Case 1

Consumption Prediction of a detached Residential building

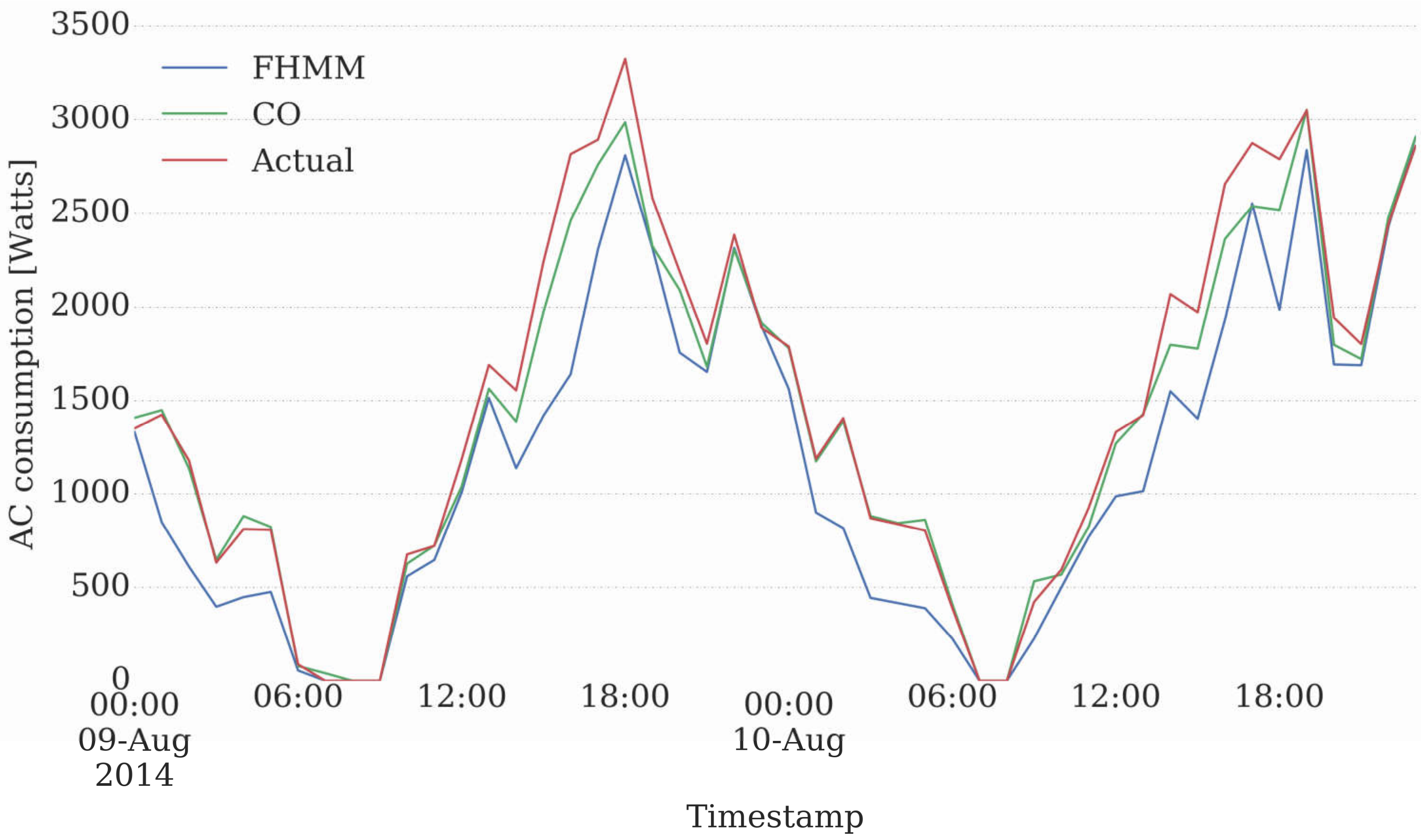
Dataset – Pecan Street Dataport

PECAN STREET DATAPORT FROM PECAN STREET

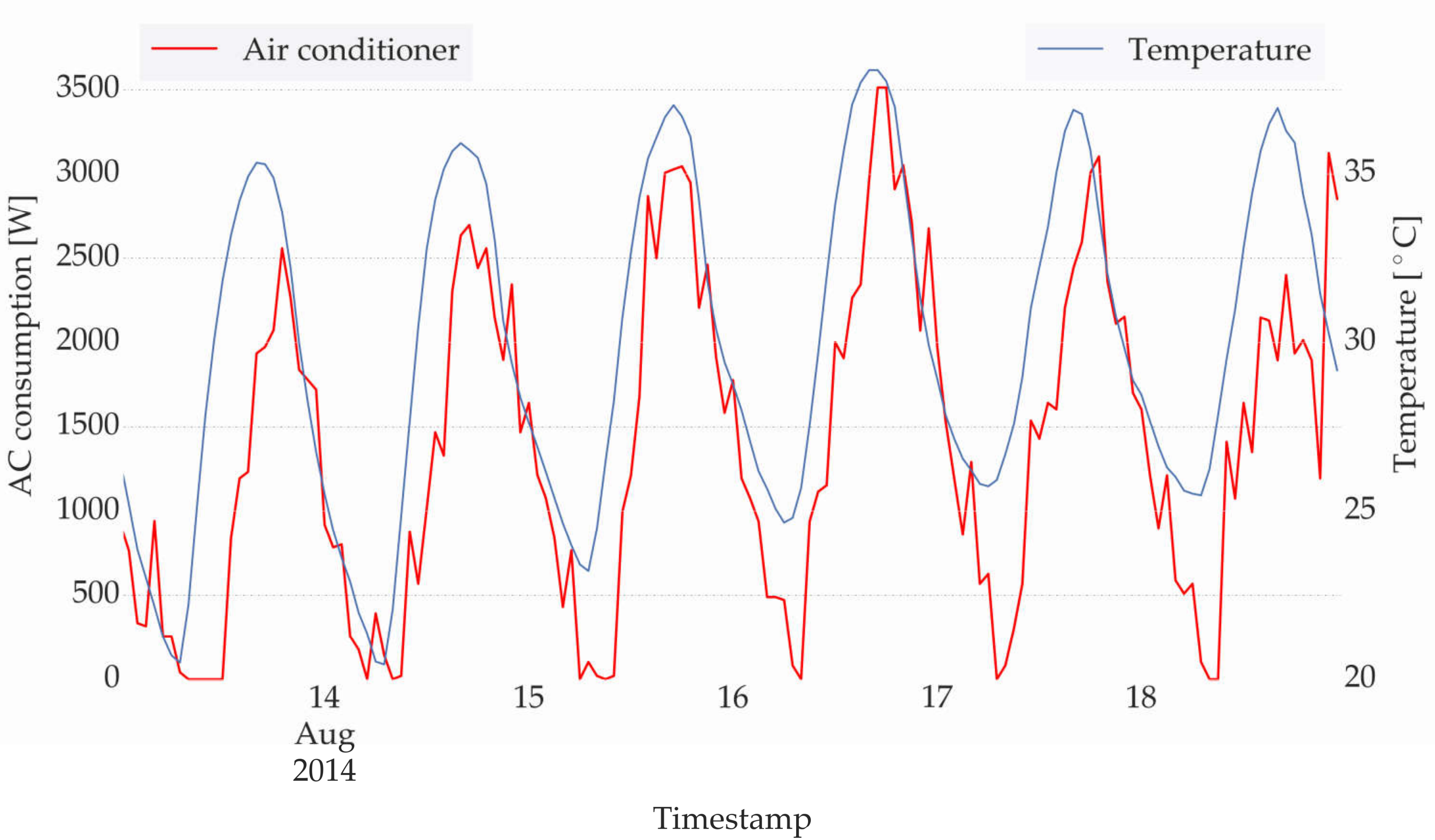
- Location: Austin, Texas
- Time-stamped data including:
- ✓ Aggregate consumption
- ✓ Appliance-by-appliance consumptions including Air conditioner
- ✓ Ambient temperature

- Time-stamped PV generation from buildings: represents solar irradiation

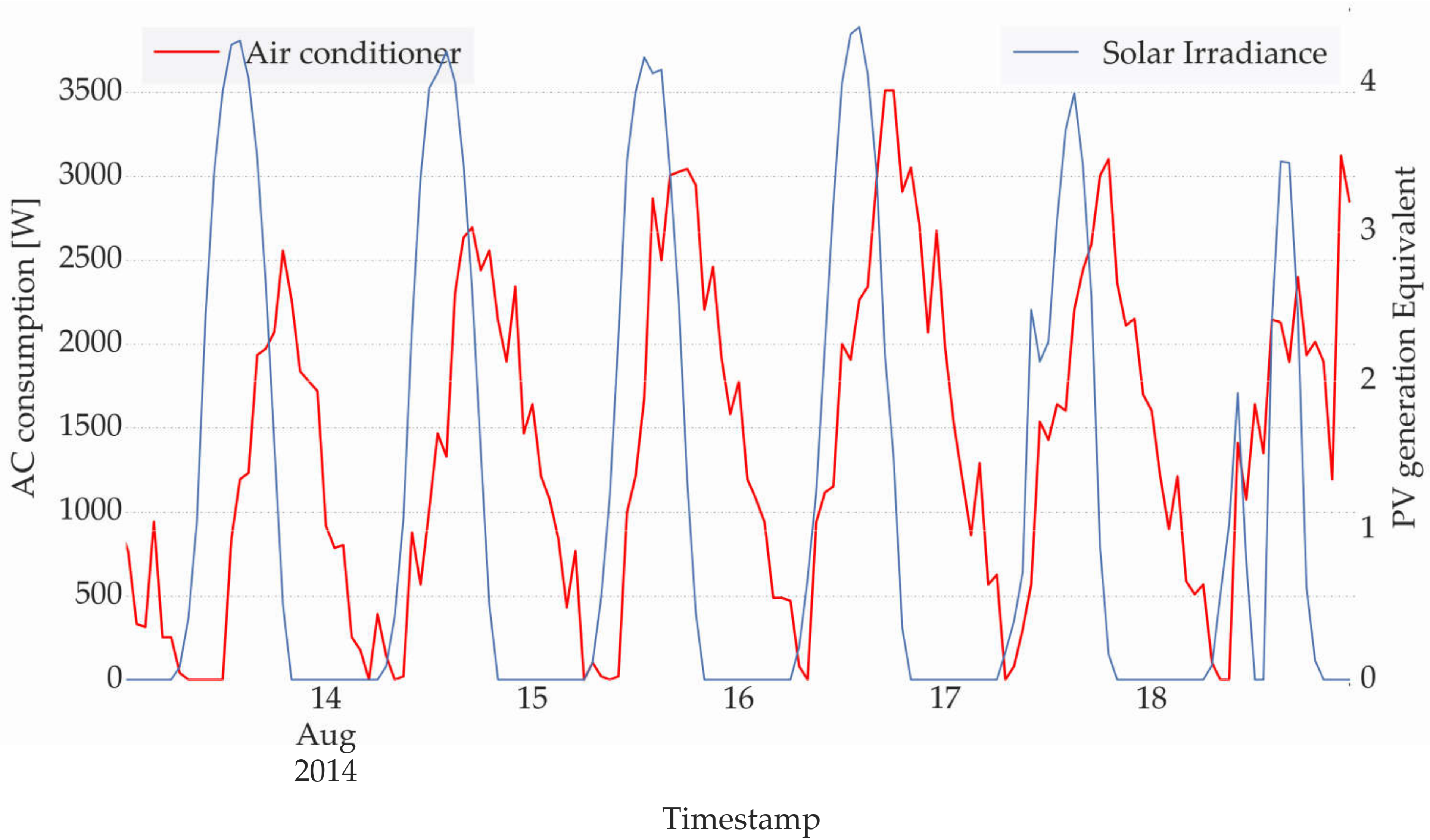
Air conditioner disaggregation results



Correlation investigation and feature selection



Correlation investigation and feature selection



Preprocessing: Features Selection

Independent Variables

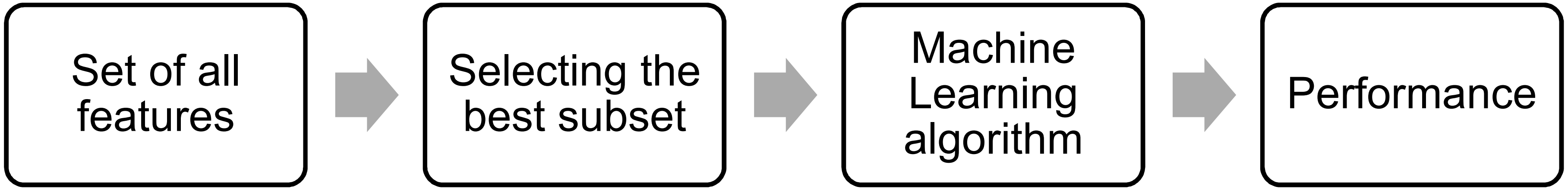
Dependent Variable

S.No	Date	Time	Hour Of day (0-23)	Day or Night (1/0)	Temperature (°C)	Irradiance (0-1)	Relative Humidity (0-100)	Energy consumed by AC (W)
1	01-01-2017	14:00:00	14	1	44	0.83	60	2400
2	01-01-2017	15:00:00	15	1	44.5	0.85	60	2500
3	…	…	…	…	…	…	…	…

$$Energy\ consumed(t) = \mathrm{F}(Date \quad time(t), Hour(t), Temperature(t), IRR(t), RH(t))$$

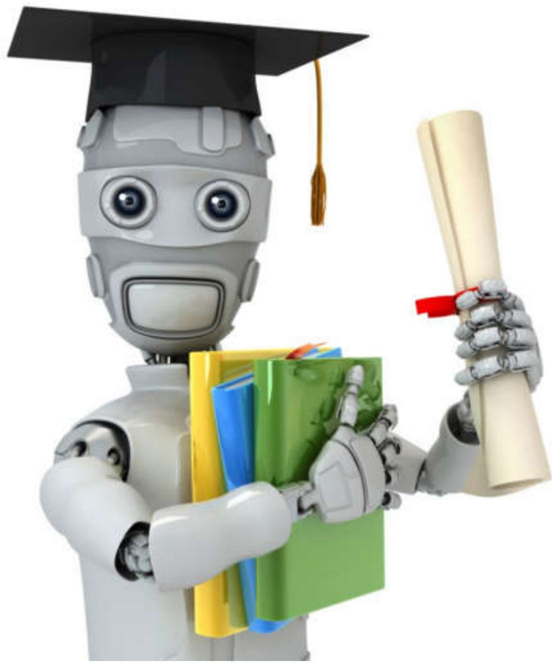
Question:
What other independent variables affect Energy consumed by an Air conditioner ?

Review of Steps



1. Identifying set of all features >> Domain knowledge
2. Selecting the best subset >> Correlation (Person, Spearman)
3. Learning algorithm >> Performance of the model

A Rapid Introduction to Machine Learning*



*Important Note: This introduction is a simplified presentation directly extracted from Andrew Ng's Stanford ML course

What is Machine Learning ???

- **Simple informal Definition (Arthur Samuel)**
  - ✓ the field of study that gives computers the ability to learn without being explicitly programmed
- **A more accurate definition (Tom Mitchel)**
  - ✓ A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

- **Example: playing checkers.**
  - ✓ E = the experience of playing many games of checkers, T = the task of playing checkers.
  - ✓ P = the probability that the program will win the next game.

- **Main Machine Learning Categories**
  - ✓ Supervised Learning
  - ✓ Unsupervised Learning

- **Supervised Learning**
- ✓ in supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output. Supervised learning problems are categorized into :
- **Regression**
- ✓ In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function.
- ❑ Example: Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output

- **Classification**
- ✓ In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.
- ❑ Example: given a picture of tumour we would like to decide whether it is malignant or benign

- **Unsupervised learning**

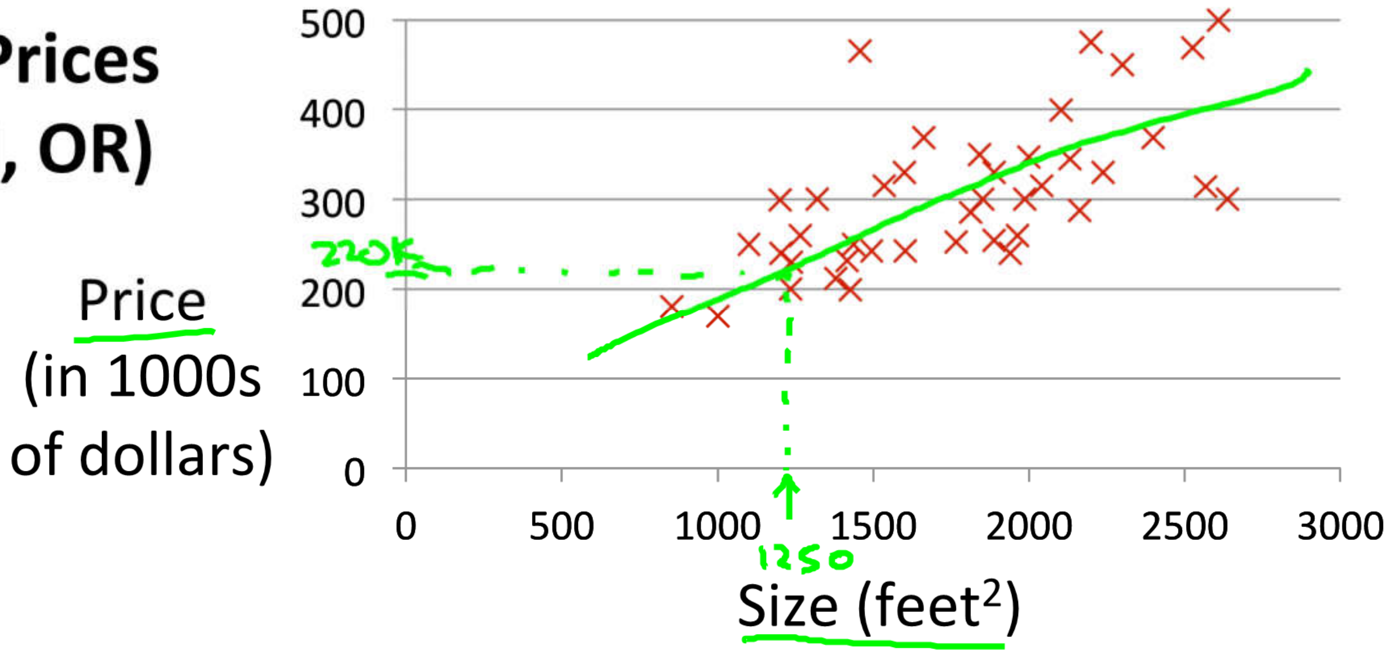
✓ Unsupervised learning, allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

✓ We can derive this structure by clustering the data based on relationships among the variables in the data.

✓ With unsupervised learning there is no feedback based on the prediction results, i.e., there is no teacher to correct you.

- Example: (Clustering) Take a collection of 1000 essays written on the US Economy, and find a way to automatically group these essays into a small number that are somehow similar or related by different variables, such as word frequency, sentence length, page count, and so on.

Housing Prices
(Portland, OR)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Classification: Discrete-valued output

*Source: Andrew Ng's Machine Learning Course

Training set of housing prices (Portland, OR)

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

m = 47

Notation:

- **m** = Number of training examples
- **x**'s = "input" variable / features
- **y**'s = "output" variable / "target" variable

(x, y) - one training example

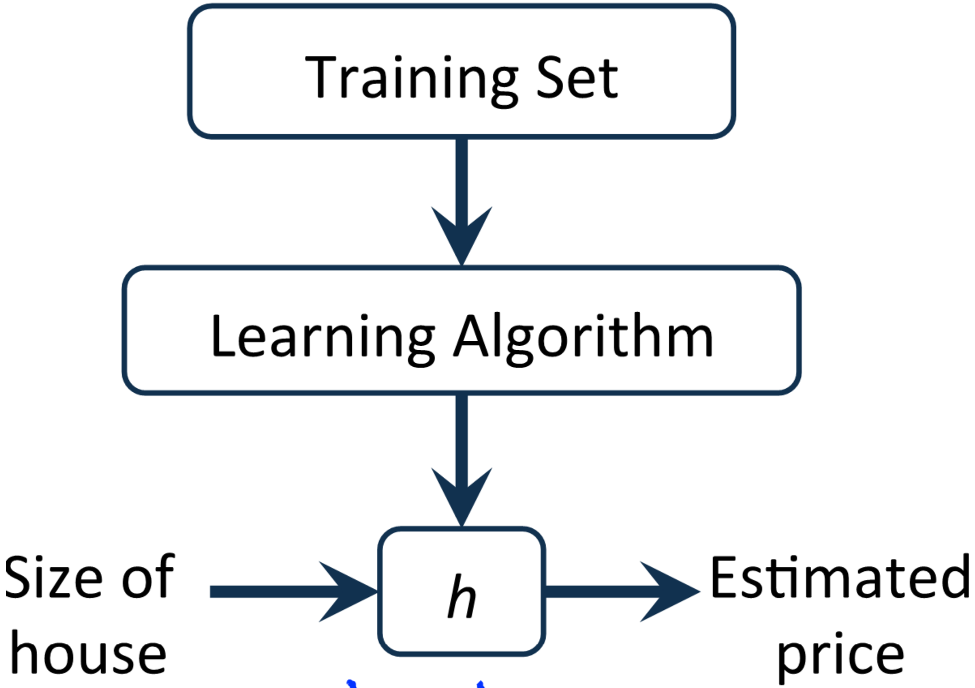
$(x^{(i)}, y^{(i)})$ - i^{th} training example

$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$

*Source: Andrew Ng's Machine Learning Course



x

hypothesis

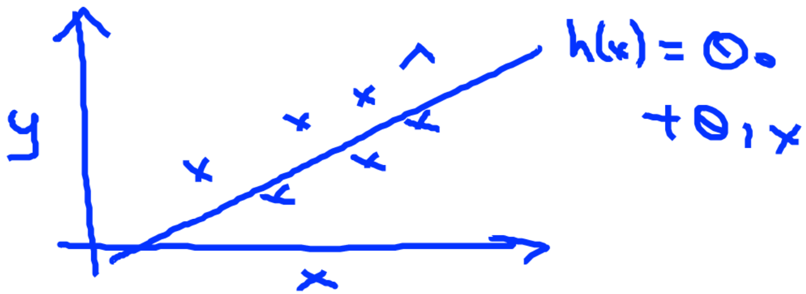
(estimated value of y)

h maps from x's to y's.

How do we represent *h* ?

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Shorthand: $h(x)$



Linear regression with one variable. (x)

Univariate linear regression.

└ one variable

*Source: Andrew Ng's Machine Learning Course

Training Set

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

$m = 47$

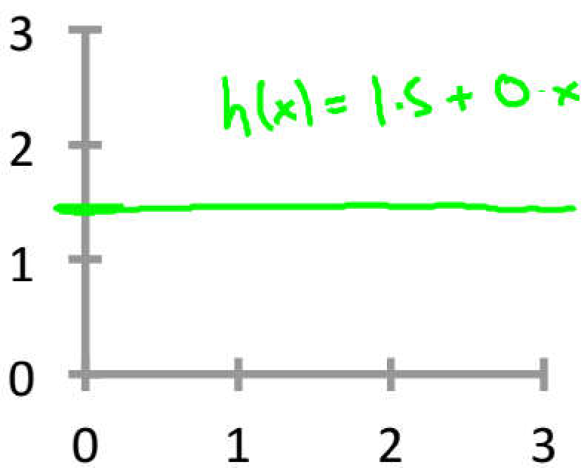
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's: Parameters

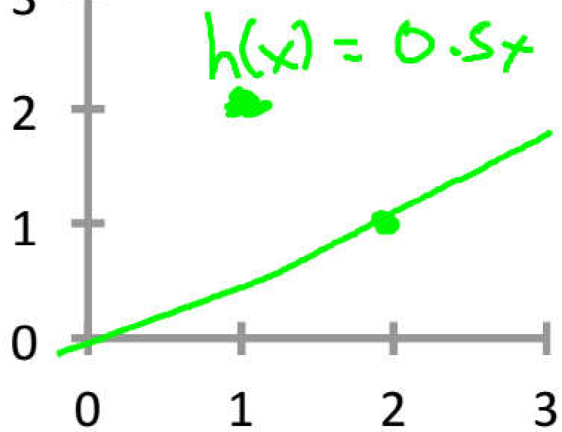
How to choose θ_i's ?

*Source: Andrew Ng's Machine Learning Course

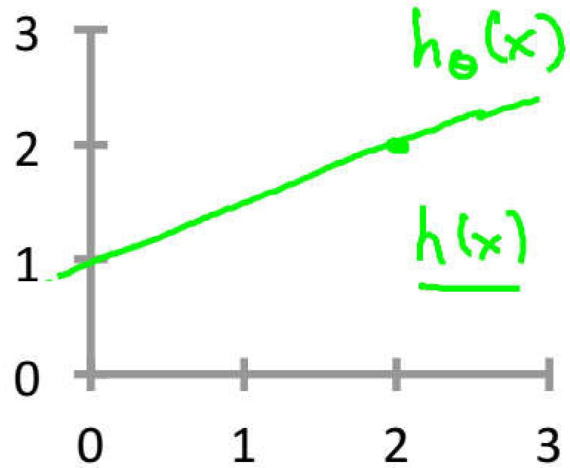
$$h_\theta(x) = \theta_0 + \theta_1 x$$



$\theta_0 = 1.5$
$\theta_1 = 0$



$\theta_0 = 0$
$\theta_1 = 0.5$



$\theta_0 = 1$
$\theta_1 = 0.5$

*Source: Andrew Ng's Machine Learning Course

Training Set

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

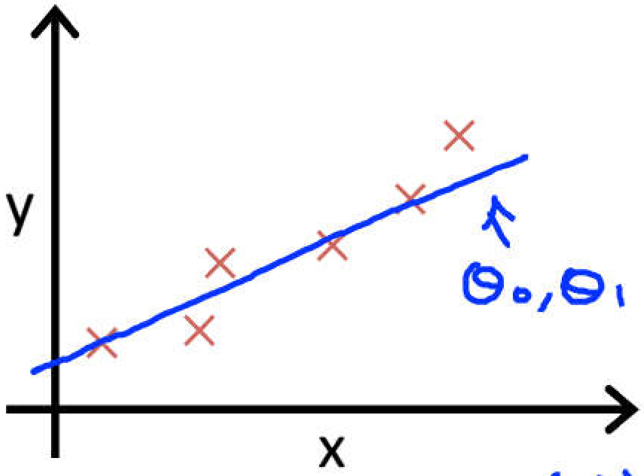
m = 47

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's: Parameters

How to choose θ_i's ?

*Source: Andrew Ng's Machine Learning Course



$(x^{(i)}, y^{(i)})$

Idea: Choose θ_0, θ_1 so that
$h_\theta(x)$ is close to y for our
training examples (x, y)

x, y

$$\underset{\theta_0, \theta_1}{\text{minimize}} \; \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

m: # training examples

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$\underset{\theta_0, \theta_1}{\text{minimize}} \; \underbrace{J(\theta_0, \theta_1)}_{\text{Cost function}}$$

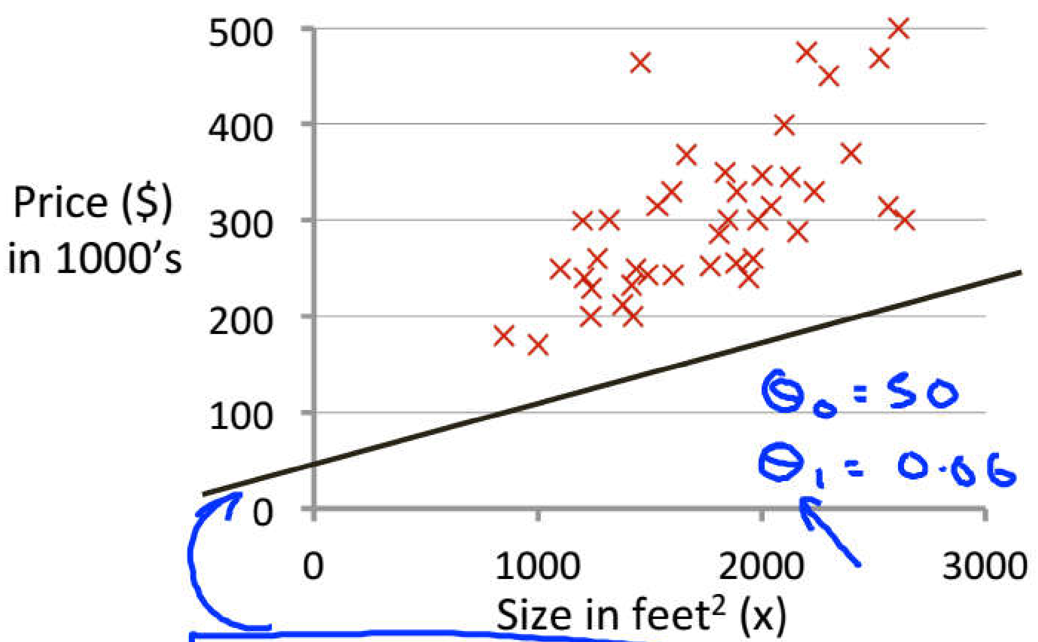
Squared error function

*Source: Andrew Ng's Machine Learning Course

Linear Regression: One Variable

$h_\theta(x)$

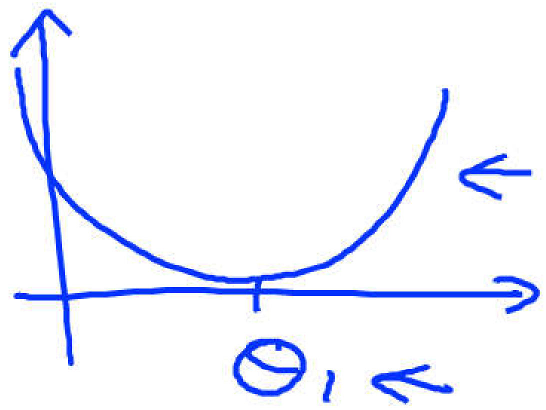
(for fixed θ_0, θ_1, this is a function of x)



$$\boxed{h_\theta(x) = 50 + 0.06x}$$

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



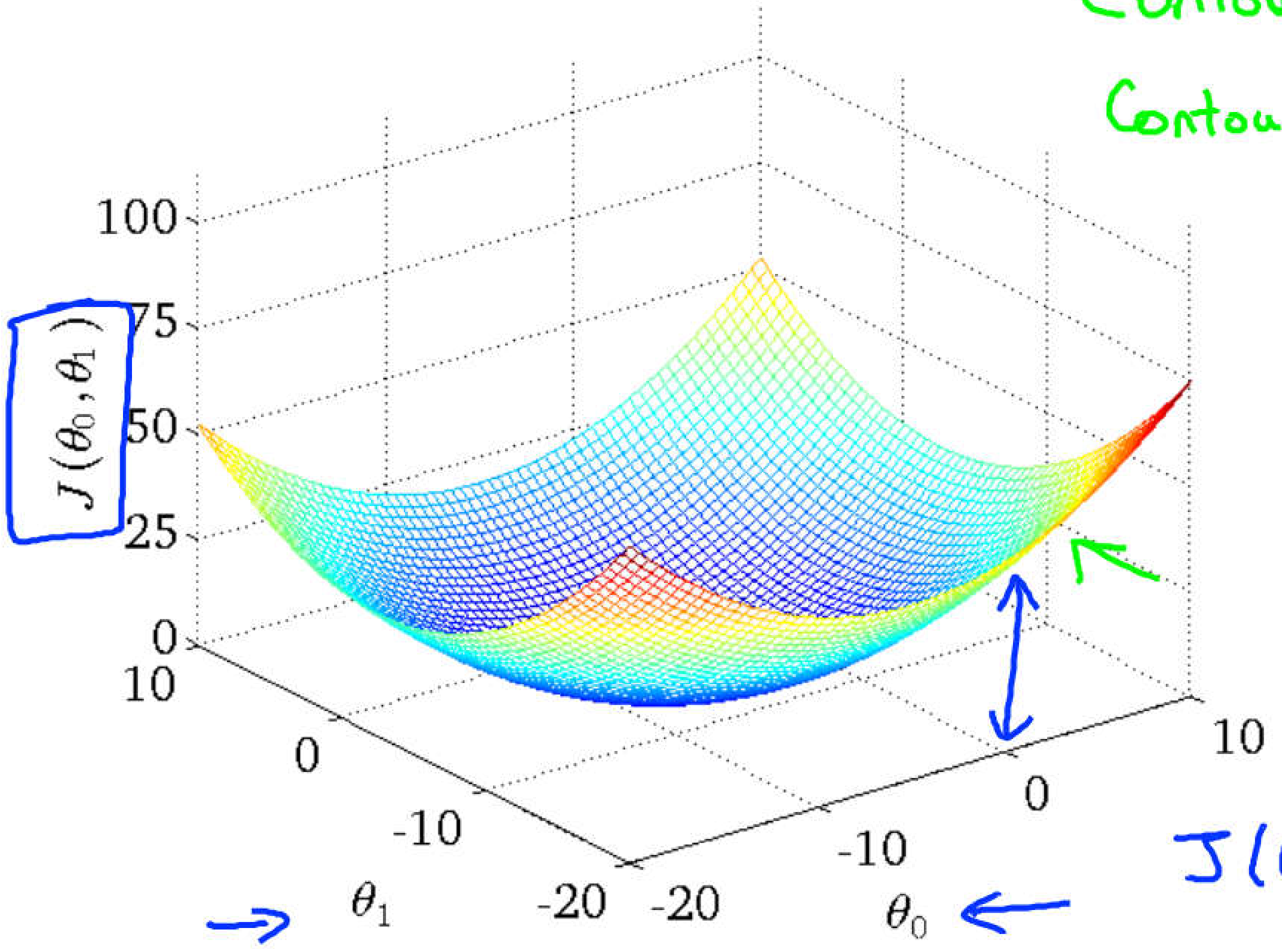
$$\boxed{\theta_0, \theta_1}$$

*Source: Andrew Ng's Machine Learning Course

Linear Regression: One Variable

Contour plots

Contour figures -



$J(\theta_0, \theta_1)$

*Source: Andrew Ng's Machine Learning Course

Have some function $J(\theta_0, \theta_1)$ $J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ $\min_{\theta_0, \ldots, \theta_n} J(\theta_0, \ldots, \theta_n)$

Outline:

- Start with some θ_0, θ_1 (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

  until we hopefully end up at a minimum

*Source: Andrew Ng's Machine Learning Course

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}$$ (simultaneously update $j = 0$ and $j = 1$)

}

learning rate

derivative

$\min_{\theta_1} J(\theta_1)$

$\theta_1 \in \mathbb{R}$.

*Source: Andrew Ng's Machine Learning Course

Linear Regression: One Variable

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(\text{for } j = 1 \text{ and } j = 0)$$

}

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

***Source: Andrew Ng's Machine Learning Course**

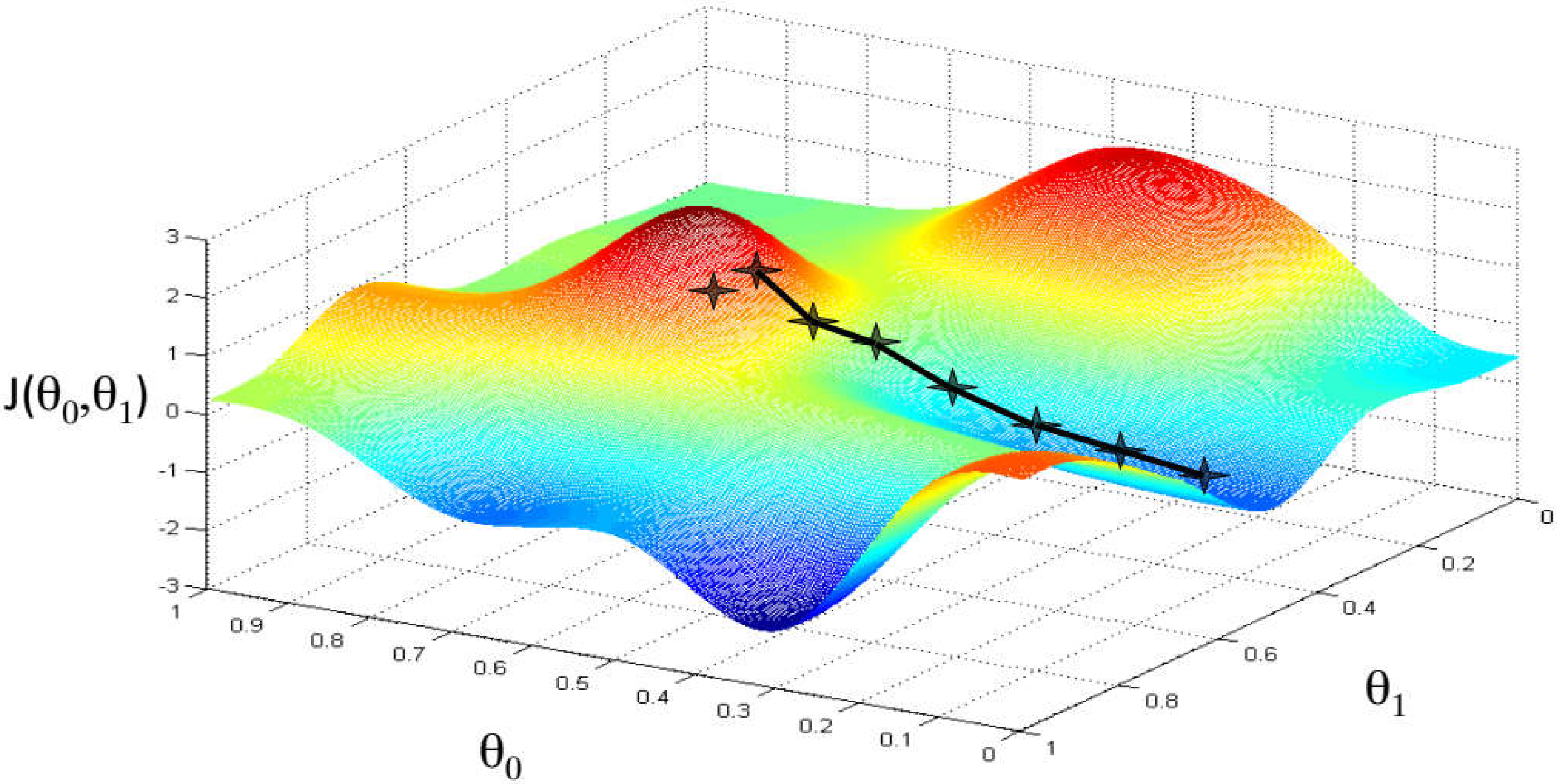
Gradient descent algorithm

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_0 := \theta_0 - \alpha \underbrace{\boxed{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)}}_{\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)} \\
&\quad \theta_1 := \theta_1 - \alpha \underbrace{\boxed{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}}}_{\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)} \\
&\}
\end{aligned}
$$

update θ_0 and θ_1 simultaneously

*Source: Andrew Ng's Machine Learning Course

Linear Regression: One Variable

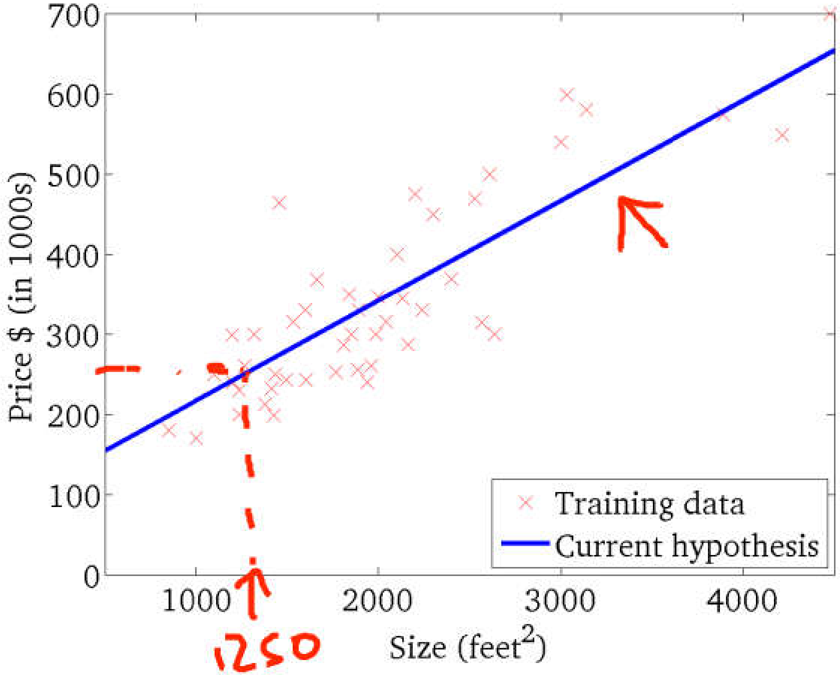


*Source: Andrew Ng's Machine Learning Course

Linear Regression: One Variable

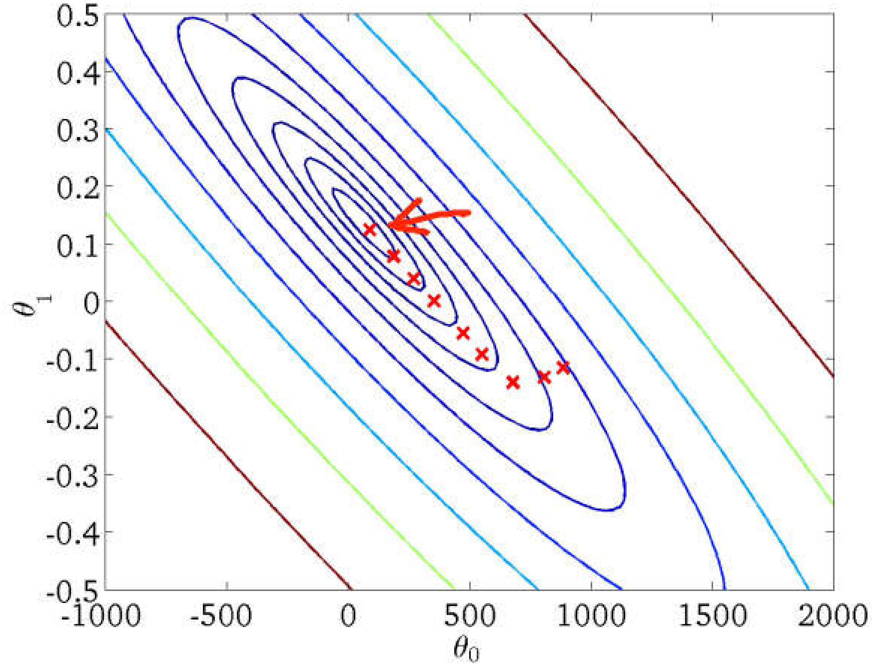
$$h_\theta(x)$$

(for fixed θ_0, θ_1, this is a function of x)



$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



*Source: Andrew Ng's Machine Learning Course

Multiple features (variables).

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

$m = 47$

Notation:

- n = number of features $n = 4$
- $x^{(i)}$ = input (features) of i^{th} training example.
- $x_j^{(i)}$ = value of feature j in i^{th} training example.

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_3^{(2)} = 2$$

*Source: Andrew Ng's Machine Learning Course

Hypothesis:

Previously: $\cancel{h_\theta(x) = \theta_0 + \theta_1 x}$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

E.g. $\underline{h_\theta(x)} = \underline{80} + \underline{0.1} x_1 + \underline{0.01} x_2 + 3x_3 - 2x_4$

age

*Source: Andrew Ng's Machine Learning Course

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

$x_0 = 1$

Parameters: $\cancel{\theta_0, \theta_1, \ldots, \theta_n}$ θ n+1 - dimensional vector

Cost function:

$$\cancel{J(\theta_0, \theta_1, \ldots, \theta_n)} = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$J(\theta)$

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} \cancel{J(\theta_0, \ldots, \theta_n)}\ J(\theta)}$$

} (simultaneously update for every $j = 0, \ldots, n$)

*Source: Andrew Ng's Machine Learning Course

$x_0 = 1$

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\cancel{\theta_0, \theta_1, \ldots, \theta_n}$ θ n+1 - dimensional vector

Cost function:

$$\cancel{J(\theta_0, \theta_1, \ldots, \theta_n)} = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$J(\theta)$

Gradient descent:

Repeat $\{$

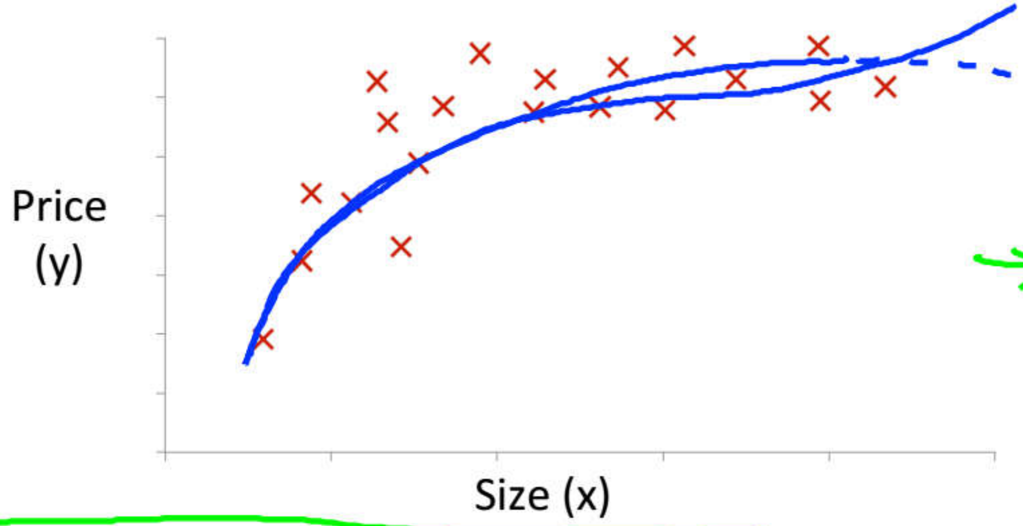
$$\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} \cancel{J(\theta_0, \ldots, \theta_n)} \; J(\theta)}$$

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

*Source: Andrew Ng's Machine Learning Course

Linear Regression: Polynomial Regression

Polynomial regression



$$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$$

$$\rightarrow \boxed{\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3}$$

$$\boxed{h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3}$$

$$= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$$

$$\left.\begin{aligned} \rightarrow x_1 &= (size) \\ \rightarrow x_2 &= (size)^2 \\ \rightarrow x_3 &= (size)^3 \end{aligned}\right\}$$

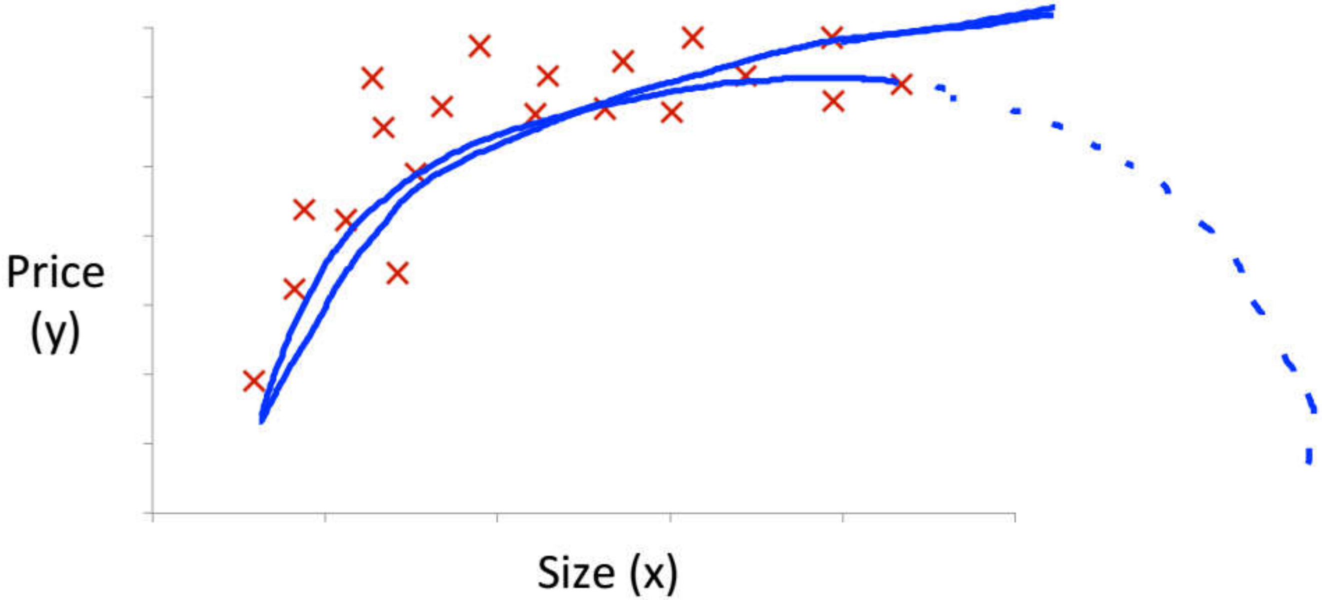
Size: 1 - 1000

$Size^2$: 1 - 1000,000

$Size^3$: 1 - 10^9

*Source: Andrew Ng's Machine Learning Course

Choice of features



$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$

$$\boxed{h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}}$$

*Source: Andrew Ng's Machine Learning Course

Linear Regression: But what if we want to try multiple non-linear features!!

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \ldots)$$

x_1 = size
x_2 = # bedrooms
x_3 = # floors
x_4 = age
$\ldots$
x_{100}

$n = 100$

$\rightarrow x_1^2,\ x_1x_2,\ x_1x_3,\ x_1x_4 \ldots x_1x_{100}$

$x_2^2,\ x_2x_3 \ldots$

≈ 5000 feature $\quad O(n^2) \quad \approx \frac{n^2}{2}$

$\rightarrow x_1^2, x_2^2, x_3^2, \ldots, x_{100}^2$

$\rightarrow x_1x_2x_3,\ x_1^2x_2,\ x_{10}x_{11}x_{17}, \ldots$

$O(n^3)$

$170{,}000$

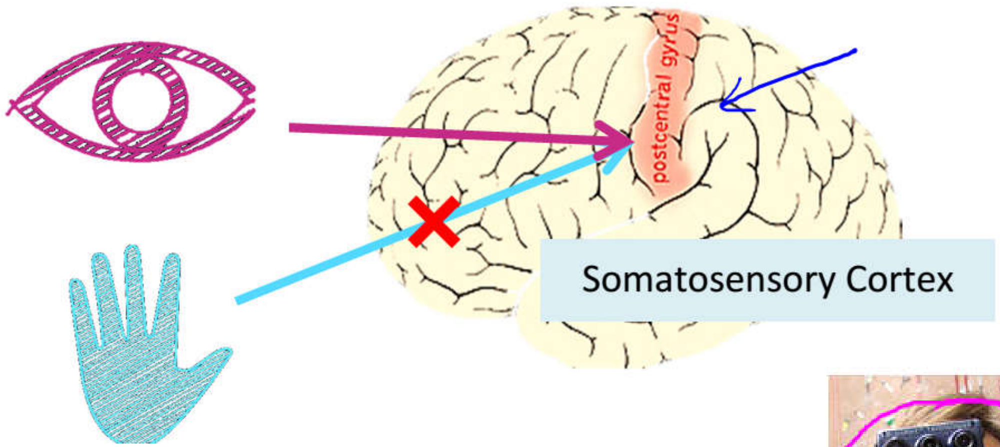
Andrew Ng

*Source: Andrew Ng’s Machine Learning Course

❖ First Solution

✓ Let's have a look at what happens in the nature! → Human brain !

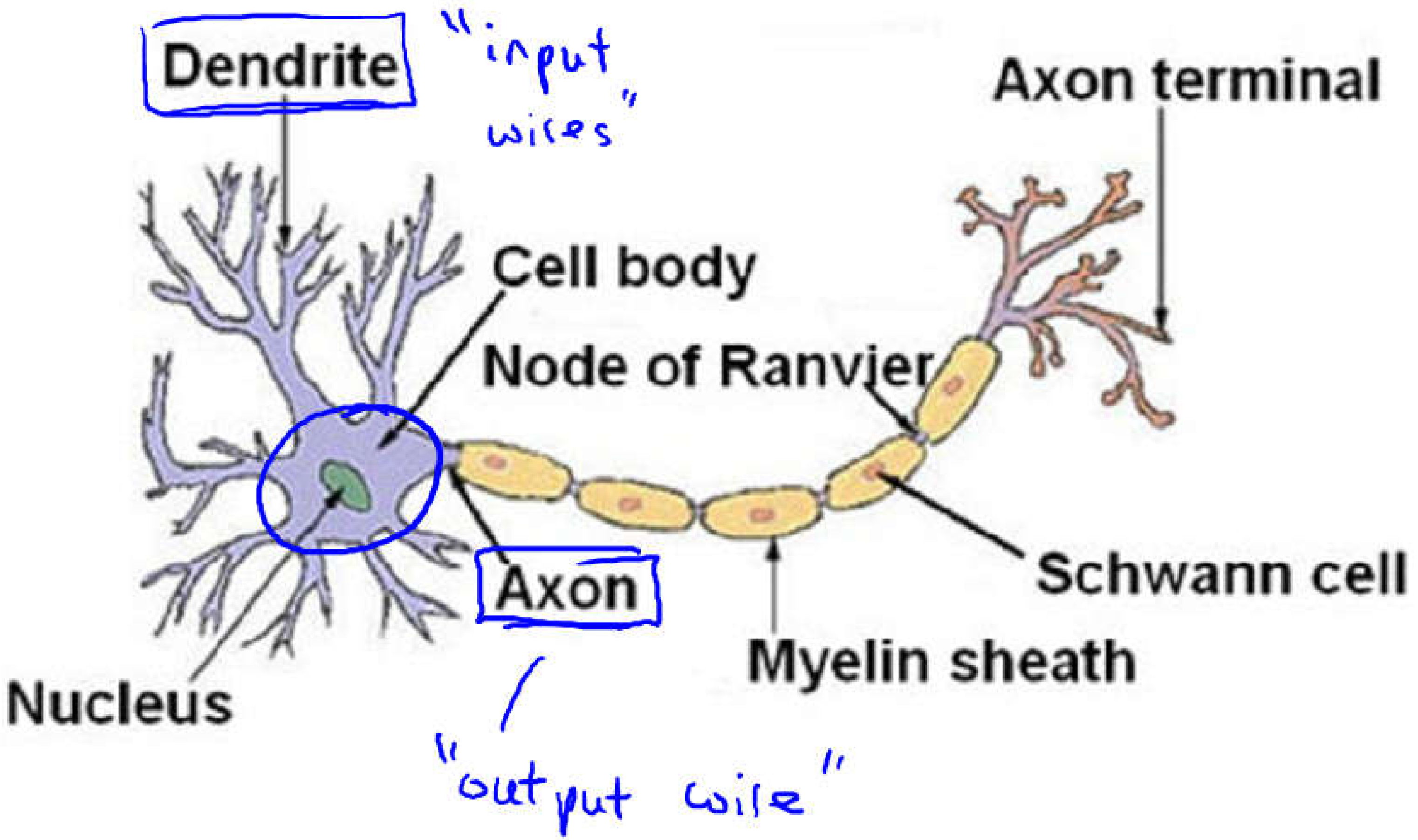
The "one learning algorithm" hypothesis



Seeing with your tongue

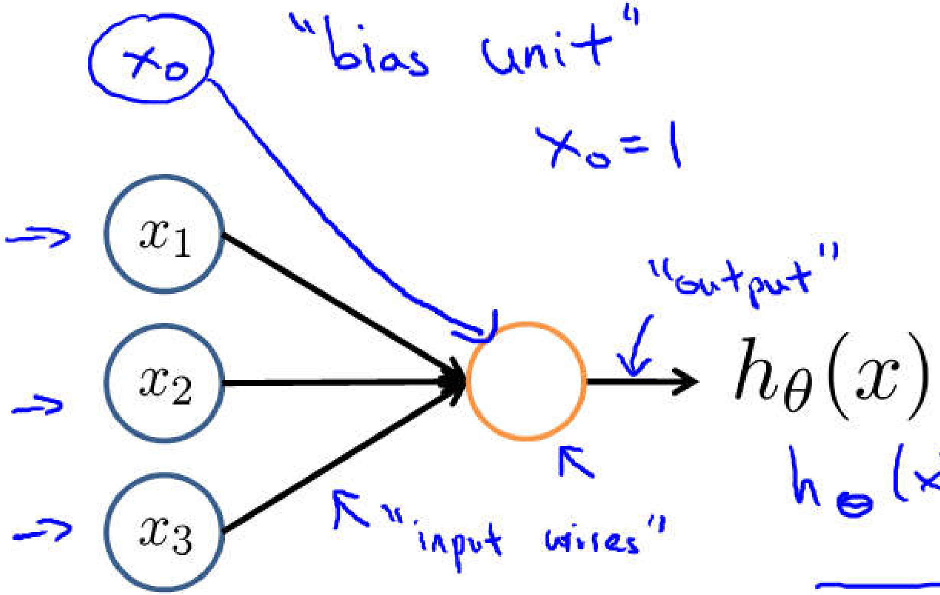
*Source: Andrew Ng's Machine Learning Course

Neurons in Brain



*Source: Andrew Ng's Machine Learning Course

Neuron model: Logistic unit



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

"weights"

(parameters

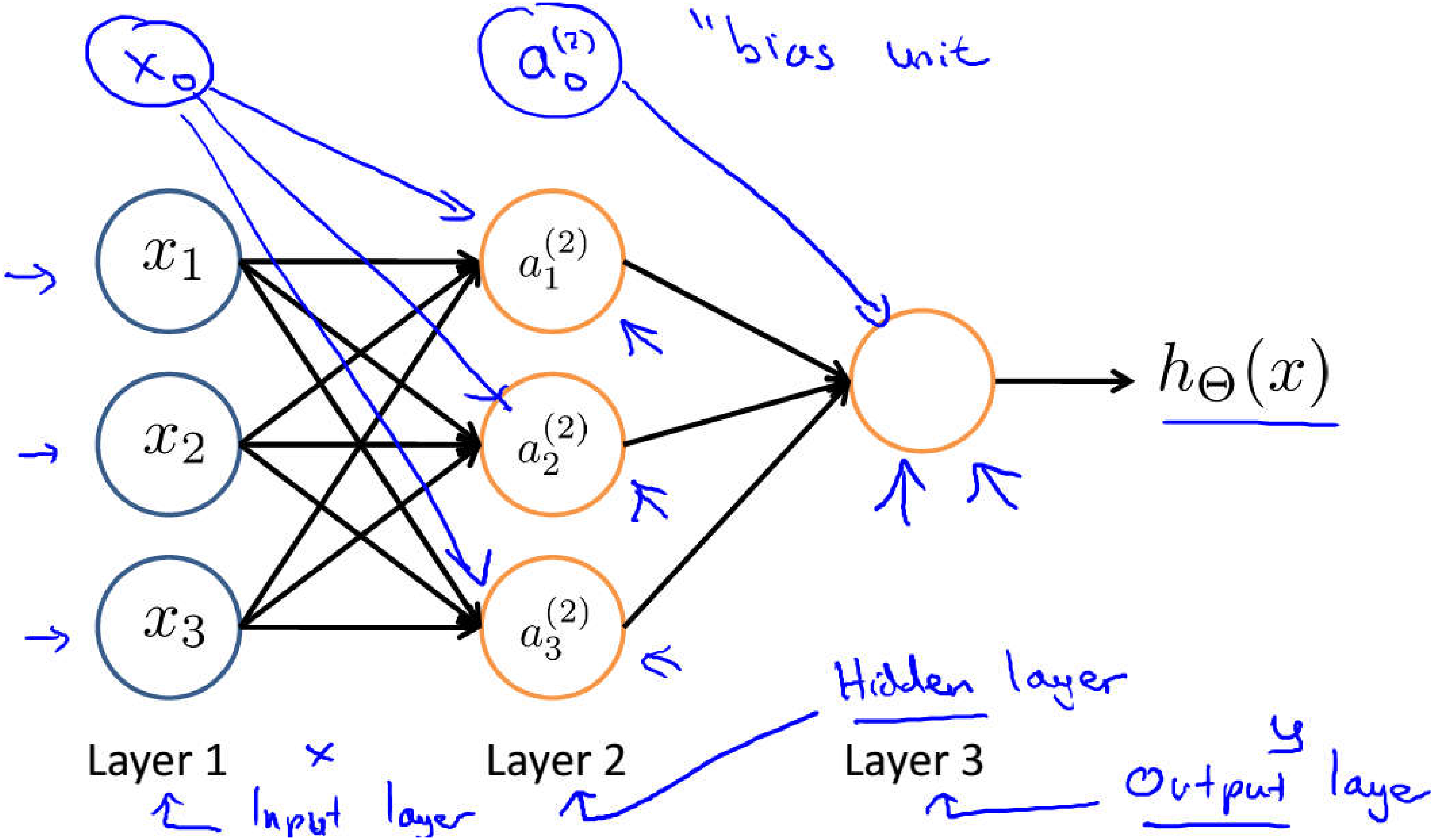
$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Sigmoid (logistic) activation function.

$$g(z) = \frac{1}{1+e^{-z}}$$

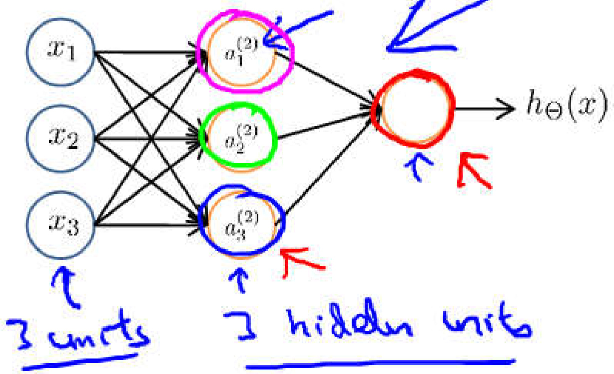
*Source: Andrew Ng's Machine Learning Course

Neural Networks



*Source: Andrew Ng's Machine Learning Course

Neural Network



→ $a_i^{(j)}$ = "activation" of unit i in layer j

→ $\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

$\Theta^{(1)} \in \mathbb{R}^{3\times 4}$ $h_\Theta(x)$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

$\Theta^{(2)}$

→ If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $\boxed{s_{j+1}} \times (\boxed{s_j} + 1)$.

$s_{j+1} \times (s_j + 1)$

*Source: Andrew Ng's Machine Learning Course

Evaluating your hypothesis

Dataset:

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

70% Training set

30% Test set

$$(x^{(1)}, y^{(1)})$$
$$(x^{(2)}, y^{(2)})$$
$$\vdots$$
$$(x^{(m)}, y^{(m)})$$

$$(x_{test}^{(1)}, y_{test}^{(1)})$$
$$(x_{test}^{(2)}, y_{test}^{(2)})$$
$$\vdots$$
$$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$$

m_{test} = no. of test example

$(x_{test}^{(1)}, y_{test}^{(1)})$

*Source: Andrew Ng's Machine Learning Course

Accuracy metrics- Mean squared error and Mean absolute error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n} = \frac{\sum_{i=1}^{n}|e_i|}{n}.$$ [1]

Accuracy metrics- (R2)

A data set has n values marked $y_1,\ldots,y_n$ (collectively known as y_i or as a vector $y = [y_1,\ldots,y_n]^T$), each associated with a predicted (or modeled) value $f_1,\ldots,f_n$ (known as f_i, or sometimes $\hat{y}_i$, as a vector f).

If $\bar{y}$ is the mean of the observed data:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

then the variability of the data set can be measured using three sums of squares formulas:

- The total sum of squares (proportional to the variance of the data):
$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$
- The regression sum of squares, also called the explained sum of squares:
$$SS_{\text{reg}} = \sum_i (f_i - \bar{y})^2,$$
- The sum of squares of residuals, also called the residual sum of squares:
$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

The most general definition of the coefficient of determination is

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$