Regression

Machine Learning (CS 306)

Instructor: Dr. Moumita Roy

Teaching Assistants: Indrajit Kalita, Veronica Naosekpam

Email-ids:

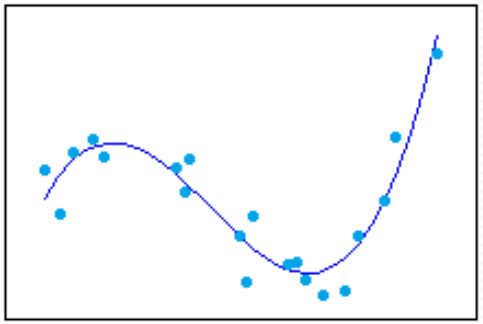
moumita@iiitg.ac.in

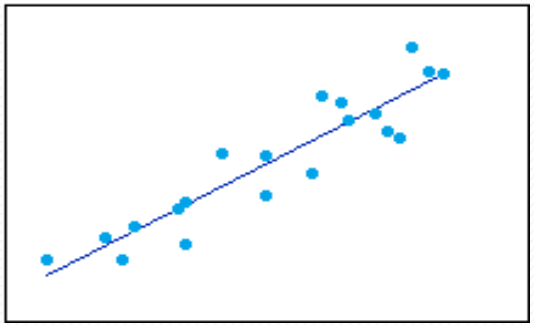
veronica.naosekpam@iiitg.ac.in

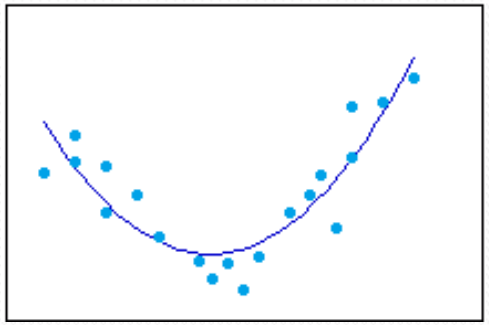
indrajit.kalita@iiitg.ac.in

Mobile No: +91-8420489325 (only for emergency quires)

Polynomial Curve fitting

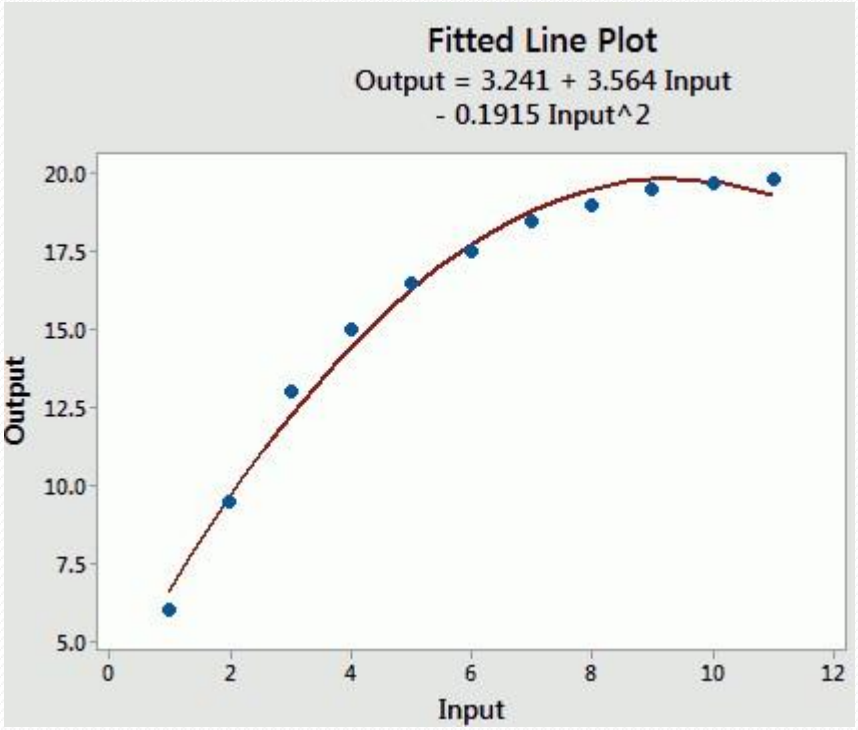






Polynomial Curve fitting (with known function)

Independent variable (Input/x)
Dependent variable (Output/y)



Structure of the polynomial (Not known)

Try to approximate polynomial structure with some (training) samples $<x_i, y_i>$

x — Independent variable / Input

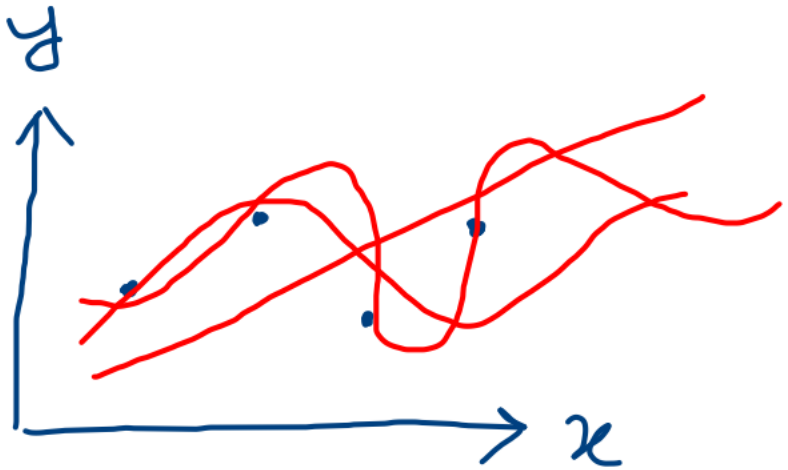
y → Dependent variable / Output

Suppose, Some set of (x, y) is provided,

x	y
0.1	5
0.6	2
0.9	1.2

Solution?

Solution



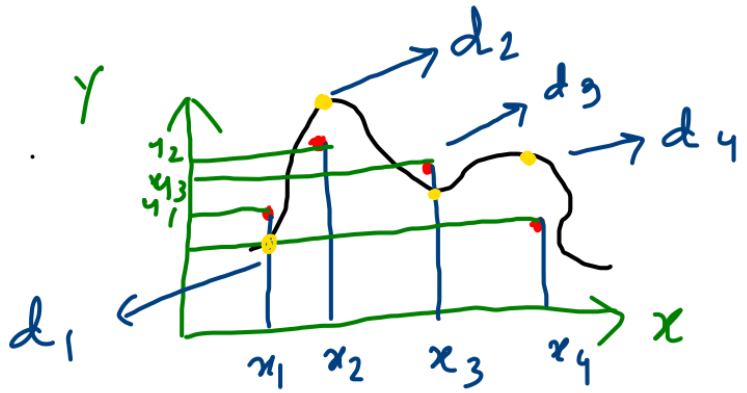
make a guess about the equation

many options??

Which one is better??

Draw curves

- find best-fit ones



x	Actual Y	Predicted d
.1	5	4
.6	2	1
.9	1.2	6
1.2	3	2

predicted value for the approximate polynomial structure ??

m = no of training samples

d_1, d_2, d_3, d_4

$$\text{Error} = \sum_{i=1}^{m} (y_i - d_i)^2$$

Sum-of-square error

Try to find a polynomial structure (equation) which minimize (Error)

To sum up

- The structure of the polynomial is not known (one independent and one dependent variables).
- Instead of it, we have some samples to approximate the structure of the polynomial
- There are many possible structure (curves/equations)

  Like, $y=a+bx$, $y=ax^2$, $y=a+bx^2+cx^3$……
- We can assume a structure and calculate error.
- Try to find a polynomial that minimizes the error (best fit curve)

→proceed with trial and error approach

Supervised Learning

- There is a set of training samples $<x_i, y_i>$
- The goal is to learn function/hypothesis $h(x_i)$ so that it is a good predictor for the corresponding value of y using the training samples

$$\boldsymbol{h: x \rightarrow y}$$

- If y is continuous, then the problem is called regression
- If y is discrete, then the problem is called classification.
- Aim-> to design learning algorithm to approximate hypothesis structure

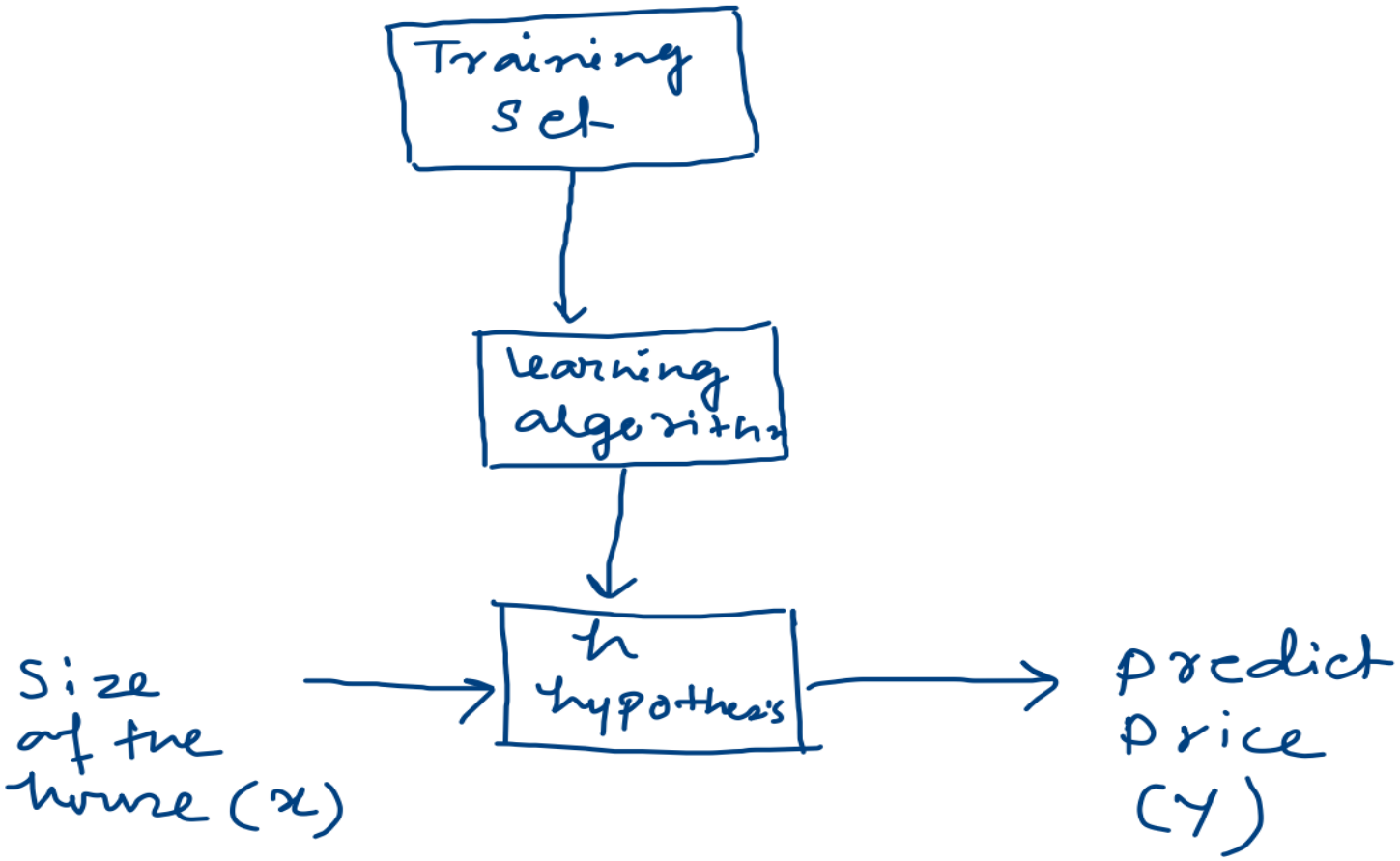
Some application for Regression

Size of house (in sq.ft)	Price (Lakh)
600	15
1120	30
2000	40
700	30

Predict the price of the house

H.W: Find some more applications of regression problem

Graphical Abstract



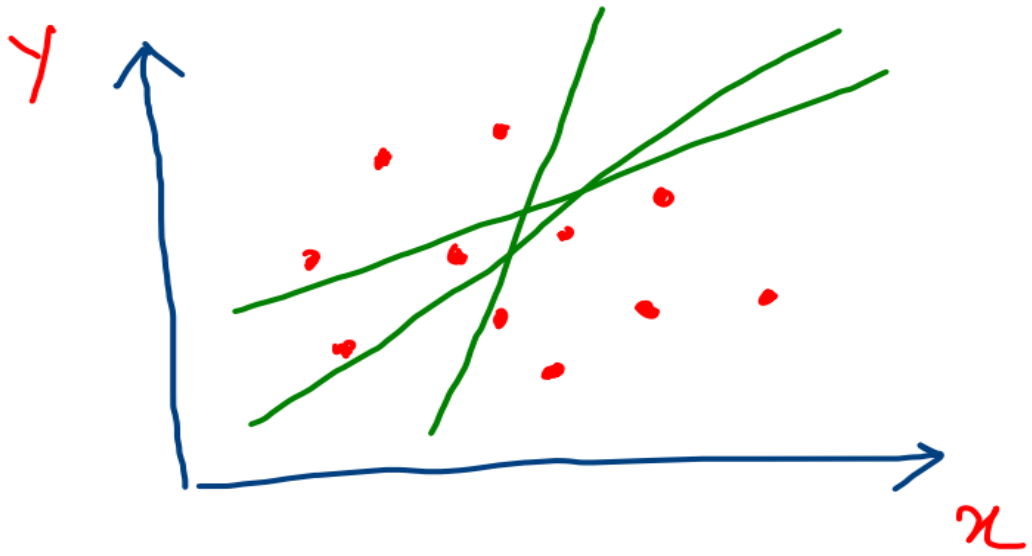
Linear Regression (univariate)

- Assume, y is a linear function of x

$$h(x)=w_0+w_1 * x$$

- Learn $h(x)$ using the training set $<x_i, y_i>$
- m=number of training sample, number of feature =1

$$J(w_0, w_1) = \frac{1}{2} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$



Linear Regression (multiple features)

Age of the house (year)	No. of bedrooms	Size of house (in sq.ft)	Price (Lakh)
2	2	600	15
10	3	1120	30
6	4	2000	40
7	2	700	30

Linear Regression (multiple features)

$$x_i = [x_{i,1} \quad x_{i,2} \quad x_{i,3} \quad \dots \quad x_{i,n}]$$

n = number of features (independent variables)

$$h(x_i) = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + w_3 x_{i,3} + \dots + w_n x_{i,n}$$

$$= \sum_{j=0}^{n} w_j x_{i,j} \quad | \quad x_{i,0} = 1$$

Training samples: $\langle x_i, y_i \rangle$ — n number of features

$$W = [w_0 \quad w_1 \quad w_2 \quad \dots \quad w_n]$$

$$J(W) = \frac{1}{2} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

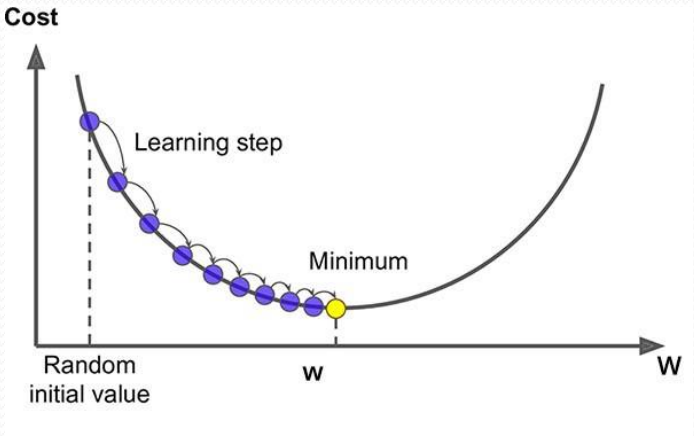
↳ minimize

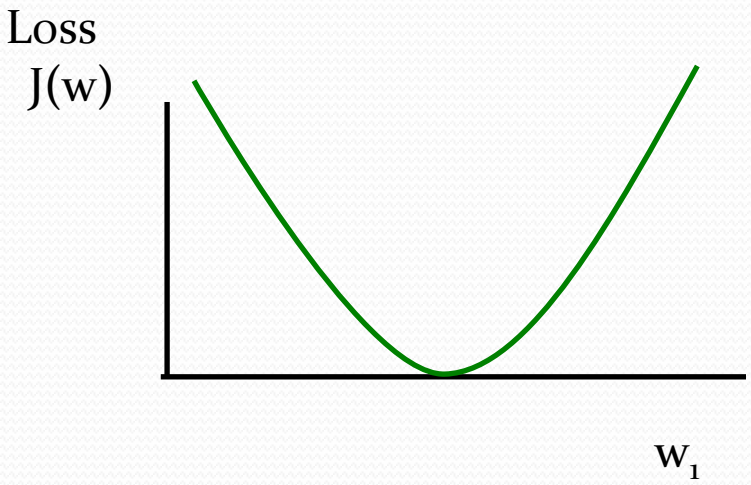
Aim

- To obtain parameter $w_0, w_1, w_2, \ldots, w_n$ for good approximation of y
- Check random initialization of w and find the best fit one by trial and error approach
- Solve the problem using linear algebra (numerical complexity)
- Fit model by minimizing sum of squared errors (least squares)

Finding the minimum

Finding the minimum

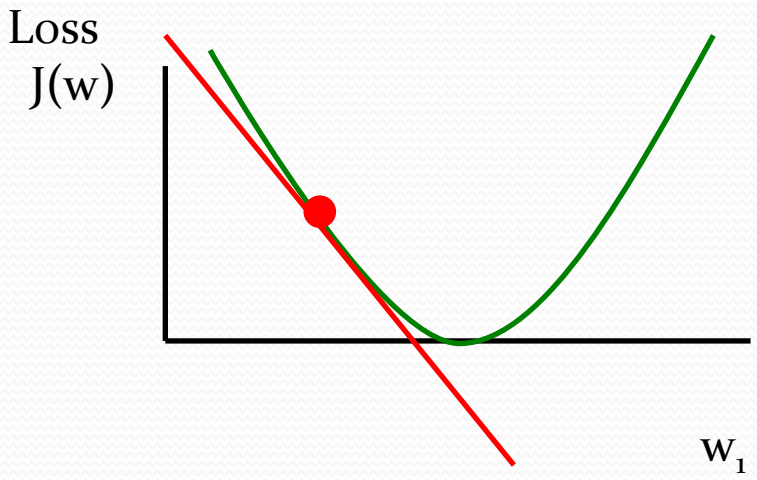




How do we do this for a function?

One approach: gradient descent

Partial derivatives give us the slope (i.e. direction to move) in that dimension (suppose w_1)



One approach: gradient descent

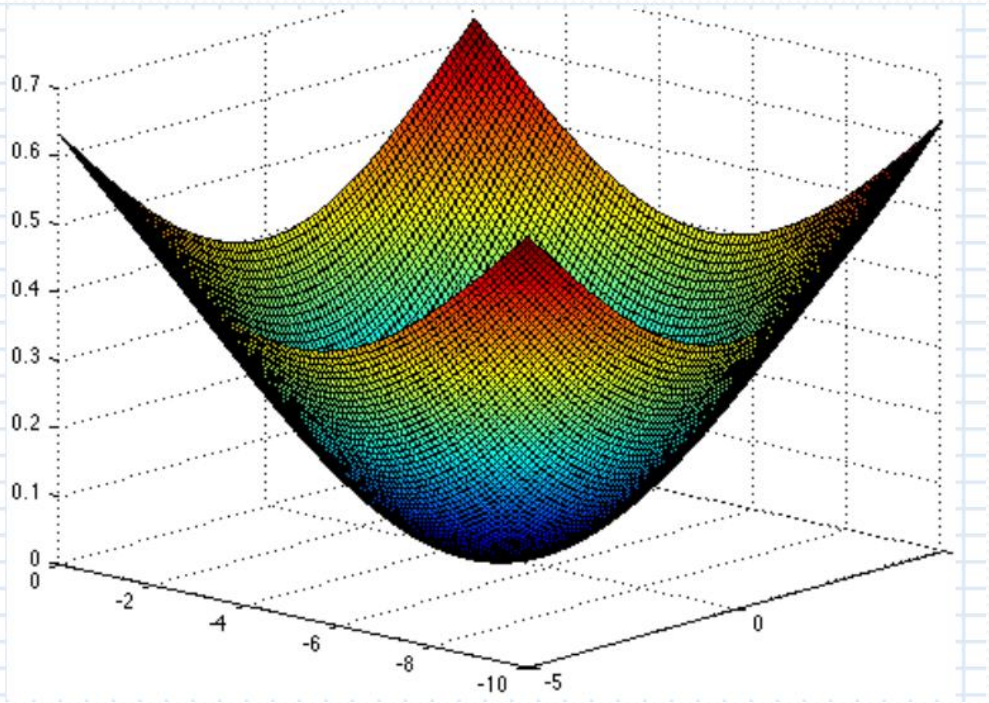
Partial derivatives give us the slope (i.e. direction to move) in that dimension

Approach:

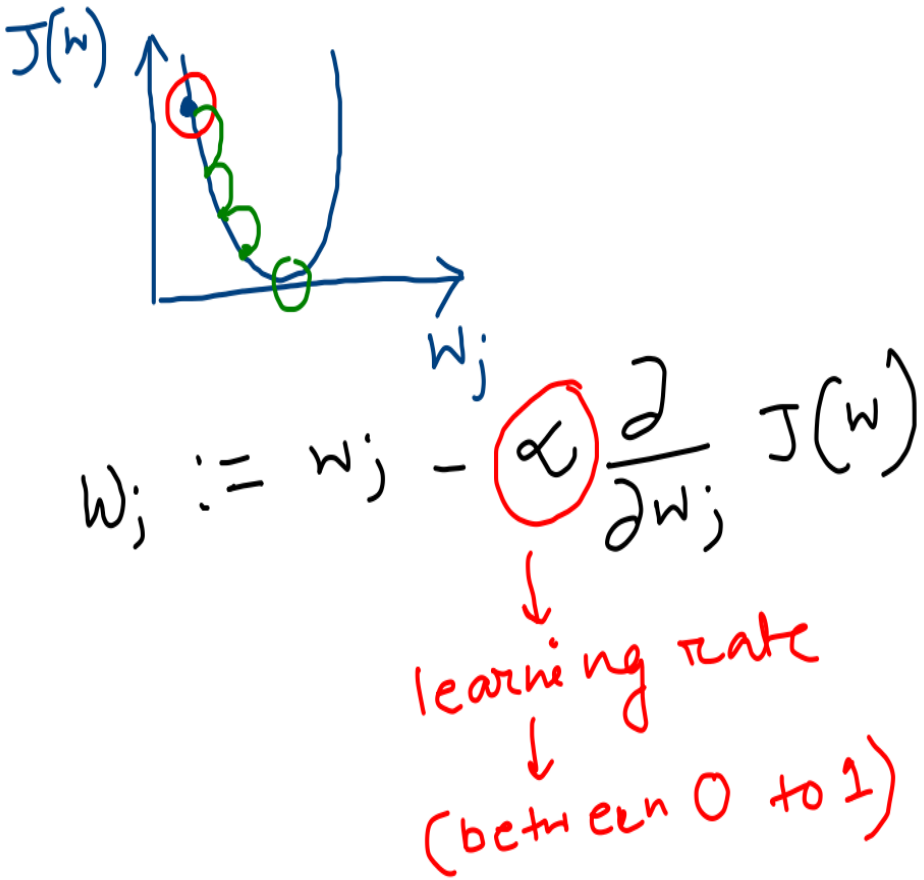
- pick a starting point (w)
- repeat:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)/opposite direction of slope

Gradient Descent for Linear Regression

- Convex loss function
- Geometrically: Error surface is bowl shaped



Gradient Descent



- Simultaneously update for $j=1,2,\ldots,n$

Gradient Descent for Linear Regression

$W = [w_0 \; w_1 \; w_2 \; \dots \; w_n]$

- randomly initialized between $[-3 \text{ to } +3]$

n = number of features

$X_1 \quad X_2 \quad \dots \quad X_m$

m = number of patterns

$X_i = [x_{i,1} \quad x_{i,2} \quad \dots \quad x_{i,j} \quad \dots \quad x_{i,n}]$

$$h(x_i) = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + \dots + w_n x_{i,n}$$

$$= \sum_{j=0}^{n} w_j x_{i,j}$$

$h(x_1) \rightarrow$

$h(x_2) \rightarrow$

$\vdots$

$h(x_m) \rightarrow$

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

MSE

↳ mean

Gradient Descent for Linear Regression

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w)$$

Update

w_1

w_2

w_3

$\vdots$

w_n

} Simultaneously

Gradient Descent for Linear Regression

$$\frac{\partial}{\partial w_j} J(w) = \frac{\partial}{\partial w_j} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right)^2 \right]$$

$$\vdots$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right) x_{i,j}$$

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right) x_{i,j}$$

$$= w_j + \alpha \frac{1}{m} \sum_{i=1}^{m} \left(y_i - h(x_i) \right) x_{i,j}$$

- if we have large number of training samples!?

Algorithmic representation for gradient descent

Step 1: Randomly initialize $w_0, w_1, ..., w_n$ and learning rate

Step 2: Calculate $h(X_i)$ for all training samples

Step 3: Calculate $J_{itr}(W)$ for itr^{th} epoch

Step 4: Update all parameters $w_0, w_1, ..., w_n$

Step 5: Repeat steps 2-5 until

$$| J_{itr}(W) - J_{itr-1}(W)| > \rho$$

Step 6: Report MSE $J_{itr}(W)$

ρ: small value (may be *0.0001*)

Epoch/training step: one time updating considering all the training samples

Gradient Descent for Linear Regression

- **Batch gradient descent:**

➢ all the training samples are taken into consideration for a single updating of all parameters (after each epoch).

➢ Problem for huge number of training samples

- **Stochastic gradient descent:**

➢ Update parameters by considering just one training sample at time (repeat epoch by epoch till convergence)

$$w_j := w_j + \alpha (y_i - h(x_i)) x_{i,j}$$

↳ simultaneously for all $w_1, w_2 \dots w_n$

Algorithmic representation for stochastic gradient descent

Step 1: Randomly initialize $w_0, w_1, \ldots, w_n$ and learning rate

Step 2: Initialize $itr = 0$ (itr stands for epoch)

Step 3: $J_{itr}(W)=0$

Step 5: Repeat Steps 6-8 for all the training samples

Step 6: Calculate $h(X_i)$ for i^{th} training sample

Step 7: Update $J_{itr}(W)= J_{itr}(W) +(h(X_i)-Y_i)^2$

Step 8: Update all parameters $w_0, w_1, \ldots, w_n$

Step 9: $J_{itr}(W)=(1/2m)\ J_{itr}(W)$

Step 10: Check convergence if $| J_{itr}(W) - J_{itr-1}(W)|<\rho$ then stop and report MSE $J_{itr}(W)$; else go to step 11

Step 11: $itr=itr+1$ and repeat steps 3-10

ρ: small value (may be *0.0001*)

Exercise 1

Design a linear regression model using (batch mode) gradient descent approach by considering the following dataset and use this model to predict the output for the input pattern [3 12]

Feature 1	Feature 2	Output
1	2	6
2	10	24

For demonstration, note the followings:

1. Learning rate: 0.1
2. Number of training steps/epochs:2
3. All parameters are initialized as 1

Parameters vs. HyperParameters

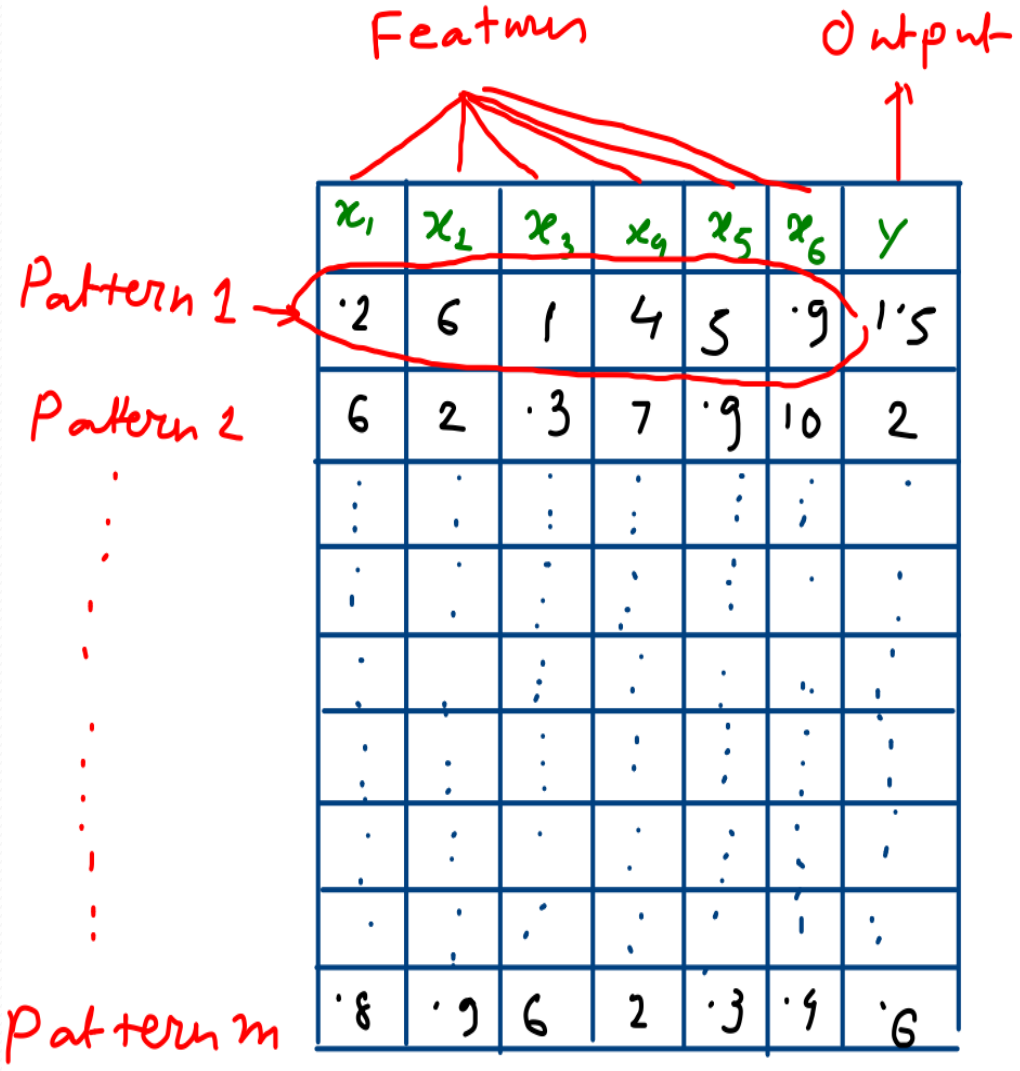
- **Parameter:** A model parameter is a variable of the selected model which can be estimated by fitting the given data to the model.

  Example: $w_0, w_1, w_2, \ldots, w_n$

- **HyperParameter:** A model hyperparameter is the parameter whose value is set before the model start training. They cannot be learned by fitting the model to the data.

  Example: learning rate, number of epochs

Training, Test, Validation Set



Split randomly

training set (20%.)

validation Set (10%.)

test set (70%.)

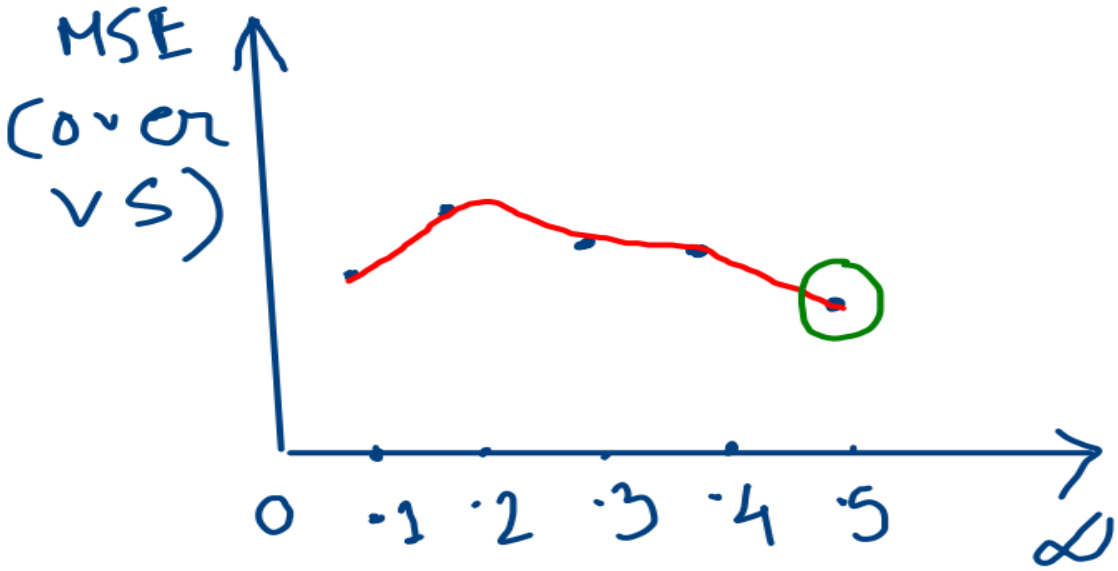
Training, Test, Validation Set

- **Training set (TS)**: samples/patterns are used to train the model (by updating the parameters).

- **Validation set (VS):** samples/patterns are used to evaluate the performance of models with different values of hyperparameters (hyperparameter tuning/model configuration). It's also used to detect overfitting during the training stages.

- **Test set (TeS):** samples/patterns are used to evaluate the final performance (Mean square error) of the models (after hyperparameter tuning).

Example of Hyperparameter Tuning



TS → 20%

VS → 10%

Calculate MSE over 70% (TES)
using $\alpha = .5$

Final Result

Training samples (%)	α	Train MSE	Test MSE
10%	.5	?	?
40%	.6	?	?
60%	.3	?	?
80%	.8	?	?

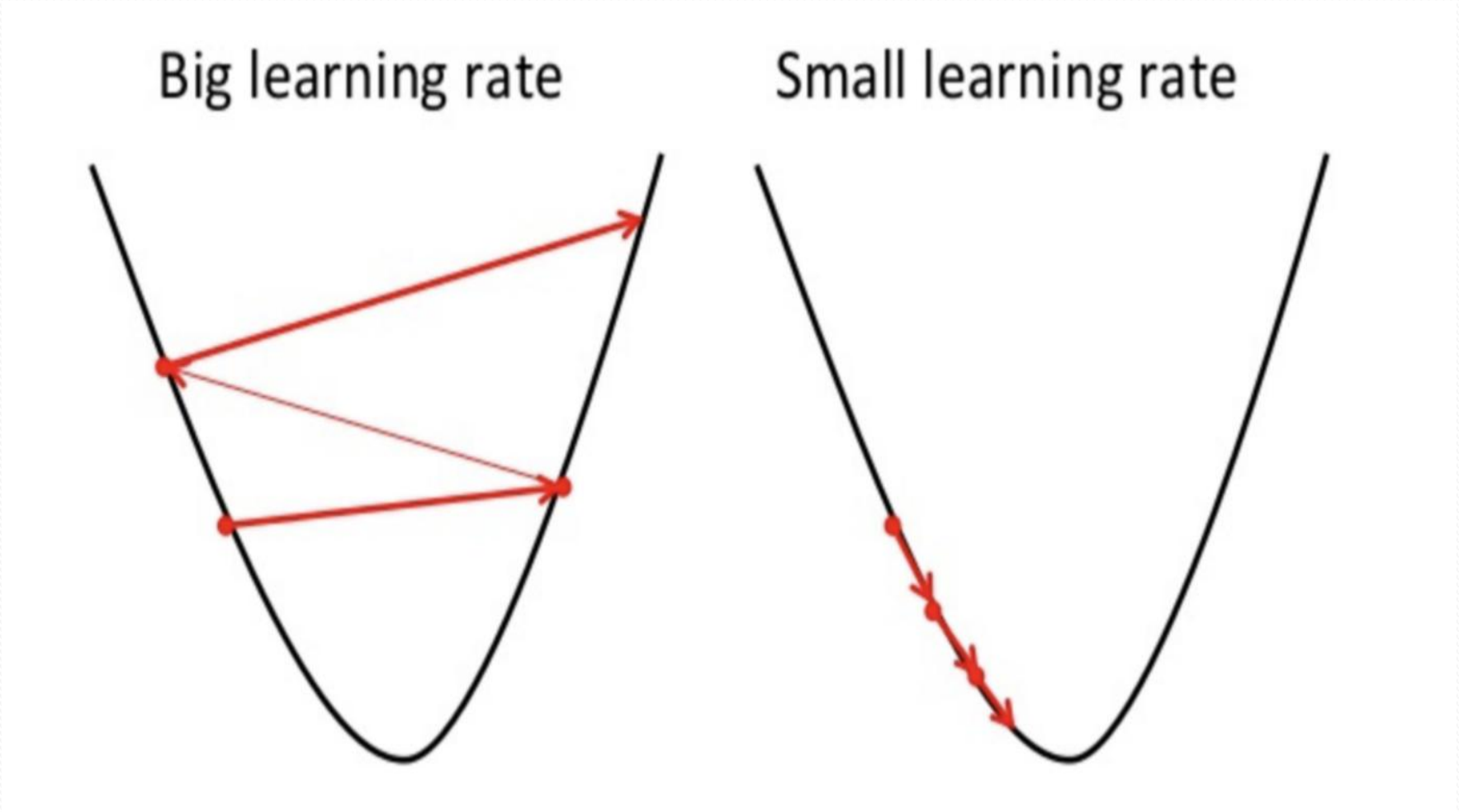
↓

After Convergence

Some observations

- Is there any affect of initial parameter values in performance?
- Is there any affect of different random set of training samples (considering same percentage) in performance of the model?
- How to check these issues during experimentation?

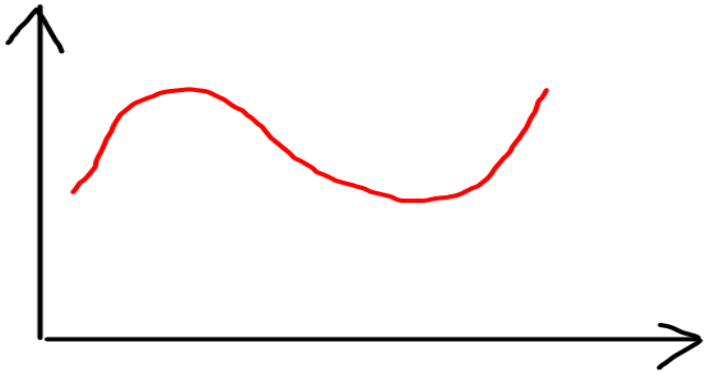
Setting of learning rate



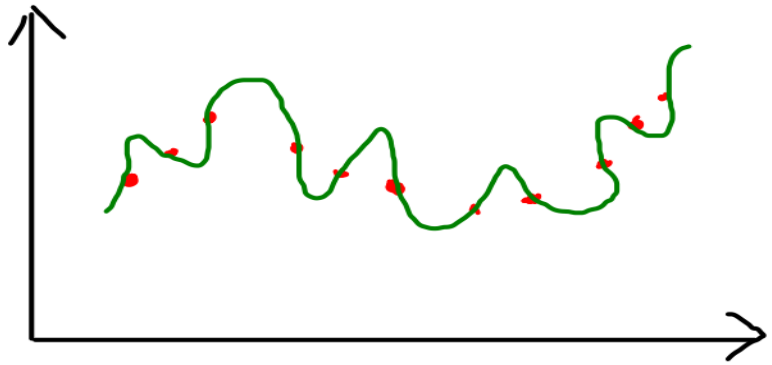
Overfitting and Underfitting

- Overfitting: model performs well using training samples but does not perform well on the testing samples. The validation set is also used to detect overfitting during the training stages.

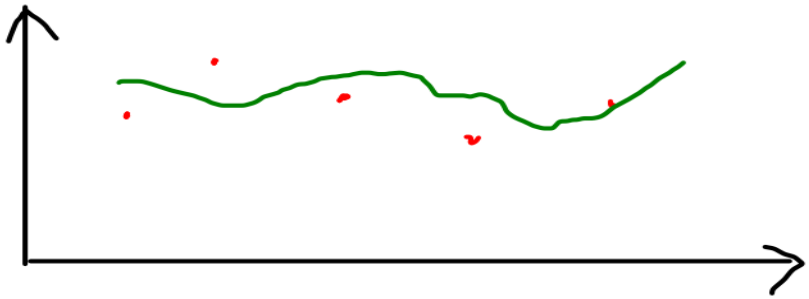
- Underfitting: model does not perform well on training samples as well as testing samples (may be due to very insufficient training samples).



→ actual curve
(we have training samples; not exact polynomial structure)

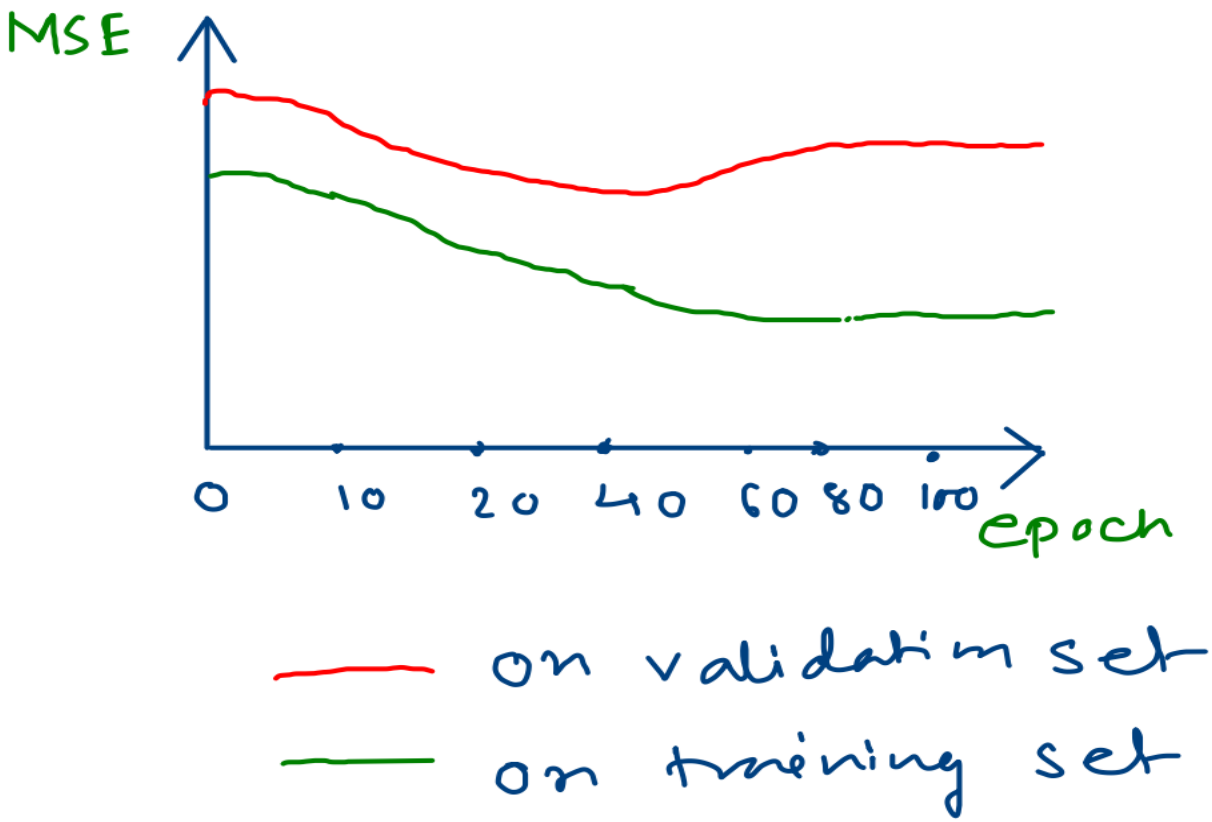


→ overfitting
→ training error in zero
→ but not able to fit the curve



→ underfitting
→ poor performance in training / testing

Checking overfitting



Feature Scaling (Normalization)

- Feature scaling is a method used to normalize the range of independent variables or features of data.
- It can speed up the gradient descent.
- Absolute maximum: Find maximum value in the dataset (considering features only) and divide all the features by the maximum value.

x_1	x_2	y
1·2	200000	1·2
1·3	400 00	6
1·4	50 000	1·2
2	10 0000	1·6