NOTES

Date : 20-11-23

Machine Learning Specialization

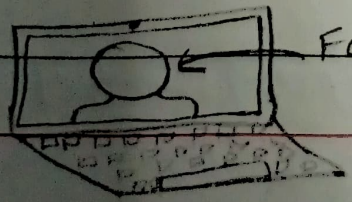
1). Supervised Machine Learning

Week-1

Overview of ML

Machine Learning is a field of study in AI concerned with the development & study of statical algorithms that can we to teach machine to perform a task without human instruction.

for example, anytime you want to find out something like "how do i make a shushi roll?" you can do a web search on google to find out. And that work so well because this machine Leoning software has figured out how to rank web pages.



Face recognision

NOTES Date :

Applications of Machine Learning

→ Image Recognition
→ Speech Recognition
→ Automatic Language Translation
→ self driving car
→ Recommendation systems, etc.

Supervised Vs Unsupervised ML

Machine Learning : "Field of study that gives computers the ability to learn without being explicitly programmed".

Machine learning Algorithm

Supervised Learning

Unsupervised Learning

Supervised Learning

X ⟶ Y

(Input) (output) Label

Learn from being given "right answers"

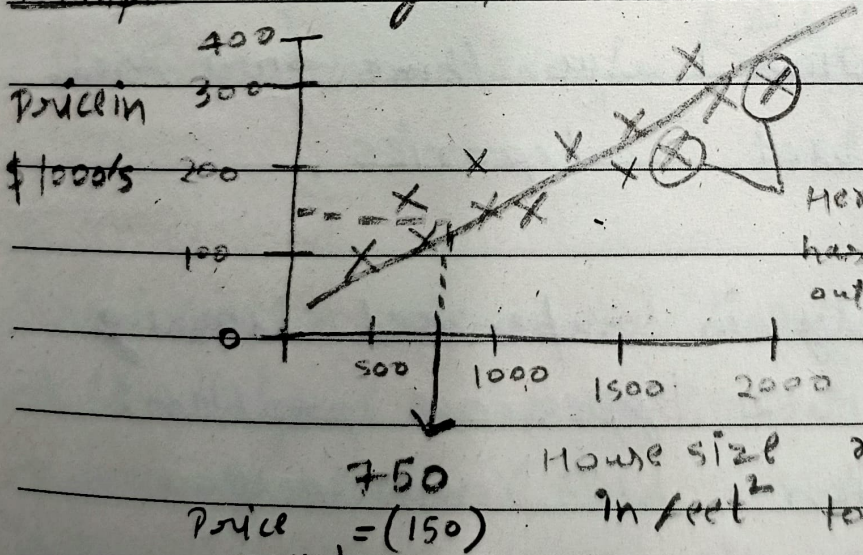
NOTES

Date :

- Supervised Machine Learning or more commonly, supervised learning, refers to algorithm that learn X to Y or input to output mappings.

- The key characteristic of supervised learning is that you give your learning algorithm examples to learn from. That includes the right answer, whereby right answers, means that a correct label Y for a given input X.

Example: Housing Price Prediction



Here we already has some input & output from which our model created a regression line to predict house prices.

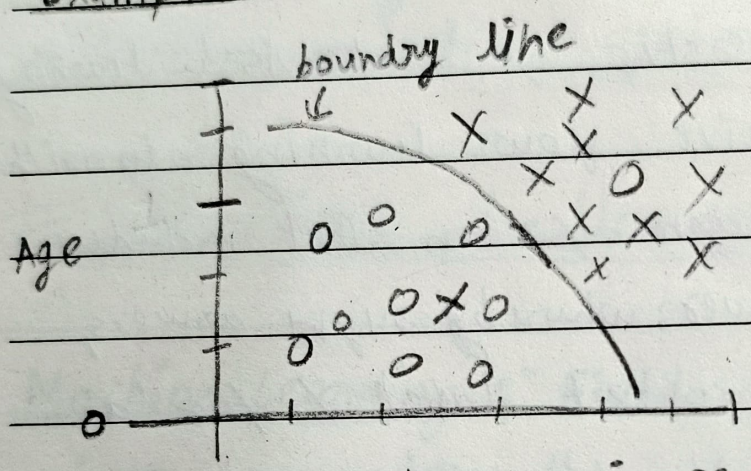
Regression

NOTES

Date :

• Another example of a supervised learning can be a "classification" problem.

Example: Breast cancer detection



(1) malignant or benign (0)

size	diagnosis
2	0
5	1
1	0
⋮	⋮

• classification algorithms are used to predict categories.

• So mainly in supervised learning we have 2 types of problem:

→ Regression

→ classification.

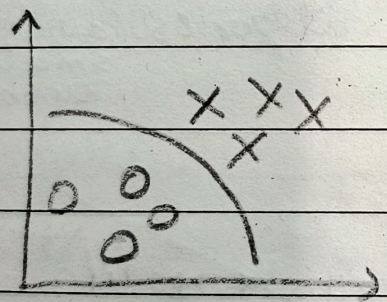
NOTES

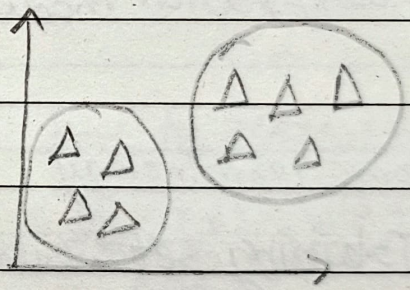
Date :

Unsupervised Learning

- Unlike supervised learning, unsupervised learning model are given unlabeled data & allowed to discover patterns & insights without any explicit guidance or instruction.

supervised Learning	Unsupervised learning
Learn from data labeled with the "right answers"	Find something interesting in unlabeled data





clustering

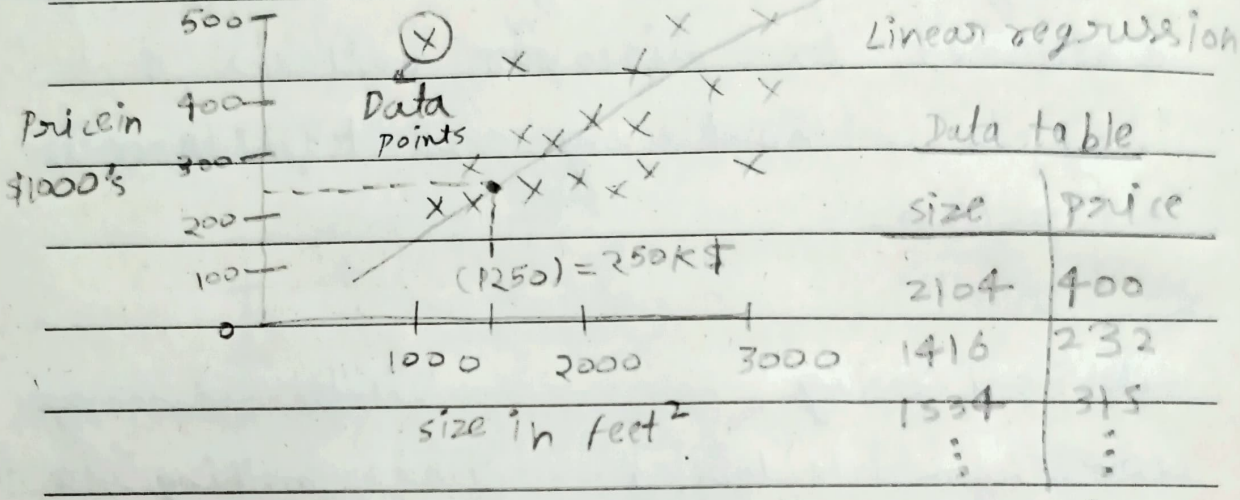
- Some type of unsupervised learning are
  - → clustering (group similar data points together).
  - → Anomaly detection (Find unusual data points).
  - → Dimensionality reduction (compress data using fewer numbers).

NOTES

Date : 21-11-23

Regression Model

House sizes & prices



Data table

size	price
2104	400
1416	232
1534	315
⋮	⋮

- Regression model predicts numbers. (Infinity)
- Classification model predicts categories. (discrete) small numbers

Some terminology

- Training set : Data used to train the model

x size in feet²	y price in $1000's	
(1) 2104	400	m = 47
(2) 1416	232	
(3) 1534	315	
⋮	⋮	
(47)		

- Notation : x = "input" variable / feature

y = "output" variable

"target" variable

$\Rightarrow x = 2104, \; y = 400$

NOTES

Date :

m = number of training examples

(x, y) = single training example

⇒ (x, y) = (2104, 400)

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

(1^{st}, 2^{nd}, 3^{rd}, ...)

⇒ $x^{(1)} = 2104$, $y^{(1)} = 400$

$(x^{(1)}, y^{(1)}) = (2104, 400)$

$(x^{(2)}, y^{(2)}) = (1416, 232)$

⇒ $x^{(2)} \neq x^2$ not exponent

f = function

$\hat{y}$ = "y-hat" predicted value

Model

training set → features, target

learning algorithm

x → [f] → $\hat{y}$ "y-hat"

function

feature

prediction (estimated y)

target

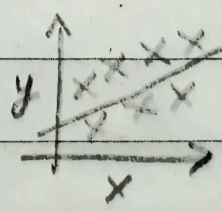
size → [f] → price (estimated)

How to represent f?

$f_{w,b}(x) = wx + b$

$f(x)$

• for now let w & b are some numbers which helps us to find the $\hat{y}$.



$f_{w,b}(x) = wx + b$

$f(x) = wx + b$

linear fn

• Linear regression with one variable. (size)

• Univariate linear regression

NOTES Date :

Cost Function

Training set:

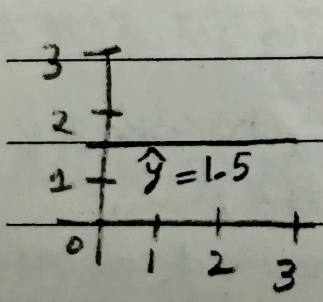
size in feet² (x)	Price $1000's (y)
2104	460
1416	232
1534	315
852	178
⋮	⋮

Model: $f_{w,b}(x) = wx + b$

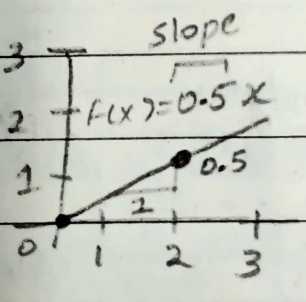
w, b : parameters — what do "w, b" do?

[coefficients / weights]

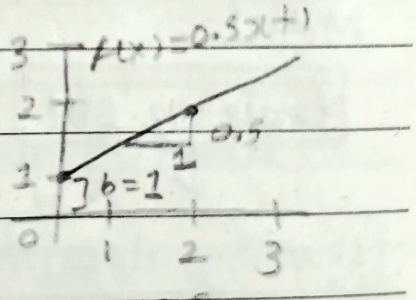
$\Rightarrow f_{w,b}(x) = wx + b$



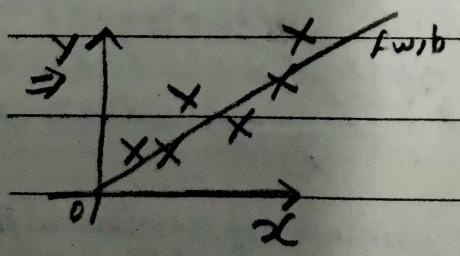
$\Rightarrow w = 0$
$b = 1.5$



$w = 0.5$
$b = 0$



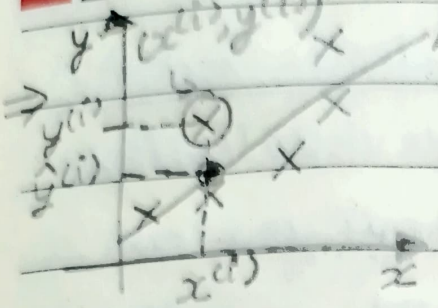
$w = 0.5$
$b = 1$



- So w & b are some parameters which helps in finding a $f_{w,b}$ line by changing these parameters again & again

NOTES

Date :



$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$

★ so, now how should we know the correct value of w & b for the linear regression model?

⇒ so, the ans is "Cost function".

- Cost function helps in finding the difference b/w the actual & predicted value & the minimum the cost fⁿ implise more accurate model.

⇒ Cost function : Squared Error Cost function

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(\underbrace{\hat{y}^{(i)} - y^{(i)}}_{\text{error}})^2, \quad m = \text{no. of train example}$$

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

NOTES

Date : 22-11-23

Train the Model with Gradient Descent

★ In last part we saw visualizations of the cost function j & how you can try different choices of the parameters "w" & "b".

Gradient Descent

- It would be nice if we had a more systematic way to find the values of "w" & "b", that results in the smallest possible cost, j of w, b.

- It turns out that there's an algorithm called "gradient descent" that you can use to do that.

★ Gradient Descent is a machine learning algorithm which helps in minimizing the cost function as much as possible to find the best value of 'w' & 'b'.

NOTES

Date :

Outline of Gradient descent

1). start with some w, b (set w=0, b=0)

2). keep changing w, b to reduce J(w,b) [by eqⁿ]

3). until we settle at or near a minimum (~0)

• Gradient descent algorithm

Learning rate (alpha)

$$\Rightarrow w = w - \alpha \frac{d}{dw} J(w,b)$$

assignment operator

Derivative term of cost fⁿ J(w,b)

$$\Rightarrow b = b - \alpha \frac{d}{db} J(w,b)$$

- Repeat until convergence
- simultaneously update w & b

★ Correct way to implement: simultaneously update

$$\Rightarrow temp\_w = w - \alpha \frac{d}{dw} J(w,b)$$

using old value not new one

$$temp\_b = b - \alpha \frac{d}{db} J(w,b)$$

we do this because in updating value of 'b' we have to use old value of 'w'.

$$w = temp\_w$$
$$b = temp\_b$$

updating the value of 'w' & 'b'

NOTES Date :

• More intuition about gradient descent:

repeat until convergence {

$$w = w - \alpha \boxed{\frac{d}{dw} J(w,b)}$$ "derivative term"

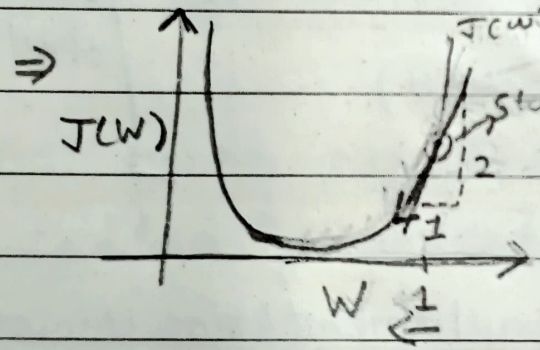
learning rate

$$b = b - \alpha \frac{d}{db} J(w,b)$$

} simultaneously update

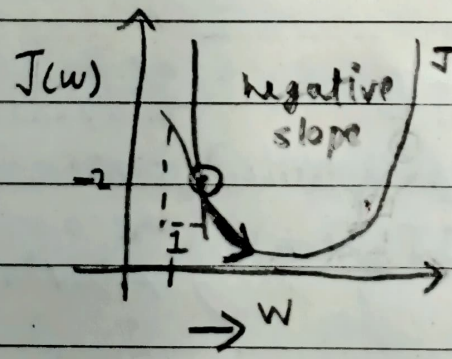
Suppose we have cost fn of $J(w)$,

$$\Rightarrow w = w - \alpha \frac{d}{dw} J(w)$$



$w = w - \alpha \frac{d}{dw} J(w)$

$w = w - \alpha \cdot (+ \text{number})$



$w = w - \alpha \cdot (- \text{number})$

NOTES

Date :

Learning Rate (alpha α)

- Alpha also known as the learning rate, is a hyperparameter that determines the step size at each iteration of the gradient descent algorithm.

- It controls how quickly or slowly the algorithm converges to the optimal solution. The value of alpha is crucial as it can greatly impact the optimization process

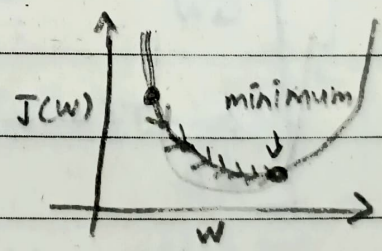
⇒ repeat until convergence {

$$w = w - \boxed{\alpha} \frac{d}{dw} J(w)$$

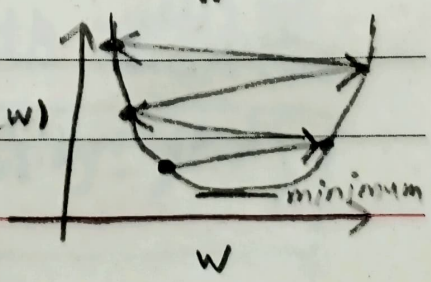
alpha

} simultaneously update

- if α is too small...
  ⇒ Gradient descent may be slow



- if α is too large...
  ⇒ Gradient descent may never reach minimum.



NOTES

Date : 23-11-2023

Gradient Descent for Linear Regression

Let's Recap:

- Linear regression model

$$f_{w,b}(x) = w \cdot x + b$$

- cost Function

$$J(w,b) = \frac{1}{2m} \sum_{i=0}^{m-1} \left(f_{w,b}(x^{(i)}) - y^{(i)} \right)^2$$

- Gradient descent algorithm

repeat until convergence {

$$w = w - \alpha \frac{d}{dw} J(w,b)$$

$$b = b - \alpha \frac{d}{db} J(w,b)$$

} simultaneously update

$\Rightarrow$ so, now derivatives of:

$$\frac{d}{dw} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

$$\frac{d}{db} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)} \right)$$

so, how we get these terms?

NOTES

Date :

Let's find out: (optional)

$\frac{d}{dw} J(w,b)$

↑ Here we can see that we have to do partial derivatives with respect to 'w'.

$$\Rightarrow \frac{d}{dw} J(w,b) = \frac{d}{dw}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2\right]$$

$$= \frac{d}{dw}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)^2\right]$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right) \times 2x^{(i)}$$

$$\boxed{\frac{d}{dw} J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)x^{(i)}}$$

similarly, with respect to 'b',

$$\Rightarrow \frac{d}{db} J(w,b) = \frac{d}{db}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2\right]$$

$$= \frac{d}{db}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)^2\right]$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right) 2$$

$$\boxed{\frac{d}{db} J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)}$$

NOTES

Date :

- So, now final gradient descent algorithm for linear Regression is :

⇒ repeat until convergence {

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$$

}; simultaneously update.

NOTES

Date : 28-11-2023

Week 2

Multiple Linear Regression

Recap: In previous week we have working with single variable feature:

size in feet² (x)	Price ($) in 1000$ (y)
2104	400
1416	232
1534	315
852	178
...	...

$\gg f_{w,b}(x) = wx + b$

- But if we have multiple features like:

size in feet²	Number of bedrooms	Number of floors	Age of home in years	Price
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...
X_1	X_2	X_3	X_4	

$\Rightarrow X_j$ = j^{th} feature // x_1, x_2, x_3, x_4

n = number of features // $n = 4$

$\vec{x}^{(i)}$ = feature of i^{th} training example

// $\vec{x}^{(2)} = [1534, 3, 2, 40]$

$x_j^{(i)}$ = value of feature j in i^{th} training example // $x_3^{(2)} = 2$

NOTES

Date :

Model:

previously: $f_{w,b}(x) = wx + b$

$\Rightarrow f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$

example:

$f_{w,b}(x) = 0.1x_1 + 4x_2 + 10x_3 + -2x_4 + 80$

↓ #size ↓ #bedrooms ↓ #floors ↓ #years ↓ base price

- so, from above we can say that, if we have n features then model will look like:

$f_{w,b}(x) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$

$\Rightarrow \vec{w} = [w_1 \; w_2 \; w_3 \cdots w_n]$] vectors

$\vec{x} = [x_1 \; x_2 \; x_3 \cdots x_n]$]

b = b is a number

$$\Rightarrow f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

(↑ dot product)

★ A Linear Regression with multiple Feature is known as "multiple linear regression".

NOTES

Date : 29/11/23

Parameters & features:

$\vec{w} = [w_1, w_2, w_3]$

b is a number

$\vec{x} = [x_1, x_2, x_3]$

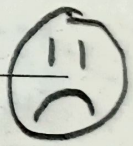
Numpy

```
⇒ w = np.array([1.0, 2.5, -3.3])
  b = 4
  x = np.array([10, 20, 30])
```

① without vectorization

$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

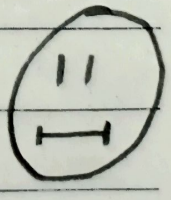
```
⇒ f = w[0] * x[0] + w[1] * x[1] +
      w[2] * x[2] + b
```



② without vectorization (with loop)

$f_{\vec{w},b}(\vec{x}) = \sum_{j=1}^{n} w_j x_j + b$

```
⇒ f = 0
  for j in range(0, n):
      f = f + w[j] * x[j]
  f = f + b
```

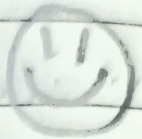


NOTES Date :

With Vectorization

$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

$\Rightarrow$ `f = np.dot(w,x) + b`



Let's analyse code:

without vectorization

```
⇒ for j in range(0,16):
      f = f + w[j] * x[j]
```

t_0 `f + w[0] * x[0]`

t_1 `f + w[1] * x[1]`

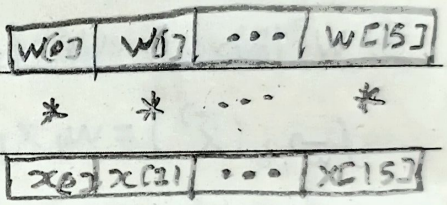
...

t_{15} `f + w[15] * x[15]`

vectorization

```
⇒ np.dot(w,x)
```

t_0



t_1

`w[0]*x[0] + w[1]*x[1] + ... + w[15]*x[15]`

NOTES

Date :

Gradient Descent for Multiple Regression

vector notation

Parameters: $\vec{w} = [w_1 \cdots w_n]$

b is a number

Model : $f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

Cost Function : $J(\vec{w}, b)$

Gradient Descent: repeat {

$$w_j = w_j - \alpha \frac{d}{dw_j} J(\vec{w}, b)$$

$$b = b - \alpha \frac{d}{db} (\vec{w}, b)$$

}; simultaneously update

Gradient Descent

one feature

repeat {

$$w = w - \alpha \frac{1}{m} \sum_{i=0}^{m} (f_{\vec{w},b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$$

}; simultaneously update

NOTES

Date :

multible features

repeat {

$$\vec{w} = \vec{w} - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) \cdot \vec{x}^{(i)}$$

$$b = b - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$$

}; simultaneously update

NOTES

Date :

Gradient Descent in Practice

Feature scaling

. Feature scaling is the process of normalizing the range of feature in a dataset. Real-world dataset are more often various in range, degree, etc. Therefore, in order for machine learning models to interpret these features on the same scale, we need to perform feature scaling.

Let,

$\hat{price} = w_1 x_1 + w_2 x_2 + b$ $\quad x_1$: size(feet^2)

size, #bedrooms $\quad x_2$: # bedrooms

$\Rightarrow x_1$: [range = 300 − 2,000] large

x_2: [range = 0 − 5] small

House: $x_1 = 2000$, $x_2 = 5$, price = \$500k

$\Rightarrow$ size of the parameters w_1, w_2 ?

$\Rightarrow$ suppose,

$w_1 = 50$, $w_2 = 0.1$, $b = 50$

$\hat{price} = 50 * 2000 + 0.1 * 5 + 50$

$= \$100{,}050.5k > \$500k$

NOTES

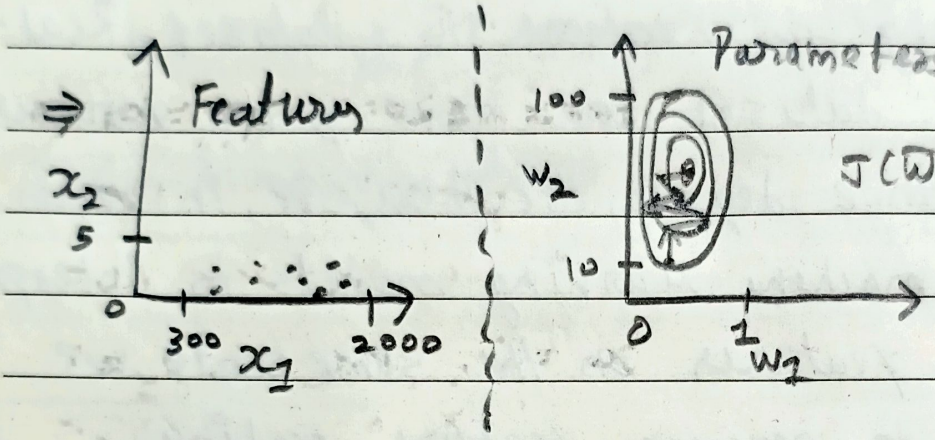
Date :

⇒ suppose,

$w_1 = 0.1$, $w_2 = 50$, $b = 50$

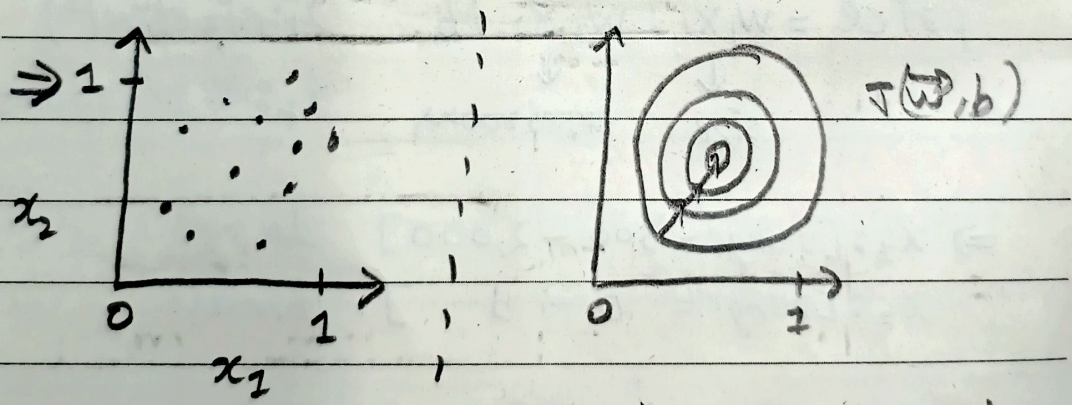
$\widehat{price} = 0.1 * 2000 + 50 * 5 + 50$

$= \$500k$ more reasonable



After feature scaling

[doing some transformation]



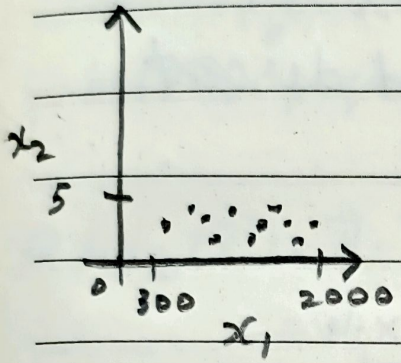
much more faster & accurate than previous.

NOTES

Date :

How do we do feature scaling?

⇒ One way to do this is using "Mean normalization"



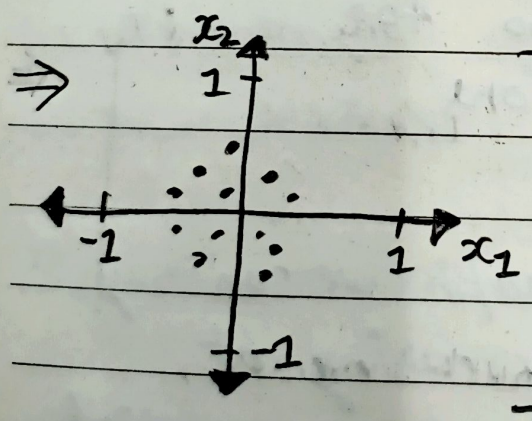
$300 \le x_1 \le 2000$ $\quad$ $0 \le x_2 \le 5$

⇒ average $\mu_1 = 600$ $\quad$ $\mu_2 = 2.3$

$$x_1 = \frac{x_1 - \mu_1}{2000 - 300} \qquad x_2 = \frac{x_2 - \mu_2}{5 - 0}$$

(Max − Min) $\quad$ (max − min)

$-0.18 \le x_1 \le 0.82$ $\quad$ $-0.46 \le x \le 0.5$



$$\Rightarrow x = \frac{x - \text{average}}{\text{Max} - \text{Min}}$$

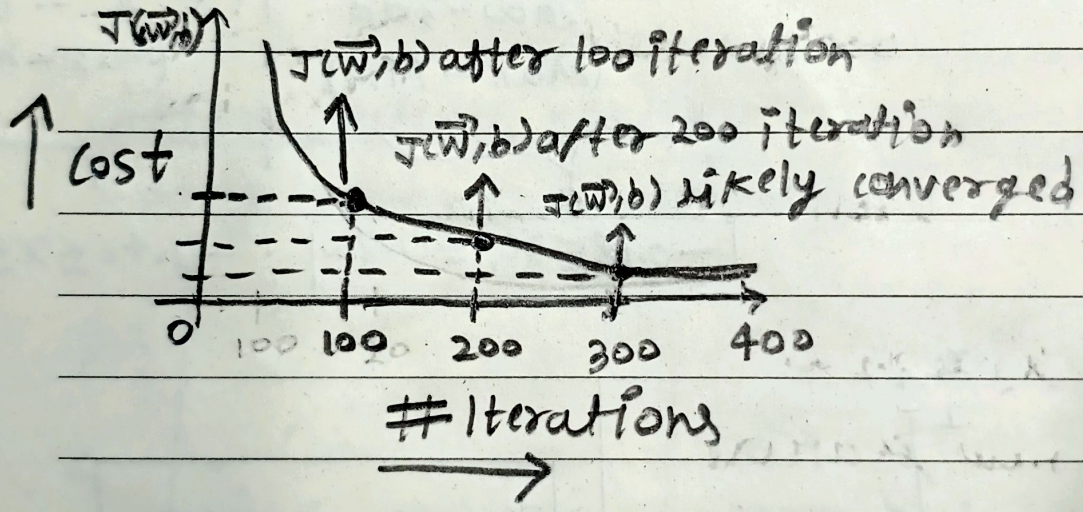
"Mean Normalization"

NOTES Date :

- Now after learning so many things, how should we know our algorithm (Gradient desent) is converging?

⇒ To make sure gradient descent is working correctly.

objective: min $J(\vec{w}, b)$



- From above graph we can say that if cost is decreasing at every iteration then our gradient descent working properly, else there is some issue in our gradient descent algorithm.

NOTES

Date :

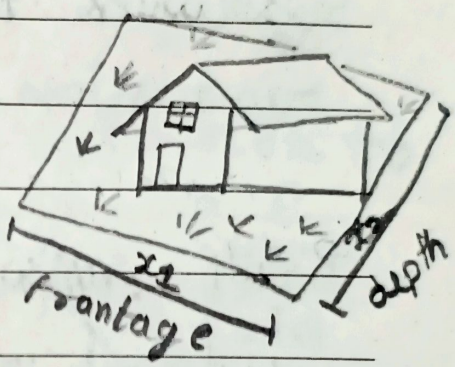
Feature engineering

- Feature engineering refers to manipulation - addition, deletion, combination, mutation - of your data set to improve machine learning model training, leading to better performance & greater accuracy.

$\Rightarrow f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

$= w_1 x_1 + w_2 x_2 + b$

frontage depth



$\Rightarrow$ area = frontage x depth

$x_3 = x_1 x_2$

new feature

$\Rightarrow f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

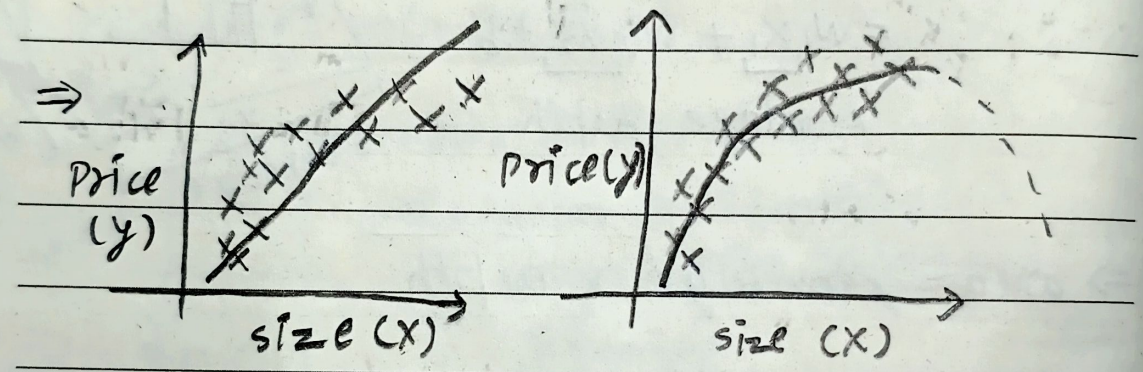
$\Rightarrow$ • Feature engineering: Using intuition to design new features, by transforming or combining original features.

NOTES

Date :

Polynomial regression

- Polynomial regression is a form of Linear regression where only due to the Non-linear relationship between dependent & independent variables; we add some polynomial terms to linear regression to convert it into polynomial regression.



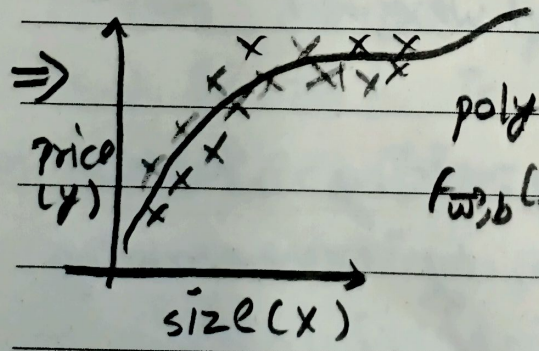
Linear Regression

$f_{w,b}(x) = w \cdot x + b$

Polynomial Regression

$f_{\vec{w},b}(\vec{x}) = w_1 \cdot x_1 + w_2 x_2^2 + b$

x_1: size, x_2^2: size^2



polynomial regression

$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2^2 + w_3 x_3^2 + b$

x_1: size, x_2^2: size^2, x_3^2: size^3

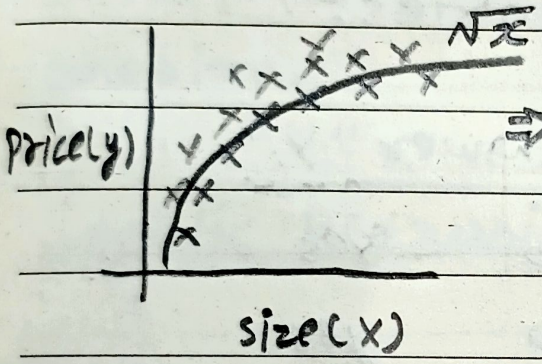
NOTES

Date :

★

- when we are using polynomial regression then it is very necessary to use feature scaling.

- we can also use another eqn for polynomial regression:



$$\Rightarrow f_{w,b}(x) = w_1 x + w_2 \sqrt{x} + b$$

$w_1 x$ ↑ size

$w_2 \sqrt{x}$ ↑ $\sqrt{\text{size}}$

```
⇒ X_poly = np.column_stack (
                [x**i for i in range(degree+1)]
            )
```

NOTES

Date : 2-12-2023

Week - 3

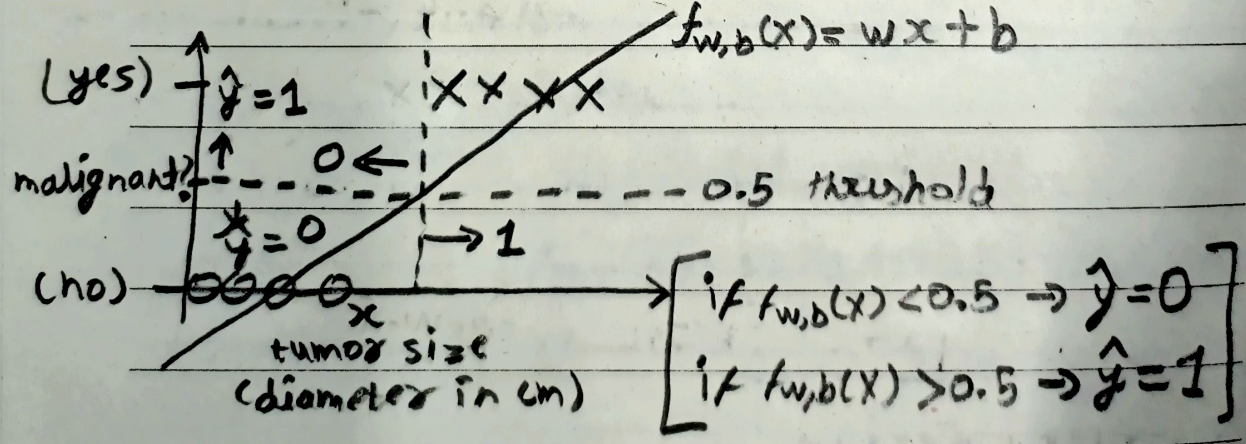
Classification with logistic regression

Classification

- Classification problem is another type of problem in supervised learning where feature input X we have y categories.

Question (X)	Answer "y"	
Is this email spam?	no	yes
Is this transaction fraud?	no	yes
Is the tumor malignant?	no	yes

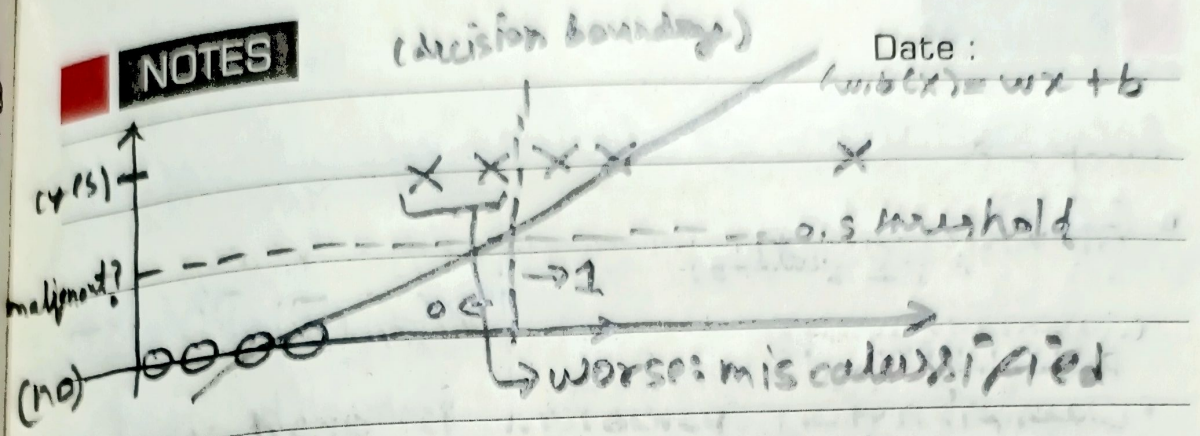
- "y" can only be one of two values
- "binary classification" (0 or 1)



- If above data set over "Linear Regression" is working good.

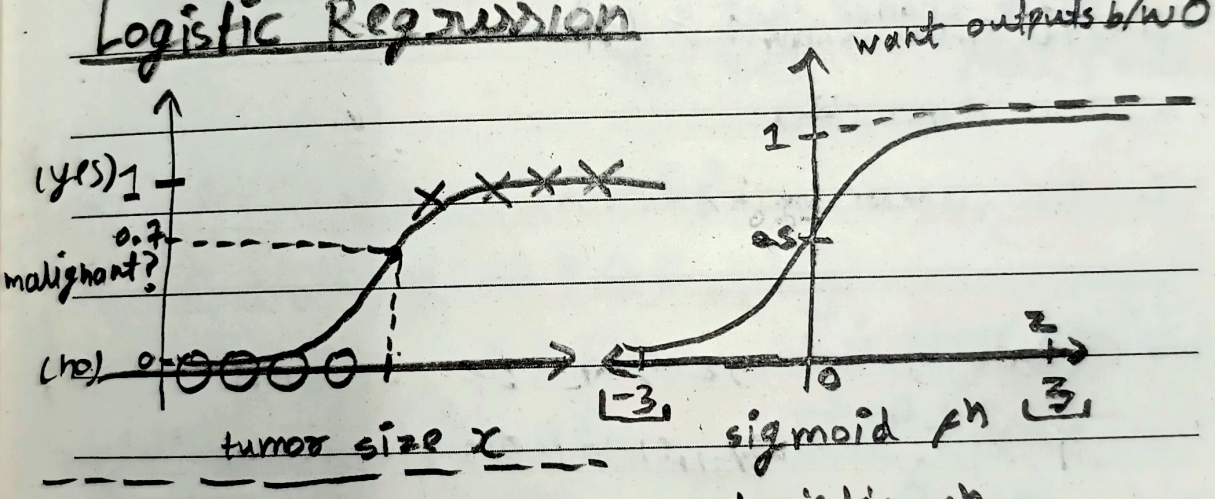
NOTES

Date :



- Now "Logistic Regression" comes in picture. Logistic Regression is a technique to solve classification problem.

Logistic Regression



$f_{\vec{w},b}(\vec{x})$

$z = \vec{w} \cdot \vec{x} + b$

z

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

$\Rightarrow g(z) = \frac{1}{1+e^{-z}}, \quad 0 < g(z) < 1$

- outputs b/w 0 & 1 only.

$\Rightarrow f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w} \cdot \vec{x} + b}_{z})$

$= \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$

"Logistic Regression"

NOTES

Date :

Interpretation of logistic regression output

- $f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$

"probability" that class is 1

- Example:

  x is "tumor size"

  y is 0 (not malignant)

  or 1 (malignant)

$\Rightarrow f_{\vec{w},b}(\vec{x}) = 0.7$

70% chance that y is 1

$\Rightarrow f_{\vec{w},b}(\vec{x}) = P(y=1|\vec{x};\vec{w},b)$

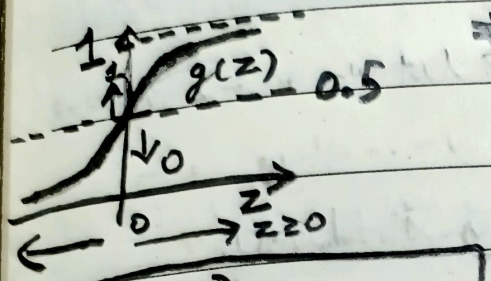
probability that y is 1, given input $\vec{x}$, parameters $\vec{w}, b$.

$\Rightarrow P(y=0) + P(y=1) = 1$

NOTES

Date : 3-12-2023

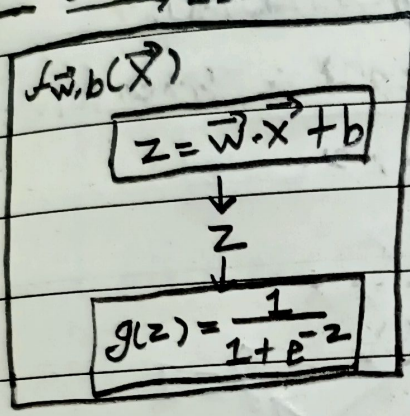
Decision boundary



$\Rightarrow f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w}.\vec{x}+b}_{z}) = \frac{1}{1+e^{-(\vec{w}.\vec{x}+b)}}$

$= P(y=1 | \vec{x}; \vec{w}, b)$

0 or 1?



- To know is $y=1$ or 0 we take a threshold let say 0.5.

$\Rightarrow$ Is $f_{\vec{w},b}(\vec{x}) \geq 0.5$? ← threshold

Yes: $\hat{Y}=1$ No: $\hat{Y}=0$

$\Rightarrow$ when is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

$g(z) \geq 0.5$

$z \geq 0$ [∴ above graph]

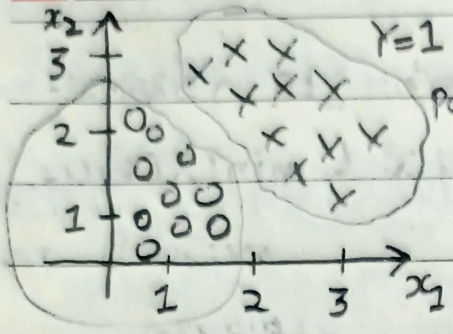
$\vec{w}.\vec{x}+b \geq 0$ $\quad \vec{w}.\vec{x}+b < 0$

$\hat{y}=1$ $\quad \hat{y}=0$

- In Logistic Regression, "Decision Boundry" is a linear line, which seperates the possitive & negative points/class.

NOTES

Date :



$f_{\vec{w},b}(\vec{x}) = g(z)$

$= g(\vec{w}.\vec{x} + b)$

$= g(w_1 x_1 + w_2 x_2 + b)$

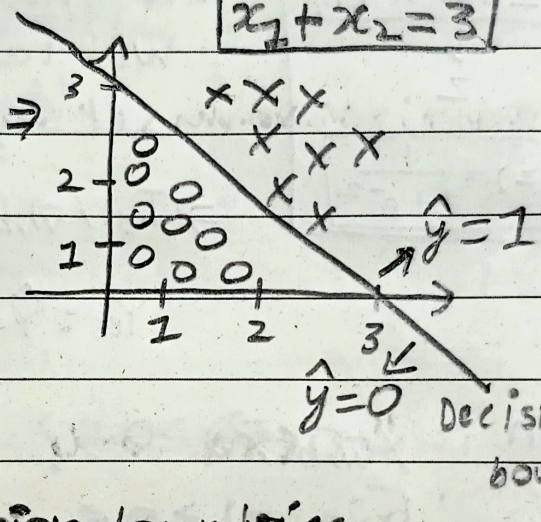
$\Rightarrow$ let $w_1, w_2 = 1$ & $b = -3$

Decision boundry $\Rightarrow$ $\boxed{z = \vec{w}.\vec{x} + b = 0}$

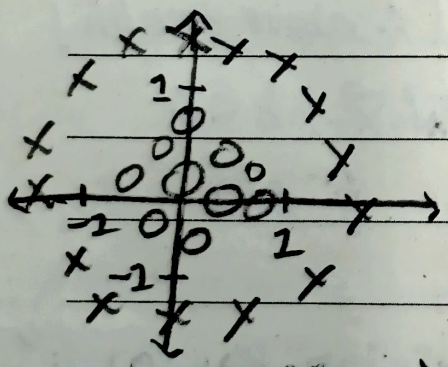
$z = x_1 + x_2 - 3 = 0$

Example

$\boxed{x_1 + x_2 = 3}$



Non-linear Decision boundries



$z = w_1 x_1^2 + w_2 x_2^2 + b$

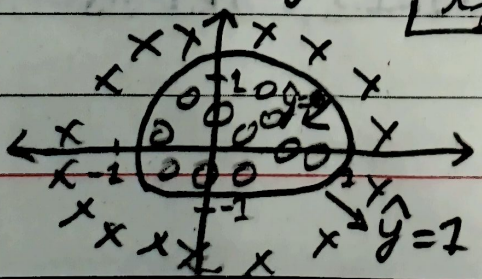
$\Rightarrow f_{\vec{w},b}(\vec{x}) = g(z) = g(\underbrace{w_1 x_1^2 + w_2 x_2^2 + b}_{z})$

let $w_1, w_2 = 1$ & $b = -1$

$\Rightarrow z = x_1^2 + x_2^2 - 1$

Decision Boundry $z = x_1^2 + x_2^2 - 1 = 0$

$\boxed{x_1^2 + x_2^2 = +1}$



$\Rightarrow x_1^2 + x_2^2 \geq 1$: $x_1^2 + x_2^2 < 1$

$\hat{y}=1$: $\hat{y}=0$

outside : inside

NOTES

Date :

code for creating decision boundaries:

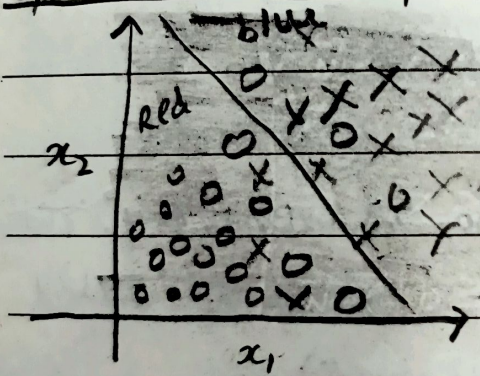
2-D plot

```
x_values = np.linspace(min(X[:,0]),
                       max(X[:,1]),
                       100)

y_values = -(w_updated[0] * x_values +
             b_updated) / w_updated[1]

plt.plot(x_values, y_values, label="Decision boundary")
```

If we want to plot graph like this:



⇒ # creating decision boundary ==========

```
x_values = np.linspace(min(X[:,0]), max(X[:,0]), 100)
y_values = -(w[0] * x_values + b) / w_updated[1]
plt.plot(x_values, y_values, label="DB", color="black")
```

NOTES

Date :

```
# scatter plot of data points.
sns.scatterplot(x="x1", y="x2", hue='class', data=data
```

seaborn library

```
# meshgrid (for more attractive ==========
x_min, x_max = data['x1'].min()-1, data['x1'].max()+1
y_min, y_max = data['x2'].min()-1, data['x2'].max()+1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

Z = np.dot(np.c_[xx.ravel(), yy.ravel()], w) + b
Z = np.sign(Z)
Z = np.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3, color=['blue', 'red'])

plt.title("Decision boundary")
plt.xlabel("Feature 1 (x1)")
plt.ylabel("Feature 2 (x2)")
plt.legend()
plt.grid(True)
```

NOTES Date :

Cost Function for Logistic Regression

Training set

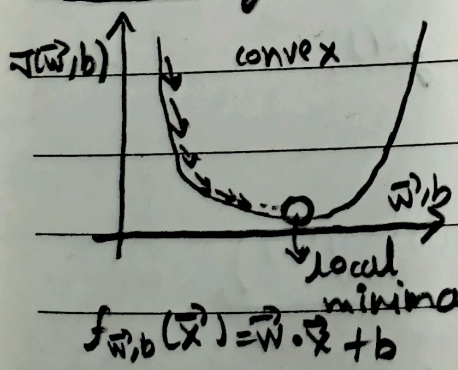
	tumor size (cm)	...	Patient's age	malignant?	$i=1,\dots,m$ ← training example
$i=1$	10		52	1	$j=1,\dots,n$ ← features
:	2		73	0	target y is
:	5		55	1	0 or 1
$i=m$	...		...	...	$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$

⇒ How to choose $\vec{w} = [w_1\ w_2\ \cdots\ w_n]$ & b?

Squared error cost fn

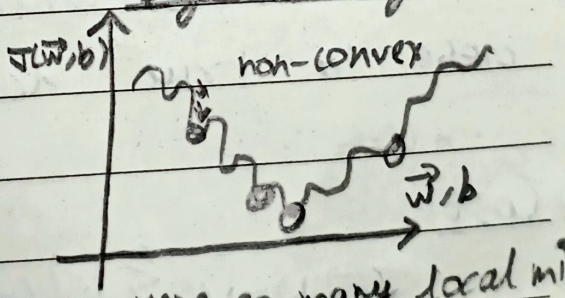
$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{2}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2$$

Linear Regression



$f_{\vec{w},b}(\vec{x}) = \vec{w}\cdot\vec{x} + b$

Logistic Regression



Here so many local minima

$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$

- If we are using squared error cost fn for logistic regression, it will create so many local minima which will create the problem.

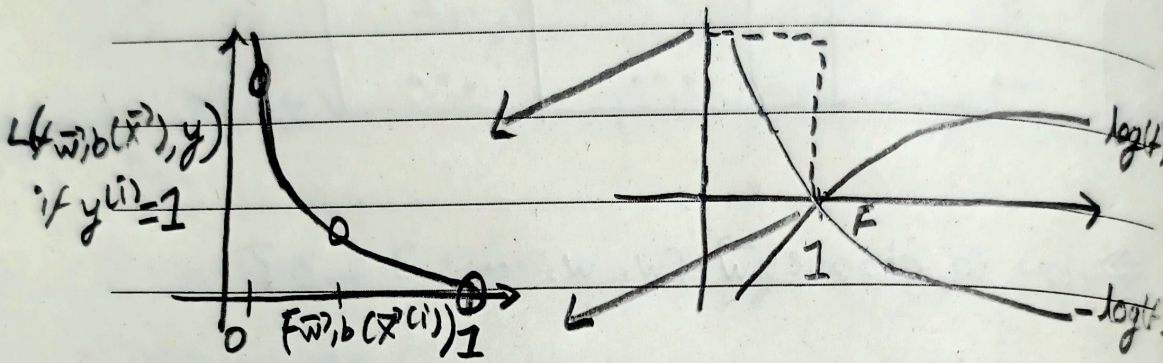
NOTES

Date :

- So, for logistic Regression we have loss function.

Logistic loss function

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)}=1 \\ -\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)}=0 \end{cases}$$



As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$ then loss $\to 0$ ☺

As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$ then loss $\to \infty$ ☹

⇒ Loss is lowest when $f_{\vec{w},b}(\vec{x}^{(i)})$ predicts close to true label $y^{(i)}$.

⇒ **Cost**

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)}_{\text{Loss}}$$

$$= \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)}=1 \\ -\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)}=0 \end{cases}$$

it can reach a global minimum (convex)

NOTES

Date :

Simplified loss function

$$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = \begin{cases} -\log(f_{\vec{w},b}(\vec{x}^{(i)})) & \text{if } y^{(i)}=1 \\ -\log(1-f_{\vec{w},b}(\vec{x}^{(i)})) & \text{if } y^{(i)}=0 \end{cases}$$

$$\Rightarrow L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = -y^{(i)}\log(f_{\vec{w},b}(\vec{x}^{(i)})) - (1-y^{(i)})\log(1-f_{\vec{w},b}(\vec{x}^{(i)}))$$

Let $y^{(i)}=1$,

$$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = -\log(f_{\vec{w},b}(\vec{x}^{(i)}))$$

Let $y^{(i)}=0$,

$$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = -\log(1-f_{\vec{w},b}(\vec{x}^{(i)}))$$

Proved.

$\Rightarrow$ Simplified cost function

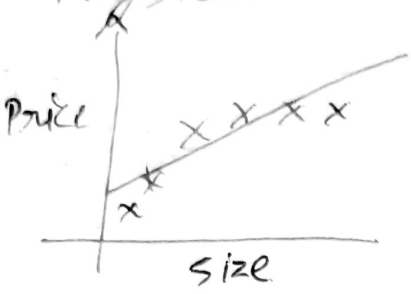
$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left[L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)})\right]$$

$$\Rightarrow J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(f_{\vec{w},b}(\vec{x}^{(i)})) + (1-y^{(i)})\log(1-f_{\vec{w},b}(\vec{x}^{(i)}))\right]$$

$\uparrow$ ('-') — (Loss function)

The problem of overfiting

Regration Examble



$w_1 x + b$

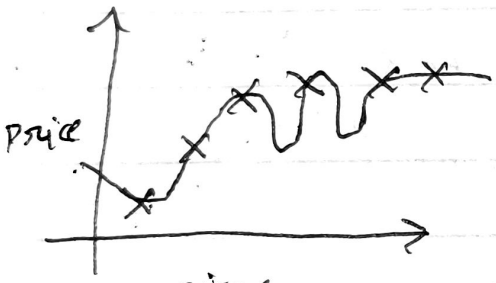
Underfit

- Does not fit the training set well
  → high bias

$w_1 x + w_2 x^2 + b$

Just right

- Fits training set pretty well
  → Generalization



$w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

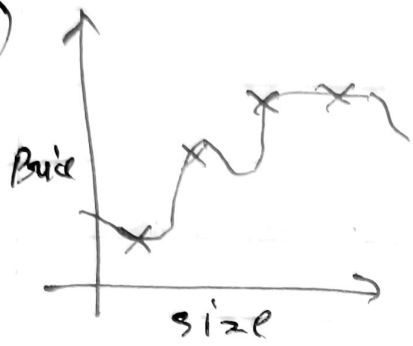
- Fits the training set extremely well

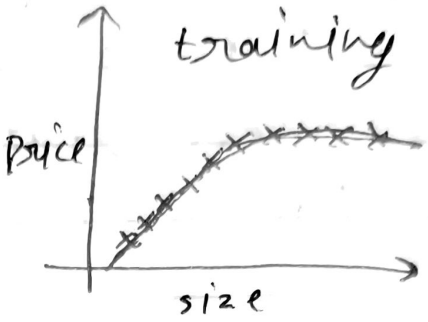
Overfit

→ high variance

Addressing overfitting

① collect more training example

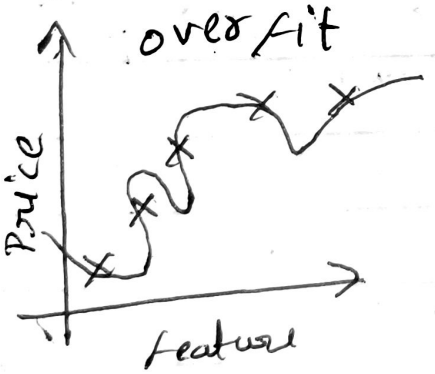




② select feature to include/exclude

All feature ↓ insufficient data ↓ overfit	selected feature + insufficient data ↓ just right.

③ * Regularization (Reduce the size of parameters)



$f(x) = 28x - 385x^2 + 39x^3 - 174x^4 + 100$

large values for w_j

$f(x) = 13x - 0.023x^2 + 0.00014x^3 - 0.001x^4 + 10$

small values for w_j

summarise

options:

1). collect more data

2). select features

3). Reduce size of parameters

Cost function for Regularization

$w_1, w_2, w_3 \ldots\ldots, w_{100}, b$ n features $n=100$

regularization term

$$J(\vec{w}, b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{m} w_j^2$$

"lambda" $\lambda > 0$

regularization parameter

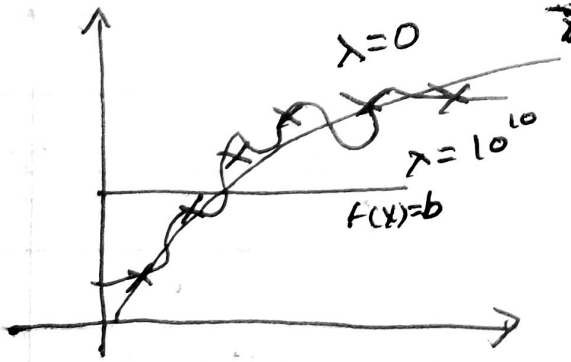
mean squared error

regularization term

$$\Rightarrow \min_{\vec{w}, b} J(\vec{w}, b) = \min_{\vec{w}, b}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{i=1}^{n} w_j^2\right]$$

fit data ↘ ↙ keep w_j small

λ balance both goals



choose $\lambda = 0$, overfit

choose $\lambda = 10^{10}$, underfit

choose λ in b/w, just right

Regularized linear regression

cost fn:

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

Gradient descent

repeat {

$$w_j = w_j - \alpha \frac{d}{dw_j} J(\vec{w},b) \rightarrow = \frac{1}{m}\sum_{i=0}^{m}\left(f_{(\vec{w},b)}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m} w_j$$

$$b = b - \alpha \frac{d}{db} J(\vec{w},b) \rightarrow = \frac{1}{m}\sum_{i=0}^{m}\left(f_{(\vec{w},b)}(\vec{x}^{(i)}) - y^{(i)}\right)$$

} simultaneous update

dont have to regularize b.

Implementation of gradient descent

repeat {

$$w_j = w_j - \alpha\left[\frac{1}{m}\sum_{i=0}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m} w_j\right]$$

$$b = b - \alpha\left[\frac{1}{m}\sum_{i=0}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)\right]$$

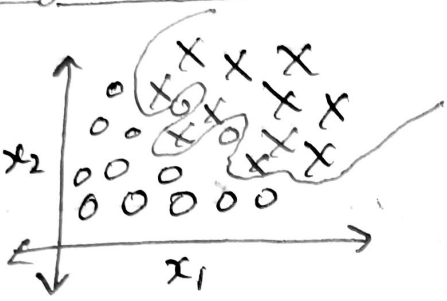
} simultaneous update

option 1 {

rewrite: $w_j = w_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_i^{(i)}\right]$

rest of the previous code.

Regularized Logistic Regression



$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 x_2 + \dots + b$$

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-z}}$$

cost f^n:

simple cost f^n

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f(\vec{x}^{(i)})\right)\right] + \underbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2}_{\text{Regularized term}}$$

Gradient descent

repeat {

$$w_j = w_j - \alpha\left[\frac{d}{dw_j}J(\vec{w},b)\right] = \frac{1}{m}\sum_{i=1}^{n}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m}w_j$$

$$b = b - \alpha\left[\frac{d}{db}J(\vec{w},b)\right] = \frac{1}{m}\sum_{i=1}^{n}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

} simultaneous update

$$\Rightarrow f_{\vec{w},b} = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}} = \frac{1}{1+e^{-z}}$$