Data X

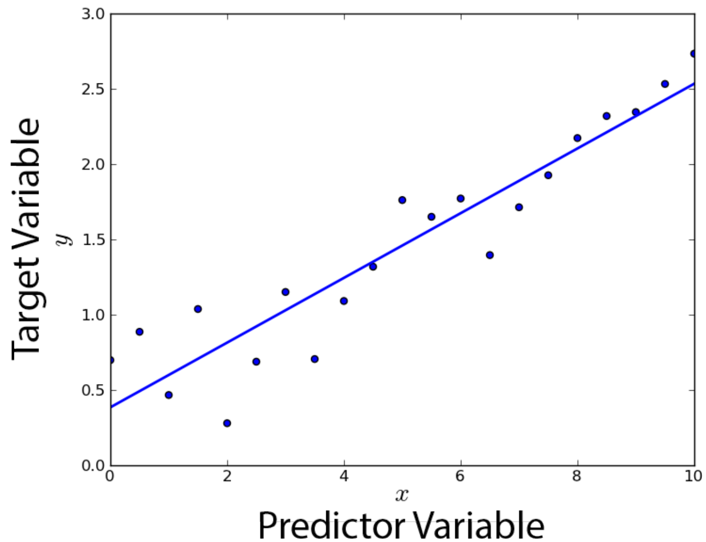
Prediction

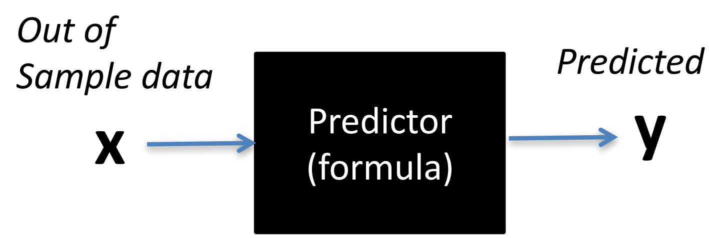
Data, Signals, and Systems

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

Introduction to Prediction

Prediction

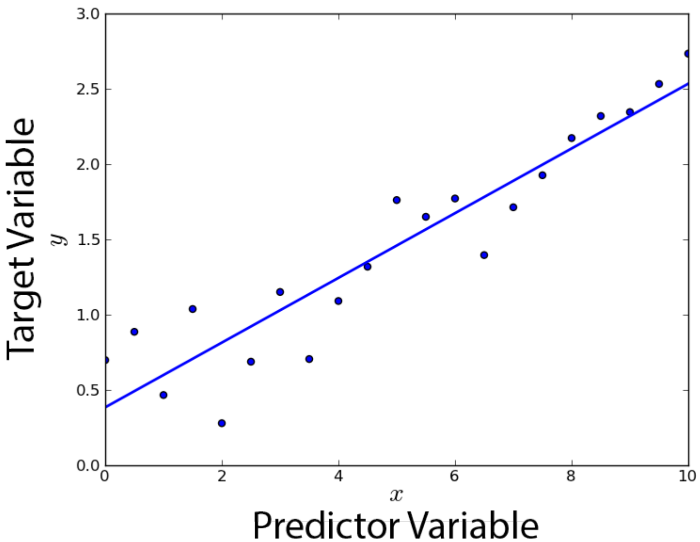




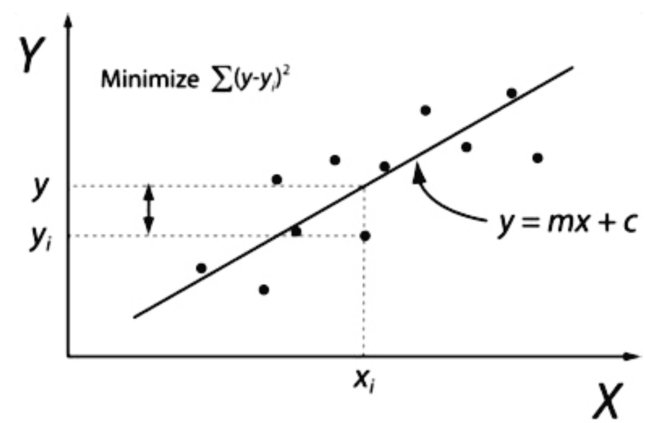
Our Goal: Working with *out of sample data*

Data We Might Have
(In Sample)

Prediction



Data We Might Have
(In Sample)

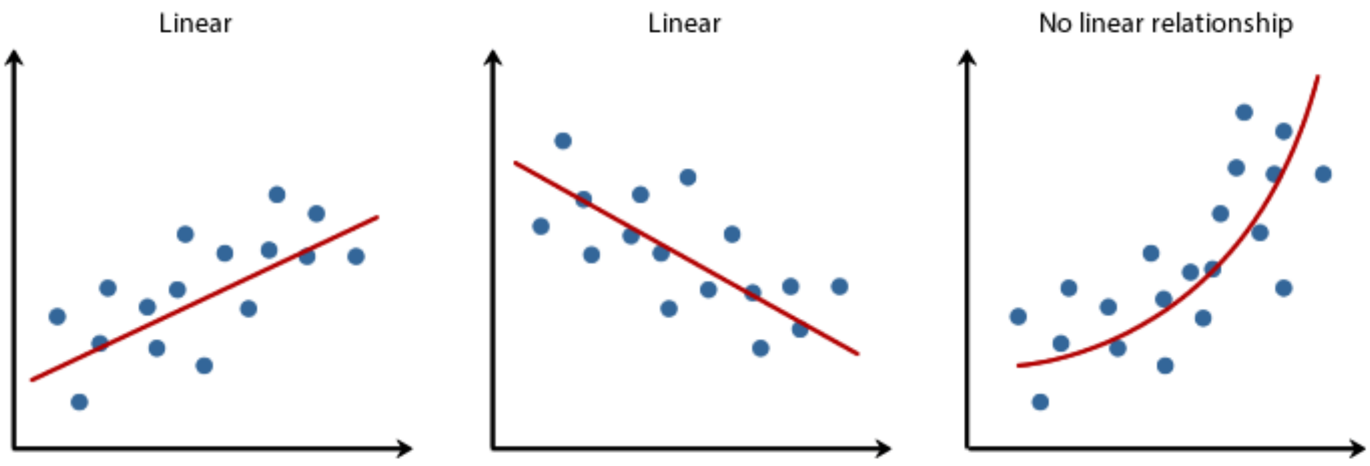


One way to make a prediction:

Choose a line that best fits the sample data

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

*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)

Image: Laerd Statistics, 2014

Best Linear Predictor (if you just have 2 Variables)

This turns out to be
the best linear predictor:

$$L(Y \mid X) = \mathbb{E}(Y) + \frac{\operatorname{cov}(X,Y)}{\operatorname{var}(X)}[X - \mathbb{E}(X)]$$

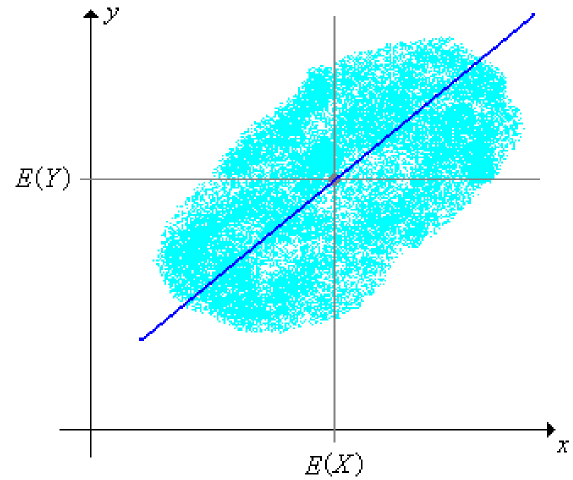
It's a line:

Runs though point: $(E[X],\ E[Y])$

slope: $m = \frac{COV(X,Y)}{VAR(X)}$

y-intercept = $E[Y] - mE[X]$

= $E[Y] - COV(X,Y) * \frac{E[X]}{VAR(X)}$



The distribution regression line

Remember:
COV(X,Y) = E[(X-μ_x) (Y-μ_y)]
COV(X,Y) = E [X Y] – E[X] E[Y]

COV(X,X) = VAR(X)
COV(AX,Y) = ACOV(X,Y)

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

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

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

$Cov(X,Y) = E[XY] - E[X]E[Y]$
$= 21 - 4 * 6.2 = -3.8$

Var(X) = 18-16 = 2

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

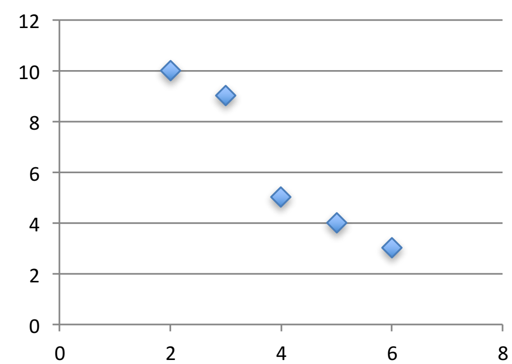
$$m = \frac{Cov(X,Y)}{Var(X)} = \frac{-3.8}{2} = -1.9$$

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

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

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

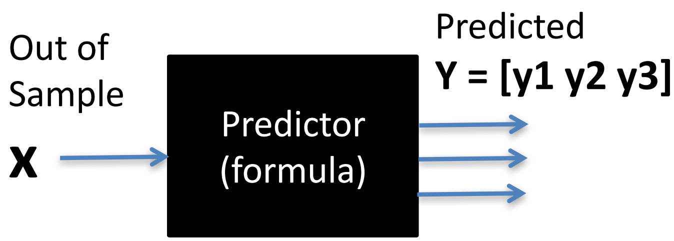


Code Sample

```
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
```

Prediction: Multiple Inputs and Outputs

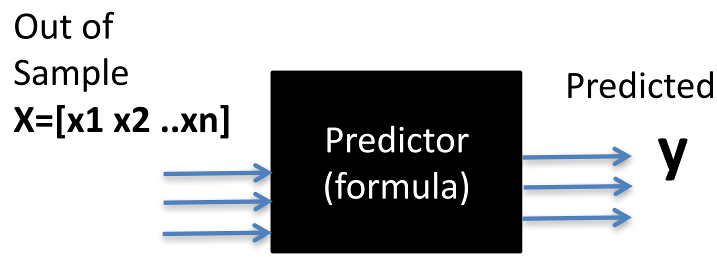


No problem,
we use multiple predictors.

y1 = g1(x)

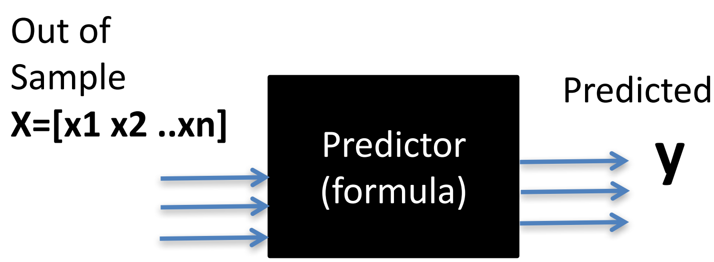
y2 = g2(x)

y3 = g3(x)

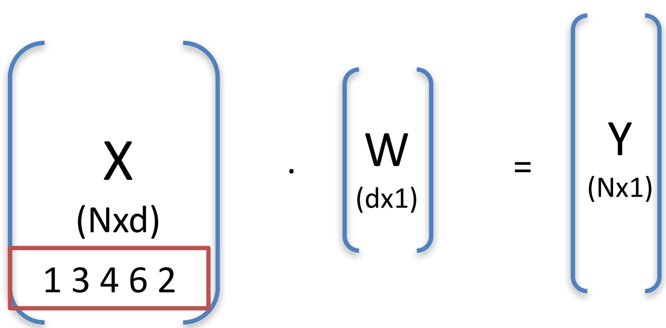


?

Prediction: Multiple Inputs and Outputs



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



$$x_{1,1}w_1 + x_{1,2}w_2 + x_{1,3}w_3 = y_1$$
$$x_{2,1}w_1 + x_{2,2}w_2 + x_{2,3}w_3 = y_2$$

…

…

$\mathbf{x}_i$

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

In Sample Data
which we have

- d features
- N samples

Use for training

X (Nxd) $\xrightarrow{f(X)}$ Y Nx1

Results in Y
(which we know for N samples)

Sometimes measured
Y includes error or noise
Y = f(X) + e()

We don't know:

P(X), the distribution of X

Function f:
f: X->Y

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

Person 1
Person 2
Person 3

Person N

Sex	Age	Marital ...	Occupation	Job Time	Checking	Savings	Good/Bad Mark
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	[illegible]	[illegible]	[illegible]	No	Bad
male	39.5	Married	[illegible]	[illegible]	[illegible]	No	Good
male	36.5	Married	Blue Collar	3.5	No	No	Good
male	52.42	Married	Blue Collar	3.75	No	No	Good

In Sample

Out of Sample

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"

An ML Framework

In Sample Data
which we have
- d features
- N samples

Use for training

X
(Nxd)

Y
(N x 1)

Results in Y
(which we know)

Sometimes measured
Y includes error or noise

We don't know:

P(X), the distribution of X

Function f:
f: X->Y

We use this to train:
(x1,y1), (x2,y2)…(xN,yN)

H = Hypothesis Set
All Possible Algorithms
Candidate Formulas

Algorithm:
Pick a g(x)

Error measure

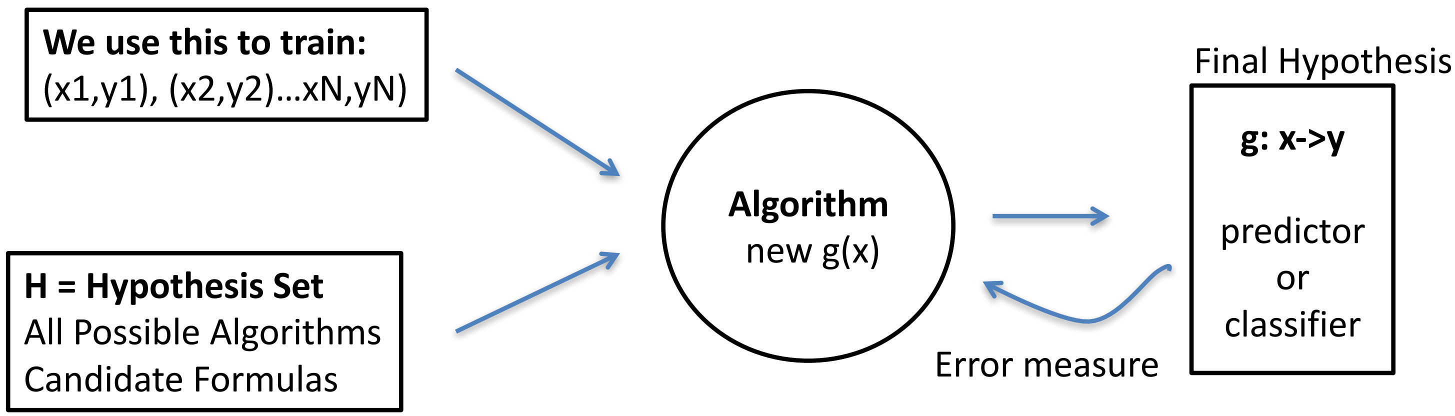
Final Hypothesis

g: x->y

predictor
or
classifier

- We try different functions g until
- *g(x) is close to f(x)*
- For any out of sample **x**, we can predict a **y** or classify it

An ML Framework



Actual Estimated

- **g(x) is our estimate of f(x)**
- **One way to measure error: Error(g) = E[f(x)-g(x)]** over all in-sample x.
  This Error(g) is a type of loss function, there are many loss functions
- Error_in_sample(g) is expected to be similar to Error_out of sample[g]

Example: Prediction with Regression

		$x(i,1)$	$x(i,2)$	$x(i,3)$	$y(i,1)$
Data:					
ID	Name	**Age**	**years w employer**	**Income**	**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

$$\underset{(N \times d+1)}{\mathbf{X}} \quad \mathbf{W} \quad \underset{N\text{x}1}{\mathbf{Y}}$$

$$\begin{pmatrix} 1\ 25\ 3\ 50 \\ 1\ 23\ 2\ 60 \\ 1\ 28\ 1\ 80 \\ 1\ 25\ .5\ 59 \end{pmatrix} \cdot \begin{bmatrix} w0 \\ w1 \\ w2 \\ w3 \end{bmatrix} = \begin{pmatrix} 660 \\ 580 \\ 425 \\ 320 \end{pmatrix}$$

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

$$\underset{(N \times d+1)}{\mathbf{X}} \qquad \mathrm{W} \qquad \underset{N\times1}{\mathbf{Y}}$$

$$\begin{array}{c} \\ \mathbf{X_2} \\ \\ \\ \end{array} \begin{pmatrix} 1 & 25 & 3 & 50 \\ \boxed{1 \;\; 23 \;\; 2 \;\; 60} \\ 1 & 28 & 1 & 80 \\ 1 & 25 & .5 & 59 \end{pmatrix} \cdot \begin{bmatrix} w0 \\ w1 \\ w2 \\ w3 \end{bmatrix} = \begin{bmatrix} 660 \\ 580 \\ 425 \\ 320 \end{bmatrix}$$

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

Train with this to “calculate” **W**

Out of sample
(a row of numbers)

$\mathbf{X_i}$

→ **Predictor:** $g(x) = XW$ →

Estimated y
(a real number)

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

The Math: Linear Regression

Predictor:
g(x) = XW

OK, but better to measure squared error

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

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

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

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

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

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

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

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

X^T = dxN

X = Nxd

(X^TX) = dxd

$(X^TX)^{-1}$ = dxd

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

= dxd · dxN · Y= Nxm outputs

= [dxN] x [Nxm] = dxm

Y estimated = X W =

Xout is Nout x d · W = dxm = Nout x m

Example 2: Prediction with Regression

$$\underset{(N \times d+1)}{\mathbf{X}} \qquad \mathbf{W} \qquad \underset{N\times 1}{\mathbf{Y}}$$

$$\mathbf{X} = \begin{pmatrix} 1 & 25 & 3 & 50 \\ 1 & 23 & 2 & 60 \\ 1 & 28 & 1 & 80 \\ 1 & 25 & .5 & 59 \end{pmatrix} \cdot \begin{bmatrix} w0 \\ w1 \\ w2 \\ w3 \end{bmatrix} = \begin{pmatrix} 660 \\ 580 \\ 425 \\ 320 \end{pmatrix}$$

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

Train with this to "calculate" **W**

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

Out of sample
(a row of numbers)

$\mathbf{X}_i$

→ **Predictor:** g(x) = XW →

Estimated y
(a real number)

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

Code Sample

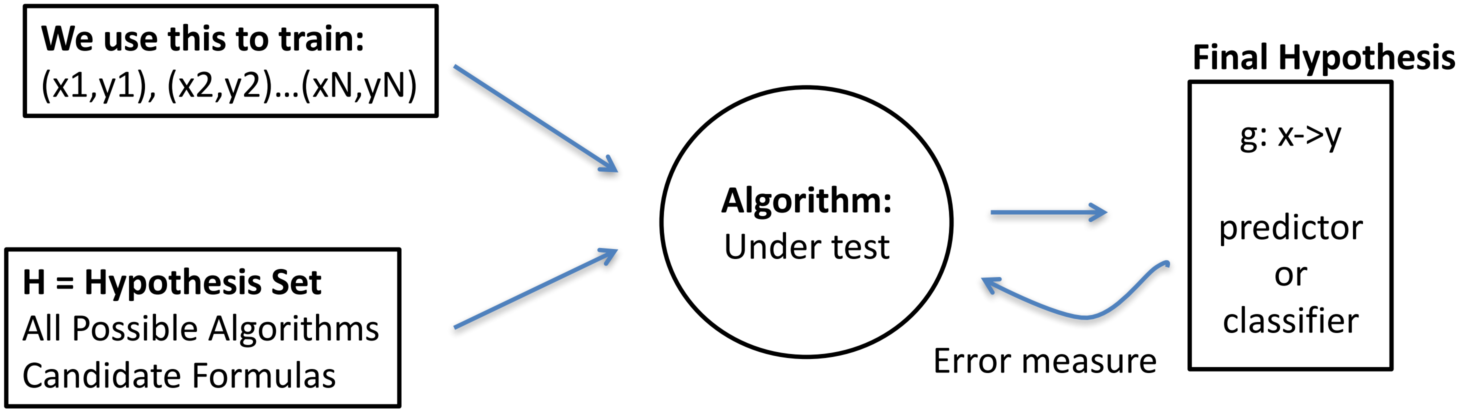
```python
import numpy as np

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 ]
```

$$x_{0,}w_0 + \; x_{1,}w_1 + x_2w_2 + x_3w_3 = y_i$$

Data:		x(i,1)	x(i,2)	x(i,3)	y(i,1)
ID	Name	**Age**	**years w employer**	**Income**	**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

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



g can be chosen from,

Linear estimators:

- Any weighted sum
- The best fit line or plane

Non-linear functions:

- Neural Networks
- MLE
- Any function

- We try different functions g until
- *g(x) is close to f(x)*
- For any out of sample **x**, we can predict **Y** or classify it

End of Section