Introduction

I) Introduction

II) Linear regression & Decision Trees

III) Instance based learning feature Selection

IV) Probability & Bayes Learning

V) Support Vector Vector Machines

VI) Neural Network

VII) Introduction to computation Learning Theory

VIII) Clustering

Scanned with CamScanner

1> Introduction:

a> Introduction:

b> Different types of learning

c> Hypothesis space, Inductive Bias

d> Evaluation, Training & test set, Cross-Validation

⇒ 1950: Samuel's checker-playing program

⇒ 1960: - Neural Network: Rosenblatt's perceptron:
: Pattern Recognition
: Minsky & Papert prove limitation of perceptron

⇒ 1970: → Symbolic Concept induction
→ Expert system and knowledge acquisition bottleneck.
→ Quinlan's ID3
→ Neural language processing (symbolic)

⇒ 1980: → Advanced decision tree & rule learning
→ Learning & planning & problem solving.
→ Resurgence of neural network.
→ Valiant's PAC learning theory
→ focus on experimental methology.

⇒ 1990: ML & Stats
→ Support vector machines
→ Adaptive agent and web applications
→ Text learning
→ Reinforcement learning
→ Ensembles
→ Bayes net learning

→ New algorithms → neural N/W
→ Deep learning

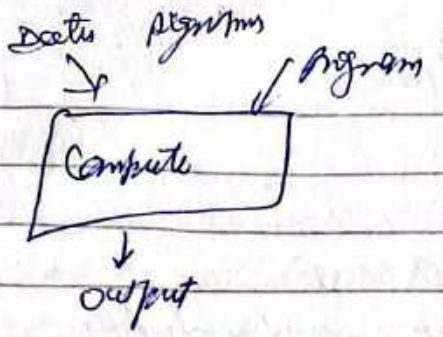
→ New H/W → GPU's

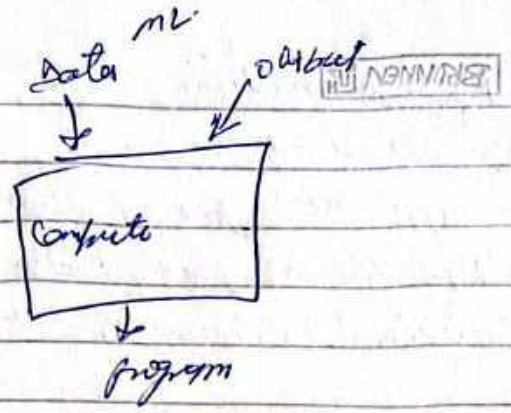
→ Cloud Enabled → availability of Big data

→ 2009 - Google built self driving car

→ 2011 → Watson on jeopardy

→ 2014 → Human vision sup surpassed by ML systems.





Learning – the ability to improve behaviour based on experience

M.L – explore algorithm

↳ learn / build model from data

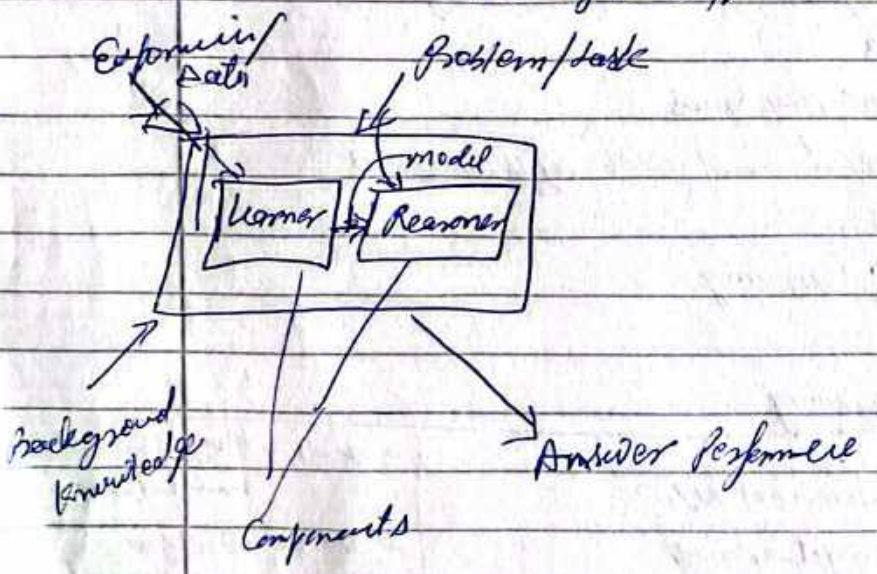
↳ model use for different task (prediction, decision making or solving tasks)

Algorithm definition by Mitchelle

→ A computer program is said to learn from experience E (Data) with respect to some class of tasks T and performance measure P

if its performs on tasks in T as measure by P improves with experience E.

Task → like – Prediction
– classification
Acting in different Environment



Medicine:

Diagnose a disease.

→ input: symptoms, lab measurements, test results, DNA tests ...

→ output: one of set of possible diseases, or "none of the above".

Vision

→ Say what object appear in a image

→ Convert hand-written digits to characters 0 - 9

→ detect where object appear in a image

Robot Control:

Design autonomous mobile robots that learn to navigate from their own experience

NLP

→ detect where entities are mentioned in NL

→ detect what facts are expressed in NL

→ detect if a product/movie review is positive negative, or neutral

Speech recognition.

Machine translation

Financial

Predict if a stock will rise or fall.

Step.

i) choose the training experience (data) — each generation → more difficult to learn. — feature.

ii) choose the target function (target is what learned)

iii) choose how to represent the target function — Class of functions, also called → hypothesis language

iv) choose a learning algorithm to infer the target function

02 Different types of learning

Broad types of learning

⇒ Supervised learning
- x, y (pre-classified training examples
- Given an observation x, what is the best label for y?

in learning we learn data

$\underset{\text{Input}}{X}$; $\underset{\text{Op}}{Y}$ — what is the correspondence Y based on X.

→ labelled data

⇒ Unsupervised learning
- X
- Given a set of x's, cluster or summarize them

There is not labelled data in unsupervised

⇒ Reinforcement learning

Agent who acting in a environment.

⇒ Semi-supervised learning — labelled data & unlabelled data

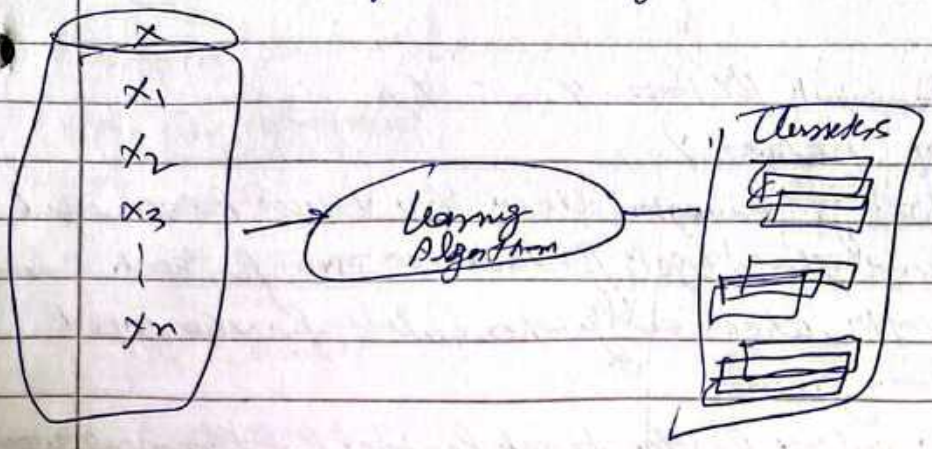
Supervised learning

x	y
input 1	output 1
2	2
3	3
⋮	⋮
Input n	output n

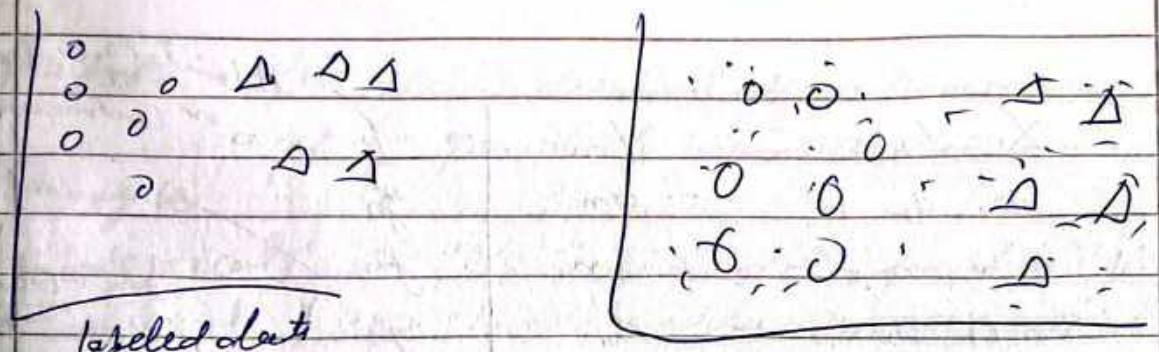
→ Learning Algorithm → model

New Input x → model → output y

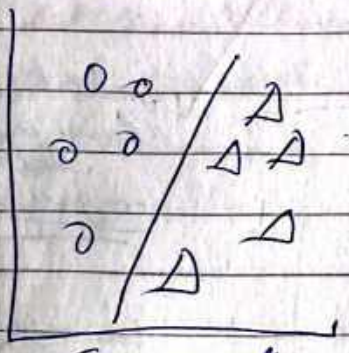
Unsupervised learning



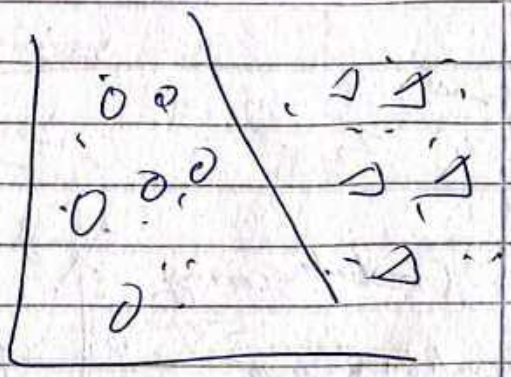
Semi-supervised learning



labeled data

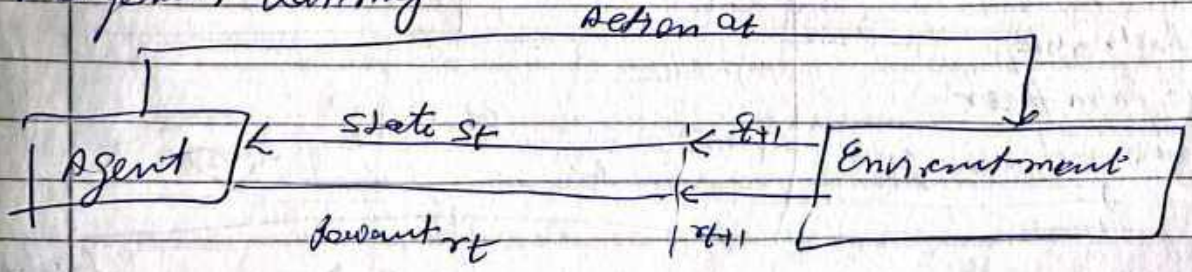


Supervised learning



Semi supervised learning

Reinforcement learning



Supervised Learning

⇒ Given

- a set of input features $x_1, \ldots, x_n$
- A target features Y
- a set of training examples where the values for the input features and the target features are given for each example.
- a new example, where only the values for the input features are given.

⇒ Predict the values for the target features for the new example.

- Classification when Y is discrete
- regression when Y is continuous

Instance	features X_1	X_2	...	X_n	Y
I_1	a_1	a_2	...	a_n	y_1
I_2	b_1	b_2	...	b_n	y_2
I_3	c_1	c_2	...	c_n	y_3
	...				
Test Instance	z_1	z_2	...	z_n	

Y — It could be discrete or continuous valued

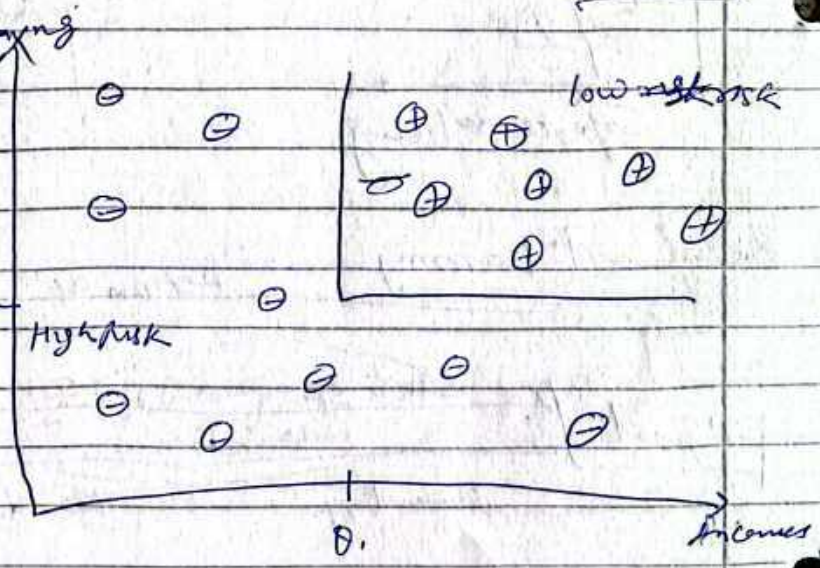
A increasing data results in classification

Predict whether patient have particular disease — regression

Example: Credit scoring

Classification example

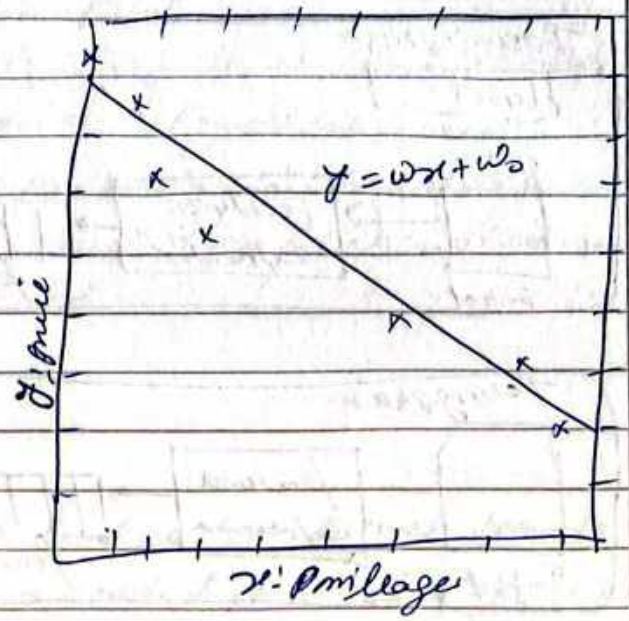
Differentiating b/w low risk and high risk customer from their incomes and savings



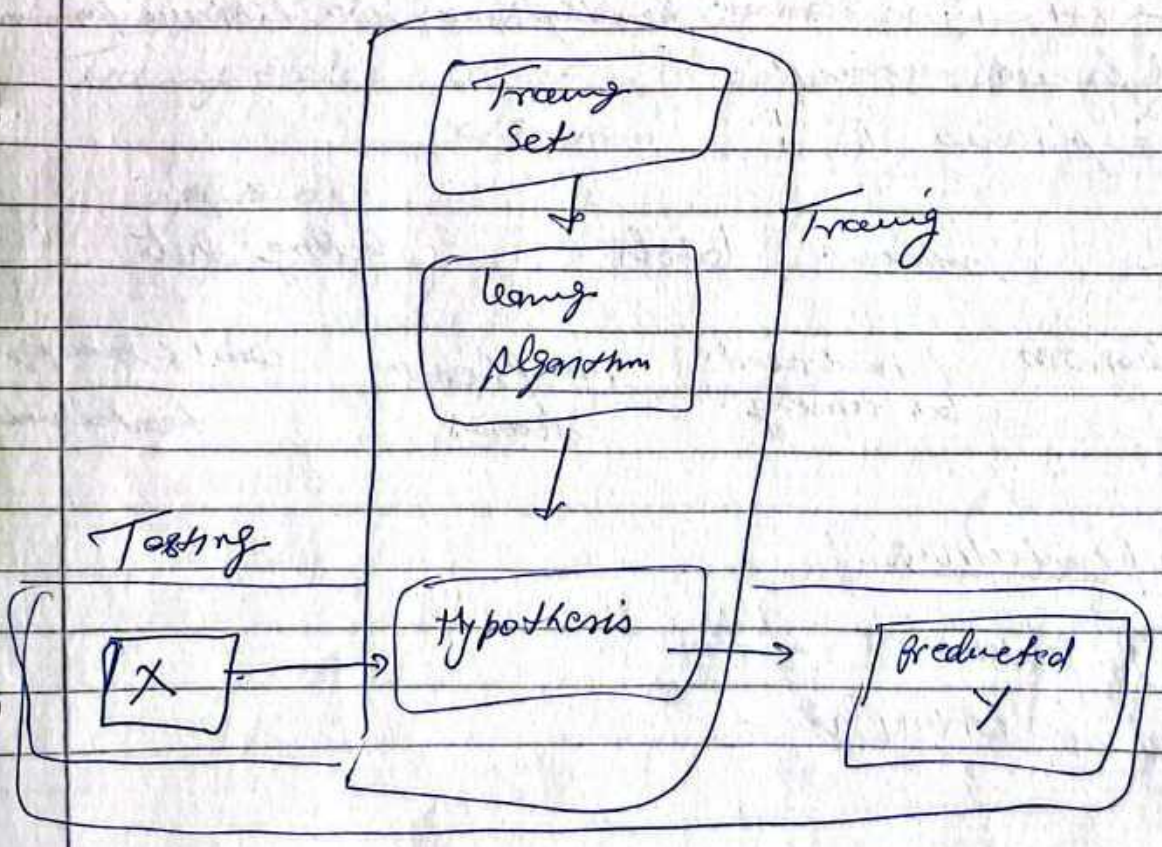
Discriminant: If income > θ_1 AND Saving > θ_2
Then low risk else high risk.

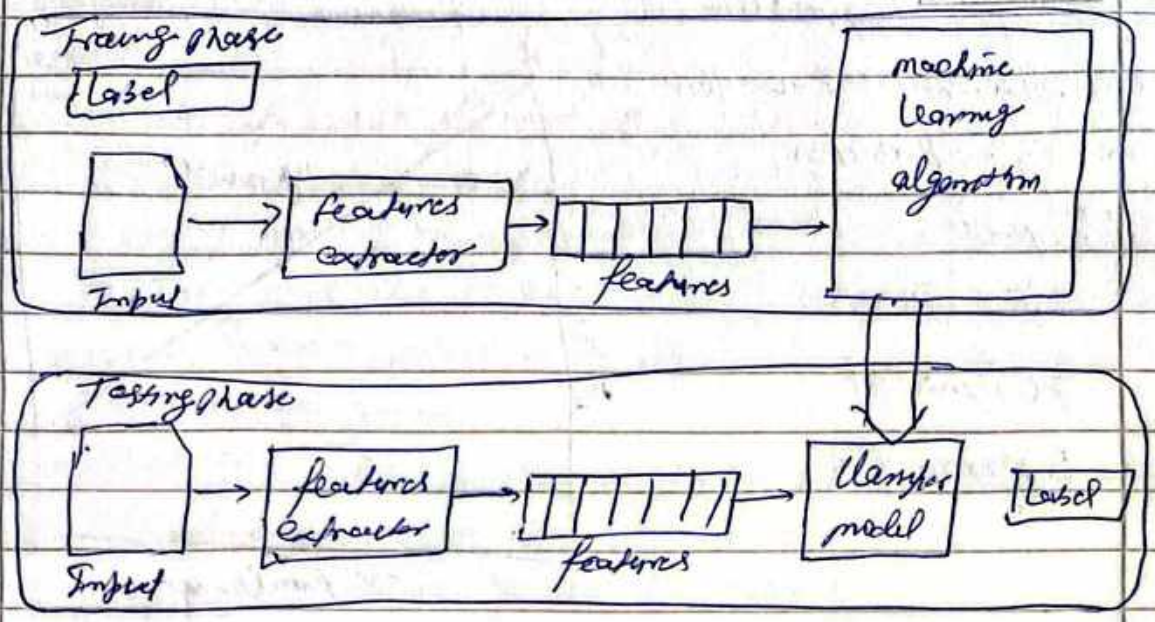
Regression

- Example: price of a used car
  - x: car attributes
  - y: price
  - $y = g(x, \theta)$
  - $g(\,)$ model
  - θ parameters



- Features: often, the individual observations are analyzed into a set of quantifiable properties which are called features. may be
  - categorical (e.g. "A", "B", "AB" or "O", for blood type)
  - ordinal (e.g. "large", "medium", or "small")
  - integer-valued (e.g. the number of words in a text)
  - real-valued (e.g. height)





Classification Learning

- Task T
  - Input: a set of instance $d_1, \ldots, d_n$
    - an instance has a set of features
    - we can represent an instance as a vector $d = \langle x_1, \ldots, x_m \rangle$
  - output: a set of predictions $\hat{y}_1, \ldots, \hat{y}_n$
    - one of a fixed set of constant values:
    - $\{+1, -1\}$ or {Cancer, Healthy} or {rose, hibiscus, jasmine}
- Performance metric P:
- experience E:

Task	Instance	labels	Getting data
medical diagnosis	patient record: lab reading	risk of heart diseases	wait & look for heart diseases

RP representations:

(I) features -

(II) function on class -

Hypothesis space:

- One way to think about a supervised learning machine is as a device that explores a "hypothesis space".
  - Each setting of the parameters in the machine is a different hypothesis about the function that maps input vectors to output vectors.

Terminology

- features: The number of features or distinct traits that can be used to describe each item in a quantitative manner
- feature vector: n-dimensional vector of numerical features that represent some object.
- Instance Space X: Set of all possible objects describable by features.
- Example (x, y): Instance x with label $y = f(x)$
- Concept C: Subset of object from X ($C \subset C$ is unknown)
- Target function f: maps each instance $x \in X$ to target label $y \in Y$
- Example (x, y): instance x with label $y = f(x)$
- Training Data S: Collection of examples observed by learning algorithm.

used to do

03 Hypothesis Space & Inductive Bias

Inductive learning / Prediction

Given examples $(\hat{x}, y)$ or $(\hat{x}, f(\hat{x}))$

y is the o/p attribute.

$\hat{x}$ is for a particular instance x comprises of the values of the different features of that instance

Classification: $f(\hat{x})$ is discrete
Regression: $f(\hat{x})$ is continuous.

Probability estimation $f(\hat{x})$ = probability of $\hat{x}$

Inductive Problem

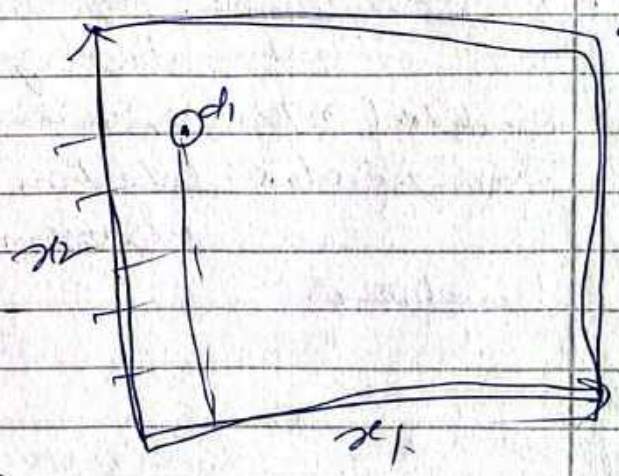
function of features.

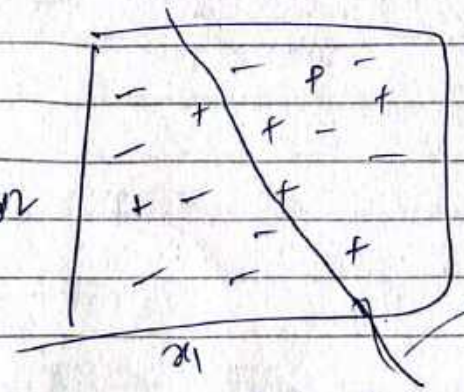
features properties that describes each instance and each instance can be described in a quantitative manner using feature.

more features than just features vector

<u>feature vectors</u>

lets say two features x_1, x_2

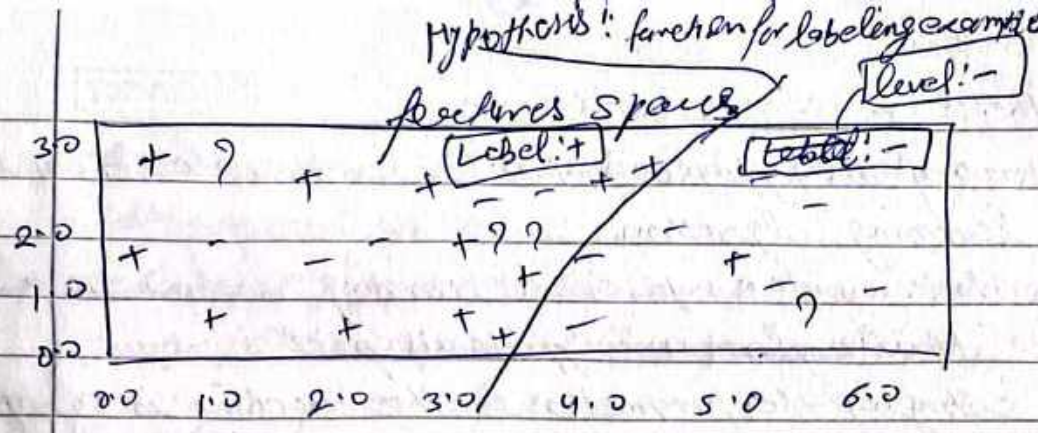




this function ~~and~~ this in inductive learning about.

we want to learn ~~so that~~ feature or function could be separate the

Hypothesis: function for labeling examples.



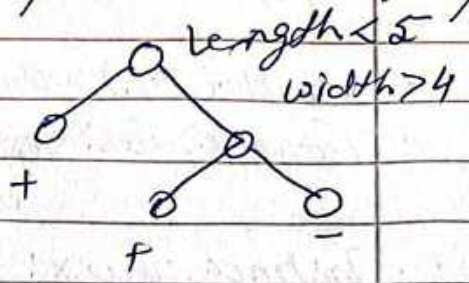
We could have other functions also

Hypothesis space: Set of legal Hypotheses

<u>Representations</u> → function represent in terms of features.

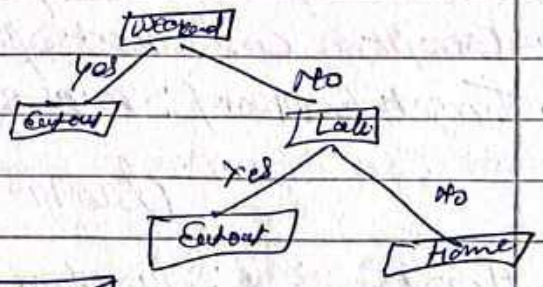
<u>features</u>: We have to define type of function or language of function

features

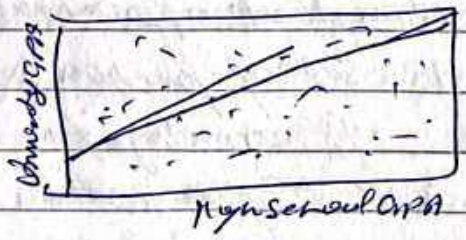


language

① Decision Tree



2. Linear function.



3. multivariate linear function.

4. Single layer perceptron

5. multi-layer neural networks

Hypothesis space:

→ The space of all hypothesis that can, in principle, be output by a learning algorithm.

→ We can think about a supervised learning machine as a device that explores a "hypothesis space".

- Each setting of the parameters in the machine is a different hypothesis about the function that maps input vectors to output vectors.

Hypothesis space:

$\mathcal{H}$ $\quad$ $\underbrace{h \in \mathcal{H}}_{\text{output of learning alg}}$

Terminology:

- Example (x, y): instance x with label $y = f(x)$
- Training data: collection of example observed by learning algorithm.
- Instance space X: set of all possible objects describable by features
- Concept c: subset of objects from X (c is unknown)
- Target function f: maps each instance $x \in X$ to target label $y \in Y$.

Classifier

- Hypothesis h: function that approximates f.
- Hypothesis space $\mathcal{H}$: set of function we allow for approximating f.
- The set of hypothesis that can be produced can be restricted further by specifying a language bias.
- Input training set $S \subseteq X$
- Output: A hypothesis $h \in \mathcal{H}$

Hypothesis spaces.

→ If there are 4(n) input features, there are 2^{16} (2^{2^n}) possible ~~Bot~~ Boolean functions.

→ We can not figure out which one is correct unless we see every possible input-output pair 2^4 (2^n)

Hypothesis language : It reflects an inductive bias of the learner.

Inductive Bias

→ Need to make assumptions.

* Experience alone does not allow us to make conclusions about unseen data instances.

→ Two types of bias:

* Restriction : Limit the hypothesis space.
* Preference : Impose ordering on hypothesis space.

Inductive Learning

→ Inductive learning: Inducing a general function from training examples

* Construct hypothesis h to agree with c on the training examples
* A hypothesis is consistent if it agrees with all training examples.
* A hypothesis said to generalize well if it correctly predicts the value of y for novel example.

→ Inductive learning is an ill posed problem:

* Unless we see all possible examples the data is not sufficient for an inductive learning algorithm to find a unique solution.

Inductive Learning Hypothesis

→ Any hypothesis h found to approximate the target function well over a sufficiently large set of training examples D will also approximate the target function well over other unobserved examples.

Scanned with CamScanner

Underfitting and Overfitting:

Scanned with CamScanner

04- Evaluation and Cross-validation

Experimental Evaluation of Learning Algorithms.

~~Evaluation the~~

→ Evaluating the performance of learning systems is important because

* Learning systems are usually designed to predict the class of "future" unlabeled data points.

→ Typical choices for performance evaluation:

* Error
* accuracy
* Precision/Recall

→ Typical choices for sampling methods:

* Train/Test sets
* k-fold Cross-validation

H, S

function — $h(x) = \hat{y}$

Experiental ~~Evaluation~~ Evaluation.

Error	Evaluation on sample
accuracy	Training set
precision	Test set
	Cross-validation

Evaluating predictions.

→ Suppose we want to make a prediction of a value for target feature on example x:

* y is the observed value of target feature on example x.
* $\hat{y}$ is the predicted value of target feature on example x.
* How is the error measured?

predicted $\hat{y}$ y — actual

if $\hat{y}$ differed from y then is error

→ ~~Absolute~~ Absolute error measured by $|h(x) - y|$ - single training example.

if have n no. of error then

Absolute error $= \frac{1}{n}\sum |(x-y)|$

② Sum of squares error

$$\boxed{\frac{1}{n}\sum_{i=1}^{n}(h(x)-y)^2}$$

* both useful for regression ~~prob~~ problem.

for classification problem ~~&~~ look no. of misclassification.

③ No. of misclassifications

$$\frac{1}{n}\sum_{i=1}^{n}\delta(h(x),y)$$

* confusion matrix

Hypothesis Class \ True Class	POS	NEG
POS	TP	FP
NEG	FN	TN
Sampled	P	N

TP True Positive → TP
True Negative → TN
False Positive → FP
False Negative → FN

$$\text{Accuracy} = \frac{TP+TN}{P+N}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{P}$$

* What data evaluating the learning Algorithm

⇒ Sample error and True error

→ The sample error of hypothesis f with respect to target function c and data sample S is:

$$error_S(f) = \frac{1}{n} \sum_{x \in S} \delta(f(x), c(x))$$

→ The true error (denoted $error_D(f)$) of hypothesis f with respect to target function c and distribution D, is the probability that h will misclassify and instance drawn at random according to D.

$$error_D(f) = Pr_{x \in D}[f(x) \neq c(x)]$$

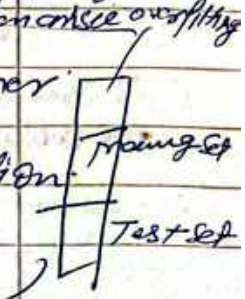
If small test set then chance overfitting

Sample Error

split the examples to train the learner

Training set } drawn from distribution

Test set } to test the learner



True error

If test on small test set then variance will be high.

Difficulties in evaluating hypothesis with limited data

- Bias in the estimate: The sample error is a poor estimator of true error.

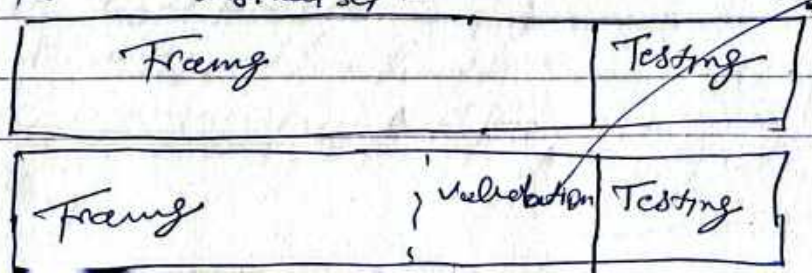
  — ⇒ test the hypothesis on an independent test set.

- We divide the examples into:
  - Training examples that are used to train the learner.
  - Test example that are used to evaluate the learner.
- Variance in the estimate: The smaller the test set, the the greater the expected variance.

Tomalec ~~&~~ training set & test set proper ~~wa~~ evolve scheme Cross validation

validation set

original set



tuning the parameters

One tuning is done then use data for training

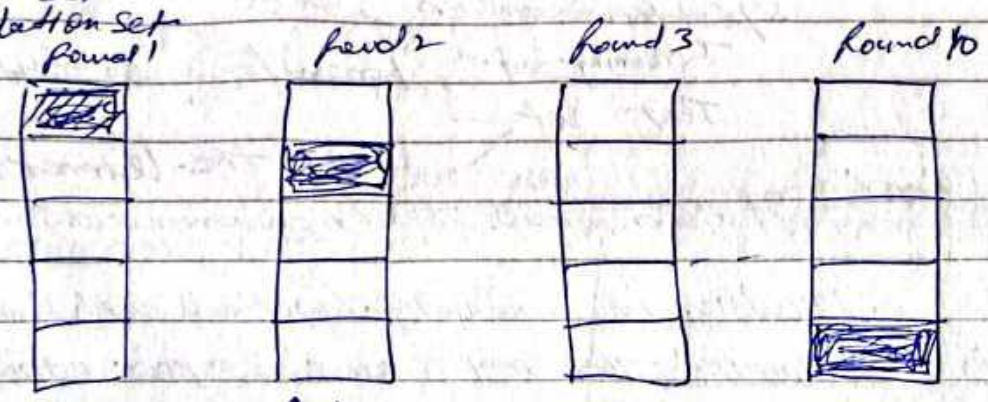
Validation helps to use all the available data.

So when we do validation if splitting data in first part validation fails to use all the available data so we come up with a scheme called cross validation:

K-fold Cross Validation

round i use S_i for testing
$S - S_i$ for training

training set
validation set



	round 1	round 2	round 3	round 10
validation accuracy	93%	90%	91%	95%

Final accuracy = Average (Round 1, Round 2 --)

Trade-off

- In machine learning, there is always a trade-off b/w
  - complex hypothesis that fit the training data well.
  - simple hypothesis that may generalise better.
- As the amount of training data increases, the generalization error decreases.

2

from Bishop's book on machine learning

02. Linear Regression

Regression is supervised learning problem where have given example X, Y and learn a function which predict Y.

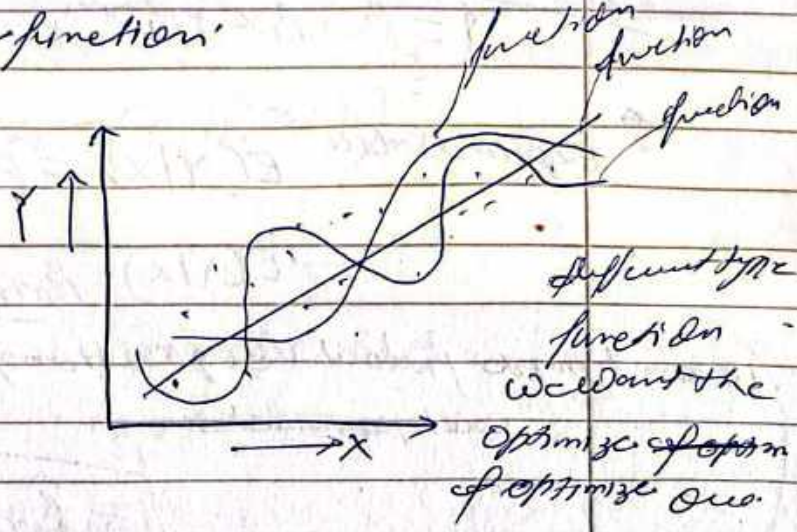
X, Y

$f \quad X \rightarrow Y$ – continuous.

Simplest function is linear function.

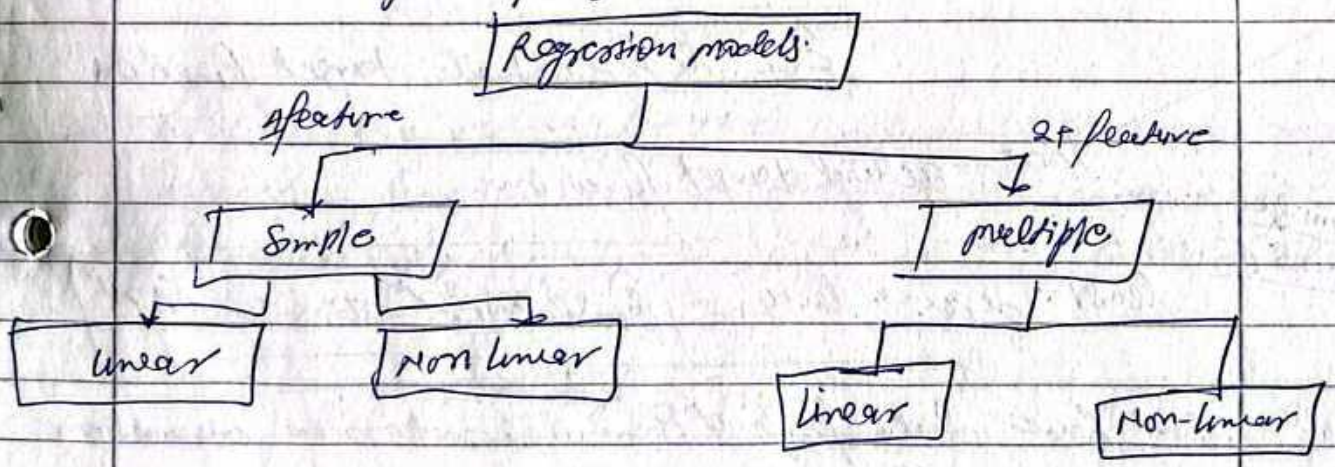
→ when single feature.

Simple regression.



degree polynomial

Types of Regression models



Linear Regression

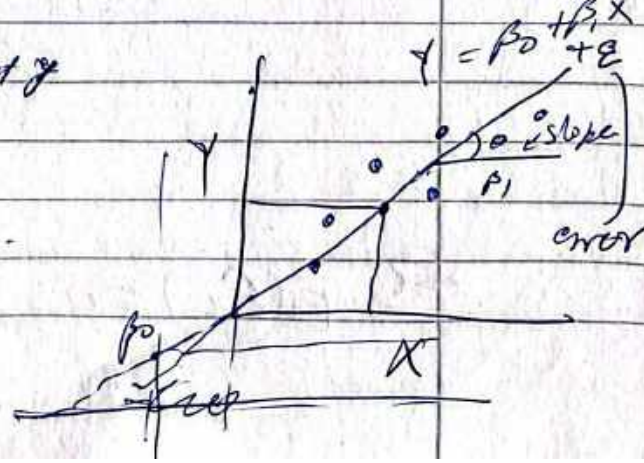
→ Given an input x compute an output y

→ for example:

→ Predict height from age

→ Predict house price from house area.

→ predict distance from wall from sensors.



$\beta_0 \rightarrow$ Y intercept of the population line

$\beta_1 \rightarrow$ slope $\rightarrow$ population-slope

$\varepsilon \rightarrow$ random error

for multiple linear regression $\rightarrow$ we have more of the independent variable.

p-predictor variables

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p + \varepsilon$$

Expected value $E(Y|X) = \beta_0 + \beta_1 x$ - in case of 1 variable

$E(Y|X) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots \beta_p x^p$ when have p variable.

We are trying to find out the equation of line that is estimated value of each parameters.

$$\boxed{= \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \cdots \hat{\beta}_p x^p}$$

Estimate for actual target function

That is actual target function

minimize SSE

least square line $= \boxed{\hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \cdots \hat{\beta}_p x^p}$

We dont have data perfect then we make some assumption

Assumption $Y = \beta_0 + \beta_1 x + \varepsilon$

$$d_1 = y_1 = \beta_0 + \beta_1 x_1 + \varepsilon_1$$
$$d_2 = y_2 = \beta_0 + \beta_1 x_2 + \varepsilon_2$$
$$\vdots$$
$$d_n = y_n = \beta_0 + \beta_1 x_n + \varepsilon_n$$

$E(\varepsilon_i) = 0$

residual error have mean zero

→ $\sigma(\varepsilon_i) = \sigma_\varepsilon$

→ errors are independent.

→ ε_i are normally distributed.

This noise called Gaussian Noise or white noise.

given training point (x_i, y_i)

$$\sum_{d \in D} \left(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)\right)^2$$

How to learn parameters.

parameters β_0, β_1

find a closed form solution.

How do we "learn" parameters

· for the 2-d problem

$$Y = \beta_0 + \beta_1 X$$

→ To find the values for the coefficients which minimize the objective function we take the partial derivates of the objective function (SSE) with respect to the coefficients set these to 0, and solve

$$\boxed{\beta_1 = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2}}$$

closed form solution

$$\boxed{\beta_0 = \frac{\sum y - \beta_1 \sum x}{n}}$$

multiple linear Regression:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

$$h(x) = \sum_{i=0}^{n} \beta_i x_i$$

→ There is a closed form which requires matrix inversion, etc.
→ There are iterative techniques to find weights
* delta rule (also called LMS method) which will update towards the objective of minimize the SSE.

Well know method

delta rule.

LMS - least mmmum slop.

linear Regression

$$h(x) = \sum_{i=0}^{n} \beta_i x_i$$

to learn the parameters $\theta(\beta_i)$?

→ To make $h(x)$ close to y, for the available training examples.
→ define a cost function $J(\theta)$

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

→ find θ that minimize $J(\theta)$

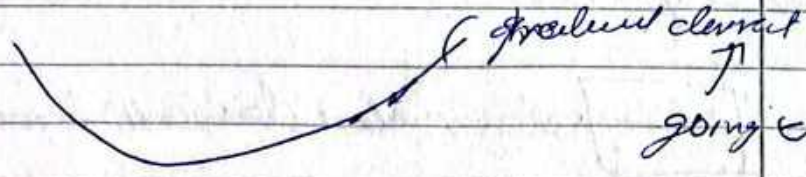
LMS Algorithm

→ start a search algorithm (e.g. gradient descent algorithm) or with initial guess of θ.
→ Repeatedly update θ to make $J(\theta)$ smaller, until it converges to minima

$$\beta_j = \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\theta)$$

→ J is a convex quardratic function, so ~~that~~ has a single global minima. gradient descent eventually converges at the global minima.

→ At each iteration this algorithm takes a step in the direction of steepest descent (-ve direction of gradient)



<u>LMS update Rule.</u>

→ If you have only one training example (x, y)

$$\frac{\partial}{\partial \theta} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2} (h(x) - y)^2$$

$$= 2 \cdot \frac{1}{2} (h(x) - y) \frac{\partial}{\partial \theta_j} (h(x) - y)$$

$$= (h(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y_i \right)$$

$$= (h(x) - y) x_j$$

→ For a single training example, this gives the update

$$\theta_j = \theta_j + \alpha (y^{(i)} - h(x^{(i)})) x_j^{(i)}$$

m training examples.

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h(x^{(i)})) x_j^{(i)}$$

}

Batch Gradient Descent: look at every example on each step.

Stochastic gradient descent.

→ Repeatedly run through the training set.

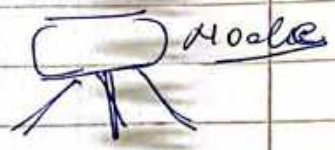
→ Whenever a training point is ~~con~~ encountered update the parameters according to the gradient of the error with respect to that training example only.

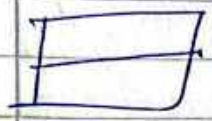
Repeat {
for $i = 1$ to m do
$\theta_j := \theta_j + \alpha (y^{(i)} - h(x^{(i)})) x_j^{(i)}$ (for every j)
} until error for convergence.

06: Introduction to Decision Trees

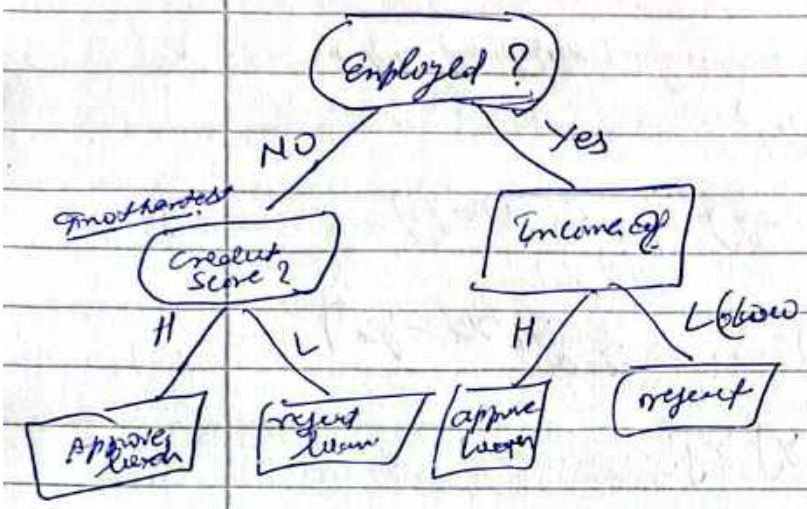
Decision Tree - is a classifier. (Tree structured classifier)

Two Node
- Decision Nodes
→ Leaf Nodes.





Leaf Node → classification tree.



Issues

→ Given some training examples, what decision tree should be generated ?

→ One proposal: prefer the smallest tree that is consistent with the data (bias)

→ possible method:

- search the space of decision trees for the smallest decision tree that fits the data.

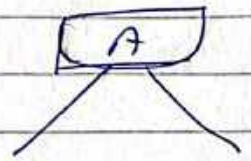
07 Learning Decision Tree

Top-Down Induction of Decision Tree ID3

1. A ← The "best" decision attribute for next node.
2. Assign A as decision attribute for node.
3. For each value of A create new descendant.
4. Sort training examples to leaf node according to the attribute value of the branch.
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

D

$A = A_1 A_2 - A_1$



Principled Criterion

→ Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.

→ Information gain.

* measure how well a given attribute separates the training examples according to their target classification.
* This measure is used to select among the candidate attributes at each step while growing the tree.
* Gain is measure of how much we can reduce uncertainty (value lies between 0, 1)

Entropy → information of purity or measure of uncertainty.

$$\boxed{-\log_2 p} \text{ bits}$$

p → positive examples
$(1-p)$ → negative example

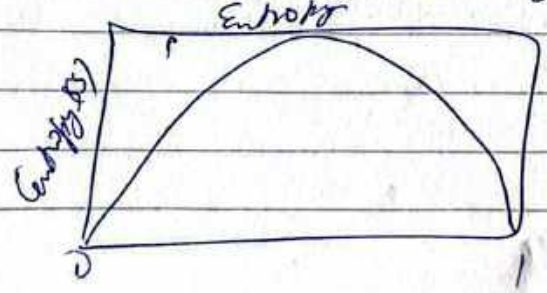
→ A measure for
- uncertainty
- Purity
- information content

→ Information theory: optimal length code assigns $(-\log_2 p)$ bits to message having probability p

→ S is a sample of training examples.
- p_+ is the proportion of positive examples in S
- p_- is the proportion of negative example in S

→ Entropy of S: average optimal number of bits to encode information about certainty / uncertainty about S

$$\text{Entropy}(S) = p_+(-\log_2 p_+) + p_-(-\log_2 p_-) = -p_+\log_2 p_+ - p_-\log_2 p_-$$



Information Gain :

Gain (S, A) : expected reduction in entropy due to partitioning S on attribute A

$$Gain(S,A) = Entropy(S) - \sum_{v \in value(A)} |S_v| / |S| \, Entropy(S_v)$$

ID3 Algorithm

Splitting Rule : GINI Index

. GINI Index

→ measure of node impurity

$$GINI_{node}(Node) = 1 - \sum_{c \in Classes} [p(c)]^2$$

$$GINI_{split}(A) = \sum_{v \in values(A)} \frac{|S_v|}{|S|} GINI(Nv)$$

Scanned with CamScanner

08 overfitting

overfitting:

If learn a decision tree and continues splitting nodes that tree become larger and larger when tree becomes larger if you use the tree for classifying the training example the error on the training example reduce but it may happen that after some ~~time~~ time as grow the tree the true error will ~~increase~~ increase.

A hypothesis h is said to overfit the training data if there is another hypothesis h'.

s.t. h' has more error than h on training data

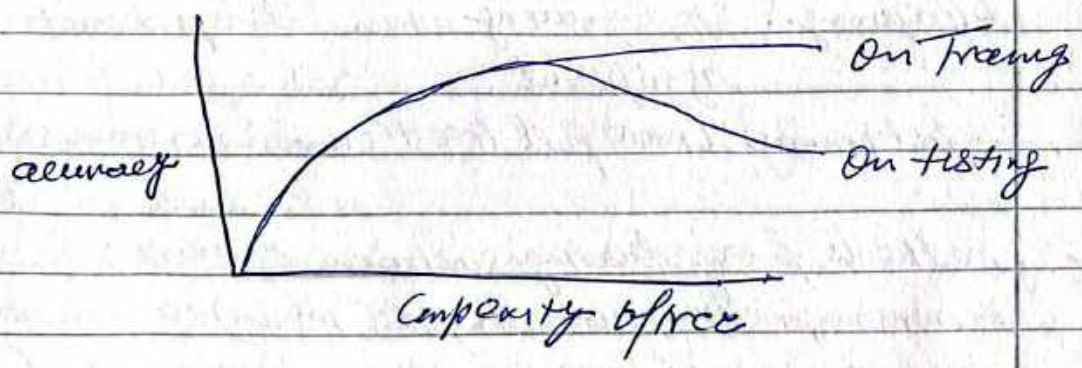
but h' has less error than h on test data

Overfitting

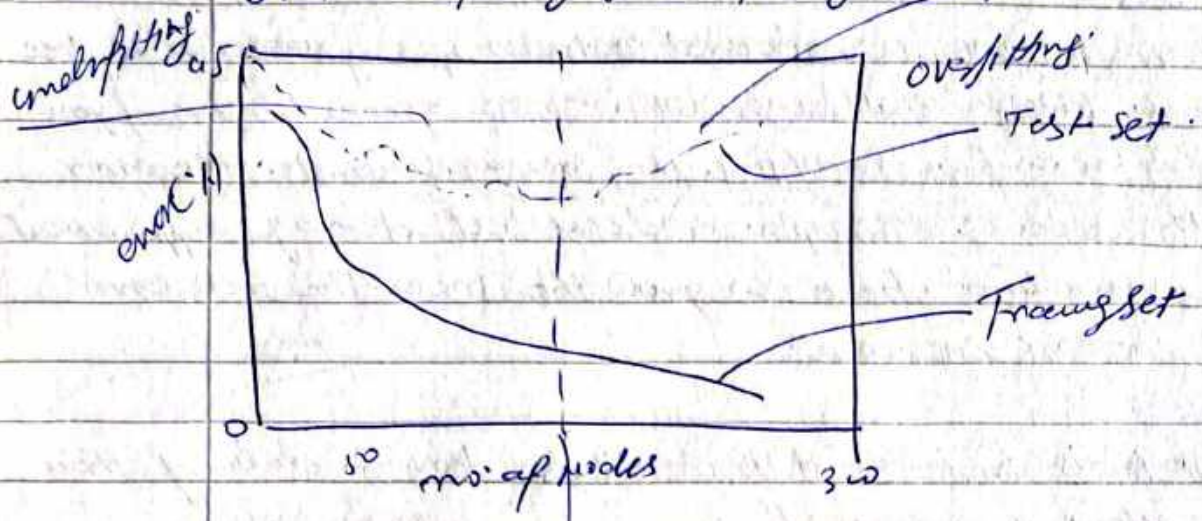
→ learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.

* There may be noise in the training data
* may be based ~~on~~ on insufficient data.

→ A hypothesis h is ~~also~~ said to overfit the training data if there is another hypothesis, h', such that h has smaller error than h', on the training data but h has larger error on the test data than h'.



over underfitting & overfitting



Underfitting: When a model is too simple, both training & test error are large.

middle good fit.

Notes of overfitting:

→ overfitting results in decision trees that are more complex than necessary

→ Training error no longer provides a good estimate of how well the tree will perform on previously unseen records.

Avoid overfitting:

→ How can we avoid overfitting a decision tree?

* Prepruning: Stop growing when data split not statistically significant

* postpruning: Grow full tree then remove nodes.

→ methods for evaluating subtrees to prune:

* minimize description length (MDL):

minimize: size(tree) + size(misclassifications(tree))

* Cross-validation

Model Selection & Generalization

→ Learning is an ill-posed problem; data is not sufficient to find a unique solution.

→ The need for inductive bias, assumptions about $\mathcal{H}$.

→ Generalization: How well a model performs on new data.

→ Overfitting: $\mathcal{H}$ more complex than C or f.

→ Underfitting: $\mathcal{H}$ less complex than C or f.

Triple Trade-off

→ There is a trade-off between three factors:

* Complexity of $\mathcal{H}$, $c(\mathcal{H})$,
* Training set size, N,
* Generalization error, E on new data

Notes on Overfitting

→ Overfitting happens when a model is capturing idiosyncrasies of the data rather than generalities.

* Often caused by too many parameters relative to the amount of training data.
* E.g. an order-N polynomial can interpolate any $N+1$ data points.

Dealing with Overfitting:

→ Use more data

→ Use a training set.

→ Regularization.

→ Be a Bayesian

Penalize large weights in linear Regression.

→ Introduce a penalty term in the loss function.

$$E(\vec{w}) = \frac{1}{2} \sum_{n=0}^{N-1} \{t_n - y(x_n, \vec{w})\}^2$$

Regularized Regression:

1. L2 - Regularization or Ridge Regression:

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \frac{\lambda}{2}\|\vec{w}\|^2$$

2. L1 - Regularization:

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \lambda|\vec{w}|,$$

Scanned with CamScanner

09 Python Exercise on Decision Tree and Linear Regression

Scikits learn machine learning in python

3

12 K-Nearest Neighbour

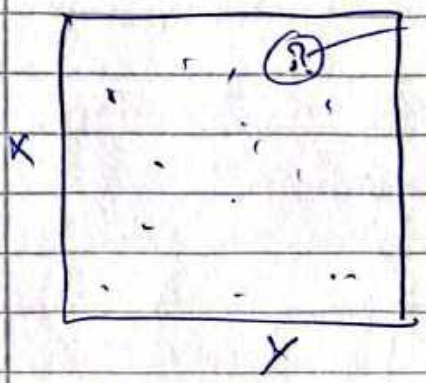
Instant base learning → first in K-Nearest Neighbour Algorithm.

(x_i, y_i)

$(x_i, f(x_i))$

Instant base learning also called lazy algorithm.
~~we don't learn immediately~~

lazy algorithm does not come up with a model approx → ~~other that~~ when it get test instant it uses the stored instant in memory in order to find the possible y.



what is closest instant in terms of the X value

Find Similar Instant

K-Nearest neighbour: → learning algorithm have two faces, training face & testing face.

uses face.

~~Training face~~ phase → normally the model is learned
→ save the examples.

Prediction time: We get the test instants and find the K- training examples so we get test instant.

for test instant we give x_t.
Consider we find training example (x_i, y_i) that is ~~closest~~ closest to x_t.
the we predict ~~the~~ y_1 as output y_t.

3

12 K-Nearest Neighbour

Instant base learning → forthm in K-Nearest Neighbour Algorithm.

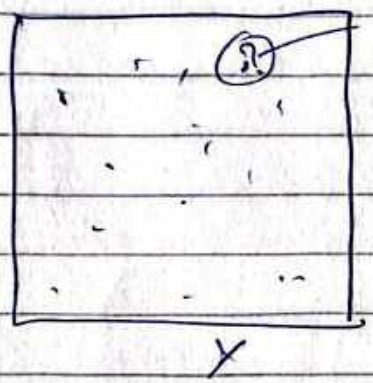
(x_i, y_i)

$(x_i, f(x_i))$

instant base learning also called lazzy algorithm.

~~We dont learn immediately~~

lazzy algorithm does not come up with a model approx rather ~~that~~ when get test instant it uses the stored instant in memory in order to find the possible y.



what is closest instant in terms of the X value

Find Similar Instant

K-Nearest neighbour: → learning algorithm have two faces, training face & testing face

uses face.

~~Training face~~ phase → normally the model is learned
→ save the examples.

Prediction time: We get the test instants and find the K-training examples so we get test instant.

for test instant we give x_t.

Compared we find training example (x_i, y_i) that is ~~closest~~ closest to x_t.

the we predict ~~the~~ y_i as output y_t.

Find the k training ex $\{(x_1, y_1), (x_2, y_2) \ldots (x_k, y_k)\}$ which are closest to x_t.
Predict as the output y_t.

<u>Classification</u>: Predict the majority class from $(y_1, y_2 \ldots y_k)$

<u>Regression</u>: Predict the average of $\{y_1, y_2 \ldots y_k\}$

Inductive Assumption.

→ Similar inputs map to similar outputs.
+ If not true ⇒ learning is impossible
+ If true ⇒ learning reduces to defining "similar".

→ Not all similarities created equal
+ predicting a person's weight may depend on different attributes than predicting their IQ.

~~Basic k-Nearest neighbour classification.~~

<u>Voronoi diagram.</u>

What is the decision boundry?

<u>Improvement:</u>

→ Weighting examples from the neighborhood.
→ measuring "closeness".
→ finding "close" examples in a large training set quickly.

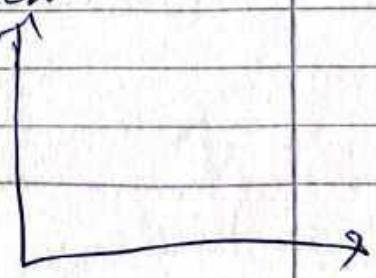
Euclidean Distance

$$Dist(C_1, C_2) = \sqrt{\sum_{i=1}^{N} (attr_i(C_1) - attr_i(C_2))^2}$$

$$Prediction_{test} = ?$$

· Average of k points more reliable when:

→ noise in attributes.
→ noise in class labels.
→ class partially overlap.



Book from Hastie, Tibshirani, Friedman 2001, P418

equal weights to all attributes?

- only if the scale of the attributes are similar and the differences.
- Scale attributes to equal range and variance should be similar.

→ Classes are spherical.

what if not above cases.

then use.

① use larger k

② use weighted distance function

③

Small k. Captures fine structure of problem space better and may be necessary for small training set

large k: less sensitive to noise (particularly class noise) and we get better probability estimates for discrete classes.

④ larger training set allows to use of larger k.

Weighted Euclidean Distance

$$D(C_1, C_2) = \sqrt{\sum_{i=1}^{n} w_i \left(attr_i(C_1) - attr_i(C_2)\right)^2}$$

- large weights ⇒ attribute is more important
- small weights ⇒ attribute is less important
- zero weights ⇒ attribute does not matter

- Weights allow kNN to be effective with axis-parallel elliptical classes.
- Where do weights come from?

Distance-weighted kNN

- Tradeoff b/w small and large k can be difficult.
  → use large k, but more emphasis on nearer neighbors?

$$\text{Prediction}_{test} = \frac{\sum_{i=1}^{k} w_i \times \text{classes}_i}{\sum_{i=1}^{k} w_i} \quad \left(\text{or } \frac{\sum_{i=1}^{k} w_i * \text{values}_i}{\sum_{i=1}^{k} w_i}\right)$$

$$w_k = \frac{1}{\text{Dist}(c_k, c_{test})}$$

locally Weighted Averaging

- Let k = number of training points
- Let weight fall-off rapidly with distance.

$$\text{Prediction}_{test} = \frac{\sum_{i=1}^{k} w_i * \text{classes}_i}{\sum_{i=1}^{k} w_i} \quad \left(\text{or } \frac{\sum_{i=1}^{k} w_i * \text{values}_i}{\sum_{i=1}^{k} w_i}\right)$$

$$w_k = \frac{1}{\text{kernelwidth} \cdot \text{Dist}(c_k, c_{test})}$$

- kernelwidth controls size of neighborhood that has large effect on value (analogous to k)

13 Feature Selection

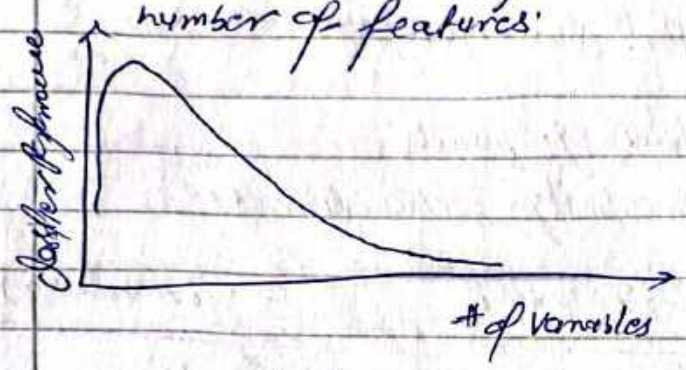
Feature Reduction:

feature contain information about large more features ⇒ { more information
{ better discrimination power

may not allways case.

~~Cor~~ Curse of Dimensionality:

- Number of training examples is fixed
  ⇒ the classifier's performance usually will degrade for a large number of features.



→ Irrevalent features
→ Redundant features } → Can confuse learner
→ limited training examples
→ limited computational resources.

~~Curse~~

To overcome of the curse of dimensionality we use feature reduction,

Two type of feature reduction:

i) feature selection
ii) feature extraction

Feature Selection

$$F = \{x_1, x_2 \ldots x_m\}$$

$$F' \subset F = \{x_1', x_2', \ldots x_m'\}$$

subject

Optimizer

Feature extraction: → It transforms ~~the~~ or project original set of features into a new ~~subset~~ subspace which has ~~no~~ smaller no. of dimensional.

Projection to $M < N$ dimensions.

→ either improve or maintain accuracy.
→ Simplifier Simplify classifier complexity.

2^N possible subsets — can be use

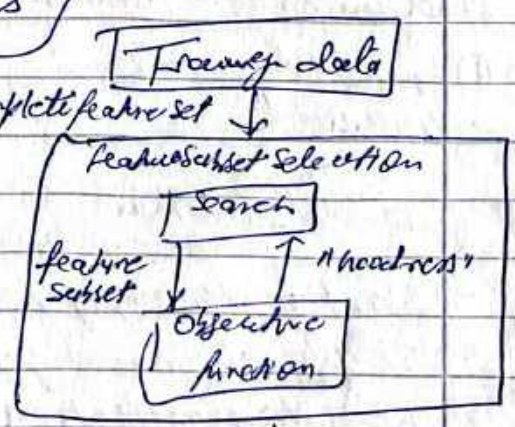
→ optimimum method
→ Heuristic
→ Randomized.

Evaluation →

(I) unsupervised method → Called filter method
(II) supervised method → wrapper method

Subset selection.

- d entire features.
- There are 2^d possible subsets.
- Criteria to decide which subset is the best.
  - Classifier based on these m features has the lowest probability of error of all such classifiers.
- Can't go over all 2^d possibilities.
- Need some heuristics.



feature selection steps.

feature selection is an optimization problem.

- step 1. Search the space of possible feature subsets.
- step 2. Pick the subset that is optimal or near optimal with respect to some objective function.

final feature subset

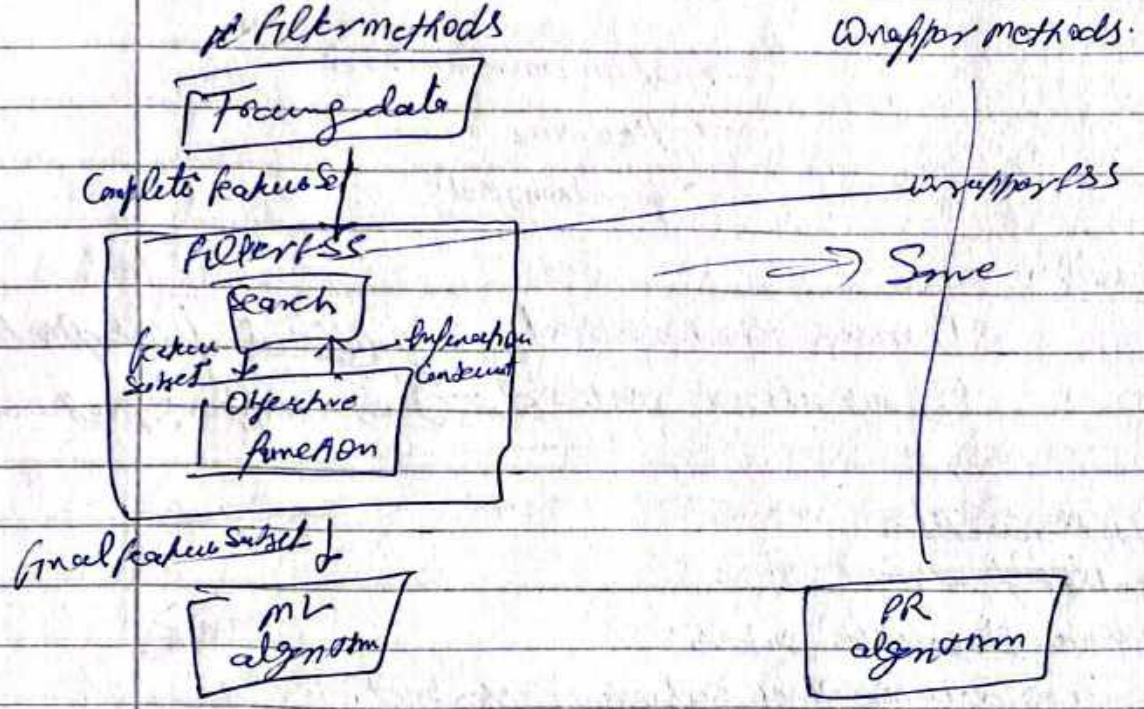
PR Algorithm

(PR- Pattern Recognition)

Evaluating feature subset.

- Supervised (Wrapper method)
  - → Train using selected subset.
  - → Estimate error on validation data set.
- Unsupervised (filter method)
  - → Look at input only
  - → Select the subset that has the most information.

Evaluating Strategies



→ find uncorrelated features.

① forward

→ backward

forward selection.

- Start with empty feature set.
- Try each remaining feature
- → Estimate classification/regression error for adding each specific feature.
- → Select feature that give maximum improvement
- → stop when there is no significant improvement

Backward Selection:

→ Start with full feature set.
→ Try removing features from
→ Drop feature with smallest impact on error.

Feature Selection:

Univariate (look at each feature independently of others)

- Pearson Correlation Coefficient.
- F-Score
- Chi-square
- Signal to noise ratio
- mutual information.
- Etc.

- Rank features by importance.
- Ranking cut-off is determined by user.

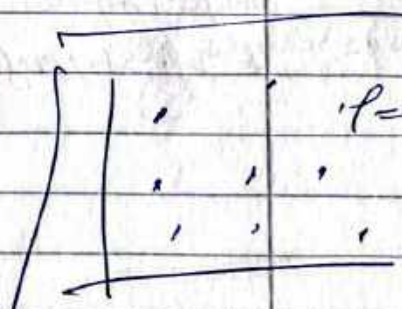
* Univariate methods measure some type of correlation b/w two random variables.

- the label (y_i) and fixed feature (x_{ij} for fixed j)

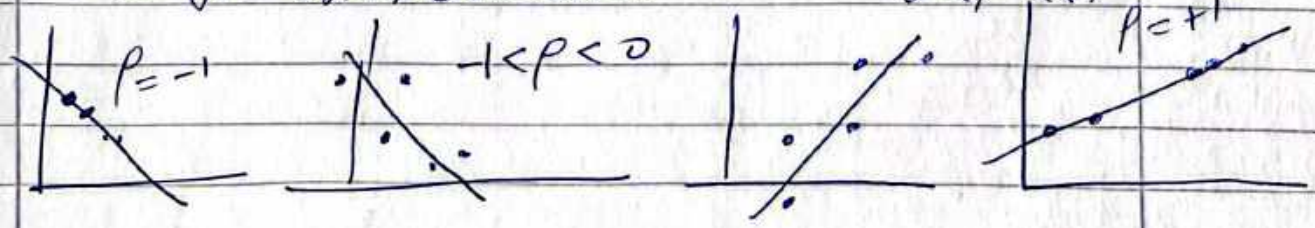
Pearson Correlation Coefficient:

- measure the correlation b/w two variables
- Formula for Pearson correlation =

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$



- The correlation r is b/w +1 and −1.
- +1 means perfect positive correlation.
- −1 in the other direction.



Signal to noise ratio

→ difference in means divided by difference in standard deviation s/w the two classes.

$$S2N(X,Y) = (\mu_X - \mu_Y)/(\sigma_X - \sigma_Y)$$

→ large values indicate a strong correlation.

~~Multi~~ multivariate feature selection.

→ multivariate (Consider all features simultaneously)

→ Consider the vector w for any linear classifier.

→ classification of a point x is given by $w^T x + w_0$.

→ Small entries of w will have little effect on the dot product and therefore those features are less relevant.

→ for example if $w = (10, 101, -9)$ then features 0 and 2 are contributing more to the dot product than feature 1.

* A ranking of features given by this w is 0, 2, 1.

→ The w can be obtained by any of linear classifiers.

→ A variant of this approach is called recursive feature elimination:

* Compute w on all features.
* Remove feature with smallest w_i
* Recompute w on reduced data.
* If stopping criterion not met then go to step 2

14 Feature Extraction

We have n dimensional feature — projection matrix

$$\underset{X}{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}} \rightarrow \underset{Z}{\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{pmatrix}} \rightarrow f \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

- new feature

$m \times n$

original feature

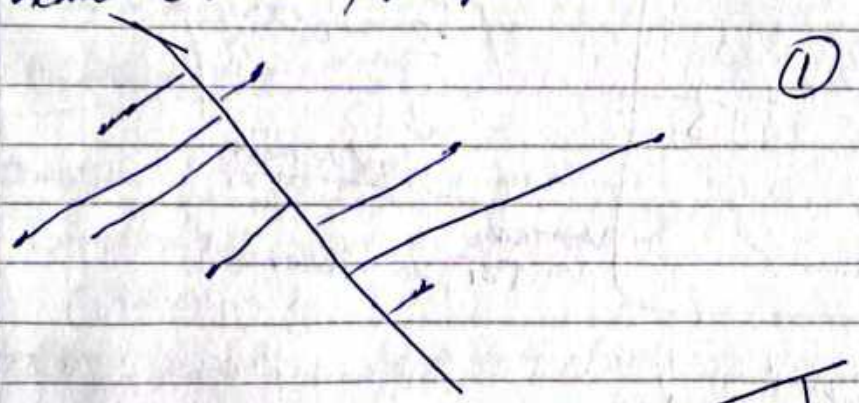
A need → find a projection matrix ω

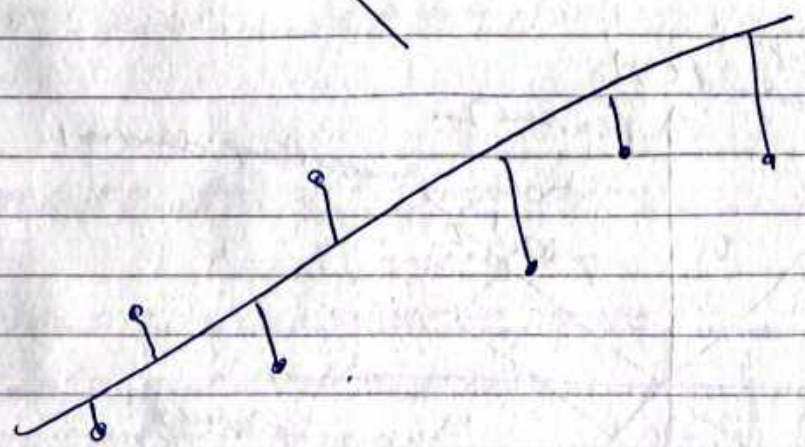
s.t. $\bar{z} = \omega^T \bar{x}$

we expect new features is correlated and cannot be reduced further.

→ large variance

Geometric interpretation of principal component (PC)





larger variance

we would go with ②

Algebric ~~definition~~ definition of PCs

Given a sample of p observations on a vector of N variables

$$\{x_1, x_2, \ldots, x_p\} \in R^N$$

define the first principal component of the sample by the linear transformation:

$$z_1 = w_1^T x_j = \sum_{i=1}^{N} w_{i1} x_{ij}, \quad j = 1, 2, \ldots, p.$$

Where the vector

$$w_1 = (w_{11}, w_{21} \ldots w_{N1})$$

$$w_j = (x_{1j}, x_{2j} \ldots x_{Nj})$$

is chosen such that $Var[z_1]$ is maximum.

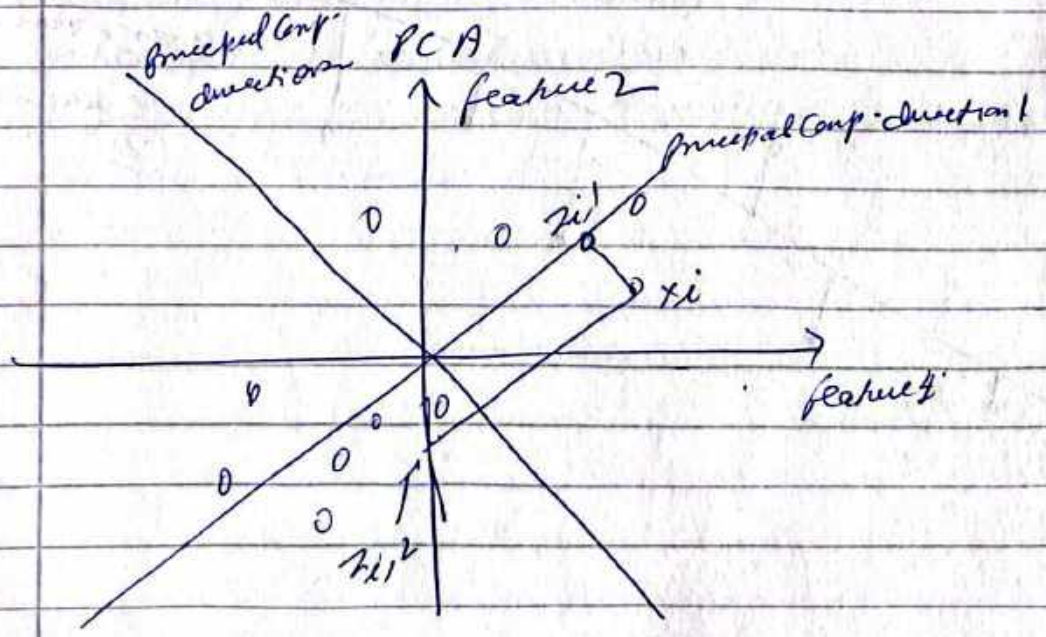
Principal Components

$$\boxed{\begin{array}{l} x_1 \ldots \{ \ldots \} \\ x_2 \ldots \{ \ldots \} \\ \vdots \\ x_p \ldots \{ \ldots \} \end{array}}$$

$$\boxed{\bar{z} = w^T \bar{x}}$$

want to map

~~variance~~ $Var[z_1]$ is maximum.



PCA

1. maximize total variance.
2. minimize correlation.

Choose $m \leq N$ orthogonal directions which maximize total variance.

For data $Cov(x) = \Sigma$

→ Select m largest eigenvectors

PCA for image compression

$p=1$, $p=2$, $p=4$, $p=8$, $p=16$, $p=32$, $p=64$, $p=100$

Q Is PCA a good criterion for classification?

1. Data variation determines the projection direction.
2. What's missing?

* Class information.

What is a good projection?

→ Similarly, what is a good criterion?

* Separating different classes.

* means & scatter after projection.

Good projection.

→ means are as far away as possible.

→ scatter is as small as possible.

→ Fisher Linear Discriminant

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

15. Collaborating Filtering.

Recommender System:

1. Item Recommendation
2. Rating Prediction

U: set of users

S: set of items. $p: U \times S \rightarrow R$

learn p from data
use it predict the utility value of the item to each user.

There are broad two type of recommendation systems.
1) Content based Recommendation System
1) Collaborating filtering " "

→ Content Based:

— based on content similarity.

Collaborative filtering:

→ find a similar users.

Neighborhood formation phase

→ Let the record (or profile) of the target user be u (represented as a vector), and the record of another user be v (UGT)

→ The similarity b/w the target user, u and a neighbour, v, can be calculated using the Pearson's Correlation coefficient.

$$Sim(u,v) = \frac{\sum_{i \in C} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in C} (r_{u,i} - \bar{r}_u)} \sqrt{\sum_{i \in C} (r_{v,i} - \bar{r}_v)^2}}$$

Scanned with CamScanner

Recommendation phase:

→ Use the following formula to compute the rating prediction of item i for target user u.

→

$$P(u,i) = \bar{r}_u + \frac{\sum_{v \in V} Sim(u,v) \times (r_{v,i} - \bar{r}_v)}{\sum_{v \in V} |Sim(u,v)|}$$

where V is the set of k similar users, r_v is the rating of user v given to item i,

Item-based CF

→ The item-based approach works by comparing items based on their pattern of ratings across users. The similarity of items i & j is computed as follows:

$$Sim(i,j) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u)(\bar{r}_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}}$$

Recommendation phase

→ After computing the similarity b/w items we select a set of k most similar items to the target item and generate a predicted value of user's rating:

$$P(u,i) = \frac{\sum_{j \in J} r_{u,j} \times Sim(i,j)}{\sum_{j \in J} Sim(i,j)}$$

where J is the set of k similar items.

16 Python Exercise of KMP & LCA

Scanned with CamScanner

(4)

10 Bayesian learning

Tom Michell Book

Probability: Modeling Concepts.

Bayesian probability → Partial Beliefs.

Bayesian Estimation → Calculate Validity of a probability.

(i) Prior estimate

(ii) New relevant evidence

Bayes Theorem:

$$P(h|D) = \frac{P(D|h) \cdot P(h)}{P(D)}$$ - Prior Probability

$$P(hD) = \boxed{P(h) \cdot P(D|h)}$$

$$= P(Dh) = \boxed{P(D) \cdot P(h|D)}$$

ex

$P(Cancer) = 0.008$

$P(\neg Cancer) = 1 - 0.008 = 0.992$

$P(+|Cancer) = 0.98$ $\quad P(\neg|Cancer) = 1 - 0.98 = 0.02$

$P(+|\neg Cancer) = 1 - 0.97 = 0.03$ $\quad P(\neg|\neg Cancer) = 0.97$

~~P(C+~~

$$P(Cancer|+) = \frac{P(+|Cancer) \cdot P(Cancer)}{P(D)} = \frac{(.98) \cdot (0.02)}{\cancel{P(D)}}$$

$$P(\neg Cancer) = \frac{P(\neg|Cancer) \cdot P(\neg Cancer)}{P(D)} = \frac{(0.03) \cdot (0.992)}{\cancel{P(D)}}$$

=

Maximum A Posteriori (MAP) Hypothesis

$$P(h|D) = \frac{P(D|h) P(h)}{P(D)}$$

The Goal of Bayesian learning: The most probable hypothesis given the training data (maximum A Posterior hypothesis)

Scanned with CamScanner

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

$$= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)}$$

$$= \arg\max_{h \in H} P(D|h)P(h)$$

$$h_{ML} = \arg\max_{h \in H} P(h|D)$$

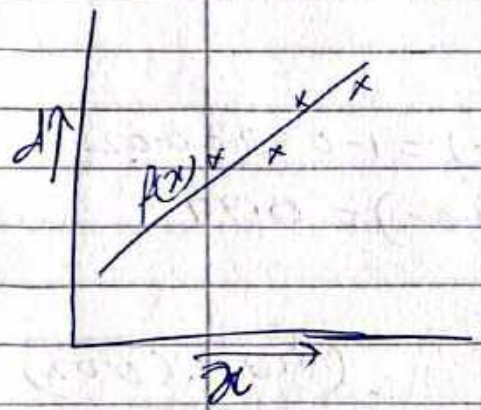
* learn a real-valued function

$f \rightarrow$ target function

ML - maximum likelihood

(x_i, d_i) $\quad d_i = f(x_i) + \varepsilon_i$ $\quad \left\{ \sigma^2 \text{ invariance corresponding to } \varepsilon \text{ error term} \right\}$

$d_i \sim N(f(x_i), \sigma^2)$



$$h_{ML} = \arg\max_h P(D|h)$$

$$= \arg\max_h \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d - h(x)}{\sigma}\right)^2}$$

$$= \arg\max_h \sum_{i=1}^{m} -\frac{1}{2}\ln(2\pi\sigma^2) - \frac{1}{2}\left(\frac{d - h(x_i)}{\sigma}\right)^2$$

this part does not play role b/c its constant

$$\boxed{= \arg\min_h \sum_{i=1}^{m} (d_i - h(x_i))^2}$$

Bayes optimal classifier:-

Training data (or learn) h_{map} — most probable classification

h_1, h_2, h_3

$$\begin{cases} P(h_1/D) = 0.4 \\ P(h_2/D) = 0.3 \\ P(h_3/D) = 0.3 \end{cases}$$

$$\boxed{\underset{v_j \in V}{\arg\max} = \sum_{h \in H} P(v_j/h) \cdot P(h_i/D)}$$

Bayes optimal classifier

Gibbs Sampling:-

Choose a hypothesis randomly according to $P(h/D)$ use it to classify the new instance

19 Naive Bayes

$$P(h|D) = \frac{P(D|h)\cdot P(h)}{P(D)}$$

$$P(Y|\bar{X}) \propto P(\bar{X}|Y)\cdot P(Y)$$

$$= \underbrace{P(X_1 X_2 \cdots X_n|Y)}_{\text{Joint probability}}\cdot P(Y)$$

Naive Bayes:

$$= P(X_1|Y)\cdot P(X_2|X_1 Y)\cdots P(X_n|X_1\cdots X_{n-1} Y)\cdot P(Y)$$

$$= P(X_1|Y)\cdot P(X_2|Y)\cdot P(X_n|Y)\cdot P(Y) \quad \text{— Assumption}$$

$$\boxed{P(X_i|X_j Y) = P(X_i|Y)}$$

$$Y^{new} = \underset{y_k}{\arg\max}\ \boxed{\prod_{i=1}^{d} P(X_i^{new}|Y=y_k)}\cdot \underbrace{P(Y=y_k)}_{\text{prior probability}}$$

loc have discrete values of X

$$P(X_i = x_{ij}|Y=y_k)$$

$$P(Y=y_k) \rightarrow (n-1)$$

$\boxed{2\times n \rightarrow \text{estimates}}$

Classify (X^{new})

$$y^{new} \leftarrow \underset{y_k}{\arg\max}\ P(Y=y_k)\prod_i P(X_i^{new}|Y=y_k)$$

$$y^{new} \leftarrow \underset{y_k}{\arg\max}\ \pi_k \prod_i \theta_{ijk}$$

→ possibilities must sum to 1, so need estimate only n-1 parameters

Maximum likelihood estimates (MLE's):

$$\hat{\pi}_k = \hat{P}(Y=y_k) = \frac{\#\{D \in Y=y_k\}}{|D|}$$

number of items in set D for which $Y=y_k$

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij}|Y=y_k) = \frac{\#\{D \in X_i = x_{ij} \wedge Y=y_k\}}{\#\{D \in Y=y_k\}}$$

Scanned with CamScanner

20 Bayesian Network

Bayes Network

Graphical Model

Scanned with CamScanner

21 Python Exercising of Naive Bayes

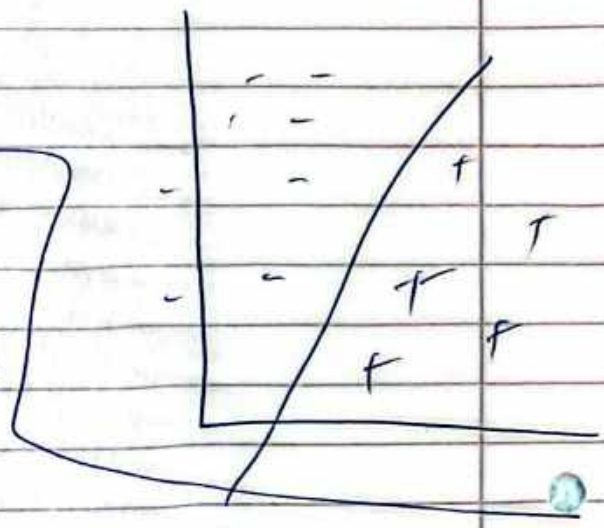
Scanned with CamScanner

(2)

23 Logistic Regression

Linear Regression → $h(x) = \sum_{i=0}^{n} \beta_i x_i$

Classification problem



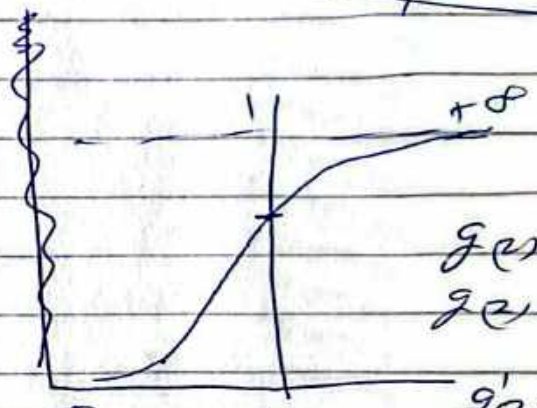
Logistic Regression

Sigmoid function or logistic Regression

$$\boxed{g(z) = \frac{1}{1+e^{-z}}}$$

or

$$\boxed{\sigma(z) = \frac{1}{1+e^{-z}}}$$



$g(z) \to 1$ as $z \to \infty$

$g(z) \to 0$ as $z \to -\infty$

$$h_\beta(x) = g(\beta^T x)$$

$$= \frac{1}{1+e^{-\beta^T x}}$$

$$g'(z) = \frac{1}{(1+e^{-z})^2 \times e^{-z}}$$

$$= \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right)$$

logistic function → $g'(z) = g(z)(1-g(z))$

$$P(y|x) = h(x)^y (1-h(x))^{1-y}$$

$$h(x) = \frac{1}{1+e^{-\beta^T x}}$$

learning by using gradient ascent method.

need to learn conditional probability.

$h_\beta(x \cdot \beta)$ is our estimate of $P(Y|X)$

We have to learn optimal value of β.

→ likelyhood of β

probability

$$L(\beta) = p(Y|X;\beta)$$

product of

$$= \prod_{i=1}^{m} p(y_i|x_i\beta)$$

$$= \prod_{i=1}^{m} (h(x_i))^{y_i} - (1-h(x_i))^{1-y}$$

$$\boxed{l(\beta) = \sum_{i=1}^{m} y_i \log(h(x_i)) + 1 - y_i \log(1-h(x_i))}$$

log likelyhood

We have to maximize this then have to use gradient descent

We have to use derivative

$$\boxed{\beta = \beta + \alpha \nabla_\beta l(\beta)}$$

$$\frac{\partial}{\partial\beta}(l(\beta)) = \left(y \frac{1}{g(\beta^T x)}\right) - (1-y)\frac{1}{1-g(\beta^T x)} \frac{\partial}{\partial\beta_j} g(\beta^T x)$$

$$= \left(y \frac{1}{g(\beta^T x)}\right) - (1-y)\frac{1}{1-g(\beta^T x)} (1-g(\beta^T x)) \frac{\partial}{\partial\beta} \beta^T x$$

$$= (y(1-g(\beta^T x)) - (1-y)g(\beta^T x))x_j$$

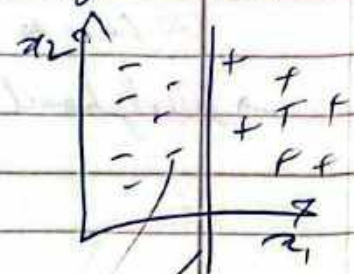
$$\boxed{= (y - h_\beta(x))x_j}$$

$$\boxed{\beta_j = \beta_j + \alpha(y - h_\beta(x))x_j}$$

Stochastic gradient ascent for logistic regression.

24 Introduction on Support vector Machine

If have clever way for overfitting, & can work with relatively larger no of features without require too much computation.



decision surface

less confident of feature near to decision surface.

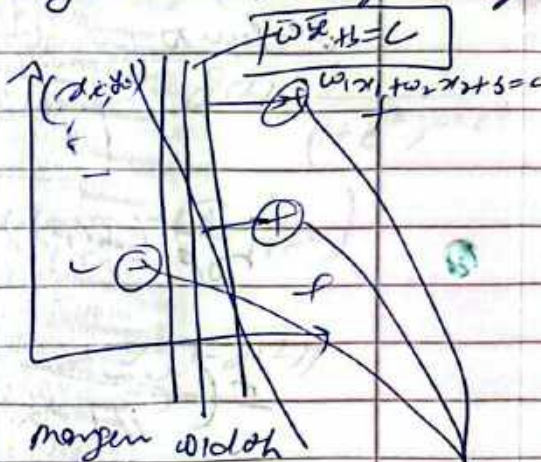
logistic regression:

$$P(y=1|x) = h(x) = \sigma(\beta^T x)$$

Predict 1 when $h(x) > 0.5$

0 " $h(x) < 0.5$

⇒ If have large no of features you to considered Bayesian learning. Bayesian learning can look all possible classifier weight then by posterior probability and combine them to answer. but it is computationally intractable. we want good alternative to Bayesian learning which is computationally attractive. So we use SVM.



Closest to surface points are called support vectors.

What we call functional margin?

functional margin: distance of (x_i, y_i) from (w, b)

the point with respect to surface

(x_i, y_i) w.r.t. (w, b)

$\gamma^i = y_i(w^T x_i + b)$

are normal distance for surface

(x_i, y_i)

(x_i, y_i) margin large means have large confidence

(w_1, w_2)

$2x + 3y + 1 = 0$
$4x + 6y + 2 = 0$
$20x + 30y + 10 = 0$

then of scaled the equation hyperplanes same but functional margin becomes low

$w^T x + b = 0$

normalized distance.

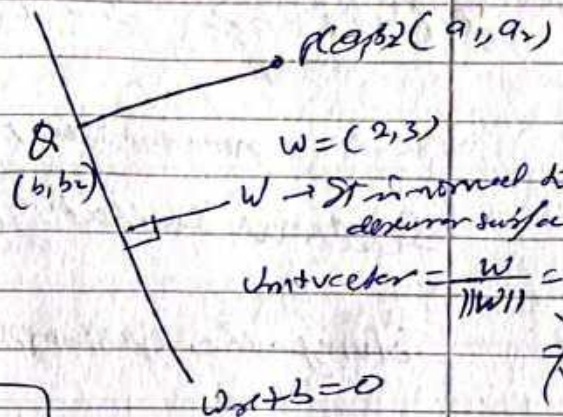
for equal margin of set of point.

$$S = \{(x_1, y_1) \ldots (x_m, y_m)\}$$

$$\gamma = \min_{i=1}^{m} \gamma^i \longrightarrow \text{geometric margin}$$

definitely functional margin suffers from these drawbacks so we cannot use the geometric margin

Geometric margin:



$$P = Q + \gamma \frac{w}{\|w\|}$$

why what

$$\cancel{(b_1, b_2) = (}$$

$$\boxed{(a_1, a_2) = (b_1, b_2) + \gamma \frac{w}{\|w\|}}$$

$$w^T\left((a_1, a_2) - \gamma \frac{w}{\|w\|}\right) + b = 0$$

$$\Rightarrow \gamma = \frac{w^T(a_1, a_2) + b}{\|w\|}$$

$$= \frac{w^T}{\|w\|}(a_1, a_2) + \frac{b}{\|w\|}$$

$$\boxed{\gamma = y \cdot \frac{w}{\|w\|}(a_1, a_2) + \frac{b}{\|w\|}}$$

we scale w when $\|w\| = 1$

$$\boxed{\gamma = y(w^T(a_1, a_2) + b)}$$

Scanned with CamScanner

(w, b)

$\frac{\gamma}{||w||}$ is the geometric margin

$wx_i + b \geq \gamma$ for (+) points

$wx_j + b \leq \gamma$ for (−) points

$\frac{\gamma}{||w||}$ we actually do maximize

maximize $\frac{1}{||w||} \equiv$ minimize $||w||$
$= w \cdot w$

$y_i(wx_i + b) \geq \gamma$ for $i = 1, 2 \ldots m$

normalize so that $\gamma = 1$

Scale w so that geometric margin is $\frac{1}{||w||}$

Solving the optimization problem:-

$$\text{minimize } \frac{1}{2}||w||^2$$

$$\text{s.t. } y_i(w^T x_i + b) \geq 1$$

→ optimization problem with convex quadratic objectives and linear constraints.

→ Can be solved using Q.P.

→ lagrange duality to get the optimization problem's dual form,

* allows us to use kernels to get optimal margin classifiers to work efficiently in very high dimensional spaces.

* allows us to derive an efficient algorithm for solving the above optimization problem that will typically do much better than generic Q.P software.

25. SVM: The Dual formulation

Lagrangian Duality in brief.

The primal problem $\min_\omega f(\omega)$

$$\text{s.t. } g_i(\omega) \le 0, \quad i=1,\dots,k$$
$$h_i(\omega) = 0, \quad i=1,\dots,l$$

The generalized lagrangian:

$$L(\omega,\alpha,\beta) = f(\omega) + \sum_{i=1}^{k} \alpha_i g_i(\omega) + \sum_{i=1}^{l} \beta_i h_i(\omega)$$

the α's ($\alpha_i \ge 0$) and β's are called the Lagrange multipliers

$$\max_{\alpha,\beta,\alpha\ge 0} L(\omega,\alpha,\beta) = \begin{cases} f(\omega) & \text{if } \omega \text{ satisfies primal constraints} \\ 0 & \text{otherwise} \end{cases}$$

A rewritten primal

$$\min_\omega \max_{\alpha,\beta,\alpha_i\ge 0} L(\omega,\alpha,\beta)$$

The primal problem: $p^* = \min_\omega \max_{\alpha,\beta,\alpha_i\ge 0} L(\omega,\alpha,\beta)$

The Dual problem: $d^* = \max_{\alpha,\beta,\alpha_i\ge 0} \min_\omega L(\omega,\alpha,\beta)$

Theorem (Weak duality):

$$d^* = \max_{\alpha,\beta,\alpha_i\ge 0} \min_\omega L(\omega,\alpha,\beta) \le \min_\omega \max_{\alpha,\beta,\alpha_i\ge 0} L(\omega,\alpha,\beta) = p^*$$

Theorem (Strong duality)

If there exist a saddle point of $L(\omega,\alpha,\beta)$ we have $\boxed{d^* = p^*}$

The KKT Conditions.

If there exists some saddle points of L, then it satisfies the following "Karush-Kuhn-Tucker" (KKT) conditions:

$$\frac{\partial}{\partial \omega_i} L(\omega,\alpha,\beta) = 0, \quad i = 1,\dots,k$$

$$\frac{\partial}{\partial \beta_i} L(\omega,\alpha,\beta) = 0, \quad i=1,\dots,l$$

$$\alpha_i g_i(\omega) = 0, \quad i=1,\dots,m$$

$$g_i(\omega) \le 0, \quad i=1,\dots,m$$

$$\alpha_i \ge 0, \quad i=1,\dots,m$$

Scanned with CamScanner

Theorem: If w^*, α^* and β^* satisfy the KKT condition, then it is also a solution to the primal and dual problems.

Support vectors.

→ only a few α_i's can be nonzero.

→ Call the training data points whose α_j's are nonzero the support vectors.

$$\boxed{\alpha_i g_i(w) = 0, \quad i = 1, \dots, m}$$

if $\alpha_i > 0$ then $g(w) = 0$

$$\text{minimize } L_p(w, b, \alpha_i) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(w^T x_i + b) - 1)$$

$$\text{s.t } \alpha_i \geq 0$$

minimize w.r.t w & b fixed α

$$\frac{\partial L_p}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial L_p}{\partial b_m} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$L_p(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j) - b \sum_{i=1}^{m} \alpha_i y_i$$

$$L_p(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

The dual problem.

Now we have the following dual opt problem:

$$\max J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

$$\text{s.t. } \alpha_i \geq 0, \; i = 1 \dots k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

This is a qp quadratic programming problem.

→ A global maximum of α_i can always be found

→ Once we have the lagrange multipliers $\{\alpha_i\}$ we can reconstruct the parameters vector w as a weighted combination of the training examples.

$$\boxed{w = \sum_{i=1}^{m} \alpha_i y_i x_i} \qquad \boxed{w = \sum_{i \in SV} \alpha_i y_i x_i}$$

→ for testing with a new data z

← compute

$$w^T z + b = \sum_{i \in SV} \alpha_i y_i (x_i^T z) + b$$

and classify z as class 1 if the sum is positive, and class 2 otherwise

Note: w need not be formed explicitly.

→ The discriminant function is:

$$\boxed{g(x) = w^T x + b = \sum_{i \in SV} \alpha_i x_i^T x + b}$$

→ It relies on a dot product b/w the test point x and the support vectors x_i.

→ Solving the optimization problem involved computing the dot products $x_i^T x_j$ b/w all pairs of training points.

→ The optimal w is a linear combination of a small number of data points.

26. SVM: Maximum Margin with Noise

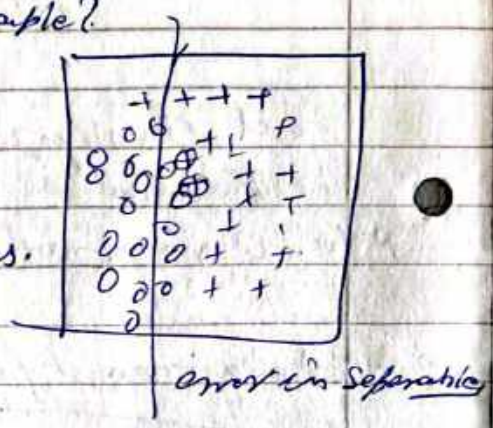
Linear SVM formulation:-

Find $\bar{w}$ and b such that $\frac{2}{\|w\|}$ is maximized
or $\frac{1}{2}\|w\|$ or $\frac{1}{2}\sqrt{w \cdot w}$ is minimized

for each training point

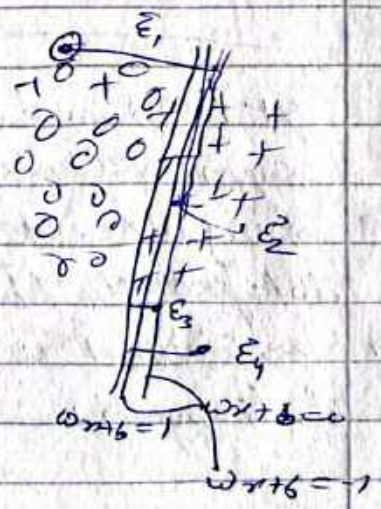
$$y_i(w x_i + b) \geq 1$$

What if the data is not linearly separable?
or noisy data points?



Extend the definition of maximum margin to allow non-separating planes.

$w \cdot w + C \#$ training errors.



finally
finally

$$w \cdot w + C \cdot \sum_{k=1}^{m} \xi_k$$

$$m = \frac{2}{\sqrt{w \cdot w}}$$

C controls the relative importance of maximizing the margin and fitting the training data.
Controls overfitting.

minimize $\quad w \cdot w + C \sum_{k=1}^{m} \xi_k$

m constraints:

$$\left[\begin{array}{l} w \cdot x_k + b \geq 1 - \xi_k \text{ if } y_k = 1 \\ w \cdot x_k + b \leq -1 + \xi_k \text{ if } y_k = -1 \end{array}\right\}$$

$$y_k(w \cdot x_k + b) \geq 1 - \xi_k, \; k = 1, \ldots m$$

$\xi_k \geq 0, \ldots k = 1, m$

Lagrangian:

$L(w, b, \xi, \alpha, \beta)$

$$= \frac{1}{2} \cdot w \cdot w + C \sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} \alpha_i [y_i (x \cdot w + b) - 1 + \xi_i] - \sum_{i=1}^{m} \beta_i \xi_i$$

α_i's & β_i's are lagrange multipliers (≥ 0).

Dual formulation:

find $\alpha_1, \alpha_2, \ldots \alpha_m$ s.t.

$$\boxed{\max_{\alpha} J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)}$$

Linear SVM or Soft SVM

s.t. $\alpha_i \geq 0, i = 1, \ldots, m$

$$\boxed{\sum_{i=1}^{m} \alpha_i y_i = 0}$$

Noise Accounted.

s.t. $0 \leq \alpha \leq C, i = 1, \ldots m$

$$\boxed{\sum_{i=1}^{m} \alpha_i y_i = 0}$$

Solution to Soft margin Classification:

→ x_i with non-zero α_i could be support vectors.

→ Solution to the dual problem is

$$w \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$b = y_k (1 - \xi_k) - \sum_{i=1}^{m} \alpha_i y_i x_i x_k$$

for any k s.t. $\alpha_k > 0$

for classification $f(x) = \sum_{i=1}^{m} \alpha_i y_i x_i \cdot x + b$

(no need to compute w explicitly)

Scanned with CamScanner

27 Nonlinear SVM & kernel function.

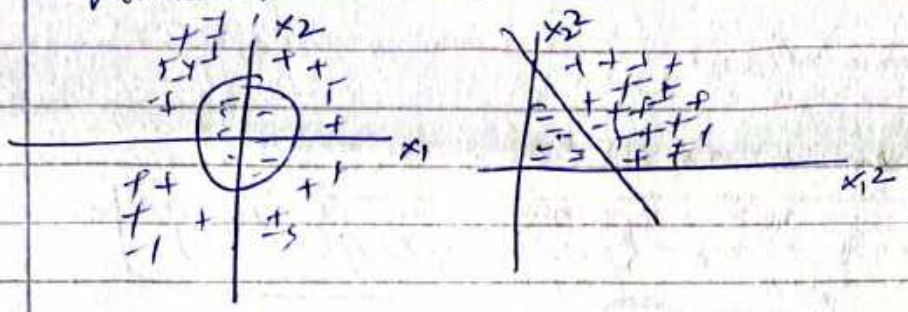
$x \longrightarrow \phi(x)$

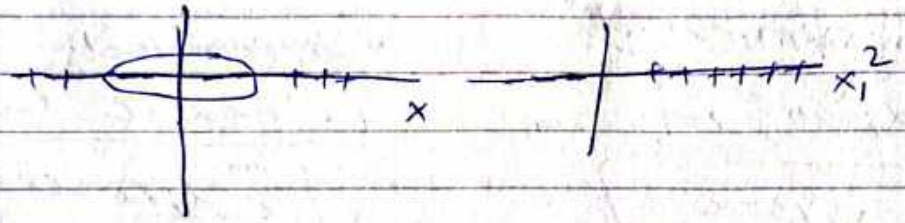
Mapping to higher dimension feature space

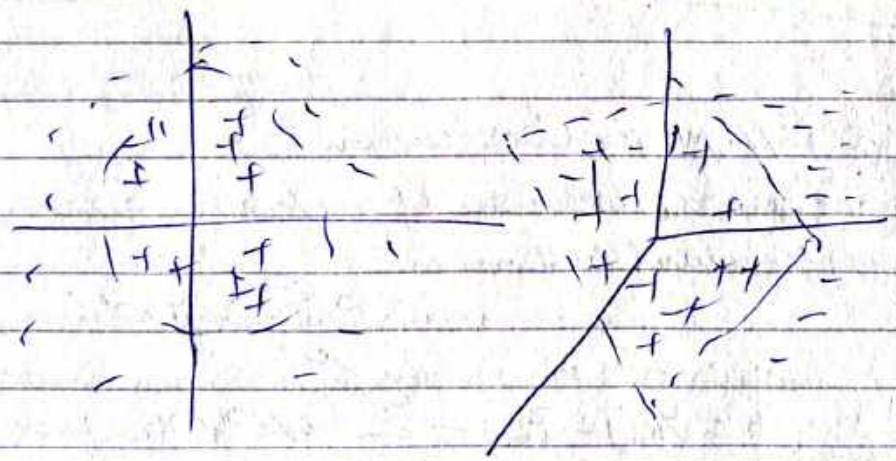
Computation cost ?

kernel function $\phi : x \rightarrow \phi(x)$

Non-linear SVM : feature space







kernel

$x_a \quad x_b$

$\phi(x_a) \; \phi(x_b)$

$K(x_a, x_b) = \phi(x_a) \cdot \phi(x_b)$

called kernel function

$x_a \; x_b$

$\phi(x_a) \cdot \phi(x_b)$

Non linear SVMs: The Kernel Trick.

$$g(x) = w^T \phi(x) + b = \sum_{i \in SV} \alpha_i \boxed{\phi(x_i)^T \phi(x)} + b$$

→ We only use the dot product of feature vectors in both the training & test.

→ A kernel function is defined as a function that corresponds to a dot product of two features vectors in some expanded feature space:

$$K(x_a, x_b) = \phi(x_a) \cdot \phi(x_b)$$

often $K(x_a, x_b)$ may be very inexpensive to compute even if $\phi(x_a)$ may be extremely high dimensional.

2-dimensional vectors $\bar{x} = [x_1 x_2]$

let $K(x_i, x_j) = (1 + x_i x_j)^2$

We need to show that $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$

$$K(x_i, x_j) = (1 + x_i x_j)^2$$

$$= 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2}$$

$$= [1 \; x_{i1}^2 \; \sqrt{2} x_{i1} x_{i2} \; x_{i2}^2 \; \sqrt{2} x_{i1} \; \sqrt{2} x_{i2}] \cdot [1 \; x_{j1}^2 \; \sqrt{2} x_{j1} x_{j2} \; x_{j2}^2 \; \sqrt{2} x_{j1} \; \sqrt{2} x_{j2}]$$

$$= \phi(x_i) \cdot \phi(x_j)$$

where $\phi(x) = [1 \; x_1^2 \; \sqrt{2} x_1 x_2 \; x_2^2 \; \sqrt{2} x_1 \; \sqrt{2} x_2]$

Commonly used kernel function.

- Linear kernel: $K(x_i, x_j) = x_i \cdot x_j$
- Polynomial of power p: $K(x_i, x_j) = (1 + x_i \cdot x_j)^p$
- Gaussian (radial-basis function):

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$

Scanned with CamScanner

→ Sigmoid

$$K(x_i, x_j) = \tanh(\beta_0 x_i \cdot x_j + \beta_1)$$

In general, functions that satisfy mercer's condition can be kernel functions.

Kernel functions

→ Kernel function can be thought of as a similarity measure b/w the input objects.

→ Not all similarity measures can be used as kernel function.

→ Mercer's condition states that any positive semi definite kernel $K(x, y)$, i.e

$$\sum_{i,j} K(x_i, x_j) c_i c_j \geq 0$$

→ Can be expressed as a dot product in a high dimensional space.

Performance

→ SVM work very well in practice

* The user must choose the kernel function and its parameters.

→ They can be expensive in time and space for big datasets.

* The computation of the maximum margin hyper plane depends on the square of the number of training cases.
* We need to store all the support vectors.

→ The kernel trick can be also be used to do PCA in a much higher dimensional space, then giving a non-linear version of PCA in the original space.

multi-Class Classification:

- SVMs can only handle two-class outputs.
- learn N SVM's
  - SVM 1 learns Class 1 vs REST
  - SVM 2 learns class 2 vs REST
  - ...
  - SVM N learns Class N vs REST
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Scanned with CamScanner

28: SVM: Solution to the Dual problem

The SMO algorithm:-

The SMO algorithm can efficiently solve the dual problem.
First we discuss Coordinate Ascent:

Coordinate Ascent:-

→ Consider solving the unconstrained optimization problem:

$$\max_{\alpha} W(\alpha_1, \alpha_2 \ldots \alpha_m)$$

loop until convergence: {

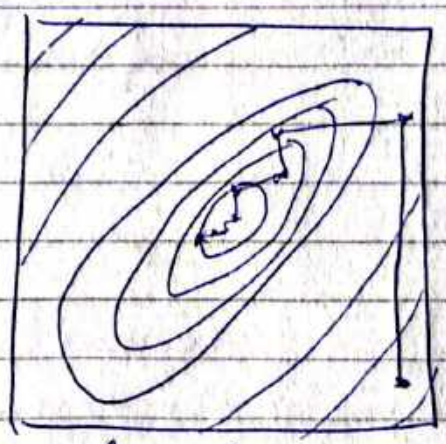
for $i = 1$ to m {

$$\alpha_i = \arg\max_{\hat{\alpha}_i} W(\alpha_1, \ldots \hat{\alpha}_i \ldots \alpha_m);$$

}

}

Coordinate ascent



→ Ellipses are the contours of the function.
→ At each step, the path is parallel to one of the axes.

The SMO algorithm.

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i x_j)$$

$$\text{s.t. } 0 \le \alpha_i \le C, \quad i = 1 \ldots m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

→ choose a set of α_i's satisfying the constraints.
→ α_1 is exactly determined by the other α's.
→ We have to update at least two of them simultaneously to keep satisfying the constraints.

Scanned with CamScanner

Repeat till convergence {

1. Select some pair α_i & α_j to update next (using a heuristic that tries to pick the two that will lead to make the biggest progress towards the global maximum).

2. Re-optimize $W(\alpha)$ with respect to α_i & α_j, while holding all the other α_k's $(k \neq i, j)$ fixed.

3

→ The update to α_i & α_j can be computed very efficiently.

Jupyter

1 tensorflow for deep learning

RHP: tflearn.org

The MNIST Data base

36 Introduction to Computation Learning Theory

Goal of Learning Theory:

→ To understand
* What kinds of tasks are learnable?
* What kind of data is required for learnability?
* What are the (space, time) requirements of the learning algorithm?

→ To develop and analyze models.
* Develop algorithms that provably meet desired criteria.
* Prove guarantees for successful algorithms.

Learnable

Which tasks?
What type of data required?
Resource requirement → space, time

In machine learning, we design algorithm.
(11) Confidence — Generalization ability

How harder to formally talk about this task we require some setting or learning model so learning we talk about in this is PAC → Probably Approximately Correct.

PAC model:-

error region $c \oplus h$

We want error region small

$\cancel{P(c \oplus h) \le \varepsilon}$

$Pr(P(c \oplus h) \le \varepsilon) \ge (1-\delta)$

$0 \le \varepsilon \le 1/2$
error parameter

$0 \le \delta \le 1/2$
confidence parameter

c → ideal class which we want to learn

Prototypical Concept Learning Task.

→ Given

* Instance X (e.g. $X = R^d$ or $X = \{0,1\}^d$
* Distribution D over X
* Target function c
* Hypothesis space H
* Training examples $S = \{(x_i, c(x_i))\}$ x_i i.i.d from D

→ Determine

* A hypothesis $h \in H$ s.t. $h(x) = c(x)$ for all x in S?
* A hypothesis $h \in H$ s.t. $h(x) = c(x)$ for all x in X?

→ An algorithm does optimization over S, find hypothesis h.

→ Goal: find h which has small error over D.

Consistent hypothesis

→ Can we be certain about how the learning algorithm generalizes?

→ We would have to see all the examples.

→ Inductive beyond the training data is impossible unless we add more assumptions (e.g., priors over H)

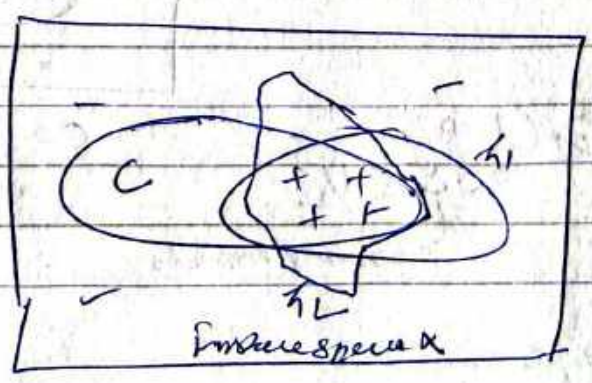
We need a bias!

Function Approximation:

→ How many labeled examples in order to determine which of the 2^{2^n} hypothesis is the correct one?

→ All 2^n instance in X must be labeled!

→ Inductive inference: generalizing beyond the training data is impossible unless we add more assumptions (e.g. bias)



$H = \{h : X \to Y\}$

$|H| = 2^{|H|} = 2^{2^n}$

Error of a hypothesis :-

The true error of hypothesis h, with respect to the target concept c and observation distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \Pr_{x \sim \mathcal{D}}[c(x) \neq h(x)]$$

In a perfect world, we'd like the true error to be 0.

Bias: fix hypothesis space H

c may not be $H \Rightarrow$ find h close to c

A hypothesis h is approximately correct if

$$error_{\mathcal{D}}(h) \leq \varepsilon$$

$h \in H$

$c \notin H$

$\boxed{h \text{ close to } c}$

PAC model

→ Goal: h has small error over $\mathcal{D}$.

→ True error: $error_{\mathcal{D}}(h) = \Pr_{x \sim \mathcal{D}}(h(x) \neq c^*(x))$

→ How often $h(x) \neq c^*(x)$ over future instance drawn at random from $\mathcal{D}$.

→ But, can only measure:

Training error : $error_S(h) = \frac{1}{m} \sum_i I(h(x_i) \neq c^*(x))$

How often $h(x) \neq c^*(x)$ over training instances.

→ Sample Complexity: bound $error_{\mathcal{D}}(h)$ in terms of $error_S(h)$

Probably Approximately Correct Learning

→ PAC learning concern efficient learning

→ We would like to prove that

* With high probability an (efficient) learning algorithm will find a hypothesis that is approximately identical to the hidden target concept.

→ We specify two parameters, ε & δ and require that with probability at least $(1-\delta)$ a system learns a concept with error atmost ε.

Sample Complexity for Supervised Learning

Theorem

$$m \geq \frac{1}{\varepsilon}\left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right)\right]$$

labeled examples are sufficient so that with prob. $1-\delta$, all $h \in H$ with $error_D(h) \geq \varepsilon$ have $error_S(h) > 0$.

→ Inversely linear in ε

→ logarithmic in $|H|$

→ ε error parameter: D might place low weight on certain part of the space

→ δ confidence parameter: there is a small chance the examples we get are not representative of the distribution.

Statistical Learning Way:

$$error_D(h) \leq \frac{1}{m}\left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right)\right]$$

37. Sample Complexity: finite Hypothesis space.

Sample Complexity: finite Hypothesis Space Realizable Case.

PAC: How many examples suffice to guarantee small error whp.

Theorem
$$m \geq \frac{1}{\epsilon}\left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right)\right]$$

Labeled examples are sufficient so that with prob. $1-\delta$, all $h \in H$ with $error_D(h) \geq \epsilon$ have $err_S(h) > 0$.

Statistical learning way:

With probability at least $1-\delta$, all $h \in H$ s.t. $err_S(h) = 0$ we have
$$err_D(h) \leq \frac{1}{m}\left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right)\right]$$

Sample Complexity: inconsistent finite $|H|$

→ for a single hypothesis to have misleading training error
$$Pr[error_D(f) \leq \epsilon + error_D(f)] \leq e^{-2m\epsilon^2}$$

→ We want to ensure that the best hypothesis has error bounded in this way.

→ So consider that any one of them could have large error.
$$Pr[(\exists f \in H)\ error_D(f) \leq \epsilon + error_D(f)] \leq |H| e^{-2m\epsilon^2}$$

→ from this we can derive the bound for the number of samples needed
$$m \geq \frac{1}{2\epsilon^2}\left(\ln|H| + \ln\left(\frac{1}{\delta}\right)\right)$$

Consistent Case

Theorem $m \geq \frac{1}{\epsilon}\left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right)\right]$

Inconsistent Case → What if there is ~~no~~ no perfect h?
$$m \geq \frac{2}{2\epsilon^2}\left(\ln(|H|) + \ln\left(\frac{2}{\delta}\right)\right)$$

Scanned with CamScanner

Sample Complexity : example

→ P : conjunction of n Boolean literals. Is P PAC-learnable?

$$|H| = 3^n$$

$$m \geq \frac{1}{\varepsilon}\left(n \ln 3 + \ln\left(\frac{1}{\delta}\right)\right)$$

→ Concrete examples:

1. $\delta = \varepsilon = 0.05$, $n = 10$ gives 280 examples
2. $\delta = 0.01$, $\varepsilon = 0.05$, $n = 10$ gives 312 examples

→ Result holds for any consistent learner, such as Find-S.

38 VC Dimension

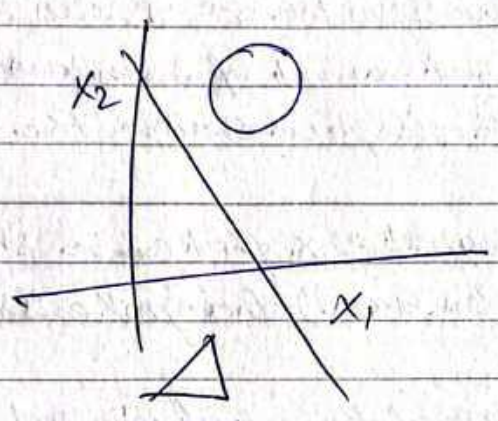
Infinite H & VC Dimension

Sample Complexity : Infinite Hypothesis Spaces

→ Need some measure of the expressiveness of infinite hypothesis spaces.

→ The Vapnik-Chervonenkis (VC) dimension provides such a measure, denoted VC(H)

→ Analagous to $\ln|H|$, there are bounds for sample complexity using VC(H).

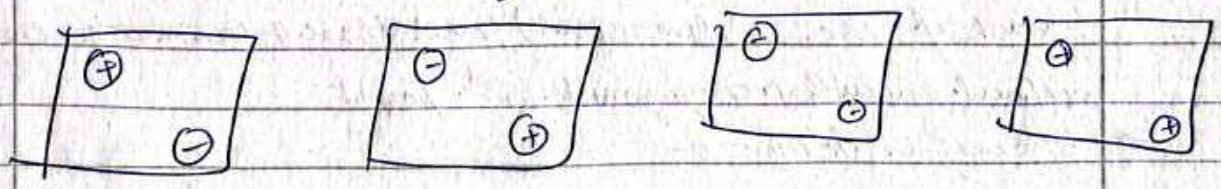


Shattering:

→ Consider a hypothesis for the 2-class problem.

→ A set of N points (instances) can be labeled as + or − in 2^N ways

→ If for every such labeling a function can be found in H consistent with this labeling, we say that the set of instances is shattered by H.



VC Dimension

→ The Vapnik-Chervonenkis dimension, VC(H) of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite subsets of X can be shattered then $VC(H) = \infty$

→ If there exists at least one subset of X of size d that can be shattered then $VC(H) \geq d$.

→ If no subset of size d can be shattered, then $VC(H) < d$.

⇒ for a single intervals on the real line, all sets of 2 instances be shattered, but no set of 3 instances can, so $VC(H) = 2$.

→ An unbiased hypothesis space shatters the entire instance space.

→ The larger the subset of X that can be shattered, the more expressive (and less biased) the hypothesis space is

⇒ The VC dimension of the set of oriented lines in 2-d is three.

→ Since there are 2^m partitions of m instances, in order for H to shatter instances: $|H| \geq 2^m$

→ Since $|H| \geq 2^m$, to shatter m instances, $VC(H) \leq \log_2 |H|$

→ No. of five instances can be shattered since there can be at most 4 distinct ~~example~~ extreme points (min & max on each of the 2 dimensions) and these 4 cannot be included without including any possible 5th point.

→ Therefore $VC(H) = 4$.

→ Generalizes to axis-parallel hyper-rectangles (conjunction of intervals in n dimensions): $VC(H) = 2n$

Upper Bound on Sample Complexity with VC

→ Using VC dimension as a measure of expressiveness, the following number of examples have been shown to be sufficient for PAC learning (Blumer et al. 1989):

$$\frac{1}{\varepsilon}\left(4\log_2\left(\frac{2}{\delta}\right)+8VC(H)\log_2\left(\frac{13}{\varepsilon}\right)\right)$$

→ Compared to the previous result using $\ln|H|$, this bound has some extra constants and extra $\log_2(1/\varepsilon)$ factor. Since $VC(H) \leq \log_2|H|$, this can provide a ~~tit~~ tighter upper bound on the number of examples needed for PAC learning

31 Introduction to Ensembles

multiple classifier or multiple learners
↳ Generate base learner

→ Combine the decisions

→ Different algorithms
→ Different hyper parameters
→ Different representations
→ Different training sets.

2-Class classification

error of each learner : ϵ

$$\boxed{\epsilon > 0.5}$$

$L_1 - \epsilon$
$L_2 - \epsilon$
$L_3 - \epsilon$
⋮
$L_k - \epsilon$

Bias: Bias is a measure of how flexible model is.
→ So if model is very flexible or very powerful so bias is low

Variance: – Variance high when if you give different subsets of data as training set the models outputs are very different then we say variances are very high.

→ usually those hypothesis for which bias is low they have variance ←

Reduce { Bias
Variance

Ensembles – which use multiple trained models

If we achieve low bias & variance what does it give us, we will get less overfitting.

⇒ Combining weak learners.

$\epsilon < 0.5$

→ Assume n independent ~~learners~~ learners, each with error 0.3 (accuracy 0.7)

→ If all n learner give the same o/p $1-(1-.7)^{10}$

$1-.3^{10}$

$n=10$

→ $n1$ say class 1
$n2$ say class 2

$n1 > n2$

$P(\text{class } 1) = 1 - \text{Binomial}(n, n_2, .7)$

$P(r) = {}^nC_r \, p^r (1-p)^{n-r}$

$= {}^nC_m \cdot .7^{n_2} \cdot .3^{n_1}$

1. Unweighted voting
2. Weighted voting

weight ∝ accuracy

weight ∝ 1/variance

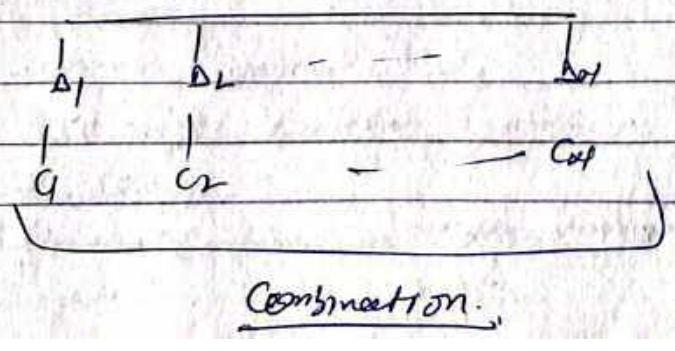
$y = \sum_{j=1}^{M} w_j d_j$ each $\begin{cases} w_j \geq 0 \\ \sum_j w_j = 1 \end{cases}$

Bayesian learning

$P(C_i|x) = \sum_{\text{all model}} P(C_i|x, M_j) \cdot P(M_j)$

Sum of

$D = \{x_i \; y_i\}$



Combination.

Random forest Algorithm:

Scanned with CamScanner

40: Bagging & Boosting

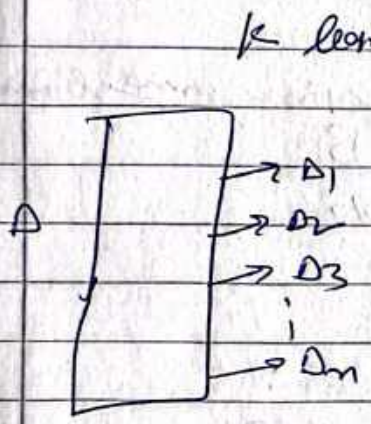
Bagging:

→ Bagging = "bootstrap aggregation"
— draw N items from X with replacement.

→ Desired learners with high variance (unstable)
— Decision trees & ANNs are unstable.
— K-NN is stable.

→ Use bootstrapping to generate L training sets and train one base-learner with each (Breiman, 1996).

→ Use voting.



draw random samples with replacement

for bagging we use unstable learner.

Sample of size n

$(1-\frac{1}{n})^n$

Boosting:

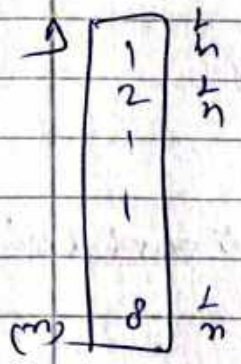
→ An iterative procedure. adaptively change distribution of training data.
— Initially, all N records are assigned equal weights.
— Weights change at the end of boosting ~~rounding~~ round.

→ On each iteration t:
— Weight each training example by how incorrectly it was classified.
— learn a hypothesis: h_t
— A strength for this hypothesis: α_t

→ Final classifier:
— A linear combination of the votes of the different classifiers weighted by their strength.

→ "Weak" learners

— P(correct) > 50%, but not necessarily much better



$D_1 - h_1 \quad \alpha_1$

$D_2 - h_2 \quad \alpha_2$

$D_3 - h_3 \quad \alpha_3$

Ada Boost:

→ Boosting can turn a weak algorithm into a strong learner.

→ input: $S = \{(x_1, y_1), \ldots (x_m, y_m)\}$

→ $D_t(i)$: weight of ith training example.

→ Weak learner A

→ for $t = 1, 2, \ldots T$

— Construct D_t on $\{x_1, x_2 \ldots\}$

— Run A on D_t producing $h_t : X \to \{-1, 1\}$

ϵ_t = error of h_t over D_t.

Given: $(x_1, y_1) \ldots (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$

for $t = 1, \ldots, T$:

— Train weak learner using distribution D_t.

— Get weak classifier $h_t : X \to \mathbb{R}$

— Choose $\alpha_t \in \mathbb{R}$

— update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Scanned with CamScanner

41 Introduction to Clustering

In the earlier so far we have talk about supervised learning. In supervised learning we have some data which are labeled and we tried to learn a function and we need to come up with method to label unseen unseen correctly.

Unsupervised Learning :- one specific type of unsupervised learning which is called Clustering.

Clustering :-

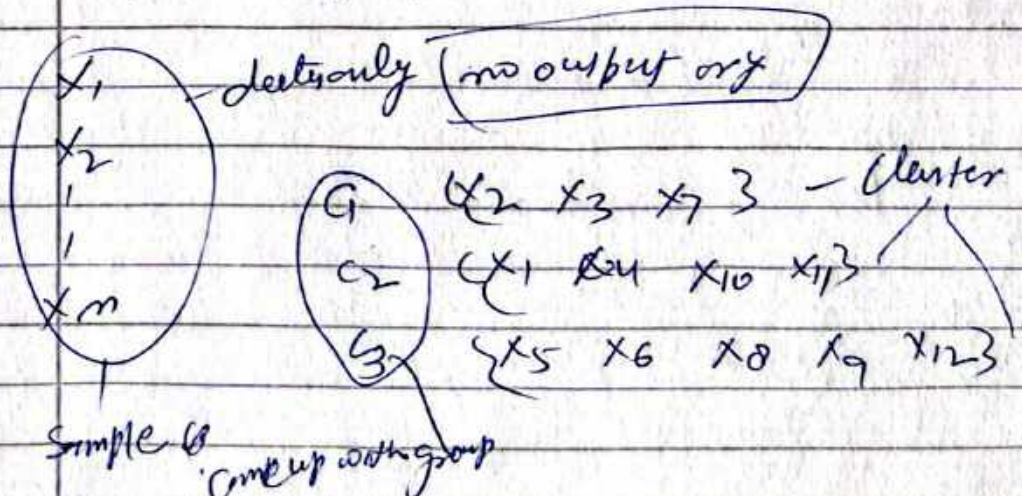
Unsupervised Learning :-

- Data with no target attribute. Describe hidden structure from unlabeled data.
- Explore the data to find some intrinsic structures in them.

Clustering : The task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other clusters.

Usual for

- Automatically organizing data
- Understanding hidden structure in data
- Preprocessing for further analysis.



⇒ Clustering ~~tat~~ Could be used to organized data.

example google news
Gene expression clustering

Aspects of clustering.
→ A clustering algorithm such as
- Partitional clustering eg. k means.
- Hierarchical clustering eg. AHC
- mixture of Gaussians.
→ A distance or similarity function
- Such as Euclidean, minkowski, cosine
→ Clustering quality.
- Inter-clustering distance ⇒ maximized.
- Intra-clustering distance ⇒ minimized

* The quality of a clustering result depends on the algorithm, the distance function, and the application.

Major Clustering Approaches:
→ Partitioning: construct various partitions and then evaluate them by some criterion.
→ Hierarchical: Create a hierarchical decomposition of the set of objects using some criterion.
→ model-based: Hypothesize a model for each cluster and find best fit of models to data.
→ Density-based: Guided by connectivity and density functions.
→ Graph-Theoretic: Clustering

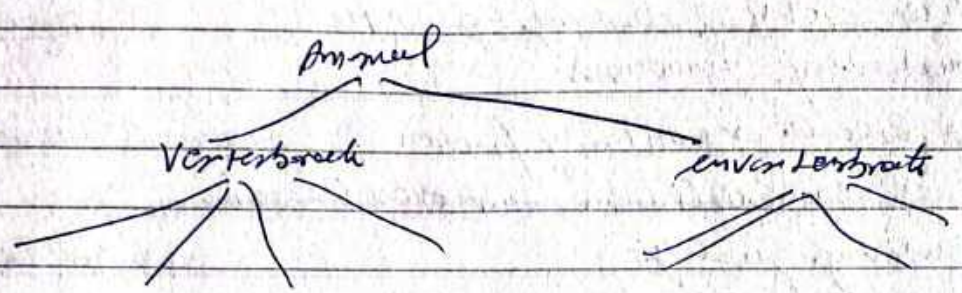
Partitioning Algorithms:

→ Partitioning method: Construct a partition of a database D of n objects into a set of k clusters.

→ Given a k, find a partition of k clusters that optimizes the chosen partitioning criterion.

- Global optimal: exhaustively enumerate all partitions
- Heuristic method: k-means (MacQueen, 1967)

Hierarchical Clustering



Model Based Clustering:

- A model is hypothesized.
- e.g. Assume data is generated by a mixture of underlying probability distributions
- Fit the data to model

Density Based:

e.g. DBSCAN

→

Graph Theoretic Clustering - weights of edges b/w items (nodes) based on similarity.
- E.g. look for minimum cut in a graph

(Dis/similarity measures:

→ Distance metric (Scale - dependent)

- Minkowski family of distance measures

$$d(x_i, x_j) = \left(\sum_{s=1}^{m} |x_{is} - x_{js}|^p \right)^{1/p}$$

Manhattan ($p=1$), Euclidean ($p=2$)

- cosine distance

$$\text{Cosine}(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\| \cdot \|x_i\|}$$

→ Correlation Coefficients (scale-invariant)

→ Mahalanobis distance:

$$d(x_i, x_i) = \sqrt{(x_i - x_j)\Sigma^{-1}(x_i - x_j)}$$

→ Pearson correlation

$$r(x_i, x_j) = \frac{Cov(x_i, x_j)}{\sigma_{x_i}\sigma_{x_j}}$$

Quality of Clustering

→ Internal evaluation:

- assign the best score to the algorithm that produces clusters with high similarity within a cluster and low similarity b/w clusters, e.g. Davies-Bouldin index

$$DB = \frac{1}{n}\sum_{i=1}^{k} \max_{j \neq i} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

→ External Evaluation:

- evaluated based on data such as known class labels and external benchmarks, e.g. Rand index, Jaccard index, f-measure

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN}$$

Scanned with CamScanner

42 K means Clustering

Partitioning! - k means algorithm. (MacQueen, 1967) Centroid

Given K objects $X = \{x_1, x_2 \ldots x_m\}$

output clusters $S = \{S_1, S_2 \ldots, S_k\}$

S_i is represented by cluster centre μ_i

Optimization Criteria WCSS (within cluster sum of square)

$$\boxed{\underset{S}{\operatorname{argmin}} = \sum_{i=1}^{k} \sum_{x \in S_i} \|x_i - \mu_i\|^2}$$

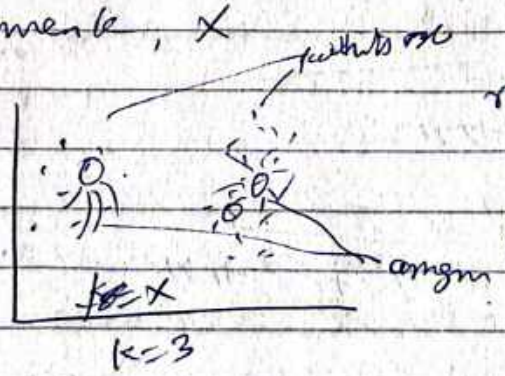
(no. of cluster)

K-means Algorithm

Given k

1. Randomly choose k data points (seeds) to be the initial cluster centres.
2. Assign each data point to the ~~closet~~ closest cluster centre.
3. Re-compute the cluster centres using the current cluster memberships.
4. If a convergence criterion is not met, go to 2.

Given k, X



Stopping / Convergence Criterion.

OR

1. no. re-assignments of data points to different clusters.
2. no. (or minimum) change of centroids.
3. minimum decrease in the sum of squared error.

$$SSE = \sum_{i=1}^{k} \sum_{x \in S_i} \| x_i - \mu_i \|^2$$

Similarity / Distance measures

→ Distance metric (Scale-dependent)

– Minkowski family of distance measures:

$$d(x_i, x_j) = \left(\sum_{s=1}^{n} |x_{is} - x_{js}|^p \right)^{1/p}$$

Manhattan ($p=1$), Euclidean ($p=2$)

– Cosine distance – mainly when dealing with text documents.

→ Correlation coefficients (Scale-invariant)

– Mahalanobis distance

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^{\Sigma^{-1}} (x_i - x_j)}$$

→ Pearson Correlation.

$$r(x_i, x_j) = \frac{Cov(x_i, x_j)}{\sigma_{x_i} \sigma_{x_j}}$$

Convergence of k-means

→ Recomputation monotonically decreases the sum square error since

(m_j is number of members in cluster j):

$\sum (x_i - a)^2$ reaches minimum for:

$$\sum -2(x_i - a) = 0$$

$$\sum x_i = \sum a = m_j a$$

$$a = \frac{1}{m_j} \sum x_i = c_j$$

→ [illegible]

Time Complexity:

→ Computing distance b/w two items is $O(n)$ where n is the dimensionality of the vectors.

→ Reassigning clusters: $O(kmn)$ distance computations, or $O(kmn)$

→ Computing centroids: Each item gets added once to some centroid: $O(mn)$

→ Assume these two steps are each done once for t iterations: $O(tknm)$

Advantages:

→ Fast, robust easy to understand.

→ Relatively efficient: $O(tkmn)$

→ Gives best result when data set are distinct or well separated from each other.

Disadvantages:

→ Requires apriori specification of the ~~no~~ number of cluster ~~centrs~~ centers.

→ Hard assignment of data points to clusters.

→ Euclidean distance measure can unequally weight underlying factors.

→ Applicable only when mean is defined i.e, fails for categorical data

→ only local optima

43 Agglomerative Hierarchical Clustering

Scanned with CamScanner