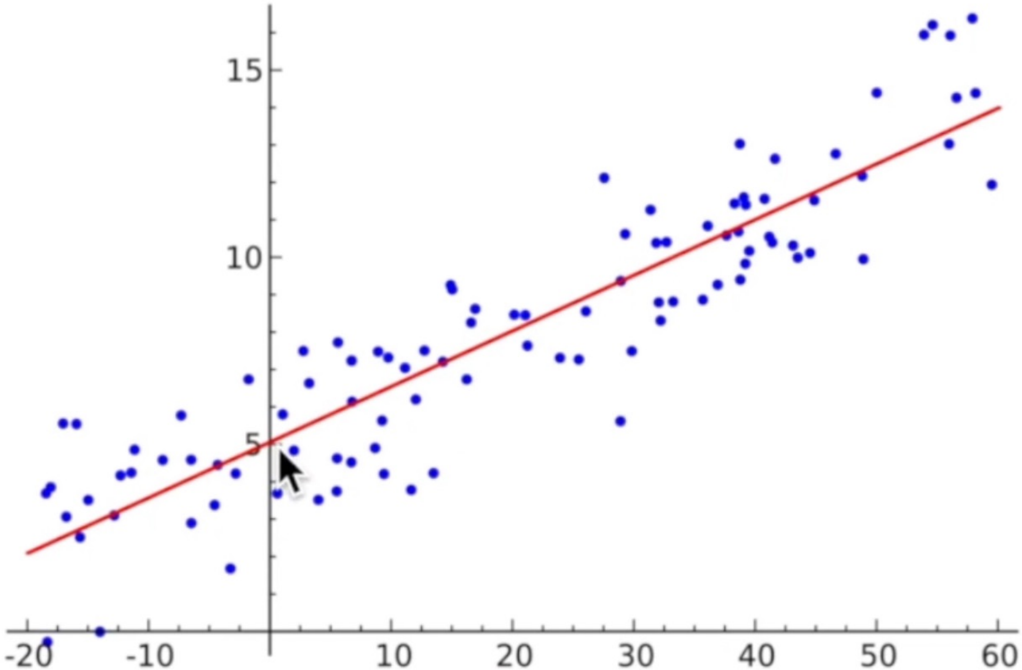
Machine Learning

- Limitations of explicit programming
  - Spam filter: many rules
  - Automatic driving: too many rules
- Machine learning: "Field of study that gives computers the ability to learn without being explicitly programmed" Arthur Samuel (1959)

Supervised/Unsupervised learning

- Supervised learning:
  - learning with labeled examples - training set

Linear Regression



$$y = ax + b$$

https://en.wikipedia.org/wiki/Linear_regression

Cost function

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx_i - y_i)^2$$

$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x_i) - y_i)^2$$

Recap

- Hypothesis $$H(x) = Wx + b$$
- Cost function $$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx_i - y_i)^2$$
- Gradient descent $$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(W(x_i) - y_i)x_i$$

Hypothesis using matrix

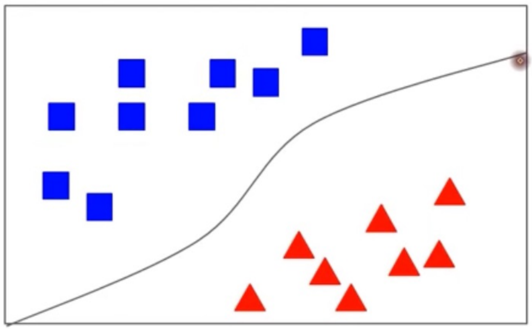
$$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1w_1 + x_2w_2 + x_3w_3 \end{pmatrix}$$

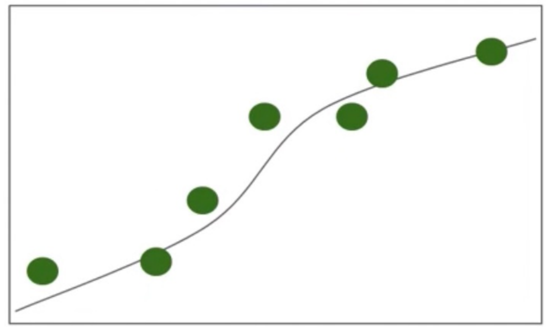
$$H(X) = XW$$

Logistic vs Linear

What is the difference between logistic and linear?



VS



Discrete (Counted)
Shoe Size /The number of workers in a company

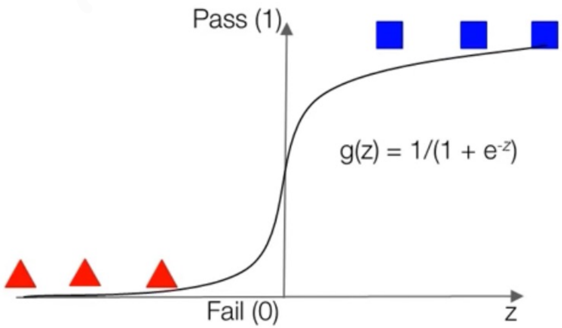
Continous (Measured)
Time / Weight / Height

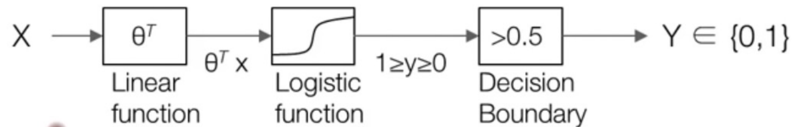
[Python Code]

```
Logistic_Y= [[0], [0], [0], [1], [1], [1]] # One Hot
Linear_Y = [828.659973, 833.450012, 819.23999, 828.349976, 831.659973] # Numeric
```

Sigmoid (Logistic) function

g(z) function out value is between 0 and 1



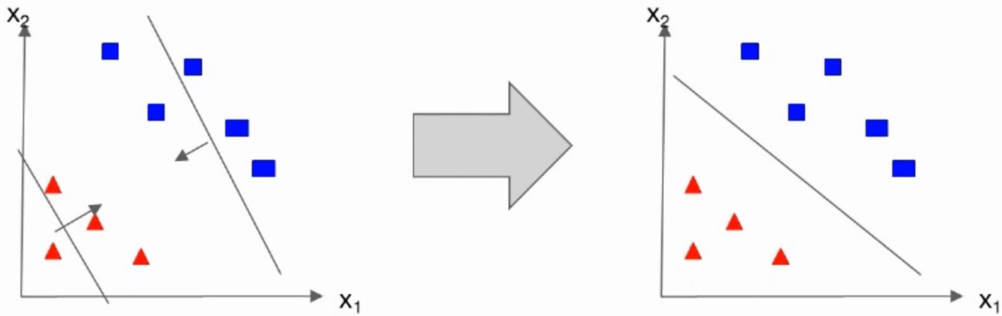


Where we define g(z) -> z is a real number -> $g(z) = e^z/(e^z + 1) = 1/(1 + e^{-z})$

```
[Tensorflow Code]
hypothesis  = tf.sigmoid(z) # z=tf.matmul(X, θ) + b
hypothesis  = tf.div(1., 1. + tf.exp(z))
```

Cost Function

the cost function to fit the parameters(θ)



Given the training set how to we chose/fit θ? $h_\theta(x) = y$ then Cost = 0

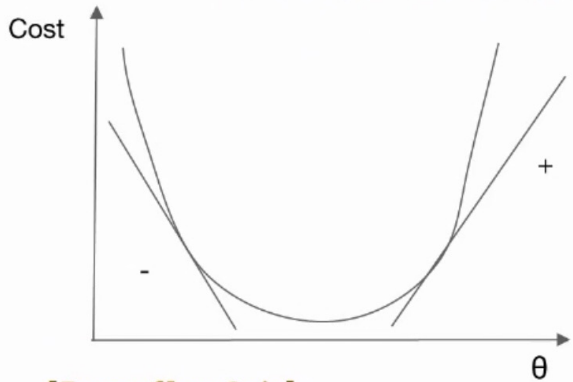
$$cost(h_\theta(x),y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

[Tensorflow Code]

```
def loss_fn(hypothesis, labels):
    cost = -tf.reduce_mean(labels * tf.log(hypothesis) + (1 - labels) * tf.log(1 - hypothesis))
    return cost
```

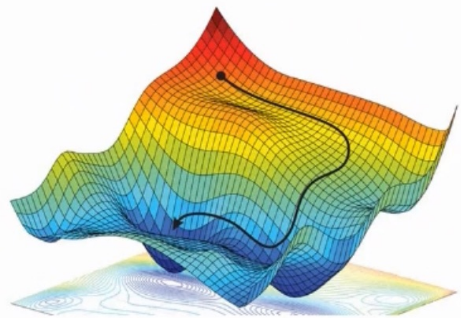
Optimization

How to minimize the cost function



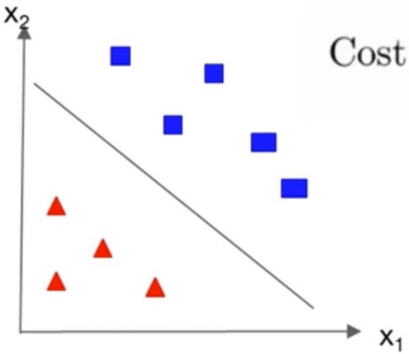
$$\mathrm{cost}(h_{\theta}(x),y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

$$\text{Repeat } \left\{ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \right\}$$



```
[Tensorflow Code]
def grad(hypothesis, labels):
   with tf.GradientTape() as tape:
       loss_value = loss_fn(hypothesis, labels)
   return tape.gradient(loss_value, [W,b])
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
optimizer.apply_gradients(grads_and_vars=zip(grads,[W,b]))
```

Summary



$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$