Regression Model

1

Simple Linear Regression

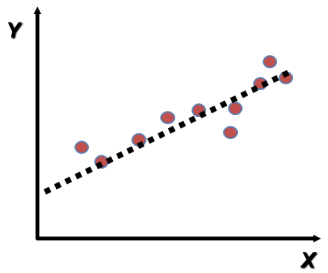
- Two variable case: x – independent (predictor) variable, y – dependent (output variable)

  Given $\{x_i, y_i\}$, i= 1, N
  Find a_0 and a_1 to fit the following model
  $\hat{y}_i = a_0 + a_1 x_i$; i =1, N
  Through simple minimization, the following solution is obtained

$$a_1 = \frac{\sum_i x_i y_i - (\sum_i x_i)(\sum_i y_i)/N}{\sum_i x_i^2 - (\sum_i x_i)^2/N}$$

Steps?

$$a_0 = \bar{y} - a_1 \bar{x}$$



2

Numerical Example

Number of employees (x)	Total contract sales (in $10,000's) y
10	13
9	11
8	12
7	10
6	8
5	6
4	5
3	3
2	4
1	3

$\hat{y}i = 0.79 + 1.218\ xi$

$y_i - \hat{y}_i$ Residual

The prediction error is called the *residual sum of square error*. The *residual variance* is a measure of error fluctuation. The square root of the residual variance is called the *standard error of the estimate*.

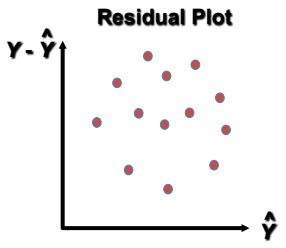
$SSE = \Sigma(y_i - \hat{y}_i)^2$

3

Assessing Model Goodness

Residuals should have…

- Randomness
- Constant Variance
- Normal Distribution



SSR stands for Sum of Squares due to Regression. Measures how far are predicted values from mean of the dependent variable

$SSR = \Sigma(\hat{Y}_i - \bar{Y})^2$

$SST = \Sigma(Y_i - \bar{Y})^2$

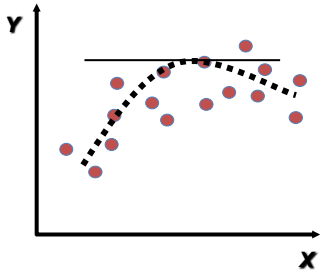
SST stands for Sum of Squares Total. Similar to variance.

$R^2 = SSR/SST$

Coefficient of Determination; it measures the proportion of variation in Y explained by (the variation in) X. A value close to 1 is considered good. Why?

4

Regression model need not be linear

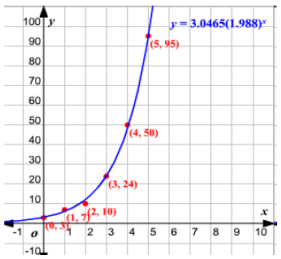


Quadratic regression model:

$Y = b_0 + b_1 X + b_2 X^2$

Logarithmic regression model: Y = a + b (ln x)

Exponential regression model: $Y = a*b^x$



5

Indicator (Dummy) Variables

Dummy, or indicator, variables allow for the inclusion of qualitative variables in the model

For example: $I = \begin{cases} 1 & \text{if female} \\ 0 & \text{if male} \end{cases}$

Model with Indicator variable:

$Y = b_0 + b_1 X + b_2 I$

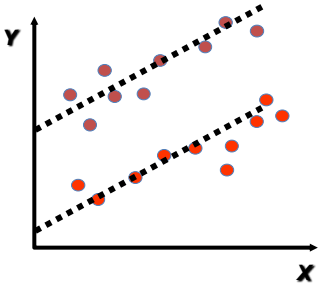
Rewrite the model as:

- For $I = 0$,

  $Y = b_0 + b_1 X$

- For $I = 1$,

$Y = (b_0 + b_2) + b_1 X$



6

Multiple Linear Regression

- When the number of predictor variables is more than one, then the term multiple linear regression is used
- In general, the multiple regression equation of y on x_1, x_2, …, x_k is given by:

  $y = a_0 + a_1 x_1 + a_2 x_2 + \ldots\ldots\ldots\ldots\ldots + a_k x_k$
- Representing a's and x's as vectors, and given a set of training examples, we can find **a** by minimizing

$$J(\mathbf{a}) = \|\mathbf{Y} - \mathbf{X}\mathbf{a}\|^2$$

Such a minimization can be carried out following the LMS rule discussed earlier. For small k, it can be also done using the pseudo-inverse approach

7

Solution Through Pseudo-inverse Matrix

$$\mathbf{Y} = \mathbf{X}\mathbf{a} \rightarrow \|\mathbf{Y} - \mathbf{X}\mathbf{a}\|^2$$

Applying minimization, we get

$$2\mathbf{X}^t(\mathbf{Y} - \mathbf{X}\mathbf{a}) = \mathbf{0} \rightarrow \mathbf{X}^t\mathbf{Y} = \mathbf{X}^t\mathbf{X}\mathbf{a}$$

$$\mathbf{a} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{Y} = \mathbf{X}^*\mathbf{Y}$$

$$\mathbf{X}^* = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t \leftarrow \text{Psuedo-inverse}$$

8

Example: Predicting the Number of Credit Cards

- The table on right shows a small part of a credit card company's database
- Build a model to predict the number of credit cards held by a customer

	# of CCs	Family Size (x1)	Income (x2)
1	4	2	14
2	6	2	16
3	6	4	14
4	7	4	17
5	8	5	18
6	7	5	21
7	8	6	17
8	10	6	25

Predicted # of Credit Cards = a0 + a1x1 + a2x2

9

$$\mathbf{X} = \begin{matrix} 1 & 2 & 14 \\ 1 & 2 & 16 \\ 1 & 4 & 14 \\ 1 & 4 & 17 \\ 1 & 5 & 18 \\ 1 & 5 & 21 \\ 1 & 6 & 17 \\ 1 & 6 & 25 \end{matrix}$$

$$\mathbf{X}^t = \begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 4 & 4 & 5 & 5 & 6 & 6 \\ 14 & 16 & 14 & 17 & 18 & 21 & 17 & 25 \end{matrix}$$

$$(\mathbf{X}^t\mathbf{X}) = \begin{matrix} 8 & 34 & 142 \\ 34 & 162 & 631 \\ 142 & 631 & 2616 \end{matrix}$$

$$(\mathbf{X}^t\mathbf{X})^{-1} = \begin{matrix} 3.5015 & 0.0899 & -0.2117 \\ 0.0899 & 0.1044 & -0.0301 \\ -0.2117 & -0.0301 & 0.0191 \end{matrix}$$

$$\mathbf{X}^* = \begin{matrix} 0.7168 & 0.2933 & 0.8966 & 0.2613 & 0.1395 & -0.4958 & 0.4411 & -1.2529 \\ -0.1221 & -0.1822 & 0.0866 & -0.0036 & 0.0708 & -0.0194 & 0.2052 & -0.0352 \\ -0.0041 & 0.0342 & -0.0642 & -0.0068 & -0.0178 & 0.0396 & -0.0669 & 0.0861 \end{matrix}$$

10

Y = [4 6 6 7 8 7 8 10]t

a = X*Y =

0.4817
0.6322
0.2158

Ŷ = [4.76 5.19 6.03 6.68 7.52 8.17 7.94 9.67]^t

11

Linear Regression Model Recapped

$\mathbf{x} \in \mathcal{R}^d \rightarrow \hat{y} = w_0 + \mathbf{w}^t\mathbf{x} \rightarrow y \in \mathcal{R}$ Target output

$\hat{y}$ Actual output

Find the model parameters using *n* pairs of examples:

$$\mathbf{x}_i, y_i, i = 1, n$$

12

Model:

$$\hat{y} = w_0 + \mathbf{w}^t\mathbf{x}$$

Model Parameters:

$$w_0, \mathbf{w}$$

Cost Function:

How do we choose model parameters?

By defining a cost function and selecting those parameter values that minimize the cost function

We will perform minimization using gradient descent

$$J(w_0, \mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - (w_0 + \mathbf{w}^t\mathbf{x}_i))^2$$

13

$$J(w_0, \mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - (w_0 + \mathbf{w}^t\mathbf{x}_i))^2$$

$$\frac{\partial J}{\partial w_0} = -\sum_{i=1}^{n}(y_i - (w_0 + \mathbf{w}^t\mathbf{x}_i))$$

$$\frac{\partial J}{\partial \mathbf{w}} = -\sum_{i=1}^{n}(y_i - (w_0 + \mathbf{w}^t\mathbf{x}_i))\mathbf{x}_i$$

We can now write the updating rule. To simplify writing equations, the bias term is absorbed in weight/parameter vector by appending 1 to X vector. We will assume this too for convenience. With this understanding, we have

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{w}^t\mathbf{x}_i)^2$$

$$\frac{\partial J}{\partial \mathbf{w}} = -\sum_{i=1}^{n}(y_i - \mathbf{w}^t\mathbf{x}_i)\mathbf{x}_i$$

Updating using all examples is known as *batch gradient descent*

14

LMS Rule

- Lets do solution updating after every training example. In that case:

$$\frac{\partial J_i}{\partial \mathbf{w}} = -(y_i - \mathbf{w}^t \mathbf{x}_i)\mathbf{x}_i$$

- Thus, updating rule becomes

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \alpha(y_i - \mathbf{w}_{old}^t \mathbf{x}_i)\mathbf{x}_i$$

This updating rule is also known delta or *Widrow-Hoff rule*

15

```
# This example does linear regression for a problem with 2 independent and one dependent variables.
N = 32
alpha = 1.3
beta = np.array([0.5, 1.9])
```

```
x0 = np.ones((N,1))
x_data = np.random.randn(N,2)
X = np.hstack((x0,x_data))# Absorbs contant term of 1
Y = x_data.dot(beta) + alpha + 0.1*np.random.randn(1)
```

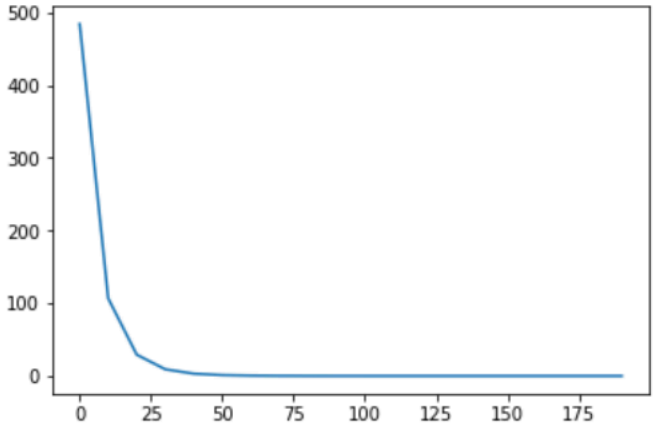
```
w = np.random.randn(3)# Initialize weight vector
learning_rate = 1e-3
result =[]
loss = 0
for t in range(200):
    # Forward pass: compute predicted y
    y_pred = X.dot(w)
    # Compute and print loss
    loss = np.square(y_pred - Y).sum()
    if t % 10 == 0:
        print(t, loss)
        result.append(loss)
  # Backprop to compute gradients of w1 and w2 with respect to loss
    grad_y_pred = 2.0 * (y_pred - Y)
    grad_w = X.T.dot(grad_y_pred)

    # Update weights
    w -= learning_rate * grad_w
```

16

```
print(w)
```

```
[1.25914417 0.50003894 1.89945083]
```

```
plt.plot(np.arange(0,200,10),result)
plt.show()
```



17

```
# Predict the power output of a plant based on historical readings
Data is in Excel file. After reading, print first two rows
```

```
#Read an Excel worksheet
file = '/Users/isethi/Downloads/CCPP/ccpp.xlsx'# Select the file
xl = pd.ExcelFile(file)#This loads the file
df = xl.parse('Sheet1')# Loads sheet1 as dataframeprint(df.shape)
print(df[0:2])
```

```
      AT      V       AP      RH      PE
0  14.96  41.76  1024.07  73.17  463.26
1  25.18  62.96  1020.04  59.08  444.37
```

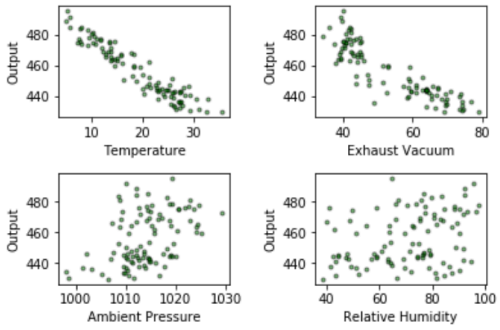
```
#Correlation matrix
corr = df[0:100].corr()
print(corr)
```

```
       AT      V      AP      RH      PE
AT  1.000  0.863 -0.448 -0.486 -0.950
V   0.863  1.000 -0.368 -0.236 -0.881
AP -0.448 -0.368  1.000 -0.012  0.432
RH -0.486 -0.236 -0.012  1.000  0.306
PE -0.950 -0.881  0.432  0.306  1.000
```

18

Visualize each independent variable wrt output variable

```
data = np.array(df[0:100])
X = data[:,:-1]
Y = data[:,-1]
heading = ['Temperature', 'Exhaust Vacuum', 'Ambient Pressure', 'Relative Humidity']
for i in range(4):
    plt.subplot(2,2,i+1)
    plt.scatter(X[:, i], Y,c = 'g',marker = ".",edgecolor='k', alpha = 0.50)
    plt.xlabel(heading[i])
    plt.ylabel('Output')
plt.subplots_adjust(wspace=0.50, hspace=0.50)
plt.show()
```



19

Lets divide the data into training and test sets

```
split = int(0.7*X.shape[0])
print (split)
```

```
70
```

```
X_train = X[:split,:]
Y_train = Y[:split]
X_test = X[split:,:]
Y_test = Y[split:]
```

```
from sklearn.linear_model import LinearRegression
regr = LinearRegression()
regr.fit(X_train, Y_train)
Y_pred = regr.predict(X_test)
```

```
print('Coefficients: \n', regr.coef_)
print('Intercept: \n', regr.intercept_)
mse = np.mean((Y_test-Y_pred)**2)
print('mse: \n',mse)
```

```
Coefficients:
 [-2.32024685 -0.0303063  -0.13458604 -0.2022868 ]
Intercept:
 653.7252298935637
mse:
 18.431693401310625
```

20

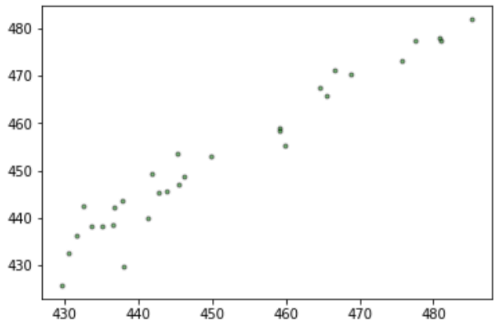
```
print('regr.Score: \n',regr.score(X_test,Y_test))# Score closed to 1 is a perfect fit;
#close to 0 means no linear relationship
```

```
regr.Score:
 0.9365276088114
```

```
from sklearn.model_selection import cross_val_score
np.mean(cross_val_score(LinearRegression(), X_train, Y_train, cv=10))
```

```
0.8938592147745295
```

```
plt.scatter(Y_test, Y_pred,c = 'g',marker = ".",edgecolor='k', alpha = 0.50)
plt.show()
```



21

Subset Selection in Regression

- Should we use all features or a subset in our multiple regression model?
  - Same problem as that of feature selection
    - Backward and forward stepwise selection methods
    - PCA
    - Shrinkage methods

22

Shrinkage Methods

- Basic idea?
  - Use additional constraints to reduce (shrink) coefficients
  - Since shrinking is a continuous operator compared to stepwise selection, better regression results with less variability are obtained
  - Ridge Regression
    - Minimize least squares subject to: $\sum_{j=1}^{k} a_j^2 \le s$
  - The LASSO (Least Absolute Shrinkage & Selection Operator)
    - Minimize least squares subject to: $\sum_{j=1}^{k} |a_j| \le s$

http://statweb.stanford.edu/~tibs/lasso.html

23

Shrinkage Methods

- Also help in avoiding overfitting
- The term *regularization* is also used in many instances

A closed form solution exists for ridge regression

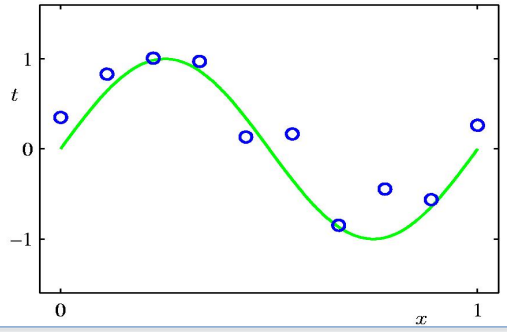
In Lasso, the solution is obtained via quadratic programming

The lasso does better when the response is a function of only a relatively small number of predictors

24

An Example Illustrating Use of a Shrinkage Method

We are given 10 (x,y) pair of points to fit a regression model

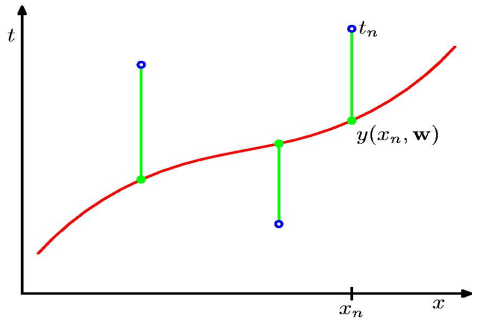


Model

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

25

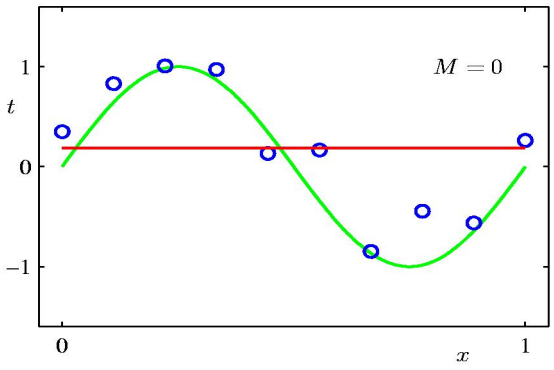
Sum-of-Squares Error Function



$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

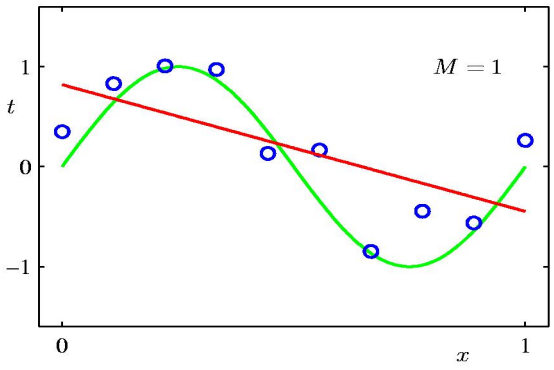
26

0th Order Polynomial



27

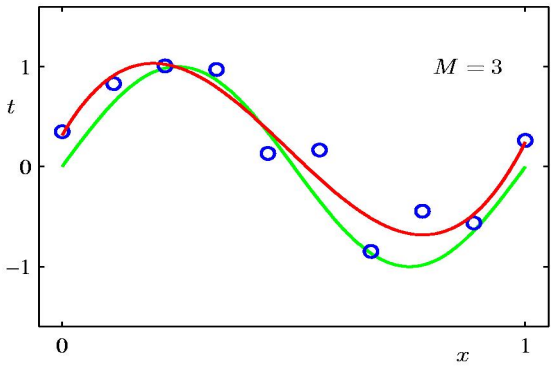
1st Order Polynomial



28

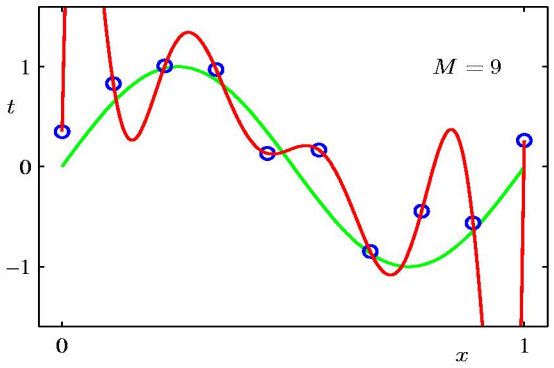
3rd Order Polynomial

Looks like a good fit



29

9th Order Polynomial



Zero error fit. Is this the best model for this problem?

30

Polynomial Coefficients

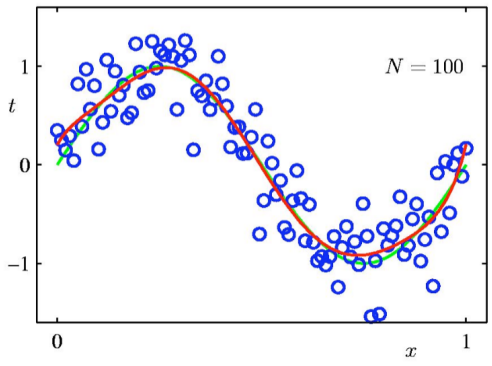
	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

These are the values of the fitted coefficients. Note extremely high values for the 9-th order model

31

Data Set Size: $N = 100$

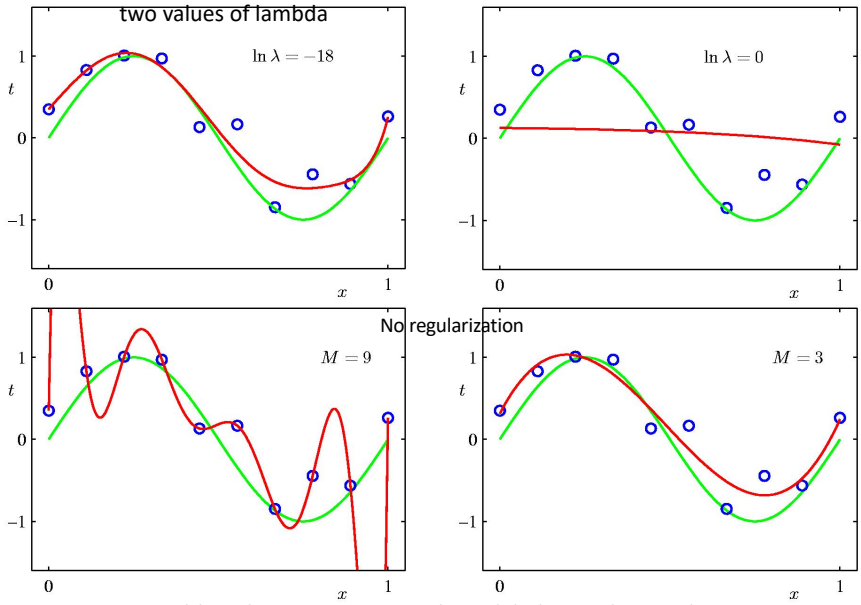
9th Order Polynomial



Overfitting reduces with 100 pairs of (x,y) values

32

9-th order model fitting with regularization/ridge regression for two values of lambda



Although we are using a 9-th model, the result is similar to 3-rd order model without shrinkage

33

Polynomial Coefficients

Shrinkage effect on coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

34

Logistic Regression

- Lets consider the following model used in linear regression. But in this case the dependent/outcome variable (y) is a discrete variable (*good/bad customer*)

  $y = a_0 + a_1 x_1 + a_2 x_2 + \ldots\ldots\ldots\ldots\ldots + a_k x_k$

- Lets look at the following expression

$$p = \frac{e^{(a_0 + a_1 x_1 \ldots + a_k x_k)}}{1 + e^{(a_0 + a_1 x_1 \ldots + a_k x_k)}}$$

- The quantity *p* will always lie in the range 0-1 and thus can be interpreted as the probability of outcome *y* being good or bad.

35

Logistic Regression

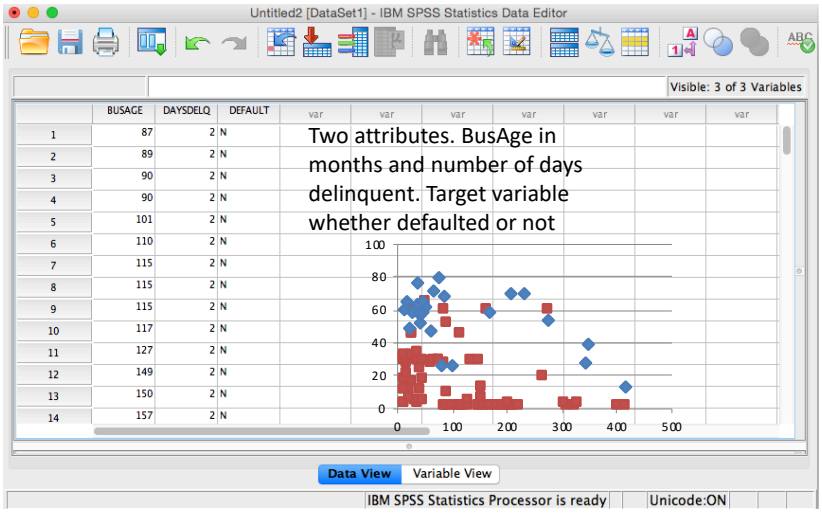
- By simple rewriting, we get:

  $log(p/(1-p)) = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_k x_k$

- This ratio is called *log odds*
- The parameters of the logistic model are determined using the likelihood maximization based cost function optimized via gradient search

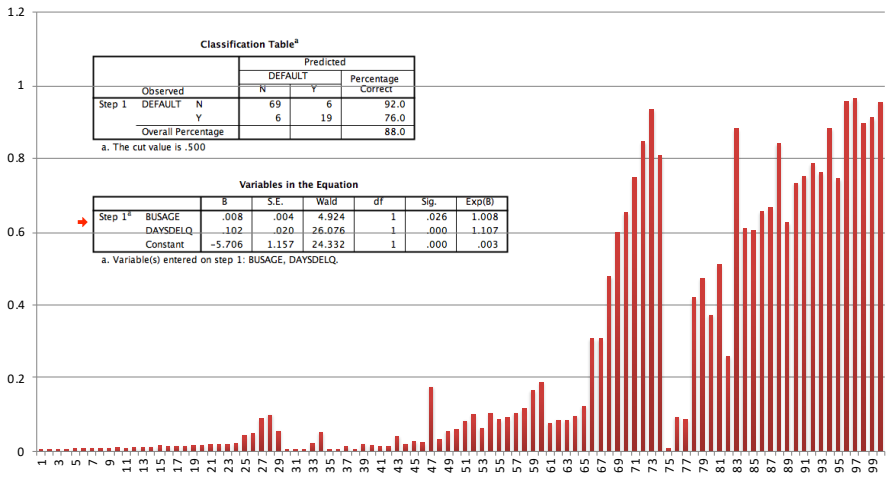
36

Credit Default Example



37

Credit Scoring Example



38

Using the Model

- What is the probability of a business defaulting given that business has been with the bank for 26 months and is delinquent for 58 days?

BUSAGE: 0.008; DAYSDELQ: 0.102; Intercept: -5.076

Plug the model parameters to calculate *p*

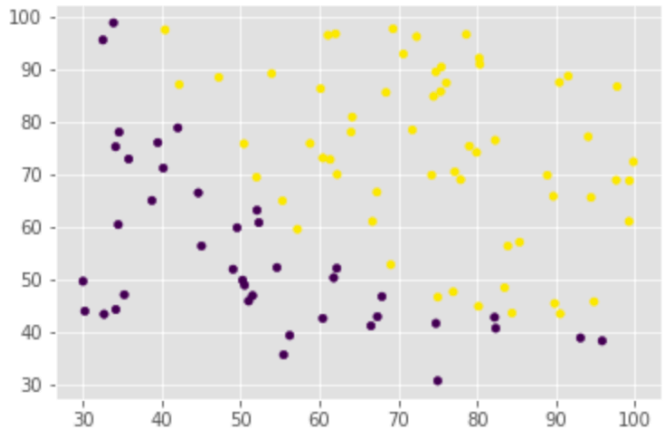
$e^{0.008*26+0.102*58-5.706}/(1+e^{0.008*26+0.102*58-5.706})$

0.603

39

Python Example using 2-D Data

```
plt.scatter(data[:,0],data[:,1],c=data[:,2],s=20)
plt.show()
```



40

```
X = data[:,0:2]
y = data[:,2]
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(C=10).fit(X, y)# C controls regularization
```

```
print(clf.score(X,y))#This gives the mean accuracy.
```

```
0.91
```

```
prob = clf.predict_proba(X)
```

```
plt.figure(figsize=(12,6))
h = np.arange(100)
plt.bar(h,prob[:,0],color='g')
plt.title("Probability Plot for Training Examples for Class '0'")
plt.show()
```

41


```
#Print weights
w = clf.coef_[0]
print(w)
```

```
[0.09432673 0.08817996]
```

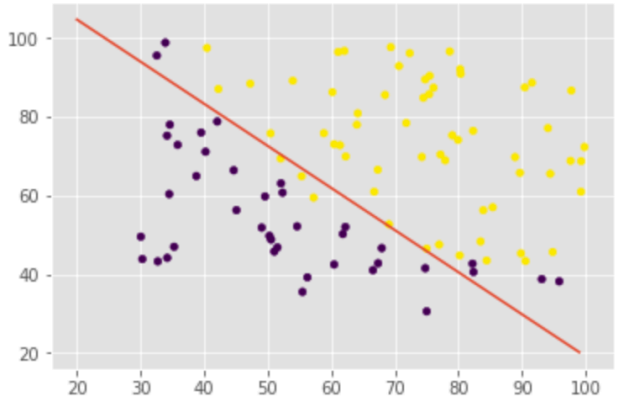
```
#Print intercept value
w0 = clf.intercept_
print(w0)
```

```
[-11.12335114]
```

42

```
xp = np.arange(100)
yp = -w[0]*xp/w[1]-w0/w[1]
```

```
plt.plot(xp[20:100],yp[20:100])
plt.scatter(data[:,0],data[:,1],c=data[:,2],s=20)
plt.show()
```



43

Example

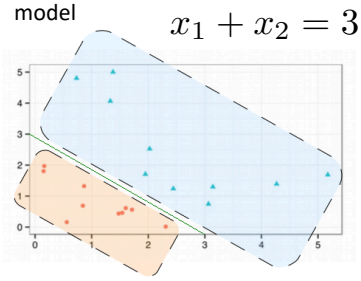
$$\mathbf{w} = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

Coefficients in a logistic regression model

What will be the prediction, 0 or 1, of the logistic regression model for an input $[2\ 5]^T$?

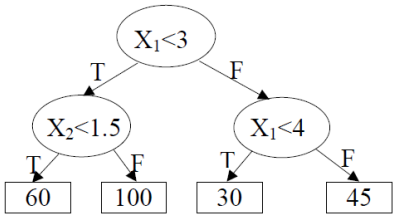
How does the decision boundary look like in x_1-x_2 space?

$$x_1 + x_2 = 3$$

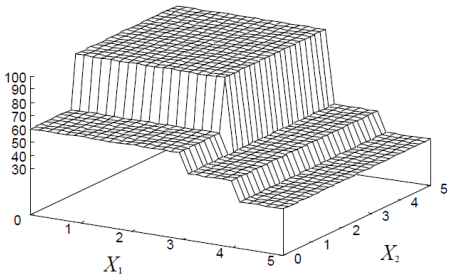


44

Regression Tree

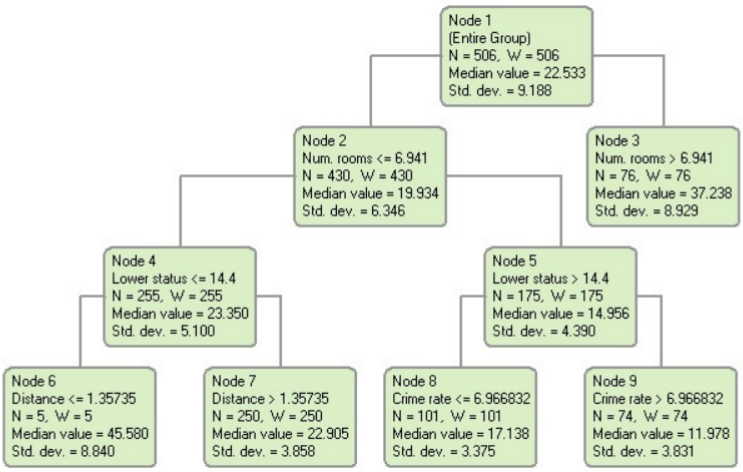


This tree roughly corresponds to the following regression surface (assuming that there were only the predictor variables X_1 and X_2) :



45

Regression Tree: Another Example



46

Regression Tree

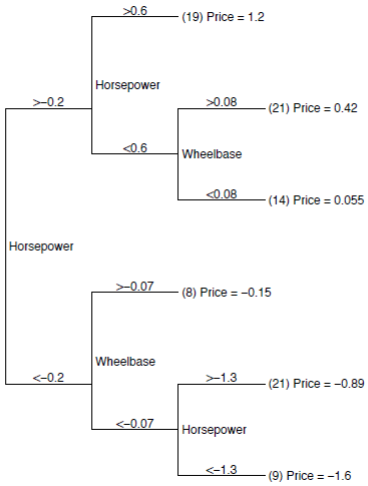


Figure 1: Regression tree for predicting price of 1993-model cars. All features have been standardized to have zero mean and unit variance. Note that the order in which variables are examined depends on the answers to previous questions. The numbers in parentheses at the leaves indicate how many cases (data points) belong to each leaf.

47

Regression Tree

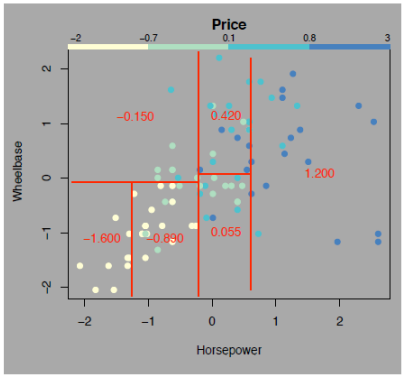


Figure 2: The partition of the data implied by the regression tree from Figure 1. Notice that all the dividing lines are parallel to the axes, because each internal node checks whether a single variable is above or below a given value.

48

Can we use knn approach for regression?

49