Riggression = predicting continuous output

Linear Regression (parmitric)

(parmitric) → gets the paramiter from learning procces

hyperperparamiter → you provide the value not from learning

$\hat{y} = w_1 x_1 + w_2 x_2 + b$

$\hat{y} = \theta_1 x_1 + \theta_2 x_2 + \theta_0$

\# of predicters = # of features + 1

* to evaluate a line ⇒ predict on the traing set and calculate the Loss. (How far our prediction from the correct)

Per sample → $Loss = |y - \hat{y}|$ (L^1)

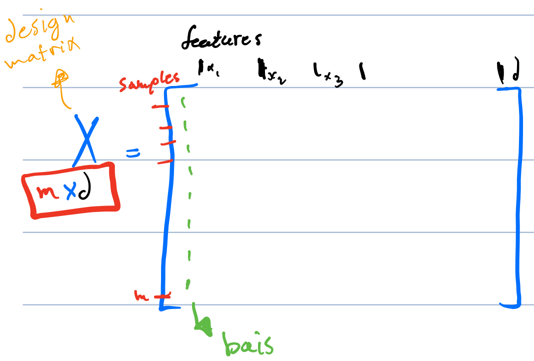
for all samples → $Cost(J) = \frac{1}{m} \sum^{m} |y_i - \hat{y}_i|$

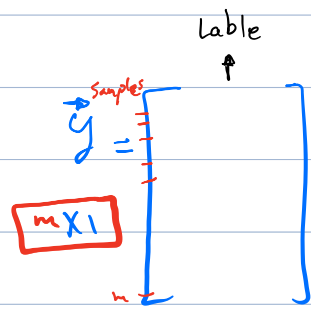
(m = # of samples)

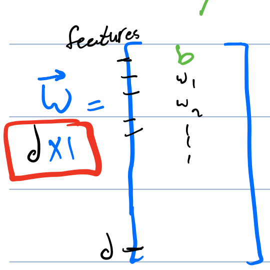
RSS $Loss = (y - \hat{y})^2$ (L^2) (better)

$cost(J) = \frac{1}{2m} \sum^{m} (y_i - \hat{y}_i)^2$

A residual sum of squares (RSS) is a statistical technique used to measure the amount of variance in a data set that is not explained by a regression model.







$J = (Xw - y)^T (Xw - y)$

$= X^T w^T X w - X^T w^T y - y^T X w + y^T y$ (are equal)

$\frac{dJ}{dw} = 2 w X^T X - 2 X^T y$

$2 w X^T X - 2 X^T y = 0$

$w X^T X = X^T y$

the minimum: $w = (X^T X)^{-1} X^T y$ → very rare way to get w ??

closed form: no hyperparameters; no need for feature scalling

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

$A \cdot A^T = A^2$

- Sometimes there is no inverse
- inverse is costly (d^3)

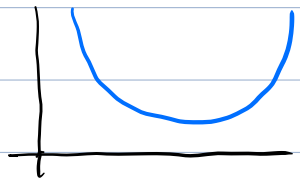
m < d

no problems with big # of d

Local Search (Iterative solution):-

Gradiant desent :-

better than closed-form because the graph is confex (one minimum)



Features must be scaled to avoid streached graph → more time to reach the solution

However, the solution doesn't change

* big step = overshoot, may fail to converge
* small step = more time

Step 1: Initial w, b randomly

step 2: calculate J (High value inital)

step 3: calculate $\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}$

step 4: $w_j \leftarrow w_j - \alpha \frac{\partial J}{\partial w_j}$

$b \leftarrow b - \alpha \frac{\partial J}{\partial b}$

Step 5: repeat 2 to 4 until $J^k - J^{k+1} \leq \epsilon$ (10^{-3})

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

$$\frac{\partial J}{\partial w_j} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) \cdot x_j$$

$$\frac{\partial J}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)$$

10

5

=

initially

✱ Sleep	✱ Study	Score	$\hat{y}$	Loss	$\frac{\partial J}{\partial w_1}$	$\frac{\partial J}{\partial w_2}$	$\frac{\partial J}{\partial b}$
6	6	9	0	64	-48	-48	-8
5	6	8	0	64	-40	-48	-8
7	8	9	0	81	-63	-72	-9
7	7	7	0	49	-49	-49	-7
				$J = 32.25$	-50	-44.25	-8

$w_1 \leftarrow 0 - 0.01 \times (-50)$

$w_1 = +0.5$

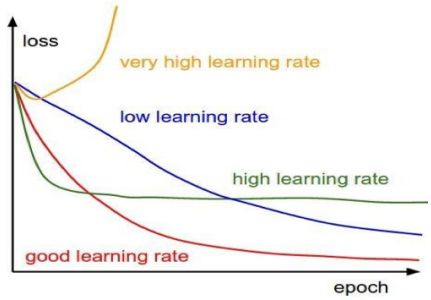
$w_2 \leftarrow 0 - 0.01 \times (-44.25)$

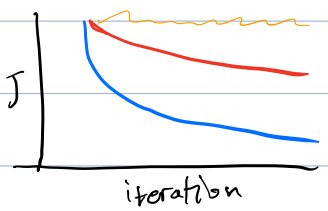
$w_2 = +0.4425$

$b \leftarrow 0 - 0.01 \times (-8)$

$b = 0.08$

then update $\hat{y}$ and repeat





apropriate α
small α
high α

✱ Polynomial regression

in Linear Regression => we assumed that features have linear relation with predictor

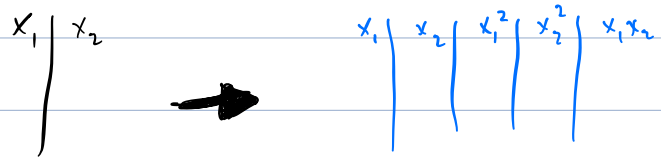
if you have complex relation :- **features expantion**

expand before scaling

✱ Study → (✱ study)^2

adding new feature which is squared (or something else)

then do linear Regression
but plot in linear plot



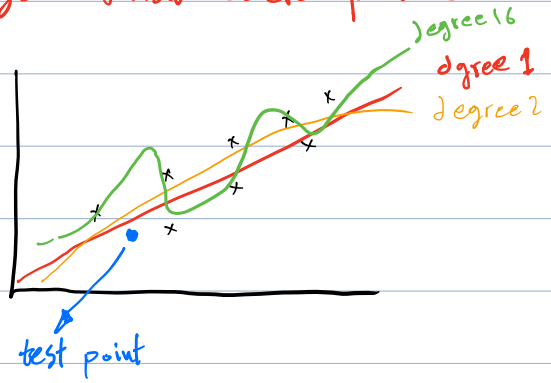
in Real live, no need to expand more than square:-

- not that complecated
- it's hard to Knew the relationship

※ over fitting

if you used expanded features
you may have Overfitting ⇒ same as when you have lots of features

you will start with line until you get very comlecated curve that goes throw each point (each lable in training set) overfitting

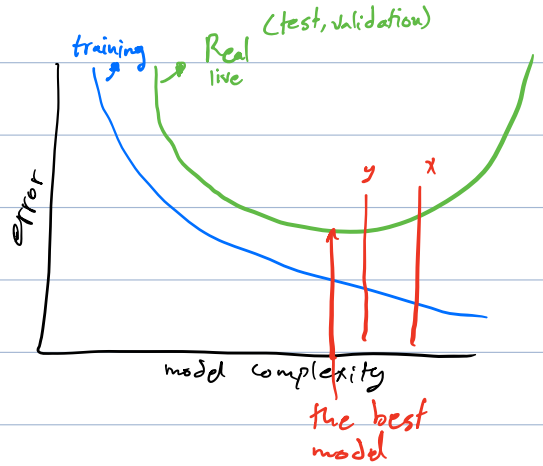


the semptom of it ⇒ the constant with coefficients become bigger

What is Overfitting ??

why model x is over fitting ?
because there is a better model (y) in testing set

The criteria is to reduce error in Generalization set



How to deal with Overfitting ??

① Regularization
② add more training data

Regularization (techniques to deal with overfitting): [Penalize for large value of w]

① Ridge Regularization

→ Ridge Regression

→ L^2

$\hat{y} = b_0 + w_0 x_1 - 2 x_2$

$\|w\| = \sqrt{w_0^2 + 2^2}$

$\|w\|^2 = w_0^2 + 2^2$

for training

$$J = \frac{1}{2m} \sum (\hat{y}^{(i)} - y^{(i)})^2 + \lambda \|w\|_2^2$$

Regularization hyperparameter — don't include bais

↳ we get it using trail and error

if λ is high → underfitting

if λ is 0 → no Regularization (overfitting)

① closed-formula

$$J = (Xw - y)^T (Xw - y) + \lambda w^T w$$

fields of ML: { define cost; optimize cost; Regularization }

$$w = (X^T X + \lambda I)^{-1} X^T y$$

Identity matrix

→ by adding λI always you can get the inverse

→ but still you have the computation problem when d is large

$$w_j \leftarrow w_j - \alpha \frac{\partial J}{\partial w_j} \rightarrow \frac{1}{m} \sum (\hat{y} - y) \cdot x_j + \lambda w_j$$

$$b \leftarrow b - \alpha \frac{\partial J}{\partial b} \rightarrow \frac{1}{m} \sum (\hat{y} - y)$$

$$w_j \leftarrow w_j (1 - 2\lambda) - \alpha \frac{\partial J}{\partial w_j}$$

※ Steps

① start with simple model

② go more complex (polynomial) until get the best model

③ go 2 more dgree then regularize it

↳ why? to reduce the bais and regularization will reduce the variance

The Sources of error:-

① bais² (simple model → high bais) (complex model → low bais)
How far the avrg model from actual data

① variance how much each model different from each other
How far each model from actual
[simple model → low variance
[complex model → high variance

③ irreducable error ϵ
- noisy in lables
- lack of information
- Inherctence between data

*Ideal ⇒ low bais + low variance

feature Selection:-

* more features with less data set leads to overfitting

ways:-

① Chroustic approch:- (doesn't give the optimal)
Randomly

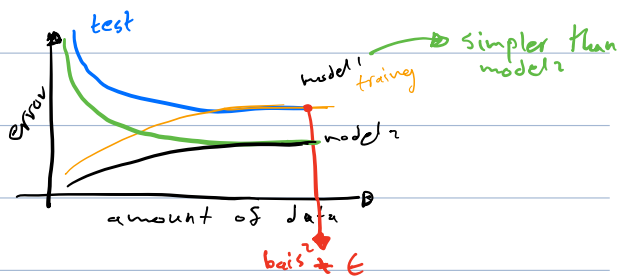
② forward features selection
Start with 1 feature then add the best feature

③ Backward features dedection
start with all features then delete the worst feature

* if you have more data: you can reduce the variance

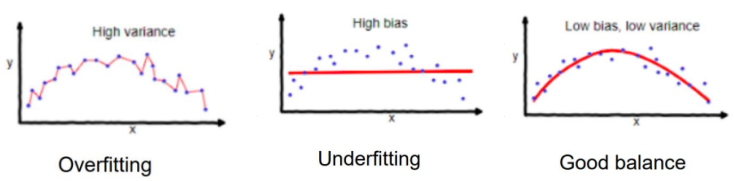
* more data ⇒ more training error

* more data ⇒ low test Generalization error



* theoritically ⇒ you can have 0 variance but you cant have 0 bais

All of them are slow the better way is Lasso



② Lasso Regularization

$$J = \frac{1}{2m} \sum (\hat{y} - y)^2 + \lambda \|w\|_1$$

more λ → more features become zero

→ don't have closed-form solution or graliant

→ we solve it by Coordinate descent

→ it will do feature selection automatically

→ so it's deal with overfitting

- ❑ Lasso tends to select amongst highly correlated features arbitrarily → randomly (without any reason)
- ❑ Empirically, ridge has better predictive performance

to get the best solution of-

Lasso → (then) Ridge or Elastic net

(combin Lasso and Ridge)

$$J = \frac{1}{2m} \sum (\hat{y}^i - y^i)^2 + \lambda \left(\alpha \|w\|_2^2 + (1-\alpha) \|w\|_1 \right)$$

- ❑ Perform training and check the error:
- ❑ If error is high: (underfitting) [when $d > m$]
  - ➢ Add more features
  - ➢ More complex model by polynomial
- ❑ If overfitting:
  - ➢ Add more data
  - ➢ Perform regularization

Bias and variance