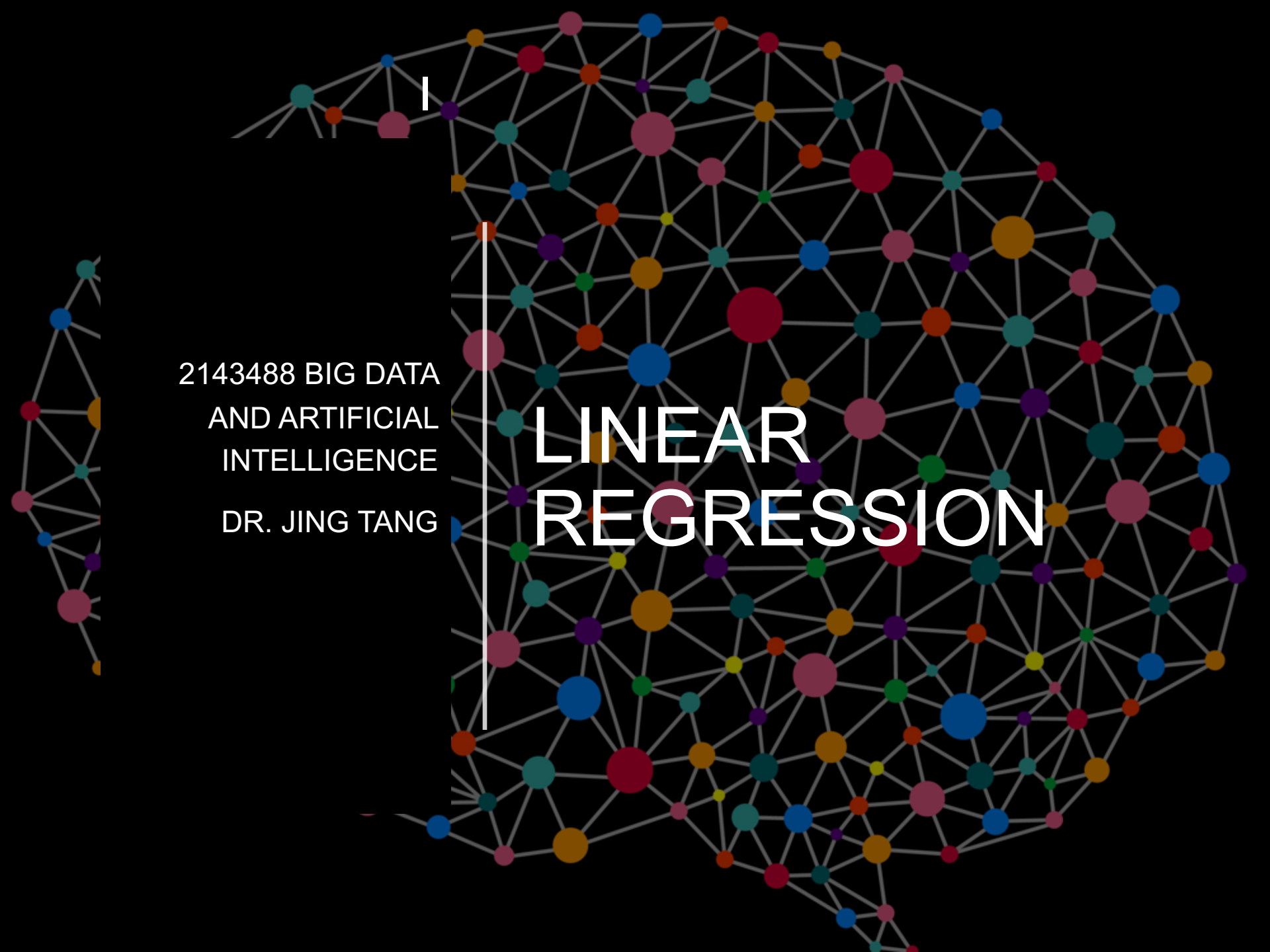


I

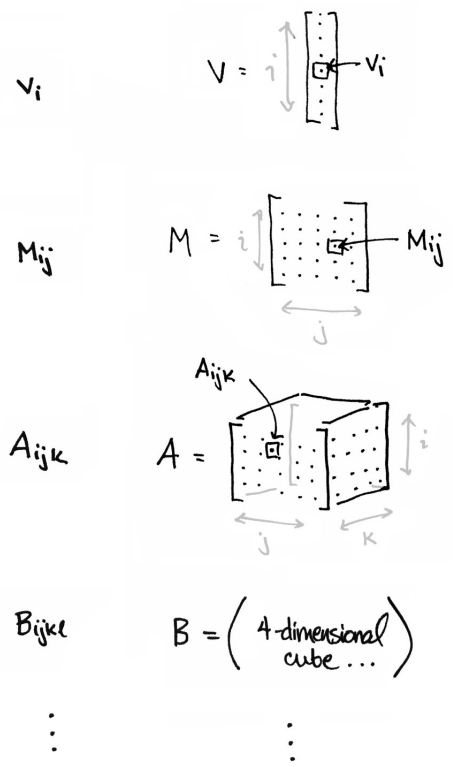
2143488 BIG DATA AND ARTIFICIAL INTELLIGENCE

DR. JING TANG

LINEAR REGRESSION

SCALAR, VECTOR, MATRIX, TENSOR

- **Scalar**: A single base data item. i.e., 2
- **Vector**: A one-dimensional array (essentially a list) of data items. i.e., [2,3,4,5] (access items using a *zero-based index*).
- **Matrix**: An array of two dimensions (essentially a $m \times n$ table) of data items.
- **Tensor**: A generalized matrix with a rank (num. of dimensions) >= 0.



3-D: numpy. array([[[1,2],[3,4]],[[5,6],[7,8]]])

MATRIX MULTIPLICATION

ELEMENT-BY-ELEMENT

- a=numpy.array([[1,2,3],[4,5,6])
- b=numpy.array([[1,2,3],[4,5,6])
- print(a*b)
- [[1 4 9]

  [16 25 36]]

DOT PRODUCT

- a=numpy.array([[1,2,3],[4,5,6])
- b=numpy.array([[1,2,3],[3,4,5],[5,6,7])
- print(a.dot(b))
- [[22 28 34]

  [49 64 79]]

REGRESSION

- In regression the output is **continuous**
  - Function Approximation – attractive simple expression
  - ML borrows it from statistics
- Many models could be used – Simplest is linear regression
  - Fit data with the best **hyper-plane** which "goes through" the points
  - For each point the differences between the predicted point and the actual observation is the **residue**

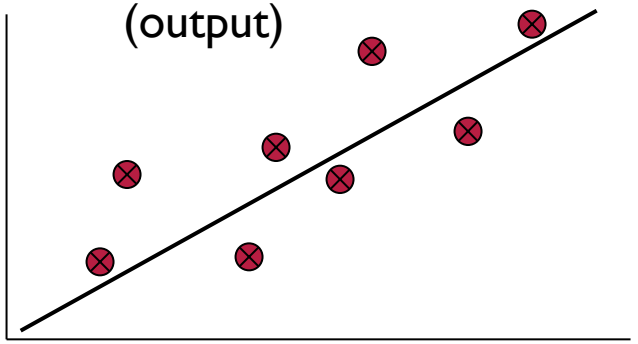
Regression → continuous dataset
→ or discrete y → very big range of number

ordinal data rather than discrete
↳ ไม่รู้ค่าห่างกันแต่ไหน

discrete :
↳ รู้ค่าห่างกันแต่ไหน

↳ If you don't have y or y is not numerical, you can't do linear regression

y – dependent variable (output)



x – independent variable (input)

You might have $x_1, x_2, x_3 \ldots$

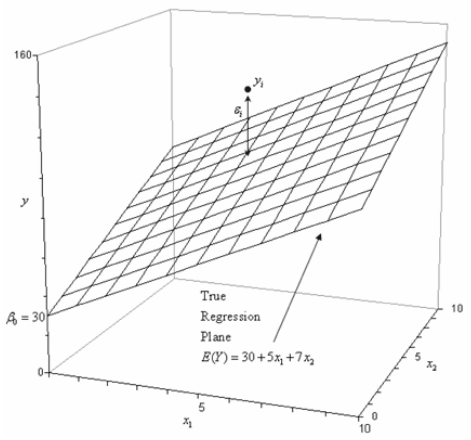
- can't visualize using a graph
- can visualize relation of x's and y in a hyperplane

one x

multiple x

SIMPLE VS. MULTIPLE LINEAR REGRESSION

- For now, assume just one (input) **independent** variable x, and one (output) **dependent** variable y
- Simple: with a single feature as a predictor
  - $y = \beta_0 + \beta_1 x + \varepsilon$
  - Shown as a **line** in a coordinate xy-plane
- Multiple: with more than one feature
  - $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon$
  - Shown as a hyperplane ($n - 1$ dimensions) in a n multidimensional space.



$$y = \vec{x}\vec{\beta} + \varepsilon,$$
$$\vec{x} = [1, x_1, x_2, \dots],$$
$$\vec{\beta} = [\beta_0, \beta_1, \beta_2, \dots]^T$$

SIMPLE LINEAR REGRESSION

- Which line should we use?
  - Choose an objective function
  - For simple linear regression we choose **Mean Squared Error (MSE)**
  - Thus, find the line which **minimizes** MSE (e.g., least squares)

not mean absolute error (MAE) - it MAE

↳ most common

$$\frac{\sum(predicted_i - actual_i)^2}{n} = \frac{\sum(residue_i)^2}{n}$$

$$\frac{\sum(\hat{y}_i - yi)2}{n} = \frac{\sum(\varepsilon_i)^2}{n}$$

HOW DO WE "LEARN" PARAMETERS (1)

- For the 2D problem (line) there are coefficients for the bias and the independent variable (*y*-intercept and slope)

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

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

$$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \qquad \beta_0 = \bar{y} - \beta_1 \bar{x}$$

HOW DO WE "LEARN" PARAMETERS (2)

- In matrix form:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$Y = X\hat{\beta}$$

$$(X^T X)^{-1} X^T Y = (X^T X)^{-1} X^T X \hat{\beta}$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y \quad \rightarrow \text{calculate stuff}$$

$$var(\hat{\beta}) = \begin{bmatrix} \sigma^2(\beta_0) & \sigma(\beta_0, \beta_1) \\ \sigma(\beta_1, \beta_0) & \sigma^2(\beta_1) \end{bmatrix} = \sigma^2 (X^T X)^{-1}$$

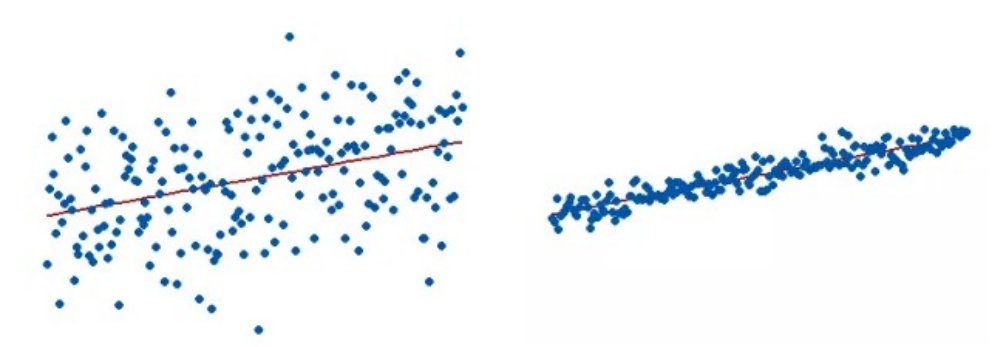
Evaluate your model

R-SQUARED AND THE GOODNESS-OF-FIT

- $R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} = 1 - \frac{\sum(y-\hat{y})^2}{\sum(y-\bar{y})^2}$ ($\hat{y}$: expected)
- R^2 is always between 0 (worse) and 1 (best)
- R^2 does not a unique indicator for GOF
  - R^2 is always low in fields as human behavior studies
  - Residuals may still have patterns when R^2 is high

- In different models, R^2 may be different.
  - physics ≈ 90%.
  - social sci ≈ 35%.

Want R^2 as big as possible



Discipline	r meaningful if	R_2 meaningful if
Physics	r < -0.95 or 0.95 < r	$0.9 < R^2$
Chemistry	r < -0.9 or 0.9 < r	$0.8 < R^2$
Biology	r < -0.7 or 0.7 < r	$0.5 < R^2$
Social Sciences	r < -0.6 or 0.6 < r	$0.35 < R^2$

check whether the coefficient is important or not

put variables in a similar range, but doesn't have to be exactly the same. Ex: [0, 10], [-3, 3]

COEFFICIENTS AND T-TEST

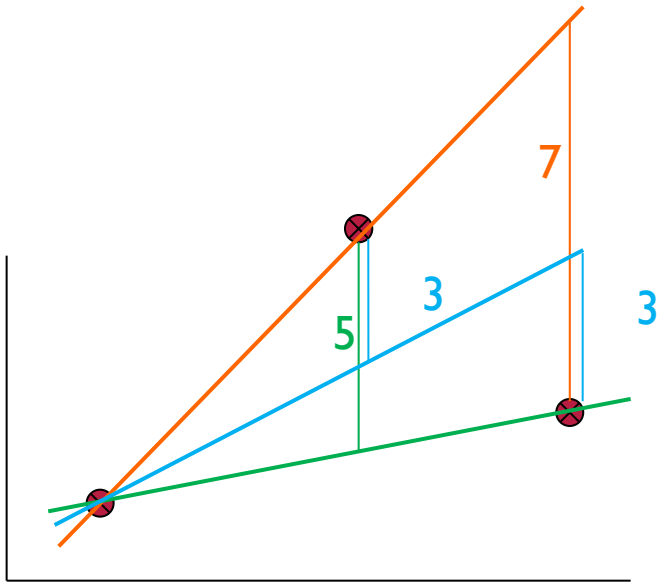
- One nice advantage of linear regression models (and linear classification) is the potential to look at the **coefficients to give insight into** which input variables are most important in predicting the output
- The variables with the largest magnitude have the highest correlation with the output
  - A large positive coefficient implies that the output will increase when this input is increased (positively correlated)
  - A large negative coefficient implies that the output will decrease when this input is increased (negatively correlated)
  - A small or 0 coefficient suggests that the input is uncorrelated with the output (at least at the 1st order)
- Linear regression can be used to find best "**indicators**"
- However, be careful not to confuse correlation with causality

· Linear Regression prefers MSE over MAE
· MSE punishes when data is located far from the regression line

Mean-squared error

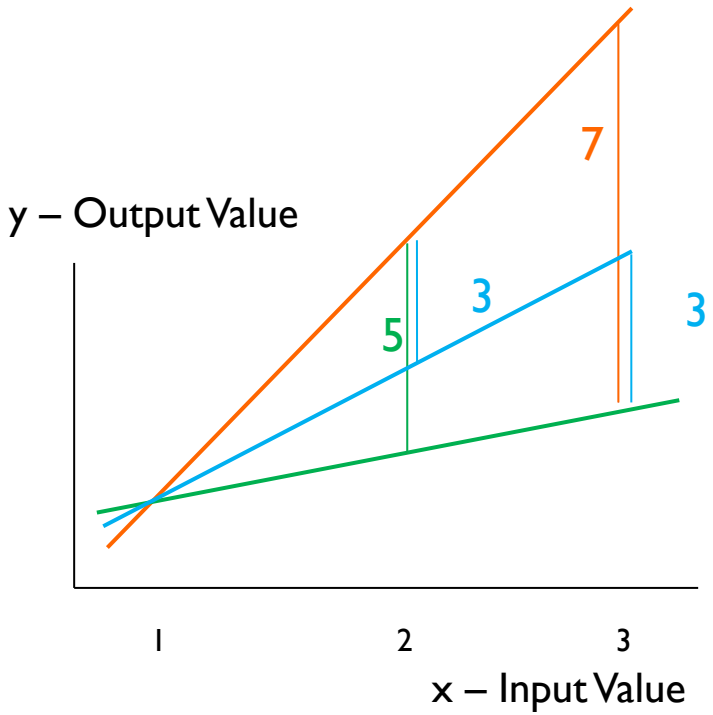
MSE AND LINEAR REGRESSION

- MSE chooses to square the difference of the predicted vs actual. Why square?
  - Don't want residues to cancel each other
  - Could use absolute or other distances to solve problem
    - $MAE = \frac{\sum |\hat{y}_i - y_i|}{n}$
  - MSE leads to a parabolic error surface which is good for **gradient descent**
- Which line would least squares choose?
  - There is always one “best” fit



SSE AND LINEAR REGRESSION GENERALIZATION

- In generalization all *x* values map to a *y* value on **the chosen regression line**



FITTING A LINEAR MODEL VIA NORMAL EQUATIONS VS. GRADIENT DESCENT

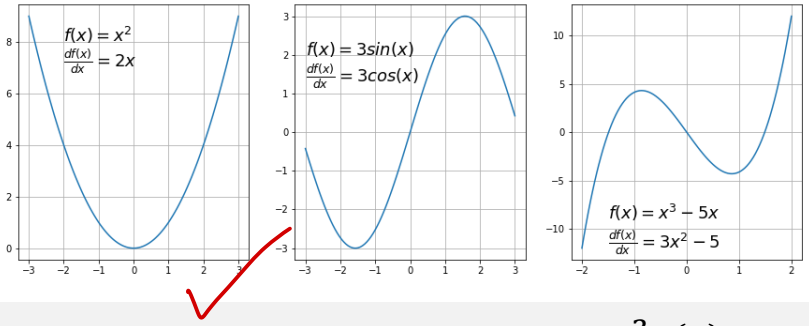
- Normal Equations:
  - $X_{M\times N}$ Applying in big data
    - $M = rows\ of\ observations > 1M$
    - $N = num.of\ predictors > 1K$
  - $X^T X\ is\ a\ M\times M\ matrix, it\ is\ too\ expensive\ to\ invert\ it$
    Gotta inverse the matrix, too big, machine learning model too slow
- Gradient Descent (GD):
  Don't directly calculate:
  - an iterative first-order optimization algorithm used to find a local minimum/maximum of a given function
  - commonly used in ML/DL to minimize a cost/loss function (e.g., in a linear regression)

GRADIENT DESCENT 1

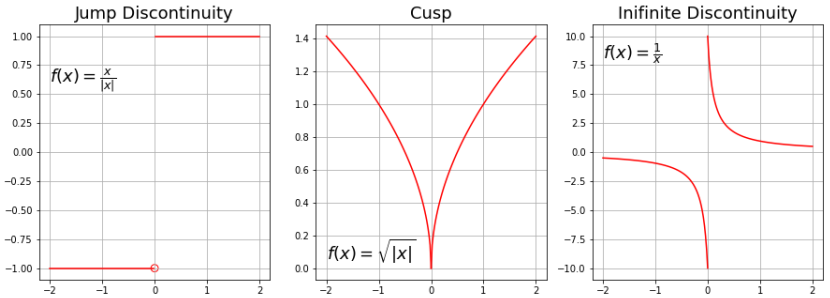
needs to be convex function to be able to use gradient descent

- It only works for **differentiable** and **convex** functions

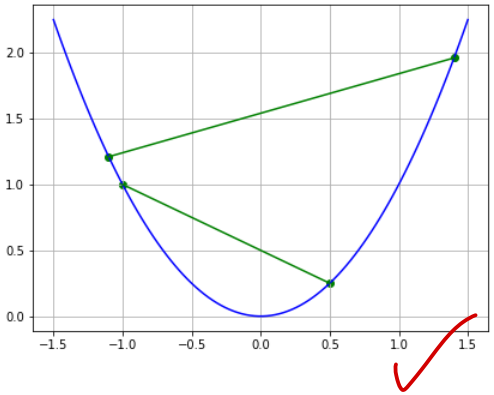
differentiable



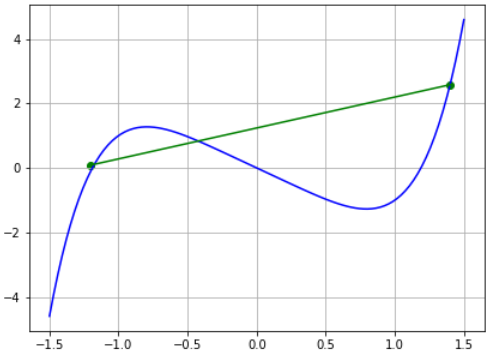
non-differentiable



convex $\frac{d^2 f(x)}{dx^2} > 0$



non-convex



GRADIENT

- a slope of a curve at a given point in a specified direction
- Univariable function: **first derivative at a selected point**
- Multivariable function: **a vector of derivatives at a selected point**
  - $\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$

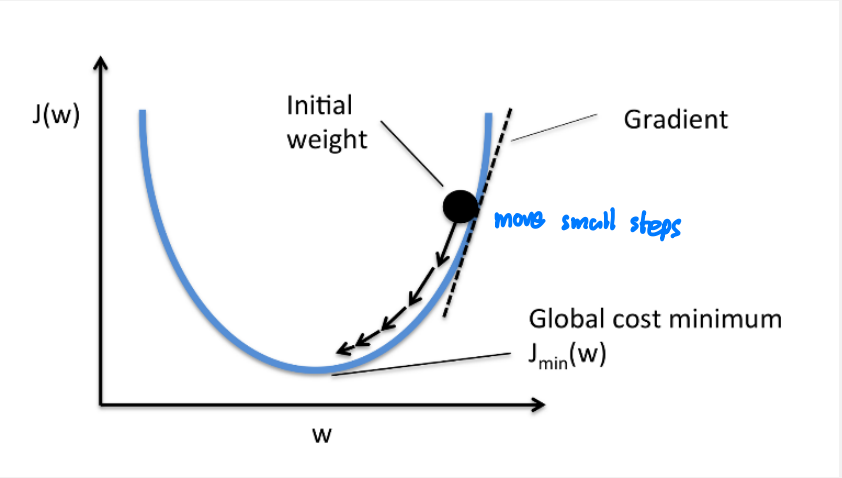
GRADIENT DESCENT 2

- Save time for calculations
- Cost function:
- "MSE" $J(\omega) = \frac{1}{2n}\sum(X\omega - y)^2$
- Repeat until converge {

learning rate (hyper parameter)

$$\omega_j := \omega_j - \eta \nabla J(\omega)$$
$$= \omega_j - \frac{\eta}{n}\sum(X\omega - y)x_j$$

}

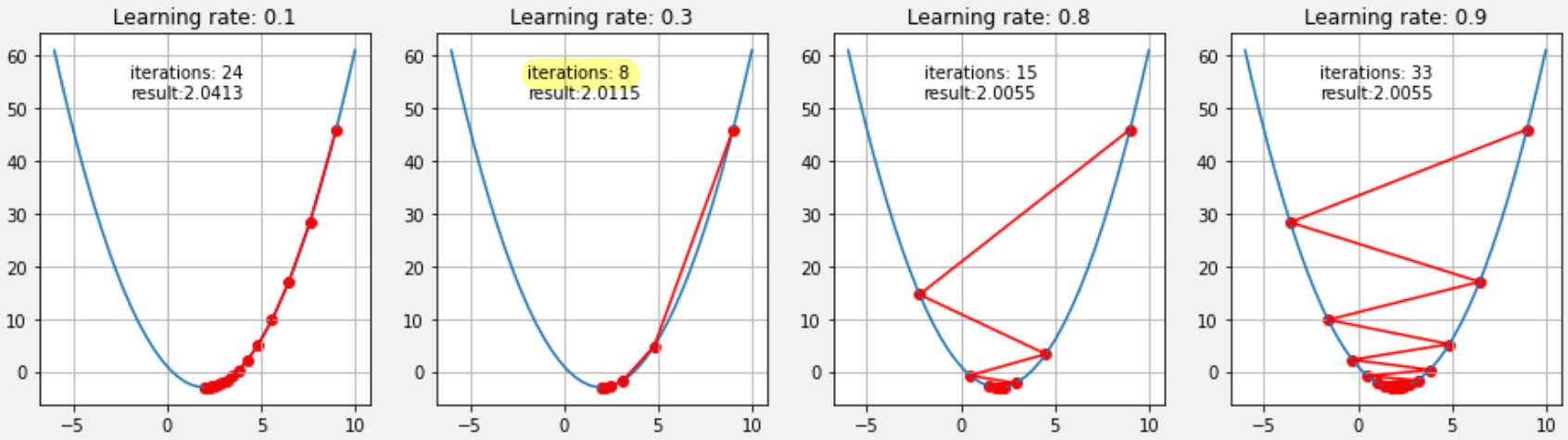


$\eta = learning\ rate$

- By trial and error → try to find the best answer
  small learning rate, small movement
- can't be too big or too small, you can adjust learning rate

GRADIENT DESCENT 3

- The small η the longer GD converges, or may reach maximum iteration before reaching the optimum point
- If η is too big the algorithm may not converge to the optimal point (jump around) or even to diverge completely.



STOCHASTIC GRADIENT DESCENT

- GD is quite costly for very large training datasets too.
  - Consider a dataset with 10,000 records and 10 features.
  - The sum of squared residuals consists of 10000 terms. To compute the derivative of this function with respect to each of the features, so it needs 10000 * 10 = 100,000 computations per iteration.
  - If it takes 1000 iterations, in effect we have 100,000 * 1000 = 100000000 computations to complete the algorithm.
- Stochastic Gradient Descent (SGD):
  - "Stochastic" means random
  - Randomly shuffle and select samples to calculate derivatives
  - It may go "zig-zag" if the cot surface is visualized in a 2D space

LINEAR REGRESSION ASSUMPTIONS

most important for LR

- **Linear Relationship** between y and x_i
  - Scatter plot

1 x for 1 graph, if there's no linear relationship between x_i and y, throw away

- **Multivariate Normality**: Residuals are normally distributed
  - Histogram / Q-Q plot
- **No Multicollinearity**: All x_i are independent with each other

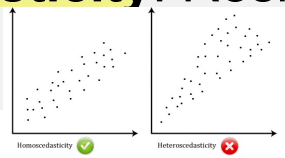
no relationship

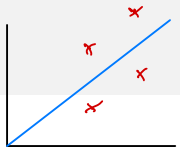
  - Correlation matrix / Variance Inflation Factor (VIF)
- **No Auto-correlation**: y_{j+1} is independent from y_i
  - Scatter plot / Durbin-Watson test

$GDP_t = GDP_{t-1}$

- has autocorrelation: can't do this in linear regression
- in time-series: OK can use

- **Homoscedasticity**: Residuals are equal across the regression line

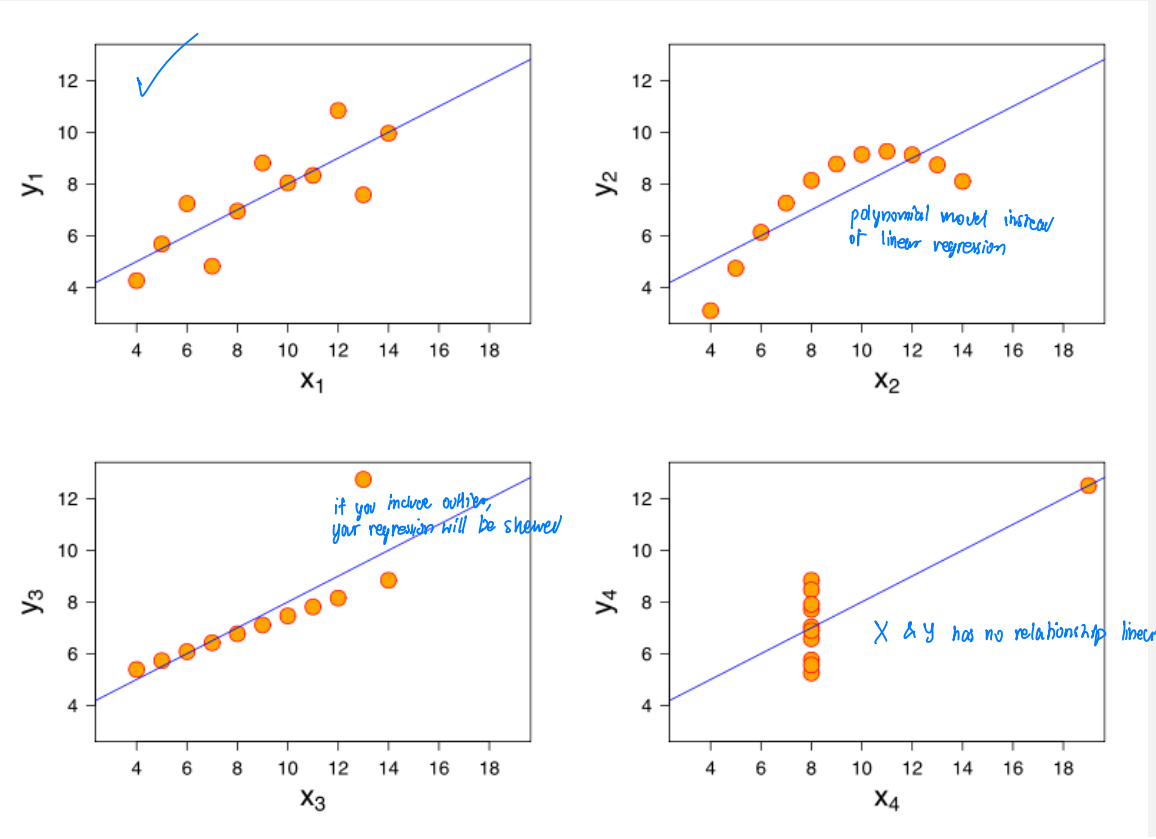




similar variance. y distributed in the same range

RELATIONSHIP BETWEEN Y AND X:
ANSCOMBE'S QUARTET

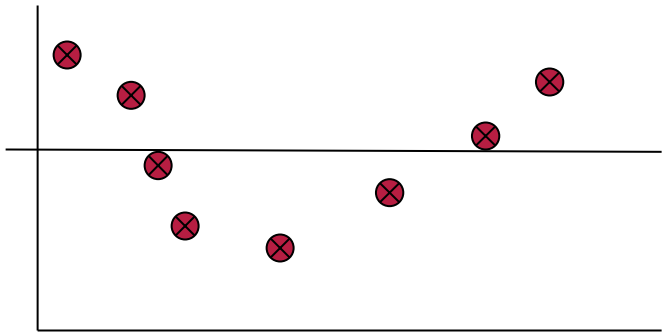
What lines "really" best fit each case? – different approaches



NON-LINEAR TASKS

- Linear Regression will not generalize well to the task above
- Needs a non-linear surface
- Could do a feature pre-process as with the quadric machine
  - For example, we could use an **arbitrary polynomial** in x
  - Thus, it is still linear in the coefficients, and can be solved with delta rule, etc.
  - What order polynomial should we use? – Overfit issues occur as we'll discuss later

$$Y = \beta_0 + \beta_1 X + \cdots + \beta_n X^n + E$$



PROCEDURE 1

- Load data: Merge & Concatenation
- Explore data: “Garbage in, garbage out”
  - Size (col & row) :
    - At least an order of magnitude more examples than trainable parameters
    - **Simple models on large data sets** generally BEAT fancy models on small data sets
  - Quality
    - Realiability: null, duplication, bad label, bad feature value
    - Feature Representation
    - Distribution

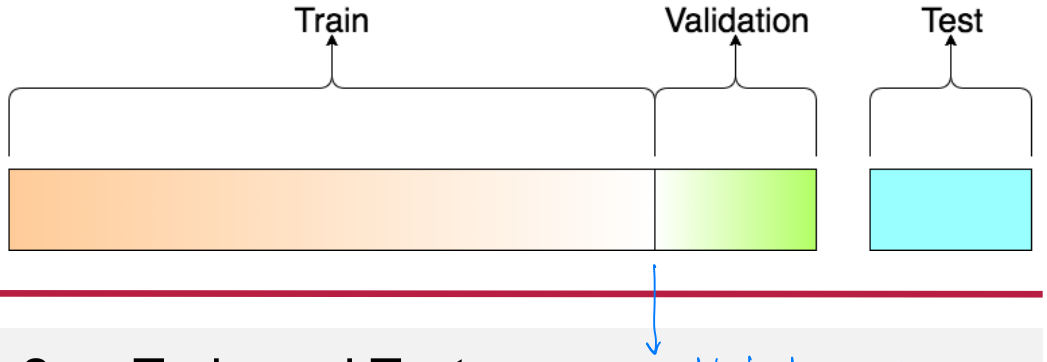
PROCEDURE 2

- Preprocess data:
  - Split train-test
  - Encoding data columns:
    - Integer → Float; Categorical → Binary Values
  - Fill in missing value in train dataset
  - Normalize data in train dataset
  - Select features in train dataset
- Model:
  - Debug a ML model to make the model work for train dataset
  - Preprocess (same as train dataset) and predict test dataset
  - Evaluate regression result of test dataset

TRAIN, VALIDATION, TEST

- A machine learning model aims to make good predictions on new, previously **unseen data**
- **Training Dataset** used to fit the model
- **Validation Dataset** used to fine-tune the hyperparameters
  - Often seen as a part of training set
  - Values or configurations of hyperparameters are set before training, as the learning rate in gradient descent
- **Test Dataset** *used to provide an unbiased evaluation*
  - The test set is large enough
  - You don't cheat by using the same test set over and over

DATA SPLIT

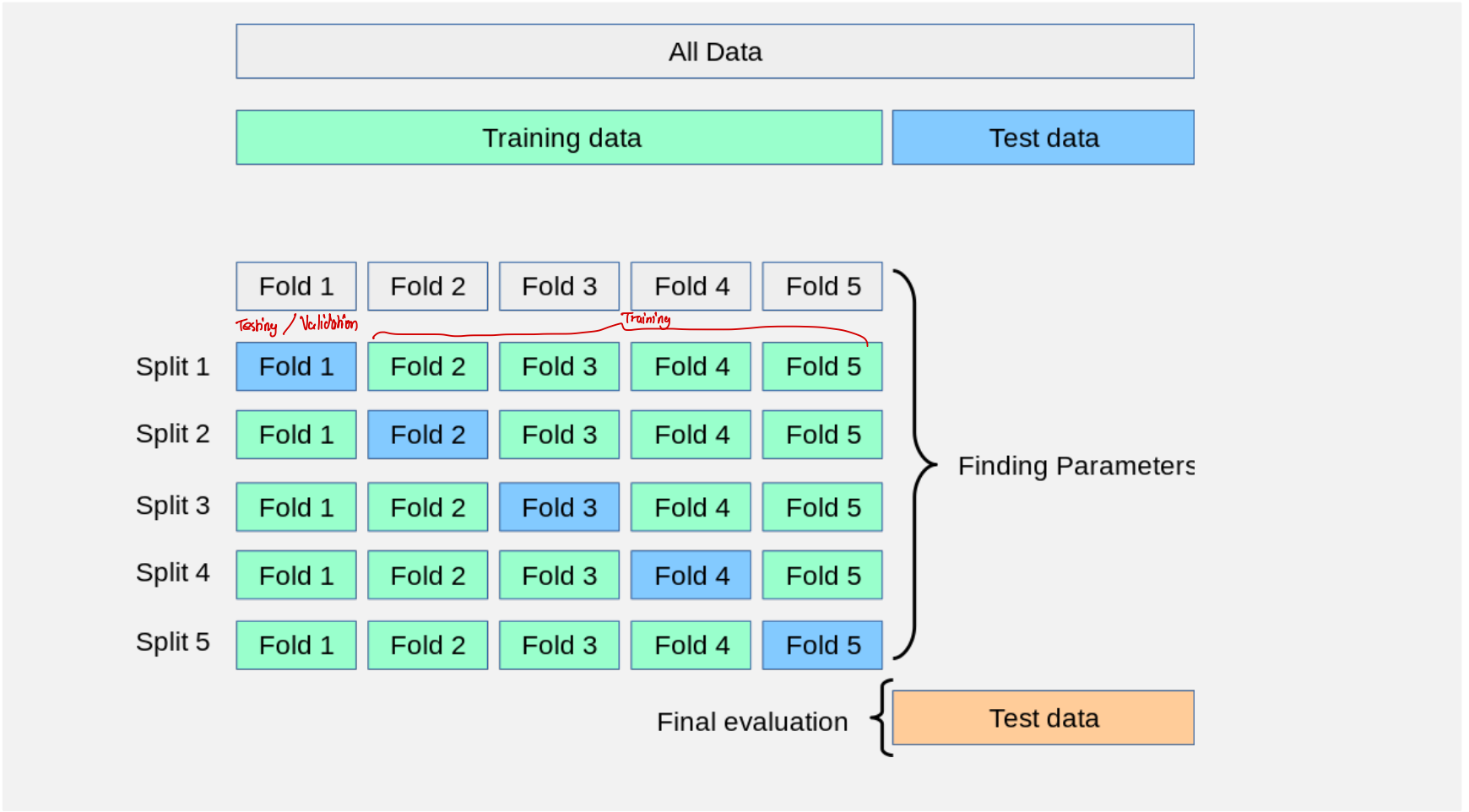


- First split their dataset into 2 — Train and Test
- Then keep aside the Test set
- Randomly choose X% of their Train dataset to be the actual Train set and the remaining (100-X)% to be the Validation set
  - Often used values of X are 10%, 20%, 30%, 40%
  - Shuffle the dataset ?
  - Stratify the dataset based on the distribution of label ?
- Or Cross Validation (if dataset is small)

Midterm. ask what's cross validation

K-FOLD CROSS VALIDATION

- Rather than separating into training and validations set 4 sets for training, 1 for validation
- More robust
- Problem: must run the model k times, makes model slow
- use for small-medium size data
  - else, just use normal train & test & validation split.



BEFORE LINEAR REGRESSION

1. **Linear Assumption**: Scatter plot forms a linear pattern
2. **Remove Noisey**:
   - Y and X_i have a correlation coefficient > 0.5 or <-0.5
3. **Remove Collinearity:**
   - X_i and X_j have a correlation coefficient [-0.5,0.5]
4. **Gaussian Distributions:** Mean & SD close to Pop.
5. **Rescale Input**:
   - Standardization: Mean=0 and SD=1
   - Normalization: [0,1]

SCATTER PLOT

Figure 9-1a
Perfect Positive Linear Relationship

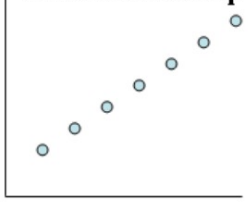


Figure 9-1b
Perfect Negative Linear Relationship

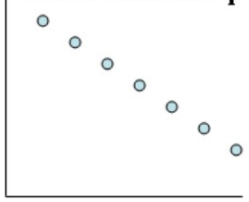


Figure 9-1c
Strong Positive Linear Relationship

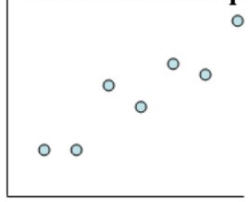
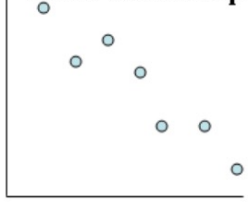


Figure 9-1d
Strong Negative Linear Relationship



Good for linear regression
x/y

Figure 9-1e
Weak Positive Linear Relationship

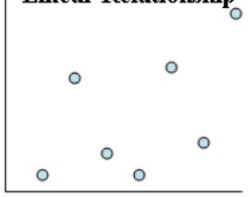


Figure 9-1f
Weak Negative Linear Relationship

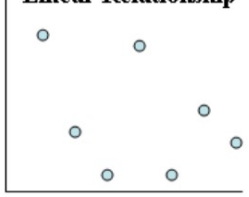


Figure 9-1g
No Relationship

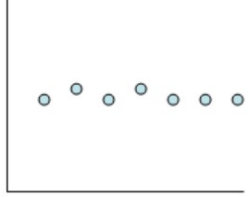
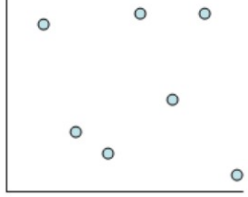


Figure 9-1h
No Relationship



Good for x/x
when there's no relationship

Figure 9-1i
Nonlinear (Negative) Relationship

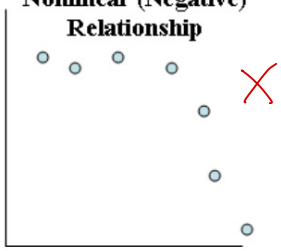
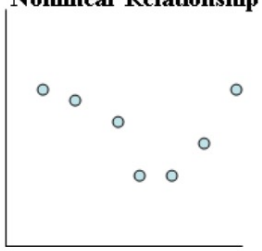
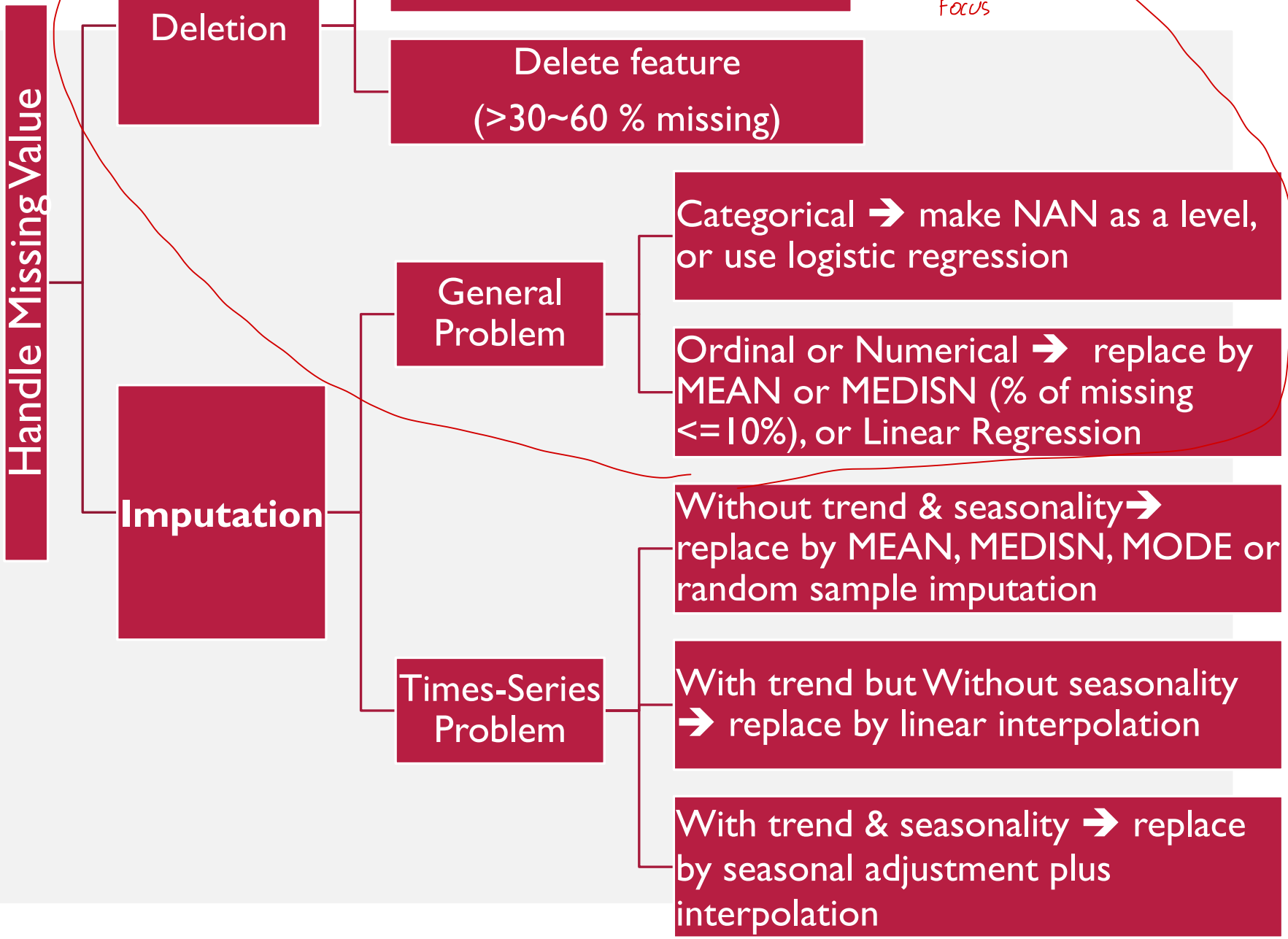


Figure 9-1j
Nonlinear Relationship



Not good for Y/X, X/X


Handle Missing Value
Deletion
Delete sample (in Big Data)
Delete feature (>30~60 % missing)
Midterm Focus
Imputation
General Problem
Categorical → make NAN as a level, or use logistic regression
Ordinal or Numerical → replace by MEAN or MEDISN (% of missing <=10%), or Linear Regression
Times-Series Problem
Without trend & seasonality→ replace by MEAN, MEDISN, MODE or random sample imputation
With trend but Without seasonality → replace by linear interpolation
With trend & seasonality → replace by seasonal adjustment plus interpolation

REGRESSION EVALUATIONS 1

- $Mean\ Absolute\ Error(MAE) = \frac{1}{n}\sum |y_i - \hat{y}_i|$
  - Same unit as the output variable
  - Most Robust to outliers
  - No differentiable
- $Mean\ Squared\ Error(MSE) = \frac{1}{n}\sum (y_i - \hat{y}_i)^2$
  - Differentiable
  - A squared unit of the output variable
  - Penalizes the outliers

REGRESSION EVALUATIONS 2

- $Root\ Mean\ Squared\ Error(RMSE) = \sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2}$
  - Same unit of the output variable
  - Penalizes the outliers
- $Mean\ Absolute\ Percentage\ Error(MAPE) = \frac{1}{n}\sum\left|\frac{y_i - \hat{y}_i}{y_i}\right| = \frac{1}{n}\sum\frac{|y_i - \hat{y}_i|}{y_i}$
  - Easy to judge
  - Most widely used

REGRESSION EVALUATIONS 3

- $R^2 = 1 - \frac{SSR}{SSM}$
  - % of variance explained by model
  - Tend to include all features
- $Adjusted\ R^2 = 1 - \left[\frac{n-1}{n-k-1} \times (1 - R^2)\right]$
  - K increases by adding some features

HW4: APPLY LINEAR REGRESSION TO PREDICT **THE NUMBER OF CRITICAL REVIEWS** ON IMDB

- movie_title : Title of the Movie
- duration: Duration in minutes
- director_name : Name of the Director of the Movie.
- director_facebook_likes : Number of likes of the Director on his Facebook Page.
- color: Film colorization. 'Black and White' or 'Color'
- genres: Film categorization like 'Animation', 'Comedy', 'Romance', 'Horror', 'Sci-Fi', 'Action', 'Family'
- actor_1_name: Primary actor starring in the movie
- actor_1_facebook_likes : Number of likes of the Actor_1 on his/her Facebook Page.
- actor_2_name: Other actor starring in the movie
- actor_2_facebook_likes : Number of likes of the Actor_2 on his/her Facebook Page.
- actor_3_name: Other actor starring in the movie
- actor_3_facebook_likes : Number of likes of the Actor_3 on his/her Facebook Page.
- num_critic_for_reviews : Number of critical reviews on imdb Predict this for y
- num_voted_users: Number of people who voted for the movie
- cast_total_facebook_likes: Total number of facebook likes of the entire cast of the movie.
- language : English, Arabic, Chinese, French, German, Danish, Italian, Japanese etc
- country: Country where the movie is produced.
- gross: Gross earnings of the movie in Dollars
- budget: Budget of the movie in Dollars
- title_year: The year in which the movie is released (1916:2016)
- imdb_score: IMDB Score of the movie on IMDB
- movie_facebook_likes: Number of Facebook likes in the movie page.

HW3: SOLUTION STEPS 1

- Load data:
- Explore data:
  - Head()/tail()
  - info()→data_type & size
  - describe() & value_counts() plot → distribution
  - isna()→missing value

HW3: SOLUTION STEPS 2

- Preprocess data:
  - Split train-test
  - Encoding data features:
    - Integer → Float; ~~(Categorical → Binary Value)~~
  - Select features in train dataset
    - Only numerical features
    - Correlation – use correlation matrix or heat map to select x that is highly correlated with y (>0.5 or <-0.5), and then ignore x which is highly correlated with another x
  - Fill in missing value in train dataset
  - Scale features (comparable units) in train dataset

can exclude, only include a few important categorical data
↓ change them to binary then use it
ex: Director: pick the top 5 directors to recode the column.

HW3: SOLUTION STEPS 3

- Model:
  - Debug the LR model to make the model work for train dataset
  - Preprocess (based on train dataset) and predict test dataset
  - Evaluate regression result of test dataset
  - Interpret the regression function

can submit multiple files
- code
- powerpoint
  - ↳ what preprocessing
  - ↳ what data
  - ↳ why?
  - ↳ evaluate. good enough?
    - ↳ if not, why? explain. how to improve?

- Purpose: Try. Don't have to have very high R^2
- "Social Science"