Introduction to Data Science CS61 June 12 - July 12, 2018

Dr. Ash Pahwa

Lesson 5: Regression

Lesson 5.2: Regression – Python/Scikit-Learn

Outline

- Example 1: Single Variable Regression: Price/Demand Data set
  - R
    - Matrix Solution in R
    - Regression using ‘lm’ function in R
- Example 2: Single Variable Regression: Price/Demand Data set
  - Python/Scikit-Learn
    - Matrix Solution in Python/Scikit-Learn
    - ‘Regression’ function in Python/Scikit-Learn
- Example 3: Multi Variables Regression: Advertising Dataset
  - R
    - Regression using ‘lm’ function in R
  - Python/Scikit-Learn
    - Split Train/Test in Python/Scikit-Learn
    - Regression using ‘Regression’ function in Python/Scikit-Learn

Closed Form Solution

Linear Regression Using Matrices

2 Variable Regression

- Systems of Linear Equations
  - $y_1 = (b + mx_1) + e_1$
  - $y_2 = (b + mx_2) + e_2$
  - …
  - $y_n = (b + mx_n) + e_n$

X	Y
x_1	y_1
x_2	y_2
x_3	y_3
…	…
x_n	y_n

- $Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_n \end{bmatrix} \quad A = \begin{bmatrix} b \\ m \end{bmatrix} \quad E = \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_n \end{bmatrix}$

- $\boldsymbol{A = (X^T X)^{-1} X^T Y}$

Price	Demand
$49	124
$69	95
$89	71
$99	45
$109	18

Example 1: R

Matrix Approach

Using ‘lm’ Function

Example-2
Solution in R – Using Matrix

Regression Equation
$y = -1.6948x + 211.2707$

$$A = (X^T X)^{-1} X^T Y$$

```
> price = c(49, 69, 89, 99, 109)
> demand = c(124, 95, 71, 45, 18)
> plot(price,demand)
>
> ################################
> Oneprice = c(1,1,1,1,1,price)
> Xprice = matrix(Oneprice,nrow=5)
> Xprice
     [,1] [,2]
[1,]    1   49
[2,]    1   69
[3,]    1   89
[4,]    1   99
[5,]    1  109
>
> ###############################
> demand = c(124, 95, 71, 45, 18)
> Ydemand = matrix(demand,nrow=5)
> Ydemand
     [,1]
[1,]  124
[2,]   95
[3,]   71
[4,]   45
[5,]   18
```

```
> #########################
> z1 = t(Xprice)%*%Xprice
> z1
     [,1]  [,2]
[1,]    5   415
[2,]  415 36765
> # comput the inverse of z
> det(z1)
[1] 11600
> invz1 = solve(z1)
>
> ##########################
> z2 = t(Xprice)%*%Ydemand
> z2
      [,1]
[1,]   353
[2,] 25367
> ##########################
> ans = invz1 %*% z2
> ans
           [,1]
[1,] 211.270690
[2,]  -1.694828
>
```

Example-2
Solution in R – Using 'lm' command

```
> summary(lm(demand~price))

Call:
lm(formula = demand ~ price)

Residuals:
      1       2       3       4       5 
-4.2241  0.6724 10.5690  1.5172 -8.5345 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 211.2707    14.7215  14.351 0.000733 ***
price        -1.6948     0.1717  -9.872 0.002210 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.269 on 3 degrees of freedom
Multiple R-squared:  0.9701,	Adjusted R-squared:  0.9602 
F-statistic: 97.46 on 1 and 3 DF,  p-value: 0.00221
```

Regression Equation

$y = -1.6948x + 211.2707$

Example 2: Python/Scikit-Learn

Matrix Approach

Setup
NumPy + Matplotlib

Jupyter

Setup

```
In [2]: # Common imports
        import numpy as np
        np.random.seed(42)

        # To plot pretty figures
        %matplotlib inline
        import matplotlib
        import matplotlib.pyplot as plt
```

Spyder

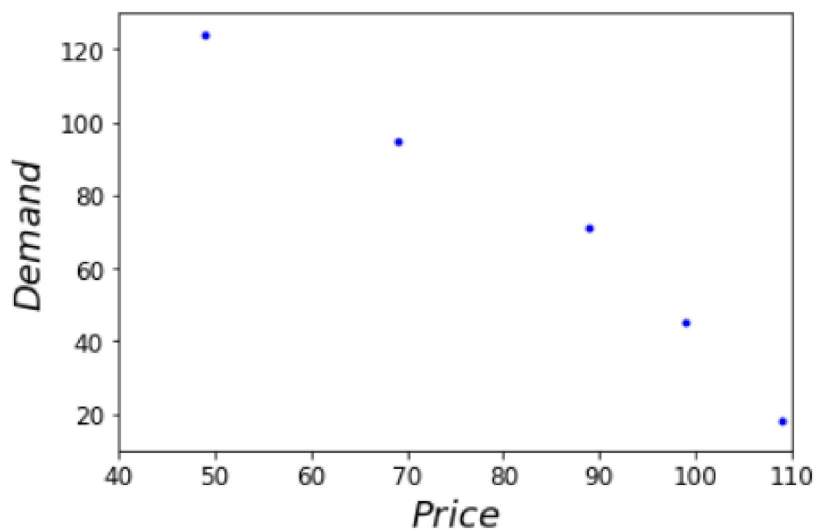
```
import numpy as np
import matplotlib.pyplot as plt
```

Read and Plot the Dataset Jupyter

Linear regression using Matrix

In [14]:
```
X = [49, 69, 89, 99, 109]
y = [124, 95, 71, 45, 18]
```

In [19]:
```
plt.plot(X, y, "b.")
plt.xlabel("$Price$", fontsize=18)
plt.ylabel("$Demand$", rotation=90, fontsize=18)
plt.axis([40, 110, 10, 130])
plt.show()
```

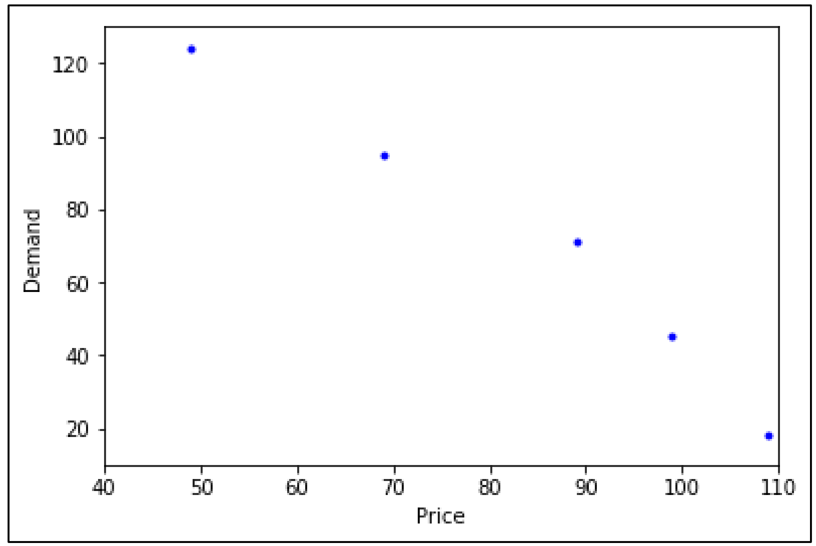


Price	Demand
$49	124
$69	95
$89	71
$99	45
$109	18

Read and Plot the Dataset Spyder

Price	Demand
$49	124
$69	95
$89	71
$99	45
$109	18

```
X = [49, 69, 89, 99, 109]
y = [124, 95, 71, 45, 18]

plt.figure()
plt.plot(X, y, "b.")
plt.xlabel('Price')
plt.ylabel('Demand')
plt.xlim([40, 110])
plt.ylim([10, 130])
```



- Solution for Least RSS = $A = (X^T X)^{-1} X^T Y$

Matrix Solution

```
In [21]: X_b = np.c_[np.ones((5,1)), X]
         print(X_b)

[[   1.   49.]
 [   1.   69.]
 [   1.   89.]
 [   1.   99.]
 [   1.  109.]]

In [22]: X_T_X = X_b.T.dot(X_b)
         print(X_T_X)

[[  5.00000000e+00   4.15000000e+02]
 [  4.15000000e+02   3.67650000e+04]]

In [25]: X_T_X_I = np.linalg.inv(X_T_X)
         print(X_T_X_I)

[[  3.16939655e+00  -3.57758621e-02]
 [ -3.57758621e-02   4.31034483e-04]]
```

- Solution for Least RSS = $A = (X^T X)^{-1} X^T Y$

Matrix Solution

```
In [8]: X_T_Y = X_b.T.dot(y)
        print(X_T_Y)

        [   353.   25367.]

In [9]: answer = X_T_X_I.dot(X_T_Y)
        print(answer)

        [ 211.27068966   -1.69482759]
```

Regression Equation

$$y = -1.6948x + 211.2707$$

Example 2: Python/Scikit-Learn

Using Regression Function

Setup

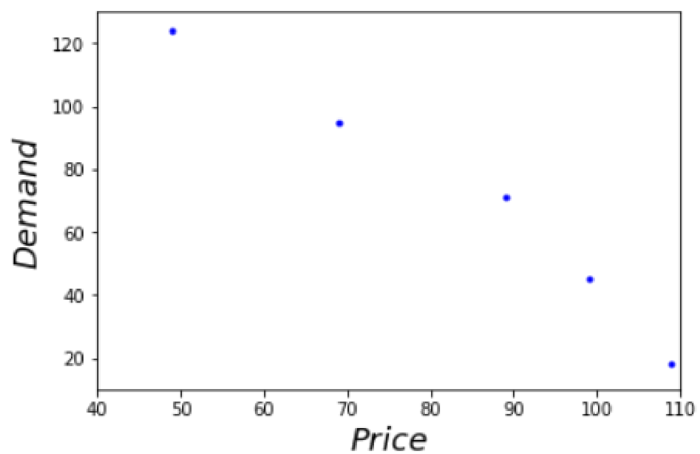
Linear regression

In [19]:
```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.cross_validation import train_test_split

# To plot pretty figures
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
```

Read and Plot the Dataset

```
In [6]: X = [49, 69, 89, 99, 109]
        y = [124, 95, 71, 45, 18]
```

```
In [7]: plt.plot(X, y, "b.")
        plt.xlabel("$Price$", fontsize=18)
        plt.ylabel("$Demand$", rotation=90, fontsize=18)
        plt.axis([40, 110, 10, 130])
        plt.show()
```



DataFrame

```
In [8]: import pandas as pd
        df_x = pd.DataFrame(X)
        print(df_x)

     0
0   49
1   69
2   89
3   99
4  109
```

```
In [9]: df_y = pd.DataFrame(y)
        print(df_y)

     0
0  124
1   95
2   71
3   45
4   18
```

```
In [18]: df_x.describe()
Out[18]:
```

	0
count	5.000000
mean	83.000000
std	24.083189
min	49.000000
25%	69.000000
50%	89.000000
75%	99.000000
max	109.000000

```
In [11]: df_y.describe()
Out[11]:
```

	0
count	5.000000
mean	70.600000
std	41.440319
min	18.000000
25%	45.000000
50%	71.000000
75%	95.000000
max	124.000000

Regression

```
In [12]: reg=linear_model.LinearRegression()

In [13]: reg.fit(df_x, df_y)

Out[13]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [14]: print(reg.coef_)
         print(reg.intercept_)

[[-1.69482759]]
[ 211.27068966]
```

Regression Equation

$$y = -1.6948x + 211.2707$$

Example 3: R

Multi Variable Regression

Raw Data

	A	B	C	D	E
1		TV	Radio	Newspape	Sales
2	1	230.1	37.8	69.2	22.1
3	2	44.5	39.3	45.1	10.4
4	3	17.2	45.9	69.3	9.3
5	4	151.5	41.3	58.5	18.5
6	5	180.8	10.8	58.4	12.9
7	6	8.7	48.9	75	7.2
8	7	57.5	32.8	23.5	11.8
9	8	120.2	19.6	11.6	13.2
10	9	8.6	2.1	1	4.8
11	10	199.8	2.6	21.2	10.6
12	11	66.1	5.8	24.2	8.6
13	12	214.7	24	4	17.4
14	13	23.8	35.1	65.9	9.2
15	14	97.5	7.6	7.2	9.7
16	15	204.1	32.9	46	19
17	16	195.4	47.7	52.9	22.4
18	17	67.8	36.6	114	12.5

Read Data

```
Ad = read.csv("Advertising.csv")
Ad
#fix(Ad)

Sales = Ad$Sales

TV = Ad$TV
Radio = Ad$Radio
Newspaper = Ad$Newspaper

#####################################################
model1 = lm(Sales ~ TV)
summary(model1)
plot(TV,Sales,pch=21,col="blue",bg="red")
abline(model1,col="blue",lwd=3)

#####################################################
model2 = lm(Sales ~ Radio)
summary(model2)
plot(Radio,Sales,pch=21,col="blue",bg="red")
abline(model2,col="blue",lwd=3)

####################################################
model3 = lm(Sales ~ Newspaper)
summary(model3)
plot(Newspaper,Sales,pch=21,col="blue",bg="red")
abline(model3,col="blue",lwd=3)

####################################################
```

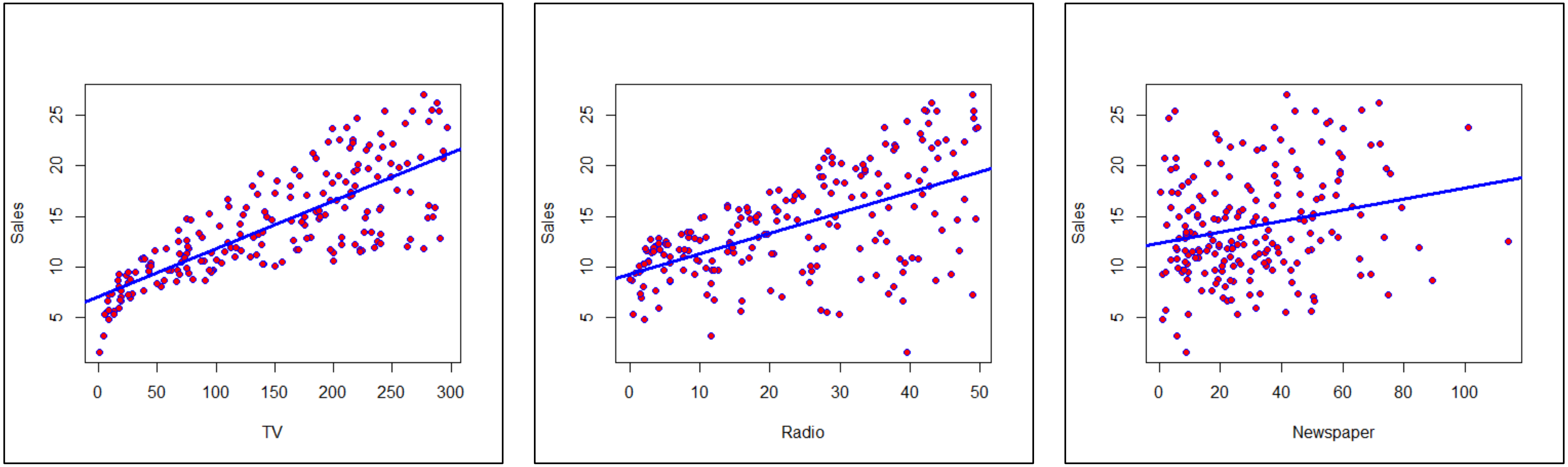
Regression

```
> summary(model1)

Call:
lm(formula = Sales ~ TV)

Residuals:
    Min      1Q  Median      3Q     Max
-8.3860 -1.9545 -0.1913  2.0671  7.2124

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.032594   0.457843   15.36   <2e-
TV          0.047537   0.002691   17.67   <2e-
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.

Residual standard error: 3.259 on 198 degrees
Multiple R-squared:  0.6119,       Adjusted R
F-statistic: 312.1 on 1 and 198 DF,  p-value:
```

```
> model2 = lm(Sales ~ Radio)
> summary(model2)

Call:
lm(formula = Sales ~ Radio)

Residuals:
     Min       1Q   Median       3Q      Max
-15.7305  -2.1324   0.7707   2.7775   8.1810

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.31164    0.56290  16.542   <2e-16 ***
Radio        0.20250    0.02041   9.921   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.275 on 198 degrees of freedom
Multiple R-squared:  0.332,       Adjusted R-squared:  0.3287
F-statistic: 98.42 on 1 and 198 DF,  p-value: < 2.2e-16
```

```
> model3 = lm(Sales ~ Newspaper)
> summary(model3)

Call:
lm(formula = Sales ~ Newspaper)

Residuals:
     Min       1Q   Median       3Q      Max
-11.2272  -3.3873  -0.8392   3.5059  12.7751

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.35141    0.62142   19.88  < 2e-16 ***
Newspaper    0.05469    0.01658    3.30  0.00115 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.092 on 198 degrees of freedom
Multiple R-squared:  0.05212,      Adjusted R-squared:  0.04733
F-statistic: 10.89 on 1 and 198 DF,  p-value: 0.001148
```

Plot Data



Sales
25
20
15
10
5
0
50
100
150
200
250
300
TV
Sales
25
20
15
10
5
0
10
20
30
40
50
Radio
Sales
25
20
15
10
5
0
20
40
60
80
100
Newspaper

Multi Variable Regression

```
> model4 = lm(Sales ~ TV + Radio + Newspaper)
> summary(model4)

Call:
lm(formula = Sales ~ TV + Radio + Newspaper)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.8277 -0.8908  0.2418  1.1893  2.8292 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.938889   0.311908   9.422   <2e-16 ***
TV           0.045765   0.001395  32.809   <2e-16 ***
Radio        0.188530   0.008611  21.893   <2e-16 ***
Newspaper   -0.001037   0.005871  -0.177     0.86    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.686 on 196 degrees of freedom
Multiple R-squared:  0.8972,	Adjusted R-squared:  0.8956 
F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16

>
```

$$\boxed{y = 2.9388 + (0.0457 * TV) + (0.1885 * Radio) - (0.001 * Newspaper)}$$

Example 3: Python/Scikit-Learn

Using Regression Function

Setup

Linear regression

In [1]:
```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.cross_validation import train_test_split

# To plot pretty figures
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
```

Read Data

Read the data File

In [13]:
```
data = pd.read_csv("Advertising.csv", index_col=0)
```

In [14]:
```
data.head()
```

Out[14]:

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

In [15]:
```
data.tail()
```

Out[15]:

	TV	Radio	Newspaper	Sales
196	38.2	3.7	13.8	7.6
197	94.2	4.9	8.1	9.7
198	177.0	9.3	6.4	12.8
199	283.6	42.0	66.2	25.5
200	232.1	8.6	8.7	13.4

In [10]:
```
data.shape
```

Out[10]: (200, 5)

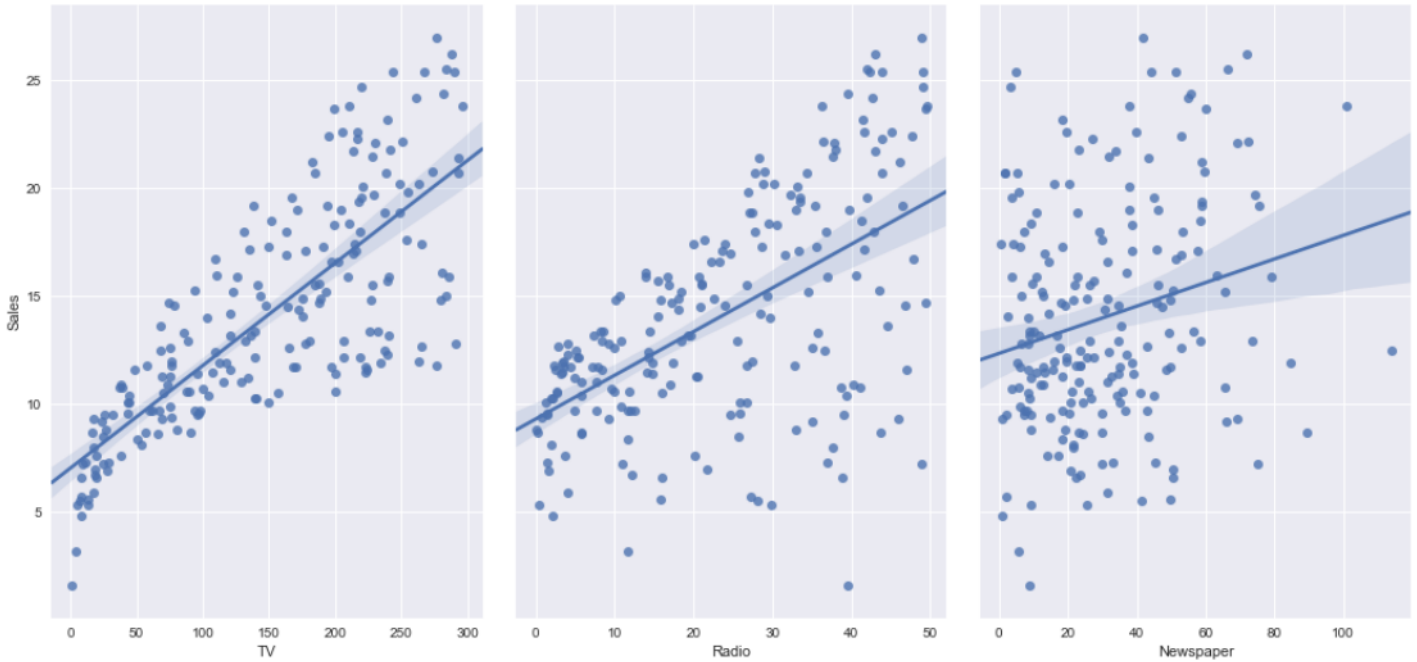
Plot Data

Plot the Data

In [16]:
```
import seaborn as sns
%matplotlib inline
```

In [19]:
```
sns.pairplot(data,x_vars=['TV','Radio','Newspaper'], y_vars='Sales', size=7, aspect=0.7, kind='reg')
```

Out[19]: `<seaborn.axisgrid.PairGrid at 0x1f571fb4278>`



Predictor Variables

Data Preparation

In [26]:
```
feature_cols = ['TV', 'Radio', 'Newspaper']
```

In [27]:
```
X = data[feature_cols]
```

In [36]:
```
X = data[['TV','Radio', 'Newspaper']]
```

In [37]:
```
X.head()
```

Out[37]:

	TV	Radio	Newspaper
1	230.1	37.8	69.2
2	44.5	39.3	45.1
3	17.2	45.9	69.3
4	151.5	41.3	58.5
5	180.8	10.8	58.4

In [40]:
```
print (type(X))
print (X.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
(200, 3)
```

Response Variable

```
In [41]: y = data['Sales']

In [42]: y = data.Sales

In [43]: y.head()

Out[43]: 1    22.1
         2    10.4
         3     9.3
         4    18.5
         5    12.9
         Name: Sales, dtype: float64

In [44]: print (type(y))

         <class 'pandas.core.series.Series'>

In [46]: print (y.shape)

         (200,)
```

Split Data
Training and Test

Splitting X and y into training and testing sets

In [52]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=1)
```

In [53]:
```
print (X_train.shape)
print (y_train.shape)
print (X_test.shape)
print (y_test.shape)
```

```
(150, 3)
(150,)
(50, 3)
(50,)
```

Build Regression Model Using Training Data

Linear Regression

```
In [55]: linreg = linear_model.LinearRegression()

In [56]: linreg.fit(X_train, y_train)

Out[56]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Interpret the Model Coefficients

```
In [59]: print (linreg.intercept_)
         print (linreg.coef_)

2.87696662232
[ 0.04656457  0.17915812  0.00345046]

In [62]: print (zip(feature_cols, linreg.coef_))

<zip object at 0x000001F572FB7E48>
```

y = 2.877 + (0.047 x TV) + (0.179 x Radio) + (0.003 * Newspaper)

$$y = 2.9388 + (0.0457 * TV) + (0.1885 * Radio) - (0.001 * Newspaper)$$

Prediction of the Test Data

Making Prediction

In [30]:
```
y_pred = linreg.predict(X_test)
print (y_pred)
```

```
[ 21.70910292  16.41055243   7.60955058  17.80769552  18.6146359
  23.83573998  16.32488681  13.43225536   9.17173403  17.333853
  14.44479482   9.83511973  17.18797614  16.73086831  15.05529391
  15.61434433  12.42541574  17.17716376  11.08827566  18.00537501
   9.28438889  12.98458458   8.79950614  10.42382499  11.3846456
  14.98082512   9.78853268  19.39643187  18.18099936  17.12807566
  21.54670213  14.69809481  16.24641438  12.32114579  19.92422501
  15.32498602  13.88726522  10.03162255  20.93105915   7.44936831
   3.64695761   7.22020178   5.9962782   18.43381853   8.39408045
  14.08371047  15.02195699  20.35836418  20.57036347  19.60636679]
```

Model Evaluation

Root mean Square Error (RMSE) is defined as follows.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[f(x_i) - y_i)]^2}$$

Here $f(x_i)$ is the computed value and y_i is the true (observed) value.

Model Evaluation

In [31]:
```
import numpy as np
np.sqrt(sum((y_test-y_pred)**2)/50)
```

Out[31]: 1.4046514230328953

Regression Without Splitting Data into Training and Testing

```
linreg = linear_model.LinearRegression()

linreg.fit(X, y)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

print (linreg.intercept_)
2.93888936946

print (linreg.coef_)
[ 0.04576465  0.18853002 -0.00103749]
```

Python $\boxed{y = 2.9388 + (0.0457 * TV) + (0.1885 * Radio) - (0.001 * Newspaper)}$

R $\boxed{y = 2.9388 + (0.0457 * TV) + (0.1885 * Radio) - (0.001 * Newspaper)}$

Summary

- Example 1: Single Variable Regression: Price/Demand Data set
  - R
    - Matrix Solution in R
    - Regression using 'lm' function in R
- Example 2: Single Variable Regression: Price/Demand Data set
  - Python/Scikit-Learn
    - Matrix Solution in Python/Scikit-Learn
    - 'Regression' function in Python/Scikit-Learn
- Example 3: Multi Variables Regression: Advertising Dataset
  - R
    - Regression using 'lm' function in R
  - Python/Scikit-Learn
    - Split Train/Test in Python/Scikit-Learn
    - Regression using 'Regression' function in Python/Scikit-Learn