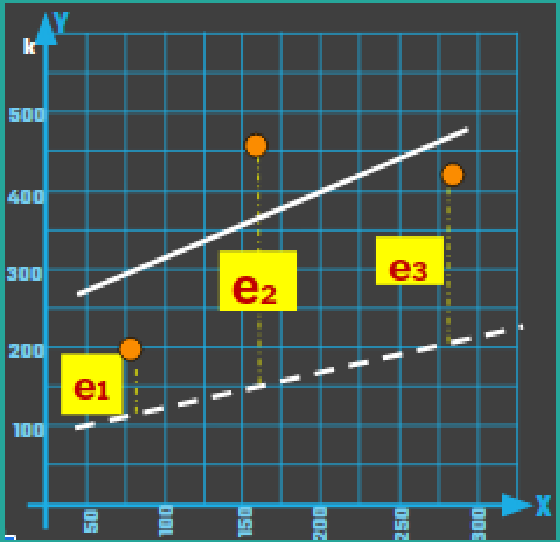
Regression

Outlines:

Linear Regression



x y

Area	Price (k)
100	200
150	400
200	450
250	550
300	450

Parametric solution

$y = m\,x + b$

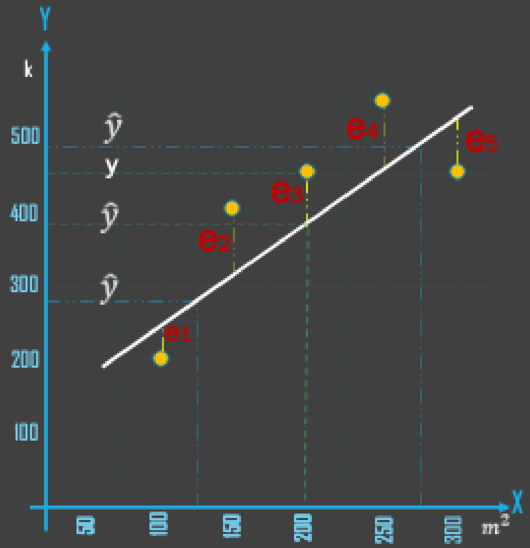
$y = w_0 + w_1\,x$ —> (1)

Error

$e = \hat{y} - y$

$e1 + e2 + e3 + e4 + e5$

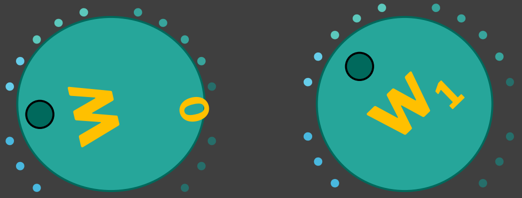
$E = \sum_{i=1}^{n} (\hat{y} - y) \rightarrow (2)$

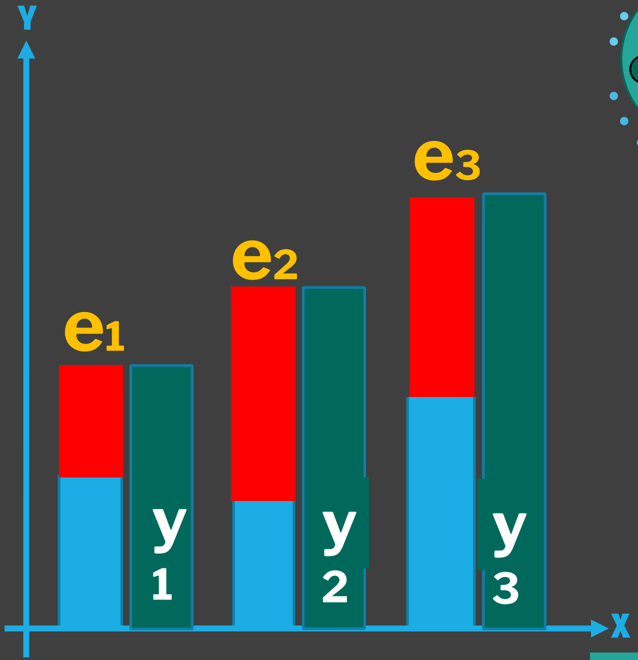


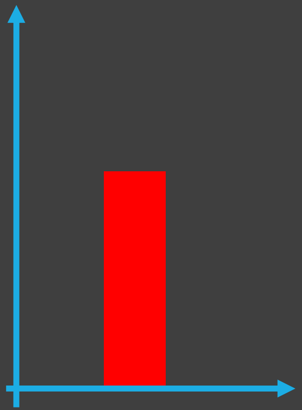
Hassan Badawy

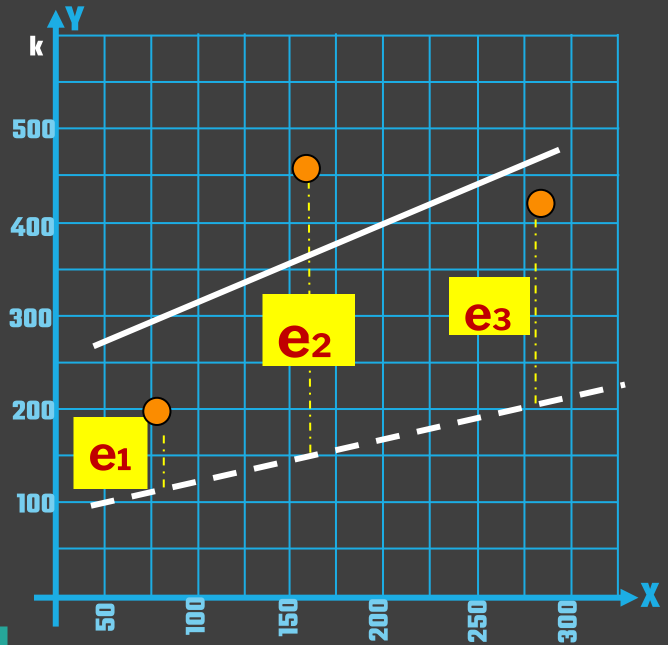
Title

$$y = w_0 + w_1 x$$







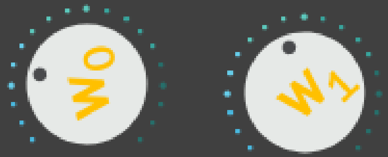


$$e = \sum_i e = \sum_i (\hat{y} - y)$$

Hassan Badawy

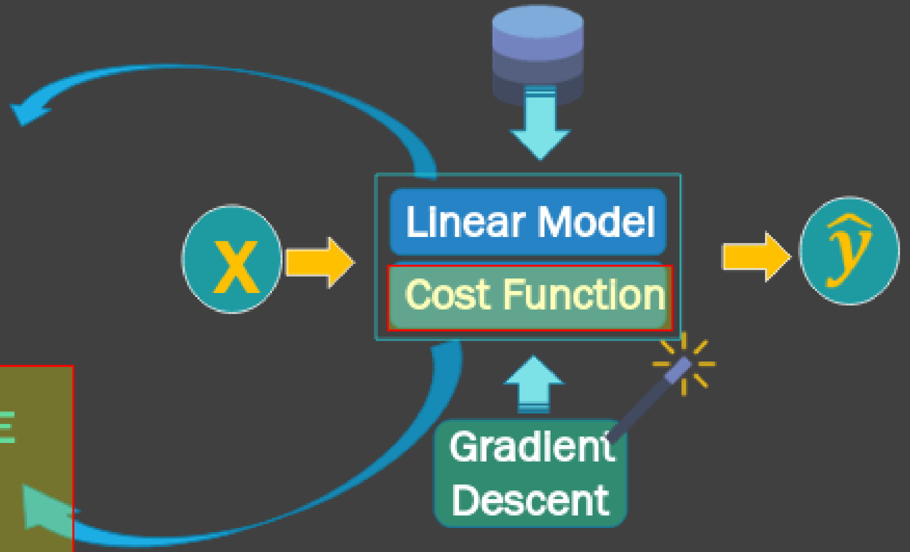
Linear Model

$$y = w_0 + w_1 x \longrightarrow (1)$$



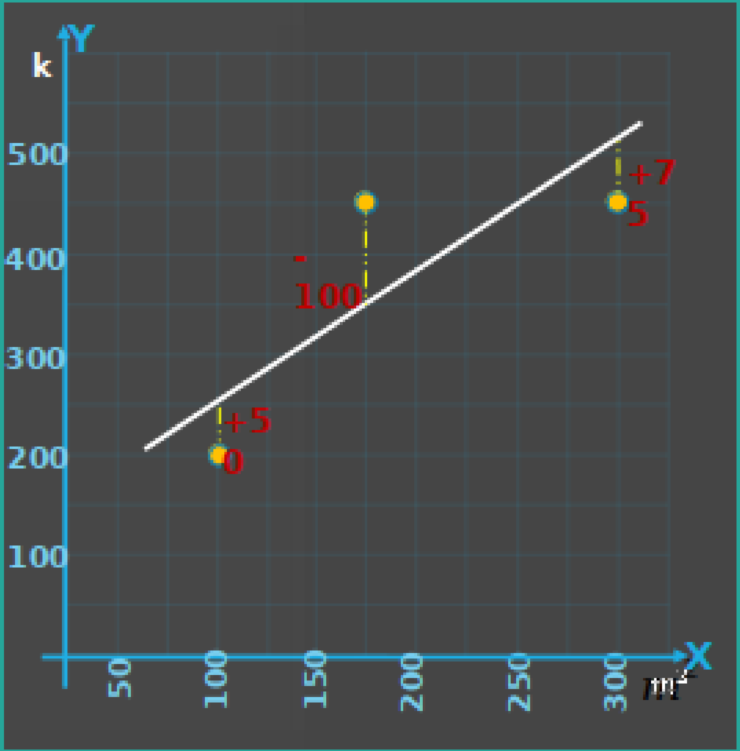
Error or Cost Function MSE

$$WMSE = \frac{1}{2n}\sum_{i=1}^{n} e_i^{2} = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$



Hassan Badawy

Error (cost) Functions



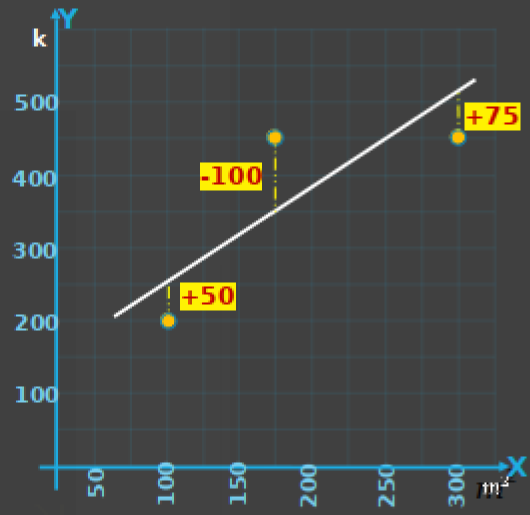
Standard Error

$$e_i = \hat{y}_i - y_i$$

$$E = e_1 + e_2 + e_3$$

$$E = +50 - 100 + 75 = 25$$

$$E = \sum_i e_i = \sum_{i=1}^{n} (\hat{y}_i - y_i)$$



Hassan Badawy

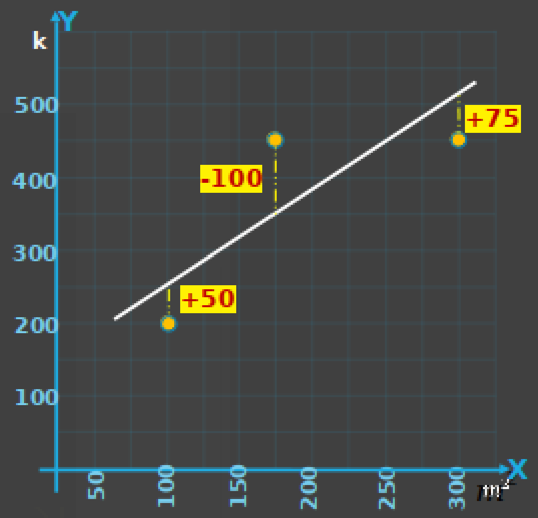
Mean Absolute Error MAE

$$MAE = \frac{|e_1|+|e_2|+|e_3|}{3}$$

$$MAE = \frac{|+50|+|-100|+|+75|}{3}$$

$$MAE = \frac{50+100+75}{3} = 75$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i| = \frac{1}{n}\sum_{i=1}^{n}|(\hat{y} - y)|$$



Hassan Badawy

Mean Square Error MSE

$$MSE = \frac{(e_1)^2+(e_2)^2+(e_3)^2}{3}$$

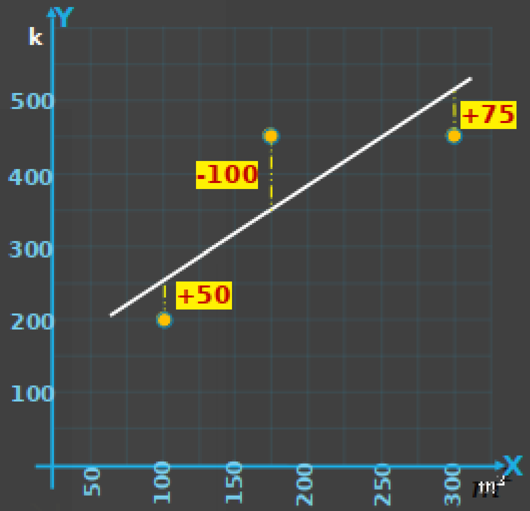
$$MSE = \frac{(+50)^2+(-100)^2+(+75)^2}{3}$$

$$MSE = \frac{2500+10000+5625}{3} = 6041.67$$

$$MSE = \frac{1}{2n}\sum_{i=1}^{n} e_i{}^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

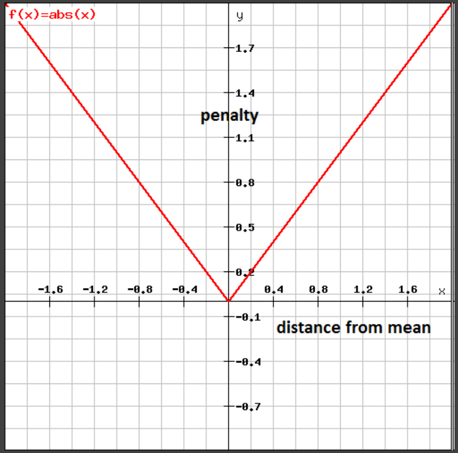
$$WMSE = \frac{MSE}{w} = \frac{MSE}{2} = 320.83$$

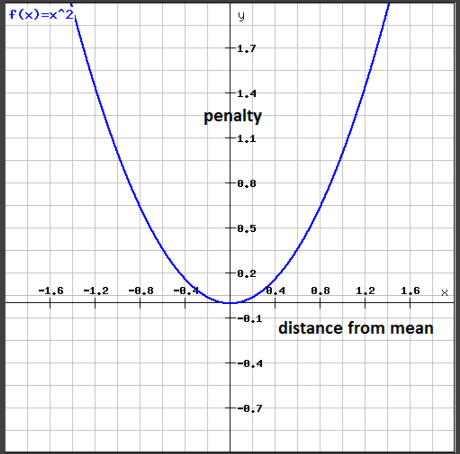
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i{}^2} = sqrt(6041.67)$$



Hassan Badawy

SE	MAE	MSE	WMSE	RMSE
25	75	6041	3020	78
$\sum_i e_i$	$\frac{1}{n}\sum_{i=1}^{n}\lvert e_i\rvert$	$\frac{1}{n}\sum_{i=1}^{n}{e_i}^2$	$\frac{1}{2n}\sum_{i=1}^{n}{e_i}^2$	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}{e_i}^2}$





Hassan Badawy

Why Squared Error?

Mean Error ME:

$$\frac{+4+4\ -4\ -4}{4}=0$$

ME = 0

Mean Absolute Error MAE:

$$\frac{|+4|+|+4|+|-4|+|-4|}{4}$$

$$\frac{4+4+4+4}{4}=4$$

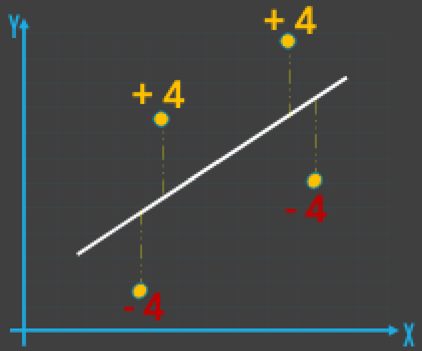
MAE = 4

Mean Square Error MSE

$$\frac{(+4)^2+(+4)^2+(-4)^2+(-4)^2}{4}$$

$$\frac{16+16+16+16}{4}=16$$

MSE = 16



	ME	MAE	MSE
[+4, +4 , -4, -4]	0	4	16
[+7, +1 , -2, -6]	0	4	22.5

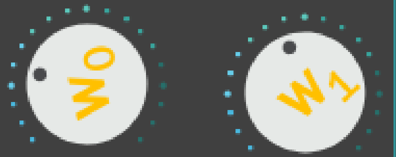
- Accurate
- Sensitive
- Differentiable
- Convex

Hassan Badawy

Title

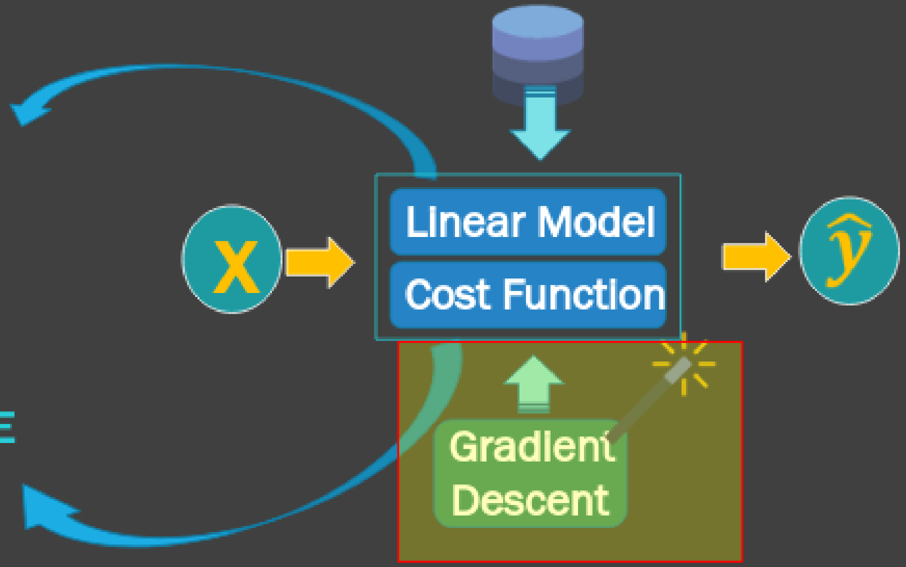
Linear Model

$y = w_0 + w_1 x$ —> (1)



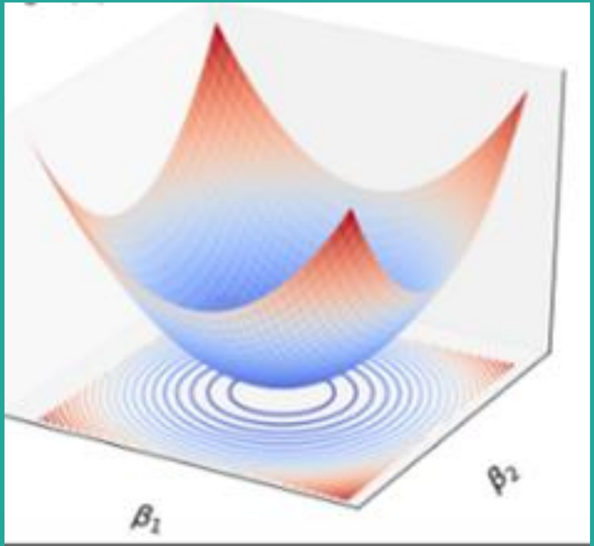
Error or Cost Function MSE

$$WMSE = \frac{1}{2n}\sum_{i=1}^{n} e_i^{\,2} = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$



Hassan Badawy

Gradient Descent

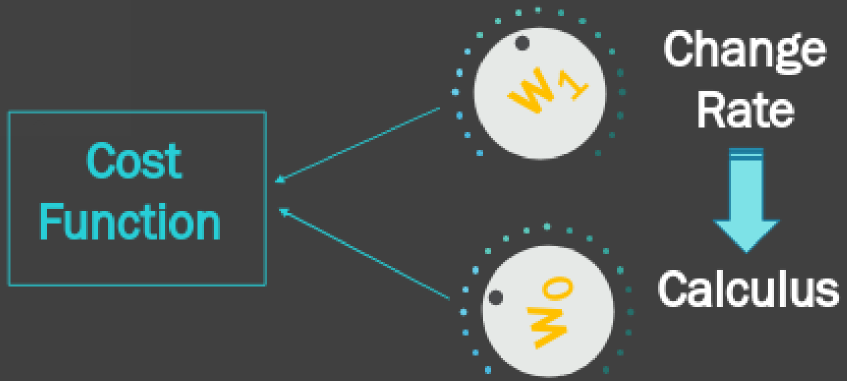


Error or Cost Function MSE

$$MSE = \frac{1}{2n}\sum_{i=1}^{n} e_i{}^2 = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Linear Model

$$y = w_0 + w_1 x$$



Hassan Badawy

Cost Function J(w)

$$WMSE = \frac{1}{2n}\sum_{i=1}^{n} e_i{}^2 = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$$J(w) = \frac{1}{2n}\sum_{i=1}^{n}(\mathrm{h}(w)_i - y_i)^2$$

Gradient Descent $\frac{\delta j(w)}{\delta w}$

$$\frac{\delta j(w)}{\delta w}$$

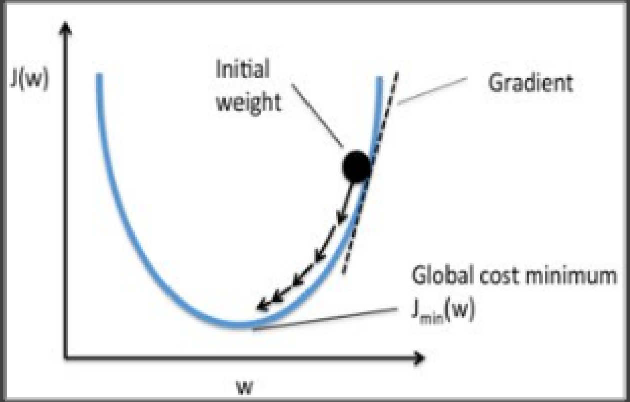
w_0

Partial Derivative

w_1

Linear Model

$\hat{y}_i = h(w) = w_0 + w_1\,x$



Hassan Badawy

$$\frac{\delta j(w)}{\delta w_0} = \frac{\delta}{\delta w} J(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(\left(\underset{1}{\overset{v}{\boxed{w_0}}} + \underset{0}{\overset{c}{\boxed{w_1}}}\, x\right)_i - y_i\right)^2$$

$$\frac{\delta j(w)}{\delta w_0} = \frac{2}{2n}\sum_{i=1}^{n}\left(\left(w_0 + w_1\, x\right)_i - y_i\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\left(w_0 + w_1\, x\right)_i - y_i\right)$$

$$\frac{\delta j(w)}{\delta w_1} = \frac{\delta}{\delta w} J(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(\left(\underset{0}{\overset{c}{\boxed{w_0}}} + \underset{x}{\overset{v}{\boxed{w_1}}}\, x\right)_i - y_i\right)^2$$

$$\frac{\delta j(w)}{\delta w_1} = \frac{1}{n}\sum_{i=1}^{n}\left(\left(w_0 + w_1\, x\right)_i - y_i\right) x^i$$

Hassan Badawy

$$\frac{\delta j(w)}{\delta w_0} = \frac{1}{n}\sum_{i=1}^{n}\left(\left(w_0 + w_1 x\right)_i - y_i\right)$$

$$\frac{1}{n}\sum_{i=1}^{n}\left(\boxed{\left(w_0 + w_1 x\right)_i} - y_i\right) x_i$$

$$\boxed{h(w) = w_0 x_0 + w_1 x_1}$$

$$\frac{\delta j(w)}{\delta w_1} = \frac{1}{n}\sum_{i=1}^{n}\left(\left(w_0 + w_1 x\right)_i - y_i\right) x_i$$

$$\boxed{\frac{\delta j(w)}{\delta w} = \frac{1}{n}\sum_{i=1}^{n}\left(\left(w_0 + w_1 x\right)_i - y_i\right) x_i}$$

Gradient $\boxed{w = w - \alpha \frac{\delta j(w)}{\delta w}}$ Descent

Hassan Badawy

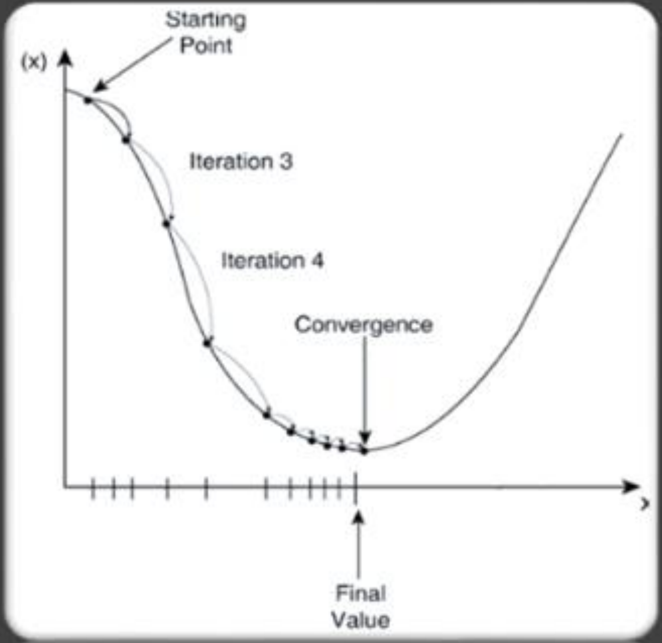
Gradient Descent Algorithm

1- Start with random W0, W1.

$$w_0 = w_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left((w_0 + w_1 x)_i - y_i \right) x_i$$

$$w_1 = w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left((w_0 + w_1 x)_i - y_i \right) x_i$$

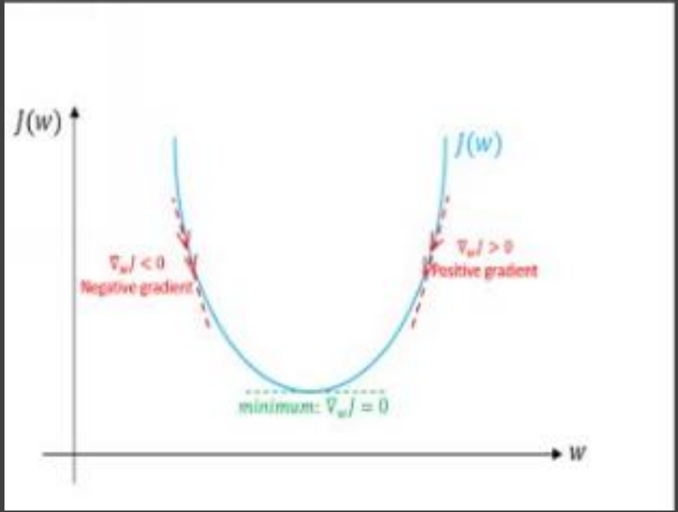
2- Update both W0, W1 Simultaneously.



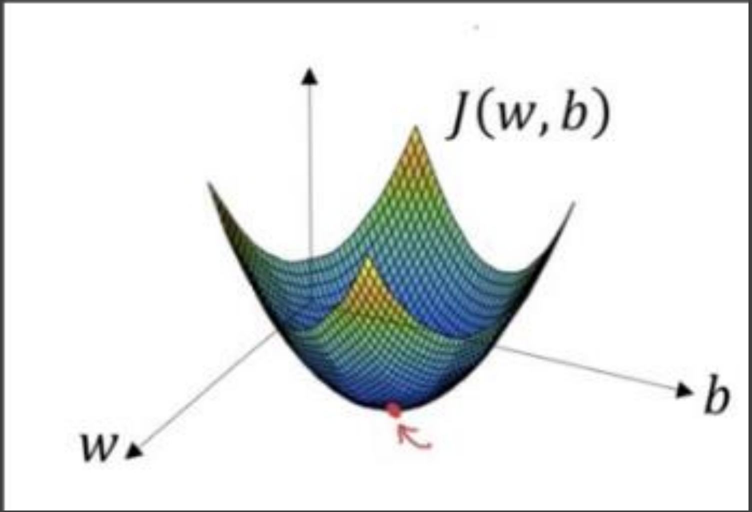
Hassan Badawy

$$w_1 = w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left((w_0 + w_1 x)_i - y_i \right) x_i$$

2d plan:



3d plan:



Hassan Badawy

$$w_1 = w_1 - \alpha \frac{1}{n}\sum_{i=1}^{n}\left((w_0 + w_1 x)_i - y_i\right)x_i$$

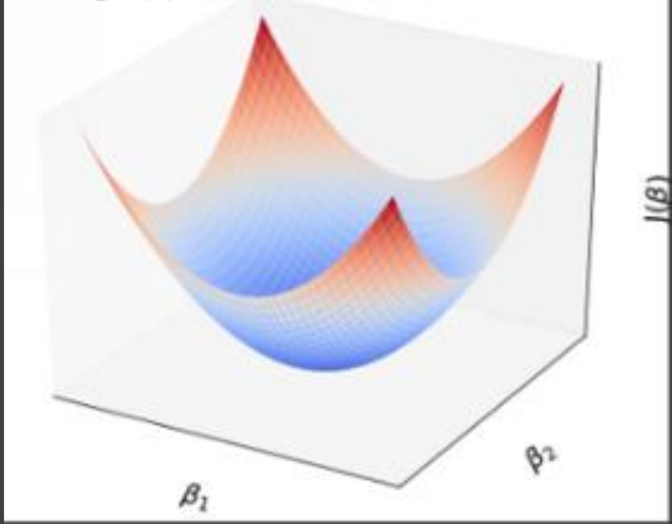


Fig 1(a): Gradient Descent in 3-dim

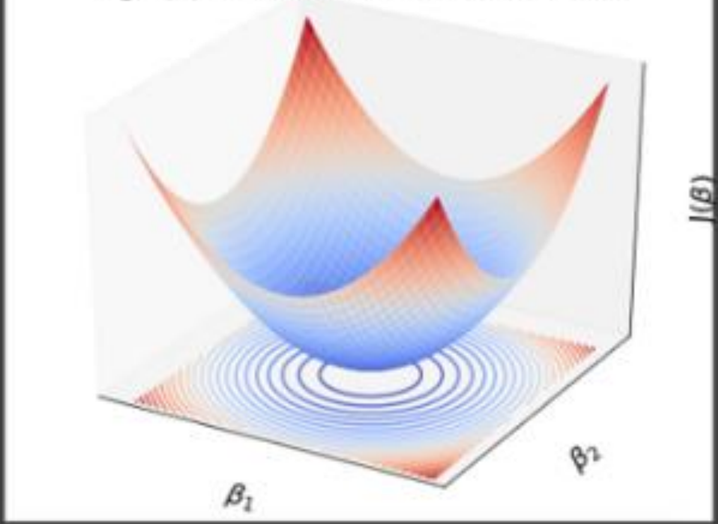


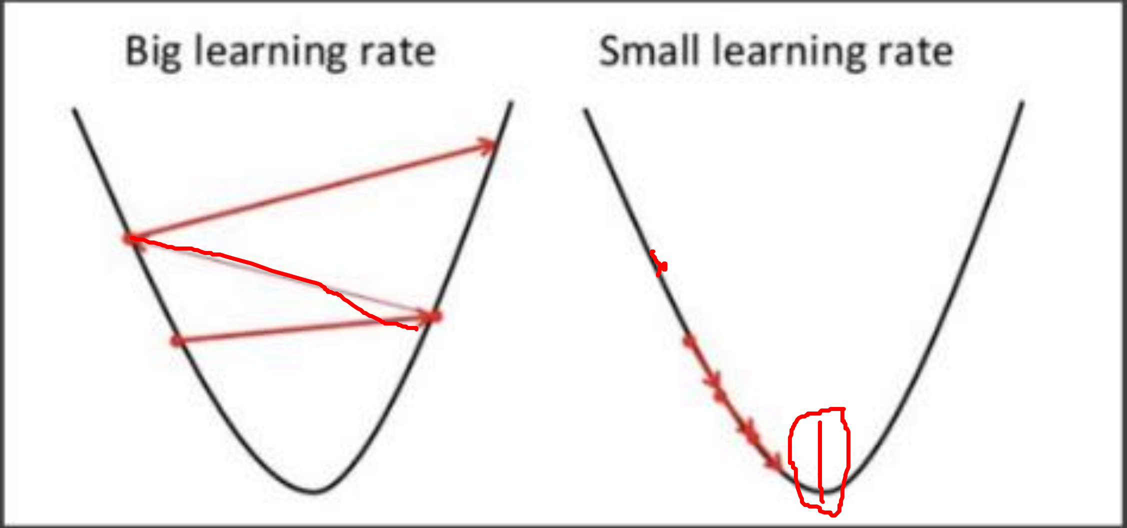
Fig 1(b): Gradient Descent as Contour

Hassan Badawy

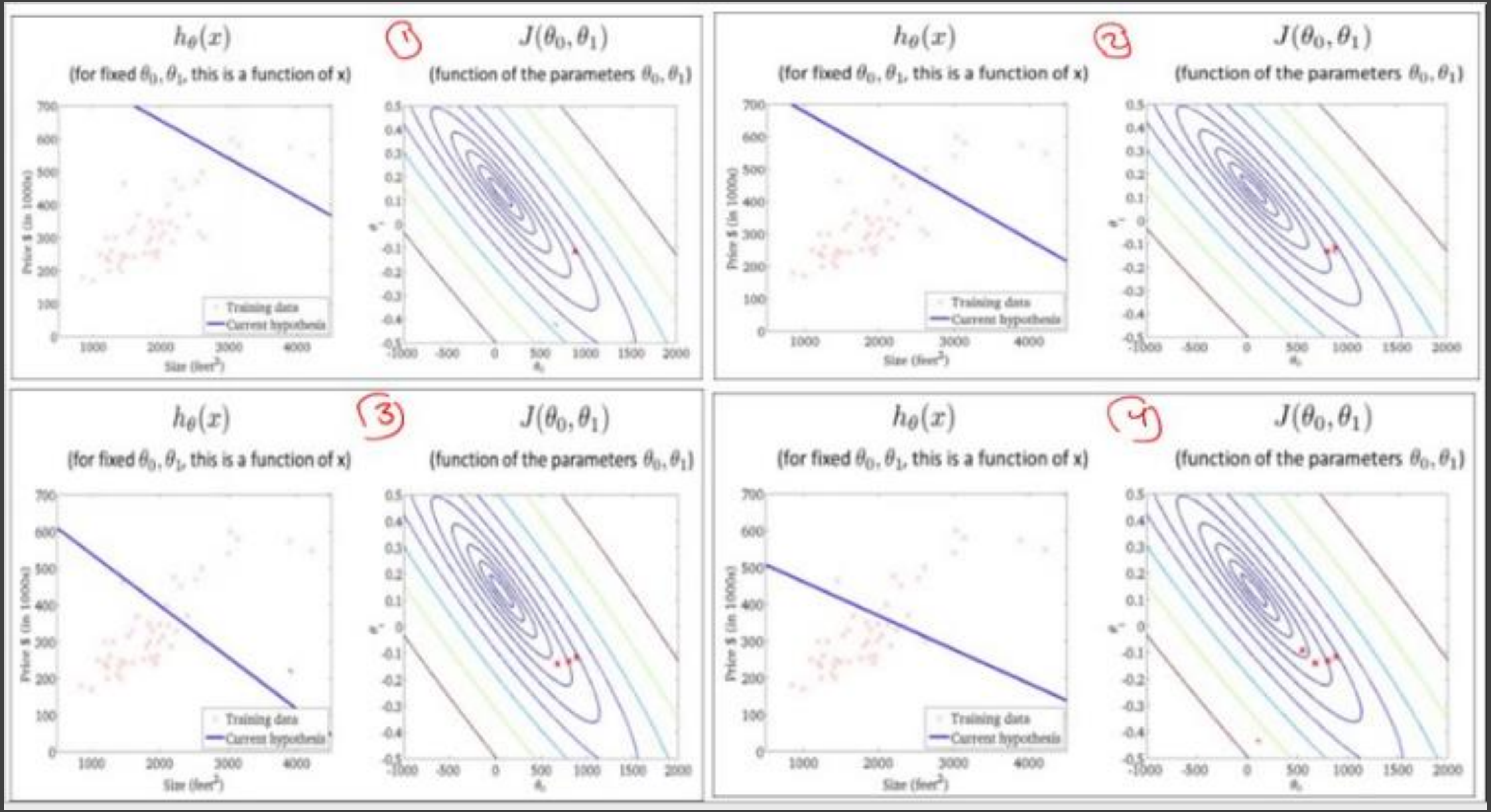
Title

What about Alpha?

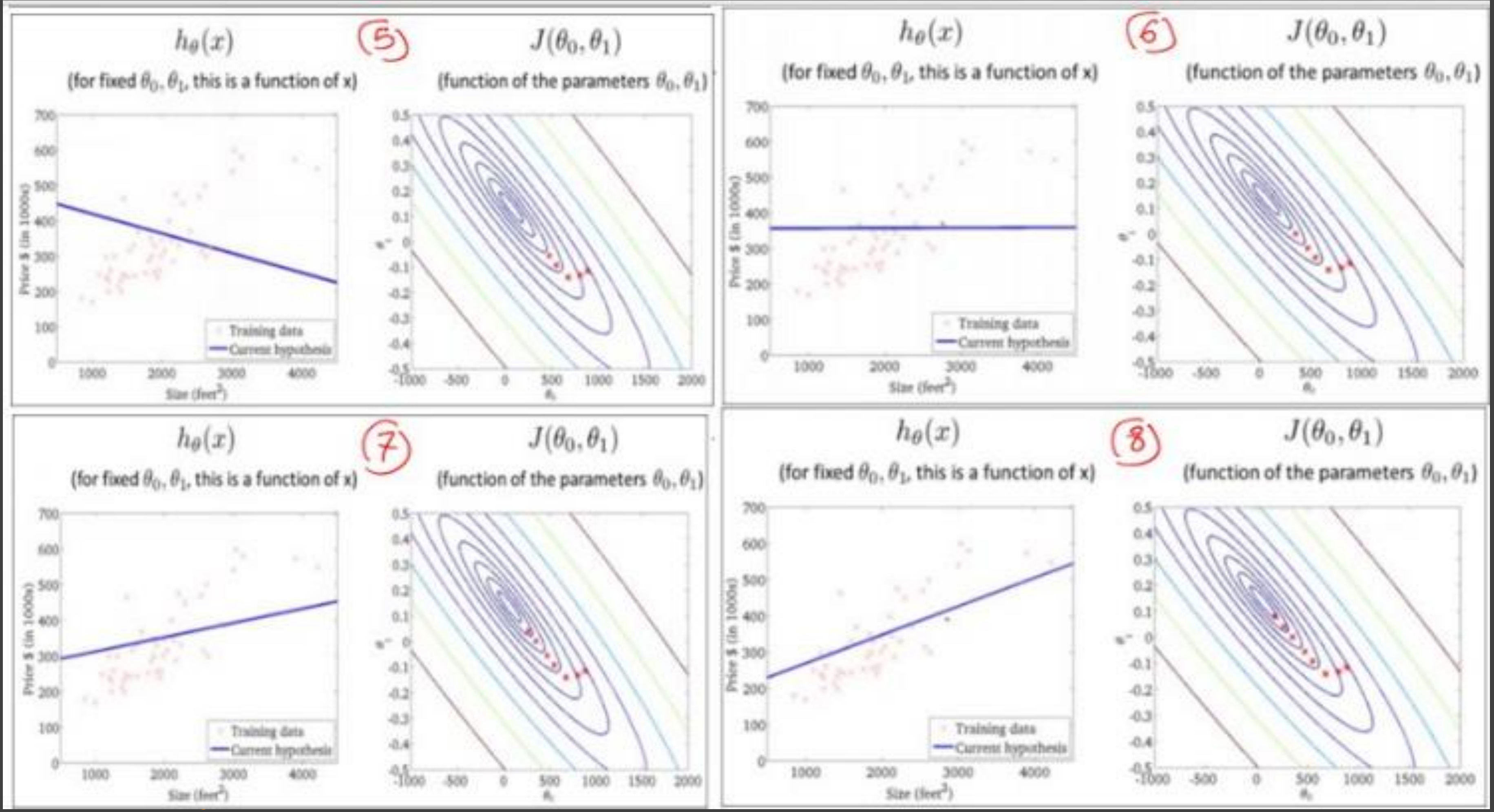
$$w_1 = w_1 - \alpha \frac{1}{n}\sum_{i=1}^{n}\left(\left(w_0 + w_1 x\right)_i - y_i\right)x_i$$

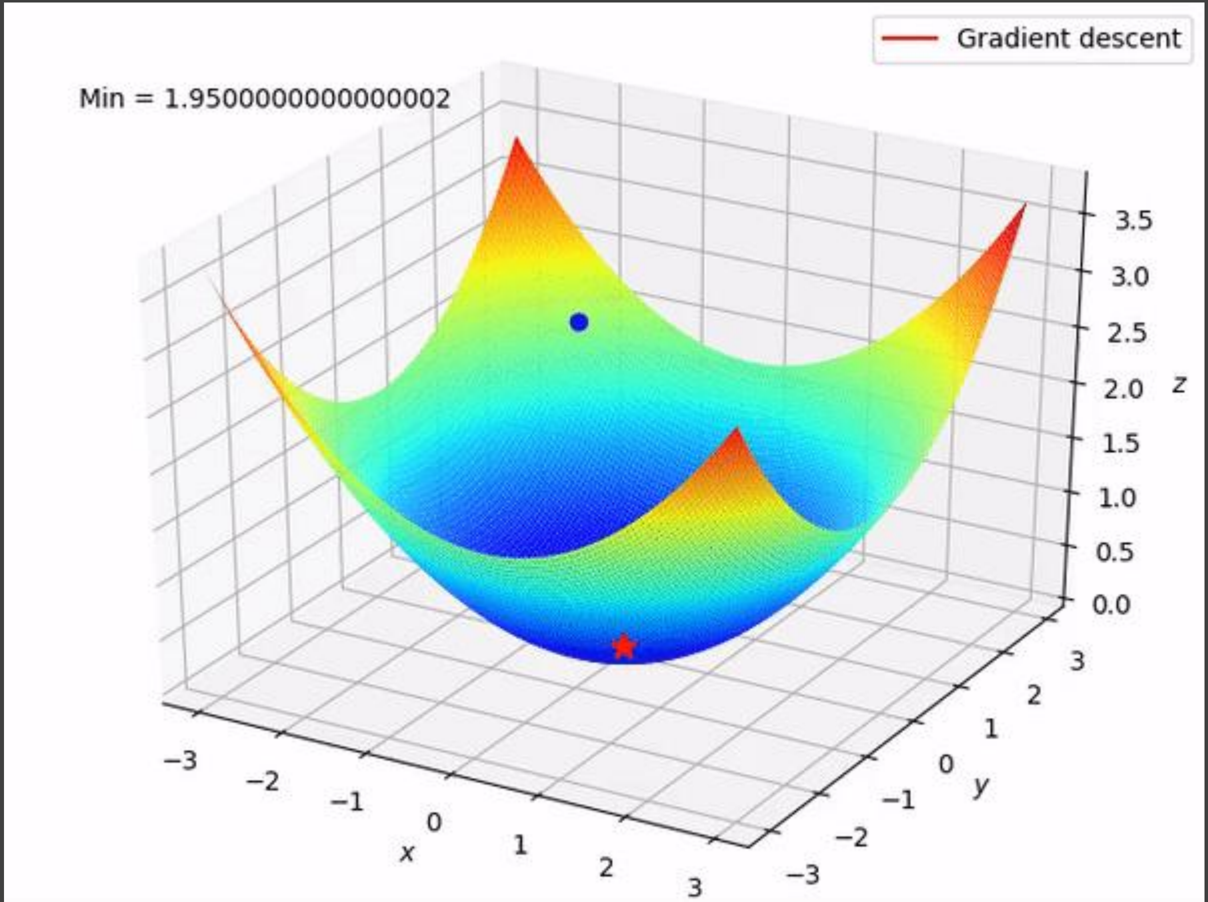


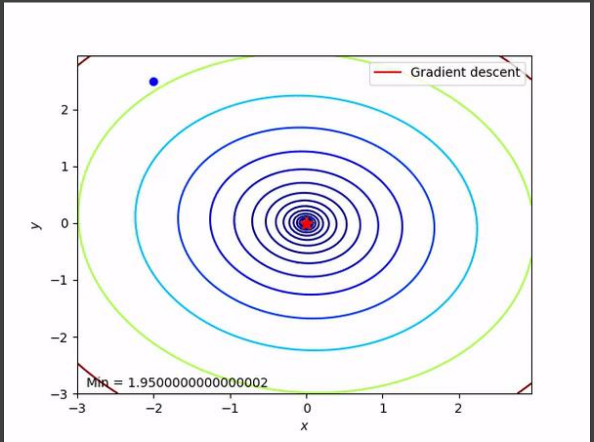
Hassan Badawy

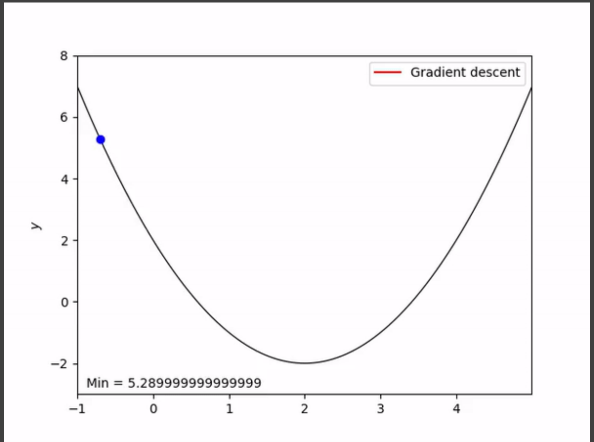


1
$h_\theta(x)$
(for fixed θ_0, θ_1, this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Training data
Current hypothesis
Size (feet^2)
Price $ (in 1000s)
2
$h_\theta(x)$
(for fixed θ_0, θ_1, this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Training data
Current hypothesis
Size (feet^2)
Price $ (in 1000s)
3
$h_\theta(x)$
(for fixed θ_0, θ_1, this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Training data
Current hypothesis
Size (feet^2)
Price $ (in 1000s)
4
$h_\theta(x)$
(for fixed θ_0, θ_1, this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Training data
Current hypothesis
Size (feet^2)
Price $ (in 1000s)


5
$h_\theta(x)$
(for fixed θ_0, θ_1, this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis
6
$h_\theta(x)$
(for fixed θ_0, θ_1, this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis
7
$h_\theta(x)$
(for fixed θ_0, θ_1, this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis
8
$h_\theta(x)$
(for fixed θ_0, θ_1, this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis

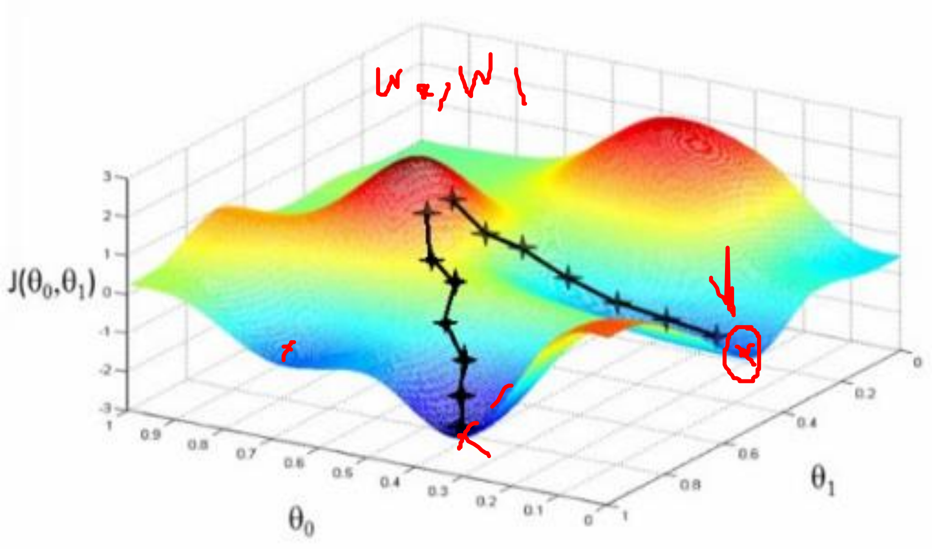


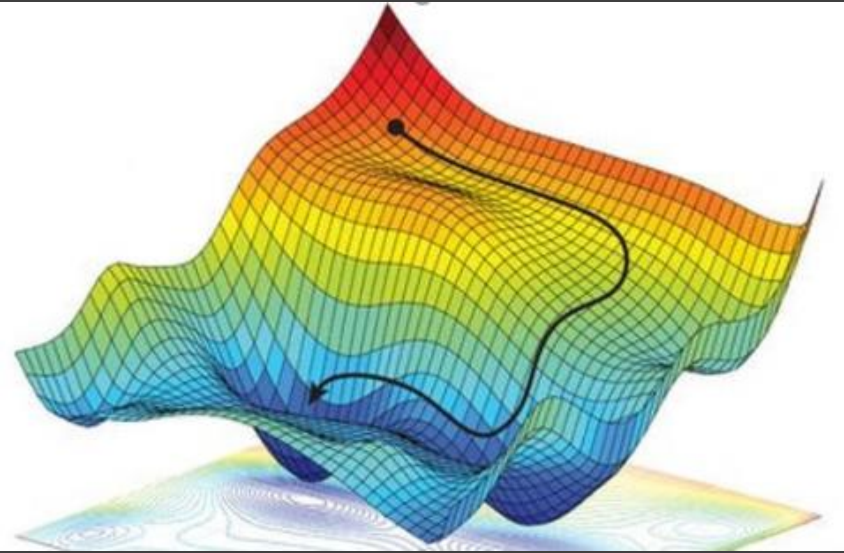




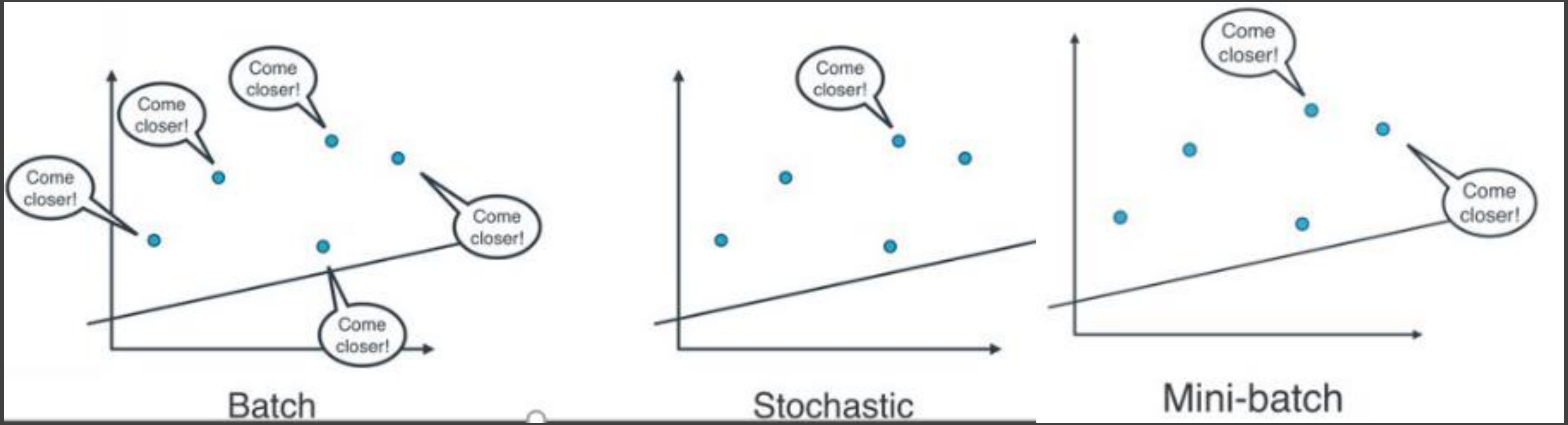
Hassan Badawy

The Problem of non-convex function



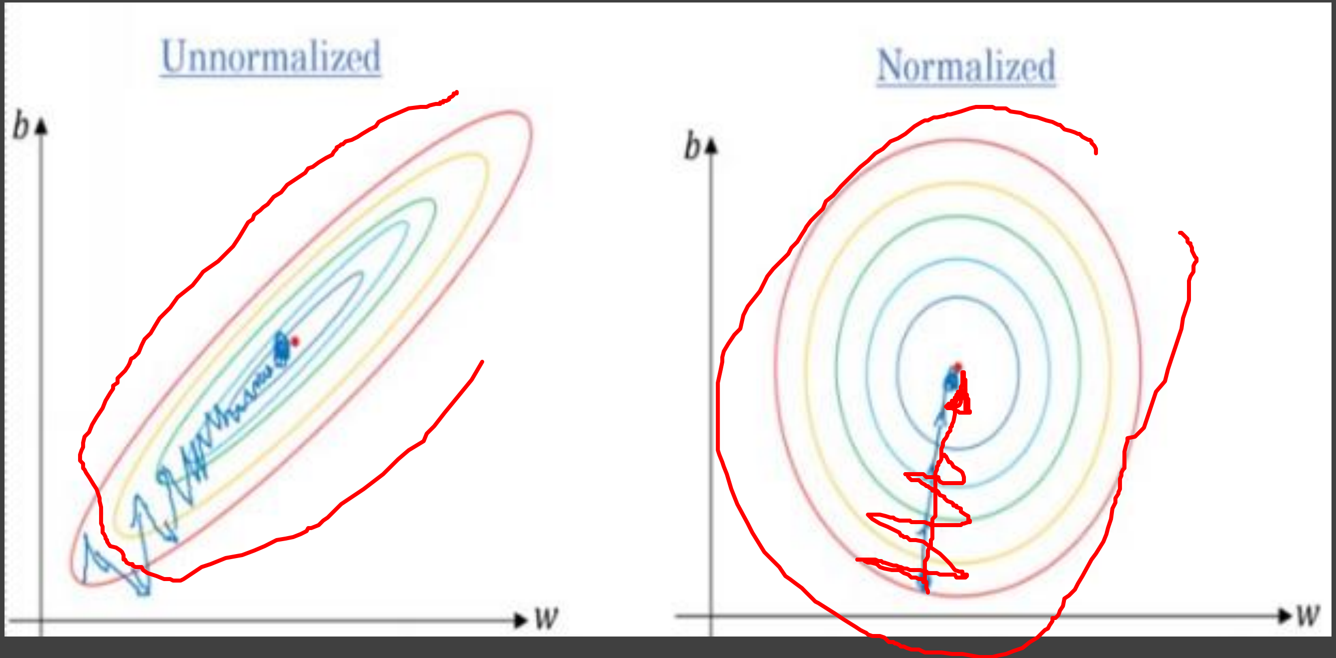


Batch vs Mini-Batch vs Stochastic Gradient Descent SGD



Hassan Badawy

Data Normalization and contour plan:



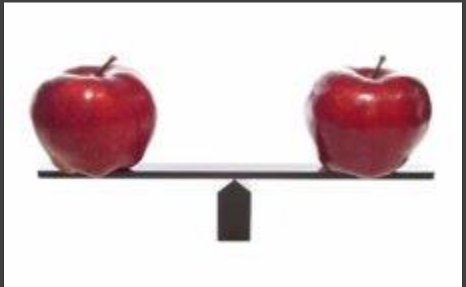
Hassan Badawy

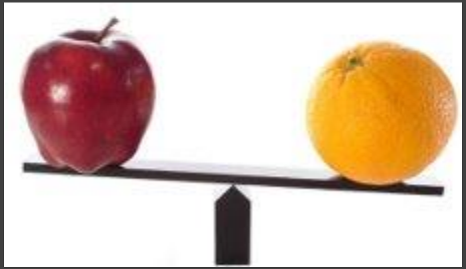
Features Scaling

Why Scaling?

Simple Feature Scaling (range : 0 to 1):-

$$X_{new}=X_{old} / X_{max}$$





Hassan Badawy

Normalization

$$x' = \frac{x - \text{mean}(x)}{\max(x) - \min(x)}$$

This distribution will have values between **-1 and 1** with **μ=0**.

Standardisation and **Mean Normalization** can be used for algorithms that assumes zero centric data like **Principal Component Analysis(PCA)**.

العمود

R Rang

G

355

0

$$\frac{x_i - 122}{255}$$

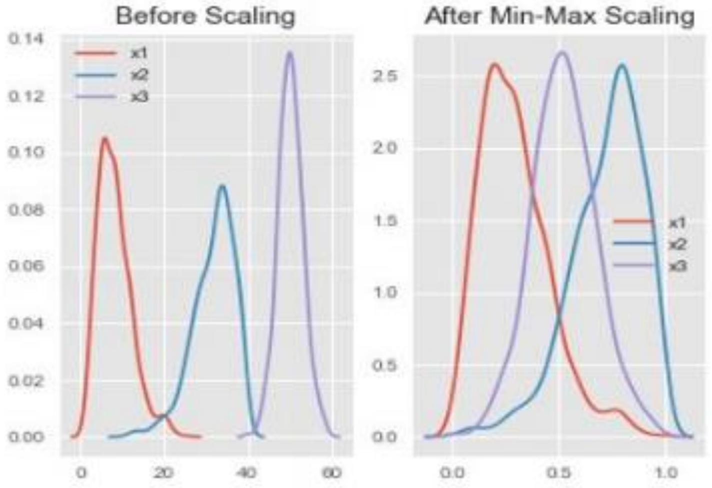
$$\frac{x}{255}$$

Hassan Badawy

Min-Max Scaler

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

This scaling brings the value between 0 and 1.



very sensitive to outliers.

Hassan Badawy

Unit Vector Scaler

$$x' = \frac{x}{||x||}$$

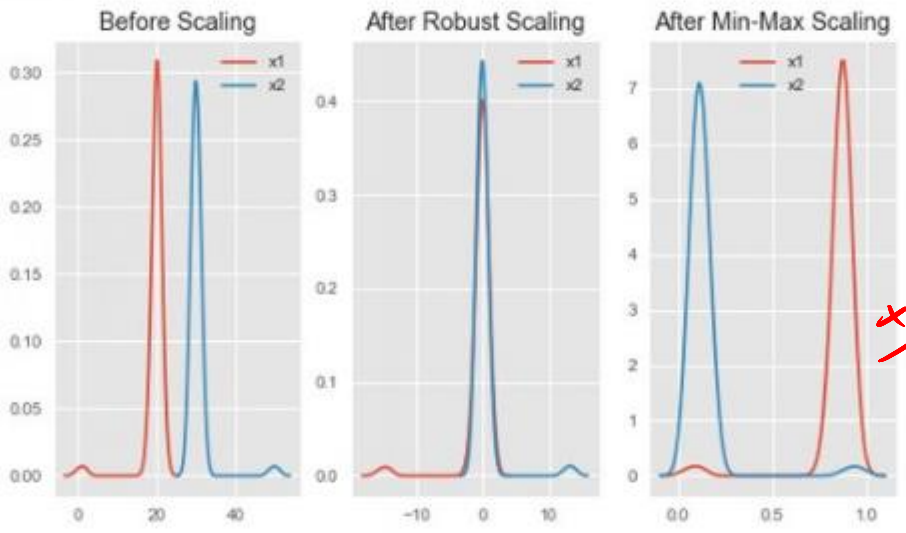
Min-Max Scaling and **Unit Vector** techniques produces values of range [0,1]. When dealing with features with hard boundaries this is quite useful. For example, when dealing with image data, the colors can range from only 0 to 255.

Hassan Badawy

Robust Scaler

The Robust Scaler uses statistics that are robust to outliers:

$(xi–Q1(x))/(Q3(x)–Q1(x))$



$$\frac{x_i - Q_1}{Q_3 - Q_1} \quad IQR$$

5t.

± 1.5 IQR

1.5

$\sim \boxed{ND}$

Hassan Badawy

Standardization

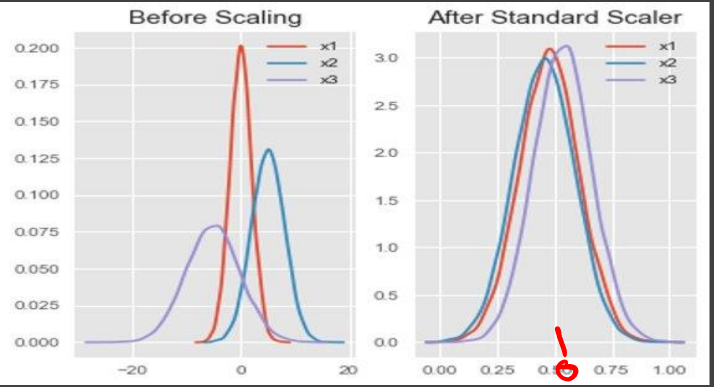
Standardization replaces the values by their Z scores.

$$x' = \frac{x - \bar{x}}{\sigma}$$

This redistributes the features with their mean **μ = 0** and standard deviation **σ = 1** . `sklearn.preprocessing.scale` helps us implementing standardisation in python.

Range most freq: [-3:3]
Full Range : [-4:4]

```
# Features Standarization
X = (X - np.mean(X))/np.std(X)
```



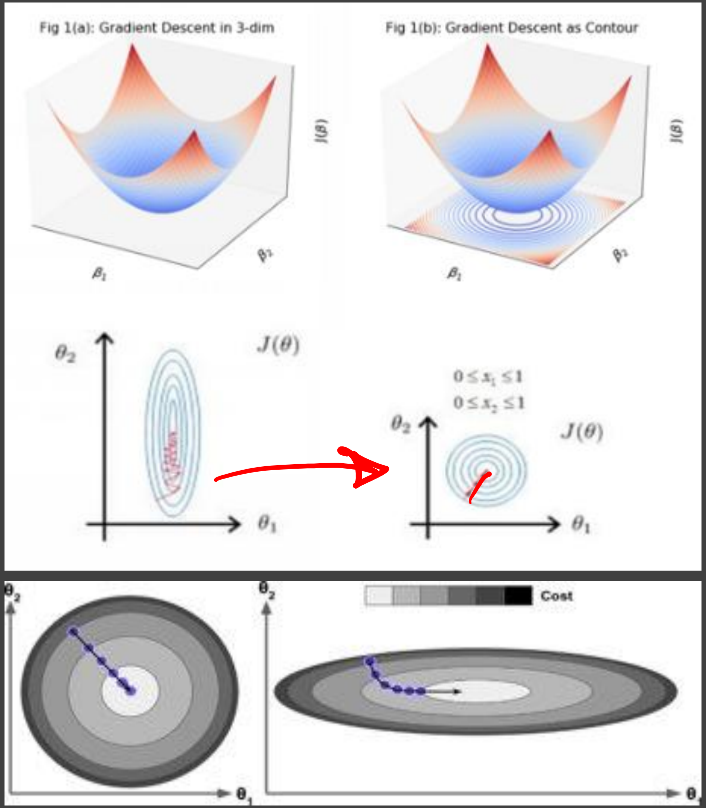
When To Scale?

Rule of thumb I follow here is any algorithm that computes distance or assumes normality, **scale your features!!!**

- **k-nearest neighbors** with an Euclidean distance measure is sensitive to magnitudes and hence should be scaled for all features to weigh in equally.
- Scaling is critical, while performing **Principal Component Analysis(PCA)**. PCA tries to get the features with maximum variance and the variance is high for high magnitude features. This skews the PCA towards high magnitude features.

Hassan Badawy

When To Scale?



Hassan Badawy

When To Scale?

- We can speed up **gradient descent** by scaling. This is because θ will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to the optimum when the variables are very uneven.
- **Tree based models** are not distance based models and can handle varying ranges of features. Hence, Scaling is not required while modelling trees.
- Algorithms like **Linear Discriminant Analysis(LDA), Naive Bayes** are by design equipped to handle this and gives weights to the features accordingly. Performing a features scaling in these algorithms may not have much effect.

feature scaling is necessary while using ridge and lasso regression; basically using regularization parameters(L2 and L1 norm). Standard least squares doesn't often require them but while using regularization with Standard least squares, scaling is necessary.

Hassan Badawy

Scaling by Python

```python
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)
```

```python
import pandas as pd
data=pd.DataFrame({
    'Years of Experience':[1,12,17,27,19,7,21,11,3],
    'Salary':[150000, 1200000, 1750000, 3000000, 2100000,
              900000, 2550000, 1150000, 350000]
})
data
```

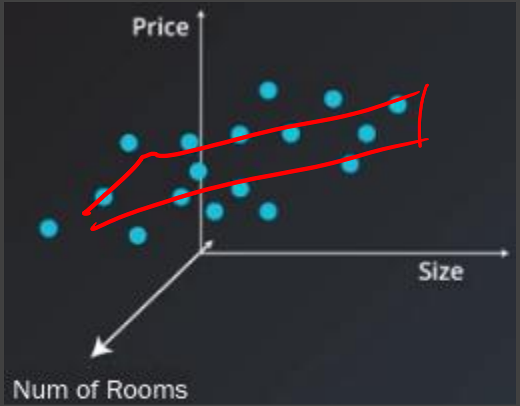
	Years of Experience	Salary
0	1	150000
1	12	1200000
2	17	1750000
3	27	3000000
4	19	2100000
5	7	900000
6	21	2550000
7	11	1150000
8	3	350000

```python
#Z-Score
from scipy.stats import zscore
zscore(data)
```

```
array([[-1.48716274, -1.43827593],
       [-0.13643695, -0.28643631],
       [ 0.47752932,  0.31690826],
       [ 1.70546186,  1.6881459 ],
       [ 0.72311583,  0.7008548 ],
       [-0.75040322, -0.61553334],
       [ 0.96870234,  1.19450035],
       [-0.2592302 , -0.34128581],
       [-1.24157623, -1.21887791]])
```

Multi-linear Regression

$$y = w_0\, x_0 + w_1\, x_1 + w_2\, x_2$$

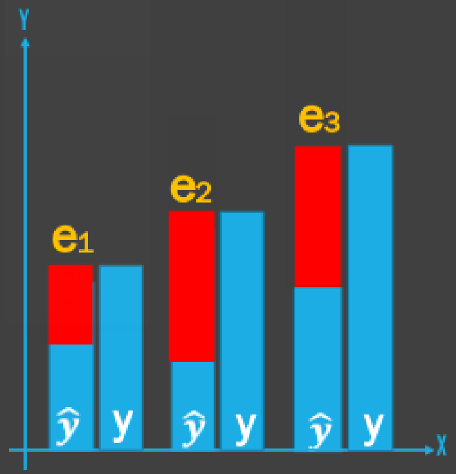


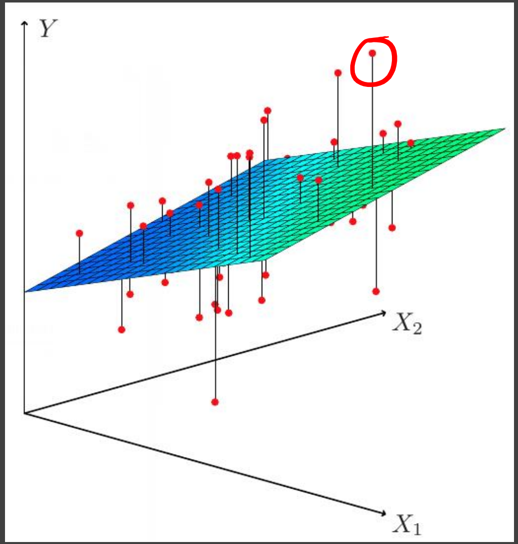
$$W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \end{bmatrix}$$

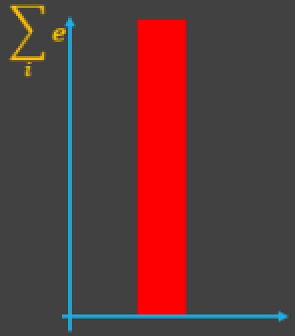
x_1	x_2	y
Area	Num of Rooms	Price (k)
100	2	200
150	3	400
200	4	450
250	5	550
300	7	450

Hassan Badawy

$$y = w_0\, x_0 + w_1\, x_1 + w_2\, x_2$$







$$e = \sum_i e = \sum_i \hat{y} - y$$

Hassan Badawy

Hassan Badawy

Linear Model $\hat{y} = w_0 x_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$

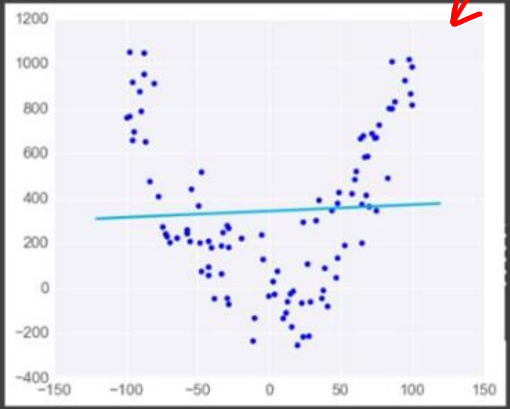
Cost Function $$J(W) = \frac{1}{n}\sum_{i}(\hat{y} - y)^2$$

Gradient Descent $$\boxed{w = w - \alpha \frac{\delta j(w)}{\delta w}}$$

$$X = \begin{pmatrix} x_0^{(0)} & x_1^{(0)} & \dots & x_n^{(0)} \\ x_0^{(1)} & x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \times W = \begin{pmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{pmatrix} = y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

Linear Regression Cons

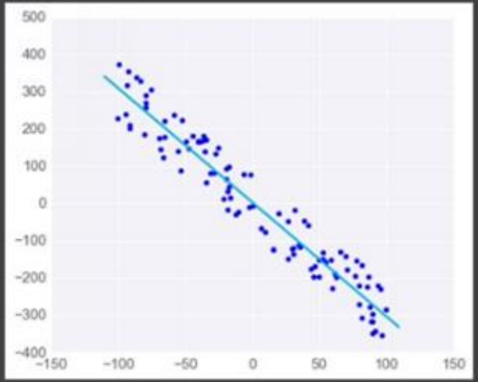
It works well when data is linear

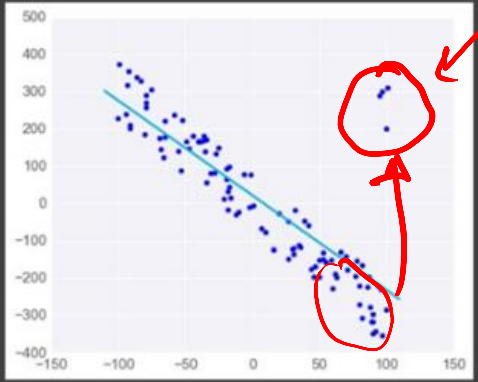


Feature Mapping

Polynomial Regr..

CEO





It is sensitive to Outliers

Hassan Badawy

Dummy Variable Trap

Independent features			Dependent feature
Height(cm)	Age	City	Weight(lb)
138	10	Chicago	70
142	15	Boston	72
140	14	Phoenix	80

Boys health data

With OHE City_Chicago+City_Boston+City_Phoenix =1

Dummy variables count = Category count — 1

Independent features					Dependent feature
Height(cm)	Age	City_Chicago	City_Boston	City_Phoenix	Weight(lb)
138	10	1	0	0	70
142	15	0	1	0	72
140	14	0	0	1	80

Hassan Badawy

https://medium.com/@amanrai77/dummy-variable-trap-9068c3f366fe

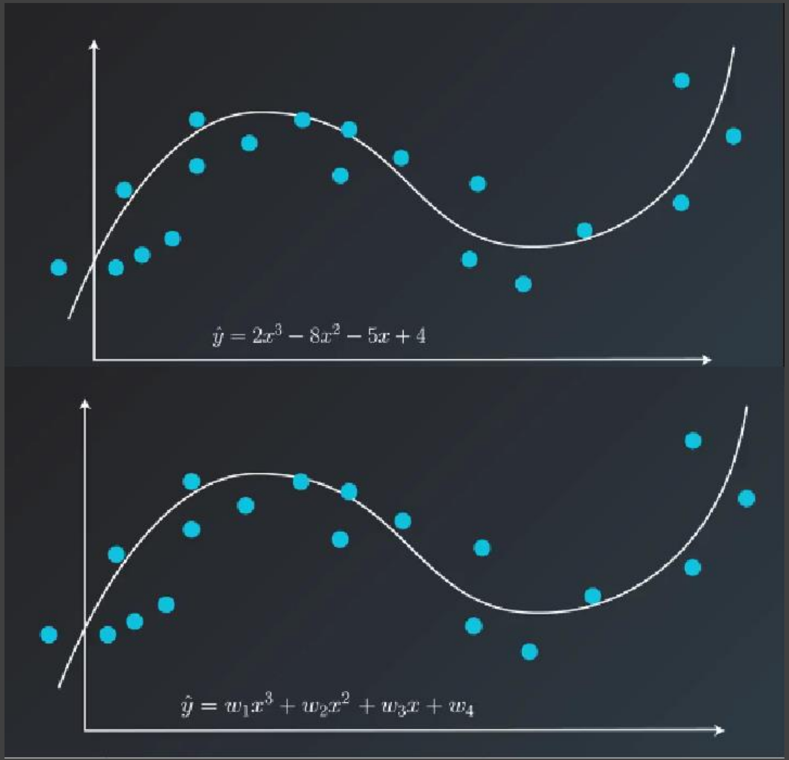
Dummy Variable Trap in Python

```
# Import pandas library
import pandas as pd

#Call get_dummies function
ModifiedData_final = pd.get_dummies(BoysHealthData,drop_first =True)

#View modified data
ModifiedData_final
```

	Height(cm)	Age	Weight(lb)	City_Chicago	City_Phoenix
0	138	10	70	1	0
1	142	15	72	0	0
2	140	14	80	0	1

Hassan Badawy

Polynomial (Multi-Linear) Regression



$$w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_1^3 + \ldots + w_n^n x_n^n$$

Hassan Badawy

```python
# Importing the dataset
dataset = pd.read_csv('data/Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values.reshape(-1, 1)
dataset
```

executed in 109ms, finished 09:44:03 2019-04-13

```python
X.shape, y.shape
```

executed in 16ms, finished 09:44:07 2019-04-13

```
((10, 1), (10, 1))
```

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

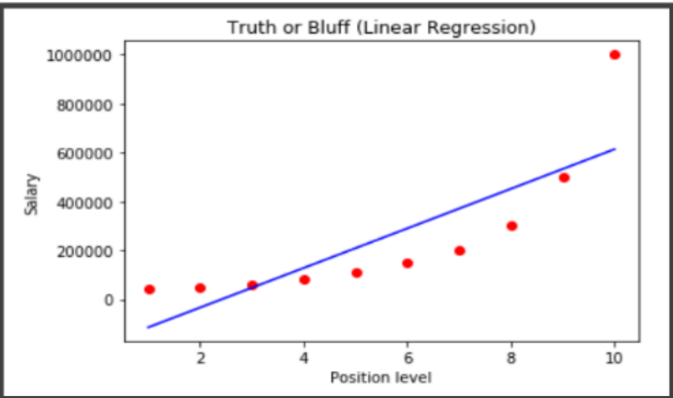
Hassan Badawy

```python
# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
y_predict_lin = lin_reg.predict(X)
```

executed in 31ms, finished 09:38:42 2019-04-13

```python
# Visualising the Linear Regression results
plt.scatter(X, y, color = 'red')
plt.plot(X, y_predict_lin, color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

executed in 141ms, finished 09:38:45 2019-04-13



Hassan Badawy

SVM

```
poly=sklearn.PolynomialFeature(degree=2)

poly_X=poly.fit_transform(X)
```

1
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0

	0	1	2	3	4
0	1.0	1.0	1.0	1.0	1.0
1	1.0	2.0	4.0	8.0	16.0
2	1.0	3.0	9.0	27.0	81.0
3	1.0	4.0	16.0	64.0	256.0
4	1.0	5.0	25.0	125.0	625.0
5	1.0	6.0	36.0	216.0	1296.0
6	1.0	7.0	49.0	343.0	2401.0
7	1.0	8.0	64.0	512.0	4096.0
8	1.0	9.0	81.0	729.0	6561.0
9	1.0	10.0	100.0	1000.0	10000.0

Poly

Hassan Badawy

Hassan Badawy

```python
# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
poly_reg.fit(X_poly, y)
```

executed in 16ms, finished 09:38:55 2019-04-13

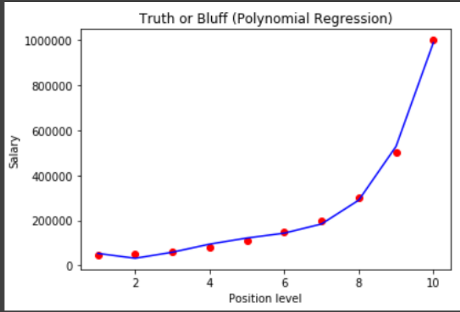
```
PolynomialFeatures(degree=4, include_bias=True, interaction_only=False)
```

```python
# Predicting a new result with Polynomial Regression
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
y_pred_poly = lin_reg_2.predict(poly_reg.fit_transform(X))
```

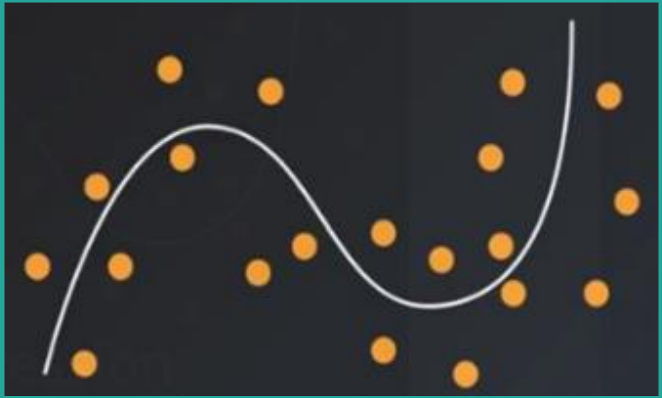
```python
# Predicting a new result with Linear Regression
pred1 = lin_reg.predict(np.array([[6.5]]))
pred2 = lin_reg_2.predict(poly_reg.fit_transform(np.array([[6.5]])))
pred1, pred2
```

executed in 31ms, finished 09:51:40 2019-04-13

```
(array([[330378.78787879]]), array([158862.45265153]))
```



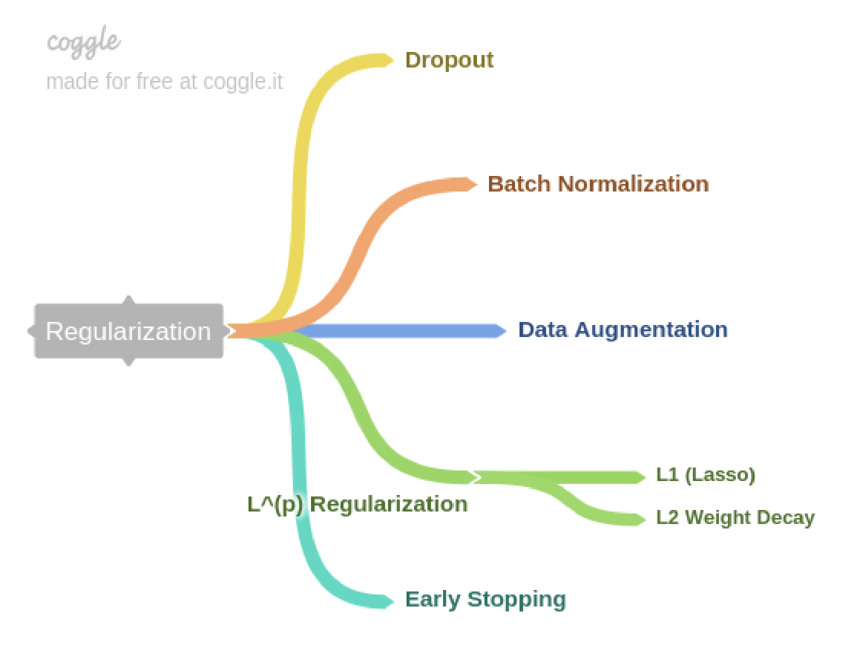
Bias-Variance [Underfit-Overfit] Models



- Overfitting
- Well-fitting
- Underfitting
- Lasso
- Ridge

Regularization

Regularizations are techniques used to generalize your model and reduce the error by fitting a function appropriately on the given training set and avoid overfitting.

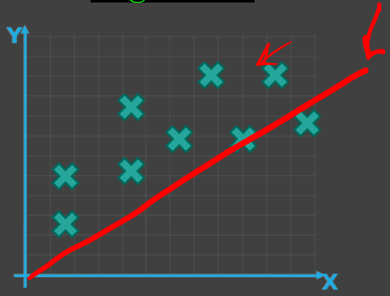


Hassan Badawy

Impact of Bias - Variance

```
poly=sklearn.PolynomialFeature(degree=2)

poly_X=poly.fit_transform(X)
```

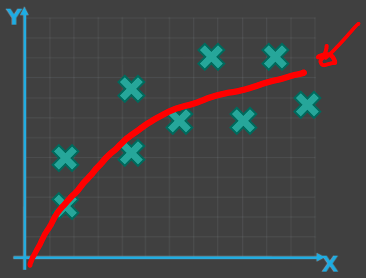
degree=1



$$w_0 + w_1 x$$

High Biase
(Undefitting)

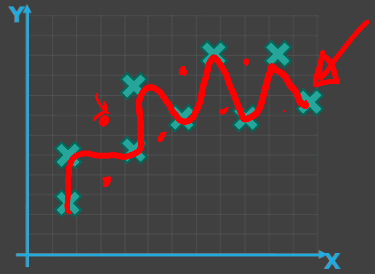
degree=2



$$w_0 + w_1 x + w_2 x^2$$

Well-fitting

degree=4



$$w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$

High Variance
(Overfitting)

Hassan Badawy

Overfitting Problem

If our model is suffer from Overfitting we can said that our model is a summarizing model than a predictive model.

Alternatively If our model is suffer from Underfitting we can say that our is not trained and consider be a random model.

Overfitting Problem Solving

- Use Regularization Techniques (L1, L2, Elastic Net)
- Model Validation (Data Splitting, K-fold)
- Reduce number of features

Hassan Badawy

The regularization parameter λ

With regularization, take cost function and modify it to shrink all the parameters and Add a term at the end, This regularization term shrinks every parameter

L1

$$J(w)_{L1} = min_w \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y(i))^2 + \lambda \sum_{j=1}^{n} |w_j|$$

$$J(w)_{L2} = min_w \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y(i))^2 + \lambda \sum_{j=1}^{n} w_j^2$$

b to b

w_0 to

w[1:]

L2

λ should be chosen carefully

Hassan Badawy

Automating λ

$$w_j = w_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y(i))^2 x_j^{(i)}$$

The term $(1 - \alpha\frac{\lambda}{m})$ Usually is a number less than 1.

Usually learning rate alpha is small and m is large

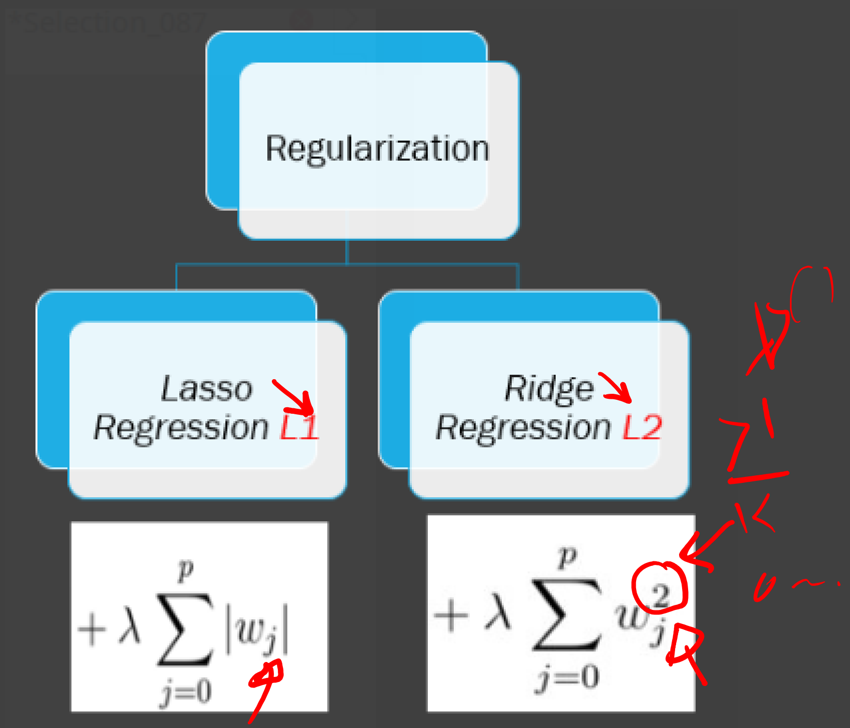
So the result is going to be (1- a small number) ~ 0.99 or 0.95

This means that w_j gets multiplied by 0.99 that means we take the most value of last weight.

If w_j multiplied with 0.01, that mean we almost forget the last weight and start learn from the beginning, thats means that our model don't learn anything and it is go through underfitting

Hassan Badawy

Lasso Regression L1



Lasso Regression L1

$$J(w)_{L1} = min_w \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y(i))^2 + \lambda \sum_{j=1}^{n} |w_j|$$

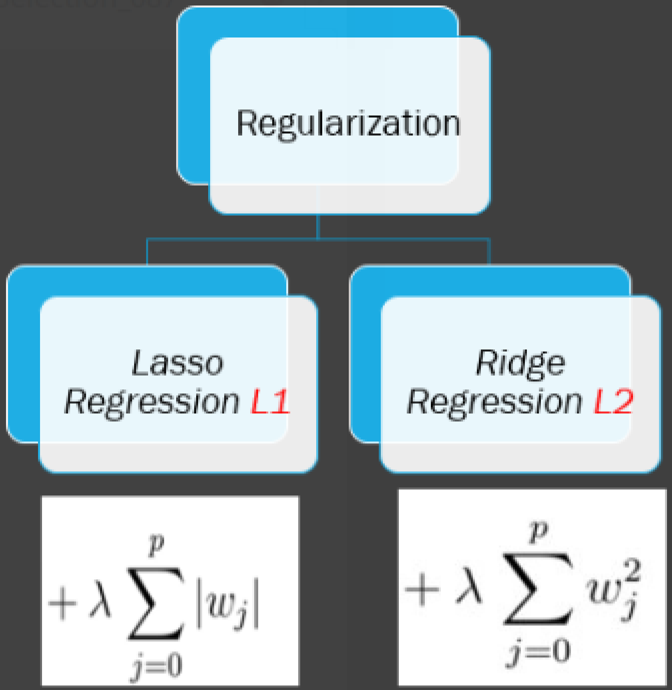
Use sklearn's Lasso class to fit a linear regression model to the data, while also using L1 regularization to control for model complexity.

```
from sklearn.linear_model import Lasso
lasso_reg = Lasso()
lasso_reg.fit(X, y)
reg_coef = lasso_reg.coef_
```

Hint: Lambda is called alpha in sklearn and it equal to 1.0
If alpha = 0 is equivalent to an ordinary least square, solved by the LinearRegression object

Hassan Badawy

Ridge Regression L2



Ridge Regression

$$J(w)_{L2} = min_w \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y(i))^2 + \lambda \sum_{j=1}^{n} w_j^2$$

Use sklearn's Ridge class to fit a linear regression model to the data, while also using L2 regularization to control for model complexity.

```
from sklearn.linear_model import Ridge
Ridge _reg = Ridge()
Ridge _reg.fit(X, y)
reg_coef = Ridge _reg.coef_
```

Hint: Lambda is called alpha in sklearn and it equal to 1.0
If alpha = 0 is equivalent to an ordinary least square, solved by the LinearRegression object

Hassan Badawy

Elastic Net Regression

Ridge + Lasso

$$L = \sum(\hat{Y}i - Yi)^2 + \lambda\sum w^2 + \lambda\sum |w|$$

Elastic Net combines characteristics of both lasso and ridge. Elastic Net reduces the impact of different features while not eliminating all of the features.

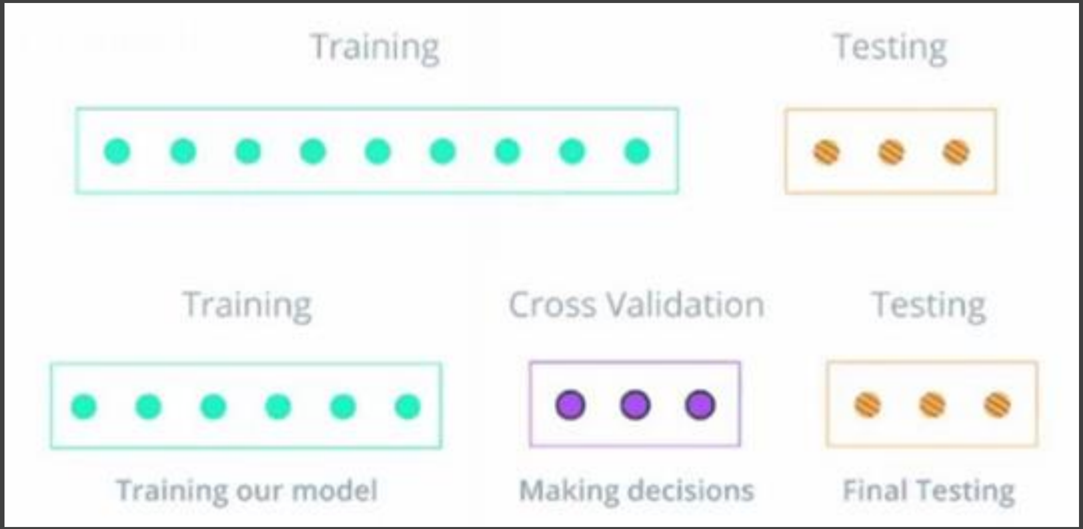
```
sklearn.linear_model.ElasticNet(alpha=1.0, l1_ratio=0.5)
```

```
alpha = a + b and l1_ratio = a / (a + b)
```

Hassan Badawy

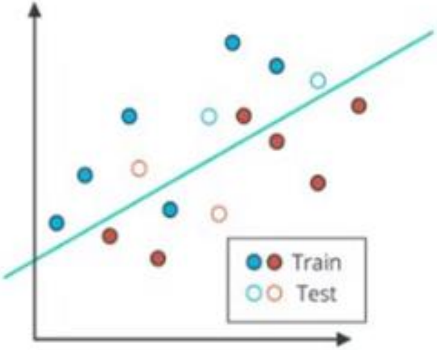
Model Cross Validation

Cross Validation



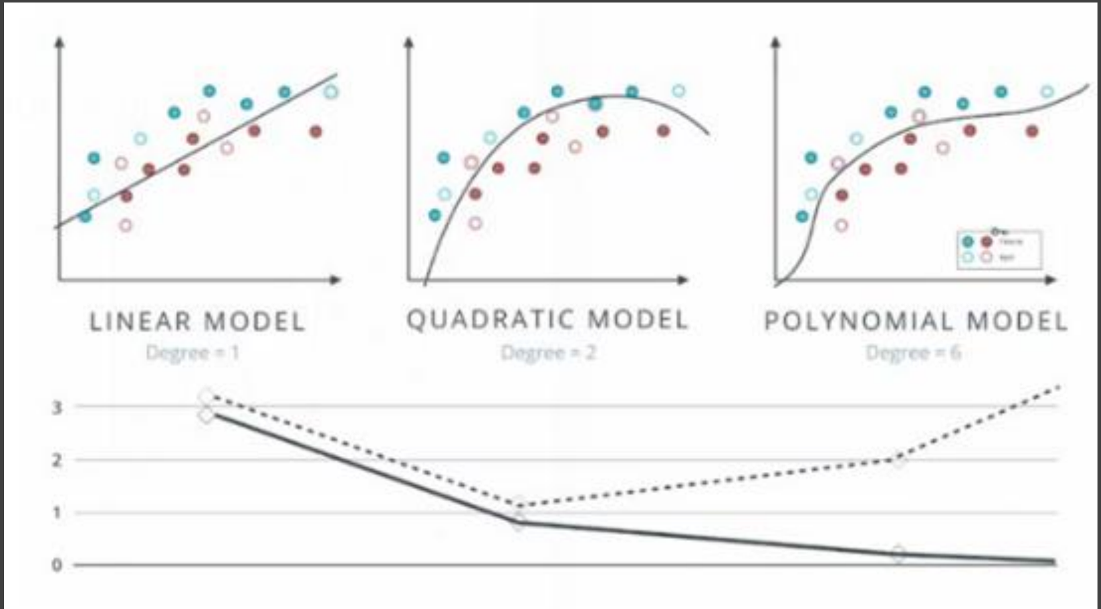
Sklearn Cross Validation

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(X,
                 y,
                 test_size = 0.25)
```

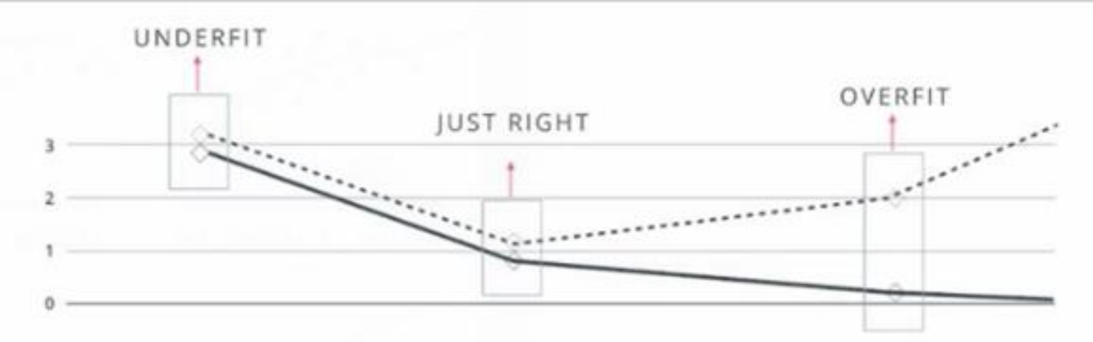


Hassan Badawy

Which Model is Better

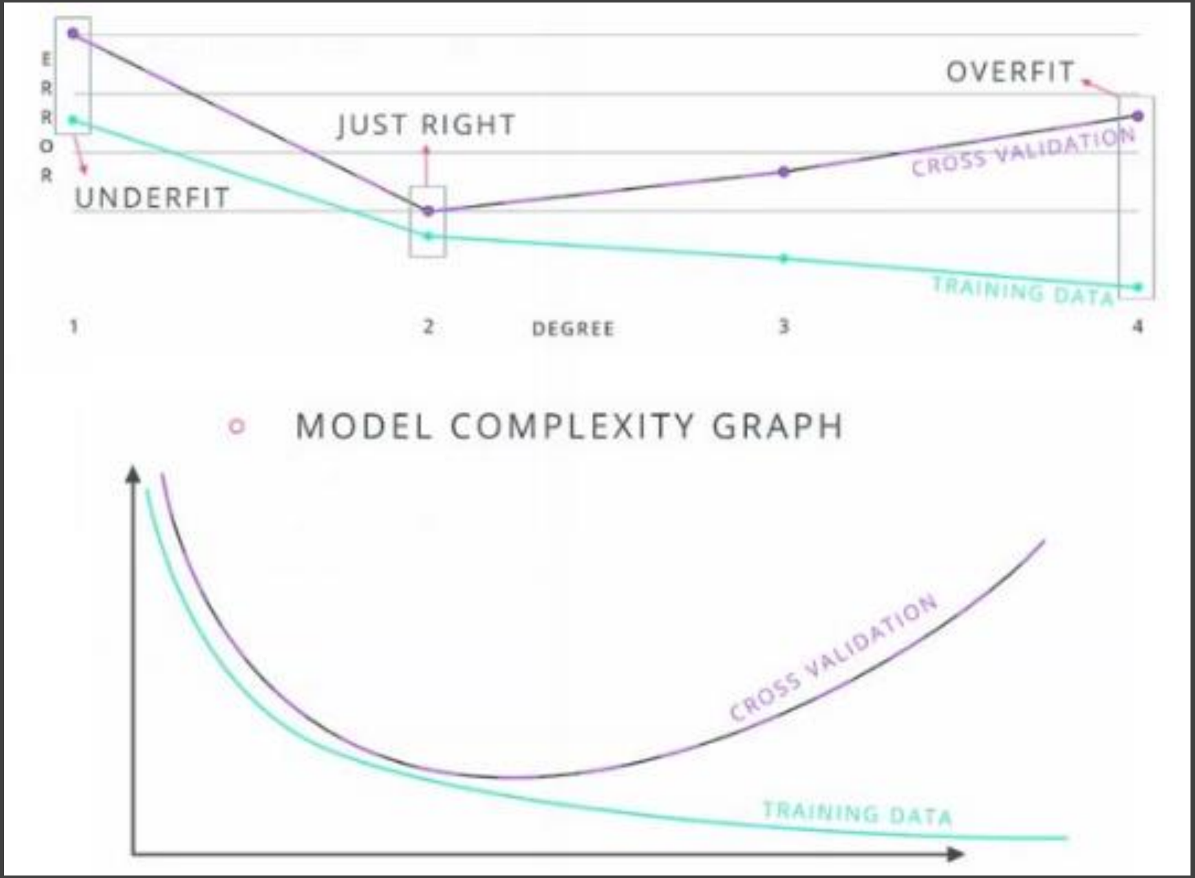


Model Complexity Graph



Hassan Badawy

Model Complexity Graph



Hassan Badawy

Regression Evaluation Metrics

- Mean Absolute Error MAE
- Mean Squared Error MSE
- Root Mean Squared Error RMSE
- R-Squared
- Adjusted R-Squared

Hassan Badawy

Regression

- MSPE
- MSAE
- R Square
- Adjusted R Square

Classification

- Precision-Recall
- ROC-AUC
- Accuracy
- Log-Loss

Unsupervised Models

- Rand Index
- Mutual Information

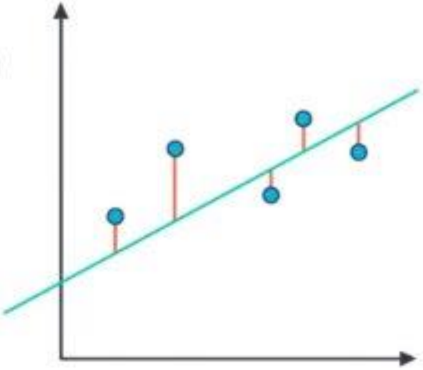
Others

- CV Error
- Heuristic methods to find K
- BLEU Score (NLP)

Mean Absolute Error MAE

$$MAE = \frac{1}{n}\sum_{i}^{n} |\hat{y}_i - y_i|$$

```
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression

classifier = LinearRegression()
classifier.fit(X,y)

guesses = classifier.predict(X)

error = mean_absolute_error(y, guesses)
```



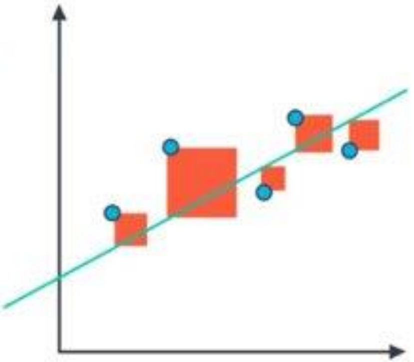
Hassan Badawy

Mean Squared Error MSE

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

$$RMSE = \sqrt{\frac{1}{n}\sum_{i}^{n}(\hat{y}_i - y_i)^2}$$

```
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression

classifier = LinearRegression()
classifier.fit(X,y)

guesses = classifier.predict(X)

error = mean_squared_error(y, guesses)
```



Hassan Badawy

R-Squared

BAD MODEL

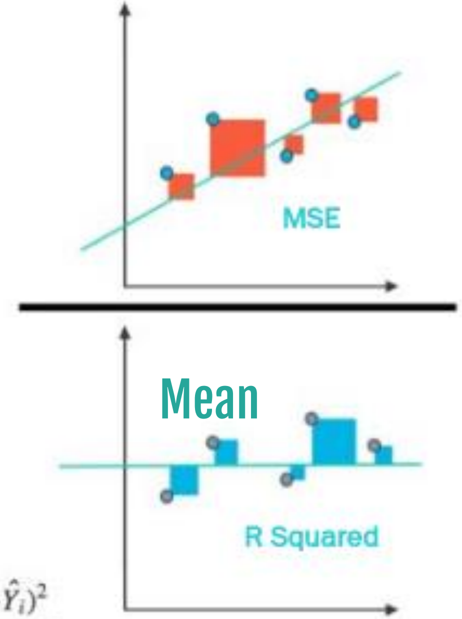
The errors should be similar.
R2 score should be close to 0.

GOOD MODEL

The mean squared error for the linear regression model should be a lot smaller than the mean squared error for the simple model.

R2 score should be close to 1.

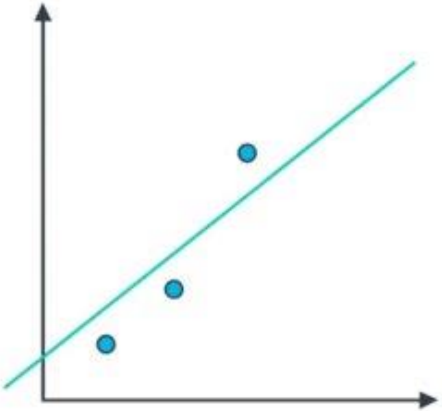
R2 = 1 -



$$\hat{R}^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i-\bar{Y})^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(Y_i-\bar{Y})^2}$$

Hassan Badawy

```python
from sklearn.metrics import r2_score

y_true = [1, 2, 4]
y_pred = [1.3, 2.5, 3.7]

r2_score(y_true, y_pred)
```



```python
def r_squared(m,X,y):
    yhat = m.predict(X)
    print(yhat)
    SS_Residual = sum((y-yhat)**2)
    SS_Total = sum((y-np.mean(y))**2)
    r_squared = 1 - (float(SS_Residual))/SS_Total
return r_squared
```

Hassan Badawy

Adjusted R-Squared

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

R^2 – Goodness of fit
(greater is better)

$y = b_0 + b_1{*}x_1$

$y = b_0 + b_1{*}x_1 + b_2{*}x_2 \leftarrow \boxed{+\, b_3{*}x_3}$

Problem:

SS_{res} -> Min $\longrightarrow$ R^2 will never decrease

Hassan Badawy

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Number of independent variables or features

$$\text{Adj } R^2 = 1 - (1 - R^2)\frac{n - 1}{n - p - 1}$$

p - number of regressors

n - sample size

Adj R2 Penalize your system when you use a feature doesn't related to your output (have noise or random samples)

Hassan Badawy

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

If R2 = 1 then
Adj = R2 = 1

$$\text{Adj } R^2 = 1 - (1 - R^2)\frac{n - 1}{n - p - 1}$$

p - number of regressors
n - sample size

The more features The more System Penalization

Hassan Badawy

Feature Elimination

Dimensions Reduction (Feature Engineering)

5 methods of building models:

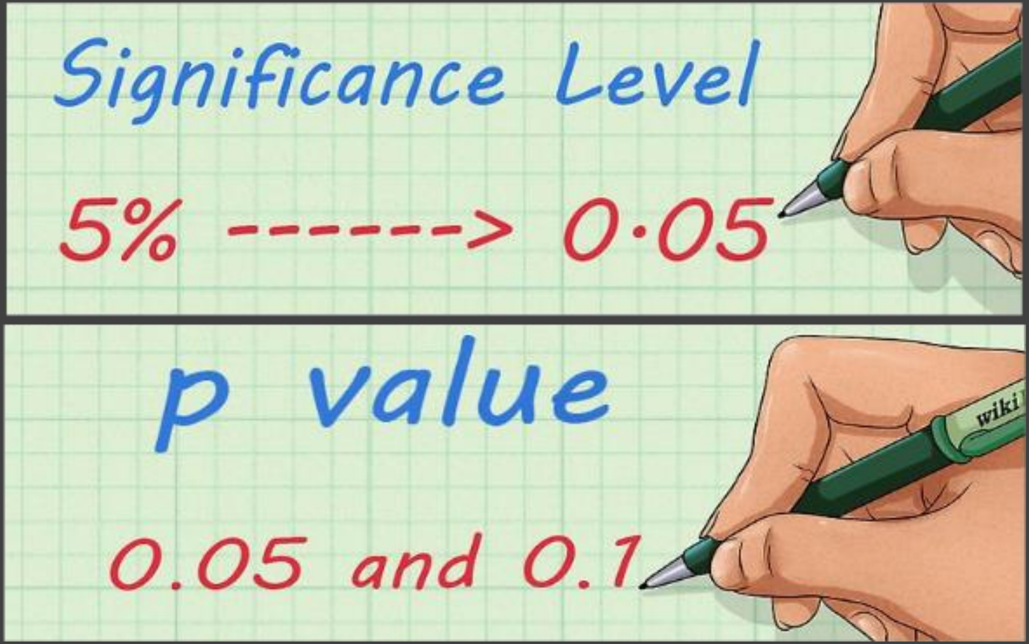
1. All-in
2. Backward Elimination
3. Forward Selection
4. Bidirectional Elimination
5. Score Comparison

Stepwise Regression

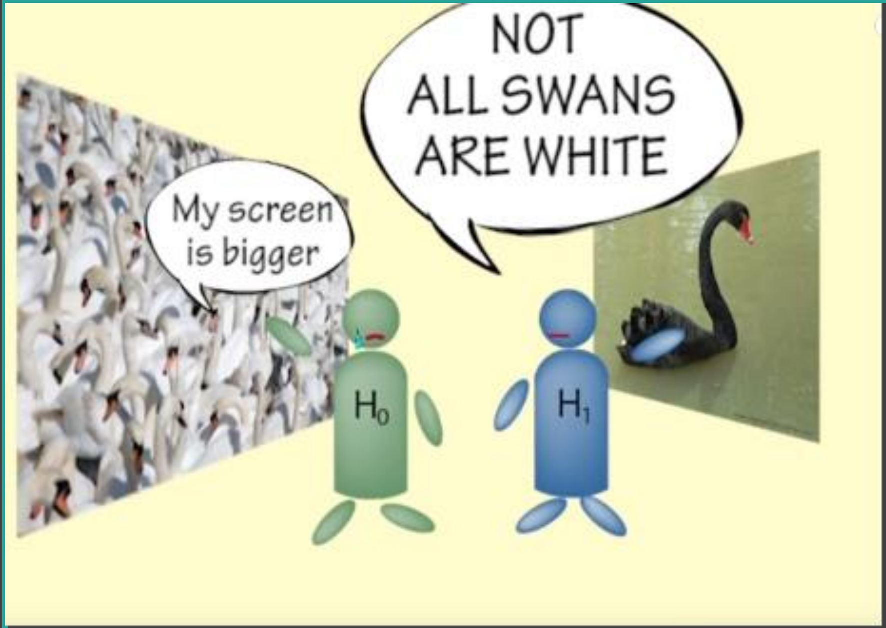
Feature Engineering and P - Value

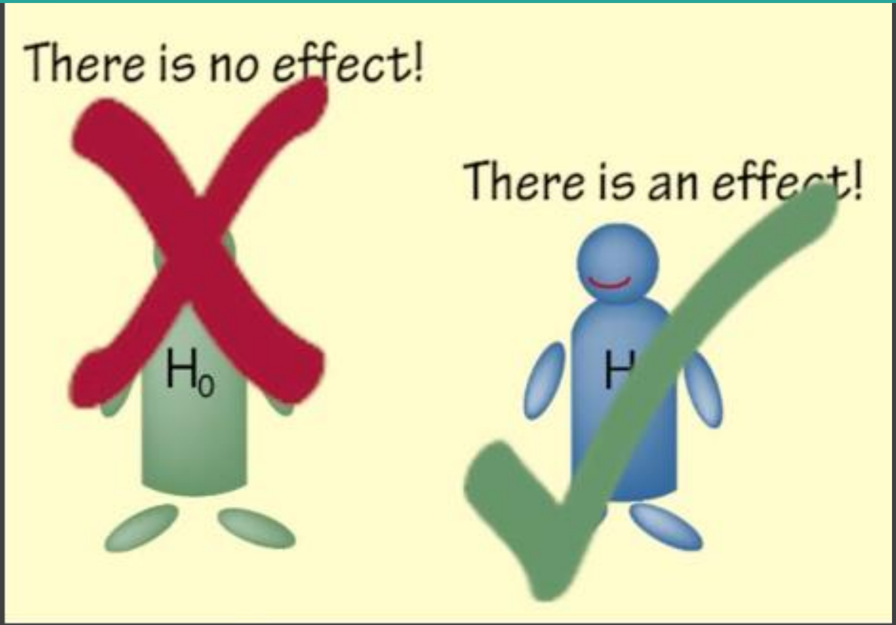
P value is a statistical measure that helps scientists determine whether or not their hypotheses are Statistically Significant or not

if the P value of a data set is below a certain pre-determined amount (like, for instance, 0.05), scientists will reject the "null hypothesis" of their experiment



Hassan Badawy



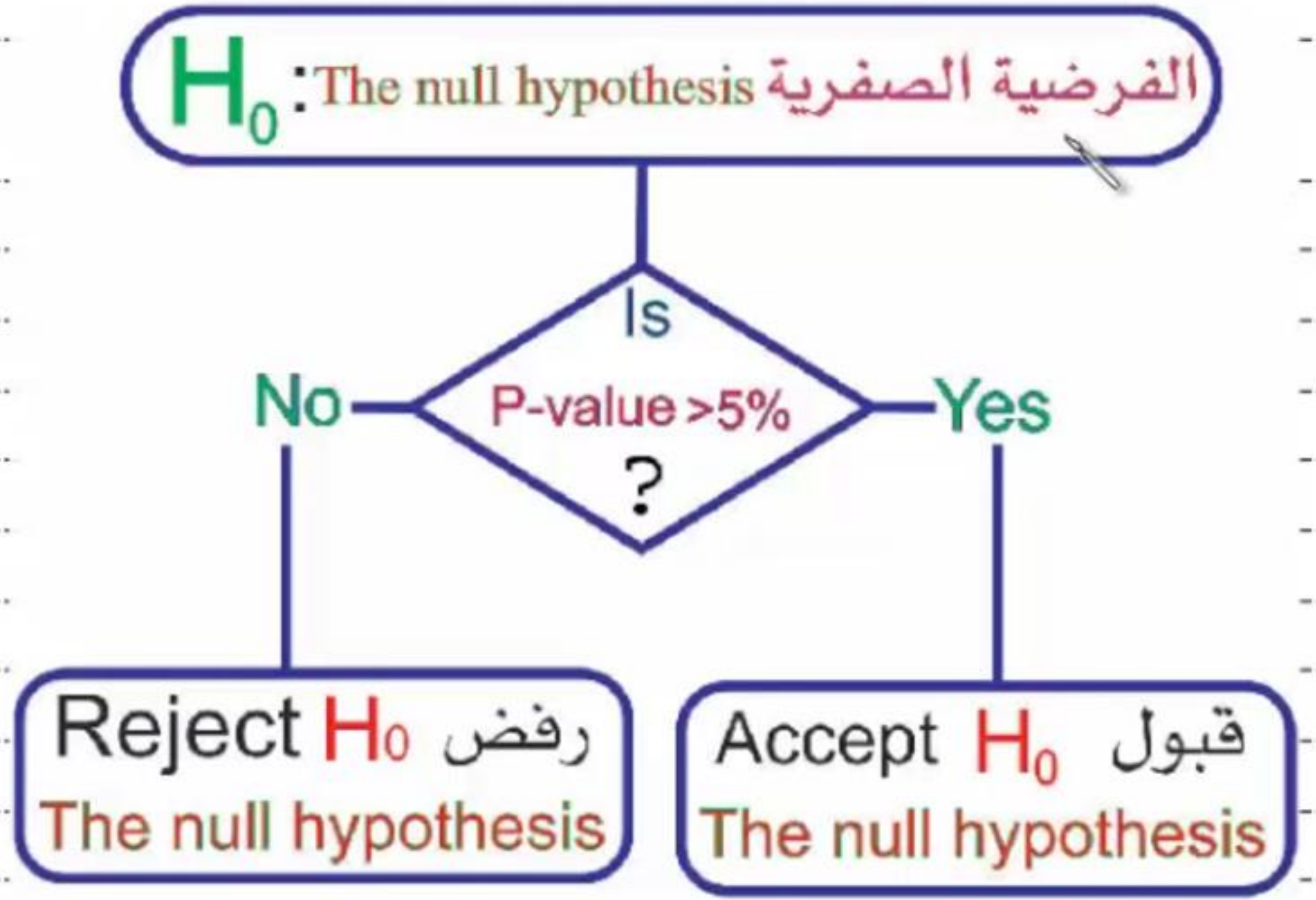


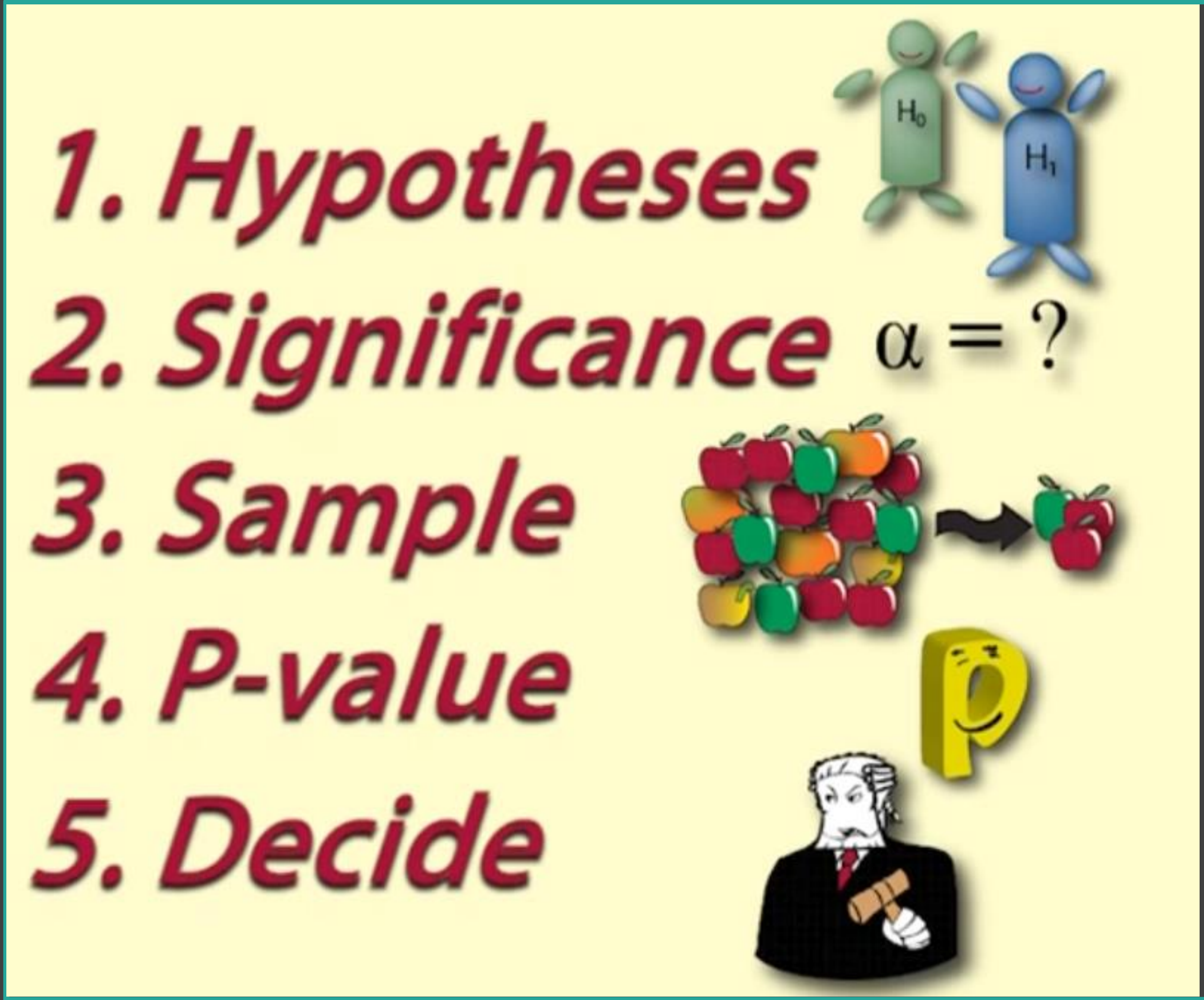
Level of Significance

α

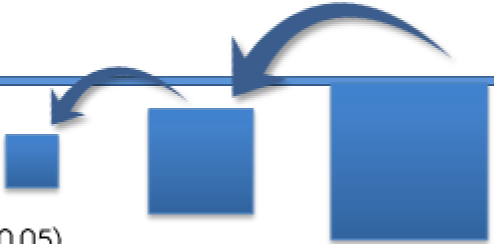
0.05

Hassan Badawy


H0 :The null hypothesis الفرضية الصفرية
Is
P-value >5%
?
No
Yes
Reject H0 رفض
The null hypothesis
Accept H0 قبول
The null hypothesis


1. Hypotheses
H_0
H_1
2. Significance
$\alpha = ?$
3. Sample
4. P-value
p
5. Decide

Building A Model



Backward Elimination

STEP 1: Select a significance level to stay in the model (e.g. SL = 0.05)

STEP 2: Fit the full model with all possible predictors

STEP 3: Consider the predictor with the highest P-value. If P > SL, go to STEP 4, otherwise go to FIN

STEP 4: Remove the predictor

STEP 5: Fit model without this variable*

FIN: Your Model Is Ready

Building A Model

Forward Selection

STEP 1: Select a significance level to enter the model (e.g. SL = 0.05)

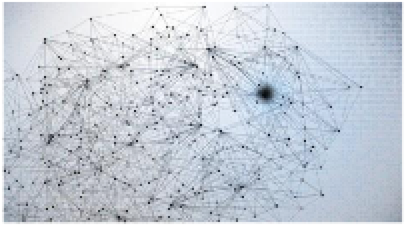
STEP 2: Fit all simple regression models $\mathbf{y} \sim \mathbf{x}_n$ Select the one with the lowest P-value

STEP 3: Keep this variable and fit all possible models with one extra predictor added to the one(s) you already have

STEP 4: Consider the predictor with the <u>lowest</u> P-value. If P < SL, go to STEP 3, otherwise go to FIN

FIN: Keep the previous model

Building A Model



All Possible Models

STEP 1: Select a criterion of goodness of fit (e.g. Akaike criterion)

STEP 2: Construct All Possible Regression Models: 2^N-1 total combinations

STEP 3: Select the one with the best criterion

FIN: Your Model Is Ready

Example:
10 columns means
1,023 models

1 Backward Elimination Model

```
# Building the optimal model using Backward Elimination
# Backward Elimination with p-values only
import statsmodels.formula.api as sm
SL = 0.05
X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)
X_opt = X[:, [0, 1, 2, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

executed in 500ms, finished 14:24:30 2019-04-09

Dep. Variable:	y	R-squared:	0.951
Model:	OLS	Adj. R-squared:	0.945
Method:	Least Squares	F-statistic:	169.9
Date:	Tue, 09 Apr 2019	Prob (F-statistic):	1.34e-27
Time:	14:24:30	Log-Likelihood:	-525.38
No. Observations:	50	AIC:	1063.
Df Residuals:	44	BIC:	1074.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.013e+04	6884.820	7.281	0.000	3.62e+04	6.4e+04
x1	198.7888	3371.007	0.059	0.953	-6595.030	6992.607
x2	-41.8870	3256.039	-0.013	0.990	-6604.003	6520.229
x3	0.8060	0.046	17.369	0.000	0.712	0.900
x4	-0.0270	0.052	-0.517	0.608	-0.132	0.078
x5	0.0270	0.017	1.574	0.123	-0.008	0.062

The Biggest one and > SL

Omnibus:	14.782	Durbin-Watson:	1.283
Prob(Omnibus):	0.001	Jarque-Bera (JB):	21.266
Skew:	-0.948	Prob(JB):	2.41e-05
Kurtosis:	5.572	Cond. No.	1.45e+06

Hassan Badawy

```python
X_opt = X[:, [0, 1, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

executed in 38ms, finished 14:24:56 2019-04-09

Check P-Value If > SL

Drop the Feature

```python
X_opt = X[:, [0, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

executed in 46ms, finished 14:25:04 2019-04-09

Check P-Value If > SL

Drop the Feature

```python
X_opt = X[:, [0, 3, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

executed in 38ms, finished 14:25:11 2019-04-09

Check P-Value If > SL

Drop the Feature

```python
X_opt = X[:, [0, 3]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

executed in 47ms, finished 14:25:22 2019-04-09

Hassan Badawy

Task : Automate the Back Elimination Processes

```
regressor_OLS = sm.OLS(y, x).fit()
maxVar = max(regressor_OLS.pvalues).astype(float)
adjR_before = regressor_OLS.rsquared_adj.astype(float)
```

```
if (regressor_OLS.pvalues[j].astype(float) == maxVar):
```

Hassan Badawy