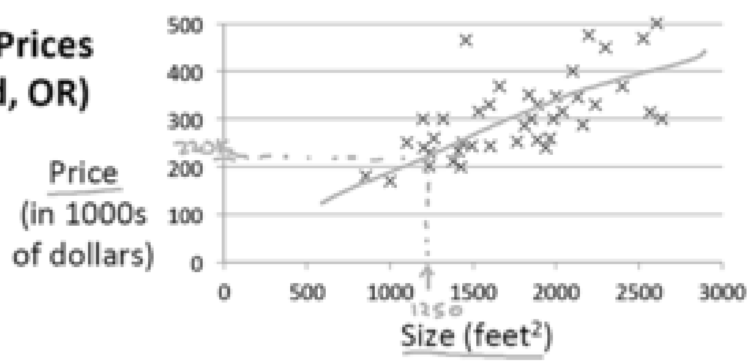
Linear Regression with One Variable

MOHAN M J

Housing Prices (Portland, OR)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Classification: Discrete-valued output

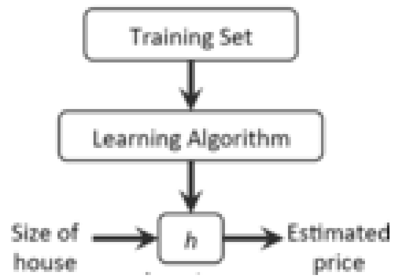
Model Representation

Predict real-valued output

In regression problems, we are taking input variables and trying to fit the output onto a continuous expected result function

Linear regression with one variable or 'univariate linear regression'

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$$



Modeling

How to fit best possible model to our given data?

Training Set

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

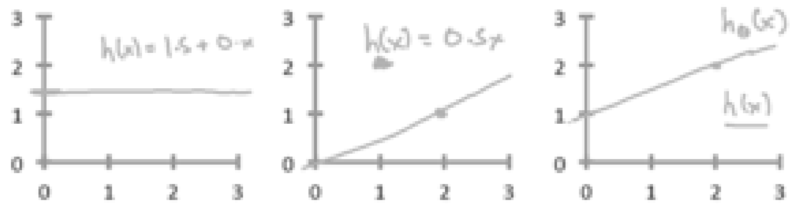
m = 47

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's: Parameters

How to choose θ_i's ?

$h_\theta(x) = \theta_0 + \theta_1 x$



$\theta_0 = 1.5$
$\theta_1 = 0$

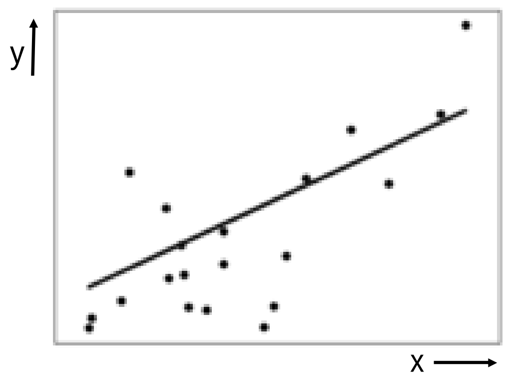
$\theta_0 = 0$
$\theta_1 = 0.5$

$\theta_0 = 1$
$\theta_1 = 0.5$

Cost Function

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

The straight line can be seen in the plot, showing how linear regression attempts to draw a straight line that will best minimize the residual sum of squares between the observed responses in the dataset.



Cost Function

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

We can measure the accuracy of model by using a cost function

This takes an average of all the results of the model with inputs from x's compared to the actual output y's

Difference between the predicted value and the actual value

"Squared error function“ or "Mean squared error“

We will be able to concretely measure the accuracy of our predictor function against the correct results using ‘Cost Function’

Training data set is scattered on the x-y plane. We are trying to make straight line which passes through this scattered set of data. The objective is to get the best possible line => **Minimize the cost function**

Cost Function - Intuition

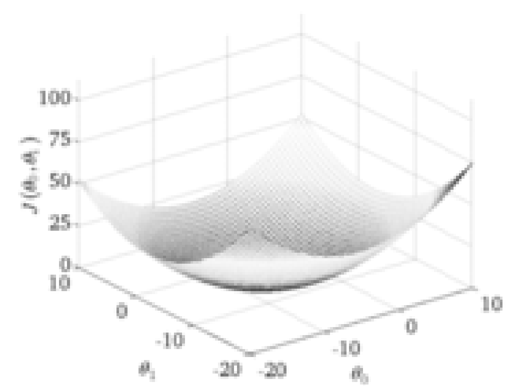
Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

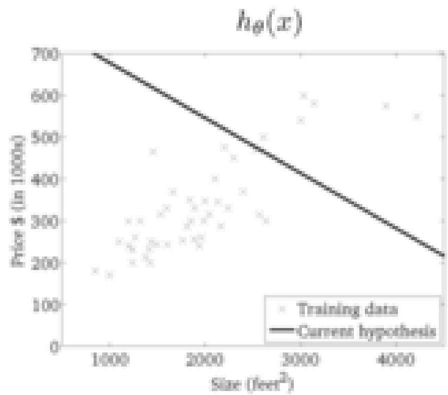
Cost Function:

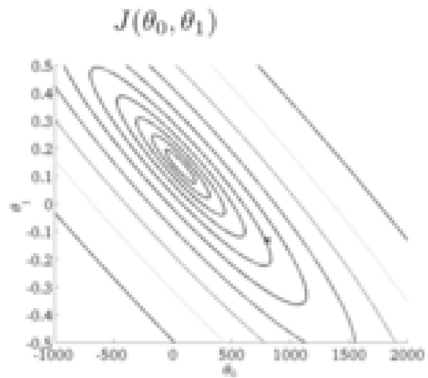
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$



Cost Function - Intuition





Gradient Descent

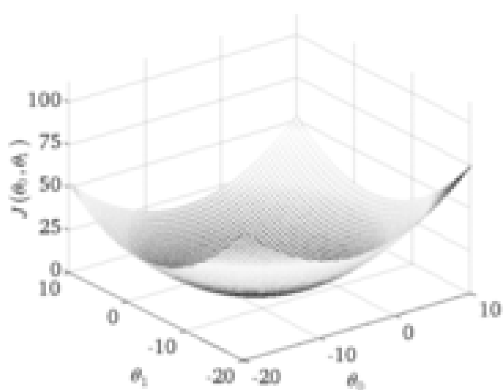
The gradient descent algorithm is:

repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

where

j=0,1 represents the feature index number



Gradient Descent for Linear Regression

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

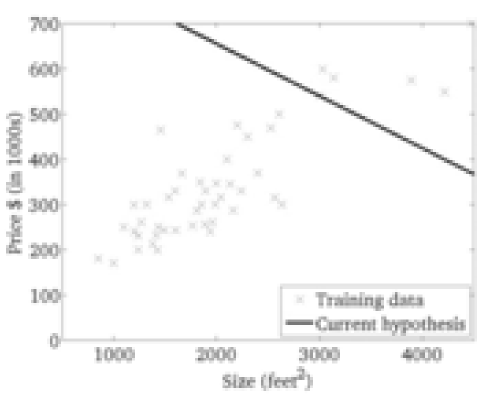
}

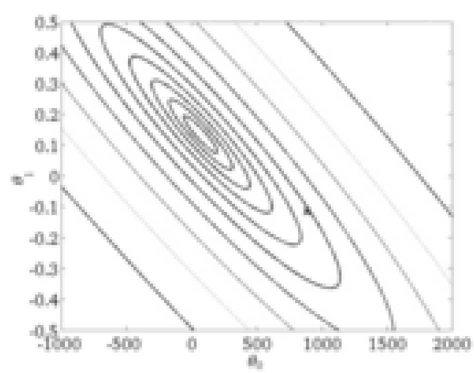
Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Gradient Descent Algorithm





Gradient Descent for Linear Regression

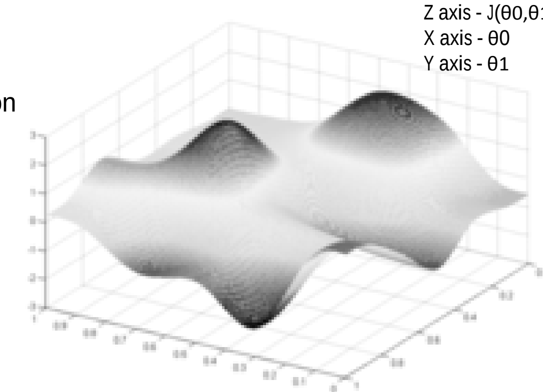
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

Substituting actual cost function and model function

Repeat until convergence:

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

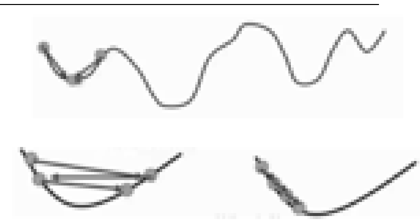
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i)$$

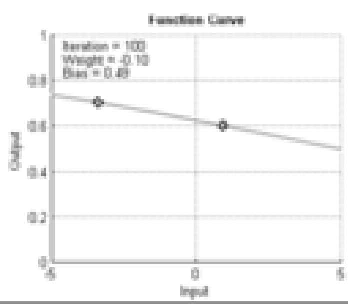


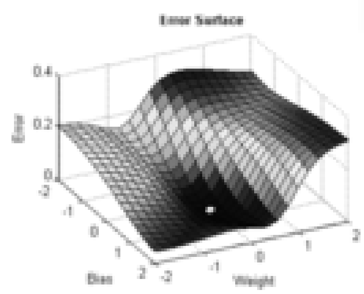
Z axis - J(θ0,θ1)
X axis - θ0
Y axis - θ1

Gradient Descent Intuition

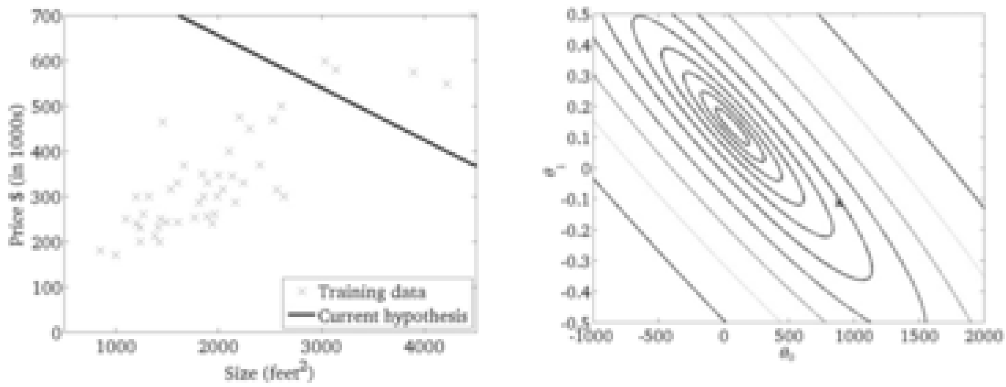
- ➢α : Learning Rate
- ➢Sensitive Starting Point
- ➢As we approach a local minimum, gradient descent will automatically take smaller steps







Gradient Descent Algorithm

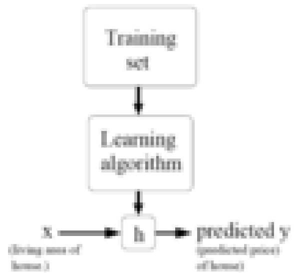


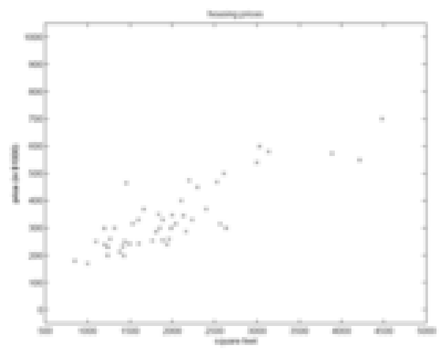
Linear Regression with Multiple Variables

Intro

When the target variable that we are trying to predict is continuous, such as our housing example, we can call the learning problem a regression problem

Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮





Intro

Living area (feet^2)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

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

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta).$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}\left(h_\theta(x) - y\right)^2\\ &= 2\cdot\frac{1}{2}\left(h_\theta(x) - y\right)\cdot\frac{\partial}{\partial\theta_j}\left(h_\theta(x) - y\right)\\ &= \left(h_\theta(x) - y\right)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right)\\ &= \left(h_\theta(x) - y\right)x_j\end{aligned}$$

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}.$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$$h(x) = \sum_{i=0}^{n}\theta_i x_i = \theta^T x,$$

Repeat until convergence {

$$\theta_j := \theta_j + \alpha\sum_{i=1}^{m}\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

EXERCISE 1:

The effect of temperature and reaction time affects the %yield. The data collected is given in the Mult-Reg_yield file. Develop a model for %yield in terms of temperature and time

STEP1: Read Data

```
import pandas as mypanda
from scipy import stats as mystats
import matplotlib.pyplot as myplot
from pandas.tools.plotting import scatter_matrix
from statsmodels.formula.api import ols as myols
myData=mypanda.read_csv('.\datasets\Mult_Reg_Yield.csv')
myData
tmp=myData.Temperature
yld =myData.Yield
time=myData.Time
scatter_matrix(myData) # Correlation analysis
myplot.show()
```

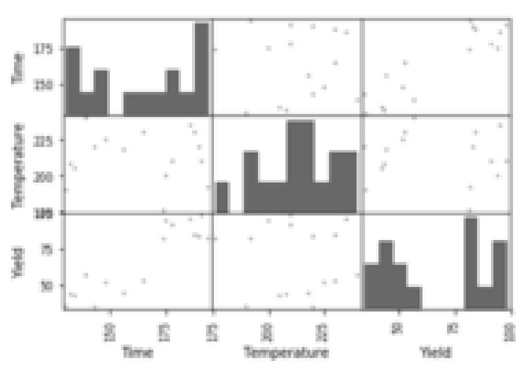
Correlation Analysis

```
np.corrcoef(tmp,yld)
array([[ 1.        , -0.05456951],
       [-0.05456951,  1.        ]])

np.corrcoef(time,yld)
array([[ 1.        ,  0.89671196],
       [ 0.89671196,  1.        ]])

np.corrcoef(time,tmp)
array([[ 1.        , -0.00756007],
       [-0.00756007,  1.        ]])
```

```
scatter_matrix(myData)
myplot.show()
```



STEP 2: Regression Output

```
mymodel=myols("yld ~ time + tmp",myData)
mymodel=mymodel.fit()
mymodel.summary()
```

Dep. Variable:	yld	R-squared:	0.806
Model:	OLS	Adj. R-squared:	0.777
Method:	Least Squares	F-statistic:	27.07
Date:	Wed, 21 Mar 2018	Prob (F-statistic):	2.32e-05
Time:	12:06:08	Log-Likelihood:	-59.703
No. Observations:	16	AIC:	125.4
Df Residuals:	13	BIC:	127.7
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-67.8844	40.587	-1.673	0.118	-155.566	19.797
time	0.9061	0.123	7.344	0.000	0.640	1.173
tmp	-0.0642	0.164	-0.392	0.702	-0.418	0.290

Omnibus:	1.984	Durbin-Watson:	1.957
Prob(Omnibus):	0.371	Jarque-Bera (JB):	0.970
Skew:	-0.078	Prob(JB):	0.616
Kurtosis:	1.804	Cond. No.	3.91e+03

Regression Output

```
mymodel=myols("yld ~ time ",myData).fit()
mymodel.summary()
```

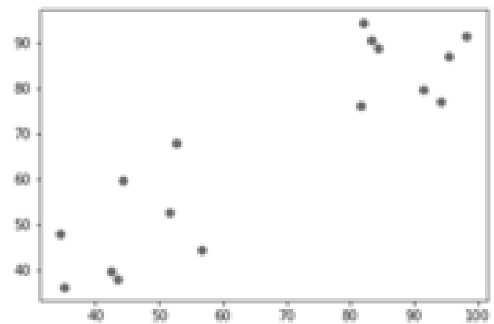
Dep. Variable:	yld	R-squared:	0.804
Model:	OLS	Adj. R-squared:	0.790
Method:	Least Squares	F-statistic:	57.46
Date:	Thu, 14 Sep 2017	Prob (F-statistic):	2.55e-06
Time:	10:12:02	Log-Likelihood:	-59.797
No. Observations:	16	AIC:	123.6
Df Residuals:	14	BIC:	125.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-81.6205	19.791	-4.124	0.001	-124.067	-39.174
time	0.9065	0.120	7.580	0.000	0.650	1.163

Omnibus:	1.894	Durbin-Watson:	2.055
Prob(Omnibus):	0.388	Jarque-Bera (JB):	0.969
Skew:	-0.127	Prob(JB):	0.616
Kurtosis:	1.822	Cond. No.	1.21e+03

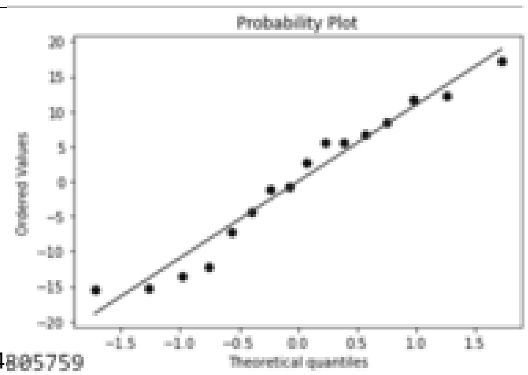
STEP 3:

```
pred=mymodel.predict()
res=yld-pred
res
myplot.scatter(yld,pred)
myplot.show()
# There need to be strong positive
correlation between actual and fitted
response
```



Residual Analysis

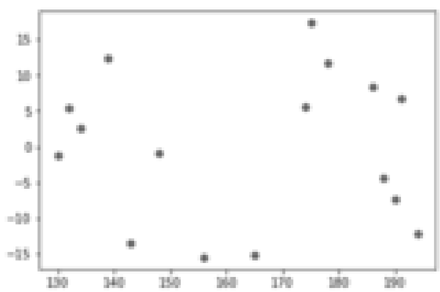
```
mystats.probplot(res,plot=myplot)
myplot.show()
```



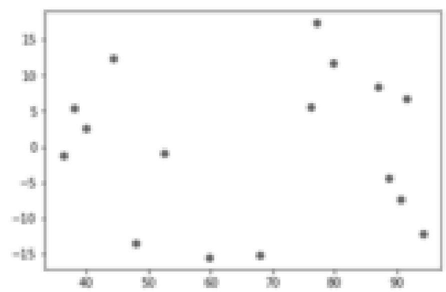

```
#Normality Test
mystats.normaltest(res)
Out [] NormaltestResult(statistic=1.8944805759
       902918, pvalue=0.38780979136720556)
```

Model Adequacy Check

There should not be any pattern or trend, the points should be distributed randomly



```
#Residual vs independent variables
myplot.scatter(time,res)
myplot.show()
```



```
#Residual vs fitted
myplot.scatter(pred,res)
myplot.show()
```

THANK YOU