21/10/16

Job roles -

- Data mining expert
- Machine learning expert
- Predictive Analyst
- Data Scientist

} Hr.

- BIG DATA Analyst → + Aware of BigData ML platform (Spark / H_2O)

Hadoop being replaced by "Spark"

Mahout - obsolete

- AI programming
- SAS analyst
- Cognitive learning expert

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/algorithmica-repository/datascience

Assignments regularly on weekends

No fake experience. Genuine working knowledge.

Do "Proof of Concept" innovation.

R	Python
Promoted by Microsoft	Promoted by Open source community
Good libraries for basics	Good libraries for advanced work

22/10/16

R language
- R Studio (IDE)
- R runtime (execution)

0.99.903 - Latest
R 3.1.3 or latest

Python language
- Jupyter (IDE)
- Python runtime

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

Scanned by CamScanner

Kaggle.com / Competitions

"Knowledge Base" questions

Titanic : m/c learning from disaster

D/L train.csv, test.csv

R studio supports calculator work also.

(eg) >1+2
3

(eg) 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 predictors
- 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.

gendar_class_model.csv — submission format - Passenger | Survived

"Make submission" — 10 entries per day per person.

R Code

Save program in '.R' file.

"read.csv" function to read file

```
titanic_test =
> read.csv("test.csv")
```
→ give full path - use '/' forward slash for windows & linux 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");
```
↳ all rows, but 2 cols

↳ Wrong data format, extra row ID column.

```
write.csv(—same—, row.names = FALSE)   row.names = F.
```

Submission gives about 62.6% correct with 'majority' model.

Scanned by CamScanner

Now by "random" strategy.

titanic_test$Survived = sample(c(0,1), 418, replace=T)
                                  ↑ #rows
                                  ↳ to re-use guessed values.

We get 50.7% accuracy now.

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

—x—

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 (age)

max = 80 ⇒ realistic age

70% of this course is on "Predictive Analytics",
~~others are~~

Predictive Analytics
- Classification (es) - Will this customer respond to this offer? - Will something happen?
- Regression (es) - How much will the house cost? - How much something will happen?
- Recommendation - based on similarity principle. Identify people who are similar in the products they purchased or liked. Recommend similar products for like people

Laptop recommended — 8 GB RAM
2 core / 4 core
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 take better decisions / a better product
So, data has inherent value and cannot be discarded
- Amazon, Yahoo, Google, etc. are into data insights.

Data examples/sources
- Products purchased - customers order 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

Scanned by CamScanner

- Server logs...
  - Predict how much load a server next month
- IoT devices (Internet of things)
  - 65+ billion devices on Internet by 2010. Upto 230 billion by 2
  - Alert us before heart stroke, how to make devices smarter
  - Smart city – automatically find parking slot

(Less than 1% of demand for datascientists is met.)
Need diff. thinking style, not just tools

Structured – Transactional data
Semi structured – log data, XML/JSON data, Sensor data
Unstructured – Image, voice, email, Video, Chats, Blogs;

(eg) Facial keypoints 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

Deep Analytics

Predictive – Predicts "multiple futures" based on decision makers actions
(eg) Titanic survivors
Cost (for forecasting probabilistic predict in nature)

Descriptive (to summarize what happened (eg) segment pattern, outlier detection) clustering / feature reduction

Prescriptive (Less seen in this course)
- recommends one or more courses of action & shows the likely outcome of each decision
- predicts multiple futures based on

Python – Continuum Analytics company – "Anaconda" distribution

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)

decision maker's action
- 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.

Scanned by CamScanner

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

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

Recommended hardware – Quadcore, preferably i7 (at 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.

Menu

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

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 std languages like Java, but is here as data 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 ← numeric, character, logical
b) Containers / DataStructures → Array (3 or more dimensions), Vector (1D array) – all data of same type, Dataframe (to read excel sheets), Matrix – for rows/cols (2D array), List – most flexible, Factor – for enum types
c) Control statements
d) closures – specific to functional programming

Scanned by CamScanner

a = 10 # R treats this internally as a vector of size 1.
class(a) # type of variable — numeric

b = "abc"
class(b) # character

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

c = TRUE
class(c) # logical.

Data types

Continuous
- value not bound/in range
(eg) None, Age, SibSp, Parch

Categorical / Discrete
— only one from a set of values
(eg) Gender, Pclass, Survived

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

v = c(10.5, "abc", TRUE).

For heterogenous 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.

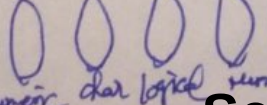
e1	e2	e3	e4	e5
10	20	30	40	50
1	2	3	4	5

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 heterogenous vectors



Scanned by CamScanner

```
ids = 1:4
names = c("S1", "S2", "S3", "S4")
Students = data.frame(ids, names)
class(Students) # data.frame
# find structure of frame
str(Students)
# find dimensions of frame
dim(Students)

Students[1,] # details of first row
Students["names"] # all clums
Students[ids>2,] # all rows satisfying the condition
```

shortcut → Students$names

To cross-classify values in dataframes, for tabular data, use "xtabs" → cross-tabulation

use "as.factor" to convert 'continuous' data to exam types.

```
titanic_train$Survived = as.factor(titanic_train$Survived)
                                    $Sex
```

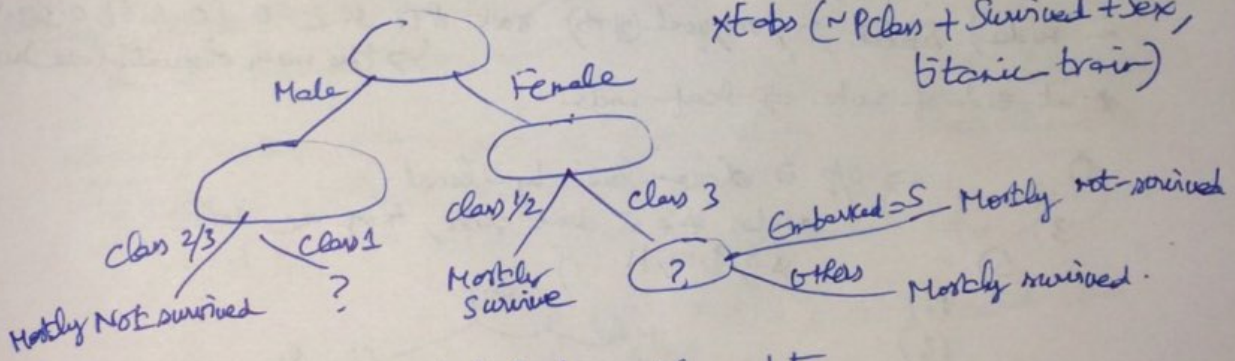
```
xtabs(~Sex + Survived, titanic_train)
```

...

```
titanic_test$Survived = ifelse(titanic_test$Sex=="female",1,0)
```

↳ forecast females as survivors.

```
xtabs(~Pclass + Survived + Sex,
      titanic_train)
```



We can use "Tableau" tool to explore data

Though humans can attempt to find logic, some issues in it

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

28/10/24

Titanic problem: check if "Fare" helps to improve logic

Need Machine Learning to handle large data

Data scientists

Scanned by CamScanner

Machine learning algorithm:

rpart → recursive partitioning.

library(rpart)

tree_model1 = rpart(Survived ~ Sex + Pclass + Embarked + Fare, titanic_train)

Need logic for this — Use these to get logic

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

CART (advantages)
- 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. (eg) $(x_2 - x_1) \leq 0$ 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.

21/10/16

Machine learning of classification trees

- Machine learning algorithms

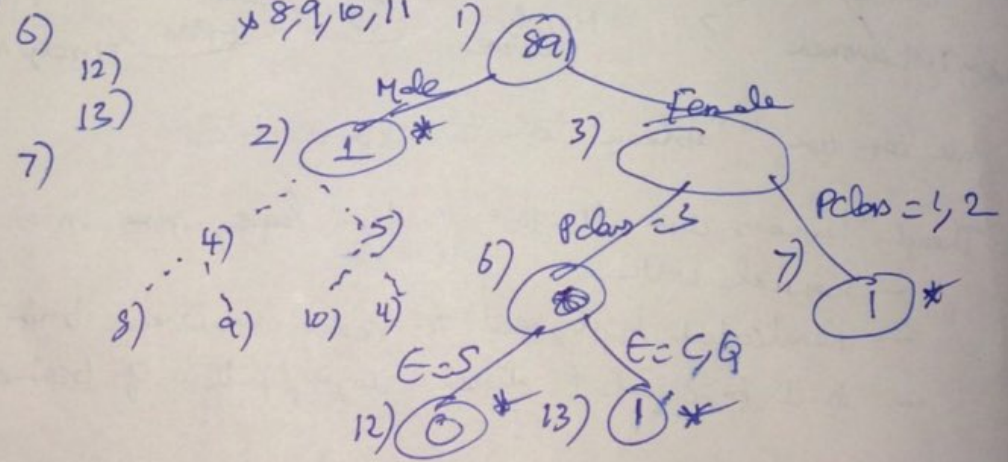
Output of rpart

- node) split, n, loss, yval (yprob)

node ← root 891 342 0 (0.6161 0.3839) ← predicted o/p.
loss of correct o/p for this many
↳ this many observations are in node

* at end of node ⇒ leaf node

1) 2) 3) 6) 12) 13) 7) → o/p is shown level-by-level.
Nodes 4 & 5 don't exist, they are called 8, 9, 10, 11



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

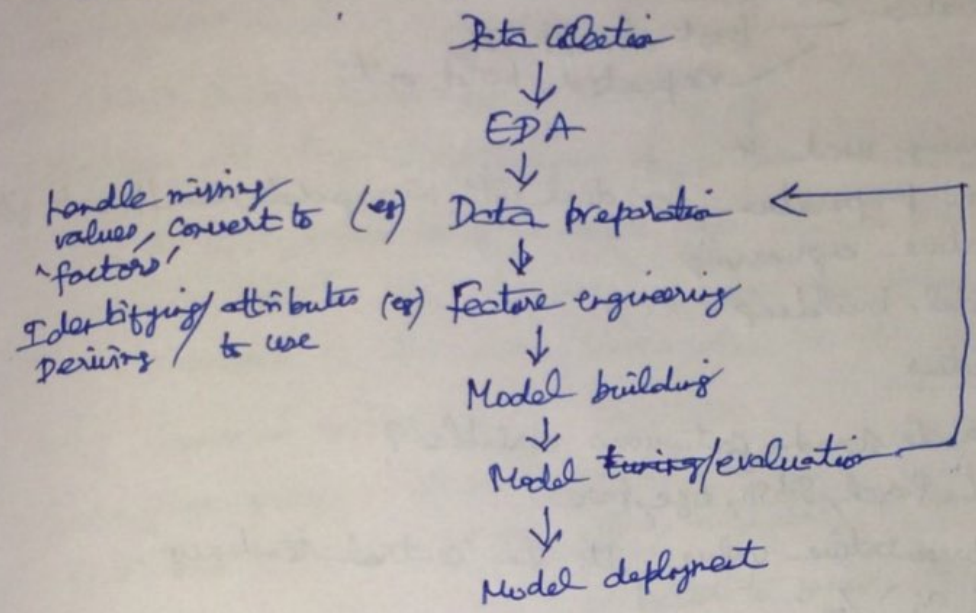
(eg) If female, but we don't know Pclass, then predict as "1" (survived)

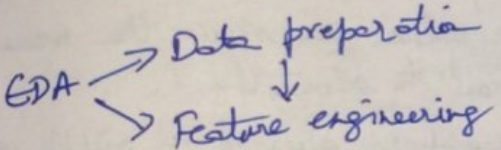
Scanned by CamScanner

Methods of machine-learnt approaches
- Trees (eg) 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%? (validation set)
(train set)

train() method from 'caret' package for model evaluation
↳ tells if attributes have missing values.

read.csv("train.csv", na.strings = c("NA", "")))
tells that "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	0.8004784	
0.0307	0.786	
0.444	0.718	

Scanned by CamScanner

Predictive data analytics
- Classification
- Regression
- Recommender systems
- Association analysis / Market basket analysis

Need to learn "statistics" to evaluate models
- mean, median, quartile, moments, SD, variance
- evaluation
  - cross validation
  - boot strapping
  - repeated hold out

Machine learning used in
- data preparation (eg) deal with missing data. Can / How to fill the
- feature engineering
- model building

1/11/16 Statistics

How to understand continuous variables?
- Parch, 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 are not enough to measure 'Spread'. We need other values like
- min
- max
} Range
- Standard deviation
- Inter quartile range (IQR)

Outliers impact mean but not median.

10, 15, 17, 19, 23, 40, 50, 60, 70, 80, 90, 100, 110

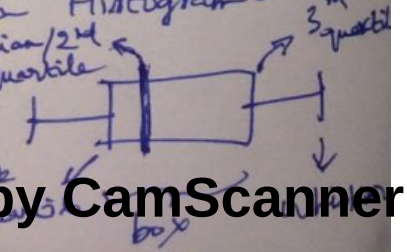
1st quartile: 25% < , < 75%
Median, 2nd quartile: 50% < , < 50%
3rd quartile: 75% < , < 100%

[1st quartile, 3rd 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'.



Scanned by CamScanner

'ggplot2' is grammar of graphics 'library'.

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

X11() # gives new window to draw plots

ggplot(titanic_train) + geom_histogram(aes(x=Fare)) + Coord_flip()

Warning: stat_bin() using 'bins = 30' — dividing data range into 30 bins and giving count of data in each bin) → 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 quartiles. 1st & 3rd 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 ⟨ MAD / √MSD

- find ~~median~~ mean
- take diff. of mean and each value
- Sum the 'absolute' value of the diffs, Take the 'median' of the absolute deviation ⇒ MAD (Median absolute deviation)
- Another is 'Standard' deviation ⇒ √MSD
  - Square root of 'Mean squared deviation'.

Standard deviation (SD) = Root mean squared deviation
= √Variance

Variance = Mean squared deviation

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

$\frac{d}{dx}(x)$ → not easy to find

$\frac{d}{dx}(x^2)$ → easier to find.

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

— We'll use this for 'kurtosis'.

Scanned by CamScanner

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

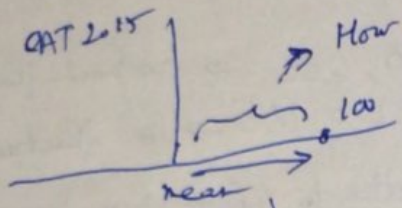
→ which of these toppers is better? BOTH are 100%ile percentile

Idea 1) mad(score, centre = max(score)) = Median(abs(x-
2) max(score) − mean(score − max(score))
this is all scores after the max score.

3) $\sqrt{\frac{\sum [x_i - \max(x)]^2}{n}}$ x = score set.

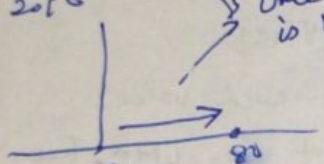
2/11/16

1) CAT 2015 — How many 'SD' units away from mean?



This tells how better the toppers are when compared with the rest.

CAT 2016 → Unit of comparison is 'sd' units'.



→ How many 'SD' units away from mean?

Z-score $z_i = \frac{x_i - \bar{x}}{sd(x)}$

↳ To compare values across diff scales.

Use 'preProcess' from 'caret' package to apply 'Z' finding logic in all columns of a dataframe.

Cat_2015_Z = (Cat_2015 − mean(Cat_2015)) / sd(Cat_2015)

Histogram : continuous variable :: Bar chart : categorical variable.

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

1) Exploring categorical variables
- Use a bar chart to visualize categorical variables.

xtabs(~ survived, titanic_train) ≡ to

ggplot(titanic_train) + geom_bar(aes(x = survived))

Scanned by CamScanner

- Use bar chart to also explore multivariate relationships among categorical variables

(eg) 2-vars
xtabs(~Sex + Survived, titanic_train) (or) better still
xtabs(~Survived + Sex, titanic_train) ≡
ggplot(titanic_train) + geom_bar(aes(x=Sex, fill=Survived))

The survivors & non-survivors are differentiated by color in each bar.

(eg) 3-variables
xtabs(~Pclass + Survived + Sex, titanic_train) ≡
ggplot(titanic_train) + geom_bar(aes(x=Pclass, fill=Survived))
+ facet_grid(Sex ~ .)

(eg) 4-variables
xtabs(~Embarked + Survived + Pclass + Sex, titanic_train) ≡
ggplot(titanic_train) + geom_bar(aes(x=Embarked, fill=Survived))
+ facet_grid(Sex ~ Pclass)

2) Exploring relationships between categorical and continuous variables
- Use boxplots or histograms.

(eg) 2 variables
xtabs(~Fare + Survived, titanic_train)
≡ ggplot(titanic_train) + geom_box(aes(x=Survived, y=Fare))
(or) ggplot(titanic_train) + geom_histogram(aes(x=Fare)) +
facet_grid(Survived ~ .) [or (. ~ Survived)]

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

(eg) 3 variables
xtabs(~Fare + Survived + Sex, titanic_train)
≡ ggplot(titanic_train) + geom_histogram(aes(x=Fare)) +
facet_grid(Survived ~ Sex)

(eg) 4+ variables - plot gets messy
xtabs(~Fare + Survived + Sex + Embarked, titanic_train)
≡ ggplot(titanic_train) + geom_histogram(aes(x=Fare)) +
facet_grid(Survived ~ Sex + Embarked) [or (Survived + Sex ~ Embarked)]

3) Exploring continuous vs continuous variables
- Use scatterplots
ggplot(titanic_train) + geom_point(aes(x=Fare, y=Age))

Scanned by CamScanner

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?

Model evaluation → resampling techniques: Repeated holdout, Cross validation, Bootstrapping } all 3 are resampling techniques.

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.

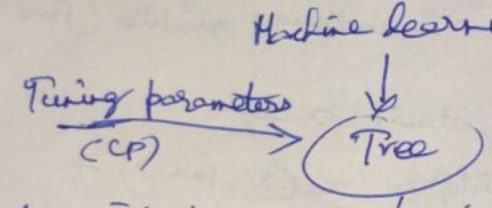
Re-sampling – Sampling 10 units many times.
└ 10, 10, 10, 10, 10

Re-sample or 'train' data

Randomly choose 80% & 20% train-test split multiple times and see how model performed on test or 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
1st iteration: 891 → 802 (build tree), 89 (check how tree predicts on this 89)
10th iteration
Many such iterations, find average prediction on '89' test samples.

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.

Scanned by CamScanner

If there are multiple tuning parameters, cp × ③trees ⑤ params.
use all 3×3 = 9 combinations.

The 'train' function does all this work.

How to split train-test %?
No math. But heuristics suggest 70-30% or 75-25%.

cp=0 ⇒ 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 ⇒ gives tree with only one node - All people 'dead' or 'alive'.

Different tuning parameters give the best accuracy for different data sets.

- In train() function, resampling is done for each tuning parameters combination. For a given data set, then the "tuning parameters" that work best on that dataset are chosen. 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 & H_2O.

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 kfold cv', best #iterations for each tuning parameter = 10. Based on heuristic
- For 'bootstrapping', best # iterations = 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 if logic discoverable with less "train" data.

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

Scanned by CamScanner

In repeated holdout & cross-validation, "stratification" is preferred - Ensures % of survivors & non-survivors (target attr) is equal across train and test data. 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 & test.

This issue is solved using K-fold cross validation.

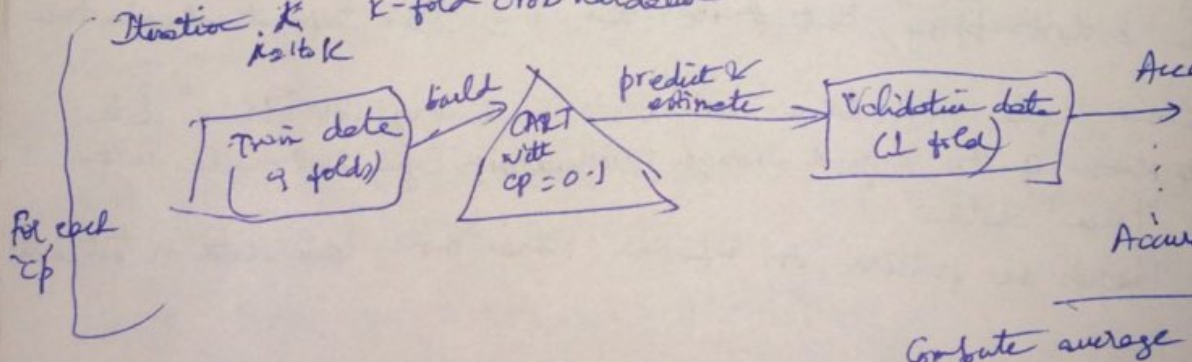
- Break 891 into 10 parts |89|89|89|---|89|
- 1st iteration: take 1st part as test
  2nd iteration: take 2nd part as test
  ⋮
  10th iteration: take 10th part as test
  } Covers all samples
- Number of folds = Number of iterations ⇒ This is done
- As per heuristics, # iterations = 10 works well in practice. (K=10 or K=5 works well) (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 ⇒ test data size = 1. Therefore, this is called "leave-one-out" when K=n.
- Make sure each of the 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).

(fx) Logistic regression estimates the probability of particular outcomes, the dependent variable is categorical.

7/11/16 Parameter tuning : CART trees



Compute average accuracy cp=0.1

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

Scanned by CamScanner

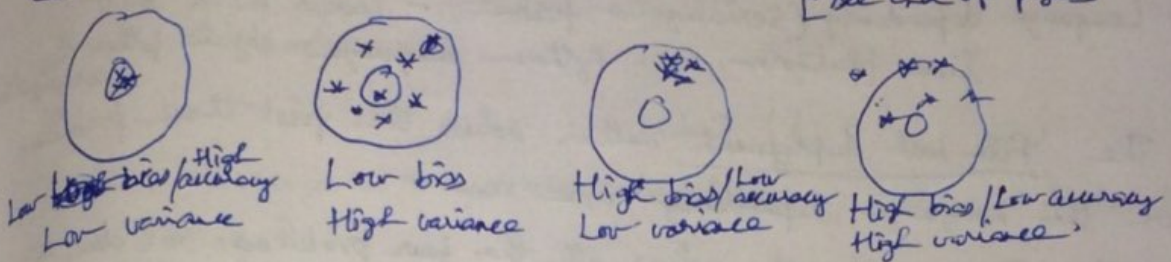
Human learning evaluation : Bias & Variance

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

Suppose a student, solves P2 - 50%, P3 - 80%, P4 = 90%, P5 - 30%
⇒ High variance + low accuracy/bias

Student$_2$ P2 - 60%, P3 - 65%, P4 - 63%, P5 - 58%
⇒ low variance + low accuracy/bias

Bias = How accurately can one solve variations of the problem?
Variance = How consistently can one solve the variations of problem? [see end of page]



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

Approach 2 (Resampling) - Give student a variation of the problem P1; like P2, P3, P4, P5, ...

Bias ⇒ How powerful/accurate is the model?
Variance ⇒ How sensitive the model is to the training set?

	cp	Accuracy	Kappa
A)	0.0	0.8215595	0.6187985
2)		↓ Bias	

b. See "tree_model$resample" to understand the variance. For each of the 10 iterations (of the best tuning parameter), we see 10 diff accuracy values ⇒ 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, chose one with lower variance.

train(tree_model$finalModel, titanic_test, type="class")

(*) Bias = How much on average are the predicted values different from the actual value?
Variance = How different will the predictions be for diff. samples?

Underfit ⇒ same result for all inputs ⇒ low variance & high bias
Overfit ⇒ high variance & low bias

Scanned by CamScanner

Model deployment

- using .RData
- using 'file hash 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 R serialization

The 'File hash 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 hash 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.rpart   # neural network writing in PMML format
```

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 hash 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.

Scanned by CamScanner

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 occurences
- Bayes — Used by machines
  Toss coin once, I'll tell probability
  Toss coin the 2nd time, I'll tell its probability.

9/11/16

Random ⇒ 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 experiment, posterior probability — Some formula correcting prior probability & evidence

(2, 3 — n times)

Machines learn using 'Bayes' approach.

Classical approach

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

Probability of Spade & face card selected from a deck of cards $= \frac{13 + 12 - 3(\text{common})}{52} = \frac{22}{52}$

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

Scanned by CamScanner

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 $= \frac{74}{48+74}$

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

4) Pick random <25 years; prob of female

$P(F|<25) = \frac{48}{98} = \frac{48/238}{98/238} = \frac{P(F \text{ and } <25)}{P(<25)}$

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

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

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

$\Rightarrow P(A \text{ and } B) = P(A|B)P(B)$ } These are equivalent
$P(B \text{ and } A) = P(B|A)P(A)$

D - Democratic, S = Senators

$P(D \text{ and } S) = P(D|S) \cdot P(S)$
or
$P(S|D) \cdot P(D)$ ← This formula not applicable here as we don't have $P(D)$ or $P(S|D)$.

18.7% Senators → 49% Democrats (out of 18.7% senators)

$\Rightarrow P(S) = 0.187$
$P(D|S) = 0.49$

$= 0.187 * 0.49$

Chain Rule

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

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

Scanned by CamScanner

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

$P(A)$ = prior probability
$P(B)$ = new evidence / normalizing constant
$P(B|A)$ = likelihood.
$P(A|B)$ = posterior probability

This is Bayesian way of saying posterior ∝ prior probability.
Updates 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?

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

$$= \frac{0.9}{P(T=+ve)} * 0.02 = \frac{0.9 * 0.02}{(0.02*0.9)+(0.98*0.1)} = 0.155$$

10/11/16

$P(T=+ve) = P(C=yes) * P(T=+ve \mid C=yes)$ [0.02, 0.9] = $P(T=+ve$ and with cancer)
$= P(C=yes \,\&\, T=+ve)$
$+ P(C=no \,\&\, T=+ve)$

$+\ P(C=no) * P(T=+ve \mid C=no)$ [0.98, 0.1] $P(T+ve$ and without cancer)

Writing functions in R.

```
functionName = function(....)
{
....
}
```

```
add = function(a,b)
{ return (a+b)
}
```

% population with cancer	Test reliability 90%	98%	100%	
2%	→ 15%	→ 56%	→ 100%	} Probability that patient has cancer.
60%	→ 93%	→ 98.6%		

Is Coin fair or not?

both sides are heads

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

Scanned by CamScanner

$$P(C=\text{Fair}|H) = \frac{P(H|C=F) * P(C=\text{Fair})}{P(H \& \text{fair}) + P(H \& \text{double headed})} \quad (= P_H$$

$$= \frac{0.5 * 0.9}{P(H) * P(C=\text{Fair}|H) + P(DH) * P(C=DH}$$

Denom = (or) $P(\text{Fair}) * P(H|\text{Fair}) + P(DH) * P(H|$

$0.(0.9 * 0.5) + (0.1 * 1)$

Probability rules

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

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

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

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

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

10, 10, 20, 20, 30 → This view useful for random experiments

Probabilistic view

Average $= \frac{10+10+20+20+30}{5}$ $\quad \frac{2}{5} * 10 + \frac{2}{5} * 20 + \frac{1}{5} * 30$

$= 18$ $\quad = 18.$

Lottery

lottery won by picking 6 numbers from 1 to 49.

To play lottery, need \$1.00

If win, prize money = \$2 million.

Expected value, $E(X) = \mu$ = 'weighted average' of possible values of X

$\mu = p_1 x_1 + p_2 x_2 + \cdots + p_n x_n$. ($X$ can take values $x_1, x_2, \ldots, x_n$)

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

Probability (lottery win) $= \frac{1}{C(49,6)} = 0.000\ldots01$

$E(X) = (2 \text{ million} * 0.000\ldots1) + (-1 * 0.9999\ldots9)$

lose \$1.00 with high probability

$= -0.86$ USD

Playing once ⇒ expect to lose \$0.86

-ve expected value never good!

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

Scanned by CamScanner

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?

⇒ 0.1 = double-headed probability

After 1st toss: We got 'H'

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

W.K.T. $P(\text{Head} \mid \text{Coin} = \text{Fair}) = 0.5$. If fair coin, $P(\text{head}) = 0.5$. This is likelihood.

$P(\text{Coin} = \text{fair}) = 0.9$. This is 'prior probability'.

$P(\text{Heads}) = P(\text{Heads}) * P(\text{Coin} = \text{fair} \mid \text{Heads}) + P(\text{Heads}) * P(\quad)$ // This is 'normalization constant'

$= P(\text{Heads and fair coin}) + P(\text{Heads and double-headed})$

$P(\text{Heads and fair coin}) = P(\text{Heads}) \times P(\text{fair coin} \mid \text{Heads})$ → We don't know these values. Avoid.

(or)

$P(\text{fair coin}) \times P(\text{Heads} \mid \text{fair coin})$ → We know these values. Use this.

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

$P(\text{Heads and fair coin}) = P(\text{fair coin})\,P(\text{Heads} \mid \text{fair coin})$

$= 0.9 \times 0.5$.

$P(\text{Heads and double-headed}) = P(\text{Heads}) \times P(\text{double headed} \mid \text{Heads})$ — Avoid. We don't know these

(or)

$P(\text{double headed}) \times P(\text{Heads} \mid \text{double headed})$ — Use.

$P(\text{Heads and double headed}) = P(\text{double headed}) \times P(\text{Heads} \mid \text{double headed})$

$= 0.1 \times 1$

↳ If coin is double-headed, we always get 'Heads'.

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

$= 0.8$

R code

```
posterior = function (prior, likelihood, normalize_factor)
{ return (prior * (likelihood/normalize_factor))
}
```

After 1st toss, $P(\text{Coin} = \text{fair} \mid \text{Heads}) = \text{posterior}(0.9, 0.5, (0.9*0.5 + 0.1*1))$

$= 0.8$.

$\therefore$ After 2nd toss, we have prior = $P(\text{fair coin}) = 0.8$.

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

We got 'Heads' in 2nd toss. $= 0.67$

Scanned by CamScanner

After 2nd toss (where we got 'Heads' again), prior = P(fair coin) = 0.67

∴ P(Coin = Fair | Heads) = posterior (0.67, 0.5, (0.67 * 0.5 + 0.33 * 1))
= 0.5.

After 4th toss (where we got 'Tails' now), prior = P(fair coin) = 0.5

∴ P(Coin = Fair | Tails) = posterior (0.5, 0.5, (0.5 * 0.5 + 0.5 * 0)) = 1

P(fair coin) P(Tails | fair coin) P(Tails | double headed coin)

Observation

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

0.9 → 0.8 → 0.67 → 0.5 → 1
(Initial) (Head) (Head) (Head) Tails)

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

↔ P(Coin = Fair | Heads) = posterior (1, 0.5, (1 * 0.5 + 0 * 1)) = 1

Also, P(Coin = Fair | Tails) = posterior (1, 0.5, (1 * 0.5 + 0 * 0)) = 1

These show the practicality / relevance of Bayes theorem of probab

11/11/16

Random variables

Discrete (eg) Coin toss, Patient has cancer?

Continuous

Addition rule: P(A or B) = P(A) + P(B) − P(A and B)

Multiplication rule P(A and B) = P(B|A) P(A)

How machines build classification tree?

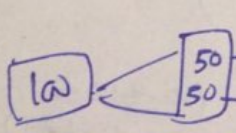
- Use 'impurity / pure' measure of nodes built by splitting data based on some attribute condition.
- Entropy
- Gini Index
- Misclassification error

to measure purity, we use these different methods

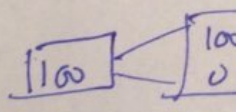
CART algorithm uses 'Gini index'.

k = no. of distinct values of X

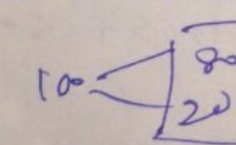
Entropy of X, $E(X) = \sum_{i=1}^{k} -p_i \log_2 p_i$ ["Shannon's" formula]



$E(X) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = -\log\left(\frac{1}{2}\right) = \log 2 = 1$



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



$E(X) = -\frac{4}{5}\log\left(\frac{4}{5}\right) - \frac{1}{5}\log\left(\frac{1}{5}\right) = 0.72$

$= 0.72$ [illegible] $= 0.47$

$E(X) =$ [illegible]

Scanned by CamScanner

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

100 — 99
100 — 1

So, We see that entropy exactly measures purity.

classification tree — we that 'variable' child splits data (node) to give 'minimum' entropy / maximum purity.

Why 'Decision trees'?

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

Divide by an attribute → easy if attribute has 'discrete' values.

compute 'weighted entropy'.

Gain (attribute) = Entropy (parent node) − weighted entropy of child nodes.

Choose attribute with 'maximum gain'.

Weighted entropy of child nodes $= \sum_{i=1}^{n} E(\text{child node}_i) * \frac{(\text{size of child node}_i)}{(\text{size of parent node})}$ where n = # of child nodes

Keep dividing those child nodes which are not yet pure.

How do divide with 'continuous' variables?

(eg) Titanic: Fare ≤ 5 & > 5
or
Fare ≤ 8 & > 8
etc.

} We try various divisions and choose the one with best gain.

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]

⇒ 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.

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

Two steps for growing decision tree

1) Growing phase ⇒ maximize purity of leaves
2) (Post) Pruning phase — minimizing the true "error" rate.

rpart → Recursive partitioning

Scanned by CamScanner

Issues with decision tree learning

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

Diff 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 of C4.5

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

- 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 ⇒ Takes longer time

↳ CART grows deeper trees, but C4.5 grows wider trees.

Problem with multi-way split in C4.5 — node size smaller and has more fr
But smaller nodes ⇒ more purity

Since the small nodes are pure, they become 'leaf' or hold 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).

Scanned by CamScanner

Assignment 5

19/11/16

1) 12 trucks – 4 faulty brakes. If an inspector chooses two of the trucks for brakes, what is the prob. that neither one is faulty?

$$\frac{{}^{8}C_2}{{}^{12}C_2} \text{ (or) } 1 - \ldots = \frac{7\times 4}{11\times 6} = \frac{14}{33}$$

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

contingency table

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

$$P(H<25) = 248/500.$$

$$P(W>25K \mid H>25K) = \frac{P(W>25K \text{ and } H>25)}{P(H>25K)} = \frac{54/500}{252/500}$$

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

3) chain rule

$P(A/B) = P(A \& B) = P(A|B)P(B)$

$P(A,B,C) = P(A|B,C)P(B,C) = P(A|B,C)P(B|C)P(C)$.

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

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

$P(\text{Pass } 3^{rd} \mid \text{Pass } 1^{st} \& 2^{nd}) = 0.7$.

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

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

5) Rain forecast for Mario's marriage.

Rained only 5 days each year $(5/365) = 0.14$.

Weatherman correctly forecasts rain 90% of time

Wrongly forecasts rain 10% of time.

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

$$P(\text{Rain} \mid \text{forecast} = \text{rain}) = \frac{P(\text{forecast} = \text{rain} \mid \text{Rain}) \, P(\text{Rain})}{P(\text{forecast} = \text{rain})}.$$

Assignment 4 – Univariate data

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

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

Scanned by CamScanner

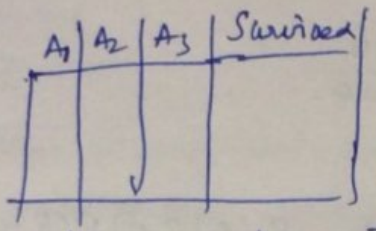
Confusion matrix - output of train method.
Gives the # of true-true, true-ve, false-true, false-ve values in 'validation' set.

		Predicted true	Predicted -ve
(True)	Given true	100 cell1	40 cell2
(False)	Given -ve	20 cell3	150 cell4

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

treemodel$ resampledCM → cell1 cell2 cell3 cell4
100 40 20 150

15/11/15



Recursive partitioning

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 (using attribute) = Entropy (original node) − Entropy (split 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').

Entropy Vs GiniIndex

$1 - \sum_i p_i^2 = Gini = \sum_i \left(\sum_{j \neq i} p_i p_j\right)$

10 Survivors
0 non-survivors

Entropy = $1 \log 1 - 0 \log 0$
= 0

GiniIndex
$p(1) \cdot p(0) = 1 \cdot 0$
= 0

8 − 2
Survived Died

$E = -\frac{8}{10}\log\left(\frac{8}{10}\right) - \frac{2}{10}\log\left(\frac{2}{10}\right)$
= 0.72

$\frac{8}{10} \times \frac{2}{10} = 0.$

5 − 5
Survived Died

$E = -\frac{5}{10}\log\left(\frac{5}{10}\right) - \frac{5}{10}\log\left(\frac{5}{10}\right)$
= 1

$\frac{5}{10} \times \frac{5}{10} = 0.$

3 − 7

$E = -\frac{3}{10}\log\left(\frac{3}{10}\right) - \frac{7}{10}\log\left(\frac{7}{10}\right)$
= 0.88

$\frac{3}{10} \times \frac{7}{10} = 0.$

(4) (0.1, 0.9) ... (0.3, 0.7)
For 2-valued nodes
⇒ Entropy ∈ [0, 1]

(Only for 2-valued nodes, 0.25]
GiniIndex ∈ [0, 0.5]

These have diff

Scanned by CamScanner

$Gini(t) = \sum_v p(v) \sum_{i \neq j} p(i|v)\,p(j|v) = \sum_v p(v) \sum_i \left(\sum_{j \neq i} p(i|v)\,p(j|v) \right)$

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

CART uses 'Gini gain' for splitting criteria.

C4.5 — categorical attributes — multiway cutting
continuous attributes — binary cutting.

CART : 'cp' :: C4.5 : 'c'

cost-complexity pruning (based on validation set)

$R_\alpha = MC + \alpha |T|$. For CART, α = '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, so don't do 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 usually
— Tree depth. (max depth) — controlled usually.

rpart.control — minsplit, maxdepth
default = 20, default = 30

(*) 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 → binary split; C4.5 → multiway split

(*) 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 = Gain / Entropy(attribute)
— CART solves the bias by always doing only a 2-way split.

(?) Is Gini Index = $1 - \sum_i p(i)^2$ ✓ (or) $\sum_{i \neq j} p(i)\,p(j)$? ✗ $\left\{ \sum p(i)\,p(j) \right\}$

Scanned by CamScanner

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

Model overfitting – model discovers irrelevant patterns (due to noise) 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.

Misclassification cost (subtree) = $\sum_{\text{leaves } j} \sum_{x_i \in \text{leaf}_j} 1(y_i \neq \text{leaf's class})$

y_i = actual value of x_i
leaf's class = does the leaf say 'survived' or 'not survived'.

→ This alone not sufficient because larger trees typically have low misclassification and a large number of leaves.
- So need a better tree with smaller # leaves while allowing misclassification cost to rise a bit.

⇒ cost (subtree) = $\sum_{\text{leaves } j} \sum_{x_i \in \text{leaf}_j} 1(y_i \neq \text{leaf's class}) + \alpha$ (# leaves in subtree)

control term – regularization

For CART, α is 'cp'.

If $\alpha = cp = 0$ ⇒ I am asking for deepest possible tree.
When cp ↑, # leaves ↓
When $\alpha \to \infty$, tree will have only the 'root node'.

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

Due to using probabilities, maximum misclassification cost = 1.
instead of 1 so, $cp \in [0,1]$ & $cp \in [0, \frac{(k-1)}{k}]$ k = # values target attr

In cost (subtree), why (# leaves in subtree) & not (# leaves in subtree)
Our measure of complexity is "# leaves", so this is enough. This does not make sense

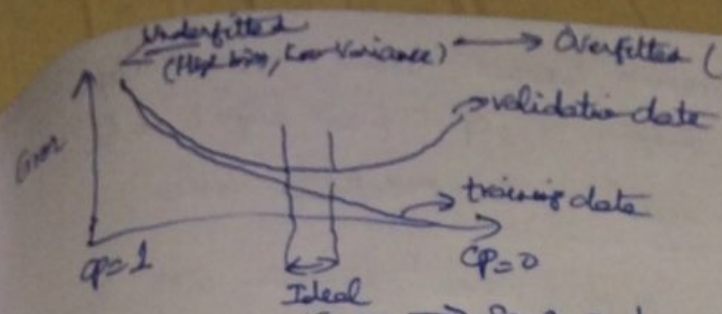
Even with slight ↑ in 'cp', tree becomes less complex and accuracy can go down. Need correct 'cp' to balance overfitting & underfitting

If there is a 'lot of noise' in train data, use higher 'cp' values 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.

Scanned by CamScanner



↓ because its very sensitive to data as it has learnt too much.

→ So, a good model should have 'medium bias' and not 'low bias'.

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.

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

If a tree has 200 nodes, and a leaf has 25 nodes — 15 correct prediction, 10 wrong prediction

So, error of leaf = $\frac{10}{200}$.

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

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

to be 1 only.

CART cost(subtree) = "formula" + (cp * (# leaves in subtree))
"formula" $\in$ [0, 1] value range

So, when cp = 1, even a tree with 2 leaves will have a cost of ≥ 2; but a tree with only 1 leaf (just root node) will have a cost ≤ 2. Post-pruning minimizes cost(tree).
So, when cp = 1, we will always have tree with just 1 node.

16/11/16

Randomness

cross-validation uses 'randomisation' during stratification.

$X_{i+1} = (aX_i + b) \% m$
↳ This sequence looks 'random'.

- a = 3
- X_0 = seed value
- b = 5

X_0 = seed · # set · seed ("X_0 value")

Pseudo-random generation

To make "cross-validation" look the same, use the same "randomness"

Scanned by CamScanner

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

Bayes learning → Bayes model — uses probability tables.

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

P(S=0 | Gender=Male, Pclass=1, Fare=120) = 0.3

P(S=1 | Gender=Male, Pclass=1, Fare=120) = 0.7 → We predict 'Survived'.

Instead of 'trees', use 'conditional probability' for predicting.

$$= \frac{P(\text{Male}, \text{class 1}, \text{Fare 120} \mid S=0) * P(S=0)}{P(\text{Gender=Male}, \text{Pclass=1}, \text{Fare=120})}$$

$$\frac{P(\text{Male}, \text{class 1}, \text{Fare 120} \mid S=1) * P(S=1)}{P(\text{Male}, \text{class 1}, \text{Fare=120})}$$

Interested to know which is bigger. So, compare only Numerator.

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

P(S=0) = ? } Know 'train' data the prior probabilities
P(S=1) = ? }

P(Male & class 1 & Fare 120) ≐ P(Male) P(class 1) P(Fare 120) — "if" all are independent

P(Male & class 1 & Fare 120 | S=0) = P(Male | S=0) P(class 1 | S=0) P(Fare 120 | S=0)

Illy for "S=1"

↓

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

	Outlook	Temperature	Humidity	Wind	→ predict Play Tennis ?
(eg)	Sunny	Hot	High	Light	No.
	Overcast	Hot	High	Light	Yes

Scanned by CamScanner

Naive Bayes model

Prior probabilities

Conditional probability tables — Machine learnt directly from data

Outlook

$P(\text{Play}=\text{Yes}) = 9/14$, $P(\text{Play}=\text{No}) = 5/14$

$P(\text{Sunny}|\text{p}) = 2/9$ $P(\text{Sunny}|\text{n}) = 3/5$

$P(\text{overcast}|\text{p}) = 4/9$ $P(\text{overcast}|\text{n}) = 0$

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

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

$P(\text{high}|\text{p}) = 3/9$ $P(\text{high}|\text{n}) = 4/5$

$P(\text{normal}|\text{p}) = 6/9$ $P(\text{normal}|\text{n}) = 2/5$.

(test) $P(\text{play?} | \langle \text{rain, hot, high, light} \rangle)$

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

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

Naive Bayes' assumes that conditions are independent. So,

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

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

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

— 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].

Denominator $= P(\text{rain, hot, high, light}) = P(\text{rain, hot, high, light, play=yes}) + P(\text{rain, hot, high, light, play=no})$ — same as 'likelihood' * 'estimate'

$\left[P(A|B) = \dfrac{P(B|A)\,P(A)}{P(B|A)\,P(A) + P(B|\bar{A})\,P(\bar{A})} \right]$

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)

$h_{MAP} \equiv \underset{h \in H}{\text{argmax}}\, P(h|D)$

$= \text{"} \dfrac{P(D|h)P(h)}{P(D)}$

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

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

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

Scanned by CamScanner

ML hypothesis. $P(h_i) = P(h_j) \; \forall i, j$.

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

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

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

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

Ages – 20, 23, 24, 27, 35, 32, 21, ... → 20–25, 25–30, 30–35, 35–40

$P(Age = 25.5 ?)$. Split 'range' in intervals

$$P(Age = 25.5) = \frac{\#\text{Passengers in } (25-30)}{\#\text{Passengers}}$$ → How to decide on interval?

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

Can't use 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'

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

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

σ_{ji} = Std. "

21/11/16 Naives Bayes conditional probability tables

$Fare	[,1]	[,2]
0	22.11789	31.38821
1	48.39541	66.597
	Mean μ	Std-deviation σ

$P(S = 0 \mid Pclass = 3, Fare = 12.5, Sex = male, Embarked = Q)$

Need to compute $P(Pclass = 3 | S = 0) \; P(Fare = 12.5 | S = 0) \; P(Sex = Male | S = 0) \; P(Embarked = Q | S = 0)$

→ Use the conditional probability tables

If probability = 0 in conditional table, final probability will be 0. So, ensure non-zero by adding 1. Increase weights of all outcomes → add '1' at numerator & add 'N' at denominator

→ Laplace

Scanned by CamScanner

In Naive Bayes, 'fL' parameter = Laplace correction. Other parameters are 'usekernel' & 'adjust'. fL ∈ [0,1] but we mostly use '1'. - recommended by Laplace

usekernel = FALSE ⇒ assume 'normal distribution' and use Gauss equation

usekernel = TRUE ⇒ split 'continuous' data into bins.

'adjust' ⇒ How much of 'bin size' to use? Makes sense only when usekernel = TRUE.

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

Advantages of 'tree approach' vs 'Naive Bayes/other learning'

- No formal distributional assumptions. In NB, we should if distrbn 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

- Newer 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 classifier' 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.

Scanned by CamScanner

To find probability of continuous variable, 2 ways

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

Distribution based — Need to know the distribution

Correct here

This is "wrong" correction

N=14 P(play) = 9/14, P(no play) = 5/14

Conditional probability tables

Outlook	play	no play
P(sunny | p) = 2/9	P(sunny | n) = 3/5	
P(overcast | p) = 4/9	P(overcast | n) = 0/5	
P(rain | p) = 3/9	P(rain | n) = 2/5	

Temperature		
P(hot | p) = 2/9	P(hot | n) = 2/5	
P(mild | p) = 4/9	P(mild | n) = 3/5	
P(cool | p) = 3/9	P(cool | n) = 1/5	

Humidity

....

(After Laplace correction)

Outlook	
P(sunny | p) = 3/23	P(sunny | n) = 4/19
P(overcast | p) = 5/23	P(overcast | n) = 1/19
P(rain | p) = 4/23	P(rain | n) = 3/19

Temperature	
P(hot | p) = 3/23	P(hot | n) = 3/19
P(mild | p) = 5/23	P(mild | n) = 3/19
P(cool | p) = 4/23	P(cool | n) = 2/19

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 the 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: $3/9 < 2/5$

↓ correction, Laplace

$4/23 > 3/19$

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

Scanned by CamScanner

After Laplace correction

$P(\text{play}) = 10/16$, $P(\text{no play}) = 6/16$. [Adding 2 — 1 play, 1 no play].

$= \frac{9+1}{14+2}$ $= \frac{5+1}{14+2}$. // Correction usually NOT done for this; this is 'apriori'.

Outlook # other opposite likelihoods also change

$P(\text{sunny}|p) = 3/12 = \frac{2+1}{9+3}$

$P(\text{overcast}|p) = 5/12$

$P(\text{rain}|p) = 4/12$

$P(\text{sunny}|n) = 4/8 = \frac{3+1}{5+3}$

$P(\text{overcast}|n) = 1/8$

$P(\text{rain}|n) = 3/8$. (*)

(Adding 1 sample for each of sunny, overcast, rain)

(*) Temperature, Humidity also get corrected

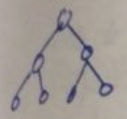
In general, to make Laplace correction, we add an initial count of 1 to the total of all instances with a given attribute value, and we add the number of distinct values of the same attribute to the total number of instances in the group.

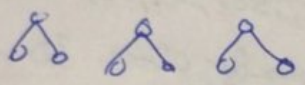
22/11/16

Probability distributions

- discrete distributions
- Continuous distributions.

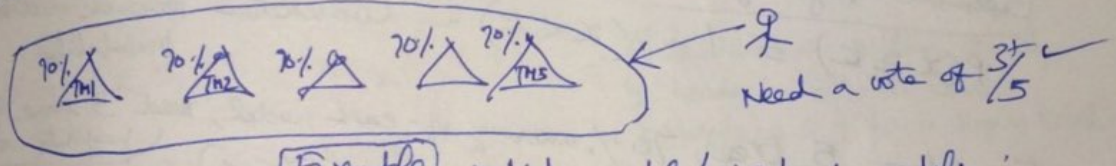
Single tree → Multiple trees (Forest). Is this better?





Why have forests instead of a single tree?

Suppose I've 5 classifiers which each 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



Ensemble → Meta model / group of models.

Will majority vote > 70% or not?

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

$= \frac{5C_3}{2^5}$ (or) $5C_3 (1/2)^5$ → Each of the $5C_3$ has $(1/2)^5$ 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 distributions is "Binomial distribution"

- 'n' experiments, independent of each other, only 2 possible outcomes
- p (of one outcome) ⇒ 1−p (of other outcome).

Scanned by CamScanner

experiments → # heads

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

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

↳ R → 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(Heads) = 0.8? [Cumulative probability]

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

$P(X \le k) = pbinom(2,5,0.8) = dbinom(0,5,0.8) + dbinom(1,5,0.8) + dbinom(2,5,0.8)$

Generate random number from binomial distribution/sequence

- rbinom(4,10,0.8).

Pascal Δ – binomial coefficients

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

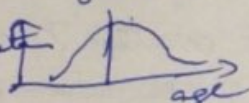
1 2 1 – $2C_0, 2C_1, 2C_2$

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' → randomly chose 30 values from (1 to 30). ↳ bad.



Generate 'ages' in normal distribution with mean '25' and some variations.

Ensemble – majority vote

$P(x \ge k) = 1 - P(x < k)$ – Cumulative binomial distribution probability

Ensemble – 5 trees, 70% accuracy of each model, need '3' same for majority vote

$n = 5, k = 3$.

$P(X \ge 3) = 1 - P(X \le 2) =$

$1 - pbinom(2,5,0.7)$.

How to make models independent in ensemble? — Bagging / 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(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(Heads of ensemble) = ?

⇒ P(Heads of ensemble) = P(3+ heads) = 1 − P(≤2 heads)

= P(Accuracy of ensemble) = $1 - pbinom(2,5,0.7)$

$= 0.837$

See how accuracy increased from 70% to 83.7%, with

Scanned by CamScanner

1-pbinom(500, 1001, 0.7) = 1
1-pbinom(1000, 2001, 0.7) = 1
} Since 100% accuracy achieved with 500 models, no need to build a large ensemble.

1-pbinom(50, 101, 0.7) = 0.9999871 ⇒ 101 models are sufficient!

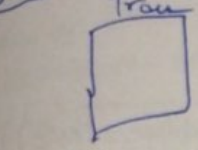
The real challenge is "How to build completely independent models?" — "Bagging" idea

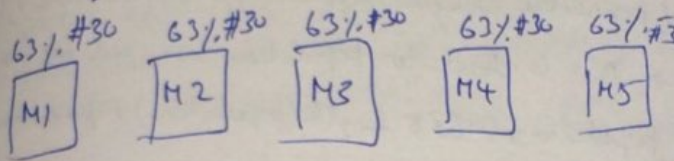
29/8/16 Discrete prb distribn patterns
- Binomial, Poisson, -ve binomial, geometric

Continuous prob distribn patterns
- Normal, Gamma, Exponential, Beta, Lognormal, chi-square, Fisher's F, Student's T

Cumulative probability, pbinom() used for p-value

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





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"

Bootstrap + aggregation = Bagging
Bagging + random feature selection = Random forest

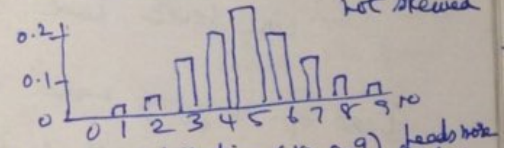
'aggregation' means 'majority vote'.
bootstrap = resampling with replacement

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

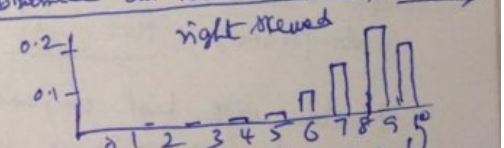
binomial-distribution = function(n, p)

```
{
  x = 0:n
  b = dbinom(x, n, p)
  barplot(b, names.arg = x)
}
```

binomial distribution (10, 0.5)



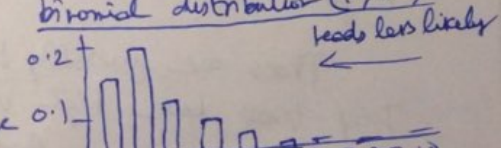
binomial distribution (10, 0.9) heads more likely →



rbinom(4, 10, 0.8) # 4 random #s in [0, 10] in binomial distbn 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

binomial distribution (10, 0.1) ← heads less likely



rbinom(4, 10, 0.5) # mostly in [3, 7] range
> 4 7 3 6, 7 5 3 4

rbinom(4, 10, 0.2) # mostly in [1, 3] range
> 4 3 4 3, 2 0 2 1

Obviously, rbinom() makes sense only with replace = TRUE. So, its not an option when p(heads) = 0.1, we (heads)

Scanned by CamScanner

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

We can also empirically observe that the # of unique entries in the bootstrapped sample (of size n) ≈ 0.632 n [= (1 - 1/e)]
So, 63% is unique. The remaining 37% is used for validation by bagging() method. ↳ "out-of-bag" data

```
n = 100000
bsample = sample(1:n, n, replace = TRUE)
length(unique(bsample))/n  # 0.63155.

population = 1:10000
mean(population)  # 5000.5
mean(sample(population, length(population), replace = TRUE))
        ↳ This is closer to 'population mean' and has less variance
mean(sample(population, 0.63 * length(population), replace = FALSE))
```

↳

24/11/16

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

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

Bias of bagging model ≪ 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 models

Models with high variance are good candidates to 'bag' into a forest, so variance is 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

Scanned by CamScanner

Why NB not bagged to improve accuracy?
- Hard to have 'independent' NB models. Can use '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$Survived, method="treebag", trControl=resampling_strategy)
# resampling_strategy = trainControl(method="boot", number=10)
bag_tree_model$finalModel$mtrees[[5]]$bindx # indexes of the 63% selects
                                  $ 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=resampling_strategy)
```

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

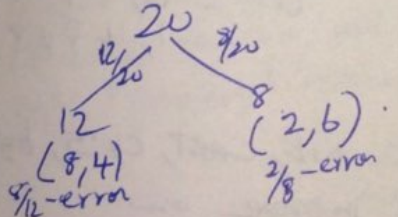
Post pruning in CART

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

always '0' if all leaves are pure. overfitting!

6-survived, 4-not survived
node class = survived ⇒ misclassification error of this node = 4/10.

(6,4)



$\left(\frac{4}{12} * \frac{12}{20}\right) + \left(\frac{2}{8} * \frac{8}{20}\right)$ → for these two leaves.

Modified above formula to overcome overfitters

$$\text{Cost(subtree)} = \left(\sum_{\text{leaves } j} \text{misclassification error at node } j * \text{proportion of data instances reached to node } j\right) + \alpha(\#\text{leaves in subtree})$$

Scanned by CamScanner

Only one node in tree (20) (12,8) → Cost = 8/20 (20/20) + 0(1)

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

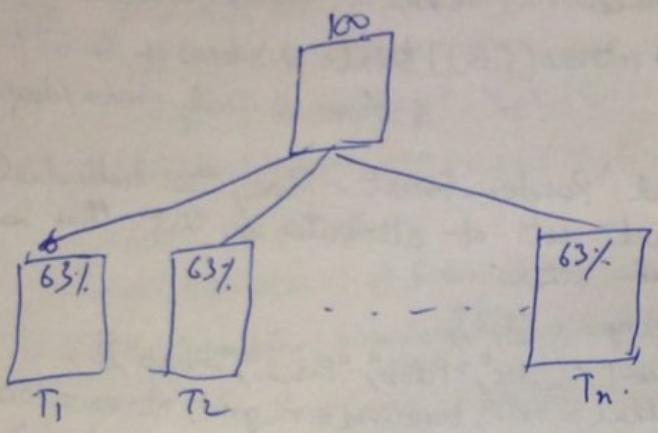
I) (4,4) → (1,3) (3,1)

II) (4,4) → (4,2) (0,2)

'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' da



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

P(sample is OOB) = 0.37.
P(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' h

OOB Error Estimate

i	M_1	M_2	M_3	M_4	M_5	$\hat{Y}$	Y	ERR
1	·	+	·	+	-	+	+	0
2	·	-	·	—	—	—	—	0
3	+	·	—	—	—	—	+	1

'·' = OOB
+ = prediction is +ve
- = 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} = Proportion (ERR)

$Accuracy_{OOB} = 1 - ERR_{OOB}$

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'.

Scanned by CamScanner

Data preparation (DP)
- type casting, adding levels
- Handle missing data → Adhoc → Median/Mean, ML Approach, Multiple Imputation
- Normalisation / Standardisation → 0-1, (Z-score)
- Transformation of data.

Feature Engineering (FE)
- Filter features → automated → regressive feature elimination, Neural Network
- Automatic feature selection → EDA/Manual elimination
- Feature creation → variance
- Feature transformation → Correlation

× extraction (Deep learning)
→ Neural networks (Deep learning approach)
→ RFE (Recursive feature elimination)
→ PCA

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

In Random Forest, "~~boot~~". if we say resampling strategy (method = "boot", number = 10)
how is the 4/10^4 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 × 100 = 1000 trees overall.

Why is 10 random forests × 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 minimise the chances 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 these 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 — use 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]

Scanned by CamScanner

K-Nearest Neighbors

No tree/probability based learning. Find those who are close 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 the 891 passengers in 'train'. Among the 'nearest' neighbors, find their 'majority vote' or survived/non-survived. Predict the majority 'survived'.

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

KNN applicable only to 'continuous' data because we cannot 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'.

↓ Majority vote of K neighbors ↓ Average of K-nearest neighbors

→ But all the predictor variables must be 'continuous' so that distances can be measured.

Advantages 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, till, deal successfully with changes to the problem domain. But CART/C4.5 need changes to recompute 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.

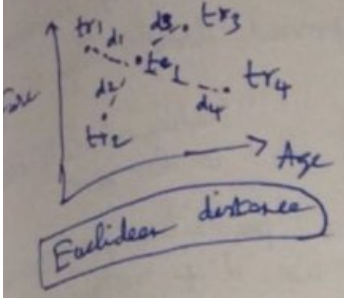
Scanned by CamScanner

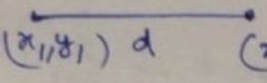
In KNN, how to measure closeness/similarity between 2 passengers?

For every passenger in 'test', compute distances from each 'train' passenger.

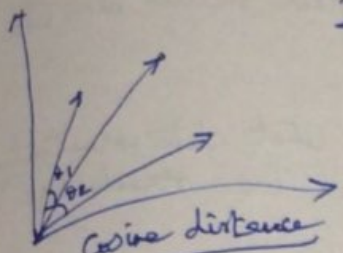
- Euclidean distance
- Cosine distance
- Correlation distance
- ⋮

$tr_i \Rightarrow i^{th}$ train passenger
$te_i \Rightarrow i^{th}$ test passenger.





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



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) —— (x_2, y_2, z_2) is

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

(*) In KNN, choose '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 (#2) is not always valid.

Weighted KNN → 1st closest neighbor has more weightage than 2nd 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.

(*) 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 ⇒ 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.

Scanned by CamScanner

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

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

$\hat{y}_i$ = actual y_i value

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 for restaurant revenue.

1/12/16 Mean absolute error (MAE) $= \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$; '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 also like 'Median absolute error'.

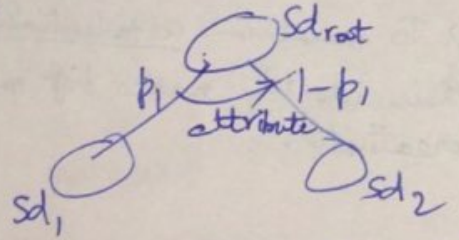
2/12/16 What model we build decides what data normalization to do. For eg, in KNN (distance-based) we standardize numbers into z-score.

5/12/16

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

How to use 'Tree learning' for regression?

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



$SD_1 p_1 + SD_2(1-p_1)$ = SD 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 prune regression trees. Instead, we just use 'threshold SD'. We can prune

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

Scanned by CamScanner

		SD	
outlook	overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

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

$S(\text{Hours}, \text{Outlook}) = P(\text{Sunny}) * S(\text{Sunny}) + P(\text{Rainy}) * S(\text{Rainy}) + P(\text{Overcast}) * S(\text{Overcast})$

$= \left(\frac{4}{14} \times 3.49\right) + \left(\frac{5}{14} \times 7.78\right) + \left(\frac{5}{14} \times 10.87\right)$

$= 7.66$

SD (overall data) = 9.32

SD reduction using outlook = SDR = 9.32 − 7.66 = 1.66.

output of 'rpart' or regression

node), split, n, deviance, yval

↳ SD ↳ average

In 'treebag' approach, we take 'average' of all trees for 'regression'.

How does a random forest compute 'variable importance'?

[Quoting from https://topepo.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 then 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 (not used to decide final model, but gives some diff information)

regression – RMSE, Mean absolute error, median absolute error, R^2.

Feature engineering
- Filtration ← EDA: variance, correlation
- Feature transformation → PCA
- Feature creation

What parts of data analytics lifecycle can be automated?

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

Scanned by CamScanner

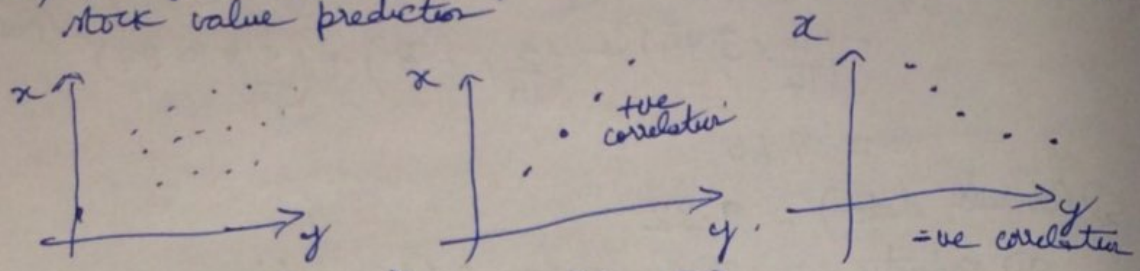
- Model building - tuning of models needs manual work

Correlation in Regression (the strength of the

Correlation 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.

(eg) 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 $x = y$, $\text{covariance}(x,y) = \text{Covariance}(x,x) =$

$$\frac{\sum(x_i-\bar{x})^2}{n} = \text{Variance}(x)$$

→ Apply $\bar{z}$-score standardization to get covariance or = correlation

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

$$= \frac{1}{n}\frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sigma_x \sigma_y}$$

Covariance values are dominated by 'scale'. So, normalise to understand relationship.

"Standardized covariance" is called "correlation".

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

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

(A.) varImp() in CART / rpart.

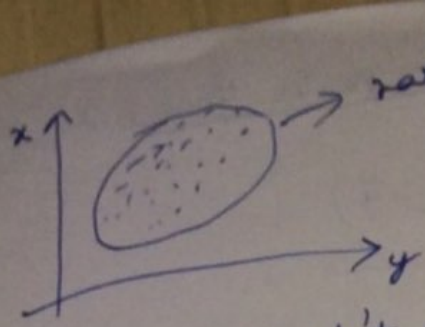
- This is computed in a way similar to that of random forest. The variable columns are permuted, and if accuracy does not change much on the permuted values, that variable has low importance.

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

(A.) 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.

Scanned by CamScanner



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, ie, all values are same, it is not useful for prediction

Correlation also used to filter features

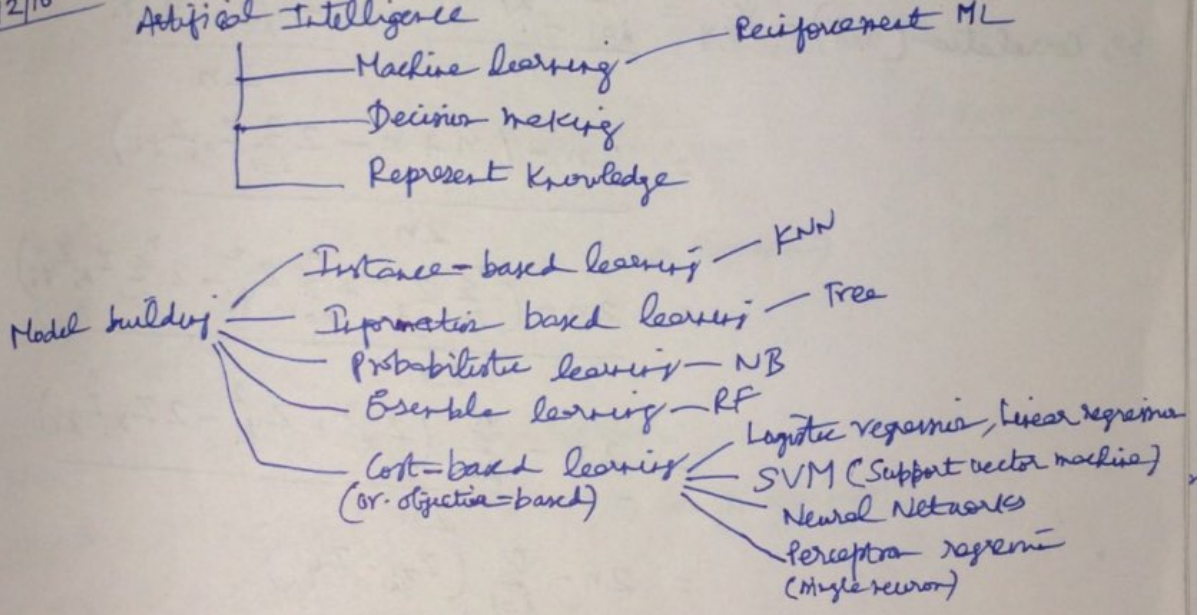
- remove redundant features

	1	2	3	4	5
1	1	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



Scanned by CamScanner

Why is correlation coefficient not greater than 1?

We know that

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

$$\text{correlation}(x,y) = \text{Covariance}(Z_x, Z_y) \quad ; \quad Z_{x_i} = \text{Z-score}(x_i) = \frac{x_i-\bar{x}}{\sigma_x}$$

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

$$= \frac{\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{\sigma_x}\right)\left(\frac{\bar{x}-\bar{x}}{\sigma_x}\right)}{n}$$

$$Z_{\bar{x}} = \frac{\bar{x}-\bar{x}}{\sigma_x} = 0$$

$$Z_{\bar{y}} = \frac{\bar{y}-\bar{y}}{\sigma_y} = 0.$$

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

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

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

So, $\text{Correlation}(x,y) = r_{x,y} = \frac{\sum_{i=1}^{n} Z_{x_i} Z_{y_i}}{n} = \frac{2\sum_{i=1}^{n} Z_{x_i} Z_{y_i}}{2n}$

$$= \frac{2n - \left(n + n - 2\sum_{i=1}^{n} Z_{x_i} Z_{y_i}\right)}{2n}$$

$$= \frac{2n - \left(\sum_{i=1}^{n} Z_{x_i}^2 + \sum_{i=1}^{n} Z_{y_i}^2 - 2\sum_{i=1}^{n} Z_{x_i} Z_{y_i}\right)}{2n}$$

$$= \frac{2n - \sum_{i=1}^{n}\left(Z_{x_i}^2 + Z_{y_i}^2 - 2 Z_{x_i} Z_{y_i}\right)}{2n}$$

$$= \frac{2n - \sum_{i=1}^{n}\left(Z_{x_i} - Z_{y_i}\right)^2}{2n}$$

$$= 1 - \frac{1}{2n}\left[\sum_{i=1}^{n}\left(Z_{x_i} - Z_{y_i}\right)^2\right]$$

The RHS in the term is non-negative.

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

so, $r_{x,y} \leq 1.$ — (1)

Scanned by CamScanner

Also, $r_{x,y} = \frac{\sum_{i=1}^{n} z_{x_i} z_{y_i}}{n} = \frac{2\sum_{i=1}^{n} z_{x_i} z_{y_i}}{2n}$

$$= \frac{-2n + \left(n + n + 2\sum_{i=1}^{n} z_{x_i} z_{y_i}\right)}{2n}$$

$$= \frac{-2n + \left(\sum_{i=1}^{n} z_{x_i}^2 + \sum_{i=1}^{n} z_{y_i}^2 + 2\sum_{i=1}^{n} z_{x_i} z_{y_i}\right)}{2n}$$

$$= \frac{-2n + \sum_{i=1}^{n} \left(z_{x_i}^2 + z_{y_i}^2 + 2 z_{x_i} z_{y_i}\right)}{2n}$$

$$= \frac{-2n + \sum_{i=1}^{n} \left(z_{x_i} + z_{y_i}\right)^2}{2n}$$

$$= -1 + \frac{1}{2n}\left[\sum_{i=1}^{n} \left(z_{x_i} + z_{y_i}\right)^2\right]$$

The RHS term is non-negative.

So, $r_{x,y}$ = −1 + (a non-negative value)

$\Rightarrow r_{x,y} \geq -1$ — (2).

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

12/12/16

To get data — Kaggle.com, TopCoder.com

Classification ✓ (only Linear Regression)

Regression ✓

Linear Regression
NN
SVM
Perceptron
→ Cost-based

Need 'optimisation theory' background.

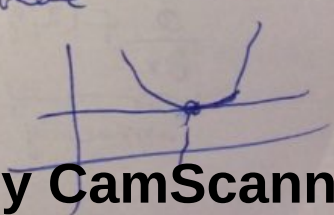
Objective / Cost-based learning / Loss-based learning

Objective function
- Minimise error / loss
- Constraints

(eg) $f(x) = x^2 + 2x + 3$.

Find value of x that minimises this value.

Find all possible values of 'x'
- Not practical with multiple variables



Scanned by CamScanner

$f(x,y) = x^2 + 2xy + xy^2 + 10$ → Need a 3-D plot for x, y & f.

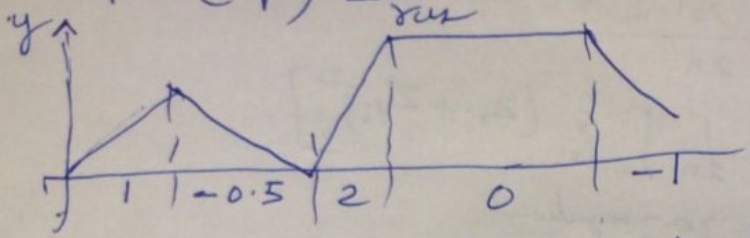
To minimise $f(x) = x^2 + 2x + 3$

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

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

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

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.

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

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

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

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

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'.

$\frac{\partial}{\partial x} f(x,y)$ or $\frac{\partial}{\partial y} f(x,y)$ – partial differential equations

$\frac{\partial}{\partial x}$: assume 'y' constant; $\frac{\partial}{\partial y}$: assume 'x' constant

Scanned by CamScanner

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 derivatives – to compute slope of a curve at any point in multiple dimensions.

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

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

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

Need "gradient" to compute slopes using 'x' & 'y'.

13/12/16

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 solns and pick the optimal one (Adhoc (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 $x = 2, 3$.

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

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

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

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

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

(y constant) $\frac{\partial f}{\partial x} = 6x^2 + 8xy + 8y \;|_{(1,2)} = 6.1^2 + 8.1.2 + 8.2 = 38$

(x constant) $\frac{\partial f}{\partial y} = 4x^2 + 8x \;|_{(1,2)} = 4.1^2 + 8.1 = 12$

Slope of line at (x,y) in 'y' direction

To find min

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

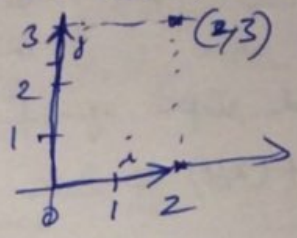
Generally hard to solve these equations. Non-linear and hard to solve. So, we use "numerical methods" approaches

Scanned by CamScanner

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 $(2*\hat{i}) + (3*\hat{j})$.



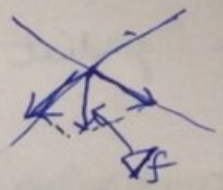
"∇" gradient symbol.

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

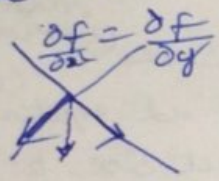
Gradient points us to the direction to move.

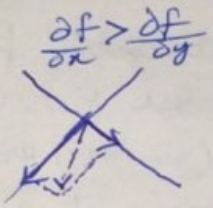
In 1-variable equation (represented in 2D), slope = gradient

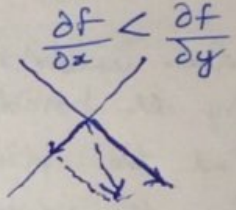
But in 2+ variable equation (3D+) gradient determined using combination of slopes.

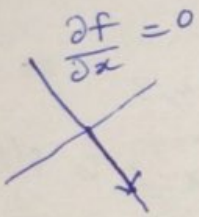


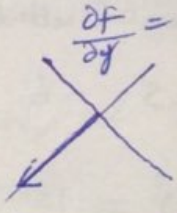
Jump in small steps along the 'gradient' to move towards the minima/maxima







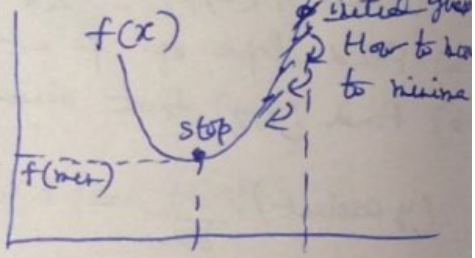




14/12/16 Solving optimization problems: Guess and refine

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.

Scanned by CamScanner

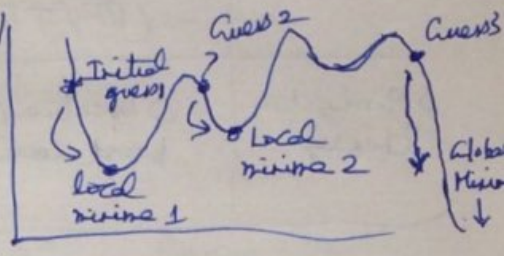
How to choose direction?
- gradient provides solution

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

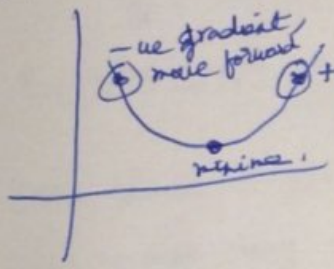
Plot function to see structure. Make multiple initial guesses [Applies to 'minima' problems only, not 'maxima']



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

↖ defines no multiplier, constant multiplier, adaptive multiplier. (stepFactor = 1)

(-) is used because



So need $x_{n+1} = x_n - (\text{something} * \text{gradient})$

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,

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

- 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)
  - $(x_n - x_{n-1}) <$ some threshold, ie, x_n has converged sufficiently

Scanned by CamScanner

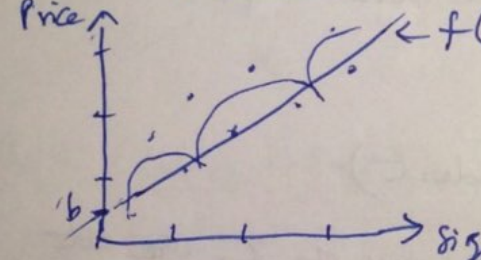
15/12/16

		Classification	Regression
Probability + Gloppy Gain	Information based (Tree) learning	CART / C4.5	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, BoostedTree, Custom	
Optimization theory	Objective/Cost based learning	Logistic regression, SVM, Perceptron, NN	Linear/Multivariate regression, SVM, NN.

Restaurant revenue - Cost-based approach. [Linear Regression]

Offers	Size	# of rooms	Price
.	1000	3	15,00,000
.	800	2	12,00,000
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Assume only Size & Price Price =

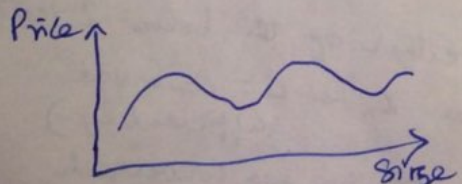


$f(\text{Size}) = m(\text{Size}) + b$

$= w_1(\text{Size}) + w_0$ ← base price

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

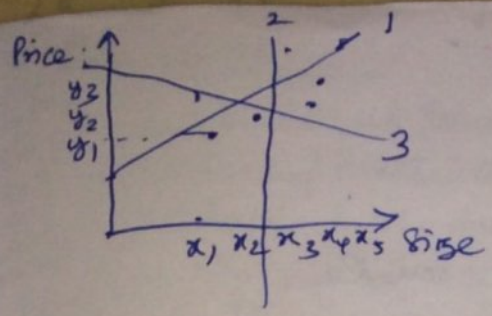
If curve fitting, use $f(\text{Size}) = w_2 \text{Size}^2 + w_1 \text{Size} + w_0 = \text{Price}$



$\text{Price} = w_0 + w_1 \text{Size} + w_2 \text{Size}^2 + \ldots + w_n \text{Size}^n$

Need to solve 'n' equations to solve $w_0, w_1, w_2, \ldots, w_n$.

Scanned by CamScanner



Best learning is with line 1

To quantify best fit,
use $(y_i - \hat{y}_i)^2$ instead of ~~Absolute~~ $(y_i - \hat{y}_i)$. Why?
Because we cannot differentiate on '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

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

$y_i = \text{Price}_i$.

So, for restaurant revenue, to minimize

$E(w_0, w_1) = \sum_{i=1}^{n} (\text{Price}_i - w_0 - w_1 \text{Size}_i)^2$ ← Solve minimization with $\frac{\partial E}{\partial w_0}$ and $\frac{\partial E}{\partial w_1}$

[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)

(xi) 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, Eg

Linear model = $w_0 + w_1 \text{Input1}_i + w_2 \text{Input2}_i + w_3 \text{Input3}_i$

Polynomial model → Could choose 2nd degree polynomial for Input1, and linear model for Input2 also.
(eg) $w_0 + w_1 \text{Input1}_i + w_2 \text{Input1}_i^2 + w_3 \text{Input2}_i$

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

Scanned by CamScanner

Google page ranking uses "Eigen Vector".

16/12/16 **Regression Model**

Y = response/outcome variable

$X1, X2, X3, \ldots X_p$ = explanatory/input variable

$Y = f(X_1, X_2, X_3, \ldots, X_p) + e$ → approximate general relation

Linear relationship is

$Y = b_0 + b_1X_1 + b_2X_2 + \ldots + b_pX_p + e$

For restaurant revenue, use cross-validation to estimate accuracy.
Get RMSE for each of the 10 CV iterations and take average.

(x) 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

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

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.

(xi) 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, problem, i.e. poor predictor

↳ In LR, if a predictor does not have linear relationship) it may still have non-linear relationship like ^2, ^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.

Scanned by CamScanner

Guess and refine

Only for ML probs not Math problems
- Batch Gradient Descent – gradient computation uses 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 $\sum$ involved.

If train data is very small, use 'Batch gradient' descent.
If very big, use Stochastic descent
↓
10000 iterations × 1 random sample each
due to less time for an iteration, converges faster

↓
100 iterations × 10000 samples each
converges slowly as iteration takes longer time

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

(*) 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 highly 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 $\alpha = 0.01$

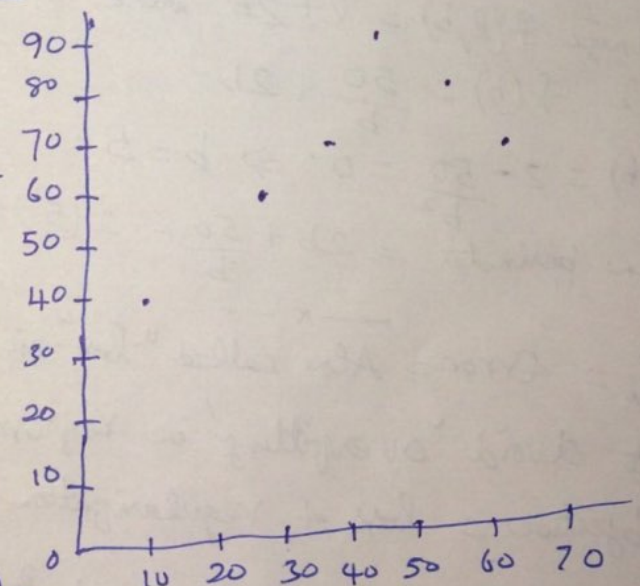
$$E(w_0, w_1) = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

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

$$\nabla E = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1} \right]$$

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

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



This is the gradient. To move towards minima,

$$w_0 = w_0 - \alpha \frac{\partial E}{\partial w_0}, \quad w_1 = w_1 - \alpha \frac{\partial E}{\partial w_1}$$

Batch gradient descent

$$= w_0 - \alpha \left(\sum_{i=1}^{n} -2 (y^{(i)} - w_0 - w_1 x_1^{(i)}) \right) = w_1 - \alpha \sum_{i=1}^{n} -2 (y^{(i)} - w_0 - w_1 x_1^{(i)}) x_1^{(i)}$$

Scanned by CamScanner

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

$$w_0 = w_0 - \alpha \cdot -2\left(y^{(i)} - w_0 - w_1 x_1^{(i)}\right)$$

$$w_1 = w_1 - \alpha \cdot -2\left(y^{(i)} - w_0 - w_1 x_1^{(i)}\right) x_1^{(i)}.$$

We shuffle the input data points.
In Epoch 1, use (shuffled) Input 1 for gradient in iteration 1, Input 2 for gradient in iteration 2, etc.
Before Epoch 2, again shuffle, and use similar approach.
We can stop in the mid. of an epoch also if enough convergence happened.
Take more 'epochs' than in batch gradient since delta change is smaller. But each epoch is faster.

Stochastic mini-batch gradient descent — Batch size = 3

One ~~epoch~~ - (1,4,5) & (2,3,6) - 2 iterations.
Show calculations for 1 epoch only.

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

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

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

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

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

$y_i - \hat{y}_i$ = Error = Also called "loss" in some books.
How to avoid 'overfitting' in regression?

(%) objective = loss + regularization

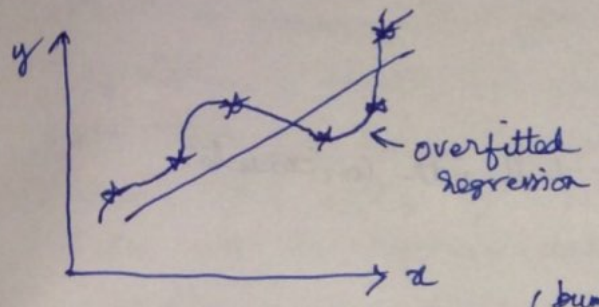
In ML approach - Information based, Probability, Nearest Neighbors, Ensemble, objective based (most dominant approach)

Mathematically show that,
Information based, Prob, NNeighbors, Ensemble can all be transformed / mapped to 'objective based' approach. But this mapping is not possible for deep learning

Scanned by CamScanner

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

Regularisation
- L1 regularization (Lasso)
- L2 regularisation (Ridge)
- L1+L2 regularisation (Elastic Net)
- Dropout (used in Neural Network)



If curve has too much ups and downs (bumpy curves) $\Rightarrow$ $w_0, w_1, w_2, \ldots$ values 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, regularise the w_i values.

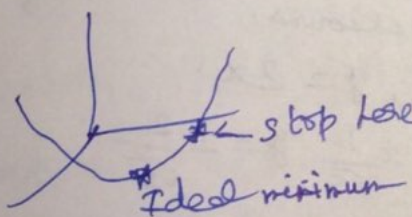
Objective (w_0, w_1) = minimise $\sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)^2$

such that $w_0 \leq \ldots$
$w_1 \leq \ldots$
} This detail differentiates L1, L2, L1+L2, dropout etc!

Regularization $\equiv$ Constraining weights/slopes.

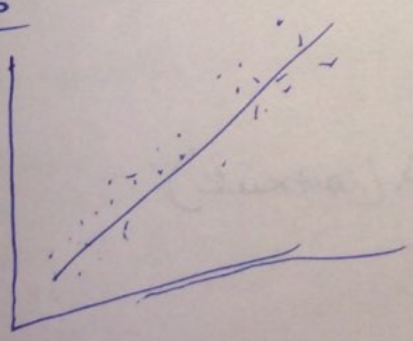
Can control regularization using parameters.

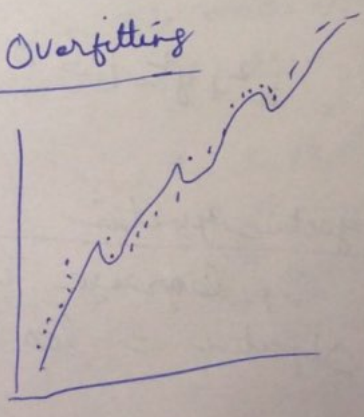
Regularisation has 'limited' advantage in linear regression
- can be used to identify unimportant predictors



Underfitting vs Overfitting

20/12/16





Scanned by CamScanner

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 $w_0, w_1, w_2, \ldots$
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

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

$x + y = 5 \rightarrow$ Equality Constraint

$x \leq 3, y \leq 5 \rightarrow$ Inequality constraint.

Approach with Constraint

- Convert constrained optimization problem to unconstrained version and solve it.

Lagrange idea (for equality constraint)

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

($f(x,y)$: Plane equation; $x^2 + y^2 = 1$: Cylindrical shape)

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

$$\frac{\partial \Lambda}{\partial x} = 1 + 2\lambda x = 0$$

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

$$\frac{\partial \Lambda}{\partial \lambda} = x^2 + y^2 - 1 = 0$$

3 equations, 3 unknowns.
Eliminate λ; $y = 2x$.
Solving, $x = \pm\frac{1}{\sqrt{5}}$ $y = \pm\frac{2}{\sqrt{5}}$.

Objective function

By lagrange idea.

Objective = loss function + λ(constraint)

Scanned by CamScanner

Common regularisation for linear models.

* Ridge constraint: $\sum_{i=1}^{p} \beta_i^2 < s$.

* Lasso constraint: $\sum_{i=1}^{p} |\beta_i| < s$.

- A tuning parameter alpha/lambda imposes a penalty on the size of co-efficients.
- Instead of minimizing "loss" (MSE), minimise "loss + penalty".
- Tiny alpha ⇒ almost like linear model
- Huge alpha → Underfitting. Coefficients shrunk 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 plane

Intersection of this shape with the gradient descent path gives constrained minima. Since 'plane' in Lasso, perhaps w_i can be 0.

(*) method = "lm" – linear regression – may not be doing regularisation: check!
– uses "mini-batch" approach.

(Error) Plot of $w_0, w_1, \ldots$ gives a 3D curve. Each point on the 3D curve corresponds to a particular (w_0, w_1) value → This value corresponds to a line equation of the input variables – (eg) $2w_1 + 3$.

21/12/16 Regression – Model Building

- KNN
- Information based
- Objective based learning
  - Linear/Polynomial regression ⇒ Objective & subject to regularisation
    - Ridge
    - Lasso
    - Elastic Net
  - Logistic Regression (cross entropy loss)
  - SVM (Hinge loss)
  - Neural Network
  - Perceptron

Used for all objective based learnings
- Batch Gradient Descent
- Stochastic GD
- Mini batch GD *

Loss (Squared loss) $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Scanned by CamScanner

Linear models bad - Can we build a curve?

Ensemble learning
- Bagged tree
- Random forest = bagged tree + randomisation at split level
- Gradient Boosting (GBM)

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

Boosted Models

Train Data

DataSet 1 → Model 1 → DataSet 2 → Model 2 → DataSet 3 → Model 3 → DataSet 4 → Model 4 → ...

Each model corrects the mistakes/shortcomings of its predecessor.
Build models in a sequence. "Additive models" + "Stagewise".

DataSet 1

Size	Price	$\hat{Price}$
1000	10 lakh	9 lakh
500	5 lakh	4 lakh
600	8 lakh	6 lakh

DataSet 2

Size	Price	$\hat{Price}$
1000	1 lakh	94000
500	1 lakh	93000
600	2 lakh	1,74,000

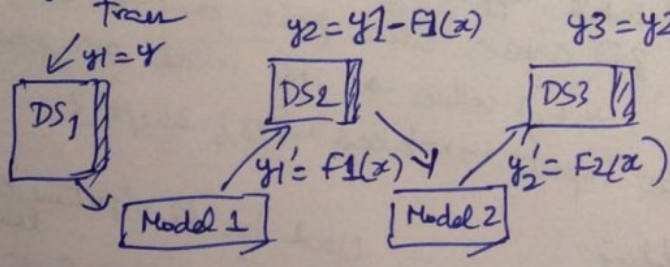
Price - $\hat{Price}$ of Dataset 1

Models = Tuning parameter. Too many models → overfit.

In classification, If dataset$_i$ (model$_i$) has correct predictions, we don't carry them to dataset$_{i+1}$ (x_i)

But in regression, Even if model$_i$ has some "good enough predictions", we still carry those 'good enough' inputs to dataset$_{i+1}$ → the gap could be close to zero, but that's fine since the model$_{i+1}$ will not use it for learning.

Many 'Boosted ideas' present - 'Gradient Boosted' model. (integrated with caret "gbm")
(classic AdaBoost, XGBoost (not integrated with caret))



$y_i - F_1(x_i) = \}$ Residuals of model$_i$
or $y_i - y_i' = \}$

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

$J = \frac{1}{2}\sum_{i=1}^{m}\left(y_i - F(x_i)\right)^2$

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

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

Scanned by CamScanner

~~For any~~ Benefit of GB algorithm using gradient is that it allows us to consider other loss fns and derive other corresponding algorithms accordingly.

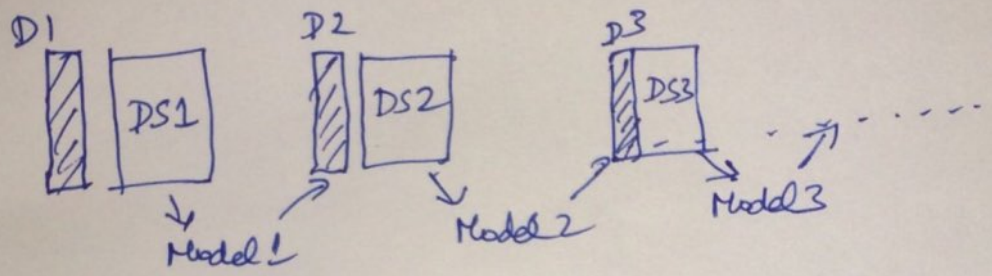
Boosting gives best performance on weak / underfitted models. (high bias / low variance)

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'
↓ 'Bagging' " — use full depth.

"Ada Boost" — Adaptive boosting used for classification



D'n — weights of each sample.
D1 — all samples equal weight
D2 — samples correctly predicted by Model 1 — low weight
└ samples wrongly predicted by model 1 — High weight
Model2 tries harder to classify these samples correctly.

Final output = weighted summation of all models output.

Why not take only misclassified data to DS2?
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 $\Sigma -p_i \log p_i$, 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.

Scanned by CamScanner

22/12/16

Ensemble

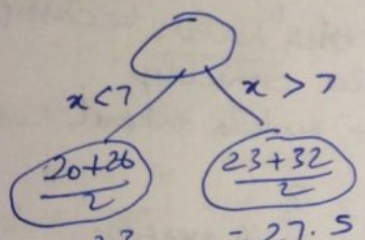
- Bagging
- Bagging + random feature for split } Improve Any overfitted Model (eg) deep trees
- Boosting (Weak Models) – gives better result than Random Forest.
  Use 'underfitted trees' – depth of '3 to 7'
  - Gradient Boost – Regression + classification
  - Ada Boost, C5.0 – only classification
  - 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)

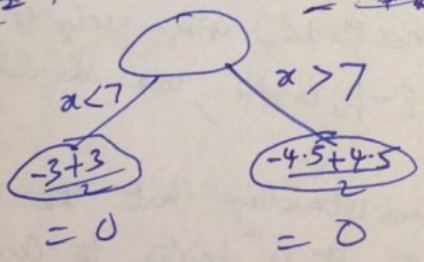


x	y	$\hat{y}$
5	20	23
7	23	27.5
9	32	27.5
6	26	23

Round 2 → $(y-\hat{y})$ of round 1 $= -\frac{\partial L}{\partial F_1}$

x	y_1	$\hat{y}_1$
5	-3	0
7	-4.5	0
9	4.5	0
6	3	0

$F_2(x)$ Shallow tree – ~~First Model~~.



Final Model = Model 1 + Model 2 + ...

(eg) $F(8) = f_{Model\,1}(8) + f_{Model\,2}(8) + \cdots$

$= 27.5 + 0 + \cdots$

We can also take

$F(8) = f_{Model\,1}(8) + \alpha F_{Model\,2}(8) + \alpha F_{Model_3}(8) + \cdots$

↑ learning rate

Scanned by CamScanner

So, $F = F + \alpha \frac{\partial L}{\partial F_1} + \alpha \frac{\partial L}{\partial F_2} + \cdots$ α = 'shrinkage parameter'

Small $\alpha \Rightarrow$ More rounds needed

$\alpha \in [0,1]$.

(3) XGBoost = "Regularised" gradient boost.
- like 'cp' for trees & Lasso/Ridge for linear regression.
- Popular in Kaggle

(4) Overfitted models not good for boosting $\Rightarrow$ Data is already very good and do not need to boost.

(5) If $\alpha = 1$, we may not get good convergence, like step-factor in gradient computation. So, tune 'α' carefully.

Boosting is a general thought applicable to many models like trees, linear regression, logistic regression etc.

(eg) blackboost — mstop, maxdepth
(# Trees) (Max Tree depth)

C5.0 — improvement over C4.5 — Boosting added.

eXtreme Gradient Boosting

xgbLinear (boosting on linear regression)
- nrounds (# boosting iterations) — boosting parameter
- lambda (L2 regularisation) } 'elastic' — L1+L2 for linear models
- alpha (L1 regularisation) }
- eta (learning rate / shrinkage) — boosting parameter

XgbTree
- nrounds
- max_depth
- eta (Shrinkage)
- gamma (Minimum loss reduction) — individual tree control
- Colsample_by tree, min_child_weight — tree control params.

Stochastic Gradient boosting:

gbm (boosting without regularisation)
- n.trees (# Boosting Iterations)
- interaction.depth (Max Tree Depth)
- shrinkage
- n.minobsinnode (Min Terminal node size).

Adaboost — of historical interest only — only for classification.
Classification done better with Gradient boost or Xtreme Gradient Boost.

Scanned by CamScanner

classification with gradient boost

→ only 1 column has 1.

x	y
5	1
7	0
9	1
6	1
8	0

⇒ (One-hot encoding)

x	y_1	y_2
5	0	1
7	1	0
9	0	1
6	0	1
8	1	0

target variable → vector

First thing to do for classification with boosting

Use probability values.

$F(x)$ < Logistic Regression
< Shallow Trees.

prob. of 0, prob. of 1.

Round 1

x	y	$\hat{y}$
5	1	1
7	0	1
9	1	0
6	1	1
8	0	0

x	y_1 y_2	$\hat{y}_1$ $\hat{y}_2$
5	0 1	0.3 0.7 → try to decrease $\hat{y}_1$, increase $\hat{y}_2$
7	1 0	0.7 0.3 → try to increase $\hat{y}_1$, decrease $\hat{y}_2$
9	0 1	0.6 0.4
6	0 1	0.2 0.8
8	1 0	0.9 0.1

$(y_1 - \hat{y}_1,\ y_2 - \hat{y}_2)$ → input for round 2

This approach here on, 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 Trees + randomisation at split level.

For splitting a node, pick the best variable among randomly selected features.

"gbm" – boosting + no regularisation.
– interaction.depth = tree depth.
– shrinkage = c(0.1, 0.3, 1)
– n.minobsinnode = 5, prepruning strategy – don't split node if < 5 samples.

If we give 'small' shrinkage, increase the number of trees.

Plot(knnmodel)

Plot(gbmmodel) – gives great picturisation.

method = "xgbLinear" – extreme boosting on linear models.

Know "variable Importance" across models & we can play around with them.

↳ Remove all the "low importance" variables and re-build

Plot(VarImp(gbmmodel))

Scanned by CamScanner

For classification, gbm, xgb give better performance than Ada, C5.0.

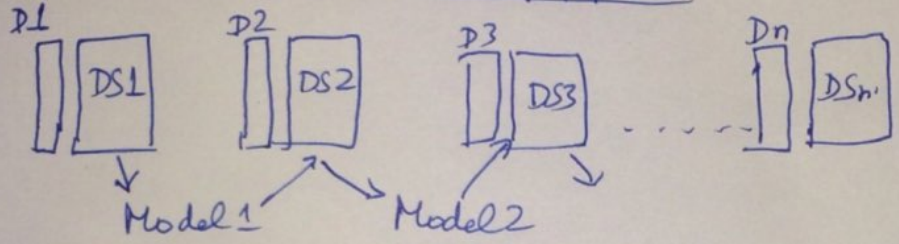
26/12/16

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 } Both classification & regression
– Neural Network

Learning AdaBoost since its very popular



D_i = distribution of weights. Sum of weights = 1.
$D_1 = (1/n, 1/n, 1/n, \ldots)$ – all have equal weights at start.

At later D_i, $(i>1)$, weights of correct classifications is decreased
weights of wrong classifications is increased.

Regularise AdaBoost by the number of iterations.

AdaBoost → Adaptive Boosting : algorithm adapts weights & base learners and training examples.

(*) For AdaBoost to work, y_i input and output $h_x(i)$ should be +1 or −1 only. If we use (0,1) instead of (−1,1), formula won't work.

$\Rightarrow y_i h_x(i) = +1 \Rightarrow$ correct classification
$-1 \Rightarrow$ misclassification

(*) See PPT for details of AdaBoost. ⇒ Ada works only for binary classification

rpart – CART – does weighted learning. Multiply weights with entropy.
So, Ada works on top of both CART and C4.5, but normally CART is used.

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 its doing 'average' of k-similar observations

Scanned by CamScanner

For classification, gbm, xgb give better performance than Ada, C5.0.

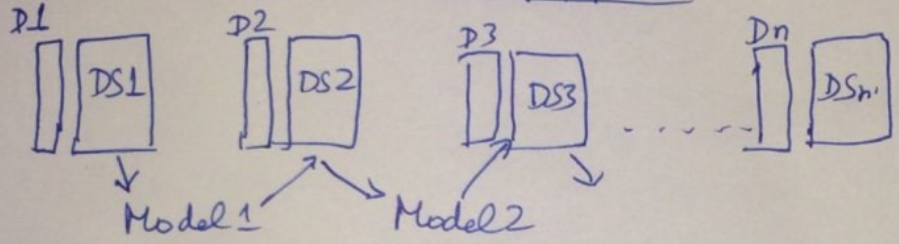
26/12/16

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 } Both classification & regression
– Neural Network

Learning AdaBoost since its very popular



D_i = distribution of weights. Sum of weights = 1.
$D_1 = (1/n, 1/n, 1/n, \ldots)$ – all have equal weights at start.

At later D_i, $(i>1)$, weights of correct classifications is decreased
weights of wrong classifications is increased.

Regularise AdaBoost by the number of iterations.

AdaBoost → Adaptive Boosting : algorithm adapts weights & base learners and training examples.

(*) For AdaBoost to work, y_i input and output $h_x(i)$ should be +1 or −1 only. If we use (0,1) instead of (−1,1), formula won't work.

$\Rightarrow y_i h_x(i) = +1 \Rightarrow$ correct classification
$-1 \Rightarrow$ misclassification

(*) See PPT for details of AdaBoost. ⇒ Ada works only for binary classification

rpart – CART – does weighted learning. Multiply weights with entropy.
So, Ada works on top of both CART and C4.5, but normally CART is used.

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 its doing 'average' of k-similar observations

Scanned by CamScanner

27/12/16

How weights of samples used in AdaBoost

Entropy = $\Sigma p_i \log p_i$

↳ Instead of p_i, use the weights done.

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.

In linear regression, Lasso 'eliminates' useless features (close to zero)

In polynomial regression, Lasso 'smoothens the curves'.

28/12/16 Regularisation – control of overfitting

When does overfitting happen?

1) Tree Learning – Very deep trees ⇒ 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) → 'fraction of sample' used to grow each tree
(Linear Regression) → Lambda (L2) & Alpha (L1).

5) Probabilistic learning – NB not impacted much by overfitting
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 Regression – Large number of features (x_i) ⇒ 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, ie, w_i values
– w_i values look similar across resamples

(L1) Ridge Constraint:
$\sum_{i=1}^{n} w_i^2 < S$, n = # features
If 'S' too low ⇒ underfitting

(L2) Lasso constraint
$\sum_{i=1}^{n} |w_i| < S$;

Scanned by CamScanner

The idea of L1, L2 regularisation is applicable in many areas - linear regression, logistic regression, SVM, NN.

'fraction' in "lasso" (linear) regression.

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \text{fraction} \left(\sum_{i=1}^{b} |w_i| \right).$$

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 content should not be lost (or) 3D → 2D image. VOB → MP4 file

(eg2) Clustering

- find group of customers who are identical. Create a scheme to attract this group.
- download internet content; group them by category

(eg3) Outlier detection

- In restaurant revenue problem, some revenues are unusually high; detect them

(eg4) Feature extn from unstructured data

- Given an image, identify interesting characteristics.
- Show interesting insights into data.

Reinforcement learning - another category like supervised, unsupervised.

Frequent pattern mining - Supervised / Unsupervised - Subjective choice

(i) In predictive analytics, use of 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.

(ii) Unsupervised learning - Can be an apple by itself and not necessary to be a part of supervised learning. (eg) extract "car number" from image.

(iii) In reinforcement learning, there is interaction with environment. So, keep learning from interactions. It involves 'some extra' on top of unsupervised learning.

Scanned by CamScanner

Principal component analysis (PCA) – for feature extraction

PCA works only on numerical data. So, transform categorical to numeric (using dummy vars) before using PCA.

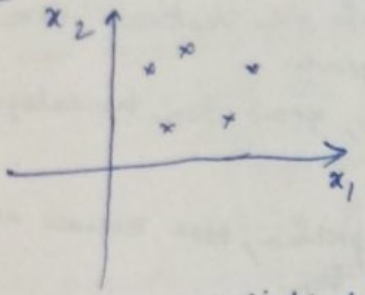
PCA gives error on 'zero variance' features. Remove them first.

Transform/Reduce both train and test data using PCA. Then proceed with the usual process.

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 algos. have target. Try for representative train data.

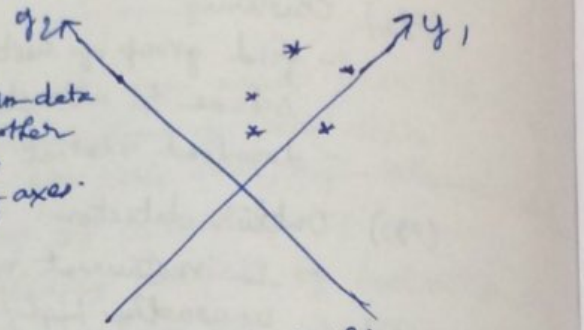
↳ In Kaggle, binding rows of train & test and doing feature engineering ⇒ people will get knocked out in private leaderboard.

30/12/16



x_1, x_2 are not prioritised
All are equally important

Transform data into another set of axes.

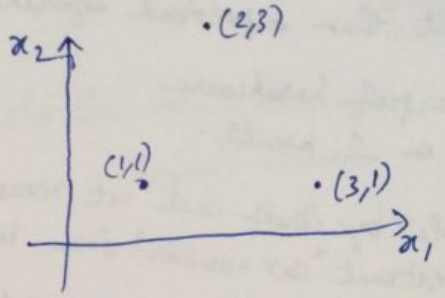


y_1 most important (Principal component 1)
y_2 second most imp. (Principal component 2)

– By ordering by importance, can ignore the least important features.

PC1 > PC2 > PC3 > · · · – · · > PC1000

Pattern in data – Quantification
– Sum of variances of data along each dimension

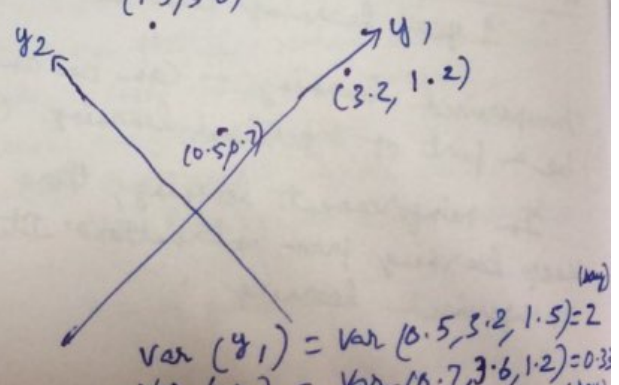


$(1,1) \equiv 1\hat{i} + 1\hat{j}$

$\text{Var}(x_1) + \text{Var}(x_2)$
$= \text{Var}(y_1) + \text{Var}(y_2)$
$= 2.33$.

Now, can ignore y_2. We have a 1-D axis of y_1 only.

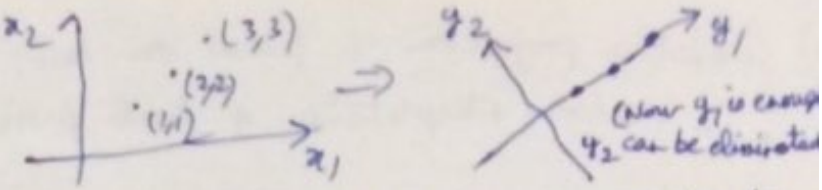
$\text{Var}(x_1) = \text{Var}(1,2,3) = 1$
$\text{Var}(x_2) = \text{Var}(1,3,1) = 1.333$



$\text{Var}(y_1) = \text{Var}(0.5, 3.2, 1.5) = 2$ (avg)
$\text{Var}(y_2) = \text{Var}(0.7, 3.6, 1.2) = 0.33$ (avg)

Scanned by CamScanner

(1,1) (2,2) (3,3)
Var(x_1) = Var(1,2,3)
Var(x_2) = Var(1,2,3)



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

- Genomics – thousands of genes, millions of DNA polymorphisms
- Documents – thousands of words, millions of bigrams
- Images – millions of pixels.

Principal components

Find the new co-ordinate 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 co-ordinate 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; aka, non-linear PCA.

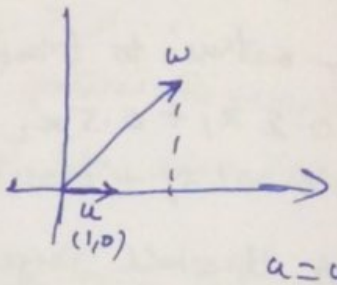
Understanding PCA

- How to project a point on a different axis?
- How to find axis that captures variances in descending order?

Scanned by CamScanner

1) Computing projection of points on new axis
- Vector interpretation of points provides solution.

Find projection of w vector along 'u' direction
= w·u (=dot product ≡ shadow)
= $w \cos\theta = w_1 u_1 + w_2 u_2$



u = unit vector

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 vectors V.

Maximise $var(V) = V^T A V$ subject to $V^T V = 1$
$\Rightarrow$ unit vector.

$var(V, \lambda) = V^T A V - \lambda(V^T V - 1)$

Apply calculus and solve for V and λ.

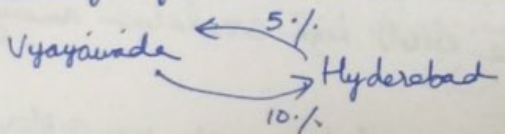
$V = [0\ 1]$ $V^T = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow V \cdot V^T = [1]$
$V = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $V^T = [0\ 1] \Rightarrow V^T V = [1]$

$V^T A V$ = Variance of all data points along V direction

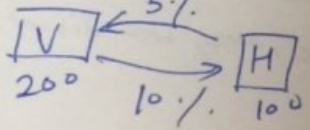
Search Engine Optimization (SEO) for improving "Page Rank"
- involves hacking of page rank algorithm

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?



$H_0 = 100$, $V_0 = 200$ (initial population)
$H_1 = 100 \times 95\% + 200 \times 10\% = 115$
$V_1 = 200 \times 90\% + 100 \times 5\% = 185$
$H_2 = 115 \times 95\% + 185 \times 10\% = 127.75$
$V_2 = 185 \times 90\% + 115 \times 5\% = 172.25$

$H_{t+1} = 0.95 H_t + 0.1 V_t$
$V_{t+1} = 0.05 H_t + 0.9 V_t$

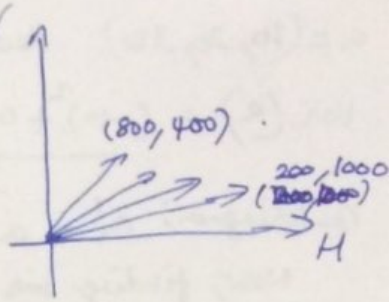
$$\begin{bmatrix} H_{t+1} \\ V_{t+1} \end{bmatrix} = \begin{bmatrix} 0.95 & 0.1 \\ 0.05 & 0.9 \end{bmatrix} \begin{bmatrix} H_t \\ V_t \end{bmatrix}$$

After many years, equilibrium [illegible]

Scanned by CamScanner

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, $MV = \lambda V$ No change in direction. Magnitude impacted by λ.
↳ Eigen value

How does Google model Eigen Vectors?

After people start visiting web sites, they keep jumping websites and finally land on some important pages - these important pages have high page ranks.

People jumping across websites & finally going to important sites ≡ matrix multiplications leading 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 ≡ Eigen value of 1.

In PCA context,
100 features ⇒ 100 dimensions → 100 eigen vectors.
Eigen vectors with max. eigen values are the most important principal components.

$$f(v, \lambda) = v^T A v - \lambda (v^T v - 1).$$

$$\frac{\partial f}{\partial \lambda} = v^T v - 1$$

PCs are eigen vectors of matrix A.

$v^T v - 1 = 0 \Rightarrow v^T v = 1$

$$\frac{\partial f}{\partial v} = 2Av - 2\lambda v.$$

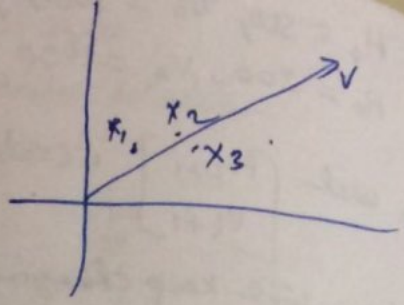
$2Av - 2\lambda v = 0$
$\Rightarrow Av = \lambda v$
↳ eigen value

$n \times n$ matrix has 'n' eigen vectors.

In PCA, we need to understand what 'A' is.
A is a $n \times n$ matrix where n = # features/dimensions.

Scanned by CamScanner

$X^{(1)} \cdot V, \; X^{(2)} \cdot V, \; X^{(3)} \cdot V, \ldots$



$a = (10, 20, 30)$ Mean $(a) = 20$

$\text{Var}(a) = \frac{(-10)^2 + 0^2 + (10)^2}{3} = 66.67$

Centering a: $a' = a - \bar{a} = (-10, 0, 10)$.

Now, finding var is easier :.

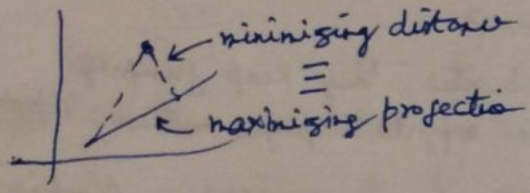
$X \cdot V$ = Projection of X on vector V.

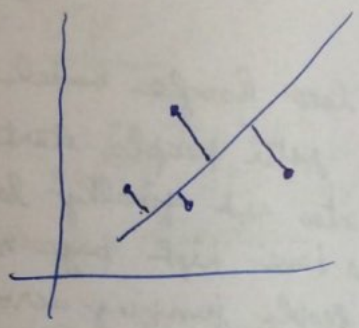
$(X^{(1)} - \bar{X}) V, \; (X^{(2)} - \bar{X}) V, \; (X^{(3)} - \bar{X}) V, \ldots$

$$\text{Variance} = \frac{\sum_{i=1}^{n} (x^{(i)} - \bar{x})^2}{n} = V^T \cdot A \cdot V$$

↳ A = Covariance matrix

3/1/2017 **Meaning of Capturing variance**





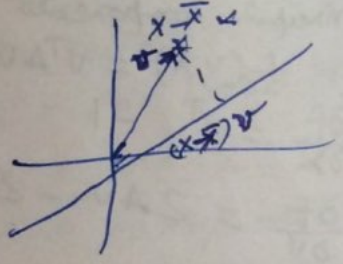
Eigen values of eigen vectors determine the importance of the principal components

X1	X2	
1	2	$\text{covar}(X1,X1) = \text{var}(X1) = (1-2)^2 + (2-2)^2 + (3-2)^2/3$
2	3	$\text{covar}(X2,X2) = \text{Var}(X2) = (2-3)^2 + (3-3)^2 + (4-3)^2/3$
3	4	$\text{covar}(X1,X2) = \text{covar}(X2,X1) = (1-2)(2-3) + (2-2)(3-3) + (3-2)(4-3)/3$

Co-variance matrix

$$\begin{bmatrix} \text{Cov}(X1,X1) & \text{Cov}(X1,X2) \\ \text{Cov}(X2,X1) & \text{Cov}(X2,X2) \end{bmatrix}$$ → Symmetric always since $\text{Cov}(X1,X2) = \text{Cov}(X2,X1)$.

$$\begin{bmatrix} \text{Cov}(X1,X1) & \text{Cov}(X1,X2) & \text{Cov}(X1,X3) \\ \text{Cov}(X2,X1) & \text{Cov}(X2,X2) & \text{Cov}(X2,X3) \\ \text{Cov}(X3,X1) & \text{Cov}(X3,X2) & \text{Cov}(X3,X3) \end{bmatrix}$$



$$\text{var}(V) = \sum_{\text{each point}} [(x - \bar{x})^T \cdot V]^2$$

$$\text{var}(V) = \sum [(x^{(i)} - \bar{x})^T \cdot V]^2$$

(*)
$$= \sum V^T (x^{(i)} - \bar{x})(x^{(i)} - \bar{x})^T V$$

$$= V^T \left[\sum_i (x^{(i)} - \bar{x})(x^{(i)} - \bar{x})^T \right] V$$

$$= V^T A V \quad \text{where } A = \sum (x^{(i)} - \bar{x})(x^{(i)} - \bar{x})^T$$

= Covariance matrix

Scanned by CamScanner

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.

Maximize $f(v,\lambda) = v^T A v - \lambda (v^T v - 1) = 0$

$\frac{\partial f}{\partial \lambda} = -v^T v + 1 = 0$

$\frac{\partial f}{\partial v} = 2Av - 2\lambda v = 0$

$\Rightarrow Av = \lambda v$

$\Rightarrow v$ is eigen vector of A, λ is eigen value

$V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots v_n \end{bmatrix}$.

$V^T V = v_1^2 + v_2^2 + \cdots + v_n^2$

$\therefore \frac{\partial}{\partial v}(V^T V) = 2v$.

$\lambda_1 = 73.718$, $\lambda_2 = 0.384$, $\lambda_3 = 0.298$

↓ 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 λ_1 is the 1st principal component.

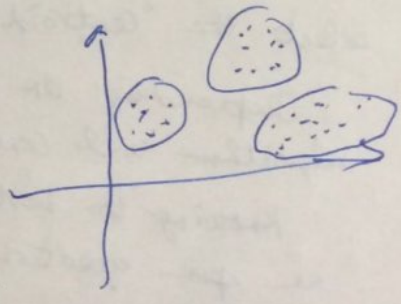
PCA cannot remove the impact of outlier. PCA also makes sense only when there is linear relationship among the features.

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, $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_d$ with the corresponding eigen vectors $(u_1, u_2, \ldots, u_d)$
4. The principal component matrix is: $U = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_d \\ | & | & & | \end{bmatrix}$.

(X) 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.

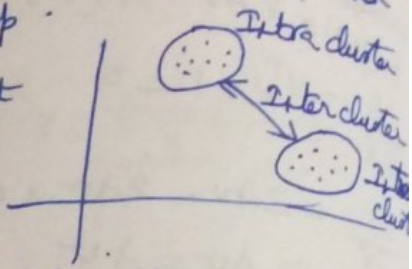
Scanned by CamScanner

3) Correlated variable clustering
- CorrPlot 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

Similarity $\propto \frac{1}{\text{Distance}}$

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 minimise the sum of distances of the points to their respective centroid
- Number of clusters, K, must be specified.

Algo
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-existant.
4) Repeat steps 2-3 with the new centroids.
5) Repeat the above steps until the membership (to cluster) of points gets stabilized.

There is mathematical proof that distance is minimized when we select the "centroid" a 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 = Within Sum of Squares]

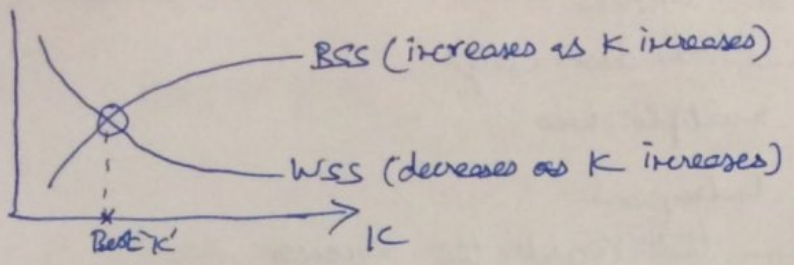
Scanned by CamScanner

Separation – measured between cluster sum of squares.

$$BSS = \sum_i |C_i| (m - m_i)^2.$$

[BSS = Between sum of squares]

where $|C_i|$ is size of cluster i, m is the centroid of the whole data set.



- BSS + WSS = constant
- WSS (coherion) 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 → Transform it to structured matrix-like data and run clustering on it.

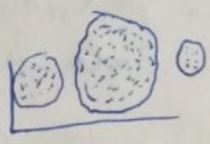
Deep neural network (Word2Vec) converts a text document into vector format.

Image2Vector – Converts Images to Vector Graphics.

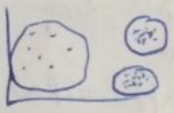
Understanding video content – Videos are automatically captioned

Caret = classification + Regression

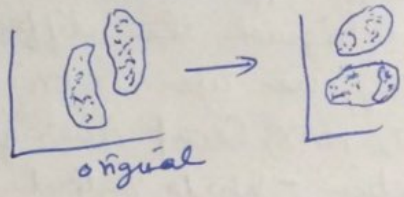
Limitations of K-means



- Different cluster sizes
- Different densities in different clusters (Overcome using DBScan algorithm using density as the core math)



- Non-globular shapes not detected
- problems with outliers.



Use many clusters & combine manually some of the clusters to overcome some of the issues

Clustering

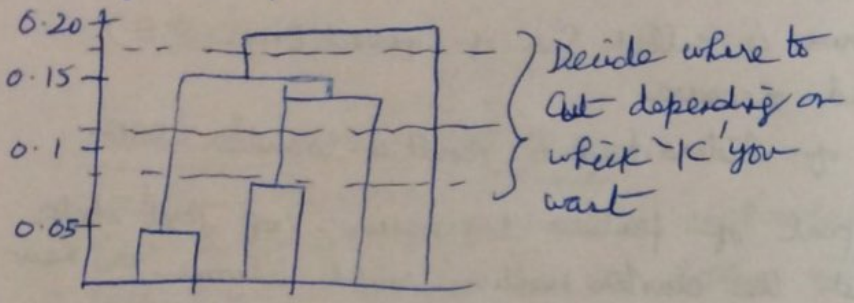
- Hierarchical approach (eg) hclust
- Density based (eg) DBScan
- Partitioning approach (eg) K-means

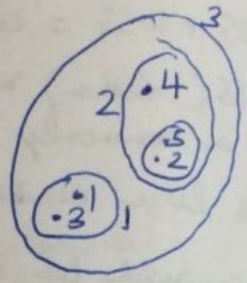
Scanned by CamScanner

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 sequences 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.

corrplot (correlation, order = "hclust")

Scanned by CamScanner

K-means, inspite of its limitations, is most popular due to its simplicity.

Use the R method hclust().

"Factor Analysis" not used any more. PCA better. Even PCA is being replaced by auto encoders.

9/1/2017

Recommenders

Rating prediction
Top-N Recommendations

Neural Turing Machines – In 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. (eg) 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

Scanned by CamScanner

Recommenders are more predictive than descriptive because we are 'forecasting', for eg, what movies I will watch.

Associative analysis (Market basket analysis) – What things do people buy together? 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 – Not seen in this course – For agent-based learning (eg) car driving, game playing. Not just deployed 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.

10/1/2017

Recommender
- rating prediction (rating of not watched movies)
- top N recommendation (recommend buys based on what is bought).
  └ done based on 'rating prediction' output.

Scanned by CamScanner

Mean based predictors

- Global mean rating
- Per-user mean rating
- Per-movie mean rating
- Random rating

} Ad hoc predictors · Low variability · Underfitted

Collaborative filtering approach

(rating 1-5 scale)

	$Movie_1$	M_2	M_3	M_4
$User_1$	5	4	?	-?
$User_2$	-	3	5	-
$User_3$	-	-	5	5
$User_4$	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_4 movie

Here, $User_2$ and $User_4$ are closer to $User_1$.

$User_2$ ($Weight_2$), $User_4$ ($Weight_4$) ↳ Similarity with $User_1$.

$5 \, Weight_2 + 4 \, Weight_4 = User_1$ M_3 rating

} User-based collaborative filtering (UBCF)
Adv: Serendipity; subtle discoveries

To suggest movie for $User_1$, predict rating for M_3 and M_4;
Recommend the highest predicted movie.

* Predict rating for an item for an user u, give weight to each other user's 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 $User_1$ may not have given a rating to a movie - then we cannot predict.

Rating for an 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:

$$pred(u,i) = \frac{\left(\sum_{v \in k\text{-similar users}(u)}\right) usersim(u,v) * r_{vi}}{\left(\sum_{v \in k\text{-similar users}(u)}\right) usersim(u,v)}$$

Denominator used to normalise rating in range 1-5.

Can modify formula to include user or item bias factor.

(ex) $$\frac{(0.9 \times 5) + (0.9 \times 4) + (1 \times 5)}{0.9 + 0.9 + 1}$$

Scanned by CamScanner

UBCF : Top-N recommendations

Find N items 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 $I_{recmd} = I_{candidate} - I_{purchased}$
- Reorder items in I_{recmd} using following formula and recommend first n items :

$$pred(u,i) = \frac{\sum_{v \in Ksimilarusers(u)} usersim(u,v) * r_{vi}}{\sum_{v \in similarusers(u)} usersim(u,v)}$$

Issues with UBCF

1) User cold-start problem : How to find similar users for the new user on-board?

If rating matrix has 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.

11/1/2017

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* not directly modification)
- User profile normally contains less ratings than a product profile. Hence, addresses sparsity issue
- Improves scalability - item similarity more stable than user similarity ; enables precomputing of item-item similarity

Scanned by CamScanner

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:

$$pred(u,i) = \frac{\sum_{j \in Ksimilaritems(i)} itemsim(i,j) * r_{uj}}{\sum_{j \in Ksimilaritems} Itemsim(i,j)}$$

IBCF: Top-N recommendations

Find N items most likely to be purchased by an user u

- Get all items purchased by u: $I_{purchased}$
- For each item ~~purchased~~ in $I_{purchased}$, find K most similar items: $I_{candidate}$
- Remove unavailable items from $I_{candidate}$
- ....

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

(eg) 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. (eg) static profiles of movies and static profiles of users.

Idea: 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 users interest changes

Lack of serendipity

Scanned by CamScanner

(*) A combination of IBCF 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 → [ ] = more features, less input.

Narrow format → [ ] = less features, more input.

Narrow format

id	UserID	MovieID	Rating
	u_1	m_1	5
	u_1	m_2	6
	⋮	⋮	
	u_2	m_1	
	u_2	m_2	

Wide format

	m_1	m_2	..	m_n
u_1				
u_2				
⋮				

To transform narrow → wide format:
- library (reshape2)
- acast() function

'recommenderlab' doesn't understand 'real-valued' matrix. Need to convert it to "realRatingMatrix".

Recommenderlab tried to create APIs for all recommendation problems. Design inspired by 'caret' package.

Java: int ≡ Integer.

Illy, matrix of real numbers ≡ 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.

(*) Recommenders & Imputation both deal with missing values. But, because of different domains, our approach/algorithm is different.

(*) 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. [Evaluate 'rating']

- We make some available (non-NA) ratings as validation set. They are masked (held out) and use RMSE to evaluate.

Scanned by CamScanner

For holdout set,
- We mark / holdout only on a subset of users. We mask 'some' of the ratings of the user-subset.

method = "cross-validation", given = 2.

Recommenderlab
- separate 'train' and 'evaluate' steps.

cross-validation NOT done for parameter tuning of 'given'. It is purely to evaluate - 'given' value based on 'sparsity' of input matrix.

The hyper parameters to tune are '# similar user/items'.

Evaluate "Top-N recommendation"
- Order of recommended movie does not matter. Just check if 'liked' movie is recommended or not.
- given = 2, goodRating = 2.5 } Neither are hyperparameters; purely for evaluation
  - given → based on sparsity of matrix
  - goodRating → based on business decision

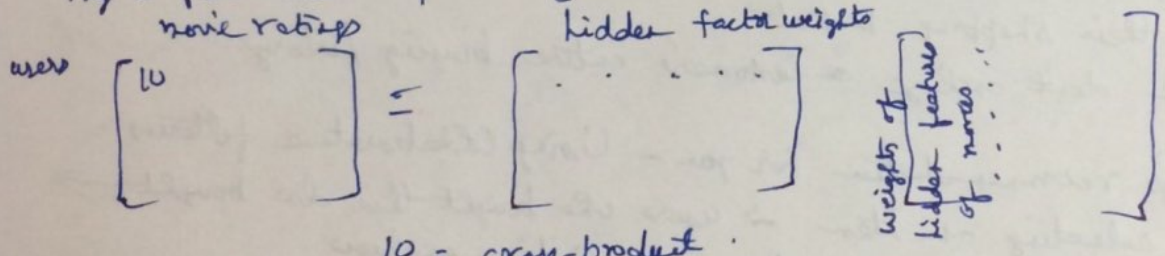
13/1/2017 To decide on method for distance, like "cosine", "Euclidean", etc use 'evaluate' and find RMSE.

UBCF takes long time to compute similarity scores; Use Spark / h_2O.

Recommender() — computes similarity matrix
predict() — uses similarity matrix to predict.

Latent factor approach -

Try to find hidden factors behind user's ratings



10 = cross-product.

SVD = Singular Value Decomposition — unfit for our problem since the input is 'sparse' matrix.

Optimisation-based value decomposition —

$r = p \times q$

5×6 matrix = 5×3 matrix × 3×6 matrix → Here '3' hidden features

Need to tune the '# hidden features'.

In production environment, still 'collaborative filtering' is used.

Latent-factor used only in competition.

Scanned by CamScanner

$$\begin{bmatrix} 5 & NA & NA \\ NA & 4 & NA \\ NA & NA & 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 1 \end{bmatrix}$$

$\rightarrow 6+6 < 12$ Unknowns.

Use the non-NA values to guess the right-side matrixes.

$$f(p_{11}, p_{12}, p_{21}, p_{22}, p_{31}, p_{32}, q_{11}, q_{12}, q_{13}, q_{21}, q_{22}, q_{33})$$
$$= \left[r_{11} - (p_{11}q_{11} + p_{12}q_{21})\right] + \left[r_{22} - (p_{21}q_{12} + p_{22}q_{22})\right]^2 + \left[r_{33} - (p_{31}q_{13} + p_{32}q_{23})\right]^2.$$

Minimize 'f' using available values.

Although powerful and found in 2002, not used widely since poor 'interpretability'; can't explain to business heads.

$$E = \sum e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2$$

16/1/2017

Assignment -1 Recommenders

- Euclidean similarity = $1/$ Euclidean distance
  $= 1/\sqrt{(\text{item}_1 \text{ diff})^2 + (\text{item}_2 \text{ diff})^2 + \dots}$.

Problem 1

Problem 3 SVD() 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 → users who bought this also bought → Based on association analysis

For set of transactions T, goal to find all rules having

support $\geq$ minsup threshold
confidence $\geq$ minconf threshold

Metrics:

{bread} ⇒ {milk}
{soda} ⇒ {chips}
{bread} ⇒ {jam}

≡ {people who bought item} ⇒ {also bought}

Support (S)

- Fraction of transactions that contain both X and Y; $\{X\} \Rightarrow \{Y\}$

Confidence (C)

- Measures how often items in Y appear in transactions that contain X

Scanned by CamScanner

Support, $S = \frac{\sigma(\{Bread, Milk\})}{\#\text{ transactions}} = 0.38$ — 38% transactions have both bread and milk.

Confidence, $C = \frac{\sigma(\{Bread, Milk\})}{\sigma(\{Bread\})} = 0.75$. — 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) (Unsupervised ML)
- If 'Milk' is bought by only 30% people, (less than threshold) then don't consider {Milk, Jam}, {Milk, Yogurt} associations.
- Selection of 'support threshold' has huge impact on computation time.

A✓ B✓ C✓ D✗ E✓

AB✗ AC✓ AE✗ BC✓ BE✓ CE✓ → Won't get ACE✗ since AE is involved → BCE (using support)

BCE → BC→E, BE→C, CE→B, B→CE, C→BE, E→BC — Select rule based on confidence

Transactions
ACD
BCE
ABCE
BE
ABCE

Freq Item	Support
BCE, AC	60%
BC, CE, A	60%
BE, B, C, E	80%

Rule	Confidence
BC→E	100%
BE→C	75%
CE→B	100%
B→CE	75%
C→BE	75%
E→BC	75%

Association analysis also used to place items nearby in supermarkets

When user bought B, should we show C or E first?
- Depends on confidence of B→C vs B→E.

To overcome Cold Start
1) Show most popular items.

Scanned by CamScanner

17/1/2017

dplyr useful to preprocess data; almost SQL-like syntax.

Basket analysis - quantity not considered since people decide quantity. We only recommend based on association.

18/1/2017

Dominant learning types (Supervised) → Recommender

Predictive
- Information based
- Probability based
- Ensemble (eg) tree bag, bagging, boosting
- NN learning (Nearest neighbor) (eg) KNN
- Objective based (eg) deep learning strategy, linear regression, Logistic regression, Neural network learning, SVM

Unsupervised learning (no target)
- Dimensionality reduction (eg) PCA
- clustering
- Association analysis
- Outlier detection

Hard to evaluate 'Unsupervised learning'.

Reinforcement learning
(eg) fully automated cars
- no target
- learn automatically by interacting with environment.

Two main types
1) Q-learning
2) Policy-based

Objective learners
- function (parameters) = ?
- subject to constraints

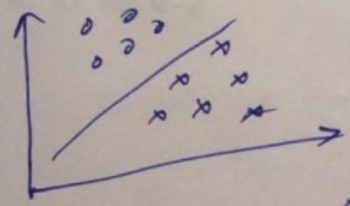
Linear regression

$y = w_1 x_1 + w_2 x_2$

$f(w_1, w_2)$ = squared loss = $\sum [y_i - f(w_i, x_i)]^2$

SVM & Logistic regression (for binary classification → can be extended for multi-class)

Line-based classification



If line equation is $f = w_0 + w_1 x_1 = 0$

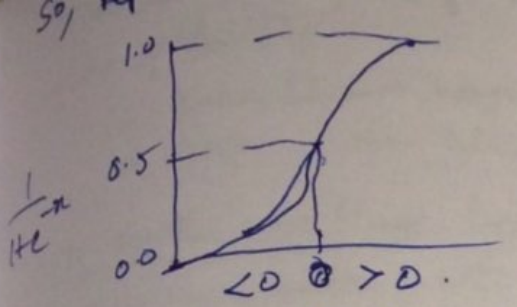
Then $f() = 1 \rightarrow$ class 1

$= 2 \rightarrow$ class 2

OK, but does not give probability that a point is class 1 or class 2.

Scanned by CamScanner

So,



Line still used to separate points in logistic regression.

But probability interpreted by using sigmoid (like eqn 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'.

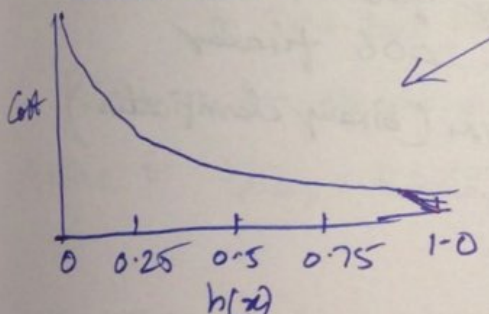
Ideal LR separation line:
- all class 1 points on one side
- all class 2 points on other side
- Identify loss function accordingly
- Constraint is L1, L2, or, L1+L2.

Objective fn : Log loss

$h(x)$ = Logistic fn = prediction

$$\text{Cost}(h(x), y) = \begin{cases} -\log(h(x)) & \text{if } y = 1 \\ -\log(1-h(x)) & \text{if } y = 0 \end{cases}$$

For $y=1$

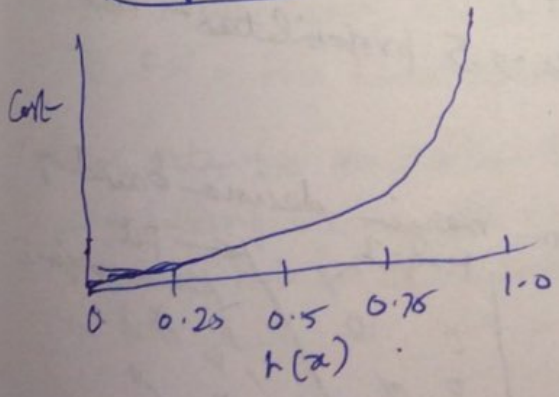


1) If $y=1$ and $h(x)=1$, Cost = 0
But for $h(x)=0$, cost = ∞.

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.

For $y=0$



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

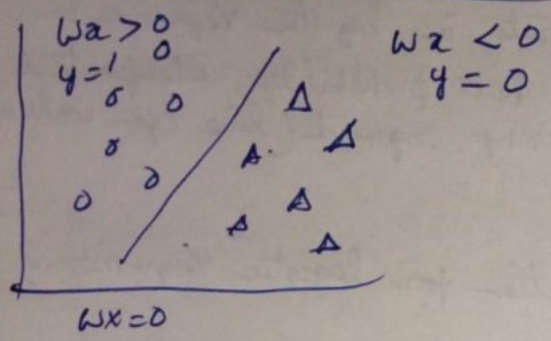
=

Cross entropy.

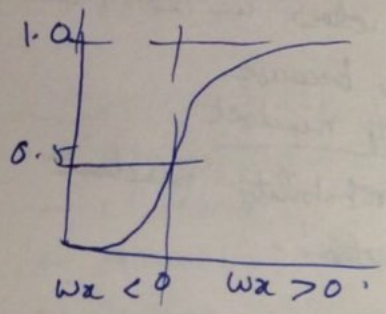
Scanned by CamScanner

19/1/2017

Linear discriminator – only linear boundaries



Line fit → How much error do each of the points contribute to the line?,



Logistic regression

$$f(w_0, w_1, \ldots, w_n) = -y \log \hat{y} - (1-y) \log (1-\hat{y})$$

$$\hat{y} = h(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)$$

Support vector machine (SVM)

- linear model
- Kernels + SVM = no – 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:

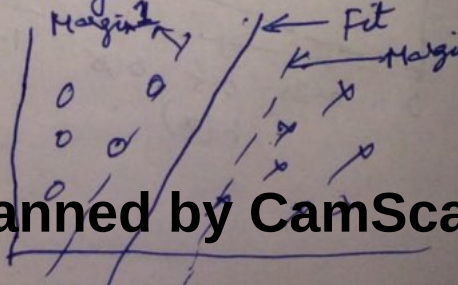
Build 5 models: C_1 vs $[C_2 - C_5]$

C_2 vs $[C_1, C_3, C_4, C_5]$

C_5 vs $[C_1 C_2 C_3 C_4]$.

Run 'test' against all 5 models → 5 probabilities – Max. probability determines class.

SVM good – Discover maximum margin decision boundary.



Scanned by CamScanner

SVM does better classification than LR. But it does not give 'probability'.

For probability → Logistic regression
is classification ← 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

Regression — RMSE
R^2

False +ve rate (FPR) $= \frac{FP}{TN+FP}$

TPR = Recall $= \frac{TP}{TP+FN}$

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

Mostly based on confusion matrix

	classified +ve	classified -ve
Original +ve	True +ve (TP)	False -ve (FN)
Original -ve	False +ve (FP)	True -ve (TN)

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{TP+TN}{\text{Total}}$$

Error = 1 − 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.
(eg) FN = Has heart disease, but we say 'no heart disease'.

Error rate too simplistic to select between models
- misclassification costs are not same (eg) FN worse than FP
- class imbalance datasets used (in 'train' data) ⇒ biased decision
- test data not representative

Scanned by CamScanner

1) Non-symmetric misclassification cost
- Matrix often difficult; different for diff circumstances
- Should we try a large # of matrices?

2) Imbalanced dataset (train)
3762 - No → 3731 (✓) + 32 (✗)
238 - yes → 229 (✗) + 9 (✓)
serious issue with just 'accuracy'

3) Test data not representative
- Does not present all cases, edge-cases, use-cases.

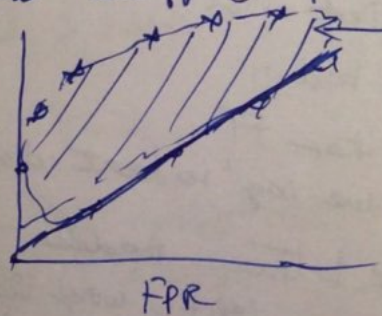
Alternative Metrics

1) ROC curve (Receiver operating characteristic)
- discovered for assessing the received 'signal quality' in electronics - curve between FPR & Recall measures (False +ve 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 classifiers } Requirements/
- applicable when classifier gives probability } Prerequisites

TPR (True +ve rate) = TP / Positives = Recall
FPR (False +ve rate) = FP / Negatives
Graph drawn by computing confusion matrix values using various cut-off (or probability threshold to classify as true or -ve)



AUC = Area under curve
- Found by plotting ROC curve

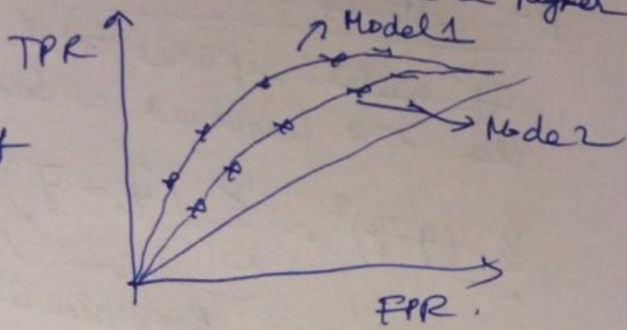
(*) log loss → Its an objective fn; not a metric.

AUC - more area is better
- also predicts valid TPR & FPR combination for use (TPR max & FPR min)

Scanned by CamScanner

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 ⇒ we will miss fewer true ~~data~~ points
* FPR - of those that are known as -ve, how many are predicted as +ve?
  - Higher FPR ⇒ will misclassify more -ve data points.

AUC = 0.6, 0.7, 0.8 ⇒ ? More is better.

≡ P(positive instance to have a higher score than

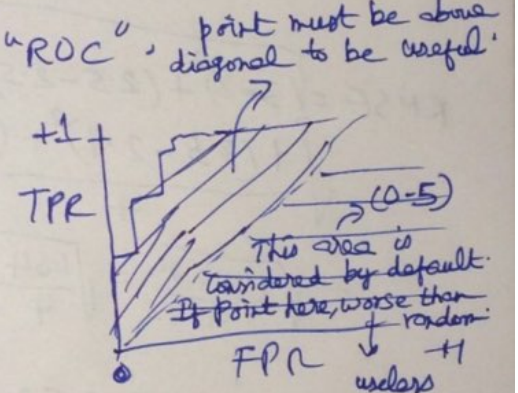
R library: ROCR, PROC. (PROC).

In train() method, use metric = "ROC".

ROC(.) → gives AUC value.

So, AUC value = [area below diagonal line (always) + area above line

= 0.5 + area ↓ ∈[0, 0.5].



(*) So, finally AUC ∈ [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 notion of -ve (and hence FPR) is not well defined
- F-score is harmonic mean.

$$F = \frac{2}{1/P + 1/R} = \frac{2PR}{R+P}$$

- harmonic mean
[it is taken for average of percentages]
↳ mathinsight.org

Regression metrics

- RMSE = square root of average squared residuals
- R^2

Scanned by CamScanner

R^2 - Capturing variance

If model captures total variance of target variable then it is considered as best model.

$$\sum_{i=1}^{n}(y-\bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2 + \sum_{i=1}^{n} e_i^2$$

Total sum of squares (SST) | Regression SS (SSR) | Error SS (SSE)

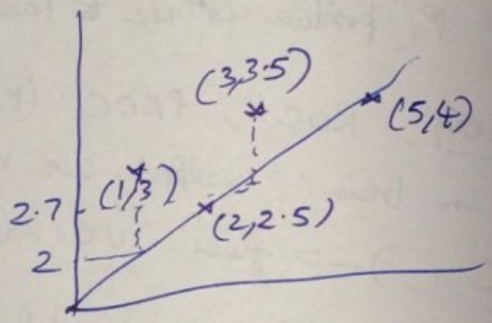
$$SST = SSR + SSE.$$

$SST = SSR \Rightarrow$ perfect fit.

$$R^2 = \frac{SSR}{SST} \qquad 0 < R^2 < 1.$$

If $R^2 = 1$, best fit.

$$= 1 - \frac{SSE}{SST}$$



$$RMSE = \sqrt{\frac{(3-2)^2 + (2.5-2.5)^2 + (3.5-2.7)^2 + (4-4)^2}{4}}$$

$$= \sqrt{\frac{1^2 + 0.8^2}{4}} = \sqrt{\frac{1.64}{4}} = \sqrt{0.41}$$

$Var(y) = Var(3, 2.5, 3.5, 4)$ $\quad \bar{y} = mean = \frac{13}{4} = 3.25.$

$$SST = (3-3.25)^2 + (2.5-3.25)^2 + (3.5-3.25)^2 + (4-3.25)^2.$$

$Var(\hat{y}) = Var(2, 2.5, 2.7, 4)$ w.r.t same $\bar{y} = 3.25$

$$SSR. = (2-3.25)^2 + (2.5-3.25)^2 + (2.7-3.25)^2 + (4-3.25)^2$$

$$R^2 = \frac{SSR}{SST}$$

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

Scanned by CamScanner