Linear Regression

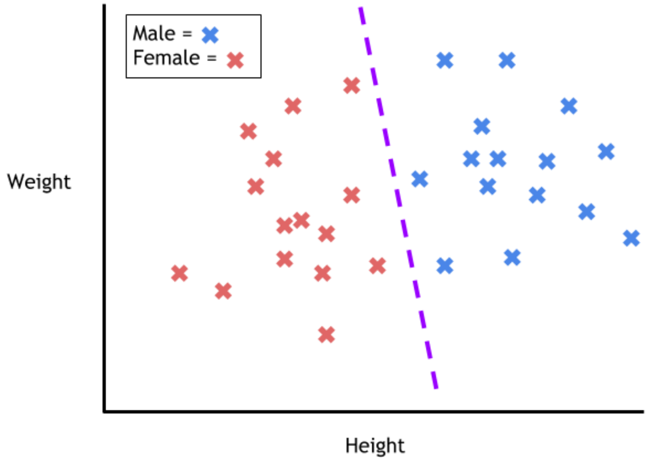
Simi S

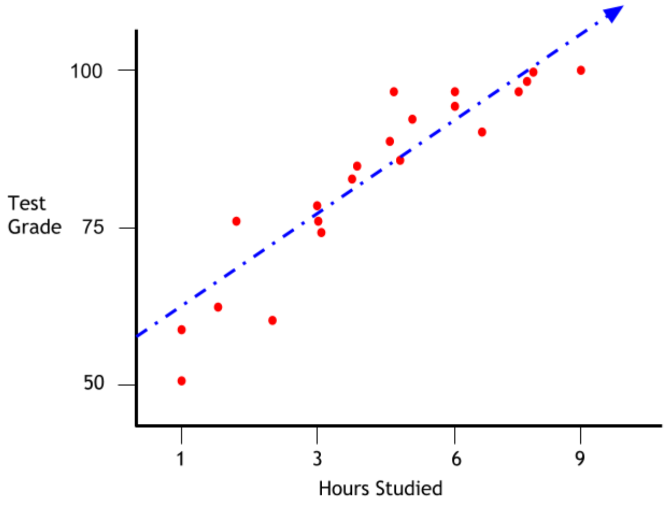
Department of Computer Science& Engineering, School of Engineering, Amritapuri

What, Where, How Linear Regression?

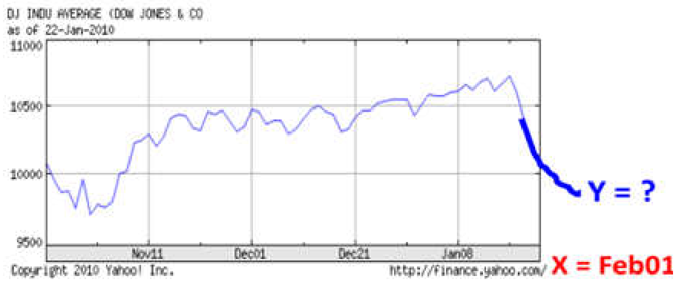
Supervised Learning: Classification vs Regression

- Approximate the mapping function y = f(X)
- Classification: output(s) is discrete.
- Regression: output is continuous or real valued





Sports
Science
News

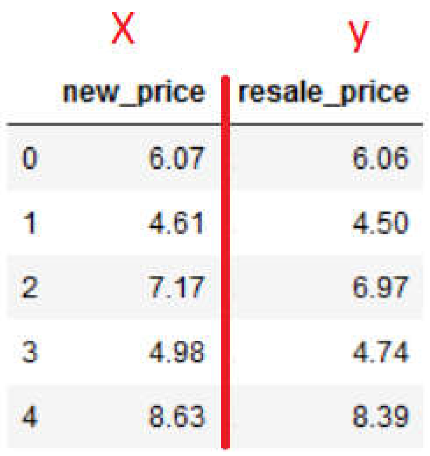


Classification:

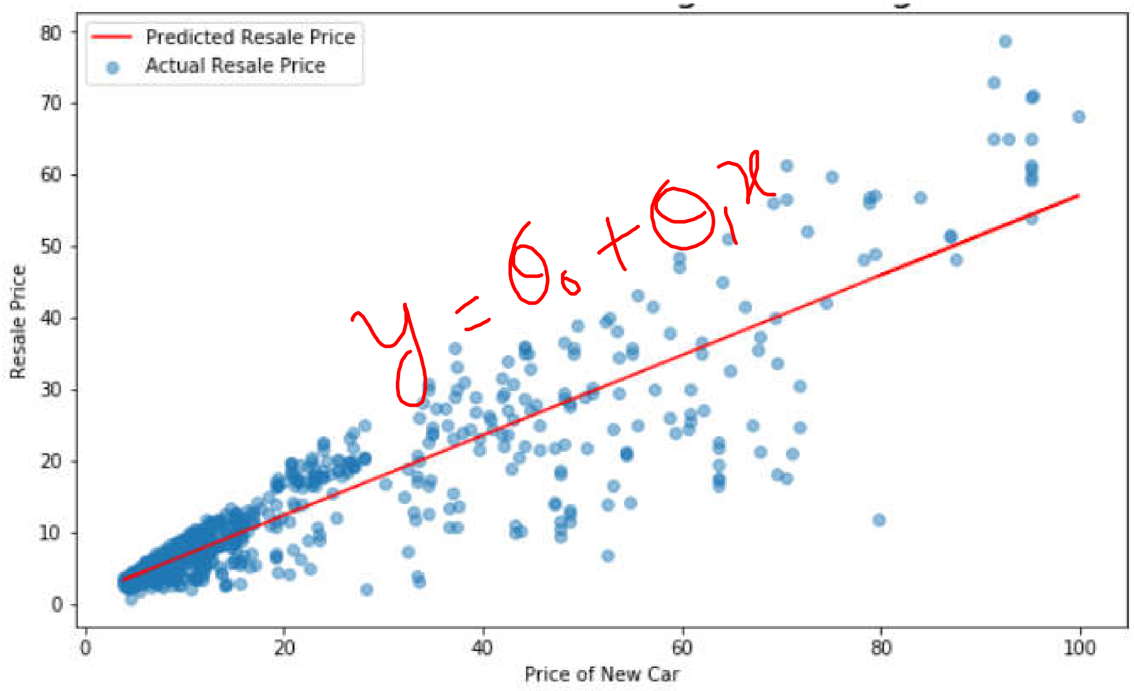
Regression:

What is Linear Regression?

- Linear Regression is a way finding relationship between one or more variables giving the model with the ability to predict outputs for inputs it has never seen before.



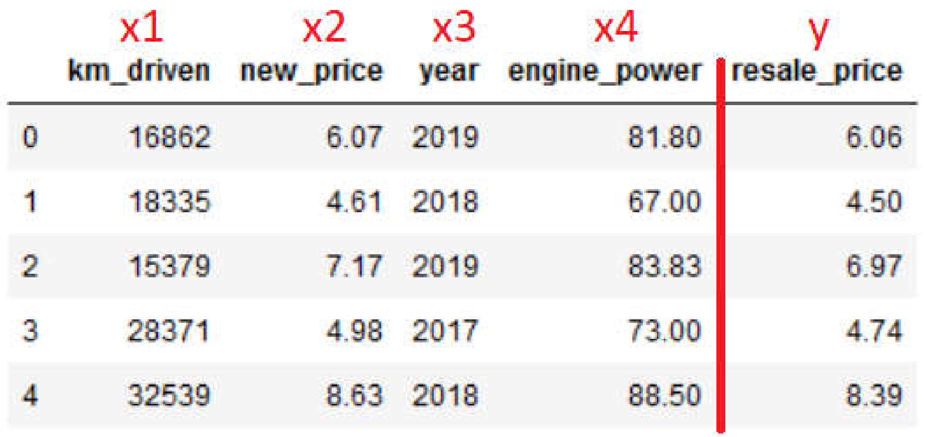
	X	y
	new_price	resale_price
0	6.07	6.06
1	4.61	4.50
2	7.17	6.97
3	4.98	4.74
4	8.63	8.39



Learn θ_0 and θ_1

Model and parameters

What is Linear Regression?



	x1	x2	x3	x4	y
	km_driven	new_price	year	engine_power	resale_price
0	16862	6.07	2019	81.80	6.06
1	18335	4.61	2018	67.00	4.50
2	15379	7.17	2019	83.83	6.97
3	28371	4.98	2017	73.00	4.74
4	32539	8.63	2018	88.50	8.39

- Model the variable Y in terms of four other variables X1, X2, X3, X4 as **Y = *f*(X1, X2, X3, X4)- multiple regression.**
- Y is called Dependent Variable or Response Variable
- X1, X2, X3,X4 called Independent Variable or Explanatory Variable
- Goal: determine the mathematical relationship between response variables Y and regressors Xi

Linear Regression Hypothesis

univariate

$$y = \theta_0 + \theta_1 x_1$$

multivariate

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$$

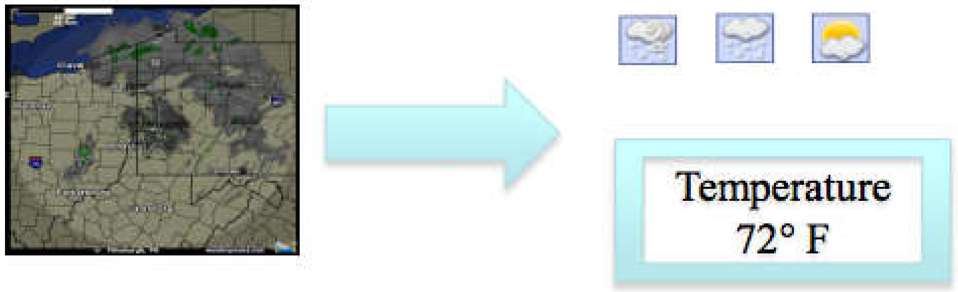
Assume x_0 = 1

Where can Linear Regression be used?

Understand and quantify the relationship between variables in data

Stock market

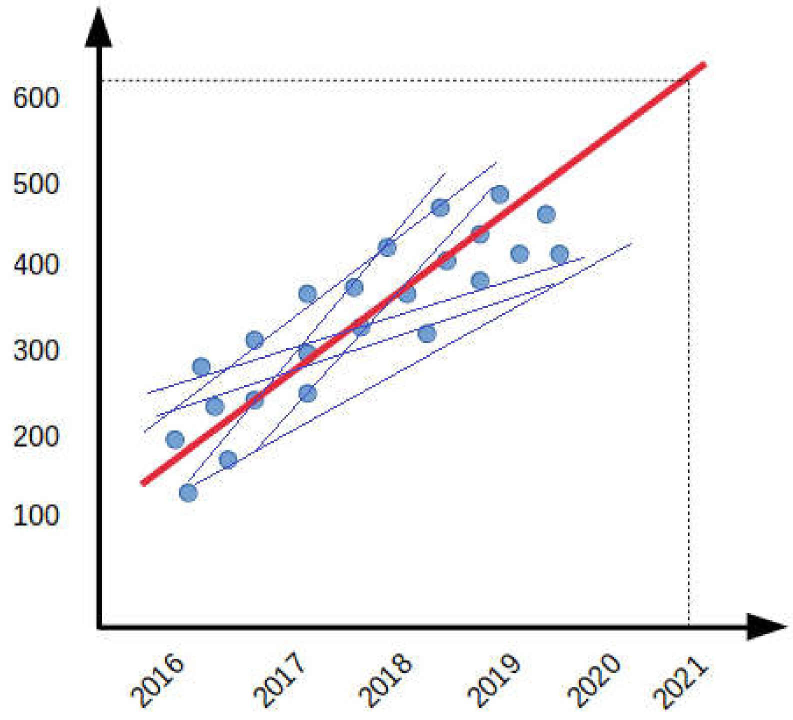
Weather prediction



Predict the temperature at any given location

How does linear regression works?

- Find best fitting line
- For best fit line, error between the predicted values and the observed values is minimum
- Define error function as cost function
- Learning algorithm find model parameter θ such that it minimize the cost.

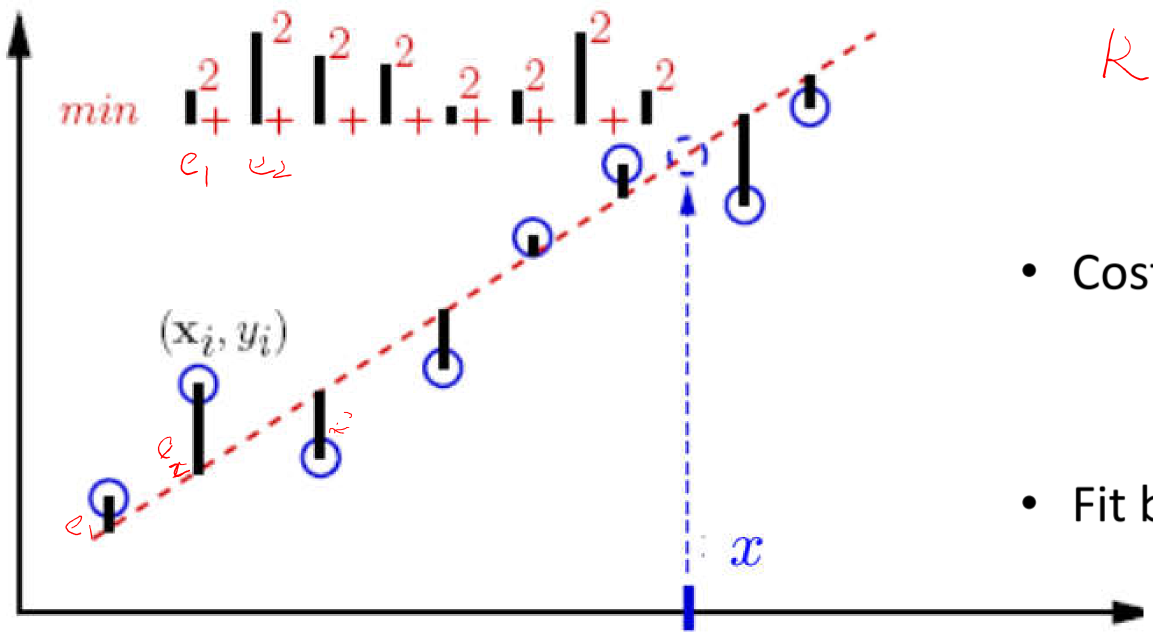


h(X) = θ0 + θ 1.X

Cost Function

$e_i = h_\theta(x_i) - y_i$

$RSS = e_1^2 + e_2^2 + \cdots + e_n^2$



- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

How do we determine the best fit line?

Linear Regression Summary

Data Set : <x1,x2, …, xd, y>

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j \quad \theta^T x$$

Assume x_0 = 1

Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$

Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

Derivative of Cost Function

Minimize cost function by taking its derivatives with respect to the θj's, and setting them to zero

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

$$(X\theta - y)^2$$

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$Cost = (X\theta - y)^T (X\theta - y)$$

$$\frac{\partial J_\theta}{\partial \theta} = \frac{\partial}{\partial \theta}\left[(X\theta - y)^T (X\theta - y)\right]$$

$$\frac{\partial J_\theta}{\partial \theta} = 2X^T X\theta - 2X^T y$$

$$Cost'(\theta) = 0$$

$$2X^T X\theta - 2X^T y = 0$$

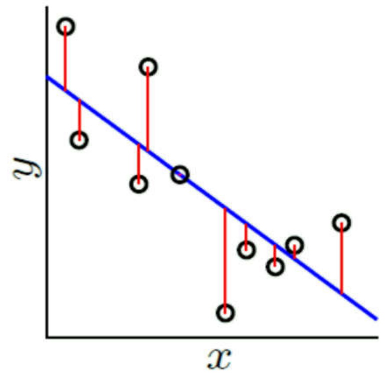
$$2X^T X\theta = 2X^T y$$

$$\boxed{\theta = \left(X^T X\right)^{-1} . \left(X^T y\right)}$$

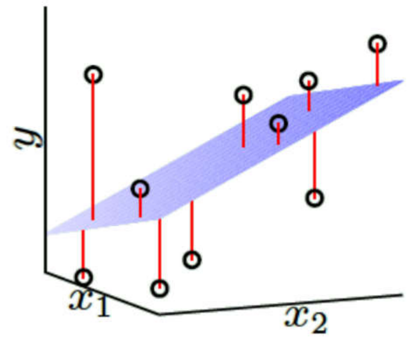
Ordinary Least Square Estimation

OLS Linear Regression Algorithm:

1. From the training data set, construct the input matrix ***X*** and the output vector ***Y***



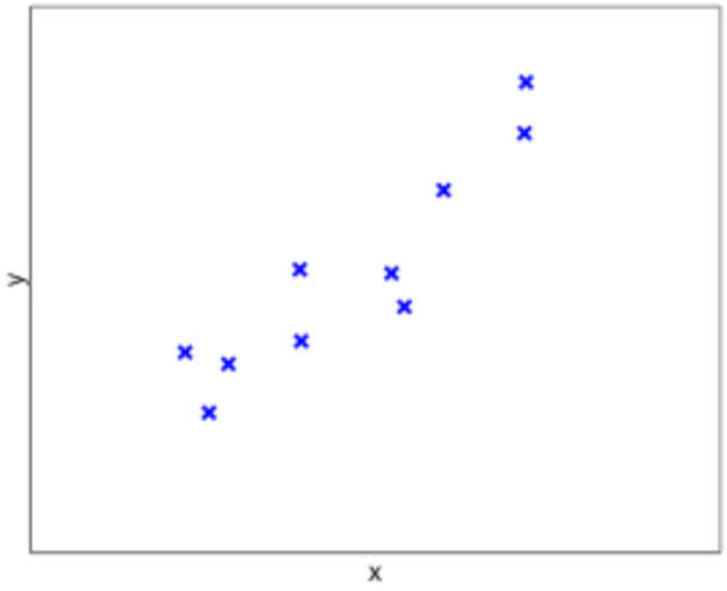
2. Assuming $\boldsymbol{X}^T\boldsymbol{X}$ is invertible (positive definite and non-singular), compute $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$



3. Return $\hat{\theta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$

Numerical Example

$$\theta = (X^T X)^{-1} X^T \vec{y}.$$



Observe x	*Predict* y
0.86	2.49
0.09	0.83
-0.85	-0.25
0.87	3.10
-0.44	0.87
-0.43	0.02
-1.1	-0.12
0.40	1.81
-0.96	-0.83
0.17	0.43

Numerical Example

$$X^T X =$$

$$\begin{bmatrix} 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0.86 & 1 \\ 0.09 & 1 \\ -0.85 & 1 \\ 0.87 & 1 \\ -0.44 & 1 \\ -0.43 & 1 \\ -1.10 & 1 \\ 0.40 & 1 \\ -0.96 & 1 \\ 0.17 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4.95 & -1.39 \\ -1.39 & 10 \end{bmatrix}$$

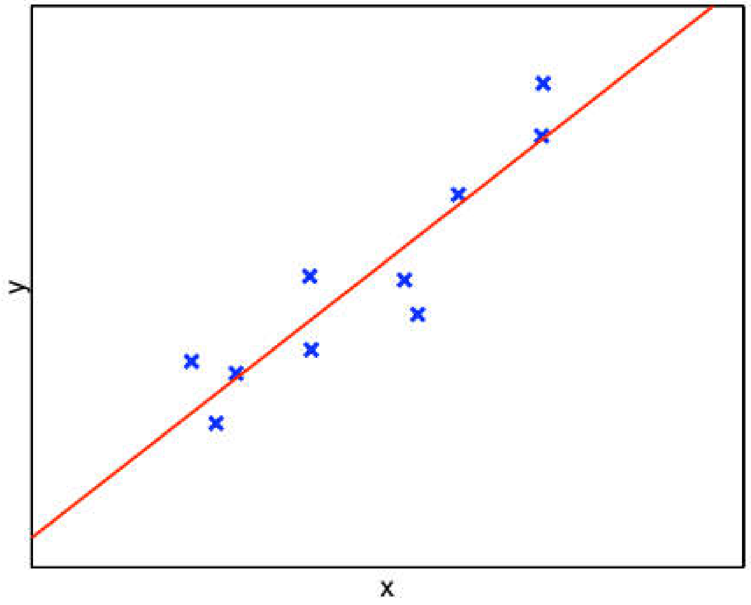
$$X^T Y =$$

$$\begin{bmatrix} 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

$$= \begin{bmatrix} 6.49 \\ 8.34 \end{bmatrix}$$

$$\theta = (X^T X)^{-1} X^T Y = \begin{bmatrix} 4.95 & -1.39 \\ -1.39 & 10 \end{bmatrix}^{-1} \begin{bmatrix} 6.49 \\ 8.34 \end{bmatrix} = \begin{bmatrix} 1.60 \\ 1.05 \end{bmatrix}$$

So the best fit line is $y = 1.60x + 1.05$.



Computational cost of OLS

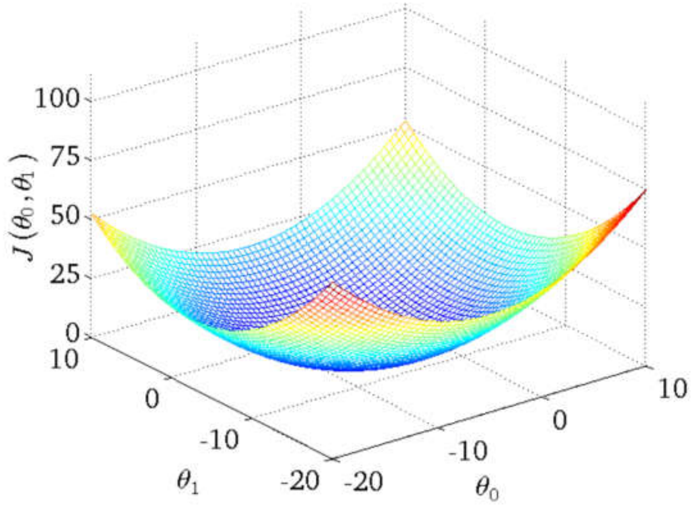
- What operations are necessary?
  - Overall: 1 matrix inversion + 3 matrix multiplications
- What if X is too big to compute this explicitly (e.g. m ~ 10^6)?
- Assuming for now that X is reasonably small so computation and memory are not a problem. Can we always evaluate this?
  - To have a unique solution, we need XTX to be nonsingular.

Optimization Objective of Linear Regression

Cost Function

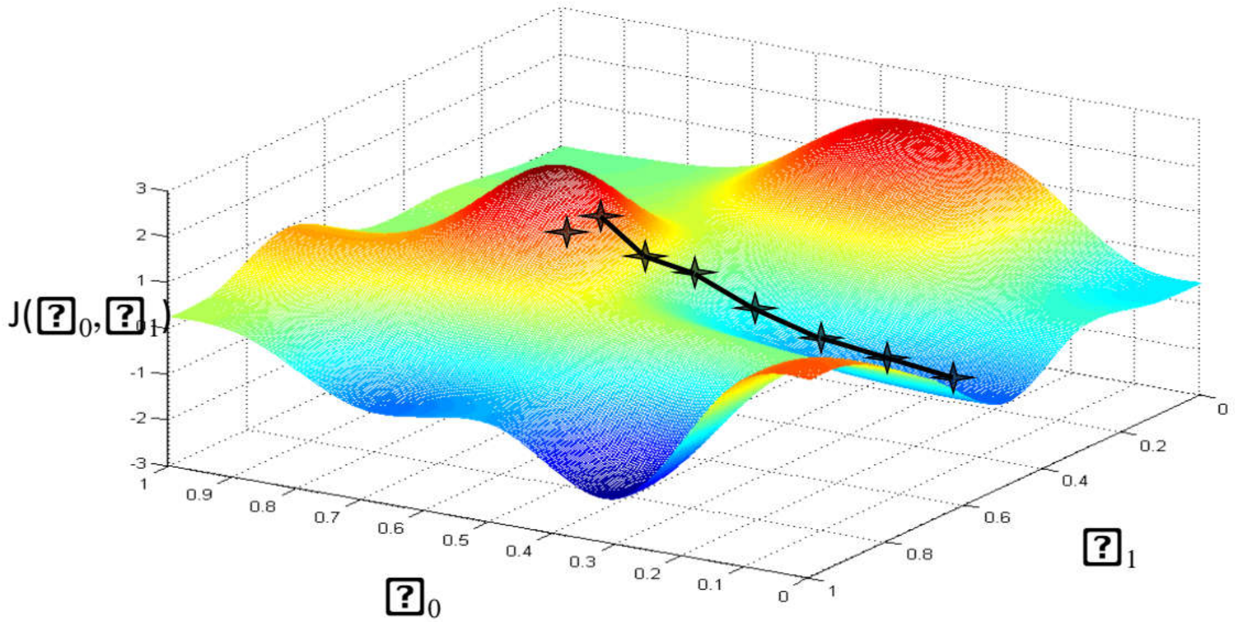
$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$



Search Algorithm

- Choose initial value for $\boldsymbol{\theta}$
- Until we reach a minimum:
  - Choose a new value for $\boldsymbol{\theta}$ to reduce $J(\boldsymbol{\theta})$

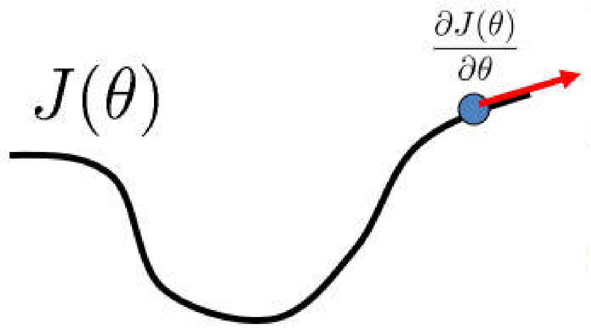


Instead of directly finding that minimum (using the closed-form equation), we can take small steps towards the minimum

How to choose direction of search?

Gradient is the rate of change of a function. It's a vector (a direction to move) that

- Points to the direction of greatest increase of a function
- Is zero at a local maximum or local minimum (because there is no single direction of increase)



- Choose a direction in which J(θ) is decreasing
- Derivative $\frac{\partial J(\theta)}{\partial \theta}$

- Positive => increasing
- Negative => decreasing

Gradient of Cost Function

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

$$= \frac{1}{2m}\sum_j (\overbrace{y^{(j)} - \theta_0 \underline{x}_0^{(j)} - \theta_1 \underline{x}_1^{(j)} - \dots}^{e_j(\theta)})^2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial}{\partial \theta_0}\frac{1}{2m}\sum_j (\, e_j(\theta)\,)^2$$

$$= \frac{1}{2m}\sum_j 2e_j(\theta)\,\frac{\partial}{\partial \theta_0} e_j(\theta)$$

$$\frac{\partial}{\partial \theta_0} e_j(\theta) = \underset{0}{\cancel{\frac{\partial}{\partial \theta_0} y^{(j)}}} - \frac{\partial}{\partial \theta_0}\theta_0 x_0^{(j)} - \underset{0}{\cancel{\frac{\partial}{\partial \theta_0}\theta_1 x_1^{(j)}}} - \dots$$

$$= -x_0^{(j)}$$

$$\frac{\partial J}{\partial \theta_0} = (h_\theta(x) - y)\, x_j$$

Gradient Descent Algorithm

Gradient descent is an iterative optimization algorithm

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

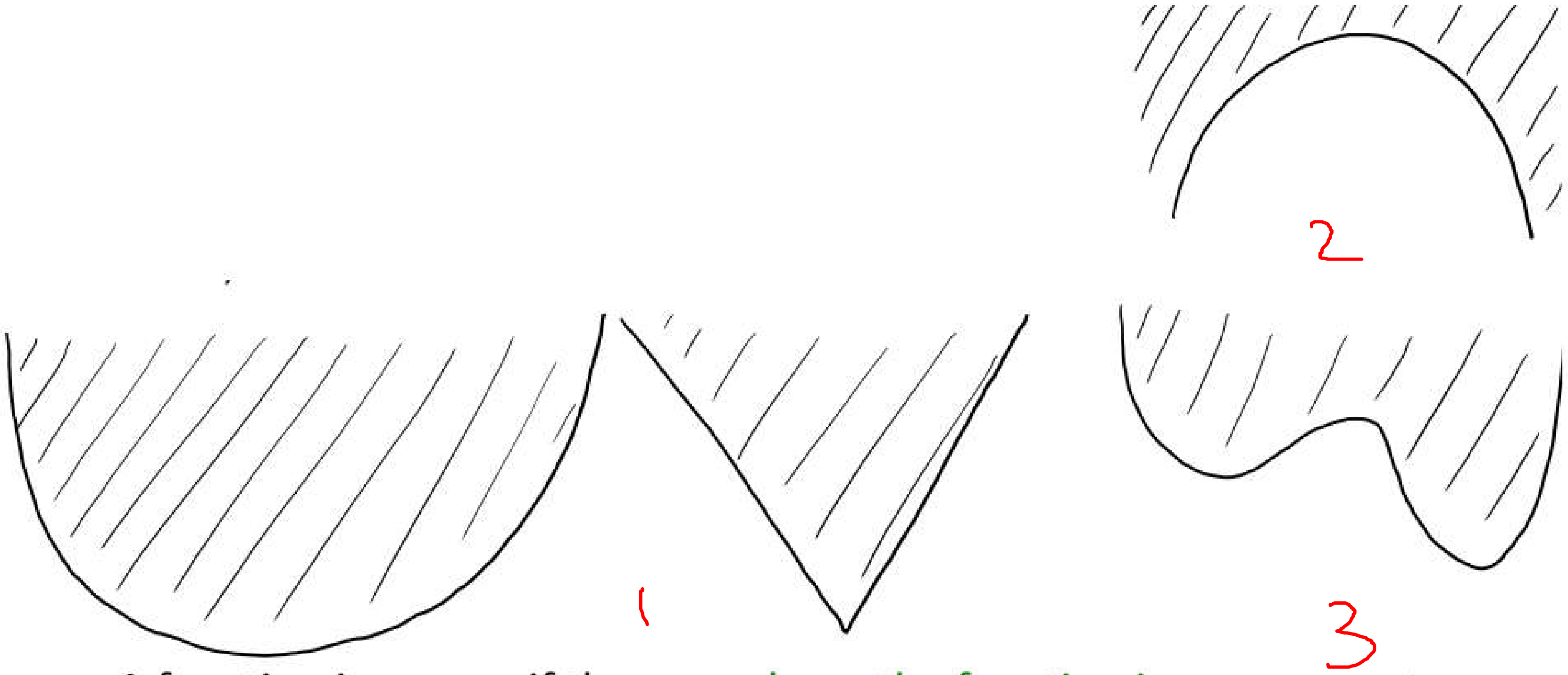
$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update for j = 0 ... d

– It starts with a "guess" θ^0.
– It uses the gradient $\nabla J(\theta^0)$ to generate a better guess θ^0.
– It uses the gradient $\nabla J(\theta^1)$ to generate a better guess θ^1.
– It uses the gradient $\nabla J(\theta^1)$ to generate a better guess θ^1.
...
– The limit of θ^t as 't' goes to ∞ has $\nabla J(\theta^t) = 0$.

- It converges to a global optimum if 'J' is "convex".

Convex Functions



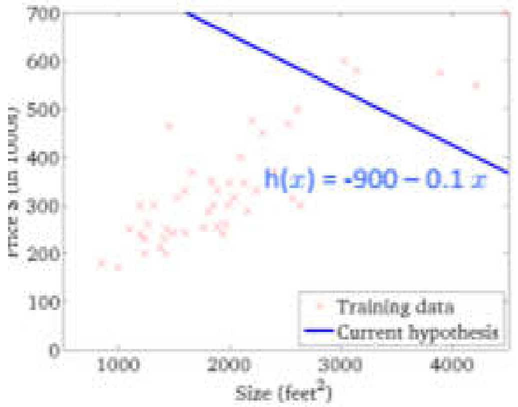
- A function is convex if the area above the function is a convex set.
  - All values between any two points above function stay above function.

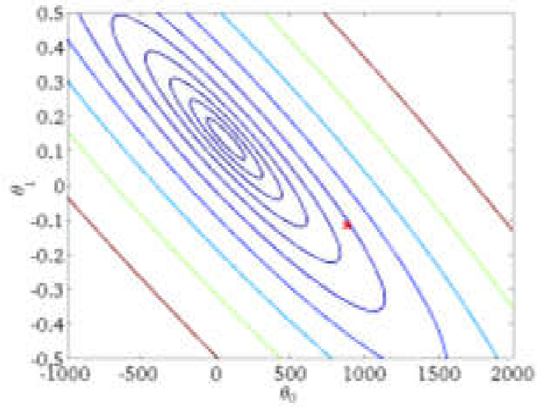
How do we know if a function is convex?

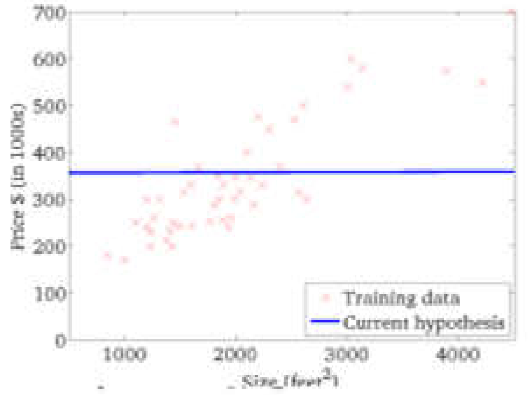
- Some useful tricks for showing a function is convex:
  - 1-variable, twice-differentiable function is convex iff f''(w) $\geq$ 0 for all 'w'.
  - A convex function multiplied by non-negative constant is convex.
  - Norms and squared norms are convex.
  - The sum of convex functions is a convex function.
  - The max of convex functions is a convex function.
  - Composition of a convex function and a linear function is convex.

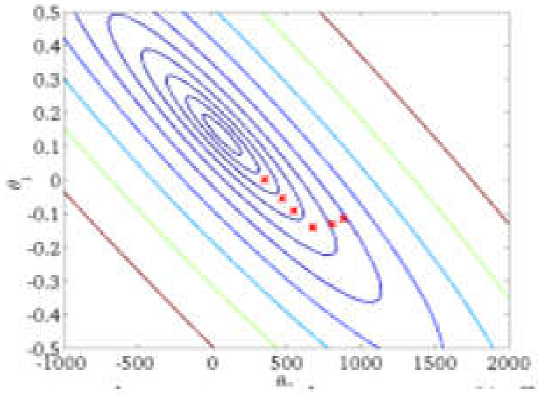
•not true that multiplication of convex functions is convex:

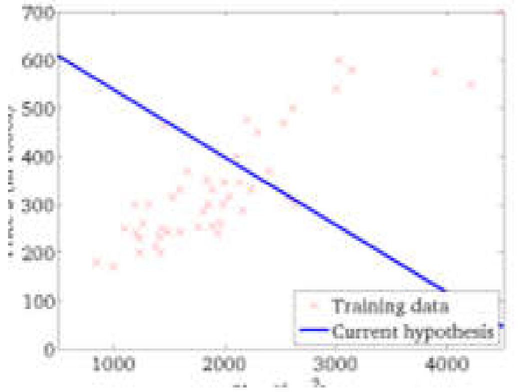
- If f(x)=x (convex) and g(x)=x2 (convex), f(x)g(x) = x3 (not convex).

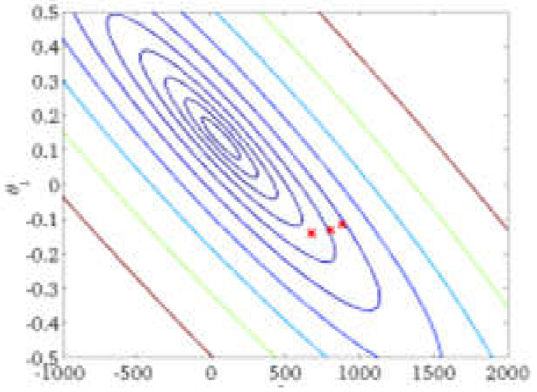


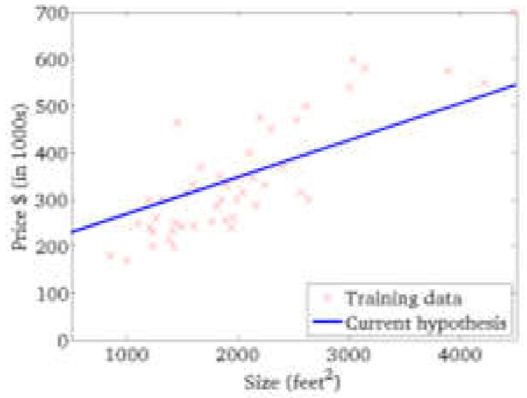


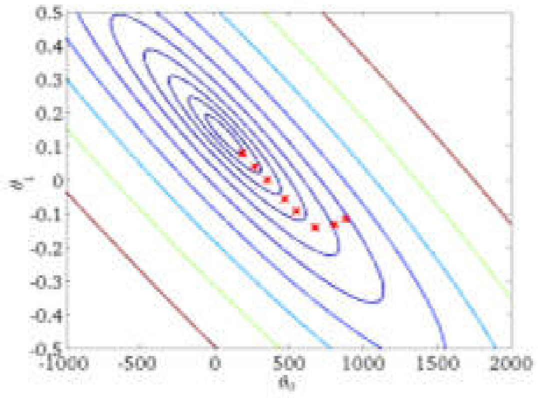






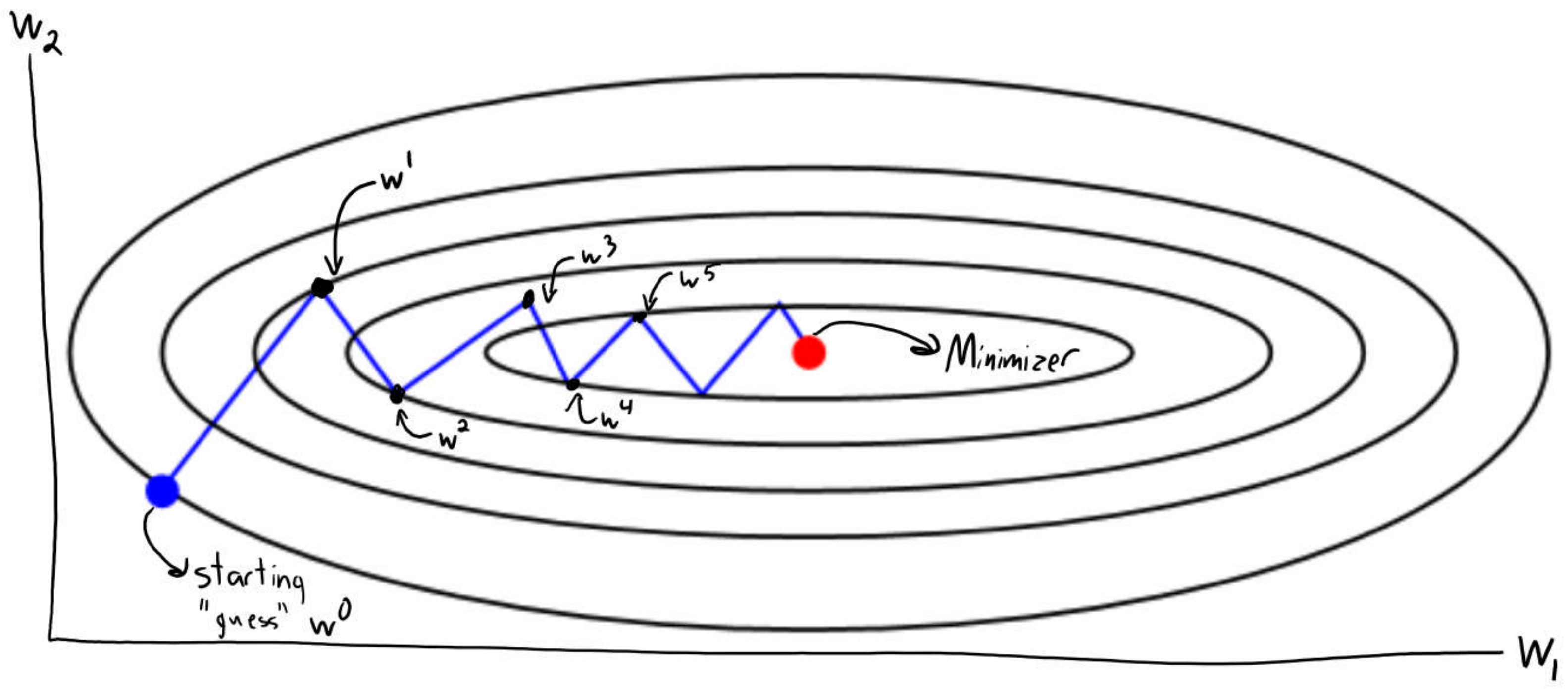




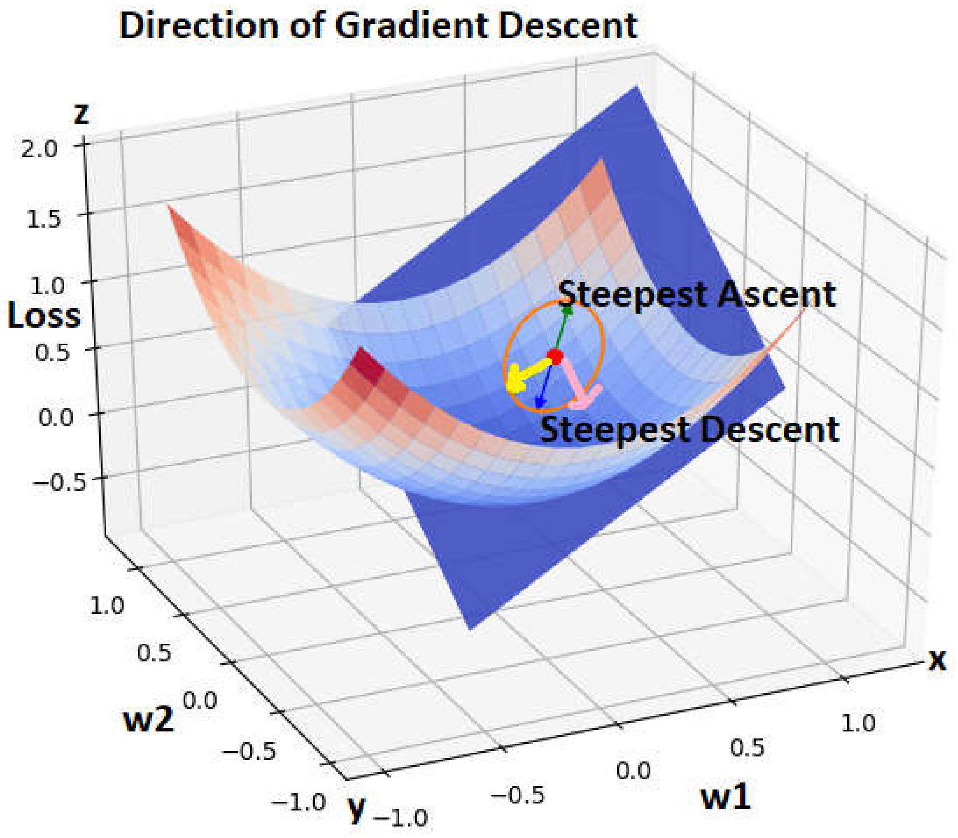


Data space   Parameter Space

Gradient Descent Summary



Gradient Descent: Review



- Assume we have only two parameters
- Optimization is finding parameters for which the value of cost function is minimum
- Find the parameters at the bottom most point the function
- This direction of ascent is given by the Gradient at that point.
- Exact opposite of that give direction of descent
- decide the size of the step to take

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

Gradient descent Algorithm

1.initialize the model parameters with some random values.

2.Measure how the cost function changes with change in it's parameters.

compute the partial derivatives of the cost function w.r.t to the parameters $\vartheta_0, \vartheta_1, \ldots, \vartheta_n$

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h(x^i) - y^i)$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m}(h(x^i) - y^i)x_1^i$$

similarly, the partial derivative of the cost function w.r.t to any parameter can be denoted by

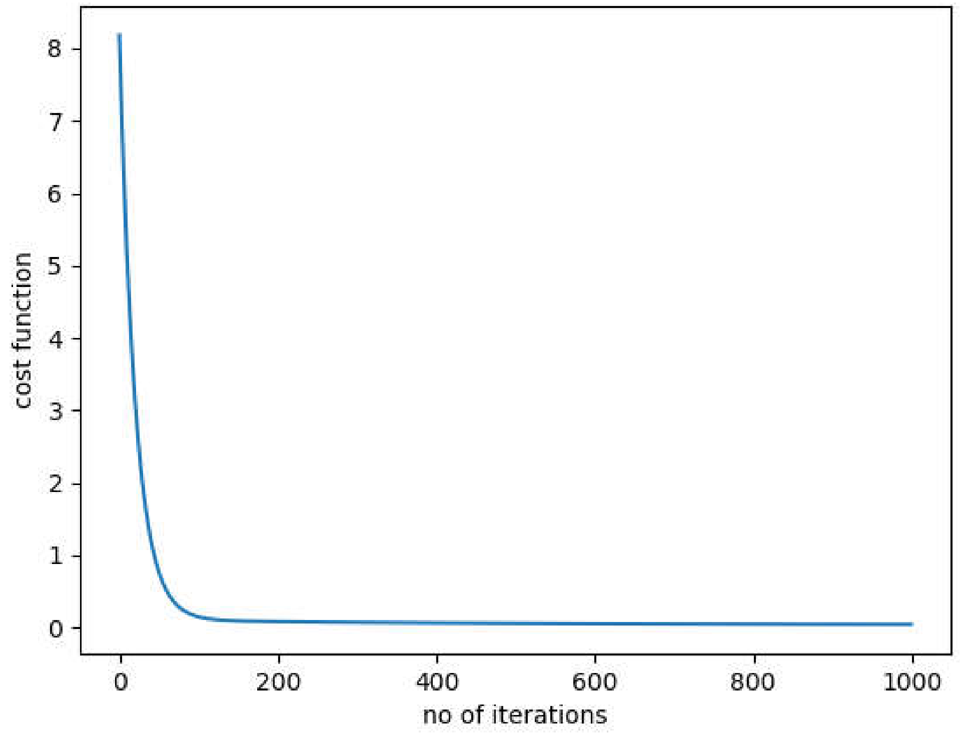
$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h(x^i) - y^i)x_j^i$$

3. After computing the derivative we update the parameters as given below

$$\theta_0 = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}(h(x^i) - y^i) \qquad \theta_1 = \theta_1 - \frac{\alpha}{m}\sum_{i=1}^{m}(h(x^i) - y^i)x_1^i$$

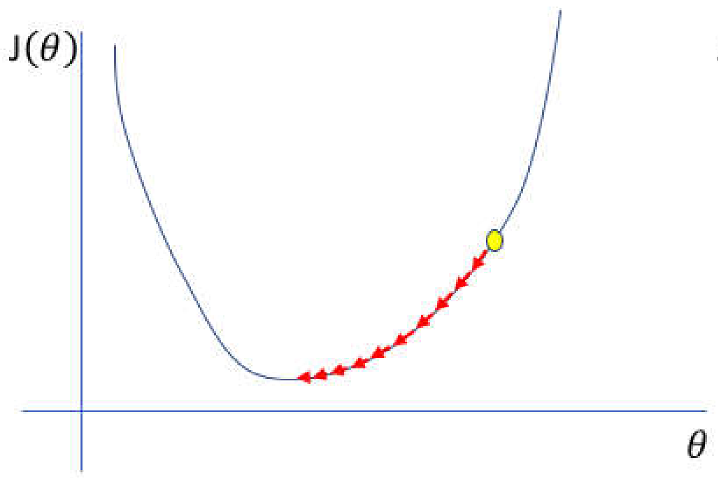
repeat the steps 2,3 until the cost function converges to the minimum value

Convergence of cost function



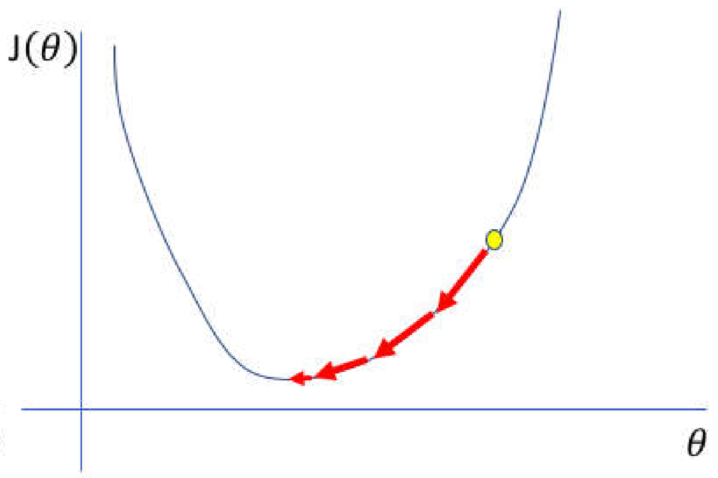
Impact of learning rate

Too low



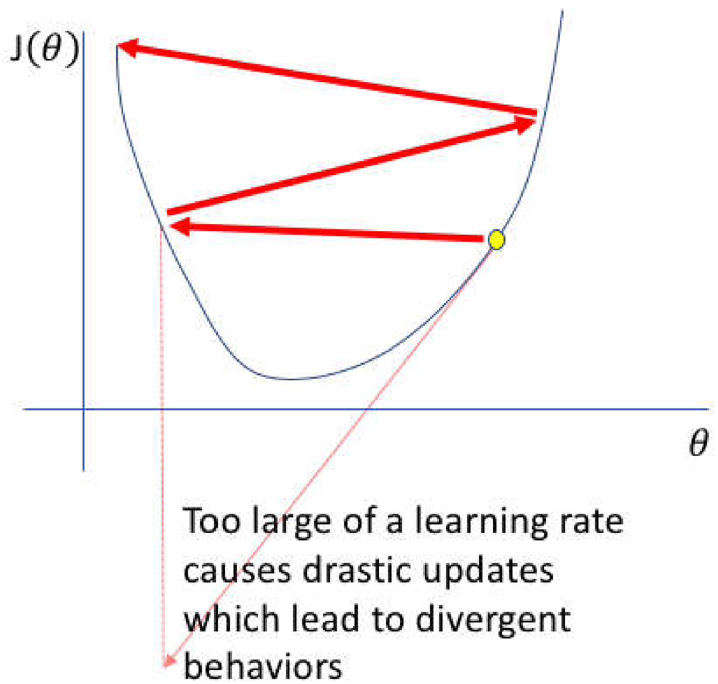
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

Too high

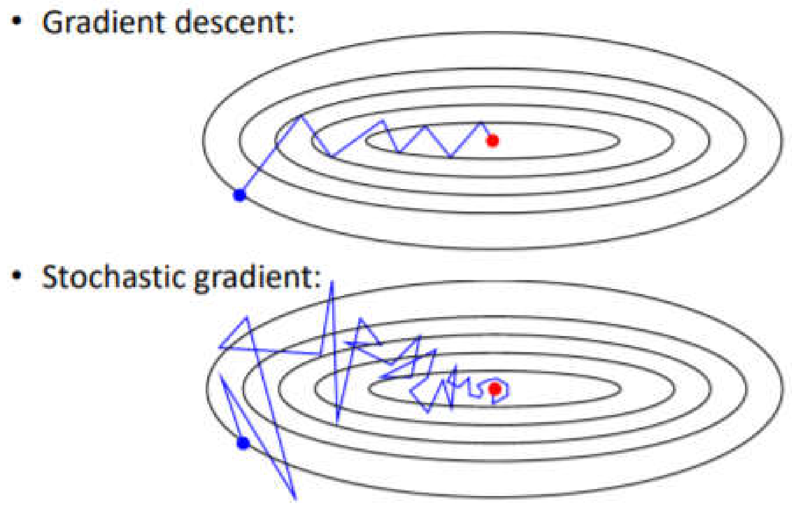


Too large of a learning rate causes drastic updates which lead to divergent behaviors

Variants of Gradient Descent

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini Batch Gradient Descent

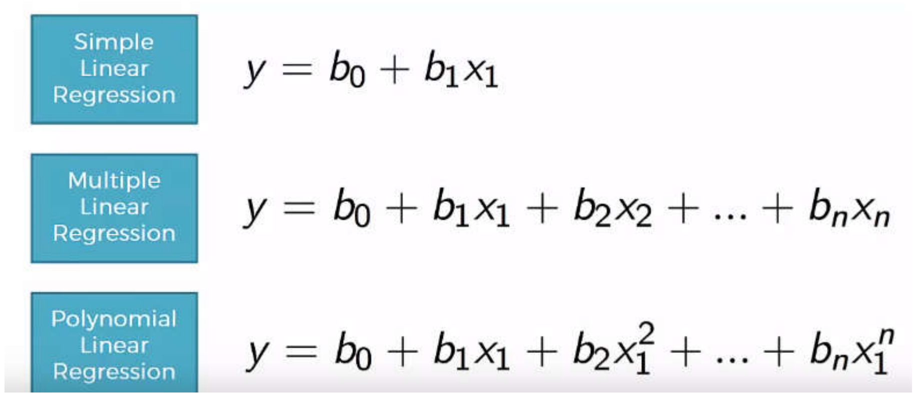
$$\frac{\partial J}{\partial \theta_0} \quad = \quad (h_\theta(x) - y)\, x_j$$



Gradient descent optimization algorithms

- Momentum
- Adagrad
- Adadelta
- RMSprop

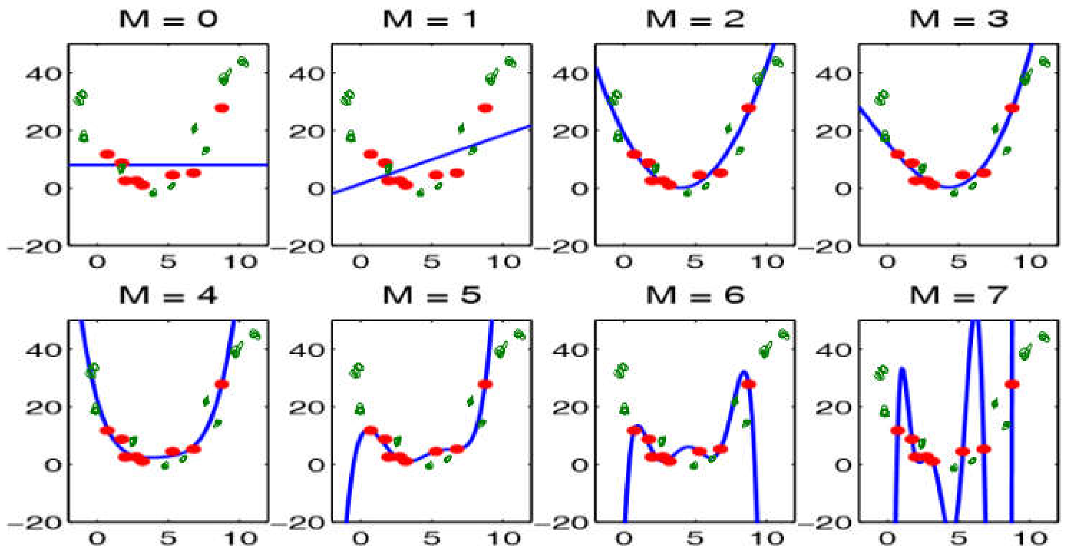
Types of Regression Models



Simple Linear Regression $y = b_0 + b_1 x_1$

Multiple Linear Regression $y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$

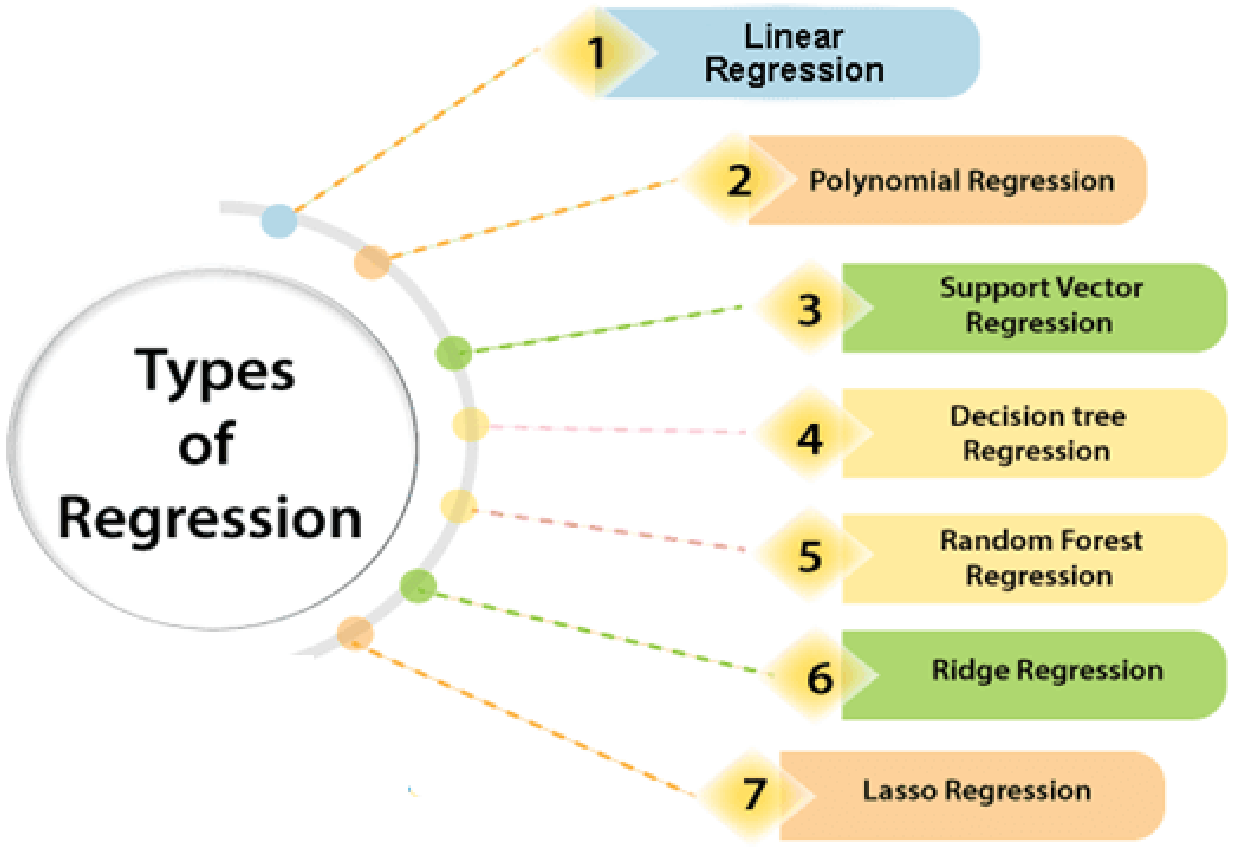
Polynomial Linear Regression $y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$



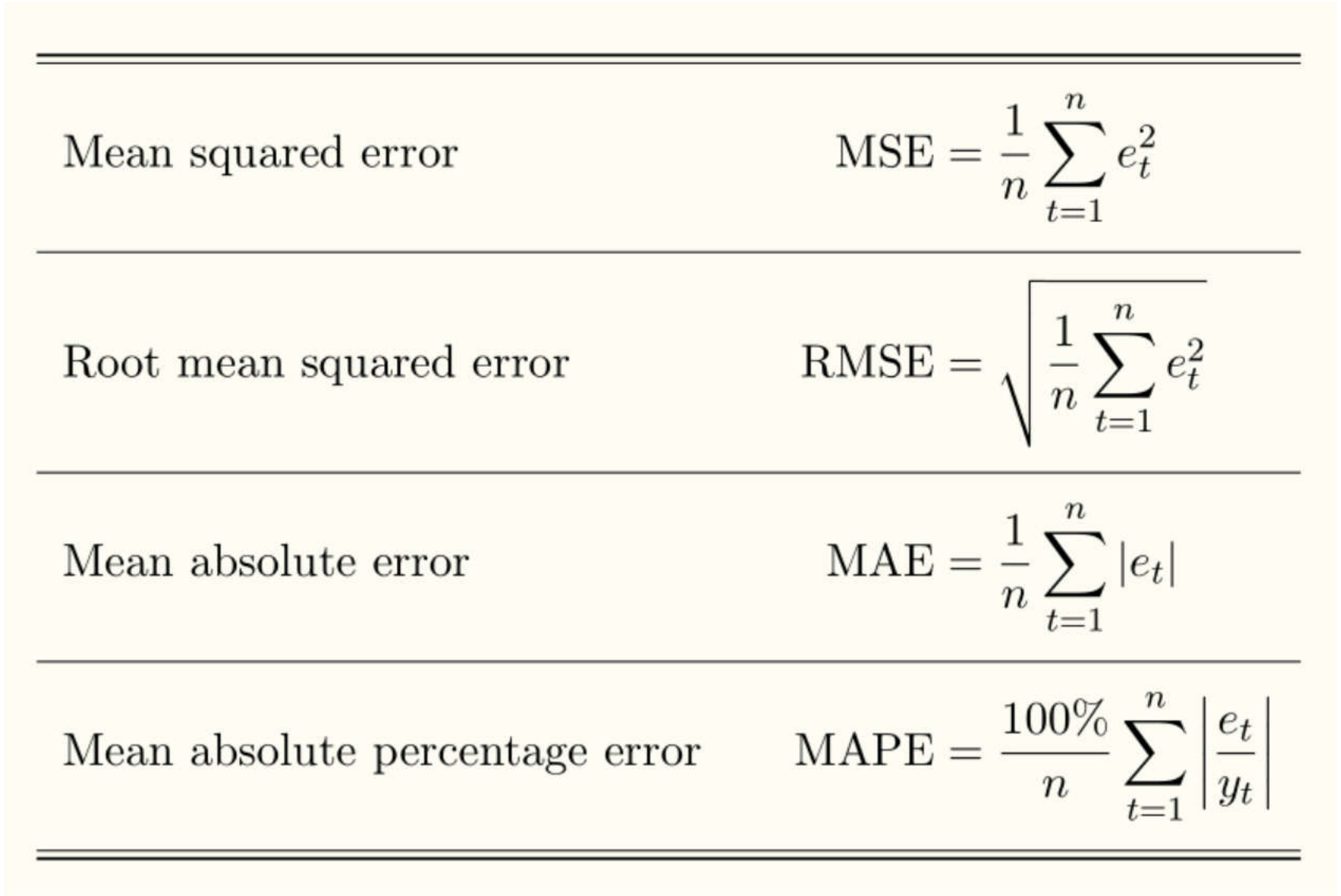
Assumptions in linear regression model

- **Linearity**: The relationship between X and the mean of Y is linear.
- **Homoscedasticity**: The variance of residual is the same for any value of X.
- **Independence**: Observations are independent of each other.
- **Normality**: For any fixed value of X, Y is normally distributed.

Types of Regression Models



Evaluating the performance of the model



Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Python Scikit learn

```python
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```python
print(regressor.intercept_)
```

```python
print(regressor.coef_)
```

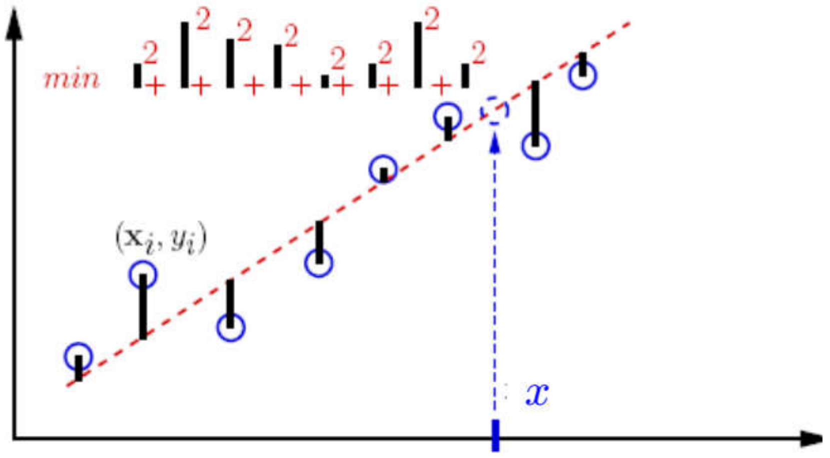
```python
y_pred = regressor.predict(X_test)
```

```python
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

Summary Linear Regression

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$$

- Hypothesis representation
- Ordinary Least Square Estimation
- Cost function and Properties
- Gradient Descent Algorithm
- Finding direction of minimum
- Impact of learning rate
- Variants of Gradient Descent Algorithm
- Performance evaluation
- Implementation in scikit



References

- *Kevin P. Murphy, “Machine Learning, a probabilistic perspective”, The MIT Press, 2012*
- *Lecture notes from Stanford http://cs229.stanford.edu/materials.html*
- *https://machinelearning-blog.com/2018/01/24/linear-regression/*