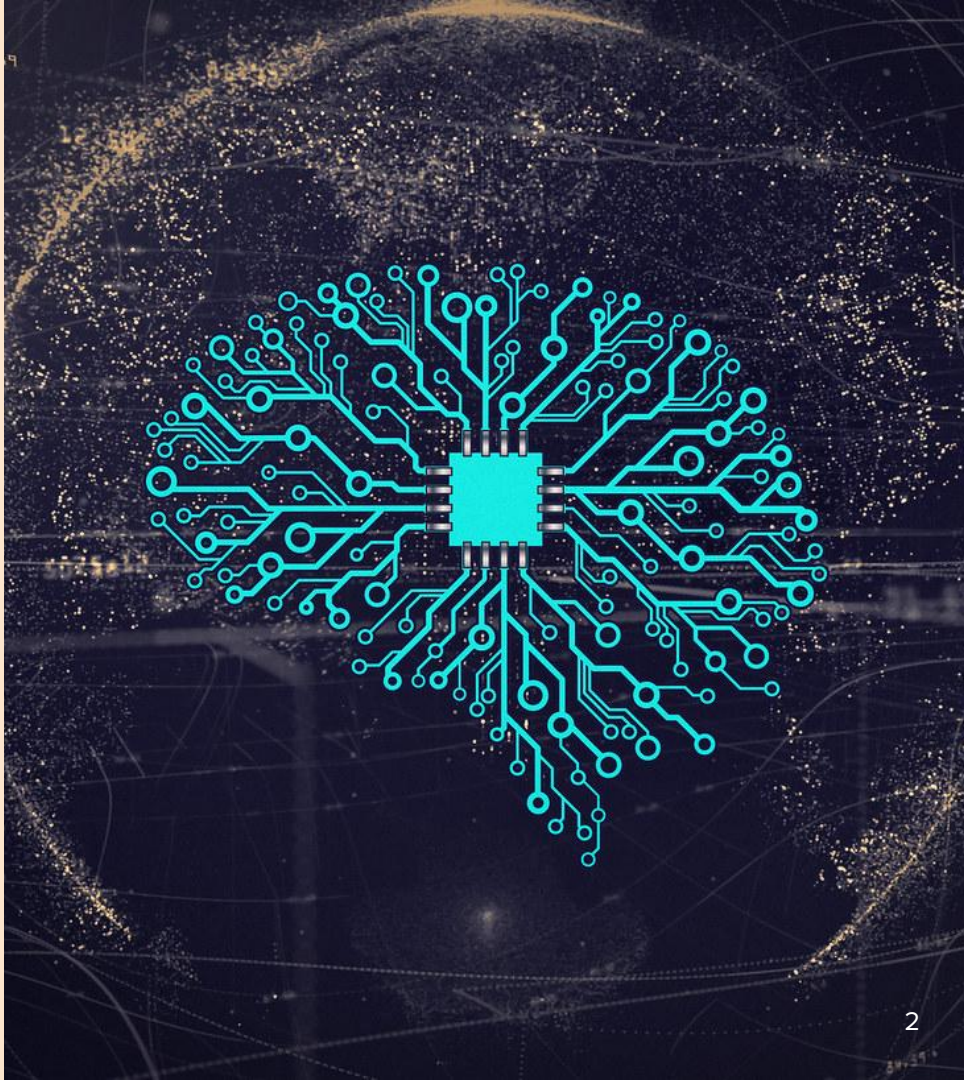
Linear Regression & Logistic Regression

Dr. Paisit Khanarsa

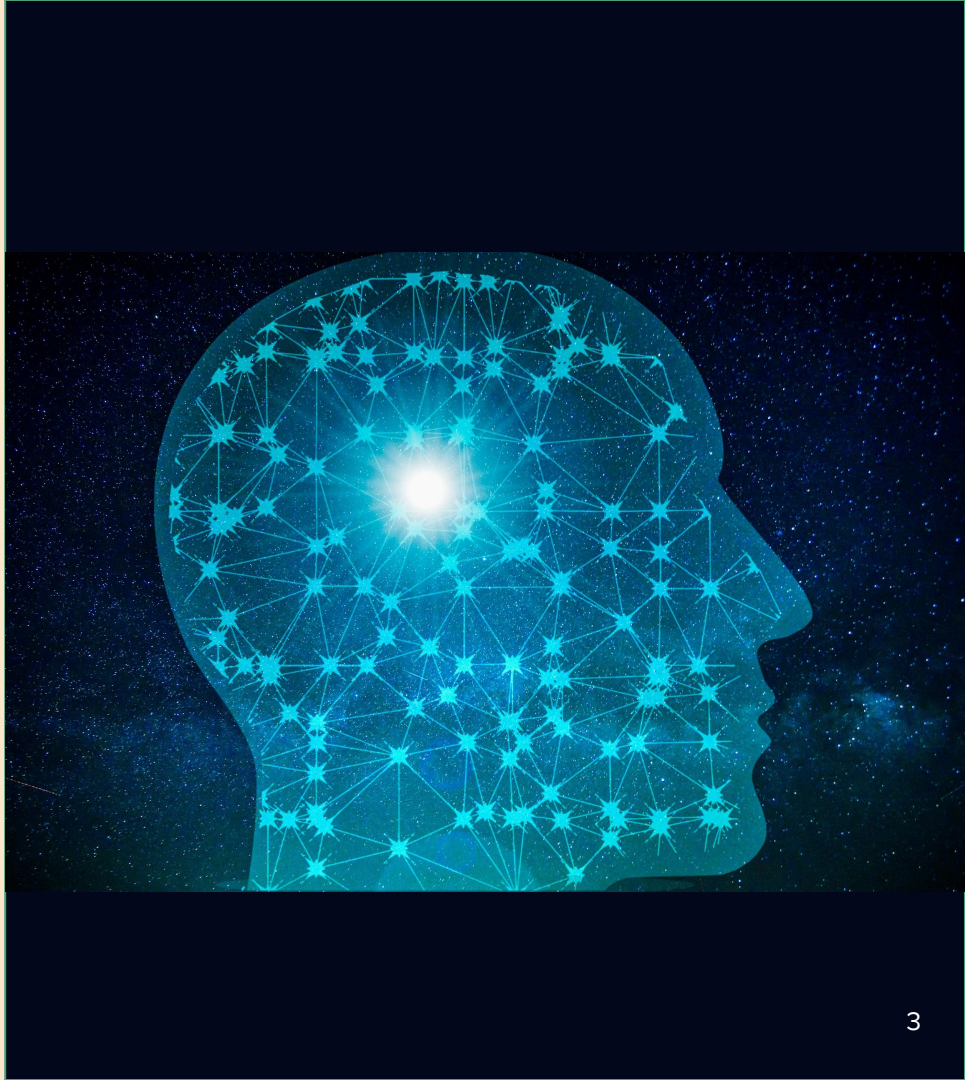
Fibo, Kmutt

Outline

- Machine learning basics
- Cost function
- Over fitting
- Linear regression
- Regression performance
- Logistic regression
- Classification performance



Machine Learning Basics



The basic example

An agent has been selling 300 houses in the last years and want to be able to predict of a house by just knowing the size of the house.

i	Size(feet^2)	Price (dollar)
1	8450	208500
2	9600	181500
3	11250	223500
4	9550	140000
...	...	...

The basic example

In general,

x_i: one feature (input) at i index

y_i: one target (output) at i index

i: sample index

i	x Size(feet^2)	y Price (Dollar)
1	8450 = x_1	208500 = y_1
2	9600 = x_2	181500 = y_2
3	11250 = x_3	223500 = y_3
4	9550 = x_4	140000 = y_4
...	...	...

The basic example

In general,

x^j_i: multi feature (input) at i index

y^k_i: multi target (output) at i index

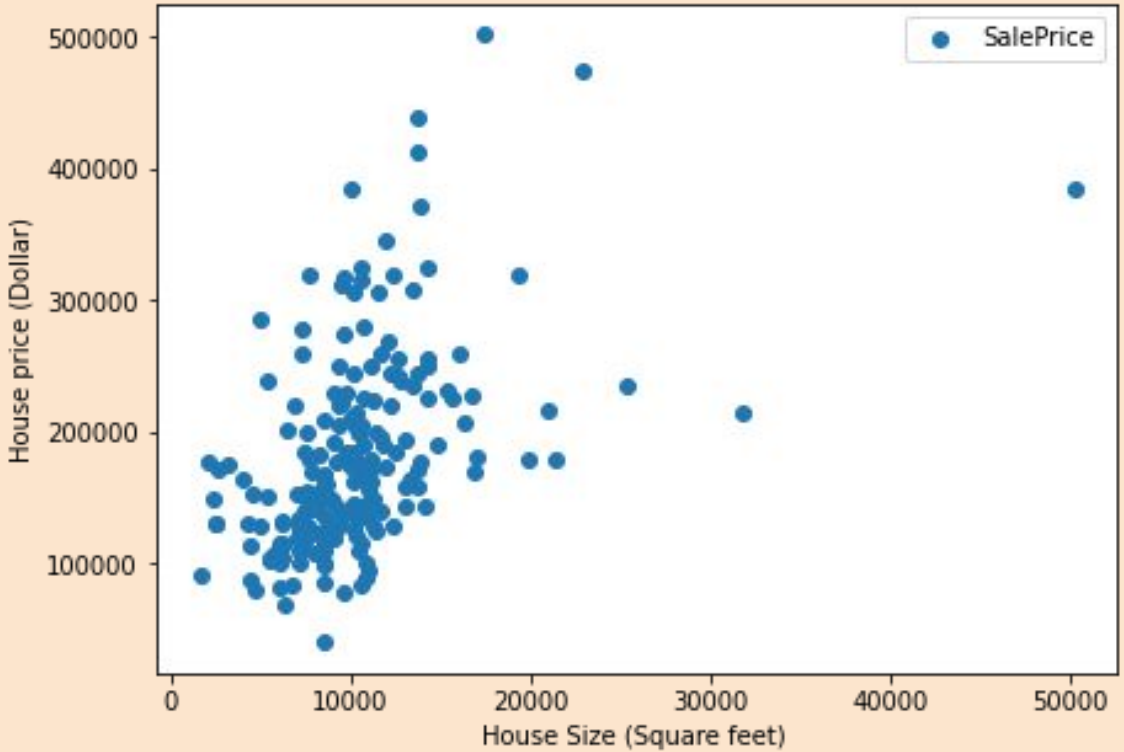
j: feature index

k: target index

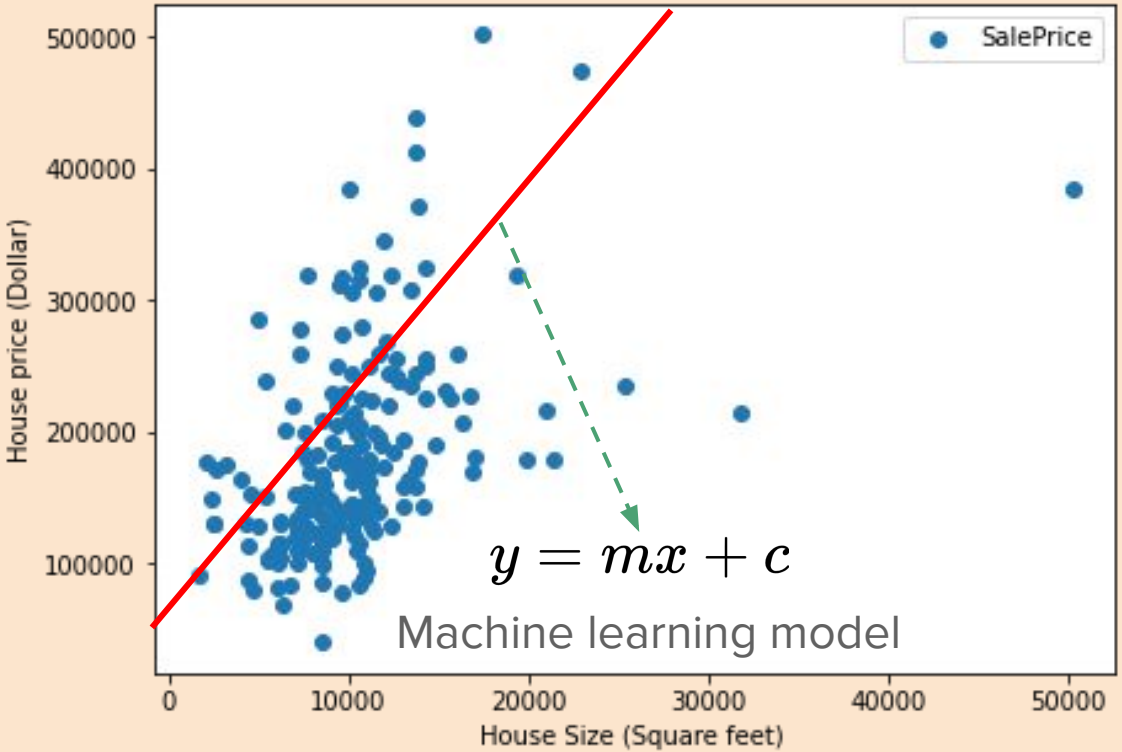
i: sample index

i	$x^{(1)}$ # Bath	$x^{(2)}$ #Bed	$x^{(3)}$ Size(feet^2)	$y^{(1)}$ Sell price (Dollar)	$y^{(2)}$ Buy price (Dollar)
1	$1= x^{(1)}_1$	$1= x^{(2)}_1$	$8450 = x^{(3)}_1$	$208500 = y^{(1)}_1$	$207000 = y^{(2)}_1$
2	$1= x^{(1)}_2$	$2 = x^{(2)}_2$	$9600 = x^{(3)}_2$	$181500 = y^{(1)}_2$	$180000 = y^{(2)}_2$
3	$1= x^{(1)}_3$	$1= x^{(2)}_3$	$11250 = x^{(3)}_3$	$223500 = y^{(1)}_3$	$221500 = y^{(2)}_3$
4	$2 = x^{(2)}_4$	$3= x^{(2)}_4$	$9550 = x^{(3)}_4$	$140000 = y^{(1)}_4$	$120500 = y^{(2)}_4$
...			...	...	

The basic example



The basic example



The basic example

Output

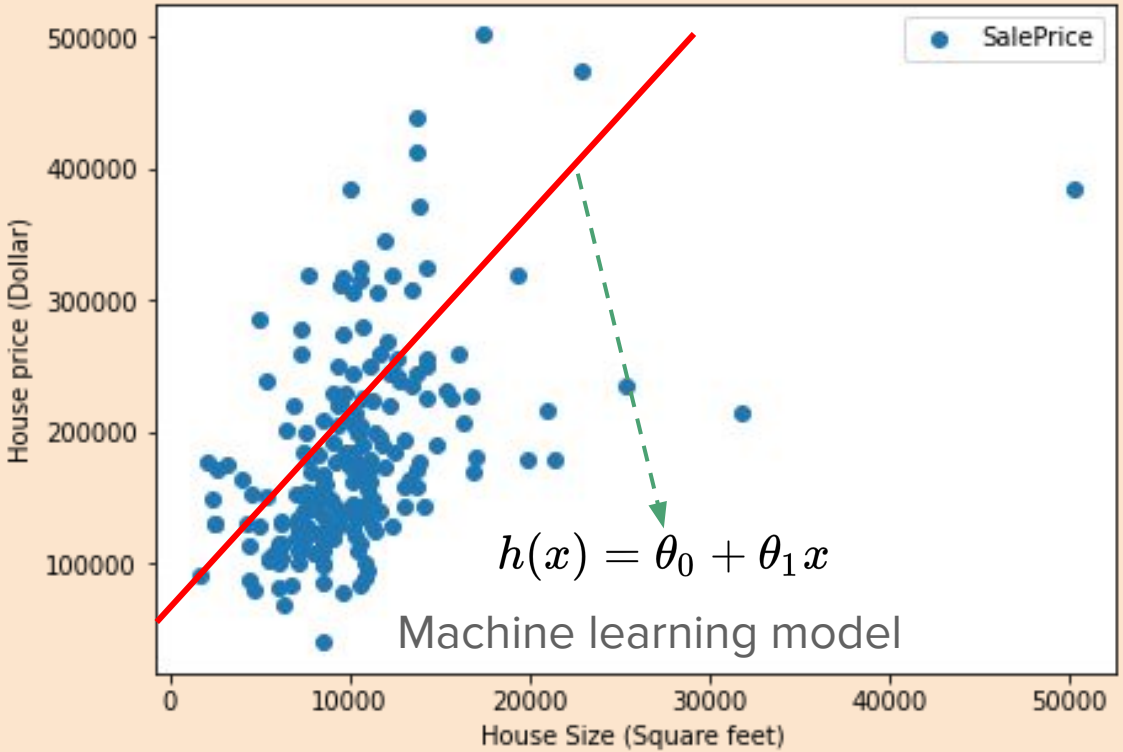
Input

$$h(x) = \theta_0 + \theta_1 x$$

Parameters of the model

$\theta_0 = 50000$

$\theta_1 = 15$

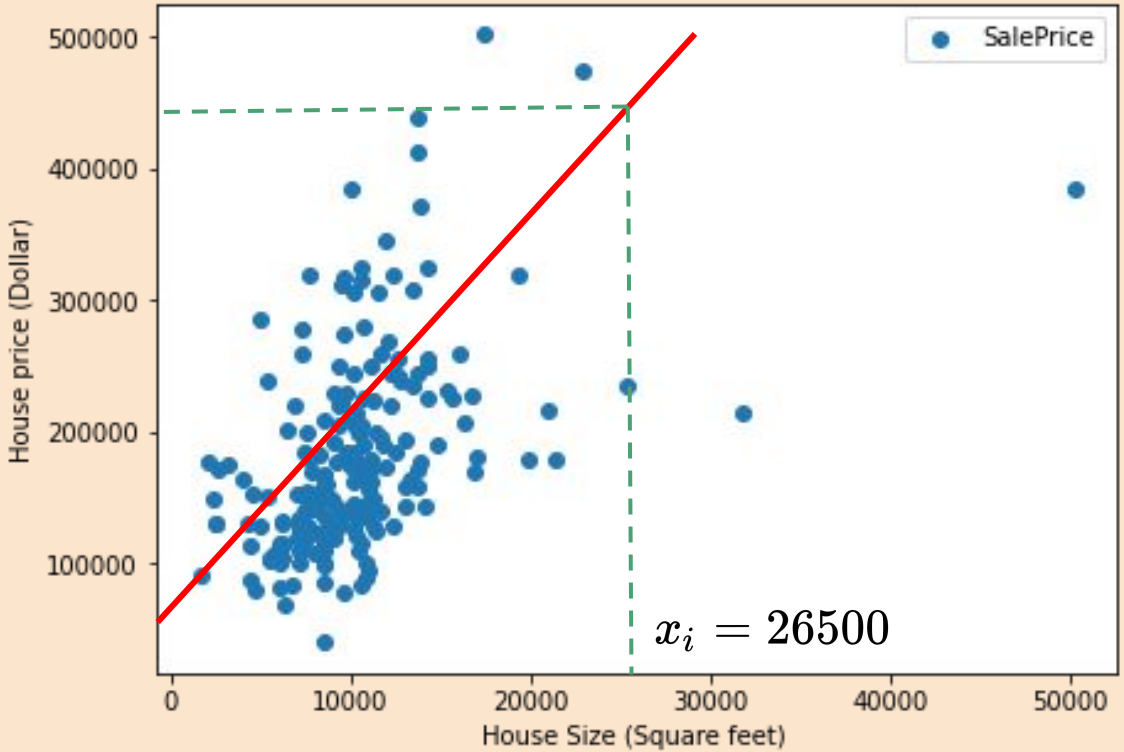


The basic example

$$Model : h(x_i) = 50000 + 15x_i$$

$$x_i = 26500$$

$$h(x_i = 26500) = 50000 + 15 \times 26500$$
$$= 447500$$



The basic example

$$Model: h(x_i) = 50000 + 15x_i$$

$$x_i = 26500$$

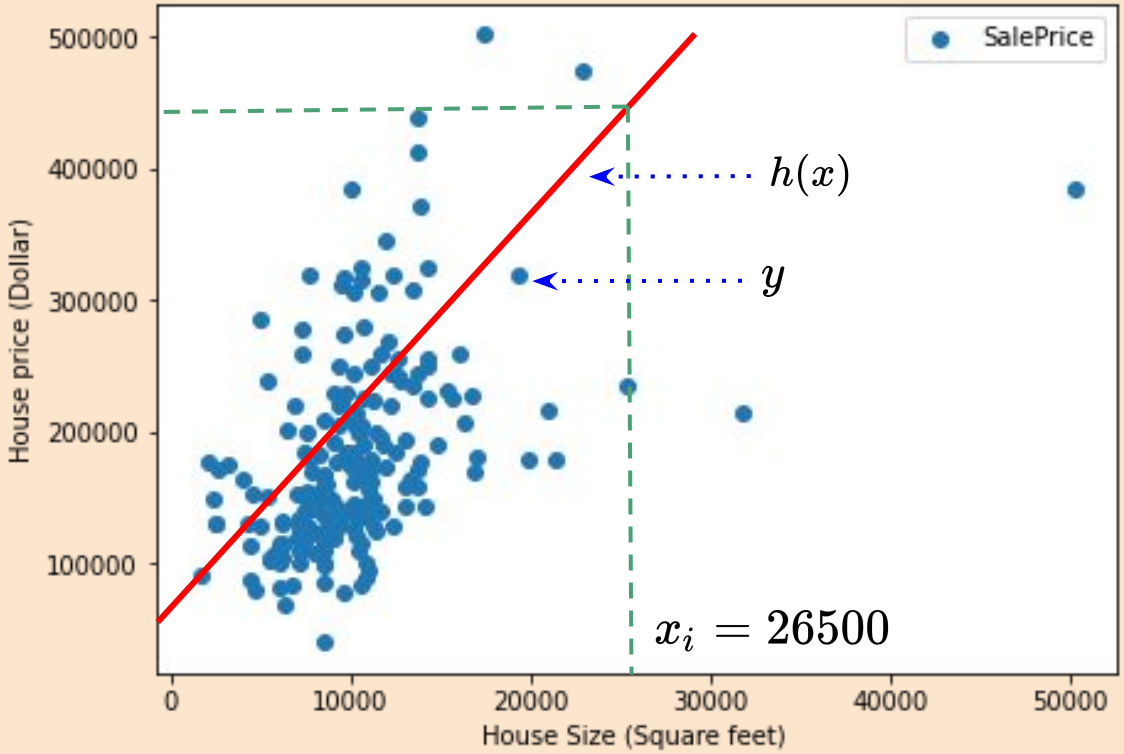
$$h(x_i = 26500) = 50000 + 15 \times 26500$$
$$= 447500$$

Notice that

$h(x)$ and y are not the same

$h(x)$: predicted values

y : actual values



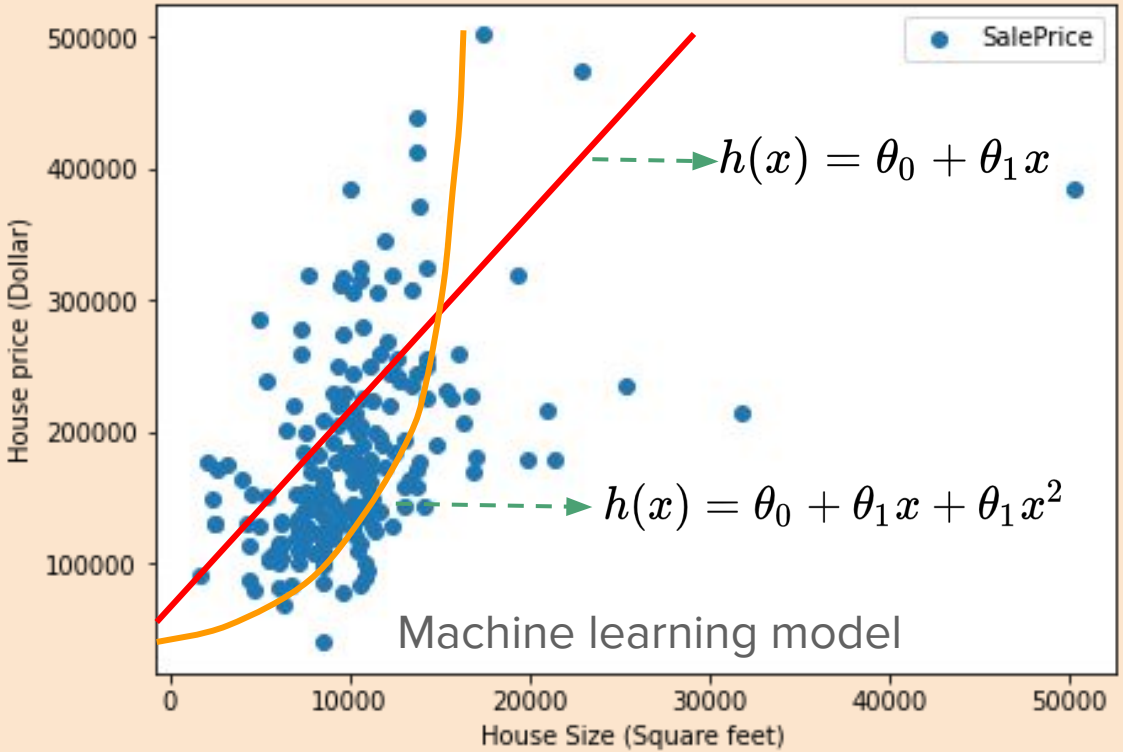
The basic example

$$h(x) = \theta_0 + \theta_1 x$$

$$h(x) = \theta_0 + \theta_1 x + \theta_1 x^2$$

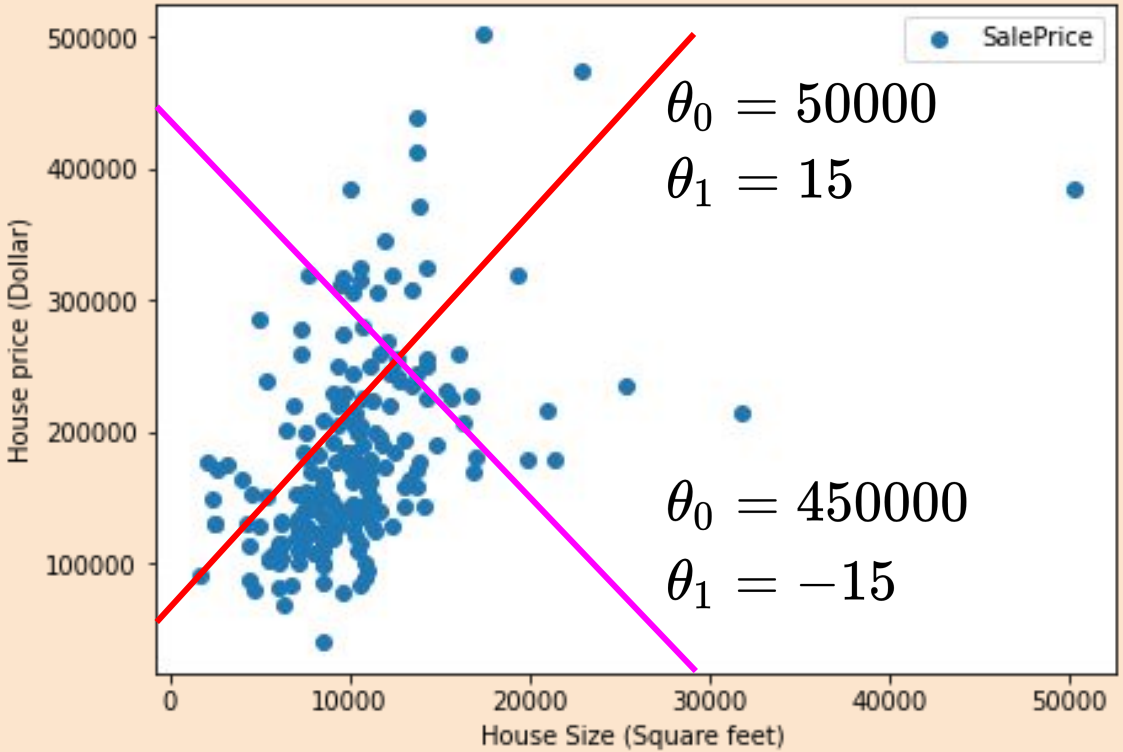
Model is a function of both features(x) and parameters (theta).

$$h(x; \theta)$$

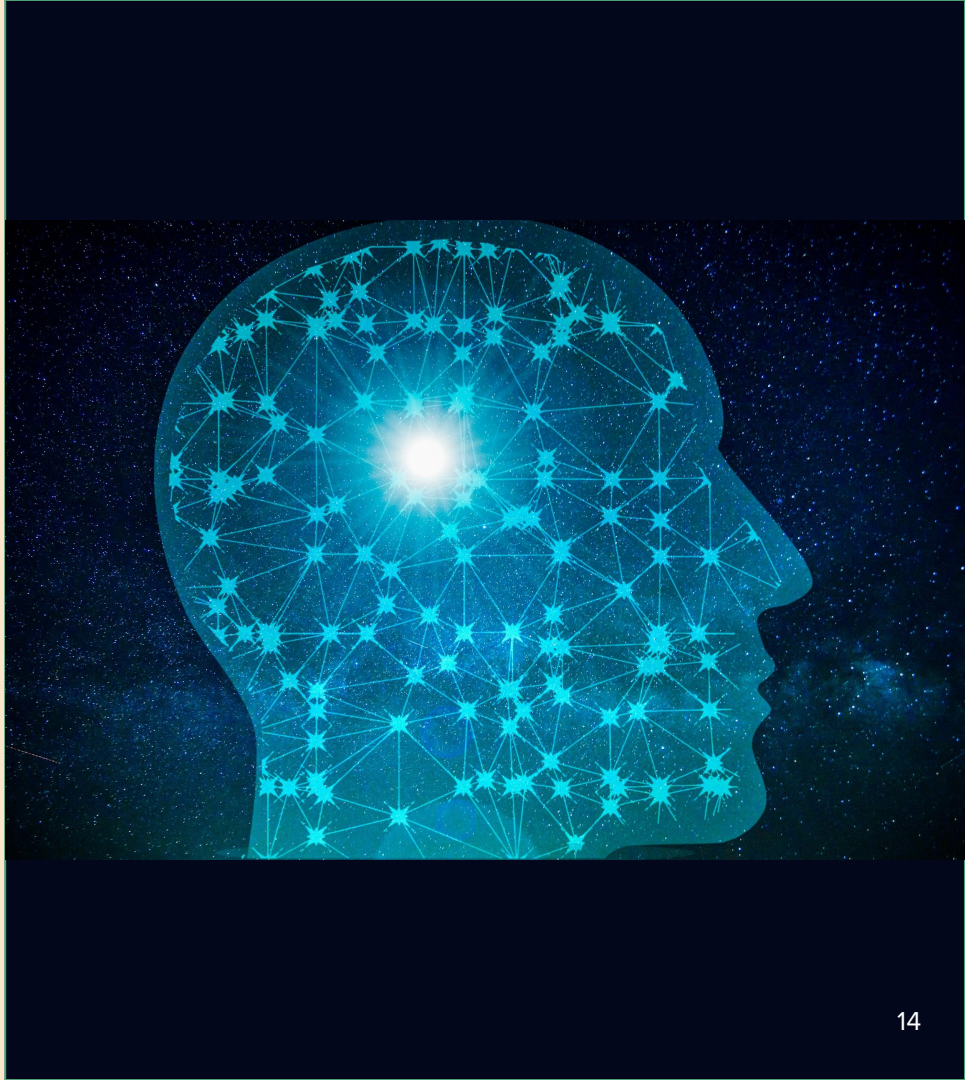


Good and bad models

$$h(x) = \theta_0 + \theta_1 x$$



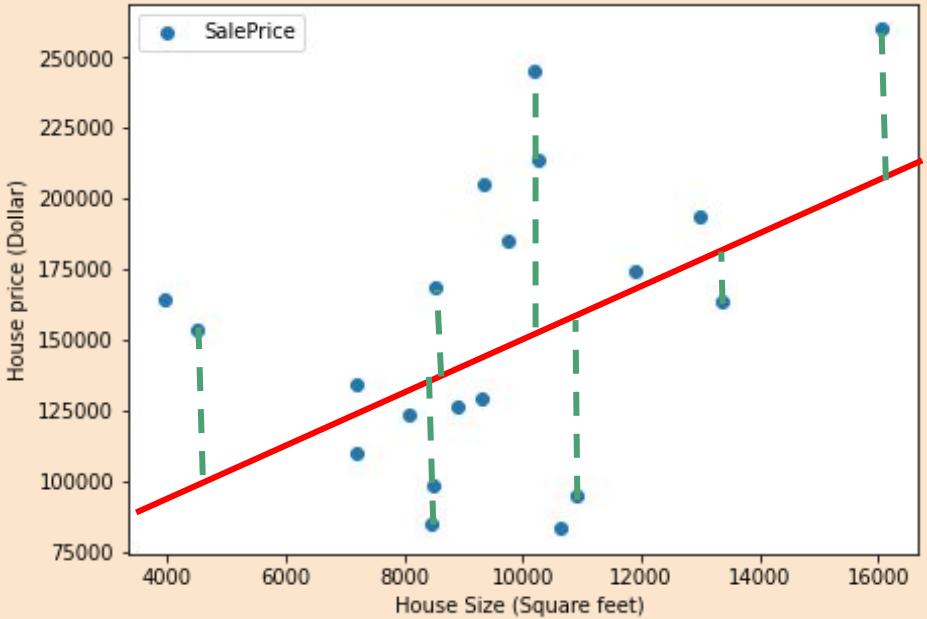
Cost Function



Cost function

Cost function (loss function) is a measure of whether a model is a good fit to the data.

For example, a famous cost function is called ‘squared error’ function that takes the difference between what you predict and the actual data value and square it.

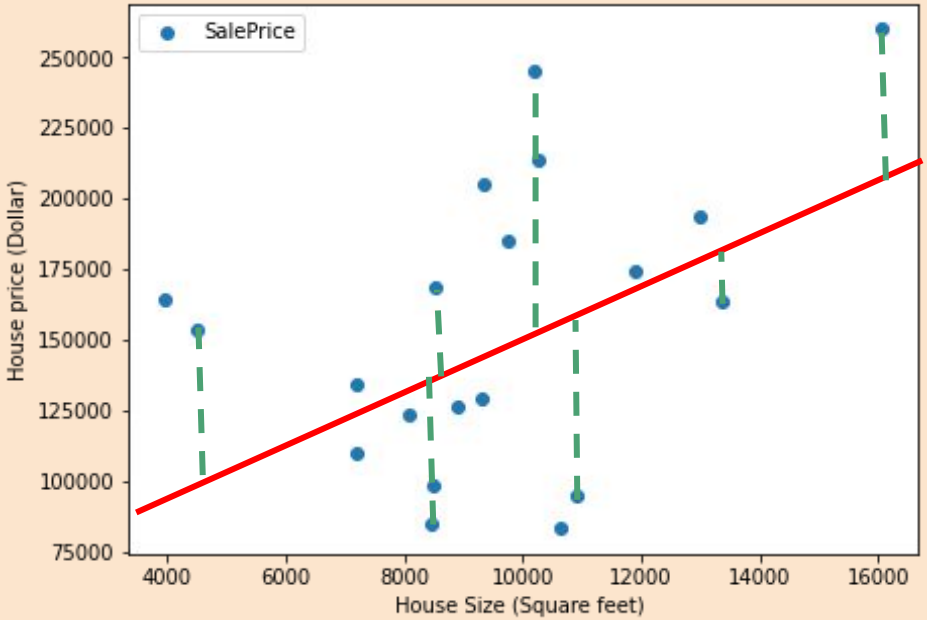


$$E = \sum_i (h(x_i) - y_i)^2$$
$$= Cost\ function$$

Cost function

Cost function (loss function) is a measure of whether a model is a good fit to the data.

For example, a famous cost function is called ‘squared error’ function that takes the difference between what you predict and the actual data value and square it.



$$E = \sum_i (h(x_i) - y_i)^2$$
$$= Cost\ function$$

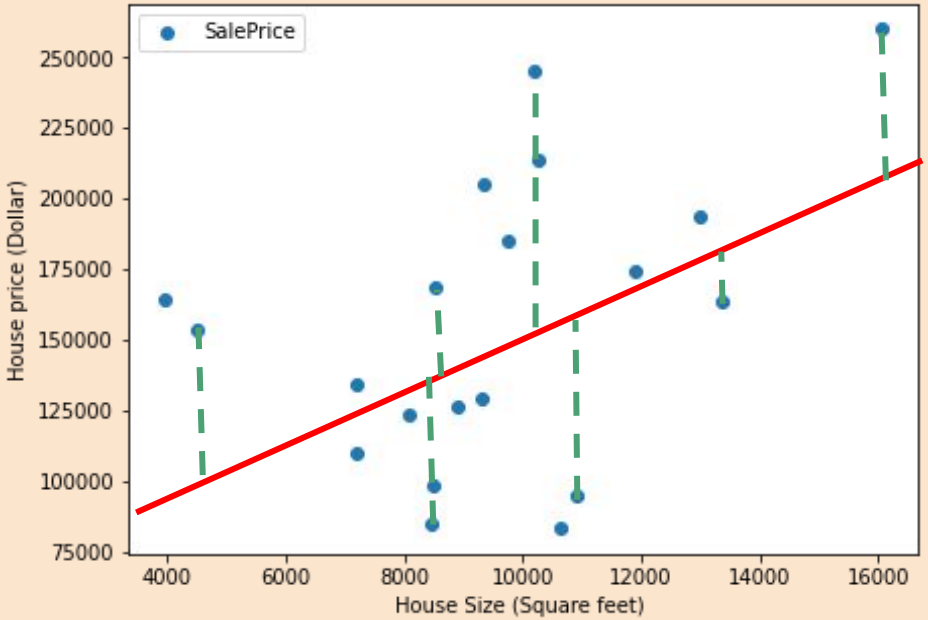
Low cost function ➔ good model

High cost function ➔ bad model

Cost function

Cost function (loss function) is a measure of whether a model is a good fit to the data.

For example, a famous cost function is called ‘squared error’ function that takes the difference between what you predict and the actual data value and square it.



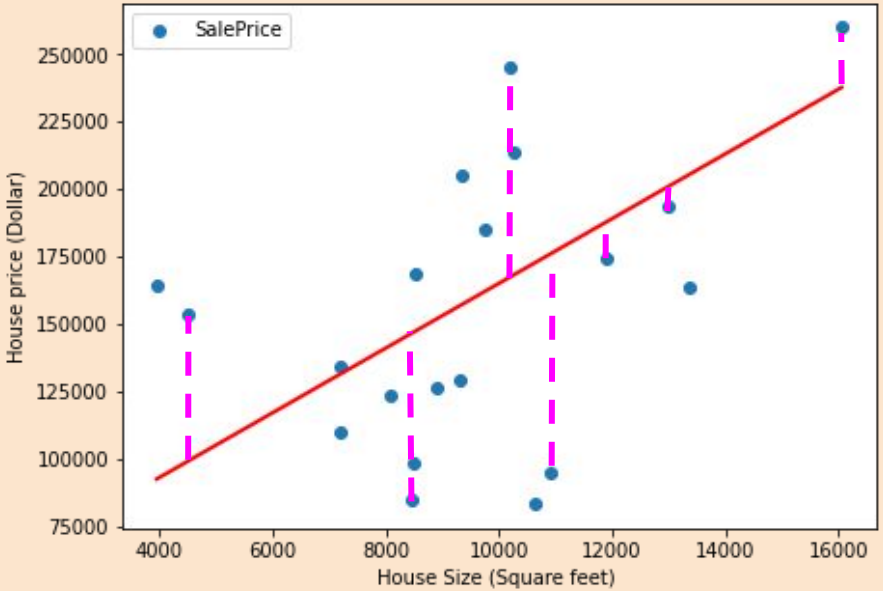
$$Root\ mean\ square\ error\ (RMSE)$$

$$Cost(\theta) = \sqrt{\frac{1}{m}\sum_i (h(x_i) - y_i)^2}$$

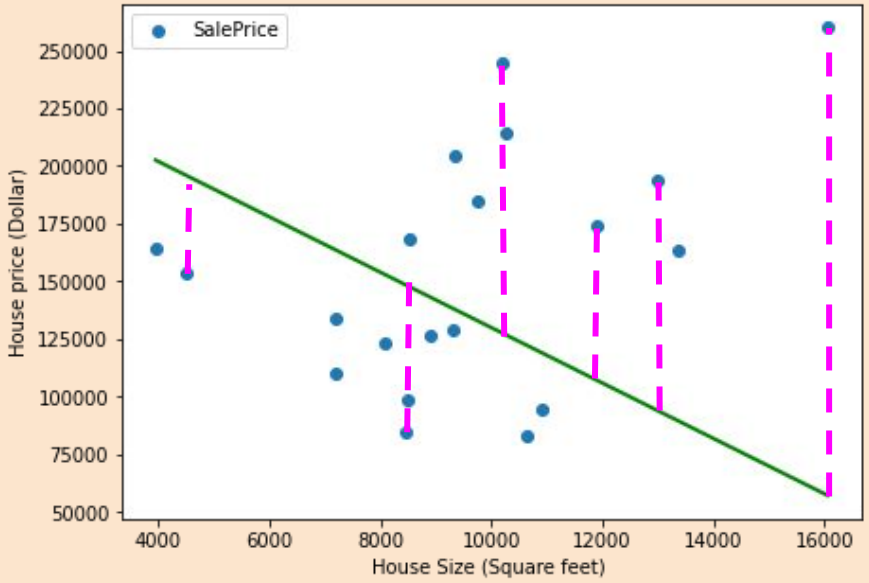
m : # of sample in the dataset

Lowering cost function

$$Model : h(x_i) = \theta_0 + \theta_1 x_i$$



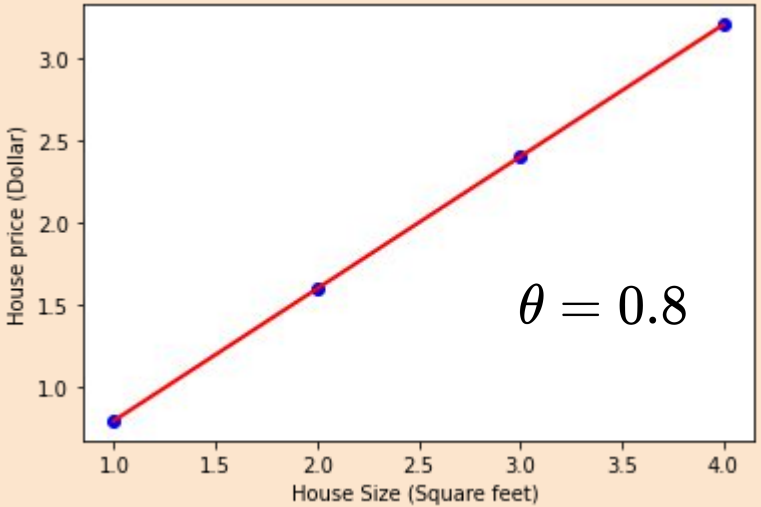
$$\theta_0 = 45000$$
$$\theta_1 = 12$$



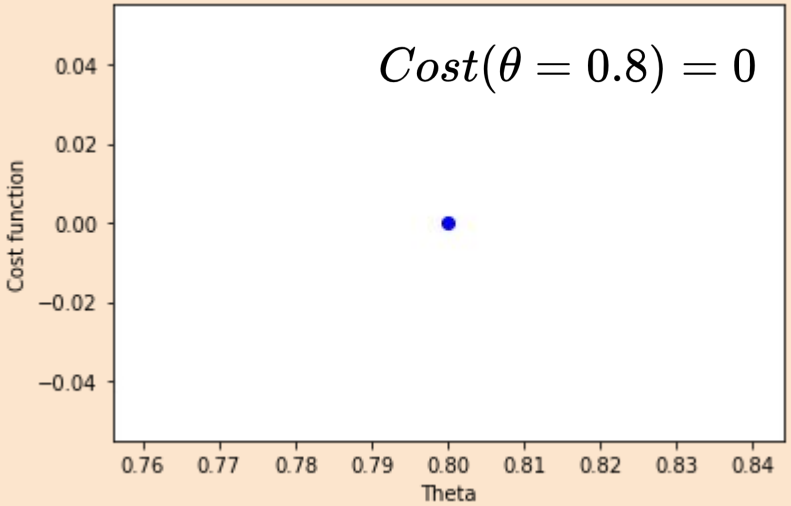
$$\theta_0 = 250000$$
$$\theta_1 = -12$$

Minimizing cost function

$Model : h(x) = \theta x$



$Cost(\theta)$



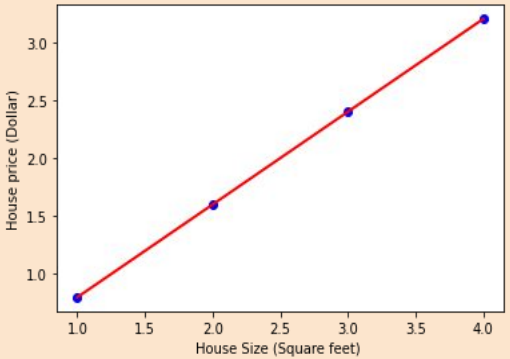
$x = 1, 2, 3, 4$

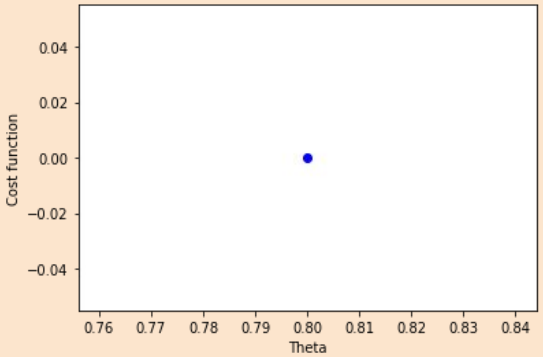
$y = 0.8, 1.6, 2.4, 3.2$

Minimizing cost function

$\theta = 0.8$

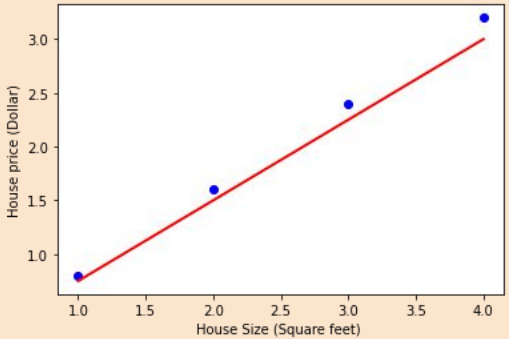
$Cost(\theta = 0.8) = 0$

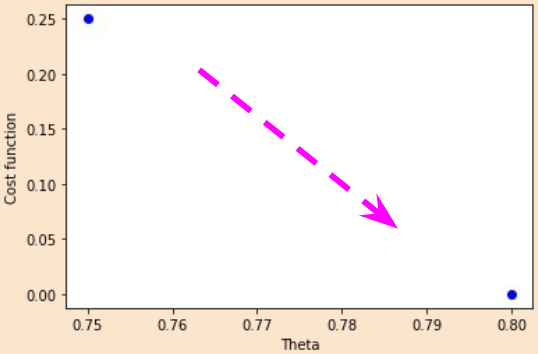




$\theta = 0.75$

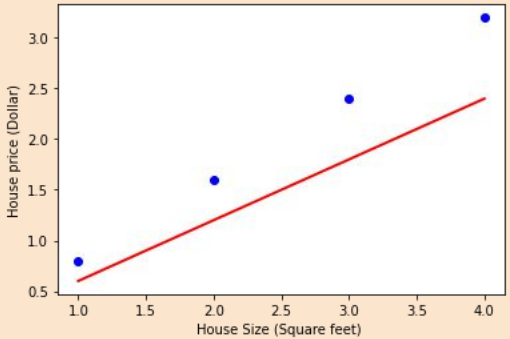
$Cost(\theta = 0.75) = 0.25$

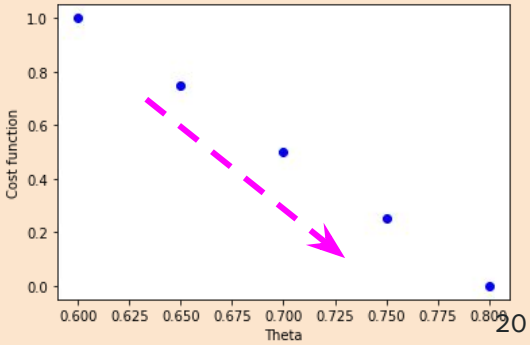




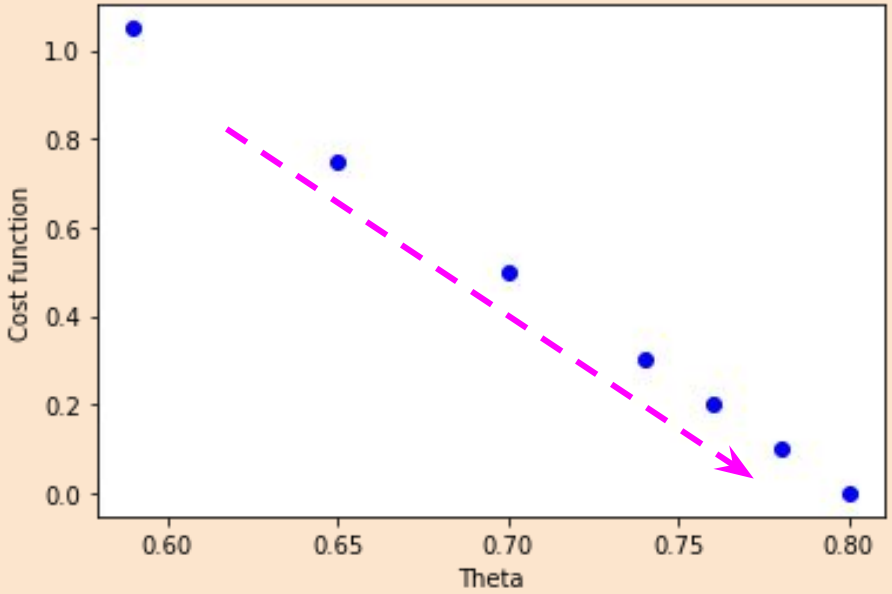
$\theta = 0.6$

$Cost(\theta = 0.6) = 1$



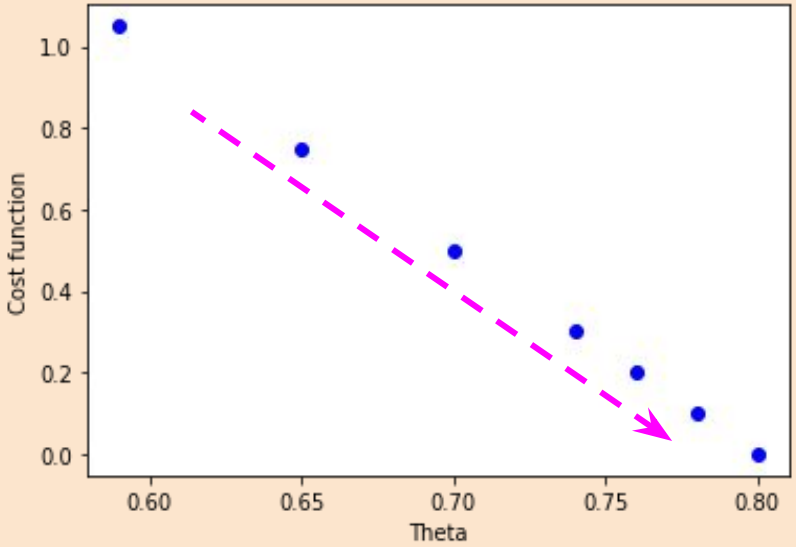


Minimizing cost function



- Best-fitted model has lowest cost function.
- Training machine learning models = minimizing cost function
- Often accomplished by using ‘optimization algorithms’

Minimizing cost function



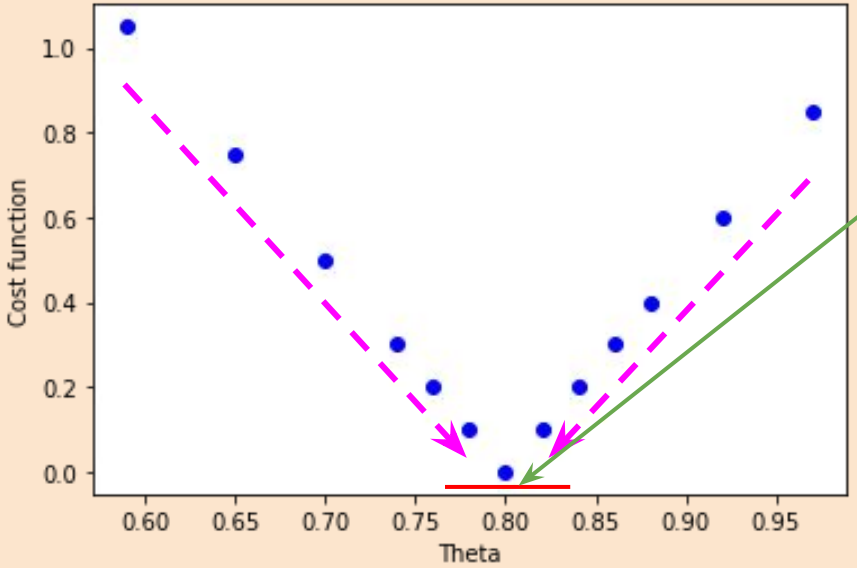
$$Model : h(x_i) = \theta x_i$$

$$Parameter \ : \ \theta$$

$$Cost\ function \ : \ Cost(\theta) = \sqrt{\frac{1}{m}\sum_i (h(x_i) - y_i)^2}$$

- Best-fitted model has lowest cost function.
- Training machine learning models = minimizing cost function
- Often accomplished by using ‘optimization algorithms’

Minimizing cost function



$$\frac{\partial Cost(\theta)}{\partial(\theta)} = 0$$

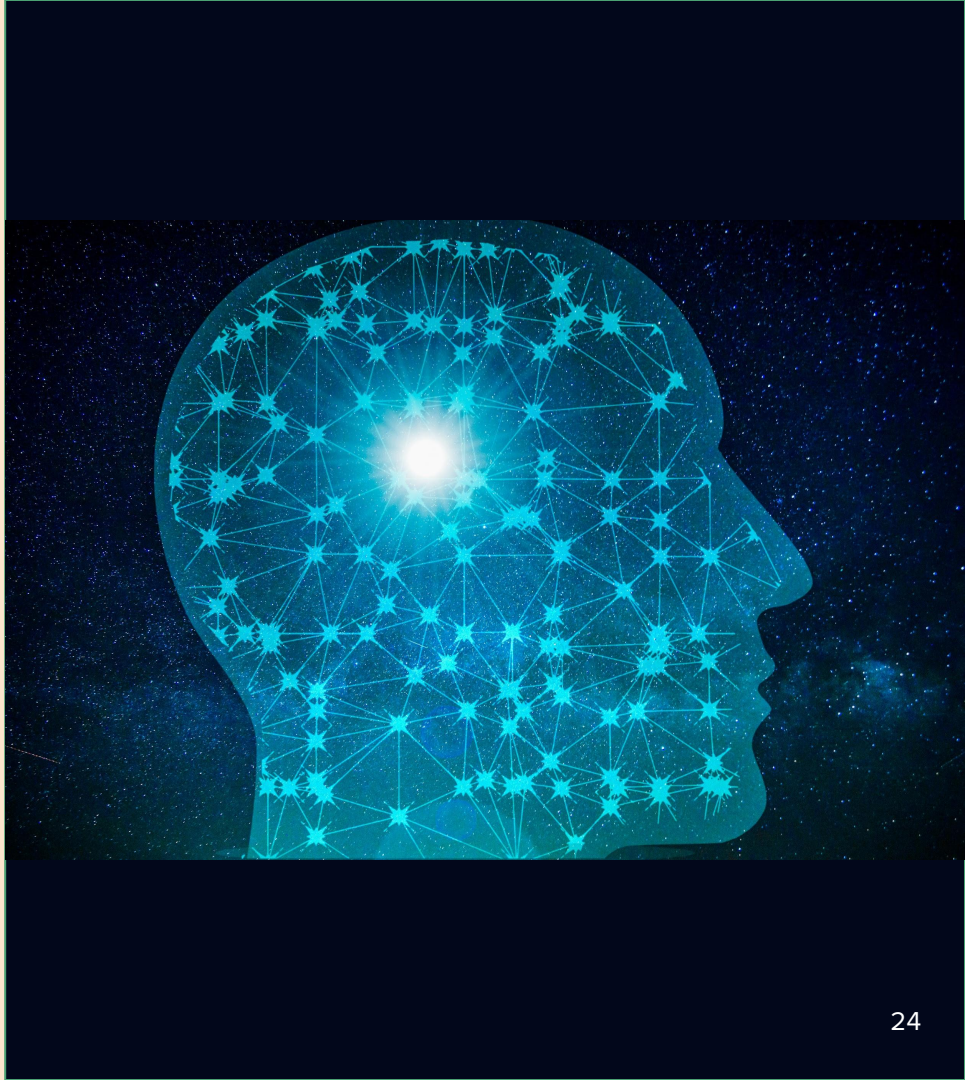
$$Cost(\theta) = \sum_i (h(x_i) - y_i)^2$$

$$= \sum_i (\theta x_i - y_i)^2$$

$$\frac{\partial Cost(\theta)}{\partial(\theta)} = \sum_i 2(\theta x_i - y_i)x_i = 0$$

$$\theta = 0.8$$

Over Fitting

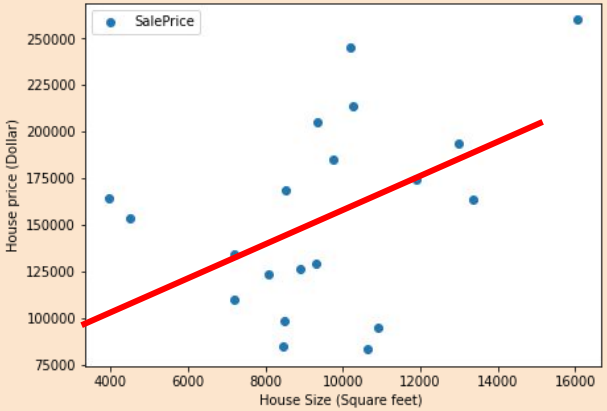


Overfitting

In machine learning, we need to minimize cost function, however sometimes, we have the opposite problem when the cost function is too low.

Underfit

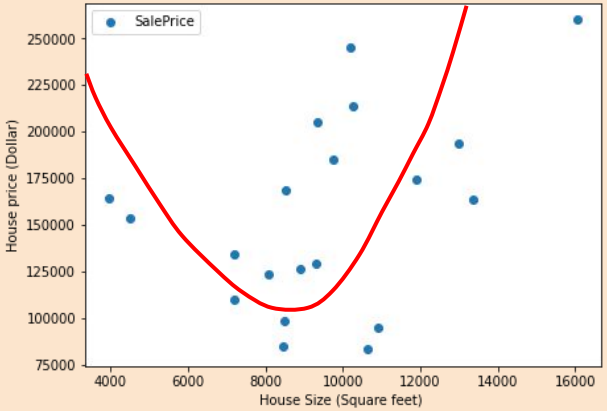
$Cost = 0.5$



$h(x) = \theta_0 + \theta_1 x$

Right

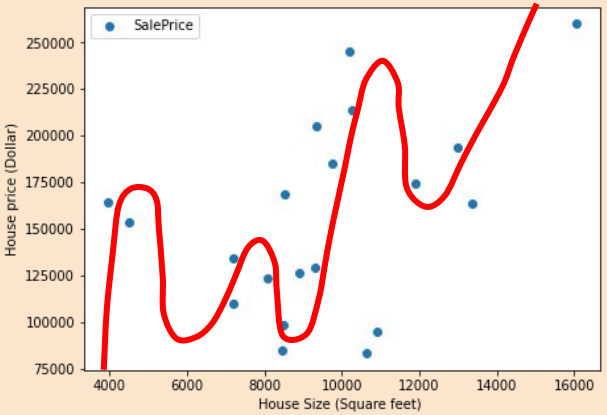
$Cost = 0.04$



$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

Overfit

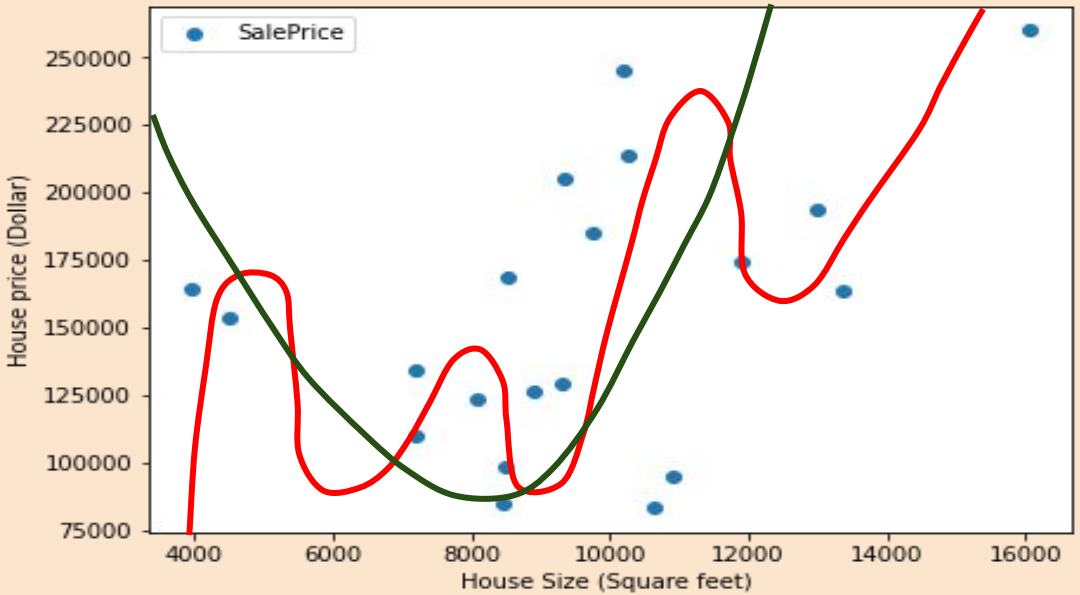
$Cost = 0.01$



$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_9 x^9$

Overfitting

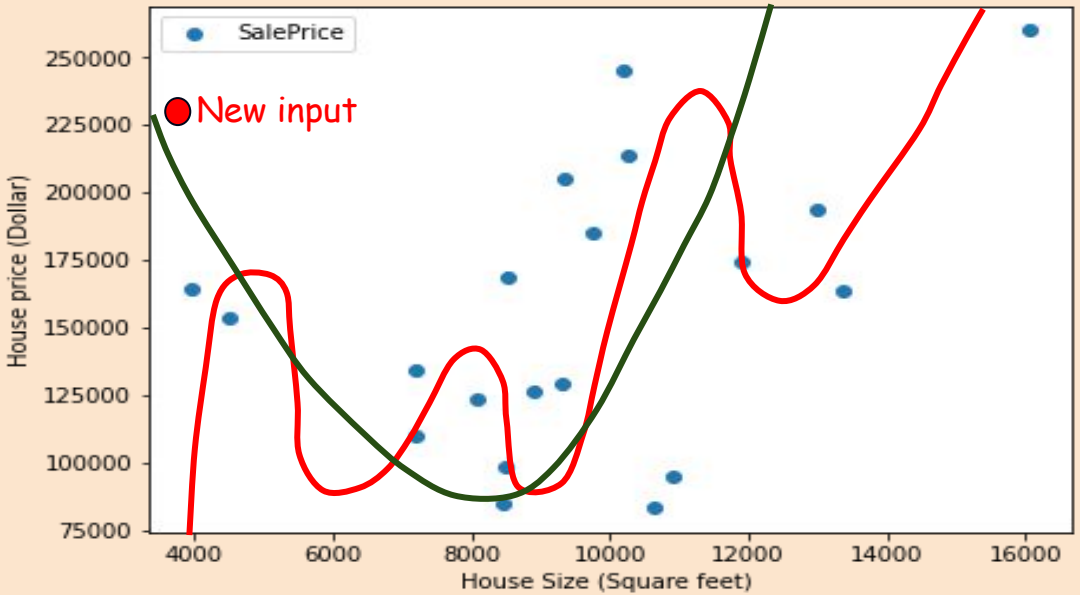
In machine learning, we need to minimize cost function, however sometimes, we have the opposite problem when the cost function is too low.



- Training data: good predictions
- Data outside training set: bad predictions

Overfitting

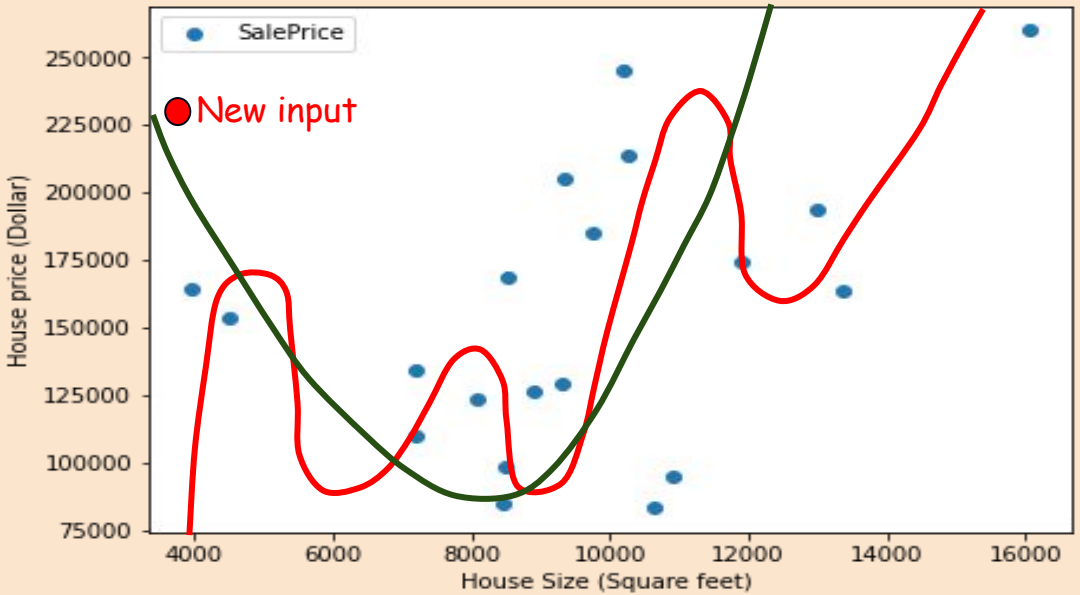
In machine learning, we need to minimize cost function, however sometimes, we have the opposite problem when the cost function is too low.



- Training data: good predictions
- Data outside training set: bad predictions

Overfitting

In machine learning, we need to minimize cost function, however sometimes, we have the opposite problem when the cost function is too low.

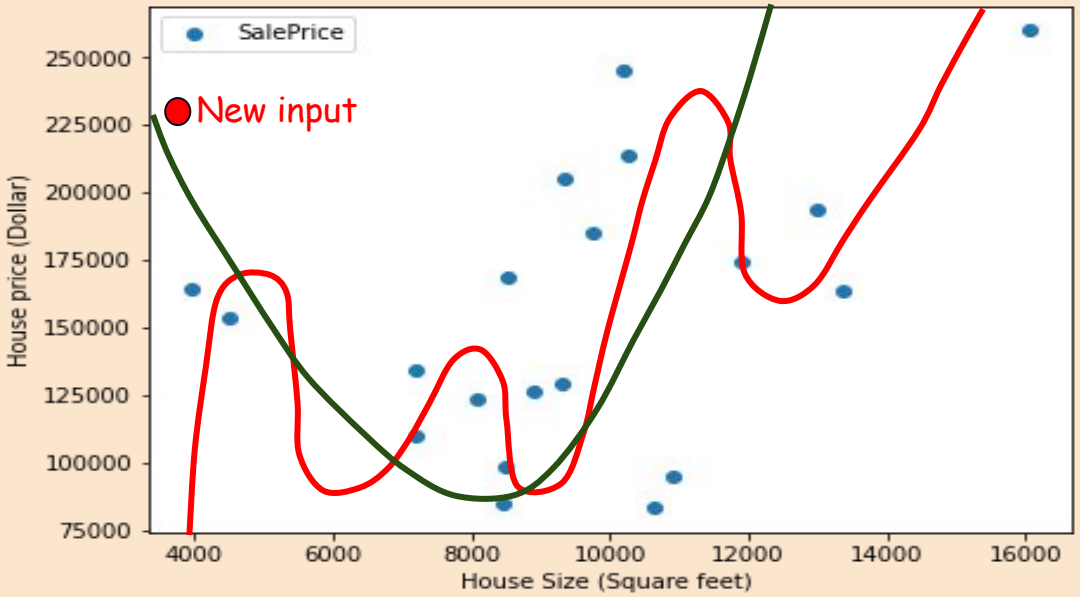


Overfitting model

- Signal to noise
- Memorize noize

Overfitting

In machine learning, we need to minimize cost function, however sometimes, we have the opposite problem when the cost function is too low.

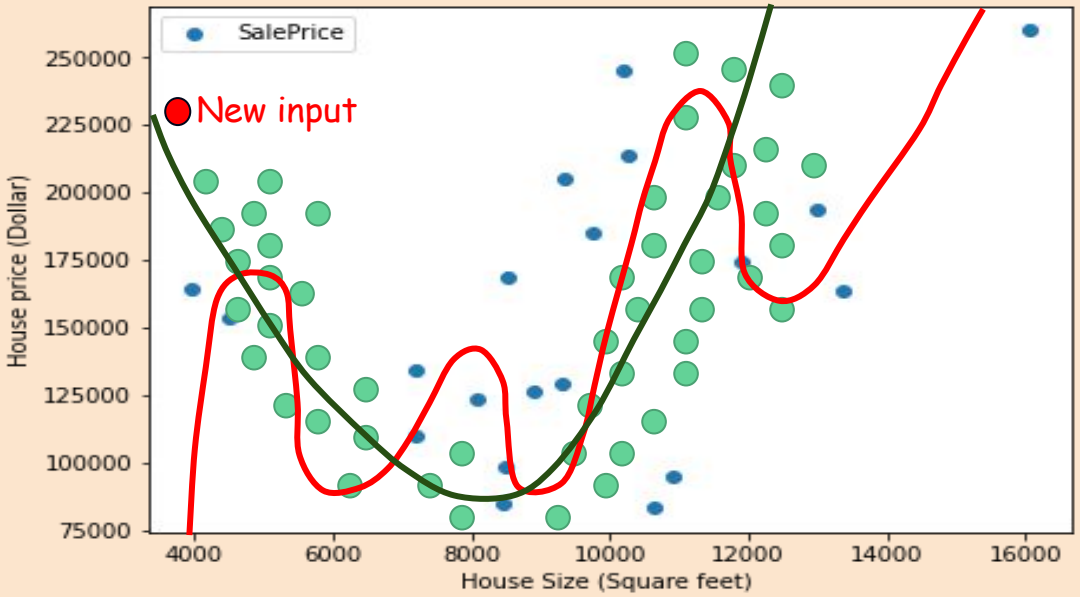


Fact about over fitting

- The more parameters you have, the more likely your model will overfit.
- The more data you have (compared to parameters) the less likely your model will overfit, because noise will be more likely to average out.

Overfitting

In machine learning, we need to minimize cost function, however sometimes, we have the opposite problem when the cost function is too low.

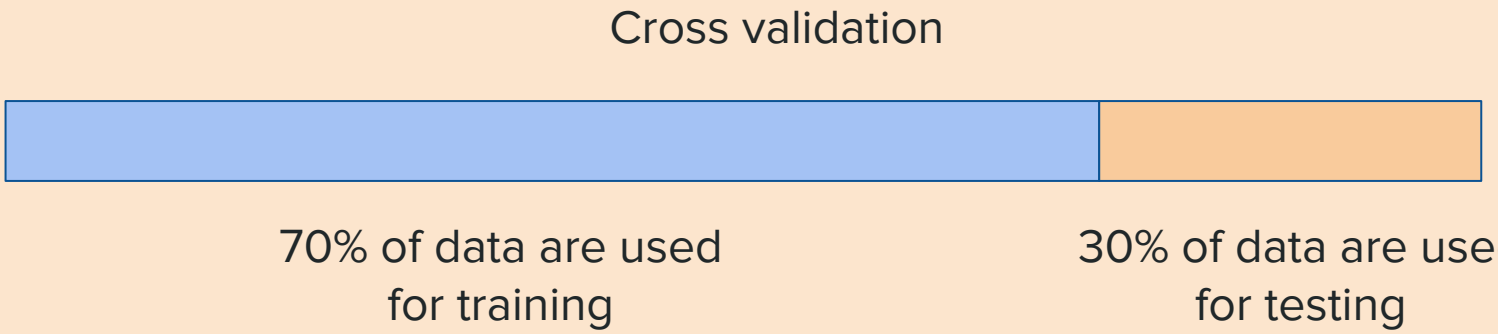


Fact about over fitting

- The more parameters you have, the more likely your model will overfit.
- The more data you have (compared to parameters) the less likely your model will overfit, because noise will be more likely to average out.

Avoid overfitting

Cross-validate your models: train models with one set of data (training set) and test them with another set of data (test set).

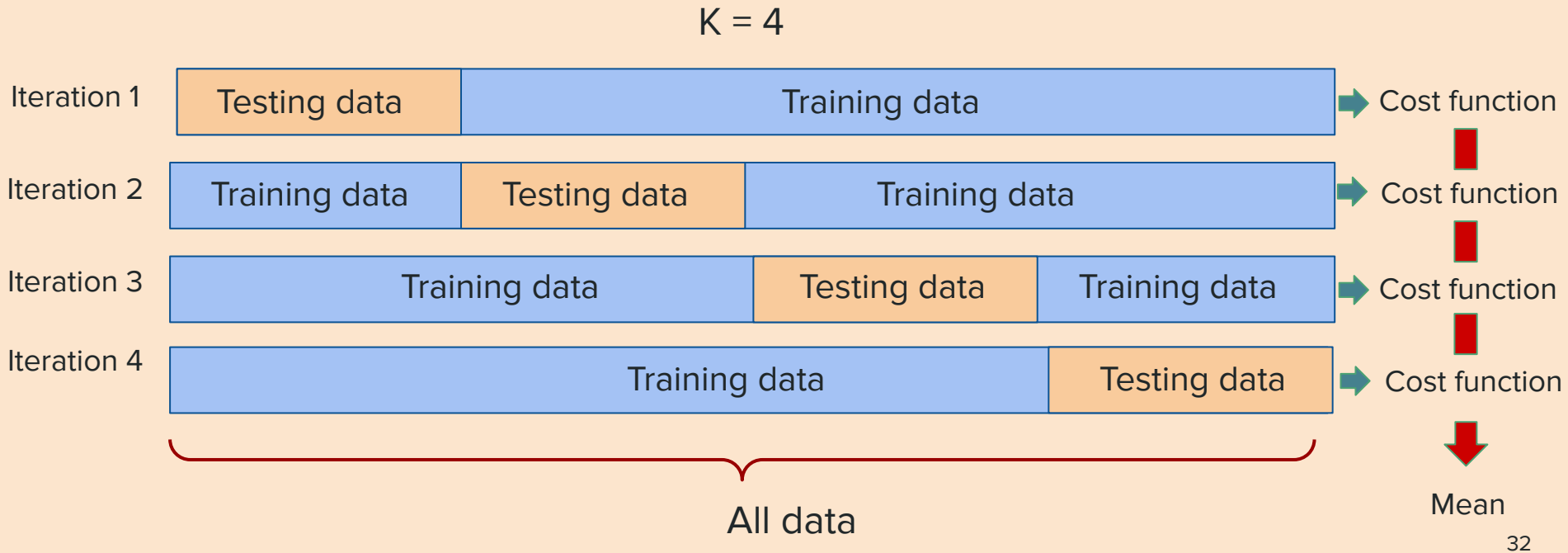


Overfitting model

- Train data → low cost but test data → high cost

Avoid overfitting

- F-fold cross validation: divide data into K chunks.
- Fit the model K times, each time use one chunk as test data, the rest as training data.



Overfitting

- Get more data
- Reduce the number of features (select only a few features that are more powerful)
- Reduce the number of parameters (using an algorithm to remove parameters that not necessary)
- Constrain the parameter set in the optimization process (regularization)

Regularization

$$Cost(\theta_0, \theta_1, \ldots, \theta_j) = \frac{1}{m} \sum_i (h(x_i) - y_i)^2 + \lambda \sqrt{\sum_j ||\theta_j||^2}$$

- Regularization is a mathematical way to reduce overfitting automatically, by limiting the influence of each feature.
- To minimize this cost function, you have to minimize both first and second terms.
- Minimize the second term means less overfitting.

Regularization

Error term

Regularization term

$$Cost(\theta_0, \theta_1, \ldots, \theta_j) = \frac{1}{m} \sum_i (h(x_i) - y_i)^2 + \lambda \sqrt{\sum_j ||\theta_j||^2}$$

The norm of cost function

The norm of parameter vector

- Regularization is a mathematical way to reduce overfitting automatically, by limiting the influence of each feature.
- To minimize this cost function, you have to minimize both first and second terms.
- Minimize the second term means less overfitting.

Regularization

Error term

Regularization term

$$Cost(\theta_0, \theta_1, \ldots, \theta_j) = \frac{1}{m} \sum_i (h(x_i) - y_i)^2 + \lambda \sqrt{\sum_j ||\theta_j||^2}$$

The norm of cost function

The norm of parameter vector

- Lambda is called ‘regularization parameter’
- Because we adjust lambda to adjust the weight of the two cost function terms
- High lambda: severely limit parameter size
- Low lambda: allow parameters to scale up more freely, give more importance to lowering error.

Regularization

Error term

Regularization term

$$Cost(\theta_0, \theta_1, \ldots, \theta_j) = \frac{1}{m} \sum_i (h(x_i) - y_i)^2 + \lambda \sqrt{\sum_j ||\theta_j||^2}$$

The norm of cost function

The norm of parameter vector

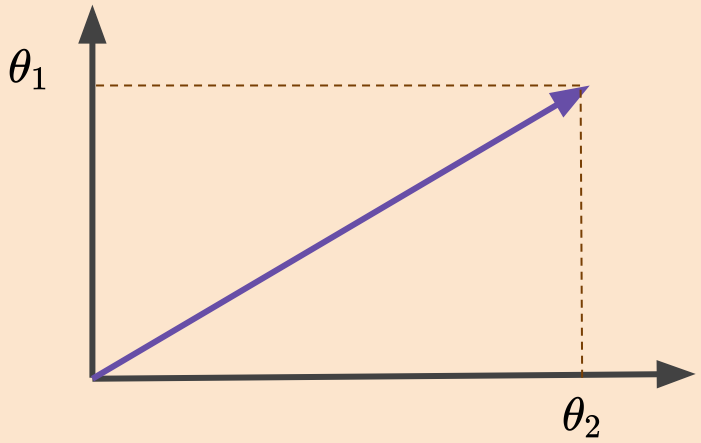
Example

Model overfit at lambda = 0.0001 ➔ adjust lambda =0.0005 ➔ less overfitting

Regularization

$$Cost(\theta_0, \theta_1, \ldots, \theta_j) = \frac{1}{m} \sum_i (h(x_i) - y_i)^2 + \lambda \sqrt{\sum_j ||\theta_j||^2}$$

The norm of parameter vector

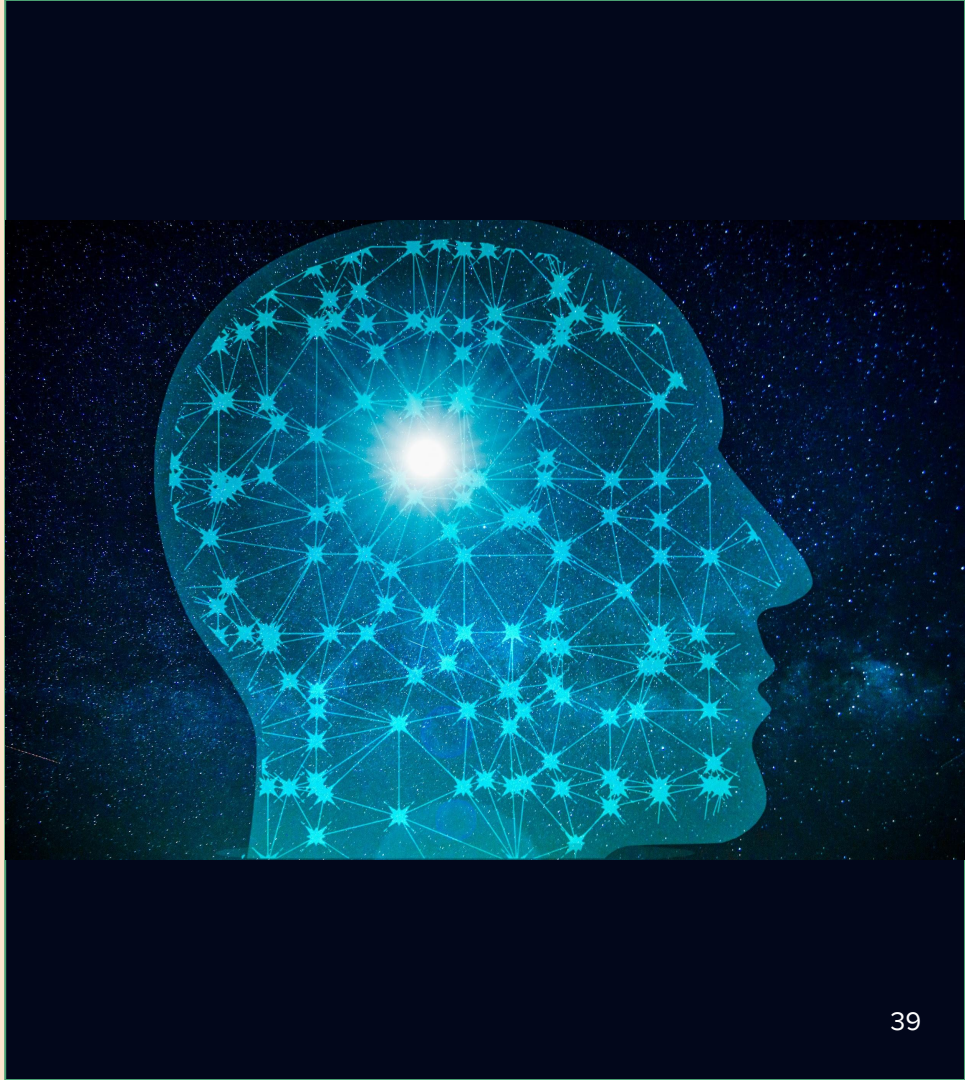


Euclidean norm

$$norm = \sqrt{\theta_1^2 + \theta_2^2}$$

$$= magnitude\ of\ < \theta_1, \theta_2 >$$

Linear Regression



Linear regression model

Output

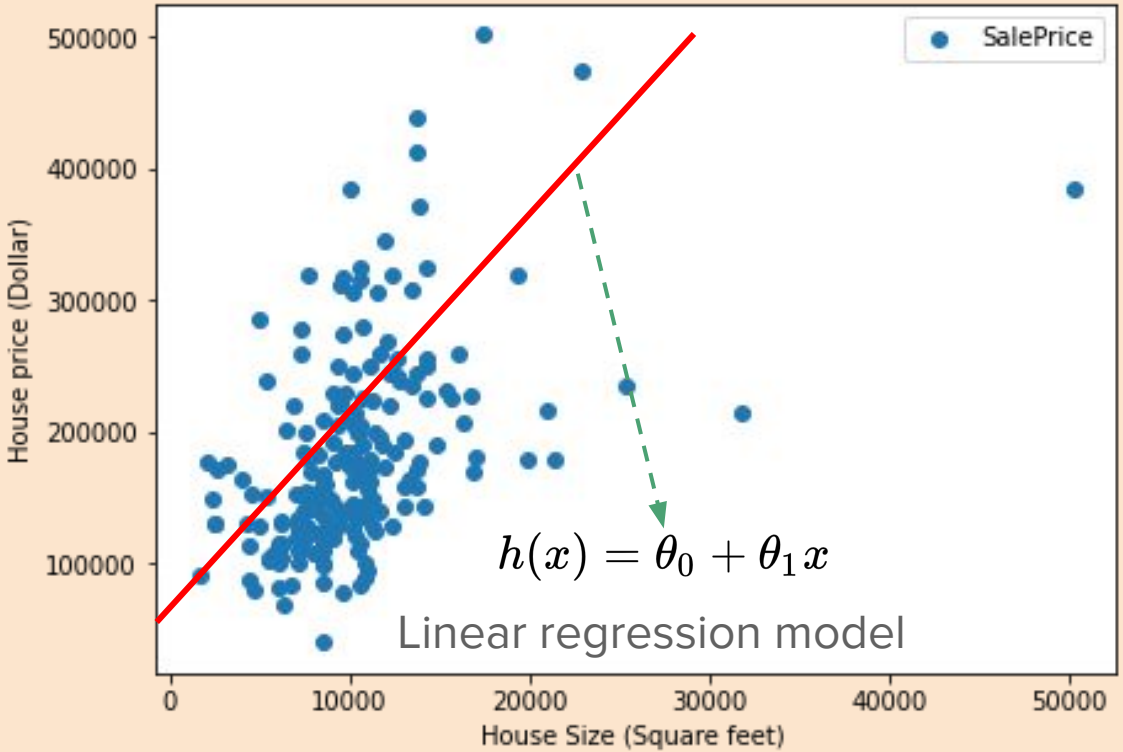
Input

$$h(x) = \theta_0 + \theta_1 x$$

Parameters of the model

$\theta_0 = 50000$

$\theta_1 = 15$



Linear regression model

$$h(\vec{x}) = \underbrace{\theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n}_{\text{Regression}}$$

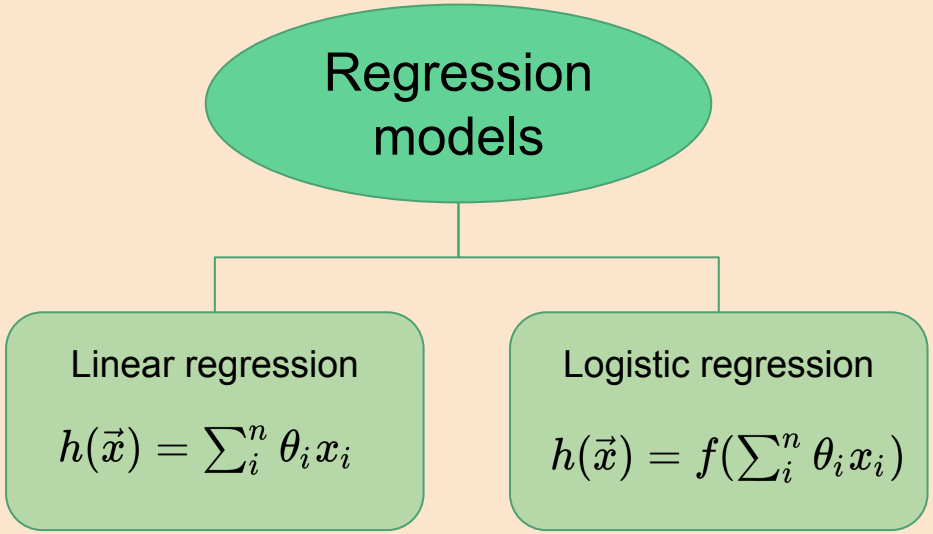
$$= \sum_i^n \theta_i x_i = \vec{\theta}^T \vec{x}$$

where $\vec{x} = [1, x_1, x_2, \ldots, x_n]^T$

$\vec{\theta} = [\theta_0, \theta_1, \ldots, \theta_n]^T$

n is the number of features

$\vec{x}, \vec{\theta} \in \Re^{n+1 \times n}$



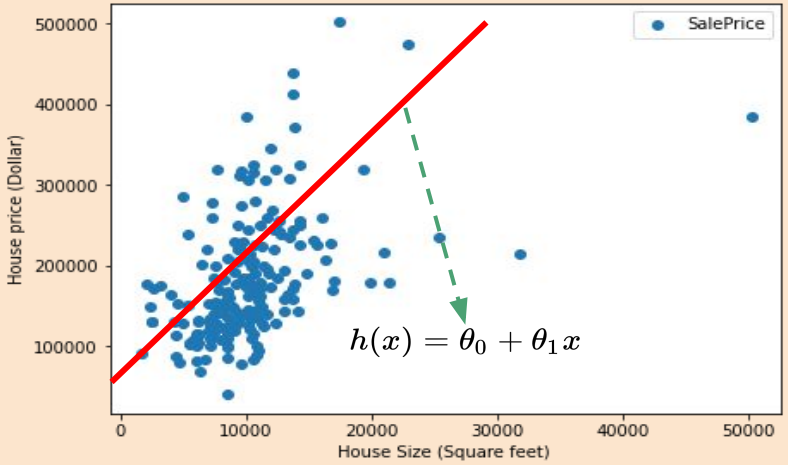
Linear regression model

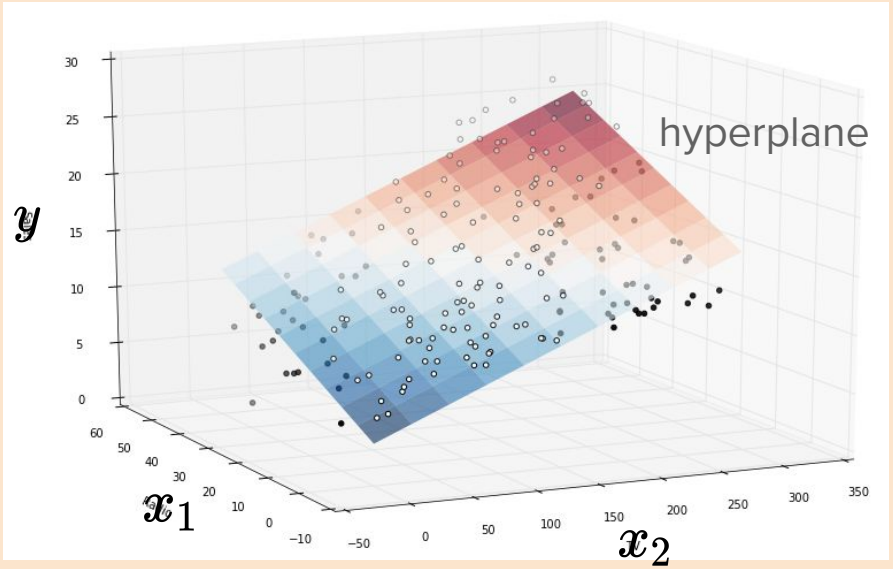
$$h(\vec{x}) = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$$

$$= \sum_i^n \theta_i x_i = \vec{\theta}^T \vec{x}$$

where $\vec{x} = [1, x_1, x_2, \ldots, x_n]^T$

$\vec{\theta} = [\theta_0, \theta_1, \ldots, \theta_n]^T$





$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

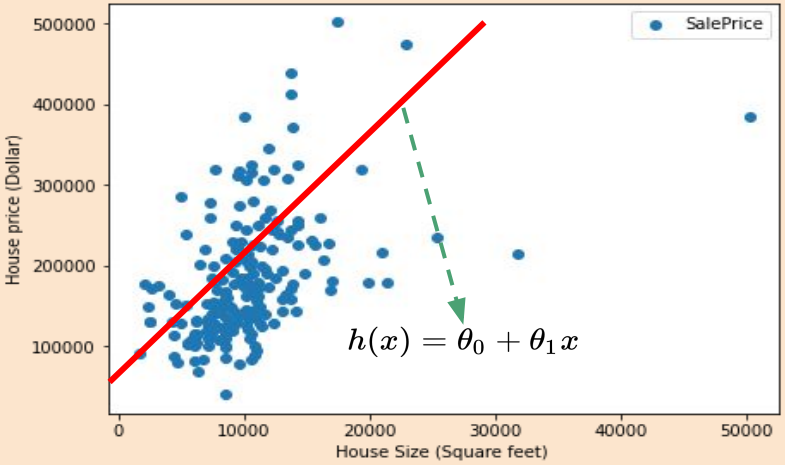
Linear regression model

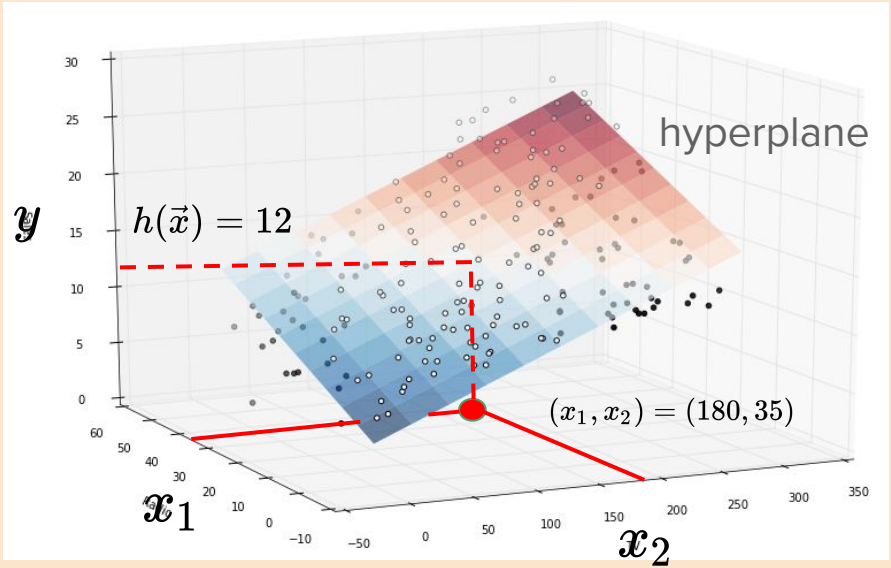
$$h(\vec{x}) = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$$

$$= \sum_i^n \theta_i x_i = \vec{\theta}^T \vec{x}$$

where $\vec{x} = [1, x_1, x_2, \ldots, x_n]^T$

$\vec{\theta} = [\theta_0, \theta_1, \ldots, \theta_n]^T$

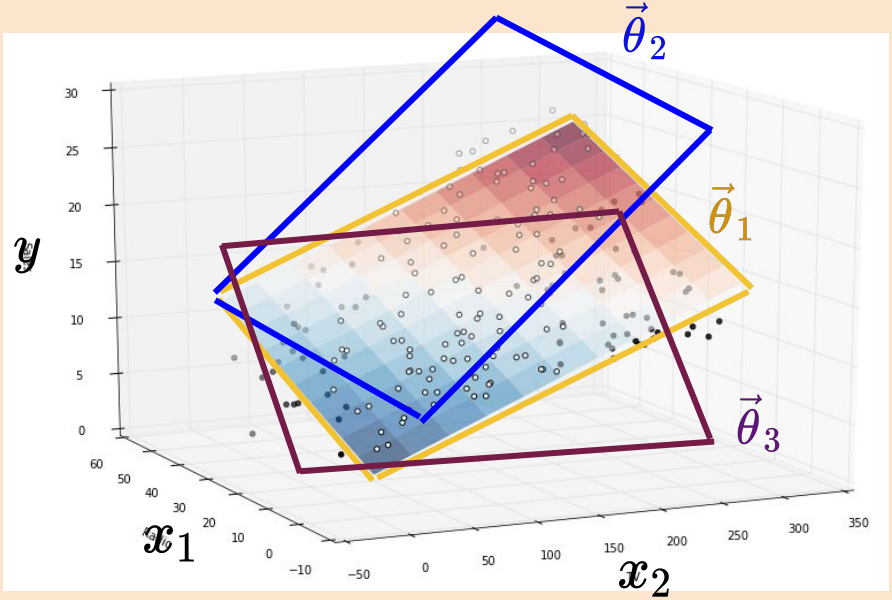




$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

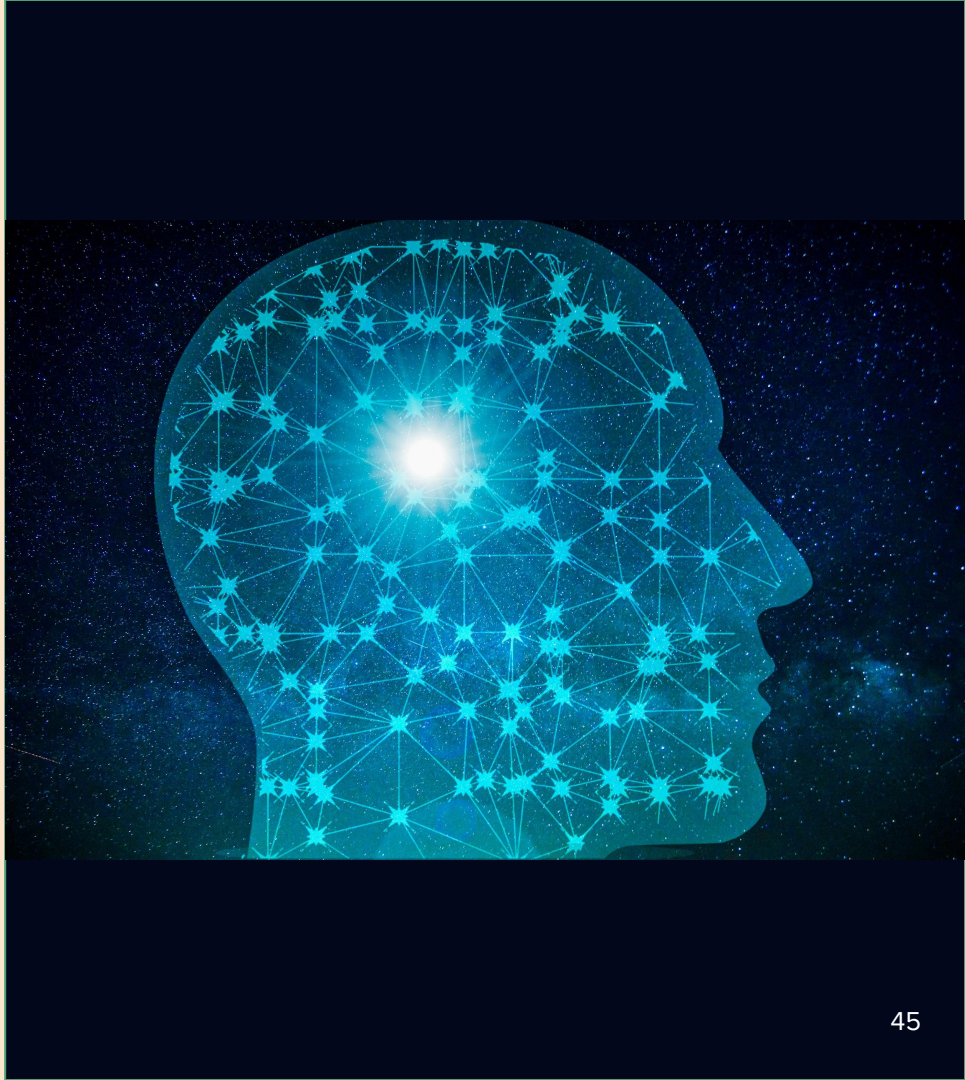
Linear regression model

- Cost function: Mean square error (MSE), Root mean square error (RMSE)
- Find $\vec{\theta}$ to minimize cost function



$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Regression Performance



Regression performance

- R-squared: how close the data are to the fitted regression line.
- R-squared = the percentage of the response variable variation that is explained by a linear model.

R-squared = Explained variation / Total variation

Regression performance

- R-squared: how close the data are to the fitted regression line.
- R-squared = the percentage of the response variable variation that is explained by a linear model.

$$Explained\ variation = Total\ variance - \sum_i (y_i - h(\vec{x}_i))^2$$

(*Total variation*)

R-squared = Explained variation / Total variation

$$Total\ variation = Variance\ of\ y = s^2 = \sum_i (y_i - \bar{y})^2$$

Regression performance

- R-squared: how close the data are to the fitted regression line.
- R-squared = the percentage of the response variable variation that is explained by a linear model.

R-squared = Explained variation / Total variation

- R-squared → 0 (model explains none of the variability of data) → bad model
- R-squared → 1 (model explains all of the variability of data) → good model

Regression performance

Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

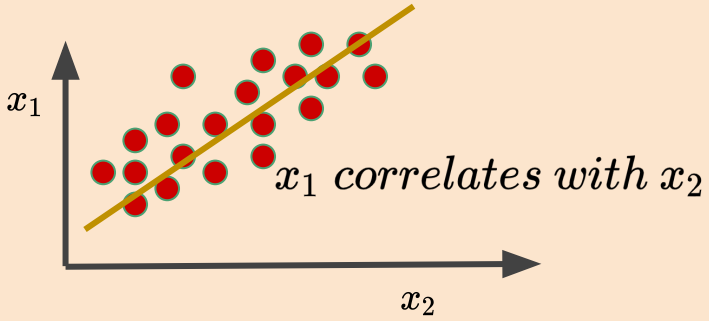
where $e_t = (y_t - \hat{y}_t)$

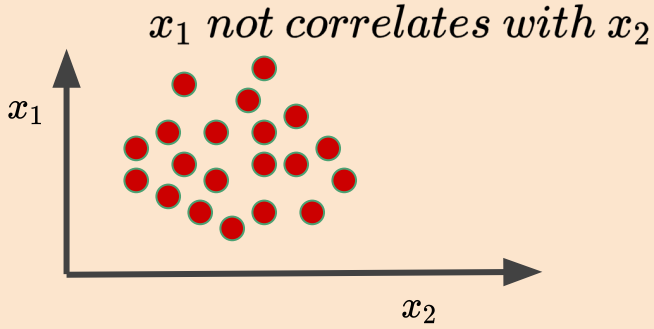
Regression remark

- We use mean square error or root mean square error as a cost function of linear regression.
- Remember that linear regression model assume all features and labels are **normally distributed**.
- All features should have **equal variance** and are **not correlated** it each other.

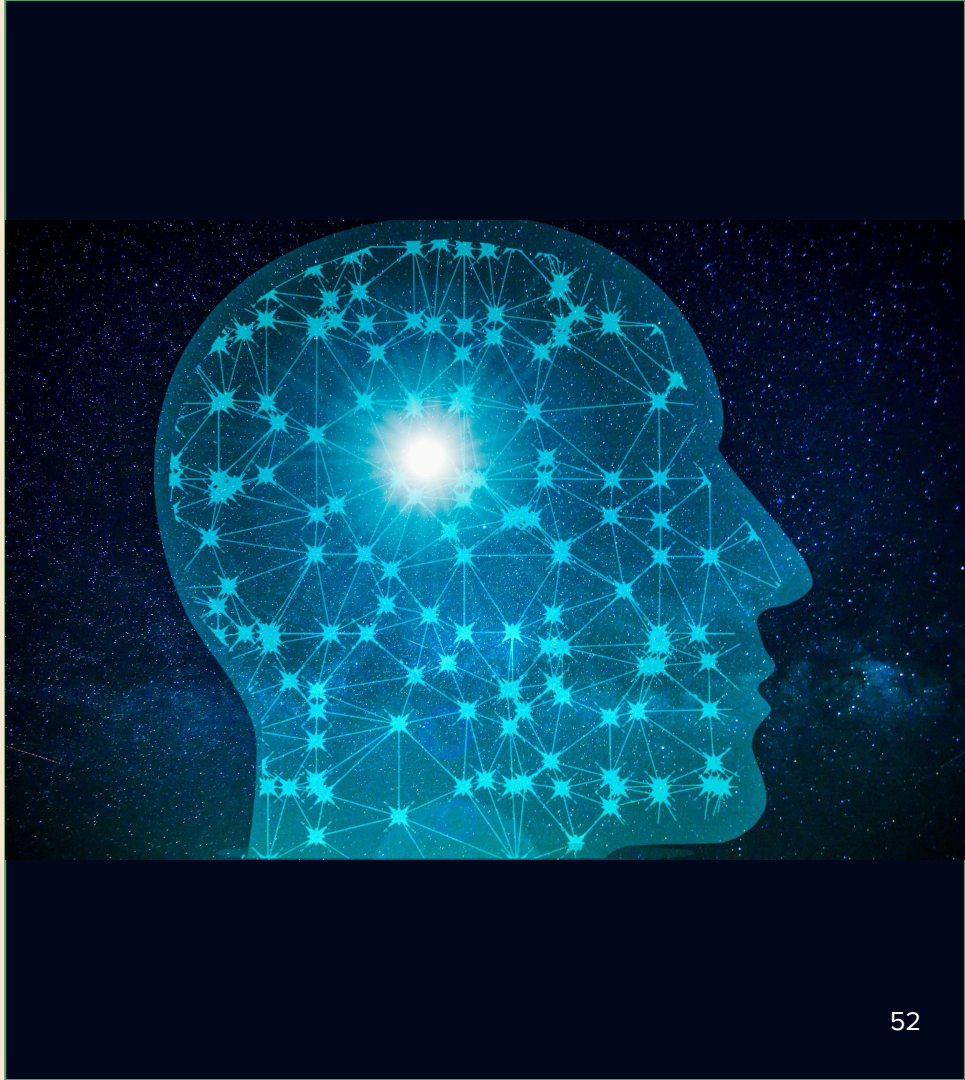
Regression remark

- We use mean square error or root mean square error as a cost function of linear regression.
- Remember that linear regression model assume all features and labels are **normally distributed**.
- All features should have **equal variance** and are **not correlated** it each other.





Logistic Regression



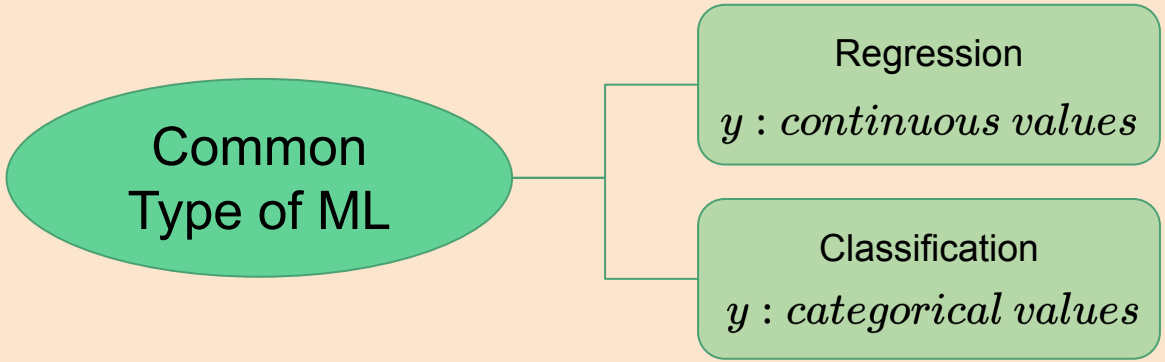
Logistic regression model

Logistic regression is similar to linear regression, however it is used for classification problem.

$$h(\vec{x}) = f(\theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n)$$

where $\vec{x} = [1, x_1, x_2, \ldots, x_n]^T$ $\vec{\theta} = [\theta_0, \theta_1, \ldots, \theta_n]^T$ n is the number of features

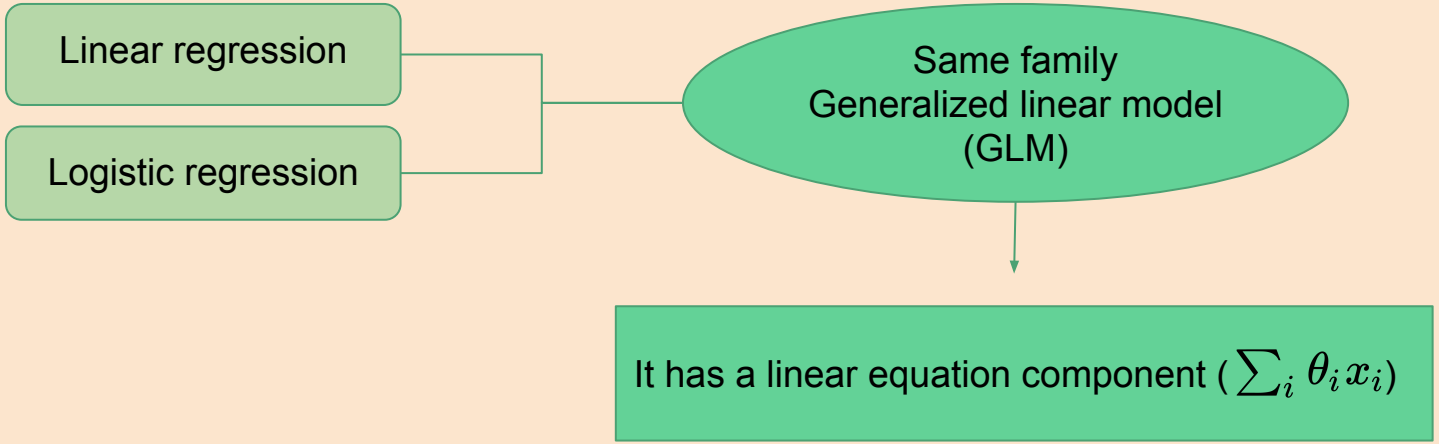
$\vec{x}, \vec{\theta} \in \Re^{n+1 \times n}$



Logistic regression model

Logistic regression is similar to linear regression, however it is used for classification problem.

$$h(\vec{x}) = f(\theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n)$$



Logistic regression model

Logistic regression is similar to linear regression, however it is used for classification problem.

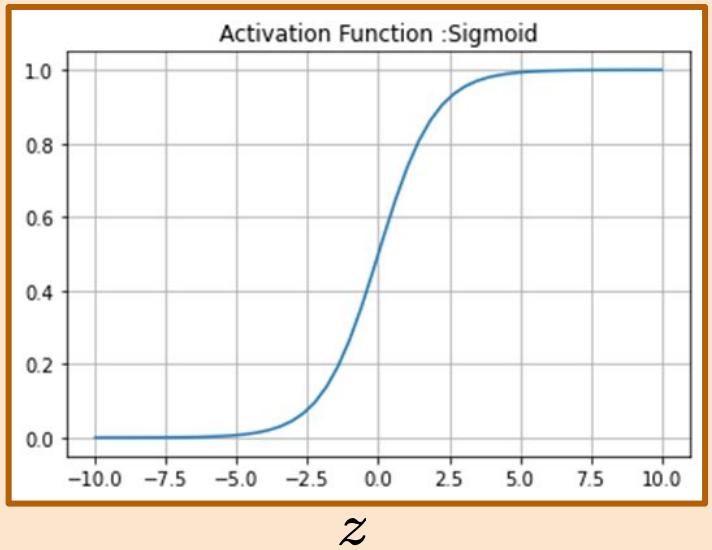
- Linear equation: $\sum_i \theta_i x_i = z$
- Every model in GLM: $h(\vec{x}) = f(z)$
- Linear regression model: $h(\vec{x}) = f(z) = f(\sum_i \theta_i x_i)$

  let $f(z) = z$ then $h(\vec{x}) = \sum_i \theta_i x_i$
- Logistic regression model: $f(z) = ?$ ➜ Logistic function

Logistic regression model

Logistic regression is similar to linear regression, however it is used for classification problem.

$$f(z) = h(\vec{x}) = f(\theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n)$$



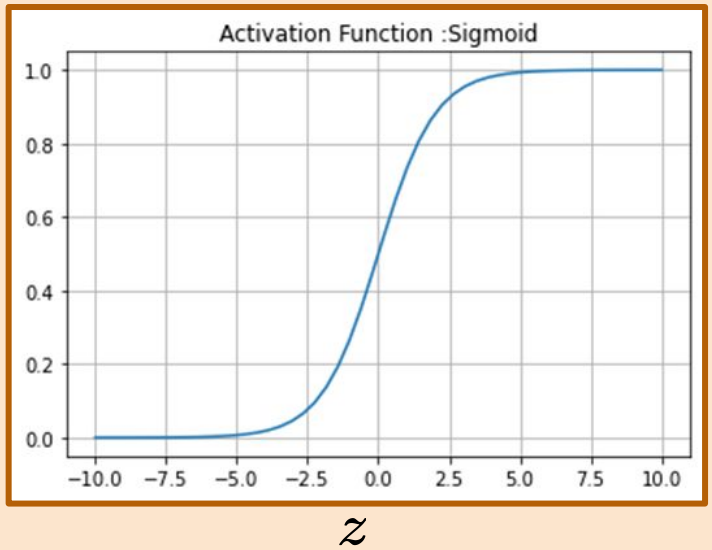
$$f(z) = \frac{1}{1+e^{-z}}$$

- Domain: $z \in \Re$
- Range: $f(z) \in (0, 1)$
- $z \to -\infty$ then $f(z) \to 0$
- $z \to \infty$ then $f(z) \to 1$

Logistic regression model

Logistic regression is similar to linear regression, however it is used for classification problem.

$$f(z) = h(\vec{x}) = f(\theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n)$$



Probability

Sample 1	$x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}$	...	$h(\vec{x}^{(1)})$
Sample 2	$x_1^{(2)}, x_2^{(2)}, \ldots, x_n^{(2)}$	...	$h(\vec{x}^{(2)})$
...	...	...	...
Sample *i*	$x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}$	...	$h(\vec{x}^{(i)})$

$if\ h(\vec{x}^{(i)}) = 0.8\ :$

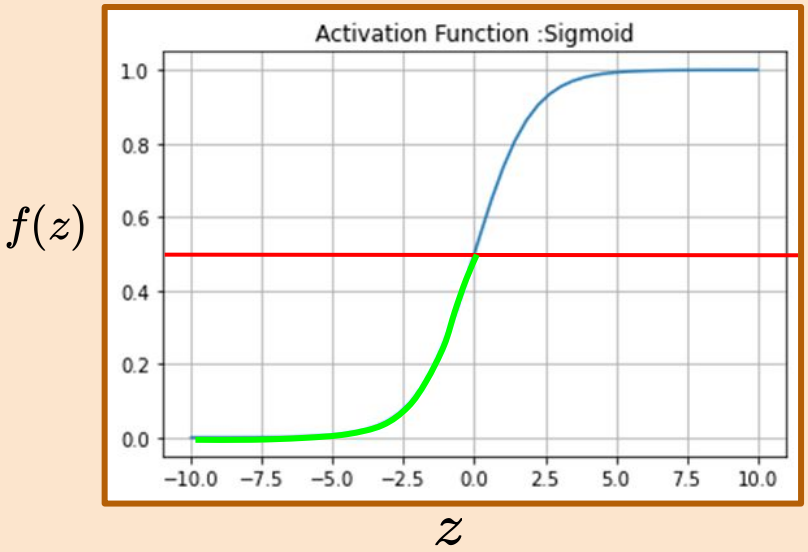
$Sample\ i \equiv\ Class\ 1\ : 80\%\ or\ 0.8$

$Sample\ i \equiv\ Class\ 0\ : 20\%\ or\ 0.2$

Logistic regression model

Logistic regression is similar to linear regression, however it is used for classification problem.

$$f(z) = h(\vec{x}) = f(\theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n)$$



Example

Sample *i*	x_1^i	x_2^i	...	$h(\vec{x}^{(i)})$	y
1	3	A	...	0.6	1
2	6	A	...	0.3	0
3	12	B	...	0.5	1

$$Dicision\ rule:$$

$$y = \begin{cases} 1, & \text{if } f(z) = h(\vec{x}^{(i)}) \geq 0.5 \text{ or } z \geq 0 \\ 0, & \text{if } f(z) = h(\vec{x}^{(i)}) < 0.5 \text{ or } z < 0 \end{cases}$$

A simple example

Let's look at a simple logistic regression procedure.

Sample i	age $x_1^{(i)}$	income $x_2^{(i)}$	dependents $x_3^{(i)}$	spending $x_4^{(i)}$	history y
1	40	50	0	30	1
2	25	40	2	35	1
3	18	10	0	12	0
4	34	22	1	10	1
	55	**65**	**4**	**60**	**?**

A simple example

We first to do preprocessing, such as normalization and standardization

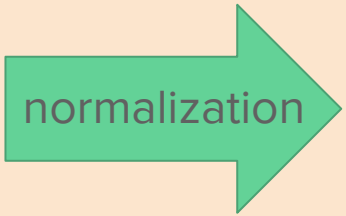
i	$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	$x_4^{(i)}$	y
1	40	50	0	30	1
2	25	40	2	35	1
3	18	10	0	12	0
4	34	22	1	10	1

A simple example

We first to do preprocessing, such as normalization and standardization

$$new\ x_j^{(i)} = \frac{x_j^{(i)}}{max(x_j^{(i)} \mid i{=}1,2,3,4\ and\ j{=}1,2,3,4)}$$

i	$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	$x_4^{(i)}$	y
1	40	50	0	30	1
2	25	40	2	35	1
3	18	10	0	12	0
4	34	22	1	10	1



i	$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	$x_4^{(i)}$	y
1	0.44	0.63	0	0.6	1
2	0.28	0.5	0.5	0.7	1
3	0.2	0.13	0	0.24	0
4	0.38	0.28	0.25	0.2	1

$$max(x_j^{(i)} \mid i = 1,2,3,4\ and\ j = 1,2,3,4) = 91$$

A simple example

Then, fit the logistic regression to the data. Suppose after fitting, here are the wight numbers.

$$h(\vec{x}) = f(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4)$$

$$= f(0 + 0.7x_1 + 0.6x_2 - 0.1x_3 - 0.2x_4)$$

A simple example

$$h(\vec{x}) = f(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4)$$

$$= f(0 + 0.7x_1 + 0.6x_2 - 0.1x_3 - 0.2x_4)$$

Let make prediction for a single customer.

j	Raw data	Normalization	θ_j	$\theta_j x_j$
age x_1	40.00	0.44	0.7	0.31
income x_2	57.00	0.63	0.6	0.38
dependent x_3	0.00	0.00	-0.1	0.00
spending x_4	55	0.60	-0.2	-0.12
			Sum:	0.57
			$h(\vec{x})$	0.64

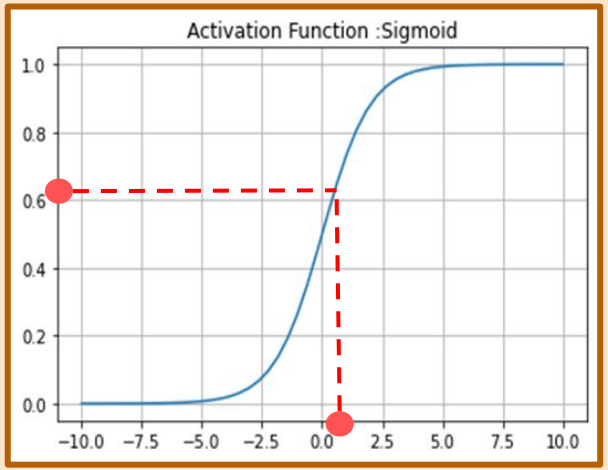
A simple example

$$h(\vec{x}) = f(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4)$$

$$= f(0 + 0.7x_1 + 0.6x_2 - 0.1x_3 - 0.2x_4)$$

Let make prediction for a single customer.

j	Raw data	Normalization	θ_j	$\theta_j x_j$
age x_1	40.00	0.44	0.7	0.31
income x_2	57.00	0.63	0.6	0.38
dependent x_3	0.00	0.00	-0.1	0.00
spending x_4	55	0.60	-0.2	-0.12
			Sum:	0.57
			$h(\vec{x})$	0.64



A simple example

$$h(\vec{x}) = f(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4)$$

$$= f(0 + 0.7x_1 + 0.6x_2 - 0.1x_3 - 0.2x_4)$$

Let make prediction for a single customer.

j	Raw data	Normalization	θ_j	$\theta_j x_j$
age x_1	40.00	0.44	0.7	0.31
income x_2	57.00	0.63	0.6	0.38
dependent x_3	0.00	0.00	-0.1	0.00
spending x_4	55	0.60	-0.2	-0.12
			Sum:	0.57
			$h(\vec{x})$	0.64

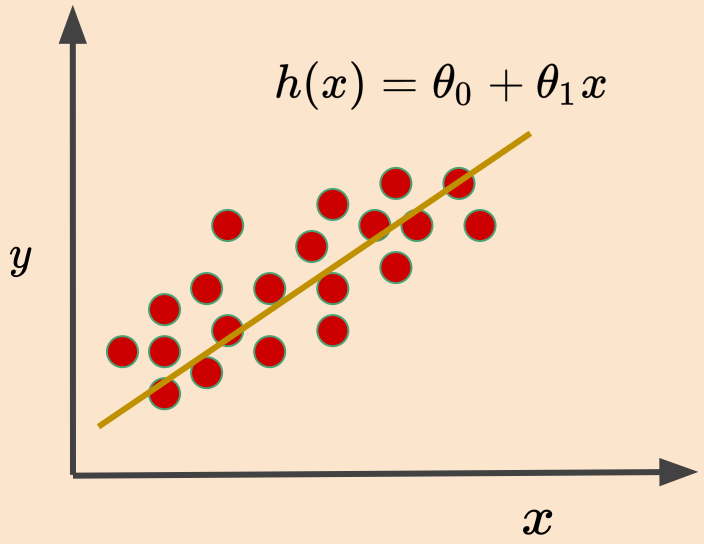
$h(\vec{x})$ Indicates the probability of customer being good.

$$Dicision\ rule:$$

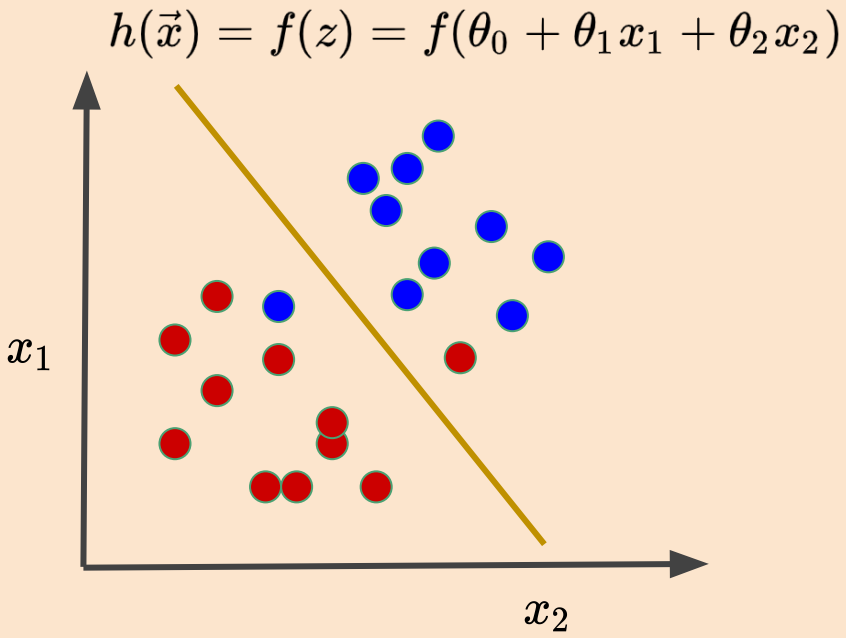
$$y = \begin{cases} 1, & \text{if } h(\vec{x}^{(i)}) \geq 0.5 \\ 0, & \text{if } h(\vec{x}^{(i)}) < 0.5 \end{cases}$$

➜ This is a good customer !

A simple example



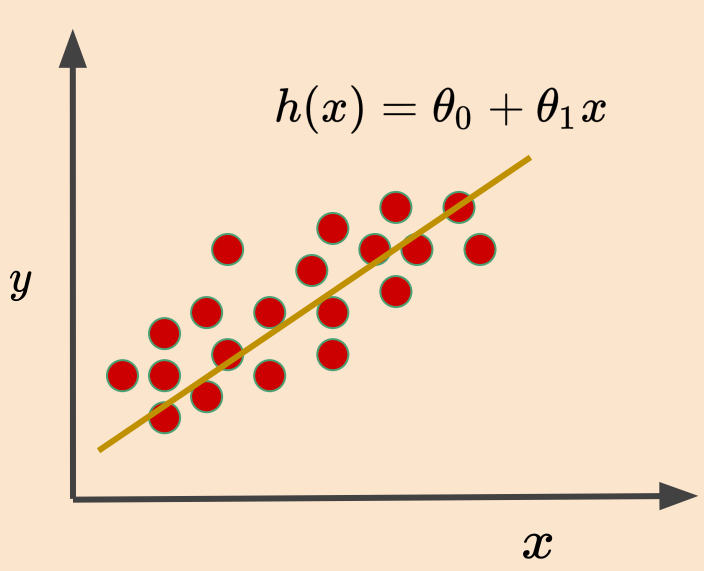
Linear regression



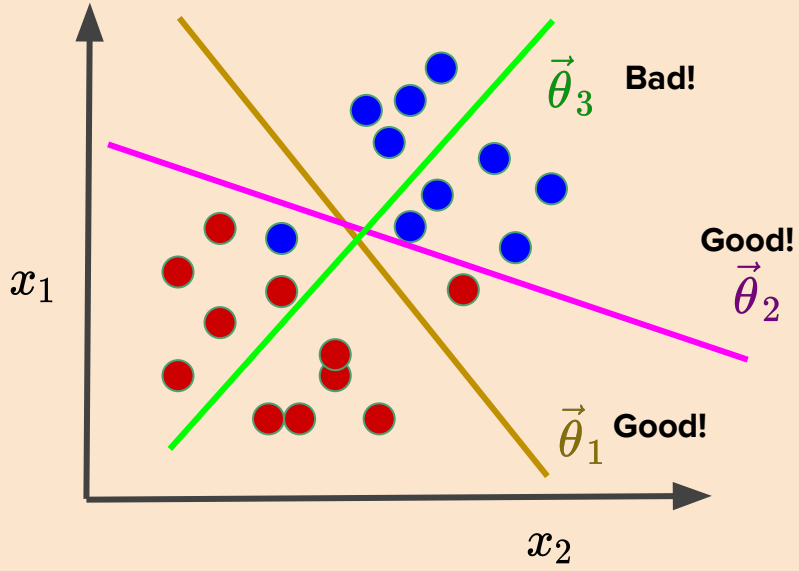
Logistic regression

A simple example

$$h(\vec{x}) = f(z) = f(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$



Linear regression



Logistic regression

Classification cost function

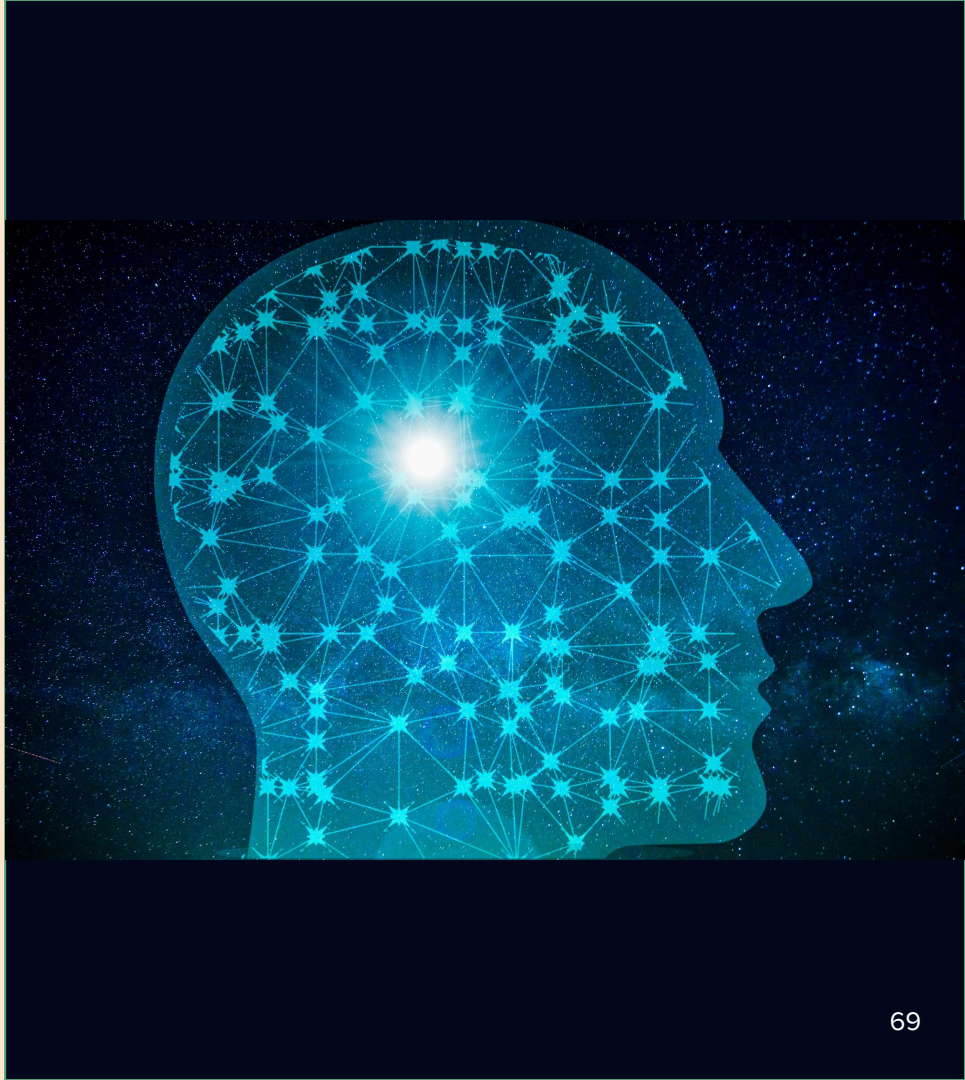
Cost function: sum of errors from classifying sample *i* : $\sum_i E_i$

$$E_i = -(y^{(i)} log(h(\vec{x}^{(i)})) + (1 - y^{(i)}) log(1 - h(\vec{x}^{(i)})))$$

The cost function is ‘cross entropy’ error, defining how actual classes match your class predictions.

$h(\vec{x}^{(i)})$	y	**Cost**
0.8	1	0.10
0.1	1	1.00
0.1	0	0.05

Classification Performance



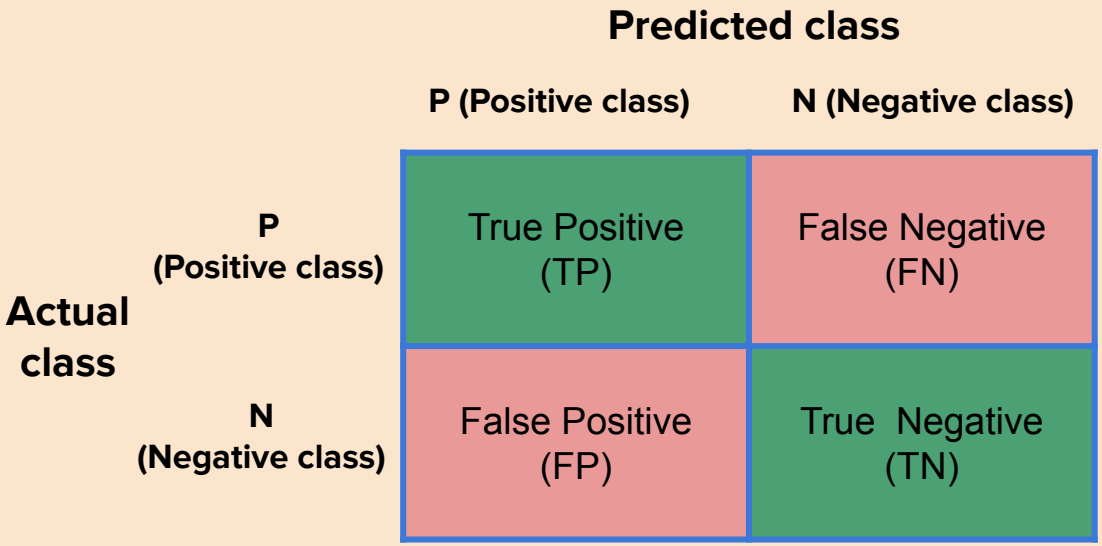
Classification Performance

- Log loss (‘cross entropy’ error)
- Accuracy
- Precision
- Recall
- F1-score

Classification Performance

- Log loss (‘cross entropy’ error)
- Accuracy
- Precision
- Recall
- F1-score

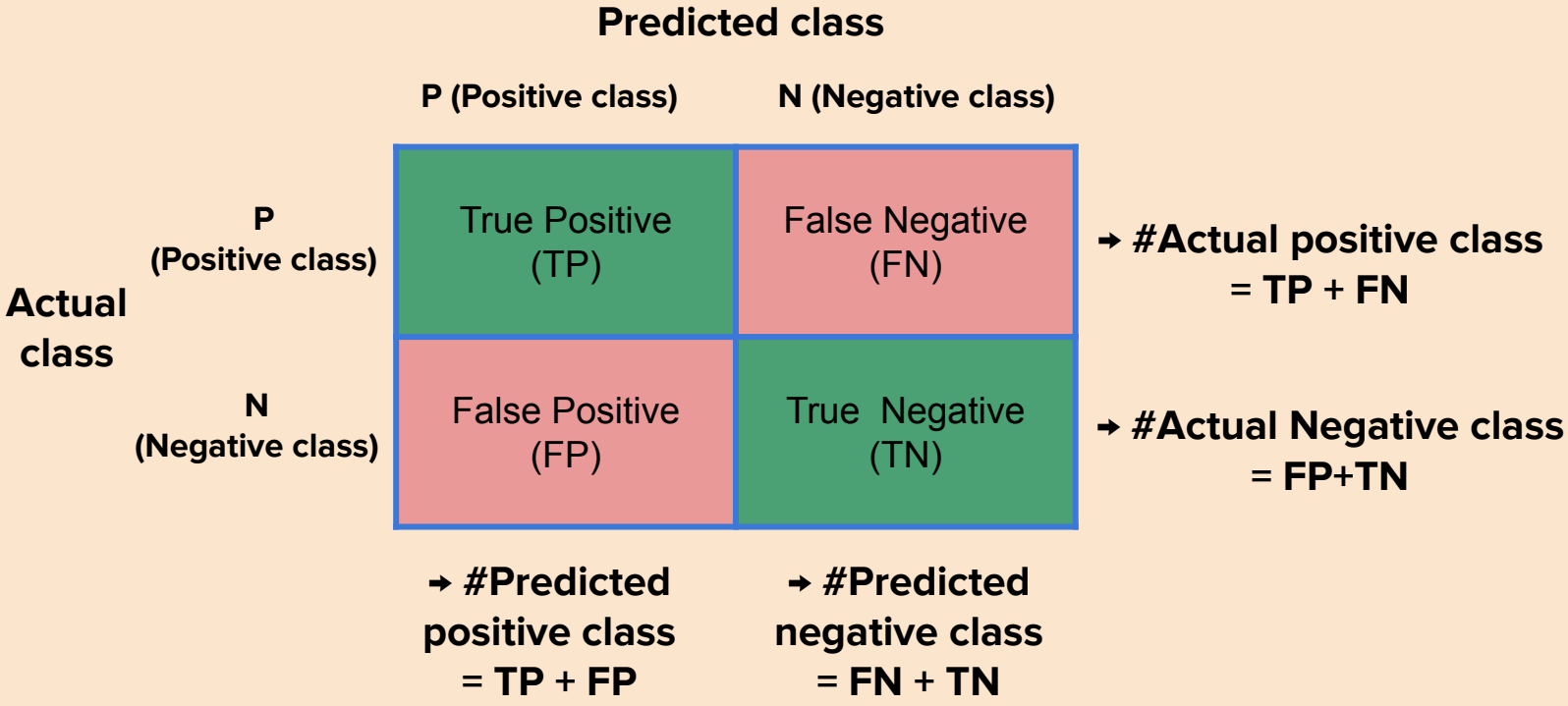
Confusion Matrix



$$P \equiv Positive\ class \equiv label\ 1$$
$$N \equiv Negative\ class \equiv label\ 0$$

Classification Performance

Confusion Matrix



Classification Performance

Confusion Matrix

Actual class \ Predicted class	P (Positive class)	N (Negative class)
P (Positive class)	True Positive (TP)	False Negative (FN)
N (Negative class)	False Positive (FP)	True Negative (TN)

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN}$$

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision+Recall}$$

Classification Performance

Example 1

Actual class \ Predicted class	P (Positive class)	N (Negative class)
P (Positive class)	70 (TP)	30 (FN)
N (Negative class)	35 (FP)	65 (TN)

$$Accuracy = \frac{70+65}{70+30+35+65} = 0.675$$

$$Precision = \frac{70}{70+35} = 0.67$$

$$Recall = \frac{70}{70+30} = 0.7$$

$$F1 - score = 2 \times \frac{0.67 \times 0.7}{0.67+0.7} = 0.685$$

- # Actual positive class: 100
- # Actual negative class: 100

Classification Performance

Example 2

		Predicted class	
		P (Positive class)	**N (Negative class)**
Actual class	**P (Positive class)**	0 (TP)	5 (FN)
	N (Negative class)	5 (FP)	995 (TN)

$$Accuracy = \frac{0+995}{0+5+5+995} = 0.99$$

$$Precision = \frac{0}{0+5} = 0$$

$$Recall = \frac{0}{0+5} = 0$$

$$F1 - score = 2 \times \frac{0 \times 0}{0+0} = 0$$

- # Actual positive class: 5
- # Actual negative class: 1000

Good luck 😉


Uppsala universitet 1852–1916
Studenten, Staden och Samingen
UPPSALA UNIVERSITE