CHAPTER -1 Prof. Ranjitha U.N

1

→ Machine learning ⟹ Succefull understanding of how to make computers (machines) learn.

→ Machine learning opens up new levels of competence & customization.

Ex * Machines learing from medical records for most effective treatments.

* House learning to optimize energy usage

* Speech recognition where ML algorithms outperform other approaches.

* In the datamining applications, ML algorithms are used to discover valuable knowledge from large database.

→ Specific achievements being recognition of spoken words detect fraudulent use of credit cards, drive autonomous vehicles etc.

→ Machine learning involves training with the examples observed, hypotheses under consideration and of expected error in learned hypotheses.

Well-posed learning problems

A computer is said to learn from experience E task w.r.t to task T and performance measured by P

The performance P with tasks T increases with experience E.

Example 1 : A computer program made to learn checkers game.

E = playing practice games

T = playing checkers game

P = ability to win against the oponent

2: Filtering of spam in email

Marking the email as spam, machine learns to categorize the spam emails.

T = Classifying emails as spam or not spam

E = watching you label emails as spam or not

P = The no. of emails correctly classified as spam or not spam.

3. Learning to classify new astronomical structures

Sky Image cataloging and Analysis Tool (SKICAT), the purpose of SKICAT is to enable & maximize the extraction of meaningful information from a large DB (TB of data)

Image → Removal of noise → segmentation →

Feature extraction → Classification → manual analysis

Classification method : decision tree classifier

4. Learning to drive autonomous vehicles

2016 – Tesla's model crashed with truck

2018 – Uber's self driving hit a pedestrain (Arizona)

(who jumped of the bushes)

Deep learning algorithms found in self driving cars

use numerous examples to learn the rules of their domain. As they spend time on the road, they classify the info they gather & learn to handle different situations. ⇒ Technology fails during its evolution.

Task T : Driving on lane using vision sensors.

Performance P : Distance travelled before an error

E : a sequence of images and steering commands recorded while observing a human driver.

Successful applications of machine learning:

Artificial Intelligence

Bayesian methods → $P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$ ⇒ statistical inference is based on this

computational complexity theory → classifies computational problems based on inherent difficulty

Control theory → dynamic systems with feedback influence

Information theory → coding of information

Philosophy

Psychology & neurobiology

Statistics.

Learning ⇒ training

⇒ define precisely a class of problems that encompasses interesting forms of learning, explore algorithm that solve such problem & to understand the fundamental structure of learning problem & processes.

Designing a learning system

Illustration is done using program to learn to play chess

The different steps involved in designing a learning system are
1. choosing the training experience
2. choosing the target function
3. choosing a representation for the target function
4. choosing a function approximation algorithm

1. choosing the training experience:-

Type of training
- degree to which learner controls the sequence of training ex/
- representation of distribution of examples

Training experience E : playing practice games against itself

Type of training

→ Training experience has significant impact on success or failure of the learner.

→ Training experience provides direct or indirect feedback

→ For the example considered, direct ~~feedback~~ examples consists of individual checkers board states and the correct move for each. (direct feedback is observable & measurable)

→ Indirect info consists of the move sequences & the final outcome of various games played. The correctness of the move sequences must be inferred indirectly from the fact that the game was eventually won or lost.

Learning from direct training feedback is typically easier than learning from indirect feedback.

Degree to which the learner controls the sequence of training examples.

→ The learner might rely on the teacher to select informative board states and to provide correct move for each. or the learner may have complete control over both board states & moves (ie when it learns by playing against itself.

→ The control influences in choosing the optimal training examples. The settings can be varied for learning which includes training from a random process, queries to an expert teacher, learner autonomously exploring its environment etc.

Optimal representation of the distribution of the examples

→ The training experience E must be fully representative of the distribution of situations over which it will later be tested. (Ex expecting a move played by a champion)

→ Performance P will be appreciative with the mastery over wide range of distribution of examples.

Note: Assumption - system will train by playing games against itself ⇒ generation of as much training data as time permits.

2. Choosing the target function

→ It deals with exactly what type of knowledge will be learned.

→ For example, if a picture of a digit is fed as an input to a function, digit value is given out as output ie target function maps data to its target value.

→ In the checkers game, among all the legal moves, the type of info to be learned [target function] is to choose the best move for any given board state. This function = choose move

ChooseMove : B → M [Program accepts I/P from the set of legal moves B & produces O/P being a legal move in set M]

→ Choose move will be a very difficult state to learn from as it is an indirect training experience

→ As an alternative to above mentioned difficulty an evaluation function is designed that assigns a numerical score to any given board state. This function is V

V : B → R (R = set of real no)

The target function V can be used to choose the best successor state & therefore the best legal move

The target value V(b) for an arbitrary board state b in B, is as follows

1. If b is a final board state that is won then V(b) = 100
2. If b is a final board state that is lost then V(b) = -100
3. If b is a final board state that is drawn, then V(b) = 0
4. If b is not a final state in the game, then V(b) = V(b'). b' is the best final board state that can be achieved starting from b & playing optimally.

→ For case 4., optimal line of play has to be searched all the way to the end of the game which results in nonoperational definition

→ Learning operational description may be difficult so, some approximation function for the purpose of learning is designed which is called function approximation $\hat{V}$. [ideal function = V]

3. Choosing a Representation for the target function

Here we need to choose a representation that the learning program will use to describe the function $\hat{V}$.

→ * $\hat{V}$ can be represented using collection of rules
* quadratic polynomial function of predefined board features or
* an artificial neural network.

→ Expressive representation allows close approximation as possible to ideal target function V. But for more expressive representation, more training data, the program will require.

→ Here, $\hat{V}$ is calculated as a linear combination of the following board features

x_1: no. of black pieces on the board
x_2: " " red " " "
x_3 no. of black kings on the board
x_4: no. of red kings on the board
x_5: no. of black pieces threatened by red
x_6: no. of red pieces threatened by black

$$\hat{V}(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_6 + w_6x_6$$

$w_0 - w_6$ = numerical co-ff or weights to be chosen by learning algorithm.

4. Choosing a function approximation algorithm

→ In order to train the target function $\hat{V}$, a set of training examples are required ie $V_{train}(b)$ (b = board state)

→ Each training example is an ordered pair of the form
$\langle b, V_{train}(b) \rangle$

→ For example consider a training example which describes a board state b in which black has won the game

ie ⇒ $V_{train}(b) = +100$

$\langle\langle x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0 \rangle, +100 \rangle$

Assigning training values

→ the procedure involves i) estimating training values and
ii) Adjusting the weights.

Estimating training values

→ It is easy to assign a value to board states that correspond to the end of the game but difficult to assign values to more numerous intermediate board state.

→ So assign value to intermediate board state, successor board state successor(b) following current board state b (ie the board state following the program's move and the opponent's response)

$V_{train}(b) \leftarrow \hat{V}(successor(b))$

successor(b) calculated by $\hat{V}$
next move would be calculated by the move which maximizes V
opponents move = move which minimizes V

Adjusting the weights

→ TO suit the training examples, appropriate weights ha to be chosen which is done by 'learning algorithms'

→ The weights are chosen in such a way that the error 'E' between the training values and the values predicted by the hypothesis $\hat{V}$ is minimized.

→ In the example considered, an algorithm that will incrementally refine the weights as new training examples become available is considered. One such algorithm is the LMS training rule.

LMS weight update rule

For each training ex $\langle b, V_{train}(b) \rangle$

Use the current weights to calculate $\hat{V}(b)$

For each weight w_i update it as

→ $w_i \leftarrow w_i + \eta \left(V_{train}(b) - \hat{V}(b)\right) x_i$

$\eta \rightarrow$ small constant (0.1)

when $\left[V_{train}(b) - \hat{V}(b)\right] = 0 \Rightarrow$ no wts are changed

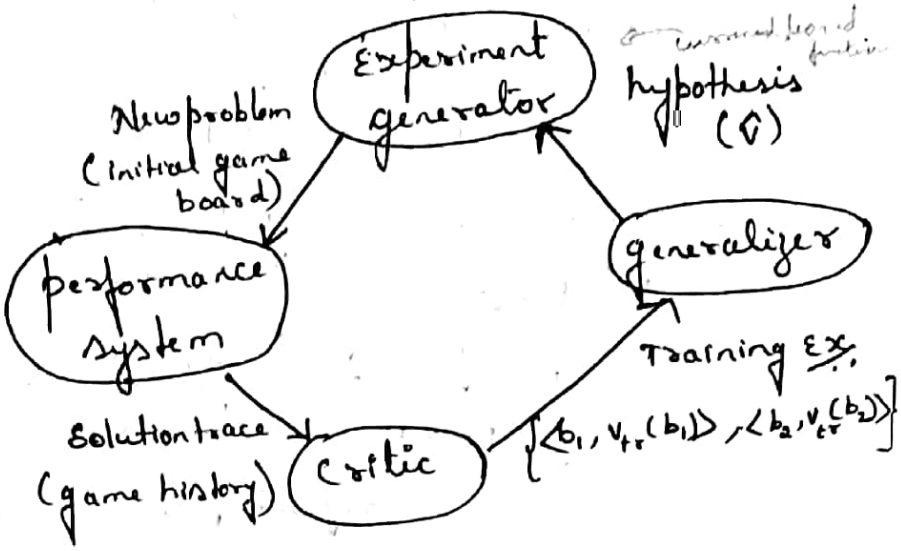
when $\left[V_{train}(b) - \hat{V}(b)\right] = +ve \Rightarrow \hat{V}(b)$ is low

then each weight is increased in proportion to the value of its corresponding feature

Note: If some feature x_i is zero then its weight is not altered regardless of error.

The final design

The checkers learning system can be described by four distinct program modules viz performance system, critic, generalizer and experiment generator.



Performance system:-
It is the module that must solve the given performance task by using the learned target function [i.e $\hat{V}$]
It takes new problem as input and produce a trace of its solution as output

Critic :- i/p = history or trace of the game
o/p = training examples of the target function
i.e $V_{train}(b) \leftarrow \hat{V}(successor(b))$

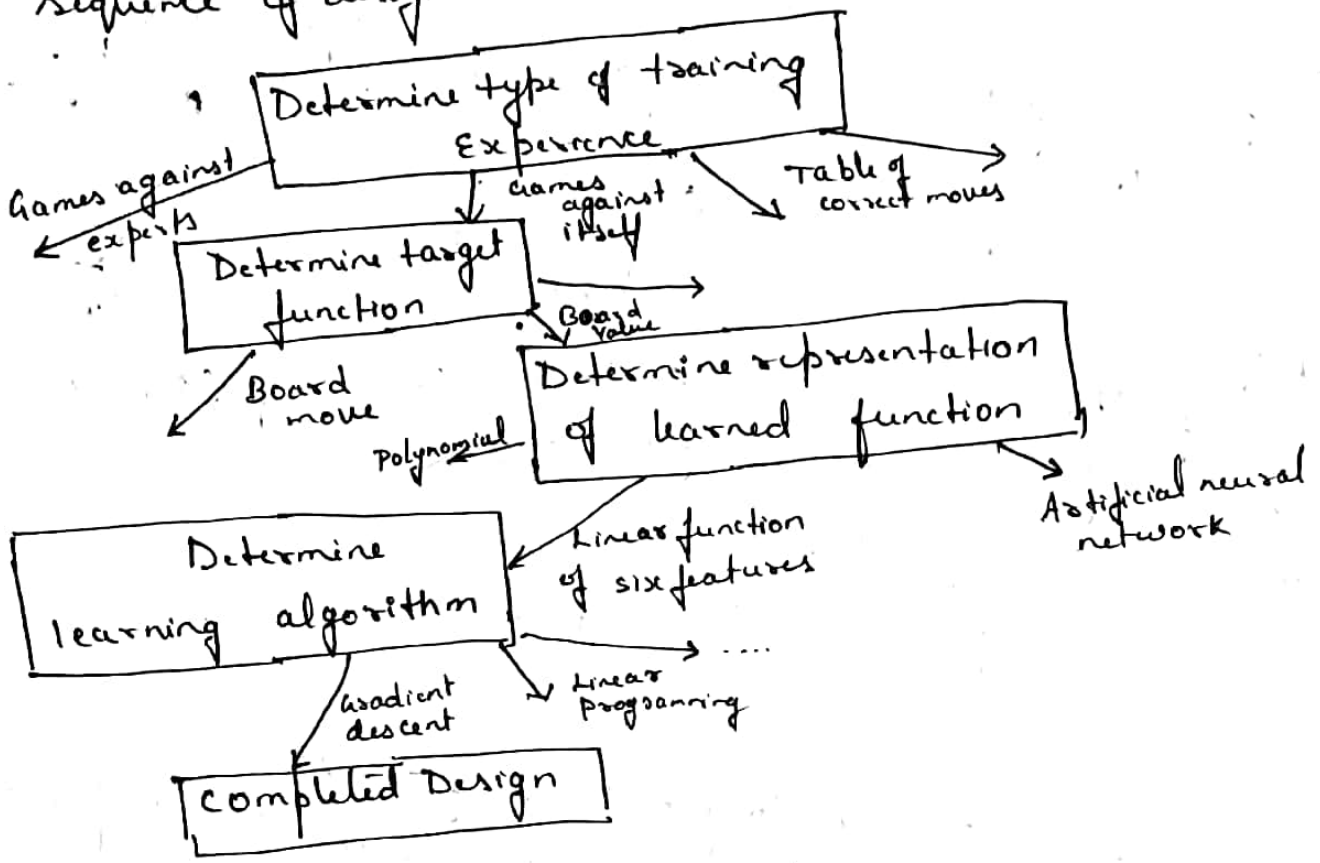
Generalizer:- i/p = training examples
o/p = hypothesis of the estimate of the target fn.

Ex: LMS algorithm → o/p hypothesis is $\hat{V}$ defined by weights $w_0 \dots w_6$.

Experiment generator:- i/p = currently learned function (current hypothesis)

o/p = initial board state (new problem)

the sequence of design choice



Perspective and issues in machine learning

Useful perspective → large space [observed data & apriori knowledge by learner] of possible hypotheses is searched through to get the best fit case.

Ex: In the checkers game, the large space considered for hypothesis consists of all evaluation functions that can be represented by some choice of values for the weights w_0 to w_6. Note:- Learner's task is to search through the vast space to locate hypothesis.

LMS algorithm tunes the weights, adding correction to each weight each time the hypothesized evaluation function predicts a value that differs from the training value.

Note Algorithms that search hypothesis space could be linear functions, decision trees, ANN etc).

Issues in Machine learning

→ what algorithms exist for learning general target function

↔ what would be settings for the algorithm to yield best coverage. which algorithm perform best for which type of problems.

→ How much training data is sufficient. Confidence in learned hypothesis evaluation.

→ priori knowledge's usefulness when it is approximately correct.

→ strategy to choose a useful next training experience

→ what specific functions should the system attempt to learn.

→ How to automatically alter its representation to improve its ability.

Examples of machine learning applications

There are different types of learning. Machine learn caters to different types of applications based on the learning method.

1. Learning Associations

Learns a conditional probability of the form $P(Y|X)$, where Y is the product conditioned on X.

Ex In supermarket chain, one application is basket analysis [associating products bought by customers. If people who buy X typically also buy Y, if he does not, he/she is a potential Y customer.

$P(\text{chips}|\text{beer}) = 0.7 \Rightarrow$ 70% customers who buy beer also buy chips. → rule

→ $P(Y|X, D)$ where D is the set of customer attributes such as gender, age, marital status so on.

→ In case of webportal based applications, a link user is likely to click can be downloaded in advance for faster access.

→ web search

Ex. of algorithms → Apriori, Eclat, FP growth

These mining frequent itemsets

→ supervised (if modified & the task is to learn the mapping from i/p to o/p)

Classification :- Deals with identifying to which set of categories (sub-population), the new observation belongs to.

→ For classification problems, the training data is used and based on this a rule is formed. The main application of the above is prediction.

→ Ex :- A bank setting a rule to issue loan to customer
Bank calculates the risk using the info about the customer
Info, may be age, income, collaterals etc.
Based on these data, the rule set would be

[IF income $> \theta_1$ AND savings $> \theta_2$ THEN low-risk ELSE high-risk]

learning association

$P(Y|X)^{\text{attrib}} = 0$ for low-risk & 1 for high risk

$P(Y=1|X=x) = 0.8 \Rightarrow$ 80% probability of being at high risk.

→ the data set dealt in classification may be bi-class (male/female, spam/non-spam) or may be multiclass (blood group identification)

→ some examples of classification problems are speech recognition, pattern recognition (hand writting) document classification etc, face recognition, medical diagnosis, outlier detection (finding instances that do not obey the general rule → fraudulent)

Types of classification algorithm

1. Linear classifier : Naive Bayes classifier
2. SVM → supervised → used even for regression → soft marginal, maximum marginal, kernel based etc
3. Decision tree → supervised → Recursively devides the working area (plot) into subparts until devided into classes that are pure
4. Boosted tree
5. Random forest → works with decision trees, difference b/w decision tree & this is it does splitting feature node randomly; eg finding a place to visit from friends advice & suggestion
6. Neural Network → Mathematical model, RNN, Radial basis function NN, feed forward NN.
7. Nearest Neighbor → An object is classified by a majority vote of its neighbors.

Regression → supervised

→ Regression problems have output as a number

→ Consider an example of a system that can predict the price of a used car.

Inputs are the car attributes (brand, year, mileage etc)

If x denote the car attributes

$y \longrightarrow$ price of the car

$y = wx + w_0$

→ Regression is a supervised learning, where

$y = g(x|\theta)$, g is the model

$\theta \rightarrow$ parameters

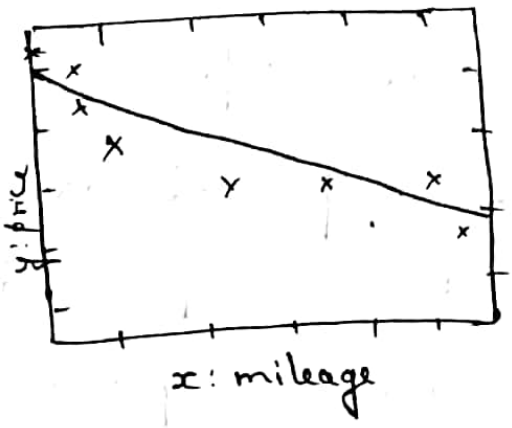
y = number, if its regression problem.

y = class code, if its classification.

Note: Classification is about predicting a label & Regression is about predicting a quantity.

Generally machine learning optimizes the parameters, θ, such that the approximation error is minimized..

→ In the fig, the model is linear with eqn: $y = wx + w_0$.



It can also be quadratic

$$y = w_2x^2 + w_1x + w_0$$

→ Autonomous car, navigation of a mobile robot. where o/p y = angle by which the steering wheel should be turned at each time.

→ Coffee roaster

Inputs → bean type, time, temperature etc

o/p → customer satisfaction in bean that is roasted.

optimal roasting can be found by having various check points which is called response surface design.

→ Recommendation system for movies, real estate value, drug response modelling.

→ Examples of regression algorithms linear regression, regression trees, lasso regression & multivariate regression.

Note:	supervised	unsupervised
Discrete	classification	clustering
continuous	regression	Dimensionality reduction

Unsupervised Learning

→ In machine learning the three major types of learni. are supervised, unsupervised and semisupervised..

→ supervised learning aim is to map from input to an output whose correct values are provided by a supervisor [the task involves learning the mapping from the input to the output].

→ Unsupervised learning has (no supervisor) only i/p data and the aim is to find regularities in the input. It deals with the task of inferring a function that describes the structure of unlabelled data (not classified or categorized)

→ One of the application of unsupervised learning is density estimation. One method for density estimation is clustering.

→ In a company with varied range of customers, customers with similar attributes ~~of the customers~~ are grouped together ie to the same group ⇒ customer segmentation along with customers, their products & services are also considered which leads to CRM system.

Application of Clustering is image compression also A clustering algorithm (program) groups pixels with similar colours in the same group. Document clustering also is an example also Bioinformatics

Reinforcement learning

→ some systems require sequence of correct actions to reach the goal to generate a policy. Such learning methods are called reinforcement learning algorithms. ex game playing, Robot navigation

→ Reinforcement learning is difficult when the system has unreliable and partial sensory info.

Supervised learning

Concept learning and the general-to-specific Ordering

Concept learning can be formulated as a problem of searching through a predefined space of potential hypotheses to find the hypothesis that best fits the training examples.

In precise it is about inferring a boolean-valued function from training examples of its input and output.

Ex: function defined over all animals, true for birds & false for other animals.

A concept learning task

* target concept → Days on which Aldo enjoys his favorite water sport.

  Each hypothesis → vector of six constraints

* Note: ? → any value is acceptable, a single value, ∅ → no value is acceptable. There are the possible hypothesis for each attribute.

* If some instance x satisfies all the constraints of hypothesis h, then x is a positive example h(x) = 1.

  If ~~hypothesis~~ attributes are <?, cold, High, ?, ?, ?> then hypothesis is Aldo enjoys his favorite sport only on cold days with high humidity.

Ex	Sky	Airtemp	humidity	wind	water	Forecast	Enjoy sport
1.	Sunny	warm	Normal	strong	warm	same	yes
2.	Rainy	cold	High	strong	warm	change	NO

Ex 1 → positive case , Ex 2 → Negative case.

* Most general hypothesis → Everyday is a positive example
(?, ?, ?, ?, ?, ?)

Most specific hypothesis → No day is positive example
(ϕ, ϕ, ϕ, ϕ, ϕ, ϕ)

* The concept or function to be learned is called the target concept c.
The set of items over which the concept is defined is denoted by X.
$c : X \rightarrow \{0, 1\}$; c(x) = 1 : if enjoy sport = yes.

* Training examples are indicated by the ordered pair $\langle x, c(x) \rangle$, denoted by D

* H → indicates hypothesis class in which hypothesis h is found.
The goal of the learner (learning algorithm) is to find a hypothesis h such that h(x) = c(x) for all x in X

* Note: In the enjoy sport learning task, attrib sky has three possible values and all other 5 attributes have 2 possible values
∴ the instance space X contains $3 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 96$ distinct instances.

X = cross product of all feature values

* Note: Most practical learning tasks involve much larger, sometimes infinite hypothesis spaces. the A good learning algorithm is capable of searching very large hypothesis spaces to find the hypothesis that best fit the training data.

General-to-specific Ordering of hypotheses

Objective: Avoid explicit enumeration of every hypothesis [By making use of naturally occurring structure over the hypothesis space].

Consider a hypothesis

$h_1 = \langle$ sunny, ?, ?, strong, ?, ? $\rangle$

$h_2 = \langle$ Sunny, ?, ?, ?, ?, ? $\rangle$

h_2 is more general hypotheses than h_1.

for x being an instance in X &
h being an hypotheses in H
x satisfies h iff $\underline{h(x) = 1}$

Generally, given h_j and h_k being two hypotheses, h_j is more general than or equal to h_k iff any instance that satisfies h_k also satisfies h_j.

$$\forall x \in X \; [h_k(x) = 1 \rightarrow (h_j(x) = 1)]$$

$$h_j \geq_g h_k \text{ iff } (h_j \geq_g h_k) \wedge (h_k \not\geq_g h_j)$$

Find-S: ~~Finding a maximally specific hypothesis~~

* consider three hypothesis h_1, h_2 & h_3

$h_1 = \langle$ sunny, ?, ?, strong, ?, ? $\rangle$

$h_2 = \langle$ sunny, ?, ?, ?, ?, ? $\rangle$

$h_3 = \langle$ sunny, ?, ?, ?, cool, ? $\rangle$

$h_2 >_g h_1$ also $h_2 >_g h_3$

Find-S: Finding a maximally specific hypothesis

Objective: To use a more_general_than ($>_g$) partial ordering to organise the search for a hypothesis consistent with observed training examples.

Note: D: Set of training example is $\{x, c(x)\}$

* For the training example below, we find the s.

1. sunny, warm, Nor, str, warm, same, Yes
2. sunny, warm, high, str, warm, same, Yes
3. Rainy cold, high, str, warm, change, NO
4. sunny warm, high, str, cool, change, Yes

step 1: Initialize - h to most specific hypothesis in H

$h \leftarrow \langle \phi, \phi, \phi, \phi, \phi, \phi \rangle$

Step 2: Compare the attributes of ~~the~~ h with each example, replace h attributes with the attributes of the example if the target concept (true case) is not met.

→ In Ex 1 ∅ does not satisfy Ex 1 attrib

so $h \leftarrow (\phi, \phi, \phi, \phi, \phi, \phi)$ becomes

$h \leftarrow$ (sunny, warm, Normal, strong, warm, same)

→ Consider Ex 2 from H to further generalize h

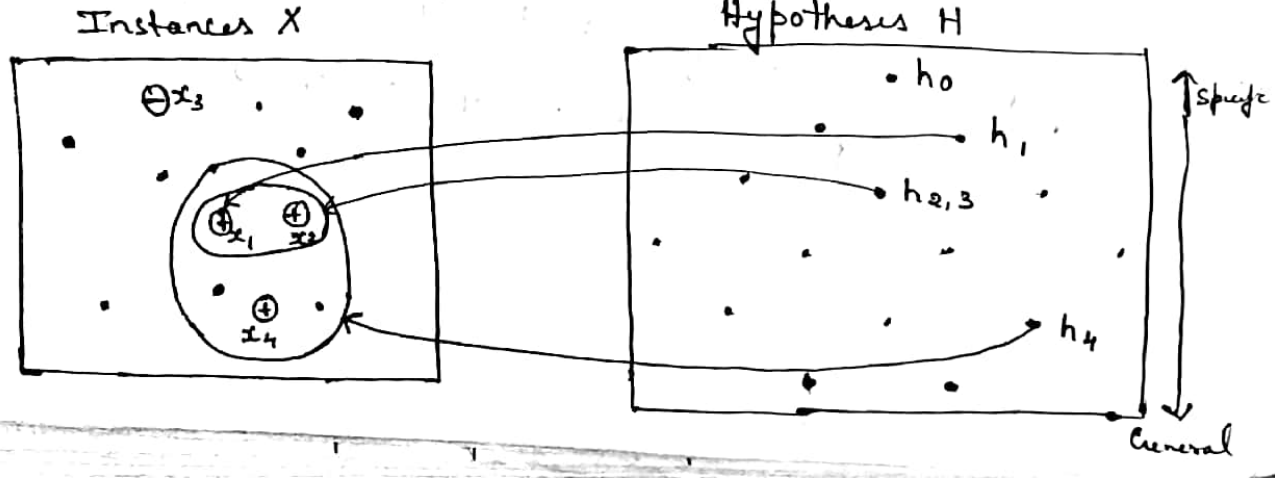
$h \leftarrow$ (sunny, warm, ?, strong, warm, same)

→ Consider Ex 3, FIND-S algorithm ignores negative examples.

→ Consider Ex 4,

$h \leftarrow \langle$Sunny, Warm, ?, strong, ?, ?$\rangle$

Conclusion: FIND-S algorithm moves from hypothesis to hypothesis, searching from the most specific to progressively more general hypotheses



Property of FIND-S algorithm : For a hypothesis space H, FIND-S is guaranteed to output the most specific hypothesis within H.

Questions left unanswered by FIND-S

1. Has the learner converged to the correct target concept?
2. Why prefer most specific hypothesis?
3. Are the training examples consistent?
4. What if there are several maximally specific consistent hypotheses?

or (2)

Algorithm (FIND-S)

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   - → For each attribute constraint a_i in h:
     - → if constraint a_i in h is satisfied by x, then do nothing
     - → Else replace a_i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h.

Supervised learning

Problem: Set of cars, group of people to whom the cars are shown.

Case: If cars shown = family cars ⇒ positive ex
If cars shown ≠ family cars ⇒ Negative ex
} Labeled data

Class learning: Finding a description through positive examples = class learning

Knowledge extraction: In this case, the features that separate a family car from other type of cars are extracted.
for ex attrib are price & engine power

Representation: Each car (family car) is represented using two numeric values

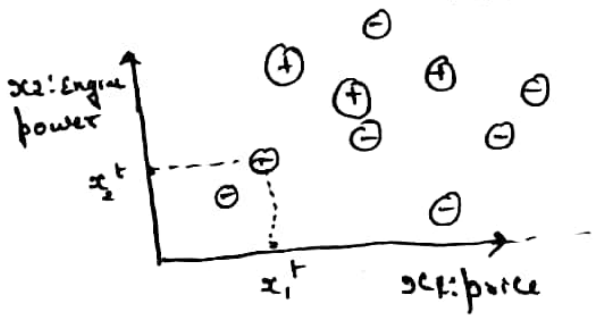
$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad r = \begin{cases} 1 & \text{if } x \text{ is positive example} \\ 0 & \text{if } x \text{ is negative example} \end{cases}$$

∴ Each car is an ordered pair (x, r), the training set contains N such examples.

$$X = \{x^t, r^t\}_{t=1}^{N}$$

t Indexes different examples in the set

Plot of training data set

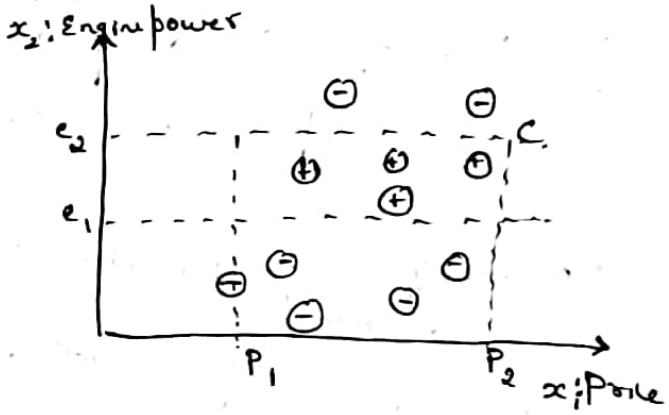


Hypothesis class: For a car to be family car, its price and engine power should be in a certain range

ie $(P_1 \leq \text{price} \leq P_2)$ AND $(e_1 \leq \text{engine power} \leq e_2)$ —Eq (1)

Equation (1) fixes the hypothesis class $\mathcal{H}$.

Hypothesis: The learning algorithm then find the particular hypothesis $h \in \mathcal{H}$



Linear Regression

Regression is predictive modelling which investigates the relationship between a dependent and independent variable.

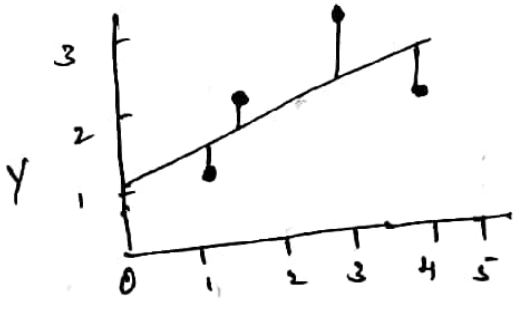
Regression analysis needs a curve or line to the data points in such a manner that the differences b/w the distances of data points from the curve or line is minimized.

Linear regression establishes relationship between dependent variable (y) and one or more independent variables (x) using a best fit straight line (also known as regression line)

$$y = \underbrace{a}_{\text{intercept}} + \underbrace{bx}_{\text{slope}} + \underbrace{e}_{\text{error}}$$

How to obtain bestfitline :- Accomplished by least square method, where the best fit line is found by observing data by minimizing the sum of squares of the vertical deviations from each data point to the line

$$\min_{w} \|X_w - y\|_2^2$$



other Reg ex
→ step wise Reg
→ Ridge "
→ polynomial "

Note : Logistic regression is used when the dependent variable is binary (0/1, T/F, Yes/No)

Example of Learning association algorithm.

Apriori These algorithms mine the frequent itemset.

Consider the transaction DB

Customer	Items
C_1	Milk, Egg, bread, chip
C_2	Egg, popcorn, chip, coke
C_3	Egg, bread, chip
C_4	milk, Egg, bread, popcorn, chip, coke
C_5	Milk, Bread, coke
C_6	Egg, bread, coke
C_7	milk, bread, chip
C_8	milk, Egg, Bread, butter, chip
C_9	milk, Egg, butter, chip

Assume minsupport = 30%. $\Rightarrow$ 3. Generate canditates

C_1

Itemset	Sup count
Milk	6
Egg	7
Bread	7
popcorn	2
Butter	2
chip	7
coke	4

$\longrightarrow$

Itemset	
milk	6
Egg	7
Bread	7
chip	7
coke	4

C_2

Itemset	Supcount	
milk, Egg	4	
milk, bread	5	
milk, chip	5	
milk, coke	2	X
Egg, bread	5	
Egg, chip	6	
Egg, coke	3	
Bread, chip	5	
Bread, coke	3	
chip, coke	2	X

Note: weka → Loa...
supermarket · arff
→ select associate
(default apriori
→ start.

Generate C_3 ~~Itemset~~

Itemset	Supcount		
milk, Egg, bread	3		3
milk, Egg, chip	4		4
milk, egg, coke	1	X	
milk, bread, chip	4		4
milk, bread, coke	2	X	
egg, bread, chip	4		4
Egg, bread, coke	2	X	
Egg, chip, coke	2	X	
Bread, coke, chip	1	X	

Generate C_4

Itemset	Supcount	
milk, Egg, bread, chip	4	∴ these items are

sold frequently.

Consider the following set of points $\{(-2,-1),(1,1),(3,2)\}$ (2)

a) Find the least square regression line for the given data points.

b) Plot the given points & the regression line

Solu

x	y	xy	x^2
-2	-1	2	4
1	1	1	1
3	2	6	9
$\Sigma x = 2$	$\Sigma y = 2$	$\Sigma xy = 9$	$\Sigma x^2 = 14$

$$a = \frac{[n\Sigma xy - \Sigma x \Sigma y]}{[n\Sigma x^2 - (\Sigma x)^2]} = \frac{3\times 9 - 2\times 2}{3*14 - 2^2} = \frac{23}{38}$$

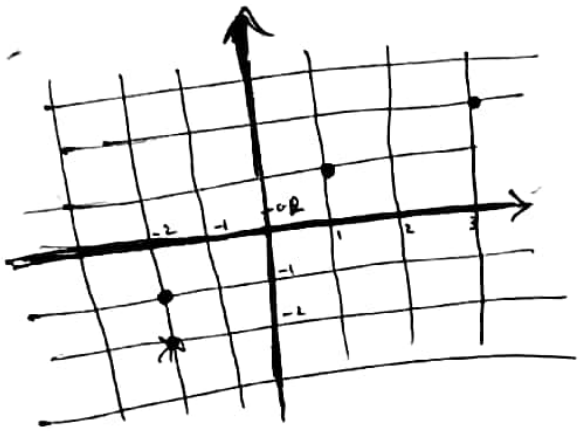
$$b = \left(\frac{1}{n}\right)(\Sigma y - a\Sigma x) = \frac{1}{3}\left(2 - \left(\frac{23}{38}\right)*2\right) = \frac{5}{19}$$

$y = ax + b$ → graph

$$y = \underbrace{\frac{23}{38}}_{\text{slope}} x + \underbrace{\frac{5}{19}}_{\text{intercept}}$$

0.6, 0.2

slope = slantness = $\frac{\text{change in } y\text{ value}}{\text{change in } x\text{ value}}$



* the sales of a company (in million dollars) for each year are shown in the table below.

x (Year)	2005	2006	2007	2008	2009
y (sales)	12	19	29	37	45

a) Find Least square regression line $y = ax + b$

b) Use the least square regression line as a model to estimate the sales of the company in 2012

Solu change the variable x into t i.e $t = 2005 - x$ (scalability)

i.e

t	0	1	2	3	4
y	12	19	29	37	45

t	y	ty	t^2
0	12	0	0
1	19	19	1
2	29	58	4
3	37	111	9
4	45	180	16
$\Sigma x = 10$	$\Sigma y = 142$	$\Sigma xy = 368$	$\Sigma x^2 = 30$

$$a = \frac{n\Sigma ty - \Sigma t \Sigma y}{n\Sigma t^2 - (\Sigma t)^2}$$

$$= \frac{5 * 368 - 10 * 142}{5 * 30 - 10^2} = 8.4$$

$$b = \frac{1}{n}(\Sigma y - a\Sigma x)$$

$$= \frac{1}{5}(142 - 8.4 \times 10)$$

$$= 11.6$$

b) In 2012 $t = 2012 - 2005 = 7$

estimated sales in 2012 $y = 8.4 * 7 + 11.6$

$= 70.4$ million dollars

Note: Apriori Algorithm works on 2 steps
→ join (cross join)
→ Prune

Ex ②

Transaction — min support cost :-2

T_1 — 1, 2, 5

T_2 — 2, 4

T_3 — 2, 3

T_4 — 1, 2, 4

T_5 — 1, 3

T_6 — 2, 3

T_7 — 1, 3

T_8 — 1, 2, 3, 5

T_9 — 1, 2, 3

Solu —

iteration 1

Item	count
I_1	6
I_2	7
I_3	6
I_4	2
I_5	2

iteration 2 combination

Item	count
I_1, I_2	4
I_1, I_3	4
I_1, I_4	1 ✗
I_1, I_5	2
I_2, I_3	4
I_2, I_4	2
I_2, I_5	2
I_3, I_4	0 ✗
I_3, I_5	1 ✗

⟹

I_1, I_2 — 4
I_1, I_3 — 4
I_1, I_5 — 2
I_2, I_3 — 4
I_2, I_4 — 2
I_2, I_5 — 2

1, 2, 3 → 2 ✓

1, 2, 4 → 1

1, 2, 5 → 2 ✓

1, 3, 4 → 0

1, 3, 5 → 1

1, 4, 5 → 0

2, 3, 4 →

2, 4, 5

⋮

I_1, I_2, I_3

I_1, I_2, I_5

} Frequent itemsets

Disadvantage:- Useless combination

Solution :- F.P-growth.

271 299
317 311
286 556
472
290
256
276
473
475