

21/10/16

### Job names -

- Data mining expert
- Machine learning expert
- Predictive Analyst
- Data Scientist
- BIG DATA Analyst → + Aware of Big data ML platform  
(Spark / H2O)

Hadoop being replaced by "Spark"

- AI programming
- SAS analyst
- Cognitive learning expert

Machine - oriented

Course diff from BIG DATA developer - just scripting

We'll learn about R / Python - scripting - most popular  
Hands-on data science work.

To learn:

Kaggle.com (Competitions)

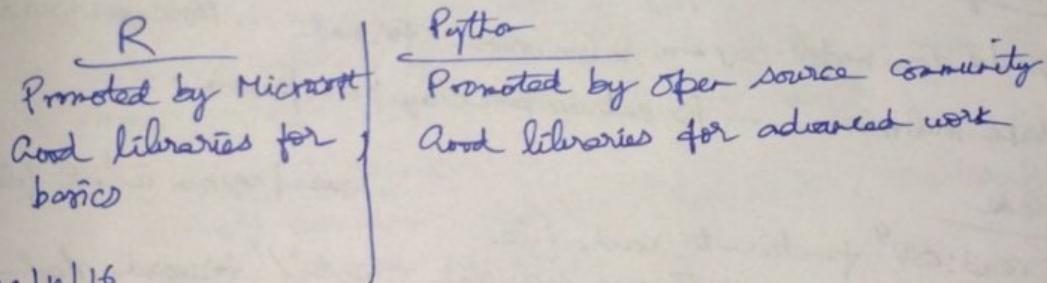
topcoder.com

<https://github.com/algoithmic-repository/datascience>

Assignments regularly or weekends

No face experience - Generic working knowledge

Do "Proof of Concept" innovation



### R language

- └ R Studio (IDE)
- └ R runtime (executive)

0.99.903 - Rblast

R 3.1.3 or latest

### Python language

- └ Jupiter (IDE)
- └ Python runtime

### Scala language

- └ Scala IDE (many IDEs)
- └ Scala runtime

Kaggle.com / Competitions

wifi pwd  
algorithms

"Knowledge Base" questions

Titanic : ML learning from disaster

D/L train.csv, test.csv

R studio supports calculator work flow

(eg)  $1 + 2$

$\vdots$

(\*) Predictions — Astrologers / Nostradamus

- email spam or not
- will it rain tomorrow?
- will this person default on loans?
- will my product sell well? How much will sell next month?

SibSp - Sibling Spouse

Parch - Parent/children

Kaggle > Leaderboard — top predictors

100% predictors are bluffers - got o/p by googling

Titanic survivor predictions

- Simple approach is 'say all died' since most died (train.csv)  
Majority prediction

Scripting languages are interactive - Hence best for interactive analysis unlike C++, Java etc.

genderclassmodel.csv ← submission format - Passenger | Survived  
genderclassmodel.csv ← submission format - Passenger | Survived  
"Make Submission" - 10 entries per day per person.

R Code

Save program as 'R' file

"read.csv" function to read file  
titanic-test =  $\nearrow$  give full path - use '/' forward slash  
> read.csv("test.csv")  $\nearrow$  for windows slashes compatibility  
dim(titanic-test)

titanic-test\$Survived = 0 (adds a new column 'Survived' with value 0)  
write.csv(titanic-test[, c("PassengerID", "Survived")],  
"submission.csv");

$\hookrightarrow$  wrong data format, extra row ID columns.  
 $\hookrightarrow$  all rows, but 2 do

write.csv ( -same, rownames = FALSE ) row.names = F .

Submission gives about 62% correct with 'majority' model.

Now by "random" strategy.

titanic-test & survived = sample(c(0,1), 418, replace = T) # rows

We get 50.7% accuracy now.

↳ to re-use  
guessed values.

So, "majority vote" is better than "random" guess.

Try another approach ('-R file) - use "train" for knowledge

titanic-train = read.csv("train.csv")

dim(titanic-train)

summary(titanic-train) → Age: NA's 177 → no data for these  
177 people

max = 80 → realistic age

70% of this course is on "Predictive Analytics".

Others are

Laptop recommended

Classification - 8 GB RAM (es) Regression - based on similarity principle.  
will this customer respond to this offer? How much will the house cost? Identifying people who are similar in the modelling will happen, products they purchase or liked. Recommend similar products for Mr. beetle

Graphics card - 'CUDA' support

for 'Data Science', logic is important. API is not important.

Book reading is not so good.

Try to solve a problem. After you are exhausted, read a book to learn ideas from others.

25/10/16 Why Data?

33% business revenue based on recommendations.

Get competitive advantage by providing personalized services

Get insights from data to make better decisions/a better product

So, data has interest & value and cannot be discarded

- Amazon, Yahoo, Google, etc. are into data insights.

Data examples/sources

- Products purchased - customers & their data (structured data)

- Social networks (Unstructured data)

- Posts in social media

- Pictures/Videos/Emails/Voice data

- YouTube automating video analysis - Horror/Romance/..?

↳ Data scientists need to know tools to handle structured & unstructured data)

↳ Mobiles

- GPS data

- Mobile apps data

- Voice data

→ CCTV cameras

- automatic challans for traffic violations

→ Server logs (Petabytes)

- semi-structured data

- Server logs...
  - Predict how much load a server next month
- IoT devices (Internet of things)
  - 65+ billion devices on Internet by 2010. Up to 230 billion by 2020
  - Alert us before heart stroke, how to make devices smart
  - Smart city - automatically find parking slot

(less than 1% of demand for data scientists is net.)  
Need diff thinking style, not just tools

Structured - Transactional data

Semi Structured - Log data, XML/JSON data, Search data

Unstructured - Image, voice, email, Video, Chats, Blogs,

(eg) Facial keypoint detection - Recognize nose, lips, cheeks etc. to identify emotions.

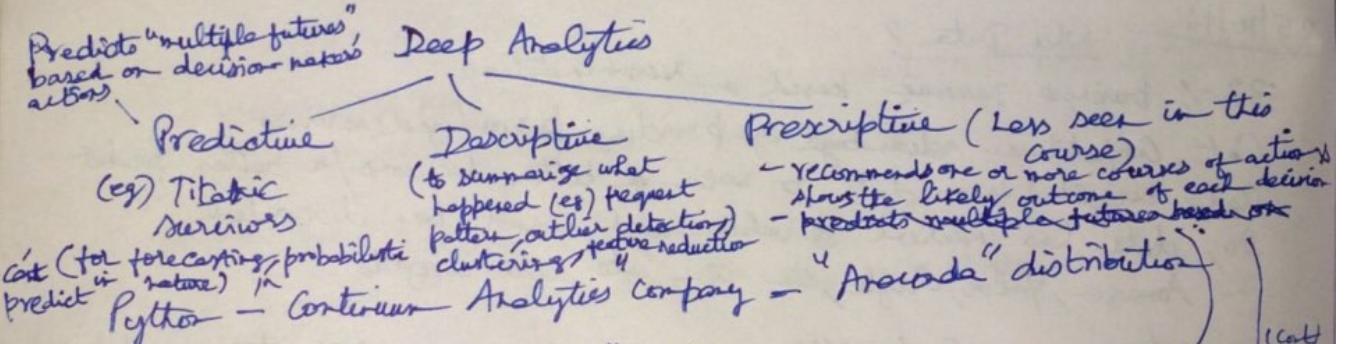
Digit recognizer - Identify numbers written by hand

Dogs vs Cats - Discover if image of dog or cat

Unstructured data analysis is the most dominant problem

Financial analysis - mostly structured data analysis.

Unstructured analysis needs NLP, Image processing



Install "RStudio Desktop" version

Python - 2.7 version - 64 bit installer (use this)  
3.5 version incompatible with 2.7 version

read.csv {utils} → package name.  
↳ group of related fns.

"R" comes with some default fns.

library() → gives available packages

?read.csv → gives help on read.csv fn

- (cont)
- Requires predictive model + two additional components: actionable data & feedback system
  - Since a prescriptive model can predict the possible consequences based on a diff choice of action, it can also recommend the best course of action for any pre-specified outcome.

In windows, use '/' or '\\' for path in R.  
"titanic-test" is a data-frame object / Frame container  
str(titanic-test) → gives structure of the frame container

TL/Hadoop / BI teams work on gathering data for use by the data scientists.

Recommended hardware - Quadcore, preferably i7 (not dual core)  
NVIDIA GPU to run intense computations using CUDA

26/10/16

Environment / workspace - Collection of all objects in current session

Package - group of related functions

- |- install package → goes to C:\...\R\3.1.3\library Tools > Install Packages (or) install.packages(package)
- |- load package → this available to current session library(package) (or)
- |- explore package for - ls ("package: packageName").require(package)
- |- how to know what packages are installed? library()
- |- how to know what packages are loaded to current session? search()

Menu

Tools > Install Package

- Install from CRAN

(or)

- Install from downloaded .zip, .tar, .gz files

Data to load to R environment (variable values) is in 'RData' file.  
This is not in most languages like Java, but is here as date is very important.

To save data in environment,  
save.image ("./temp1.RData") # "R workspace" file

↓ To load this

Session > Load workspace

To learn any scripting language

a) basic types

    character  
    logical

b) Containers / Data Structures

c) control statements

d) closures - specific to functional programming

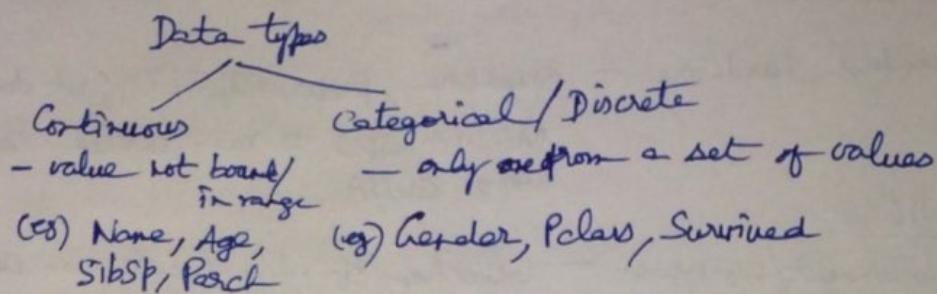
Array (3 or more dimensions)  
Vector (1D array) - all data of same type  
Dataframe (to read excel sheets)  
Matrix - for rows/columns (2D array)  
List - most flexible  
Factor - for enum types

```

a = 10 # R treats this internally as a vector of size 1.
class(a) # type of variable. - numeric
b = "abc"
class(b) # character
c = TRUE
class(c) # logical.

```

d = T(a) TRUE  
class(d) # logical



Vectors for homogenous data types. If we mix, there is a "forced" type casting.

v = c(10, 5, "abc", TRUE).

For heterogeneous values, use a "List" instead of vector

age = c(10, 20, 30)

age = 10 + age # add 10 to each age

age2 = c(age, 15) # add 15 to age and assign to new vector

rm(age2) # remove object from session

age3 = c(15, 12, 25, 19)

sort(age3) # returns vector in sorted order

age3 = sort(age3) # to modify vector in sorted order

gc() # garbage collection

27/10/16

Vectors - elements of same type

- elements can be accessed by index or name.

e <sub>1</sub>	e <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	e <sub>5</sub>
10	20	30	40	50
1	2	3	4	5

v1 = c(10, 20, 30)

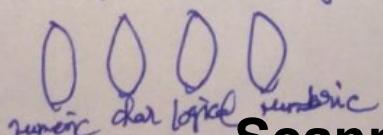
names(v1) = c("e1", "e2", "e3")

⇒ v1[2] & v1["e2"] both are the same 20 values.

Every container in R has "named" access apart from "indexed" access

v1[1:2] # 10, 20. Subsetting

Dataframe - Group of heterogeneous vectors



$\text{ids} = 1:4$   
 $\text{names} = c("S1", "S2", "S3", "S4")$   
 $\text{Students} = \text{data-frame}(\text{ids}, \text{names})$   
 $\text{class}(\text{Students}) \# \text{data-frame}$   
 $\# \text{find structure of frame}$   
 $\text{str}(\text{Students})$   
 $\# \text{find dimensions of frame}$   
 $\text{dim}(\text{Students})$

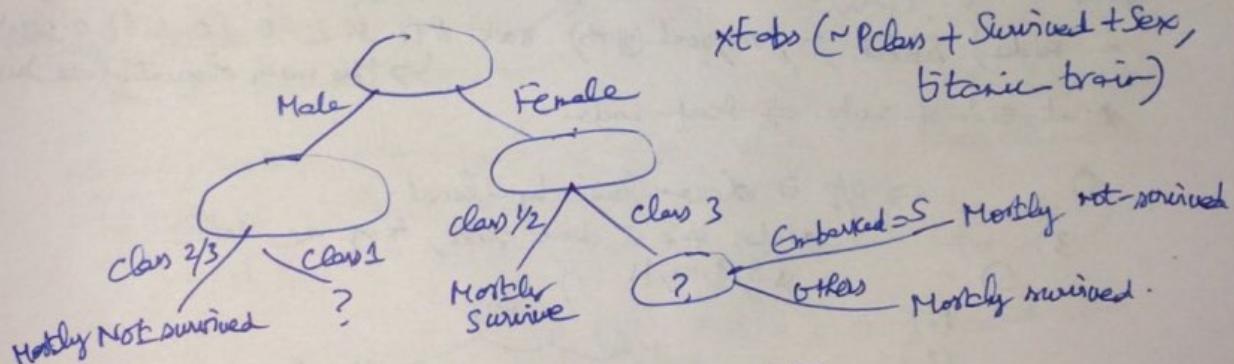
$\text{Students}[1,] \# \text{details of first row}$   
 $\text{Students}[, "names"] \# \text{all columns}$   
 $\text{Students}[\text{ids} > 2,] \# \text{all rows satisfying the condition}$   
 shortcut  $\rightarrow \text{Students}[\text{names}]$

To cross-classify values in dataframes, for tabular data,  
use "`xtabs`"  $\rightarrow$  cross-tabulation

use "`as.factor`" to convert 'continuous' data to exam types.  
 $\text{titanic\_train} \$ \text{Survived} = \text{as.factor}(\text{titanic\_train} \$ \text{Survived})$   
 $\text{by} \$ \text{Sex}$

`xtabs(~Sex + Survived, titanic_train)`

...  
 $\text{titanic\_test} \$ \text{Survived} = \text{ifelse}(\text{titanic\_test} \$ \text{Sex} == "female", 1, 0)$   
 $\hookrightarrow \text{forecast females as survivors.}$



We can use "Tableau" tool to explore data

Though lenses can attempt to find logic, more issues is it

- manual work
- hardcoded logic, need to rediscover whenever train data changes
- is it practical to discover logic/pattern if train data is big?

28/10/16  
 Titanic problem: check if "fare" helps to improve logic

Need Machine Learning to handle large data

Date scientists

Machine learning algorithm:

rpart → recursive partitioning

library (rpart)

breastcancer = rpart (Survived ~  $\text{Sex} + \text{Pclass} + \text{Embarked} + \text{Fare}$ , titanic\_train)

Need logic for this Use these to get logic

rpart uses the "CART" algorithm to generate decision trees.

CART - can work with both continuous & discrete data

- can handle missing values

- can easily handle outliers. Outliers have a negative effect on some statistical models like Principal Component Analysis (PCA) and linear regression. But CART will isolate outliers in a separate node.

- solves the 'overfitting' problem by controlling the minimum no. of observations in each node ("minsplit" in rpart)

(disadvantages)

- splits by only one variable. (e.g.)  $(x_2 - x_1) \leq 0$  can't be used as a condition

- May have unstable decision trees:

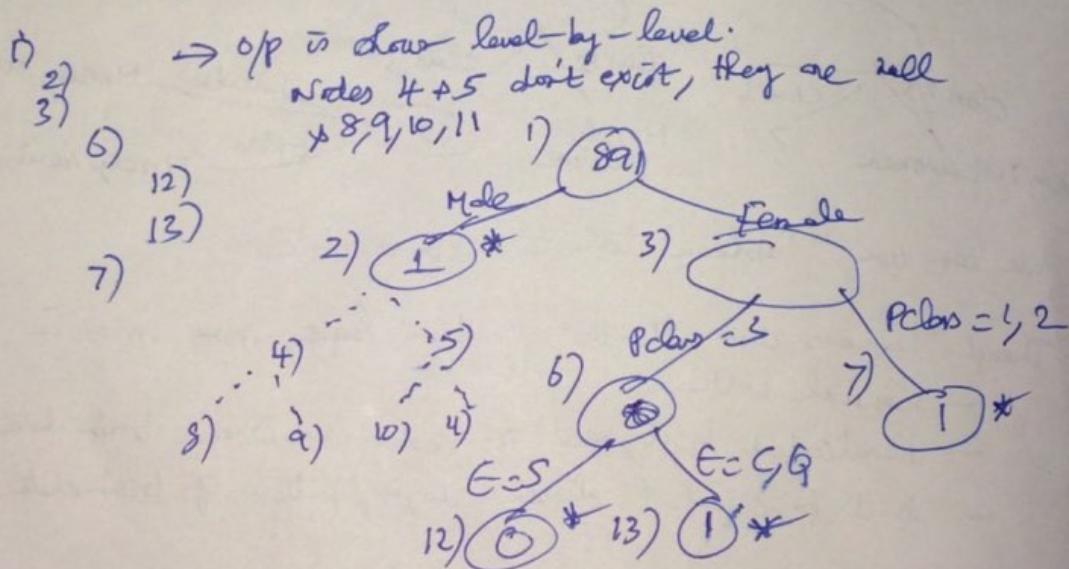
Insignificant changes to training data, such as removing some observations can drastically change the decision tree

31/10/16

Machine learning of classification trees

- Machine learning algorithms loss of correct o/p predicted o/p.  
output of rpart node for this many observations

- node split, n, loss, yval(gprob) root 891 342 0 (0.6161 0.3838)  
\* at end of node  $\Rightarrow$  leaf node



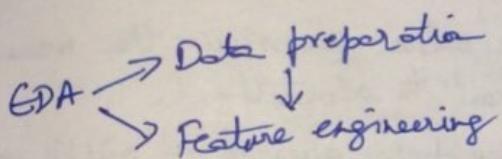
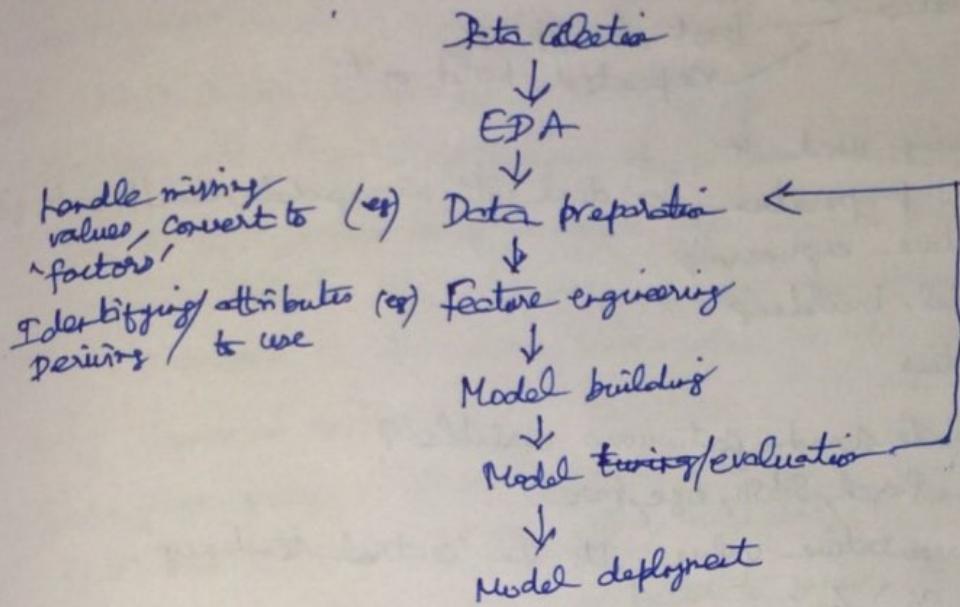
Predicted o/p for non-leaf nodes to deal with missing data and still give predictions.

(e.g.) If female, but we don't know Pclass, then predict as "L" (survived)

## Methods of machine-learnt approaches

- Trees (or) O/P of rpart()
- Neural networks
- Others

tree model = rpart(Survived ~ Sex + PClass + Embarked + Fare, titanic\_train,  
method = "class")  
Data analytics lifecycle

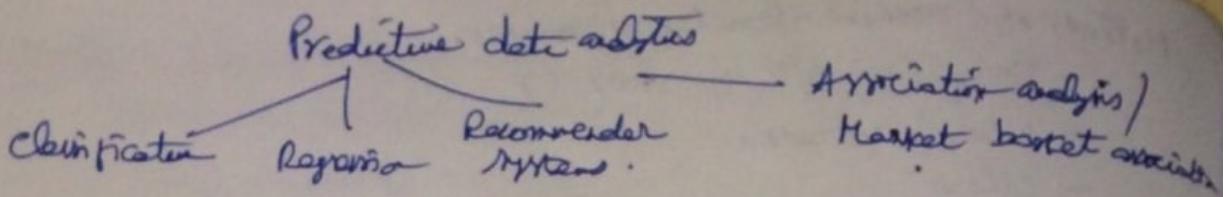


Model evaluation to be done on train data  
Can we divide train data into 80% and 20%?  
(train set) (validation set)

train() method from 'caret' package for model evaluation  
↳ tells if attributes have missing values.  
read.csv ("train.csv", na.strings = c("NA", "na"))  
tells that "NA" & "na" be treated as empty values.

O/P of train(Survived ~ Sex + PClass + Embarked + Fare, titanic\_train,  
method = "rpart")  
- Resampling: Bootstrapped

cp	Accuracy	Kappa
0.013157	0.804784	
0.0307	0.786	
0.444	0.718	



Need to learn "statistics" to evaluate models

- ↳ mean, median, quartile, moments, SD, variance
- ↳ evaluation
  - ↳ cross validation
  - ↳ bootstrapping
  - ↳ repeated hold out

Machine learning used in

- data preparation (e.g) deal with missing data. Can/How to fill them?
- feature engineering
- model building

11/11/16      Statistics

How to understand continuous variables?

↳ Porch, SibSp, Age, Fare

Mean - representative value. It's the "central tendency".

$$= \left( \sum_{i=1}^n a_i \right) / n$$

- problematic since an 'extreme' value can skew the mean towards it. We use 'Median' to solve this.

Median - 'middle value'. If 'even' numbers, average of 'middle values'

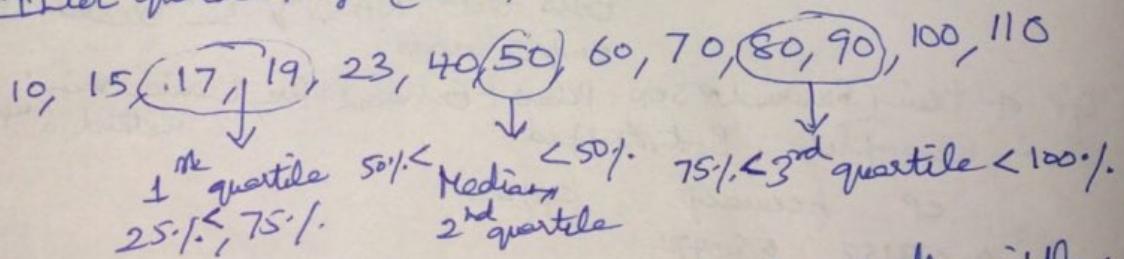
If 'Mean ≈ Median' → no extreme values; uniform distribution.

But mean, median do not enough to measure 'Spread'. We need other values like

outliers impact mean but not median

- min } Range
- max }
- Standard deviation

- Inter quartile range (IQR)

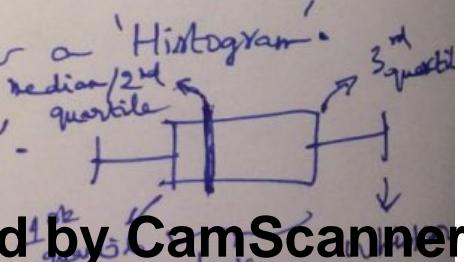


[1<sup>st</sup> quartile, 3<sup>rd</sup> quartile] - range of 50% values in the middle.

Statistics are there help humans to quickly understand a range of values.

To know about the entire data, draw a 'Histogram'.

Box-Whisker plot - gives 'quartiles' -



In 'histogram', specify only X-axis. Y is automatically the counts at various 'X' values.

X1() # gives new window to draw plots

ggplot(titanic\_bair) + geom\_histogram(aes(x=fare)) + coord\_flip()

warning: stat\_bin() using 'bins = 30' — dividing data range into 30 bins

→ shows lesser count as fare goes up. We can change the 'bin'.

Histogram also shows distribution of values.

In box plots, values below & above the whisker boundary are called 'outliers'. The boundary of 'box' are the  $1^{\text{st}}$  &  $3^{\text{rd}}$  quartiles. The 'median' is the dark/thick value in the box.

Why 'Standard deviation'?

Tata stock price — 100, 150, 70, 50, 200 → too much fluctuation, risky

MS stock price — 100, 120, 150, 150, 160 → less fluctuation, less risky.  
→ better to invest stocks here.

One way to measure fluctuation  $\leftarrow \text{MAD}$

- find ~~median~~ mean  $\sqrt{\text{MSD}}$
- take diff. of mean and each value
- sum the 'absolute' value of the diffs. Take the 'median' of the absolute deviations  $\Rightarrow \text{MAD}$  (Median absolute deviation)
- Another is 'Standard' deviation  $\Rightarrow \sqrt{\text{MSD}}$ 
  - square root of 'Mean squared deviation'.

Standard deviation = Root mean squared deviation

$$(\text{SD}) = \sqrt{\text{Variance}}$$

Variance = Mean squared deviation

In 'standard deviation' approach, we give more 'weightage' to larger differences. Also, SD preferred over MAD w.r.t. 'differentiation' calculations / derivatives

$$\frac{d}{dx}(x)$$

$$\frac{d}{dx}(x^2)$$

$\rightarrow$  easier to find.

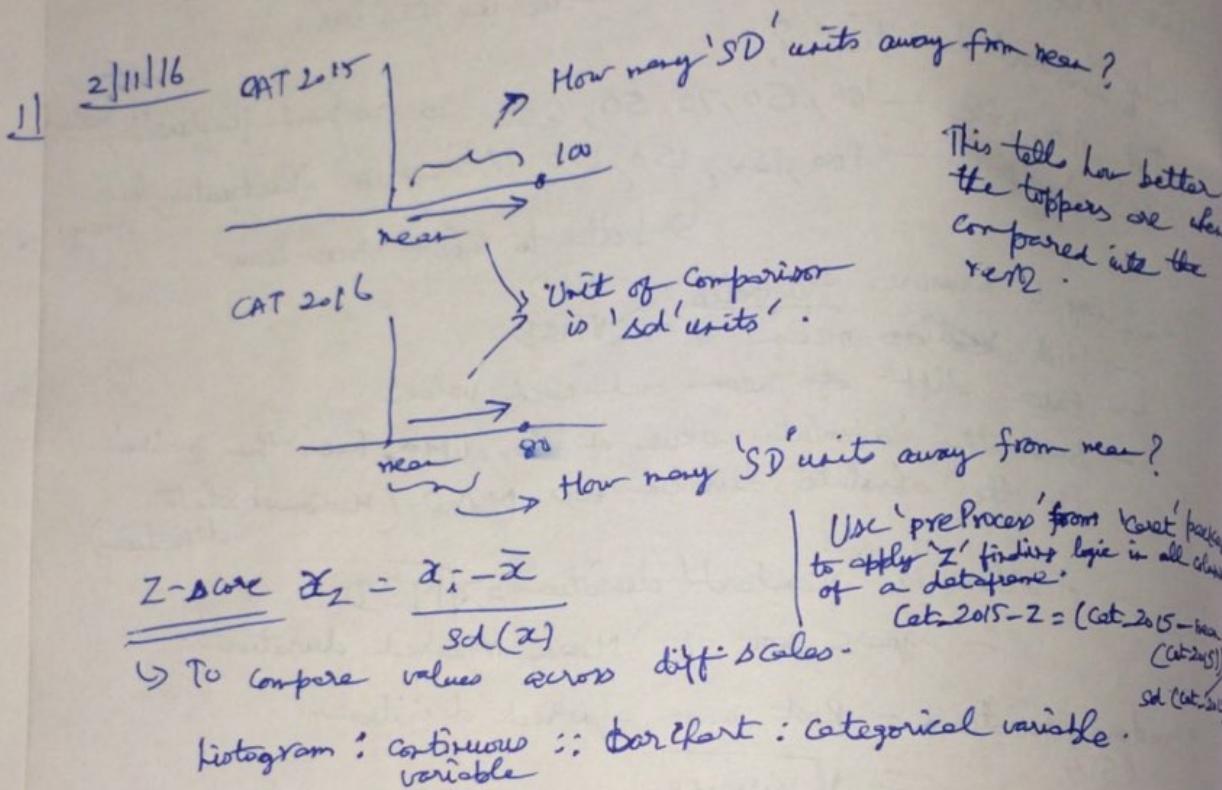
Another approach for 'fluctuation' =  $\sqrt[4]{\sum (x - \bar{x})^4}$ .

— We'll use this for 'movements'.

CAT 2016 score	CAT 2015 score
100	86
99	82
98	81
97	80

which of these toppers is better  
BTR are 60th percentile

- Idea 1)  $\text{med}(\text{score})$ , centre =  $\max(\text{score})$ ) = Median  
 2)  $\max(\text{score}) - \text{mean}(\text{score}) = \frac{\sum (\text{score} - \text{mean}(\text{score}))}{n}$   
 this is all scores other than  
 max score.
- 3)  $\sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$ .  $x = \text{Score set}$ .



For visualizing data,  
non-technical users : Tableau

There is a 'plot' function in graphics package, but is harder to use. So, we use 'ggplot' for uniform grammar for graphics.

### Exploratory data analysis with graphs

- Exploring categorical variables
  - Use a bar chart to visualize categorical variables.
  - $\text{xtabs}(\sim \text{Survived}, \text{titanic\_train}) \equiv$  to
  - $\text{ggplot}(\text{titanic\_train}) + \text{geom\_bar}(\text{aes}(x = \text{Survived}))$

- Use bar chart to also explore multivariate relationships among categorical variables
- (eg)  $x\text{tabs}(\sim \text{Sex} + \text{Survived}, \text{titanic\_train})$  for better still  
 $x\text{tabs}(\sim \text{Survived} + \text{Sex}, \text{titanic\_train}) \equiv$   
 $\text{ggplot}(\text{titanic\_train}) + \text{geom\_bar}(\text{aes}(\text{x} = \text{Sex}, \text{fill} = \text{Survived}))$
- The survivors & non-survivors are differentiated by color in each bar.
- (eg) 3-variables  
 $x\text{tabs}(\sim \text{Pclass} + \text{Survived} + \text{Sex}, \text{titanic\_train}) \equiv$   
 $\text{ggplot}(\text{titanic\_train}) + \text{geom\_bar}(\text{aes}(\text{x} = \text{Pclass}, \text{fill} = \text{Survived}))$   
 $+ \text{facet\_grid}(\text{Sex} \sim \cdot)$

- (eg) 4-variables  
 $x\text{tabs}(\sim \text{Embarked} + \text{Survived} + \text{Pclass} + \text{Sex}, \text{titanic\_train}) \equiv$   
 $\text{ggplot}(\text{titanic\_train}) + \text{geom\_bar}(\text{aes}(\text{x} = \text{Embarked}, \text{fill} = \text{Survived}))$   
 $+ \text{facet\_grid}(\text{Sex} \sim \text{Pclass})$

2) Exploring relationship between categorical and continuous variables

(eg) 2 variables - Use box plots or histograms.

- (eg) 2 variables  
 $x\text{tabs}(\sim \text{Fare} + \text{Survived}, \text{titanic\_train})$   
 $\equiv \text{ggplot}(\text{titanic\_train}) + \text{geom\_box}(\text{aes}(\text{x} = \text{Survived}, \text{y} = \text{Fare}))$
- (eg)  $\text{ggplot}(\text{titanic\_train}) + \text{geom\_histogram}(\text{aes}(\text{x} = \text{Fare})) +$   
 $\text{facet\_grid}(\text{Survived} \sim \cdot)$  [or ( $\sim \text{Survived}$ )]

Note that histogram works only with continuous variables ('Fare' in this case).

- (eg) 3 variables  
 $x\text{tabs}(\sim \text{Fare} + \text{Survived} + \text{Sex}, \text{titanic\_train})$   
 $\equiv \text{ggplot}(\text{titanic\_train}) + \text{geom\_histogram}(\text{aes}(\text{x} = \text{Fare})) +$   
 $\text{facet\_grid}(\text{Survived} \sim \text{Sex})$
- (eg) 4+ variables - plot gets messy  
 $x\text{tabs}(\sim \text{Fare} + \text{Survived} + \text{Sex} + \text{Embarked}, \text{titanic\_train})$   
 $\equiv \text{ggplot}(\text{titanic\_train}) + \text{geom\_histogram}(\text{aes}(\text{x} = \text{Fare})) +$   
 $\text{facet\_grid}(\text{Survived} \sim \text{Sex} + \text{Embarked})$  [or ( $\sim \text{Survived} + \text{Sex}$ )  
 $\sim \text{Embarked})]$

- 3) Exploring continuous vs continuous variables  
- Use scatterplots  
 $\text{ggplot}(\text{titanic\_train}) + \text{geom\_point}(\text{aes}(\text{x} = \text{Fare}, \text{y} = \text{Age}))$

3/11/16 Machine learning not needed in some stages of life cycle  
'Data collection'.

Type casting to factors, for Sex, Pclass, Embarked, Survived are part of 'data preparation'.

Model evaluation - how well does it perform on unseen data?

re-sampling techniques

Model evaluation

Repeated Holdout } all 3 are resampling  
Cross validation } techniques.  
Bootstrapping }

Sampling - If a company manufactured  $10^6$  units, it will randomly choose about 10 or 100 units to assess overall quality of units.

Six Sigma - Only 1 in  $10^6$  units have defects; great quality.

Re-sampling - Sampling 10 units many times.

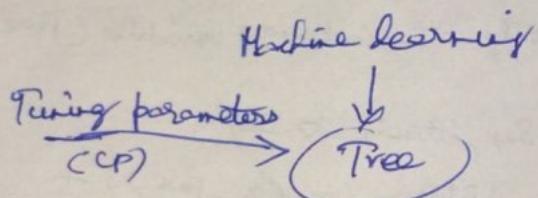
↳ 10, 10, 10, 10, 10

Re-sample on 'train' data

Randomly choose 80% + 20%, train-test split multiple times and see how model performed on test on average.

Output of 'train'

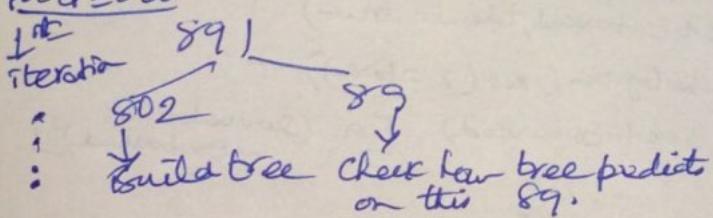
CP - Complexity parameter for tree learning.



How to find the right tuning parameter?

- Once found, build tree model on entire data.

For  $CP=0.00$



10 iterations  
Many such iterations, find 'average' prediction on '89' test sample.

For each 'CP' value, do 10 iterations.

Choose best 'CP' (highest average accuracy over the 10 iterations), say, '0.05'.

It built 10 trees and tested each on 89 test samples.

Finally, take  $CP = 0.05$ , and build tree on all 891 data.

↳ Can't evaluate this final tree model as there is no 'test' data left.

But, we predict its accuracy as average performance on 10 iterations over 89 test samples.

If there are multiple tuning parameters, of  $3 \times 3 = 9$  combinations,  $\text{③}^{\text{tun}} \times \text{③}^{\text{par}}$ .

The 'train' function does all this work.

How to split train-test %?

No math. But heuristics suggest 70-30% or 75-25%.

$cp=0 \Rightarrow$  gives a deep tree. This is over-fitting, caused by learning from 'noise'.

'train', by default, uses only 3 cp values, which it chooses using heuristics.

$cp=1 \Rightarrow$  gives tree with only one node - All people 'died' etc. Different tuning parameters give the best accuracy for different data sets.

- In train() function, resampling is done for each tuning parameters combination on a given dataset, then the "tuning parameters" that work best on that dataset are chosen, and the best tuning parameters are then used on the entire dataset (and not just some 'train' data subset) to generate the final "classification tree".

We, data scientists, need to experiment with various tuning parameter combinations on a dataset using our experience / heuristics. This requires a LOT of computing power and time. Hence the need for distributed platforms like Spark & H2O.

When we experimented with cp values - seq(0, 1, 0.01) - 100 values, the best accuracy was still lower than what was got using 'three default cp values' chosen by heuristics based on data.

4/11/16

If we have a ready-made model and the sample that was used to build the model, it is not possible to estimate the model's prediction accuracy on unseen data.

- For 'repeated holdout', best # iterations for each tuning parameter = 10. Based on heuristic
- For 'bootstrapping', best # iterations  $n = 25$ . Default values in 'train'

Sometimes people give large 'test' data and very less "train" data. Don't know if this is good enough. Depends on logic discoverable with less "train" data.

There's no pattern or whether there's more train data or test data.

In repeated holdout & cross-validation, "stratification" is preferred - Ensures % of survivors & non-survivors (target attr) is equal across train and test data.  $\text{train}()$  does this by default but only stratifies by 'Survived'. Cannot stratify by 'Sex' but we offset skewness by multiple iterations.

### Cons of 'Repeated Holdout'

- Some samples may end up always on train  $\Rightarrow$  <sup>always on</sup> test.

This issue is solved using K-fold cross validation.

- Break 891 into 10 parts  $\boxed{89|89|89| \dots |89}$

- 1<sup>st</sup> iteration : take 1<sup>st</sup> part as test  
2<sup>nd</sup> iteration : take 2<sup>nd</sup> part as test } covers all samples  
10<sup>th</sup> iteration : take 10<sup>th</sup> part as test }

- Number of folds = Number of iterations  $\rightarrow$  This is done
- As per heuristics, # iterations = 10 works well in practice -

$(K=10 \text{ or } K=5 \text{ works well}) \quad (n=150)$

- If there is a very small sample of size just 150, we cannot ignore any data. So, select  $K=n$ , i.e., # iterations = sample size. So, each fold has only 1 sample  $\Rightarrow$  test data size = 1. Therefore, this is called "leave-one-out" when  $K=n$ .

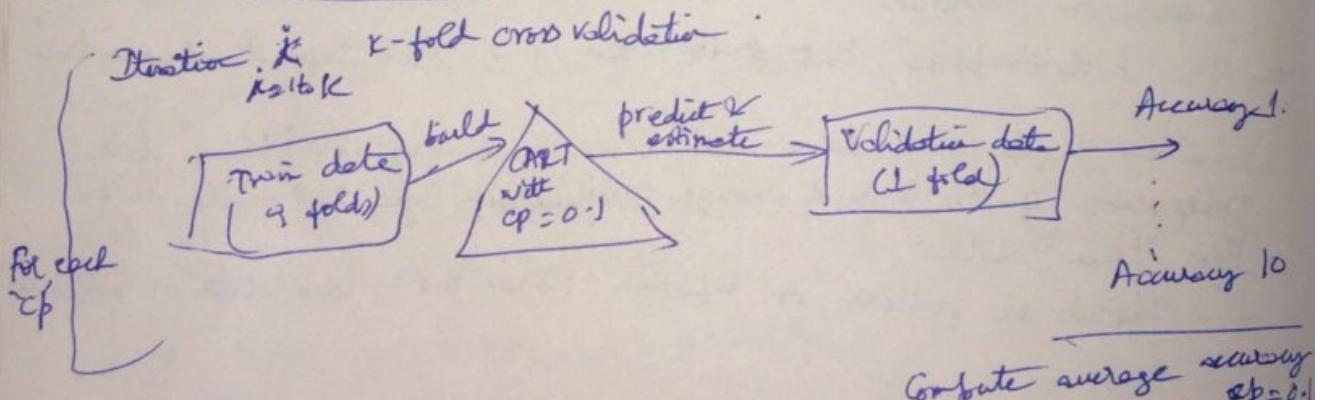
- Make sure each of the  $\Rightarrow$  K-folds is stratified.

For cross-validation method = "CV"

Bootstrap is a general-purpose tool for quantifying the uncertainty in a statistical model (getting standard errors).

- Logistic regression estimates the probability of particular outcomes; the dependent variable is categorical.

7/11/16 Parameter-tuning : CART tree



Select 'cp' with best average accuracy. Build tree on entire training + test data using that 'cp'.

Scanned by CamScanner

## Human learning evaluation : Bias & Variance

If teacher teaches problem 'P1', and asks students to solve problems P2, P3, P4, P5, etc.

Suppose  $n$  students, solves P2 - 50%, P3 - 80%, P4 - 90%, P5 - 30%.

$\Rightarrow$  high variance + low accuracy/bias

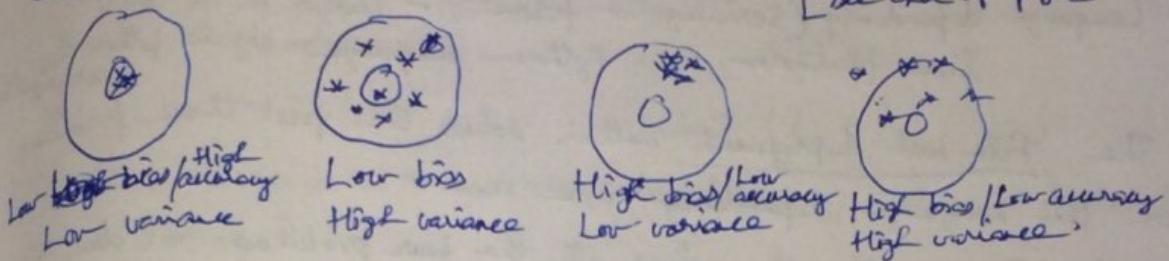
Student 1: P2 - 60%, P3 - 65%, P4 - 63%, P5 - 58%.

$\Rightarrow$  low variance + low accuracy/bias

Bias = How accurately can we solve variations of the problem?

Variance = How consistently can we solve the variations of problem?

[see end of page]



Approach 1 (Resubstitution) - Give student the same problem P1 to solve.

Approach 2 (Cross-validation) - Give student a variety of the problem P1, like P2, P3, P4, P5, ...

Bias  $\Rightarrow$  How powerful/accurate is the model?

Variance  $\Rightarrow$  How sensitive the model is to the training set?

	cp	Accuracy	Keppel
1)	0.0	0.8215699	0.687985
2)		$\downarrow$	

Q. See "tree-model & resample" to understand the variance - for each of the 10 iterations (of the best two-j parameter), we see 10 diff. accuracy values  $\Rightarrow$  variance of the model'

\* We can ask the train() method to not always choose the 'best' average accuracy, but also one with low variance. We need to write 'custom method' for that. But, by default, train() only chooses best average accuracy.

If for two diff 'cp' values, we get the same average prediction accuracy, choose one with lower variance.

train(tree-model & final-model, titanic-test, type = "class")

\* Bias = how much or on average are the predicted values different from the actual value?

Variance = how different will the prediction be for diff. samples?

Underfit  $\Rightarrow$  same result for all inputs  $\Rightarrow$  low variance & high bias

Prediction Overfit  $\Rightarrow$  high variance & low bias

## Model deployment

- └ using .RData
- └ using 'file based deployment' — most popularly used
- └ using 'Predictive model markup language' (PMML) - XML format

### Problems with .RData file based deployment

- 1) No control over .RData export — must export all objects
- 2) Platform dependence — Will .RData get loaded in a different OS?
- 3) No control over .RData import — must import all objects
- 4) Language dependency (Serialization format) — Coded in 'R' but live platform uses Python language — objects follow serialization

The 'File based deployment' method solves the first three problems. But the 'language dependency' issue remains.

The PMML approach solves all the four problems. But due to compatibility issues, is not widely used. So, file based deployment is most common.

### .RData based approach

- 1) Export objects to workspace (.RData file) using "save.image(filename)"  
- Run [Session > Save workspace] and see the command generated.
- 2) Import objects from workspace (.RData file) from another session/file using "load(filename)"  
- Run [Session > Load workspace] for the command generated.

8/11/16

PMML net # send network writing in PMML format

- rpart

We have different functions for different models in PMML.

In PMML, there are many versions for the same object type, which are incompatible with each other. Hence not used widely.

There is no way to do PMML import in R as of now.

In file based deployment, data is stored as (key, value) pairs.  
'key' is the variable/object name

How is the "classification tree" grown by 'rpart'? How do machines learn?

Pattern recognition using different strategies.

'Tree learning' algorithms

For diff attributes, machine has to calculate the "purity".  
Quantify the 'purity'.

	Sex	PClass	Survived
Passenger 1	M	1	0
2	M	2	1
3	F	1	1
4	F	2	1
5	F	3	1
			0

### Sex analysis

Male	1-0 1-1	?
Female	2-1 1-0	

### PClass analysis

class 1	1-1 1-0	- ?
class 2	2-1	
class 3	1-0	

Quantities to compute purity (most commonly used) - Based on probability

- Entropy
- Gini Index
- Misclassification rate

Understanding probabilities

frequentist / classical approach → run many times; observe # of occurrences

Bayes → Used by machine

To ~~be~~ coin once, I'll tell probability

To ~~be~~ coin the 2nd time, I'll tell its probability

9/11/16

Random  $\Rightarrow$  outcome unknown

'Bayes' view - don't conduct trials for a large # of times; not practical.

- 1) Start with 'prior probability' - can be 0.7, 0.5, etc. for coin toss
- 2) Conduct random experiment - coin toss
- 3) After experiments posterior probability - Some formula connecting prior probability & evidence

Machines learn using 'Bayes' approach

classical approach

$$\text{Probability} = \frac{\# \text{ of favorable outcomes}}{\text{Total } \# \text{ of outcomes}}$$

$$\text{Probability of A mode & face card} = \frac{13 + 12 - 3(\text{common})}{52} = \frac{22}{52}$$

selected from a deck of cards

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$$

## Conditional probabilities

Age	Female	Male
< 25 years	48	50
≥ 25 years	74	66

1) Probability of picking male =  $\frac{50+66}{50+66+48+74} = (238)$

2) Probability of picking < 25 years person =  $\frac{48+50}{238}$

3) Pick random female; probability ≥ 25 years =  $\frac{74}{48+74}$

$$P(A \text{ given } B) = \frac{P(A \text{ and } B)}{P(B)} \Rightarrow P(A \text{ and } B) = P(A|B) * P(B)$$

4) Pick random < 25 years, prob of female  
 $P(F|<25) = \frac{48}{98} = \frac{48/238}{98/238} = \frac{P(F \text{ & } <25)}{P(<25)}$

If A & B are independent,  $P(A|B) = P(A)$

$$\therefore P(A \text{ and } B) = P(A|B) P(B) \\ = P(A) P(B)$$

$$P(\text{getting head 6 times in coin toss}) = P(\text{getting head}) * P(\text{head}) * \dots * P(\text{head}) \\ = \left(\frac{1}{2}\right)^6 = \frac{1}{64}$$

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)}$$

$$P(B|A) = \frac{P(B \text{ and } A)}{P(A)}$$

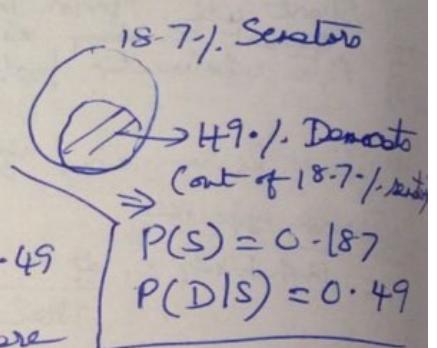
$$\Rightarrow P(A \text{ and } B) = P(A|B) P(B) \quad \left. \begin{array}{l} \text{(or)} \\ P(B|A) \end{array} \right\} \text{These are equivalent}$$

D - Democratic, S = Senator

$$P(D \text{ and } S) = P(D|S) * P(S)$$

$$P(S|D) * P(D) = 0.187 * 0.49$$

This formula not applicable here as we don't have  $P(D)$  or  $P(S|D)$ .



$$\frac{\text{Chain Rule}}{P(A \text{ and } B)} = P(A, B) = P(A|B) P(B)$$

$$P(A \text{ and } B \text{ and } C) = P(A|B|C) = P(A|B,C) P(B|C) \\ = P(A|B,C) P(B|C) P(C)$$

$$\Rightarrow P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

~~P(A)~~  
P(Cancer)

P(A) = prior probability

P(B) = from evidence / normal counts

P(B|A) = likelihood.

P(A|B) = posterior probability

This is Bayesian way of linking posterior & prior probability.  
Update our belief about A, in the light of new information B.

Cancer Diagnosis  
Cancer test 90% reliable (90% people w/ cancer get +ve test result)  
10% people w/o cancer also get +ve result)

- 2% total population have Cancer

- Suppose we have a +ve test result

- What is the probability our patient has cancer?

$$P(C=\text{yes} | T=+\text{ve}) = \frac{P(T=+\text{ve} | C=\text{yes}) * P(C=\text{yes})}{P(T=+\text{ve})}$$

= 0.155

$$10/11/16 = \frac{0.9 * 0.02}{P(T=+\text{ve})} = \frac{0.9 * 0.02}{(0.02 * 0.9) + (0.98 * 0.1)}$$

$$P(T=+\text{ve}) = P(C=\text{yes}) * P(T=+\text{ve} | C=\text{yes}) + P(C=\text{no}) * P(T=+\text{ve} | C=\text{no})$$

$$= P(C=\text{yes} \& T=+\text{ve}) + P(C=\text{no} \& T=+\text{ve})$$

$$= \frac{0.02}{0.98} + \frac{0.9}{0.1}$$

$$= P(T=+\text{ve with cancer}) + P(T=+\text{ve without cancer})$$

Writing fractions in R.

functionNone = function(...){  
{  
...  
}}

add = function(a,b)  
{  
value (a+b)  
}

% population with cancer	Test reliability		
90%	90%	78%	100%

2%	$\rightarrow 15\%$	$\rightarrow 50\%$	$\rightarrow 100\%$	Probability that patient has cancer.
66%	$\rightarrow 93\%$	$\rightarrow 98.6\%$		

both sides are heads  
↑

Is coin fair or not?

We have a coin that can be either fair or double headed.  
We had prior belief of it being fair with 0.9 probability.

- If we get heads after tails, what is  $P(\text{fair coin})$ ?
- If we get heads again after 2nd tail,  $P(\text{fair coin}) = ?$
- Does prior probability have impact on analysis?

$$P(C=Fair|H) = \frac{P(H|C=Fair)}{P(H \text{ & Fair}) + P(H \text{ & double-headed})} = \frac{P(H|C=Fair)}{P(H) * P(C=Fair|H) + P(\cancel{DH}) * P(C=Double|H)}$$

Denom =  $P(Fair) * P(H|Fair) + P(DH) * P(H)$   
Nominator =  $0.9 * 0.5 + 0.1 * 1$

### Probability Rules

Range  $0 \leq P(A) \leq 1$

Complementary probability  $P(A^c) = 1 - P(A)$

Decomposition  $P(A) = P(A \text{ and } B) + P(A \text{ and } B^c)$ .

$P(T=\text{true}) = P(T=\text{true with cancer}) + P(T=\text{true without cancer})$

If random experiment, we find "expected value" instead of mean.

Probabilities given  
for random experiments

$$\begin{array}{ccc} 10, 10, 20, 20, 30 & \xrightarrow{\text{This view}} & \frac{2}{5} * 10 + \frac{2}{5} * 20 + \frac{1}{5} * 30 \\ \text{Average} = \frac{10+10+20+20+30}{5} & & = 18 \end{array}$$

Lottery lottery works by picking 6 numbers from 1 to 49.

To play lottery, need \$1.00

If win, prize money = \$2 million

Expected value,  $E(X) = \mu = \text{"weighted average" of possible values of } X$

$$\mu = p_1x_1 + p_2x_2 + \dots + p_nx_n. \quad (X \text{ can take values } x_1, x_2, \dots, x_n)$$

$$< \sum_{i=1}^n p_i x_i$$

$$\text{Probability (lottery win)} = \binom{49}{6} = \frac{1}{C(49,6)} = 0.000\dots01$$

$$E(X) = (2 \text{ million} * 0.000\dots01) + \underbrace{(-1 * 0.9999\dots9)}_{\text{lose } \$1.00 \text{ with high probability}}$$

$$= -0.86 \text{ USD}$$

Playing once  $\Rightarrow$  expect to lose \$0.86

-ve expected value never good!

- you shouldn't play since you are expected to lose money

We have prior belief of a coin being fair at 0.9 probability. The coin could actually be 'fair' or double-headed (heads on both sides). Suppose we toss the coin 4 times and get an observation of H, H, H, T respectively. How does our prior belief (0.9=fair probability) change with each of the 4 tosses?

After 1<sup>st</sup> toss: We got 'H'

$$\begin{array}{l} \Rightarrow \\ 0.1 = \text{double-headed probability} \end{array}$$

$$P(\text{Coin} = \text{Fair} | \text{Heads}) = \frac{P(\text{Head} | \text{Coin} = \text{Fair}) * P(\text{Coin} = \text{Fair})}{P(\text{Heads})}$$

W.K.T.  $P(\text{Head} | \text{Coin} = \text{Fair}) = 0.5$ . If fair coin,  $P(\text{head}) = 0.5$ . This is  $P(\text{Coin} = \text{fair}) = 0.9$ . This is 'prior probability'. Likelihood

$$\begin{aligned} P(\text{Heads}) &= \frac{P(\text{Heads}) * P(\text{Coin Fair} | \text{Heads})}{P(\text{Heads}) * P(\text{Coin Double} | \text{Heads})} // \text{This is} \\ &\quad \text{normalization constant.} \\ &= P(\text{Heads and fair coin}) + \\ &\quad P(\text{Heads and double-headed}) \end{aligned}$$

$$P(\text{Heads and fair coin}) = P(\text{Heads}) * P(\text{fair coin} | \text{Heads}) \rightarrow \begin{array}{l} \text{We don't know} \\ \text{these values. Avoid!} \end{array}$$

$$(or) \quad P(\text{fair coin}) * P(\text{Heads} | \text{fair coin}) \rightarrow \begin{array}{l} \text{We know these} \\ \text{values. Use this.} \end{array}$$

$$[P(A \text{ and } B) = P(A)P(B|A) \text{ or } P(B)P(A|B)]$$

$$\begin{aligned} P(\text{Heads and fair coin}) &= P(\text{fair coin}) P(\text{Heads} | \text{fair coin}) \\ &= 0.9 * 0.5. \end{aligned}$$

$$P(\text{Heads and double-headed}) = P(\text{Heads}) * P(\text{double-headed} | \text{Heads}) - \begin{array}{l} \text{Avoid. We} \\ \text{don't know these} \end{array}$$

(or)

$$P(\text{double-headed}) * P(\text{Heads} | \text{double-headed}) - \text{Use.}$$

$$\begin{aligned} P(\text{Heads and double-headed}) &= P(\text{double-headed}) * P(\text{Heads} | \text{double-headed}) \\ &= 0.1 * 1 \end{aligned} \rightarrow \begin{array}{l} \text{If coin is} \\ \text{double-headed, we} \\ \text{always get 'Heads'}. \end{array}$$

$$\therefore P(\text{Coin} = \text{Fair} | \text{Heads}) = \frac{0.5 * 0.9}{(0.9 * 0.5) + (0.1 * 1)} =$$

$$= 0.8$$

R code  
Posterior = function(prior, likelihood, normalize-factor)

$$\{ \text{return } (\text{prior} * (\text{likelihood} / \text{normalize-factor})) \}$$

{

$$\text{After 1<sup>st</sup> toss, } P(\text{Coin} = \text{fair} | \text{Heads}) = \text{posterior}(0.9, 0.5, \frac{(0.9 * 0.5 + 0.1 * 1)}{0.1 * 1})$$

$$= 0.8.$$

∴ After 2<sup>nd</sup> toss, we have prior =  $P(\text{fair coin}) = 0.8$ .

$$P(\text{Coin} = \text{Fair} | \text{Heads}) = \text{posterior}(0.8, 0.5, \frac{(0.8 * 0.5 + 0.2 * 1)}{0.2 * 1})$$

$$\text{We got 'Heads' in 2nd toss.} = 0.67.$$

After 2<sup>nd</sup> toss (where we got 'Heads' again), prior =  $P(\text{fair coin}) = 0.67$   
 $\therefore P(\text{coin} = \text{fair} | \text{Heads}) = \text{posterior } (0.67, 0.5, (0.67 * 0.5 + 0.33 * 1)) = 0.5$

After 4<sup>th</sup> toss (where we got 'Tails' now), prior =  $P(\text{fair coin}) = 0.5$

$$\therefore P(\text{coin} = \text{fair} | \text{Tails}) = \text{posterior } (0.5, 0.5, (0.5 * 0.5 + 0.5 * 0)) = 1$$

$P(\text{fair coin})$      $P(\text{Tails} | \text{fair coin})$      $P(\text{Tails} | \text{double-headed coin})$

### Observations

We see how our belief in the coin's fairness reduced from 0.9 to 0.5 to 0.67 to 0.5 to 0.5. We saw only 'Heads'. But the value of 'Tails' confirmed it's a fair coin.

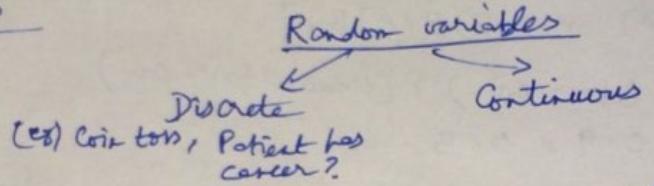
From the 5<sup>th</sup> toss onwards, even if we get 'Heads' any number of times, our belief in 'fair coin' will not change.

$$\hookrightarrow P(\text{coin} = \text{Fair} | \text{Heads}) = \text{posterior } (1, 0.5, (1 * 0.5 + 0 * 1)) = 1$$

$$\text{Also, } P(\text{coin} = \text{Fair} | \text{Tails}) = \text{posterior } (1, 0.5, (1 * 0.5 + 0 * 0)) = 1$$

These show the practicality/relevance of Bayes' theorem of probability.

11/11/16



$$\text{Addition rule: } P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$$

$$\text{Multiplication rule: } P(A \text{ and } B) = P(B|A) P(A)$$

How machines build classification tree?

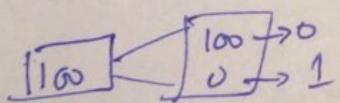
- Use 'impurity/pure' measure of nodes built by splitting data based on some attribute condition.
- Entropy  $\rightarrow$  to measure purity; we use these different methods
- Gini Index
- Misclassification error

CART algorithm uses 'Gini index'.

$$\text{Entropy of } X, E(X) = \sum_{i=1}^k -P_i \log_2 P_i \quad [ \text{"Shannon's formula"} ]$$

$$\boxed{100} \xrightarrow{\quad} \begin{cases} 50 & \rightarrow 0 \\ 50 & \rightarrow 1 \end{cases} \quad E(X) = -\frac{1}{2} \log \frac{1}{2} - \frac{1}{2} \log \frac{1}{2} = -\log \left( \frac{1}{2} \right) = \log 2 = 1$$

$$E(X) = -\log 1 = 0.$$



$$\boxed{100} \xrightarrow{\quad} \begin{cases} 100 & \rightarrow 0 \\ 0 & \rightarrow 1 \end{cases} \quad E(X) = -4/5 \log \left( \frac{4}{5} \right) - 1/5 \log \left( \frac{1}{5} \right) = 0.72$$

$$\boxed{100} \xrightarrow{\quad} \begin{cases} 80 & \rightarrow 0 \\ 20 & \rightarrow 1 \end{cases} \quad E(X) = -\frac{8}{10} \log \left( \frac{8}{10} \right) - \frac{2}{10} \log \left( \frac{2}{10} \right) = 0.47$$

$$E(x) = -\frac{99}{100} \log \left( \frac{99}{100} \right) - \frac{1}{100} \log \left( \frac{1}{100} \right) = 0.08.$$

So we see that entropy correctly measures purity.  
Classification tree - we take 'variable' which splits data to give (node)  
'minimum entropy / maximum purity'.

Why 'Decision trees'?

- Intuitive
- Faster than other methods
- Good performance. Accuracy comparable to superior to other methods.

Divide by an attribute, compute 'entropy' of each child and compute 'weighted entropy'.

$$\text{Gain (attribute)} = \text{Entropy (parent node)} - \text{Weighted entropy of child nodes.}$$

Choose attribute with 'maximum gain'.

$$\text{Weighted entropy} = \sum_{\text{child nodes}}^n E(\text{node}_i) \frac{\text{size of child}_i}{\text{size of parent node}} \quad \begin{cases} \text{where } n = \# \text{ of child nodes} \\ \text{size of child}_i \\ \text{size of parent node} \end{cases}$$

Keep dividing those child nodes which are not yet pure.

How do divide into 'continuous' variables?

$$\text{(eg) Titarie: } \begin{aligned} \text{Fare} &\leq 5 \quad \text{or} \quad \geq 5 \\ &\text{Fare} \leq 8 \quad \text{or} \quad \geq 8 \\ &\text{etc.} \end{aligned} \quad \begin{cases} \text{We try "various" divisions and} \\ \text{choose the one with best} \\ \text{gain.} \end{cases}$$

If a "continuous" variable has 'n' distinct values, we try to split with each of the 'n-2' values [we don't try min & max values]

$\Rightarrow$  Tree with "continuous" variable takes much more time than with "discrete" variables

'CP=0' grows tree deeply, but it itself does not ensure 100% purity of nodes. There are other parameters to control the tree "tree".

Very deep tree  $\Rightarrow$  learning from noise; Performs well on 'train' data but poorly on 'test' data  $\Rightarrow$  Overfitting

Two steps for growing decision tree

- 1) Growing phase  $\Rightarrow$  maximize purity of leaves
- 2) (Post) Pruning phase - minimizing the true "error" rate

split  $\rightarrow$  Recursive partitioning

## Topics with decision tree learned

- 1) Choosing the splitting criterion
  - Impurity based
  - Information gain
  - Statistical measure of association

- 2) Binary or multiway splits
  - Split parent node in 'two child nodes' or 'many child nodes'?

- 3) Finding the right sized tree
  - Pre-pruning
  - Post-pruning

Different algorithms for growing 'Decision tree':

- ID3
  - C4.5
  - C5.0
  - CART
- } Differ in the choice of 'splitting criteria', 'Number of splits' and "Pruning Strategy".

↳ classification and regression tree

CART, C4.5 are very popular. C5.0 also popular, its improvement.

C5.0 differs from C4.5 only in the way 'pruning' is done.

C4.5

CART uses 'binary' split, but C4.5 uses 'multiway' splits.

If discrete variable with 3 values, how CART does 'binary' split?

color = R, G, B (say) 'R' in one node, 'GB' in another.

So, use different combinations for binary split  $\Rightarrow$  Takes longer time.

$\Rightarrow$  CART grows deeper trees, but C4.5 grows wider trees.

Problem with multi-way split — node size smaller and has more branches in C4.5 But smaller nodes  $\Rightarrow$  more purity

Since the small nodes are pure, they become 'leaf' or 'old'. A lot of importance in the decision tree. But they may not necessarily help in 'predicting' the target variable (which is the goal of the decision tree).

Assignment 5  
 10/11/16  
 Q) 12 trucks - 4 faulty brakes. If an inspector chooses two of the trucks for brake, what's the prob. that neither one is faulty?  
 $\frac{8C_2}{12C_2} \text{ (or) } 1 - \dots = \frac{7 \times 4}{11 \times 6} = \frac{14}{33}$

2) 500 married couples. Conditional probability. Regd. their salaries.

contingency table

		W < 25K		W > 25K
		H < 25K	H > 25K	
H < 25K	W < 25K	212	36	W > 25K
	W > 25K	198	54	

$$P(W < 25) = 248/500$$

$$P(W > 25K | H > 25K) = \frac{P(W > 25K \text{ and } H > 25K)}{P(H > 25K)}$$

$$= \frac{54/500}{252/500}$$

$$P(W > 25K | H < 25K) = \frac{36}{248}$$

3) chain rule  
 $P(A \cap B) = P(A \& B) = P(A|B)P(B)$   
 $P(A, B, C) = P(A|B,C)P(B,C)P(C)$

$$P(\text{Pass 1st exam}) = 0.9$$

$$P(\text{Pass 2nd} | \text{Pass 1st}) = 0.8$$

$$P(\text{Pass 3rd} | \text{Pass 1st and 2nd}) = 0.7$$

$$P(\text{Pass all three exams}) = P(1^{\text{st}} \text{ exam}) P(2^{\text{nd}} \text{ exam}) P(3^{\text{rd}} \text{ exam})$$

$$= 0.9 \times 0.8 \times 0.7 = 0.504$$

5) Rain forecast for Mario's marriage.

$$\text{Rained only 5 days each year } (5/365) = 0.14$$

Weatherman correctly forecasts rain 90% of time

Wrongly forecasts rain 10% of time

a)  $P(\text{rain on Mario's wedding}) = P(\text{Rain forecast} = \text{rain}) + P(\text{Rain forecast} = \text{no rain})$

$$P(\text{rain} | \text{forecast} = \text{rain}) = \frac{P(\text{forecast} = \text{rain} | \text{Rain}) P(\text{rain})}{P(\text{forecast} = \text{rain})}$$

Assignment 4 - Univariate Stats

1. I. If marks increased by 5% for all students,  
 'mean' increases, 'SD' also increases.

Z-score can be calculated only on 'normal distribution'.

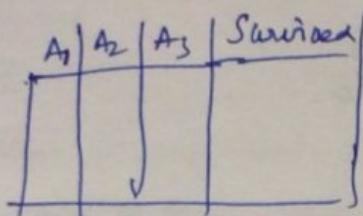
Confusion matrix - output of train method  
Gives the # of true-true, true-ve, false-ve, false-ve  
values in 'Validation' set.

		Predicted true	Predicted -ve
		Given true	
		100 cell 1	40 cell 2
(True)	Given true		
(False)	Given -ve	20 cell 3	150 cell 4

$$\text{Accuracy} = \frac{100 + 150}{100 + 150 + 20 + 40}$$

tree model / resampled CM  $\rightarrow$  cell 1 cell 2 cell 3 cell 4  
100 40 20 150

15/11/15



Recursive partitions

Calculate the information gain (reduction in impurity) using each of the 3 attributes  $A_1, A_2, A_3$  and use the attribute that has max information gain.

Information gain = Entropy(original node) - Entropy(split attribute using attribute)

'Entropy' measures 'impurity'.

We do not grow classification trees to perfection (all pure nodes) to avoid 'overfitting'.

'Gini Index' is another measure of purity (just like 'Entropy').

$$1 - \sum_i p_i^2 = \text{Gini} = \frac{\sum_i p_i(1-p_i)}{\sum_i p_i} = \frac{\sum_i p_i(1-p_i)}{1} = 1 - \sum_i p_i$$

$10 \text{ Survivors}$ $0 \text{ non-survivors}$  $8 - 2$ Survival Died  $5 - 5$ Survived Died  $3 - 7$	$\text{Entropy} = 1 \log 1 - 0 \log 0 = 0$  $E = -\frac{8}{10} \log \left(\frac{8}{10}\right) - \frac{2}{10} \log \left(\frac{2}{10}\right) = 0.72$  $E = -\frac{5}{10} \log \left(\frac{5}{10}\right) - \frac{5}{10} \log \left(\frac{5}{10}\right) = 1$  $E = -\frac{3}{10} \log \left(\frac{3}{10}\right) - \frac{7}{10} \log \left(\frac{7}{10}\right) = 0.88$	$\frac{8}{10} \times \frac{2}{10} = 0.16$  $\frac{5}{10} \times \frac{5}{10} = 0.25$  $\frac{3}{10} \times \frac{7}{10} = 0.21$
--	--	---

(e.g.)  $(0.1, 0.9) \dots (0.3, 0.7)$   
For 2-valued nodes

$\Rightarrow \text{Entropy} \in [0, 1]$

[Info for 2-valued nodes, 0.25]  
Gini Index  $\in [0, 1]$

These have diff. scales but same meaning

$$Gini(i) = \sum_v p(v) \left( \sum_{j \neq i} p(j|v) k(j|v) \right) = \sum_v p(v) \sum_j p(j|v) p(j|i)$$

Gini gain = Gini(original node) - Gini(split)

CART uses 'Gini gain' for splitting criteria.

C4.5 - categorical attributes - multi-way cutting  
continuous attributes - binary cutting.

CART: 'CP' :: C4.5: 'C'

cost-complexity pruning (based on validation set)

$$R_C = MC + \alpha |T| \quad \text{For CART, } \alpha = 'cp'$$

for each 'CP' value,

there are many iterations - each one has a diff validation set.  
Do pruning inside each iteration.

C4.5 uses 'Error based pruning' (without using validation set)

Why CART uses 'greedy approach' in pruning (using cost-complexity)?

- Machine time is costly. So, OK with greedy for now.

Pre-pruning: Control tree growth while growing, no late pruning

- Min. size of nodes to be split. (minsplit) - controlled usually
- Min. instances in leaves - while splitting, how many min. entries in a leaf? - not controlled
- Tree depth. (max depth) - controlled usually.

split control - minsplit, maxdepth  
 $\leq 20 \quad = 30$

④ In both CART & C4.5, we always do only 'binary' split for "continuous" attributes. This is to avoid the complexity.  
But for categorical attributes,  $CART \rightarrow$  binary split;  $C4.5 \rightarrow$  multiway split

⑤ Both the Shannon entropy (used in C4.5) and Gini index (used in CART) are biased towards splitting with attributes that have a large number of values - because they both result in large information gain.

But C4.5 and CART use diff approaches to solve this 'issue':

- C4.5 uses 'Gain ratio' instead of 'Gain'. Gain ratio =  $\frac{\text{Gain}}{\text{Entropy(attribute)}}$

- CART solves the bias by always doing only a 2-way split.

⑥ Is Gini Index =  $1 - \sum_i p(i)^2$  ✓ (or)  $\sum_{i \neq j} p(i)p(j) ? \times$   
 ~~$\sum_{i \neq j} p(i)p(j) ? \times$~~

16/11/16 Post-pruning in CART - cost complexity pruning

Model overfitting - model discovers irrelevant patterns due to noise in train data.

Underfitting - model does not discover relevant patterns.

How to control overfitting in decision trees?

Pre-pruning is very hard because no one knows where to stop. So, post-pruning is best.

$$\text{Misclassification cost (subtree)} = \sum_{\text{leaves } j} \sum_{x_i \in \text{leaf}_j} I(y_i \neq \text{leaf}'s \text{ class})$$

$y_i$  = actual value of  $x_i$

leaf's class = does the leaf say 'survived' or 'not survived'.

→ This alone is not sufficient because larger trees typically have low misclassification and a large number of leaves.

- So need another tree with smaller # leaves while allowing misclassification cost to rise a lot.

$$\Rightarrow \text{Cost (subtree)} = \sum_{\text{leaves } j} \sum_{x_i \in \text{leaf}_j} I(y_i \neq \text{leaf}'s \text{ class}) + \alpha (\# \text{ leaves})$$

For CART,  $\alpha$  is 'cp'.

If  $\alpha = cp = 0 \Rightarrow$  I am asking for deepest possible tree.

When  $cp \uparrow$ , # leaves ↓

When  $\alpha \rightarrow \infty$ , tree will have only the 'root node'.

$$\text{In CART, cost (subtree)} = \sum_{\text{leaves } j} \sum_{x_i \in \text{leaf}_j} \text{probability} (y_i \neq \text{leaf}'s \text{ class}) + cp (\# \text{ leaves})$$

Due to using probability, maximum misclassification cost = 1 instead of 1. so,  $cp \in [0, 1] \quad cp \in [0, \frac{1}{k}]$   $k = \# \text{ values in target attr}$

In cost (subtree), why (# leaves in subtree) & not (# leaves in whole tree)?

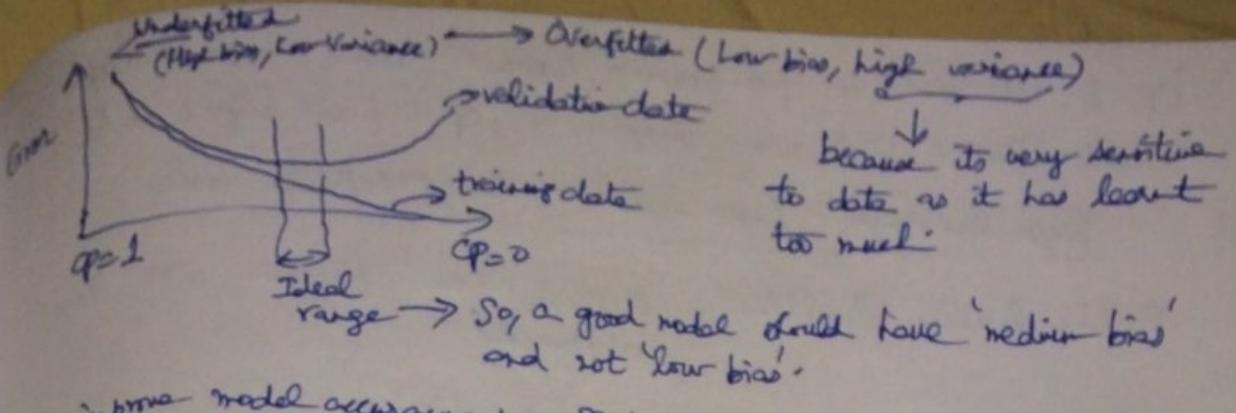
Our measure of complexity is "# leaves", so this is enough. This does not see leaves.

Even with slight ↑ in 'cp', tree becomes less complex and accuracy goes down. Need correct 'cp' to balance overfitting & underfitting.

If there is a 'lot of noise' in train data, use higher 'cp' value to avoid overfitting.

In CART post-pruning, we always collapse '2 leaf nodes' at a time because CART uses binary splits.

Post-pruning in C4.5 does not use 'validation data' - it needs probability distribution math.



To improve model accuracy from 80% to 90%.

- Need more data
  - Need better learning algorithm
  - Need better 'feature engineering'.
- ? TRY these two first. If it's not enough, finally ask for more data.

If we have 'more train data', do we need greater fine-tuning of Cp?

- Not necessarily. Depends on whether there are more relevant patterns in the 'extra' train data.

$$\text{CART cost (multtree)} = \sum_{\text{leaves}} \sum_{x_i \in \text{leaf}} \text{'probability of } x_i \text{' } (y_i \neq \text{leaf's class}) + \alpha (\# \text{leaves in subtree})$$

If a tree has 200 nodes, and a leaf has 25 nodes. 15 correct prediction  
So, error of leaf =  $\frac{10}{200}$ . 10 wrong prediction

↳ If all 200 nodes have 'wrong' prediction, will we have

$$\sum_{\text{leaves}} \sum_{x_i \in \text{leaf}} \text{'prob. of } x_i \text{' } (y_i \neq \text{leaf's class})$$

to be 1 only.

$$\text{CART cost (subtree)} = \text{"formula"} + (Cp * (\# \text{leaves in subtree}))$$

$[0, 1]$  value range

So, when  $Cp = 1$ , even a tree with 2 leaves will have a cost of  $\geq 2$ ; but a tree with only 1 leaf (just root node) will have a cost  $\leq 2$ . Post-pruning minimizes cost (tree).

So, when  $Cp = 1$ , we will always have tree with just 1 node.

16/11/16

### Randomizers

Cross-validation uses 'randomisation' during stratification.

$$X_{i+1} = (aX_i + b) \% m$$

↳ This sequence looks 'random'.

$$X_0 = \text{seed} \cdot \# \text{set} \cdot \text{seed} ("X_0 \text{ value}")$$

Pseudo-random generation

To make "cross-validation" look the same, use the same "randomizers".

'cp' is not used in pre-pruning because train() uses multiple values. This is true even though if "cp = 1  $\Rightarrow$  only 1 node".

Bayes learning  $\rightarrow$  Bayes model - uses probability tables

Machine learning algorithm - Replace 'tree' approach with 'Bayes' approach.

$$\left\{ \begin{array}{l} P(S=0 \mid \text{Gender}=\text{Male}, \text{Pclass}=1, \text{Fare}=120) = 0.3 \\ P(S=1 \mid \text{Gender}=\text{Male}, \text{Pclass}=1, \text{Fare}=120) = 0.7 \rightarrow \text{We predict 'Survived'.} \\ \text{Instead of 'trees', use 'conditional probability' for predicting} \\ \rightarrow = \frac{P(\text{Male, class 1, Fare } 120 \mid S=0) * P(S=0)}{P(\text{Gender}=\text{Male, Pclass}=1, \text{Fare}=120)} \quad \left. \begin{array}{l} \text{Interested} \\ \text{Know which} \\ \text{bigger} \cdot \text{So,} \\ \text{compose only} \\ \text{Numerator.} \end{array} \right\} \\ \rightarrow \frac{P(\text{Male, class 1, Fare } 120 \mid S=1) * P(S=1)}{P(\text{Male, class 1, Fare } 120)} \end{array} \right.$$

So, for relative comparison, we only likelihood & prior probability

$$\left. \begin{array}{l} P(S=0) = ? \\ P(S=1) = ? \end{array} \right\} \text{Know from 'Train' data the prior probabilities}$$

$$P(\text{Male} \& \text{class 1} \& \text{Fare } 120) = P(\text{Male}) P(\text{class 1}) P(\text{Fare } 120) \quad \left. \begin{array}{l} \text{"if" all are} \\ \text{independent} \end{array} \right\}$$

$$P(\text{Male} \& \text{class 1} \& \text{Fare } 120 \mid S=0) = P(\text{Male} \mid S=0) P(\text{class 1} \mid S=0) P(\text{Fare } 120 \mid S=0)$$

Why for "S=1"

$\downarrow$   
Need 'conditional probability table'.

Machines computing the probability table and learning from it is called 'Bayes learning'.

Not finding directly from 'train data' as it takes a long time.

The 'conditional probability table' is a group of tables, one for each attribute (against target attribute)

18/11/16 classification problem revisited

	<u>Outlook</u>	<u>Temperature</u>	<u>Humidity</u>	<u>Wind</u>	<u>predict</u>	<u>Play Tennis?</u>
(1)	Sunny	Hot	High	Light	No	
	Overcast	Hot	High	Light	Yes	

## Naive Bayes model

Prior probabilities

Conditional probability tables → Machine learnt directly from data

Outlook

$$P(\text{Sunny} | p) = \frac{2}{9}, P(\text{Sunny} | n) = \frac{3}{14}, P(\text{play} = \text{Yes}) = \frac{5}{14}$$

$$P(\text{overcast} | p) = \frac{4}{9}, P(\text{overcast} | n) = \frac{3}{5}$$

$$P(\text{rain} | p) = \frac{3}{9}, P(\text{rain} | n) = \frac{2}{5}$$

for Temperature, Humidity, Wind we've diff. tables

$$P(\text{High} | p) = \frac{3}{9}, P(\text{High} | n) = \frac{4}{5}$$

$$P(\text{Normal} | p) = \frac{6}{9}, P(\text{Normal} | n) = \frac{2}{5}$$

$$(a) P(\text{play}^*) < \text{rain, hot, high, light} >$$

$$\hookrightarrow P(\text{rain, hot, high, light} | \text{play}) \times P(\text{play} = \text{Yes})$$

$$P(\text{rain, hot, high, light})$$

Compute only numerator to say 'Yes' or 'No'. To find probability, compute denominator also.

Naive Bayes' assumes that conditions are independent. So,

$$P(\text{rain} | p) P(\text{hot} | p) P(\text{high} | p) P(\text{light} | p) \approx P(\text{play})$$

[?], <, >, = [find whether the above or the below is greater]

$$P(\text{rain} | n) P(\text{hot} | n) P(\text{high} | n) P(\text{light} | n) \approx P(\text{no play})$$

The greater of these two denotes whether to play or not.

In the conditional probability tables, for 'continuous' variables, we have only 2 columns, [1] and [2].

$$\text{Denominator} = P(\text{rain, hot, high, light}) = P(\text{rain, hot, high, light} | \text{play} = \text{Yes}),$$

$$[P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\bar{A})P(\bar{A})}] \quad P(\text{rain, hot, high, light} | \text{play} = \text{No})$$

If Bayes' rule, if we ignore the 'Denominator' (estimate) - This approach is called "Maximum A Posteriori" (MAP) approach.

If in Bayes' rule, if ever the 'Prior probabilities' are same along with the 'estimate', we use only the "Likelihood". This is called "Maximum Likelihood" hypothesis (ML).

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

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

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

$$P(\text{rain, hot, high, light}) = P(\text{rain, hot, high, light} | \text{play}) \times P(\text{play} = \text{Yes}) = \text{Numerator}$$

$$P(\text{rain, hot, high, light} | \text{no play}) = P(\text{rain, hot, high, light} | \text{play})$$

ML hypothesis.  $P(h_i) = P(h_i) + \delta_i$

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} P(h | D)$$
$$= \frac{P(D|h)P(h)}{P(D)}$$
$$= \frac{P(D|h)}{P(D)}$$

How to find 'probability' of continuous variable values? Need to use 'Naive Bayes' approach

Ages - 20, 23, 24, 27, 35, 32, 21, ...  
 $P(\text{Age} = 25.5?)$ . Split 'range' in intervals

20-25  
25-30  
30-35  
35-40

$$P(\text{Age} = 25.5) = \frac{\# \text{Passengers in } (25-30)}{\# \text{Passengers}} \rightarrow \begin{matrix} \text{How to decide} \\ \text{interval?} \end{matrix}$$

This is static clustering. We can try 'dynamic' clustering based on closeness.

Can't we z-score to compare against 'general population' statistics because 'titanic' population may not be normally distributed - perhaps all titanic passengers are in a particular age range?

Gauss's solution uses 'mean' and 'sd'. - Needs 'normal distribution'

$$P(X_j | C = C_i) = \frac{1}{\sqrt{2\pi\sigma_{ji}^2}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

$\mu_{ji}$ : mean of feature values  $X_j$  of examples for which  $C = C_i$

$$\sigma_{ji} = \text{std. } \sigma$$

21/11/16 Naive Bayes conditional probability tables

	[1]	[2]
0	22.11789	31.38821
1	$48.39541$ Mean $\mu$	$66.597$ Std-deviation $\sigma$

$$P(S=0 | P_{\text{class}}=3, \text{fae}=12.5, \text{Sex}=\text{Male}, \text{Embarked}=Q)$$

$$P(P_{\text{class}}=3 | S=0) P(\text{fae}=12.5 | S=0) P(\text{Sex}=\text{Male} | S=0) P(\text{Embarked}=Q | S=0)$$

Need to capture

Use the conditional probability table

If probability = 0 in conditional table, final probability will be 0. So, ensure non-zero by adding 1. Increase weights of all entries  $\rightarrow$  add '1' at numerator & add 'N' at denominator  
 $\rightarrow$  Laplace smoothing

In Naive Bayes, 'FL' parameter = Laplace corrective. Other parameters are 'external' & 'adjust'.  $fl \in [0, 1]$  but we mostly use '1'.  
- recommended by Laplace  
'external' = FALSE  $\Rightarrow$  assume 'normal distribution' and use Gauss equation  
'external' = TRUE  $\Rightarrow$  split 'continuous' data into bins.  
'adjust'  $\Rightarrow$  How much of 'bin size' to use? Makes sense only when external = TRUE.

If there is missing data in 'bining data', NB ignores it for computing probability.

- Advantages of 'tree approach' vs 'Naive Bayes/other learners'
- No formal distributional assumptions. In NB, we should fit distribution is normal or not
  - Automatic variable selection  
(Top-level variables more important: more gain)
  - Handle missing value
  - Easy to interpret if tree is small
  - The terminal (leaf) nodes suggest a natural clustering.

#### Disadvantages

- Other methods can have better 'Accuracy'.
- Instability - change data a bit & tree changes a lot. Can be offset a bit by post-pruning.

'Naive Bayes' is best used in "Text analytics" & 'spam detection' because the word columns are considered independent.

#### Advantages of NB

- Much faster to compute tables
- Competitive performance.
- Good candidate of a base learner in ensemble learning

#### Disadv. of NB

- Poor performance when attributes are 'redundant' / have high correlation.
- Need to remove a redundant attribute manually.

To find 'probability' of continuous variable, 2 ways

Kernel based

'bin' based - split data  
in bins/ranges; find count in  
each bin.

$$N=14 \quad P(\text{play}) = 9/14, P(\text{no play}) = 5/14$$

Conditional probability tables

Distribution based

\ Need to know the distributions

Covered  
topic

This is wrong.

Outlook

$$\begin{aligned} P(\text{sunny} | p) &= 2/9 & P(\text{sunny} | n) &= 3/5 \\ P(\text{overcast} | p) &= 4/9 & P(\text{overcast} | n) &= 0/5 \\ P(\text{rain} | p) &= 3/9 & P(\text{rain} | n) &= 2/5 \end{aligned}$$

Temperature

$$\begin{aligned} P(\text{hot} | p) &= 2/9 & P(\text{hot} | n) &= 2/5 \\ P(\text{mild} | p) &= 4/9 & P(\text{mild} | n) &= 3/5 \\ P(\text{cold} | p) &= 3/9 & P(\text{cold} | n) &= 1/5 \end{aligned}$$

Humidity

...

Outlook

~~$$\begin{aligned} P(\text{sunny} | p) &= 3/23 & P(\text{sunny} | n) &= 4/19 \\ P(\text{overcast} | p) &= 5/23 & P(\text{overcast} | n) &= 1/19 \\ P(\text{rain} | p) &= 4/23 & P(\text{rain} | n) &= 3/19 \end{aligned}$$~~

Temperature

~~$$\begin{aligned} P(\text{hot} | p) &= 3/23 & P(\text{hot} | n) &= 3/19 \\ P(\text{mild} | p) &= 5/23 & P(\text{mild} | n) &= 3/19 \\ P(\text{cold} | p) &= 4/23 & P(\text{cold} | n) &= 2/19 \end{aligned}$$~~

Humidity

(WRONG)

Apply Laplace correction to eliminate '0' probability.

Probability of  $\frac{a}{b} \rightarrow \frac{a+1}{b+N}$ . This is to be applied all across conditional probability tables.

In general,  $\frac{a}{b} \xrightarrow{\text{Correction}} \frac{a+c}{b+cN}$ . Laplace recommends "c=1".

Laplace correction changes probabilities; so can't be used to estimate probability. It can only be used to predict outcome like 'play' or 'no play' because the 'relative probability' is not supposed to change.

Issue:  $\frac{2}{9} < \frac{2}{5}$   
 $\frac{4}{23} > \frac{3}{19}$

↓ correction (Laplace)      Relative prob. changed. Some issue with our above way of correction.

Note: Naive Bayes can be used only for "classification" and never for regression. Even when it predicts 'probability', it is the probability of 'categorical' variables only.

Laplace correction: Make sense only when data set is large.  
 Suppose for class 'no play'; sunny = 990, overcast = 0, rain = 10 in 1000 samples. Probabilities =  $990/1000, 0/1000, 10/1000$ .

Now for "no play", we add a count of 1 to each attribute value.  
 Add 3 rows/samples where 1 entry for each of sunny, overcast, rain  
 so probabilities are  $991/1003, 1/1003, 11/1003$ .

After Laplace correction

$$P(\text{play}) = \frac{10}{16}, P(\text{no play}) = \frac{6}{16} \quad [\text{Adding } 2 \text{ to } 1 \text{ play}]$$

$$= \frac{9+1}{14+2} = \frac{5+1}{14+2} \quad // \text{Correction usually NOT done}$$

for this; this is "apriori".

Outlook # other ~~probabilities~~ likelihoods also change

$$P(\text{sunny} | p) = \frac{3}{12} = \frac{2+1}{9+3} \quad P(\text{sunny} | n) = \frac{4}{8} = \frac{3+1}{5+3}$$

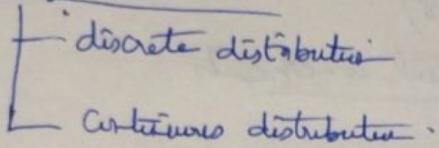
$$P(\text{overcast} | p) = \frac{5}{12} \quad P(\text{overcast} | n) = \frac{1}{8} \quad \begin{array}{l} \text{Adding 1 sample for each} \\ \text{of sunny, overcast, rain} \end{array}$$

$$P(\text{rain} | p) = \frac{4}{12} \quad P(\text{rain} | n) = \frac{3}{8} \quad \text{Temperature, Humidity also get corrected}$$

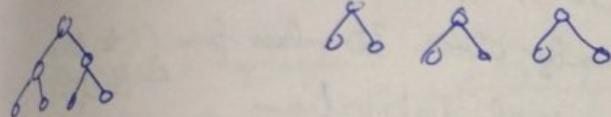
In general, to make Laplace correction, we add an initial count of 1 to the total of all instances with a given attribute value, and we add the number of distinct values of the same attribute to the total number of instances in the group.

22/11/16

### Probability distributions

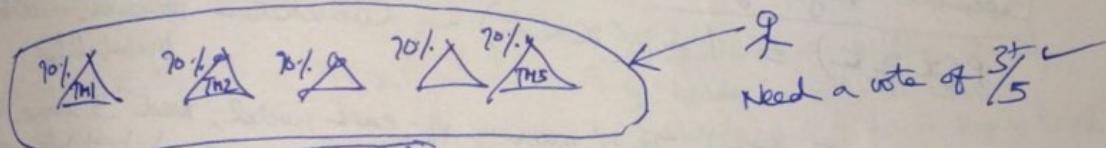


Single tree  $\rightarrow$  Multiple trees (Forest). Is this better?



Why have forests instead of a single tree?

Suppose I use 5 classifiers which classify a point correctly 70% of the time. The classifiers are fully independent and I take majority vote. How often is the majority vote correct for that point?



Will majority vote  $> 70\%$ . or not?

1) 5 coin tosses. What is the probability that 3 heads will show up?

$$= \frac{5C_3}{2^5} \quad (\text{or}) \quad 5C_3 \left(\frac{1}{2}\right)^5 \rightarrow \text{task of } 5C_3 \text{ less } \left(\frac{1}{2}\right)^5 \text{ probability}$$

$$= \text{dbinom}(3, 5, 0.5)$$

2) What is the prob. that 2 heads will show up in 5 coin tosses if

$$P(\text{head}) = 0.8 ?$$

$$5C_2 (0.8)^2 (0.2)^3 = \text{dbinom}(2, 5, 0.8)$$

This kind of thought about yes/no distribution is "Binomial distribution".

- "n" experiments, independent of each other; only 2 possible outcomes
- $p$  (# one outcome)  $\Rightarrow 1 - p$  (of other outcome)

$$B(n, p, k) = \binom{n}{k} p^k (1-p)^{n-k} = nC_k p^k (1-p)^{n-k}$$

prob. of head =  $P(X=k)$  [Works only for 'discrete' distributions]

$\hookrightarrow dbinom(2, 5, 0.8)$

Such diff. patterns are called distributions.

- 8 important patterns.

3) What is prob. that 'atmost 2 heads' show up in 5 coin tosses when  $p(\text{heads}) = 0.8$ ? [Cumulative probability]

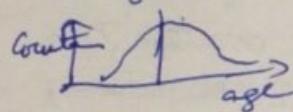
$$= B(5, 0.8, 0) + B(5, 0.8, 1) + B(5, 0.8, 2).$$

$$P(X \leq k) = \sum_{i=0}^k dbinom(2, 5, 0.8) = dbinom(0, 5, 0.8) + dbinom(1, 5, 0.8) + dbinom(2, 5, 0.8)$$

Generate random number from binomial distribution/sequence  
 $\rightarrow rbinom(4, 10, 0.8)$ .

Pascal Delta - binomial coefficients

	1	2	1	-	$2C_0, 2C_1, 2C_2$	
1	3	3	1	-	$3C_0, 3C_1, 3C_2, 3C_3$	
1	4	6	4	1	$4C_0, 4C_1, 4C_2, 4C_3, 4C_4$	
1	5	10	10	5	1	$5C_0, 5C_1, 5C_2, 5C_3, 5C_4, 5C_5$

To generate 30 'ages'  $\rightarrow$  Randomly choose 30 values from (1 to 30).  
  
 Generate 'ages' in normal distribution with mean '25' and more variations.

Ensemble-majority vote

$$P(X \geq k) = 1 - P(X < k) - \text{Cumulative binomial probability}$$

Ensemble - 5 trees, 70% accuracy of each model, need '3' same for majority vote.  
 $n=5, K=3$ .  $P(X \geq 3) = 1 - P(X \leq 2) =$

$$1 - rbinom(2, 5, 0.7)$$

How to make models independent in ensemble? Random forest

Question  
 Ensemble of 5 trees/models, each being independent. Each model has 70% accuracy. We use majority vote to decide outcome. What is the accuracy of this ensemble?

This is equivalent to : 5 coin tosses.  $p(\text{Heads})$  of each coin = 0.7. We say 'Heads' for this ensemble of 5 coin tosses if we get 'Heads' in at least 3 out of the 5 tosses.  $P(\text{Heads of ensemble}) = ?$

$$\Rightarrow P(\text{Heads of ensemble}) = P(3+ \text{Heads}) = 1 - P(\leq 2 \text{ Heads})$$

$$= P(\text{Accuracy of ensemble}) = 1 - rbinom(2, 5, 0.7)$$

$$= 0.837$$

See how accuracy increased from 70% to 83.7%, with 5 models.

$$\begin{aligned} 1 - \text{binom}(500, 1001, 0.7) &= 1 \\ 1 - \text{binom}(1000, 2001, 0.7) &= 1 \end{aligned} \quad \left. \begin{array}{l} \text{Since 100% accuracy achieved with} \\ 500 \text{ models, no need to build a} \\ \text{large ensemble.} \end{array} \right.$$

$$1 - \text{binom}(50, 101, 0.7) = 0.9999871 \Rightarrow 101 \text{ models are sufficient!}$$

The real challenge is "How to build completely independent models?"  
"Bagging" idea

Discrete prob distrib patterns

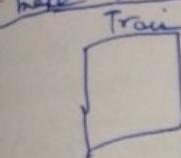
- Binomial, Poisson, negative binomial, geometric

Continuous prob distrib patterns

- Normal, Gamma, Exponential, Beta, Lognormal, Chi-square, Fisher's F, Student's T

Cumulative probability,  $\text{pbinom}()$  used for p-value

How to make models in "ensemble" independent? We've same base data only.



63%, #30

63%, #30

63%, #30

63%, #30

63%, #30

Random fraction of train  
Give 63%, random  
fraction of train to  
each model.

Also, give only some 30%  
of columns/features to  
each model

- This idea "Random forest".

'Aggregation' means 'majority vote'.

Bootstrap = resampling with replacement

$\text{dbinom}(x, n, p) = nCx p^x (1-p)^{n-x}$  = Probability of getting exactly  
 $x$  heads in  $n$  trials where  
 $p(\text{heads}) = 0.9$  for a single trial.

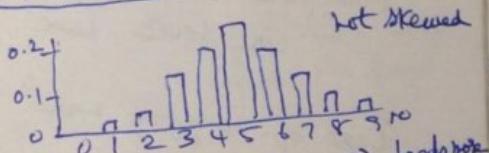
Binomial distribution = function  $(n, p)$

{  $x = 0 : n$

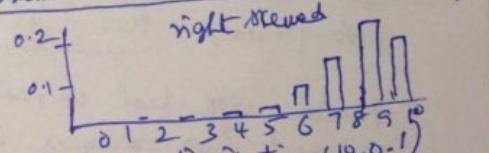
$b = \text{dbinom}(x, n, p)$

} barplot(b, names.arg=x)

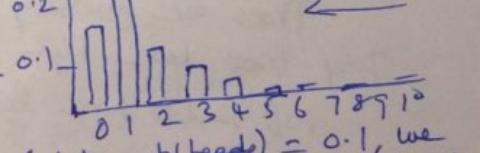
Binomial distribution  $(10, 0.5)$



binomial distribution  $(10, 0.9)$  likely



binomial distribution  $(10, 0.1)$



$\text{rbinom}(4, 10, 0.8)$  # 4 random #s in [0, 10] in binomial distribution with  $p = 0.8$

> 6, 9, 8, 10  
8, 8, 8, 5  
9, 7, 8, 9 } Pick #s mostly in [8, 10] range as they have highest probability as per right-skewed graph

$\text{rbinom}(4, 10, 0.5)$  # mostly in [3, 7] range

> 4, 7, 3, 6, 7, 5, 3, 4

$\text{rbinom}(4, 10, 0.2)$  # mostly in [1, 3] range

> 4, 3, 4, 3, 2, 0, 2, 1

Obviously,  $\text{rbinom}()$  makes sense only if replace = TRUE. So, it's not an option

#9

### Strength of bootstrap

When we do re-sampling with replacement and pick 'n' random values from a training data set of size 'n', the 'mean' of the bootstrapped sample is very close to the mean of the entire training data, and also has less variance from the entire training data.

But the 'deduped' bootstrap, although also close to mean, has much more variance.

3

We can empirically observe that the # of unique entries in the bootstrapped sample (of size n)  $\approx 0.632n$  [ $= (1 - \frac{1}{n})^n$ ]. So, 63% is unique. The remaining 37% is the "out-of-bag" data used for validation.

$$n = 100000$$

bsample = sample(1:n, n, replace=TRUE)

$$\text{length}(\text{unique}(bsample)) / n \approx 0.63155.$$

$$\text{population} = 1:10000$$

$$\text{mean}(\text{population}) \approx 5000.5$$

$$\text{mean}(\text{sample}(\text{population}, \text{length}(\text{population}), \text{replace}=TRUE))$$

↳ This is closer to 'population mean' and has less variance.

$$\text{mean}(\text{sample}(\text{population}, 0.63 * \text{length}(\text{population}), \text{replace}=FALSE))$$

↳

24/11/16

Ques

CART and C4.5 are both good. Need to experiment which is better for a particular problem. C4.5 does NOT use validation data for post-pruning - does a great job in postpruning. C4.5 widely used by Google but not so popular in open-source community.

Combining models into forest. How?

- Bagging
- Bagging with random features
- Boosting

Homogeneous vs Heterogeneous models

- Can forests have NB-model, CART & C4.5 models?

Bias of bagging model  $\ll$  Bias of underlying model. Bagging also reduces variance as it dilutes the noise. No need for post-pruning. Must not do postpruning. Noise spread across nodes.

Models with high variance are good candidates to 'bag' into a forest, so variance is ~~not~~ reduced.

Trees are generally bagged as they have more variance. Deep trees have high variance and hence good. NB models have lesser variance and are generally not bagged.

Why NB not tryed to improve accuracy?  
 Hard to have 'independent' NB models. Can we 'random feature selection' for independence, but it doesn't make logical sense as the method multiplies probabilities.  
 In the "treebag" approach, we use bootstrap to build multiple trees. All the trees are built using the same set of predictor variables. So the models are not independent. Using majority vote on the forest thus does not improve the accuracy. Hence, 'treebag' is NOT used in practice.  
 bag-tree model = train(titanic\_train[, c("Sex", "Pclass", "Fare")],  
 titanic\_train\$Survived, method = "treebag", trControl = resamplingStrategy)  
 # resamplingStrategy = trainControl(method = "boot", number = 10)  
 bag-tree model & finalModel\$ntrees[5] \$index # indexes of the 63% selected  
 " " & btree # fully grown/deep tree

Instead of treebag, we used 'Random Forest'. Here, the individual trees are grown using different sets of attributes so that they have some independence from each other.

rf-grid = expand.grid(.mtry = c(2, 3))  
 rf-model = train(titanic\_train[, c("Sex", "Pclass", "Parch", "SibSp")],  
 titanic\_train\$Survived, method = "rf", tuneGrid = rf-grid, ntree = 100,  
 trControl = resamplingStrategy)

rf-model # mtry is the # of columns tried.

rf-model & finalModel # 100 trees grown in the forest

getTree(rf-model & finalModel, 1, labelVar = TRUE)

Tree is not displayed in CART's format

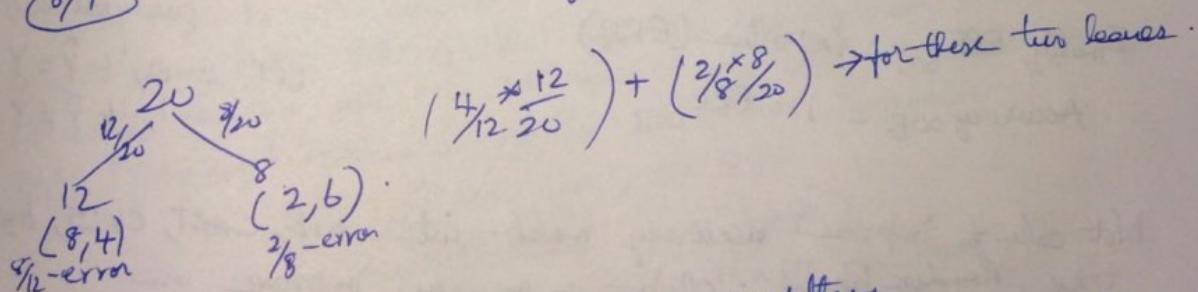
for each 'mtry' value, we build 100 trees and compute overall accuracy.

Random forests cannot handle missing data

5/11/16 Part pruning in CART

cost(subtree) =  $\sum_{\text{leaves } j}$  misclassification error at node  $j$  \* proportion of data instances reached to node  $j$ .

6-noded, 4-nor mixed  
 nodeclass = mixed  
 (6,4)  $\Rightarrow$  misclassification error of this node =  $4/10$ .  
 always  $\downarrow$  if all leaves are pure overfitting!



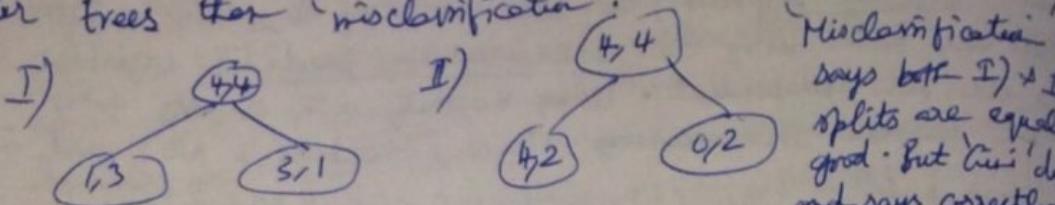
Modified above formula to overcome overfitting

cost(subtree) =  $\sum_{\text{leaves } j} \left( \frac{\text{misclassification error at node } j}{\text{proportion of data instances reached to node } j} \right) + \lambda (\# \text{leaves})$

Only one node in tree

$$20(12,8) \rightarrow \text{GSE} = \frac{8}{20} \times \left(\frac{20}{20}\right) + 0 \times 1$$

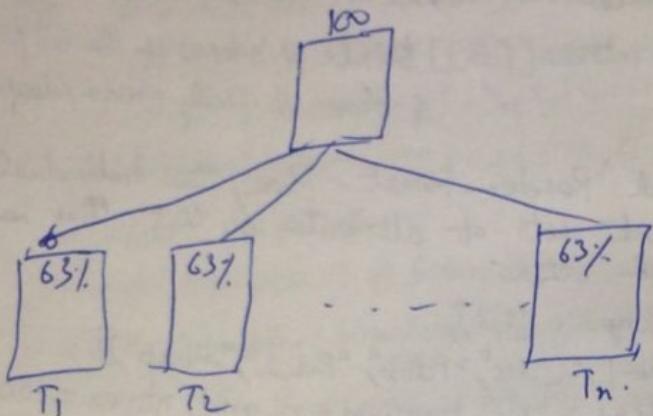
To build CART tree, why use 'impurity' instead of 'misclassification rate'? Examples found where 'impurity' gives better trees than 'misclassification'.



'Misclassification' says both I) & II) splits are equally good. But 'Gini' diff and says correctly, II) is better

28/11/16

### Evaluation of bagged models



Validation by 'out-of-bag'

A given sample in data set of 100 samples will be out-of-bag in 37% of models.

$$P(\text{sample is OOB}) = 0.37$$

$$P(\text{sample is in a bag}) = 0.63$$

So, for each sample in total given data, evaluate it in each of the 37% models, get the predictions, and use majority vote. We already know the outcome of each sample. So, Compute 'accuracy'.

OOB Error Estimate

	M <sub>1</sub>	M <sub>2</sub>	M <sub>3</sub>	M <sub>4</sub>	M <sub>5</sub>	$\hat{Y}$	Y	ERR
1	.	+	.	+	-	+	+	0
2	.	-	*	-	-	-	-	0
3	+	.	-	-	-	-	+	1

\* = OOB

+ = prediction is true

- = prediction is -ve

$\hat{Y}$  = Majority vote across +s and -s

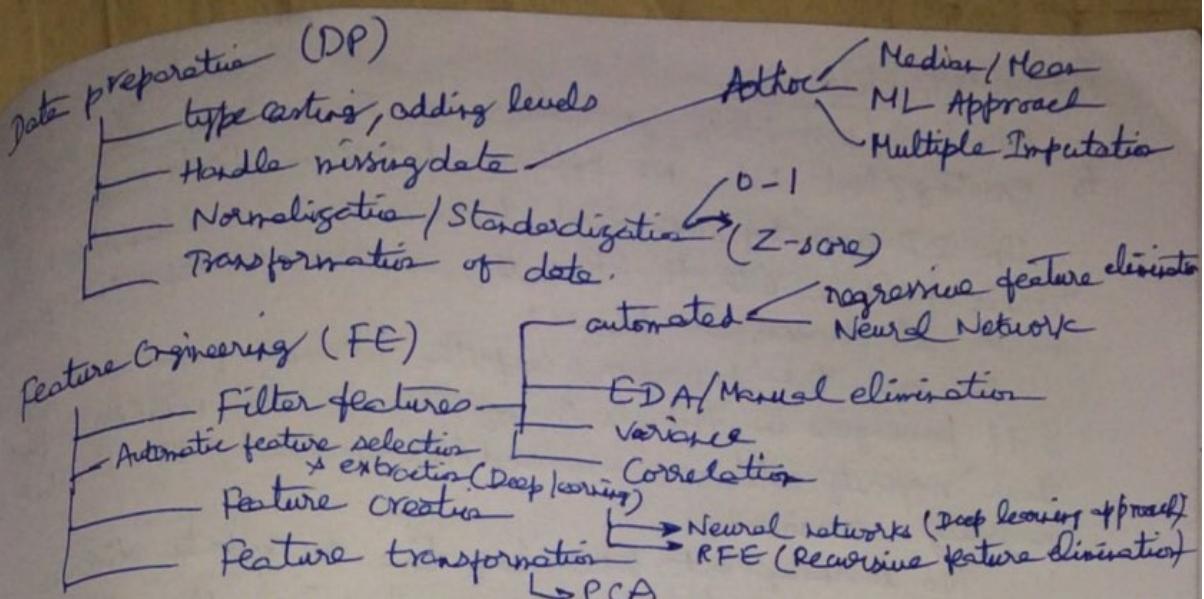
Y = Actual value of predicted data

ERR = 0, if  $\hat{Y} = Y$   
1, if  $\hat{Y} \neq Y$

Finally,  $ERR_{OOB} = \text{proportion}(ERR)$

$\text{Accuracy}_{OOB} = 1 - ERR_{OOB}$

Not able to improve accuracy much with NB, CART, C4.5, bagged tree, Random Forest. Check if we can improve on 'Data Preparation' and 'feature engineering'.



By working on DP and FE, our algorithms might perform so much better.

Is Random Forest, "boot". if we say resampling strategy (method = "boot", number = 10)  
How is the  $10^4$  used?

[Ans:] We compute OOB accuracy for the 100 trees in the forest.  
Repeat this 10 times and compute average (OOB accuracy) across 10 random forests (each forest having 100 trees built by bootstrap)  
So, in our example we have build  $10 \times 100 = 1000$  trees overall.

Why is 10 random forests  $\times$  100 trees each, better than 1 random forest with 1000 trees?

[Ans:] If the training data is small, creating 1000 trees could result in duplicate/identical trees; the bootstrap samples could be identical. We want to minimize the chance of such duplication. Hence we prefer 10 runs of random forest with 100 trees each, and taking the average accuracy across the runs.

29/11/2016 Missing data handling

- Adhoc approach
  - Mean/Median
- ML based approach
- Multiple imputation

How to handle missing 'Ages'?

If age is not dependent on other attributes, best we can do is to use 'mean' age. But if the age can be deduced from other attributes, we can use ML techniques - we train data to 'predict' age. Predicting continuous variables is called 'Regression'.

- Model Building
- Trees (CART, C4.5)
  - Probability (Naive Bayes)
  - Ensemble (Random Forest, Tree bags)
  - Instance based learning (KNN)
    - [K-Nearest Neighbor]

## K-Nearest Neighbors

No tree/probability based learning. Find those who are similar to existing/test data. No pattern detection. Instance-based learning.

In 2D geometry, we find distance in two dimensions. For titanic problem with 11 attributes, we have to represent passenger in a 11-dimensional plane.

For each 'test' passenger, compute distance from each of 891 passengers in 'train'. Among the 'nearest' neighbors, find their 'majority vote' on survived/non-survived. Predict the majority 'survived'.

No learning but takes long time to compute distance with 'every' train passenger  $\rightarrow$  'Lazy learning' approach.

KNN applicable only to 'continuous' data because we can't compute distances between categorical values.

Which 'K' in 'KNN' is best? Need to find out by trials. It is a parameter to be tuned.

K-NN can solve both 'Classification' & 'Regression'.

$\downarrow$  Majority vote of K neighbors       $\downarrow$  Average of K-nearest neighbors  
 $\Rightarrow$  But all the predictor variables must be 'continuous' so that distances can be measured.

Advantage of Lazy-learning techniques like K-NN over CART/C4.5, NB

1. We can easily add new samples to the training data, and also remove samples, deal successfully with changes to the problem domain. But CART/C4.5 need changes to trees and NB needs changes/recompute the conditional probability tables.
2. Can solve multiple problems simultaneously since the target function is approximated locally for each query to the system. Suppose 'training' data has 20 attributes and test data has only 15 attributes. And we need to predict all 5 ( $20-15$ ) attributes. Once we find out the K-NNs using the 15 (or their subset) attributes, we can immediately compute the values of all 5 attributes from the KNN values.

In CART/C4.5, we need separate decision trees for each of the 5 attributes. Also need, in NB, 5 sets of conditional probability tables.

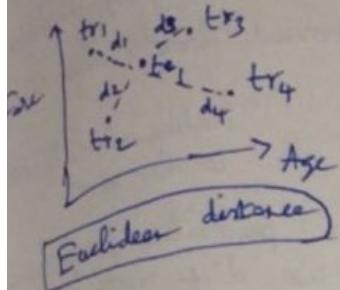
## Disadvantages

1. Large memory requirement to store entire training data
2. Slow since we need to compute distances for every training sample -  $O(n^2)$  time. (The 'training' phase is faster though since we just store the data.)

Lazy classifiers are most useful for large datasets with few attributes.

In KNN, how to measure closeness/similarity between 2 passengers?  
 for every passenger in 'test',  
 compute distances from each  
 'train' passenger.

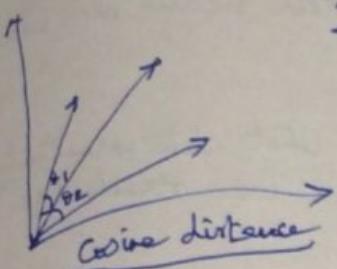
- Euclidean distance
- Cosine distance
- Correlation distance



$tr_i \rightarrow i^{\text{th}} \text{ train passenger}$   
 $te_i \rightarrow i^{\text{th}} \text{ test passenger}$

$(x_1, y_1) \quad d \quad (x_2, y_2)$

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



Angle  
 Distance from origin to point.

If angles are less, we assume closeness.  
 Problematic because although far away,  
 angle extended could be similar; methods  
 used to overcome this.

Euclidean distance for  $(x_1, y_1, z_1) \rightarrow (x_2, y_2, z_2)$  is

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Q: In KNN, chose 'K' as odd so that no issue in 'majority vote'.  
 To handle different 'target' variables, we generally use different sets of 'predictor' variables. So, this advantage is not always valid. (#2)

Weighted KNN  $\rightarrow$  1<sup>st</sup> closest neighbor has more weightage than 2<sup>nd</sup> closest etc. Diff weights for diff closeness.

Which "classification" method to use? - CART, C4.5, NB, Random forest, KNN, Weighted KNN

[Ans.] Based on experience. Successful winners use multiple approaches in a "meta-ensemble" model.

Q: Why data is "obscured" and not named? To maintain confidentiality. Understanding the columns, and lacking their identity makes big impact; depends on domain knowledge and creativity.

### Restaurant revenue competition

137 train data, 100000 test data.

Target variable is Revenue; its a 'continuous' variable  $\Rightarrow$  Regression

'Random' guess? Find the min & max values and predict

'randomly' in the range.

All factor 'levels' not in 'train' but in 'test' - common problem.

Accuracy of 'regression' submission = Root mean squared error (RMSE)  

$$= \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$
 Using KNN in regression:  
 - use 'average' of K-nearest neighbors

KNN is meaningful for 'restaurant revenue' due to similarity measure.  
 Applying KNN, our Kaggle submission score improved drastically to restaurant revenue.

1/12/16 Mean absolute error =  $\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$ ; Median absolute error is also good.

MAE & RMSE are both valid metrics to measure prediction performance in regression problems. Diff. Kaggle competitions use diff. metrics.  
 To overcome 'outlier' issues, we use other metrics like 'Median absolute error'.

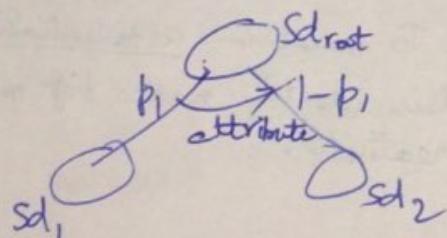
2/12/16 What model we build decides what data normalization to do. For eg, in KNN (distance-based) we standardize numbers into z-score.

5/12/16

	Classification	Regression
Tree learning (CART/C4.5)	✓ CART/C4.5	only CART ? SD ✓ CART C4.5
Probabilistic (NB)	✓	? × Not meaningful since can't find probabilities for multiple values.
Breastable (RF & BT)	✓	? ✓ Use CART trees and take average
Instance based (KNN)	✓	✓ (Avg. of neighbors)
Heterogeneous ensemble (Majority vote)		

How to use 'Tree learning' for regression?

"Impurity" for regression here is "standard deviation".



$$Sd_1 p + Sd_2 (1-p) = SD \text{ of split.}$$

using this attribute

Choose the attribute that minimizes SD of split.

We stop when leaf nodes have less than some threshold SD. Ideally, leaf nodes should have '0 SD' but it is not practical.

\* We don't usually use regression trees. Instead, we just use 'threshold SD'. We can prune the tree.

The value of leaf node in regression tree is "average" of samples in the node. [In classification, leaf node

outlook		SD	
		Overcast	Rainy
Sunny		3.49	4
		7.78	5
		10.87	5
			<u>14</u>

$$S(T, X) = \sum_{C \in X} P(C) S(C)$$

$$\begin{aligned} S(\text{Hours}, \text{Outlook}) &= P(\text{Sunny}) * S(\text{Sunny}) + P(\text{Rainy}) * S(\text{Rainy}) + P(\text{Overcast}) * S(\text{Overcast}) \\ &= \left( \frac{4}{14} \times 3.49 \right) + \left( \frac{5}{14} \times 7.78 \right) + \left( \frac{5}{14} \times 10.87 \right) \\ &= 7.66 \end{aligned}$$

$$SD(\text{overall data}) = 9.32$$

$$SD \text{ reduction} = SDR = 9.32 - 7.66 = 1.66.$$

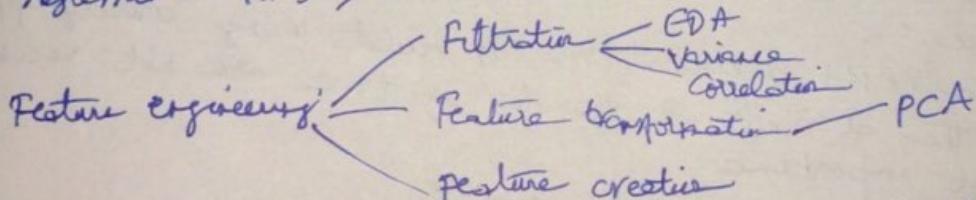
using outlook

output of 'root' or regression node), split, n, deviance, yval  
 If 'treebag' approach, we take 'average' or all trees for 'regression'.  
 $\hookrightarrow SD$        $\hookrightarrow \text{average}$

How does a random forest compute 'variable importance'?  
 [Quoting from <https://tpepo.github.io/caret/variable-importance.html>]  
 "For each tree, the prediction accuracy on the out-of-bag portion of the data is recorded. Then the same is done after permuting each predictor variable. The difference between the two accuracies are then averaged over all trees, and normalized by the standard error: for regression, the MSE is computed on the out-of-bag data for each tree, and the same computed after permuting a variable. The differences are averaged and normalized by the standard error. If the standard error is equal to 0 for a variable, the division is not done."

6/12/16

Metrics for assessment  
 classification - accuracy, AUC, Kappa  
 regression - RMSE, Mean absolute error, median absolute error,  $R^2$ .



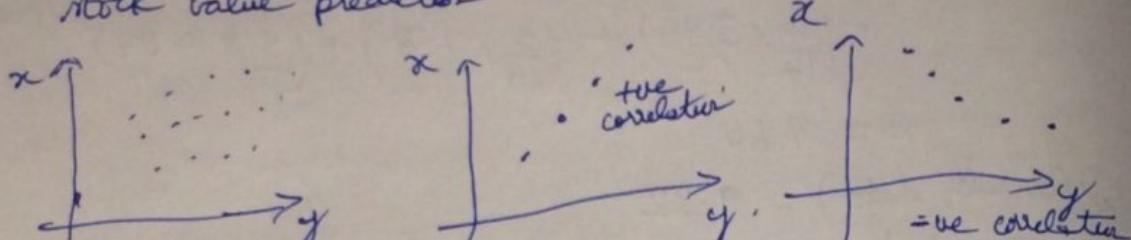
What parts of data analytic life cycle can be automated?

- Data collection needs a lot of manual work
- Data preparation - manual - depends on what model we use
- Feature engineering - for structured data, mostly manual
  - for unstructured data, it is automated mainly using NLP tools

- Model building - tuning of models needs manual work

Correlation in Regression (the strength of the relationship used to find relationship between two continuous variables. Visualise them using "scatterplots". Used by "portfolio manager" to analyse a related variable to guess the other related variable.

(ex) Analyse "IBM stock value fluctuation" to predict Microsoft stock value prediction.



$$\text{Covariance } (x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n}$$

Note that if  $x = y$ , covariance  $(x, y) = \text{covariance } (x, x) = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n} = \text{variance } (x)$

→ Apply Z-score Standardization to get covariance or  $\text{correlation}$

$$\text{correlation } (x, y) = \frac{1}{n} \sum_{i=1}^n \left[ \frac{x_i - \bar{x}}{\sigma_x} - 0 \right] \left[ \frac{y_i - \bar{y}}{\sigma_y} - 0 \right]$$

~~Covariance values are dominated by 'scale' - So, normalize to understand relationship.~~

"Standardized Covariance" is called "correlation".

$$\text{correlation } (x, y) \in [-1, +1]$$

Z-scores mostly range (~99%) in  $[-3, +3]$

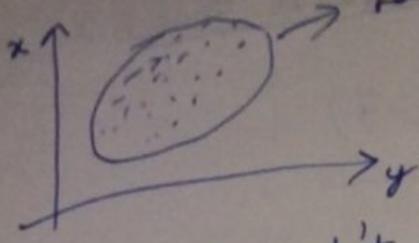
Q) varImp() in CART / rpart

- This is computed in a way similar to that of random forest. The variable columns are permuted, and if accuracy does not change much on the permuted values, that variable has low importance.

The CART decision tree is built using "greedy approach". Hence the attributes at the top of tree are not necessarily the most important.

Q) Why C4.5 does not do regression?

The post-pruning strategy there (which does not depend on input data) cannot be applied on 'continuous' target variables.



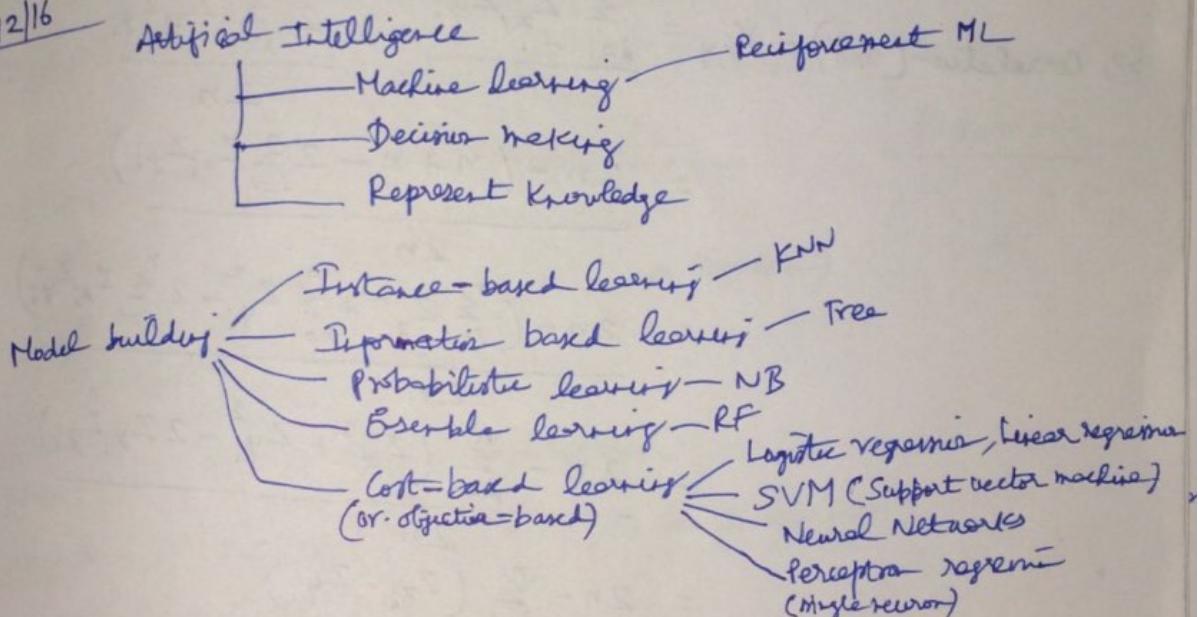
narrow ellipse means more correlation

- When investing in stocks, don't invest in those that have a high correlation — loss of one stock should not lead to loss in another stock also.
- "variance is used to filter features"
  - If a column has ~0 variance, i.e., all values are same, it is not useful for prediction.
  - Correlation also used to filter features
  - Remove redundant features

	1	2	3	4	5
1	1.1	0.9	0.95	0.5	0.6
2	0.9	1	0.8	0.9	0.8
3	0.95	0.8	1	0.92	0.9
4	0.5	0.9	0.92	1	0.8
5	0.6	0.8	0.9		1

To decide whether to remove 1 or 2, remove the one with the largest mean absolute correlation.  
— used by 'findCorrelation' function

9/12/16



Why is correlation coefficient not greater than 1?

We know that

$$\text{Covariance}(x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n}$$

$$\begin{aligned}\text{correlation}(x, y) &= \text{Covariance}(Z_x, Z_y); \quad Z_{x_i} = Z\text{-score}(x_i) = \frac{x_i - \bar{x}}{\sigma_x} \\ &= \frac{\sum_{i=1}^n (Z_{x_i} - \bar{Z}_x)(Z_{y_i} - \bar{Z}_y)}{n} \quad \sigma_x = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}} \\ &= \frac{\sum_{i=1}^n (Z_{x_i} - \bar{Z}_x)(Z_{y_i} - \bar{Z}_y)}{n} \quad \bar{Z}_x = \frac{\bar{x} - \bar{x}}{\sigma_x} = 0 \\ &\quad \bar{Z}_y = \frac{\bar{y} - \bar{y}}{\sigma_y} = 0.\end{aligned}$$

$$\therefore \text{correlation}(x, y) = \frac{\sum_{i=1}^n (Z_{x_i} Z_{y_i})}{n}.$$

$$\text{Also, } \sum_{i=1}^n Z_{x_i}^2 = \sum_{i=1}^n \left( \frac{x_i - \bar{x}}{\sigma_x} \right)^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{\sigma_x^2} = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{\left( \frac{\sum (x_i - \bar{x})^2}{n} \right)} =$$

$$\text{Similarly, } \sum_{i=1}^n Z_{y_i}^2 = n.$$

$$\begin{aligned}\text{So, Correlation}(x, y) &= r_{xy} = \frac{\sum_{i=1}^n Z_{x_i} Z_{y_i}}{n} = \frac{2 \sum_{i=1}^n Z_{x_i} Z_{y_i}}{2n} \\ &= \underbrace{2n - (n + n - 2 \sum_{i=1}^n Z_{x_i} Z_{y_i})}_{2n} \\ &= \frac{2n - \left( \sum_{i=1}^n Z_{x_i}^2 + \sum_{i=1}^n Z_{y_i}^2 - 2 \sum_{i=1}^n Z_{x_i} Z_{y_i} \right)}{2n} \\ &= \frac{2n - \sum_{i=1}^n (Z_{x_i}^2 + Z_{y_i}^2 - 2 Z_{x_i} Z_{y_i})}{2n} \\ &= \frac{2n - \sum_{i=1}^n (Z_{x_i} - Z_{y_i})^2}{2n} \\ &= 1 - \frac{1}{2n} \left[ \sum_{i=1}^n (Z_{x_i} - Z_{y_i})^2 \right]\end{aligned}$$

The RHS in the term is non-negative.

$$\Rightarrow r_{xy} = 1 - (\text{a non-negative value})$$

$$\text{so, } r_{xy} \leq 1. \quad \text{--- (1)}$$

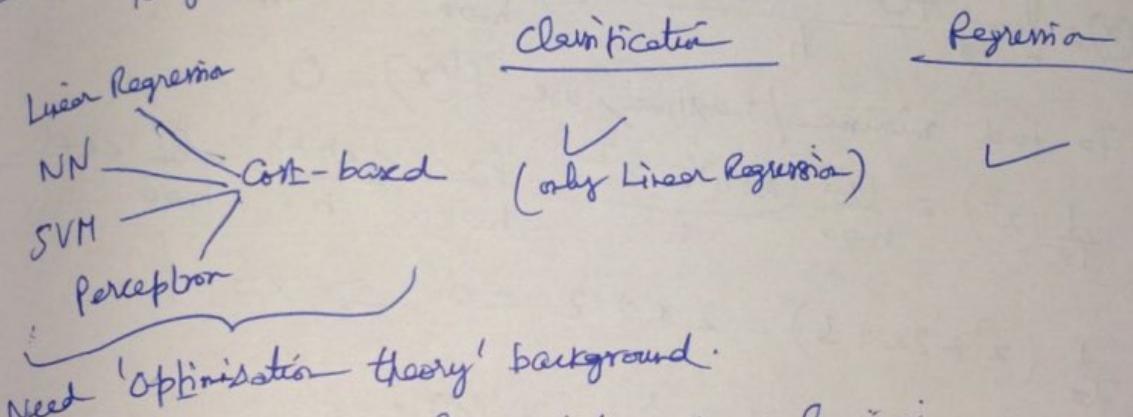
$$\begin{aligned}
 \text{Also, } r_{xy} &= \frac{\sum_{i=1}^n z_{xi} z_{yi}}{\sqrt{\sum_{i=1}^n z_{xi}^2} \sqrt{\sum_{i=1}^n z_{yi}^2}} = \frac{2 \sum_{i=1}^n z_{xi} z_{yi}}{2n} \\
 &= \frac{-2n + (n + n + 2 \sum_{i=1}^n z_{xi} z_{yi})}{2n} \\
 &= \frac{-2n + (\sum_{i=1}^n z_{xi}^2 + \sum_{i=1}^n z_{yi}^2 + 2 \sum_{i=1}^n z_{xi} z_{yi})}{2n} \\
 &= \frac{-2n + \sum_{i=1}^n (z_{xi}^2 + z_{yi}^2 + 2 z_{xi} z_{yi})}{2n} \\
 &= \frac{-2n + \sum_{i=1}^n (z_{xi} + z_{yi})^2}{2n} \\
 &= -1 + \frac{1}{2n} \left[ \sum_{i=1}^n (z_{xi} + z_{yi})^2 \right]
 \end{aligned}$$

The RHS term is non-negative.

$$\begin{aligned}
 \text{So, } r_{xy} &= -1 + (\text{a non-negative value}) \\
 \Rightarrow r_{xy} &\geq -1 \quad \text{--- (2).}
 \end{aligned}$$

$$\text{So, (1) and (2)} \Rightarrow r_{xy} \in [-1, +1]$$

12/12/16 To get data — Kaggle.com, TopCoder.com



Objective / Cost-based Learning / Loss-based Learning

Objective function

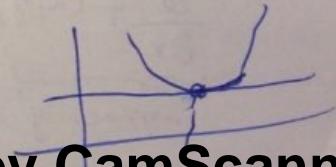
- Minimise error/loss
- Constraints

$$(ex) f(x) = x^2 + 2x + 3$$

find value of  $x$  that minimizes this value.

Find all possible values of ' $x$ '

- Not practical with multiple variables



$$f(x,y) = x^2 + 2xy + xy^2 + 10 \rightarrow \text{Need a 3-D plot to see } x, y \text{ & } f.$$

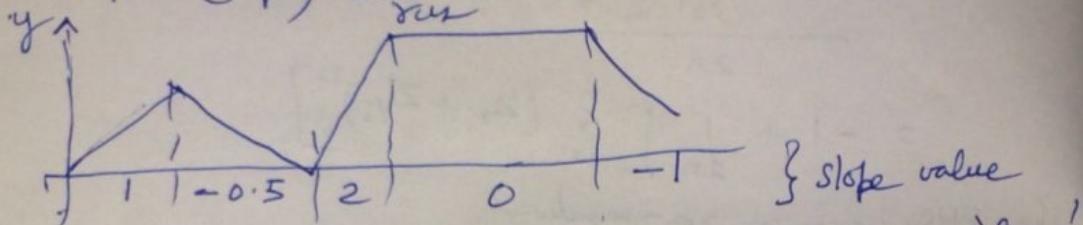
$$\text{To minimize } f(x) = x^2 + 2x + 3$$

- Explore all possible values of  $x$
- Use calculus
- Use numerical methods

Calculus need for solving 'Optimization' problems in data-science context. Also needed to study the rate-of-change in 'Physics' context.

To capture 'steepness' of line, Steepness =  $\frac{\Delta y}{\Delta x} = \frac{\text{rise}}{\text{run}}$

$$\text{Steepness (slope)} = \frac{\text{rise}}{\text{run}}$$



This is a 'piece-wise' linear function. Idea of 'line' slope still applicable.

How to find slope of curve at different points?

We use 'limit' to discover close points.

Using limit to find the slope of tangent line.

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \quad (\text{or}) \quad \lim_{h \rightarrow 0} \frac{f(x-h) - f(x)}{-h}$$

To find minima / maxima, use  $f'(x) = 0$ .

$$\frac{d}{dx}(x^2) = \lim_{h \rightarrow 0} \frac{(x+h)^2 - x^2}{h} = \lim_{h \rightarrow 0} \frac{2xh + h^2}{h} = \lim_{h \rightarrow 0} 2x + h = 2x$$

$$\frac{d}{dx}(x^2 + 2x + 3) = 2x + 2 = 0 \Rightarrow x = -1$$

$$\text{slope of } f(x) \text{ at point } 'a' = f'(a)$$

Slope of  $f(x)$  at point ' $a$ ' =  $f'(a)$ .

When dealing with  $f(x,y)$ , how to find 'slope'?

- This is a 3D picture.

- Make this a 2D by fixing value of ' $x$ ' or value of ' $y$ '.

$\underbrace{\frac{\partial}{\partial x} f(x,y)}$  or  $\underbrace{\frac{\partial}{\partial y} f(x,y)}$  — partial differential equations  
 assume ' $y$  constant'      assume ' $x$  constant'

At point of maxima & minima,  
 $\frac{\partial}{\partial x} f(x,y) = 0 = \frac{\partial}{\partial y} f(x,y)$ .

But the challenge is to solve many equations.

If 'n' variables, need 'n' partial differential equations.

Partial derivative - to compute slope of a curve at any point in multiple dimensions.

Find slope of  $f(x,y) = x^2 + y^2$

$$\frac{\partial}{\partial x} f(x,y) = 2x \quad \frac{\partial}{\partial y} f(x,y) = 2y$$

$$\text{slope } (1,1) \rightarrow 2x|_{x=1}, 2y|_{y=1} = (2,2)$$

Need "gradient" to combine slopes using ' $x+y$ '.

18/12/16

Find 'x' that minimizes  $f(x) = 1.2(x-2)^2 + 3.2$

$$f'(x) = 2.4(x-2) = 0 \Rightarrow x=2.$$

$$\therefore \min f(x) = 3.2$$

To solve unconstrained optimization problems

- Try all possible solutions and pick the optimal one (Ad-hoc, laborious)
- find the optimal soln (min/max) using slopes and equation solving (Calculus)
- Guess the solution and refine it until we get close to optimal solution (Numerical methods)

1) Find slope of  $f(x) = 2x^2 + 5x + 6$  at  $x=2, 3$ .

$$f'(x) = 4x+5.$$

$$\text{slope at } x=2 \text{ is } f'(2) = 13. \text{ Slope at } x=3 \text{ is } f'(3) = 17.$$

2) Answer  $f(x,y) = 2x^3 + 4x^2y + 8xy + 23$

a) Find slope of  $f$  at  $(1,2)$  and  $(2,1)$  - need slopes of tangential 'plane'.

b) Find  $(x,y)$  that minimizes  $f$ .

$$(y \text{ constant}) \frac{\partial f}{\partial x} = 6x^2 + 8xy + 8y \Big|_{(1,2)} = 6 \cdot 1^2 + 8 \cdot 1 \cdot 2 + 8 \cdot 2 = 38$$

$$(x \text{ constant}) \frac{\partial f}{\partial y} = 4x^2 + 8x \Big|_{(1,2)} = 4 \cdot 1^2 + 8 \cdot 1 = 12$$

Slope of line at  $(1,y)$  in 'y' direction.

To find minima

$$\left. \begin{aligned} \frac{\partial f}{\partial x} &= 0 = 6x^2 + 8xy + 8y \\ \frac{\partial f}{\partial y} &= 0 = 4x^2 + 8x \end{aligned} \right\} \text{Solve these 2 equations.}$$

Generally hard to solve these equations. ~~Non-linear~~ Non-linear  
and hard to solve. So we use numerical methods/approaches

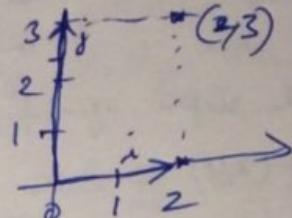
Gradient = Vector of all partial derivatives

Partial derivatives give slope along 'x' & slope along 'y'. To go towards minima of  $f(x,y)$  which direction to slide? - Get one direction based on two different slopes/direction - This "combined direction" is called gradient.

Point  $(2,3)$  can be represented by vector  $(2\hat{i}) + (3\hat{j})$ .

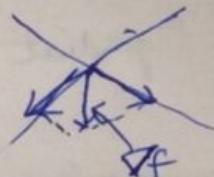
" $\nabla$ " gradient symbol.

$$\nabla f = \frac{\partial f}{\partial x} \hat{i} + \frac{\partial f}{\partial y} \hat{j}$$



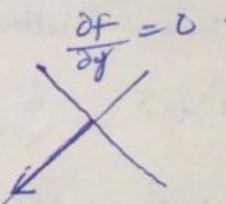
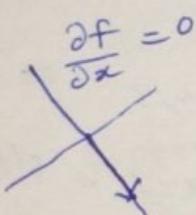
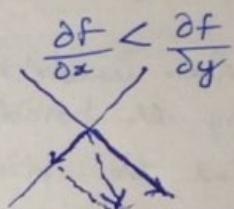
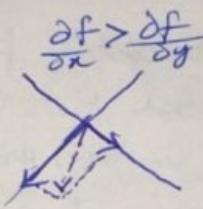
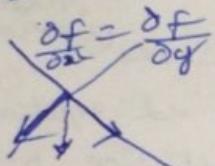
Gradient points us to the direction to move

In 1-variable equations (represented in 2D),  
slope = gradient



But in 2+ variable equations (3D+)  
gradient determined using combination of slopes.

Jump in small steps along the 'gradient' to move towards the minima/maxima



14/12/16 Solving optimisation problems: guess and refine

How to refine?

- Can I move along tangents/gradient to move towards maxima/minima?

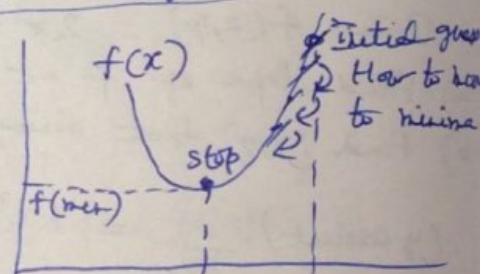
Gradient is a vector that combines multiple slopes.

Start with a point (guess)

Repeat

- Determine a descent direction
- choose a step
- Update

Until stopping criterion is satisfied.



How to choose direction?

- gradient provides solution

How much to refine?

- no multiplier - very bad, no convergence

- constant multiplier - OK but needs many iterations

- adaptive multiplier - best

When 'multiple' minima are available, the algorithm will find a particular 'minima' based on 'initial guess'. Use different initial guesses to find different "local minima"

Overall minima is called "global minima".

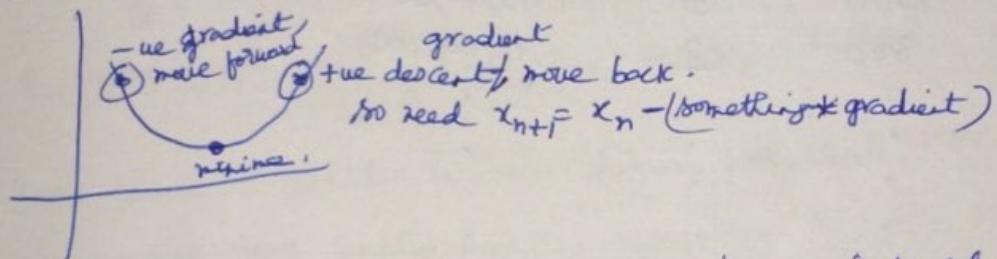
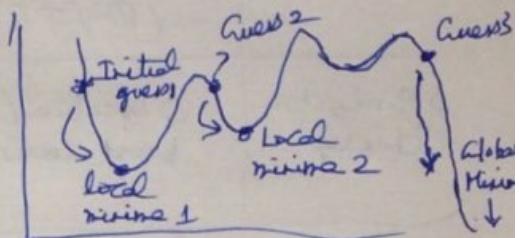
### Non-Convex functions

Plot function to see structure. Make multiple initial guesses [Applies to 'minima' problems only]

$$x_{n+1} = x_n - (\text{Step Factor} * \text{gradient})$$

(-) is used because

defines no multiplier, constant, adaptive multiplier  
(Step Factor = 1)



We treat "maxima" problem as a "minima" problem by converting  $f(x)$  to  $-f(x)$ . So, most built-in functions solve only the "minima" problem.

For 'maxima' problem,

$$x_{n+1} = x_n + (\text{Step Factor} * \text{gradient})$$

- If stepFactor = 1, we see that the values of  $x_n$  are not converging. They are diverging instead. Never use them.
- If stepFactor has small fractional values like 0.005 or 0.01,  $x_n$  [and thus  $f(x_n)$ ] converges.
- We stop the refinement based on either of the below 2 factors:
  - number of iterations done (when  $x_n$  has not converged sufficiently)
  - $(x_n - x_{n-1}) < \text{some threshold}$ , i.e.,  $x_n$  has converged sufficiently

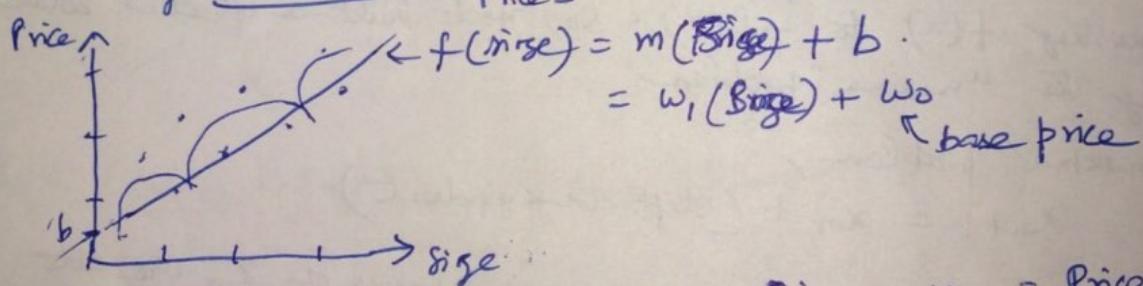
15/12/16

Probability + Bayes rule	Information based (Tree) learning	Classification CART/C4.5	Regression C4.5
Distance + Similarity	Instance based (KNN)	KNN	KNN
Probability + Bayes rule	Probabilistic learning	NB	X Not possible
Probability	Ensemble learning (RF/TB)	RF, TB, Boosted Tree, Custom	
Optimization theory	Objective/Cost based learning	Logistic regression, SVM Perceptron NN	Linear/Multivariate Regression, SVM, NN.

Restaurant revenue - Cost-based approach. Linear Regression

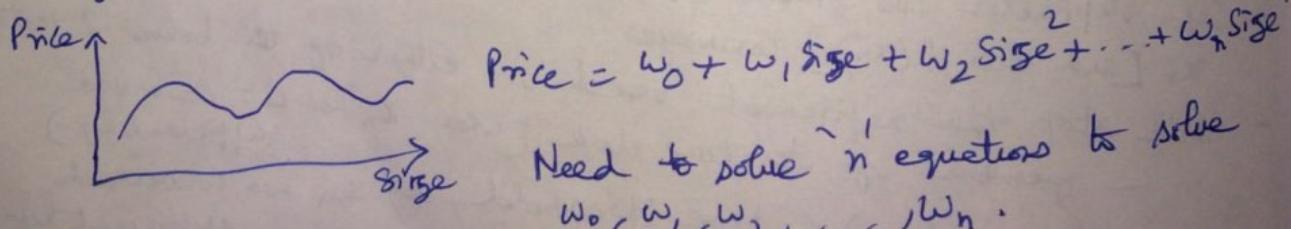
Options	Size	# of rooms	Price
:	1000	3	15,00,000
:	800	2	12,00,000
:	:	:	:
:	:	:	:
:	:	:	:

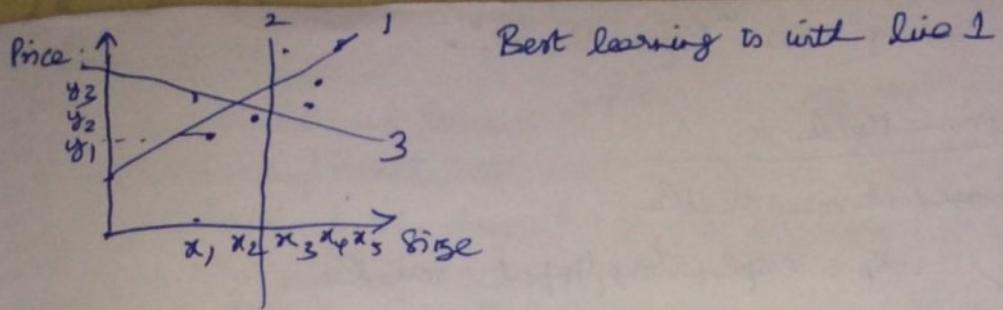
Assume only Size  $\rightarrow$  Price Price =



If line fitting, use  $f(\text{size}) = w_1 \text{Size} + w_0 = \text{Price}$

If curve fitting, use  $f(\text{size}) = w_2 \text{Size}^2 + w_1 \text{Size} + w_0 = \text{Price}$





To quantify best fit,

use  $(y_i - \hat{y}_i)^2$  instead of Absolute  $(y_i - \hat{y}_i)$ . Why?

Because we cannot differentiate or 'absolute' values. So, better to use  $(y_i - \hat{y}_i)^2$ .

Need to minimize,  $\sum_{i=1}^n (y_i - \hat{y}_i)^2 = E(w_0, w_1)$  Error = Objective function  
 $y_i - \hat{y}_i$  = residual

$$\hat{y}_i = w_0 + w_1 \text{size}_i \quad [\text{Linear Model}]$$

$$y_i = \text{Price}_i$$

So, for restaurant revenue, to minimize

$$E(w_0, w_1) = \sum_{i=1}^n (\text{Price}_i - w_0 - w_1 \text{size}_i)^2 \leftarrow \begin{array}{l} \text{Solve minimization with} \\ \frac{\partial E}{\partial w_0} \text{ and } \frac{\partial E}{\partial w_1} \end{array}$$

[Polynomial Model]

$$\hat{y}_i = w_0 + w_1 \text{size}_i + w_2 \text{size}_i^2 + w_3 \text{size}_i^3$$

$$E(w_0, w_1, w_2, w_3) = \sum_{i=1}^n (\text{Price}_i - w_0 - w_1 \text{size}_i - w_2 \text{size}_i^2 - w_3 \text{size}_i^3)^2$$

Our choice to decide between linear, polynomial, more complex models.

If we use very complex models, overfitting to noise happens.

Minimize using EDA or calculus or Numerical approach (gradient descent)

Q: We could use non-polynomial approaches using  $\sin()$ ,  $\cos()$ ,  $\log$ ,  $e^x$ , but ensure that they don't overfit.

How to avoid over-fitting in cost-based approach?  
 yet to see that. Tree overfitting solved using pruning.

If we have more variables, &

$$\text{Linear model} = w_0 + w_1 \text{Input}_1 + w_2 \text{Input}_2 + w_3 \text{Input}_3$$

Polynomial model  $\rightarrow$  Could choose 2nd degree polynomial for Input 1, and linear model for Input 2 also,

$$(ex) w_0 + w_1 \text{Input}_1 + w_2 \text{Input}_1^2 + w_3 \text{Input}_2$$

So, polynomial  $\leftarrow$  not used much due to overfitting. So,  
 linear models used widely in practice.

Google page ranking uses "Eigen Vector".

16/12/16 Regression Model

$Y$  = response/outcome variable

$X_1, X_2, X_3, \dots, X_p$  = explanatory/input variable

$Y = f(X_1, X_2, X_3, \dots, X_p) + e \rightarrow$  approximate general relation  
Linear relationship is

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \dots + b_p X_p + e$$

For restaurant revenue, use cross-validation to estimate accuracy.

Get RMSE for each of the 10 cv iterations and take average.

(\*) Why not solve 'revenue of restaurant' by solving ' $n$ ' equations, one for each row in train?

- does over-fitting
- equation solving can't handle non-linear relationships
- lot of equations to be solved - very hard

So, better to solve as a minimization problem.

Objective function for parameter estimation

$$\text{Error}(m, b) = \frac{1}{N} \sum_{i=1}^N (y_i - (mx_i + b))^2.$$

Eliminate predictors,  
Lasso Regression - tells which variables are less important

- Draw scatterplot of predictor vs target.
- If it has no linear relationship, don't use for linear regression.

(\*) There is a gap between  $y_i$  and  $\hat{y}_i$ . Let  $y_i - \hat{y}_i = e_i$ .  
We need all ' $e_i$ ' values to have normal distribution - Then the  $e_i$  values will cancel out and so need not be considered explicitly for optimization.

If  $e_i$  does NOT have normal distribution, problems, prediction poor

→ In LR, if a predictor does not have linear relationship, it may still have non-linear relationship like  $x^2, x^3$  etc.  
If we know the relationship, replace predictor $i$  with predictor $i^2$ .  
But this kind of analysis is very hard when we have multiple variables.

- Handle this by eliminating highly correlated variables
- Use CV to estimate RMSE
- If RMSE low enough, good to use LR model.

## Answer and refine

- Only for  
ML  
prob.  
not  
Math  
problem
- Batch Gradient Descent - gradient computation <sup>uses</sup> all the train samples
  - Stochastic / Probabilistic descent - gradient computation uses only one random train sample
  - Stochastic batch descent - gradient computation uses a mini-batch of random samples
- $$f(w_0, w_1) = \sum_{i=1}^n (y_i - w_0 - w_1 x_i)^2$$
- $$\frac{\partial f}{\partial w_0} = 2 \sum_{i=1}^n (y_i - w_0 - w_1 x_i) (-1)$$
- $$\frac{\partial f}{\partial w_1} = -2 \sum_{i=1}^n (y_i - w_0 - w_1 x_i) x_i$$
- $$f(x) = 1 \cdot 2(x-2)^2 + 3$$
- $$f(x, y) = x^2 + y^2$$
- $$\frac{\partial}{\partial x} f(x, y) = 2x$$

We just use 'gradient descent'  
as there is no  $\Sigma$  involved.

If train data is very small, use 'Batch gradient' descent.

If very big, use Stochastic descent

↓  
10000 iterations  $\times$  1 <sup>random</sup> sample each

due to less time for an iteration converges faster

Stochastic too optimistically, so 'Stochastic batch' is preferred.

↓  
100 iterations  $\times$  1000 samples each

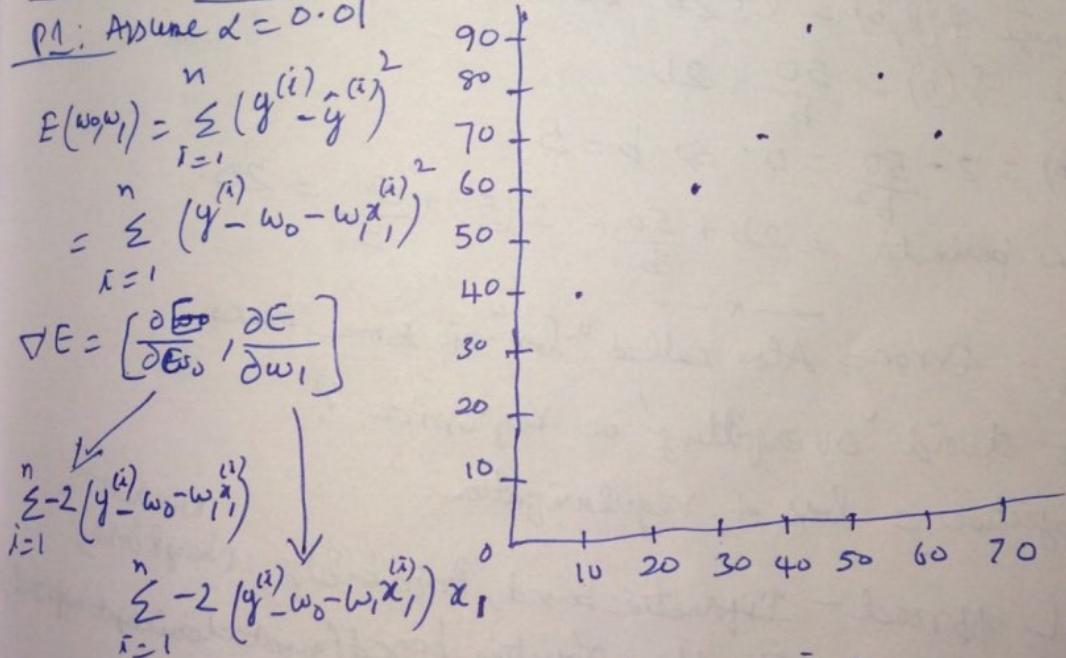
converges slowly as  
iteration takes longer time

(\*) In 'stochastic batch' descent, the size of batch can be anything (smaller than # observations), but the size must remain constant throughout the algorithm run.

(\*) If data is <sup>highly</sup> imbalanced, don't use 'bootstrap' as it does not stratify. We can still grow random forests (which internally uses bootstrap) but use cross-validation as resampling strategy.

19/12/16 Assignment 8

P1: Assume  $\alpha = 0.01$



This is the gradient. To move towards minima,

$$\text{Batch gradient descent} = w_0 - \alpha \frac{\partial E}{\partial w_0}, \quad w_1 = w_1 - \alpha \frac{\partial E}{\partial w_1}$$

$$= w_0 - \alpha \left( \sum_{i=1}^n (y_i - w_0 - w_1 x_i) \right) = w_1 - \alpha \sum_{i=1}^n -2(y_i - w_0 - w_1 x_i)x_i$$

Stochastic gradient descent - Choose only one data point instead of everything. So, equation becomes

$$w_0 = w_0 - \lambda - 2(y^{(i)} - w_0 - w_1 x_1^{(i)})$$

$$w_1 = w_1 - \lambda - 2(y^{(i)} - w_0 - w_1 x_1^{(i)}) x_1^{(i)}.$$

We shuffle the input data points.

In Epoch 1, use (shuffled) input<sub>1</sub> for gradient, input<sub>2</sub> for gradient in iteration 1, in iteration 2, etc.

Before Epoch 2, again shuffle, and use similar approach. Take more 'epochs' than in batch gradient since delta change is smaller. But each epoch is faster.

We can stop in the mid of an epoch also if enough convergence happened.

### Stochastic mini-batch gradient descent

Batch size = 3

One epoch = (1,4,5) \* (2,3,6) = 2 iterations.

Show calculation for 1 epoch only.

$$\text{Here, } w_0 = w_0 - \lambda \sum_{i=1}^3 -2(y^{(i)} - w_0 - w_1 x_1^{(i)})$$

$$w_1 = w_1 - \lambda \sum_{i=1}^3 -2(y^{(i)} - w_0 - w_1 x_1^{(i)}) x_1^{(i)}$$

The tuning parameter here is 'batch-size'.

This 'mini-batch' is most practical and widely used as it finds middle-ground between 'Batch' and 'Stochastic' approaches.

Problem 2: Find the least amount of fencing.

Minimize  $f(l,b) = l + 2b$  such that  $lb = 50$

Minimize  $f(b) = \frac{50}{b} + 2b$ .

$$f'(b) = 2 - \frac{50}{b^2} = 0 \Rightarrow b = 5.$$

$$\text{So, min. perimeter} = 2b + \frac{50}{b} = 2(5) + \frac{50}{5} = 20.$$

$y_i - \hat{y}_i = \text{Error} = \text{Also called "loss" in some books.}$

How to avoid 'overfitting' in regression?

(\*) Objective = loss + regularization

In ML approach - Information based, Probability, Neighbors, Ensemble, objective based (most dominant approach)

Mathematically show that,

Information based, Prob, Neighbors, Ensemble can all be transformed/mapped to 'Objective based' approach. But this mapping is not so intuitive in learning

To avoid overfitting in Objective-based approach, use 'Regularization'

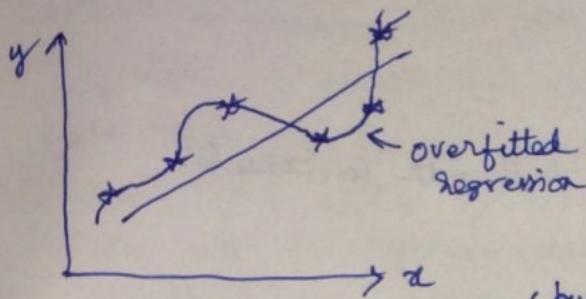
Regularisation

L1 regularization (Lasso)

L2 regularization (Ridge)

L1+L2 regularization (Elastic Net)

Dropout (used in Neural Network)



If curve has too much ups and downs  $\Rightarrow$   $w_0, w_1, w_2, \dots$  values  
(bumpy curves)  
are very high.

$$y_i = w_0 + w_1 x_1 + w_2 x_2$$

If  $w_1$  is high, if  $x_1$  increases by 1,  $y$  increase a lot (by  $w_1$ ).  
so, avoid overfitting, regularize the  $w_i$  values.

$$\text{Objective } (w_0, w_1) = \text{minimize } \sum_{i=1}^n (y_i - w_0 - w_1 x_i)^2$$

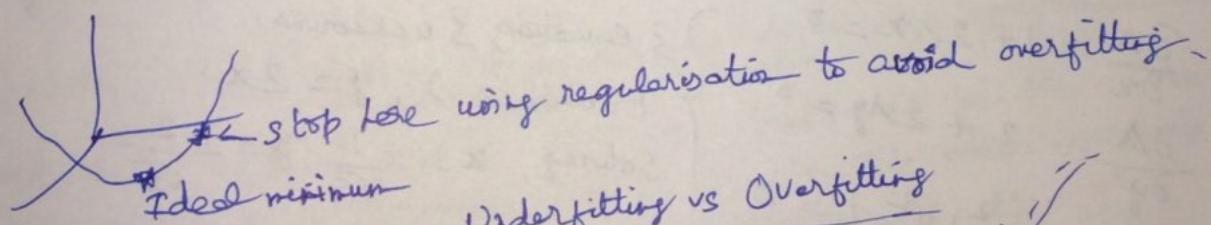
such that  $w_0 \leq \dots$   
 $w_1 \leq \dots$

This detail differentiates  
L1, L2, L1+L2, dropout  
etc!

Regularization  $\equiv$  Constraining weights/slopes.

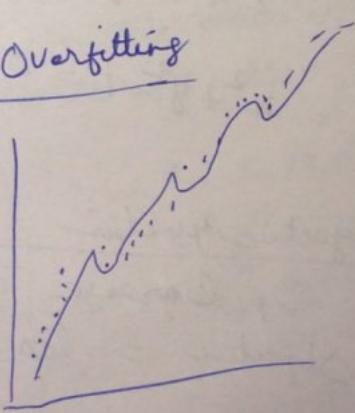
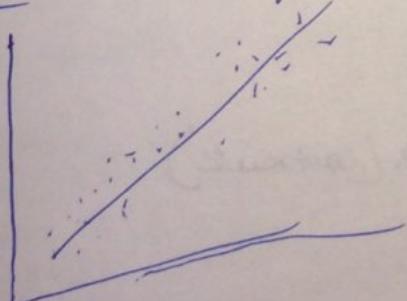
Can control regularization using parameters.

Regularization has 'limited' advantage in linear regression  
- can be used to identify unimportant predictors



### Underfitting vs Overfitting

20/12/16



- Overfitted model - may be capturing irrelevant patterns  
 KNN - very low 'K' [Reason for overfitting] Handling strategy  
 Tree - Very deep tree. Handle overfitting by pruning  
 Linear models - large co-efficients  $w_0, w_1, w_2, \dots$ .  
 Handle overfitting by constraining co-efficients  
 Tree Ensemble - Handle overfitting by growing deep trees and spread across forest of trees.

### Objective function

Objective = loss function

Use Lagrange equation to deal with constraints.

### Optimization with constraints

$$f(x, y) = x^2 + y^2$$

$$x+y=5 \rightarrow \text{Equality constraint}$$

$$x \leq 3, y \leq 5 \rightarrow \text{Inequality constraint}$$

### Approach with constraint

(\*) - Convert constrained optimization problem to unconstrained version and solve it.

Lagrange idea (for equality constraint)

Maximize  $f(x, y)$  subject to  $g(x, y) = C$

Introduce a new variable, and find a maxima

$$\Lambda(x, y, \lambda) = f(x, y) + \lambda(g(x, y) - C)$$

(eg) Maximize  $f(x, y) = x + 2y$  subject to  $x^2 + y^2 = 1$ .

Plane equation

Cylindrical shape

$$\Lambda(x, y, \lambda) = x + 2y + \lambda(x^2 + y^2 - 1)$$

$$\frac{\partial \Lambda}{\partial x} = 1 + 2\lambda x = 0 \quad \left. \begin{array}{l} 3 \text{ equations, 3 unknowns} \\ \text{Eliminate } \lambda; y = 2x \end{array} \right.$$

$$\frac{\partial \Lambda}{\partial y} = 2 + 2\lambda y = 0$$

$$\frac{\partial \Lambda}{\partial \lambda} = x^2 + y^2 - 1 = 0 \quad \left. \begin{array}{l} \text{Solving, } x = \pm \frac{1}{\sqrt{5}} \\ y = \pm \frac{2}{\sqrt{5}} \end{array} \right.$$

### Objective function

By Lagrange idea

Objective = loss function +  $\lambda(\text{constraint})$

## Common Regularization for Linear Model

- \* Ridge constraint:  $\sum_{i=1}^n \beta_i^2 \leq S$ .
- \* Lasso constraint:  $\sum_{i=1}^n |\beta_i| \leq S$ .

- A tuning parameter alpha/lambda imposes a penalty on the size of co-efficients.
- Instead of minimizing "loss" (MSE), minimize "loss + penalty".
- Tiny alpha  $\Rightarrow$  almost like linear model
- Huge alpha  $\rightarrow$  underfitting. Coefficients shrink near to zero
- Choose right 'alpha' by experimenting and tune. Different 'alpha' works for different data sets.

Lasso approach - co-efficients tend to go to '0' if useless parameters so, can be used for 'feature elimination'.

2D view of 3D - contour curve. Cut 3D at diff. levels.

How to choose between Lasso & Ridge?

If correlated variables, Lasso eliminates them. Still, can't say which approach is better. Check out both.

If no correlated variables, can't say which would be better. Try both approaches and see which one gives lower "CV" error.

Lasso only helps to identify 'useless' variables.

Ridge, uses  $\sum \beta^2$ , circle shaped

Lasso, uses  $\sum |\beta|$ , square/diamond shaped

(\*) Intersection of this shape with the gradient descent path gives constrained minima. Since 'plane' is Lasso, perhaps ' $w_i$ ' can be 0.

(\*) method = "lsm" - linear regression may not be doing regularization - uses "mini-batch" approach.

(\*) plot of  $w_0, w_1, \dots$  gives a 3D curve. Each point on the 3D curve corresponds to a particular  $(w_0, w_1)$  value  $\rightarrow$  This value corresponds to a line equation of the input variables - (eg)  $2w_1 + 3$ .

21/12/16 Regression - Model Building

Used for all objective based learnings

(cross entropy loss)  
Logistic Regression

KNN

Batch Gradient Descent

SVM  
(Hinge loss)

Information based

Stochastic GD

Neural Network

Objective based learning

Minibatch GD\*

Perceptron

Linear / Polynomial regression

$\Rightarrow$  Objective fn subject to regularization +

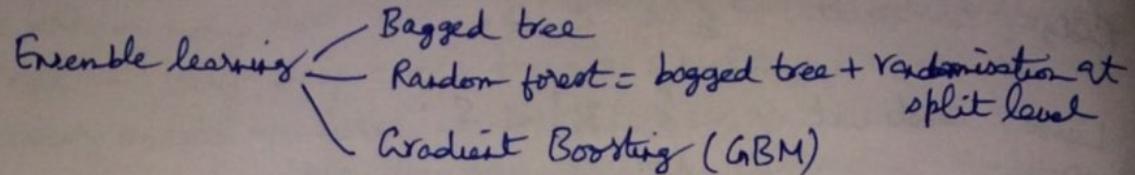
Loss (Squared loss)

Ridge

Lasso

Elastic Net

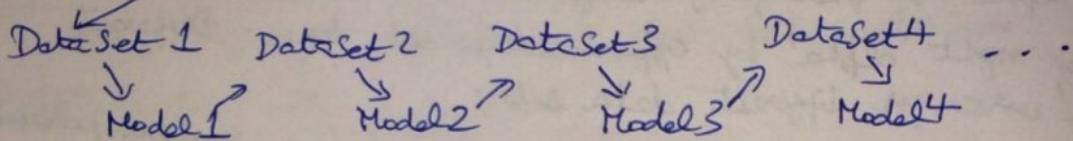
Linear models bad - Can we build a curve?



Polynomial model - Hard to model. Instead, use an ensemble of linear models.

### Boosted Models

Train Data



Each model corrects the mistakes/shortcomings of its predecessor.

Build models in a sequence: "Additive models" - stagewise.

Dataset 1

Size	Price	Price
1000	10 lacs	9 lacs
500	5 lacs	4 lacs
600	8 lacs	6 lacs

Dataset 2

Size	Price	Price
1000	1 lacs	94000
500	1 lacs	93000
600	2 lacs	174000

Price - Price of Dataset 1

# Models = Tuning parameter. Too many models  $\rightarrow$  overfit.

In classification,

(X) If dataset<sub>i</sub> has correct predictions, we don't carry them to dataset<sub>i+1</sub> (model<sub>i+1</sub>)

But in regression,

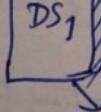
Even if model<sub>i</sub> has some "good enough predictions", we still carry those "good enough" inputs to dataset<sub>i+1</sub>  $\rightarrow$  the gap could be close to zero, but that's fine since the model<sub>i+1</sub> will not use it for learning. (class AdaBoost, XGBoost (not integrated with GB))

Many 'Boosted ideas' present - Gradient Boosted model -

(integrated with GB)

Train

$$y_1 = y$$

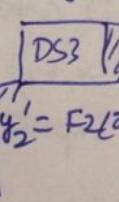
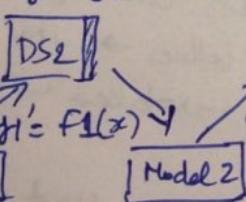


$$y_2 = y_1 - f_1(x)$$

$$y_3 = y_2 - f_2(x)$$

"gbm"

Model 1



$$y_i - f_1(x_i) = \left. \begin{array}{l} \text{Residuals of model } i \\ \text{or } y_i - y'_1 = \end{array} \right\}$$

$$J = \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i))^2$$

$y_1 - f_1(x) =$   
gradient of loss  
w.r.t  $y'_1$

$$y_i - F(x_i) = - \frac{\partial J}{\partial F(x_i)}$$

Residual  
-ve  
gradient

Same dataset used for each model but each dataset is associated with diff. values of target variable

Benefit of AB algorithms using gradient is that R allows us to consider other loss fns and derive other corresponding algorithms accordingly.

Boosting gives best performance on weak/underfitted models. (high bias / low variance)

Bagging gives best performance on overfitted models. (low bias / high variance)

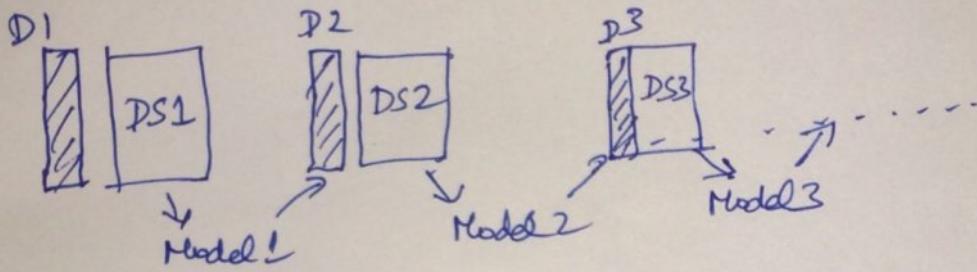
Weak model examples

- linear regression
- low depth tree.

Applying 'Boosting' to trees - use 'low depth'

or 'Bagging' or — use full depth.

"Ada Boost" — Adaptive boosting used for classification



$D_i^w$  — weights of each sample.

$D_1$  — all samples equal weight

$D_2$  — samples correctly predicted by Model 1 — low weight

samples wrongly predicted by model 1 — high weight

Model 2 tries harder to classify these samples correctly.

Final output = weighted summation of all models output.

Why not take only misclassified data to  $DS_2$ ?

Future models have lesser data to work with — will overfit.

Mobile — face detection — implemented using 'Ada Boost'.

If we build a tree (Tree Model) using weights of each sample, in entropy calculation  $E_{\text{pilospo}}$ , we should also multiply by weight.

✳ The advantage of understanding that the residual is actually "—ve gradient is that, it is "faster to compute —ve gradient" directly than to predict  $\hat{y}_i$ , take the diff  $y_i - \hat{y}_i$ , and then get the residual.

22/12/16

## Ensemble

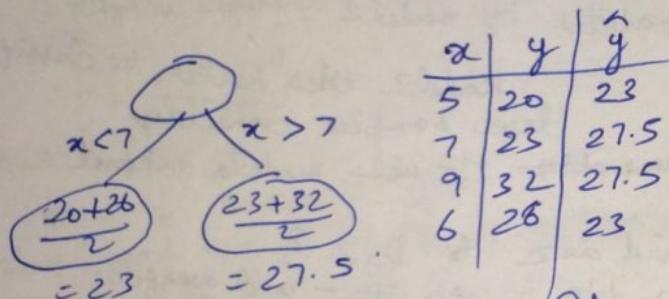
- Bagging
- Bagging + random feature } Improve  
for split Any overfitted Model (e.g) deep trees
- Boosting - gives better result than Random Forest.  
(Weak Models) Use 'underfitted trees' - depth of '3 to 7'
- Gradient Boost - Regression + denification
- Ada Boost, C5.0 only classification
- Extreme Boost (aka "Regularized Gradient Boosting") - Regression classification
- Stacking to combine heterogeneous models
  - ✖ Boosting successful only with weak models

x	y
5	20
7	23
9	32
6	26

Boosting round 1

$F_1(x)$  Linear Regression (Weak model - high bias + low variance)

Shallow Tree ✓  
(Weak Models)



$$\text{Round 2 } \rightarrow (y - \hat{y}) \text{ of round 1} = -\frac{\partial L}{\partial F_1}$$

x	$y_i$	$\hat{y}_i$	$F_1(x)$	Shallow tree	- Final Model
5	-3	0			
7	-4.5	0			
9	4.5	0			
6	3	0			

$x < 7$        $x > 7$

$\frac{-3+3}{2} = 0$        $\frac{-4.5+4.5}{2} = 0$

$$\text{Final Model} = \text{Model 1} + \text{Model 2} \dots$$

$$(e.g) F(8) = F_{\text{Model 1}}(8) + F_{\text{Model 2}}(8) + \dots$$

$$= 27.5 + 0 + \dots$$

$$\text{We can also take } F(8) = f_{\text{Model 1}}(8) + dF_{\text{Model 2}}(8) + dF_{\text{Model 3}}(8) + \dots$$

$\uparrow$  more direction

$$So, F = F + \alpha \frac{\partial L}{\partial F_1} + \alpha \frac{\partial L}{\partial F_2} + \dots$$

$\alpha$  = 'Shrinkage parameter'  
Small  $\alpha \Rightarrow$  More rounds needed

XGBoost = "Regularised" gradient boost.  $\alpha \in [0, 1]$

- like 'CP' for trees & Lasso/Ridge for linear regression.
- Popular in Kaggle

Overfitted models not good for boosting  $\Rightarrow$  Data is already very good and do not need to boost.

If  $\alpha = 1$ , we may not get good convergence, like step-factor in gradient computation. So, tune ' $\alpha$ ' carefully.

Boosting is a general thought applicable to many models like tree, linear regression, logistic regression etc.

↙  
(eg) blackboost — n\_estimators, max\_depth  
(#Trees) (MaxTreeDepth)

C5.0 — improved over C4.5 — Boosting added.

### Extreme Gradient Boosting

xgbLinear (boosting or linear regression)  
 — nrounds (# boosting iterations) — boosting parameter  
 — lambda (L2 regularization) { 'elastic' } — L1+L2 for linear models  
 — alpha (L1 regularization)  
 — eta (Learning Rate) — boosting parameter.  
 Shrinkage

### XgbTree

- nrounds
- max\_depth
- eta (Shrinkage)
- gamma (Minimum loss reduction) — individual tree control
- colsample\_bytree, min\_child\_weight — tree control params.

### Stochastic Gradient Boosting

gbm (boosting without regularisation)  
 — n\_trees (# Boosting Iterations)  
 — interaction\_depth (Max Tree Depth)  
 — shrinkage  
 — n\_minobs\_in\_node (Min Terminal Node Size)

AdaBoost — of historical interest only — only for classification

Classification done better with Gradient Boost or XGB.

Classification with gradient boost  $\rightarrow$  only 1 column has 1.

$x$	$y$	$x$	$y_1, y_2$
5	1	5	0 1
7	0	7	1 0
9	1	9	0 1
6	1	6	0 1
8	0	8	1 0

$f(x)$   $\leftarrow$  Logistic Regression  
Shallow trees

target variable  $\rightarrow$  vector

$\downarrow$   
first thing to do for classification with boosting

$\downarrow$   
Use probability values.

Round 1			$x$	$y, \hat{y}$	$\hat{y}_1, \hat{y}_2$	prob. of 0	prob. of 1
5	1	1	5	0 1	0.3 0.7	$\rightarrow$ try to decrease $\hat{y}_1$ , increase $\hat{y}_2$	
7	0	1	7	1 0	0.7 0.3	$\rightarrow$ try to increase $\hat{y}_1$ , decrease $\hat{y}_2$	
9	1	0	9	0 1	0.6 0.4		
6	1	1	6	0 1	0.2 0.8		
8	0	0	8	1 0	0.9 0.1		

$(\hat{y}_1 - \hat{y}_1, \hat{y}_2 - \hat{y}_2) \rightarrow$  input for round 2

This approach here on  $\swarrow$  is more like the 'regression' methodology.

23/12/16

To win Kaggle competitions, need to 'tune' parameters.

For method = "treebag", only tunable parameter is 'ntree'.  
resampling-strategy = trainControl (method = "oob").

No need for "cv" with treebag/Random Forest.

Random Forest = Bagged Tree + randomisation at split level.

For Splitting a node, pick the best variable among randomly selected features.

"gbm" - boosting + no regularization.

- interaction.depth = tree depth.

- shrinkage = c(0.1, 0.3, 1)

- n.trees.in.node = 5, preparing strategy - don't split node if < 5 samples.

If we give 'small' shrinkage, increase the number of trees.

Plot (knn model)

Plot (gbm model) - gives great picturisation

method = "xgbLinear" - extreme boosting or linear models.

Know "Variable Importance" across models so we can play around with them.

$\hookrightarrow$  Remove all the "low importance" variables and re-build

Plot (VarImp(gbm model))

Scanned by CamScanner

For damps, gbm, xgb give better performance than Ada, C5.0.

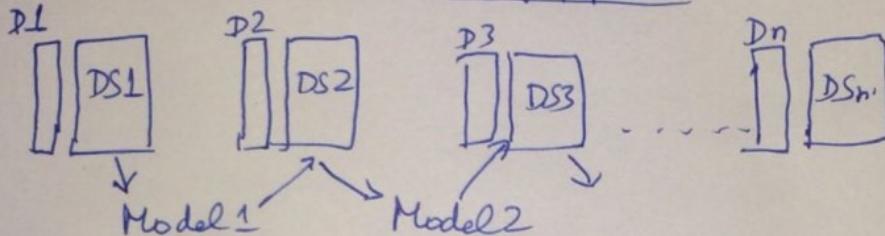
26/12/16

Linear regression regularized using Lasso/Ridge/Elastic Net.

Boosted models  
gbm  
Xgboost } Both classification & regression  
AdaBoost } Only classification

Objective-based  
- Logistic regression } Only classification  
- SVM  
- Perceptron  
- Neural Network } Both classification & regression

Learning AdaBoost since its very popular



$D_i$  = distribution of weights. Sum of weights = 1.

$D_1 = (y_1, y_2, y_3, \dots)$  - all have equal weights at start

At later  $D_i$  ( $i > 1$ ), weights of correct classifications is increased  
weights of wrong classifications is decreased

Regularize AdaBoost by the number of iterations.

AdaBoost  $\rightarrow$  Adaptive Boosting : algorithm adapts weights on base learners and training examples.  $\rightarrow y_i h_k(i) = +1, \rightarrow$  correct class  
 $\rightarrow y_i h_k(i) = -1, \rightarrow$  misclassification

For AdaBoost to work,  $y_i$  input and output  $h_k(i)$  should be +1 or -1 only. If we use (0,1) instead of ~~(0,1)~~ (-1,1), formula won't work.

\* See PPT for details of AdaBoost.  $\rightarrow$  Ada works only for binary classification.  
rpart - CART - does weighted learning. Multiply weights with entropy.  
So, Ada ~~works~~ works on top of both CART and C4.5, but normally  
CART is used.

$$\text{Sign}(x) = \begin{cases} +1, x \geq 0 \\ -1, x < 0 \end{cases}$$

\* Why not gradient boost on top of Ada for classification?  
Gradient boost itself does classification. So no need for AdaBoost.  
Perhaps an ensemble of 'gradient boost' and 'Ada boost' useful

\* Why not gradient boost on top of KNN?  
Gradient boost needs underlying 'weak models' (ie, high bias).  
But KNN is low bias since it's doing 'average' on k-nearest observations

For damps, gbm, xgb give better performance than Ada, C5.0.

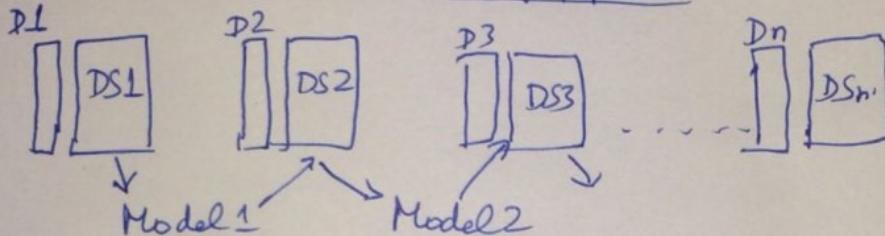
26/12/16

Linear regression regularized using Lasso/Ridge/Elastic Net.

Boosted models  
gbm  
Xgboost } Both classification & regression  
AdaBoost } Only classification

Objective-based  
- Logistic regression } Only classification  
- SVM  
- Perceptron  
- Neural Network } Both classification & regression

Learning AdaBoost since its very popular



D<sub>i</sub> = distribution of weights. Sum of weights = 1.

D<sub>1</sub> = (1/n, 1/n, 1/n, ...) - all have equal weights at start

At later D<sub>i</sub> ( $i > 1$ ), weights of correct classifications is increased  
weights of wrong classifications is decreased

Regularize AdaBoost by the number of iterations.

AdaBoost  $\rightarrow$  Adaptive Boosting : algorithm adapts weights on base learners and training examples.  $\rightarrow y_i h_k(i) = \begin{cases} +1, & \text{correct class} \\ -1, & \text{misclassification} \end{cases}$   $h_k(i)$  should be +1 or -1

For AdaBoost to work,  $y_i$  input and output  $h_k(i)$  should be +1 or -1 only. If we use (0, 1) instead of ~~(0, 1)~~ (-1, 1), formula won't work.

See PPT for details of AdaBoost.  $\rightarrow$  Ada works only for binary classification.  
rpart - CART - does weighted learning. Multiply weights with entropy.  
So, Ada ~~works~~ works on top of both CART and C4.5, but normally  
CART is used.

$$\text{Sign}(x) = \begin{cases} +1, & x > 0 \\ -1, & x < 0 \end{cases}$$

Why not gradient boost on top of Ada for classification?  
Gradient boost itself does classification. So no need for AdaBoost.  
Perhaps an ensemble of 'gradient boost' and 'Ada boost' useful

Why not gradient boost on top of KNN?  
Gradient boost needs underlying 'weak models' (ie, high bias).  
But KNN is low bias since it's doing 'average' on k-nearest observations

27/12/16  
How weights of samples used in AdaBoost

$$\text{Entropy} = \sum p_i \log p_i$$

Instead of  $p_i$ , use the weights.

Only 'trees' used for AdaBoost as the weights can be factored in easily.

For parallel processing,

library (doParallel)

cluster = makeCluster(detectCores())

registerDoParallel(cluster)

\* Lasso & Ridge Regularisation applicable to both linear and polynomial regressions.  
(close to zero)  
In linear regression, Lasso 'eliminates' useless features.  
In polynomial regression, Lasso 'smoothes the curves'.

28/12/16 Regularisation - control of overfitting

When does overfitting happen?

- 1) Tree Learning - Very deep trees  $\Rightarrow$  learnt from noise. Trees look very different for small variations of train data.  
(high variance). Low train error, high validation error.  
Solution is 'pruning' by controlling 'cp'.
- 2) 'Bagging' ensemble - Overfitting implicitly handled by bootstrap
- 3) 'Random Forest' ensemble - Overfitting handled even more efficiently by bootstrap
- 4) 'Boosted' ensemble - Overfitting controlled by 'tree depth' and underlying(Tree)  $\rightarrow$  'fraction of sample' used to grow each tree.  
(Linear regression)  $\rightarrow$  Lambda (L2) & Alpha (L1).

- 5) Probabilistic learning - NB not impacted much by overfitting  
 $\text{Prob} = p_1 \times p_2 \times p_3 \times \dots \times p_n$ .  
Even if one  $p_i$  is overfitted, doesn't impact overall product.

- 6) Objective-based learning:-

Linear Regressor - Large number of features  $\Rightarrow$  overfitting

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \dots$$

With more features, model learns local data - high variance - overfitting.

Control overfitting by restricting the feature co-efficients, i.e.,  $w_i$  values  
-  $w_i$  values look similar across resamples.

(L1) Ridge constraint:

$$\sum_{i=1}^n w_i^2 \leq S \quad , \quad n = \# \text{ features}$$

(L2) Lasso constraint  $\sum_{i=1}^n |w_i| \leq S$  ; If 'S' too low  $\Rightarrow$  underfitting

The idea of L1, L2 regularisation is applicable in many areas - linear regression, logistic regression, SVM, NN.

'fractio' in "lasso" (linear) regression.

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 + \text{fractio} \left( \sum_{i=1}^n |w_i| \right).$$

L1-SVM, L2-SVM; L1-logistic regression, L2-Neural Network

### 29/12/16 Supervised vs Unsupervised learning vs Reinforcement learning

Supervised learning - Learning driven by labelled 'target' columns.

Unsupervised learning - No 'target' column - No prediction - Only pattern detection

(eg) Many features/columns present. Are they all required?  
Can we eliminate some of them?

↳ "Feature Reduction" / "Feature Transformation" / Dimensionality Reduction  
- Data context should not be lost (eg) 3D  $\rightarrow$  2D image  
 $\cdot$  VOB  $\rightarrow$  MP4 file

(eg<sub>2</sub>) Clustering

- find group of customers who are identical. Create a scheme to attract this group.
- download internet content; group them by category

(eg<sub>3</sub>) Outlier detection

- In restaurant revenue problem, some revenues are unusually high; detect them.

(eg<sub>4</sub>) Feature extraction from unstructured data

- lies on image, identify interesting characteristics.
- Show interesting insights into data.

~~Reinforcement learning~~ - another category like supervised, unsupervised.

Frequent pattern mining - Supervised / Unsupervised - Subjective choice

Q. In predictive analytics, use supervised learning in preprocessing stage. No rules. Mix and match the approaches.

Unsupervised algorithms are more difficult than supervised algorithms.

Quantum computing - Much more powerful hardware.  
1 year learning can be learnt in 1 month.

Q. Unsupervised learning - Can be an apple by itself and not necessary to be a part of supervised learning. (eg) extract "car number" from image.

In reinforcement learning, there is interaction with environment. So, keep learning from interactions. It involves 'more extra' on top of unsupervised learning.

Principal Component Analysis (PCA) - for feature extraction.

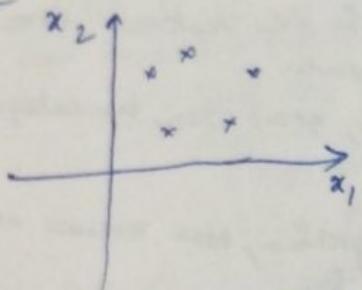
PCA works only on numerical data. So, transform categorical to numeric (using dummy vars) before using PCA.

PCA gives error on 'zero variance' features. Remove them first.

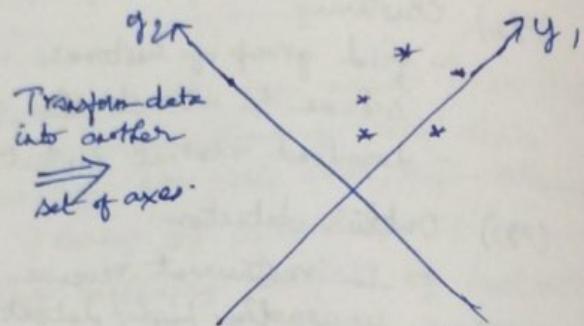
Transform/Reduce both train and test data using PCA. Then proceed with the usual process.

- ④ Decide on transformation logic using 'train' and apply same logic on 'test' - even if we have more 'test' data than 'train'. Doing otherwise is against ML approach - Some unsupervised algs. have target. Try for representative train data.
- In Kaggle, binding rows of train & test and doing feature engineering  $\Rightarrow$  people will get knocked out in private leaderboard.

30/12/16



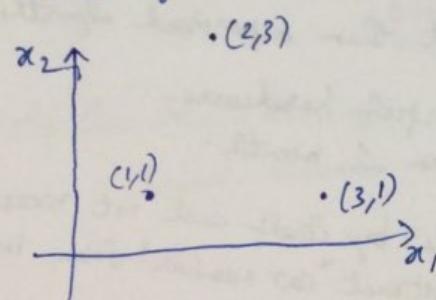
$x_1, x_2$  are not prioritised  
All are equally important



$y_1$  most important (Principal component)  
 $y_2$  second most imp. (Principal component)  
- By ordering by importance, can ignore the least important features.  
 $PC_1 > PC_2 > PC_3 > \dots > PC_{1000}$

Patter in data - Quantification

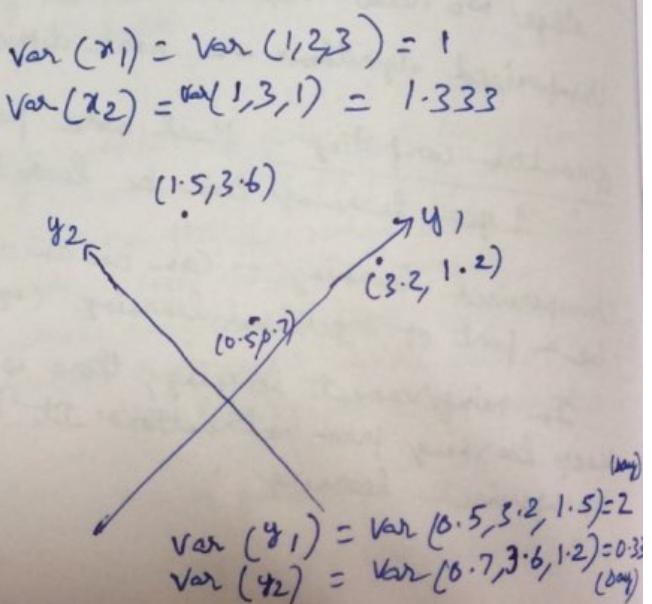
- Sum of variances of data along each dimension



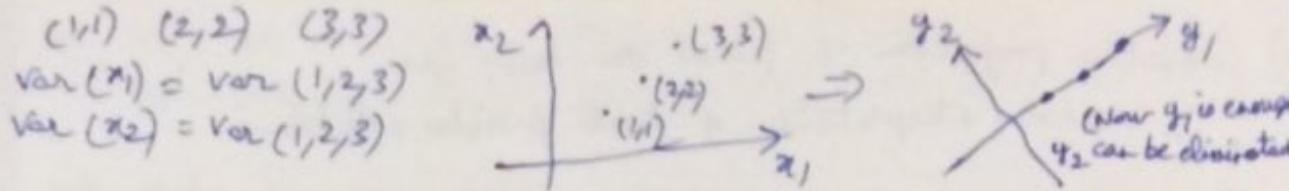
$$(1,1) = 1i + 1j$$

$$\begin{aligned} &\text{Var}(x_1) + \text{Var}(x_2) \\ &= \text{Var}(y_1) + \text{Var}(y_2) \\ &= 2.33. \end{aligned}$$

Now, can ignore  $y_2$ . We have a 1-D axis of  $y_1$  only.



Scanned by CamScanner



Use 'rotation matrix' to transform X axes to Y axes co-ordinates  
co-ordinates

$$(ex) y_1 = 0.2x_1 + 0.5x_2$$

$$y_2 = 0.001x_1 + 0.0002x_2$$

Popular PCA threshold range is [0.95, 0.99].

If threshold = 0.5, 50% variance eliminated. Most data is gone.  
Leads to underfitting.

### PCA application

- Genetics - thousands of genes, millions of DNA polymorphisms
- Documents - thousands of words, millions of bigrams
- Images - millions of pixels.

### Principal Components

Find the new coordinate axes that capture the maximum variance and co-variance

### How it works?

Total variance along X1 and X2 = (= Total variance of Y1 and Y2)  
Variance of points along X1 + Variance of points along X2.

In Y axis, the  $y_1$  coordinate captures the max. variance

### PCA assumption

Makes sense only if there exists high correlation among some of the features.

By rotating them along high-correlation axis, we capture most points along the axis.

If data is spread out, PCA fails. No matter how we rotate data, data is still spread out. So, "linear PCA".

feature reduction in non-linear / low-correlation data, use "Autoencoder" algo of neural network; etc, non-linear PCA.

### Understanding PCA

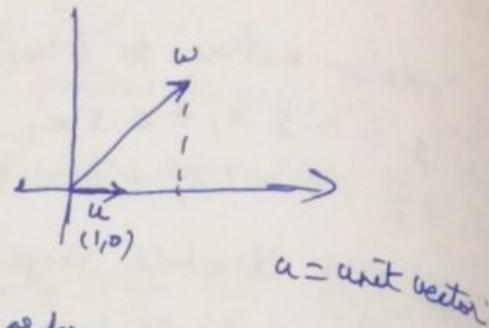
- How to project a point on a different axis?
- How to find axis that captures variances in descending order?

- 1) Computing projection of points on new axis  
 - Vector interpretation of points provides solution.

Find projection of  $w$  vector along ' $u$ ' direction

$$= w \cdot u \quad (\text{dot product} = \text{shadow})$$

$$= w \cos \theta = w_1 u_1 + w_2 u_2$$



- 2) find variances / PCA in descending order

- Eigen vectors provide an answer to this problem.

PCs are the Eigen vectors.

Formulating PCA as optimization: Maximizing variance among vector  $v$ .

Maximize  $\text{var}(v) = v^T A v$  subject to  $\underbrace{v^T v = 1}_{\Rightarrow \text{unit vector.}}$

$$\text{var}(v, \lambda) = v^T A v - \lambda(v^T v - 1)$$

Apply calculus and solve for  $v$  and  $\lambda$ .

$$v = [0, 1] \quad v^T = [0, 1] \Rightarrow v^T v = 1$$

$$v = [1, 0] \quad v^T = [1, 0] \Rightarrow v^T v = 1$$

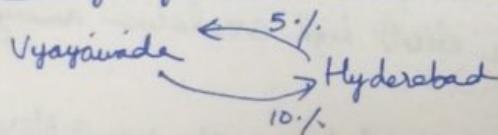
$v^T A v$  = Variance of all data points along  $v$  direction

Search Engine Optimization (SEO) for improving "Page Rank"

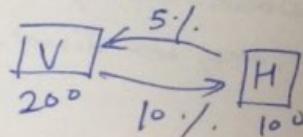
- involves hacking of page rank algorithm

Page ranking uses Eigen vectors

2/1/17 Yearly migration pattern between 2 cities



- What will be population in two cities after  $n$  years, given the initial population vector?
- What will be the long-term behavior of the above system?
- If such a long term behavior of system can be determined, does it depend on the initial population vector?



$$H_0 = 100, \quad V_0 = 200 \quad (\text{initial population})$$

$$H_1 = 100 \times 95\% + 200 \times 10\% = 115$$

$$V_1 = 200 \times 90\% + 100 \times 5\% = 185$$

$$H_2 = 115 \times 95\% + 185 \times 10\% = 127.75$$

$$V_2 = 185 \times 90\% + 115 \times 5\% = 172.25$$

$$H_{t+1} = 0.95 H_t + 0.1 V_t$$

$$V_{t+1} = 0.05 H_t + 0.9 V_t$$

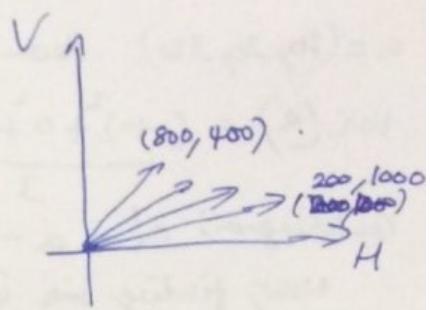
$$\left| \begin{array}{l} H_{t+1} \\ V_{t+1} \end{array} \right| = \begin{bmatrix} 0.95 & 0.1 \\ 0.05 & 0.9 \end{bmatrix} \begin{bmatrix} H_t \\ V_t \end{bmatrix}$$

After many years, equilibrium population

If  $H_0 = 500$ ,  $V_0 = 1000$ , Stability will be  $H=1000$ ,  $V=500$ .  
 $H_0 = 200$ ,  $V_0 = 1000$ , Stability will be  $H=800$ ,  $V=400$ .

After each  $\begin{bmatrix} H_{t+1} \\ V_{t+1} \end{bmatrix}$  iteration, magnitude &

direction will keep changing. But after the equilibrium is reached, magnitude & direction remain unchanged.



Eigen vector = one whose direction remains unchanged.

Ideally,  $M\vec{V} = 1 \cdot V$ .  
 ↳ Eigen vector. No change in direction & magnitude

Practically,  $MV = \lambda V$ .  
 ↳ Eigen value = impacted by  $\lambda$ . No change in direction. Magnitude

How does Google model Eigen Vectors?

After people start visiting websites, they keep jumping websites and finally land on some important pages - these important pages have high page ranks.

People jumping across websites  $\equiv$  matrix multiplications leading to finally going to important sites to stability & giving 'stable' eigen vector.

In general, a matrix acts on a vector by changing both its magnitude and its direction.

If a vector's direction is not imparted by a matrix, that vector is the matrix's eigen vector.

Stability  $\equiv$  Eigen value of 1.

In PCA context,

100 features  $\Rightarrow$  100 dimensions  $\rightarrow$  100 eigen vectors.

Eigen vectors with max. eigen values are the most important principal components.

$$f(V, \lambda) = V^T A V - \lambda (V^T V - 1)$$

$$\frac{\partial f}{\partial \lambda} = V^T V - 1$$

$$\frac{\partial f}{\partial V} = 2AV - 2\lambda V$$

$n \times n$  matrix has 'n' eigen vectors.

In PCA, we need to understand what 'A' is.  
 $A$  is a  $n \times n$  matrix where  $n = \# \text{features/dimensions}$ .

$$\begin{aligned} \text{PCs are eigen vectors of matrix } A. \\ V^T V - 1 = 0 \Rightarrow V^T V = 1 \\ 2AV - 2\lambda V = 0 \\ \Rightarrow AV = \lambda V \end{aligned}$$

↑ eigen value

$$(1) \vec{x} \cdot \vec{v}, (2) \vec{x} \cdot \vec{v}, (3) \vec{x} \cdot \vec{v}, \dots$$

$$\bar{a} = (10, 20, 30) \quad \text{Mean } (\bar{a}) = 20$$

$$\text{var}(a) = \frac{(-10)^2 + 0^2 + (10)^2}{3} = 66.67.$$

$$\text{centering } a: a' = a - \bar{a} = (-10, 0, 10)$$

Now, finding var is easier = .

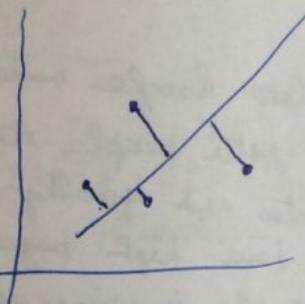
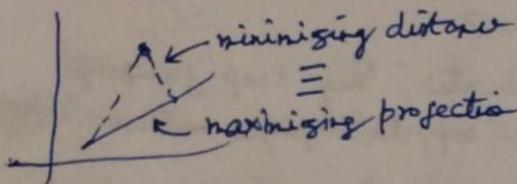
$$(1) (\vec{x} - \bar{x}) \cdot \vec{v}, (2) (\vec{x} - \bar{x}) \cdot \vec{v}, (3) (\vec{x} - \bar{x}) \cdot \vec{v}, \dots$$

$$\text{Variance} = \frac{\sum_{i=1}^n (x^{(i)} - \bar{x})^2}{n} = \vec{v}^T A \cdot \vec{v}$$

$\vec{x} \cdot \vec{v} = \text{Projection of } x \text{ vector } v,$

$\hookrightarrow A = \text{Covariance matrix}$

3/1/2017 Meaning of capturing variance



Eigen values of eigen vectors determine the importance of the principal components.

$$\begin{array}{cc} x_1 & x_2 \\ 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{array} \quad \begin{aligned} \text{covar}(x_1, x_1) &= \frac{1}{3} [(1-2)^2 + (2-2)^2 + (3-2)^2] \\ \text{covar}(x_2, x_2) &= \frac{1}{3} [(2-3)^2 + (3-3)^2 + (4-3)^2] \\ \text{covar}(x_1, x_2) &= \frac{1}{3} [(1-2)(2-3) + (2-2)(3-3) + (3-2)(4-3)] \\ &= \text{covar}(x_2, x_1) \end{aligned}$$

Covariance matrix

$$\begin{bmatrix} \text{covar}(x_1, x_1) & \text{covar}(x_1, x_2) \\ \text{covar}(x_2, x_1) & \text{covar}(x_2, x_2) \end{bmatrix} \rightarrow \text{Symmetric always nice} \\ \text{covar}(x_1, x_2) = \text{covar}(x_2, x_1)$$

$$\begin{bmatrix} \text{covar}(x_1, x_1) & \text{covar}(x_1, x_2) & \text{covar}(x_1, x_3) \\ \text{covar}(x_2, x_1) & \text{covar}(x_2, x_2) & \text{covar}(x_2, x_3) \\ \text{covar}(x_3, x_1) & \text{covar}(x_3, x_2) & \text{covar}(x_3, x_3) \end{bmatrix}$$

$$\text{var}(v) = \sum_{\text{each point } x} [(x - \bar{x})^T \cdot v]^2$$

$$\text{var}(v) = \sum [(\vec{x}^{(i)} - \bar{x})^T \cdot v]^2$$

$$= \sum_i \vec{v}^T (\vec{x}^{(i)} - \bar{x}) (\vec{x}^{(i)} - \bar{x})^T \vec{v}$$

$$= \vec{v}^T \left[ \sum_i (\vec{x}^{(i)} - \bar{x}) (\vec{x}^{(i)} - \bar{x})^T \right] \vec{v}$$

$$= \vec{v}^T A \vec{v} \quad \text{where } A = \sum_i (\vec{x}^{(i)} - \bar{x}) (\vec{x}^{(i)} - \bar{x})^T$$

## PCA as Optimization

Maximize  $\text{var}(v) = v^T A v$  subject to  $v^T v = 1$

↓  
Done to constrain  $v$ ' values. Else variance can be increased arbitrarily by scaling the values.

$$\text{Maximize } f(v, \lambda) = v^T A v - \lambda (v^T v - 1) = 0$$

$$\frac{\partial f}{\partial \lambda} = -v^T v + 1 = 0$$

$$\frac{\partial f}{\partial v} = 2Av - 2\lambda v = 0$$

$$\Rightarrow Av = \lambda v$$

⇒  $v$  is eigenvector of  $A$ ,  $\lambda$  is eigen value

$$\lambda_1 = 73.718, \lambda_2 = 0.384, \lambda_3 = 0.298$$

↓  
Keep this. Can ignore others.

Under maxima,

$$\text{var}(v) = v^T A v = v^T (\lambda v) = \lambda \quad [\because v^T v = 1]$$

Eigen vector with largest eigen value  $\lambda_1$  is the 1<sup>st</sup> principal component.

PCA cannot remove the impact of outlier - PCA also makes sense only when there is linear relationship among the features.

Using co-variance matrix

1. Find mean :  $\mu = \frac{1}{n} \sum_{i=1}^n x_i$

2. Compute covariance matrix

$$C = \frac{1}{n} X X^T = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)(x_i - \mu)^T$$

3. Find eigen values of  $C$  and arrange them into descending order,  $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_d$  with the corresponding eigenvectors  $(u_1, u_2, \dots, u_d)$

4. The principal component matrix is :  $U = \begin{bmatrix} 1 & 1 & \dots & 1 \\ u_1 & u_2 & \dots & u_d \\ 1 & 1 & \dots & 1 \end{bmatrix}$

💡 train "pca" uses the Singular Value Decomposition (SVD) algorithm instead of using co-variance matrix to compute the eigen vectors and eigen values.

Clustering : Another unsupervised algorithm

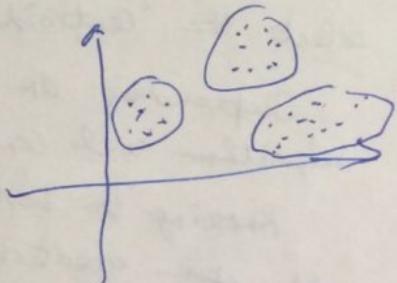
1) clustering - customer segmentation.

The cluster of customers has some close similarity.

Humans can't do this as we cannot deal with 3+ dimensions.

2) News article clustering - Downloaded multiple news articles. Cluster them as 'Sports', 'Entertainment', 'Business' etc.

These 'labels' are not given.

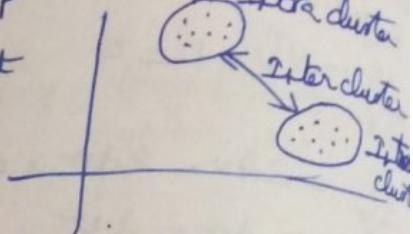


- 3) Correlated variable clustering  
 - Corplot used for 'restaurant revenue' problem

4/1/17

Cluster group - Points that have strong 'intra' relationship and weak 'inter' relationship.

Outlier detection - Find observations that look very different from others



Clustering - Intra-cluster distances are minimized  
 Inter-cluster distances are maximized

$$\text{Similarity } L = \frac{1}{\text{Distance}}$$

K-means clustering algorithm

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Objective to minimize the sum of distances of the points to their respective centroid
- Number of clusters, K, must be specified

Alg

- 1) For  $K=3$ , "randomly" select 3 points as centroids.
- 2) For each of the remaining points, find the centroid that is closest to it, and assign it to that centroid's cluster.
- 3) In each cluster,  
 re-elect a new center from that cluster. That new center (centroid) could be non-existent.
- 4) Repeat steps 2-3 with the new centroids.
- 5) Repeat the above steps until the membership ( $\xrightarrow{\text{to}} \text{cluster}$ ) of points gets stabilized.

There is mathematical proof that distance is minimized when we select the 'centroid' of cluster.

Depending on what the 'initial' centroids are, the algorithm will converge faster or slower.

Knowing the best 'K' value is a big challenge and remains an open question - How to evaluate the clustering done?

Cohesion - measured by within cluster sum of squares

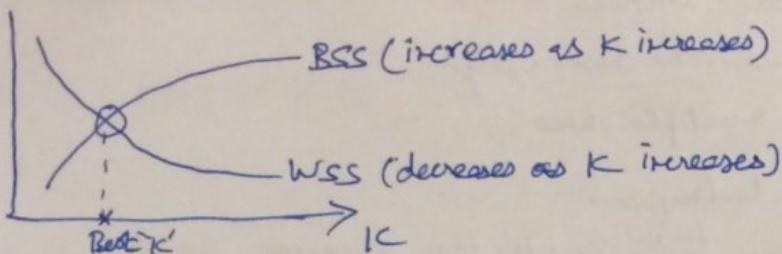
$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

$(WSS = \text{Within sum of squares})$

Separation - measured between cluster sum of squares.

$$BSS = \sum_i |C_i| (m - m_i)^2. \quad [BSS = \text{Between sum of squares}]$$

where  $|C_i|$  is size of cluster  $i$ ,  $m$  is the centroid of the whole data set.



- $BSS + WSS = \text{constant}$
- WSS (within) measure is called Sum of Squared Error (SSE) - a commonly used measure
- A large number of clusters tend to result in smaller SSE.

(\*) Clustering could be a part of feature engineering. (eg) Text data could be clustered and the 'cluster number' could become a new feature.

Text data  $\rightarrow$  Transform it to structured matrix-like data and run clustering on it.

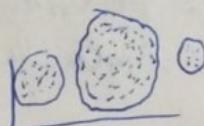
Deep neural network (Word2Vec) converts a text document into vector format.

Image2Vector - Converts Images to Vector Graphics.

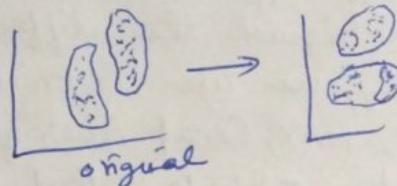
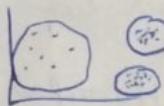
Understanding video content - Videos are automatically captured

Caret = Classification + Regression

### Limitations of K-means



- Different cluster sizes
- Different densities in different clusters  
(Overcome using DBScan algorithm using density as the core metric)
- Non-globular shapes not detected
- problems with outliers.



Use many clusters & combine hardly some of the clusters to overcome some of the issues

### clustering

Hierarchical approach  
(eg) Hclust

Density based  
(eg) DBScan

Partitioning approach  
(eg) K-means

In hierarchical clustering, we don't need to know 'K' beforehand.

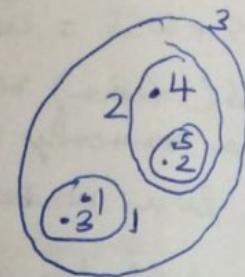
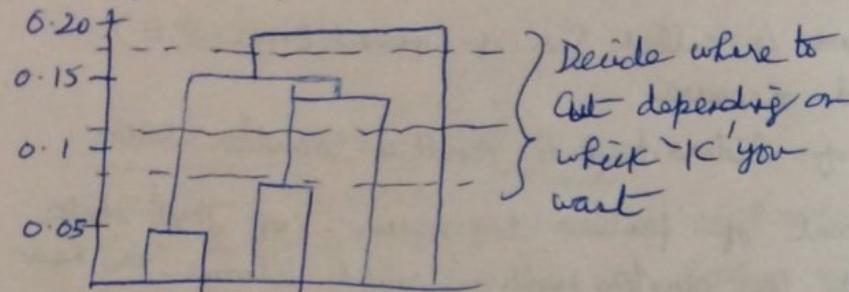
- Produces a set of nested clusters organized as a hierarchical tree.

algo (Agglomerative clustering)

1. Each point is its own cluster
2. Merge clusters which are close enough
3. Repeat step 2 multiple times

- Visualized as a dendrogram

- Tree-like diagram that records the sequence of merges or splits



### Strengths

- K not needed beforehand
- may correspond to meaningful hierarchies

Agglomerative clustering algorithm (More popular hierarchical clustering)

1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. Repeat
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. Until only a single cluster remains
- Note: Only one merge happens at a time

Key operation is the computation of the proximity of two clusters.

- Diff. approaches to defining the distance between clusters distinguish the different algorithms

Usually, we use 'centroid' based approach. Other approaches are MIN, MAX, Group Average, Other methods driven by an objective function - Ward's method uses squared error; (most popular)

The hierarchical clustering approach suffers from many of the limitations of K-means, except that 'K' need not be specified beforehand.

Even in hierarchical clustering, diff. measures like MIN, MAX, average etc. give different clusters.

Scatterplot (Correlation, order = "hclust")

K-means, despite of its limitations, is most popular due to its simplicity.

use the R method `kcluster()`

"Factor Analysis" not used any more. PCA better. Even PCA is being replaced by auto encoders.

9/11/2017

### Recommenders

Rating prediction

Top-N Recommendations

Neural Turing Machines - It progress to automate 'developer' roles but far away from practicality yet.

Manager/HR roles much easier to automate

Neural networks can understand 'voice' and convert to text. So, call center jobs are being removed.

'Narrow domains' can easily be automated

Read a resume and understand it. Understanding unstructured data.

### Descriptive analytics → understanding data

- clustering
  - Outlier detection
  - Dimensionality reduction
  - association analysis
- } unsupervised ML  
} - PCA, K-means clustering

Descriptive analytics like clustering, dimensionality reduction, outlier detection can also be a part of (preprocessing) predictive analytics.

Recommendations are a core part of business. (e.g.) YouTube, Amazon Shopping, Insurance policies, Car buying, Netflix - similar movies.  
- Suggest top N items based on similarity.

### Rating prediction (Recommendation type)

Suppose I gave ratings to 3 movies. Then machine predicts my rating to 4 other movies and suggests movie with highest predicted rating.

### Another recommendation type

If we only have info on what 3 movies I watched (but not my rating), we can still recommend.

Recommenders are more predictive than descriptive because we are 'forecasting', for eg, what movies I will watch.

Associative analysis - What things do people buy together? (Market basket analysis) Knowing this, we can place those things together in a shopping mall.

Handled using 'Apriori' algorithm

Using associative analysis also, we can 'recommend' the 'associated thing' to buy after a person buys one thing.

Reinforcement ML - For agent-based learning. Not just deployment  
- Not seen in this course (eg) car driving, game playing / do more.

### Recommenders

- Filter-based product search using Brand, Optical zoom, Mega Pixel, etc are OK but difficult for humans due to information overload.

Better to recommend products.

- Personalized recommendations - model user preferences.  
(eg) facebook homepage

### Benefits of Recommenders

- Serendipity - accidentally discover something precious fortunately
- Easier search
- Improved cross-selling

### Rating prediction ; Solution approaches

- \* Adhoc predictors - random recommendations; easiest - rarely used
- \* Collaborative filtering based predictors (eg) Amazon
  - user-user based } by combining movies watched by other users
  - item-item based } and matching users by taste
- \* Content based predictors (eg) movie type, actors etc to predict rating
- \* Latent factor based predictors (eg) Netflix
  - understanding the 'mind'; hidden factors
- \* Hybrid predictors - mix of some of them

Use 'R' package 'Recommenderlab' - not production quality yet.

10/11/2017

### Recommender

- Rating prediction (rating of not watched movies)
- top N recommendation (Recommend buys based on what is bought)
  - ↳ done based on 'rating prediction' output.

## Mean based predictors

- Global mean rating
- Per-user mean rating
- Per-movie mean rating
- Random rating

} Adhoc predictors. Low Variability. Underfitted

## Collaborative filtering approach

	M <sub>1</sub>	M <sub>2</sub>	M <sub>3</sub>	M <sub>4</sub>	(rating 1-5 scale)
User <sub>1</sub>	5	4	?	-?	
User <sub>2</sub>	-	3	5	-	
User <sub>3</sub>	-	-	5	5	
User <sub>4</sub>	4	-	4	3	

→ Rating matrix

Assumption  
People who agreed in the past are likely to agree again

Find out how 'similar' users rated the M<sub>4</sub> movie.

Here, User<sub>2</sub> and User<sub>4</sub> are closer to User<sub>1</sub>.  
(Weight<sub>2</sub>)      (Weight<sub>4</sub>)

↳ Similarity with User<sub>1</sub>.

$$5 \text{Weight}_2 + 4 \text{Weight}_4 = \text{User}_1 \text{ M}_3 \text{ rating}$$

To suggest movie for User<sub>1</sub>, predict rating for M<sub>3</sub> and M<sub>4</sub>.  
Recommend the highest predicted movie.

- \* Predict rating for an item for an user u, give weight to each other users rating based on how similar they are to user 'u'. Need to compute similarity matrix - gives pair-wise user similarity scores.

The users most similar to User<sub>1</sub> may not have given a rating to a movie - then we cannot predict.

## Rating for item i of user u?

- \* Find k-most similar users of u from user-user similarity matrix

- \* Compute prediction score of item i based on:

$$\text{pred}(u, i) = \frac{\sum_{v \in K} \text{similar users}(u) \text{ userSim}(u, v) \cdot r_{vt}}{\sum_{v \in K} \text{similar users}(u)}$$

$$\left( \sum_{v \in K} \text{similar users}(u) \right) \text{userSim}(u, v)$$

Denominator used to normalize rating in range 1-5.

Can modify formula to include user or item bias factor.

$$\text{ex)} \frac{(0.9 \times 5) + (0.9 \times 4) + (1 \times 5)}{0.9 + 0.9 + 1}$$

## UBCF : Top-N recommendation

Find N items  $\rightarrow$  most likely purchased by user u.

- Find K most similar users to u :  $U_{sim}$
- Get all items purchased by  $U_{sim}$  :  $I_{candidate}$
- Remove unavailable items from  $I_{candidate}$
- Get all items purchased by u :  $I_{purchased}$
- Take  $I_{recom} = I_{candidate} - I_{purchased}$
- Reorder items in  $I_{recom}$  using following formula and recommend first n items :

$$\text{pred}(u, i) = \frac{\sum_{v \in K \text{ similar users}(u)} \text{user}_{uv} \times p_{vi}}{\sum_{v \in K \text{ similar users}(u)} \text{user}_{uv}}$$

### Issues with UBCF

- 1) User cold-start problem : How to find similar users for the new user on-board?

If rating matrix has ~~to~~ only 'boolean' values, use association analysis to find similarity matrix.

- 2) Sparsity - When recommending from a large item set, users will have rated only some of the items. Makes it hard to find similar users - biggest problem

- 3) Item cold-start problem - Cannot predict ratings for new item till some users have rated it.

- 4) Scalability - Similarity scores are dynamic so precomputing is bad. With millions of ratings, similarity computation becomes slow.

11/11/2017

## Item-based Collaborative filtering (IBCF)

Idea - User is likely to have the same opinion for similar items

Similar items - They have similar ratings from many users.

Can the ratings of the target user for similar items be exploited for predicting an unknown rating?

- (\*) Item-based similarity is more stable than user-based similarity. Hence more widely used.

### Advantage of IBCF over UBCF

- Prevents user cold-start problem (with some \* modification)
- User profile normally contains less ratings than a product profile. Hence, addresses sparsity issue
- Improves scalability - item similarity ~~can affect the~~ user similarity; enables precomputing of item-item similarity

Item cold-start still a problem with IBCF although 3 out of 4 issues with UBCF are resolved

### IBCF: rating prediction

Rating for an item  $i$  of user  $u$ :

- \* Find  $k$ -most similar items of item  $i$  from item-item similarity matrix
- \* Compute prediction score of item  $i$  using:

$$\text{pred}(u, i) = \frac{\sum_{j \in \text{Ksimilaritems}(i)} \text{itemsim}(i, j) * r_{uj}}{\sum_{j \in \text{Ksimilaritems}(i)} \text{itemsim}(i, j)}$$

### IBCF : Top-N recommendations

Find  $N$  items most-likely to be purchased by user  $u$

- Let all items purchased by  $u$ :  $I_{\text{purchased}}$
- For each item  $\cancel{\text{purchased}} \in I_{\text{purchased}}$ , find  $k$  most similar items:  $I_{\text{candidate}}$
- Remove unavailable items from  $I_{\text{candidate}}$
- ...

30% of Amazon's sales is based on recommendations.

People who bought this also bought: Basket-analysis / Association analysis

(?) Need refinement of algorithms for each domain/category  
(e.g.) To recommend laptops, which brand to recommend?

Deep-learning is recently being applied to 'recommendations' to find hidden interests of users.

It takes 2 years to get production-grade algorithm of IBCF, because of refinement/ fine-tuning needed.

Content-based approach (outdated - replaced by collaborative filtering)

To attack cold-start issues with collaborative filtering.

Based on static data (e.g.) static profiles of movies and static profiles of users.

Idee: User likely to have similar level of interest for similar items.

- Collect multiple attributes for items and users.

Issues: Need lot of details about item & user

User cold-start

What if user's interest changes?

Lack of serendipity

④ A combination of TBCF and Content-based approach is used in practice.

After Netflix 2002 competition, a game-changer recommendation approach called "Latent factor" evolved.

12/1/2017

Wide format  $\rightarrow [ \quad ]$  = more features less input.

Narrow Long format  $\rightarrow [ \quad ]$  = less features, more input.

Narrow format			wide format		
id	User ID	Movie ID	Rating	$u_1$	$m_1 \ m_2 \dots m_n$
	$u_1$	$m_1$	5	$u_2$	
	$u_1$	$m_2$	6		
	:	:			
	$u_2$	$m_1$			
	$u_2$	$m_2$			

To transform narrow  $\rightarrow$  wide format:

- library (recommender)
- asort() function

'Recommender' doesn't understand 'real-valued' matrix. Need to convert it to "RealRatingMatrix".

Recommenderlib tried to create APIs for all recommendation problems. Design inspired by 'Caret' package.

Java: int  $\equiv$  Integer.

Thus, matrix of real numbers  $\equiv$  RealRatingMatrix

For training [Recommender() fn] and predicting, we use the same input (not like train + test). In prediction, we try to predict the 'NA' values.

⑤ Recommenders & Imputation both deal with missing values. But, because of different domains, our approach / algorithm is different.

⑥ In unsupervised ML, like clustering, evaluation is very hard. So, no evaluation step in unsupervised ML.

For Recommenders, it is like a supervised ML. So, we have some evaluation scheme.

- We make some available (non-NA) ratings as validation set. They are masked (heldout, testing, validation) to use RMSE to evaluate.

For holdout set,

- We mark / holdout only on a subset of users - we mark 'some' of the ratings of the user-subset.  
method = "cross-validation", gives = 2.

### Recommender lab

- separate 'train' and 'evaluate' steps.

cross-validation NOT done for parameter tuning of 'gives'. It is purely to evaluate - 'gives' value based on 'sparsity' of input matrix. 'Similar user/items'.

The hyper parameters to tune are 'n similar user/items'.

Evaluate "Top-N recommendation"

- Order of recommended movie does not matter. Just check if 'liked' movie is recommended or not.
  - gives = 2, goodRating = 2.5 } Neither are hyperparameters; purely for evaluation
    - ↓
    - based on sparsity matrix
    - ↓
    - based on business decision

13/11/2017 To decide or method for distance, like "Cosine", "Euclidean", etc  
we 'evaluate' and find RMSE.

UBCF takes long time to compute similarity scores; Use Spark/h2O.

Recommender () — Computer similarity matrix

`predict()` - uses similarity matrix to predict

## Latest Factor approach -

Try to find hidden factors behind user's ratings  
movie ratings      hidden factor weights

$$\text{movie ratings} = \begin{bmatrix} 10 \\ \vdots \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \end{bmatrix} \quad \begin{array}{l} \text{hidden factor weights} \\ \text{weights of hidden feature of names...} \end{array}$$

10 = cross-product

SVD = Singular Value Decomposition - unfit for our problem since the input is 'sparse' matrix.

## Optimization-based value decomposition -

$$5 \times 6 \text{ matrix} = 5 \times 3 \text{ matrix} \times 3 \times 6 \text{ matrix} \rightarrow \text{Here '3' hidden features}$$

Need to type the '# hidden features'.

In production environment, still 'Collaborative filtering' is used.  
Latent-factor used only in competition.

$$\begin{bmatrix} Y & P & Q \\ \begin{bmatrix} 5 & NA & NA \\ NA & 4 & NA \\ NA & NA & 3 \end{bmatrix} & \xrightarrow{\text{eq}} \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} & \begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 1 \end{bmatrix} \end{bmatrix} \downarrow \rightarrow 6 + 6 = 12 \text{ unknowns.}$$

Use the non-NA values to guess the right-side matrices.

$$f(p_{11}, p_{12}, p_{21}, p_{22}, p_{31}, p_{32}, q_{11}, q_{12}, q_{13}, q_{21}, q_{22}, q_{33}) \\ = [Y_{11} - (p_{11}q_{11} + p_{12}q_{21})]^2 + [Y_{22} - (p_{21}q_{12} + p_{22}q_{22})]^2 + \\ [Y_{33} - (p_{31}q_{13} + p_{32}q_{23})]^2.$$

Minimize 'f' using available values.

Although powerful and found in 2002, not used widely since

poor 'interpretability', can't explain to business heads.

16/11/2017

### Assignment - 1 Recommenders

Euclidean similarity =  $\frac{1}{\sqrt{\text{Euclidean distance}}}$

Problem 1

$$= \frac{1}{\sqrt{(\text{item}_1 \text{ diff})^2 + (\text{item}_2 \text{ diff})^2 + \dots}}$$

Problem 3 SVD (C) method

### Association analysis (Considered unsupervised ML)

- What do my customers buy?
- Which products are bought together?
- Find associations & correlations between diff items that customers place in their shopping basket
- We don't analyse a customer's entire buying history

- 1) Today's recommendation for you - Using Collaborative filtering
- 2) After selecting an item  $\rightarrow$  users who bought this also bought  $\rightarrow$  Based on association analysis

For set of transactions T, goal to find all rules having support  $\geq$  minSup threshold

Metric: Confidence  $\geq$  minConf threshold

### Support (S)

- Fraction of transactions that contain both X and Y;  $\{X\} \Rightarrow \{Y\}$

$$\begin{aligned} \{ \text{bread} \} &\Rightarrow \{ \text{milk} \} \\ \{ \text{soy} \} &\Rightarrow \{ \text{chips} \} \\ \{ \text{bread} \} &\Rightarrow \{ \text{jam} \} \end{aligned}$$

$$\{ \text{people who bought item} \} \Rightarrow \{ \text{also bought} \}$$

### Confidence (C)

- Measures how often items in X appear in transactions that contain X

$$\text{Support}, S = \frac{\sigma(\{\text{Bread, Milk}\})}{\# \text{ transactions}} = 0.38$$

38% of transactions have both bread and milk.

$$\text{Confidence}, C = \frac{\sigma(\{\text{Bread, Milk}\})}{\sigma(\{\text{Bread}\})} = 0.75$$

75% of those who bought bread also bought milk.

Among items, filtration done by 'support'.  
Among rules, filtration done by 'confidence'.

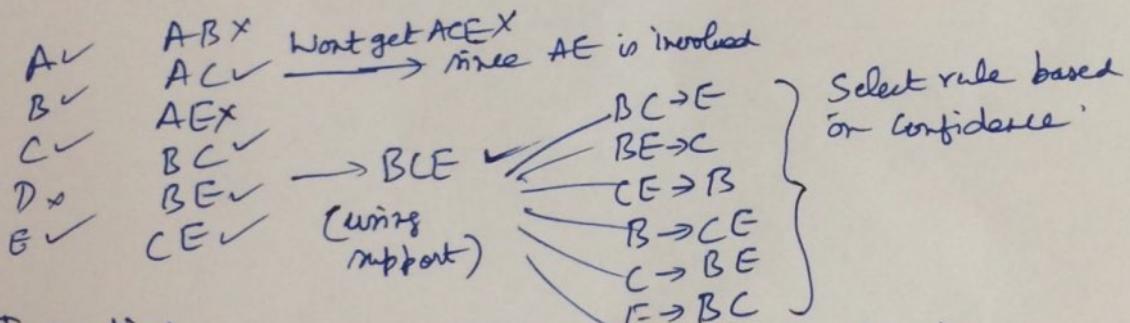
Threshold for support & confidence using 'business Decision' only.  
Not based on Statistics.

### Brute-force approach

- List all possible association rules
- Compute support / Confidence for each rule
- Prune rules that fail thresholds.
- Computationally prohibitive

Apriori (Common-sense way to improve over Brute-force approach)

- If 'Milk' is bought by only 30% people, (less than threshold) then don't consider {Milk, Jan}, {Milk, Yogurt} associations.
- Selection of 'support threshold' has huge impact on computation time.



### Transactions

	Freq Item	Support	Rule	Confidence
ACD			BC → E	100%
BCE	BCE, AC	60%	BE → C	75%
ABCE	BC, CE, A	60%	CE → B	100%
BE	BE, B, C, E	80%	B → CE	75%
ABCE			C → BE	75%
			E → BC	75%

Association analysis also used to place items nearby in supermarkets.

To overcome cold start  
1) Show most popular items.

When user bought B, should we show C or E first?  
- Depends on confidence of B → C vs B → E.

17/1/2017

dplyr useful to preprocess data; almost SQL-like syntax.

Bonnet analysis - quantity not considered since people decide quantity. We only recommend based on association.

18/1/2017

Don't care learning types (Supervised) Recommender

Predictive - Information based

- Probability based
- Ensemble (ex) tree bag, bagging, boosting
- NN learning (Nearest neighbor) (ex) KNN
- Objective based (ex) deep learning strategy

linear regression,  
logistic regression,  
Neural network learning,  
SVM

Unsupervised learning ( $\rightarrow$  target)

- Dimensionality reduction (ex) PCA
- Clustering
- Association analysis
- Outlier detection

Hard to evaluate "Unsupervised learning"

Reinforcement learning

(ex) fully automated cars

- no target
- learn automatically by interacting with environment.

Two main types

- 1) Q-learning
- 2) Policy-based

Objective function

- function (parameters) = ?
- subject to constraints

SVM -

Linear regression

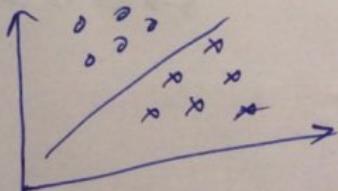
$$y = w_1 x_1 + w_2 x_2 \dots$$

$$f(w_1, w_2) = \text{squared loss} = \sum [y_i - f(w_i, x_i)]^2$$

f(w<sub>1</sub>, w<sub>2</sub>) = squared loss  $\rightarrow$  can be extended for multi-class

SVM & Logistic regression (for binary classification  $\rightarrow$  can be extended for multi-class)

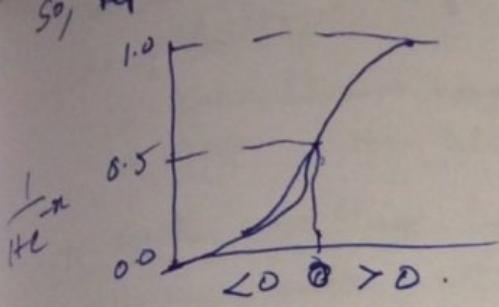
Line-based classification



If line equation is  $f = w_0 + w_1 x_1 = 0$

Then  $f() = 1 \rightarrow \text{class 1}$   
 $= 2 \rightarrow \text{class 2}$

OK, but does not give probability  
that a point is class 1 or class 2.



Line still used to separate points in logistic regression.

But probability interpreted by using sigmoid (like exp value).

What is the objective function for logistic regression?

LR only for linear separation

SVM can be extended to non-linear separation

Trees (CART) can also give 'probability' values in classification  
 - but those are not true probabilities, because  
 - trees are not built with probability mindset

Logistic regression modelled with a 'probability' mindset  
 - understandable from 'objective function'.

Ideal LR separator line:

- all class 1 points on one side
- all class 2 points on other side
- Identify loss function accordingly
- Constraint is  $L_1, L_2$ , or,  $L_1 + L_2$ .

Objective fn : Log loss

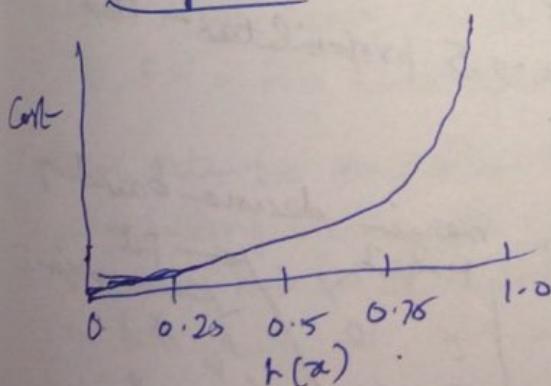
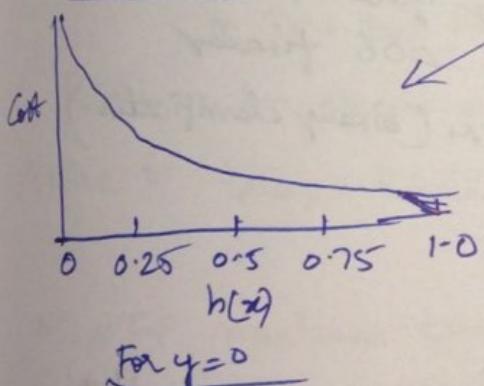
$$\text{Cost}(h(x), y) = \begin{cases} -\log(h(x)) & \text{if } y=1 \\ -\log(1-h(x)) & \text{if } y=0 \end{cases}$$

$h(x)$  = Logistic fn  
 = prediction

For  $y=1$

1) If  $y=1$  and  $h(x)=1$ , Cost=0  
 But for  $h(x)=0$ , Cost=∞.

2) If  $y=0$  and  $h(x)=0$ , Cost=0  
 But for  $h(x)=1$ , Cost=∞



Appeals to intuition

- If correct prediction, no loss
- If wrong prediction, heavy penalty.

$y$  is 0 or 1 only

Combine into a single expression

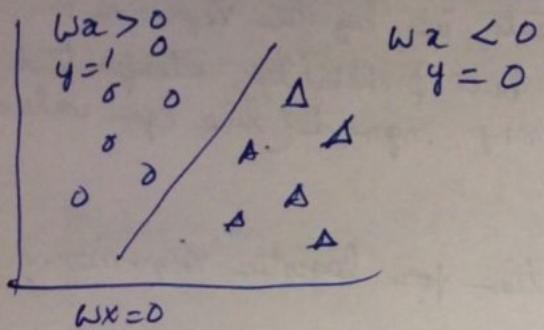
$$\text{Cost}(h(x), y) = -y \log(h(x)) - (1-y) \log(1-h(x))$$

Log loss

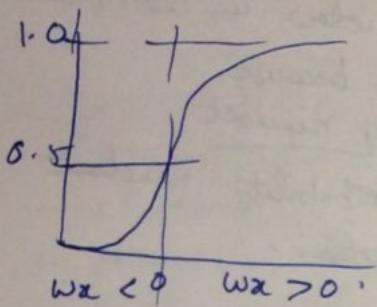
Cross entropy

19/1/2017

Linear discriminator - only linear boundaries



Line fit  $\rightarrow$  How much error do each of the points contribute to the line?



Logistic regression

$$f(w_0, w_1, \dots, w_n) = -y \log \hat{y} - (1-y) \log (1-\hat{y})$$

$$\hat{y} = h(w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n)$$

Support vector machine (SVM)

linear model



Kernel + SVM = non-linear model/boundary

Neural network issue - problem finding global minimum -  
was solved in 2006 finally

For multi-class classification with LR/SVM (binary classification).

for 5-class classification:

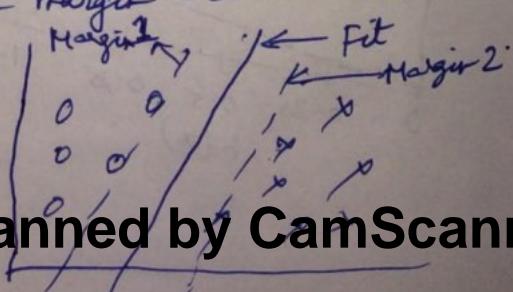
Build 5 models:  $C_1$  vs  $[C_2, C_3, C_4, C_5]$

$C_2$  vs  $[C_1, C_3, C_4, C_5]$

$C_3$  vs  $[C_1, C_2, C_4, C_5]$

Run 'test' against all 5 models  $\rightarrow$  5 probabilities - Max. probability determines class.

SVM goal - Discover maximum margin decision boundary.



(\*) SVM does better classification than LR. But it does not give 'probability'.  
 For probability → Logistic regression }  
 is classifier ← Naive Bayes }

SVM objective → Hinge loss  
 - distances matter, not left or right

NN better than SVM for non-linear classification.

### 20/1/17 Evaluation metrics

Classification — Accuracy  
 Kappa  
 ROC m  
 F-Score

$$\text{False positive rate} = \frac{FP}{TN + FP}$$

Regression — RMSE  
 $R^2$

$$\text{TPR} = \text{Recall} = \frac{TP}{TP + FN}$$

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

Mostly based on Confusion matrix

		classified +ve	classified -ve
Original +ve	Original +ve	True +ve (TP)	False -ve (FN)
	Original -ve	False +ve (FP)	True -ve (TN)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{\text{Total}}$$

$$\text{Error} = 1 - \text{Accuracy}$$

Applied to train, validate, test — Train error  
 Validation error  
 Test error.

FN & FP sometimes treated equally bad.

But, in some cases, FN is costlier than FP.

(e.g.) FN = has heart disease, but we say 'no heart disease'.

- misclassification costs are not same (e.g. FN worse than FP)
- class imbalance datasets used (in 'train' data)  $\downarrow$  biased decision
- test data not representative

- Non-symmetric misclassification cost
  - Matrix often difficult; different for diff circumstances.
  - Should we try a large # of matrices?
- Imbalanced dataset (train)
  - 3762 - No → 3731 (✓) + 32 (✗)
  - 238 - Yes → 229 (✗) + 9 (✓)

serious issue with just accuracy!
- Test data not representative
  - Does not present all cases, edge-cases, use-cases.

### Alternative Metrics

- 1) ROC curve (Receiver Operating Characteristic)
  - discovered for assessing the received 'signal quality' in electronics - curve between FPR & Recall measures (False positive rate)
- 2) PR Curve
  - Curve between Recall and Precision measures.

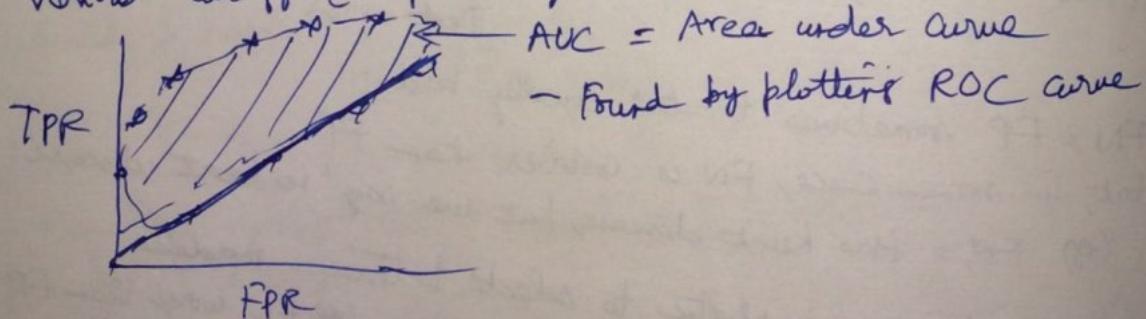
#### ROC Curve

- performance evaluation
- classifier comparison
- does not depend on misclassification cost matrix
- valid with imbalanced data sets
- applicable only for binary predictive classf } Requirements
- applicable when classifier gives probability } Prerequisites

$$TPR \text{ (True-positive rate)} = TP / \text{Positives} = \text{Recall}$$

$$FPR \text{ (False-positive rate)} = FP / \text{Negatives}$$

Graph drawn by computing confusion matrix values using various cut-offs (or probability threshold to classify as true or -ve)

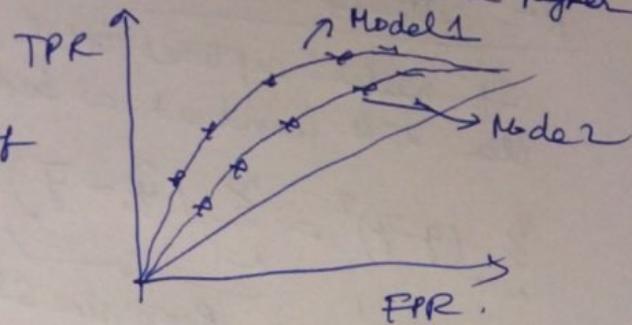


- ✳️ Deglare → It's an objective fn; not a metric.

- AUC - more area is better  
- also predicts which TPR & FPR combination to use  
(TPR max. & FPR min.)

AUC corresponds to prob. of a true instance to have a higher score than a -ve instance.

Model 1 always better than Model 2. There is no cut-off for which model 2 is better than model 1.



23/11/17  
TPR vs FPR

\* TPR = of those that are known as true, how many are predicted as true?

- Higher TPR  $\Rightarrow$  we will miss fewer true data points

\* FPR = of those that are known as -ve, how many are predicted as +ve?

- Higher FPR  $\Rightarrow$  will misclassify more -ve data points

AUC = 0.6, 0.7, 0.8  $\Rightarrow$ ? More is better

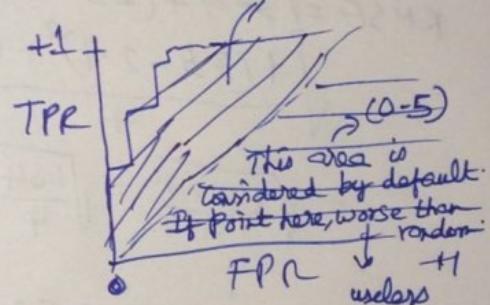
$\equiv P(\text{positive instance to have a higher score than}$

R library: ROC.R, PROC. (PROC)

In train() method, use metric = "ROC", point must be above diagonal to be useful.

roc(.)  $\rightarrow$  gives AUC value

so, AUC value =  $\frac{\text{area below line}}{\text{diagonal}}$   
always) +  
area above line  
 $= 0.5 + \frac{\text{area}}{\in [0, 0.5]}$



(\*) So, finally AUC  $\in [0, 0.5]$

Other confusion matrix-based metrics

- Recall - of those known as true, how many did you predict as true?
- Precision - of those predicted true, how many are actually true?
- Useful when ratio of -ve (and hence FPR) is not well defined
- F-score is harmonic mean.

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{R+P}$$

[it takes for average of percentages]  
 $\hookrightarrow$  mathinsight.org

Regression metrics

- RMSE = square root of average squared residuals
- $R^2$

$R^2$  - Capturing variance

If model captures total variance of target variable then it is considered as best model.

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^n e_i^2$$

Total sum of squares (SST)      Regression SS (SSR)      Error SS (SSE)

$$SST = SSR + SSE.$$

$SST = SSR \Rightarrow$  perfect fit.

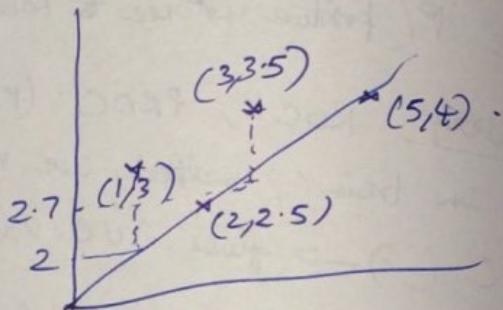
$$R^2 = \frac{SSR}{SST} \quad 0 < R^2 < 1.$$

If  $R^2 = 1$ , best fit.

$$= 1 - \frac{SSE}{SST}$$

$$RMSE = \sqrt{\frac{(3-2)^2 + (2.5-2.5)^2 + (3.5-2.7)^2 + (4-4)^2}{4}}$$

$$= \sqrt{\frac{1^2 + 0.8^2}{4}} = \sqrt{\frac{1.64}{4}} = \sqrt{0.41}$$



$$Var(y) = Var(3, 2.5, 3.5, 4) \quad \bar{y}_{\text{mean}} = \frac{13}{4} = 3.25$$

$$SST = (3-3.25)^2 + (2.5-3.25)^2 + (3.5-3.25)^2 + (4-3.25)^2$$

$$Var(\hat{y}) = Var(2, 2.5, 2.7, 4) \quad \text{w.r.t some } \bar{y} = 3.25$$

$$SSR = (2-3.25)^2 + (2.5-3.25)^2 + (2.7-3.25)^2 + (4-3.25)^2$$

$$R^2 = \frac{SSR}{SST}$$

$R^2$  has more intuitive meaning than RMSE.

$R^2$  may be more than 1, That's why we have "adjusted  $R^2$ "

This is the right metric

## 24/1/2017 Support Vector Machine (SVM)

very popular during 1994–2006 until NN took over

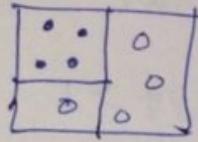
Does both

- linear classification
- non-linear (with Kernel)

Decision tree gives non-linear boundary

Logistic regression — No concept of best

line: Just finds a good separator.



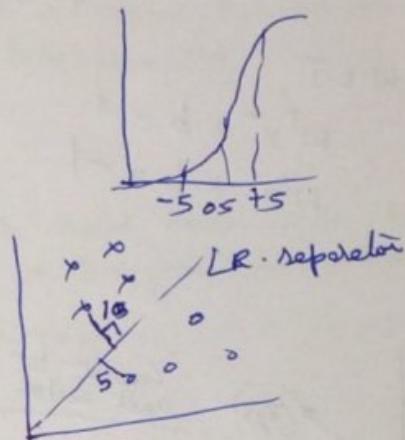
### Goal of SVM

The linear discriminant fn (classifier) with the max. margin is the best.

Why not LR?

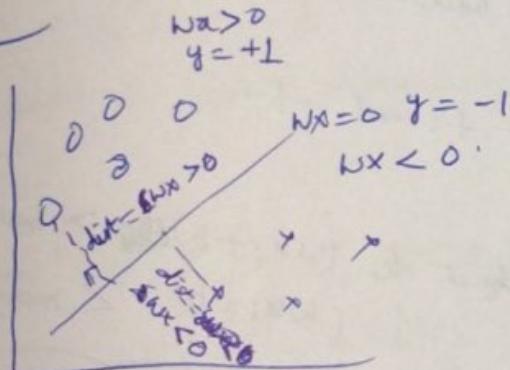
LR can't discover maximum margin.

If the distance is '5' and '10' on either side, LR can't learn further and hence stops. LR just ensures 'sufficient' gap (based on saturation inputs of +5 or -5) on either side of the line.



$$\text{distance}_i = \frac{w^T x_i}{\|w\|}$$

### SVM



$\hat{y}_i * \text{distance}_i = +ve$  for correct classification  
 $\hat{y}_i * \text{distance} = -ve \Rightarrow$  wrong classification

Goal: Maximize margin subject to  $(w^T x_i) y_i \geq 1$

$y = \text{margin}$

High/wide margin advantage:

- robust to outliers
- strong generalisation ability

Margin = width that boundary can be increased before hitting a data point



$(w^T x_i) y_i > 1$ ,  $(w^T x_i) y_i < 1$   
 $\equiv w^T x_i y_i > 1$  with proper scaling factors.

Maximize  $\frac{2}{\|w\|}$  ( $= \text{margin width}$ )  $\Rightarrow$  minimize  $\frac{1}{2} \|w\|^2$

subject to  $y_i(w^T x_i + b) \geq 1$

s.t.  $y_i(w^T x_i + b) \geq 1$   
 Take  $x^2$  (squared) for smoother learning

Marginal width

$$M = (x^+ - x^-) \cdot n$$

$$= (x^+ - x^-) \cdot \frac{w}{\|w\|} = \frac{2}{\|w\|}$$

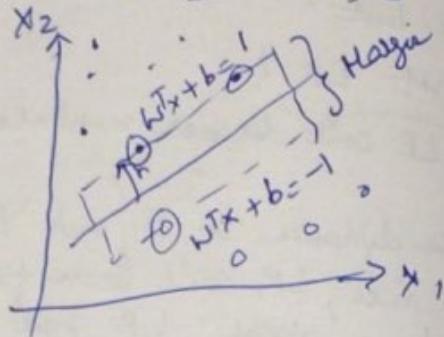
w.k.t.

$$\begin{aligned} w^T x^+ + b &= 1 \\ w^T x^- + b &= -1 \end{aligned}$$

$n = \text{unit vector}$ .

$w$  vector is  $\perp$  to ...

$$2 = [+1 - (-1)]$$



$\begin{pmatrix} 2 \\ 3 \end{pmatrix} \xrightarrow{\text{Convert to unit vector}} \left( \frac{2}{\sqrt{2^2+3^2}}, \frac{3}{\sqrt{2^2+3^2}} \right) = \left( \frac{2}{\sqrt{13}}, \frac{3}{\sqrt{13}} \right)$

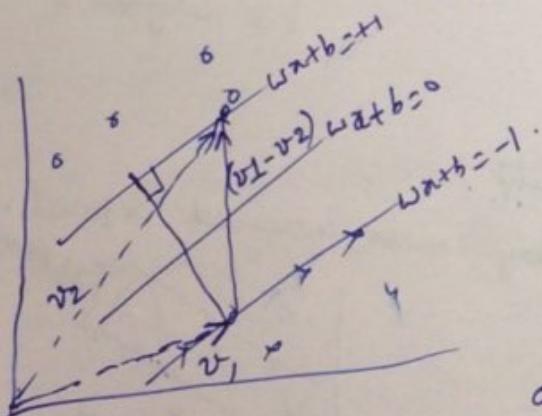
Dot product defn says 'unit vector'.

Preprocessing for SVM  
 Transform  $y_i$  values to +1 and -1.

28/11/2017

Maximize margin subject to  $(w^T x_i + b) y_i > \text{margin}, y=+1$   
 $(w^T x_i + b) y_i < \text{margin}, y=-1$

Combine as  $(w^T x_i + b) y_i > \text{margin}$



This is 'hard margin classifier' because no points are allowed within the margin.

(Hard margin SVM)

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

## Soft-margin SVM (linear classifier)

Some misclassifications allowed ; points can be in margin.

- but minimize the misclassification; the sum of slack variables.

Formulates :

$$\text{Minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$$

subject to

$$y_i (w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0.$$

-  $C$  controls overfitting.

## Soft-margin SVM with Kernel (non-linear boundary classifier)

Naive Soln : Transform input data into a higher dimension using the foll. nonlinear transformation:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

That higher dimension data could have linear separation but this is impractical  $\because$  # dimensions  $\propto$  transformation eq<sup>unknow</sup>  
can we avoid having to move to higher dimension and coming back to lower dimension?

- Use 'Kernel' trick

$$g(x) = w^T \phi(x) + b$$

Dot product in higher dimension = another operation in original dimension, i.e., square of dot product in original dimension

$$\begin{aligned} (\phi(x), \phi(w)) &= ((x_1^2, x_2^2, \sqrt{2}x_1x_2), (w_1^2, w_2^2, \sqrt{2}w_1w_2)) \\ &= x_1^2 w_1^2 + x_2^2 w_2^2 + 2x_1 x_2 w_1 w_2 \\ &= (x \cdot w)^2 \\ &= (x \cdot w)^2. \end{aligned}$$

$x = (x_1, x_2)$  } Two points in original dimension  
 $w = (w_1, w_2)$

Linear Kernel :  $K(x_i, x_j) = x_i^T x_j$  - Linear separation

Mult. Kernel { Polynomial Kernel :  $K(x_i, x_j) = (1 + x_i^T x_j)^P$  } Non-linear separation

$$\text{Laplacian (Radial-Basis Function (RBF)) Kernel : } K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

$$\text{Sigmoid : } K(x_i, x_j) = \tanh(\beta x_i^T x_j + \beta_i)$$

Complexity of SVM - Need to find out which Kernel is suitable for a problem.

Trees, KNN, SVM - all do non-linear separation

Polynomial Kernel : method = SVM Poly

## SVM output

Number of support vectors  $\equiv$  How many input values are on the two margins.