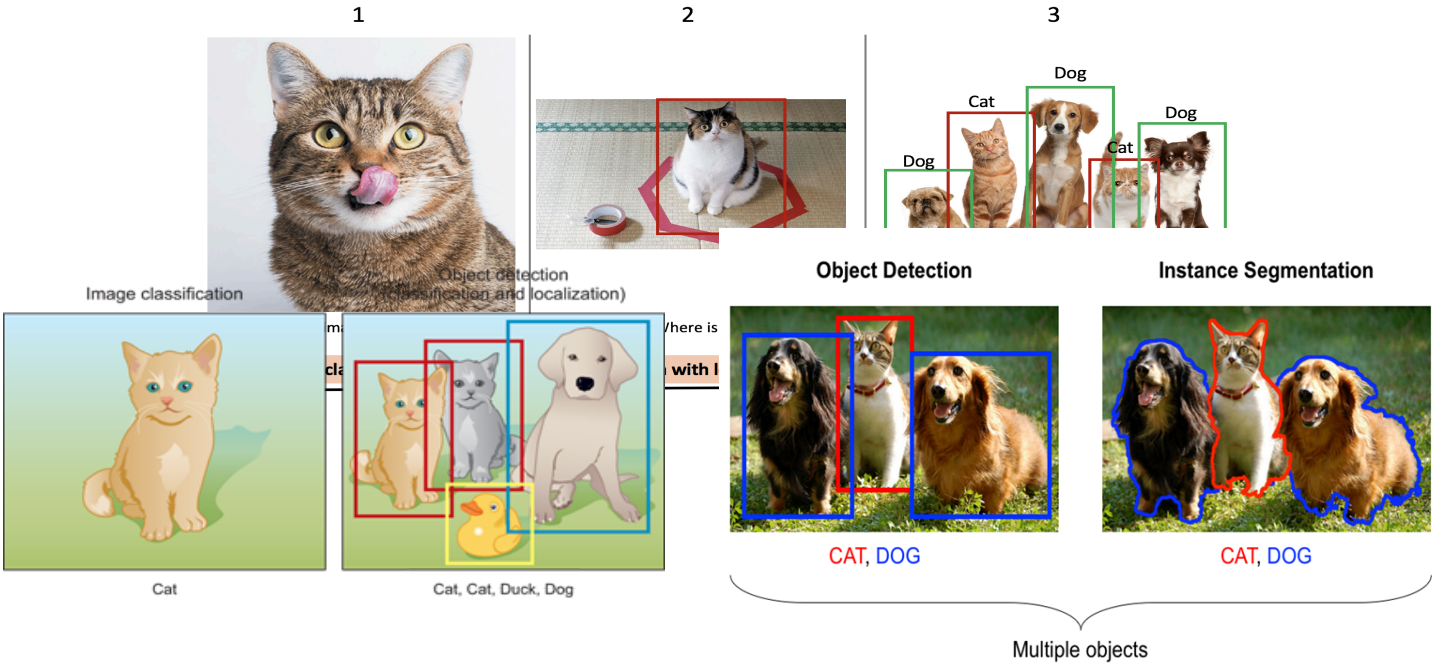
✔The ILSVRC is an annual computer vision competition developed upon a subset of a publicly available computer vision dataset called ImageNet.

✔As such, the tasks and even the challenge itself is often referred to as the

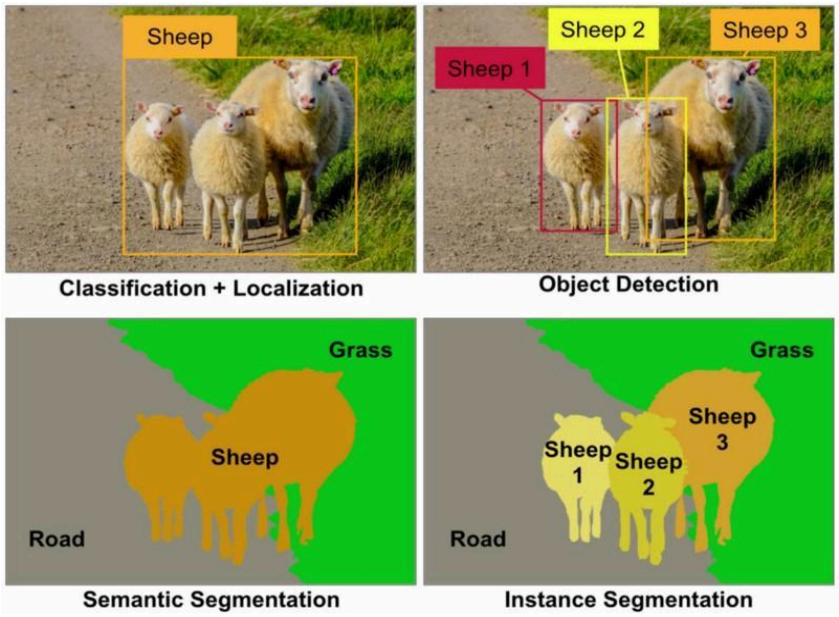
Notes| Neural Networks

ImageNet Competition.

The general challenge tasks for most years are as follows:
Image classification: Predict the classes of objects present in an image.
Single-object localization: Image classification + draw a bounding box around one example of each object present.
Object detection: Image classification + draw a bounding box around each object present.



Notes| Neural Networks



AlexNet has had a large impact on the field of machine learning, specifically in the application of deep learning to machine vision.

The paper used a CNN to get a Top-5 error rate (rate of not finding the true label of a given image among its top 5 predictions) of 15.3%. The next best result trailed far behind (26.2%).

Microsoft's ResNets achieved a top 5 error of 3.6% which is better than the human level performance on the same task.

First, you make a prediction using the CNN and obtain the predicted class ∑pclass=1

✔ Now, in the case of top-1 score, check if the top class (the one having the highest probability) is the same as the target label.

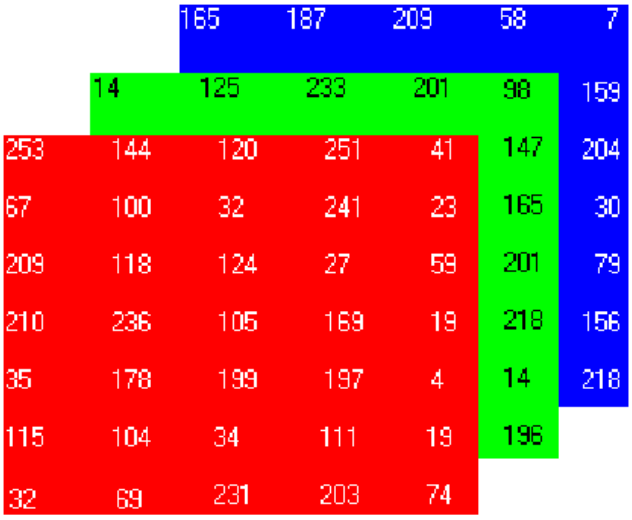
✔ In the case of top-5 score, check if the target label is one of top 5 predictions (the 5 ones with the highest probabilities).

✔ In both cases, the top score is computed as the times a predicted label matched the target label, divided by the number of data-points evaluated.

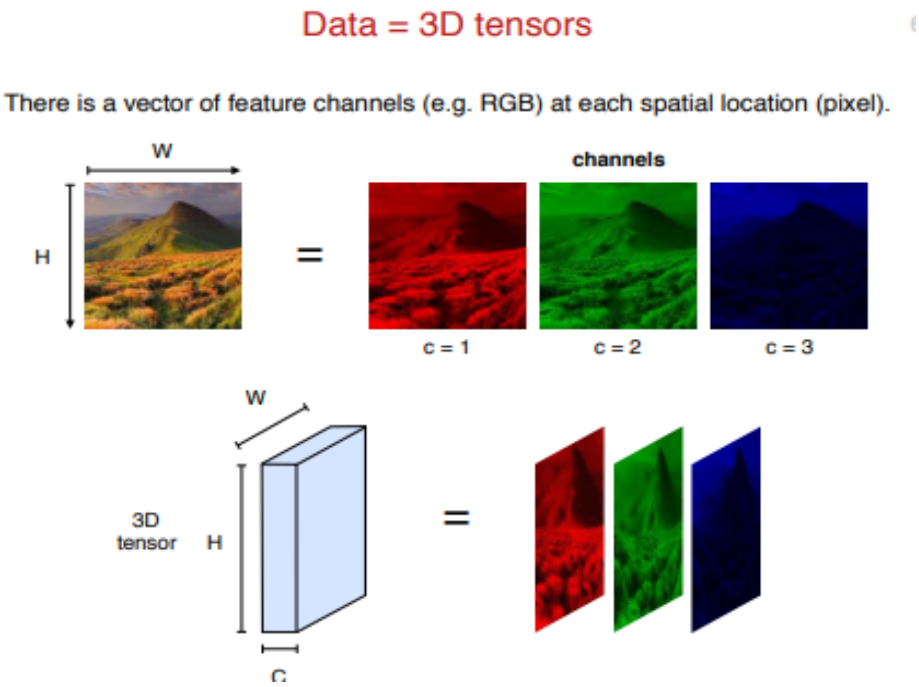
Notes| Neural Networks

What are Convolutional Neural Networks and how they work???

- CNNs are MLPs with a special structure.
- *CNN is a specialized kind of neural network for processing the data which has the grid like topology.*
- Examples of such data are :
- Images which can be considered as a 2D (224x224 grayscale) or 3-channel (coloured RGB) grid of pixels,
- Time series data which can be thought of as the 1D grid taking samples at regular time intervals



Notes| Neural Networks



Popular Databases

- CIFAR-10 (Canadian Institute For Advanced Research)dataset classification is a common benchmark problem in machine learning.
- The CIFAR10 comes bundled with Keras.
- It has 50,000 training images and 10,000 test images.
- The problem is to classify RGB 32x32 pixel images across 10 categories:
- *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.*

MNIST Database

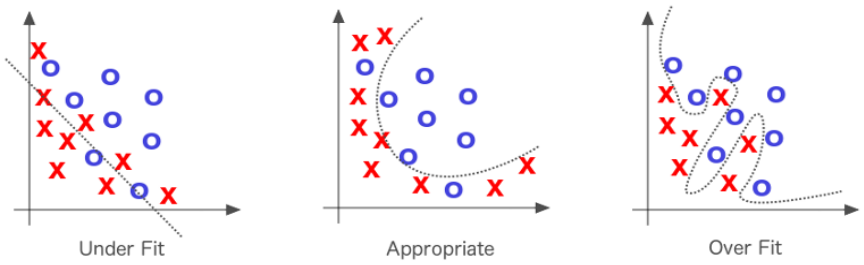
- The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems.
- The MNIST database contains 60,000 training images and 10,000 testing images.

ImageNet

Over 14 million URLs of images have been hand-annotated by ImageNet to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided.

Notes| Neural Networks

- ImageNet contains over 20 thousand categories, such as "balloon" or "strawberry“,contains several hundred images.

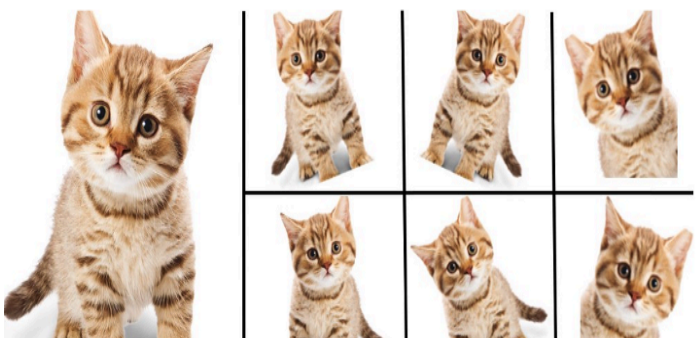


As model complexity increases , overfitting happens, generalization is poor

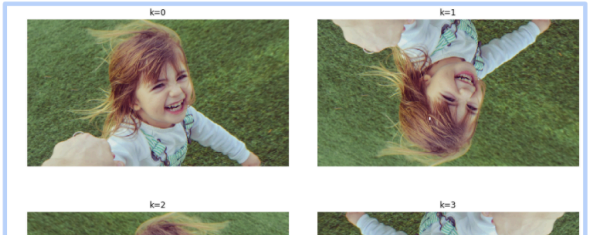
One of the major reasons for overfitting is that you don’t have enough data to train your network.
Apart from Regularization/Early stopping, another very effective way to counter Overfitting is Data Augmentation.
It is the process of artificially creating more images from the images you already have by changing the size, orientation etc of the image.
It can be a tedious task but fortunately, this can be done in Keras using the ImageDataGenerator instance.

Examples of transformations are:

✔Flipping the image horizontally / vertically
✔Rotating the image X degrees.
✔Trimming, expanding, resizing, …
✔Applying perspective deformations
✔Adjusting brightness, contrast, saturation, …
✔Adding some noise, defects, …
✔A combination of the above



Dihedral
It will rotate the image in all such possible orientations.



Notes| Neural Networks

CUTOUT REGULARISATION

A technique now commonly used on the very latest ImageNet models.

Cutout can be really seen as a form of augmenting the data to handle occlusion - an extremely common challenge in real-world applications, especially in the hot computer vision areas of robotics, surveillance, and self-driving cars.

By applying a form of occlusion to the training data, the network is effectively adapted to be more robust to it.

Main Layers used to build ConvNets

- Convolutional +ReLu Layer
- Pooling (Sub-sampling) Layer
- Fully-Connected followed by output layer (Classification)

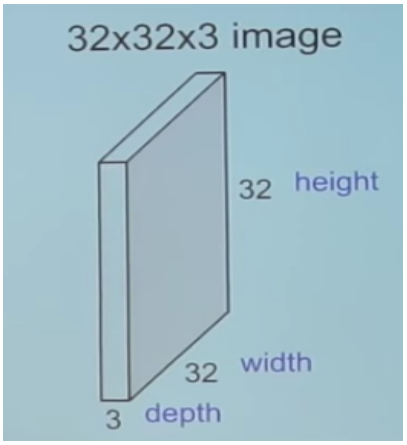
✔ *Convolutional Neural Networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of its*

Notes| Neural Networks

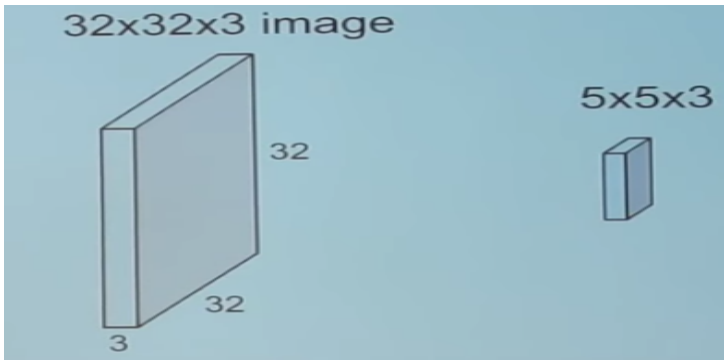
layers.

✔ The convolutional layer can be thought of as the eyes of the CNN.

✔ The layers of a CNN have neurons arranged in 3 dimensions: width, height and depth.



Unlike neural networks, where the input is a vector, in CNNs the input is a multi-channeled image (3 channeled in this case).

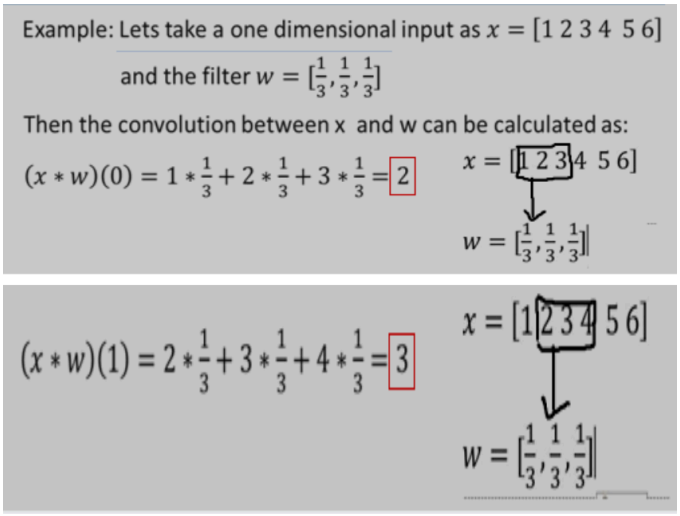


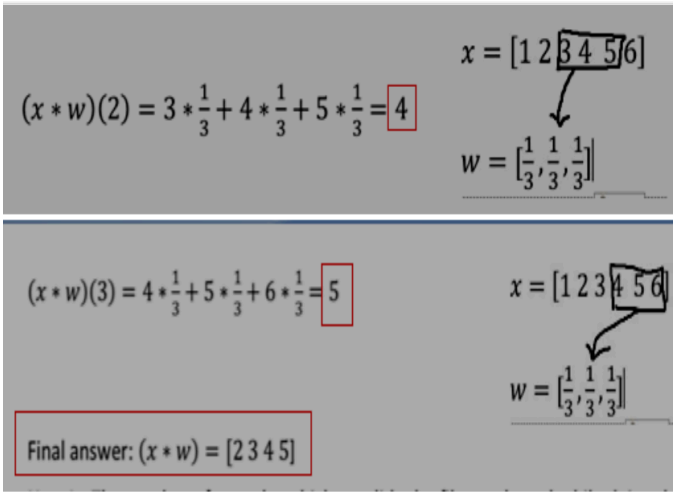
5*5*3 filter slides over the complete image and along the way it takes the dot product between the filter and chunks of the input image.

For every dot product taken, the result is a scalar.

CONVOLUTION OPERATION

Notes| Neural Networks

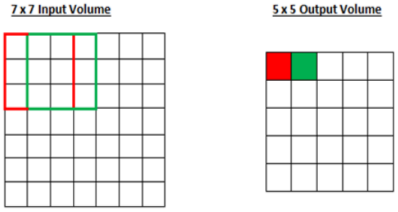




Stride: the number of pixels you skip while traversing the input image horizontally and vertically during convolution after each element-wise multiplication of the input weights with those in the filter.

It is used to decrease the input image size considerably

$$O = \frac{W - F + 2P}{S} + 1 = \frac{7 - 3 + 2 \times 0}{1} + 1 = 5$$



F is Filter size (spatial dimension)

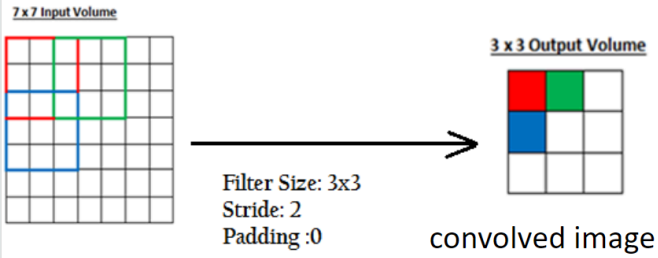
P is Padding

S is Stride

W =H is input height/length

O is the output height/length

$$O = \frac{W - F + 2P}{S} + 1 = \frac{7 - 3 + 2 \times 0}{2} + 1 = 3$$



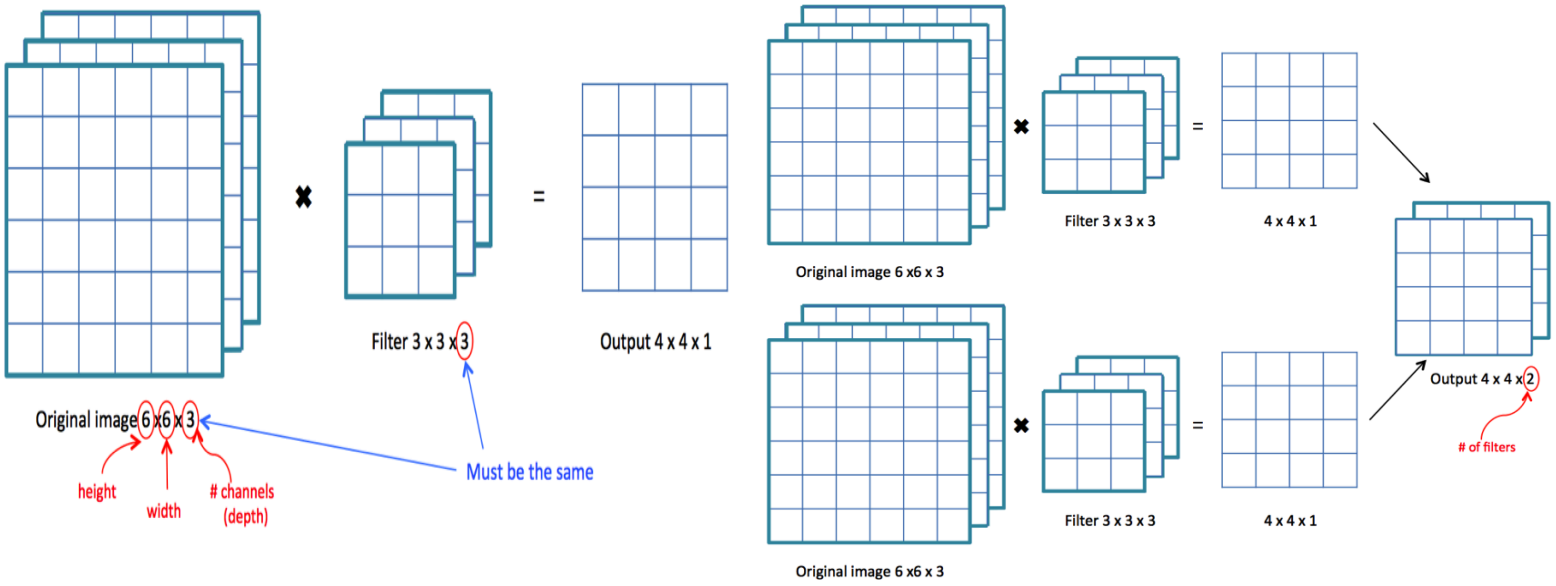
stride value is increased if you want receptive fields to overlap less and want smaller spatial dimensions.

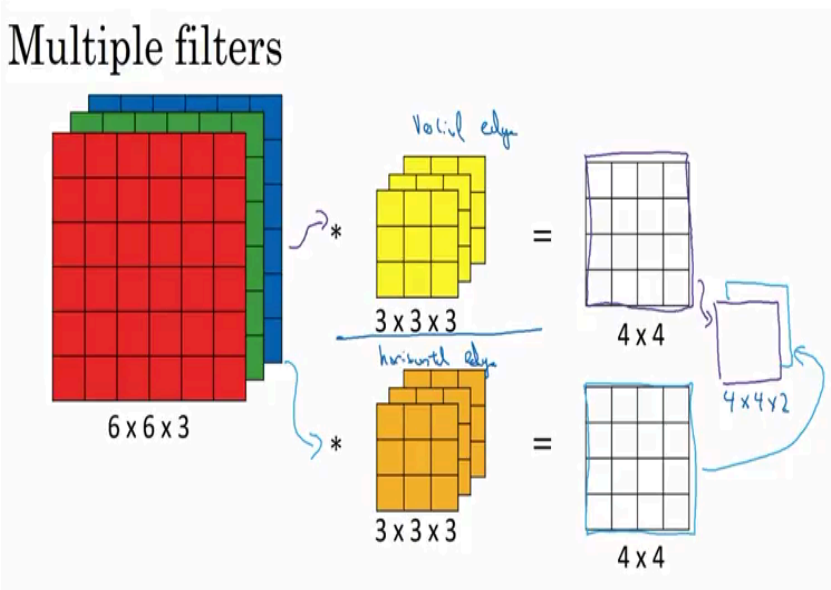
Stride is normally set in a way so that the output volume is an integer and not a fraction.

- Spatial dimensions decrease when we apply convolution. As we keep applying conv layers, the size of the volume will decrease faster.

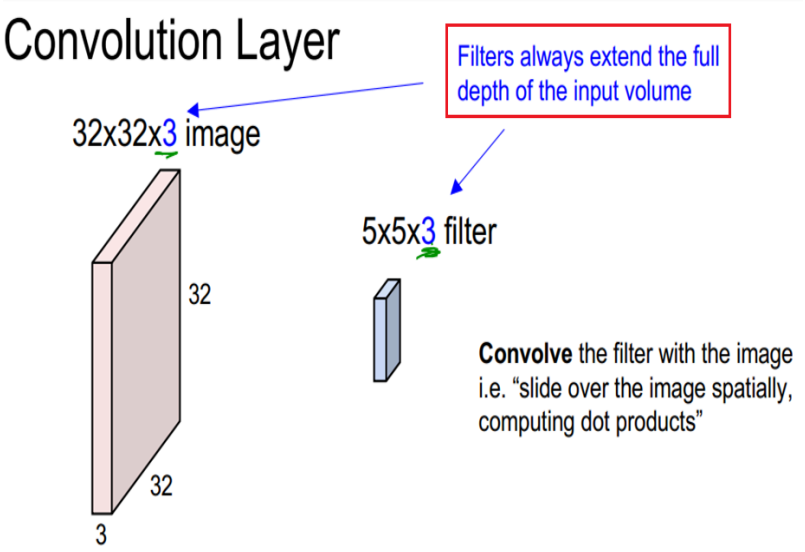
Notes| Neural Networks

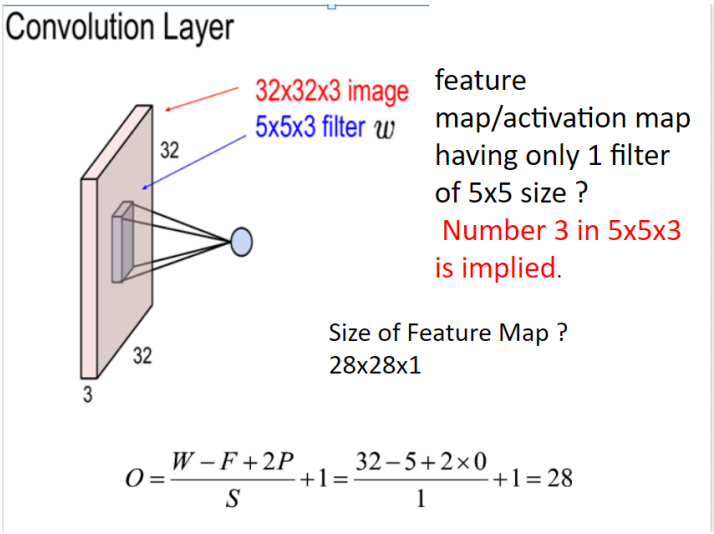
- In early layers of our network, as much information about the original input volume is to be preserved to extract low level features to keep our volume same we apply padding
- Padding is generally used to add columns and rows of zeros to keep the spatial sizes constant after convolution, doing this might improve performance as it retains the information at the borders.
- Parameters for the padding function in Keras are *Same*- output size is the same as input size by padding evenly left and right, but if the amount of columns to be added is odd, it will add the extra column to the right.

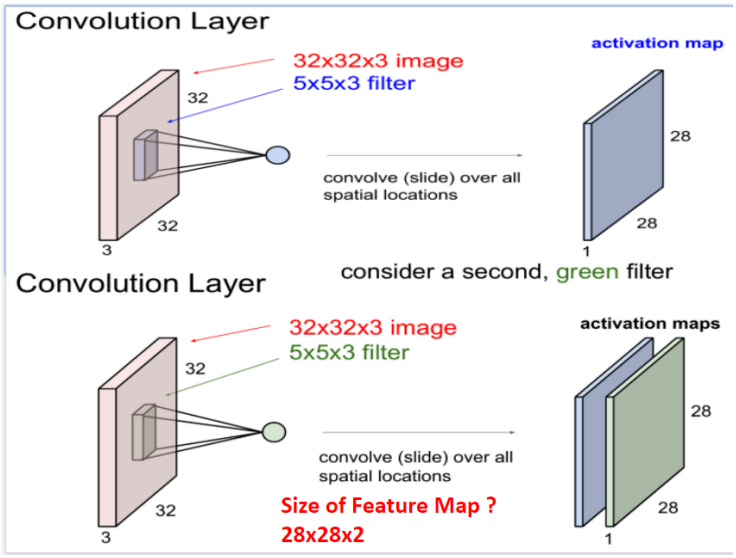


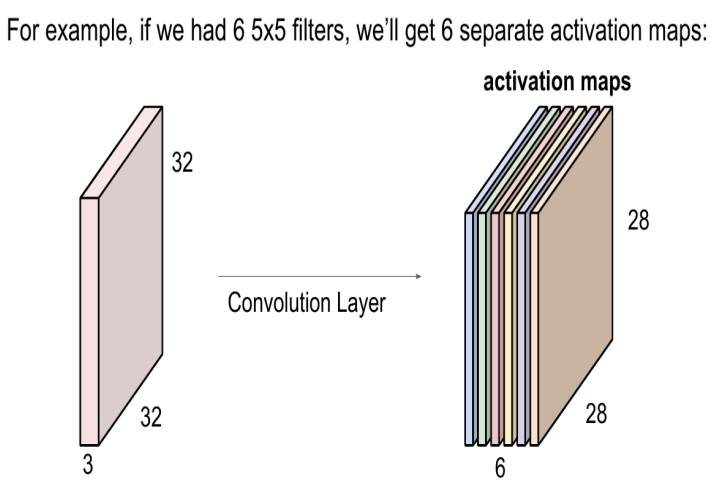


Notes| Neural Networks









✔A filter applied to an input image or input feature map always results in a single number. T/F

✔The systematic left-to-right and top-to-bottom application of the filter to the input results in a two-dimensional feature map. T/F

✔One filter creates one corresponding feature map.

Notes| Neural Networks

✔A filter must have the same depth or number of channels as the input, yet, regardless of the depth of the input and the filter, the resulting output is a single number and one filter creates a feature map with a single channel.

If the input has one channel such as a grayscale image, then a 3×3 filter will be applied in 3x3x1 blocks.

If the input image has three channels for red, green, and blue, then a 3×3 filter will be applied in 3x3x3 blocks.

If the input is a block of feature maps from another convolutional or pooling layer and has the depth of 64, then the 3×3 filter will be applied in 3x3x64 blocks to create the single values to make up the single output feature map.

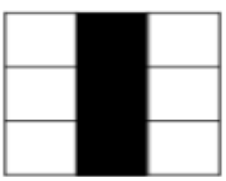
Number of Channels:

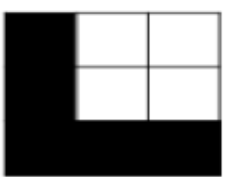
- *It is the equal to the number of color channels for the input* but
- in later stages it is equal to the number of filters we use for the convolution operation.
- More the number of filters used, more are the features learnt

✔The CNNs have several different filters/kernels consisting of trainable parameters which can convolve on a given image spatially to detect features like edges and shapes.

✔This high number of filters learn to capture spatial features from the image based on the learned weights through back propagation

Filters are nothing more than weights, which get updated.
Filters are chosen only in the sense of specifying how many filters [*hyperparameter*] at each level of the CNN.
Backprop will determine the weights of the filters

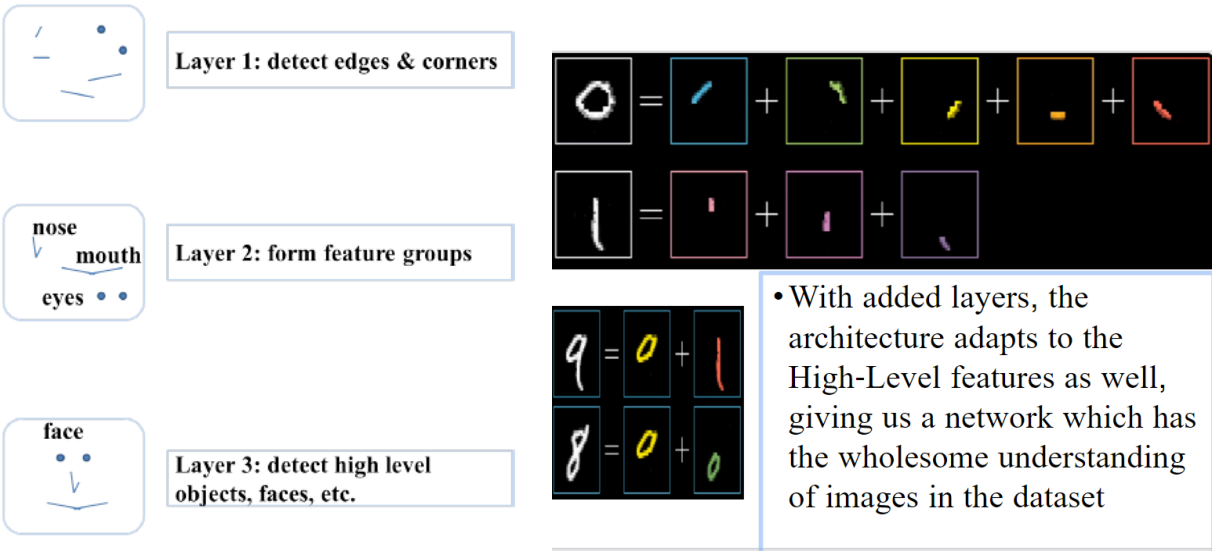




The filter on the left might activate strongest when it encounters a horizontal

Notes| Neural Networks

line the one in the middle for a vertical line in the right for edge



Smaller filters collect local information, bigger filters collect global, high-level information.

In general, we use filters with odd sizes

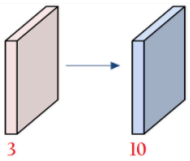
The ultimate genius of CNN is the network's ability to automatically discover and learn its own discriminative features.

The downside of CNNS is the large amounts of training samples to give a better & more precise output

Larger kernel is preferred for information that is distributed more globally
Smaller kernel is preferred for information that is distributed more locally.

Input volume: 32x32x3
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



No. of Parameters = (F x F x in-depth) x number of filters
= (5x5x3)x10 =750

No. of parameters =
Input-depth x output-depth(=no. of filters) x Filter size
= 3 x 10 x 5 x 5 =750

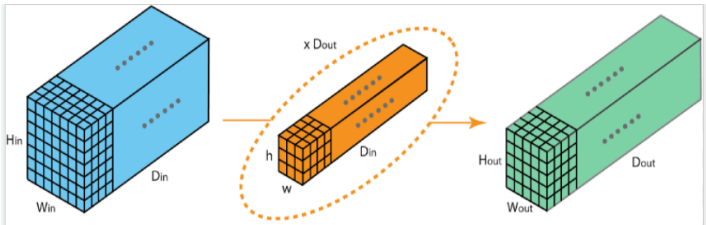
If bias is there(one bias for each filter)
No. of Parameters = [F x F x in-depth + 1] x number of filters
= [5x5x3 + 1]x10 =760

No. of parameters
=(In-depth x Filter size + 1)x out-depth (no. of filters)
= (3 x 5 x 5 + 1) x10 = 760

Notes| Neural Networks

Hyperparameters controlling the size of the output volume –
-Depth: no of filters, each learning to look for something different in the input.
-Size of filter/kernel
– the stride with which we slide the filter.

– padding the input volume with zeros around the border.



TEST PROBLEM
If all volumes of CNN have C channels(filters)
D_{in} =C
D_{out} =C

(i) number of parameters required to be trained for **single 7x7 conv layer** ?D_{in} x7x7xD_{out}
= **C(depth)x7x7xC(no. of filters=channels)**
= **$49C^2$**

(ii) number of parameters required to be trained for **three 3x3 conv layers**

1 conv layer= (C(depth)x3x3xC(no. of filters)) = $9C^2$
3 conv layers = $3x9C^2 = 27C^2$

Stacking CON layers with tiny filters as opposed to having on CON layer with big filters allows us to express more powerful features of the input, with fewer parameters.

Example: Input volume of 32x32x128 convolved with 32 conv filter (1x1) x128 is implied.
Output Volume/feature map ?

Solution: 32x32x32
shrunk the number of channels from 128 to 32.

1x1 convolution alters the depth of the input volume(*by choosing filters more or less than the depth of input volume)*

Notes| Neural Networks

What does a 1x1 convolutional layer do?

A 1x1 convolution operation without padding and with Stride of 1 ALTERS the depth dimension of an input volume (W x H x D) leaving its width and height intact [[(W-1) /1] +1=W]

n x n convolution matrix, not only squashes the depth dimension of an input volume , but -alters the width and height of the input volume too, depending on Filter size, padding and stride.

1x1 convolution alters the depth of the input volume(*by choosing filters more or less than the depth of input volume),*

Passes the input volume through a non-linearity [Applying ReLU]

(if the number of 1x1 filters is kept same as input volume, then all it does is add a nonlinearity(Relu).

Which of the two below is computationally less costly ?

(i)input (256 depth) -> 1x1 convolution (64 filters/depth) -> 4x4 convolution (256 filters/depth)

(ii)input(256 depth)->4x4 Convolution(256 filters/depth)

Assume that the computation cost is proportional to the number of weights.

case (i) **num_weights=in_depth x out_depth x kernel_size**
= 256x1x1x64 + 64x4x4x256= 2,78,528

the num_weights = in_depth x out_depth x kernel_size
= 256x4x4x256 =10,48,576

The bottom one is about ~3.7x slower.

Notes| Neural Networks

✔1x1 and 3x3 convolutional Kernels are widely used for their benfit of reducing computations, pruning parameters and improving accuracies

✔3x3 are the smallest filters that can capture the notion of left/right, up/down, and centre.

✔Although the receptive fields of 3x3 filters are small, a few continuous 3x3 layers can get the same receptive field of bigger filters.

To summarize, the Conv Layer:

- Accepts a volume of size $W_1xH_1xD_1$
- Requires four hyperparameters:
    1. Number of filters (K) =depth of output image D_2
    2. Spatial extent of Filters (FxF)
    3. Stride of filter(S)
    4. Amount of zero padding (P)
- Produces output of volume $W_2xH_2xD_2$

- Padding = Same: means the input image ought to have zero padding so that the output in convolution doesnt differ in size as input.
- we add a n-pixel borders of zero to tell that 'dont' reduce the dimension and have same as input.

- Padding = Valid: means we dont add the zero pixel padding around the input matrix, and its like saying, we are ready to loose some information.

- when we do valid padding, the convolution happens and based on kernel size we take, only the inside pixel of input is considered to be 'valid' which appears in the feature map.

Notes| Neural Networks

Padding Type	Description	Impact
Valid	No padding	Dimensions reduce
Same	Zeros around the edges	Dimensions stay the same

POOLING

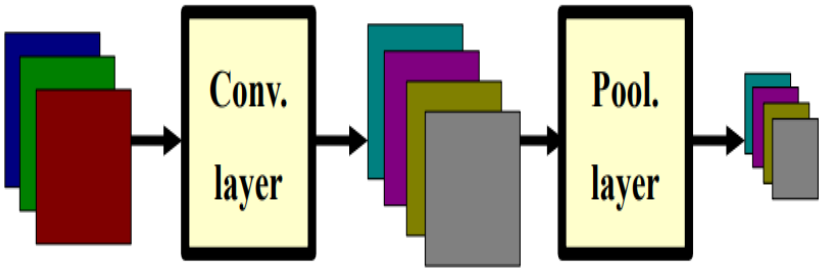
Outputs of several nearby feature detectors are combined into a local or global 'bag of features', in a way that preserves task-related information while removing irrelevant details.

Pooling is used to achieve

- ✔ Invariance to image transformations,
- ✔ More compact representations, and
- ✔ Better robustness to noise and clutter.

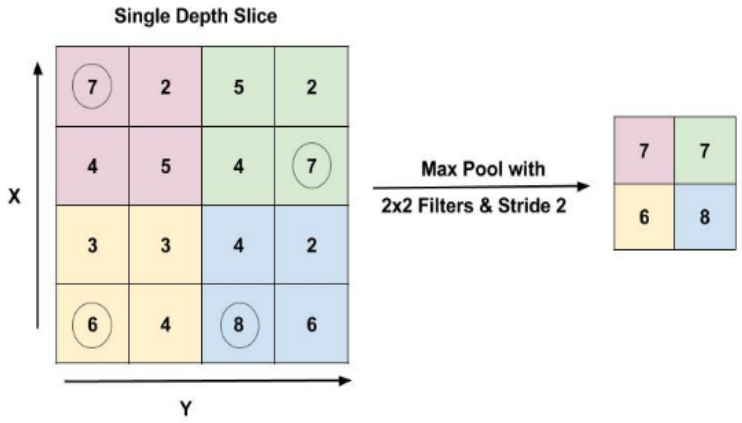
- Pooling layer is also called subsampling or downsampling
- Pooling is mostly used immediately after the convolutional layer to reduce the spatial size but retains the most important information (depth).
- Only width and height are changed not depth.
- Pooling can be of different types: Max, Average, Sum etc.

Notes| Neural Networks



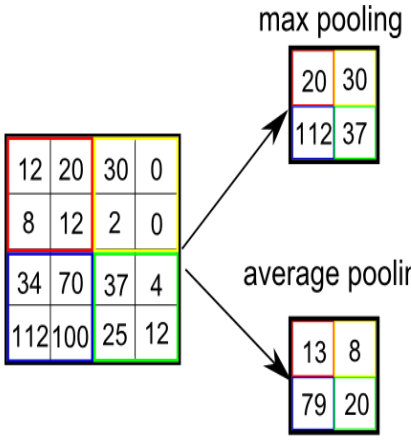
Observe number of channels before and after pooling
After Pooling, Number of channels in output will remain the same (depth of the output will remain the same as input).

In case of Max Pooling, define a window and take the largest element from the feature map within that window.



- for input volume of height h, width w and depth d, on applying max pooling with a filter of size 2x2 and stride of 2, an output volume of size [(h/2)x(w/2)]x d would result.

- For each channel apply the max pooling and then stack together all the channels.



- Average Pooling performs the average over all the pixel values for a particular portion of the image

Pooling Operation Introduces how many

Notes| Neural Networks

parameters ??

Zero parameters since it simply computes a fixed function of the input

Hyperparameters : Stride, Filter size, avg/max pooling

Advantages of pooling

- makes the input representations (feature dimension) smaller and more manageable
- reduces the number of parameters and computations in the network, therefore, controls overfitting
- makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling – since we take the maximum / average value in a local neighborhood).
- Even with a tiny kernel of 2x2 and a stride of 2, output will be two times smaller in both width and height, so area is 4 times smaller.
- Average pooling retains more information, in comparison to max pooling which simply drops 75% of input values
- Max pooling performs better, when the background is dark, as in MNIST dataset, the digits are represented in white color(value 255) and the background is black(value 0). So, max pooling is used.
- Max pooling preserves only the strongest features,

Notes| Neural Networks

✔Batch normalization - A technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch.

✔Training deep neural networks with tens of layers is challenging as they can be sensitive to the initial random weights and configuration of the learning algorithm.

✔the distribution of the inputs to layers deep in the network may change after each mini-batch when the weights are updated.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Batch Norm before activation or after the activation?

✔the original paper talks about applying batch norm just before the activation function, it has been found that applying batch norm after the activation yields better results.

✔This seems to make sense, as if we were to put an activation after batch norm, then the batch norm layer cannot fully control the statistics of the input going into the next layer since the output of the batch norm layer has to go through an activation.

✔*An exception is for the sigmoid activation function wherein you need to place the batch normalization layer before the activation to ensure that the values lie in the*

Notes| Neural Networks

linear region of sigmoid before the function is applied.

Principles/Conventions to build a CNN architecture

✔ Always start by using smaller filters is to collect as much local information as possible, and then gradually increase the filter width to reduce the generated feature space width to represent more global, high-level and representative information

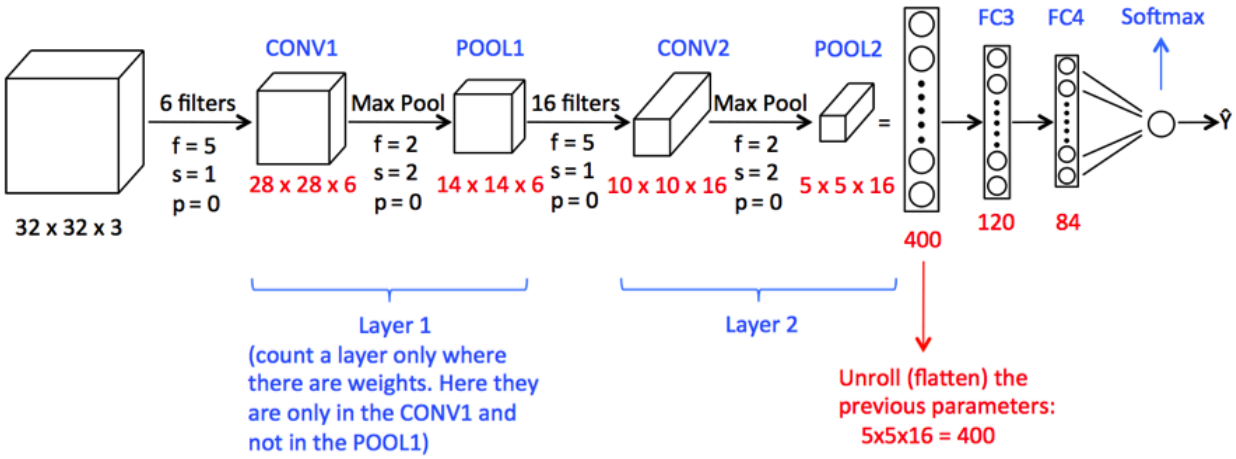
✔the number of channels should be low in the beginning such that it detects low-level features which are combined to form many complex shapes(by increasing the number of channels) which help distinguish between classes.

✔The number of filters is increased to increase the depth of the feature space thus helping in learning more levels of global abstract structures.

✔*By convention the number of channels generally increase or stay the same while we progress through layers in our convolutional neural net architecture*

✔ General filter sizes used are 3x3, 5x5 and 7x7 for the convolutional layer for a moderate or small-sized images

✔for Max-Pooling use 2x2 or 3x3 filter sizes with a stride of 2. Larger filter sizes and strides may be used to shrink a large image to a moderate size



Notes| Neural Networks

FLATTEN THE FEATURE MAP

✔To finally classify the image into a category, say cat or dog, set up a Multi-Layer Perceptron on top of the last convolution layer.

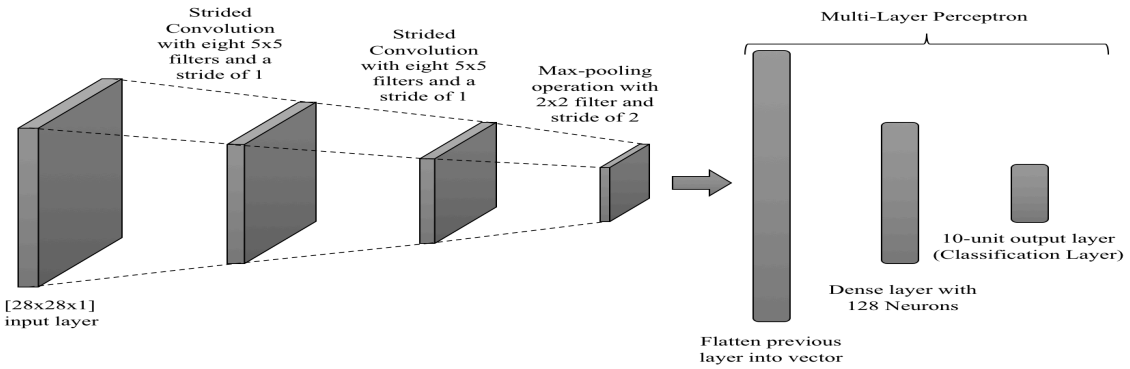
✔Feeding a MLP requires input vectors (one-dimension arrays), So "flatten" the output feature map.

- ✔ Fully Connected layer
- ✔ Every neuron in the previous layer is connected to every neuron on the next layer like normal FFNN.
- ✔ Output from the convolution and pooling layers represent high-level features of the input image.
- ✔ Purpose of the Fully Connected layer is to use features extracted from Conv+Pooling layers for classifying the input image into various classes using SOFTMAX function

✔pass 128 nxn filters[S=1,P=0] over an image of dimensions nxn. Output/result?
✔ a vector of length 128.

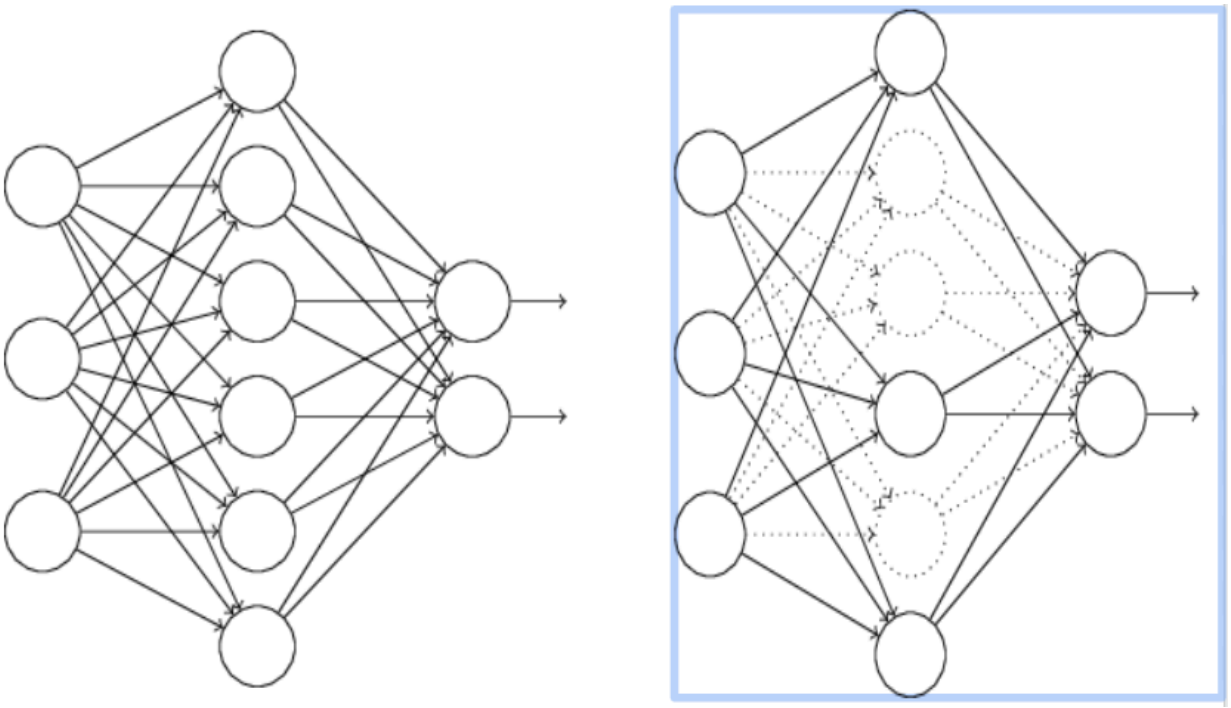
✔pass 16 filters of 5x5[S=1,P=0], over 5x5x16(output of conv layer), output map will have vector length of??
✔ 16

✔If you pass 400 filters of 5x5, over 5x5x16,
output map will have vector length of
✔400



Notes| Neural Networks

DROPOUT: Unlike L1 and L2 regularization, dropout doesn't modify the cost function.
Instead, it modifies the network itself.



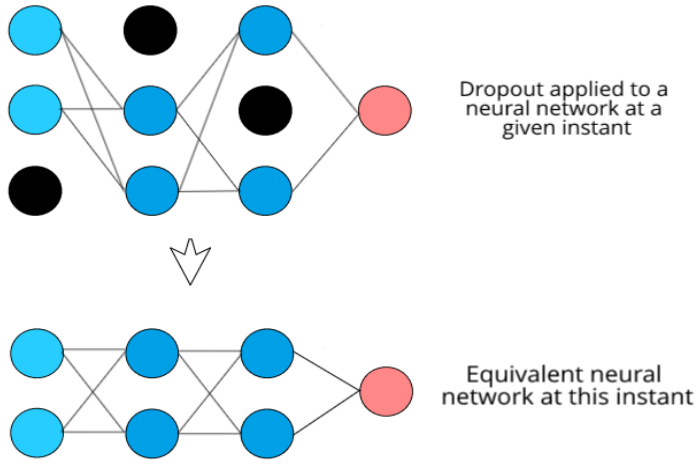
*A related heuristic explanation for dropout is given in one of the earliest papers to use the technique***ImageNet Classification with Deep Convolutional Neural Networks*, by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton (2012)*

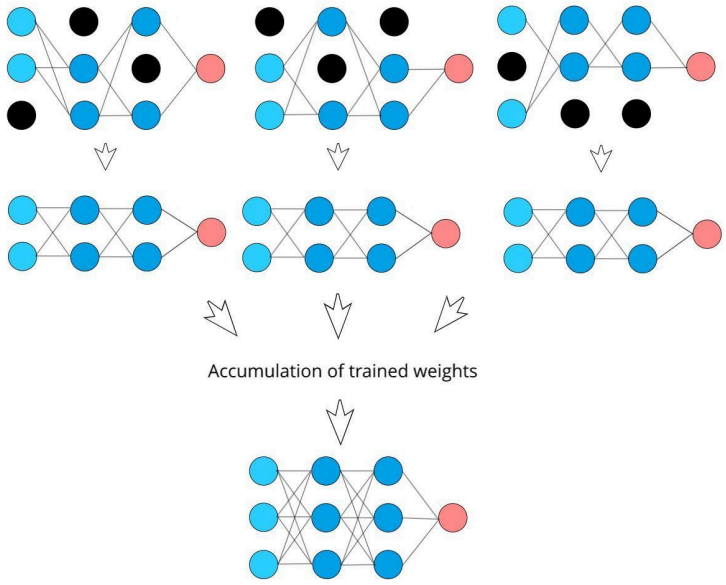
"This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons."

we can think of dropout as a way of making sure that the model is robust to the loss of any individual piece of evidence.

In this, it's somewhat similar to L1 and L2 regularization, which tend to reduce

weights, and thus make the network more robust to losing any individual connection in the network.





Notes| Neural Networks

✔What could go wrong?

✔If you turn off too many neurons, then remaining neurons simply will not be able to cope with their work and the result will only get worse.

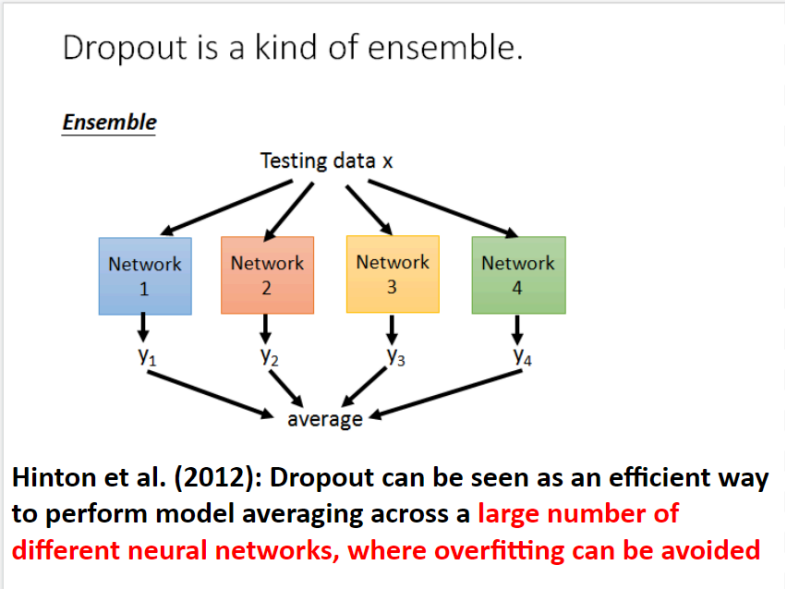
✔It can also be thought of as an ensemble technique in machine learning.

✔Remember, that the ensemble of strong-learners performs better than a single model as they capture more randomness and are less prone to overfitting.

✔But an ensemble of weak-learners more prone to retraining than the original model.

✔As a result, dropout takes place only with huge neural networks.

- ✔ if neurons are randomly dropped out of the network during training, other neurons will have to step in and handle the representation required to make predictions for the missing neurons.
- ✔ This results in multiple independent internal representations being learnt by the network.
- ✔ The effect is that the network becomes less sensitive to the specific weights of neurons.
- ✔ This in turn results in a network that is capable of better generalization and is less likely to overfit the training data.



Hinton et al. (2012): Dropout can be seen as an efficient way to perform model averaging across a large number of different neural networks, where overfitting can be avoided

Notes| Neural Networks

Transfer Learning

Why train when you can finetune ?

Notes| Neural Networks

Notes| Neural Networks

Notes| Neural Networks

Notes| Neural Networks

Notes| Neural Networks

Notes| Neural Networks

Notes| Fuzzy Logic

Fuzzy Logic Basics

"Fuzzy" means vague/ blurry.

Fuzzy Logic: A form of knowledge representation in vague linguistic terms (for phenomena which are difficult to describe mathematically)

We need a generalization of Boolean logic to handle *partial truth* values (i.e, between 0.0 to 1.0).

Example:

Age less than 40 years → YOUNG

Notes| Neural Networks

Age more than 40 years → OLD

However you don't change from young to old once you become 40 i.e, all the boundaries are blurred.

Let's consider a machine which takes inputs and outputs the required using fuzzy logic.

We