UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ECE 457B - My Course Notes: Intro. to Computer Intelligence

Prepared by:

Jianxiang (Jack) Xu []

12 April 2021

Table of Contents

Chapter 1

Introduction of Intelligent Computing

1 Fundamentals of Computational Intelligence

1.1 Introduction

Chapter 2

ML and ANN

1 Fundamentals of Machine Learning and Connectionist Modeling

1.1 AI/ML/DL

1.1.1 Intro.

Humans have developed sensory systems to perceive their environments and make decision based on what they observe, for example:

This is greatly influenced by how we perceive the patterns in our environment.

Humans have always strived to impart machines with similar capabilities as theirs for faster task execution, better predictability and higher accuracy

1.1.2 AI/ML Transforming The World

Remark 2.1.1: Why

Reason for the hugely growing role of AI/ML is the tremendous opportunities for economic development:

According to Deloitte and PwC GDP Will Set to Grow Globally by up to 14% by 2030: +15.7 trillion to the global GDP

Unthinkable only 5 to 6 years ago

Remark 2.1.2: How

Big Data and AI/ML are fueling huge growth in major innovative breakthroughs in various fields

Definition 2.1.1: AI

the effort to automate intellectual tasks normally performed by humans

Definition 2.1.2: ML

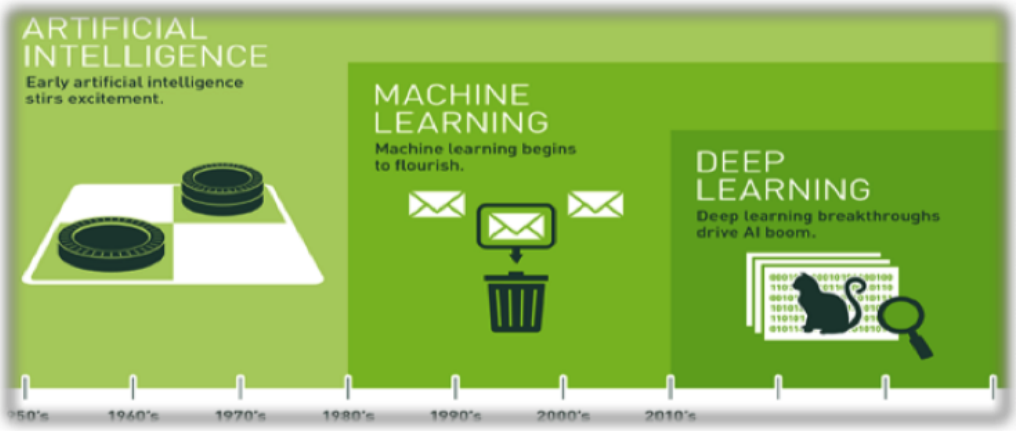
an algorithm to discover data representation rules (models) for prediction/classification

Definition 2.1.3: Deep Learning

special type of ML algorithms involving hierarchical and multi-layered data representations

Alert 2.1.1: NOTE

AI includes Machine Learning, which in turn includes Deep Learning!



1.1.3 Recent Breakthroughs

Example 2.1.1: IBM: Cat-brain

IBM's Cat-brain project is a chip like brain (SyNAPSE) to capture surrounding information in real time. Simulated the cat brain cortex with 147,456 cores and 144TB of memory: basic synaptic circuit for the brain chip.
Current work on 10^9 artificial neurons and 10^14 synapses

Example 2.1.2: Google: DeepMind

Google's DeepMind project, aims at formalizing intelligence to implement in machines and to understand the human brain. Uses deep learning technologies.
On March 2016, AlphaGo beat the top player in the world (Lee Sedol) in the Go game (Dan 9)

Example 2.1.3: IBM: Watson

IBM's Watson, QA system is used as knowledge repository, to answer virtually any inquiry (in Wikipedia and other knowledge repositories). Uses NLU and ML technologies. Has access to more than 10^9 pages content.
Won Jeopardy! Contest against top players in the world. Project Intu will equip devices and machines with Watson's powerful AI tools, hence providing them with cognitive capabilities and make them aware of their environments (ZDNET, Nov. 2016)

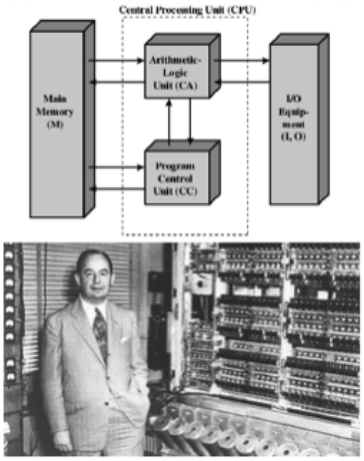
Example 2.1.4: Amazon, Google and Netflix: Big Data

Amazon, Google and Netflix are using Big Data analytics based on Deep Learning to target consumer behavior in various areas including entertainment, shopping and provide adequate recommendations to the user

1.1.4 Origin

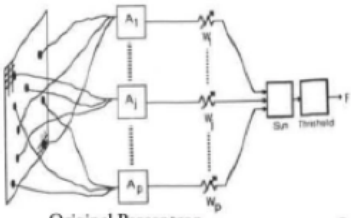
Alert 2.1.2: Origins of Modern Computing and Machine Learning

Von Neuman Computer



Father of Modern Computing

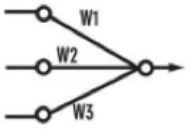
Rosenblatt Perceptron



Original Perceptron

(From Perceptrons by M. L Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press.

Frank Rosenblatt (1928-1971)



Simplified model:

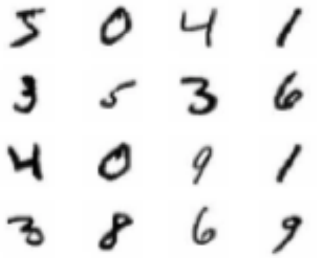
Father of Connectionist Computing

1.2 Pattern

Remark 2.1.3: Notion of a Pattern

A pattern is an abstract entity that describes a physical object such as:

- Speech Signal
- Handwritten words or digits
- Human face
- DNA sequence
- Weather information



Definition 2.1.4: Pattern Recognition

Pattern recognition (PR) is the study of how machines:

- Observe the environment
- Learn to distinguish patterns of interest
- Make correct decisions about the
- categories of the patterns

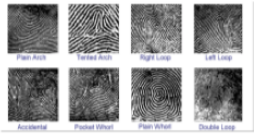
PR involves theory, algorithms, and systems that can assign patterns into categories. More theoretically inclined vs machine learning, which is more practically oriented.

Remark 2.1.4: PR Applications

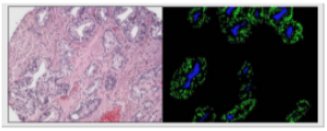
- License Plate Recognition
- Cancer Detection
- Fingerprint Recognition
- Autonomous Navigation

License Plate Recognition

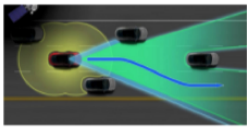
Fingerprint Recognition



Cancer Detection



Autonomous Navigation



Algorithm 2.1.1: PR Major Approaches

- Methods for Regression (Continuous Models):
  - Least Square
  - Ridge Regression
  - The Lasso Approach
- Method for Classification (Discrete Modeling):
  - Linear and Quadratic Discriminant Analysis
  - Decision Trees
  - Rosenblatt's Perceptron
  - Support Vector Machines

Remark 2.1.5: PR Main Challenges

- Wide variability of the same digits and objects.
- How to differentiate between "6" and "0"? it quickly becomes quite complicated to compile a list of heuristics.
- If we have more than 5 objects? How can we solve this problem?
- Nearly impossible to create handcrafted rules to understand each pixel.

Objective: design a machine that learns patterns from data! Solution: ML! Definition 2.1.2 Definition 2.1.5

1.3 Machine Learning

Definition 2.1.5: Machine Learning

An area of artificial intelligence (AI) that provides computers with the ability to learn from data while discovering patterns. It doesn't't follow explicitly programmed instructions.

Remark 2.1.6

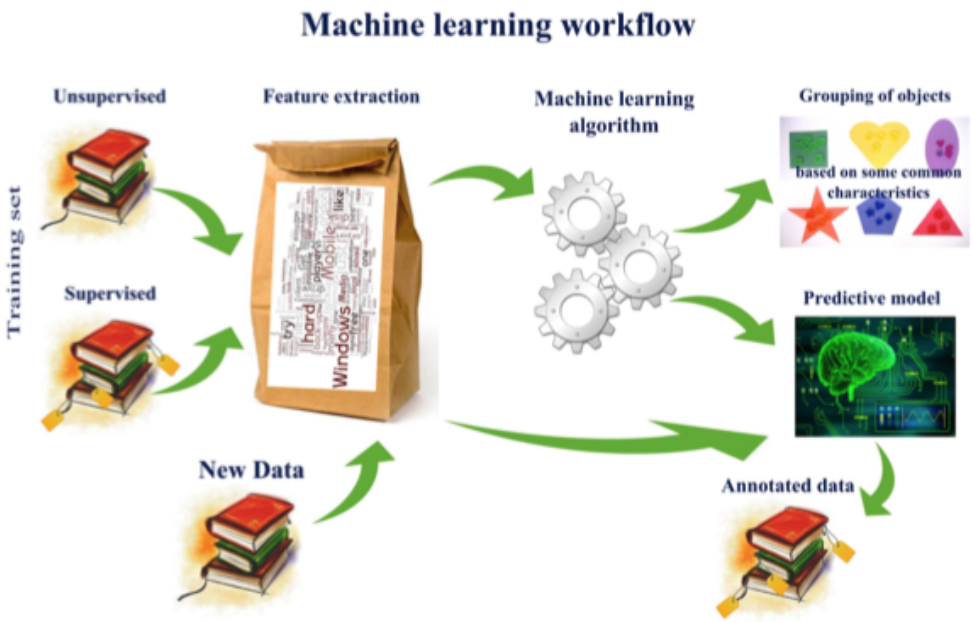
It can be applied in situations where it is very challenging or impossible to define heuristic/manual rules e.g., Face detection, Speech recognition, Stock prediction.

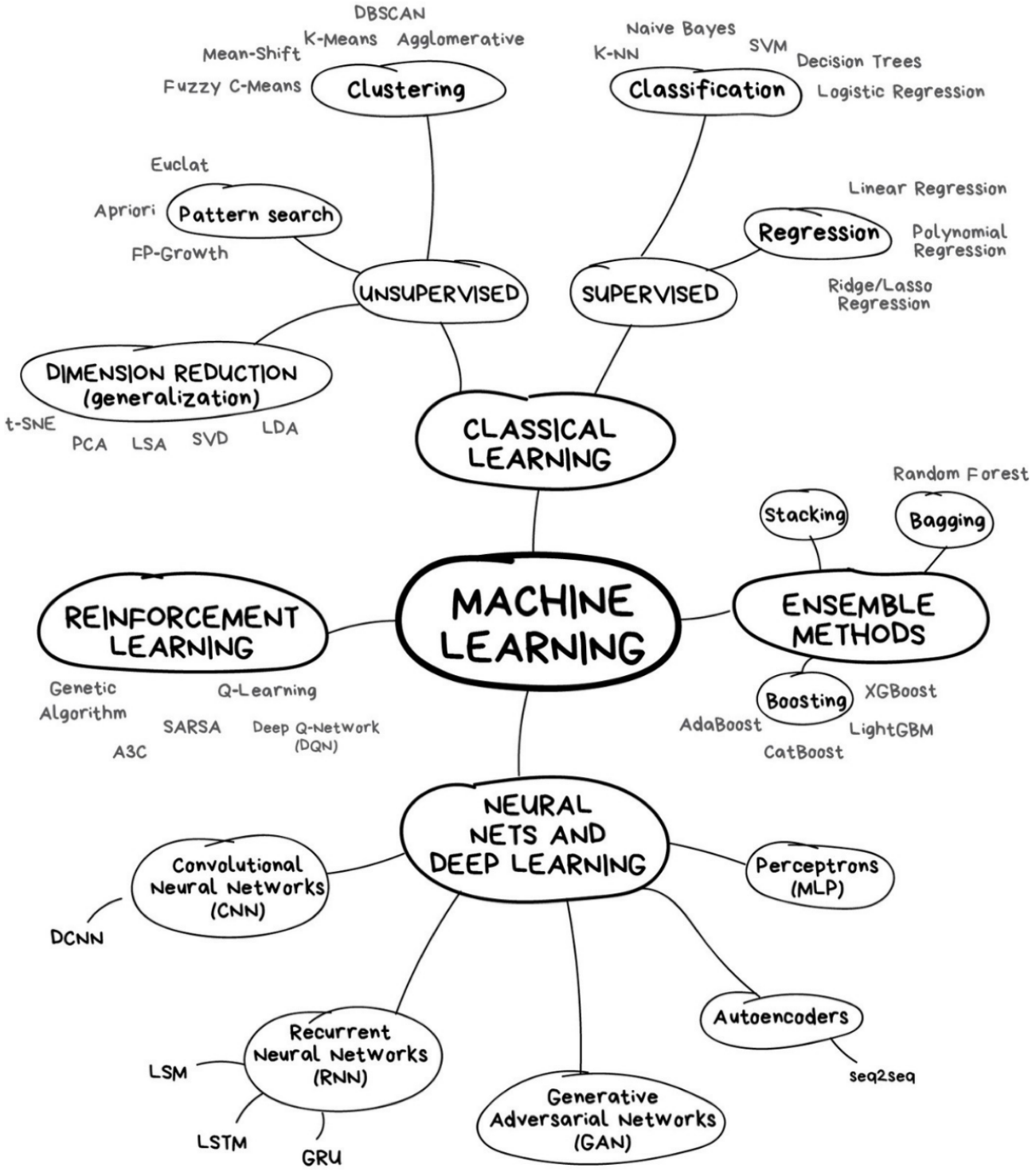
Example 2.1.5: ML: Leaning to Detect Credit Card Fraud

Formally speaking: A ML algorithm improves some measures of performance P when executing some task T through some type of experience E

- Task T: assign category of Fraud/Not Fraud to CC transaction
- Performance Measure P: Accuracy of the classifier, i.e, assign higher penalty when Fraud is labeled as Not Fraud
- Training Process E: Set of historical credit card training transactions categorized as Fraud or NotFraud

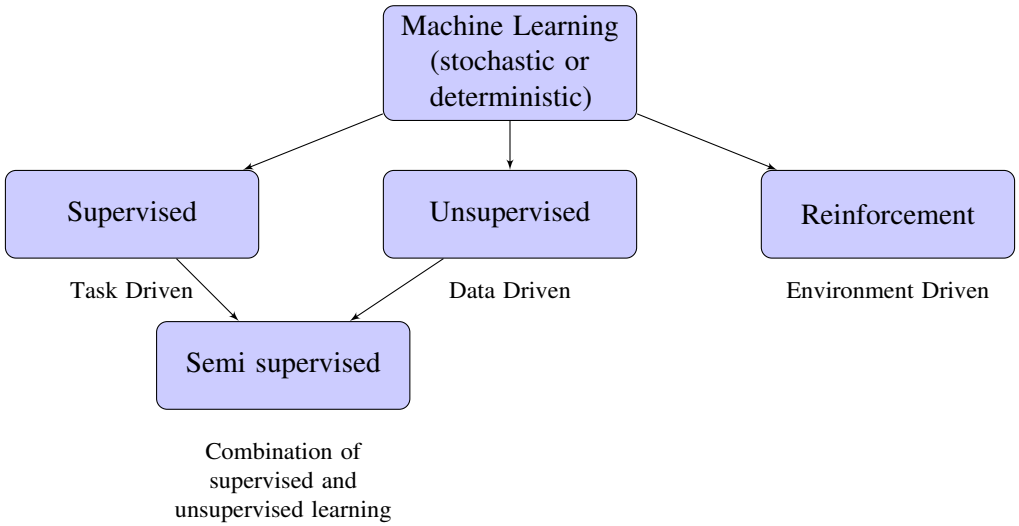
Remark 2.1.7: ML - Working Mechanism


Machine learning workflow
Unsupervised
Feature extraction
Machine learning algorithm
Grouping of objects
based on some common characteristics
Training set
Supervised
Predictive model
New Data
Annotated data



Machine Learning. Image by vas3k

Remark 2.1.8: ML - Major Classes


Machine Learning (stochastic or deterministic)
Supervised
Unsupervised
Reinforcement
Task Driven
Data Driven
Environment Driven
Semi supervised
Combination of supervised and unsupervised learning

2 Fundamentals of Artificial Neural Network

2.1 Introduction

Overview 2.2.1: ANNs

Artificial Neural Networks (ANNs) are the building blocks and the main tools for neuro-computing. they are physical cellular systems, which can acquire, store and utilize experiential knowledge.

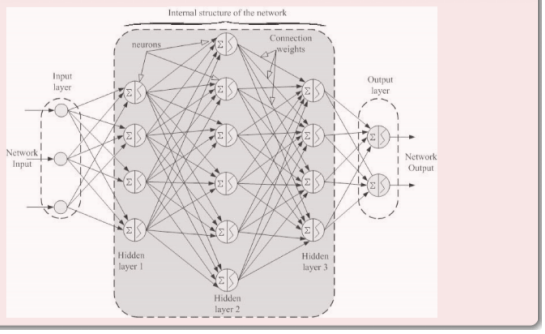
ANNs are a set of parallel and distributed computational elements classified according to topologies, learning paradigms and at the way information flows within the network.

ANNs are generally characterized by their:

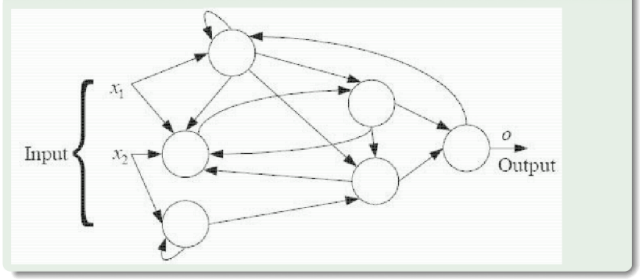
- Architecture:
  - Feedforward
  - Recurrent
- Activation functions:
  - Binary
  - Continuous
- Learning Paradigm:
  - Supervised
  - Unsupervised
  - Hybrid

Example 2.2.6: ANNs Architecture

Feedforward Flow of Information



Recurrent Flow of Information

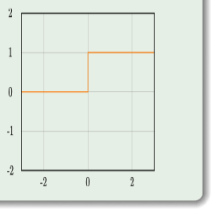


Example 2.2.7: ANNs Activation Functions

Binary Activation Functions

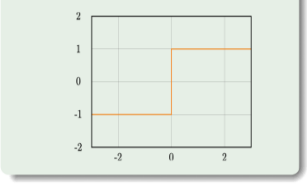
Step Function

$$\text{step}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

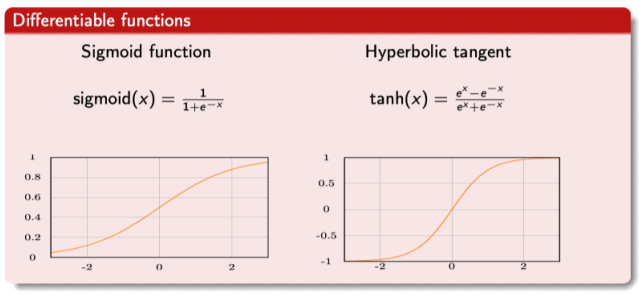


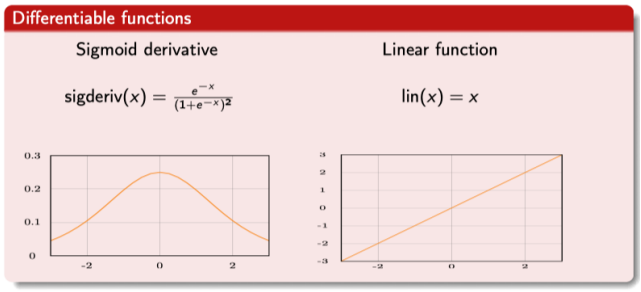
Signum Function

$$\text{sigum}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{otherwise} \end{cases}$$



Differentiable Activation Functions



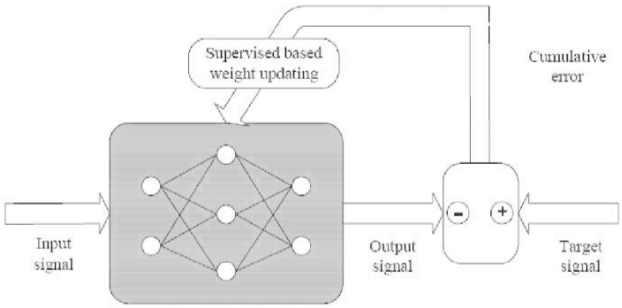


Example 2.2.8: Learning Paradigms

- Supervised
  - Multilayer perceptrons
  - Radial basis function networks
  - Modular neural networks
  - LVQ (learning vector quantization)
- Unsupervised
  - Competitive learning networks
  - Kohonen self-organizing networks
  - ART (adaptive resonant theory)
- Hybrid
  - Autoassociative memories (Hopfield networks)

Remark 2.2.9: Supervised Learning

- Training by example (ex: prior known desired output for each input pattern)
- Particularly useful for feedforward network



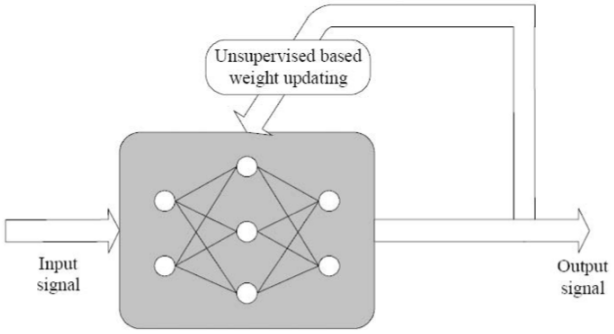
Algorithm 2.2.2: Training

- Compute error between desired and actual outputs
- Use the error through a learning rule (e.g., gradient descent) to adjust the network's connection weights
- Repeat steps 1 and 2 for input/output patterns to complete one epoch
- Repeat steps 1 to 3 until maximum number of epochs is reached or an acceptable training error is reached

Remark 2.2.11: UnSupervised Learning

- No prior known desired output
- In other words, training data composed of input patterns only
- Network uses training patterns to discover emerging collective properties and organizes the data into

clusters.



Algorithm 2.2.3: Training

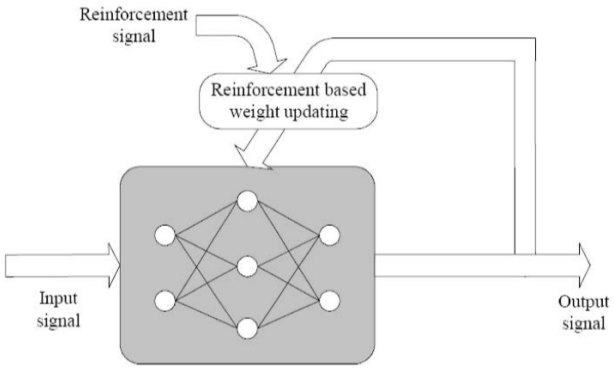
- Training data set is presented at the input layer
- Output nodes are evaluated through a specific criterion
- Only weights connected to the winner node are adjusted
- Repeat steps 1 to 3 until maximum number of epochs is reached or the connection weights reach steady state

Remark 2.2.11: Rationale

- Competitive learning strengths the connection between the incoming pattern at the input layer and the winning output node.
- The weights connected to each output node can be regarded as the center of the cluster associated to that node.

Remark 2.2.12: Reinforcement Learning

- Reinforcement learning mimics the way humans adjust their behavior when interacting with physical systems (e.g., learning to ride a bike).
- Network's connection weights are adjusted according to a qualitative and not quantitative feedback information as a result of the network's interaction with the environment or system.
- The qualitative feedback signal simply informs the network whether or not the system reacted "well" to the output generated by the network.



Algorithm 2.2.4: Training

- Present training input pattern network
- Qualitatively evaluate system's reaction to network's calculated output
  - If response is "Good", the corresponding weights led to that output are strengthened
  - If response is "Bad", the corresponding weights are weakened

Remark 2.2.13: Major Milestones

1. Late 1940's : McCulloch Pitt Model (by McCulloch and Pitt)
2. Late 1950's – early 1960's : Perceptron (by Roseblatt)
3. Mid 1960's : Adaline (by Widrow)
4. Mid 1970's : Back Propagation Algorithm - BPL I (by Werbos)
5. Mid 1980's : BPL II and Multi Layer Perceptron (by Rumelhart and Hinton)

2.2 McCullach-Pitts Model

Overview 2.2.2: McCulloch-Pitts Model

- First serious attempt to model the computing process of the biological neuron
- The model is composed of one neuron only
- Limited computing capability
- No learning capability

2.3 Perceptron

The first ML algorithm.

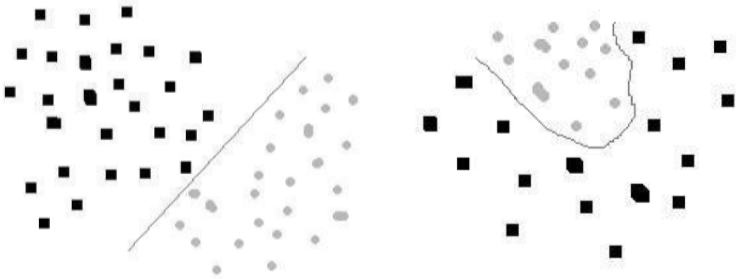
Overview 2.2.3: Perceptron

- Uses supervised learning to adjust its weigh ts in response to a comparative signal between the network's actual output and the target output
- Mainly designed to classify linearly separable patterns

Definition 2.2.6: Linear Separation

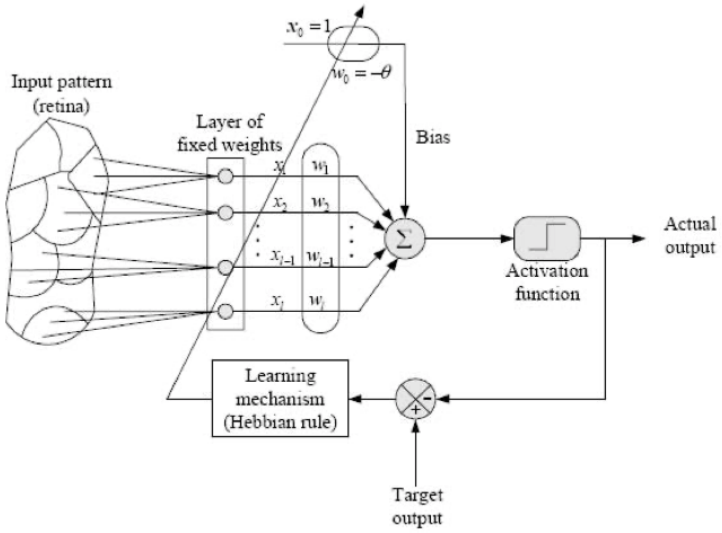
Patterns are linearly separable means that there exists a hyperplanar multidimensional decision boundary that classifies the patterns into two classes.

Linearly separable Non-linearly Separable



Remark 2.2.14: Architecture

- One neuron (one output)
- l input signals: $x_1, \ldots, x_l$
- Ajustable weights $w_1, \ldots, w_l$ and bias θ
- Binary activation function (ex: step or hard limiter function)



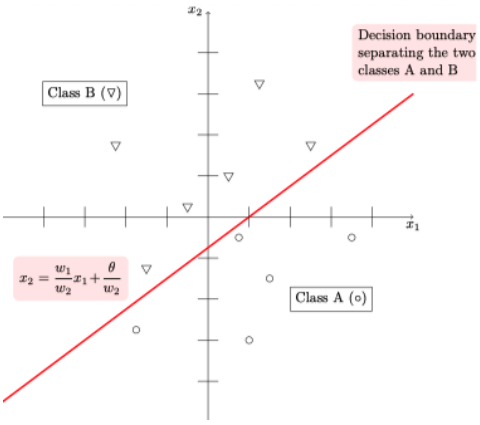
Theorem 2.2.1: Perceptron Convergence Theorem

If the training set is linearly separable, there exists a set of weights for which the training of the Perceptron

will converge in a finite time and the training patterns are correctly classified.

Example 2.2.9: 2D case

In the two-dimensional case, the theorem translates into finding the line defined by $w_1x_1 + w_2x_2 - \theta = 0$, which adequately classifies the training patterns.



Algorithm 2.2.5: Perceptron - Training Algorithm

1. Init weights and thresholds to small random values
2. Choose an input-output pattern $(x^{(k)}, t^{(k)})$ from the training data
3. Compute the network's actual output $o^{(k)} = f\left(\sum_{i=1}^{l} w_i x_i^{(k)} - \theta\right)$
4. Adjust the weights and bias according to the Perceptron learning rule:
   $\Delta w_i = \eta[t^{(k)} - o^{(k)}]x_i^{(k)}$, and $\Delta\theta = -\eta[t^{(k)} - o^{(k)}]$, where $\eta \in [0,1]$ is the perceptron's learning rate.
   If f is the signum function, this becomes equivalent to:

$$\Delta w_i = \begin{cases} 2\eta t^{(k)} x_i^{(k)} & , if\, t^{(k)} \neq o^{(k)} \\ 0 & , \text{otherwise} \end{cases} \qquad \Delta\theta_i = \begin{cases} -2\eta t^{(k)} & , if\, t^{(k)} \neq o^{(k)} \\ 0 & , \text{otherwise} \end{cases} \tag{2.1}$$

5. If a whole epoch is complete, then pass to the following step; otherwise go to Step 2.
6. If the weights and bias reached steady state ($\Delta w_i \approx 0$) through the whole epoch, then stop the learning; otherwise go through one more epoch starting from Step2.

Cons 2.2.1: Perceptron

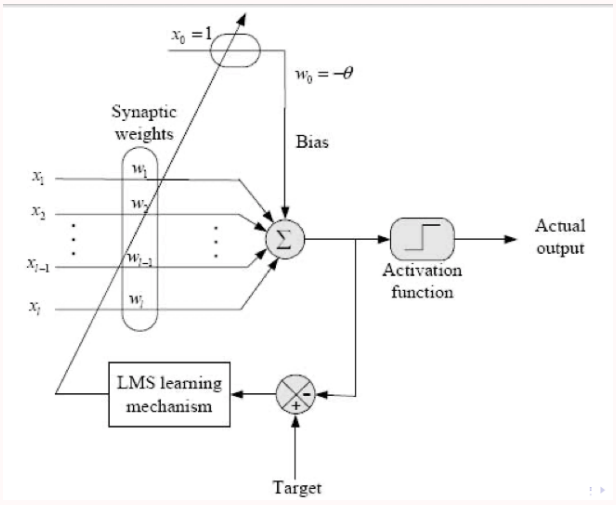
1. Cannot separate linearly non-separable patterns
2. Lack of generalization: once trained, it cannot adapt its weights to a new set of data

2.4 Adaline (Adaptive Linear Neuron)

Overview 2.2.4: Adaline

- More versatile than the Perceoptron in terms of generalization
- More powerful in terms of weight adaptation
- An Adaline is composed of a linear combiner, a binary activation function (hard limiter), and adaptive weights

Graphical Illustration



Remark 2.2.15: Learning in an Adaline

- Adaline adjusts its weights according to the least mean squared (LMS) algorithm (also known as the Widrow-Hoff learning rule) through gradient descent optimization.

  Alert 2.2.3: LMS

  Widrow-Hoff learning rule $\equiv$ LMS (Least Mean Square)

- At every iteration, the weights are adjusted by an amount of proportional to the gradient of the cumulative error of the network $E(w) \Rightarrow \Delta w = -\eta \nabla_w E(w)$
- The network's total error $E_c(w)$ for all patterns $(x^{(k)}, t^{(k)}), k = 1, 2, \ldots, n$. This is the cumulative error involving all training data and ait is between the desired response $t^{(k)}$ and the linear combiner's output $(\sum_i w_i x_i^{(k)} - \theta)$. It is expressed as:

$$E_c(w) = (1/2) \sum_k \left[t^{(k)} - (\sum_i w_i x_i^{(k)} - \theta) \right]^2 \tag{2.2}$$

- All patterns have to be represented here and the cumulative error has to be computed. This is referred as batch training and it is time consuming
- Instead we compute individual errors based on the presented pattern (k) and keep updating the weights each time a pattern in the epoch is presented. This is known as on-line training.

$$E^k(w) = (1/2) \left[t^{(k)} - (\sum_i w_i x_i^{(k)} - \theta) \right]^2 \tag{2.3}$$

- Hence, individual weights are updated using $\Delta w = -\eta \nabla_w E^k(w)$ as:

$$\Delta w_i = \eta \left(t^{(k)} - \sum_i w_i x_i^{(k)} \right) x_i^{(k)} \tag{2.4}$$

Algorithm 2.2.6: Adaline - Training Algorithm

1. Init weights and thresholds to small random values
2. Choose an input-output pattern $(x^{(k)}, t^{(k)})$ from the training data
3. Compute the network's actual output $r^{(k)} = \sum_{i=1} w_i x_i^{(k)} - \theta$
4. Adjust the weights and bias according to the LMS rule:
   $\Delta w_i = \eta \left(t^{(k)} - \sum_i w_i x_i^{(k)} \right) x_i^{(k)}$, where $\eta \in [0,1]$ is the perceptron's learning rate.
5. If a whole epoch is complete, then pass to the following step; otherwise go to Step 2.
6. If the weights and bias reached steady state ($\Delta w_i \approx 0$) through the whole epoch, then stop the learning; otherwise go through one more epoch starting from Step2.

Pros 2.2.1: LMS Algorithm

- Easy to implement
- Suitable for generalization, which is a missing feature in the Perceptron

Cons 2.2.2: Adaline

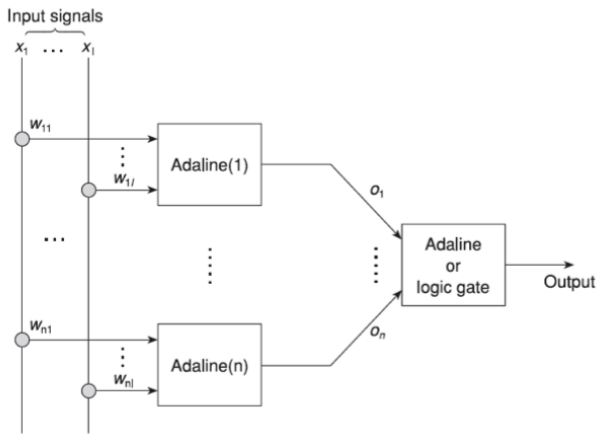
The adaline, while having attractive training capabilities, suffers also (similarly to the perceptron) from the inability to train patterns belonging to nonlinearly separable spaces.

Remark 2.2.16: Madaline

Researchers have tried to circumvent this difficulty by setting cascade layers of adaline units.

When first proposed, this seemingly attractive idea did not lead to much improvement due to the lack of an existing learning algorithm capable of adequately updating the synaptic weights of a cascade architecture of perceptrons.

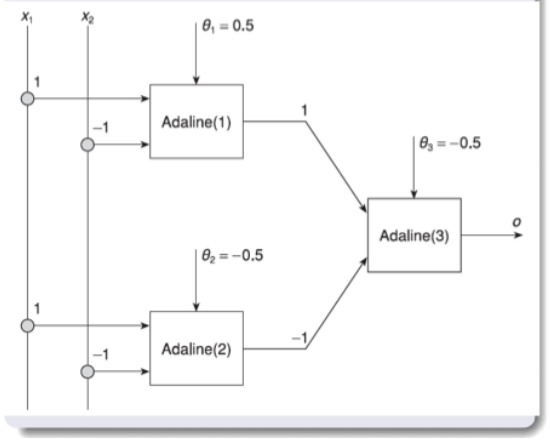
Other researchers were able to solve the nonlinear separability problem by combining in parallel a number of adaline units called a madaline.



Example 2.2.10: Madaline

- Solving the XOR logic function by combining in parallel two adaline units using the AND logic gate.

Graphical Solution



Related Binary Table

x_1	x_2	$o = x_1 XOR x_2$
0	0	1
0	1	-1
1	0	-1
1	1	1

Remark 2.2.17: Madaline

- Despite the successful implementation of the adaline and the madaline units in a number of applications, many researchers conjectured that to have successful connectionist computational tools, neural models should involve a topology with a number of cascaded layers.
- Schematics of the madaline implementation of the backpropagation learning algorithm to neural network models composed of multiplelayers of perceptrons.

Chapter 3

Classes of ML models

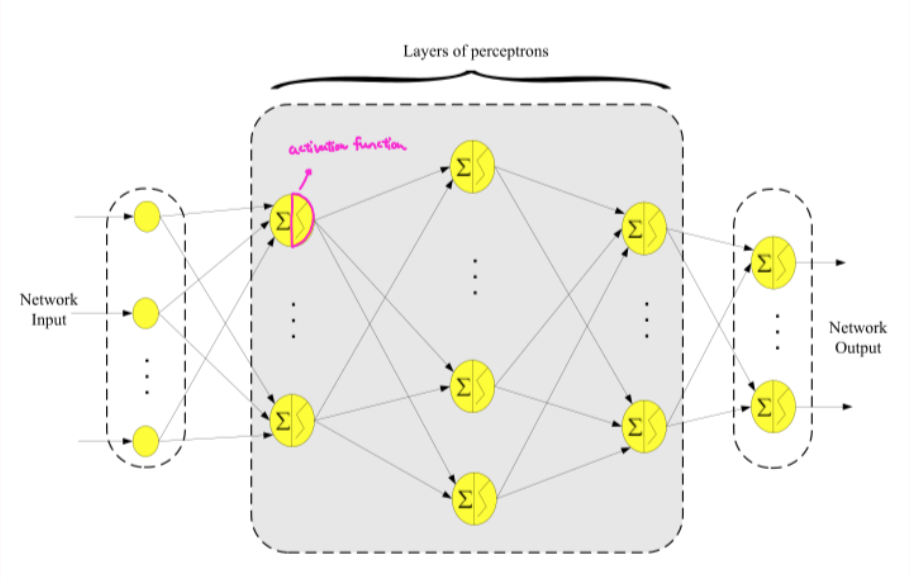
1 Major Classes of Neural Network

1.1 MLPs: Multi-Layer Perceptrons

Background

- The perceptron lacks the important capability of recognizing patterns belonging to non-separable linear spaces
- The madaline (Remark 2.2.17) is restricted in dealing with complex functional mappings and multi-class pattern recognition problems
- The multilayer architecture first proposed in the late 60s
- MLP re-emperhed as a solid connectionist model to solve a wide range of complex problems in the mid-80s
- This occurred following the reformulation of a powerful learning algorithm commonly called the Back Propagation Learning (BPL)
- It was later implemented to the multilayer perceptron topology with a great deal of success

Schematic Representation of MLP Network



Overview 3.1.1: BPL

- The BPL is based on the gradient descent technique involving the minimization of the network cumulative error.

$$E(k) = \sum_{i=1}^{q} [t_i(k) - o_i(k)]^2 \tag{3.1}$$

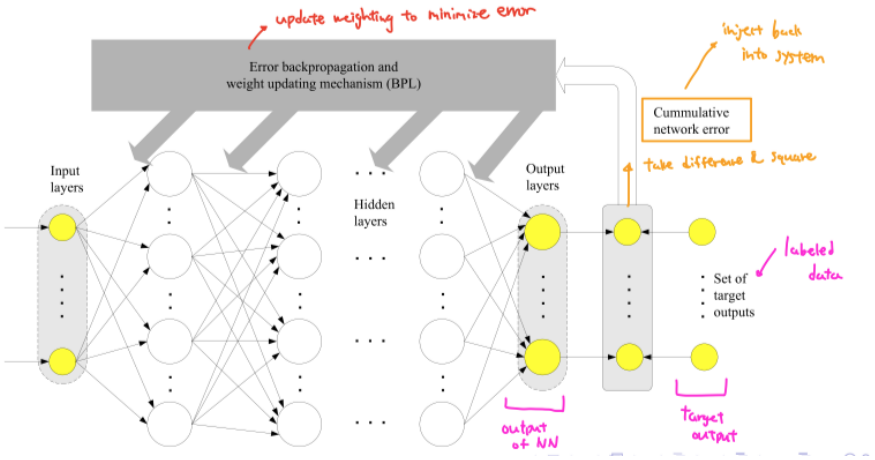
i = i-th neuron of the output layer
q = number of output neurons

- It is designed to update the weights in the direction of the gradient descent of the cumulative error

Algorithm 3.1.1: Two Stage Algorithm

1. Patterns are presented to the network
2. A feedback signal is then propagated backward with the main task of updating the weights of the layers connections according to the BPL.

Schematic Representation of BPL Network illustrating the notion of error back-propagation.



Remark 3.1.1: Objective Function

- Using the sigmoid function as the activation function for all the neurons of the network, we defined E_c as

$$E_c = \sum_{k=1}^{n} E(k) = \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{q} [t_i(k) - o_i(k)]^2 \tag{3.2}$$

- the formulation of the optimization problem can now be stated as finding the set of the network weights that minimizes E_c or $E(k)$

Offline Training:
$$\min_w E_c = \min_w \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{q} [t_i(k) - o_i(k)]^2 \tag{3.3}$$

Online Training:
$$\min_w E(k) = \min_w \frac{1}{2} \sum_{i=1}^{q} [t_i(k) - o_i(k)]^2 \tag{3.4}$$

1.1.1 BPL: On-line Training

Remark 3.1.4: BPL: On-line Training

Objective Function:

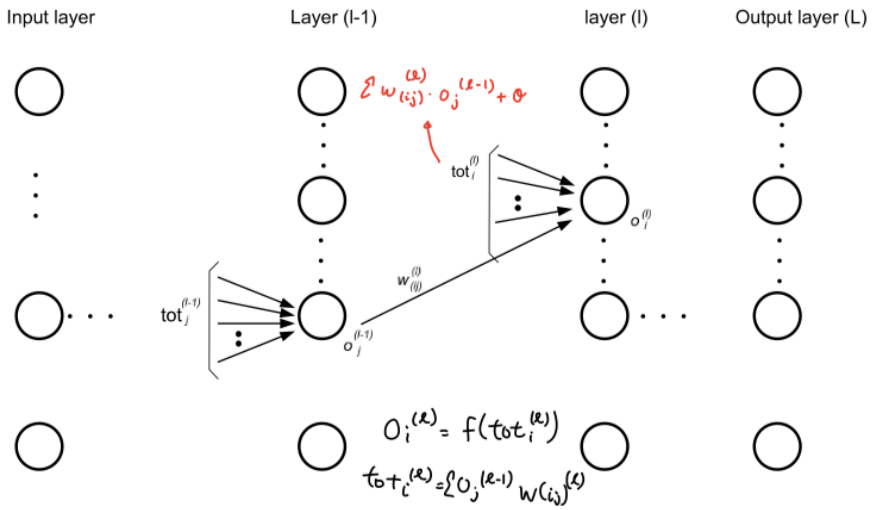
$$\min_{w} E(k) = \min_{w} \frac{1}{2} \sum_{i=1}^{q} [t_i(k) - o_i(k)]^2 \tag{3.5}$$

Remark 3.1.3: Updating Rule for Connection Weights

$$\Delta w^{(l)} = -\eta \frac{\partial E(k)}{\partial w^{(l)}} \tag{3.6}$$

l = (l-th) layer
η = learning rate
Δw_{ij}^{l} = the weight update for the connection linking the node j of layer $(l-1)$ to node i located at layer l
o_j^{l-1} = the output of the neuron j at layer $l-1$, the one located just before layer l
tot_i^l = the sum of all signals reaching node i at hidden layer l coming from previous layer $l-1$

Example 3.1.1: Illustration of interconnection between layers of MLP



Remark 3.1.4: Interconnection Weights Updating Rules

- $\Delta w^{(l)} = \Delta w_{ij}^{(l)} = -\eta \frac{\partial E(k)}{\partial w^{(l)}} = -\eta [\frac{\partial E(k)}{\partial o_i^{(l)}}][\frac{\partial o_i^{(l)}}{\partial tot_i^{(l)}}][\frac{\partial tot_i^{(l)}}{\partial w_{ij}^{(l)}}]$
- For the case where the layer (l) is the output layer (L):
  $\Delta w_{ij}^{(L)} = \eta [t_i - o_i^{(L)}][f'(tot)_i^{(L)}] o_j^{(L-1)}; \quad f'(tot)_i^{(l)} = \frac{\partial f(tot_i^{(l)})}{\partial tot_i^{(l)}}$
- By denoting $\delta_i^{(L)} = [t_i - o_i^{(L)}][f'(tot)_i^{(L)}]$ as being the error signal of the i-th node of the output layer, the weight update at layer (L) is as follows: $\Delta w_{ij}^{(L)} = \eta \delta_i^{(L)} o_j^{(L-1)}$
- In the case where f is the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$, the error signal becomes expressed as:
  $\delta_i^{(L)} = [(t_i - o_i^{(L)}) o_i^{(L)} (1 - o_i^{(L)})]$
- Propagating the error backward now, and for the case where (l) represents a hidden layer $(l < L)$, the expression of $\Delta w_{ij}^{(L)}$ becomes given by: $\Delta w_{ij}^{(l)} = \eta \delta_i^{(l)} o_j^{(l-1)}$, where $\delta_i^{(L)} = f'(tot)_i^{(l)} \sum_{p=1}^{n_{l+1}} \delta_p^{(l+1)} w_{pi}^{(l+1)}$
- Again, when f is taken as the sigmoid function, $\delta_i^{(l)}$ becomes expressed as:
  $\delta_i^{(l)} = o_i^{(l)} (1 - o_i^{(l)}) \sum_{p=1}^{n_{l+1}} \delta_p^{l+1} w_{pi}^{l+1}$

Alert 3.1.1: ...

Rest Slides are open to read: Off-line BPL for MLP

2 Variance, Bias, Underfiting, and Overfiting

2.1 Machines Learning Model

Overview 3.2.2: Machine Learning - Main Idea

The idea is to extract the best model form a set of collected data, which could predict future occurences (whether for classification, which involves discrete output or regression which involves continuous output).

The data collected is usually corrupted with noise or is recorded wrongly.

Remark 3.2.5: Types of ML Models

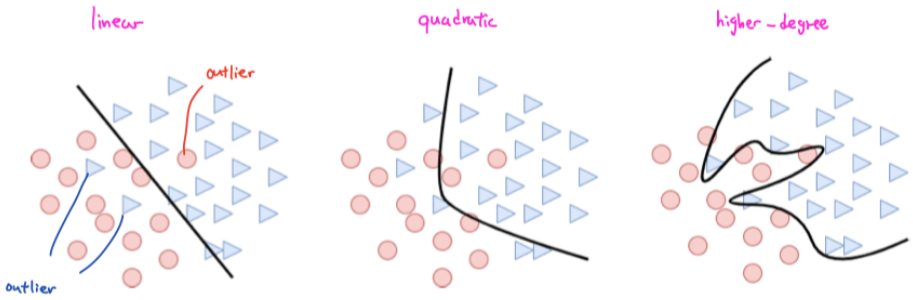
Definition 3.2.1: Classification

Given samples from two (or more classes), find a boundary (linear or non-linear) that separates them with minimum incorrect classifications.

Definition 3.2.2: Regression

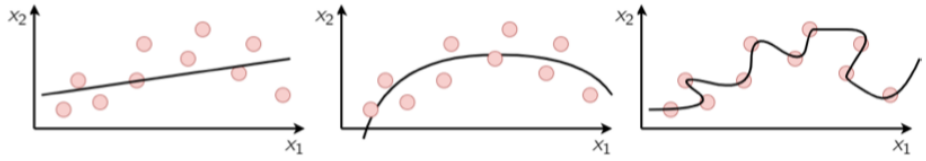
Find the continuous relationship between two variables x_1 and x_2 using only samples and no prior knowledge of their distribution.

Example 3.2.2: Learning Model for Classification



Classification of two classes with a polynomial function having different degrees of freedom.

Example 3.2.3: Learning Model for Regression



Regression of data with a polynomial function having different degrees of freedom.

Example 3.2.4: Classification

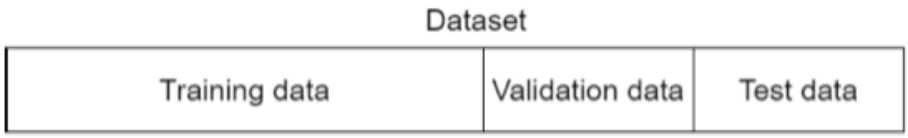
We want to teach a baby (model) the difference between tomato and cucumber. We show it several tomato and say these are tomatoes. We do the same for cucumbers. The baby learns to classify tomatoes from cucumbers. The baby can find out whether a new unseen vegetable is a tomato or cucumber. The out put of classification (whether is tomato or cucumber) is discrete.

Example 3.2.5: Regression

We want to make an artificial leg for people without leg s. Using the Electromyography (EMG) signals in a health y leg , we collect which signals result in which degrees of knee. The learned artificial leg , when is installed, can estimate the knee degree for new unlearned EMG signals. The output of regression (the knee degree) is continuous.

2.2 Training, Validation, and Test Data

Remark 3.2.6: Training, Validation, and Test Data



Dataset can be divided into disjoint training, validation, and test sets:

Definition 3.2.3: Training Dataset

used for teaching the model

Definition 3.2.4: Validation Dataset

used for either selection of hyper-parameters or checking overfitting (introduced later) during the training phase.

Definition 3.2.5: Test Dataset

used for testing the trained model (to see how good its performance is). This is never seen by the model during training and cannot affect the weights in any way.

2.3 Bias and Variance

Remark 3.2.7: Evaluating a Model

The error in predictions is defined as the difference between the predicted values and the actual (ground-truth) values. Two major metrics in direct correlation with the predition error are then used to assess the goodness of a model:

- Variance
- Bias

Definition 3.2.6: Expected Value

What we expect from a random variable to be on average (a.k.a. mean μ).

$$\mathbb{E}(\hat{X}) := \sum_{x_i \in \hat{X}} x_i \mathbb{P}(x_i) \tag{3.7}$$

Definition 3.2.7: Variance

How spread the values of a random variable are.

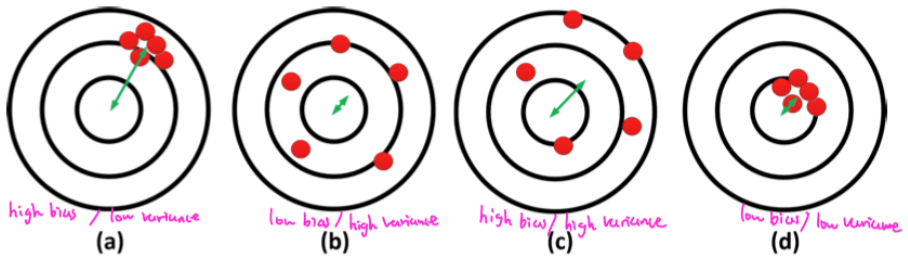
$$\mathbb{V}ar(\hat{X}) := \mathbb{E}\left((\hat{X} - \mathbb{E}(\hat{X}))^2\right) = \mathbb{E}(\hat{X}^2) - (\mathbb{E}(\hat{X}))^2 \tag{3.8}$$

Definition 3.2.8: Bias

How different the expected value of a random variable is from its actual value

$$\mathbb{B}ias(\hat{X}) := \mathbb{E}(\hat{X}) - X \tag{3.9}$$

Example 3.2.6: Variance and Bias - Dart Illustration



The dart example for (a) high bias and low variance, (b) low bias and high variance, (c) high bias and high variance, and (d) low bias and low variance.

The worst and best cases are (c) and (d), respectively.

The center of the circles is the true values of variable

The green double arrow denotes the distance to the center of the average of the predicted values.

Remark 3.2.8: Variance in ML

A variance is defined as the amount that the estimate of the target boundary (classification) or target function (regressor) will change if different data points were used.

- **High Variance**: Significant changes
- **Low Variance**: Minor changes

Remark 3.2.9: Variance in Learning Models

Usually, complex machine learning algorithms (nonlinear) have HIGH variance.

Low Variance	High Variance
Linear Regression (LRG)	Polynomial RG and Decision Trees
Linear Discriminant Analysis	Support Vector Machines (SVM)
Logistic Regression	Multi-layer Neural Networks
Naive Bayes	K-Nearest Neighbours

Remark 3.2.10: Bias in ML

A bias is the amount that the estimate of the target boundary (classification) or target function (regressor) classifies or regresses training data points correctly.

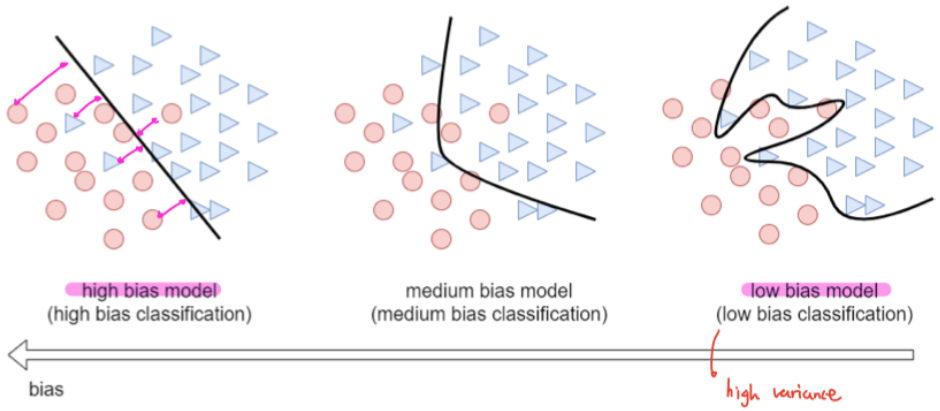
(How distinct/far the data from the estimate)

- **High Bias**: Very bad performance of the model on the training data
- **Low Bias**: Good and acceptable performance of the model on the training data

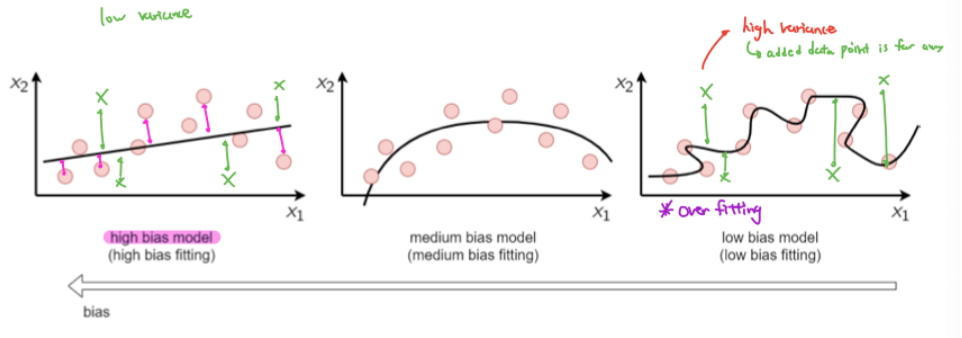
Alert 3.2.2

Both high and low biased models may perform poorly on the testing data; hence, a medium bias is usually desired.

Example 3.2.7: Bias in Classification



Example 3.2.8: Bias in Regression



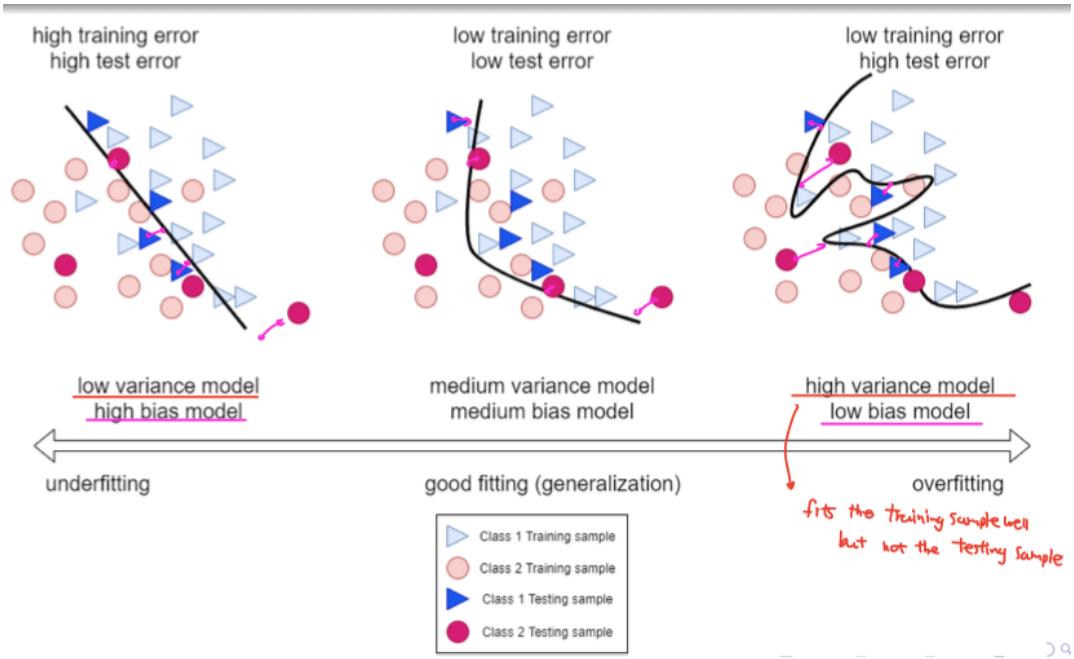
2.3.1 Underfitting, Overfitting, and Generalization

Remark 3.2.11: Overfitting

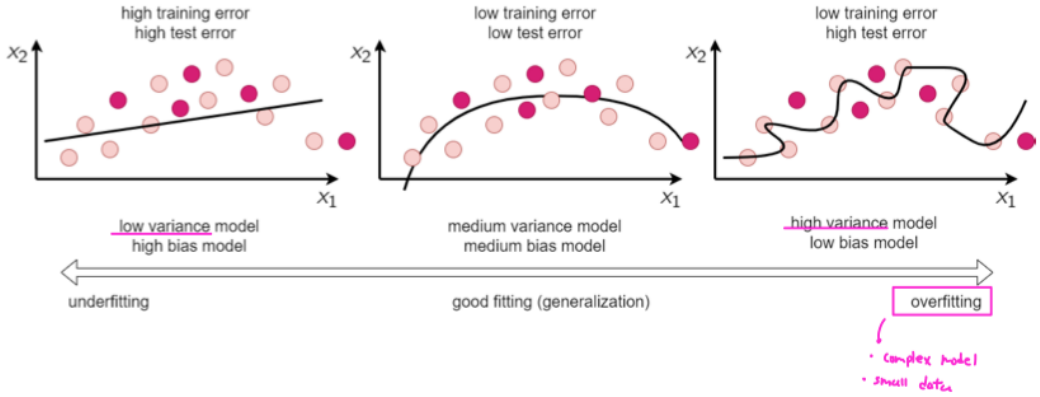
An intuitive example for overfitting is as follows. Assume the model is a baby to whom we want to teach what a coffee cup is:

- **Under-fit**: "a cup is like a glass" ⇒ confuses glass with cup
- **Good-fit**: "a coffee cup usually has a cap, too." ⇒ generalize tp new unseen coffee cups
- **Over-fit**: "Coffee cup has the shape of a leaf on it." ⇒ only able to recognize Tim Hortons cups and not Starbucks cups. ⇒ The baby(model) memorizes rather than learns.

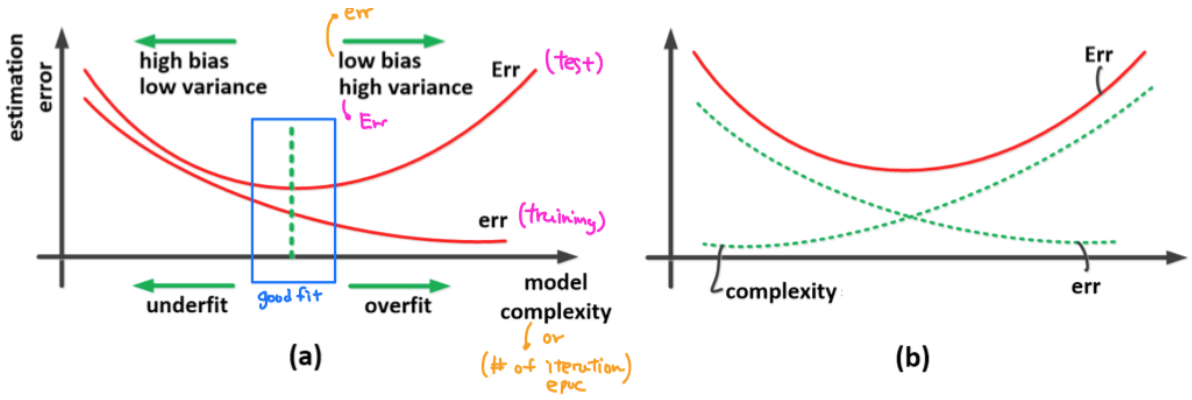
Example 3.2.9: Overfitting in Classification



Example 3.2.10: Overfitting in Regression



Remark 3.2.12: Diagram of Error in Training



(a) Training error (err) v.s. test/validation err (Err)

(b) The more complex the model gets, if it is not trained by sufficient data, we will have overfitting.

Remark 3.2.13: Underfitting vs. Overfitting

Definition 3.2.9: Underfitting

Model fails to capture the underlying trend of the data and performs poorly on both training and testing sets (high bias and low variance), Caused by:

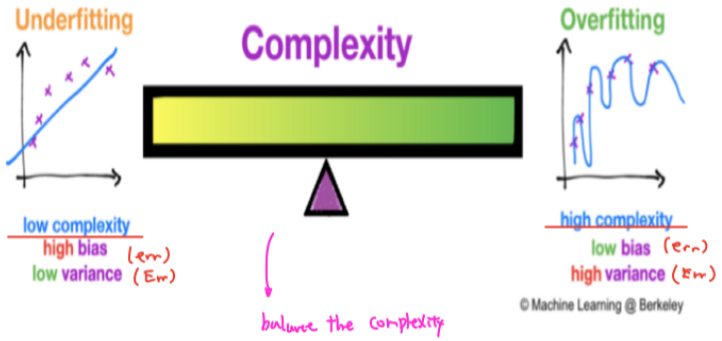
- Model too simple (linear or 1st order polynomial), Low Complexity
- Training data is not enough to capture the complete behaviour of the process

Definition 3.2.10: Overfitting

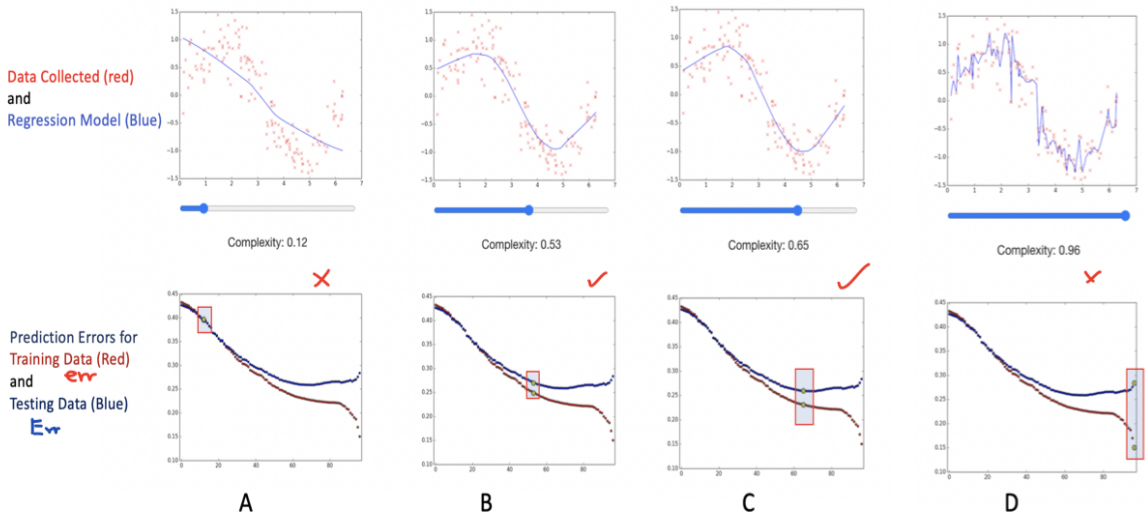
Model corresponds too closely to training dataset, capturing the noise and inaccuracies, therefore performing poorly on the testing set (no generalization - low bias and high variance).

- Model too complex and rigidly tied to training data (inflexible)
- Training data not containing enough variety to describe the entire population
- Too many training iterations (in iterative training algorithms)

Example 3.2.11: Trade-off Design: Bias vs. Variance (1)



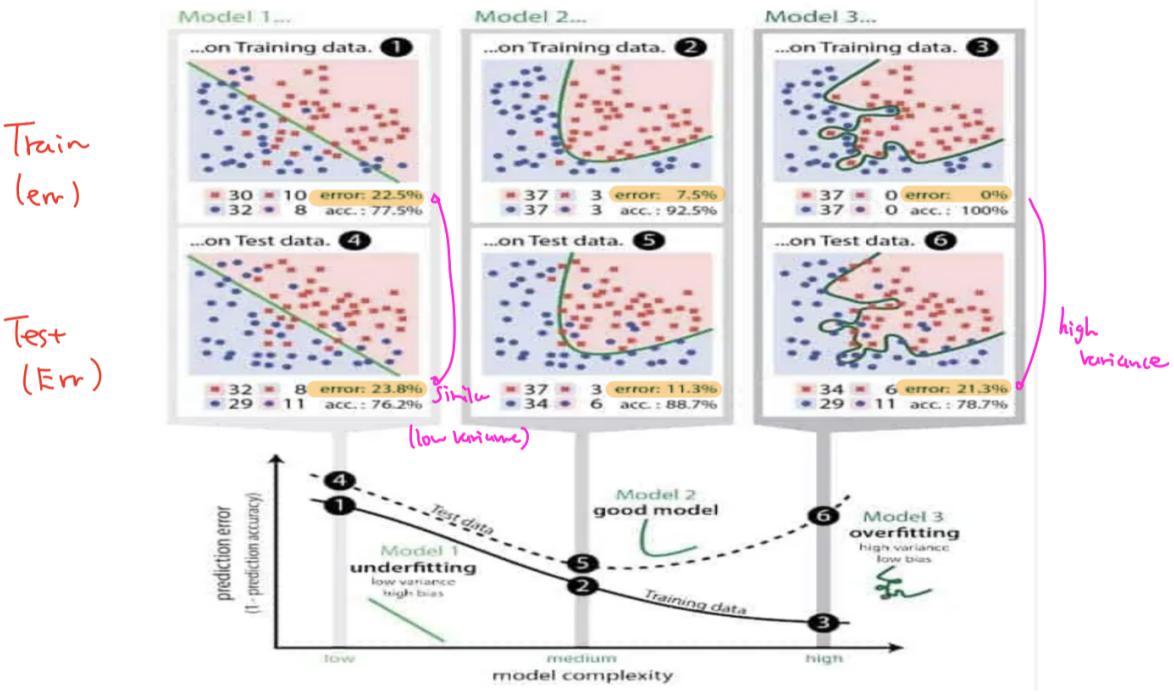
Example 3.2.12: Trade-off Design: Bias vs. Variance (2)



Training and Testing erros vs. complexity levels of proposed models

- A: Simple (High Bias) and prone to Underfitting
- D: Complex (High Variance) and prone to overfitting
- B and C: Better Genralizers of the data (medium bias and medium variance)

Example 3.2.13: Trade-off Design: Bias vs. Variance (3)



2.3.2 How to avoid Under-fitting and Over-fitting

Overview 3.2.3: Dealing with Under-fitting and Over-fitting

The nature of the algorithm dictates whether the model is high variance or high bias.

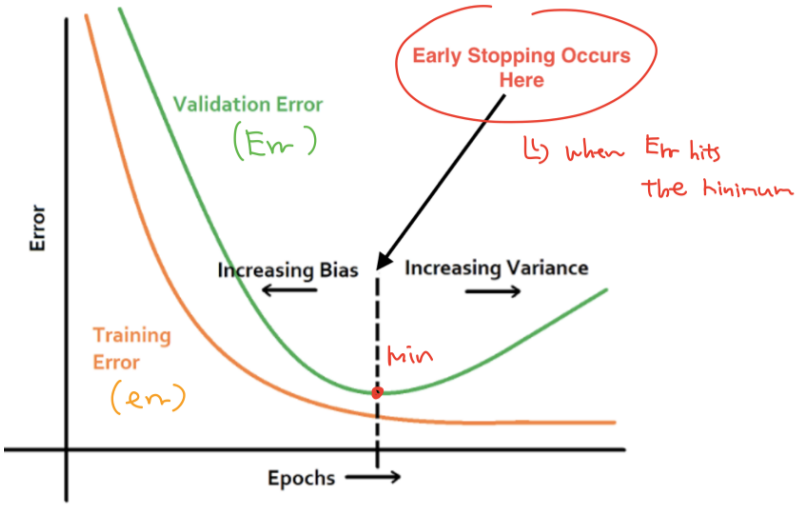
Remark 3.2.14: Solving for Under-fitting

1. Add more training data to the system to capture complex behavior of the model
2. Make the model more complex (adding nodes, layers in ANN, or use higher order polynomials

Remark 3.2.15: Solving for Over-fitting

1. Cross validation [Remark 3.2.17]
2. Training with more data
3. Simplifying features (Reduce Dimension)
4. Early Stopping [Remark 3.2.16]
5. Regularization
6. Ensembling (create multiple models and then combine them to produce improved result)
7. Less complex models

Remark 3.2.16: Early Stopping to avoid Over-fitting



1. Pick a model M
2. Run the training and validation at the same time
   (Validation uses the model that was just trained in the most recent iteration)
3. At certain point, the validation set is not improving in terms of low error performance, while training error keeps decreasing.
4. Stop the training and pick the model that led to the lowest validation error

- Repeat the process with different models (different number of nodes or different regression parameters) and get the lowest possible error.

- The model leads to the lowest training and validation error is the one to choose from.

Remark 3.2.17: Cross Validation

The model may contain some hyper-parameters which need to be optimized without using the test data during training. **Validation Dataset** is used for this.

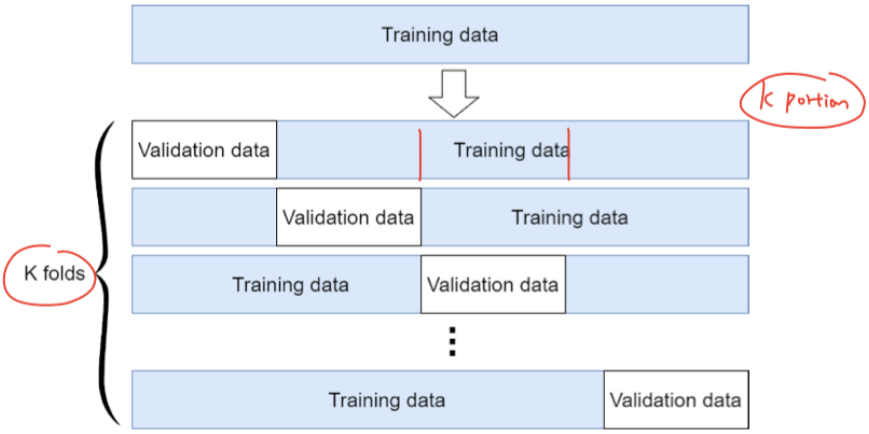
- **Cross Validation** divides the full training data into k equal portions, taking on portion for validation while the rest is used for training.

- The portion is moved and finally the model is re-trained by the parameters with best validation performance.

The well-known types are:

- K-fold cross validation [Remark 3.2.18]
- Leave-one-out cross validation

Remark 3.2.18: K-fold Cross Validation



The validation error is the average of the errors in each fold.

To build an adequate model while making use of the K-fold validation, proceed as follows:

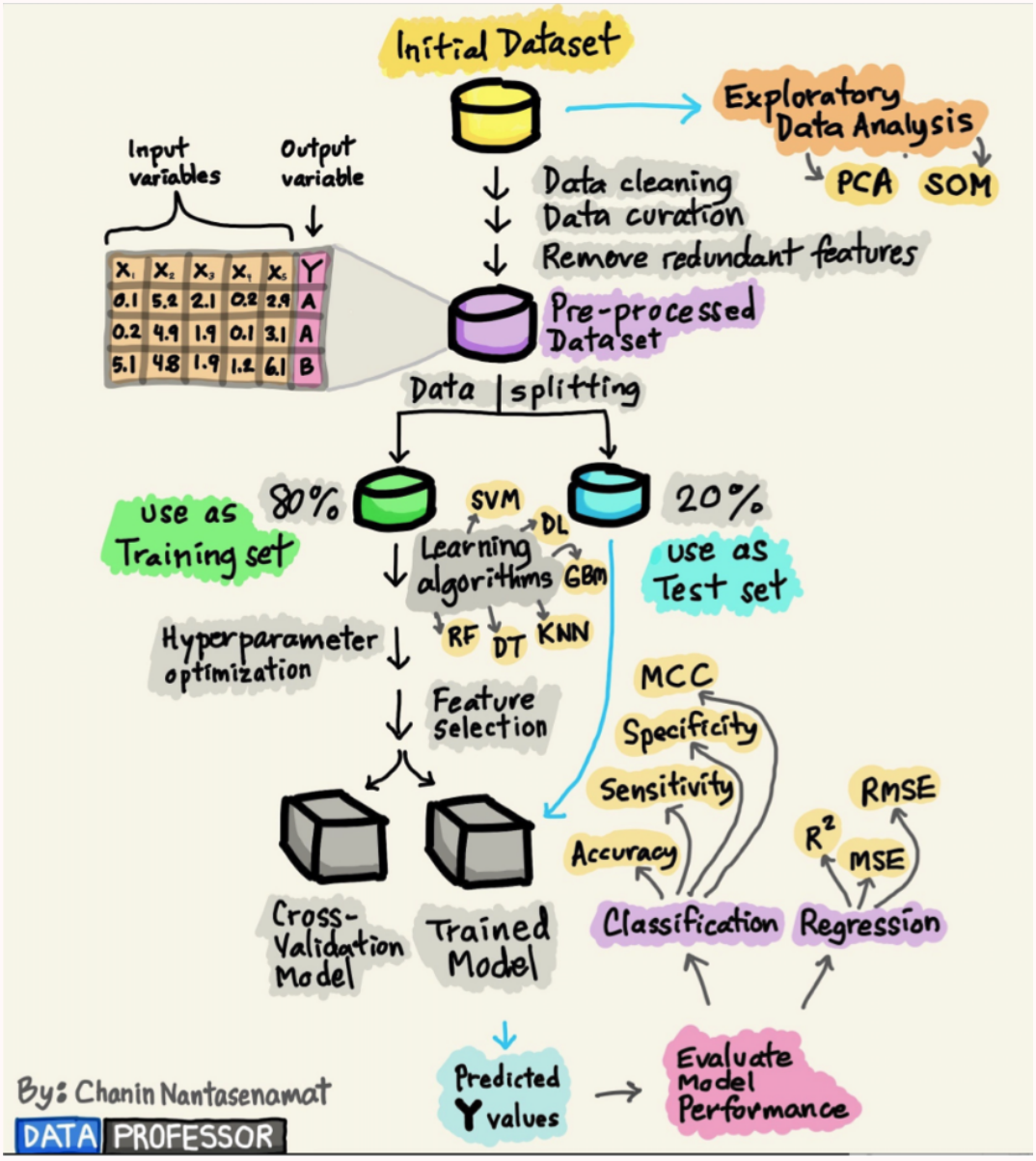
1. Choose the type of model M you want to test: linear regression, polynomials, ANN with different hidden nodes
2. For each model M, run K-fold validation and average out the validation error among all K folds. Run this operation several times to ensure the data is well mixed and there is no memorization.
3. Change the mode type/parameters and record the results with the K-fold validation. Check again the validation errors for each model.
4. The model with the smallest validation error is selected. Now train it with all the training data you have, and finally test it with testing data, for which the system never saw.

More References to Read About under/over-fitting

- Idea [Youtube]
- Theory [Youtube
- Cross-validation [Youtube]
- Overfitting [Tut Paper]
- Early Stopping [Reading]

Summary: Generally Speaking

- Under-fitting (high bias) occurs when the training data is not enough or / and when the model is too simple (smaller number of nodes in ANN or linear or first order polynomials regressors). Hence, increase the training data and increase complexity of the model gradually until good testing results are provided.
- Overfitting (high variance) occurs when the model selected (ANN or high degree polynomials) is too complex, and the training data is not enough to describe the entire population (leads to poor testing results). Overfitting also occurs when there are too many iterations (stop training when the validation/test error starts increasing). Reduce complexity of the model gradually while keeping good amount of training data.

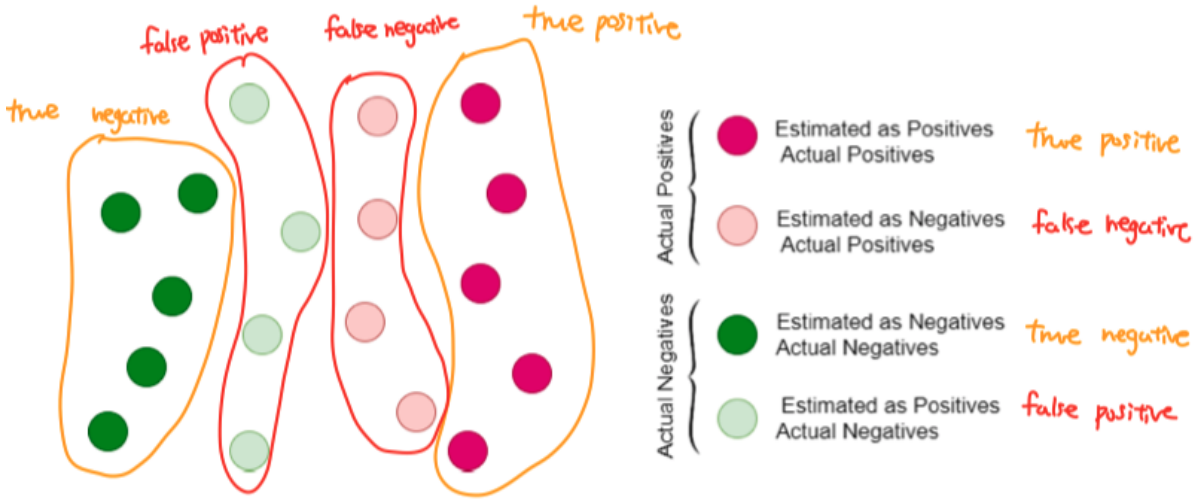


3 Performance Measures

3.1 Performance Measures for Classification

Remark 3.3.19: True/False Positive/Negatives

Consider a binary classification (positive and negative classes):

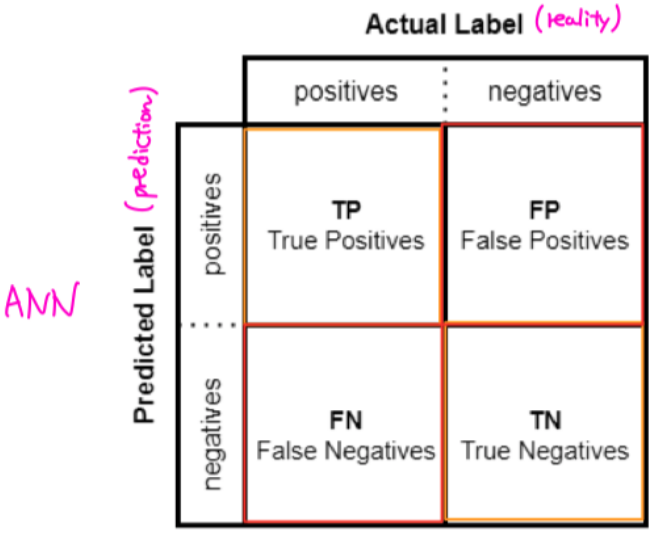


Every prediction of the model belongs to one of following categories:

- **True Positive (TP)**: predicted as positive and the estimation is correct (it is actually in positive class)
- **True Negative (TN)**: predicted as negative and the estimation is correct (it is actually in negative class)
- **False Positive (FP)**: predicted as positive and the estimation is wrong (it is actually in negative class)
- **False Negative (FN)**: predicted as negative and the estimation is wrong (it is actually in positive class)

Definition 3.3.11: Confusion Matrix

The confusion matrix is a table which is formed from the four outcomes of a binary classification (class 1 is positive and class 0 is negative) which is later used to easily calculate different metrics.



Definition 3.3.12: Accuracy

Accuracy measures what ratio of points have been correctly estimated/classified.

Equation 3.3.1: Accuracy

$$\textbf{Accuracy} = \frac{\text{Correct Predictions}}{\text{All Classes}} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.10}$$

Definition 3.3.13: Precision and Recall

Precision measures what ratio of estimations as a class are correct among all points estimated as that class.

Equation 3.3.2: Precision

$$\textbf{Precision} = \frac{\text{True Positives}}{\text{All Predicted Classes as Positive}} = \frac{TP}{TP+FP} \tag{3.11}$$

Recall measures what ratio of estimations as a class are correct among all points that actually belong to that class.

Equation 3.3.3: Recall

$$\textbf{Recall} = \frac{\text{True Positives}}{\text{All Actual Positive Classes}} = \frac{TP}{TP+FN} \tag{3.12}$$

Definition 3.3.14: F_β Score

The F_β score measures whether both precision and recall are good enough, i.e., whether truly estimated points as a class are a big percentage among both all estimated points as that class and all actual points of that class.

Equation 3.3.4: F_β Score

$$F_\beta = (1+\beta^2) \times \frac{\text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \tag{3.13}$$

Equation 3.3.5: A well-known value for $\beta = 1$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.14}$$

Definition 3.3.15: TPR and FPR Rates

TPR (**True Positive Rate**) and FPR (**False Positive Rate**) are defined as:

Equation 3.3.6: TPR

$$\textbf{TPR} = \textbf{Recall} = \frac{\text{True Positives}}{\text{All Actual Positive Classes}} = \frac{TP}{TP+FN} \tag{3.15}$$

Equation 3.3.7: FPR

$$\textbf{FPR} = \frac{\text{False Positives}}{\text{All Actual Negative Classes}} = \frac{FP}{FP+TN} \tag{3.16}$$

Remark 3.3.20: Other Performance Metrics

Error Rate: Ratio of incorrect predictions to total number of samples:

Equation 3.3.8: Error Rate

$$\textbf{ERR} = \frac{FP+FN}{TP+TN+FP+FN} \tag{3.17}$$

Specificity: Proportion of negative data points that are correctly classified as negative (True Negative Rate - TNR).

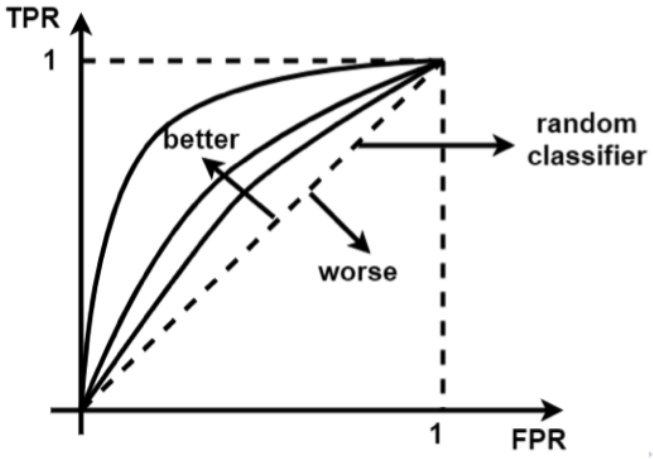
Equation 3.3.9: Specificity

$$\textbf{SP} = \frac{TN}{TN+FP} \tag{3.18}$$

Definition 3.3.16: POC and AUC

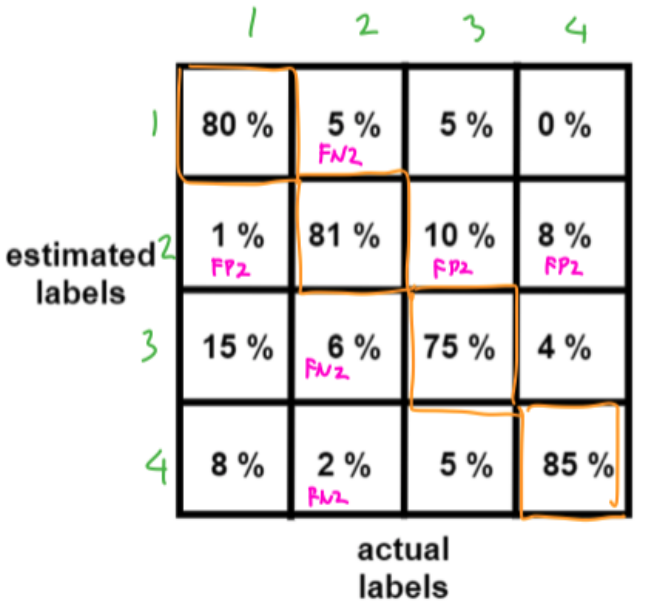
The curve of TPR vs. FPR is the ROC (Receiver Operating Characteristic) curve, We have a curve by sweeping on the threshold of classification.

AUC (Area Under Curve) is the area under ROC curve ($0 \leq AUC \leq 1$).



Remark 3.3.21: Multi-Class Confusion Matrix

When we have more than 2 classes, the points are classified to one of the classes. These points have actual class labels, too. The table of estimated labels versus actual labels creates the multi-class confusion matrix. Every column sums to 100%.

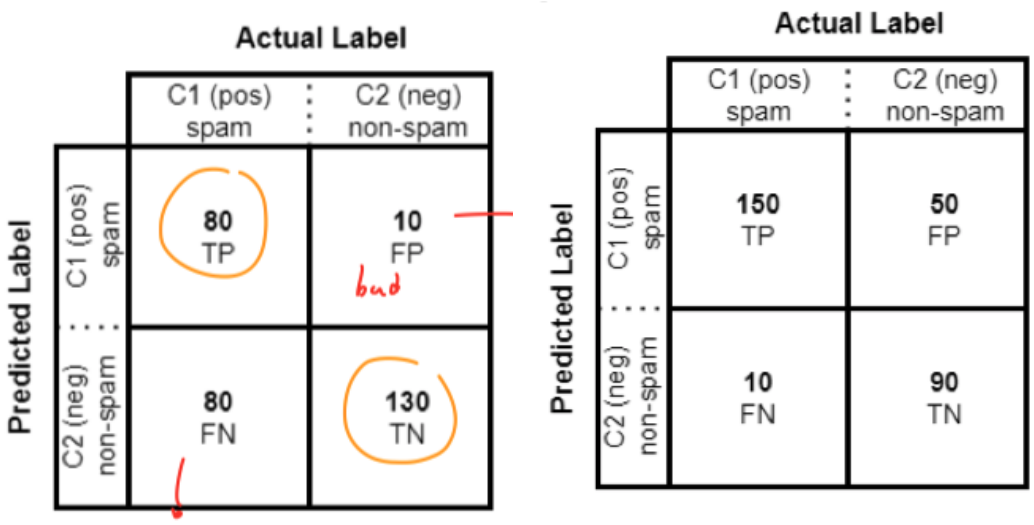

1
2
3
4
80 %
5 %
FN2
5 %
0 %
1 %
FP2
81 %
10 %
FP2
8 %
FP2
15 %
6 %
FN2
75 %
4 %
8 %
2 %
FN2
5 %
85 %
estimated labels
actual labels

3.2 Examples for Measuring Classification Performance

Example 3.3.14: Comparing Models

Assume a model (A) that classifies emails to be either spam (C1) or non-spam (C2). After testing the model, the following confusion matrix was created. Calc the accuracy of the model.

Another model (B) was trained and tested on the same dataset. Calculate the accuracy and deduce which model is better.



$$\text{Accuracy}_A = \frac{TP+TN}{TP+TN+FP+FN} = \frac{80+130}{300} = 70\% \tag{3.19}$$

$$\text{Accuracy}_B = \frac{150+90}{300} = 80\% \tag{3.20}$$

You may intuitively think that Model B is better than model A. However, lets compare their other metrics to understand the performance:

Metric	Model A	Model B
Precision	*0.89*	0.75
Recall	0.5	*0.94*
F_1 Score	0.64	*0.82*
FPR	0.11	*0.358*
TNR	*0.93*	0.64

As a result, we may conclude:

- Precision: Model A is likely to mark and email as spam and be wrong, but is more likely to miss a spam email.
- TNR: Model A is better at detecting non-spam email than Model B. Note that this can be due to simply marking all as non-spam (bias).
- Recall: Model B is better at marking emails that are spam as spam than Model A.

hence, we deduce that the choice of model depends on what kind of performance is favourable to the use-case.

Example 3.3.15: Actual and Predicted Labels

Consider a classification problem for whether a person, in a population of 10 people, is tested as infected (+ve) or not (-ve).

ID	Actually infected	Predicted infected	Outcome
1	1	1	TP
2	0	0	TN
3	0	0	TN
4	1	1	TP
5	0	0	TN
6	0	0	TN
7	0	1	FP
8	1	0	FN
9	0	0	TN
10	1	0	FN

Confusion Matrix:

$$\mathrm{TP} = 2, \mathrm{TN} = 5, \mathrm{FP} = 1, \mathrm{FN} = 2.$$

	Actually Infected	Actually Not Infected
Predicted Infected	2	1
Predicted Not Infected	2	5

The performance measures for this classifier are:

$$\mathrm{TP} = 2, \mathrm{TN} = 5, \mathrm{FP} = 1, \mathrm{FN} = 2,$$

$$\textit{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} = \frac{2+5}{2+5+1+2} = 0.7,$$

$$\textit{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{2}{2+1} = 0.67,$$

$$\textit{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{2}{2+2} = 0.5,$$

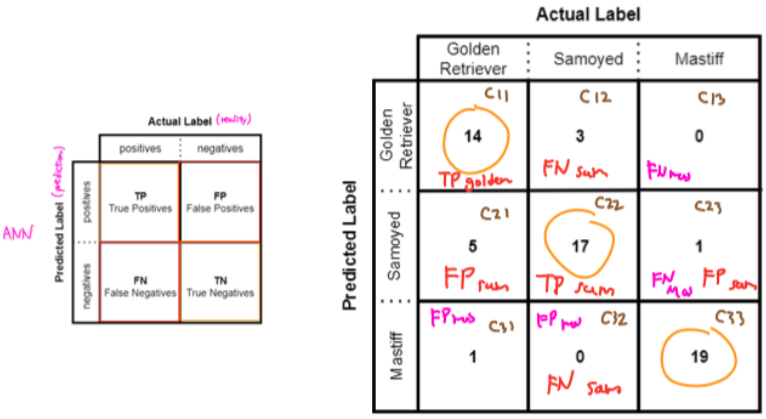
$$F_1 = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} = 2 \times \frac{0.67 \times 0.5}{0.67 + 0.5} = 0.57.$$

Comments:

- While Accuracy (0.7) and Precision (0.67) are relatively high, this is not a good classifier for this particular example
- The fact that the system is not able to catch an infected person twice (two FN) shows this is not a good classifier. One non-caught infected person, could transmit the virus to others without their knowledge (especially if no symptoms appear early on).
- The F_1 measure always takes a value between the Recall and Precision, with tendency towards the lower among either one. This is due to the fact that the F_1 measure is the harmonic average of both values.

Example 3.3.18: Multi-Class Classification

ou created a model capable of classifying images of dogs into one of three breeds: Golden Retriever, Samoyed, and Mastiff. After testing the model, the following confusion matrix was created.



To calculate various performance metrics, the TP, FP, TN, and FN have to be calculated for each label. Assume C_{ij} stands for cell in row i and column j:

	TP	FP	TN	FN
Golden Retriever	C_{11}	$C_{12} + C_{13}$	$C_{22} + C_{23} + C_{32} + C_{33}$	$C_{21} + C_{31}$
Samoyed	C_{22}	$C_{21} + C_{23}$	$C_{11} + C_{13} + C_{31} + C_{33}$	$C_{12} + C_{32}$
Mastiff	C_{33}	$C_{31} + C_{32}$	$C_{11} + C_{12} + C_{21} + C_{22}$	$C_{13} + C_{23}$

Therefore, we end up with the following values:

	TP	FP	TN	FN
Golden Retriever	14	3	37	6
Samoyed	17	6	34	3
Mastiff	19	1	39	1

These values can then be used to calculate the rest of the measures either in a one-vs-all (per-class) or all-vs-all (using the population as a whole by summing the TP, FP, TN, and FN values).

Example 3.3.17: One-vs-all

We show computation for one of the classes (Golden Retriever). Other classes are processed similarly. Total measures are the avg. rates over all the classes.

$$\text{TP} = 14, \text{TN} = 37, \text{FP} = 3, \text{FN} = 6,$$

$$\textit{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{14 + 37}{14 + 37 + 3 + 6} = 0.85,$$

$$\textit{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{14}{14 + 3} = 0.82,$$

$$\textit{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{14}{14 + 6} = 0.7,$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.82 \times 0.7}{0.82 + 0.7} = 0.76.$$

Example 3.3.18: All-vs-all

We sum all TPs, FPs, and FNs to find: TP = 50, FP = 10, FN = 10

$$\textit{Accuracy} = \frac{\text{Number of Correct Predictions (TP)}}{\text{Total Number of all Predictions}}$$

$$\textit{Accuracy} = \frac{(14+17+19)}{(14+3+0)+(5+17+1)+(1+0+19)} = 0.83,$$

$$\textit{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{50}{50+10} = 0.83,$$

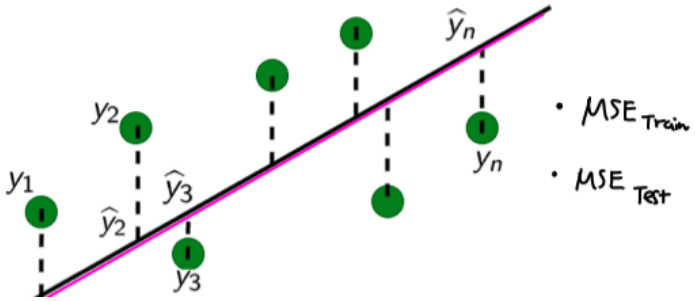
$$\textit{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{50}{50+10} = 0.83,$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.83 \times 0.83}{0.83 + 0.83} = 0.83,$$

Note: all measures become equal when we do ”all-vs-all”

3.3 Performance Measures for Regression

Definition 3.3.17: MSE (Mean Squared Error)



MSE measures error where the high errors where the high errors get bold because of the power 2.

$\Rightarrow (Outliner)^2 \rightarrow$ **Even Larger**

Treat +ve and -ve terms equally.

Equation 3.3.10: MSE

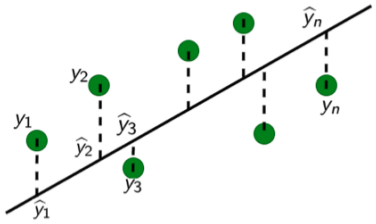
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.21}$$

RMSE (Root MSE) measures the error in the original scale of the quantity:

Equation 3.3.11: RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.22}$$

Definition 3.3.18: MAE (Mean Absolute Error)

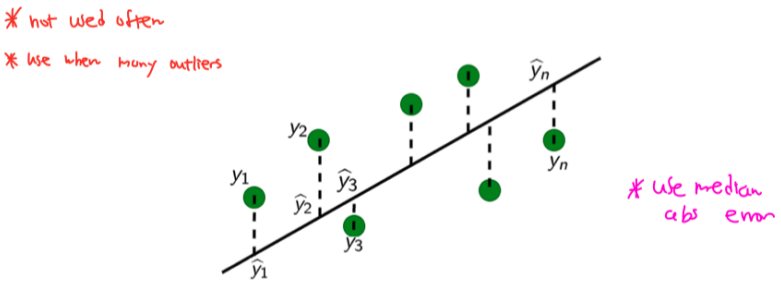


MAE measures error which is more robust to outliers than MSE because it does not have power 2. (L1 norm vs. L2 norm)

Equation 3.3.12: MAE

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.23}$$

Definition 3.3.19: MedianAE (Median Absolute Error)

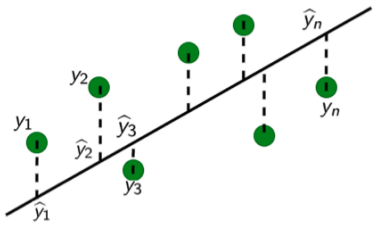


MedianAE measures error which is robust to outliers because of median operator rather than mean.

Equation 3.3.13: MedianAE

$$MAE = \text{median}\{|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|\} \tag{3.24}$$

Definition 3.3.20: R^2 Score



R^2 **Score** is similar to MSE but normalizes with the error from the mean of data.

Equation 3.3.14: MedianAE

$$R^2 = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}, \quad \text{with} \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3.25}$$

More References to Read About Performance

- Measures Summary [Python]
- Confusion Matrix [Web]

4 Support Vector Machines SVM

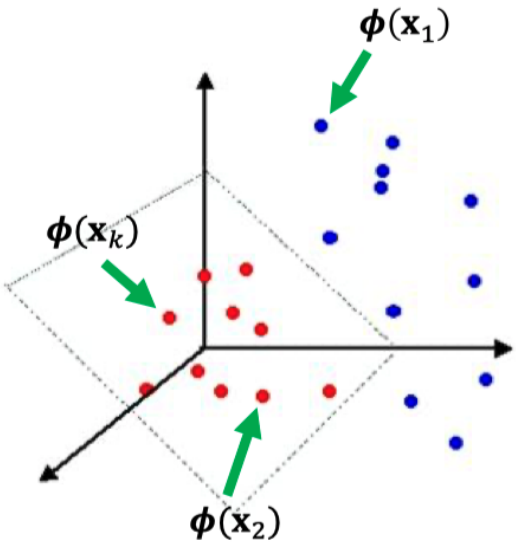
4.1 Introduction

Remark 3.4.22: SVM

- SVM is a kernel-based classifier
- Originated from the theoretical foundations of Statistical Learning Theory (SLT) and Structural Risk Minimization (SRM)
- Unlike classical approaches to minimize L1 or L2 norm of error, SVMs perform SRM to find a model with minimal VC dimension
- SVM learns during training process and generalizes to new data
- data is labeled: $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in \mathbb{R}^N xY\}$

4.2 Theory

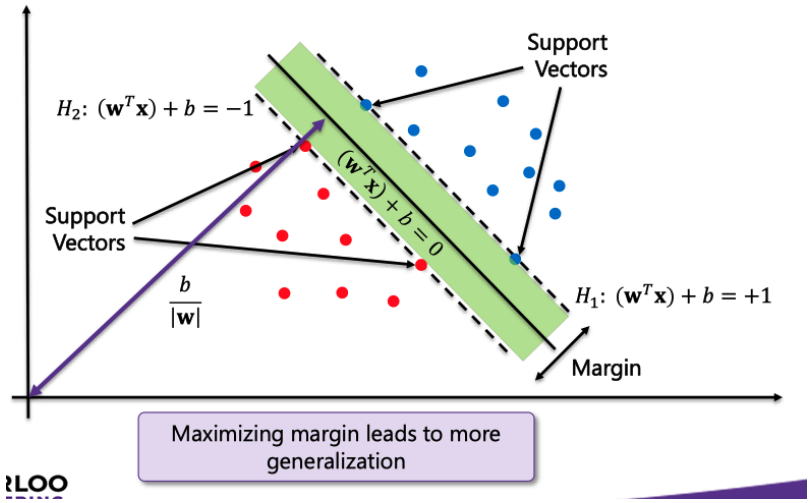
Remark 3.4.23: SVM: Basic Idea



- SVM use kernel to map data point vectors $\mathbf{x}_i$ to vectors $\phi(\mathbf{x}_i)$ in a higher-dimensional feature space
- Each $\phi(\mathbf{x}_i)$ belongs to the same group y_i as x_i does
- In the feature space, the vectors are linearly separable
- SVM obtains a hyper-plane $f(x) = \langle w, \phi(x) \rangle + b = 0$, which intersects with the manifold in the feature space to generate the decision function in the input space/
- By using a kernel, SVM does not directly work in the feature space which would be computationally expensive.

4.3 Hard Margin SVM (Original)

Definition 3.4.21: Hard Margin SVM



- Margin M: shortest distance from the discriminant hyper-plane to the closest samples from each class (support vectors), which lie on $\mathbf{w}^T\mathbf{x} + b = 0$.

$$M = \frac{|1-b|}{\|\mathbf{w}\|} - \frac{|-1-b|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{3.26}$$

- For the linearly separable case, the support vector algorithm simply looks for the separating hyper-plane with largest margin
- Thus, maximizing M translates to minimizing $\mathbf{w}$, or for mathematical convenience, $J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, i = 1, \ldots, l \tag{3.27}$$

- Hence, we are dealing with a classic quadratic programming (QP) optimization problem:

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{3.28}$$

with inequality constraints:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, i = 1, \ldots, l \tag{3.29}$$

- Reformulating the problem with Lagrange Multiplier, yield:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l} \alpha_i \left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right] \tag{3.30}$$

- The optimal saddle point (w_o, b_o, α_o) can be found using Karush Kuhn Tucker (KKT) conditions for $\alpha_i \geq 0, i = 1, \ldots, l$:

$$\frac{\partial L}{\partial \mathbf{w}_o} = 0 \rightarrow \mathbf{w}_o = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{3.31}$$

$$\frac{\partial L}{\partial b_o} = 0 \rightarrow \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.32}$$

- Now we may obtain the **dual form**:

$$L_d(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_i \tag{3.33}$$

such that: (3.34)

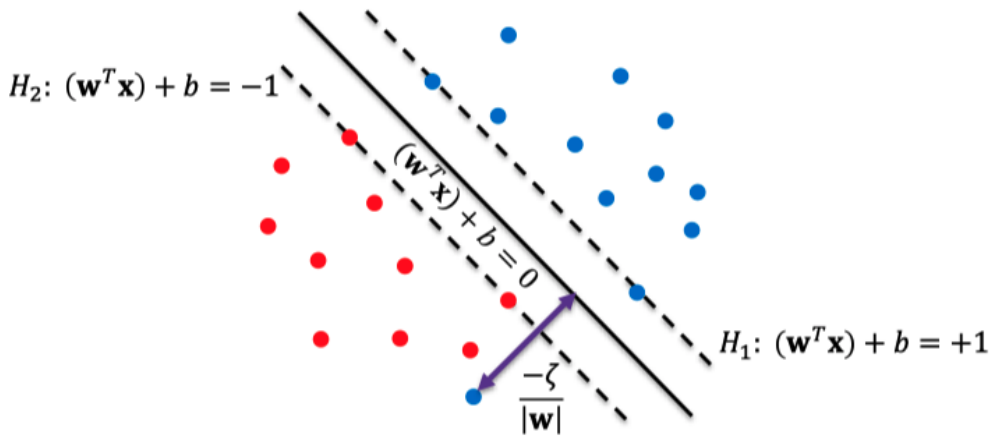
$$\sum_{i=1}^{l} \alpha_i y_i = 0 \quad \alpha_i \geq 0 \forall i = 1, \ldots, l \tag{3.35}$$

Equation 3.4.15: Final Solution with any of the standard optimization programs

$$\begin{aligned} \mathbf{w}_o &= \sum_{i=1}^{l} \alpha_o y_i \mathbf{x}_i \\ b_o &= \frac{1}{N_{sv}} \left(\sum_{s=1}^{N_{SV}} \left(\frac{1}{y_s} - \mathbf{x}_s^T \mathbf{w}_o \right) \right) \quad \forall s \in \{1, \ldots, N_{SV}\} \end{aligned} \tag{3.36}$$

4.4 Soft Margin SVM

Definition 3.4.22: Soft Margin SVM



- Optimization problem is updated to include slack variables ζ_i:

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\zeta_i \tag{3.37}$$

with inequality constraints:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \zeta_i, i = 1,\ldots,l\ \zeta_i \geq 0 \tag{3.38}$$

- Soft-margin SVM uses an error cost parameter C to address misclassification issue
- As C $\uparrow \Rightarrow$ the tolerance $\downarrow$ for misclassification of the vectors by the optimal hyperplane and a more accurate but less generalizing optimal hyper-plane is obtained, and vice versa.
- **Primal Problem**:

$$L_d(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_i \tag{3.39}$$

such that: (3.40)

$$\sum_{i=1}^{l}\alpha_i y_i = 0 \quad C \geq \alpha_i \geq 0 \forall i = 1,\ldots,l \tag{3.41}$$

- The only difference is the upper bound C on Lagrange multipliers to control overfitting
- **Primal Form**:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\zeta_i - \sum_{i=1}^{l}\alpha_i\left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \zeta_i\right] - \sum_{i=1}^{n}\beta_i\zeta_i \quad \alpha_i, \beta_i \geq 0 \forall i \in \{1,\ldots,n\} \tag{3.42}$$

- KKT conditions, derive the **Dual Form**:

$$L_d(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{3.43}$$

such that: (3.44)

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \quad C \geq \alpha_i \geq 0 \forall i = 1, \ldots, l \tag{3.45}$$

Checkout derivation from CS480 Notes

Equation 3.4.16: Final Solution with any of the standard optimization programs

$$\mathbf{w}_o = \sum_{i=1}^{l} \alpha_o y_i \mathbf{x}_i$$

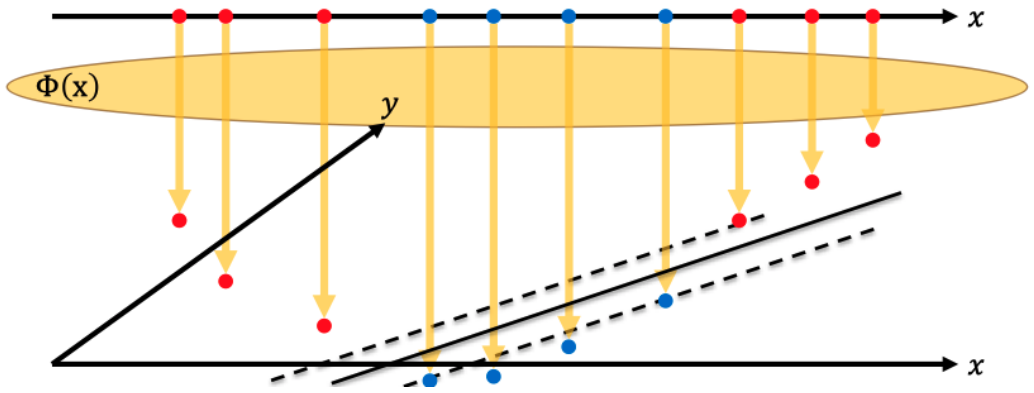
$$b_o = \frac{1}{N_{sv}} \left(\sum_{s=1}^{N_{SV}} \left(\frac{1}{y_s} 1 - \zeta_s - \mathbf{x}_s^T \mathbf{w}_o \right) \right) \; s = 1, \ldots, N_{SV} \tag{3.46}$$

with decision function:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

4.5 Kernels: Non-linear SVM

Definition 3.4.23: Non-linear Margin SVM : Kernels



- There might be cases in which the given dataset is not linearly separable using discriminant functions, hence, we may map the data to high-er-dimensional feature space, along with using kernel trick for dot product in that space.
- Linear SVM relies on dot product between two vectors x_i and x_j that might become an unmanageable operation in high-dimensional data after mapping
- A kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ provides this dot product while implicitly mapping the data points to higher-dimensional feature space

Example 3.4.19

$$K(\mathbf{x}_1, \mathbf{x}_2) = (1 + \mathbf{x}_1^T \mathbf{x}_2) = \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j) \tag{3.47}$$

$$\textbf{where: } \Phi(x) = [1, \mathbf{x}_i^2, \sqrt{2}\mathbf{x}_i\mathbf{x}_j, \mathbf{x}_j^2, \sqrt{2}\mathbf{x}_i, \sqrt{2}\mathbf{x}_j] \tag{3.48}$$

Theorem 3.4.1: Kernel

A kernel is a real-valued function $K(\mathbf{x}_1, \mathbf{x}_2)$ satisfy Mercer's condition:

There exists a mapping Φ and an expansion:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \sum_i \Phi(\mathbf{x}_1)_i \Phi(\mathbf{x}_2)_i \tag{3.49}$$

If and only if for any $g(\mathbf{x})$, such that $\int g(\mathbf{x})^2 d\mathbf{x}$ is infinite then:

$$\int K(\mathbf{x}_1, \mathbf{x}_2) g(\mathbf{x}_1) g(\mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2 \geq 0 \tag{3.50}$$

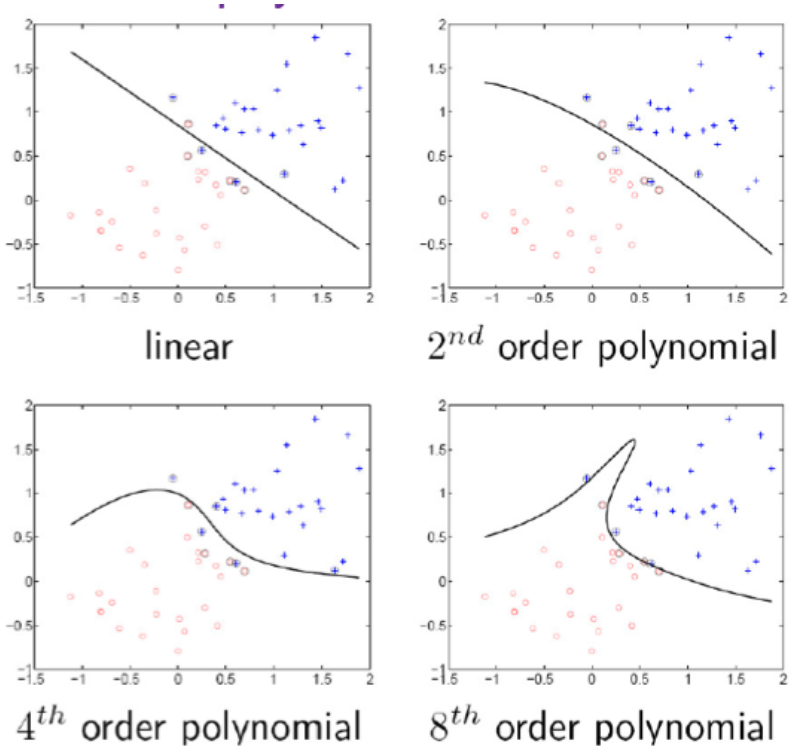
Example 3.4.20: Commonly Used Kernels

$$\textbf{Linear} \quad K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2 \tag{3.51}$$

$$\textbf{Polynomial} \quad K(\mathbf{x}_1, \mathbf{x}_2; c, d) = (c + \mathbf{x}_1^T \mathbf{x}_2)^d \tag{3.52}$$

$$\textbf{RBF (Radial-Basis Function)} \quad K(\mathbf{x}_1, \mathbf{x}_2; \sigma) = \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x}_1 - \mathbf{x}_2\|^2\right) \tag{3.53}$$

$$\textbf{Sigmoid} \quad K(\mathbf{x}_1, \mathbf{x}_2; \alpha, c) = \tanh\left(\alpha \mathbf{x}_1^T \mathbf{x}_2 + c\right) \tag{3.54}$$



More:

1. Linear Kernel 14. Power Kernel 2. Polynomial Kernel 15. Log Kernel 3. Gaussian Kernel 16. Spline Kernel 4. Exponential Kernel 17. B-Spline Kernel 5. Laplacian Kernel 18. Bessel Kernel 6. ANOVA Kernel 19. Cauchy Kernel 7. Hyperbolic Tangent (Sigmoid) Kernel 20. Chi-Square Kernel 8. Rational Quadratic Kernel 21. Histogram Intersection Kernel 9. Multiquadric Kernel 22. Generalized Histogram Intersection Kernel 10. Inverse Multiquadric Kernel 23. Generalized T-Student Kernel 11. Circular Kernel 24. Bayesian Kernel 12. Spherical Kernel 25. Wavelet Kernel 13. Wave Kernel

Remark 3.4.24: Kernels Properties

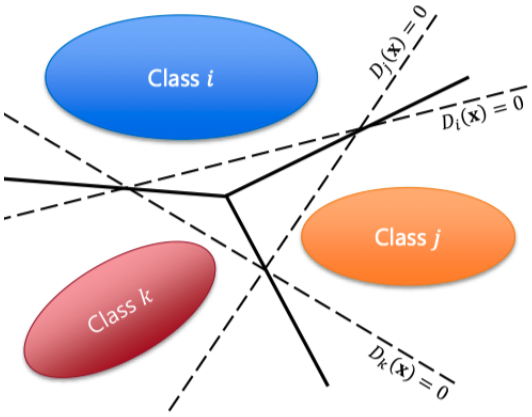
- Global kernels such as the **linear kernel** give a good prediction ability
- local kernels such as the **RNF kernel** give a good learning ability
- Every semi-positive definite symmetric function is a kernel which can be linearly combined
- Kernels can be combined to take advantage of their properties
- For instance, CombKer that is a combination of linear and RBF kernels:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1-\lambda)\langle \mathbf{x}_i, \mathbf{x}_j \rangle + \lambda \exp\{-\gamma \times \|\mathbf{x}_i - \mathbf{x}_j\|^2\} \tag{3.55}$$

As a result, it has both good prediction ability from linear kernel (global kernel) and good learning ability inherited from RBF kernel (local kernel)

4.6 Fuzzy SVM

Definition 3.4.24: Fuzzy SVM



- Same as hard-margin SVM
- Difference: Fuzzy membership functions are defined for data points near the boundary:
  - $i = j : m_{ij}(x) = \begin{cases} 1 & \forall D_j(\mathbf{x}) > 1 \\ D_j(\mathbf{x}) & \text{else} \end{cases}$
  - $m_i(\mathbf{x}) = \min_{j=1\ldots k} m_{ij}(\mathbf{x})$
  - Class label: $\text{argmax}_{i=1\ldots k} m_i(\mathbf{x})$

4.7 Summary

- SVMs provide a powerful and flexible approach to learn discriminant/approximating functions in pattern recognition tasks
- Extensions to SVMs allow for dealing with noisy, overlapping, and complex data
- Further research required on choice of Kernel functions and direct formulization of multi-class SVMs
- Much exciting research is ongoing to improve SVM. Some of the latest research topics include fuzzy SVM, genetic kernel SVM and hybridizing SVM with other soft-computing techniques such as using the genetic algorithm for initializing its parameters
- New applications of SVMs such as data fusion and density estimation are emerging

4.7.1 Pros and Cons

Remark 3.4.25: Pros and Cons of SVM

Pros:

- Flexible choice of similarity function
- well-founded theoretical approach to regularization
- SVM has sparseness and convexity properties
- SVM uses kernels to obtain complex decision functions
- SVM is able to tolerate misclassification
- Resolves overfitting through soft margin approach

Cons:

- Choice of kernel function is crucial
- SVM is vulnerability to incorrectly labeled training data
- SVM is rather sensitive to noise
- SVM has high computational complexity in the case of multi-class data
- Although SVM has good generalization performance, it can be awfully slow in test phase

4.8 Application

Example 3.4.21: SVM Applications

- Isolated Handwritten Digit Recognition
- Object Recognition
- Speaker Identification
- Bioinformatics
- Charmed Quark Detection
- Face Detection in Images
- Text Categorization
- Regression Estimation
- Time Series Prediction Tests
- Boston Housing Problem
- PET Operator Inversion Problem
- Support Vector Clustering (SVC)

4.8.1 TODO: Support Vector Clustering

Remark 3.4.26: Outline of the approach

– In data space, a transformation function is used to map the data points to a high-dimensional space (called feature space)

– In feature space, a hyper sphere with minimum radius is found that encloses a set of data points

– In data space, a set of contours are formed by mapping back the hyper sphere

– In data space, data points within each contour are associated with the same cluster

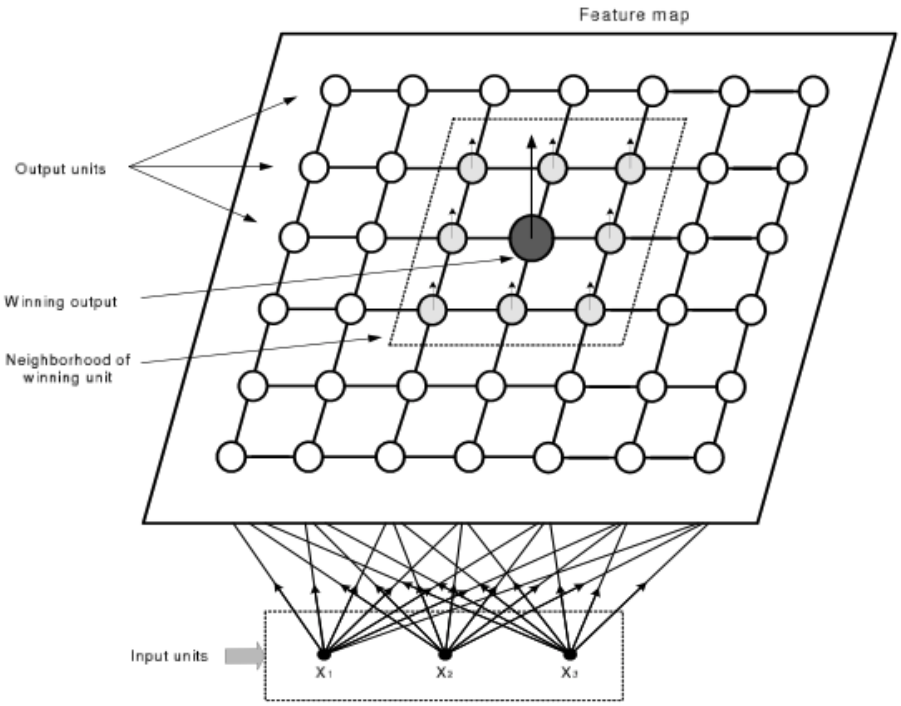
To be added from slides (pg. 35 - 45)

5 Unsupervised and Associative Memory Based

5.1 Unsupervised Learning: The Kohonen's Self-Organizing Network

5.1.1 Topology

- The Kohonen's Self-Organizing Network (KSON) also called The Kohonen's Self-Organizing Map (KSOM) belongs to the class of unsupervised learning networks.
- This means that the network, unlike supervised learning based networks updates its weighting parameters without the need for a performance feedback from a teacher or a network trainer.
- One major feature: the nodes distribute themselves across the input space to recognize groups of similar input vectors.
- Instead, using **Competitive Learning**: The output nodes compete among themselves to be fired one at a time in response to a particular input vector
- Two input vectors with similar pattern characteristics excite two physically close layer nodes $\Rightarrow$ the nodes of the KSON can recognize groups of similar input vectors.
- This generates a topographic mapping of the input vectors to the output layer, which depends primarily on the pattern of the input vectors and results in dimensionality reduction of the input space.



5.1.2 Learning Algorithm

- The learning here permits the clustering of input data into a smaller set of elements having similar characteristics (features)
- It is based on the competitive learning technique also known as the winner take all strategy
- Presume that the input pattern is given by the vector $\mathbf{x}$
- Assume w_{ij} is the weight vector connecting the input elements to an output node with coordinate provided by indices i and j.
- N_c: neighborhood around the winning output candidate, decreases at every iteration of the algorithm until

convergence occurs

Algorithm 3.5.2: KOSM Steps

1. Initialize **all weights** to small random values, set a value for the initial **learning rate** α and a value for the **neighborhood** N_c.
2. Choose an input pattern x from the input dataset
3. Select the winning unit c (the index of the best matching output unit) such that the performance index l given by the Euclidian distance from x to w_{ij} is minimized: $l = \|x - w_c\| = \min_{ij} \|x - w_{ij}\|$
4. Update the weights according to the global network updating phase from iteration k to iteration $k+1$ as:

$$w_{ij}(k+1) = \begin{cases} w_{ij}(k) + \alpha(k)[x - w_{ij}(k)] & \text{if } (i,j) \in N_c(k) \\ w_{ij} & \text{otherwise} \end{cases} \tag{3.56}$$

   $\alpha(k)$ = adaptive learning rate, $\in (0,1)$
   $N_c(k)$ = neigborhood of the unit c at iteration k

5. The learning rate and the neighborhood are decreased $\downarrow$ at every iteration according to an appropriate scheme:
   - For instance, Kohonen suggested a shrinking function: $\alpha(k) = \alpha(0)(1 - k/T)$, with T being the total number of training cycles and $\alpha(0)$ the starting learning rate bounded by one
   - As for the neighborhood, several researchers suggested an initial region with the size of half of the output grid and shrinks according to exponentially decaying behavior
6. The learning scheme continues until enough number of iterations has been reached or until each output reaches a threshold of sensitivity to a portion of the input space

Remark 3.5.27: Applications

A variety of KSONs could be applied to different applications using the different parameters of the network, which are:

- Neighbourhood size
- Shape (circular, square, diamond)
- Learning rate decaying behavior
- Dimensionality of the neuron array (1-D, 2-D, or n-D)

Given heir self-organizing capabilities based on the competitive learning rule, KSONs have been used extensively for clustering applications such as:

- Speech recognition
- Vector coding
- Robotics applications
- Texture segmentation

More References About KSOM

My personal research:

- Self-organizing Maps [Web]

5.2 Recurrent Topology: The Hop field Network

OPTIONAL*

Chapter 4

Deep Learning

1 Introduction of Deep Learning

TO PUT CONTENT

2 CNN

2.1 Introduction

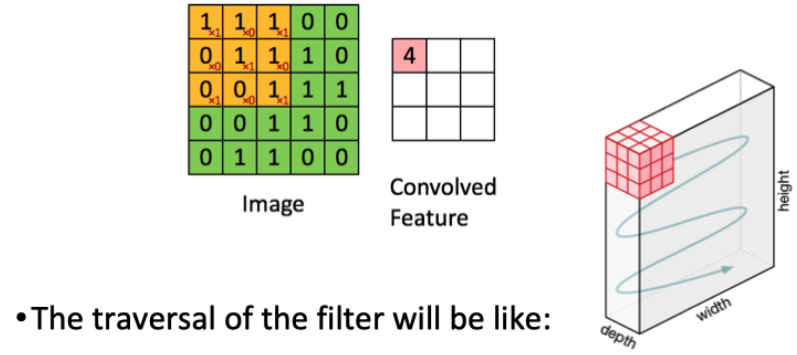
Remark 4.2.1: CNN

- a Deep Learning algorithm that takes in image as input and assign importance to various aspects in the image and be able to differentiate one from the other
- "Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers." [Goodfellow et al., 2016]
- Takes into account of spatial correlation with neighbour pixels
- CNN reduces the images into a form which is easier to process, without loosing features that are critical for a good prediction

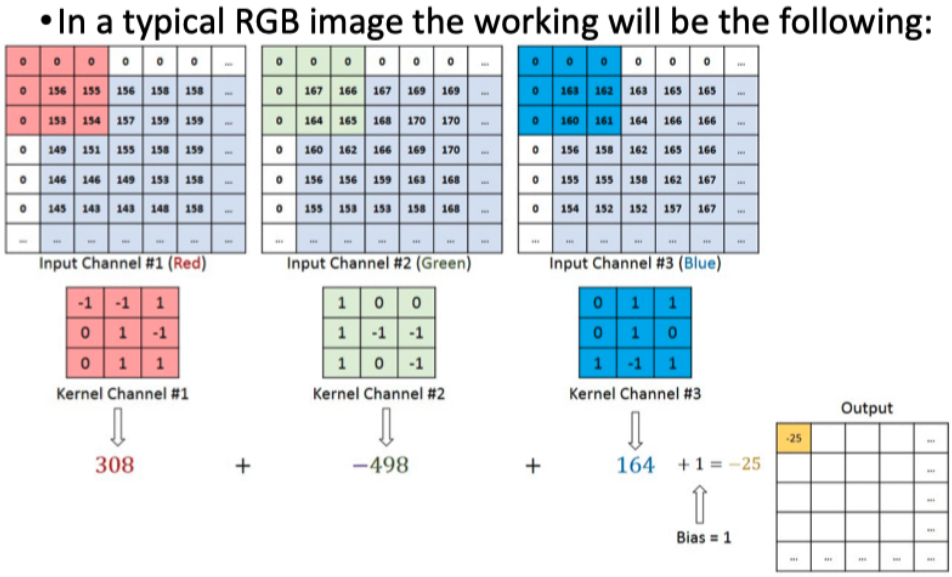
2.2 Layers

2.2.1 Convolutional Layer

To generate a feature map of image.



(a) Methodolgy



(b) Example of RGB

2.2.2 Pooling Layer

Remark 4.2.2: Types

- Max Pooling
- Average Pooling
- L2-norm pooling

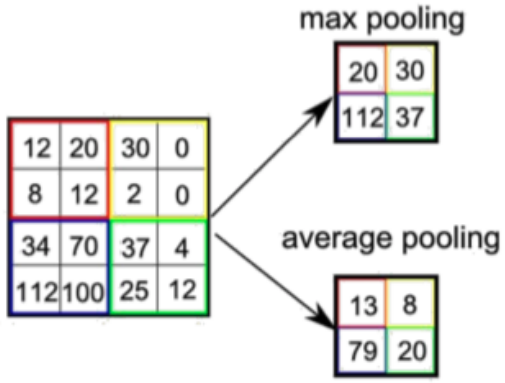


Figure 2-1. Pooling

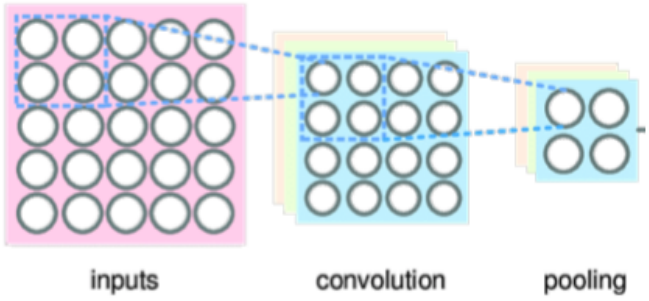
Remark 4.2.3: Effects

One might ask whether pooling will loss the information

It is found that it helps in generalizing the CNN by reducing the effects of background noise

Remark 4.2.4: Objectives

- Dimensionality reduction
- Extracting dominant features



2.2.3 FC (Fully Connected) Layer (Dense & Flatterning Layer)

Remark 4.2.5: Dense & Flatterning Layer

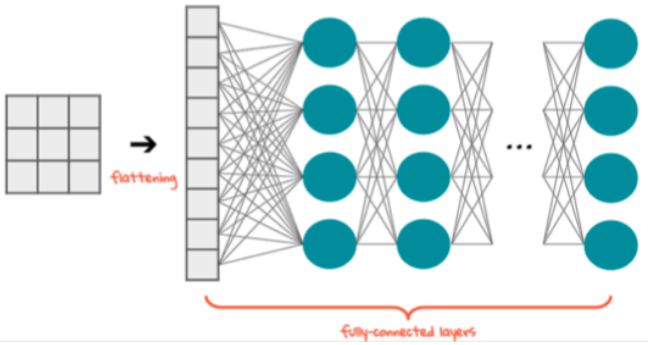
FC Layer = Flatterning Layer $\rightarrow$ Flatterning Layer

Flatterning Layer:

- Converts data $\rightarrow$ 1D
- Prep. data that can be fed to dense layer

Dense Layer:

- It connect every neuron in one layer to every neuron in another layer
- ONce trained, it can be like classifier



2.3 Regularization

Goal: to make model generalized and avoid the model over-fitting

2.3.1 Dropout

- Dropout is a method to randomly set some activations to 0 (killing or dropping)
- Dropout forces the network to explore more ways of learning patterns instead of depending too much on some features

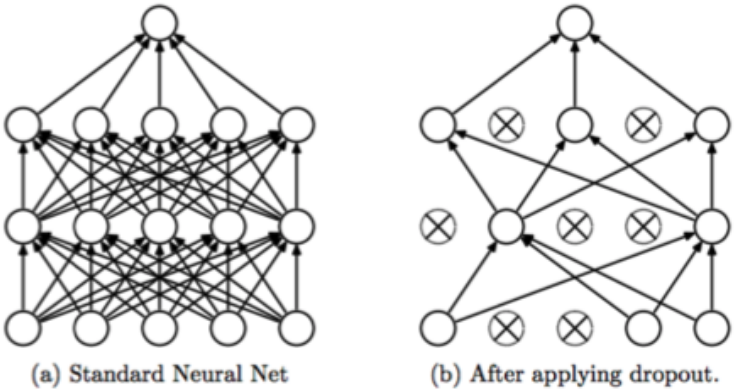


Figure 2-2. Dropout

2.3.2 Batch Normalization

- This works by normalizing every batch of the image to have zero mean and unit variance
- Improves the performance and training speed

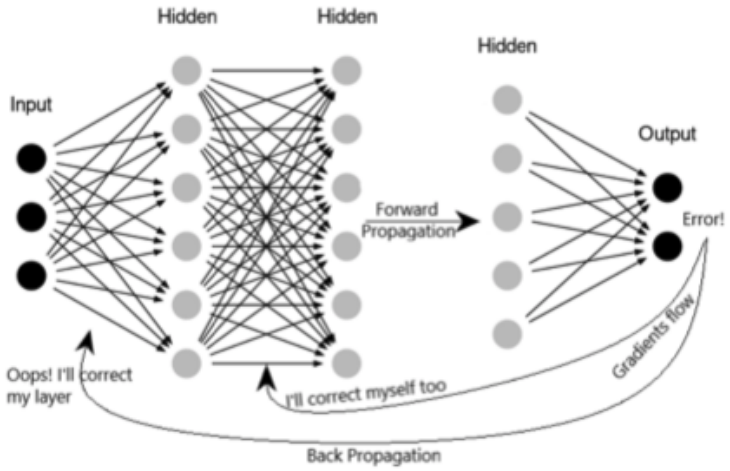


Figure 2-3. Batch Normalization

2.3.3 Data Augmentation

- Making minor modification to the samples thereby populating more data samples
- The best way to improve generalization is to have more training data. But often, data is limited. =¿ Create fake data

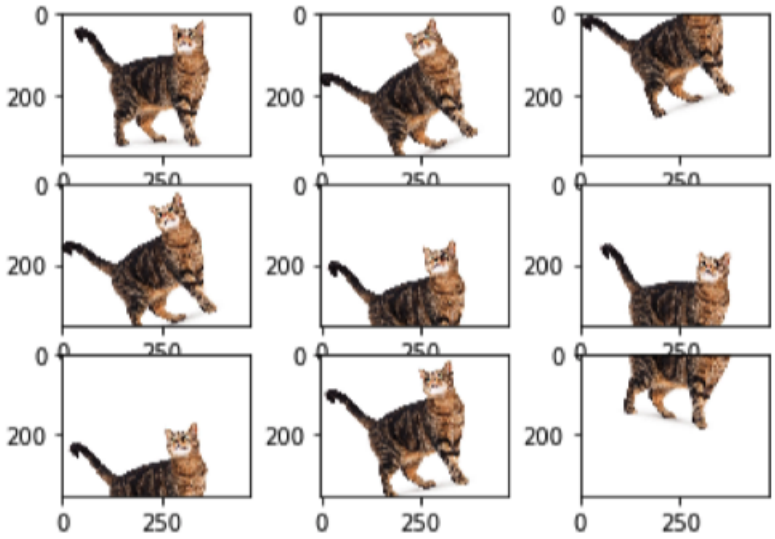


Figure 2-4. Data Augmentation

2.4 Functions

2.4.1 Activation Functions

- It is used to determine the output of neural network like yes or no.
- It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

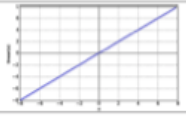
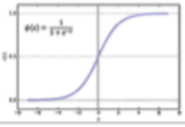
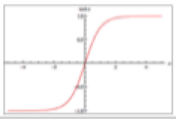
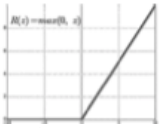
Activation Function	Graphical representation	Function
Linear		Allows multiple output
Sigmoid		Predicts the probability of the probability
tanh		Squeezes the data between -1 to +1 thereby, making the training easy
ReLU		It doesn't allow all the neurons to be activated all the time thereby, making network efficient

Figure 2-5. Activation Functions

2.4.2 Objective Functions

- An objective function gives a formal specification to the problem
- Generally it can be regarded as the function that is optimized during the training of neural network
- Some common objective functions are: MSE, Mean-Absolute-error, Cross-entropy loss

2.4.3 Optimiziers

- Optimizers are essential algorithms that help to solve predictions that have minimum cost
- Optimizers find parameters that significantly reduce the cost function while training and updating
- Commonly used: Adam, Stochastic Gradient Descent, AdaGrad, RMSProp

2.5 Problems

2.5.1 Vanishing Gradient

- A common issue with the CNN is the vanishing gradient problem
- While back-propagating if the gradient is found to less than 1, the weights in the earlier layer don't get updated resulting in poor performance
- Opposite to this is exploding gradient where the gradient is found to be large making the model unstable

2.5.2 Overfitting

- Overfitting is a modeling error that occurs when a function is too closely fit to a limited set of data points
- The opposite of overfitting is underfitting which is when the model fails to learn causing high training and testing errors

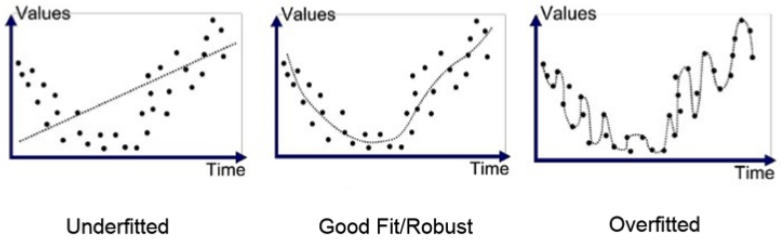


Figure 2-6. Overfitting

2.6 Extra Reference

More References About CNN

My personal research:

- CNN

Chapter 5

Fuzzy Logic

1 Introduction of Fuzzy Logic Systems

1.1 Brief:

1.1.1 Problem of Crisp Logic Rep:

- Two-state (bivalent) crisp logic uses two quantities: T/F
- Real-life: ambiguity and partial truth
- Linguistic descriptors are not crisp quantities, and tend to be quite subjective, approximate, and qualitative

1.1.2 Property of Fuzzy Rep:

- Differentiable
- Continuous
- A fuzzy set rep. as a set of ordered pairs :
  $A = \{(\mathbf{x}, \mu_A(\mathbf{x})); \mathbf{x} \in X, \mu_A(\mathbf{x}) \in [0,1]\}$
  where $\mu_A(\mathbf{x})$ is the membership function of the fuzzy set A, which represents the grade of possibility that an element x belongs to the fuzzy set A.
  - $\mu_A(\mathbf{x}) = 1 \quad \Rightarrow$ x is definitely an element of A
  - $\mu_A(\mathbf{x}) = 0 \quad \Rightarrow$ x is definitely not an element of A
  - $\mu_A(\mathbf{x}) = 0.2 \quad \Rightarrow$ x is 20% possibly to be an element of A
- (Discrete) A fuzzy set A: $A = \mu_A(x_1)/x_1 + \ldots \mu_A(x_n)/x_n = \sum_{x_i \in X} \mu_A(x_i)/x_i$
- (Continuous) A fuzzy set A: $A = \int_{x_i \in X} \mu_A(x_i)/x_i$
- $\sum$ and $\int$ do not represent summation/integration , but rather collection of members in discrete or continuous domain

1.1.3 Member functions:

1.1.3.1 Triangular

$$triangle(x|a,b,c) = \begin{cases} 0 & ,x \leq a \\ \frac{x-a}{b-a} & ,x \in (a,b] \\ \frac{c-x}{c-b} & ,x \in (b,c] \\ 0 & ,x > c \end{cases} = \max\left\{\min\left\{\frac{x-a}{b-a}, \frac{c-x}{c-b}\right\}, 0\right\} \tag{5.1}$$

1.1.3.2 Trapezoidal

$$trapezoid(x|a,b,c,d) = \begin{cases} 0 & ,x \leq a \\ \frac{x-a}{b-a} & ,x \in (a,b] \\ 1 & ,x \in (b,c] \\ \frac{d-x}{d-c} & ,x \in (c,d] \\ 0 & ,x > c \end{cases} = \max\left\{\min\left\{\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right\}, 0\right\} \tag{5.2}$$

1.1.3.3 Gaussian

$$gauss(x|\sigma,c) = \exp\left\{-0.5(\frac{x-c}{\sigma})^2\right\} \tag{5.3}$$

with σ: width, c: centre

1.1.3.4 Generalized Bell

$$gbell(x|a,b,c) = \frac{1}{1+|\frac{x-c}{a}|^{2b}} \tag{5.4}$$

with a: width, b: slope, c: the centre of the membership

1.1.3.5 Comment:

Remark 5.1.1: Member Functions Pro/Cons

- Triangular and trapezoidal: Linear $\rightarrow$ Non-Differentiable $\Rightarrow$ Computationally inexpensive
- Gaussian and Bell: Non-Linear $\rightarrow$ Differentiable $\Rightarrow$ Suitable to use with gradient-descent optimization

1.2 Fuzzy Logic Operation (Discrete, Basic)

1.2.1 Fuzzy Complement (Negation, Not)

$$\mu'_A(x) = 1 - \mu_A(x), x \in X \tag{5.5}$$

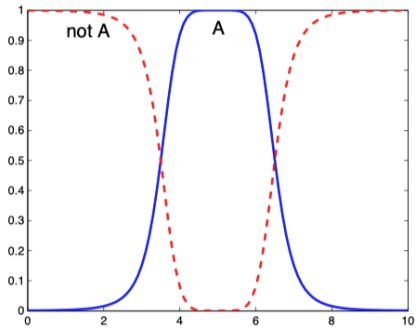


Figure 1-1. Fuzzy Complement Graphical Example

1.2.2 Union

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)), \forall x \in X \tag{5.6}$$

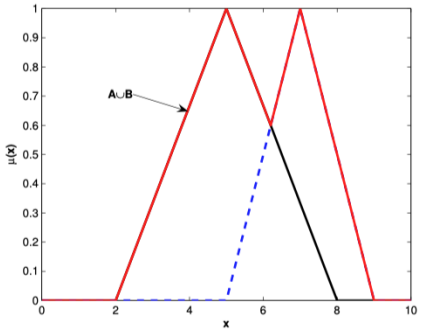


Figure 1-2. Fuzzy Union Graphical Example

1.2.3 Intersection

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)), \forall x \in X \tag{5.7}$$

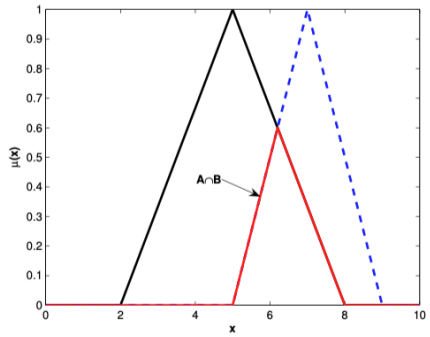


Figure 1-3. Fuzzy Intersection Graphical Example

1.3 Fuzzy Logic Operation (Generalized)

1.3.1 Generalized Fuzzy Complement

A generalized complement operation, denoted by $C : [0,1] \to [0,1]$ shall satisfy axioms:

- Boundary conditions: $C() = X$ and $C(X) =$, where X is the universe of discourse and is the null set
- Non-increasing: if $\mu_A(x) < \mu_B(x)$ then $C(\mu_A(x)) \geq C(\mu_B(x))$, and vice versa
- Involutive: $C(C(A)) = A$

1.3.1.1 Sugeno's Complement:

$$C(a) = \frac{1-a}{1+pa}, p \in (-1, \infty) \tag{5.8}$$

1.3.1.2 Yager Complement:

$$C(a) = (1-a^p)^{1/p}, p \in (0, \infty) \tag{5.9}$$

1.3.2 T-norm (Generalized Fuzzy Intersection)

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)) \tag{5.10}$$

- The intersection of two fuzzy sets A and B is given by an operation T which maps two membership functions to Equation (5.10)
- $T(\cdot)$ is known as the T-norm operator
- Consider two membership functions that are given by $a = \mu_A(x)$ and $b = \mu_B(x)$
- The t-norm operation or generalized intersection may be represented by $T(a,b)$ or more commonly aTb

1.3.2.1 T-norm Properties:

- Non-decreasing in each argument. i.e., if $a \leq b$ and $c \leq d$ then $aTc \leq bTd$.
- Commutativity: i.e., $aTb = bTa$
- Associativity: i.e., $(aTb)Tc = aT(bTc)$
- Boundary Conditions: i.e., $aT1 = a$ and $aT0 = 0$ $\Rightarrow$ take minimum

1.3.2.2 Two General Forms of T-norms:

$$1 - \min\left\{1, ((1-a)^p + (1-b)^p)^{1/p}\right\}, p \geq 1 \tag{5.11}$$

$$\max\left\{0, (\lambda+1)(a+b-1) - \lambda ab\right\}, \lambda \geq -1 \tag{5.12}$$

1.3.2.3 Four well known T-norms operators:

$$\textbf{Min} \quad T(a,b) = \min(a,b) = a \wedge b \tag{5.13}$$

$$\textbf{Algebraic product} \quad T(a,b) = axb \tag{5.14}$$

$$\textbf{Bounded product} \quad T(a,b) = 0 \vee (a+b-1) \tag{5.15}$$

$$\textbf{Basic product} \quad T(a,b) = \begin{cases} a & , \text{if } b = 1 \\ b & , \text{if } a = 1 \\ 0 & , \text{if } a,b < 1 \end{cases} \tag{5.16}$$

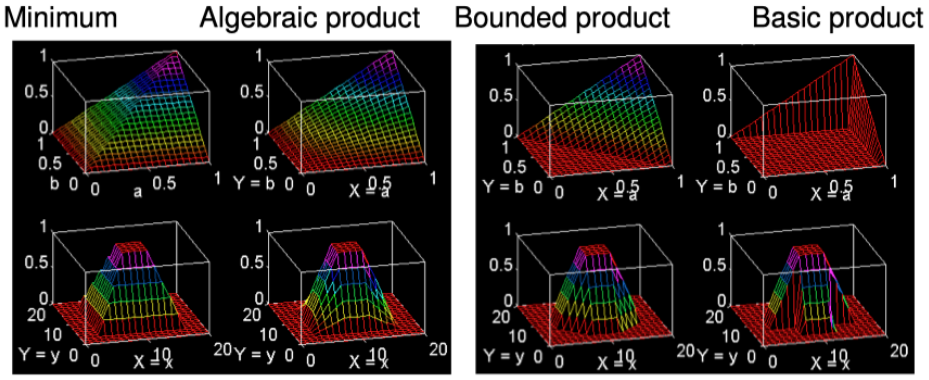


Figure 1-4. Four T-norm Operators Graphical Example

1.3.3 S-norm (Generalized Fuzzy Union)

$$\mu_{A \cup B}(x) = S(\mu_A(x), \mu_B(x)) \tag{5.17}$$

- The union of two fuzzy sets A and B is given by an operation S which maps two membership functions to Equation (5.17)
- $S(\cdot)$ aka S-norm operator, or T-conorm operator

1.3.3.1 DeMorgan's Law

Theorem 5.1.1: DeMorgan's Law

There exists a complementary $S-norm$ associated to every $T-norm$, and vice versa.

$$aSb = \overline{\bar{a}\mathrm{T}\bar{b}} = 1-(1-a)T(1-b) \tag{5.18}$$

$$aTb = \overline{\bar{a}\mathrm{S}\bar{b}} = 1-(1-a)S(1-b) \tag{5.19}$$

$$\overline{aSb} = \bar{a}T\bar{b} \tag{5.20}$$

1.3.3.2 S-norm Properties:

- Non-decreasing in each argument. i.e., if $a \leq b$ and $c \leq d$ then $aSc \leq bSd$.
- Commutativity: i.e., $aSb = bSa$

- Associativity: i.e., $(aSb)Sc = aS(bSc)$
- Boundary Conditions: i.e., $aS1 = 0$ and $aS0 = a$ $\Rightarrow$ take maximum

Proof 5.1.1: S-norm associated to the T-norm min is max

- Use DeMorgan's Law (Theorem 5.1.1): $aSb = \overline{\bar{a}T\bar{b}} = 1-(1-a)T(1-b)$
- Direct substitution of min in T: $aSb = 1-\min[(1-a),(1-b)] = \max(a,b)$
- **Q.E.D.**

1.3.3.3 Two General Forms of S-norms:

$$\min\left\{1,((a)^p+(b)^p)^{1/p}\right\}, p \geq 1 \tag{5.21}$$

$$\min\{1, a+b+\lambda ab\}, \lambda \geq -1 \tag{5.22}$$

1.3.3.4 Four well known T-norms operators:

$$\textbf{Max} \quad T(a,b) = \max(a,b) = a \vee b \tag{5.23}$$

$$\textbf{Algebraic product} \quad T(a,b) = a+b-ab \tag{5.24}$$

$$\textbf{Bounded product} \quad T(a,b) = 1 \vee (a+b) \tag{5.25}$$

$$\textbf{Basic product} \quad T(a,b) = \begin{cases} a & , \text{if } b = 0 \\ b & , \text{if } a = 0 \\ 1 & , \text{if } a,b < 1 \end{cases} \tag{5.26}$$

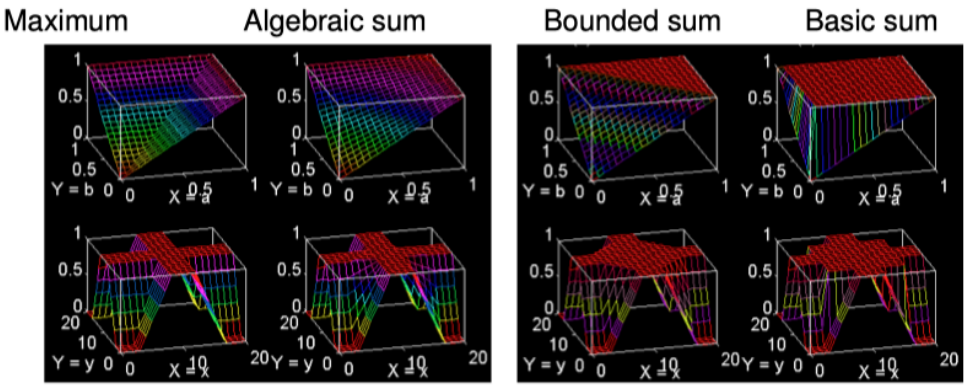


Figure 1-5. Four S-norm Operators Graphical Example

1.4 Fuzzy Logic Operation (More)

1.4.1 Grade of Inclusion

In the context of fuzzy logic, a set may be partially included in another set.

- Then, it is convenient to define a grade of inclusion of fuzzy set A in another fuzzy set B.
- This may be interpreted as the membership function of the fuzzy relation $A \subset B$ (or $A \subseteq B$)
- Specifically, a fuzzy set A is considered a subset of another fuzzy set B in universe X, iff $\mu_A(x) \leq \mu_B(x), \forall x \in X$. Denoted as $A \subseteq B$.

$$\mu_{A \subset B}(x) = \begin{cases} 1 & \text{, if } \mu_A(x) < \mu_B(x) \\ \mu_A(x) T \mu_B(x) & \text{, otherwise} \end{cases} \tag{5.27}$$

$$\mu_{A \subseteq B}(x) = \begin{cases} 1 & \text{, if } \mu_A(x) \leq \mu_B(x) \\ \mu_A(x) T \mu_B(x) & \text{, otherwise} \end{cases} \tag{5.28}$$

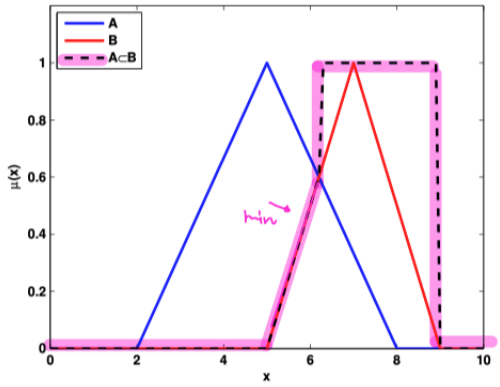


Figure 1-6. Grade of Inclusion Graphical Example

1.4.2 Grade of Equality

- A grade of equality for two fuzzy sets may be defined similar to the grade of inclusion
- This may be interpreted as the membership function of the fuzzy relation $A = B$

$$\mu_{A=B} = \begin{cases} 1 & \text{, if } \mu_A(x) = \mu_B(x) \\ \mu_A(x) T \mu_B(x) & \text{, otherwise} \end{cases} \tag{5.29}$$

1.4.3 Dilation and Contraction

Let A be a fuzzy set in the universe X with membership function μ_A

- Its Kth dilation is a fuzzy set A' in the universe X with membership function $\mu'_A(x) = \mu_A^{1/K}(x)$
- Its Kth contraction is a fuzzy set A'' in the universe X with membership function $\mu''_A(x) = \mu_A^K(x)$

$$dilation(A) = A^{1/2} = \int \frac{\sqrt{\mu_A(x)}}{x} \equiv moreorless \equiv somehow \tag{5.30}$$

$$contraction(A) = A^2 = \int \frac{(\mu_A(x))^2}{x} \equiv very \equiv too \tag{5.31}$$

1.4.4 Implication (if-then)

- Consider two fuzzy sets: A in X and B in Y
- The fuzzy implication $A \rightarrow B$ is a fuzzy relation in the Cartesian product $X \times Y$

1.4.4.1 Types of Implications

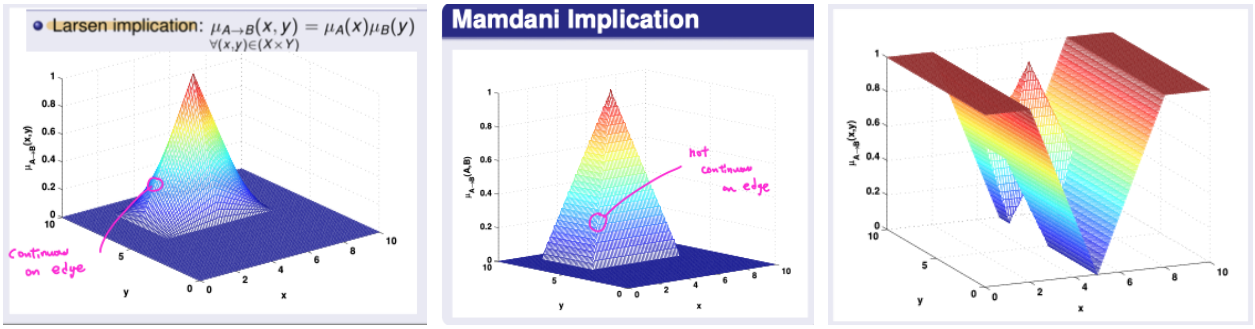
$$\textbf{Larsen implication:} \quad \underset{\forall(x,y)\in(X\times Y}{\mu_{A\rightarrow B}}(x,y) = \mu_A(x)\mu_B(y) \tag{5.32}$$

$$\textbf{Mamdai implication:} \quad \underset{\forall(x,y)\in(X\times Y}{\mu_{A\rightarrow B}}(x,y) = \min\{\mu_A(x), \mu_B(y)\} \tag{5.33}$$

$$\textbf{Zadeh implication:} \quad \underset{\forall(x,y)\in(X\times Y}{\mu_{A\rightarrow B}}(x,y) = \max\{\min[\mu_A(x), \mu_B(y)], 1-\mu_A(x)\} \tag{5.34}$$

$$\textbf{Dienes-Rascher implication:} \quad \underset{\forall(x,y)\in(X\times Y}{\mu_{A\rightarrow B}}(x,y) = \max\{1-\mu_A(x), \mu_B(y)\} \tag{5.35}$$

$$\textbf{Lukasiewicz implication:} \quad \underset{\forall(x,y)\in(X\times Y}{\mu_{A\rightarrow B}}(x,y) = \min\{1, 1-\mu_A(x)+\mu_B(y)\} \tag{5.36}$$



(a) Larsen (b) Mamdai (c) Zadeh

(d) Dienes-Rascher (e) Lukasiewicz

Figure 1-7. Types of Implications Graphical Representation

1.5 Fuzzy Logic Properties

1.5.1 Height of a Fuzzy Set

$$hgt(A) = \sup_{x \in X} \mu_A(x) \tag{5.37}$$

- The height (aka modal grade) of a fuzzy set is the maximum value of its membership function.
- The element $x* \in X$ corresponding to the modal grade of the fuzzy set (i.e., $\mu_A(x*) = hgt(A)$ is called the modal element value, or simply modal point.

1.5.2 Support Set

$$S = \{x \in X | \mu_A(x) > 0\} \tag{5.38}$$

- The **support set** of a fuzzy set is a crisp set containing all the elements (in the universe whose membership grades are greater than zero.

1.5.3 α-cut of a Fuzzy Set

$$A_\alpha = \{x \in X | \mu_A(x) \geq \alpha\},\ \alpha \in [0,1] \tag{5.39}$$

- The **α-cut** of a fuzzy set A is the crisp set denoted by A_α formed by the elements of A, whose membership function grades are greater than or equal to a specified threshold value $\alpha \in [0,1]$.

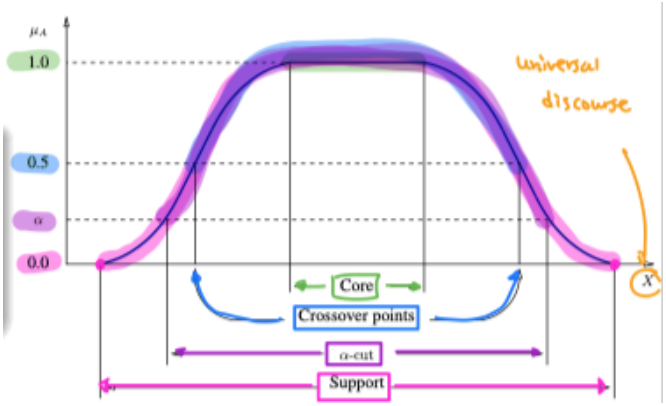


Figure 1-8. *alpha*-cut Graphical Example

1.5.4 Three Different Measures of Fuzziness

A measure of fuzziness of a fuzzy set A in the universe X defines the closeness of its membership function μ_A to the most fuzzy grade (0.5) (cross-over point).

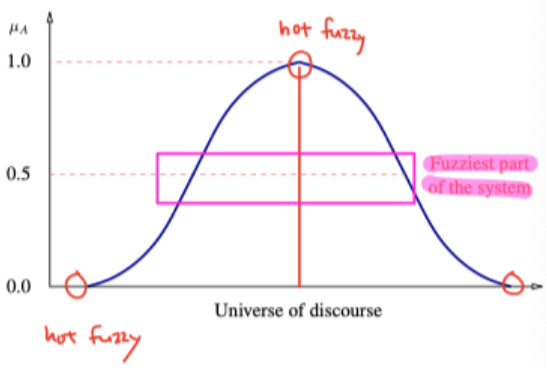


Figure 1-9. Fuzziness Graphical Example

1.5.4.1 Closeness to grade $0.5(A_1)$

$$A_1 = \int_{x \in S} f(x)dx \quad , S \text{ denotes the support set} \tag{5.40}$$

$$f(x) = \begin{cases} \mu_A(x) & , \text{ if } \mu_A(x) \leq 0.5 \\ 1 - \mu_A(x) & , \text{ otherwise} \end{cases} \tag{5.41}$$

1.5.4.2 Distance from $1/2$-cut(A_2)

$$A_2 = \int_{x \in X} \|\mu_A(x) - \mu_{A_{1/2}}(x)\|_1 dx \tag{5.42}$$

1.5.4.3 Inverse of distance from the complement (A_3)

$$A_3 = 2\int_{x \in X} \|\mu_A(x) - 0.5\|_1 dx = \int_{x \in X} |2\mu_A(x) - 1|dx = \int_{x \in X} |\mu_A(x) - (1 - \mu_A(x))|dx = \int_{x \in X} |\mu_A(x) - \mu_{\bar{A}}(x)|dx \tag{5.43}$$

1.5.4.4 Relationship Between 3 Different Measures of Fuzziness

$$A_1 = A_2 = \frac{1}{2}(S * 1 - A_3) \tag{5.44}$$

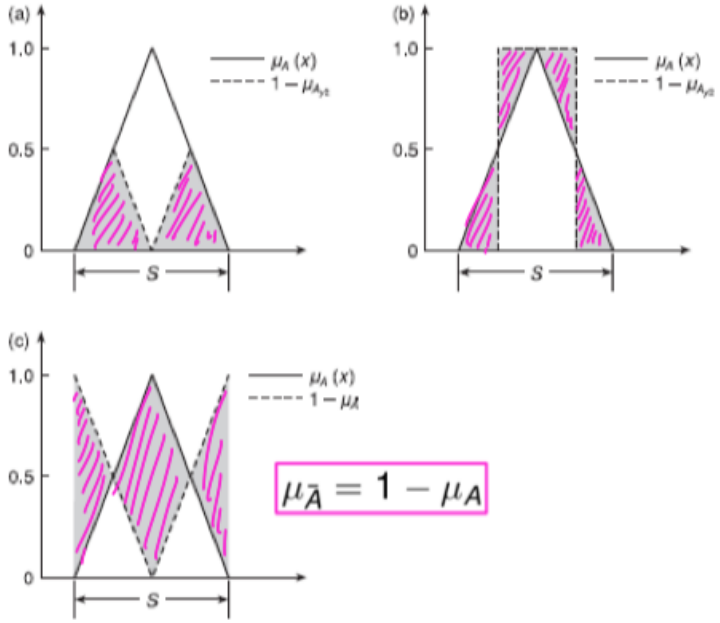


Figure 1-10. Three Measures of Fuzziness Illustration $(a)A_1, (b)A_2, (c)A_3$

1.6 Fuzzy Relation

See Examples in slide; pg 68 - 71

1.6.1 Analytical Representation of a Fuzzy Relation

- Any membership function represents a fuzzy relation int he universe (domain, or space) of definition of the particular membership function.
- The space can be one dimensional; say the real line $\mathbb{R}$ as in the case of $\mu_R(x)$, and this gives a 1-D fuzzy relation.
- This idea can be extended to higher dimensions.
- Consider two universes $X_1 = x_1$ and $X_2 = x_2$
- A crisp set consisting of a subset of ordered pairs (x_1, x_2) is a crisp relation R in the 2D Cartesian product space $X_1 \times X_2$
- We may imagine that a truth value of 1 is associated to each of these ordered pairs, giving the characteristics function (special case of membership function) of the crisp relation.

1.6.2 Projection

- Given the relation
  $R = \int_{X\times Y} \overline{\mu}_R(x,y)(x,y)$
- The first project is a fuzzy set that results by eliminating the second set Y of $X \times Y$ by projecting the relation on X:

$$R_1 = \int_X \frac{\mu_{R_1}(x)}{x}, \quad \mu_{R_1}(x) = \bigvee_y \max[\mu_R(x,y)] \tag{5.45}$$

- The second projection is a fuzzy set that results by eliminating the first set X of $X \times Y$ by projecting the relation on Y:

$$R_2 = \int_Y \frac{\mu_{R_2}(y)}{y}, \quad \mu_{R_2}(y) = \bigvee_x \max[\mu_R(x,y)] \tag{5.46}$$

- The **total projection** is a combined projection over the spaces X and Y:

$$\mu_{R_T}(x,y) = \bigvee_x \bigvee_y \max[\mu_R(x,y)] = \bigvee_y \bigvee_x \max[\mu_R(x,y)] \tag{5.47}$$

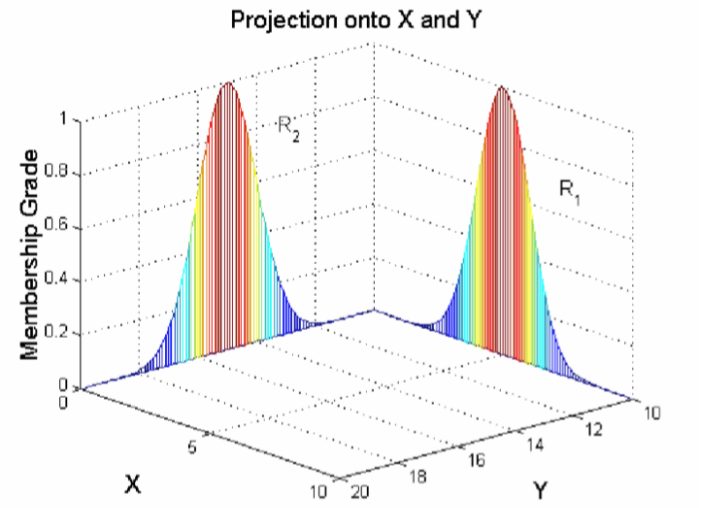


Figure 1-11. First and second projections of relation R

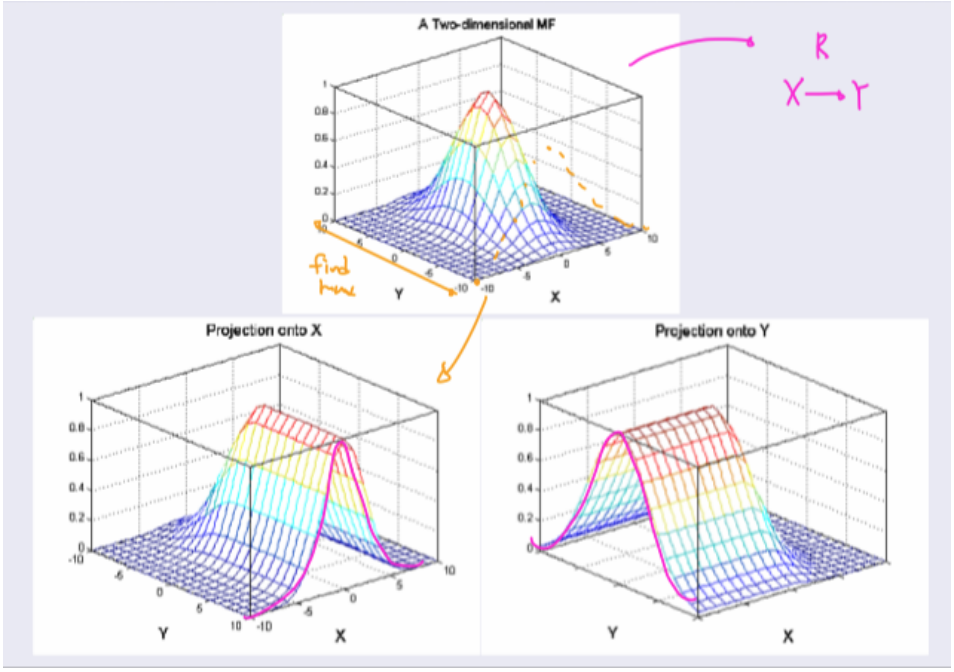


Figure 1-12. Continuous Case

$$R = \underset{y}{\begin{bmatrix} 0.1 & 0.2 & 0.4 & 0.8 & 1.0 & 0.6 \\ 0.2 & 0.4 & 0.8 & 0.9 & 0.8 & 0.6 \\ 0.5 & 0.9 & 1.0 & 0.8 & 0.4 & 0.2 \end{bmatrix}} \quad x$$

$$R^1 = \sum_i \frac{\mu_{R^1}(x_i)}{x_i} = \frac{1.0}{x_1} + \frac{0.9}{x_2} + \frac{1.0}{x_3} = \begin{bmatrix} 1.0 \\ 0.9 \\ 1.0 \end{bmatrix}$$

$$R^2 = \sum_j \frac{\mu_{R^2}(y_j)}{y_j} = \frac{0.5}{y_1} + \frac{0.9}{y_2} + \frac{1.0}{y_3} + \frac{0.9}{y_4} + \frac{1.0}{y_5} + \frac{0.6}{y_6}$$

$$= \begin{bmatrix} 0.5 & 0.9 & 1.0 & 0.9 & 1.0 & 0.6 \end{bmatrix}^T$$

$$R^T = 1$$

Figure 1-13. Discrete Case

1.6.3 Cylindrical Extension

$$C(R) = \int_{X_1 \times X_2 \times \cdots \times X_n} = \frac{\mu_R(x_1, x_2, \ldots, x_n)}{(x_1, x_2, \ldots, x_n)} \tag{5.48}$$

see example in slides Pg 90-91

1.6.4 Compositional Rule of Inference

- A typical fuzzy logic system's knowledge base is composed of a series of if-then rules.
- Each rule is a fuzzy relation employing the (if-then) fuzzy implication.
- When all individual rules (relations) are aggregated (executed), they result in one fuzzy relation represented by a fuzzy set, say K, and a multi-variable membership function.

Definition 5.1.1: Fuzzy Inference : conducted based on the compositional rule of inference (CRI)

1. In a fuzzy decision making process, the rule base K is first collectively matched with the available data.
2. Next, an inference is made on another fuzzy variable (consequent part) that is represented in the knowledge base.

Suppose that the available data (context) is denoted by the fuzzy set (or relation) D and the inference (action) is denoted by a fuzzy set (or relation) I.

1. Then, the compositional rule of the inference states that:

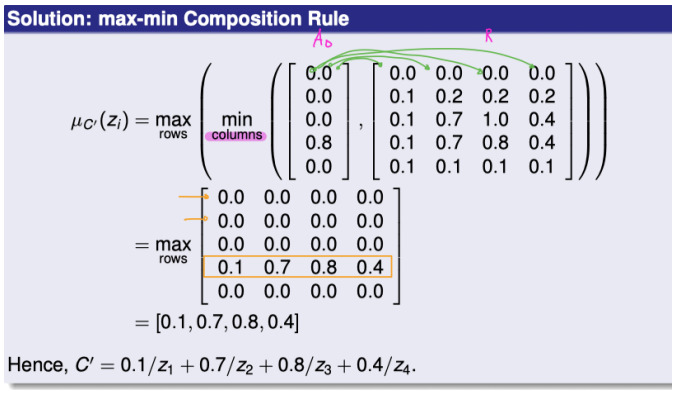
$$I = D \circ K, \quad \circ : \text{composed with} \tag{5.49}$$

2. The membership function of the inference I (also called decision, or action) is determined as

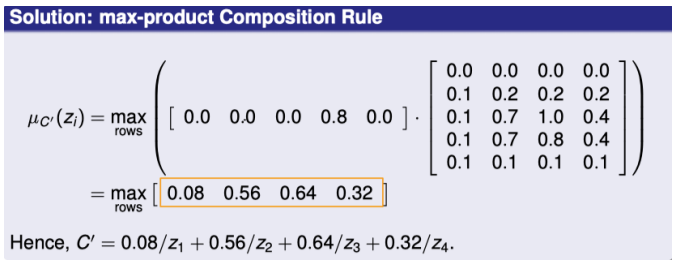
$$\mu_I = \sup_y \{\min[\mu_D, \mu_K]\} \tag{5.50}$$

   where the inference I is the output of the knowledge base decision making system.
3. This is known as the **sup-min** (or max-min) composition.
4. Another method to compute the inference I is through the **sup-product** (or max-product) composition:

$$\mu_I = \sup_y \{\mu_D \cdot \mu_K\} \tag{5.51}$$



(a) max-min



(b) max-product

Figure 1-14. Composition Rule Examples

1.6.5 Properties of Composition

- In general, any **sup-T** operator can be used to perform the compositional rule of inference, where T is a T-norm operator
- The sup-min and sup-product compositions are special cases of the **sup-T composition**, where the min and product operations are used as T-norms, respectively

- Let P and R be two fuzzy sets represented by two membership functions $\mu_p(x,z)$ and $\mu_R(z,y)$.

1.6.5.1 Sup-T and Inf-S Compositions

Definition 5.1.2: Sup-T Compositions

$$\mu_{P\circ R}(x,y) = \sup_{z\in Z}[\mu_p(x,z)T\mu_R(z,y)] \quad T: \textbf{T-norm} \tag{5.52}$$

Definition 5.1.3: Inf-S Compositions

$$\mu_{P\otimes R}(x,y) = \inf_{z\in Z}[\mu_p(x,z)S\mu_R(z,y)] \quad S: \textbf{S-norm} \tag{5.53}$$

1.6.5.2 Commutativity:

$$P\circ R = R\circ P \tag{5.54}$$
$$P\otimes R = R\otimes P \tag{5.55}$$

1.6.5.3 Associativity:

$$P\circ(Q\circ R) = (P\circ Q)\circ P \tag{5.56}$$
$$P\otimes(Q\otimes R) = (P\otimes Q)\otimes P \tag{5.57}$$

1.6.5.4 Distributivity:

$$(P\cup Q)\circ R = (P\circ R)\cup(Q\circ R) \tag{5.58}$$
(5.59)

1.6.5.5 DeMorgan's Law:

$$\overline{P\circ R} = \bar{P}\otimes\bar{R} \tag{5.60}$$
$$\overline{P\otimes R} = \bar{P}\circ\bar{R} \tag{5.61}$$

1.6.5.6 Inclusion:

$$\text{if } R_1\subset R_2 \;\Rightarrow\; P\circ R_1\subset P\circ R_2 \tag{5.62}$$

1.7 Case Studies

See Slides Pg 101-124

2 Fuzzy Inferencing and Fuzzy Control

2.1 Fuzzy Reasoning

2.1.1 Introduction

Remark 5.2.2: Fuzzy Reasoning

Fuzzy reasoning, also known as approximate reasoning (AR), is an inference procedure that derives conclusions from a set of if-then rules.

Definition 5.2.4: Composition Rule of Inference

- Let us assume that F is a fuzzy relation on $X \times Y$.
- Let A be a fuzzy set in X.
- To obtain the resulting fuzzy set B, we first construct a cylindrical extension of $A : C(A)$.
- The inference of $C(A)$ and F leads to the antecedent of the projection of $C(A) \cap F$.

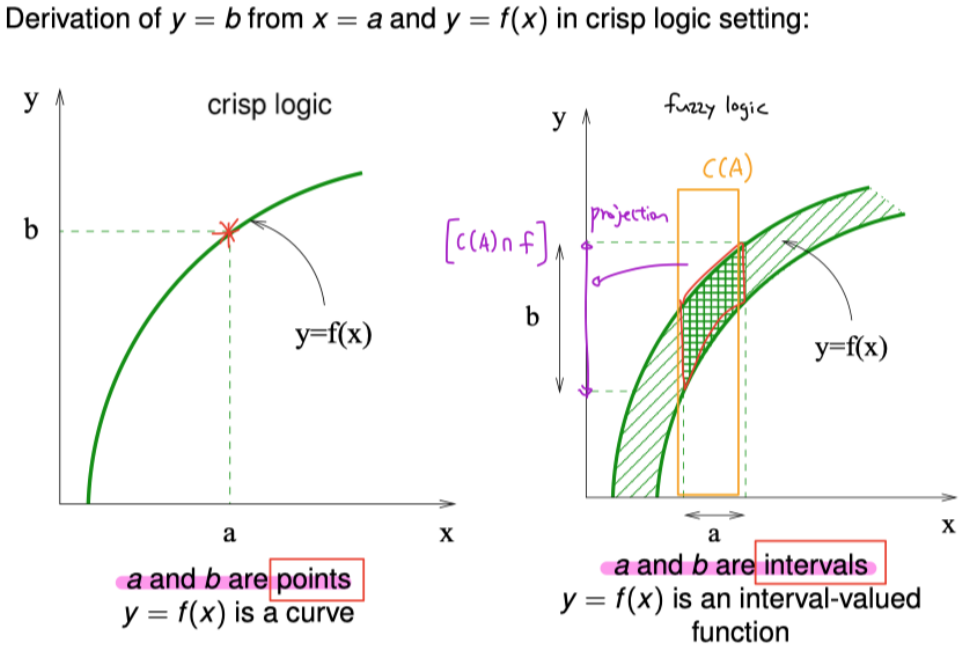


Figure 2-1. Crisp Logic vs. Fuzzy Logic

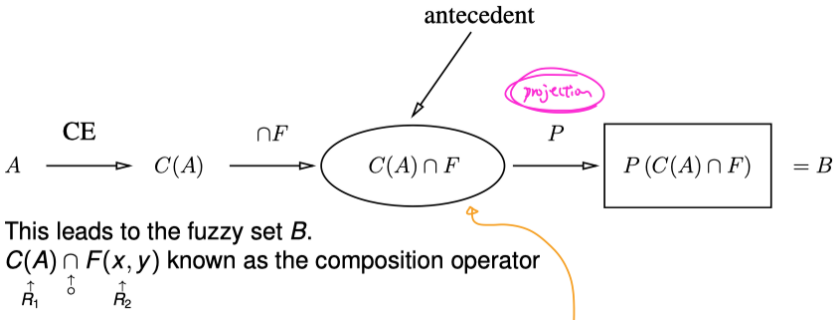


Figure 2-2. Fuzzy Reasoning Flow

Definition 5.2.5: Fuzzy Reasoning

uzzy reasoning is basically the extension of the well known composition of elements and functions:

$$\overbrace{\underbrace{\mu_C(A)}_{C(A)}}^{\text{cylinder}} \overbrace{\cap}^{\text{intersection (min)}} \overbrace{F(x,y)}^{\text{Fuzzy relationship of X and Y}} = \mu_C(A) \cap F(x,y) = \min\{\mu_{C(A)}(x), \mu_(x,y)\} \tag{5.63}$$

- Projection $\mu_C(A) \cap F(x,y)$ provides:

$$B \equiv \mu_B(y) \tag{5.64}$$
$$= \max_x \{\min[\mu_{C(A)}(x), \mu_F(x,y)]\} \tag{5.65}$$
$$= \bigvee_x [\mu_{C(A)}(x) \wedge \mu_F(x,y)] \tag{5.66}$$
$$\text{with } \vee\text{: max, and } \wedge\text{: min} \tag{5.67}$$

- This is basically max-min composition of two relations of A (unary relation) and F(binary relation)
- **Compositional Rule of Inference:** $B = A \circ F$

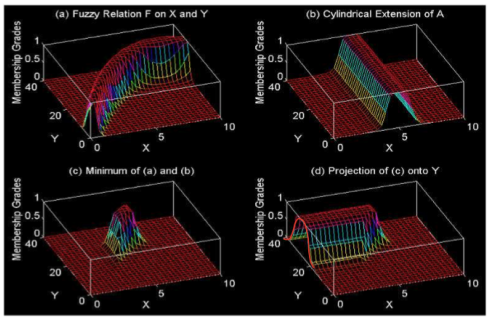


Figure 2-3. (Graphical Illustration) (a) is a fuzzy set and $y = f(x)$ is a fuzzy relation

2.1.2 Modus Ponens (MP)

- The rule of inference in conventional logic is modus ponens
- MP leads to inference of truth of a proposition B from the truth of A and the implementation $A \rightarrow B$.

2.1.2.1 Examples:

Example 5.2.1: MP

- Rule: If tomato is somehow red, then tomato is ripe $(A \rightarrow B)$
- Fact: Tomato is red (A)
- Conclusion: Tomato is ripe (B)

Remark 5.2.3: MP: Crisp Case

- Premise 1 : fact: x is A
- Premise 2 : rule: If x is A, then y is B
- Conclusion: consequence: y is B

Remark 5.2.4: MP: Fuzzy Reasoning

- Premise 1 (fact): x is A'
- Premise 2.1 (rule): If x is A, then y is B
- Conclusion (consequence): y is B'
- A' could be close to A
- B' could be close to B
- A, B, A', B' are fuzzy sets

2.1.2.2 Fuzzy Inferencing

- Let A, A', and B be fuzzy sets over X, X, and Y, respectively
- Assume that the fuzzy implication $A \rightarrow B$ be expressed as fuzzy relation $R_{X \times Y}$
- Then the fuzzy set B' included by (x is A') for the relation "x is A then y is B" is given by:

$$\mu_{B'}(y) = \max_x \{\min[\mu_{A'}(x), \mu_R(x,y)]\} \tag{5.68}$$

$$= \bigvee_x [\mu_{A'}(x) \wedge \mu_R(x,y)] \qquad [Equation\ (5.66)] \tag{5.69}$$

$$B' = A' \circ R = A' \circ (A \rightarrow B) \tag{5.70}$$

2.1.2.3 Case 1: Single Rule with Single Antecedent

$$\mu_{B'}(y) = \bigvee_x [\mu_{A'}(x) \wedge (\mu_A(x) \wedge \mu_B(y))] = \underbrace{\bigvee_x [\mu_{A'}(x) \wedge \mu_A(x)]}_{\text{degree of validity = w}} \wedge \mu_B(y) = \mathbf{w} \wedge \mu_B(y) \tag{5.71}$$

Recall Remark 5.2.4:

- Premise 1 (fact): x is A'
- Premise 2.1 (rule): If x is A, then y is B
- Conclusion (consequence): y is B'

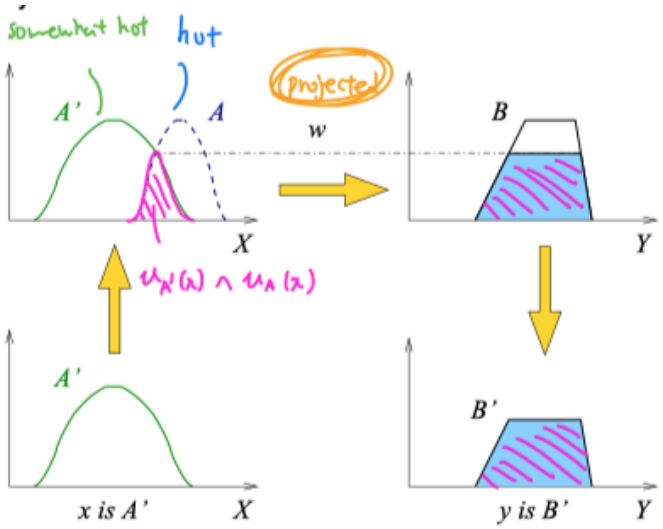


Figure 2-4. Case 1 Example

2.1.2.4 Case 2: Single Rule with Multiple Antecedent

- Premise 1 (fact): x is A' $\overbrace{\text{and}}^{\min}$ y is B'
- Premise 2.1 (rule): If x is A and y is B, then z is C
- Conclusion (consequence): z is C'

$$R: A \times B \to C \quad \Rightarrow \quad R: A \times B \times C \tag{5.72}$$

$$R = \int_{X \times Y \times Z} = \frac{\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)}{(x,y,z)} \tag{5.73}$$

$$\textbf{Result: } C': \overbrace{(A' \times B')}^{\text{data}} \circ \overbrace{(A \times B \to C)}^{\text{knowledge}} \tag{5.74}$$

$$\mu_{C'}(z) = \bigvee_{x,y} \underbrace{[\mu_{A'}(x) \wedge \mu_{B'}(y)]}_{(A' \times B')} \wedge \underbrace{[\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)]}_{A \times B \to C} \tag{5.75}$$

$$= \bigvee_{x,y} [\mu_{A'}(x) \wedge \mu_{B'}(y) \wedge \mu_A(x) \wedge \mu_B(y)] \wedge \mu_C(z) \tag{5.76}$$

$$= \underbrace{\bigvee_{x,y} [\mu_{A'}(x) \wedge \mu_A(x)]}_{\text{degree of validity of x } (w_1)} \wedge \underbrace{\bigvee_{x,y} [\mu_{B'}(y) \wedge \mu_B(y)]}_{\text{degree of validity of y } (w_2)} \wedge \mu_C(z) \tag{5.77}$$

$$= \underbrace{(\mathbf{w}_1 \wedge \mathbf{w}_2)}_{\text{firing strength}} \wedge \mu_C(z) \tag{5.78}$$

Equation 5.2.1: General Case: n antecedents

$$\mu_{C'}(z) = (\mathbf{w}_1 \wedge \mathbf{w}_2 \wedge \cdots \wedge \mathbf{w}_n) \wedge \mu_C(z) \tag{5.79}$$

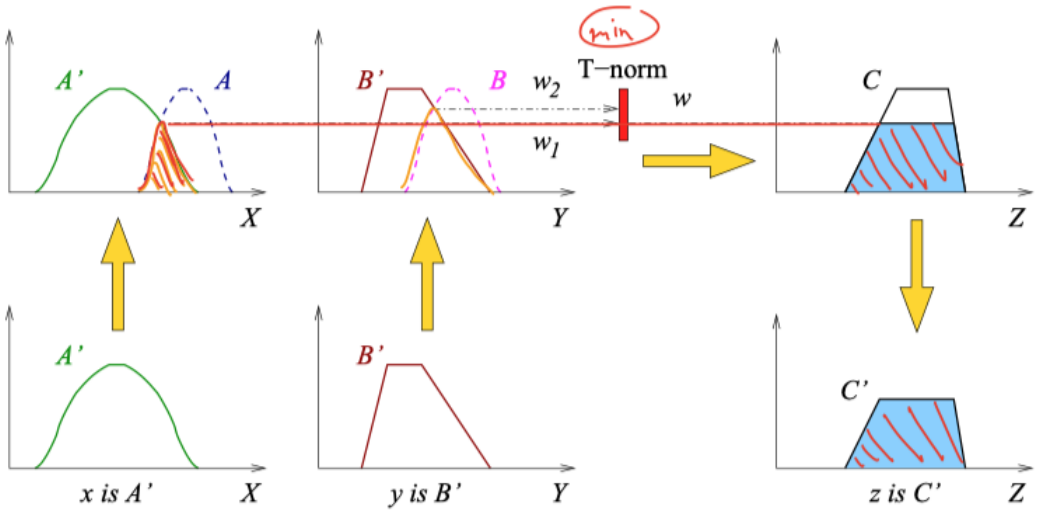


Figure 2-5. Case 2 Example

2.1.2.5 Case 3: Multiple Rule with Multiple Antecedent

- Premise 1 (fact): x is A' and y is B'
- Premise 2.1 (rule(1)): If x is A_1 and y is B_2, then z is C_1;
- Premise 2.2 (rule(2)): else if x is A_2 and y is B_2, then z is C_2
- Conclusion (consequence): z is C'

- Let $R_1 : A_1 \times B_1 \to C_2$ and $R_2 : A_2 \times B_2 \to C_2$
- Since the max-min operator is distributive over Union, we get:

$$C' = \underbrace{(A' \times B')}_{\text{antecedents}} \circ \underbrace{(R_1 \cup R_2)}_{\text{rules}} \tag{5.80}$$

$$= \underbrace{[(A' \times B') \circ R_1]}_{\substack{\text{rule 1 with} \\ \text{multiple antecedents}}} \cup \underbrace{[(A' \times B') \circ R_2]}_{\substack{\text{rule 2 with} \\ \text{multiple antecedents}}} \tag{5.81}$$

$$= C'_1 \cup C'_2, \text{ where } C'_i \text{ is the inferred consequent of rule } i \tag{5.82}$$

- As shown previously, for $C_i = (A' \times B') \circ R_i$,

$$\mu_{C'_i}(z) = \bigvee_x [\mu_{A'}(x) \wedge \mu_A(x)] \wedge \bigvee_y [\mu_{B'}(y) \wedge \mu_B(y)] \wedge \mu_C(z) \tag{5.83}$$

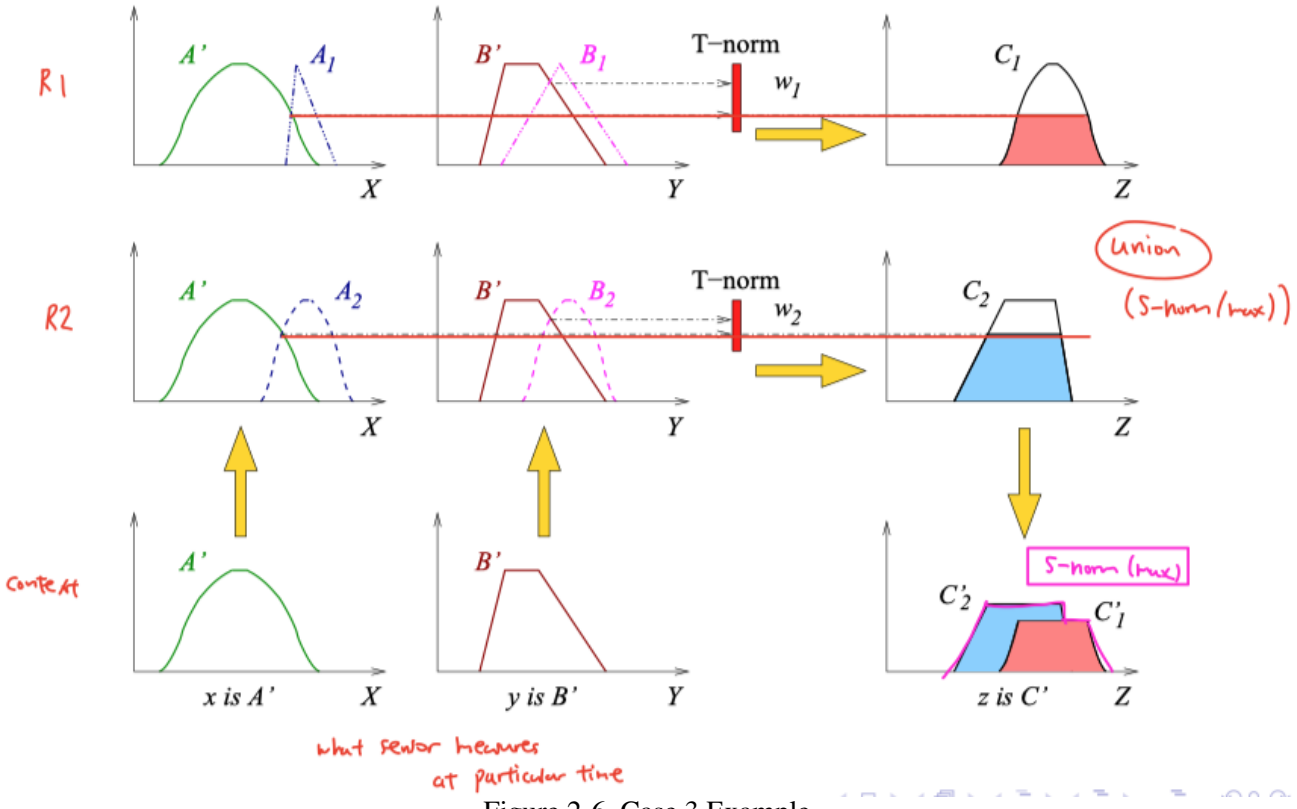


Figure 2-6. Case 3 Example

2.1.2.6 Typical Fuzzy System

Figure 2-7. Fuzzy Reasoning

2.1.2.7 Required Steps for Extended Modus Ponens (Fuzzy Reasoning)

1. Obtain degree of compatibility
   - Compare the known facts with the antecedents of the fuzzy rules → degree of compatibility
2. Find the firing strength which combines the degree of compatibility using fuzzy ”and” or ”or”
   - This indicates the degree at which the antecedent part of the rule is satisfied
3. Qualified consequent
   - Apply the firing strength to the consequent membership function of a rule to generate a qualified consequent membership function
4. Overall output MF aggregates all qualified consequent MF’s to obtain the overall MF

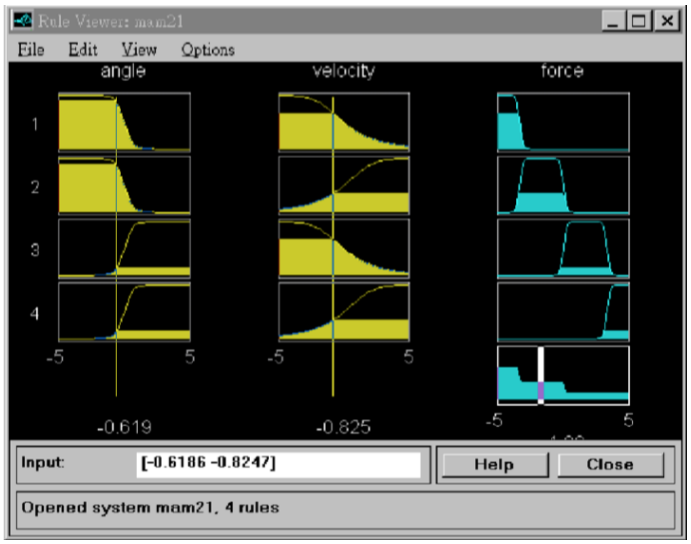


Figure 2-8. Fuzzy Reasoning - Graphical Example

2.2 Fuzzy Inference System (FIS)

2.2.1 Introduction

2.2.1.1 Basic Structure

The basic structure of any FIS is made of:

1. Rule Base
2. Database of Rules
3. Reasoning Mechanism (Fuzzification, Defuzzification)

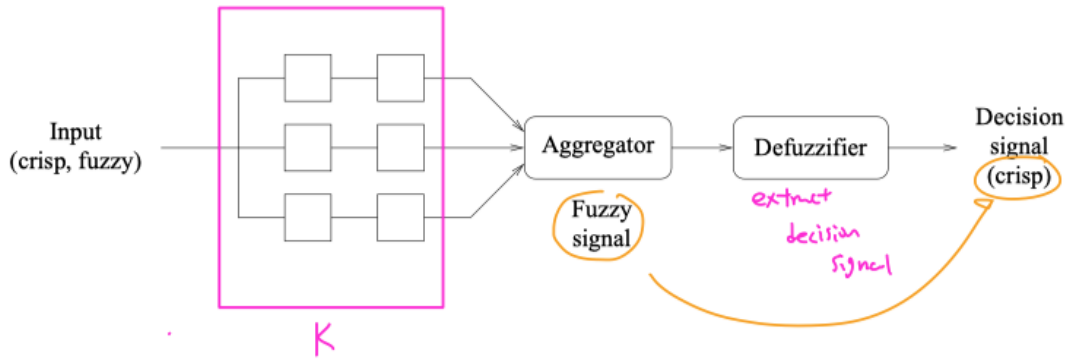


Figure 2-9. Basic Structure Diagram

2.2.2 Fuzzification

- Fuzzification refers to the representation of a crisp value by a membership function
- It is needed prior to applying the composition (CRI), when the data (measurements) are crisp values, aas common in control applications
- It may be argued that the process of fuzzification amounts to giving up the accuracy of crisp data
- This is not so in general. The reason is, a measured piece of data may not be known to be 100% accurate

2.2.2.1 Discrete Case of Fuzzification

- In the case of discrete membership functions, the crisp quantity y_0 may not correspond to one of the discrete points of the membership function of the fuzzy variable Y (or fuzzy state Y_j).
- Suppose that the crisp data value y_0 falls between the discrete values y_{j-1} and y_j of the membership function.
- Assign a membership grade of 1 for y_0, the membership grades of the fuzzified quantity F at y_{j-1} and y_j are determined through linear interpolation as $\frac{y_{j-1}-y_{j-2}}{y_0-y_{j-2}}$, $\frac{y_{j+1}-y_j}{y_{j+1}-y_0}$, respectively.
- Accordingly, the discrete membership function of the fuzzified quantity F is given by:

$$F = \left\{ \frac{\frac{y_{j-1}-y_{j-2}}{y_0-y_{j-2}}}{y_{j-1}}, \frac{\frac{y_{j+1}-y_j}{y_{j+1}-y_0}}{y_j} \right\} \tag{5.84}$$

- The approach can be extended to include more than two discrete points, thereby providing a wider membership function (greater fuzziness)

2.2.2.2 Continuous: Singleton Method

- Consider a crisp measurement y_0 of a fuzzy variable Y
- It is known that the measurement y_0 is perfectly accurate

- y_0 may be represented by a fuzzy quantity F with the singleton membership fucntion

$$\mu_F(y) = \delta(y - y_0) = \begin{cases} 1 & \text{, if } y = y_0 \\ 0 & \text{, otherwise} \end{cases} \tag{5.85}$$

- Since the measured data are not perfectly accurate, a more appropriate method of using fuzzy singleton to fuzzify a crisp value is given now:
- Suppose that a crisp measurement y_0 is made of a fuzzy variable Y. Let Y can take n fuzzy states $Y_1, Y_2, \ldots, Y_n$
- Since $Y = Y_1 \vee Y_2 \vee \cdots \vee Y_n$, the membership function of Y is given as the union of the membership functions of the individual fuzzy states:

$$\mu_Y(y) = \max_{j=1}^{n} \mu_{Y_j}(y) \tag{5.86}$$

- The membership function of the fuzzified quantity F is given according to the extended singleton method by a set of fuzzy singletons. For state j:

$$\mu_F(y) = \mu_{Y_j}(y)\delta(y - y_0) \tag{5.87}$$

2.2.2.3 Continuous: Triangular Function Method

- A triangular membership function may be used to represent the fuzzified quantity for each fuzzy state.
- A triangular membership function (continuous case) may be expressed as

$$\mu_A(y) = \begin{cases} 1 - \frac{|y - y_0|}{s} & \text{, if } |y - y_0| \le s \\ 0 & \text{, otherwise} \end{cases} \tag{5.88}$$

  - where y_0 shows the peak point, and s denotes the base length (support set)
- Again $Y = Y_1 \vee Y_2 \vee \cdots \vee Y_n$, the membership function of Y is such that, for state j:

$$\mu_F(y) = \mu_{Y_j}(y_0)\, \mu_{Y_{A_j}}(y) \tag{5.89}$$

  where

$$\mu_{A_j}(y) = \begin{cases} 1 - \frac{|y - y_0|}{s_j} & \text{, if } |y - y_0| \le s_j \\ 0 & \text{, otherwise} \end{cases} \tag{5.90}$$

  with s_j: base length for state j

2.2.2.4 Continuous: Gaussian Function Method

- A Gaussian membership function may be used to represent the fuzzified quantity for each fuzzy state.
- A Gaussian membership function (continuous case) may be expressed as

$$\mu_A(y) = \exp\left(\frac{y - y_0}{s}\right)^2 \tag{5.91}$$

  - where the smaller the $s \Rightarrow$ the sharper (or less fuzzy) the membership function.
- Again $Y = Y_1 \vee Y_2 \vee \cdots \vee Y_n$, the membership function of Y is such that, for state j:

$$\mu_F(y) = \mu_{Y_j}(y_0)\, \mu_{Y_{A_j}}(y) \tag{5.92}$$

  where

$$\mu_A(y) = \exp\left(\frac{y - y_0}{s}\right)^2 \tag{5.93}$$

with s_j: base length for state j

2.2.3 Defuzification

- Usually, the decision (control action) of a fuzzy logic controller is a fuzzy value and is represented by a membership function.
- Because low-level control actions are typically crisp, the control inference must be defuzzified for physical purposes such as actuation.
- Methods of defuzzification:
  - Centroid method
  - Mean of maxima method
  - Threshold methods

2.2.3.1 Centroid Method (Center of Gravity)

- Suppose that the membership function of a control inference is $\mu_C(c)$, and its support set is given by: $S = c|_{\mu_C(c)} > 0$
- The centroid method of defuzzification is expressed as:

Equation 5.2.2: Continuous Case

$$\hat{c} = \frac{\int_{c \in S} c\mu_C(c)dc}{\int_{c \in S} \mu_C(c)dc} \tag{5.94}$$

Equation 5.2.3: Discrete Case

$$\hat{c} = \frac{\sum_{c_i \in S} c_i \mu_C(c_i)}{\sum_{c_i \in S} \mu_C(c_i)} \tag{5.95}$$

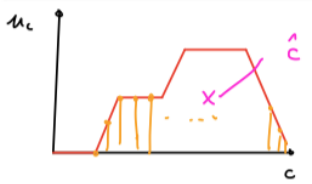


Figure 2-10. Centroid Method Illustration

2.2.3.2 Mean of Maxima Method

- If the membership function of the control inference is unimodal, the control value at the peak membership grade is chosen as the defuzzified single control action.

$$\hat{c} = c_{\max} \ \textbf{s.t.} \ \mu_C(c_{\max}) = \mu_C(c) \tag{5.96}$$

- If the control membership function is multi-modal, the mean of the control values at these peak points, weighted by the corresponding membership grades, is used as the defuzzified value.

$$c_i \ \textbf{s.t.} \ \mu_C(c_i) = \mu_i = \max_{c \in S} \mu_C(c)(c), \ \forall i = 1, 2, \ldots, P \tag{5.97}$$

Then:

$$\hat{c} = \frac{\sum_{i=1}^{P} \mu_i c_i^{max}}{\sum_{i=1}^{P} \mu_i} \tag{5.98}$$

where p: total number of modes (peaks)

2.2.3.3 Threshold Method

- Sometimes, it may be desirable to leave out the boundaries of the control inference membership function.
- Only the main core of the control inference is used, not excessively diluting or desensitizing the defuzzified value.
- The corresponding procedures of defuzzification are known as threshold methods.
- The formulae remain the same as given before. However, we use an α-cut of the control inference set not the entire support set.

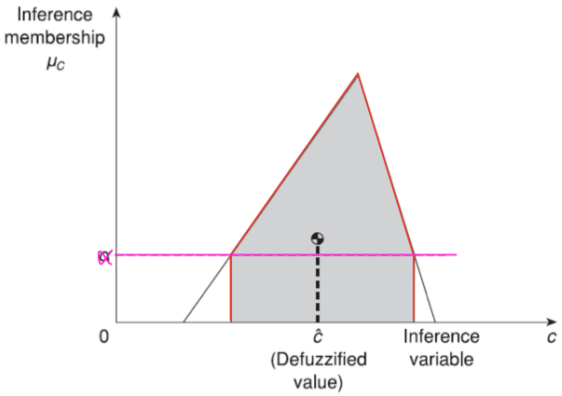


Figure 2-11. Threshold Method Illustration

2.2.3.4 Other Methods

Bisector of Area $\Rightarrow C_{BOA}$ **s.t.** $\int_{\alpha}^{C_{BOOA}} \mu_C(c)dc = \int_{C_{BOA}}^{\beta} \mu_C(c)dc$,with$\alpha = \min\{c\}$ and $\beta = \max\{c\}$ (5.99)

(SoM) Smallest of Maximum $\Rightarrow$ min of max μ_C (5.100)

(LoM) Largest of Maximum $\Rightarrow$ max of max μ_C (5.101)

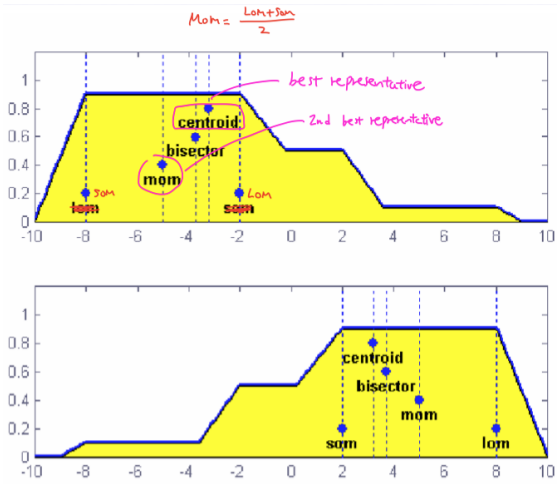


Figure 2-12. Defuzzification Methods Illustration

2.2.4 Different Inferencing Systems

- Different inferencing procedures have been used in the literature.
- The main difference among them is the aggregation and the defuzzification.
  - Mamdani Fuzzy Model
  - Sugeno Fuzzy Model
  - Tsukomoto Fuzzy Model

2.2.4.1 Mamdani Fuzzy Model

(max-min operator) for aggregation part.

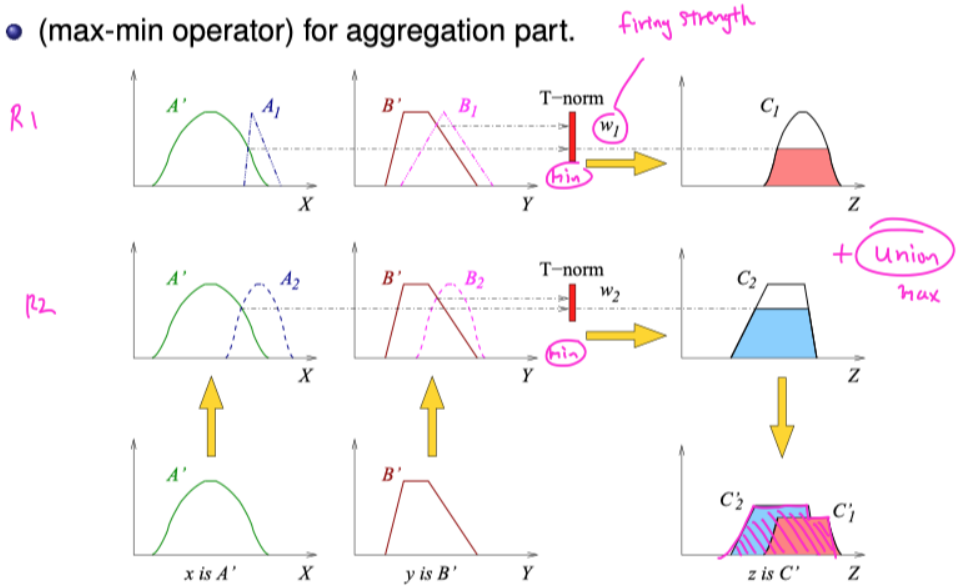


Figure 2-13. Mamdani Methods Illustration

see examples in slide 41-43

2.2.4.2 Sugeno Fuzzy Model

- The consequent in the Sungeno fuzzy model is a function of the antecedent
  - If x is A_1 and y is B_1 then $z = f_1(x,y)$
  - If x is A_2 and y is B_2 then $z = f_2(x,y)$
  - Where $f_i, i = 1,2,...,$ is a crisp polynomial function in its argument
- The overall output (aggregation) is obtained through a weighted average.

$$z = \frac{\sum_{i=1}^{\#\text{ of rules}} w_i z_i}{\sum_{i=1}^{\#\text{ of rules}} w_i} \tag{5.102}$$

- The consequent in the Sugeno fuzzy model is a function of the antecedent

Sugeno Zero Order Model: $f_i(x,y) = \text{const.}$ (5.103)

Sugeno First Order Model: $f_i(x,y) = a_i x + b_i y + c_i$ (5.104)

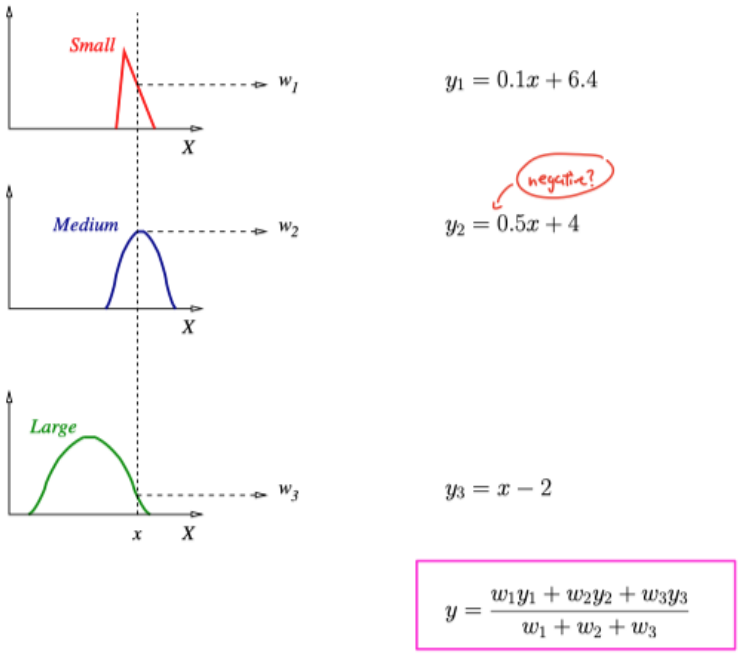


Figure 2-14. Sugeno Methods Illustration

2.2.4.3 Tsukomoto Fuzzy Model

- The output membership function is a monotonic mapping (f^{-1} exists)

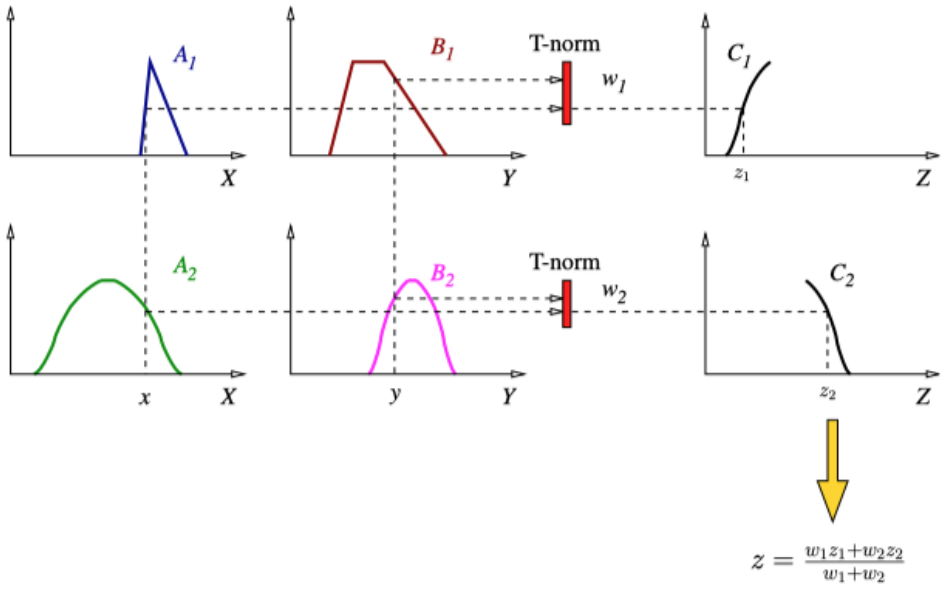


Figure 2-15. Tsukomoto Methods Illustration

2.2.5 Case Studies:

2.2.5.1 Case Studies

See Slides Pg

Glossary

AI Artificial Intelligence Definition 2.1.1.
ANN Artificial Neural Network.

BPL Back Propagation Learning.

ML Machine Learning Definition 2.1.2.
MLP Multi-Layer Perceptron.

PR Patern Recognition Definition 2.1.4.

SVM Support Vector Machines.