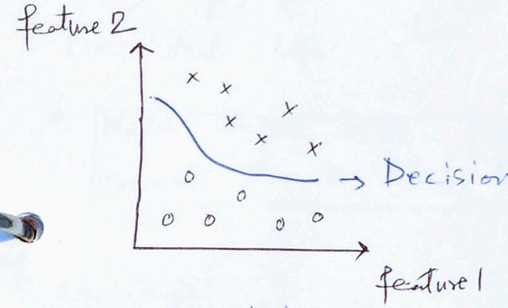
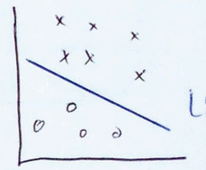
⇒ Acerous → animals w/ no horns.

⇒ Non-acerous → animals w/ horns.

feature
& attribute

* One of imp. things in ML is where to put (arrange) new data (or point) based on previous data.





* Naive Bayes is used to find decision surfaces.

* scikit-Learn:
  1- Create classifier.
  2- train it by fitting the features & Label data
  3- predict Label using test feature.

* Accuracy = $\frac{\#\text{ of points correctly classified}}{\text{all points in test set}}$

⇒ Bayes Rule:

Question:

$P(c) = 1\%$

Sensitivity $P(Pos|c) = 90\%$

Specificity $P(NEG|>c) = 90\%$

$P(c|Pos) = ?$

$P(Pos|>c) = 1 - P(NEG|>c)$
$= 0.1$

$P(>c) = 1 - P(c) = 0.99$

Sol. $P(c|Pos) = \frac{P(Pos|c)P(c)}{P(Pos)}$

$P(c|Pos) = \frac{P(Pos|c)P(c)}{P(Pos|c)P(c) + P(Pos|>c)P(>c)}$

$P(c|Pos) = \frac{0.9(0.01)}{0.9(0.01) + 0.1(0.99)}$

$\boxed{P(c|Pos) = 8\frac{1}{13}\%}$ OR 8.33%

8% chance of cancer given positive test

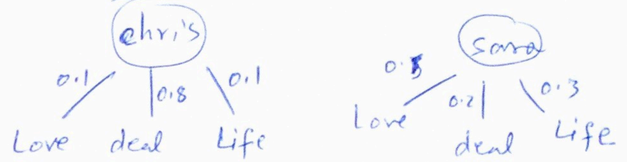
* return ↩

$r_{xy} \neq 1$

7 May (Night Shift) (2)

→ Text learning:

Naive Bayes Rule:



* love deal life are text in messages of both, difference of occurence

+ Naive Bayes tells from random email who is likely to send this email

msg: Love Life!

⇒ apriori:

P(chris) = 0.5

P(Sara) = 0.5

* Sara is most likely to have sent this message

⇒ ~~apriori~~ New msg:

LIFE Deal

→ More likely from Chris. As:

Chris = (0.1)(0.8)(0.5) = 0.04 (Life deal apriori)

Sara = (0.3)(0.2)(0.5) = 0.03

⇒ posterior:

$$P(\text{chris} \mid \text{"Life Deal"}) = \frac{P(\text{"Life Deal"} \mid \text{chris})\,P(\text{chris})}{P(\text{"Life Deal"} \mid \text{chris})\,P(\text{chris}) + P(\text{"Life Deal"} \mid \text{>chris})\,P(\text{>chris})}$$

$$= \frac{(0.1)(0.8)(0.5)}{(0.1)(0.8)(0.5) + (0.3)(0.2)(0.5)} = 0.57$$

$$P(\text{Sara} \mid \text{"Life Deal"}) = \frac{P(\text{"L D"} \mid \text{Sara})\,P(\text{Sara})}{P(\text{"L D"} \mid \text{Sara})\,P(\text{Sara}) + P(\text{"L D"} \mid \text{>Sara})\,P(\text{>Sara})}$$

$$= \frac{(0.3)(0.2)(0.5)}{(0.3)(0.2)(0.5) + (0.1)(0.8)(0.5)} = 0.428$$

$$\frac{0.1 \times 0.8 \times 0.5}{0.1 \times 0.8 \times 0.5 + 0.3 \times 0.2 \times 0.5}$$

0.07

OR $\frac{0.04}{0.04+0.03}$

OR $\frac{0.03}{0.03+0.04}$

* return q

→ New Msg:

method-1: "Love Deal"

$P(\text{chris}, \text{"Love Deal"}) = (0.1)(0.8)(0.5) = 0.04$

$P(\text{Sara}, \text{"Love Deal"}) = (0.5)(0.2)(0.5) = 0.05$

} joint

And:

$P(\text{chris} \mid \text{"Love Deal"}) = \frac{0.04}{0.04+0.05} = 0.444$

$P(\text{Sara} \mid \text{"Love Deal"}) = \frac{0.05}{0.04+0.05} = 0.555$

} posterior

Method-2:

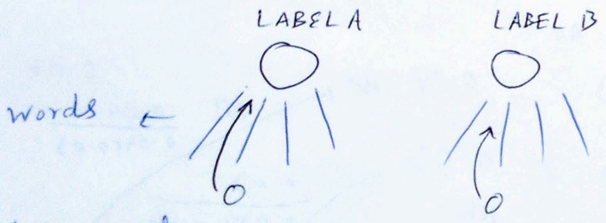
$$P(\text{chris} \mid \text{"Love Deal"}) = \frac{P(\text{"love Deal"}_{|\text{chris}})\, P(\text{chris})}{P(\text{"Love Deal"}_{|\text{chris}})\, P(\text{chris}) + P(\text{"love Deal"}_{|>\text{chris}})\, P(>\text{chris})}$$

$$= \frac{(0.1)(0.8)(0.5)}{(0.1)(0.8)(0.5) + (0.5)(0.2)(0.5)} = 0.444$$

(or P(Sara))

* Actually ~~Method~~ Method 1 & Method 2 both are same!

→ Why Naive Bayes is Naive ?



They are hidden, always in supervised Learning

* Every word gives us evidence either its person A or person B
* We multiply probabilities of all the words appearing in msg.

A ..... Ratio

B .....

Prior probabilities

+ Called Naive, b/c it ignores the order of the words in message

→ Where Naive Bayes Breaks:

In case of word order, so it wont differentiate b/w "Bull Chicago" & "Chicago Bull" although both have totally diff. meanings & "Chicago Bull" is actually 1 word not 2.

→ Project:

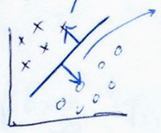
→ Based on text of email, classify which of two persons.

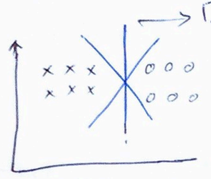
→ Naive Bayes is very good for text classification as it treats each word as a feature & where features are independent.

* return 9

⇒ Svm [Support vector machine] → Invented by Russian

* It is a supervised learning method like Naive Bayes Rule.
* It takes data & creates/forms boundary b/w 2 sets/classes.

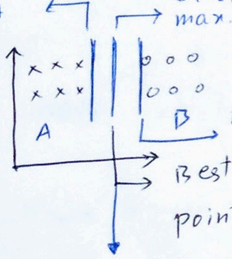




* What makes a good seperator?

Margin → max. distance to the nearest point from either of the two classes.

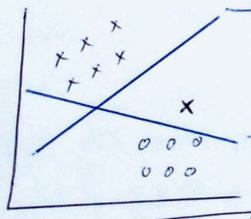
min. dist. from Class A ↔ max. dist. from Classes A & B



min. dist. from Class B

Best as it maximizes the margin to nearest point from either (both) of classes

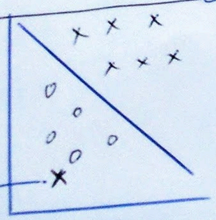
* Max. margin on either side reduces classification error.



→ provides max. margin but doesn't classify data. — Contains classification error.

→ Best (It seperates data into 2 classes & also provides margin)

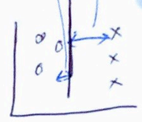
So first correct classification is performed & only then the margins are maximized.

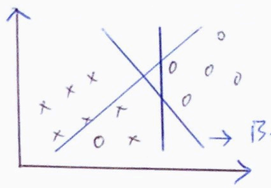


* SVM is good enough to classify, maximize margins as well as neglect outliers.

So doesn't maximize margin

{small large}





→ Best (*classifies correctly *maximizes margins *ignores outliers)

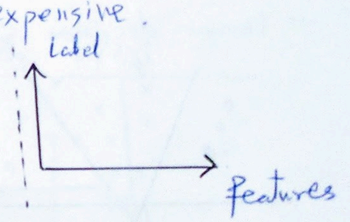
Adv. :

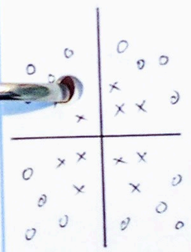
* SVM is efficient in high dimensional spaces.
* Effective when dimensions are more than number of samples.
* Diff. Kernel functions can be used for decision function.

Disadv. :

* If # of features is much greater than # of samples, it performs poorly — Hughes Phenomenon — Curse of Dimensions
* doesn't provide probability estimates. 5-fold cross validation is used for probability estimates which is expensive.

* Margins make sure that there is as much seperation among classes as possible.





* Linear SVM will not work on this kind of data.
* There is no linear Hyperplane (seperator) b/w two classes.

* input for training → features & labels
* input for predict → test features & output → predicted labels
* accuracy score → compare test labels with predicted labels

* scikit-learn ⑨
* Machine learning library.
* classific, clustering, Regression
* Naive Bayes, SVM, Random forest, KMeans ...
* Google Summer of Code
* VI → Virtual Instruments in LabView

* return

x → train-features.
y → train-Labels.
y-predicted → predicted Labels

x
y
x^2+y^2 → y-predicted

↳ non-negative (dist. from origin)

kernel trick

x has less dist. from origin
o has ~~large~~ dist. from origin

So:

x
y
→ Labels

xy plane

z
x
y
z
↳ x^2+y^2
→ Labels

xz plane

This data is non linearly seperable

* Line in xz plane corresponds to circle in xy plane

xy → here its non Linear decision Surface

xz → Linear decision Surface

Here x, y, $z=|x|$ → Linear surface of Labels

Non Linear Surface

adding new non Linear feature $|x|$

→ linear surface

if $|x|$ is used, only +ve x-axis contains all value

Kernel trick

⇒ Kernel Trick

Low dimensional

x, y ~ Kernels

non seperable

Non Linear Seperation ←

high dimensional

x1 x2 x3 x4 x5

Seperable

Linear Seperable Solution

* We use Kernel to go from low dimension to high dimensional & find Linear Seperation, using info of Linear Seperation we can find nonLinear seperation with original data

* There are several SVMs in scikit-learn, we are using SVC (Supp. vec. classifier)

* default is rbf

SVC supports these kernels:
1- Linear
2- poly
3- rbf
4- sigmoid
5- precomputed

* parameters → passed when creating classifier before fit

params for SVM
→ Kernel
→ C
→ Gamma

kernel → linear
gamma → 1000

kernel → rbf
gamma → 1000

controls tradeoff b/w smooth decision boundary & classifying training points correctly.

* Smaller C → smooth.
* Larger C → correctly classifying but Less Smooth.

Smaller C

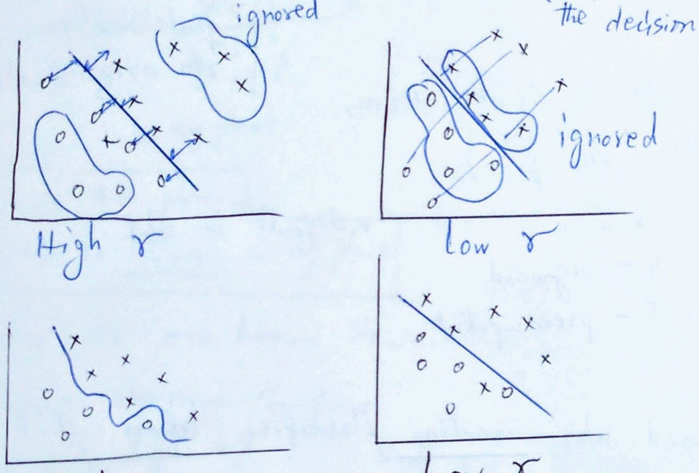
Larger C

* return y

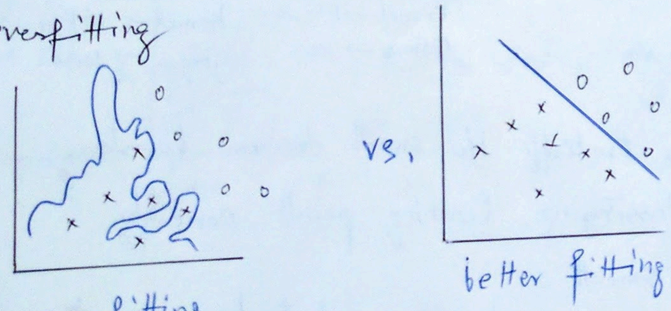
⇒ Gamma in SVM :

* defines how far the influence of a single training reaches

* Low — far reach
* high — close reach (decision boundary will be dependent on just the points closest to the decision boundary.



⇒ Overfitting



Overfitting

* when classifier takes the data too literally.
* classifier is trained on same kind of training data again & again w/o variation. So its now overfit for that data & not generalized

Kernel, gamma & C

all controls the overfitting of the data.

C → smoothness vs. best boundary

γ → higher vs. lower influence of nearby points

Kernel → Linear, poly ....

Adv. of SVM: ⑥

Disadv. of SVM:

* Noisy data, overlapping data → doesn't work good.
* With very large dataset & lots of feature, SVM can be slow. Naive Bayes can be a much better classifier for these situation as feature as independent & order doesn't matter in Naive Bayes.
* Need to count "independent evidence" when classes are too overlapping ie noisy data

---

* The more training data we have, the more is accuracy but slower is the data fitting — most of the time is consumed in training the classifier — Prediction takes very smaller time.
* In realtime usage like voice recognition & transactions blockage etc, we prefer speed over accuracy.
* naive bayes is usually way faster than SVM for large data sets.

* Using combination of high C & rbf kernel can give very good accuracy on smaller data set but on the expense of very complicated decision surface & the classifier will be much less generalized which is very bad.
  ↳ But if increase the data set size, classifier will be slow in training. — so right balance needs to be achieved.

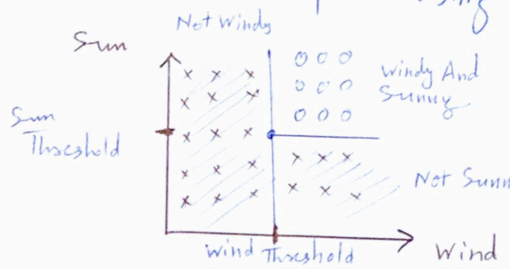
svm.predict(test_~~labels~~ feature)

* It provides us predicted class fo each data point of test_~~labels~~ features.

* gridCV → sklearn tool that finds optimal parameters.

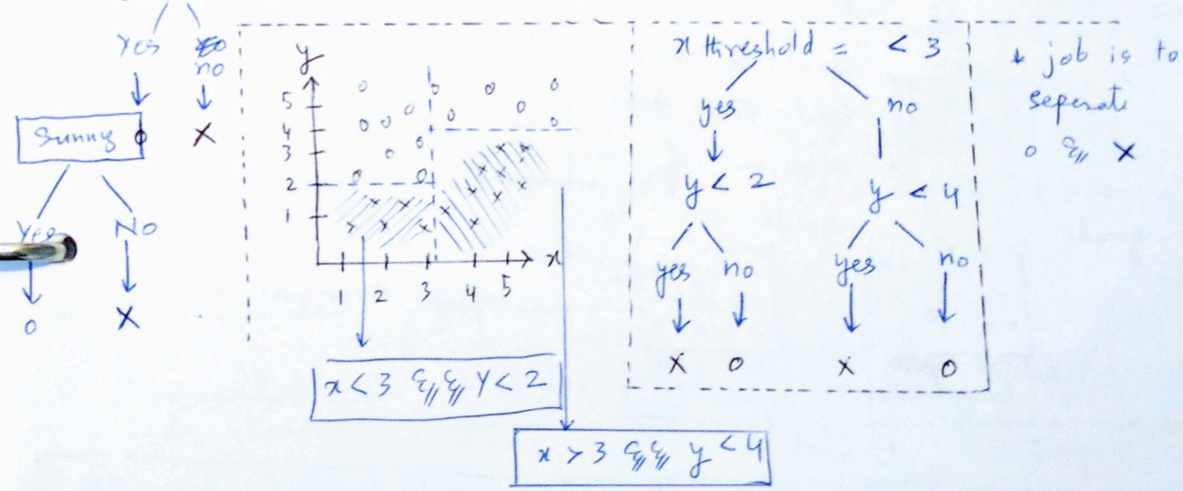
→ Decision Trees:

* In svm, we were able to change from Linear to Non Linear Decision Surface using Kernel Tricks with the help of Margins



Windsurfing → you enjoy Surfing when its Sunny & windy

* ooo → Suitable condition for wind Surfing
* Its not linearly seperable
* Decision Tree allowes to ask Multiple linear Question one after another.
* Question 1:
  Is it windy (with the threshold)
* So we classify data using data structure called Tree.



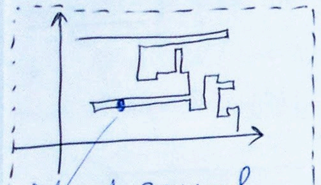
↓ job is to seperate o & x

* Cost of running tree is Logarithmic in the number of data point in training.
* Uses white box model → any observable situation can be explained by boolean Logic unlike artificial neural networkes.
* Can have overfit by creating over-complex decision trees.

* return q

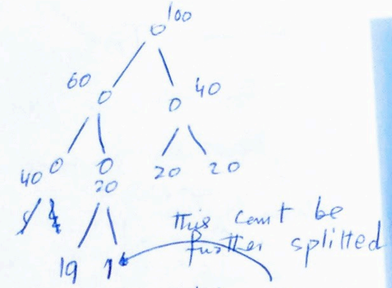
⇒ Decision Tree Parameters.

* min_sample_split → * min. value of subsample node allowed to be further splitted.
  * So 4 means node with value 4 won't be splitted further.
* default is 2.



3/4 of several nested / deep decision trees with overfit

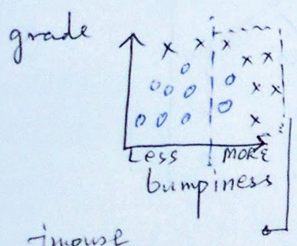
if min_sample_split = 2
Then it'll create this whole decision surface just to grab this point

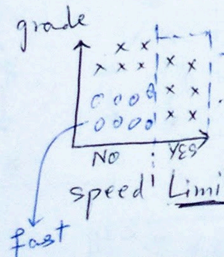


* min_sample_split = 2

⇒ Data Impurity & Entropy:

* entropy → Controls how a DT decides where to split the data.
  measure of impurity





* So decision tree tries to make subset that are as pure as possible & thats how DT actually makes its decision

$$\text{entropy} = \sum_{i} -P_i \log(p_i)$$

- i: Sum over all classes available
- p_i: Fraction of examples in a given class.
- 2 class; can be 10, e ....

$0 \leq \text{entropy} \leq 1$ (pure → impure)

Cases for Entropy:

* all examples are same class:
  entropy = 0 → pure
* examples are evenly split b/w classes:
  entropy = 1.0 → impure

* entropy ∝ 1 / organized data

Lower entropy → More organized data
Higher entropy → Less organized data

grade	bumpiness	speed limit?	speed
steep	bumpy	yes	slow
steep	smooth	yes	slow
flat	bumpy	no	fast
steep	smooth	no	fast

* $p_{slow} = 0.5$ → fraction of slow
* $p_{fast} = 0.5$ → fraction of fast

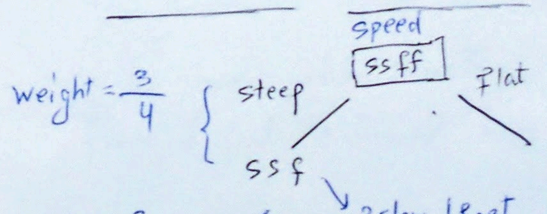
So
entropy = ~~0.5 + 0.5~~ 1.0
$= -0.5 \log_2(0.5) + -0.5 \log_2(0.5) \Rightarrow 1.0$
(impure state)

* Most impure state is when samples are evenly split b/w 2 classes.
* entropy of first (start) node (parent) (max impure) 1.0

⇒ Information Gain: (IG)

information gain = entropy(parent) − weighted average of entropy(children)

* decision tree maximizes information gain



$p_{slow} = \frac{2}{3}$
$p_{fast} = 1/3$
$\text{entropy} = -\left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right) = 0.389 + 0.528 = 0.9183$ → less organized (more impure)

* return q

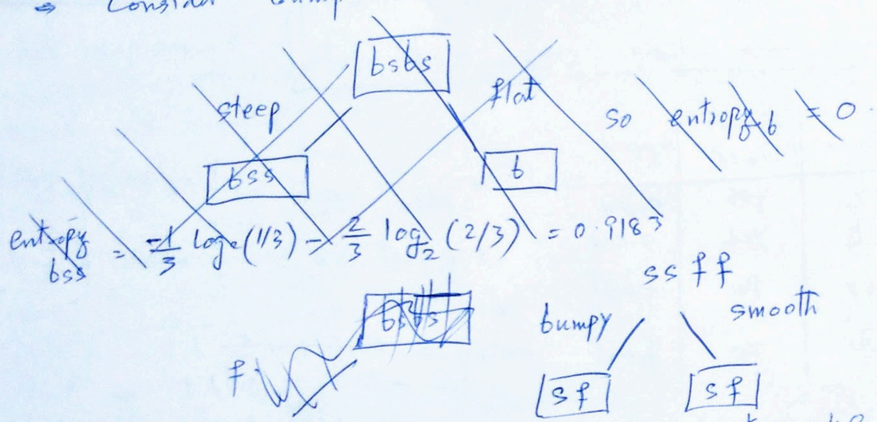
So

$IG = (\text{parent entropy}) - [\text{weight Average}](\text{children entropy})$

$IG = 1.0 - \left[\frac{3}{4}(0.9183) + \frac{1}{4}(0)\right]$

$IG = 0.311275$

⇒ Consider bumpiness



And

$I.G = 1.0 - \left[\frac{2}{4}(1.0) + \frac{2}{4}(1.0)\right]$

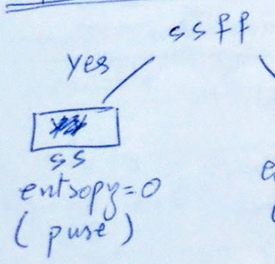
↳ weighted Sum of entropy children

entropy parent ✓

$I.G = 0$

we don't gain any information by splitting data in bumpiness for this example.

⇒ Consider Speed Limit:



So:

$I.G = 1.0 - \left[\frac{2}{4}(0) + \frac{2}{4}(0)\right]$

$I.G = 1.0$ → we get max. info by splitting data based on Speed Limit

* in scikit learn,
  in DT, criterion param → gini / entropy

(no Info) Worst $0 \leq$ Inform. Gain ≤ 1 Best (full Info)

(pure) Best $0 \leq$ entropy ≤ 1 Worst (Impure)

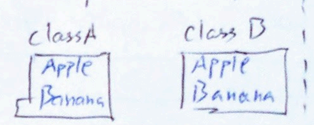
Information Gain $\propto \frac{1}{\text{entropy}}$

* for impurity classification you get no information
* for pure classification, you get max. information

⇒ pure classification

* max Info
* low entropy

⇒ Impure classif.



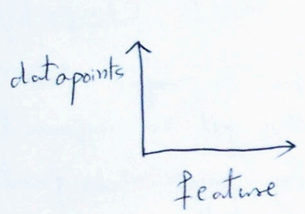
* no Info
* max entropy

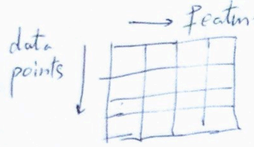
⇒ Bias-Variance Dilemma:

* Bias classifier is the one on which there is no effect of data. So whichever way you train it, it'll never do things differently.
* Variance classifier is one which is very sensitive of the data, it can only replicate the stuff it has seen before. It'll react poorly for new data as it has very little bias to generalize itself to new stuff.
* So we need something in middle so it can learn from (good variance) but could also be generalized (good bias).

⇒ Tradeoffs with Decision Tree.

* Can be easily interpretted, whitebox — addressable situations can be explained using boolean logic
* Prone to overfitting b/c of overly-complex decision tree with lots of features.
* Decision Tree classifer can be used to new classifier (ensemble ~~Unsampled~~ classifier)





ensemble classifier
several small & weak classifiers are combined together to form a single robust classifier.

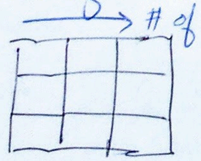
# of feature ∝ complex fit
assuming sample size is constant

⇒ Adaboost.

* Adaptive boost + meta algorithm.
* ensemble → combines (weighted sum) several weak learners (decision trees) to form a robust classifier.
* adaptive → subsequent weak learners are favored over those instances misclassified by previous classifiers.
* Hughes phenomenon → with fixed number of training samples, prediction power ∝ 1/dimensionality

Curse of Dimensionality

→ means if you have lots of features but very small training sample, then prediction is weak as features are just increasing the dimension.



→ So, rows & cols should both grow to have good prediction.

* pruning → process of removing poorly performing weak classifiers

* Bias ∝ 1/Data Dependence ; Variance ∝ Data Dependence

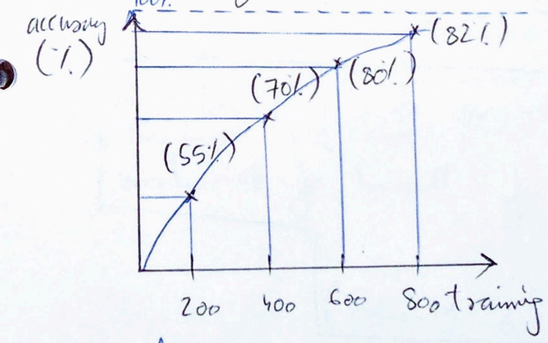
⇒ Enron fraud:

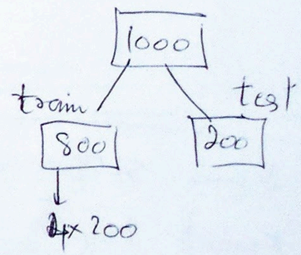
* Enron was energy trading company with 101 bn $ revenue.
* went bankrupt b/c of clever economic & financial practices.
* 10,000 of emails (real) were released as corpus & we will use these emails to train our classifiers & get insight.

⇒ Datasets & Questions:

* Person of Interest (POI): → indicted
  → settle w/o admitting guilty.
  → testified for immunity.

* Accuracy ⇔ Training set size





* Accuracy ∝ Training set size
* But we shouldn't have lots of features b/c it'll create Hughes phenomenon that too much features w/o increasing training set so just increasing dimensions & reducing prediction accuracy.

** More Data > fine tuned Algorithm

* return &

Types of Data:

Numerical – numerical values (feature → number)

Categorical – limited # of discrete value (Feature → category)

time series – temporal value (date, timestamp)
↳ features

text – words → feature
↳ we convert words feature to number using Bag of words formula etc.

* salary → numerical feature
* job title → categorical feature
* timestamp on email → time series feature
* email contents → text feature
* # of emails sent → numerical feature
* to/from field → can be categorical feature but fits better as text feature. where text is email addresses. text wrangling tools can be used.

* Jeffrey Skilling → Ceo of Enron during fraud time
* Kenneth Lay → Chairman of Board of Directors.
* Andrew Fastow → CFO
* Unfilled feature → NaN (Not a Number)

# of people ↓ → features

POI	salary	expenses	...

salary	to_messages	deferral_payments	total_payments	exercised_stock_options
bonus	restricted_stock	shared_receipt with POI	restricted_stock deferred	total_stock_value
expenses	loan_advances	from_messages	other	from_this_person_to_poi
poi	director_fees	deferred_income	long_term incentive	email
from_poi_to_this_person				

↑ features

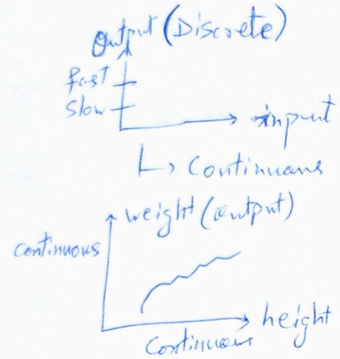
* most non-POIs come from financial statements with complete data.
* POIs were added from Enron data with very incomplete data
* classifier learner can be tricked into thinking that POIs are those which have several NaN fields — which is obviously not the case.
* So we should be very careful when introducing new feature while using data from several sources
* 1 solution is to use features common to all sources.

↳ feature selection from Multiple sources.

* return q

⇒ Regression.

* Continuous Supervised learning:
  - ↳ output.
  - * input is normally continuous but o/p can be discrete (binary) or continuous.

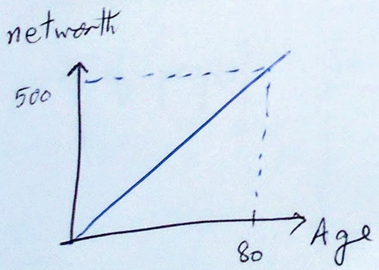


* Age → continuous
* weather → discrete (Sunny or raining)
* But most things we regard as discrete could be modelled as continuous. e.g. wealth could mean the amount of light and this is continuous.
* person wrote email → discrete
* even though we can have thousands of author but they are not continuous in time So #56 & #59 has no relation in time.
* phone number → discrete
* income → continuous

Regression → predicting relation among data & finding best fit line.

Without order → discrete	With order → continuous

* Discrete	* Continuous
fast/slow	speed in mph



net worth = m (Age) + b

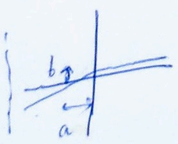
$m = \frac{500-0}{80-0} = 6.25$.

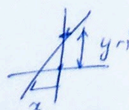
$b = 0$

So:

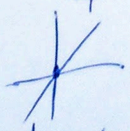
networth = 6.25 (Age)

y-intercept → height when line intersects y axis





x intercept



b = 0 (y inter.

a = 0 (x inter.

(13)

Target (out) = Slope (Input) + y-intercept

(12)

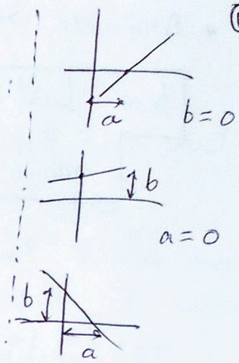
* Regression → predict output from input using:

$$y = mx + b$$

↳ Linear Regression

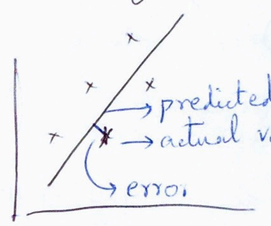
* Create
* fit
* predict

* We need to find m & b



⇒ Linear Regression Errors:

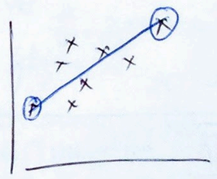
↳ actual − predicted



error → −ve means actual point is below predicted point.

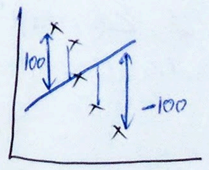
error → +ve means actual point is above predicted point.

* error on first & last data point:



Not a good fit

* Σ error on all data points



* Not a good fit
* both errors cancel each other out & result is straight line.

Σ error = 0

even its not fitting at all.

* refresh

+ Minimize Sum of Square Error — $\Sigma(error)^2$:

the best regression is:
minimizes $\sum_{\text{all training points}}(\text{actual} - \text{prediction})^2$

prediction from regression

training points

$y = (m)x + (b)$

*Methods to minimize $\Sigma(actual - prediction)^2$ OR SSE

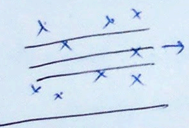
① Ordinary least Square (OLS):
→ used in Sklearn Regression.

② Gradient Desient:

$y_i = y_i + \alpha(x_i - y_i)$

$y_i = y_i + \beta(y_{i-1} + y_{i+1} - 2(y_i))$

$y_i \rightarrow$ new
$x_i \rightarrow$ old
$\alpha \rightarrow$ data smooth
$\beta \rightarrow$ weight smooth

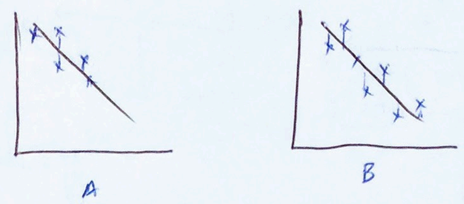

→ best regression result
↳ in terms of SVM, it has Large margin on both side

There can be multiple lines that minimizes Σ Error but only one Line will minimize $\Sigma(Error)^2$

+ Another reason for using SSE i.e $\Sigma(error)^2$ is that its much easier to code mathematically.

⇒ SSE is not perfect:

larger SSE → Worst fit



+ These both have about the same Fit
+ But just B has more data points, it will have greater SSE.

In SSE, as you add more data, the SSE usually goes up even though the fit is not necessarily bad.

#SSE gives relation of test labels & predicted labels.

r^2 gives relation b/w input & output of test data

⇒ r^2 of a regression:

+ how much Δoutput is explained by Δinput

$0 < r^2 < 1$

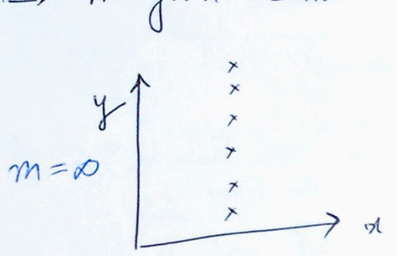
not doing good job of capturing trend in data

good description / relation b/w input & output

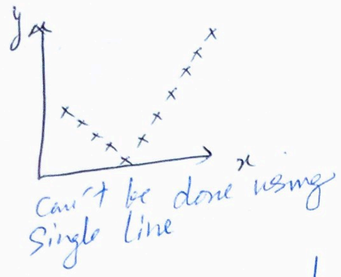
+ r^2 can be in -ve when finding fit or regression for significantly different data set.
+ r^2 is independent of # of training points.

* return q

⇒ A good Linear Regression Data

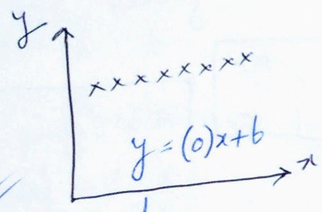


✗ no good regression

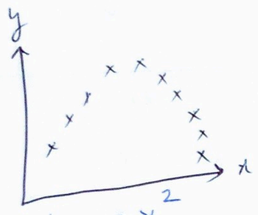


Can't be done using Single line

$y = mx + b$



✓ good

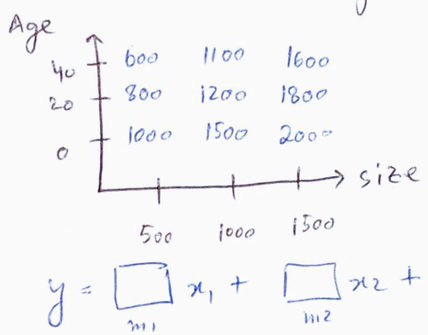


$y = ax^2$ parabola

⇒ Comparing Classification & Regression:

property	Supervised Classification	Regression
output type	Discrete	Continuous (Numbers)
what are we find	Decision Boundary	Best line of Line
evaluation	accuracy score	* Sum of Squred Error * R Square * gradient descent

⇒ Multi Variate Regression:



$y = \square x_1 + \square x_2 + \square$

m_1 m_2 m_3

Sol.

$$m_1 = \frac{1500 - 1000}{1000 - 500} = \boxed{1}$$

$$m_2 = \frac{1000 - 800}{0 - 20} = \boxed{-10}$$

And:

$$1000 = 1(500) + (-10)(0) + m_3$$

$$\boxed{m_3 = 500}$$

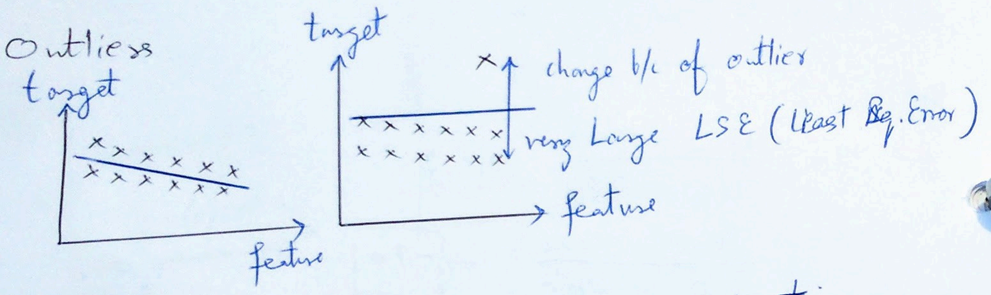
height, weight, educ; → you

* return q

⇒ * If outliers lie in training data, they change slope & intercepts.
* If outliers lie in testing data, they affect negatively the regression score.

* train → for fit
* test → for predict

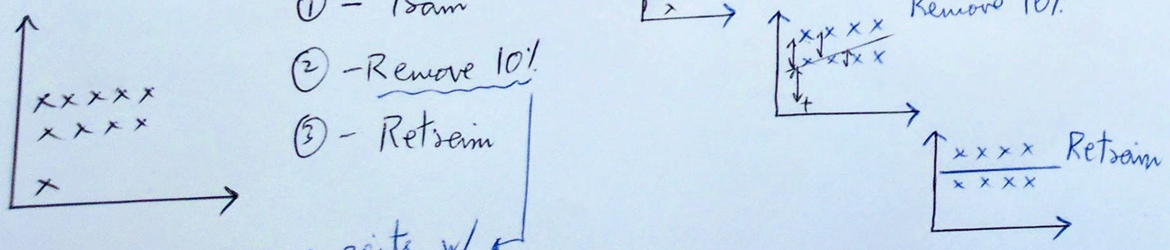
⇒ Outliers



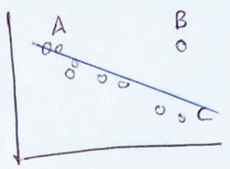
* Outliers can be caused by → Sensor Malfunction
→ Data Entry Error
→ Special Event

* In robotics & SDC, we usually ignore outliers but in other fields like fraud detection, outliers are very highly looked for.

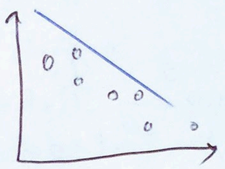
⇒ Outlier Removal:



⇒ Residual Errors



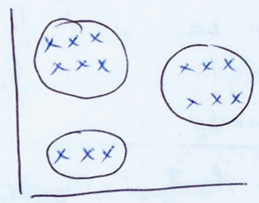
* D will have highest Residual Error after fit.



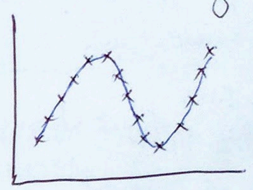
* Steeper now, as we have no outlier

⇒ Unsupervised Learning:

* Structuring data without Labels.
* with labeled data, we dont generally need Unsupervised



Clustering



Dimension Reduction

* we didn't have classes but after looking at the data we can form the clusters & say them classes.

K-Means:

* Grouping a set of objects in clusters so that objects in same cluster are similar to each other than other clusters.

* Kmeans uses Centroid model for Clustering.

* Kmeans (for Clustering) is different from K Nearest Neighbors (for classification)

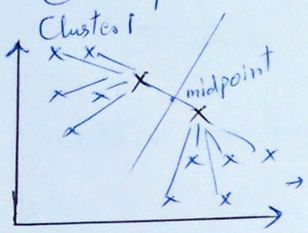
$K \leq n$ — k → # of clusters, n → # of data samples

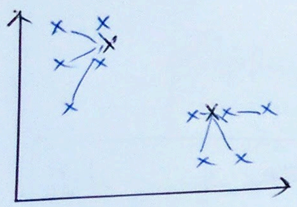
* retrain

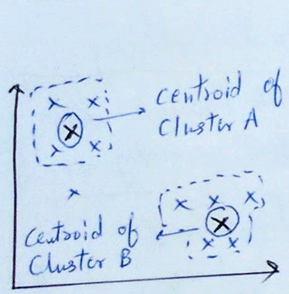
Algorithm for KMeans:

① – Assignment.

② – Optimize.







initial centroid:

* Randomly place 2 points & form the line from their midpt.
* Find difference b/w new & old centroids. centroids
* Change position of cluster point – rubber bands ties to be as short as possible.
* iterative ~~&~~ optimization.
* This is correct Centroid for both clusters.

* Initial random positions are very important & might end up with totally diffrent clustering.

So:

KMeans input → n samples, how many clusters
output → K clusters

- Minimizes WCSS (Within Cluster Sum of Square)
- in high dimension, Euclid distance becomes inflated due to Curse of Dimensions & Can be Solved by running reduction algorithms e.g PCA before clustering.
- a.k.a LLoyd's algorithm.

Machine Learning:

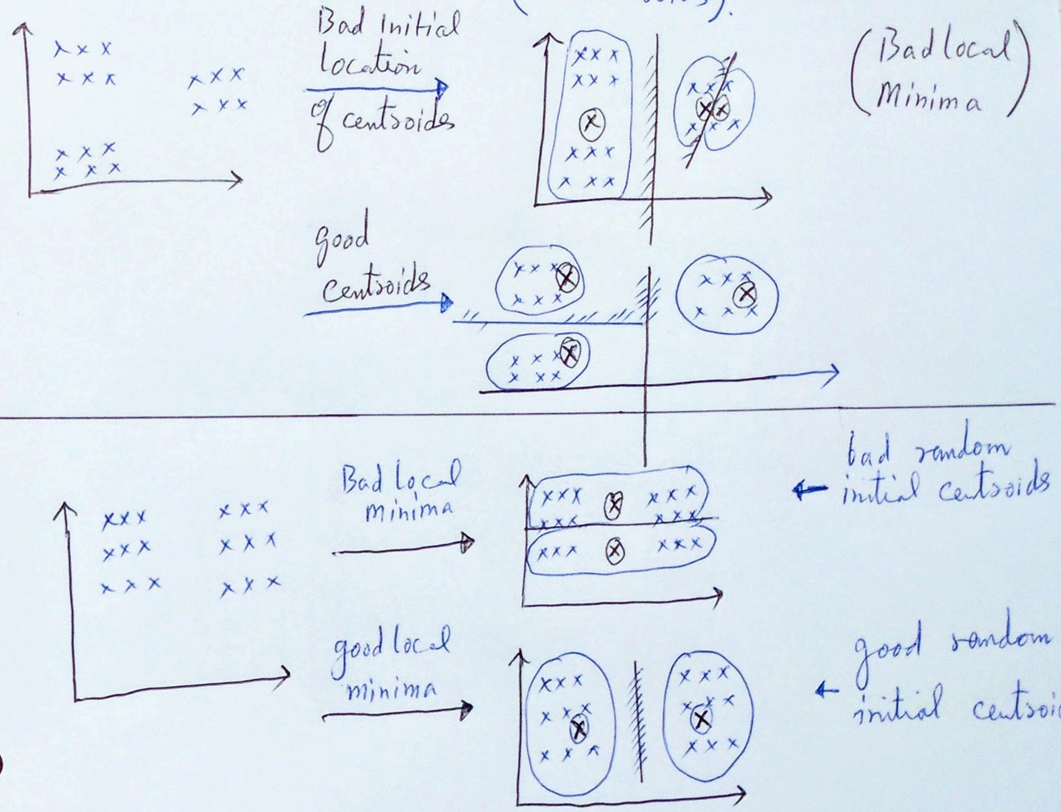
* Algorithms which learn from data.
* By making a model based on input data (training features & Labels) & predicting output. with accuracy tested against test data.

* Random forest & Adaptive Boost corrects the overfitting problem in Decision Tree (makes sometimes overComplex trees).

* iteration (optimize) steps keeps running until Centroid doesn't move anymore.

⇒ KMeans Challenges: (16)

* Uses Hill Climbing technique for local minima. So the clustering depends on when you put the initial cluster centre (or centroids).



* refresh q