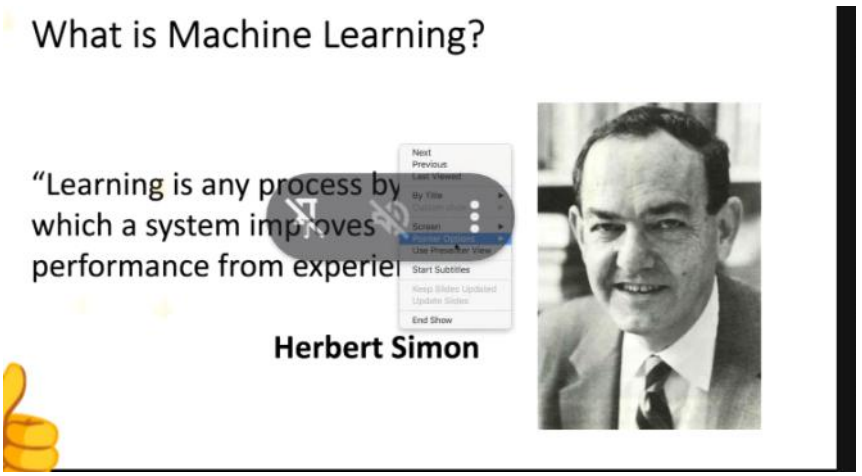
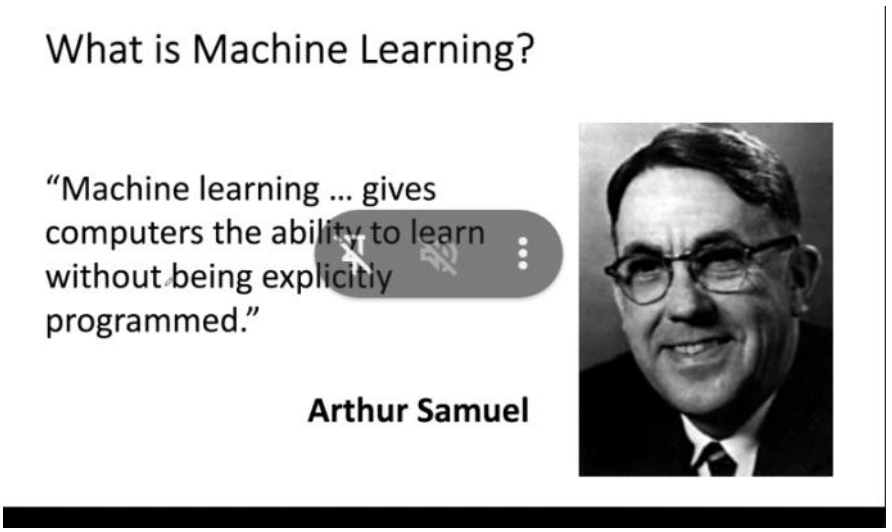
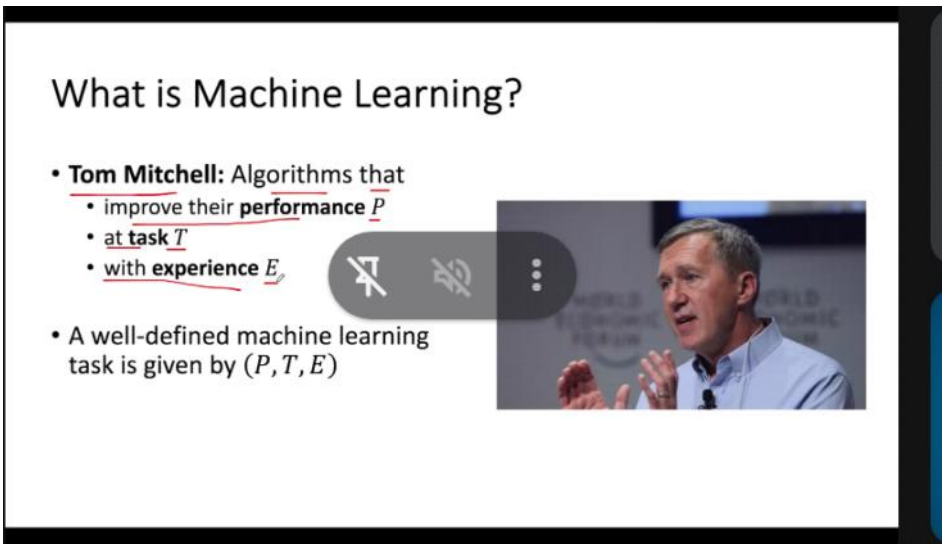
Machine Learning - 23/08

23 August 2025 14:08







Example: Prediction

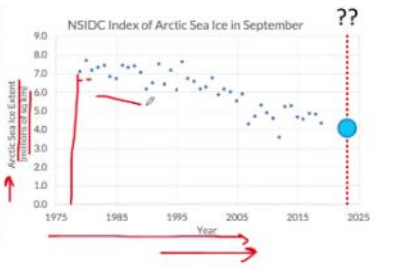
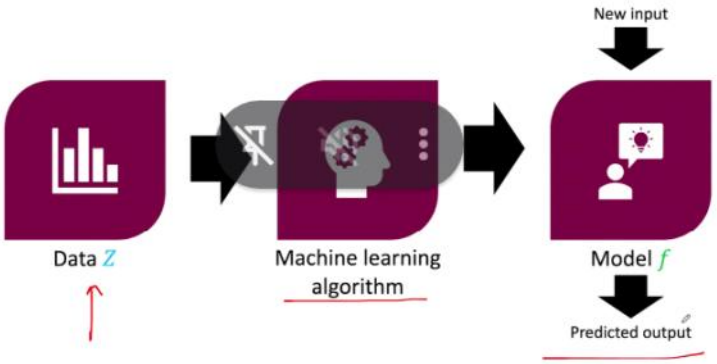
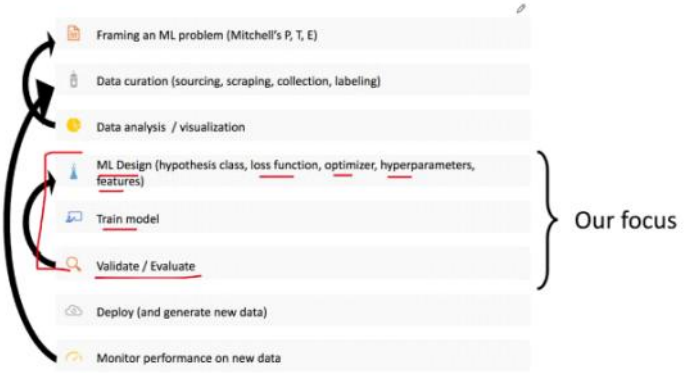


Image: https://www.flickr.com/photos/gsfc/5937599688/
Data from https://nsidc.org/arcticseaicenews/sea-ice-tools/

Machine Learning for Prediction



Machine Learning Workflow



Supervised Learning

- Given $(x_1, y_1), \ldots, (x_n, y_n)$, learn a function that predicts y given x
- **Classification:** Labels y are categories

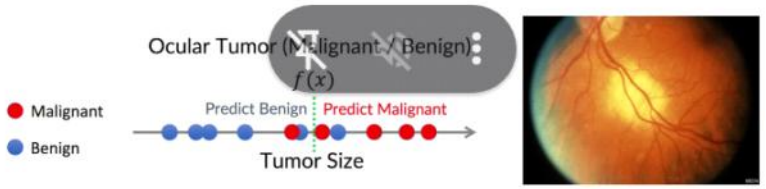


Image: https://eyecancer.com/uncategorized/choroidal-metastasis-test/

Supervised Learning

- Given $(x_1, y_1), \ldots, (x_n, y_n)$, learn a function that predicts y given x
- **Classification:** Labels y are categories

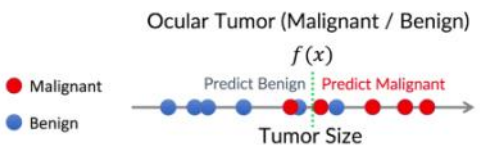
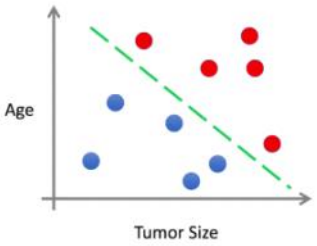


Image: https://eyecancer.com/uncategorized/choroidal-metastasis-test/

Supervised Learning

- Given $(x_1, y_1), \ldots, (x_n, y_n)$, learn a function that predicts y given x
- Inputs x can be multi-dimensional

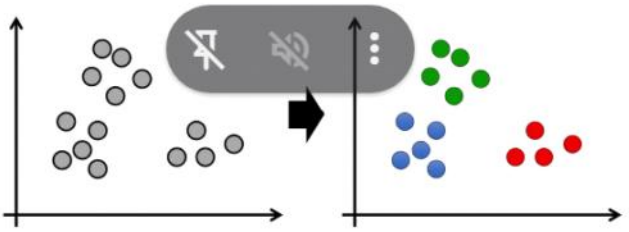


- Patient age
- Clump thickness
- Tumor Color
- Cell type
- ...

Image: https://eyecancer.com/uncategorized/choroidal-metastasis-test/

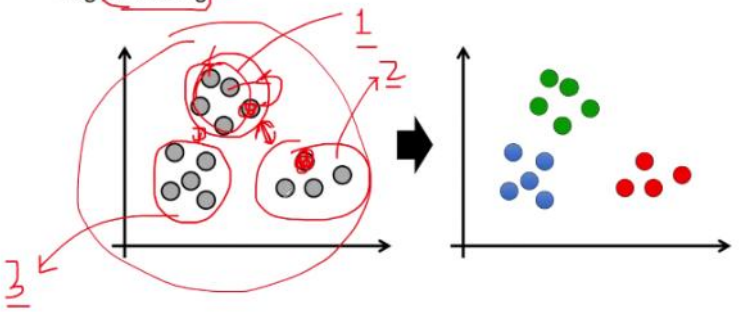
Unsupervised Learning

- Given $x_1, \ldots, x_n$ (no labels), output hidden structure in x's
  - E.g., clustering



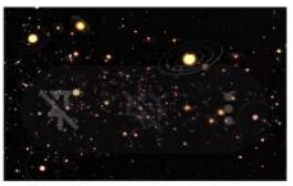
Unsupervised Learning

- Given $x_1, \ldots, x_n$ (no labels), output hidden structure in x's
  - E.g., clustering



Unsupervised Learning

Find Subgroups in Social Networks



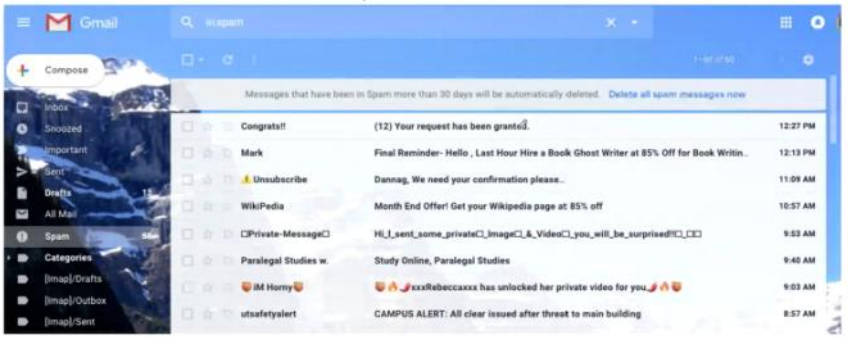
Identify Types of Exoplanets

Visualize Data

Image Credits:
https://medium.com/graph-commons/finding-organic-clusters-in-your-complex-data-networks-5c27e1d4645d
https://arxiv.org/pdf/1703.08893.pdf
https://en.wikipedia.org/wiki/Exoplanet

Problems Solved by Machine Learning Today

Spam Detection



Problems Solved by Machine Learning Today

Robotics

(Self-driving Vehicles) (Medical Surgery) (Manufacturing)

Problems Solved by Machine Learning Today

Recommendation Systems

Problems Solved by Machine Learning Today

Computer Vision Systems

e.g., self-driving vehicle on Mars

e.g., recognizing people

e.g., shopping without a cashier

Problems Solved by Machine Learning Today

Home Virtual Assistants

e.g., Amazon's Echo with Alexa

e.g., Google Home

Creating Images & Text

https://thispersondoesnotexist.com/

SYSTEM PROMPT (HUMAN-WRITTEN)

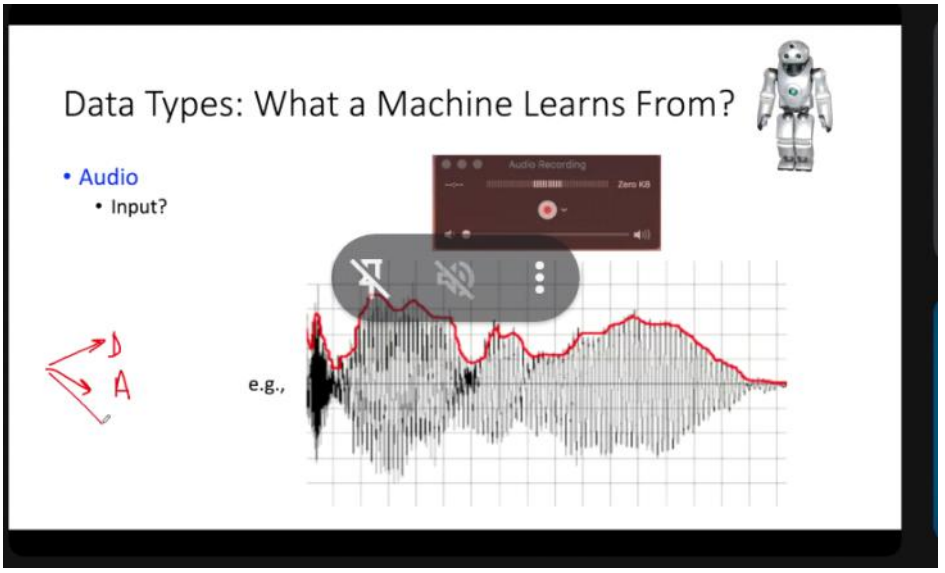
Recycling is good for the world.

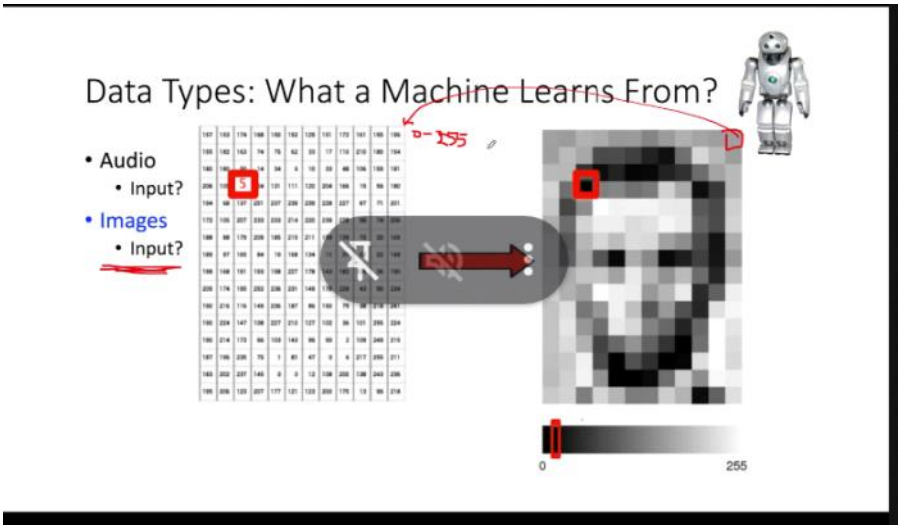
NO! YOU COULD NOT BE MORE WRONG!!

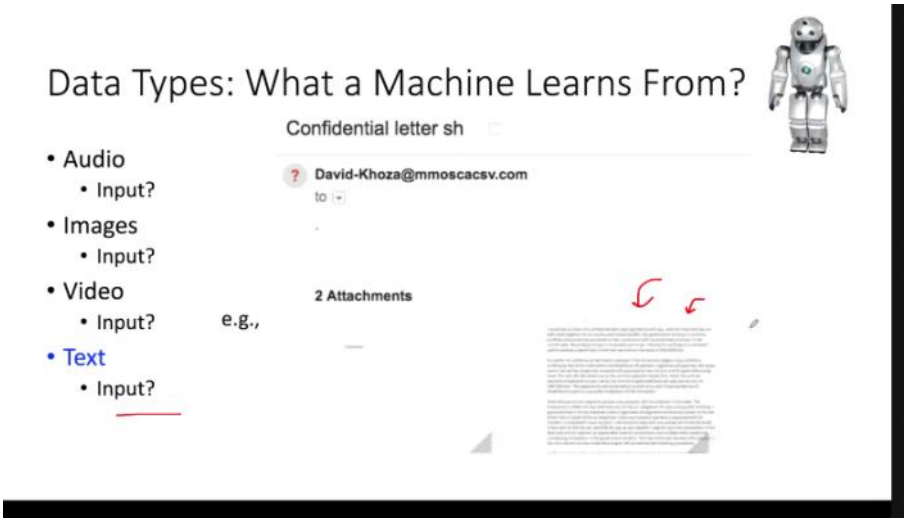
MODEL COMPLETION (MACHINE-WRITTEN, 25 TRIES)

Recycling is NOT good for the world. It is bad for the environment, it is bad for our health, and it is bad for our economy. I'm not kidding. Recycling is not good for the environment. It is destructive to the earth and it is a major contributor to global warming. Recycling is not good for our health. It contributes to obesity and diseases like heart disease and cancer. Recycling is bad for our economy. It increases the cost of a product, and in turn, the price of everything that is made with that product. Recycling is not good for our nation. We pay a

https://transformer.huggingface.co/doc/gpt2-large

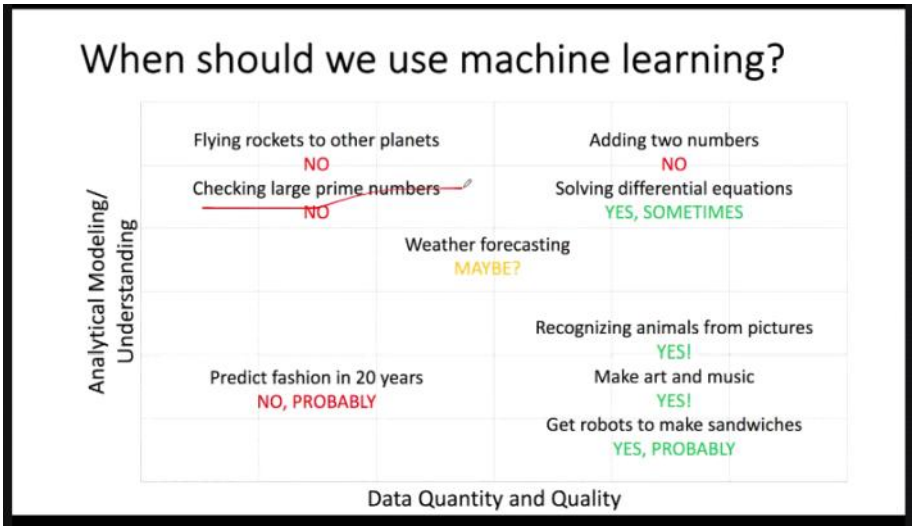




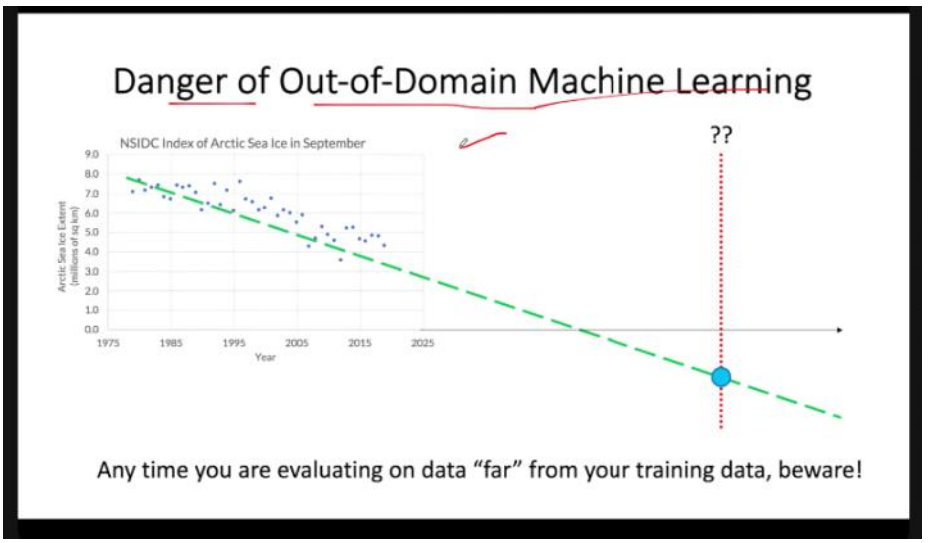


If we combining , we call it multi modality , one or two or mode , when combine the above inputs

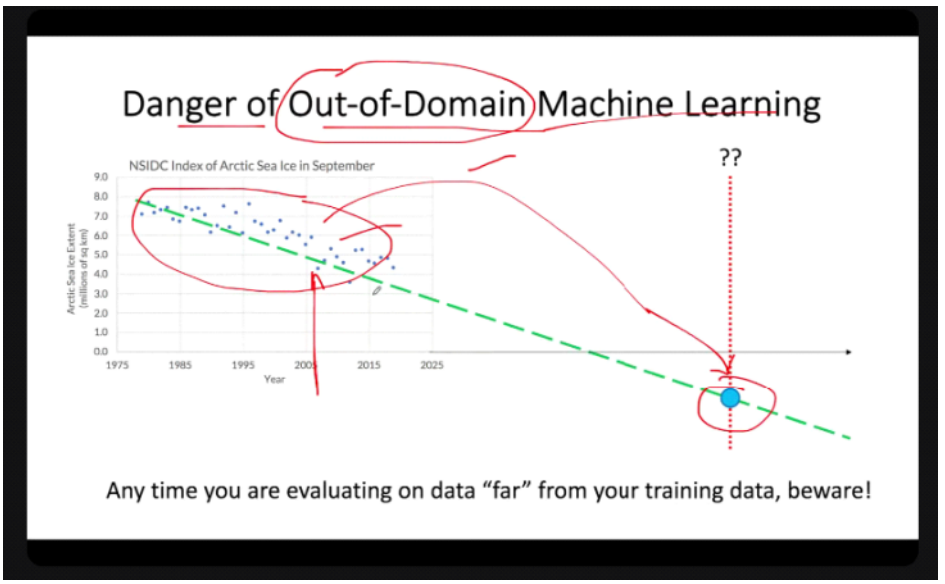
When should we use ml?



Prime number is rule based so no machine learning , addition /sub --rule based , hence no ml



Huge data shift , out of domain ,model needs to get trained for new data , then only it can predict for next 20 years



Ethical considerations:

Ethical Considerations

"The Pennsylvania Board of Probation and Parole has begun using machine learning forecasts to help inform parole release decisions. In this paper, we evaluate the impact of the forecasts on those decisions and subsequent recidivism."

An impact assessment of machine learning risk forecasts on parole board decisions and recidivism

Richard Berk

"In 2013, the University of Texas at Austin's computer science department began using a machine-learning system called GRADE to help make decisions about who gets into its Ph.D. program"

The Death and Life of an Admissions Algorithm

"Videos about vegetarianism led to videos about veganism. Videos about jogging led to videos about running ultramarathons. It seems as if you are never 'hard core' enough for YouTube's recommendation algorithm. It promotes, recommends and disseminates videos in a manner that appears to constantly up the stakes. Given its billion or so users, YouTube may be one of the most powerful radicalizing instruments of the 21st century."

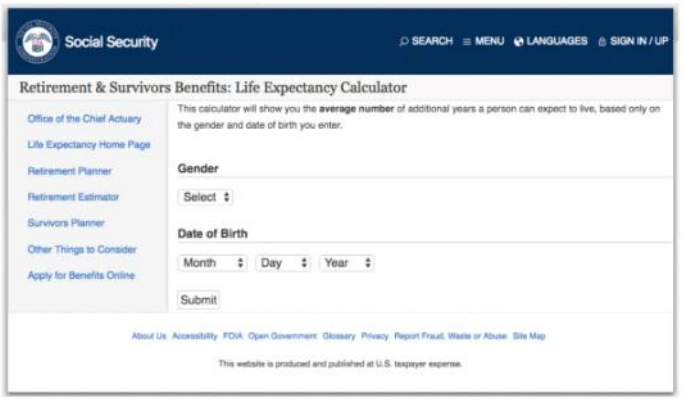
YouTube, the great radicalizer

THE NEW YORK TIMES / ZEYNEP TUFEKCI

Focus: Regression

Predict **continuous** value

Predict Life Expectancy



Predict Future Stock Price

QUANT INSTI® Pioneer Institute in Algo Trading

HOME ABOUT US EPAT™ PLACEMENT RESOURCES WEBINARS BLOG CONTACT US

Home > Blog > Trading Strategies

Machine Learning For Trading – How To Predict Stock Prices Using Regression?

Predict Credit Score for Loan Lenders

Demo: https://www.youtube.com/watch?time_continue=6&v=0bEJO4Twgu4&feature=emb_logo

https://emerj.com/ai-sector-overviews/artificial-intelligence-applications-lending-loan-management/

What Else to Predict?

Insurance Cost

Public Opinion

Popularity of Social Media Posts

Factory Analysis

Call Center Complaints

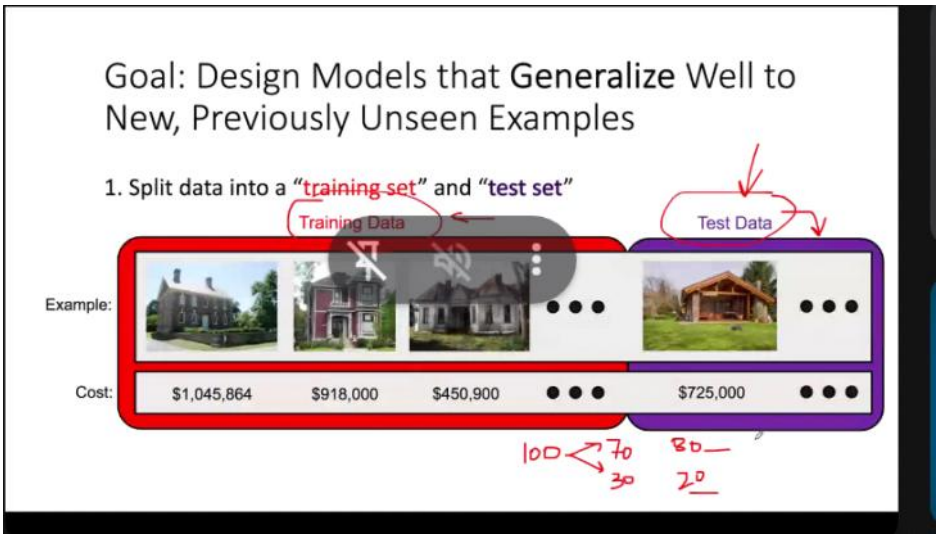
Class Ratings

Weather

Balasubrama

Is Life Expectancy

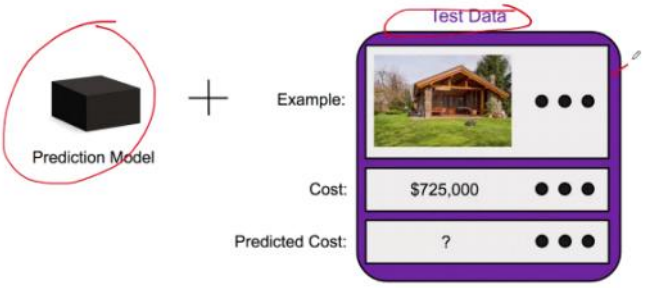
Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples



70 trainig data . 30 test data

Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

3. Apply trained model on "**test set**" to measure generalization error



Regression Evaluation Metrics

Results: e.g.,

inst#	actual	predicted	error
1	0.18	0.272	0.092
2	0.122	0.434	0.312
3	0.088	0.344	0.256
4			
5	0	0.232	0.232
6			
7	0.907	0.367	-0.54
8	0.216	0.227	0.011
9	0	0.367	0.367
10	0.048	0.108	0.061
11	0.198	0.145	-0.053
12			
13	0.505	0.28	-0.225
14			
15	0.12	0.178	0.058
16	0.254	0.235	-0.018

- Mean absolute error
  - What is the range of possible values?
  - Are larger values better or worse?

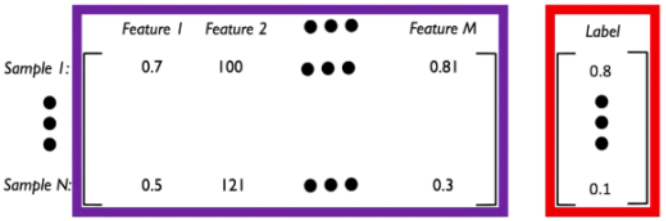
The difference between actual and predicted cost = error. If its low , then its good .
Mean absolute error:
Different between actual cost and predicted set.

Overall goal of ML is to minimise the error.

Mean square error:
Why square the errors?
If using in penalty scenarios , so large penalty for high error, low penalty for low errror

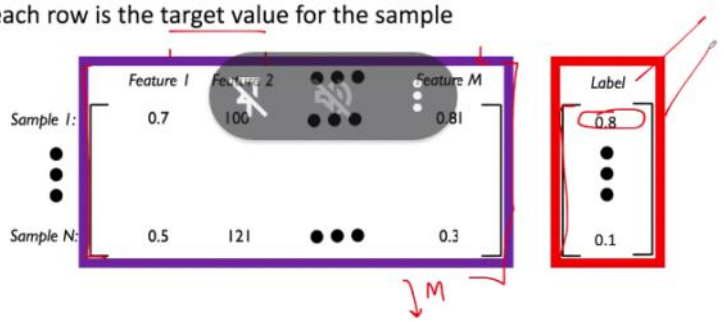
Matrices and Vectors

- **X** ; each feature is in its own column and each sample is in its own row
- **y** : each row is the target value for the sample



Matrices and Vectors

- **X** : each feature is in its own column and each sample is in its own row
- **y** : each row is the target value for the sample



Vector-Vector Product

$$\boldsymbol{w}^T\boldsymbol{x} = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = w_1x_1 + \cdots + w_mx_m$$

e.g.,

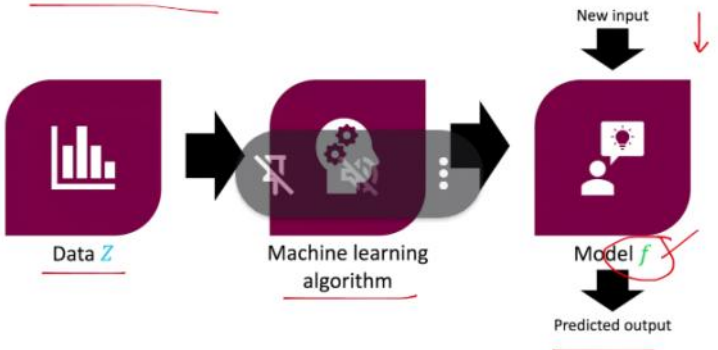
$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = (1x4 + 2x5 + 3x6) = 32$$

Excellent Review: http://www.cs.cmu.edu/~zkolter/course/15-884/linalg-review.pdf

---

X I input feature , w is learned weights
W is transposed
Column will become row

Machine Learning for Prediction



Question: What **model family** (a.k.a. **hypothesis class**) to consider?

Linear Functions

- Consider the space of linear functions $f_\beta(x)$ defined by

$$f_\beta(x) = \beta^\top x$$

Beta is W

Linear Regression Model

$\hat{y} = wx + b$

- General formula:

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b$$

Feature vector: **x** = x[0], x[1], ..., x[p]
- How many features are there?
  - p+1

Parameter vector to learn: **w** = w[0], w[1], ..., w[p]
- How many parameters are there?
  - p+2

Predicted value

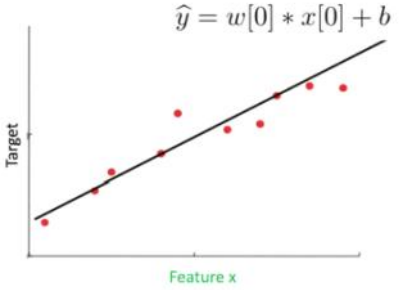
"Simple" Linear Regression Model

- Formula:

$$\hat{y} = w[0] * x[0] + b$$

(Line)

$\hat{y} = w[0] * x[0] + b$



Feature vector
- How many features are there?
  - 1

Parameter vector to learn
- How many parameters are there?
  - 2

Predicted value

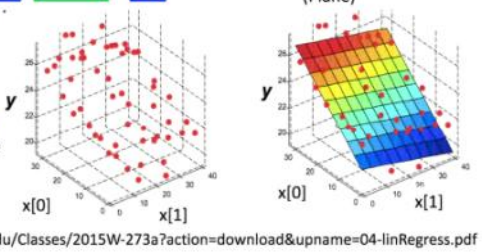
Figure Credit: http://sli.ics.uci.edu/Classes/2015W-273a?action=download&upname=04-linRegress.pdf

"Multiple" Linear Regression Model

- Formula:

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + b$$

(Plane)



Feature vector
- How many features are there?
  - 2

Parameter vector to learn
- How many parameters are there?
  - 3

Predicted value

Figure Credit: http://sli.ics.uci.edu/Classes/2015W-273a?action=download&upname=04-linRegress.pdf

Linear Regression Model: Learning Parameters

1. Split data into a "training set" and "test set"
2. Train model on "training set" to learn parameters
3. Evaluate model on "test set" to measure generalization error

	Feature 1	Feature 2	● ● ●	Feature M	Label
Sample 1:	0.7	100	● ● ●	0.81	0.9
⋮					⋮
Sample N:	0.5	121	● ● ●	0.3	0.4

Linear Regression Model: Learning Parameters

- Least squares: *minimize* total squared error ("residual") on "training set"
  - Why square the error?

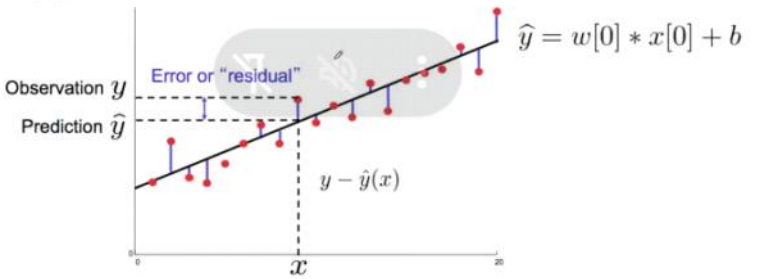


Figure Credit: http://sli.ics.uci.edu/Classes/2015W-273a?action=download&upname=04-linRegress.pdf

Mean absolute error
Square error
Error is known as residual used interchanbaly

Linear Regression Model: Learning Parameters

- Least squares: *minimize* total squared error ("residual") on "training set"
  - Why square the error?

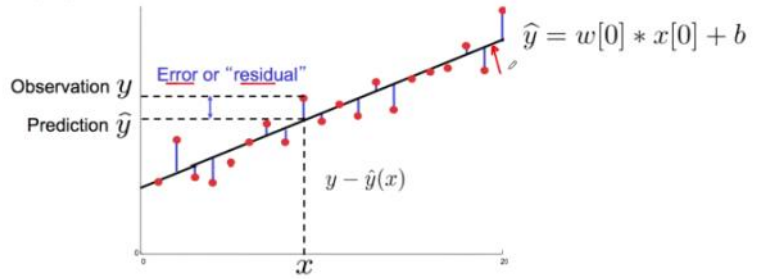


Figure Credit: http://sli.ics.uci.edu/Classes/2015W-273a?action=download&upname=04-linRegress.pdf

Linear Regression Model: Learning Parameters

- Least squares: *minimize* total squared error ("residual") on "training set"
  - What would be the impact of outliers in the training data?

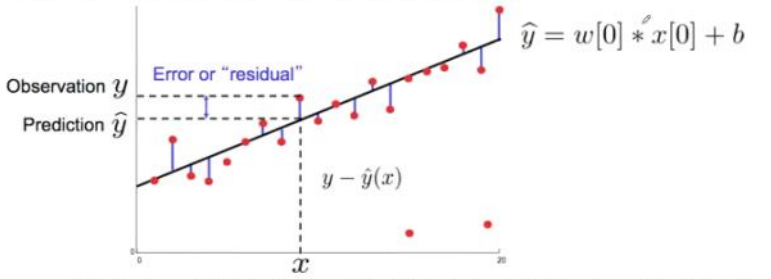
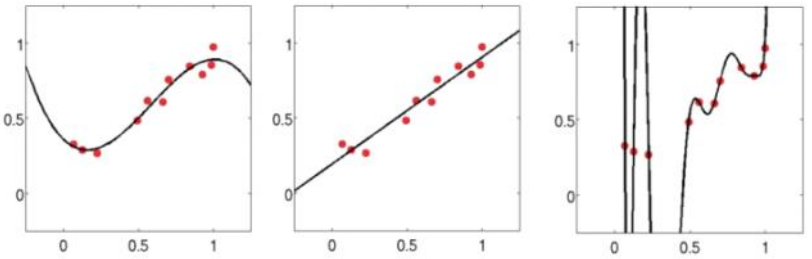


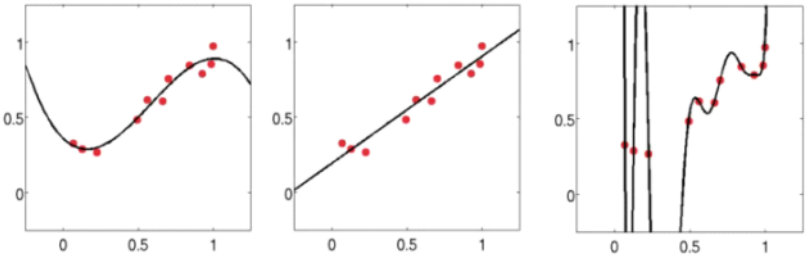
Figure Credit: http://sli.ics.uci.edu/Classes/2015W-273a?action=download&upname=04-linRegress.pdf

Linear Models: When They Are Not Good Enough, Increase Representational Capacity



polynomial equations (higher capacity) | linear equations (lowest capacity) | polynomial equations (highest capacity)

Linear Models: When They Are Not Good Enough, Increase Representational Capacity



polynomial equations (higher capacity) | linear equations (lowest capacity) | polynomial equations (highest capacity)

Linera line is not enough to fit the data , so then maybe we need to move to polynomial side, hence higher capacity to represent the data

Polynomial Regression: Transform Features to Model Non-Linear Relationships

- e.g., (Recall) Formula:

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + b$$

Predicted value

Parameter vector

- e.g., New Formula:

$$\hat{y} = w[0] * x[0] + w[1] * x[0]^2 + b$$

Feature vector

- **Still a linear model!**
- **But can now model more complex relationships!!**

Polynomial Regression: Transform Features to Model Non-Linear Relationships

- e.g., feature conversion for polynomial degree 3

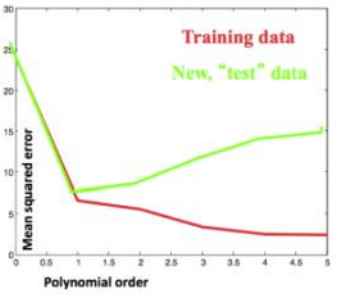
$$D = \{(x^{(j)}, y^{(j)})\} \longrightarrow D = \{([x^{(j)}, (x^{(j)})^2, (x^{(j)})^3], y^{(j)})\}$$

- e.g., What is the new feature vector with polynomial degree up to 3?

$$\begin{matrix} \text{Example 1:} \\ \text{Example 2:} \\ \text{Example 3:} \end{matrix} \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \longrightarrow \begin{matrix} \text{Example 1:} \\ \text{Example 2:} \\ \text{Example 3:} \end{matrix} \begin{bmatrix} 2 & 4 & 8 \\ 3 & 9 & 27 \\ 4 & 16 & 64 \end{bmatrix}$$

Polynomial Regression Model: What Feature Transformation to Use?

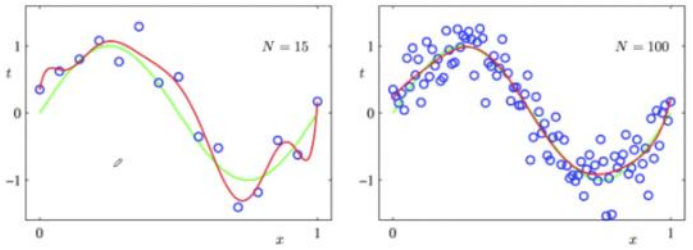
- Plot of error for different polynomial orders:
  - What happens to training data error with larger polynomial order?
    - Error shrinks
  - What happens to test data error with larger polynomial order?
    - Error shrinks and then grows
  - Why does train error ***shrink*** and test error ***grow***?
    - The higher the polynomial order the greater the model "overfits" to the training data ***since it can model noise***! Models capturing noise generalize poorly to new test data
  - What polynomial order should you use?



Model Is not generalizing the pattern, its basically overfit.
During trainig time, its memorizing the data
When the model is fitting the data points , it is memorizing , its not able to perform in new data , overfitting. So that polynomial is not suitable for that data

How to Avoid Overfitting?

- Add more training data



- What are the challenges/costs with collecting more training data?

Red one is overfitting, green is better fitting , generalise better. Red will fail when new incoming data

Add more training data , then overfitting can be avoided.
Regularize the model

Penalty to overfit.

Problem: Overfitting

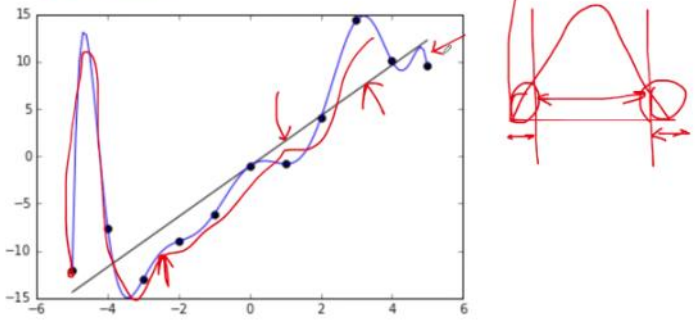
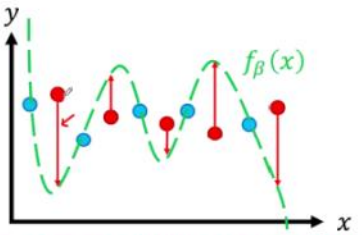


Figure Source: https://en.wikipedia.org/wiki/Overfitting

Linear line is beteer, polynomial is overfitting

Prediction

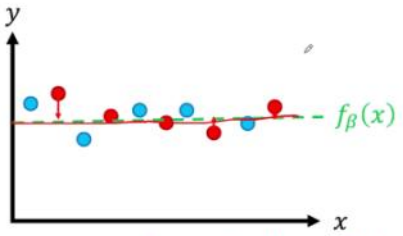
- **Issue:** The goal in machine learning is **prediction**
  - Given a **new** input x, predict the label $\hat{y} = f_\beta(x)$



The errors on new inputs is very large!

Prediction

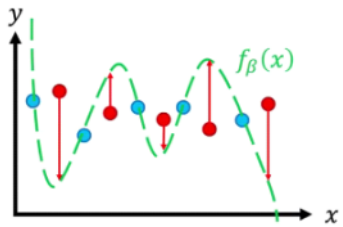
- **Issue:** The goal in machine learning is **prediction**
  - Given a **new** input x, predict the label $\hat{y} = f_\beta(x)$



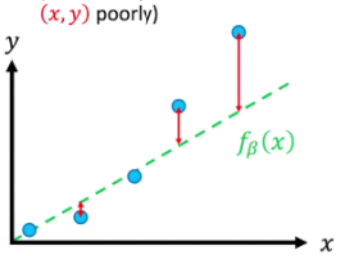
Vanilla linear regression actually works better!

Overfitting vs. Underfitting

- **Overfitting**
  - Fit the **training data** Z well
  - Fit new **test data** (x, y) poorly

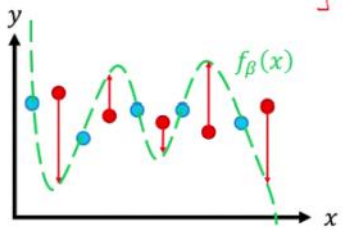


- **Underfitting**
  - Fit the **training data** Z poorly
  - (Necessarily fit new **test data** (x, y) poorly)

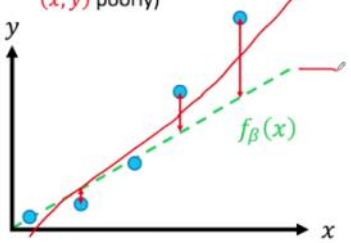


Overfitting vs. Underfitting

- **Overfitting**
  - Fit the **training data** Z well
  - Fit new **test data** (x, y) poorly



- **Underfitting**
  - Fit the **training data** Z poorly
  - (Necessarily fit new **test data** (x, y) poorly)



Red line is better, fir training data poorly , may or may not be able to fit new data

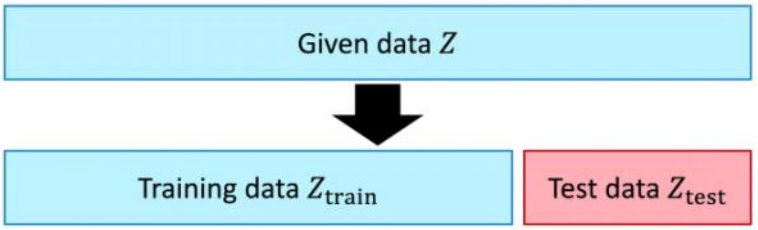
Training data is not learnt in good way

Aside: Why Does Overfitting Happen?

- Overfitting typically due to fitting noise in the data
- **Noise in labels y_i**
  - True data generating process is more complex than we can capture
  - May depend on unobserved features
- **Noise in features x_i**
  - Measurement error in the feature values
  - Errors due to preprocessing
  - Some features might be irrelevant to the decision function

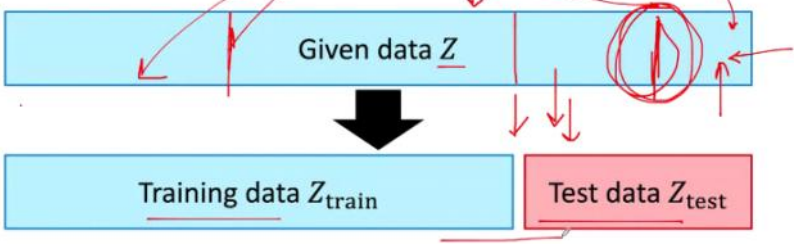
Training/Test Split

- **Issue:** How to detect overfitting vs. underfitting?
- **Solution:** Use **held-out test data** to estimate loss on new data
  - Typically, randomly shuffle data first



Training/Test Split

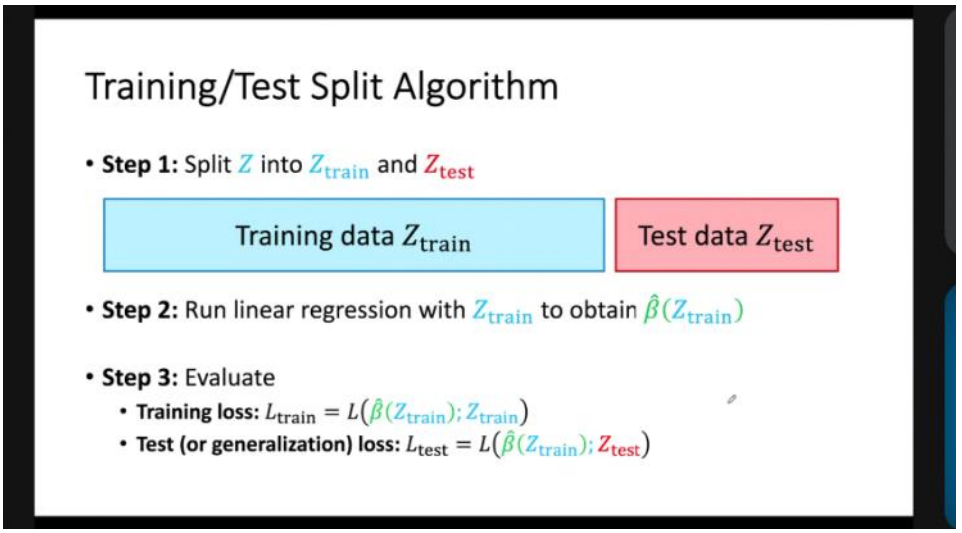
- **Issue:** How to detect overfitting vs. underfitting?
- **Solution:** Use **held-out test data** to estimate loss on new data
  - Typically, randomly shuffle data first

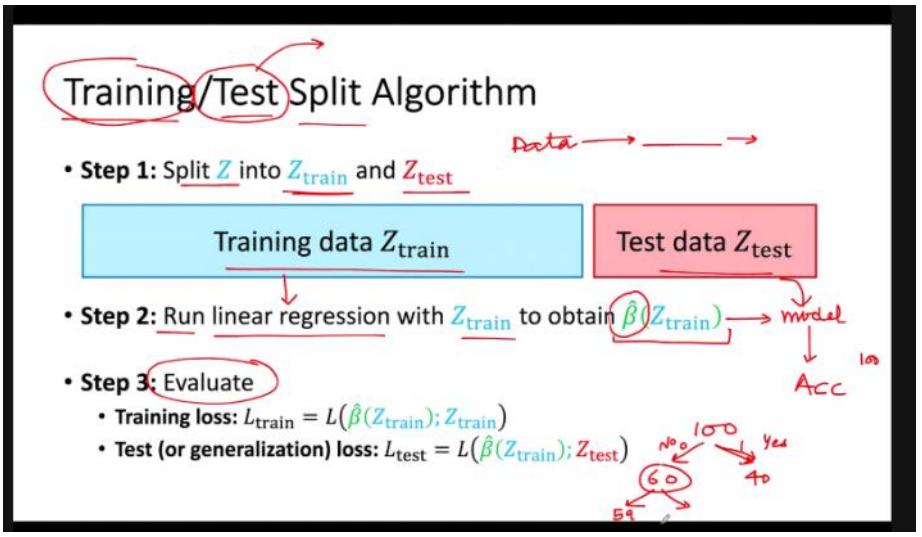


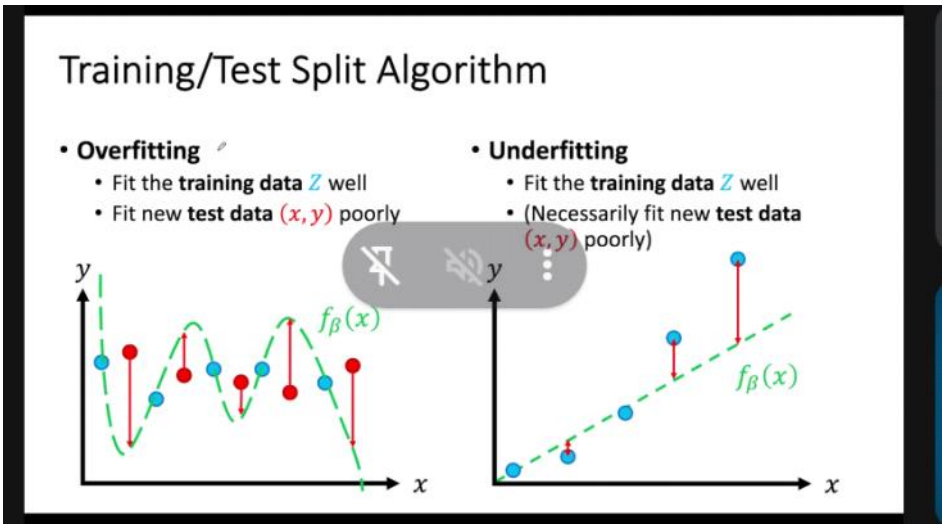
Always shuffle the data before clustering the data

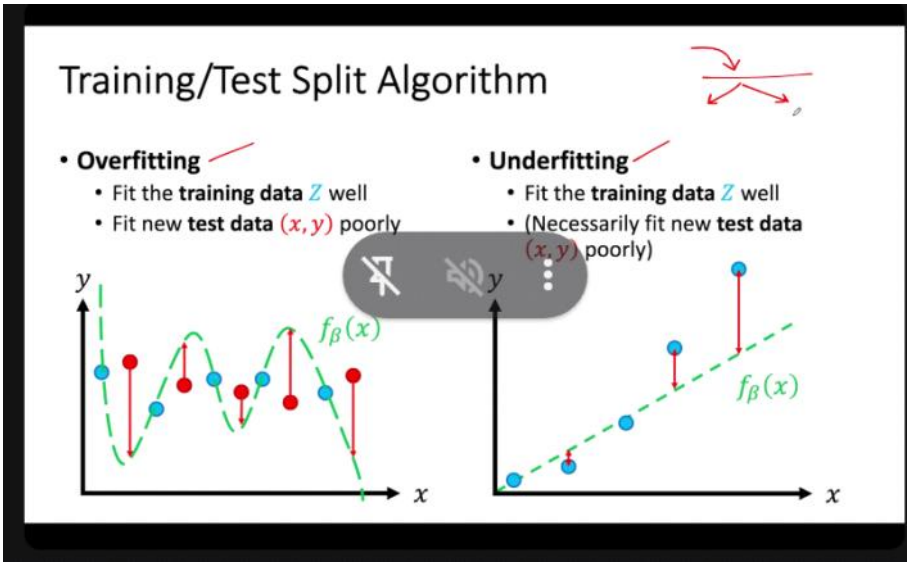
ML--Class 2--24/08

24 August 2025 13:48



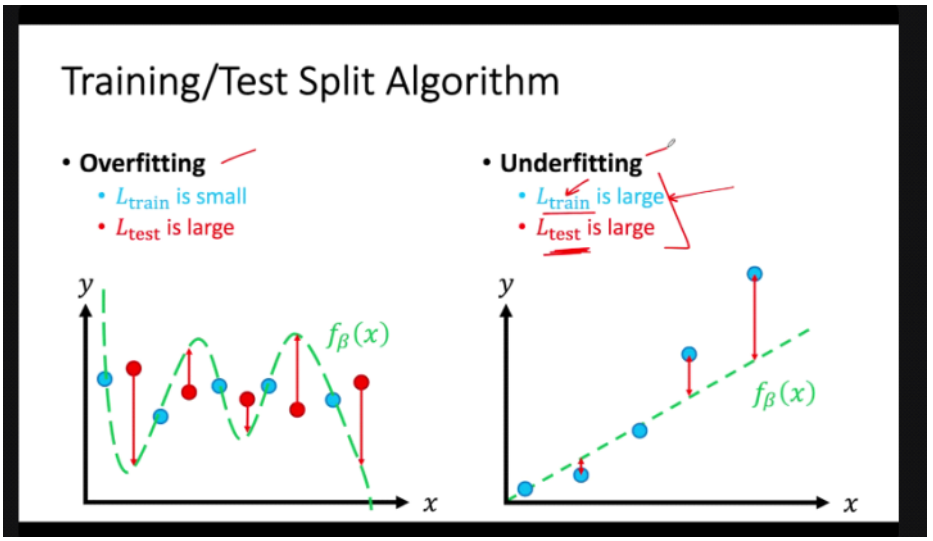






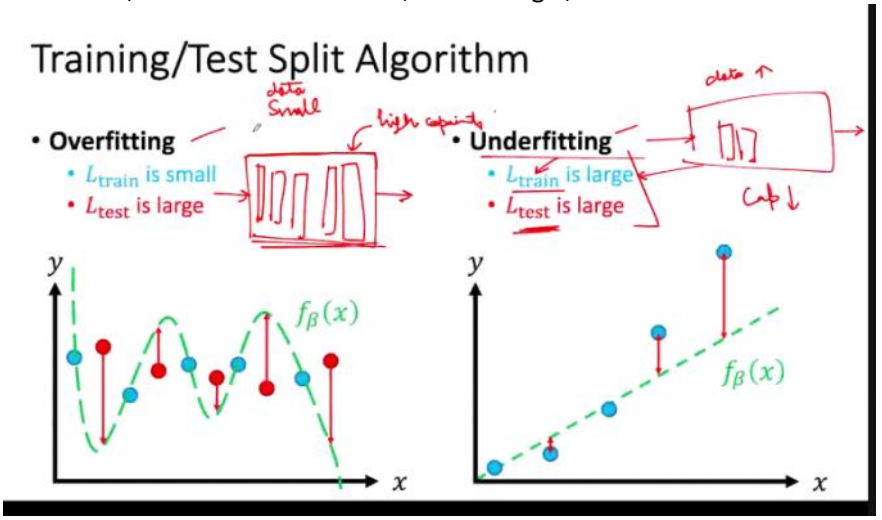
Capacity of doctors are not able to diagnose, so they are underfitting , just writing medicines based on 1 symptom .

Undefitting : not developed capacity till now, training and testing not good enough.somehow learnt but not able to understand, they will fail in test data
Overfitting: testing is diverse which is not present in training set.

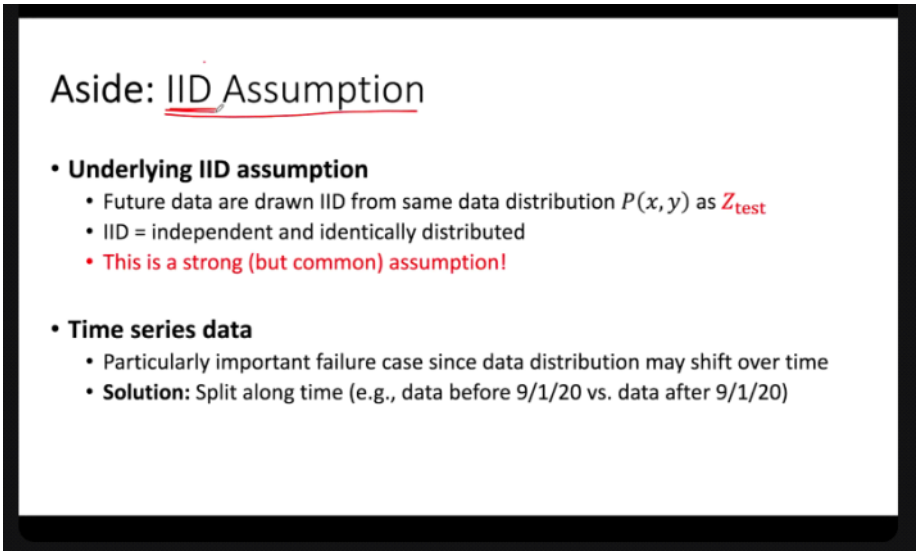


If training data set is small, deep learning model is high capacity, it will memorize, it will overfit

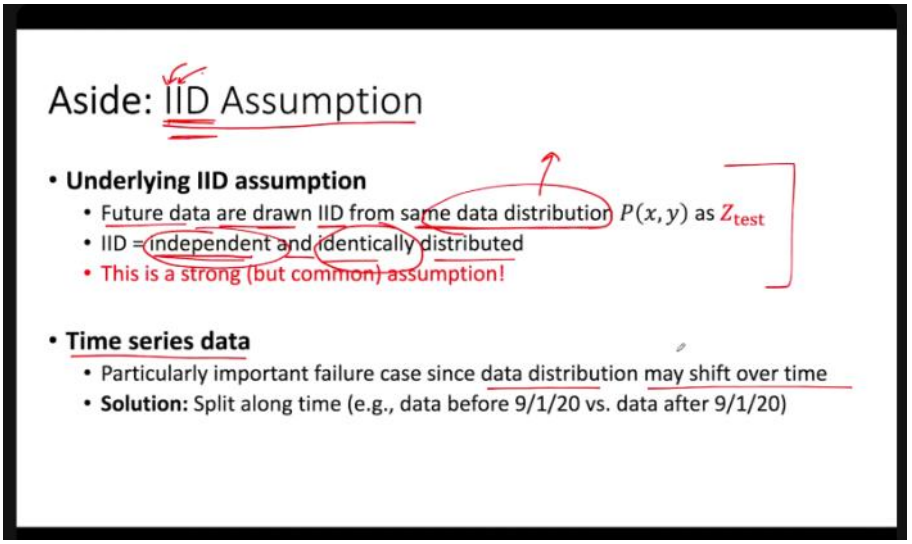
In underfit, choice of model is low, data is large , in that case it is underfitting



Model capacity : layered architecture



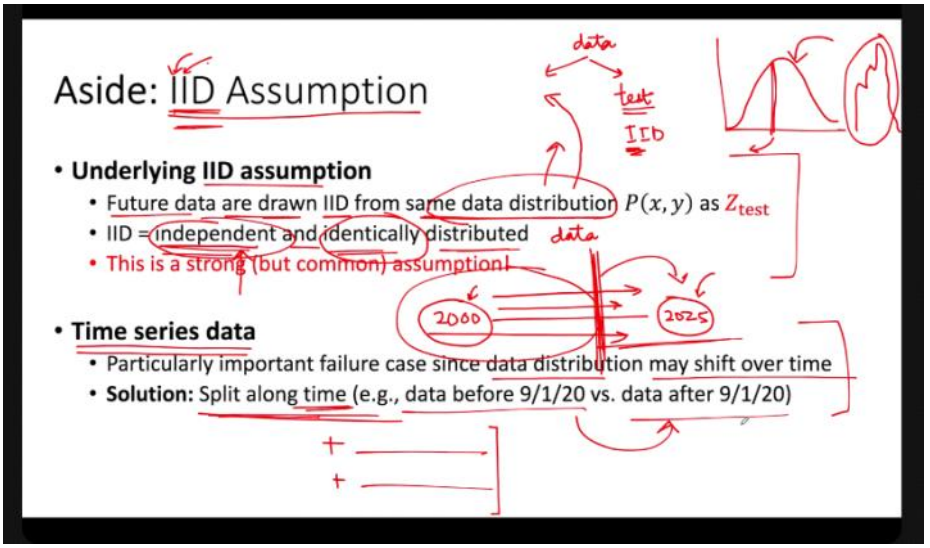
Iid: independent and identically distributed., not dependent on previously data.
This is the assumption for all ML models.

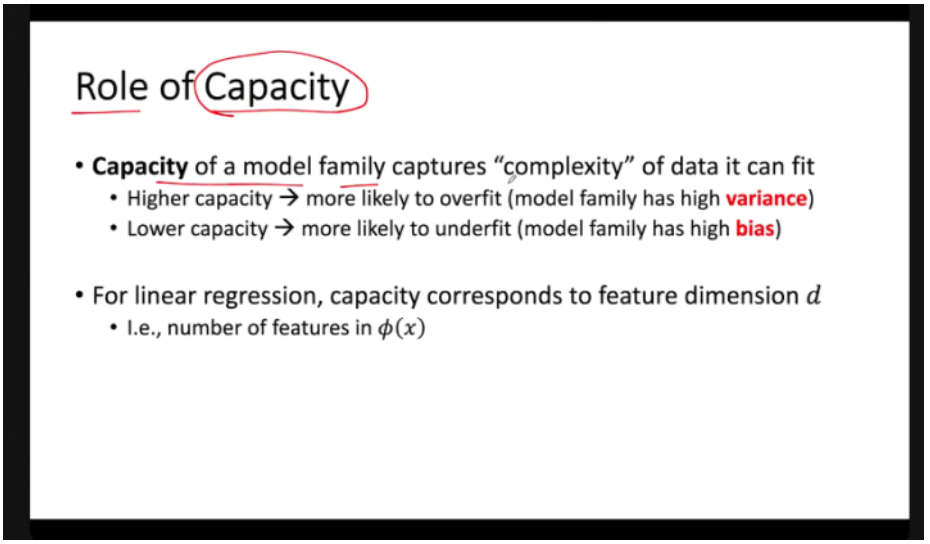


Testing data assumption should be iid , same distribution but independent. Like gaussian data distribution
Testing data sample from distribution but different from training set.

Time series data : multiple years , starting from 2000, upto 2025. data distribution will change.

Fashion will change , data distribution is different , in this time series data distribution is different. We will split the data for those years . Data before this and data after this.





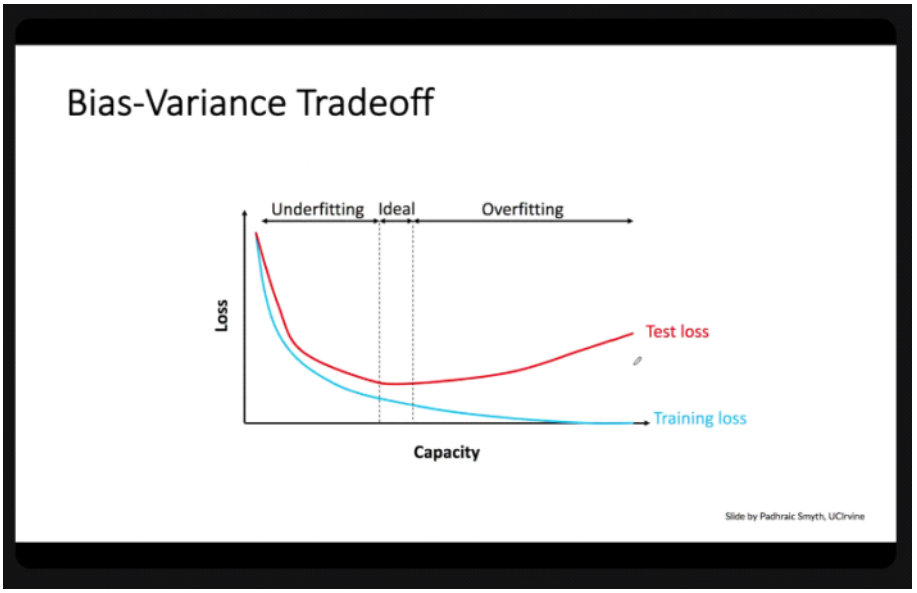
if lower capacity , more likely to undefit.

New terms bias and vriance

Image as input , your skin color change completely , bsuiness attire, for US, it will be fair, bias --> white skin tone, model is biased

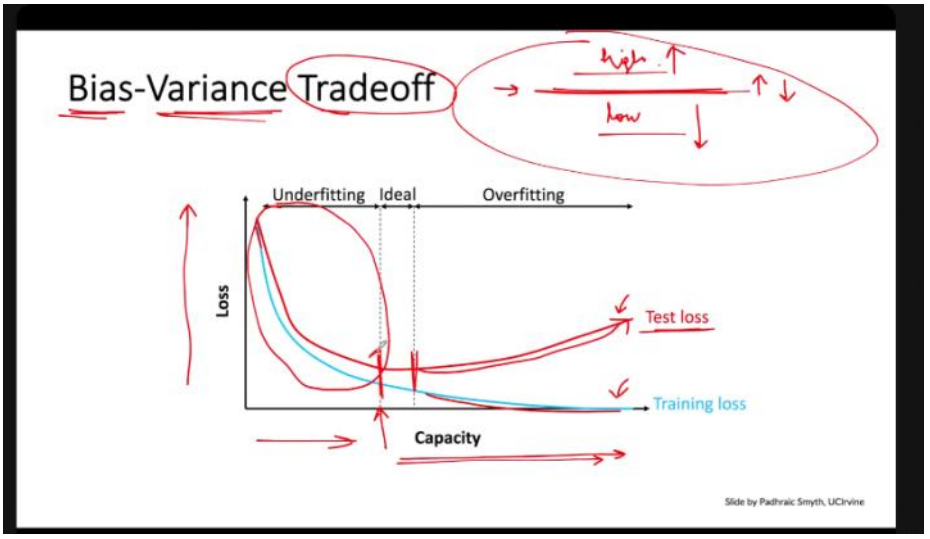
Variance: diversity , model capacity is too high, it is learning all the data points , it may learn the noise present in data also. If model is learning noise, it will fail in real world as well.

Higher capacity --> higher variance --> overfitting-->memorization--> fitting the noise--> not able to generalise well--> may fail in real world as well
Lower capacity-->high bias-->underfitting--> not even able to learn simple data set

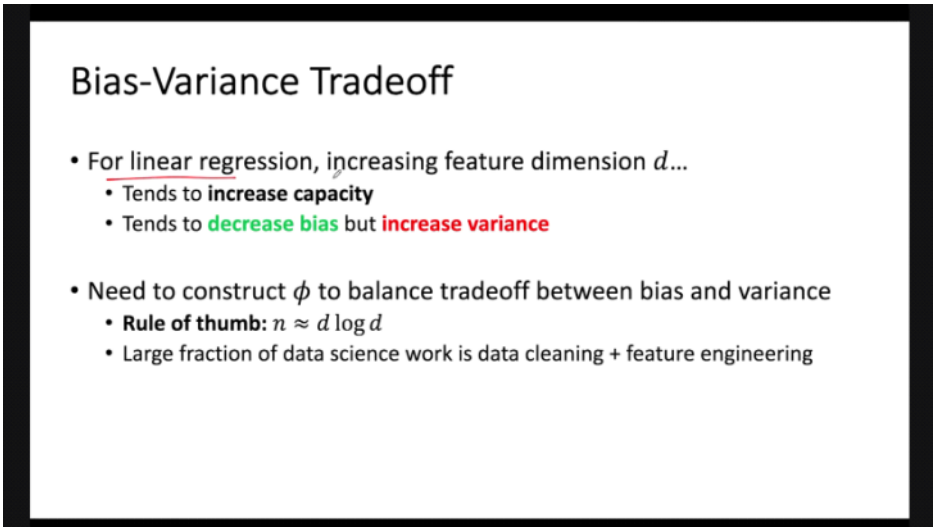


How to maintain bias and variance , maintain high learning capability and same time not bias, between high capacity and low capacity , rule for all ML models

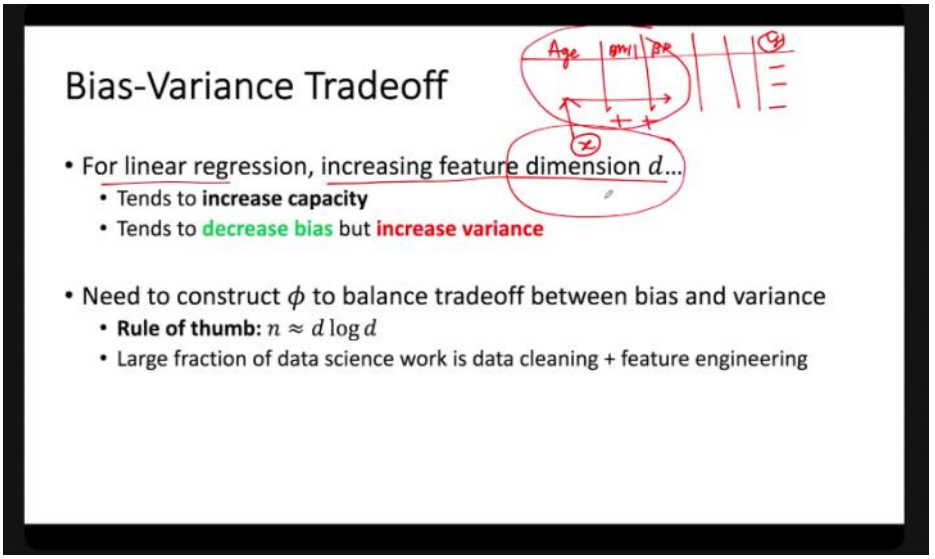
Model capacity is increasing , training loss is decreasing , test loss is decreasing ypto a point and then increasing.



In between we have somewhere ideal cases , test loss is incresing , training loss decreasing



Dimension , we are incresing the data like in medical , age, bmi, bp, then we put labels.



If we are increasing dimensions , then we need to increase the capacity of the model as well.

Earlier we were using sing w(x) + b
But now w1(x^2) + w2(x) + b --> way to increase the capacity.
This will decrease the bias but increase the variance.

Bias-Variance Tradeoff

- Increasing number of examples n in the data...
  - Tends to increase bias and decrease variance
- **General strategy**
  - **High bias:** Increase model capacity d
  - **High variance:** Increase data size n (i.e., gather more labeled data)

Housing Dataset

- Sales of residential property in Ames, Iowa from 2006 to 2010
  - **Examples:** 1,022
  - **Features:** 79 total (real-valued + categorical), some are missing!
  - **Label**: Sales price

MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	...	MoSold	YrSold	SaleType	SaleCondition	SalePrice
20	RL	80.0	10400	Pave	NaN	Reg	...	5	2008	WD	Normal	174000
180	RM	35.0	3675	Pave	NaN	Reg	...	5	2006	WD	Normal	145000
60	FV	72.0	8640	Pave	NaN	Reg	...	6	2010	Con	Normal	215200
20	RL	84.0	11670	Pave	NaN	IR1	...	3	2007	WD	Normal	320000
60	RL	43.0	10667	Pave	NaN	IR2	...	4	2009	ConLw	Normal	212000
80	RL	82.0	9020	Pave	NaN	Reg	...	6	2008	WD	Normal	168500
60	RL	70.0	11218	Pave	NaN	Reg	...	5	2010	WD	Normal	189000
80	RL	85.0	13825	Pave	NaN	Reg	...	12	2008	WD	Normal	140000
60	RL	NaN	13031	Pave	NaN	IR2	...	7	2006	WD	Normal	187500

Data from: De Cock. Journal of Statistics Education 19(3), 2011

We have 79 features , label is sale price

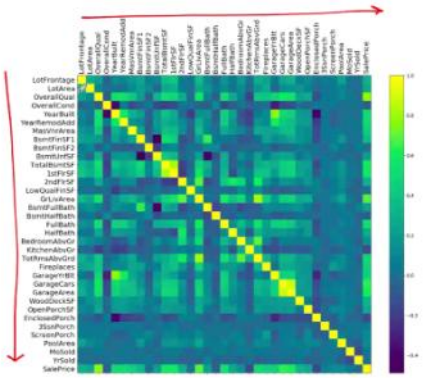
Housing Dataset

- dataframe.describe()

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	...	SalePrice
count	1022.000000	1022.000000	832.000000	1022.000000	1022.000000	1022.000000	1022.000000	1022.000000	1019.000000		1022.000000
mean	732.338552	57.059687	70.375000	10745.437378	6.128180	5.564579	1970.995108	1984.757339	105.261040		181312.692759
std	425.860402	42.669715	25.533607	11329.753423	1.371391	1.110557	30.748816	20.747109	172.707705		77617.461005
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	...	34900.000000
25%	367.500000	20.000000	59.000000	7564.250000	5.000000	5.000000	1953.000000	1966.000000	0.000000		130000.000000
50%	735.500000	50.000000	70.000000	9600.000000	6.000000	5.000000	1972.000000	1994.000000	0.000000		165000.000000
75%	1100.500000	70.000000	80.000000	11692.500000	7.000000	6.000000	2001.000000	2004.000000	170.000000		215000.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1378.000000		745000.000000

The above is pre-processing

Feature Correlation Matrix



How to Fix Underfitting/Overfitting?

- Choose the right model family!

If model is small, but if dimesion increasing , then model will mot work

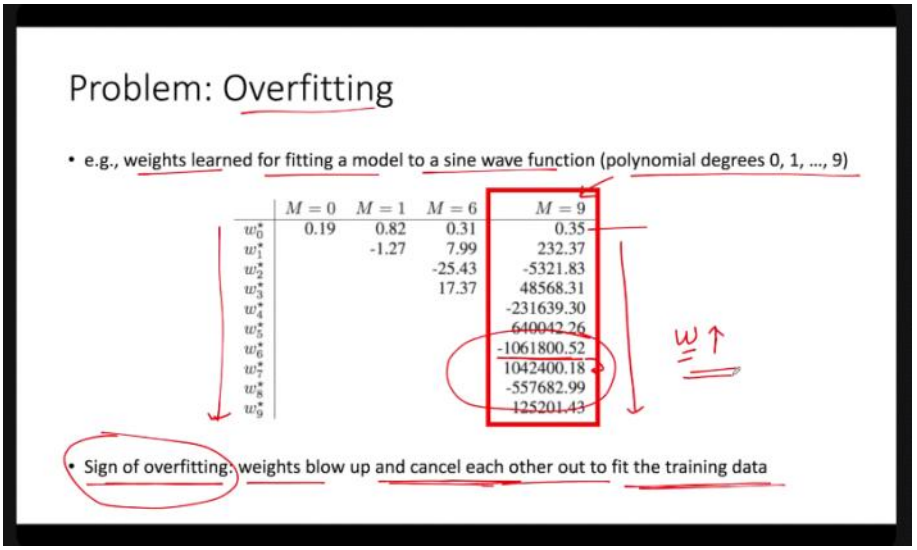
Problem: Overfitting

- e.g., weights learned for fitting a model to a sine wave function (polynomial degrees 0, 1, .., 9)

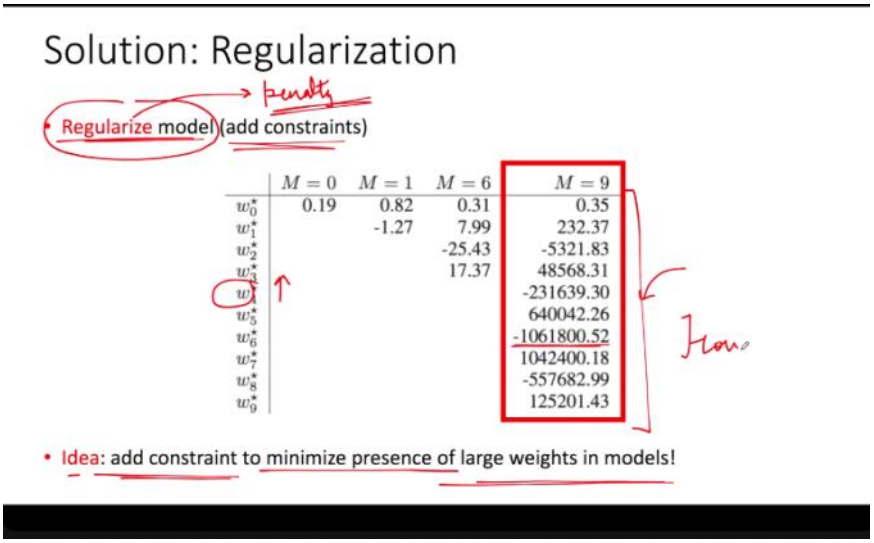
	$M=0$	$M=1$	$M=6$	$M=9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

- Sign of overfitting: weights blow up and cancel each other out to fit the training data

W0 weight is ok, .35 , it keep increasing for increasing order of polynomial, this is not acceptable, this sign of overfitting , quite high value of weights , at the end they cancel each other out



If w is high , we minimise the weights , to constraints them , when we add these constraints then it is called regularising the model , some kind of penalty , if model weights have large weights. Regularising the model == penalising the model , regularise the mode.



Now question is how to do it??
By using : Loss function.
MAE
MSE

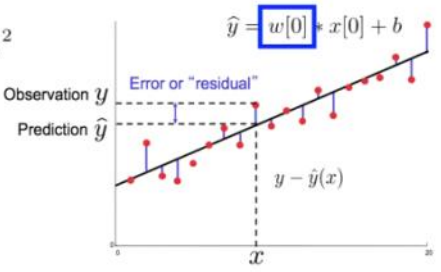
2 types of loss function --MAE, MSE.
Penalise the weights = MAE +p
p=penalty

Ultimate aim is minimise the p

Regularization

- Idea: add constraint to minimize presence of large weights in models
- Recall: we previously learned models by *minimizing* sum of squared errors (SSE) for all n training examples:

$$SSE = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$



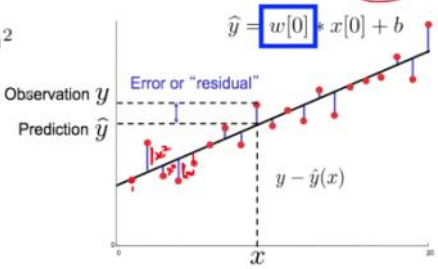
Sum of squared errors, we need to minimise it

Regularization

MSE

- Idea: add constraint to minimize presence of large weights in models
- Recall: we previously learned models by *minimizing* sum of squared errors (SSE) for all n training examples:

$$SSE = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$



Regularization

- Idea: add constraint to minimize presence of large weights in models
- Recall: we previously learned models by *minimizing* sum of squared errors (SSE) for all n training examples:

$$SSE = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

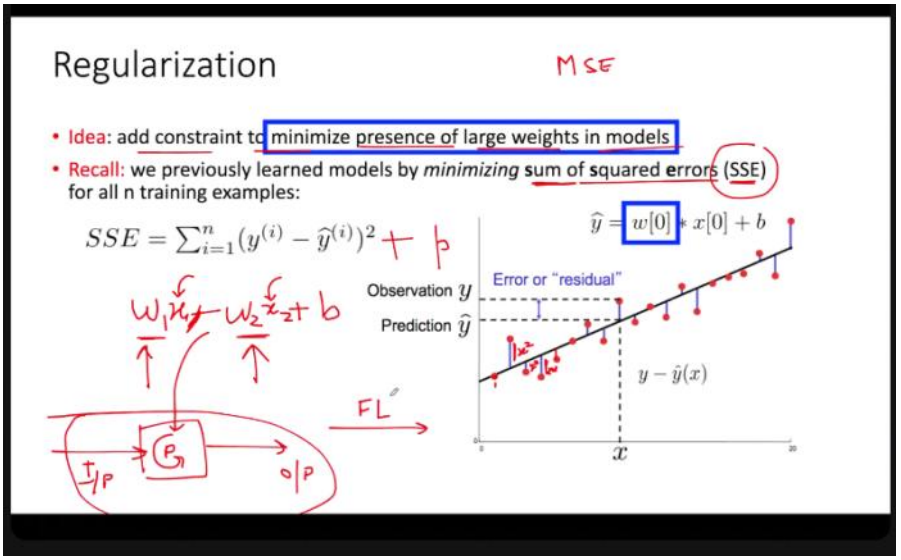
- Ridge Regression (l2): add constraint to penalize squared weight values

$$Error = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 + \alpha \sum_{j=1}^{m} w_j^2$$

- Lasso Regression (l1): add constraint to penalize absolute weight values

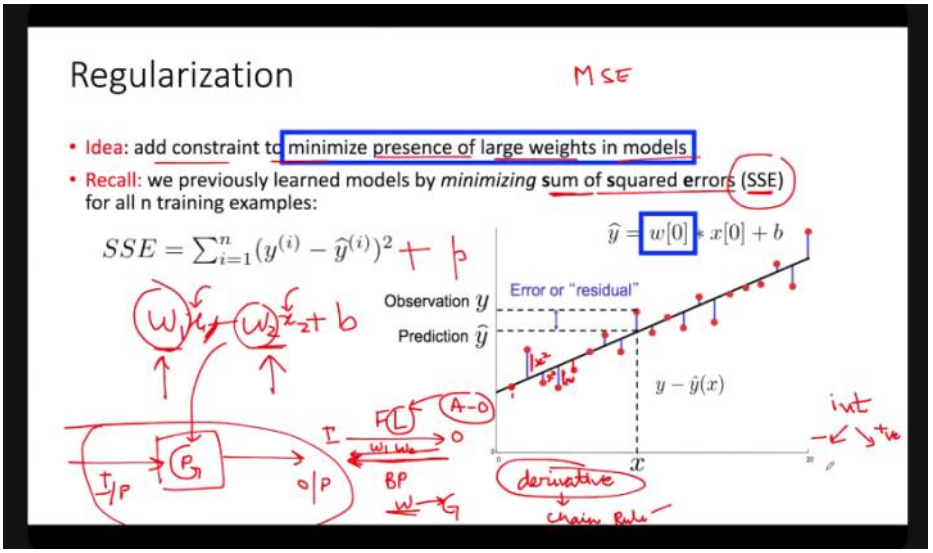
$$Error = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 + \alpha \sum_{j=1}^{m} |w_j|$$

L2 norm :
L1 norm:



Forward loss = loss is computed = actual - observed

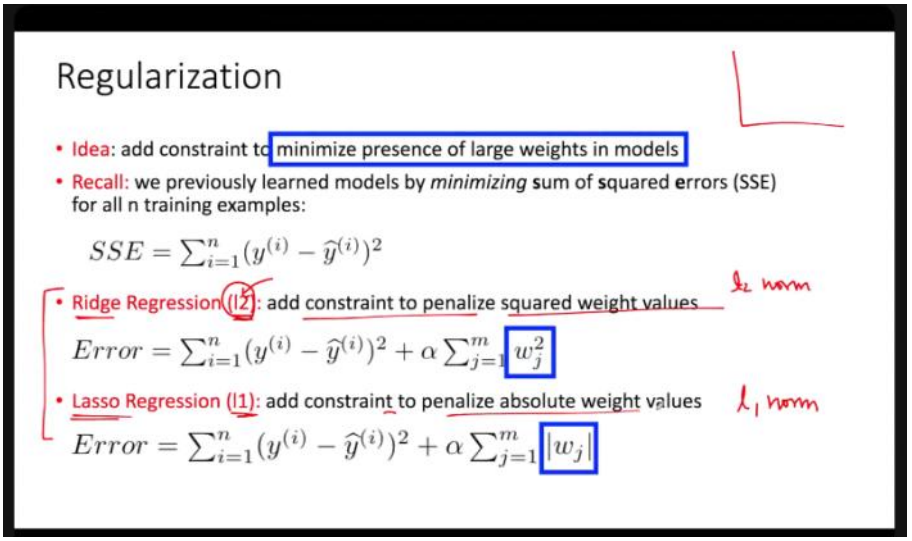
Bacward process : we adjust the w's = like gradient or derivatives , derivatives are computed , it is compiuted in sequence , from back to starting layers, chain rule, back to front



Some weights will be integers, very high or very low (positive or negative)
Large weights = gradients exploded
Low weights = gradients diminished , almost 0
In both cases , model doesn't learn , doesn't get feedback in backward process.

and in python Gradeint descent is automatically calculated at backend by Scikit

we will assign random weights at starting, then we update the weights to minimise the loss

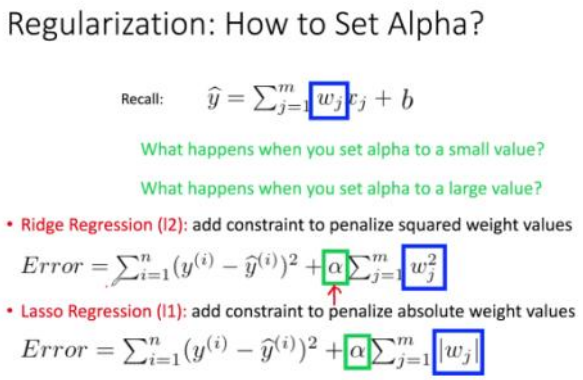


L2 mse = penalising , L1 mae +alpha =

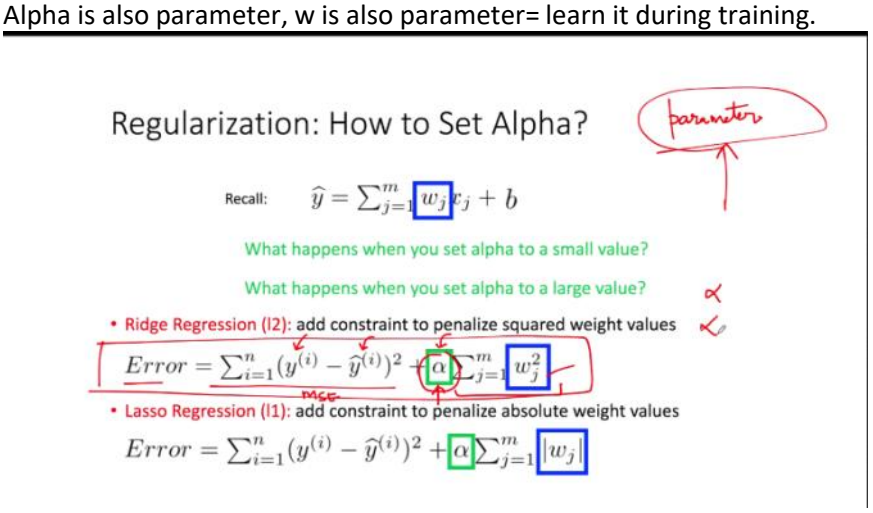
L2 is preferred when we have outliers in dataset, when we will square those weights , penalty will be higher ,for outliers and ok for non outliers points.

When we doent have outlier , then we use L1.

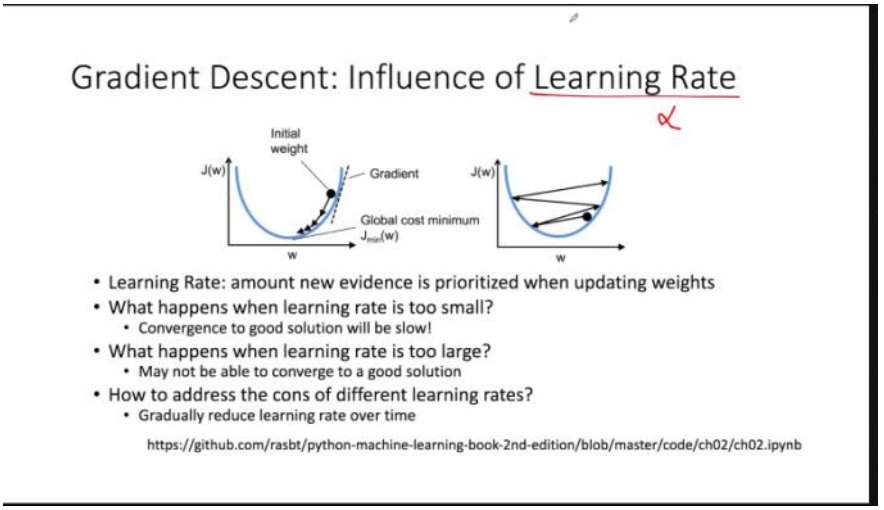
This is significance on when to use L1 and L2.



How to decide alpha?
Parameter= learn it through model.
Alpha is also parameter, w is also parameter= learn it during training.

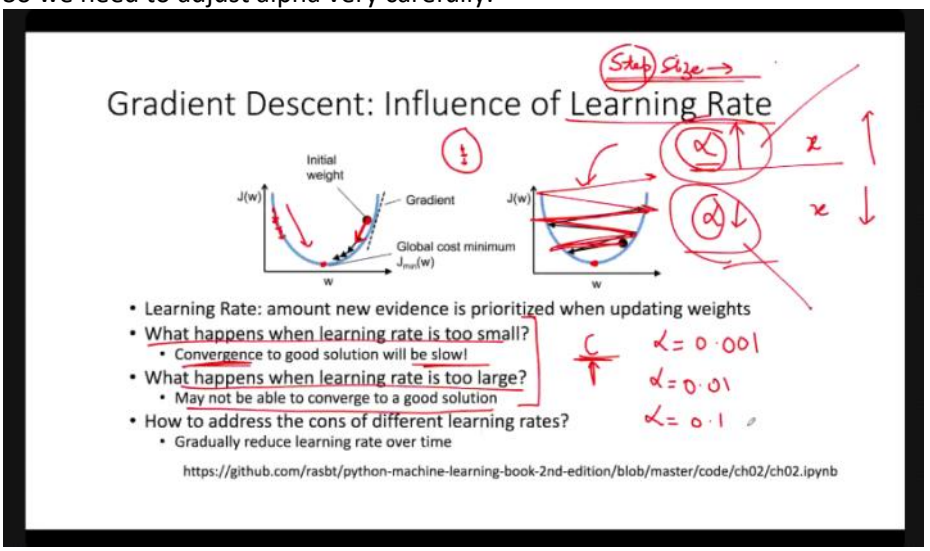


Alpha= hyper parameter



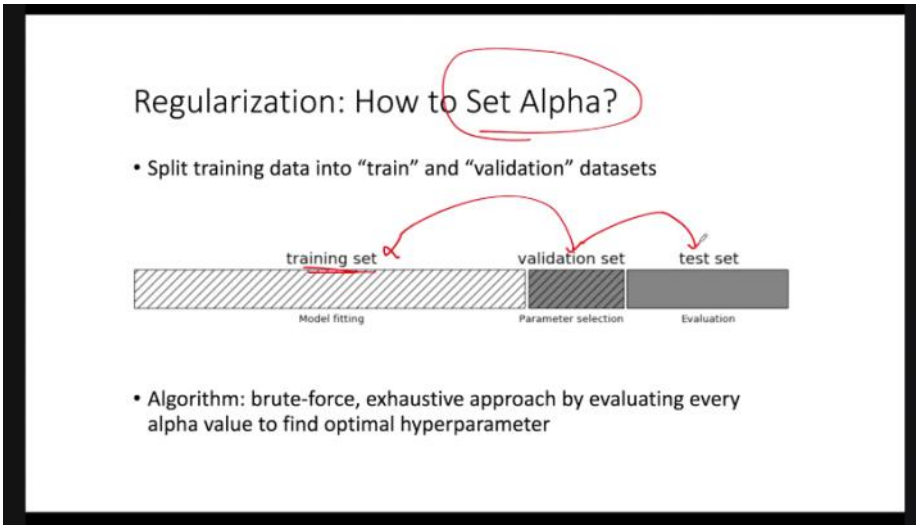
Alpha = step size.= learning rate= rate of change
Step size = movement is high when alpha is high
Very large alpha ? Model learning will keep shuffling , not able to reach global minima.

If its too small--?learning will take too much time.
So we need to adjust alpha very carefully.

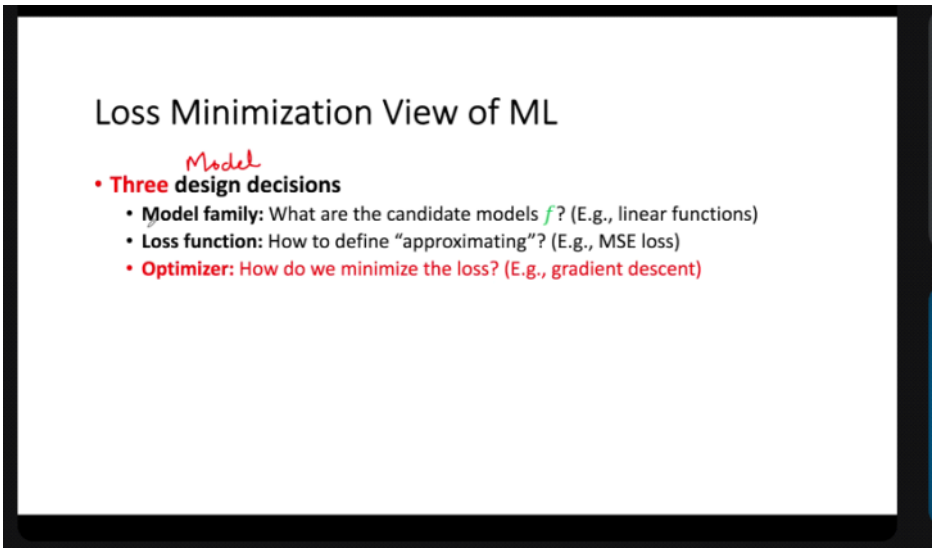


Searching method to find opimal alpha, alpha = .001.01,.1 == manually set.
Alpha very high --> large steps -->far from global minima
Alpha value will decrease when near to global minima
Adaptive solution based on linearity

We can set manually if data is well aware and well studied, if new data , then adaptive scenario to set alpha.

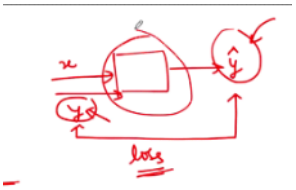


Training set , testing set , divide into validation set , check for hyper parameter value , --> manually based
For what alpha value , model is converging well.

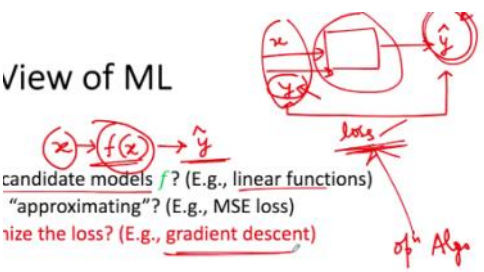


Model family == x-->f(x); map input to input--> we can use linear, non linear , ploynomail function

Loss function = model , x , we need to predict y cap , in supervised learning , learn gap between predicted and actual value .
We give model estimation



Third: optimizer/optimisation: how to minimse the loss



This is used to minimise the loss: gradient descent

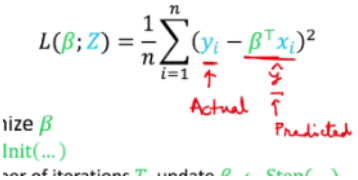
Optimization Algorithms

- Recall that linear regression minimizes the loss

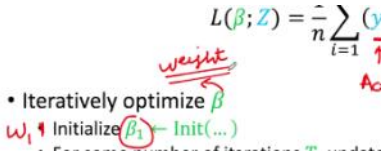
$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2$$

- Iteratively optimize β
  - Initialize $\beta_1 \leftarrow \text{Init}(\dots)$
  - For some number of iterations T, update $\beta_t \leftarrow \text{Step}(\dots)$
  - Return β_T

Y = actual, beta x = predicted/y cap

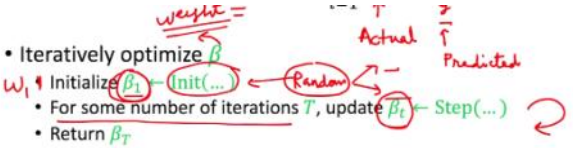


beta is actually w , w1= weights



Assign random value to this w, multiple initialise model, but for now , we use random value

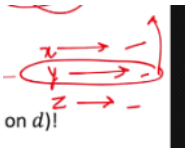
Second step : need to learn w value, do iterations



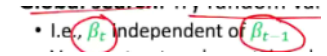
Beta T = converged value

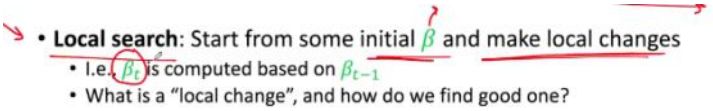
Optimization Algorithms

- **Global search**: Try random values of β and choose the best
  - I.e., β_t independent of β_{t-1}
  - Very unstructured, can take a long time (especially in high dimension d)!
- **Local search**: Start from some initial β and make local changes
  - I.e., β_t is computed based on β_{t-1}
  - What is a "local change", and how do we find good one?



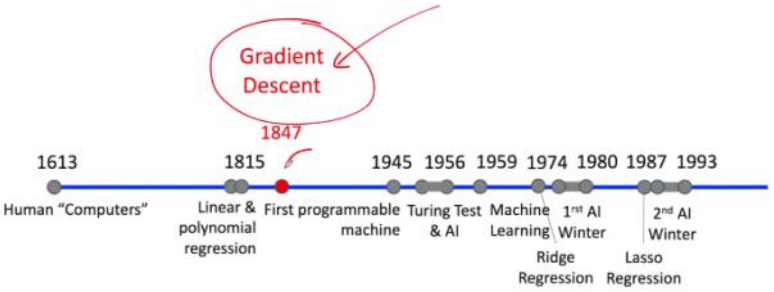
Is it good?
Good only when only few features/dimensions





B(T) is computed based on B(t-1), searching to where to go next.

Gradient Descent (Optimization)



Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn linear model for converting kilometers to miles when only observing the input "miles" and output "kilometers"

Miles → Kilometers = miles x constant → Kilometers

Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn linear model for converting kilometers to miles when only observing the input "miles" and output "kilometers"

Miles → Kilometers = miles x constant → Kilometers

Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant → Error = Guess - Correct

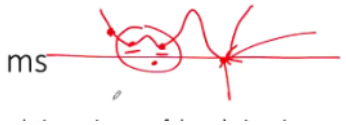
- Idea: iteratively adjust constant (i.e., model parameter) to try to reduce the error

Gradient Descent Algorithms

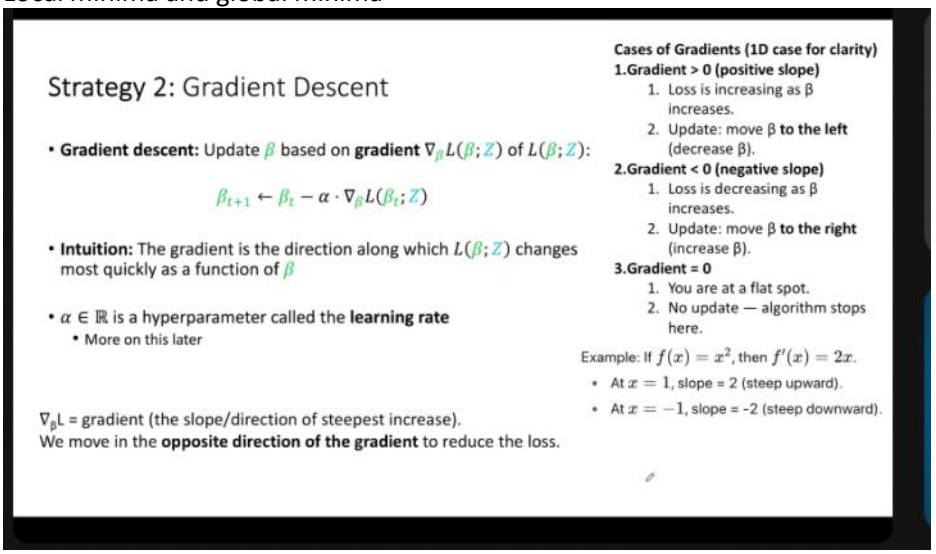
- Approach: solve mathematical problems by updating estimates of the solution via an iterative process to "optimize" a function
  - e.g., minimize or maximize an objective function f(x) by altering x

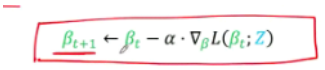
Analogy
Hiking to the bottom of a mountain range... blindfolded (or for a person who is blind)!

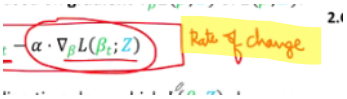
- When **minimizing** the objective function, it also is often called interchangeably the **cost function**, **loss function**, or **error function**.



Local minima and global minima

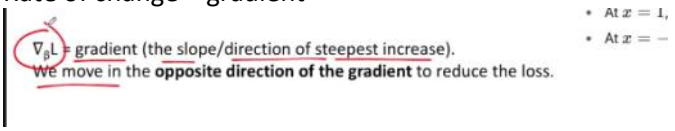






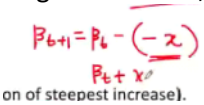
Alpha = learning rate

Rate of change = gradient

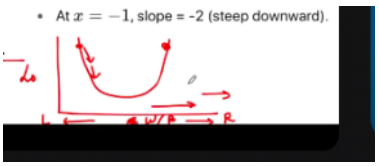


Here we have the minus , that's why opposite

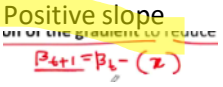
Negative means , moving towards right side

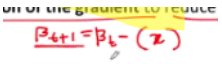


We are increasing the beta value , moving from left to right,

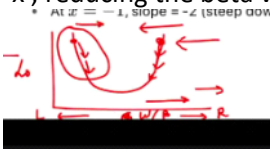


Positive slope





-x , reducing the beta value , right to left



Ml--30/08

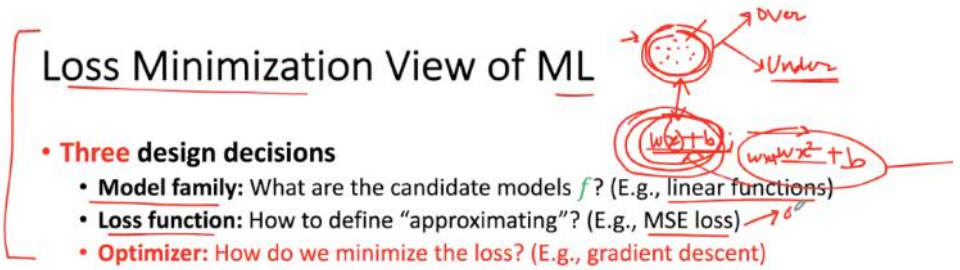
30 August 2025 14:00

Loss Minimization View of ML

- **Three design decisions**
  - **Model family:** What are the candidate models f? (E.g., linear functions)
  - **Loss function:** How to define "approximating"? (E.g., MSE loss)
  - **Optimizer:** How do we minimize the loss? (E.g., gradient descent)

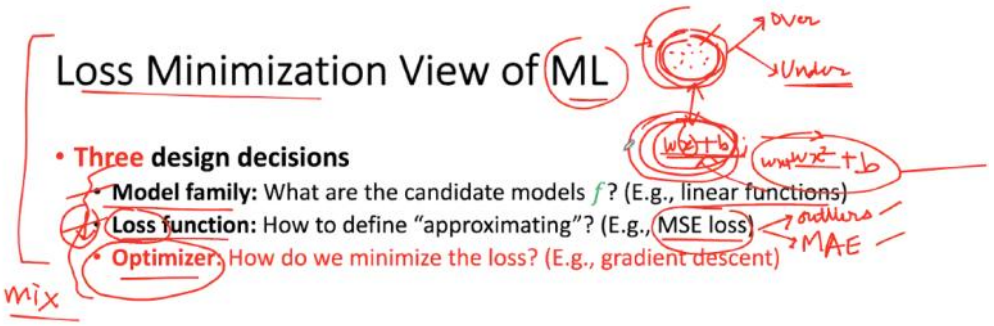
What kind of model to what kind of problem, wx+b, increase the polynomial degree, now its 1 , then we can have higher degree of polynomials

Wx^2 + b
If our dataset is very simple, there should be alignment between learn function and data, otherwise we might have underfitting(CANT LEARN COMPLEX DATA like wx+b) or overfitting(memorise the data)



MSE is for outliers , robust to outliers so select loss function accordingly

Minimise the loss: some strategy to do this , that comes in optimisation algo .



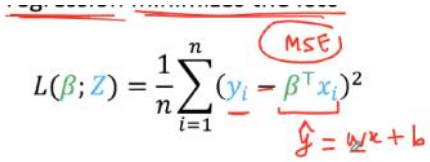
Optimization Algorithms

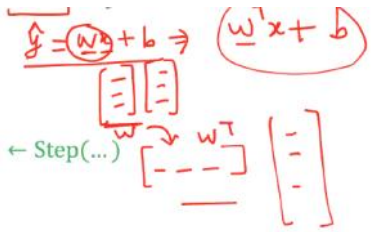
- Recall that linear regression minimizes the loss

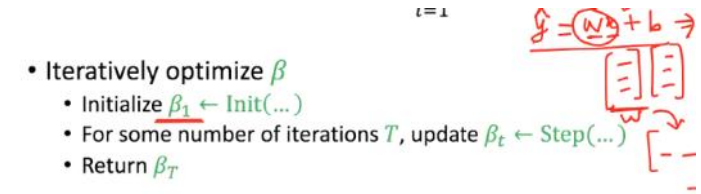
$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2$$

- Iteratively optimize β
  - Initialize $\beta_1 \leftarrow \text{Init}(\dots)$
  - For some number of iterations T, update $\beta_t \leftarrow \text{Step}(\dots)$
  - Return β_T

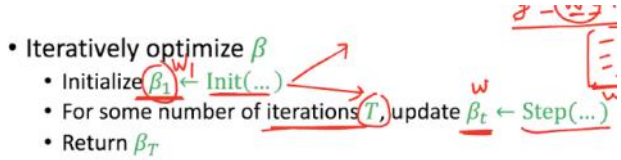
Regression --> continues values
Mse loss--> real ground truth - y predicted







Learn the beta or w



Search for minima (local search)

1. Loss minimization view of Machine Learning
2. Model selection based on data complexity (simple vs. complex data)
3. Types of loss functions:
   - MSE (Mean Square Error) - preferred when dealing with outliers
   - MAE (Mean Absolute Error) - used for simpler cases
4. Optimization algorithms to minimize loss
5. Gradient descent as an optimization method
6. Mathematical representation of MSE loss: $(Yi - W^\top \cdot X + B)^2$
7. Weight initialization for optimization algorithms

The instructor also mentioned the importance of aligning model complexity with data complexity to avoid overfitting and underfitting issues.

Loss is minimised , we get beta t , at iteration t , optimal weights

Optimization Algorithms

- **Global search**: Try random values of β and choose the best
  - I.e., β_t independent of β_{t-1}
  - Very unstructured, can take a long time (especially in high dimension d)!
- **Local search**: Start from some initial β and make local changes
  - I.e., β_t is computed based on β_{t-1}
  - What is a "local change", and how do we find good one?

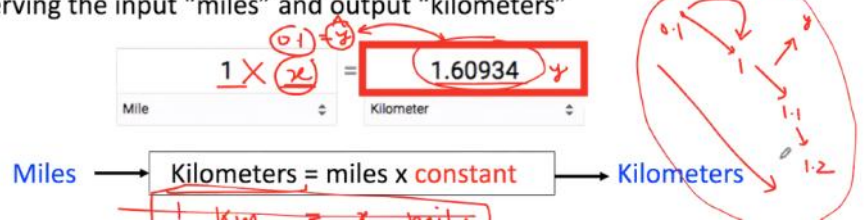
LOCAL search , next optimised weights dependent on previous weights

Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn linear model for converting kilometers to miles when only observing the input "miles" and output "kilometers"

Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn linear model for converting kilometers to miles when only observing the input "miles" and output "kilometers"



Gradient Descent – Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant → Error = Guess - Correct

- Idea: iteratively adjust constant (i.e., model parameter) to try to reduce the error

Gradient Descent Algorithms

- Approach: solve mathematical problems by updating estimates of the solution via an iterative process to "optimize" a function
  - e.g., minimize or maximize an objective function f(x) by altering x

Analogy
Hiking to the bottom of a mountain range...
blindfolded (or for a person who is blind)!

- When **minimizing** the objective function, it also is often called interchangeably the **cost function**, **loss function**, or **error function**.

He needs to compute whether uphill or downhill

- When **minimizing** the objective function, it also is often called interchangeably the **cost function**, **loss function**, or **error function**.

Strategy 2: Gradient Descent

- **Gradient descent:** Update β based on **gradient** $\nabla_\beta L(\beta; Z)$ of $L(\beta; Z)$:

$$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)$$

- **Intuition:** The gradient is the direction along which $L(\beta; Z)$ changes most quickly as a function of β
- $\alpha \in \mathbb{R}$ is a hyperparameter called the **learning rate**
  - More on this later

$\nabla_\beta L$ = gradient (the slope/direction of steepest increase).
We move in the **opposite direction of the gradient** to reduce the loss.

Cases of Gradients (1D case for clarity)

1.Gradient > 0 (positive slope)
1. Loss is increasing as β increases.
2. Update: move β **to the left** (decrease β).

2.Gradient < 0 (negative slope)
1. Loss is decreasing as β increases.
2. Update: move β **to the right** (increase β).

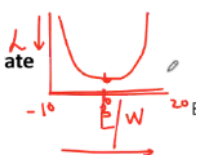
3.Gradient = 0
1. You are at a flat spot.
2. No update — algorithm stops here.

Example: If $f(x) = x^2$, then $f'(x) = 2x$.
- At $x = 1$, slope = 2 (steep upward).
- At $x = -1$, slope = -2 (steep downward).

Alpha is learning rate, deicidng the step size

$\nabla_\beta L(\beta_t; Z)$

This is gradient



$\beta_{k+1} \leftarrow \beta_k - \alpha \, (+ve)$

If gradient is positive

3.Grad
1.
2.

3.Gradient = 0

1. You are at a flat spot.
2. No update — algorithm stops here.

Global minima

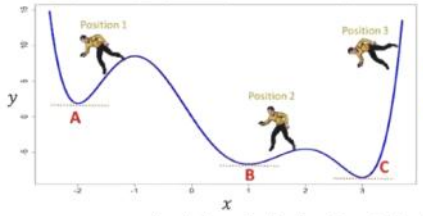
The model has converged

Derivatives== measures the rate of change of function

Example: If $f(x) = x^2$ then $f'(x) = 2x$.

Approach: Employ Calculus Concepts

- Idea: use derivatives!
  - Derivatives tells us how to change the input x to make a small change to the output f(x)
  - Functions with multiple inputs rely on a partial derivative for each input
- Gradient descent:
  - Iteratively update f(x) by moving x in small steps with the opposite sign of the derivative



Which letter is the global minimum?

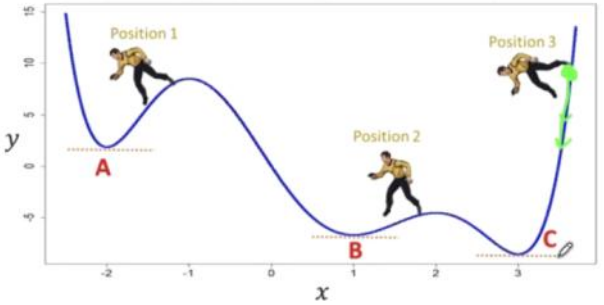
Which letter(s) are local minima?

Louis Augustin Cauchy: Compte Rendu `a l'Acad´emie des Sciences of October 18, 1847

Approach: Employ Calculus Concepts

- Idea: use derivatives!
  - Derivatives tells us how to change the input x to make a small change to the output f(x)
  - Functions with multiple inputs rely on a partial derivative for each input
- Gradient descent:
  - Iteratively update f(x) by moving x in small steps with the opposite sign of the derivative

Talking: divya Saxena



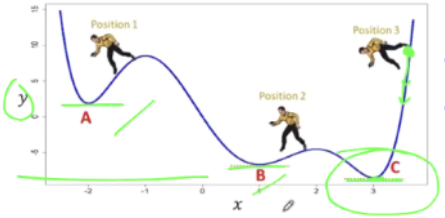
Which letter is the global minimum?

Which letter(s) are local minima?

Louis Augustin Cauchy: Compte Rendu `a l'Acad´emie des Sciences of October 18, 1847

Need to reach the optimal value

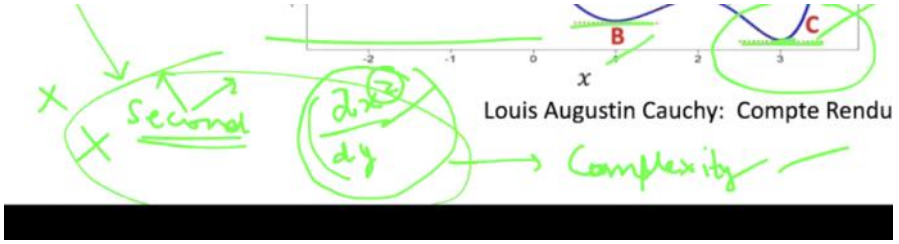
- Iteratively update f(x) by moving x in small steps with the opposite sign of the derivative



Which letter is the global minimum?

Which letter(s) are local minima?

C is global minima
a,b is local minima



Strategy 2: Gradient Descent

Talking: divya Saxe

- Choose initial value for β
- Until we reach a minimum:
  - Choose a new value for β to reduce $L(\beta; Z)$

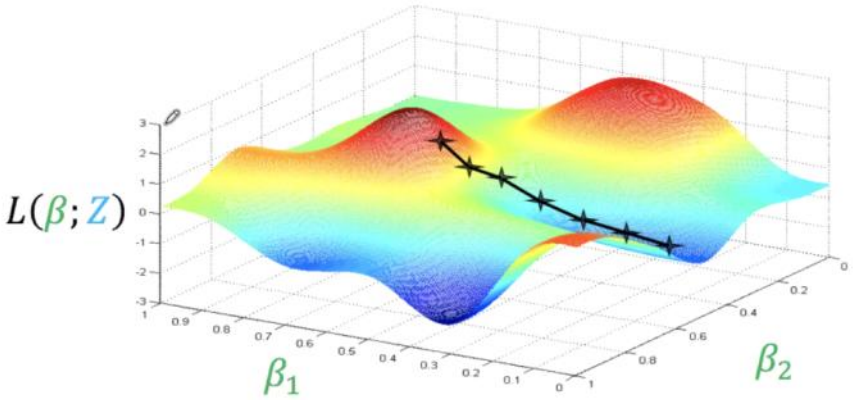
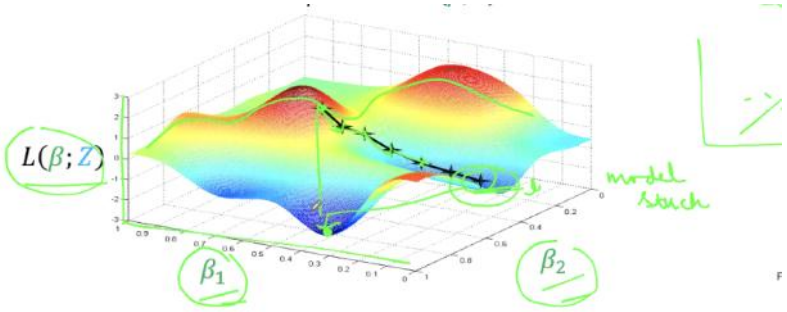


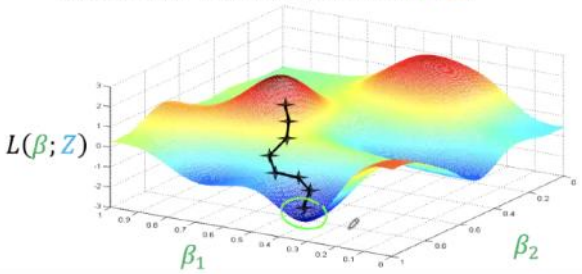
Figure by Andrew Ng

Minimise b1 abd b2
Non linear plotting
Multiple peaks and troughs



Strategy 2: Gradient Descent

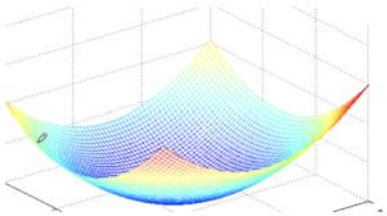
- Choose initial value for β
- Until we reach a minimum:
  - Choose a new value for β to reduce $L(\beta; Z)$



We cn do plotting for 3 variables but not for 100 variables

Strategy 2: Gradient Descent

- Choose initial value for β
- Until we reach a minimum:
  - Choose a new value for β to reduce $L(\beta; Z)$



Linear regression loss is convex, so no local minima

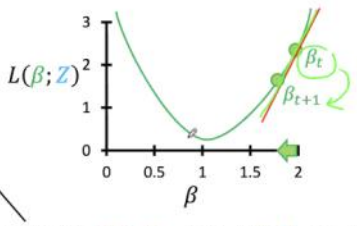
Figure by Andrew Ng

Strategy 2: Gradient Descent

- Initialize $\beta_1 = 0$
- Repeat until convergence:

$$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)$$

- For linear regression, know the gradient from strategy 1



For in-place updates $\beta \leftarrow \beta - \alpha \cdot \nabla_\beta L(\beta; Z)$, compute all components of $\nabla_\beta L(\beta; Z)$ before modifying β

Suppose our loss function is:

$$L(\beta) = (\beta - 3)^2$$

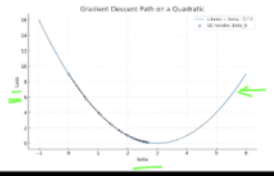
This is just a parabola (U-shape) centered at $\beta = 3$.
Our goal: find the gradient (derivative) and interpret it.

$$L(\beta) = (\beta - 3)^2$$

Differentiate with respect to β:

$$\nabla_\beta L(\beta) = 2(\beta - 3)$$

That's our gradient function.



$$\beta_{t+1} = \beta_t - \alpha \cdot \nabla L(\beta_t)$$

Let's pick:

- Start at $\beta_0 = 0$
- Learning rate $\alpha = 0.1$

Step 1:

$$\beta_1 = 0 - 0.1 \cdot (-6) = 0 + 0.6 = 0.6$$

Step 2:

$$\nabla L(0.6) = 2(0.6 - 3) = -4.8$$

$$\beta_2 = 0.6 - 0.1 \cdot (-4.8) = 0.6 + 0.48 = 1.08$$

Step 3:

$$\nabla L(1.08) = 2(1.08 - 3) = -3.84$$

$$\beta_3 = 1.08 - 0.1 \cdot (-3.84) = 1.08 + 0.384 = 1.464$$

And so on... each step gets closer to $\beta = 3$.

$\nabla_\beta L(\beta) = 2(\beta - 3)$

This is the gradient

Suppose our loss function is:

$$L(\beta) = (\beta - 3)^2$$

This is just a parabola (U-shape) centered at $\beta = 3$.
Our goal: find the gradient (derivative) and interpret it.

$$L(\beta) = (\beta - 3)^2$$

Differentiate with respect to β:

$$\nabla_\beta L(\beta) = 2(\beta - 3)$$

That's our gradient function.

$$\beta_{t+1} = \beta_t - \alpha \cdot \nabla L(\beta_t)$$

Let's pick:

- Start at $\beta_0 = 0$
- Learning rate $\alpha = 0.1$

Step 1:

$$\beta_1 = 0 - 0.1 \cdot (-6) = 0 + 0.6 = 0.6$$

Step 2:

$$\nabla L(0.6) = 2(0.6 - 3) = -4.8$$

$$\beta_2 = 0.6 - 0.1 \cdot (-4.8) = 0.6 + 0.48 = 1.08$$

Step 3:

$$\nabla L(1.08) = 2(1.08 - 3) = -3.84$$

$$\beta_3 = 1.08 - 0.1 \cdot (-3.84) = 1.08 + 0.384 = 1.464$$

And so on... each step gets closer to $\beta = 3$.

$$\beta_1 = 0 - 0.1 \cdot (-6) = 0 + 0.6 = 0.6$$

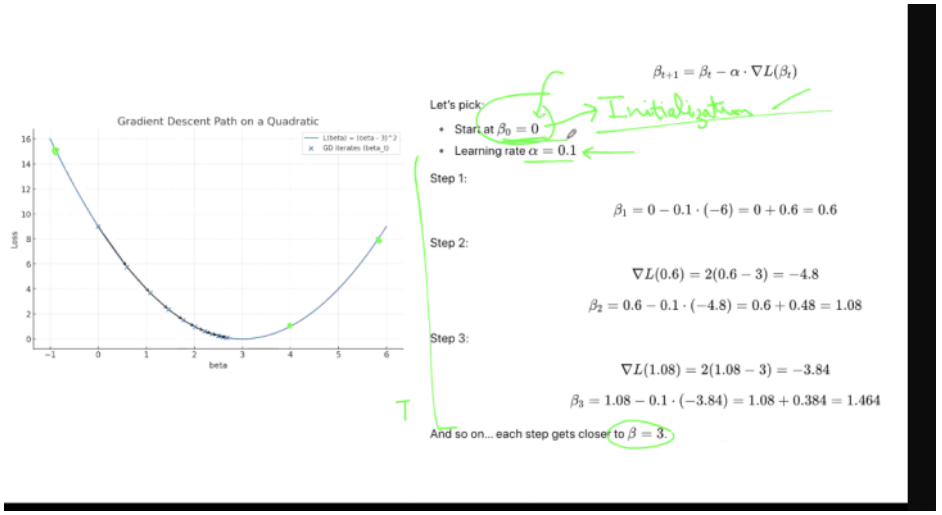
$$\nabla L(0.6) = 2(0.6 - 3) = -4.8$$

Step 3:

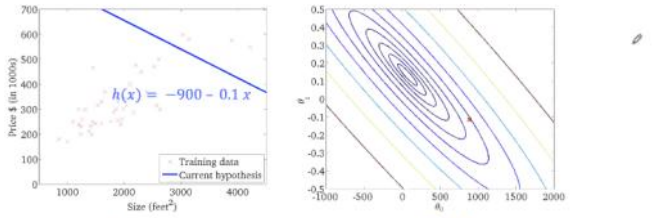
$$\nabla L(1.08) = 2(1.08 - 3) = -3.84$$

$$\beta_3 = 1.08 - 0.1 \cdot (-3.84) = 1.08 + 0.384 = 1.464$$

And so on... each step gets closer to $\beta = 3$.



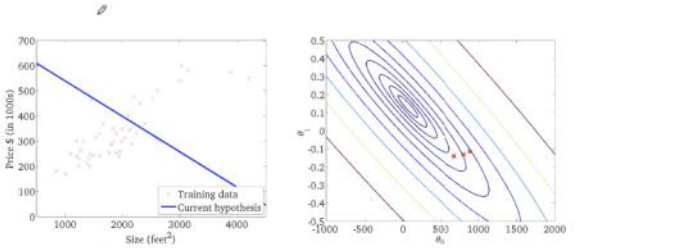
Strategy 2: Gradient Descent



$f_\beta(x)$ $L(\beta; Z)$

Slide by Andrew Ng

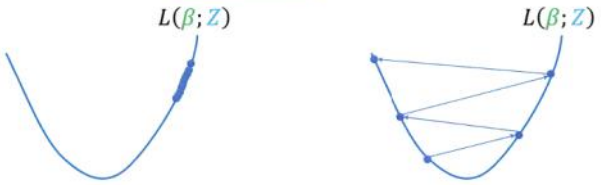
Strategy 2: Gradient Descent



$f_\beta(x)$ $L(\beta; Z)$

Slide by Andrew Ng

Choice of Learning Rate α



Problem: α too small
- $L(\beta; Z)$ decreases slowly

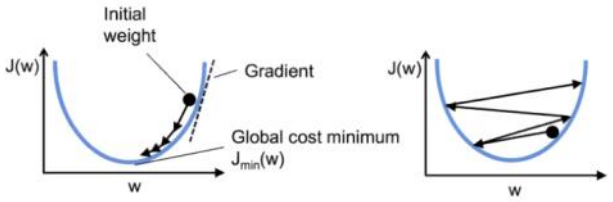
Problem: α too large
- $L(\beta; Z)$ increases!

Plot $L(\beta_t; Z_{\text{train}})$ vs. t to diagnose these problems

Choice of Learning Rate α

- α is a hyperparameter for gradient descent that we need to choose
  - Can set just based on training data
- **Rule of thumb**
  - α **too small:** Loss decreases slowly
  - α **too large:** Loss increases!
- Try rates $\alpha \in \{1.0, 0.1, 0.01, \ldots\}$ (can tune further once one works)

Gradient Descent: Influence of Learning Rate



- Learning Rate: amount new evidence is prioritized when updating weights
- What happens when learning rate is too small?
  - Convergence to good solution will be slow!
- What happens when learning rate is too large?
  - May not be able to converge to a good solution
- How to address the cons of different learning rates?
  - Gradually reduce learning rate over time

https://github.com/rasbt/python-machine-learning-book-2nd-edition/blob/master/code/ch02/ch02.ipynb

Batch Gradient Descent (BGD)

- For each step (update), use calculations over ***all training examples***
- What are strengths of this approach?
  - Does not bounce too much
- What are weaknesses of this approach?
  - Very slow or infeasible when dataset is large
- Which algorithm uses this?
  - Adaline

Stochastic Gradient Descent (SGD)

- For each step (update), use calculations from ***one training example***
- What are strengths of this approach?
  - Each iteration is fast to compute
  - Can train using huge datasets (stores one instance in memory at each iteration)
- What are weaknesses of this approach?
  - Updates will bounce a lot

In genai , image resolution size is quite bigger

Mini-batch Gradient Descent

- For each step (update), use calculations over ***subset of training examples***
- What are strengths of this approach?
  - Bounces less erratically when finding model parameters than SGD
  - Can train using huge datasets (store some instances in memory at each iteration)
- What are weaknesses of this approach?
  - Very slow or infeasible when dataset is large
- Which algorithm uses this?
  - To be explored in future classes

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap4_ann.pdf

Mini batch can also be 1

ML-31/08

31 August 2025 14:03

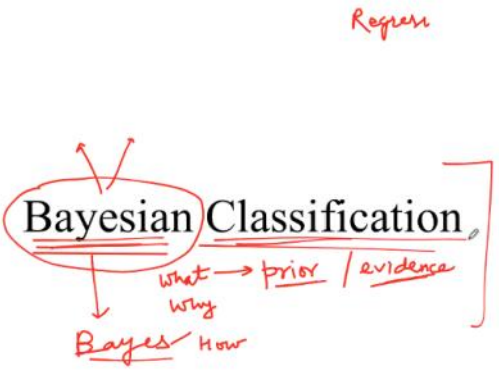
Bayesian Classification

Uncentertainty in data
Probabilitistic

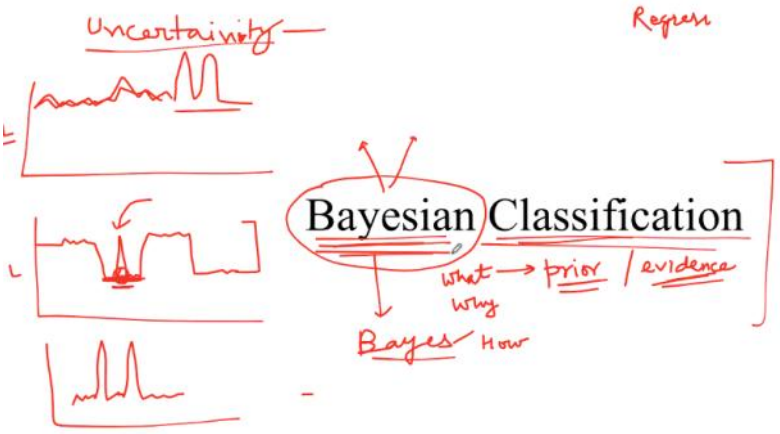
Bayes theorem
What is Bayes theorem--it is probabilistic , use to classify data based on prior knowledge/evidence

Why we need to use

How??

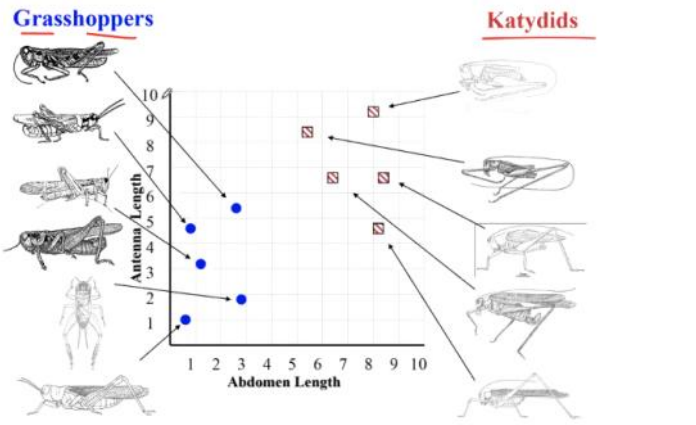


Used for predicting outcomes, using the term uncertainty from model /data perspective
For e.g.traffic data , periods of time , weekend it will be very high , weekday low

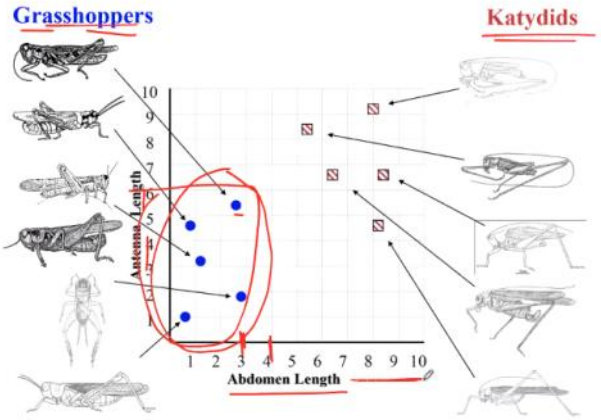


Classification is used in spam detection in email
Certain words are marked and then classified as spam

The Intuition

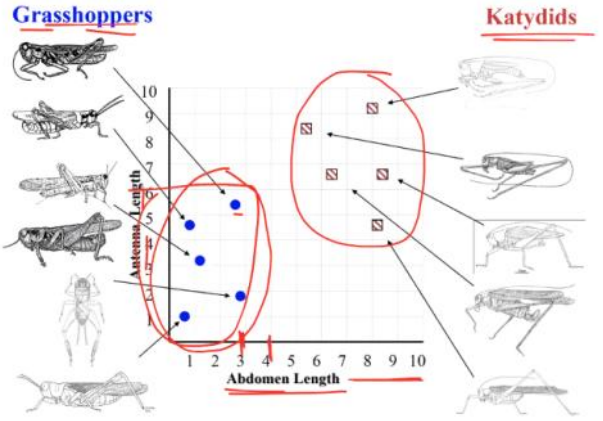


Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

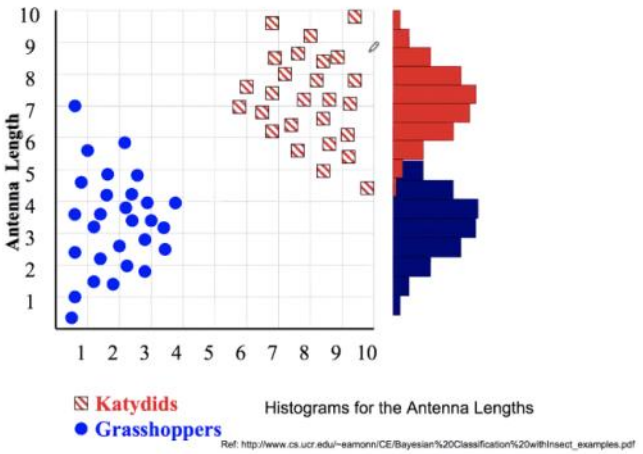


Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

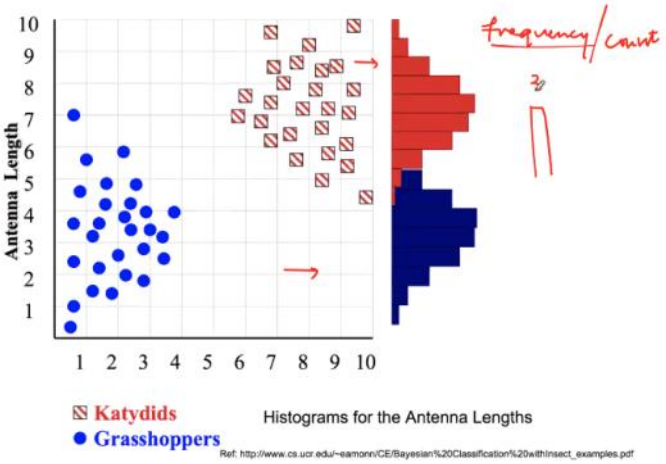
---

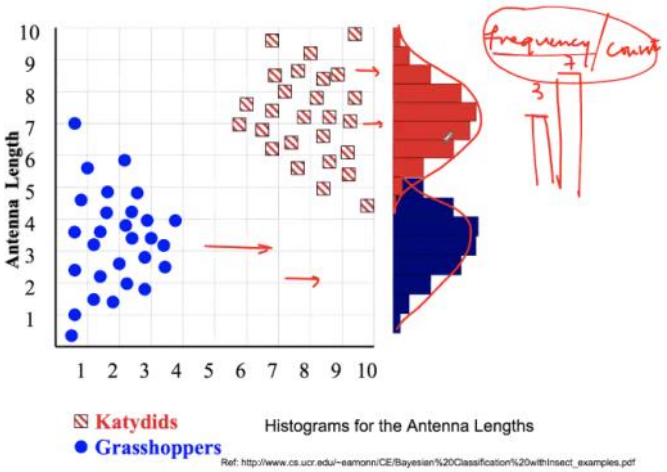


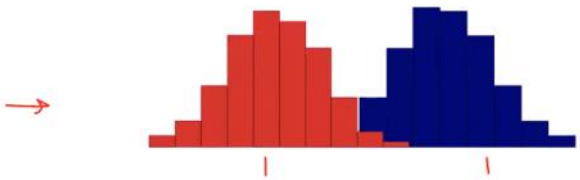
Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf



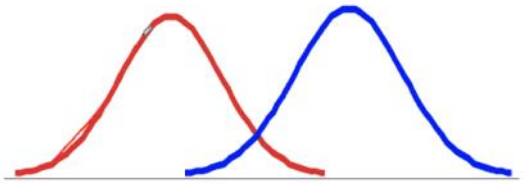
Covert this data into histogram







Summarizing the Histograms as two Normal Distributions for the ease of visualization

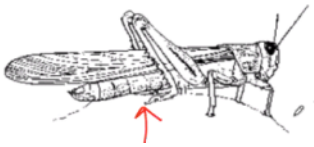


Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Which Insect?

We want to classify an insect we have found. Its antennae are 3 units long. How can we classify it?

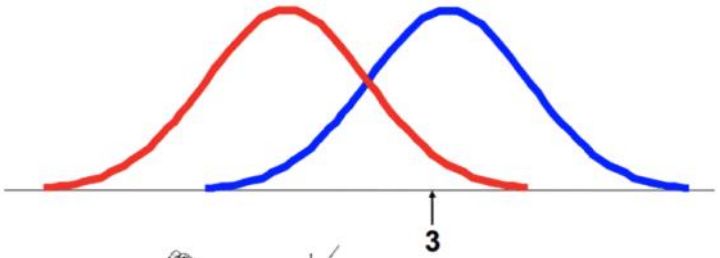
Given the distributions of antennae lengths we have seen, is it more probable that our insect is a Grasshopper or a Katydid



Antenna is 3 unit long
Now I want to know whether its grasshopper or katydid

Formal way to discuss the most probable classification:

P(Cj | d) = Probability of class Cj , given observed/evidence d

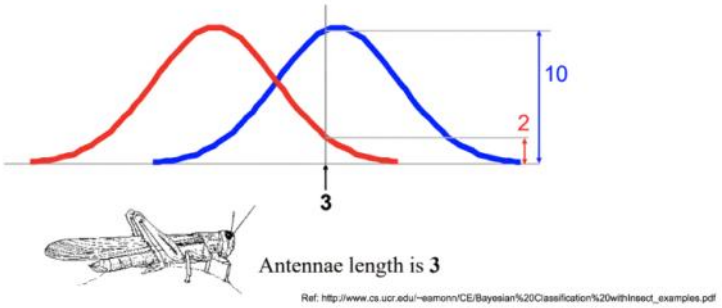


Antennae length is **3**

Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

P(Cj | d) = Probability of class Cj , given observed/evidence d

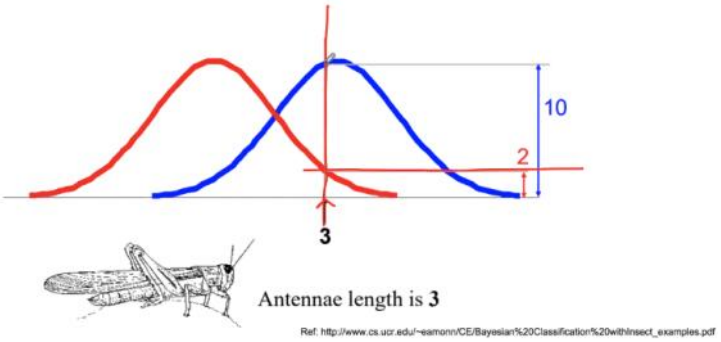
P(Grasshopper| 3) = 10 / (10 + 2) = 0.833
P(Katydid| 3) = 2 / (10 + 2) = 0.166



Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

P(Cj | d) = Probability of class Cj , given observed/evidence d

P(Grasshopper| 3) = 10 / (10 + 2) = 0.833
P(Katydid| 3) = 2 / (10 + 2) = 0.166

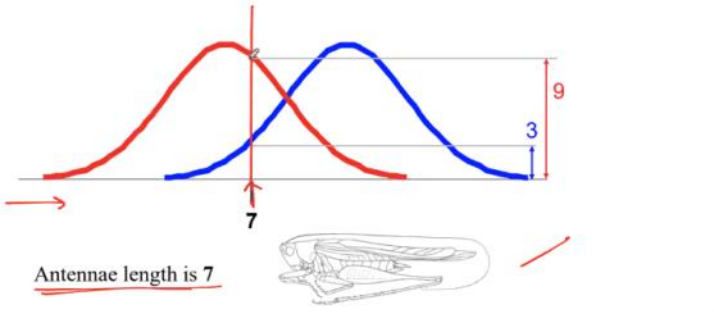


Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

3 is intersecting at 2 and 10 at red and blue resp.

P(Cj | d) = Probability of class Cj , given observed/evidence d

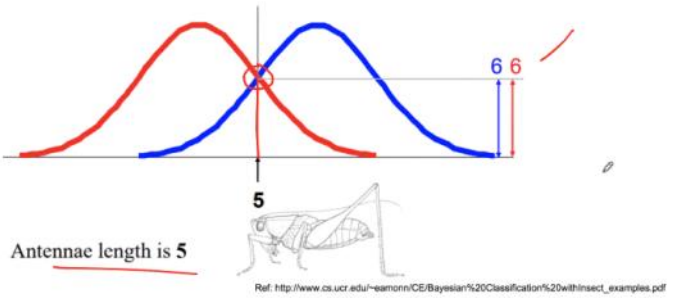
P(Grasshopper| 7) = 3 / (3 + 9) = 0.25
P(Katydid| 7) = 9 / (3 + 9) = 0.166



Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

P(Cj | d) = Probability of class Cj , given observed/evidence d

P(Grasshopper| 5) = 6/ (6 + 6) = 0.5
P(Katydid| 5) = 6/ (6 + 6) = 0.5



Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Bayesian Classification

That was a visual intuition for a simple case of the Bayes classifier, also called Naïve Bayes or Simple Bayes

Before we look into the mathematical representations, keep in mind the basic idea:

Find out the probability of the **previously unseen instance** belonging to each class, then simply pick the most probable class.

The meeting has covered the following topics:

1. Introduction to Bayesian Classification as a probabilistic method
2. What Bayes Theorem is and why it's used for handling uncertainty in data
3. Real-world applications of Bayesian Classification (like spam detection)
4. Practical example using insects (grasshoppers and katydids) to demonstrate classification based on antenna length
5. Step-by-step calculation of probabilities using the Bayes Theorem
6. How to classify new observations based on calculated probabilities
7. Handling equal probability cases in classification

The instructor has been explaining these concepts with visual aids including data distributions and histograms to help students understand the practical application of Bayesian Classification.

Bayes Theorem

Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

$p(c_j \mid d)$ = probability of instance d being in class c_j,
This is what we are trying to compute

$p(d \mid c_j)$ = probability of generating instance d given class c_j,
We can imagine that being in class c_j causes you to have feature d with some probability

$p(c_j)$ = probability of occurrence of class c_j,
This is just how frequent the class c_j is in our database

$p(d)$ = probability of instance d occurring
This can actually be ignored, since it is the same for all classes

Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Bayes Theorem

Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

$p(c_j \mid d)$ = probability of instance d being in class c_j,
This is what we are trying to compute

$p(d \mid c_j)$ = probability of generating instance d given class c_j,
We can imagine that being in class c_j causes you to have feature d with some probability

$p(c_j)$ = probability of occurrence of class c_j,
This is just how frequent the class c_j is in our database

$p(d)$ = probability of instance d occurring
This can actually be ignored, since it is the same for all classes

Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Guess the gender

Assume that we have two classes

C_1= male, C_2= female

We have a person whose sex we do not know, say "drew" or d.

Drew can be a male or a female name

Classifying drew as male or female is equivalent to asking is it more probable that drew is male or female.

I.e which is greater P(male | drew) or P(female | drew)

Drew Carey

Drew Barrymore

What is the probability of being called "*drew*" given that you are a male?

What is the probability of being a male?

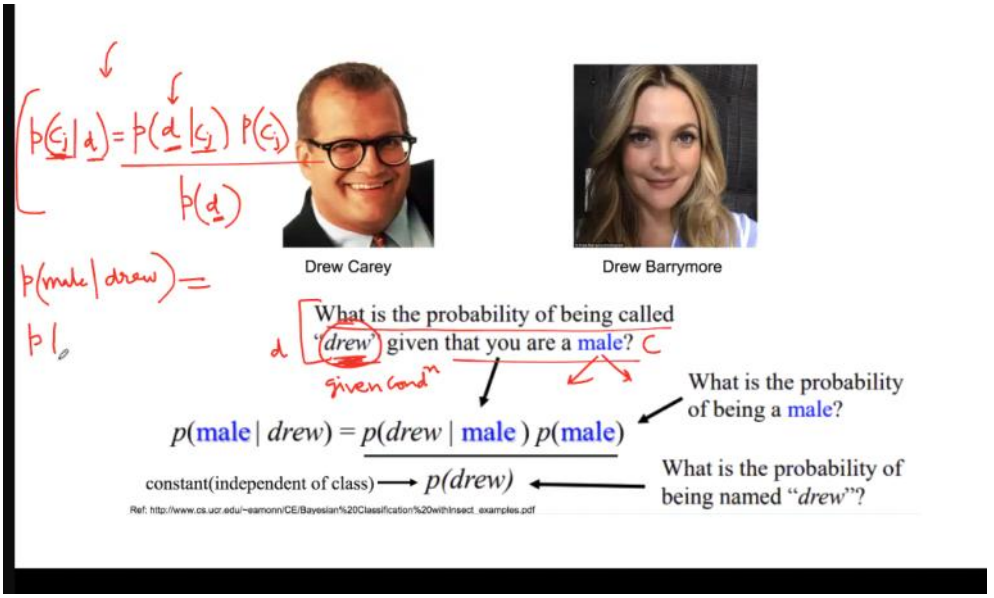
$$p(\text{male} \mid drew) = \frac{p(drew \mid \text{male})\, p(\text{male})}{p(drew)}$$

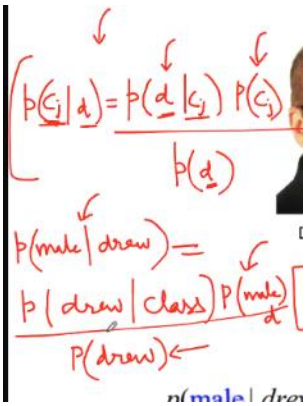
constant(independent of class)

What is the probability of being named "*drew*"?

Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Male is class
Drew is given condition, given that name is drew





The Dataset

Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.

Officer Drew

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

P(male | drew) = ((⅓) x (⅜) / (⅜)) = 0.125/(⅜)

P(female | drew) = ((⅓) x (⅜) / (⅜)) = **0.25**/(⅜)

Hence Officer Drew is more likely to be a female

Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

In the Drew name example, the instructor demonstrates Bayesian classification to determine if a person named "Drew" is more likely to be male or female.

The approach is:

1. We have a dataset with 8 instances, where the name "Drew" appears 3 times.

2. We need to calculate:
   - P(male|Drew): Probability of being male given the name is Drew
   - P(female|Drew): Probability of being female given the name is Drew

3. Using Bayes Theorem:
   P(male|Drew) = / P(Drew)

   Where:
   - P(Drew|male): Probability of having the name Drew given the person is male
   - P(male): Prior probability of being male in the dataset
   - P(Drew): Total probability of the name Drew occurring

4. From the dataset, we can see that out of 3 instances of "Drew", only 1 is male.

5. The instructor was beginning to calculate P(Drew|male) when the transcript ended, which would be 1/3 based on the data.

The example demonstrates how to apply Bayes Theorem to classify a name as more likely male or female based on prior observations in the dataset.

Officer Drew

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

P(male | drew) = ((⅓) x (⅜) / (⅜)) = 0.125/(⅜)

P(female | drew) = ((⅓) x (⅜) / (⅜)) = **0.25**/(⅜)

Hence Officer Drew is more likely to be a female

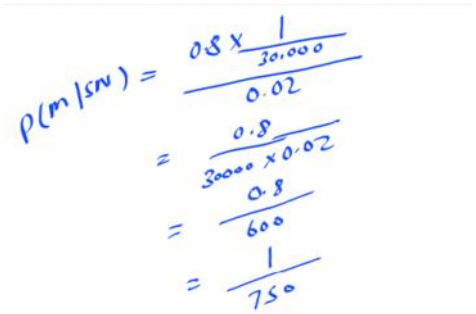
Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

Ref: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

Question: what is the probability that a patient has diseases meningitis with a stiff neck?

Given Data:

- A doctor is aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time.
- He is also aware of some more facts, which are given as follows:
  - The Known probability that a patient has meningitis disease is 1/30,000.
  - The Known probability that a patient has a stiff neck is 2%.



$$P(m|sn) = \frac{0.8 \times \frac{1}{30,000}}{0.02} = \frac{0.8}{30000 \times 0.02} = \frac{0.8}{600} = \frac{1}{750}$$

We need to compute the posterior probability that a patient has meningitis given they have a stiff neck. Use Bayes' rule.

Given (from the image / text):

- P(meningitis) = 1/30,000.
- P(stiff neck) = 2% = 0.02.
- P(stiff neck | meningitis) = 80% = 0.8.

We want P(meningitis | stiff) = P(stiff | meningitis) · P(meningitis) / P(stiff).

Plug numbers in:

P(meningitis | stiff) = 0.8 × (1/30,000) / 0.02.

From <https://sider.ai/chat>

Applications of Nearest Neighbor Methods

Applied Computing and Informatics (2016) 12, 90–108

Saudi Computer Society, King Saud University
Applied Computing and Informatics
(http://computer.org.sa)
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method

D.A. Adeniyi, Z. Wei, Y. Yongquan *

The major problem of many on-line web sites is the presentation of many choices to the client at a time; this usually results to strenuous and time consuming task in finding the right product or information on the site. In this work, we present a study of automatic web usage data mining and recommendation system based on current user behavior through his/her click stream data on the newly developed Really Simple Syndication (RSS) reader website, in order to provide relevant information to the individual without explicitly asking for it. The K-Nearest-Neighbor (KNN) classification method has been trained to be used on-line and in Real-Time to identify clients/visitors click stream data, matching it to a particular user group and recommend a tailored browsing option that meet the need of the specific user at a particular time. [...]

Saudi Computer Society, King Saud University
Applied Computing and Informatics
(http://computer.org.sa)
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method

D.A. Adeniyi, Z. Wei, Y. Yongquan *

The major problem of many on-line web sites is the presentation of many choices to the client at a time; this usually results to strenuous and time consuming task in finding the right product or information on the site. In this work, we present a study of automatic web usage data mining and recommendation system based on current user behavior through his/her click stream data on the newly developed Really Simple Syndication (RSS) reader website, in order to provide relevant information to the individual without explicitly asking for it. The K-Nearest-Neighbor (KNN) classification method has been trained to be used on-line and in Real-Time to identify clients/visitors click stream data, matching it to a particular user group and recommend a tailored browsing option that meet the need of the specific user at a particular time. [...]

Contents lists available at ScienceDirect
Journal of Cleaner Production
journal homepage: www.elsevier.com/locate/jclepro

Remaining useful life estimation of lithium-ion cells based on k-nearest neighbor regression with differential evolution optimization

Yapeng Zhou *, Miaohua Huang *, Michael Pecht b

Remaining useful life estimation is of great importance to customers who use battery-powered products. This paper develops a remaining useful life estimation model based on k-nearest neighbor regression by incorporating data from all the cells in a battery pack. A differential evolution technique is employed to optimize the parameters in the estimation model. In this approach, remaining useful life is estimated from a weighted average of the useful life of several nearest cells that share a similar degradation trend to the cell whose remaining useful life needs to be estimated. The developed method obtains a remaining useful life estimation result with average error of 9 cycles, and the best estimation only has an error of 2 cycles. [...]

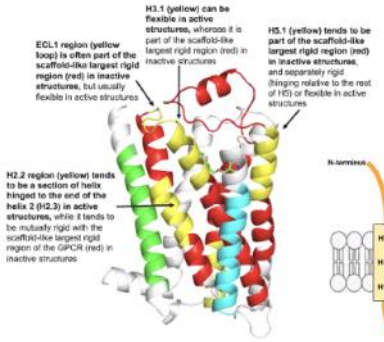
biomolecules

MDPI

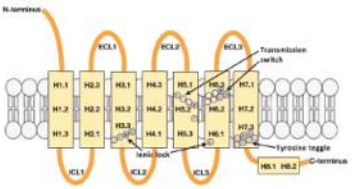
Article

Machine Learning to Identify Flexibility Signatures of Class A GPCR Inhibition

Joseph Bemister-Buxngton [1], Alex J. Wolf [1], Sebastian Raschka [1,2,*] and Leslie A. Kuhn [1,3,*]



We show that machine learning can pinpoint features distinguishing inactive from active states in proteins, in particular identifying key ligand binding site flexibility transitions in GPCRs that are triggered by biologically active ligands. [...] However, considering the flexible versus rigid state identified by graph-theoretic ProFlex rigidity analysis for each helix and loop segment with the ligand removed, followed by feature selection and **k-nearest neighbor classification**, was sufficient to identify four segments surrounding the ligand binding site whose flexibility/rigidity accurately predicts whether a GPCR is in an active or inactive state ...

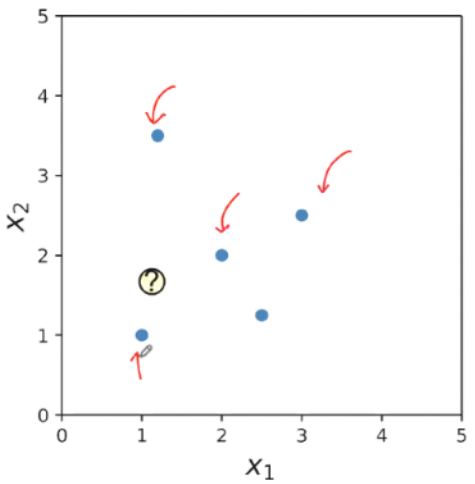


1-Nearest Neighbor

k=1

1-Nearest Neighbor

Task: predict the target / label of a new data point



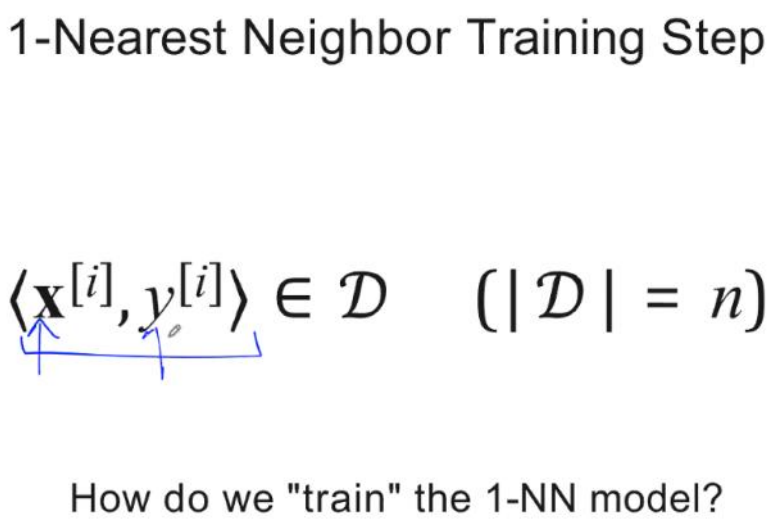
The topics discussed so far in the meeting include:

1. Bayesian Classification - a probabilistic method used for handling uncertainty in data
2. Bayes Theorem - the mathematical foundation of Bayesian classification
3. Applications of Bayesian Classification like spam detection in emails
4. Examples of Bayesian Classification with:
   - Insects (grasshoppers and katydids) classification based on antenna length

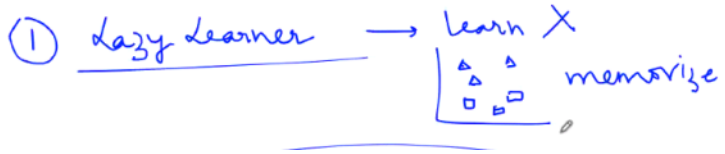
- Determining gender based on names (the Drew example)
- Medical diagnosis example (meningitis prediction based on stiff neck)

5. K-Nearest Neighbor (KNN) method - introduction and applications including:
   - Web data mining and recommendations
   - Image recognition for handwritten digits
   - Lithium cell life estimation
   - Protein classification

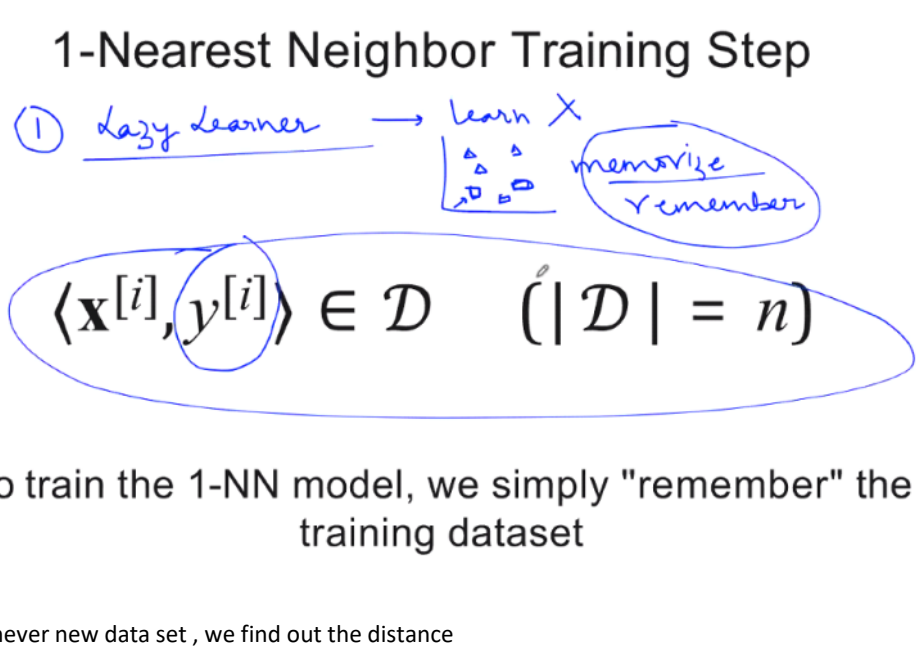
The instructor was explaining how these algorithms work through practical examples and probability calculations.



This algo is also known as lazy learner



Labelled data and memorize



Whenever new data set , we find out the distance

1-Nearest Neighbor Prediction Step

Given: $\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D} \quad (|\mathcal{D}| = n)$

$\langle \mathbf{x}^{[q]}, ??? \rangle$

Predict: $f(\mathbf{x}^{[q]})$

Algorithm:

closest_point := None

closest_distance := ∞

query point

- for $i = 1, \ldots, n$:
  - current_distance := $d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if current_distance < closest_distance:
    - closest_distance := current_distance

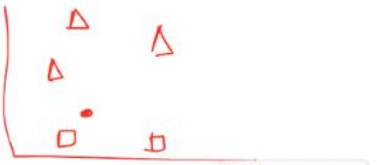
1-Nearest Neighbor Prediction Step

Given: $\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D} \quad (|\mathcal{D}| = n)$

$\langle \mathbf{x}^{[q]}, ??? \rangle$

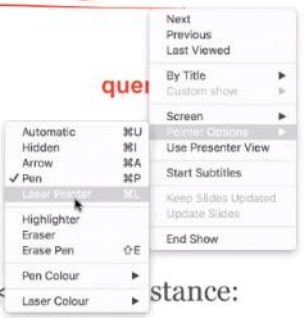
Predict: $f(\mathbf{x}^{[q]})$

Algorithm:



closest_point := None

closest_distance := ∞



- for $i = 1, \ldots, n$:
  - current_distance :=
  - if current_distance < stance:
    - closest_distance := current_distance
    - closest_point := $\mathbf{x}^{[i]}$
- return f(closest_point)

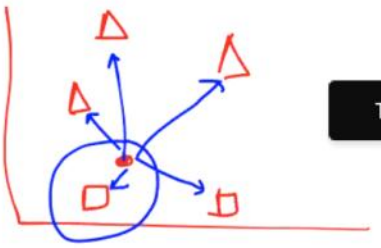
1-Nearest Neighbor Prediction Step

Given: $\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D} \quad (|\mathcal{D}| = n)$

$\langle \mathbf{x}^{[q]}, ??? \rangle$

Predict: $f(\mathbf{x}^{[q]})$

Algorithm:



Talking:

query point

- closest_point := None
- closest_distance := ∞
- for $i = 1, \ldots, n$:
  - current_distance := $d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if current_distance < closest_distance:
    - closest_distance := current_distance
    - closest_point := $\mathbf{x}^{[i]}$
- return f(closest_point)

Closest point is none

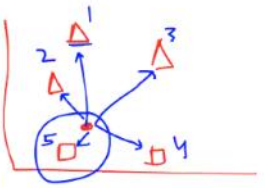
1-Nearest Neighbor Prediction Step

Given: $\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D} \quad (|\mathcal{D}| = n)$

$\langle \mathbf{x}^{[q]}, ??? \rangle$

Predict: $f(\mathbf{x}^{[q]})$

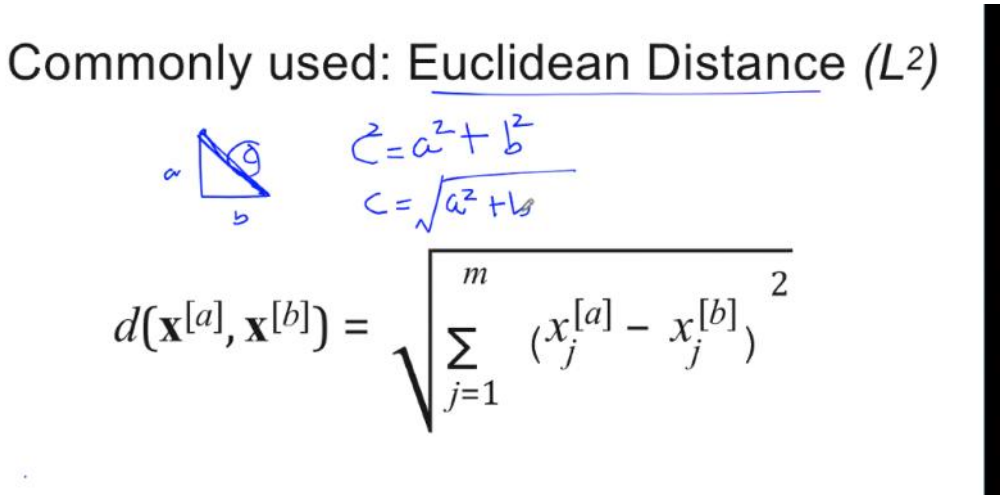
Algorithm:

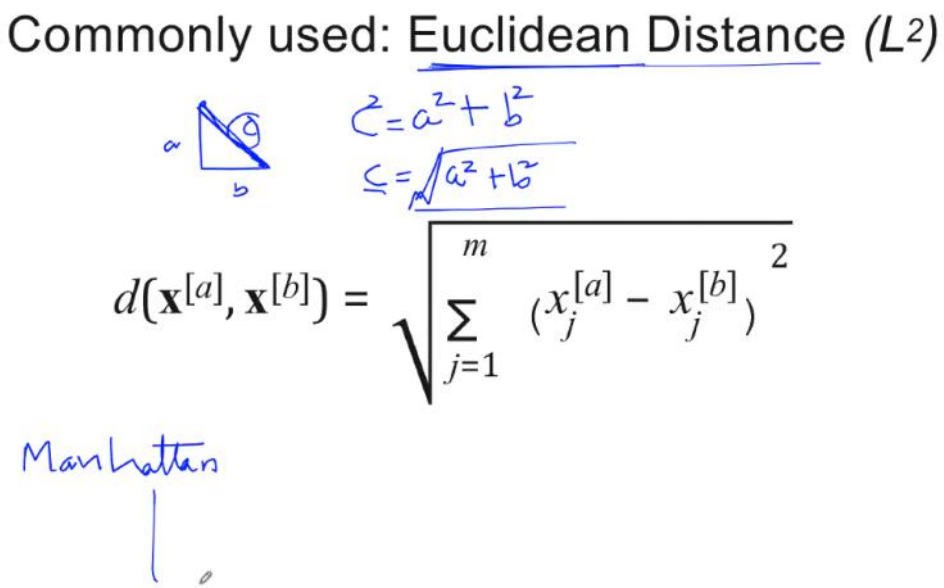


- closest_point := None
- closest_distance := ∞

query point

- for $i = 1, \ldots, n$:
  - current_distance := $d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$ new
  - if current_distance < closest_distance:
    - closest_distance := current_distance
    - closest_point := $\mathbf{x}^{[i]}$
- return f(closest_point)





K NEAREST NEIGHBOR TOPIC

K Nearest Neighbor (KNN) is a classification and regression method that:

- Is considered a "lazy learner" because it doesn't train a model but simply memorizes the training data
- Works by finding the K closest data points to a new sample and assigning the most common class among those neighbors

The basic steps of KNN:
1. Store all the training data points and their labels
2. When a new data point arrives, calculate the distance between it and all existing points
3. Find the K nearest neighbors (points with smallest distances)
4. Assign the class that appears most frequently among these K neighbors

Key points about KNN:
- Can be used for both classification and regression problems
- Is very fast and provides real-time solutions
- Has applications in web data mining, image recognition, lithium cell life estimation, and protein classification
- Distance calculation typically uses Euclidean distance (Pythagoras theorem) or Manhattan distance
- K is usually an odd number (1, 3, 5, etc.) to avoid ties

The instructor demonstrated a simple example with K=1, where a new data point is classified based on its single closest neighbor.

Nearest Neighbor Decision Boundary

How we will lern this decision boundary

Teaching assistant

31 August 2025 15:00

Teaching Assistant

- **Name:** Jacob Jojy
- **Email ID:** jacob.j@futurense.com

Applications of Nearest Neighbor Methods