

# DATA X

---

**Prediction:** Data, Signals, and Systems

Berkeley SCET

Ikhlaq Sidhu  
Chief Scientist & Founding Director,  
Sutardja Center for Entrepreneurship & Technology  
IEOR Emerging Area Professor Award, UC Berkeley



# Introduction to Prediction

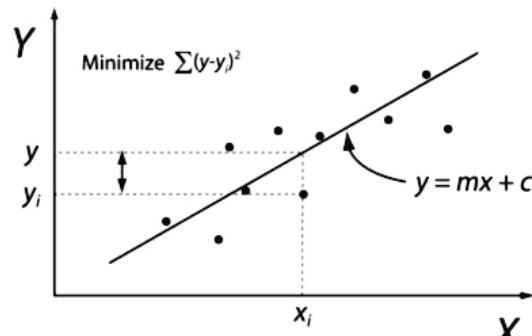
Berkeley SCET

# Prediction

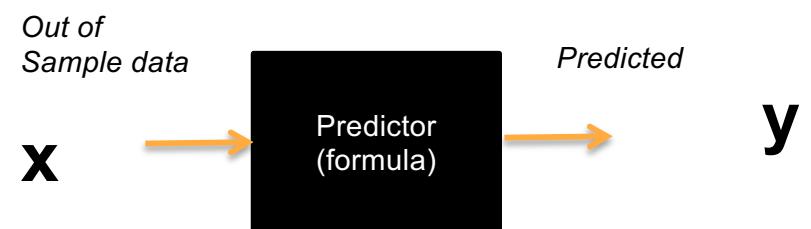
Data We Might Have  
(In Sample)

| X  | Y  |
|----|----|
| 2  | 3  |
| 5  | 9  |
| 6  | 11 |
| 8  | ?  |
| 10 | ?  |
| ?  | ?  |

Data View



Math View



**Our Goal:** Working with  
*out of sample data*

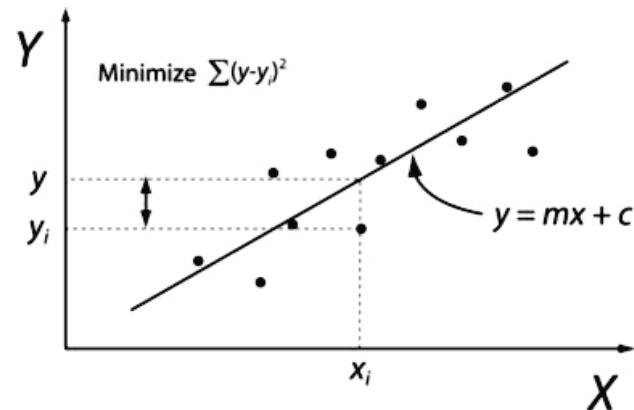
Systems View

# Prediction

Data We Might Have  
(In Sample)

| X | Y |
|---|---|
| 2 | 3 |
| 5 | 9 |
| 6 | ? |
| 8 | ? |
| ? | ? |

Data View

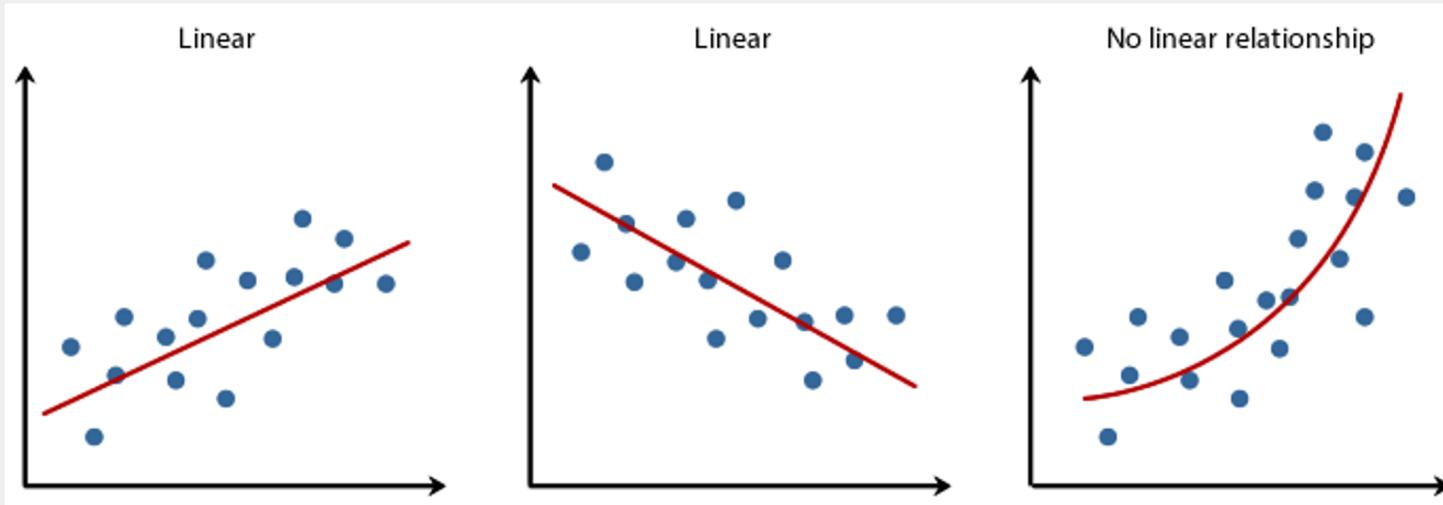


**One way to make a prediction:**

Choose a line that best fits the sample data

Then  $y(x) = mx + c$  is a predictor for a new out of sample  $x$

*Of Course, we can not assume that all data can be predicted by a **linear model***



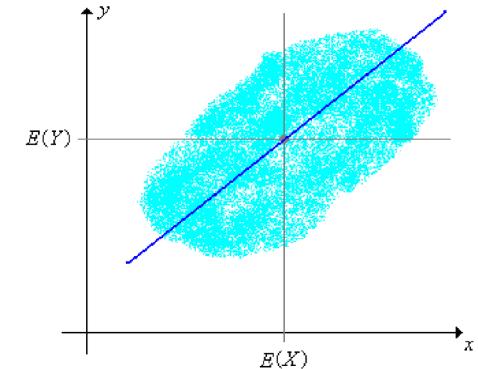
- Model might be a **poor fit** (wrong model)
- Model might be too good of a fit or **over-fit** (only works well on the in sample data)

# Best Linear Predictor

(if you just have 2 variables)

This turns out to be  
the best linear predictor:

$$L(Y|X) = E(Y) + \frac{Cov(X,Y)}{Var(X)} [X - E(X)]$$



The distribution regression line

It's a line:

**runs through point:**  $(E[X], E[Y])$

**slope:**  $m = \frac{Cov(X,Y)}{Var(X)}$

**y-intercept** =  $E[Y] - mE[X]$   
=  $E[Y] - Cov(X, Y) * \frac{E[X]}{Var(X)}$

Remember:

$$\text{Cov}(X,Y) = E[ (X-\mu_x)(Y-\mu_y) ]$$

$$\text{Cov}(X,Y) = E [XY] - E[X] E[Y]$$

$$\text{Cov}(X,X) = \text{Var}(X)$$

$$\text{Cov}(AX,Y) = A\text{Cov}(X,Y)$$

*What makes this the best linear predictor?  
How much error does this predictor have?*

Berkeley SCET

## Simple Example:

Calculate best linear predictor

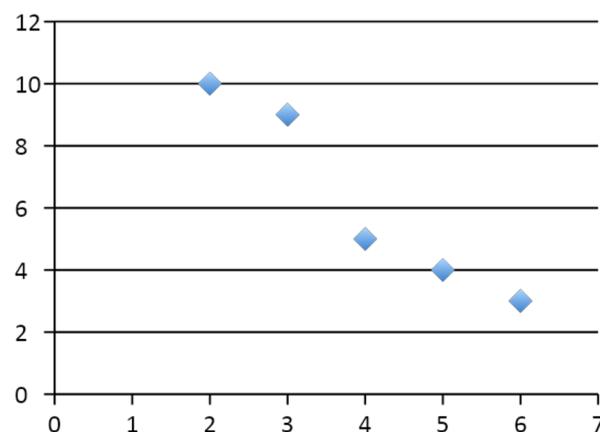
Data Set in a Table, two variables

| X | Y  |
|---|----|
| 2 | 10 |
| 4 | 5  |
| 3 | 9  |
| 5 | 4  |
| 6 | 3  |

# Example:

## Data Set in a Table 2 variables

| X    | Y    | X*Y   | X^2    | y(x) |
|------|------|-------|--------|------|
| 2    | 10   | 20    | 4      | 6.96 |
| 4    | 5    | 20    | 16     | 3.92 |
| 3    | 9    | 27    | 9      | 5.44 |
| 5    | 4    | 20    | 25     | 2.4  |
| 6    | 3    | 18    | 36     | 0.88 |
| E[X] | E[Y] | E[XY] | E[X^2] |      |
| 4    | 6.2  | 21    | 18     |      |



$$E[X] = 4, \quad E[Y] = 6.2$$

$$\begin{aligned} Cov(X, Y) &= E[XY] - E[X]E[Y] \\ &= 21 - 4 * 6.2 = -3.8 \end{aligned}$$

$$Var(X) = 18 - 16 = 2$$

$$\begin{aligned} y\text{-int} &= E[Y] - \frac{Cov(X,Y)}{Var(X)} * E[X] \\ &= 6.2 - \left( \frac{-3.8}{2} * 4 \right) = 13.8 \end{aligned}$$

$$m = \frac{Cov(X, Y)}{Var(X)} = \frac{-3.8}{2} = -1.9$$

$$y(x) = -1.9x + 13.8$$

$$E[y(x) - y_{actual})^2] = ?$$

# Code Sample

```
import numpy as np

x = np.array([2, 4, 3, 5, 6])
y = np.array([10, 5, 9, 4, 3])

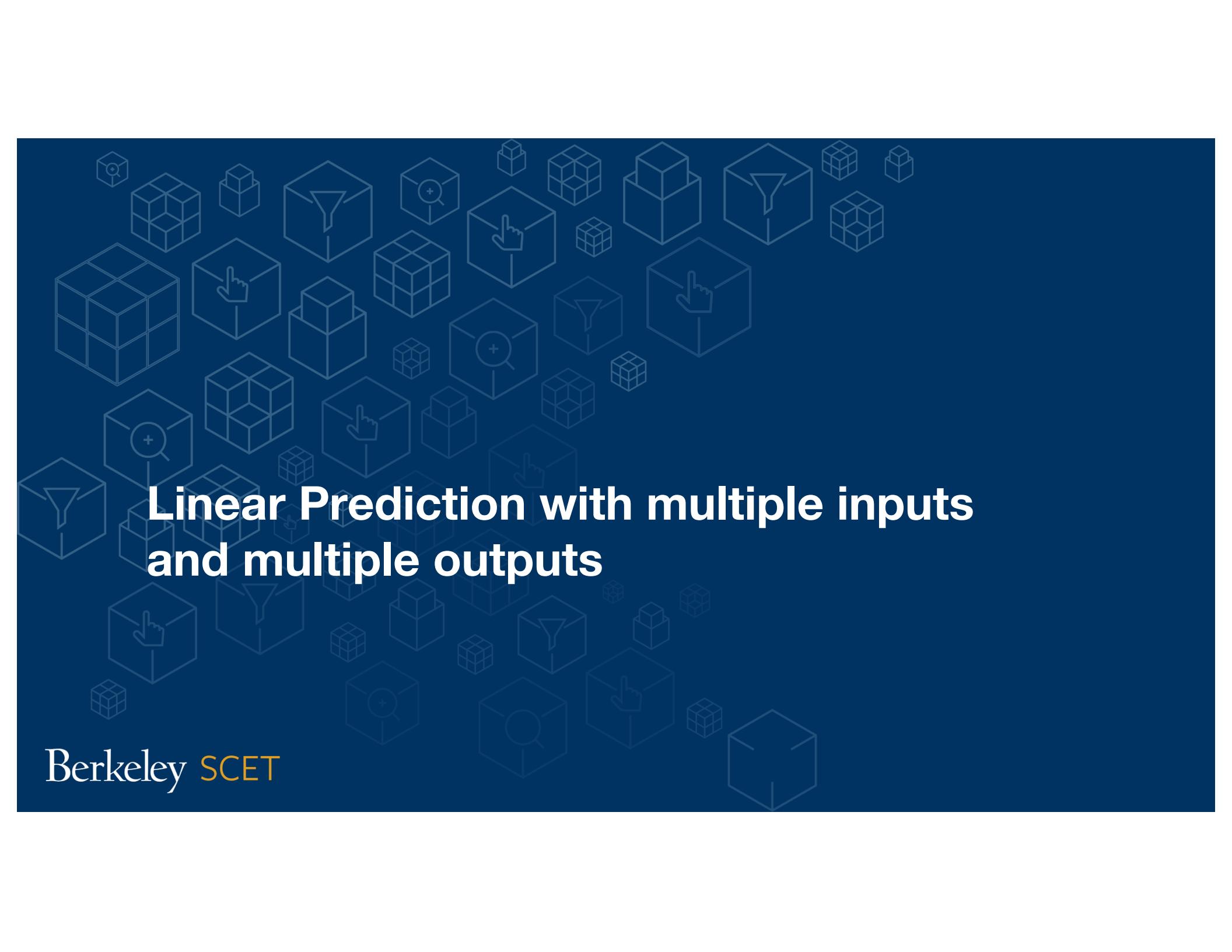
E_x = np.mean(x)
E_y = np.mean(y)

cov_xy = np.mean(x*y)-E_x*E_y

y_0 = E_y - cov_xy/np.var(x)*E_x
m = cov_xy/np.var(x)

y_pred=m*x+y_0

print "E[(y_pred-y_actual)^2] =", np.mean(np.square(y_pred-y))
E[(y_pred-y_actual)^2] = 0.54
```



# Linear Prediction with multiple inputs and multiple outputs

Berkeley SCET

# Prediction: Multiple Inputs and Outputs



No problem, we use multiple predictors.

$$\mathbf{y}_1 = g_1(\mathbf{x}) \quad \mathbf{y}_2 = g_2(\mathbf{x}) \quad \mathbf{y}_3 = g_3(\mathbf{x})$$



# Prediction: Multiple Inputs and Outputs

Out of Sample  
 $X = [x_1 \ x_2 \ \dots \ x_n]$  → Predictor (formula)

→ Predicted  
 $y$

In this case, for linear prediction, we use a matrix format:

$$\begin{bmatrix} X \\ (N \times d) \\ 1, 3, 4, 6, 2 \end{bmatrix} \cdot \begin{bmatrix} W \\ (d \times 1) \end{bmatrix} = \begin{bmatrix} Y \\ (N \times 1) \end{bmatrix}$$

$X_i$

$$\begin{aligned} x_{1,1}w_1 + x_{1,2}w_2 + x_{1,3}w_3 &= y_1 \\ x_{2,1}w_1 + x_{2,2}w_2 + x_{2,3}w_3 &= y_2 \\ \dots \\ \dots \end{aligned}$$

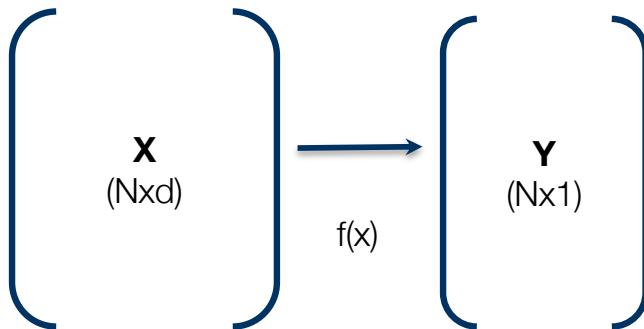
$X$  is the in-sample data. We only need to figure out  $W$ .

With  $W$ , we can estimate  $y$  for new  $x$ , i.e. out of sample data

# An ML Framework

We don't know:  
 $P(X)$ , the distribution of X  
Function  $f$ :  
 $f: X \rightarrow Y$

Berkeley SCET



Features: (d columns), age, income, zip code, ...

| Sex    | Age   | Marital ... | Occupation        | Job Time | Checking | Savings | GoodBadMark |
|--------|-------|-------------|-------------------|----------|----------|---------|-------------|
| female | 27.17 | Married     | Semi-professional | 0        | No       | Yes     | Good        |
| male   | 25.92 | Married     | Blue Collar       | 0.375    | No       | Yes     | Good        |
| male   | 23.08 | Married     | Blue Collar       | 1        | No       | Yes     | Good        |
| male   | 39.58 | Married     | Semi-professional | 0        | No       | Yes     | Good        |
| male   | 30.58 | Single      | Blue Collar       | 0.125    | No       | No      | Good        |
| male   | 17.25 | Married     | Blue Collar       | 0.04     | No       | No      | Good        |
| female | 17.67 | Single      | Semi-professional | 0        | No       | No      | Bad         |
| male   | 16.5  | Married     | Blue Collar       | 0.165    | No       | No      | Good        |
| female | 27.33 | Married     | Semi-professional | 0        | No       | No      | Good        |
| male   | 31.25 | Married     | Semi-professional | 0        | No       | Yes     | Good        |
| male   | 20    | Married     | Blue Collar       | 0.5      | No       | No      | Bad         |
| male   | 39.5  | Married     | Blue Collar       | 3.5      | No       | No      | Good        |
| male   | 36.5  | Married     | Blue Collar       | 3.75     | No       | No      | Good        |
| male   | 52.42 | Married     | Blue Collar       | 3.75     | No       | No      | Good        |

In Sample

Out of Sample

## In Sample Data:

which we have

- d features
- N samples

Use for training

**Results in Y** (which we know for N examples)

Sometimes measured Y includes error or noise

$$Y = f(x) + e()$$

## N rows:

- each row has customer information
- d features
- **Y has a value of interest**
  - *Credit score*: a number
  - *Classification*: “good customer” vs “poor customer”

# Example:

## Prediction with Regression

Data:

| ID | Name  | x(i,1)<br>Age | x(i,2)<br>years<br>w employer | x(i,3)<br>Income | y(i,1)<br>Credit Score |
|----|-------|---------------|-------------------------------|------------------|------------------------|
| 1  | John  | 25            | 3                             | 50               | 660                    |
| 2  | Alice | 23            | 2                             | 60               | 580                    |
| 3  | Bill  | 28            | 1                             | 80               | 425                    |
| 4  | Rahul | 25            | 5                             | 59               | 320                    |

$$\begin{bmatrix} \mathbf{x} \\ (\mathbf{N} \times d+1) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{w} \\ w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} \mathbf{Y} \\ \mathbf{N} \times 1 \\ 660 \\ 580 \\ 425 \\ 320 \end{bmatrix}$$

We are now going to **choose a W that gives us a predictor for Y**

This time, Y is the actual value we want to estimate

**Notice: we added an extra column of 1s?**

# Example:

## Prediction with Regression

$$\mathbf{X}_2 \cdot \begin{bmatrix} \mathbf{w} \\ w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} \mathbf{Y} \\ \mathbf{N} \times 1 \\ 660 \\ 580 \\ 425 \\ 320 \end{bmatrix}$$

This time,  $\mathbf{Y}$  is the actual value we want to estimate.?

Train with this to “calculate”  $\mathbf{w}$



Then predict (or classify) with  $\mathbf{w}$

# The Math:

## Linear Regression

**Predictor:**  
 $g(x) = Xw$

OK, but better  
to measure  
squared error

$$E_{in}(w) = E[Xw - Y]$$

$$E_{in}(w) = \frac{1}{N} \sum_{i=1}^N (w^T x_i - y_i)^2$$

# The Math:

## Linear Regression

Berkeley SCET

$$E_{in}(w) = \frac{1}{N} \|Xw - y\|^2$$

$$\nabla E_{in}(w) = \frac{2}{N} X^T (Xw - y) = 0.$$

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

$$X^T X W = X^T Y$$

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

$$Y_{estimated} = X_{out\ of\ sample} W$$

$N$  = # of  $X$  data rows

$d$  = # of features

$m$  = # of outputs in  $Y$

$$X^T = d \times N$$

$$X = N \times d$$

$$d \times d$$

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

$$d \times d$$

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

$$= d \times d \cdot d \times N$$

$$= [d \times N] \times [N \times m] = d \times m$$

$$Y = N \times m \text{ outputs}$$

$$Y_{estimated} = X W =$$

$$X_{out} = N_{out} \times d$$

$$W = d \times m$$

$$N_{out} \times m$$

# Example:

## Prediction with Regression (to be continued)

$$X = \begin{Bmatrix} X \\ (N \times d+1) \\ 1 & 25 & 3 & 50 \\ 1 & 23 & 2 & 60 \\ 1 & 28 & 1 & 80 \\ 1 & 25 & .5 & 59 \end{Bmatrix} \cdot \begin{Bmatrix} w \\ w_0 \\ w_1 \\ w_2 \\ w_3 \end{Bmatrix} = \begin{Bmatrix} Y \\ Nx1 \\ 660 \\ 580 \\ 425 \\ 320 \end{Bmatrix}$$

This time, **Y** is the actual value we want to estimate.?

Train with this to “calculate”

**w**

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

Out of  
Sample data  
(a row of numbers)  
**Xi**

**Predictor:**  
 $g(x) = XW$

**Estimated y**  
(a real number)

Then predict (or classify) with **W**

# Code Example

```
import numpy as np

x = np.array([
    [1,25,3,50],
    [1,23,2,60],
    [1,28,1,80],
    [1,25,0.5,59]
])
y = np.array([660,580,425,320])

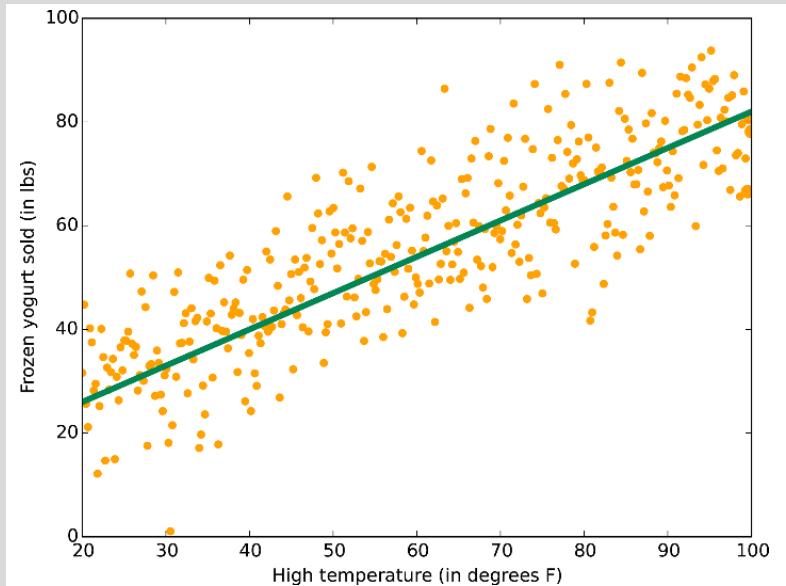
print "W = ", np.linalg.inv(x.T.dot(x)).dot(x.T).dot(y)

W =  [ 426.17283951 -16.04938272  149.44444444  3.7345679 ]
```

$$x_0w_0 + x_1w_1 + x_2w_2 + x_3w_3 = y_i$$

| Data: |       | x(i,1) | x(i,2)              | x(i,3) | y(i,1)       |
|-------|-------|--------|---------------------|--------|--------------|
| ID    | Name  | Age    | years<br>w employer | Income | Credit Score |
| 1     | John  | 1 25   | 3                   | 50     | W0 660       |
| 2     | Alice | 1 23   | 2                   | 60     | W1 580       |
| 3     | Bill  | 1 28   | 1                   | 80     | W2 425       |
| 4     | Rahul | 1 25   | 5                   | 59     | W3 320       |

# Linear Regression Illustration with Scikit Learn



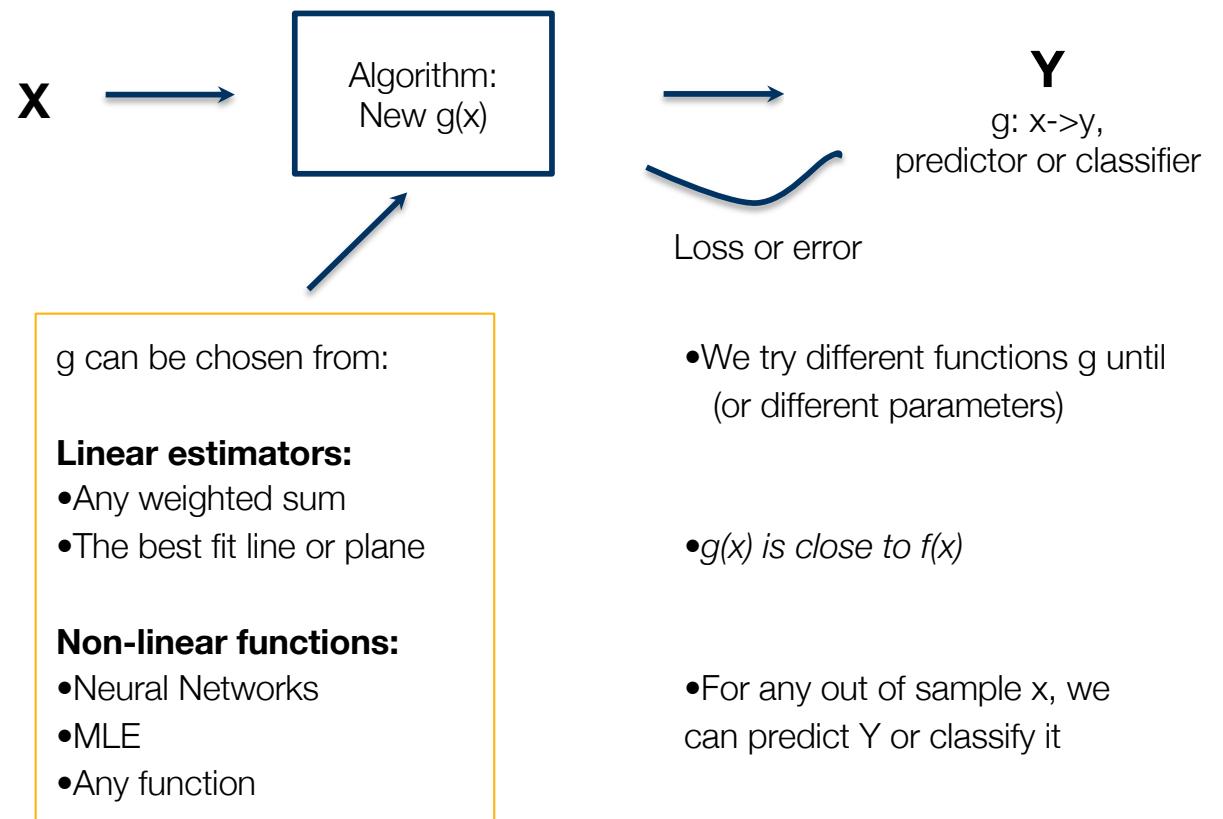
```
#Setting Linear Regression in sklearn  
from sklearn import linear_model  
  
model= linear_model.LinearRegression()  
model.fit(X_train, Y_train)  
  
Y_pred_train = model.predict(X_train)  
Y_pred_test = model.predict(X_test)  
  
# Compare Y_pred_test with Y_test for  
error.
```

Illustration Source:

<https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>

# An ML Framework

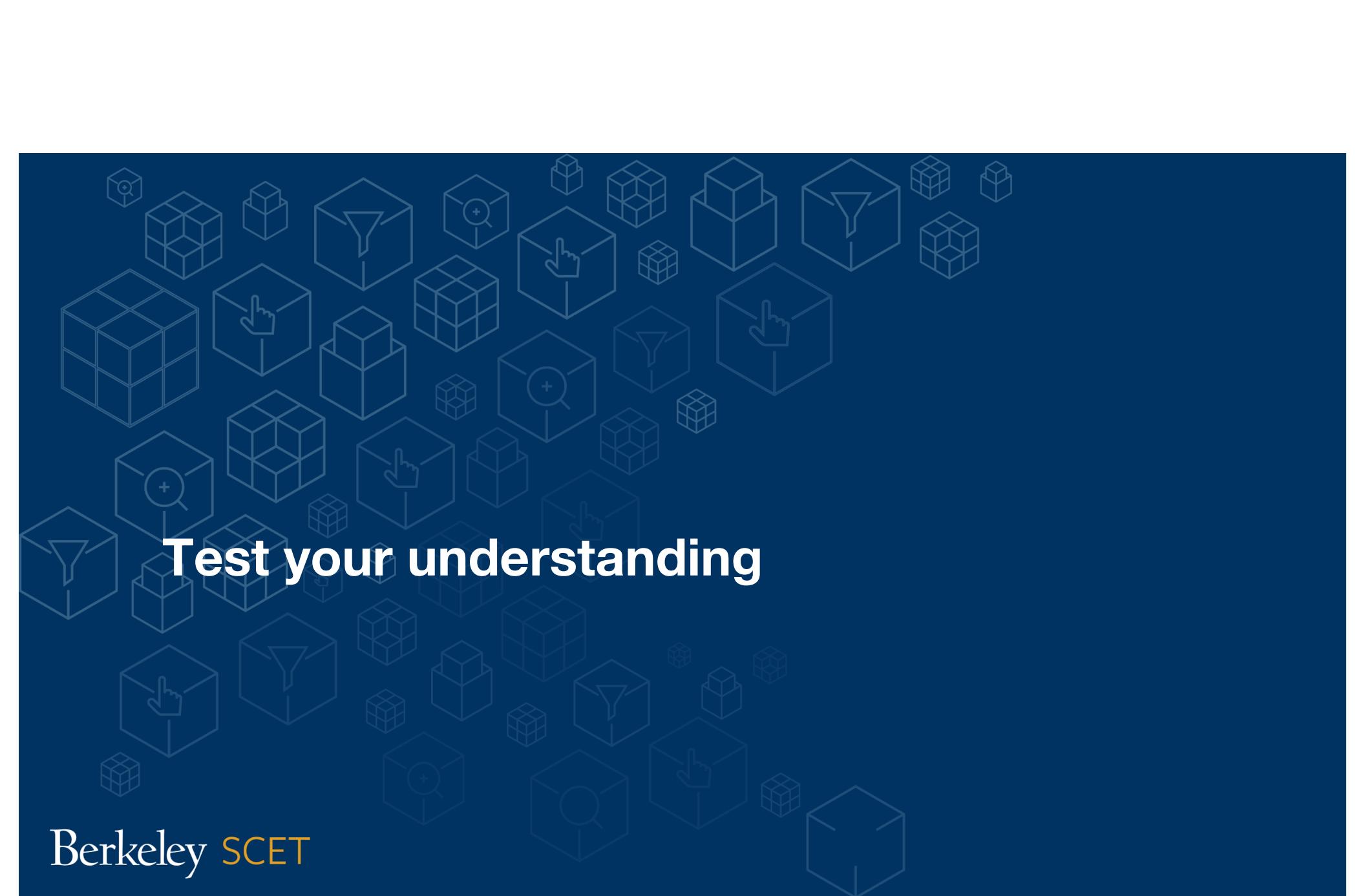
In the ML framework, there is no limit to the predictors or classifiers that can be used.





# End of Section

Berkeley SCET

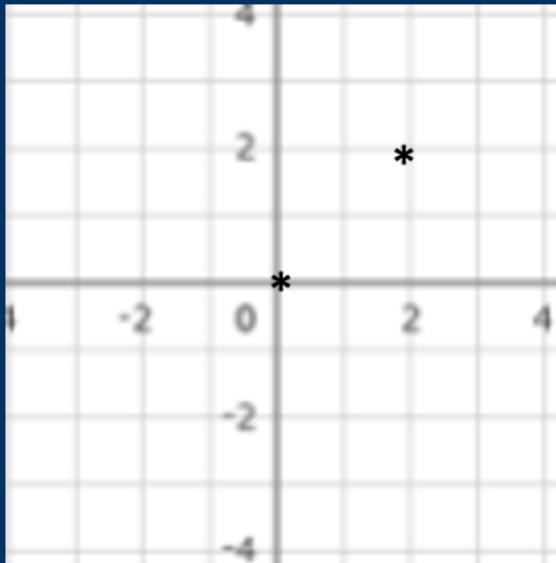


# Test your understanding

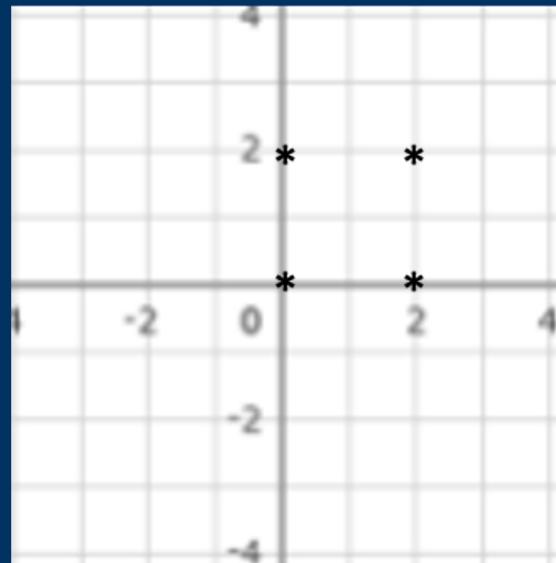
Berkeley SCET

# Covariance

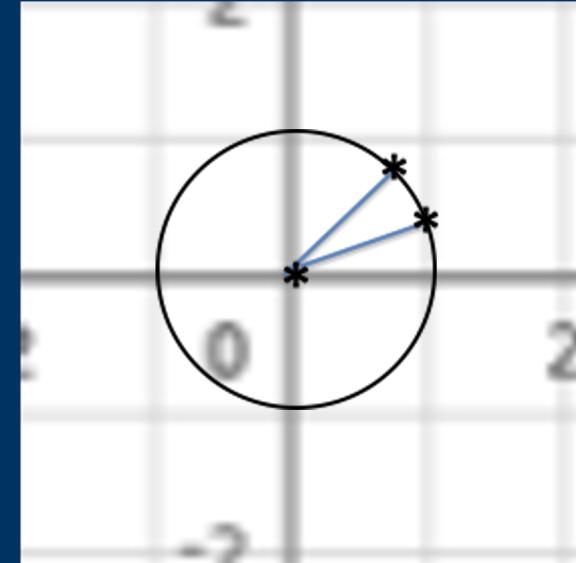
- What is the COV (X,Y) for these points:



(0,0) and (2,2)



(0,0), (2,2), (0,2), (2,0)



Points on a unit circle that are  
• 30 degrees and 45 degrees,  
and the origin  
• Or, Every point on the circle

# Multiple inputs and outputs

If  $Y_{predicted} = f(X, W) = X W$

And  $W$  is a  $4 \times 3$  matrix, then how many input features ( $d$ ) are in  $X$  and how many outputs ( $m$ ) are in  $Y$ ?

