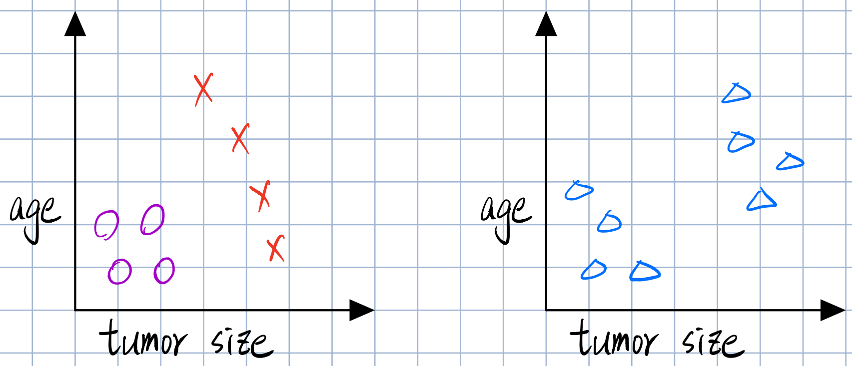
Supervised Learning v.s. Unsupervised Learning

Supervised learning is to train a function to map the input x to output y given right answers.

Regression: Predict a number, infinitely many possible outputs.

Classification: Predict categories, small number of possible outputs.
(Logistic Regression)



Unsupervised Learning tries to find "structure" in the data, which only comes with the input x, but not output y.

Clustering: Group similar data points together.

Dimentionality Reduction: Compress data using fewer numbers.

Anamoly Detection: Find unusual data points.

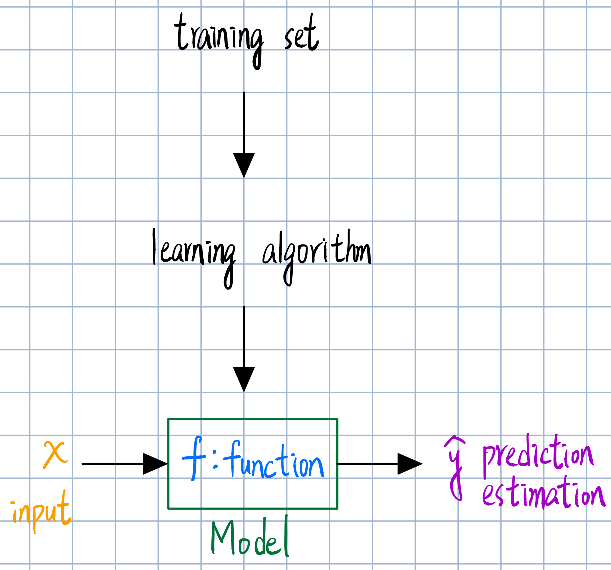
Regression Model

x: input variable, feature

y: output / target variable

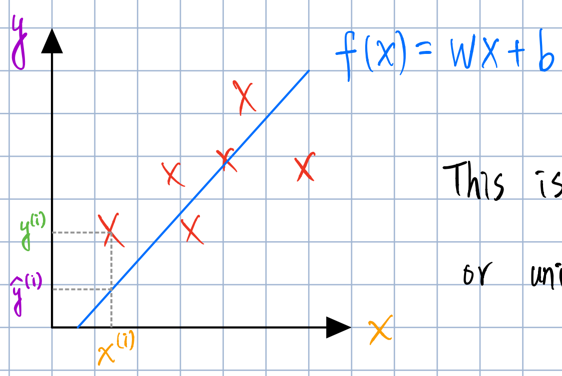
m: number of training examples.

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



How to represent function?

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



This is called linear regression with one variable.

or univariate linear regression.

Cost function: Squared error cost function

Model: $f_{w,b}(x) = wx + b$, w, b: parameters / coefficient / weights

The goal is to find w, b such that $\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$

Thus, the cost function J can be defined as

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2, \text{ where } \hat{y}^{(i)} = f(x^{(i)})$$

Goal: $\underset{w,b}{\text{minimize}} \; J(w,b)$

Cost function intuition:

Given 3 training samples $(1,1), (2,2), (3,3)$. And assume $b=0$.

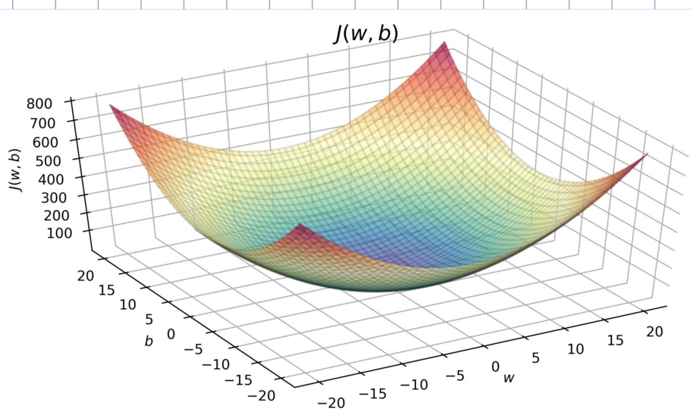
$f_w(x^{(i)}) = w x^{(i)}$ is a straight line that passes through $(0,0)$ given a fixed w.

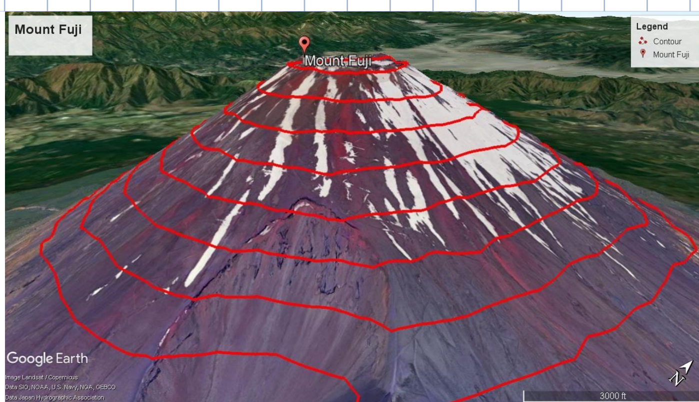
The $J(w)$ is a parabola w.r.t w.

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (w x^{(i)} - y^{(i)})^2.$$ Recall $y = ax^2 + bx + c$

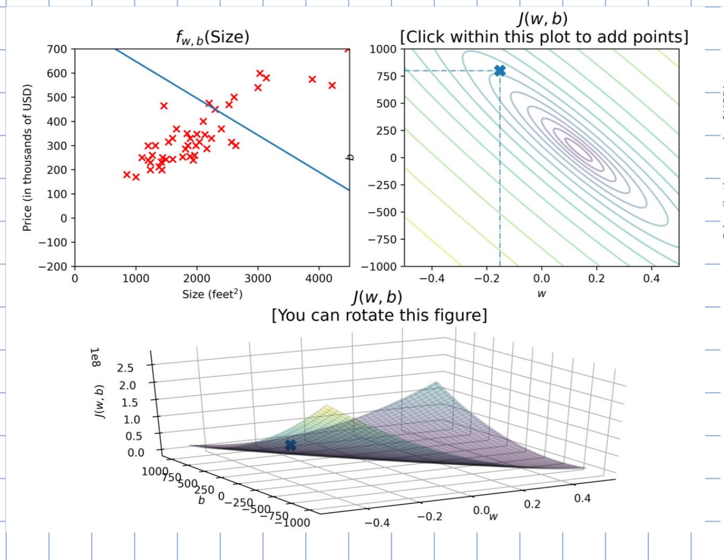
Here, b is set to 0 for simplicity. But b is ESSENTIAL in real-world problems. b is viewed as biases in neural networks. I think considering b as important as W is crucial. You should never set b as 0. Because if I were to set b as 0, first it doesn't make sense. Second, I can never find optimal weights.

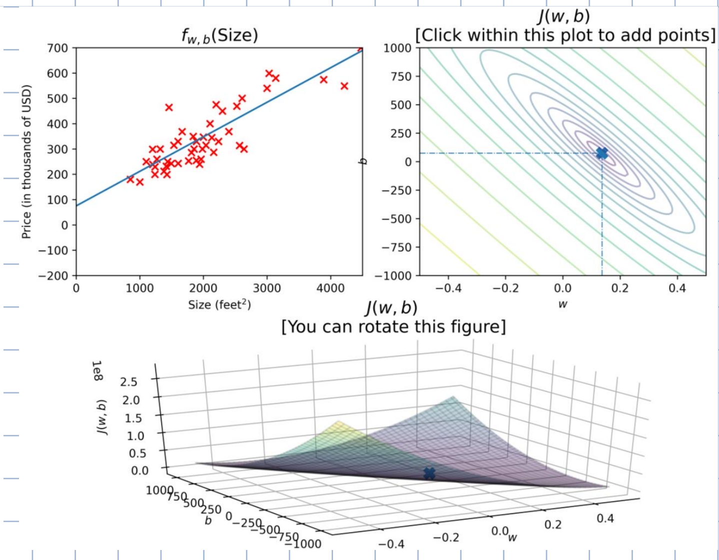
Now, here is how $J(w,b)$ looks like

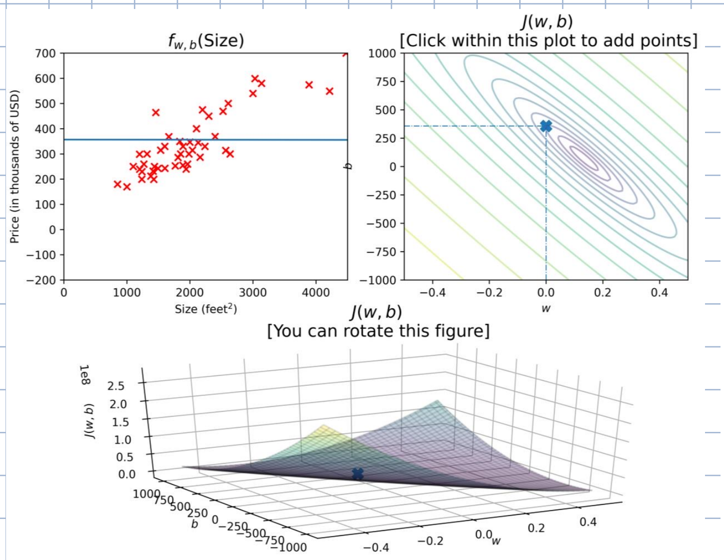


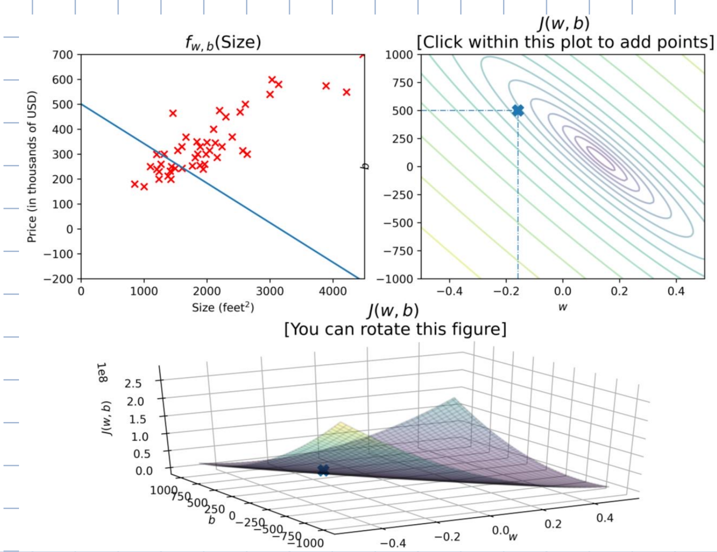


Visualization





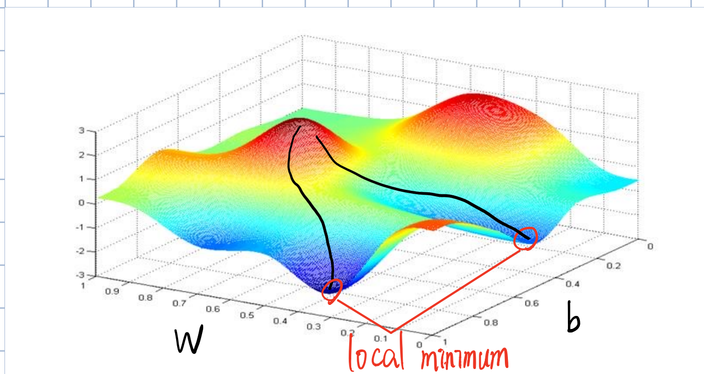




Training the Model with Gradient Descent

Given a general function $J(W_1, W_2, \ldots, W_n, b)$

Goal : $\min\limits_{W_1 \ldots W_n, b} J(W_1, W_2, \ldots, W_n, b)$



This surface is not squared error cost and is not linear regression.

Gradient Descent Algorithm.

repreat until convergence {

$$W = W - \alpha \frac{\partial J(W,b)}{\partial W}$$

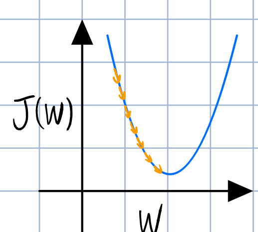
Partial Derivative

α : learning rate

$$b = b - \alpha \frac{\partial J(W,b)}{\partial b}$$

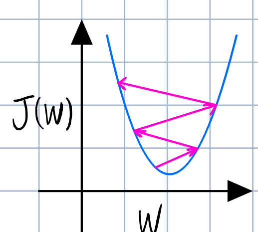
}

How to choose learning rate α ?

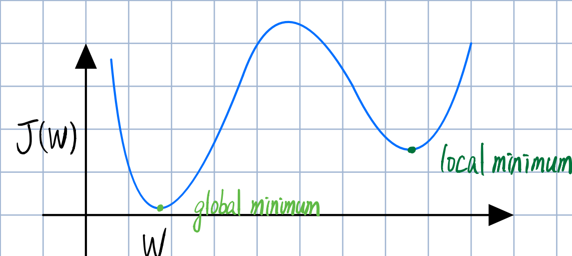


if α is too small,
gradient descent may be slow.

it can reach local minimum
with fixed learning rate.



if α is too large,
gradient descent may overshoot
fail to converge, diverge.



if it's stuck at local minimum, it can never reach
global minimum.

Combining Gradient Descent and Linear Regression

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

$$= \frac{1}{\cancel{2}m}\sum_{i=1}^{m}\left(wx^{(i)}+b-y^{(i)}\right)\cdot \cancel{2}x^{(i)} = \boxed{\frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)})-y^{(i)}\right)x^{(i)}}$$

The same derivation can be applied to b.

Thus, the gradient descent algorithm:

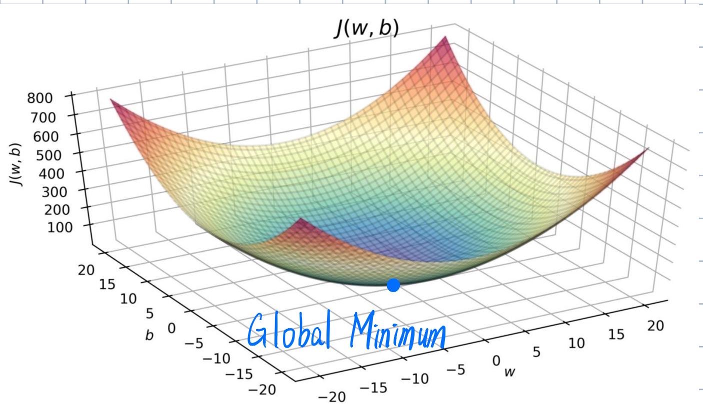
repeat until convergence {

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

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

}

The last note is that if squared error cost is used, there will always be 1 local minimum, which is a global minimum. Because it's a convex function.



Outline:

1. Start with random w, b
2. Keep changing w, b to reduce $J(w,b)$
3. Until settling at or near a minimum.

Batch gradient descent:

Each step of gradient descent uses all the training examples.