Towards Intelligence Machine:

Machine with human-like traits.

↳ casting intelligence into a m/c

In human central Nervous system (CNS) ⟷ { intelligence "sensors, control. ↳ cognitive aspects & decision-making.

Initially computer → store info.
cognitive process → develope new math tool and h/w.

Biological neuronal process are complex and difficult to understand.

Scientist can achieve only limited understanding of brain. → based on this new computer theories under Neural Nfw has been evolving.

cognitive functions of brain → based on relative grades of information.

Our neurons accept info in the form of grades rather than numbers.

The perceptions and actions → appears in ~~based on~~ the form of relative grades

Theory of fuzzy logic → provides notion of grades emulation of cognitive functions.

Appl

Neural N/W + Fuzzy logic → Cognitive Information.

Needs, Motivations and Rationale:

Collecting information is very essential.

The collected info may be categorized on the basis of nature of experience

- experimental data (example, sample, measurements, record, patterns or observations)
- Structured human knowledge (experience, expertise, heuristics)
  - ↳ expressed in (If-then rules.)

Eg: In man-machine control system, an experienced process operator employes → IF-THEN rules to control a process.

Operator estimates the important process variables and based on that it manipulates ctrl signal.

→ Industry process.

Soft Computing | ML:

To solve decision making problems:

- ↳ variables related to the problem.
- ↳ Relation among the variables are expressed in terms of mathematical equations.

Decision → given by solution of these equations

Generalizing :

Recognizing similarity b/w different situations. This is very useful b/c we can use our knowledge in situations unseen earlier.

In m/L o/p is defined by learning task. Different types of learing task appears in real-world application.

classification learning → classification based on measurement / observation.

Regression learing → prediction.

clustering → Grouping the observation.

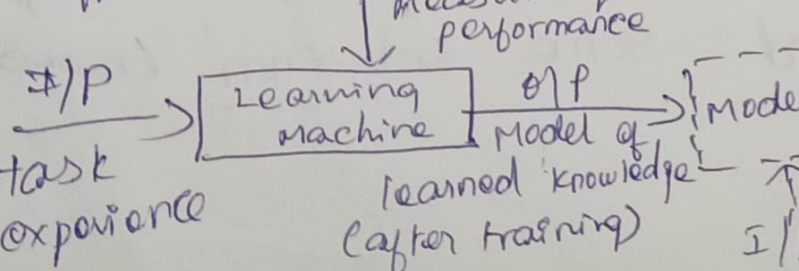
classification / Regression → Directed / supervised learing.

clustering → Unsupervised learning.

Here experience → available in the form of data

I/P → Raw data can be in any form numeric, image, video etc.

O/P → learned knowledge. → model. → used for decision-making



ML => meant for automat...

Now-a-days ocean of data to be processed. Humans are unable to extract information from them.

So human-like ability need to be incorporated into the S/W. This is the essence of M/L.

With ML a new mathematical theory has been emerged which is built on learning, memorizing adapting and generalizing.

Well-posed M/L problems:

ML concerns with the question how to improve the performance of a task through experience?

To have a well-defined learning problem, we must identify following three features:

* The learning task.
* The measure of performance
* The task experience.

} → Key concepts learning from experience.

Important aspect of learning from experience is remembering, adapting and generalizing.

Remembering and Adapting:

Recognizing the situation and used the solution which worked last time for similar type of problems and avoid some solution based on past experience.

Cross - validation:

yzhbbhb. (3)

To overcome overfitting.

k-fold

↳ Evaluate model for multiple times.

K → groups that given data sample can split.

Procedure:

1. Shuffle the dataset randomly
2. split data set into k-group.
3. For each group
   1. tak a group as test data set
   2. train with remaining group
   3. Fit the model & evaluate
   4. Retain the evaluation score and discard the model.
4. Summarize with evaluation score.

1, 2, 3, 4, 5, 6.

K = 3.

4, 3, 1, 6, 2, 5

	Train	Test	
F1	4, 3	F2 & F3	F1 = Score
F2	1, 6	F3 & F1	F2 = → Model based
F3	2, 5.	F1 & F2	F3 = on these observation

AV Scor

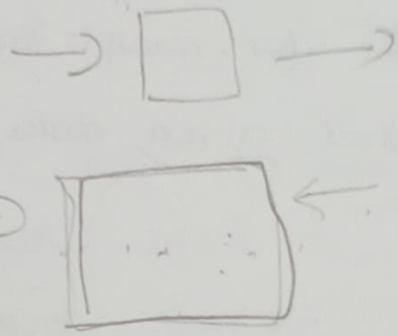
split (dataframe) — !

X_train = C1, C2

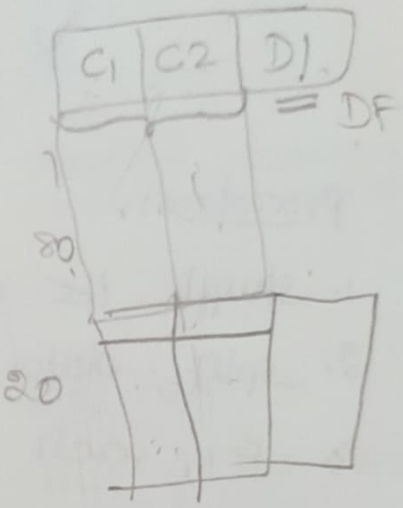
Y_train = D1

① shuffle.

② 80%.



Condition + con



m1 = LR ( )

X_test = C1, C2

Y_test = D1

m2 =

m1_pot (X_test)

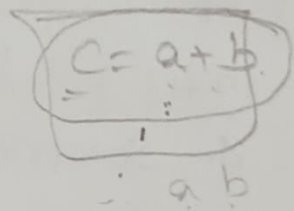
m2 = LR

MAE ( ) =

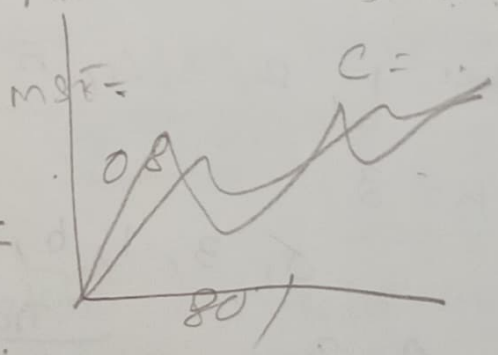
MSE (Y_test, m2) = Less

R2 ( ) = 0.92

92%



MR



MR → optimal

0.92% ↑

Application of ML in diverse fields:

M/L ⇒ ment for automatic learning from voluminous datasets.

Mining knowledge from data.

Data mining → Google. ⇒ offers info from billions of web pages. → 1st website → using PageRank algo.

Personalized product recommendations:

Eg:- Amazon ⇒ recommender system ⇒ m/l.

based on data from social n/w.

⇒ To predict suitable content for the users.

Information collected based on rating

↳ explicit rating → given by users.

↳ Implicit rating → rating estimated from social media.

Computer n/w security:

system → to find unusual patterns of activity.

Privacy-preserving data mining assist in protecting privacy-sensitive informations.

Medical diagnostics:

↳ Accurate diagonosis → challenging.

↳ Medical images. → major success.

ML + DL → More accurate diagnosis.

Finance Domain:

Helps to offer personalized services to the customers at low cost.

Predict Frauds. → M/L can scan large amount of transactional data and identify if there is any fraudulent behaviour.

Stock Market forecasting:

Prediction of stock index.

Here data is in the form of time series.

Above two domain data are independent.

So here prediction is done using the past history.

Machine vision:

Deep learning applied in major success.

Images were captured → converted to numerical values for the process.

Numerical values → pixel values

Gray scale → matrix of pixel intensity

M/L algo. ← vector representation

⇒ Biometric recognition, Medical image analysis, handwritten digit recognition system.

Speech Recognition:

Signal processing technique → Speech signal → real values.

Virtual Personal Assistants → finding info when asked over voice.

Patient Index	Headache	Muscle-pain	Temperature	Flu (5)
1	Yes	Yes	high	Yes
2	no		low	no.
3		no.	normal	
4			Very-high	
5				

Data are in Nominal representation → acceptable if table size is small.

Large → converted to some numerical value.

Direct / supervised learning → outcome for each observation is known as a priori.

expressed by an attribute called decision attribute.

Information system of this kind are called decision systems.

In the above table Flu → decision attribute which categorize patient into two classes Flu → yes or Flu - No.

Other attributes are called condition attributes

Features x_j / Instances $s^{(i)}$	x_1 x_2 . . . . x_n	Decision y
s^1		
s^2		
:		
s^N		

Values can be numeric, categorical or discrete.

Training experience is available in the

controlling a robot with voice commands.

Text Mining:

Identification of pattern in text data.

Eg: Natural Language Processing.

Spam Email filtering.

Robotics and Automation:

M/L → source of intelligence for robot.

Robots in Industrial automation, medical robots, military robots, so on.

Data Representation:

Experience → raw data.

Raw data → requires pre-processing.

this leads to information system that represents knowledge in raw data.

Data will be stored in datawarehouse.

Data warehousing provides integrated, consistent and cleaned data to machine learning algorithm.

Some application data → flat files → data tables.

Row → measurement / observation.

column → gives values of an attribute.

row → instances, samples, record, pattern, object, case, events.

column → attributes & features.

→ depends on the application.

Time series Data:

Data structure of forecasting problems.

Time series data → sequential → a sequence of observation is measured over time. and each observation is indexed by t.

The measurement are taken at fixed time interval.

$$y(t) = f(\underbrace{y(t-1), y(t-2) \cdots}_{\text{past values of the signals.}})$$

$y(t)$ → o/p at t,

NARMA → (Nonlinear, Auto-Regressive, Moving Average) Model.

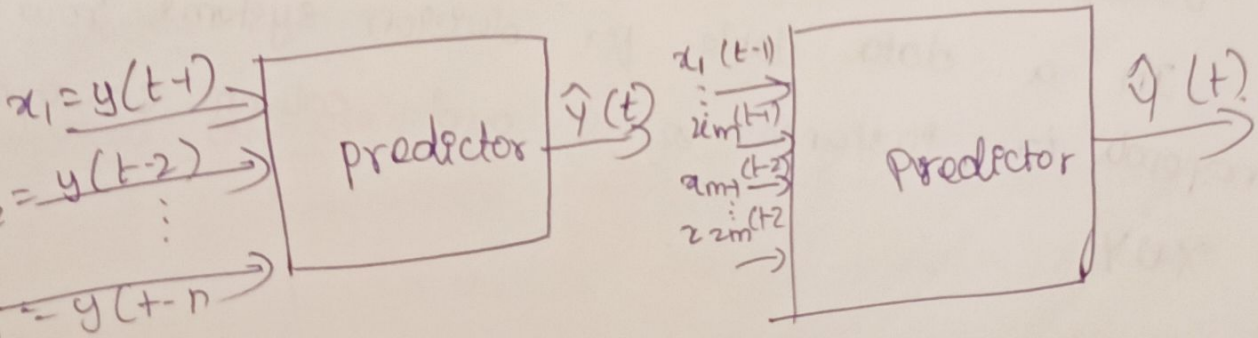
$f(\cdot)$ → very complex.

The number of Lags L in the series is application dependent.

$$\hat{y}(t) = f(y(t-1) \cdots y(t-L))$$

More pass info → prediction more accurate.
& less noise.

Time series → Regression problem.



Univariant, Bivariant, Multivariate.

Student height → one variable → 2 variable, → 2 variables

form of N examples.

$s^{(i)} \in S, i = 1, 2 \cdots N$ where S is a set of possible instances.

Each instance with n features given by a vector

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = [x_1 \; x_2 \cdots x_n]^T \in R^n$$

the set X is a finite set of feature vector $x^{(i)}$ for all possible instances.

the pair (S, X) contributes the information system

where

$$\{ s^{(i)}; i = 1, 2 \cdots N \} \in S$$

$$\{ x^{(i)}; i = 1, 2 \cdots N \} \in X$$

$$\{ x_j^{(i)}; j = 1, 2 \cdots n \} = x^{(i)} \quad V_{xj} = \{ V_{1xj}, V_{2xj} \cdots V_{dxj} \}$$

The tuple (S, X, Y) constitutes a decision system where finite set of condition attributes $x_j \in X$, and the decision attribute (output) $y \in Y$

In a data table for decision systems, row corresponds to instance of S and column corresponds to $X \cup Y$.

Structured / Unstructured Data:

Data modeling, establish → logical structuring.

↳ determine how data is stored, organized, and manuplated in DB.

Structured → Pre-defined data model.

↳ Relational model. → Tables.

↳ Easy to aggregate from various location. Easy to analyse.

Unstructured → No data model / not organized in a predefined manner.

↳ text & multimedia data.

↳ images, videos, audio etc.

Having internal structure but it doesn't fit neatly in relational database.

Semi-structured. → Doesn't fit to data model but have some organizational properties that makes it easier to analyze.

Little processing → convert to the format acceptable by machines.

Eg: JSON [Javascript object Notation].

XML, CSV, HTML.

Power of Unstructured data:

80% → unstructured.

So non-relational DB → can be used to store unstructured data.

↳ Apache, MongoDB, Hadoop etc.

DL → tools for unstructured data } → analyze and search for some structural form in the data.

Text → will not have random letters as neighbours.
↳ follow grammer.

Images → Nearby pixels are close to each other in values.

Speech → commonly used sounds → Phonics.

Natural Language processing
Computer vision
Speech synthesis
} → tool to capture some structures in unstructured data.

This is what done in DL.

Domain knowledge for Productive use of ML:

ML knowledge + Domain knowledge

Main for feature selection + feature extraction.

The key is to use features that ~~are~~

are computationally feasible

leads to good machine learning success and

reduce problem data into managable data without discarding valuable informations.

compactness of feature vector is important → Reduce computation Complexity, Generalization.

Feature with less info → leads to poor generalization.

More data → More chances for errors.

- Same range → no issues
- Beyond range → called outliers.

Some data set → large number of missing values.

- Dropping → Data loss
- Imputation → based on existing pattern.

Know the importance of each attribute.
↓ In this case
Domain knowledge places major role.

Forms of Learning:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Learning based on natural processes: Evolution, Swarming and Immune System.

Supervised Learning:

uses labelled dataset.
Model will be developed and tested with labelled dataset.

$$D = \{ s^{(i)}, y^{(i)} \} \; i = 1, \ldots N$$

$g^{(i)} = x^{(i)} : \{x_1^{(i)}, x_2^{(i)} \dots x_n^{(i)}\}$.

Two types of task in supervised/ direct learning

→ Classification [pattern recognition]

→ Regression: [numeric prediction]

Unsupervised / undirected Learning: [self-instructions].

No lables.

From the given set of i/p pt will find a patter and classify according to pt.

↙ → cluster analysis → very useful for data exploration
↓
understant structure of data

or similarities → Dimensional reduction algo.

↳ Reduces the features without affecting the generality.

Reinforcement Learning:

↳ Learning the optimal behaviour in an environment to obtain maximum reward.

↳ Learn by trial and error method.

the two aspect are

trial-and-error

cumulative reward

Agent has to take the action and learn from the result of action taken.

...ning based on natural processes:

Evolutionary computation:

Drive ideas to develop search and optimization methods that helps to solve complicated problems.

Genetic algorithm, evolution strategies, evolutionary programming and genetic programming will come under this.

Swarm Intelligence:

↳ Feature of system of unintelligent agent with inadequate individual ability

1) Intelligent behaviour ← collectively

↳ Base for Ant colony optimization: Finds shorte root for food.

2) Processing the knowledge → PSO. [particle swarm optimization]

Artificial Immune system:

↳ resist from disease.

AIS → applied to solve pattern-recognition problems and cluster data.

Natural Immune System → Match patterns of the cells and protect our body from foreign cells.

10

Supervised Learning:

Learning from observation.

$S \rightarrow$ set of observations / patterns.

$s^{(i)} \in S ; i = 1, 2 \cdots N$

A pattern is specified by n attributes / features. x_j; $j = 1, 2 \cdots n$

where each feature has real values for a pattern

Domain of $\forall x_j \in R$

Data pattern $s^{(i)}$ has feature set $\{x_1^{(i)}, x_2^{(i)} \cdots x_n^{(i)}\}$

where $x_p^{(i)} \in V_{xj}$.

Each pattern can be visualized with n numerical features as point in state space R^n.

$$x = [x_1 \cdots x_n]^T \in R^n$$

$X \rightarrow$ set of feature vector. $x^{(i)}$ for all patterns.

$\hookrightarrow$ representation space.

In supervised learning decision attribut

$y \rightarrow$ a priori

In classification problem $y \rightarrow$ given by the set

$$V_y = \{1, 2, \cdots M\}$$

o/p $y^{(i)}$ for $s^{(i)}$ takes values from the set V_y.

Thus in classification problem each is associated with $y^{(i)} \in V_y$.

For regression.

$V_y \in R$

each $x^{(i)}$ has $y^{(i)} \in R$.

↳ State space.

Different pattern $x^{(i)}$ in $X \rightarrow$ have different frequencies.

different probability distributions.

Each training examples trainer ① ⟶ learning m/c

forward op ⟶ ②

Training experience is in the form of data D.

$D: \{x^{(i)}, y^{(i)}\}$ $i = 1, 2 \ldots N$.

In general learning → reliable → training follow distribution similar to unseen data

Assume D is independently drawn and identically distributed and a m/c is defined by a function

$f: X \rightarrow Y \rightarrow$ op. space } trained machine.

↓

ip space

Able to find $\hat{y} = b(x)$ for a given patter x.

To assess success of learning, → evaluation criterions are used.

Evaluation criterions
↳ based on decisions like → errors, profits or loss, penalties or rewards etc.

In supervised learning → minimization criterion potential loss.

From set of possible ML functions
optimal one ← choose
↳ which minimizes the loss.

ML function can be defined by
$f(x, w)$
↳ adjustable parameters.

A loss function
$L(y, f(x, w))$ → measure of error b/w actual output y and estimated o/p.

$$\hat{y} = f(x, w)$$

If $\hat{y}$ → estimate o/p
y → actual o/p.

Loss associated with decision $f(x, w)$

$$E[L(y, f(x, w))] = \int_{X \times Y} L(y, f(x, w)) \underbrace{p(x, y)}_{\text{Joint probability distribution of data.}} dx\,dy = R(w)$$

(p.e) loss is calculated on entire space of x and y.
(i/p) (o/p)

A risk → predictable loss in decision theory.
R(w) → Risk function.

Problem is to find decision function f(x, w) against p(x, y) that minimizes the risk function R(w).

Thus it is easy to design optimal classifiers/numeric predictors if joint probability density function p(x, y) is known.
↓
very hard to find.

Empirical - Risk - Minimization:

Since p(x, y) → not known true risk given by R(w) → cannot be calculated.

With available info → trying to calculate the risk.

Training dataset → x, y → unknown probability distribution.

So, risk must be approximated with empirical risk, Remp(w).

$$R_{emp}(w) \triangleq \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)}, w))$$

(i.e) average over $p(x, y)$ ⟶ replaced by average over training samples.

In classification

$$L(y, f(x,w)) = \begin{cases} 0 & \text{if } y = f(x,w) = \hat{y} \\ 1 & \text{otherwise.} \end{cases}$$

↳ classification error.

In regression.

$$L(y, f(x,w)) = \frac{1}{2}(y - f(x,w))^2$$

(i.e) difference b/w real and estimated value.

$y - f(x, w)$.

But finding empirical risk is an ill-posed problem.

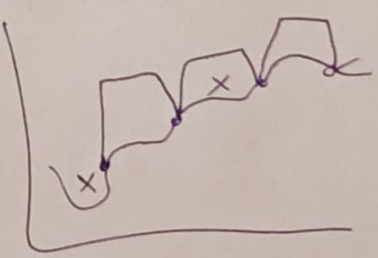
(i.e) problem may have n number of possible solutions.

This will work based on the assumption that unseen data are having similar distribution like training data. But this can not be guaranteed.

Real-world data may be incomplete, noisy and inconsistant.

Consider a regression problem.

$(x^{(1)}, y^{(1)} \ldots x^{(n)}, y^{(n)}) \in X \times Y$.



Functions which interpolate those ~~values~~ data points will be zero for Remp.

X → outside training data set.

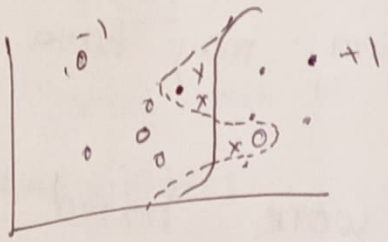
both are performing bad.

thus, interpolate functions that results in zero empirical risk can mislead.

There are many other approximating functions that will minimize empirical risk (training error) but not necessarily the true (expected) risk.

consider the classification problem ⇒ two class pattern recognition.

$f(\cdot) : X \to \{\pm 1\}$ that is, $Y = \{\pm 1\}$



dashed line → seperating correctly,

consider test points → ~~pot~~ o/p known but not used for training.

~~function~~ →

classifier → zero training error not able to get all test point right.

∴ Minimization of empirical risk (i.e training error) is thus not a solution for a learning task.

Minimization of true error → Mahesh Huddar → is the exact learning task not possible → $p(x, w)$ → unknown.

↳ solution is inductive inference.

Inductive Learning: → Learning from examples
↳ Generalized conclusion

Consider $(x^{(i)}, f(x^{(i)})); i = 1, 2 \ldots N$

returns function $h(x)$ that approximate $f(x)$

In statistical literature approximation function $h(x)$ is called hypothesis function.

$f(x)$ → true function that maps input space x to output space y.

But $f(x)$ is not known in real-word decision-making problem.

It is not easier to tell whether $h(\cdot)$ is a good approximation of $f(\cdot)$.

↳ Generalize well → predict novel patterns correctly.

∴ Generalization performance → fundamental problem in inductive learning.

↳ Judge using test dataset.

The off-training set error → used to measure generalization performance.

Assumption → Ideal hypothesis related to unseen patterns is one induced by observed training set.

Obtain general hypothesis → empirical regularities over training sets.

Induce approximation over mapping.

For a particular data set if a specific hypothesis outperforms over another, it is not necessary it will work fine for all data sets.

BIAS and VARIANCE:

Random sample → D of N and measure training error from the function.

$$R_{emp}(w) \triangleq \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^{(i)}, w))$$

Let's denote error based on D and hypothesis h as $error_D[h]$.

For various samples $(x^{(i)}, y^{(i)})$, we will get different $error_D[h]$.

$\therefore$ $error_{D_j}[h]$ → random variable.

For K, experiment, the average over K.

$$error_D[h] = E_D\{error_{D_j}[h]\}.$$

$E_D\{\cdot\}$ → expectation (or) Ensamble average.

Let's the regression function estimated is denoted $h(x : D_j)$ for training set D_j.

Due to random selection some approximation will be excellent ; some will be poor.

The natural measure of the effectiveness of the estimator can be expressed as the mean-square deviation

$$error_{D_j}[h] = [h(x:D_j) - f(x)]^2$$

The value of this quantity depends on the dataset D_j.

Average over complete ensemble of dataset as

$$error_D[h] = E_D\{[h(x:D_j) - f(x^*)]^2\}.$$

A non-zero error can arise for essentially two reasons.

1. It may be hypothesis function $h(\cdot)$ is, on an average, different from regression function $f(x)$. This is called bias.

2. It may be hypothesis function is very sensitive to particular D_j, so that for a given x approximation error is larger for some dataset and smaller for other dataset. This is called variance.

variance measure level to which the hypothesis function $h(x; D_j)$ is sensitive to specific selection of dataset.

Higher the complexity of hypothesis function, [i.e more flexible with many parameters], lower is the approximation error.

variance → reflects capability of the trained model to generalize to other data samples.

Lower the variance → estimate of $f(x)$ does not change much.

But higher the complexity of hypothesis, higher the variance.

Simpler model → higher bias, lower variance

Complex model → lower bias, higher variance.

∴ hypothesis that results in low bias and low variance is needed to minimize over all mean square error.

This is called bias-variance dilemma or bias-variance trade off.

In general finding optimal trade off is hard, but acceptable solution can be found.

$$error_{D_j}[h] = [h(x; D_j) - f(x)]^2$$

$$= [h(x; D_j) - E_D\{h(x; D_j)\} + E_D\{h(x; D_j)\} - f(x)]^2$$

$$= [h(x; D_j) - E_D\{h(x; D_j)\}]^2 + [E_D\{h(x; D_j) - f(x)\}]^2 + 2[\quad,,\quad][\quad,,\quad].$$

$$(a+b)^2 = a^2 + b^2 + 2ab.$$

We take expectation of both sides over ensemble of dataset D.

$$error_D[h] = E_D\{[h(x; D_j) - f(x)]^2\}.$$

$$= E_D\{[h(x; D_j) - E_D\{h(x; D_j)\}]^2\} + E_D\{[E_D\{h(x; D_j)\} - f(x)]^2\} - 2\,.$$

Here:

variance $\leftarrow E_D\{[h(x; D_j) - E_D\{h(x; D_j)\}]^2\}$

Bias $\leftarrow E_D\{[E_D\{h(x; D_j)\} - f(x)]^2\}.$

Bias $\Rightarrow$ measures the level to which average hypothesis function differs from desired function $f(x)$.

Occam's Razor principle and overfitting avoidance:

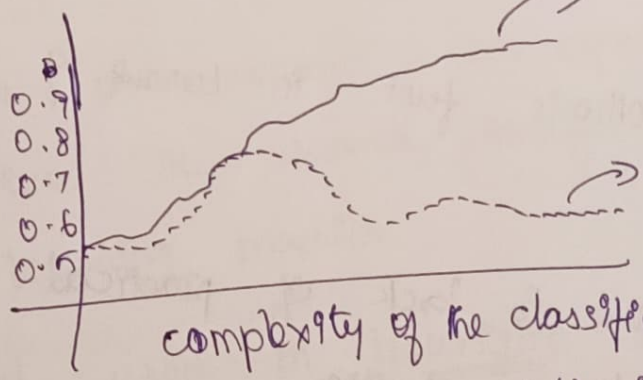
If there are two algorithms and both of them perform equally well on training set, then simpler algorithm can be expected to do better on a test set.

simpler → lesser parameter,
" training time
fewer attributes

" When a design with 'good enough' solution arrives, we not necessarily test for optimal one.

Occam's Razor principle suggest hypothesis function that avoid overfitting of train data.

Overfitting:



[more no of weights in NN,
more no of nodes in tree etc].

Accuracy get affected when test dataset lies in different state space compare to train dataset.

Heuristic search in Inductive Learning:

15/7/24 M/c Learning aim → built a Statistical model

22, 28, 29, 35, 45, ~~44~~, 48, 54, .

good generalization.

Success → depends on → hypothesis space complexity
① sample complexity.
② bias - variance trade-off.

Best generalization is achieved through the best compromise b/w requirement of small bias and small variance.

In order to find optimal bias & variance } have to control → complexity of hypothesis function.

Structural-risk minimization principle

↓

Ordered hypothesis fcnt in terms of their complexities.

In ML there is lack of practical tool for real-life practical complexities.

Bias-variance trade off is also theoretical result because lack of tools to find optimal bias and variance.

∴ Heuristic ~~strag~~ strategies ~~for~~ like overfitting avoidance - regularization, early stopping, pruning are some of heuristic tools explored today.

Search through Hypothesis space:

Heuristic → cannot be precisely predefined. uses trial-and-error method for good solution. the search for hypothesis is done with the following steps:

i) the search is first focused on class of hypothesis, chosen for learning task. Prior knowledge and experience are needed. Different hypothesis for different learning task and dataset.

ii) For each member, corresponding learning algorithm organize the search through all possible structure of learning machine.

Techniques used in heuristic search to optimize hypothesis complexity:

Regularization: → promotes smoother function by creating new criterion function not

only based on training error but also algorithmic intricacy.

It avoid extremely complex hypotheses.

New criteria can be written as

$$\bar{E} = E + \lambda \Omega$$

= error on training set + λ x hypothesis complexity.
↳ weight of penalty.

Regularization term.

when $\lambda = 0$ → no regularization. } causes overfitting
But have some variance

when $\lambda \uparrow$, variance ↓.

Till some point, this decrease in λ is useful b/c variance ↓ without subsequent ↑ in bias.

But at certain point bias starts ↑. and thus leads to underfitting.

λ ⇒ optimised using cross-validation.

Early stopping:

Training of learning machine → iterative decrease in error function.

During training session error ↓ as iteration ↑ funt

stopping iteration before attaining minimal error } → affect the hypothesis

Pruning:

Alternate solution to early stopping.

Removing certain parts of model → to prevent overfitting.

Goal is to improve model's performance, generalization and efficiency by reducing its complexity.

Evaluation of Learning System:

Some important aspect of evaluation are

Accuracy:

Generalization capability is the index of accuracy.

Robustness:

machine should perform well in all circumstances including data with noises, incomplete and with irrelavent data.

↳ These all will present in real-time data.

~~Robustness~~ based on

Computational complexity and speed:

↳ for efficiency of learning algo.

(i-e) how fast a machine can work and how much memory is needed by the machine.

Interpretability:

↳ It is subjective and difficult to evaluate.

↳ It is easy in decision tree,

but interpretability ↓ when complexity ↑.

Online Learning:

Sources → generating streaming data analyzed in real time

→ online learning system → receives i/p from real-time environment and analyze it.

Scalability:

High level of scalability is needed.

The assessment of scalability can be done with dataset size complexity.

Combining Multiple Models: Ensemble learning.

No Free Lunch → No single learning algo always gives most accurate learner.

Each algo → model → set of assumptions.

this inductive bias → Error → if assumptions do not hold for data
→ less Error → if assumptions work.

Several model → choose the best performing
→ combine them to produce ensemble of learned models.

But suitably combining multiple models, accuracy can be improved.

Even though it is counter to Occam's razor [simplicity], due to ↑ predictive performance it is applied to both classification and regression tasks.

The idea of ensembling is to build a predictive model by combining the strength of base models.

The main aim is to find set of diverse [differs in their decisions] learners that complement each other. And also the gain in overall success will not be achieved if

the base learners are not reasonably accurate in the domain of problem.

Maximizing overall accuracy of individual learners } more important < Requirement of diversity.

∴ Ensemble learning can be broken down into two tasks.

⇒ Generating a committee of base learners that complements each other.

⇒ Combining o/p of weak learners for maximum accuracy of the composite predictor, which leads to strong learner.

The Base Learners:

Different algorithms to generate set of base learners. → makes different assumption about the data.

→ different predictors [Diversity].

Eg: For a given data set different algo can be applied like NN, SVM, DecisionTree } → base learners. (or) Same algo with different hyperparameters.

Eg: NN - weight, hidden units
SVM - Kernal function
Threshold in decision tree etc.

Generate base learners by training it with different subsets of training data. (or)

Combining base Learners:

Combining decisions of various base Learners } different O/P → single prediction.

⇒ Bagging technique } → Individual learners are created independently } → bag of base learners.

⇒ Boosting technique } → new model is impacted by the performance of earlier.

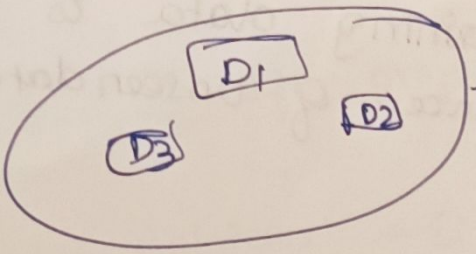
Model with accuracy > average } + new component learner

↳ Join decision rules → high accuracy

performance boosted.

⇒ Random forest:

↳ Ensemble consist of → Decision tree.



→ Group of DT forms the Random forest.

Individual tree

↳ created with random choices of attribut at each node To split.

Random forest built with } → bagging + Random attribute Selection.

↳ combines model of same type [D.T].

In stacked generalization or stacking

base learners ← different to complement each other.

↳ Different learning algorithms.

But stacking is not used widely

↓

Difficult to analyze theoretically.

Decision tree ⇒ used more often for bagging / boosting

← classification problems ← Success rate is high.

Estimating generalization errors:

Success of learning } → hypothesis space complexity + complexity of samples

Is the performance on training data is the proper indicator of performance of unseen data.

↳ Ans is -ve.

To predict the performance of learning model } → test data are used

↓

not used for building the model.

Some learning models } → 2 stages
- Building the elementary (or) base structure.
- optimize the parameter in the structure.

Seperate set of data may be required in two stages.

In such cases three datasets are considered

training data → create the structure of learning model
validation data → optimize the parameters.
test data. → compute the error rate of the optimized model.

Each of the dataset should be selected independently.

If large set of data } → training + test.

If properly trained → best model.

Metrices → test data target } compared with feature
predicted value

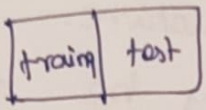
If dataset → training + validation + test
used only for optimization.

Holdout method and Random subsampling:

training + testing.

↓

1/3rd of data.



time series data → Earlier data ↓ training | Latter part ↓ testing.

consider classification problem:

Each class in the full dataset should be represented in both training and testing.

Eg: Image → cat, dog.

In training the target should ~~have~~ include both feature cat, dog.

similarly in testing also.

To ensure this random samples are selected for training and testing dataset.

This is known as stratification.

safeguard ↙ irregular representation.

More general way → Random subsampling.

↓

Holdout is iterated k times

Average accuracy is considered.

This will be useful if data set split → 50:50

But it is ideal to take more than 50% of data → training.

(22)

So a simple variant of holdout called cross-validation becomes more powerful statistical method.

K-fold cross-validation:

Given data → K folds

D_k where $k = 1 \dots K$.

Training and testing done K times.

Iteration #	Train	Test
1	$D_2 \cup D_3 \cdots D_K$	D_1
2	$D_1 \cup D_3 \cdots D_K$	D_2
⋮		
k	$D_1 \cup D_2 \cdots D_{K-1}$	D_K.

If stratification is used, it is known as stratified K-fold cross-validation.

For each K → average accuracy. The one with least error may be deployed.

K=10 → standard number.

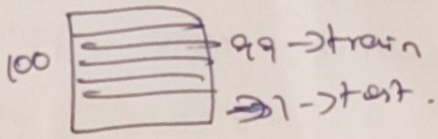
↳ But not generalized

↳ But practically used.

Leave-one-out Cross-validation.

Exceptional case of K-fold cross-validation.

Each iteration → Single sample is left out → test set.



Overall accuracy → average of N iterations

Computation ↑ → b/c whole learning process iterated N times.

Small dataset ✓
Large dataset ✗

No stratification can be applied.

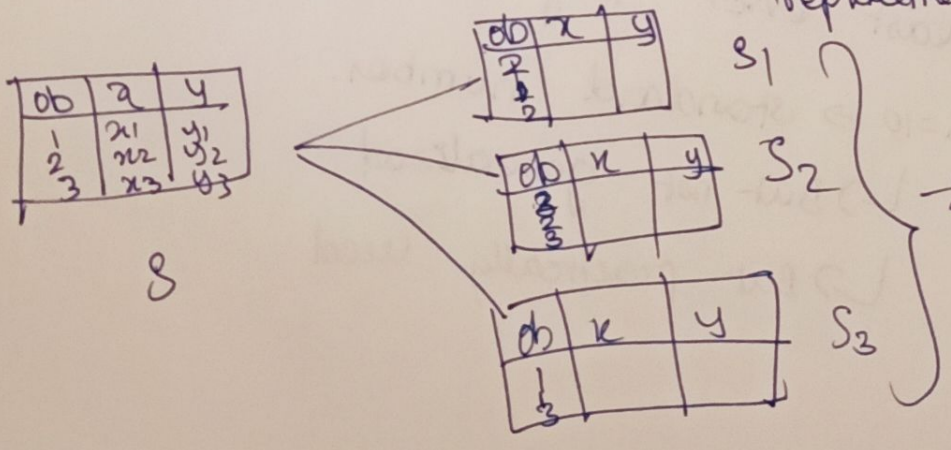
Boot strapping:

↳ Sampling with replacement.

Earlier techniques → samples → never replaced. (or) repeated.

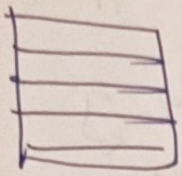
Most learning → repeated samples → impact on outcome.

Aim of bootstrapping → born. Sample dataset by replacement



any bootstrapping technique.

popular — 0.623 bootstrap.



N instance → sample

DS

N instances.

Eg: N = 10.

10 → observation.

In that sample(10) is created with a replacement.

Eg: 1, 3, 4, 3, 2, 5, 6, 5, 10, 4. → train set

Here 3 → 2 times

4 → 2 times.

5 → 2 times.

7, 8, 9. → Test set.

If this is repeated several times on an average 63.2 % of data → training

36.8 % of data → testing.

Probability that particular instance will be

picked → $1/N$

Not picked → $1 - \frac{1}{N}$

During whole cycle → picked → N

not picked $\left[1 - \frac{1}{N}\right]^N$

N → Large → probability approaches $e^{-1} = 0.368$ [base of natural algo].

Thus for large dataset

↳ test set → 36.8% of instance

train set → 63.2% " "

In 10-fold-cross validation → 90%] training.

To compensate, bootstrap combines training error with test error and gives final error estimation as

Error estimate = 0.632 × Error given by test instances

+

0.386 × Error given by train instances.

then whole procedure is repeated and results are averaged.

It works best for small dataset.

Metrics for assessing Regression (Numeric Prediction) Accuracy.

$$f = X \rightarrow Y \; ; \; f(x) = y.$$

Task is to find $h(x)$ that explains underlying data. (i.e) $h(x) \simeq y$ for all samples (x, y).

In statistics → function description of data } → regression.

In multivariate function $f(x)$ is determination of an approximating function $h(x, w)$ ↳ weight vector.

Accuracy → measured on test data set.
↓
not used in model training.

Estimating error in predictor using holdout and random subsampling, cross validation and bootstrap methods.

Several alternative metrics can be used to access the accuracy of numeric prediction.

Mean Square Error: (MSE)

Most commonly used metric.

For MSE → no statistical info are used.

Mean of training set is considered.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - h(w, x^{(i)})\right)^2$$

↑ average squared deviation of predicted value from true value.

Root Mean Square error: (RMSE)

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - h(w, x^{(i)}))^2}$$

↓

More clearly related to individual error.

Sum-of-Error Squares:

$$\text{Sum of Error Square} = \sum_{i=1}^{N}(y^{(i)} - h(w, x^{(i)})^2.$$

(Expected) Mean Square error:

When i/p and o/p are random variables this metrics can be used.

$$\text{(Expected) Mean Square error} - E\left[\sum_{i=1}^{N}(y^{(i)} - h(w, x^{(i)})^2\right].$$

E → statistical expectation operation.

Mean Absolute Error:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y^{(i)} - h(w, x^{(i)})|$$

↓

Average deviation of predicted value from true valu

Suppose model1 → $h_1(w, x)$ → works well for 95% and totally differs on other 5%

model2 → $h_2(w, x)$ small error over full range of data.

model$_2$ will be selected → Amount of variability is considered.

MSE → exaggerate → outliers.

MAE → All errors are treated equally according to their magnitude.

Metrics for assessing classification (Pattern Recognition)

Accuracy:

Regularities in data. & Recognizing patterns

ML → Pattern recog means Assigning label to given i/p pattern.

Different metrics are used to assess the accuracy of a classifier.

Misclassification Error:

Traditional classification algorithm aim to minimize the no of errors made during classification.

$$\text{Misclassification error} = \frac{\text{number of data points for which } (y^{(i)} - \hat{y}^{(i)}) \neq 0}{N}$$

↳ work well for situations where class tuples are more or less evenly distributed.

when classes are imbalanced → poor performance

b/c all misclassifications should not be treated equally.

Log Loss / cross Entropy:

Loss function → It is a cost funct. that evaluates how well algo models the data set.

Prediction
- not Good → high value
- Good → low value.

classification accuracy → count of (or) no predictions equal to actual value.

Log loss → takes uncertainty of prediction based on how much predictor varies from actual label.

Log loss ↓ robustness ↑.

Likelihood function:

Consider → observations with corresponding class labels.

parameters estimated → maximum likelihood approach.

It is computed as follows.

For each data point → compute likelihood. funt

Then likelihood funt → compute combination of likelihood for all the data points.

Estimate set of parameters which maximize the likelihood function.

Let $X = \{x^{(1)} \dots x^{(N)}\} = \{x^{(i)}, i = 1 \dots N\}$.

the probability of predicting y is given by.

$P(y^{(i)} | x^{(i)}, w)$ → parameters governing the distribution.

For whole $x^{(i)}$; $i = 1 \dots N$, $y^{(i)}$ for each $x^{(i)}$ is

$$\prod_{i=1}^{N} P(y^{(i)} | x^{(i)}, w).$$

Likelihood function can be defined by $L(w|x) = \prod_{i=1}^{N} P(y^{(i)} | x^{(i)}, w)$.

Here data → fixed, parameter → vary.

Goal → find W that maximizes L(·).

(26)

(i.e) $W^* = \arg\max_W \{L(W|x)\}$

Often $\log(L(W|x))$ will be tried to increase because it is analytically easier.

∴ W^* that maximize log-likelihood will also maximizes likelihood.

$$\log(L(W|x)) = \log\left[\prod_{i=1}^{N} P(y^{(i)}|x^{(i)}, W)\right]$$

$$= \sum_{i=1}^{N} \log P(y^{(i)}|x^{(i)}, W)$$

For binary classification problem.

$$y^{(i)} \in \{0, 1\}.$$

If $P(y^{(i)} = 1 | x^{(i)}) = \hat{y}^{(i)}$

then $P(y^{(i)} = 0 | x^{(i)}) = 1 - \hat{y}^{(i)}$

Combining these two function.

$$P(y^{(i)} | x^{(i)}, W) = \left(P(y^{(i)} = 1 | x^{(i)})\right)^{y^{(i)}} \left(P(y^{(i)} = 0 | x^{(i)})\right)^{1-y^{(i)}}$$

Here only one of the two classes will be active depending on the value of $y^{(i)} \in \{0, 1\}$.

$$\Rightarrow P(y^{(i)} | x^{(i)}, W) = \left(\hat{y}^{(i)}\right)^{y^{(i)}} \left(1 - \hat{y}^{(i)}\right)^{1-y^{(i)}}$$

this gives.

$$\log(L(w|x)) = \log\left[\prod_{i=1}^{N} (\hat{y}^{(i)})^{y^{(i)}} (1-\hat{y}^{(i)})^{1-y^{(i)}}\right]$$

$$= \sum_{i=1}^{N} \log\left[(\hat{y}^{(i)})^{y^{(i)}} (1-\hat{y}^{(i)})^{1-y^{(i)}}\right]$$

$$= \sum_{i=1}^{N} \log(\hat{y}^{(i)})^{y^{(i)}} + \log(1-\hat{y}^{(i)})^{1-y^{(i)}}$$

$$= \sum_{i=1}^{N} \left[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})\right]$$

↑ this func is maximized to get w.

Average of loss over entire dataset

$$\log \text{loss} = -\frac{1}{N} \sum_{i=1}^{N} \left[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})\right]$$

[log loss is straight forward of log-likelihood fun with maximization transfered to minimization.

$$= \sum_{i=1}^{N} -\left[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})\right]$$

log loss ↓ when predicted probability is high.

Cross entropy :→ From information theory.

log loss & cross entropy → both can be used interchangeably.

Multi class problem.

Class = M.

target output = $\{y_1, y_2 \cdots y_M\} = \{y_{qv}; q_v = 1, 2 \cdots M\}$

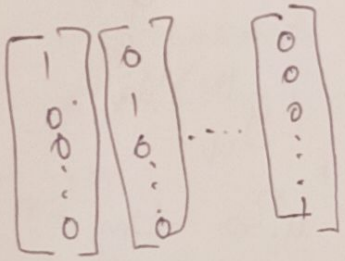
q^{th} Sample → target value $y_{qv}^{(q)}$

predicted value $P(y_{qv}^{(q)} | x^{(q)}) = \hat{y}_{qv}^{(q)}$

Common way of representing target output

↳ one-hot vector.

For M classes the target o/p are represented as M×1 vector.



Sample point that belongs to class 1, has 1 in the 1st row and others are 0.

(i.e) $y_1^{(q)} = 1$, $y_2^{(q)} = 0 \cdots y_m^{(q)} = 0$.

However, predicted output may give probability value for all M outputs, with

$$\sum_{q_v=1}^{M} y_{qv}^{(q)} = 1$$

For k^{th} target class, a good classifier will have a high value (close to 1) for k^{th} o/p and almost negligible value for all other o/p.

The probability of predicting class $y^{(i)}$ is given by

$$P(y^{(i)} | x^{(i)}, w); \; y^{(i)} \in \{y_1^{(i)}, y_2^{(i)} \dots y_M^{(i)}\}.$$

On receiving the whole series, the probability of predicting $y^{(i)}$ for each $x^{(i)}$ is

$$\prod_{i=1}^{N} P(y^{(i)} | x^{(i)}, w).$$

$\therefore$ Likelyhood function $L(w|x) = \prod_{i=1}^{N} P(y^{(i)} | x^{(i)}, w)$

For the given $x^{(i)}$ only one cell in the target o/p will be active.

$\therefore$ Predicted Probability $= P(y^{(i)} | x^{(i)}, w) = \prod_{q=1}^{M} (P(y_q^{(i)} | x^i, w))^{y_q^{(i)}}$

$$= \prod_{q=1}^{M} (\hat{y}_q^{(i)})^{y_q^{(i)}}.$$

For $\hat{y}_1^{(i)} \Rightarrow$ class label $(y_1^{(i)}=1, y_2^{(i)}=0 \dots y_M^{(i)}=0)$ and so on.

$$L(w|x) = \prod_{i=1}^{N} \prod_{q=1}^{M} (\hat{y}_q^{(i)})^{y_q^{(i)}}$$

Log likelihood is given by

$$\log(L(w|x)) = \sum_{i=1}^{N} \sum_{q=1}^{M} y_q^{(i)} \log \hat{y}_q^{(i)}.$$

Maximization is transfered to minimization and average over entire dataset is taken.

$$\text{log loss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{q=1}^{M} y_q^{(i)} \log(\hat{y}_q^{(i)}).$$