Regression

- In *regression* dependent variables is modeled as a function of some set of *explanatory* variables
- *linear regression*: the underlying model is assumed to be linear and can be expressed as matrix plus an error term

$$\mathbf{y} = \mathbf{X}h + \epsilon \tag{1}$$

$$\hat{h} = \left(X^T X\right)^{-1} X^T y \tag{2}$$

Ordinary Least Squares

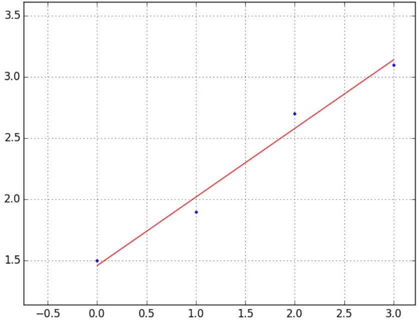
- One explanatory variable and one dependent variable the problem becomes easy to visualize
- We are given pairs of values (x, y):
  $(0, 1.5), (1, 1.9), (2, 2.7), (3, 3.1)$

$$y = Ah \tag{3}$$

$$\begin{bmatrix} 1.5 \\ 1.9 \\ 2.7 \\ 3.1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} \tag{4}$$

Sample Problem

- linear fit $y = h_0 + h_1 x$



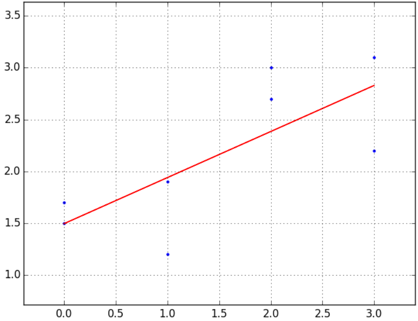
Ordinary Least Squares

- In general we may have multiple observations for a given value of the dependent variable

$$\begin{bmatrix} 1.5 \\ 1.9 \\ 2.7 \\ 3.1 \\ 1.7 \\ 1.2 \\ 3.0 \\ 2.2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} \tag{5}$$

Sample Problem

- linear fit $y = h_0 + h_1 x$



Ordinary Least Squares

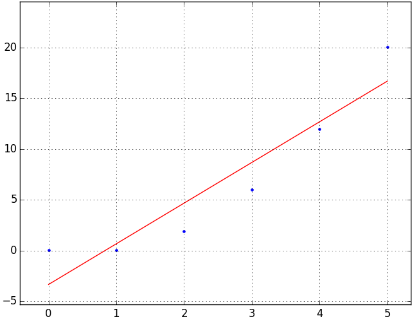
- Our data might have a more complicated relationship on the explanatory variables

$$y = Ah \tag{6}$$

$$\begin{bmatrix} 0.0319 \\ 0.0313 \\ 1.9135 \\ 5.9970 \\ 11.9835 \\ 20.0628 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} \tag{7}$$

Sample Problem

- linear fit $y = h_0 + h_1 x$



Ordinary Least Squares

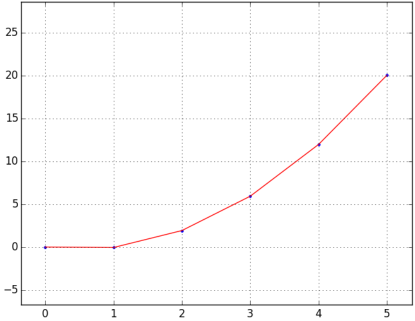
- Introduce a quadratic term in the design matrix

$$y = Ah \tag{8}$$

$$\begin{bmatrix} 0.0319 \\ 0.0313 \\ 1.9135 \\ 5.9970 \\ 11.9835 \\ 20.0628 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} \tag{9}$$

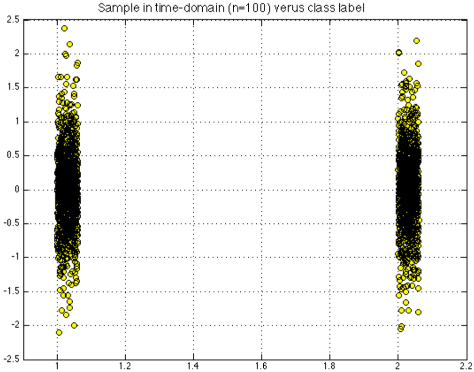
Sample Problem

- quadratic fit $y = h_0 + h_1 x + h_2 x^2$



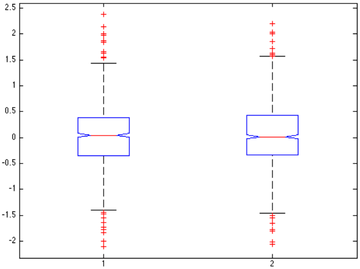
Binary Classification

- A scatter plot for a binary classification problem separates the data into to clusters
- Wilcoxon, ANOVA, Rank-sum, are ways of determining if there is a statistically significant difference between the central tendencies of the two groups



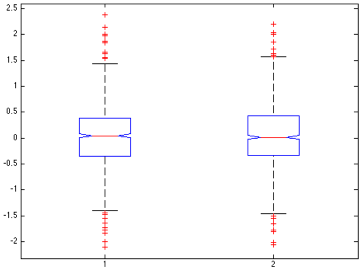
Digression on ANOVA (1)

- One-way ANOVA (*Analysis of Variance*) is a standard statistical test that can be used to measure the differences of the means of two different data sets
- In particular, a measured difference is due to random fluctuations or not
- Built into most statistics packages (SPSS etc.)
- Box-plot shown at right: same data as before



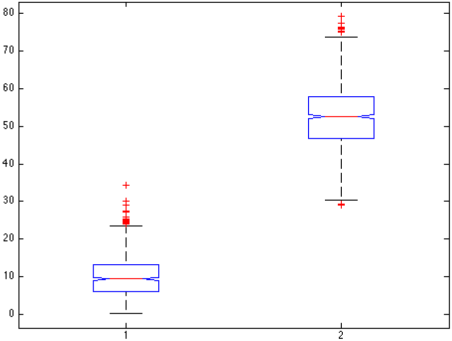
Digression on ANOVA (Box-Plots) (2)

- Box-plots provide a convenient visual summary of some data set that follows some standard conventions
- The central box extends from the first to the third quartiles (25th to 75th percentile)
- The whiskers show some idea of where the data is generally: within some number of standard deviations, within some function of the interquartile range
- The points outside of the whiskers are outliers



Digression on ANOVA (3)

- If we have a feature or variate that is highly-correlated with the class label we should see something like the plot below
- Significance of the correlation will be represented by a *p*-value
- This is the kind of data we want to present to the ML algorithm we have chosen



Slice Notation

- `np.arange` creates a sequence of incremental values
- Code above runs trials of our estimation task for different sample sizes
- *Slice notation*: `[start:end:step]`
- Colon generally means for all, works same as in matlab

Useful Python Linear Algebra Commands

```
// generate numbers according to normal dist
np.random.normal(size=(N))
// generate rand num with uniform dist
np.random.uniform(size=(N))
// matrix of zeros
np.zeros((N,2))
// element wise mult.
np.multiply
// form numpy array
S = np.array([[1/6.0, 0],[0, 1/3.0]])
// matrix multiplication
B = np.dot(A, np.dot(S,G))
// least squares
h = np.linalg.lstsq(X,y)[0]
// generate grids
xx,yy = np.meshgrid(np.linspace(-6,6,100),np.linspace(-6,6,100))
// concatenation
np.hstack((a,b))
np.vstack((a,b))
np.newaxis
// x[:,1,np.newaxis]
```

- Some useful commands for performing linear algebra calculations

Working with the Digits Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats

from sklearn import datasets

digits = datasets.load_digits()

n_samples = len(digits.images)
print(digits.images.shape)
data = digits.images.reshape((n_samples, -1))
print(data.shape)

X = data[digits.target == 2, :]
Y = data[digits.target == 8, :]
```

- Importing packages and selecting a subset of the data

Working with the Digits Dataset

```
feature1 = data[:, 35]
feature2 = data[:, 37]

R = np.corrcoef(data, rowvar=0)

e = np.random.rand(n_samples)
print((feature1 + e).shape)

r = scipy.stats.pearsonr(feature1, feature2)
print(r)
```

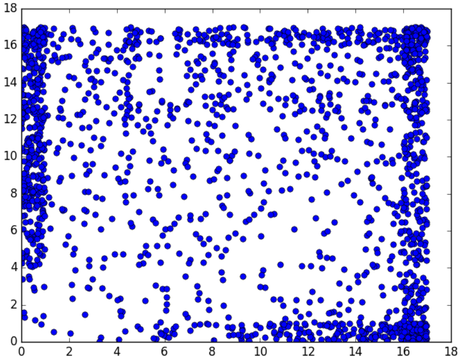
- Measuring correlation with libraries

Working with the Digits Dataset

```
plt.figure(1)
plt.plot(feature1 + e, feature2 + np.random.rand(n_samples), 'bo')

plt.figure(2)
plt.plot(X[:, 35], X[:, 36], 'bo')
plt.plot(Y[:, 35], Y[:, 36], 'ro')

plt.figure(3)
plt.imshow(R, interpolation="nearest")

plt.show()
```

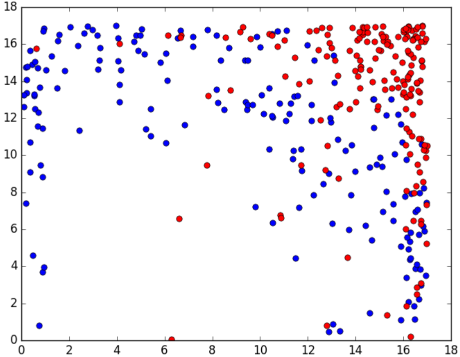
- Matlab style plotting

Correlation of Two Pixel Values



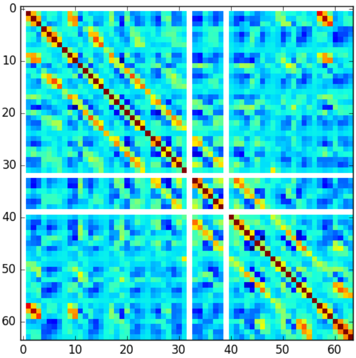
- Testing the correlation of two pixel values
- (-0.23774427860140765, 1.6459700981727808e-24)

Using pixel values as features directly



- Pixel values themselves do not separate the classes particularly well

Visualizing the Correlation or Covariance



- Rendering the Correlation as an image immediately reveals which variables are related