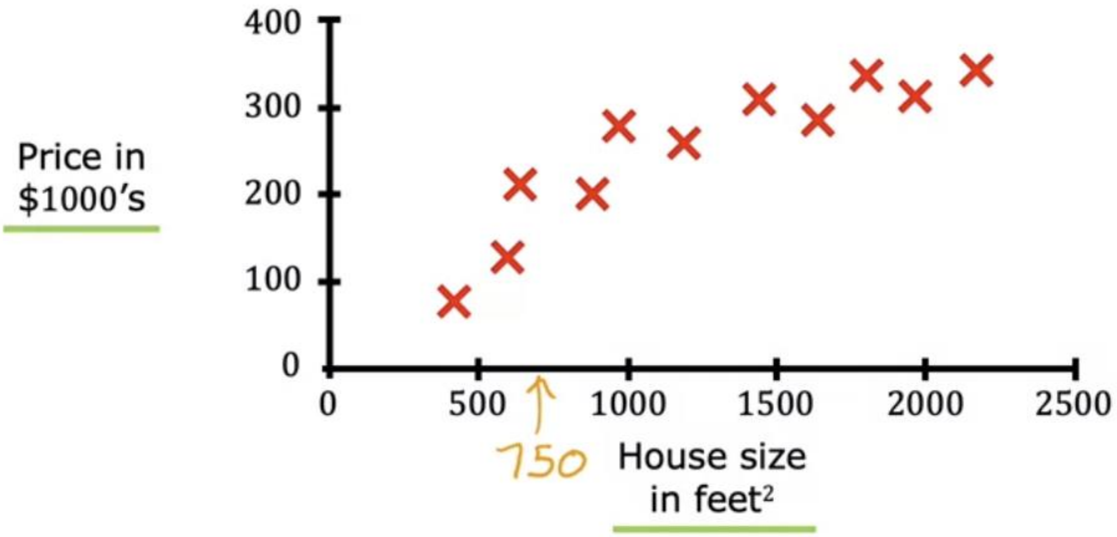
Week 1: Introduction to Machine Learning

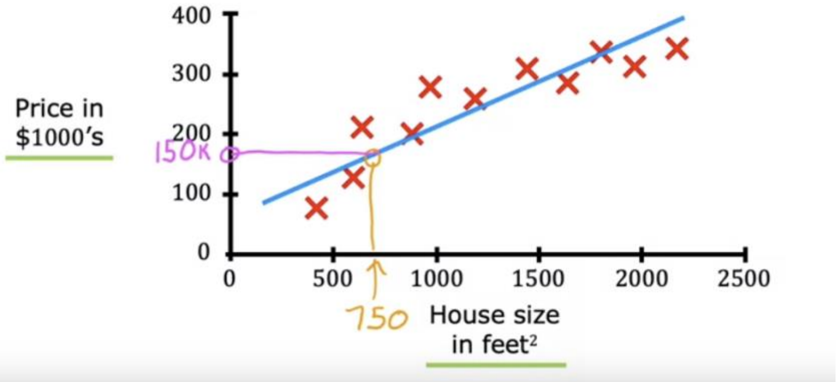
<u>Supervised vs. Unsupervised Machine Learning</u>

- Supervised Learning
  - Algorithms learn **input → output** or **X (input) → Y (output label)**
    - You give the algorithm examples with the correct output so the algorithm can learn for a given input
      - → over time the algorithm will be able to take the input variable (X) without the output variable and give a reasonable output due to the learning it did
      - Algorithms after learning can take a new input variable (X) and produce a new output variable (Y) that is reasonable
    - Example: Ad, User info (X) → Click? (0/1) – *Application is online advertising*
      - Train algorithm with information regarding user info and the ad and whether or not they click the advertisement. If they do, you know what users are likely to click on a certain kind of advertisement
  - Example

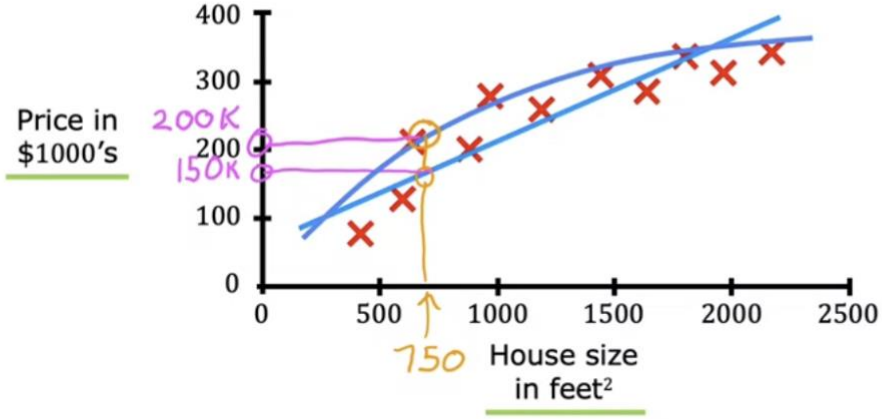


- There are some input variables and output prices already listed
- A friend wants to the know the price for a 750 ft² home

- Algorithm can provide a straight line fit to determine the price of the house = approximately $150K



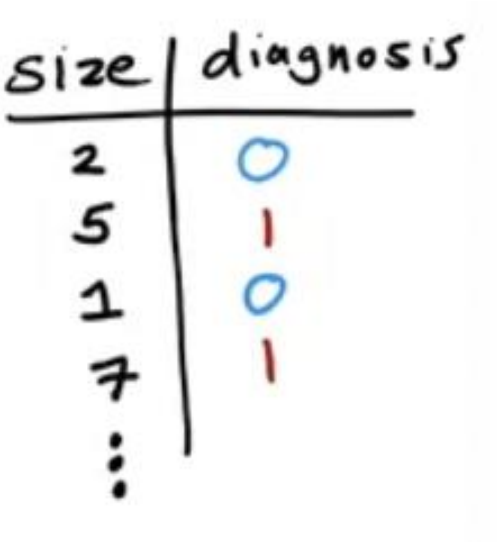
- Can also use a curved line - Price of house comes out closer to $200 K



- This is supervised learning because we gave the algorithm a data set with answers to predict a new X value
  - In this example we used a process called **regression** in order to get an output of infinitely many possible values
    - *One of the major supervised learning algorithms*

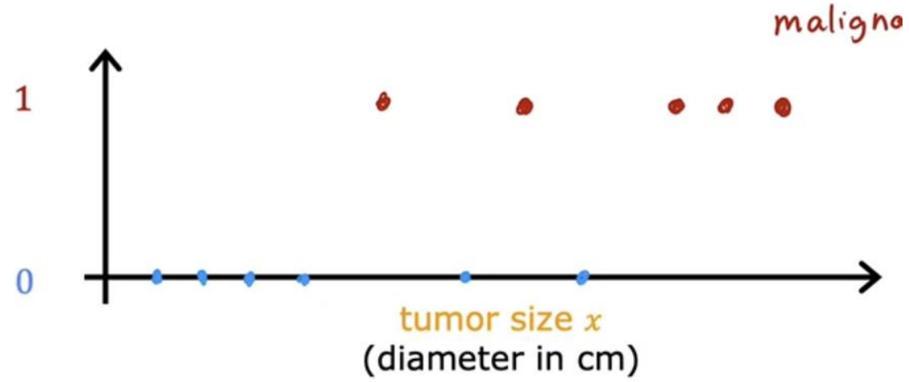
- Classification algorithm
  - Example: Trying to classify whether tumors are benign or malignant

- Given data set to algorithm

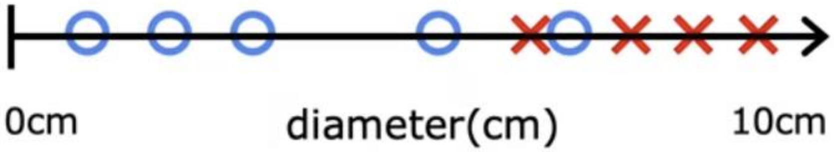


size	diagnosis
2	0
5	1
1	0
7	1
⋮	

  - Size of Tumor vs Diagnosis (0 = benign, 1 = malignant)

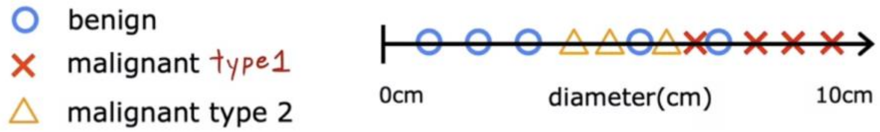


  - Blue dots on 0 axis = benign, Red dots on 1 axis = malignant
  - In this algorithm it is different since the algorithm can only predict to classify two possible diagnoses/output variables. In regression there are an infinite number of output variable possibilities
  - Another way to show this is:

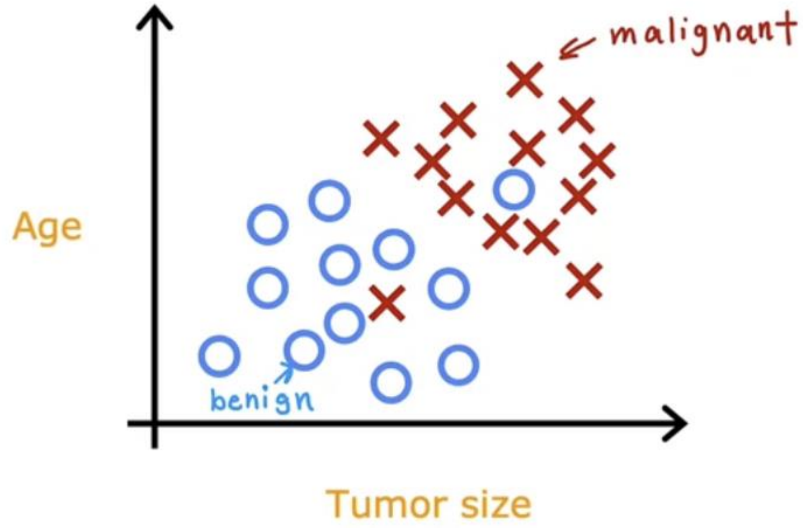


    - Algorithm will try to predict whether a new tumor (not in the data set is benign or malignant)

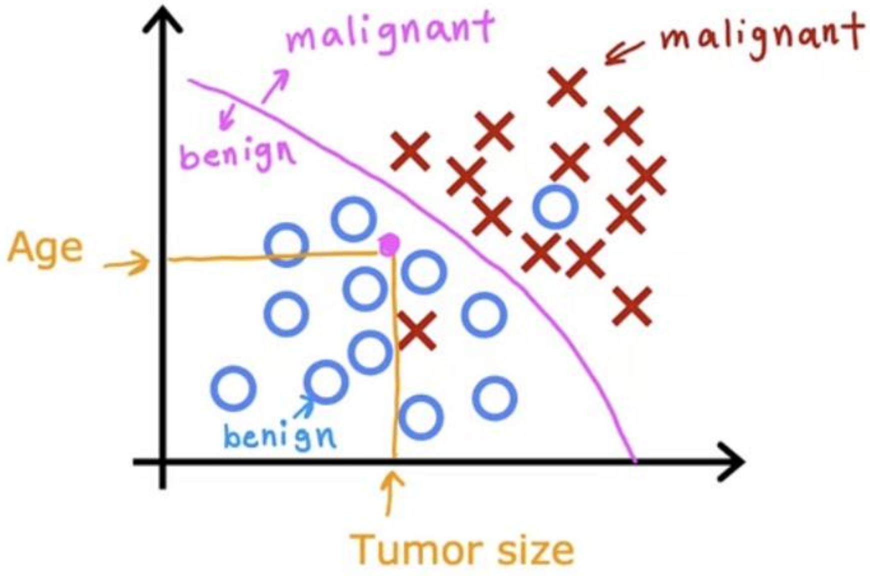
        - Classification can also provide an output where there is more than two options



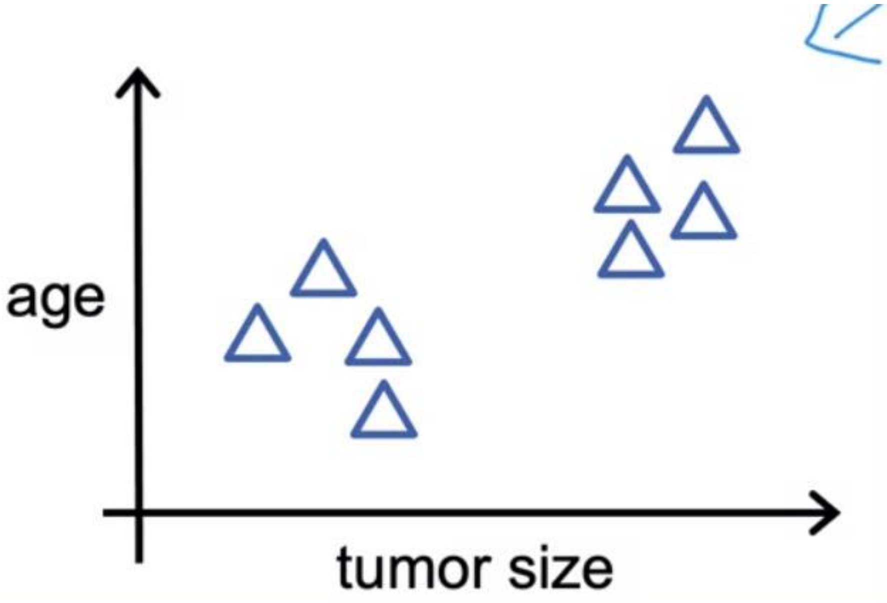
    - In classification, output is often referred to as class or category
        - Could be non-numeric
- Example: **Two or more inputs** could also be used to predict whether a tumor is malignant or benign
    - Age and tumor size are inputs



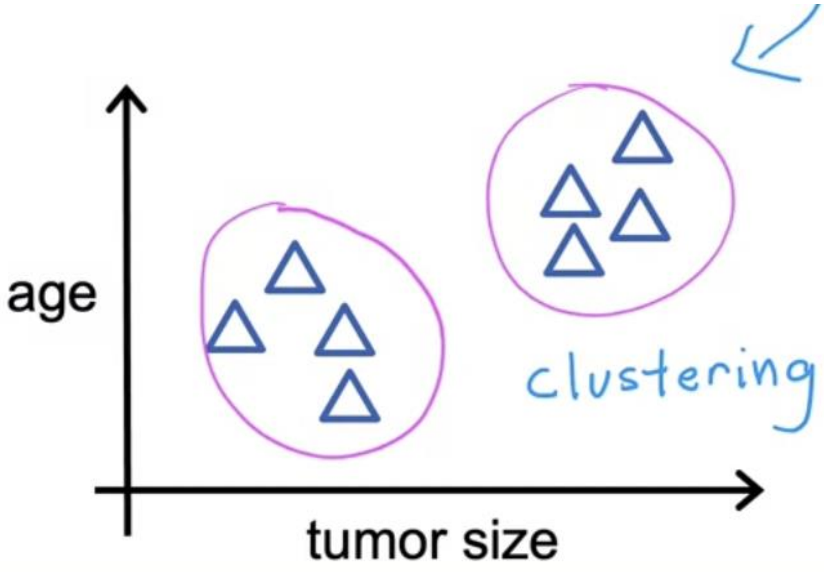
    - To predict whether a person has a tumor or not, the leanring algorithm might use a boundary approach



- Unsupervised Learning
  - Data is given without any output labels (Y value), but is given input values (X values)
    - Example: Not given information regarding whether tumors are benign or malignant. Our goal is just to find something interesting in the unlabeled data

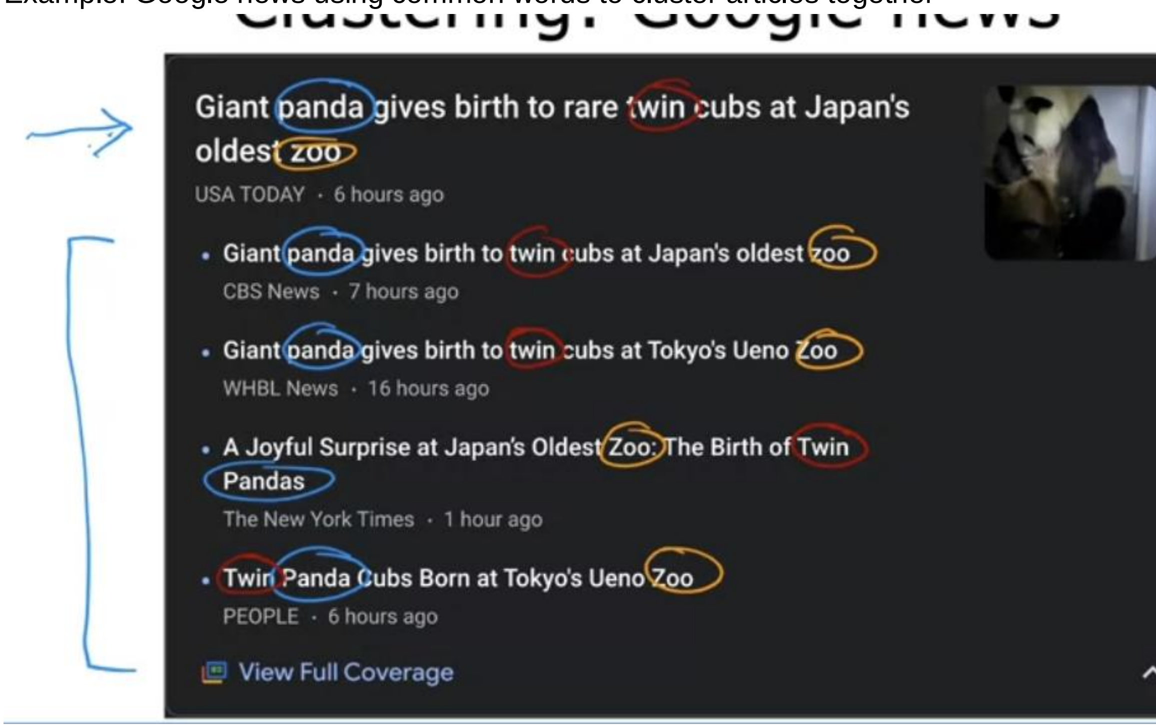


      - Algorithm may decide data can be assigned to different clusters



        - This is known as a **clustering algorithm - grouping related values together**

- Example: Google news using common words to cluster articles together



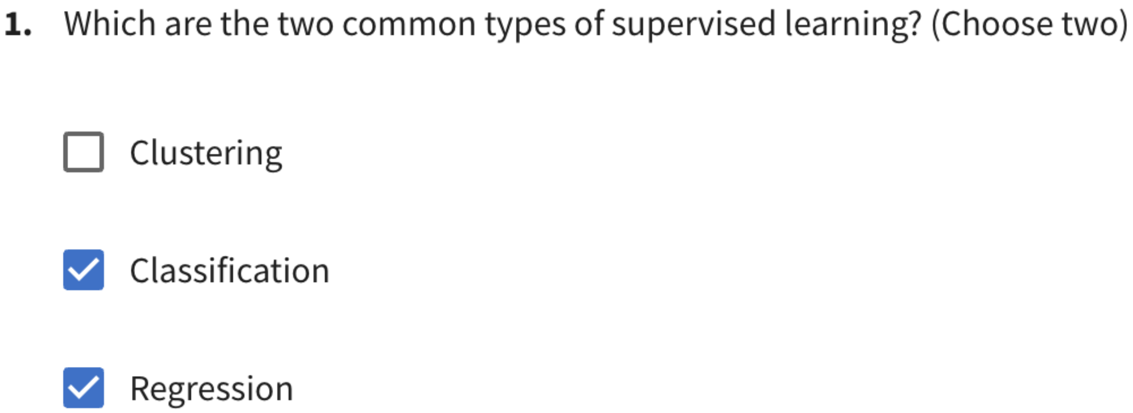
Clustering: Google news

Giant panda gives birth to rare twin cubs at Japan's oldest zoo
USA TODAY · 6 hours ago

- Giant panda gives birth to twin cubs at Japan's oldest zoo
  CBS News · 7 hours ago
- Giant panda gives birth to twin cubs at Tokyo's Ueno Zoo
  WHBL News · 16 hours ago
- A Joyful Surprise at Japan's Oldest Zoo: The Birth of Twin Pandas
  The New York Times · 1 hour ago
- Twin Panda Cubs Born at Tokyo's Ueno Zoo
  PEOPLE · 6 hours ago

- Another possible algorithm - **Anomaly Detection**
  - Used to find usual data points
- Another possible algorithm - **Dimensionality Reduction**
  - Take big data set → compress data set using fewer numbers (without losing information)

Practice Quiz: Supervised vs. unsupervised learning



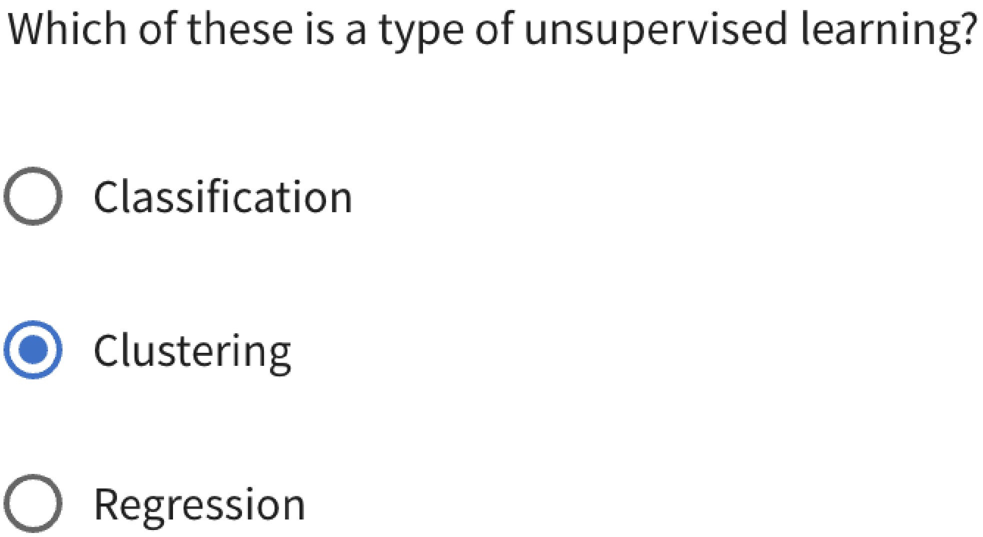
1. Which are the two common types of supervised learning? (Choose two)

□ Clustering

☑ Classification

☑ Regression

2.



Which of these is a type of unsupervised learning?

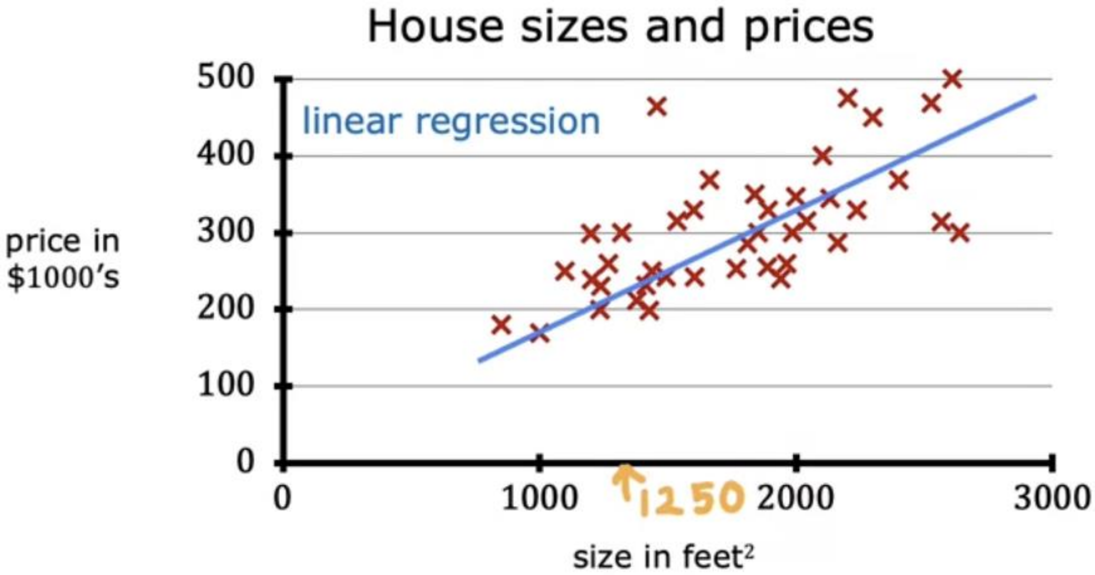
○ Classification

● Clustering

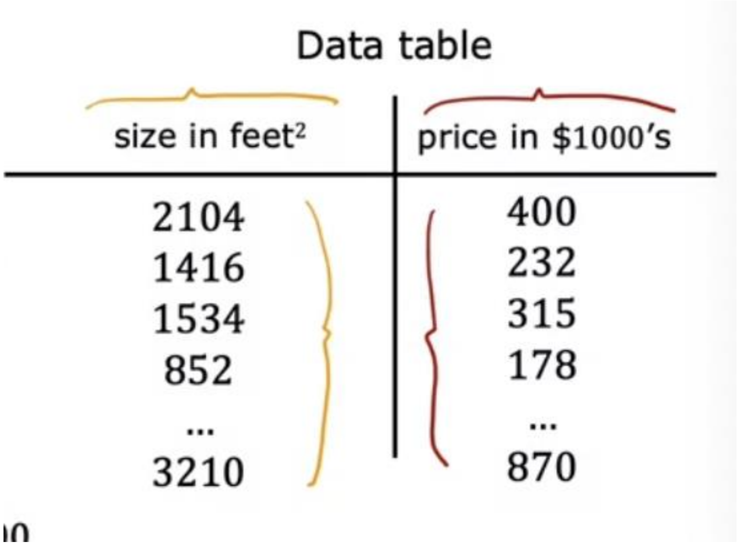
○ Regression

Regression Model

- Linear Regression Model
    - Can fit a model to a straight line (Example below)



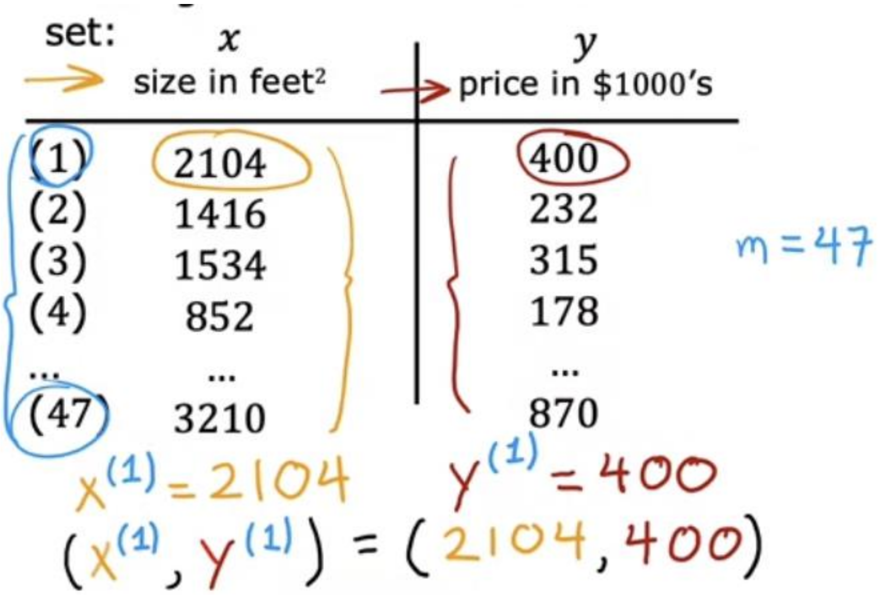
        - Cost of a 1250 ft^2 house is ~220k
        - Example of a supervised learning model since the data has "right answers"
            - Regression model since it predicts a number
                - Infinitely many possible outputs
    - Could be helpful to see the data as a data table



Data table

size in feet²	price in $1000's
2104	400
1416	232
1534	315
852	178
...	...
3210	870

- Terminology and Notation



set:	x size in feet²	y price in $1000's
(1)	2104	400
(2)	1416	232
(3)	1534	315
(4)	852	178
...	...	...
(47)	3210	870

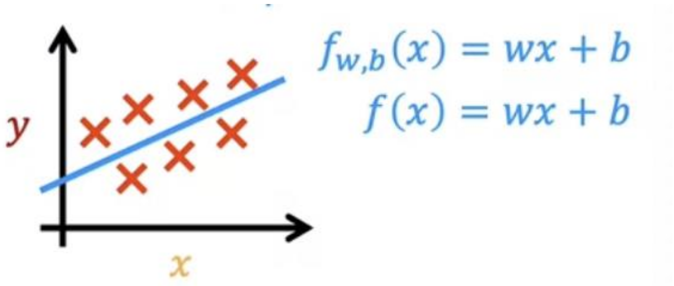
$m = 47$

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

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

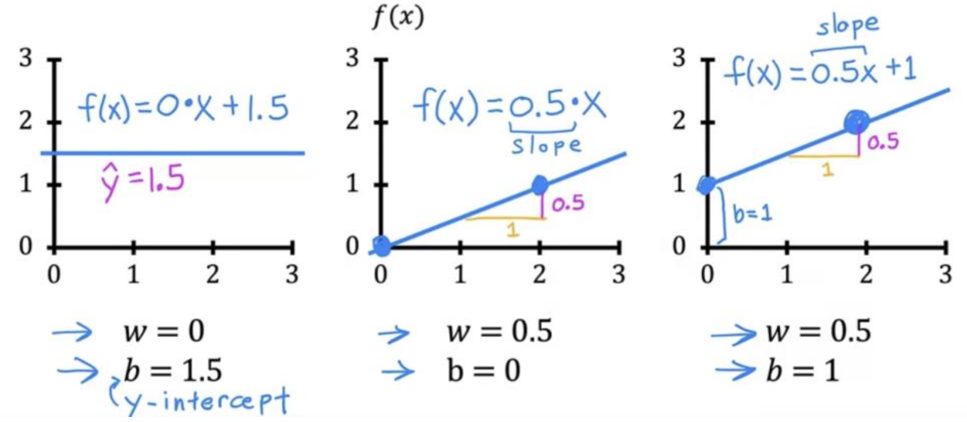
  - Terminology
    - The data set that you use to train a particular model is called a **training set**
  - Notation
    - x = “input” variable or feature
    - y = “output” variable or “target variable”
    - m = number of training examples
    - (x, y) = single training example
    - $(x^{(i)}, y^{(i)})$ = ith training example
      - i refers to a specific row in the table or the ith training example
      - Index of the training set
      - **Not exponent**
- Supervised learning has both input and outputs in a *training set* → produce a *training algorithm* → produces a *function (f)* or hypothesis or model
  - The *function* takes a new input (x) and produces a new output prediction (ŷ)
    - The output prediction is the estimated value of y (target)
      - This is the actual value of y
  - How to represent *f?*
    - To represent as a straight line
      - $f_{w,b}$ (x) = wx +b or could be written as f(x)
        - This will give us the prediction of ŷ

- Generates a best fit line

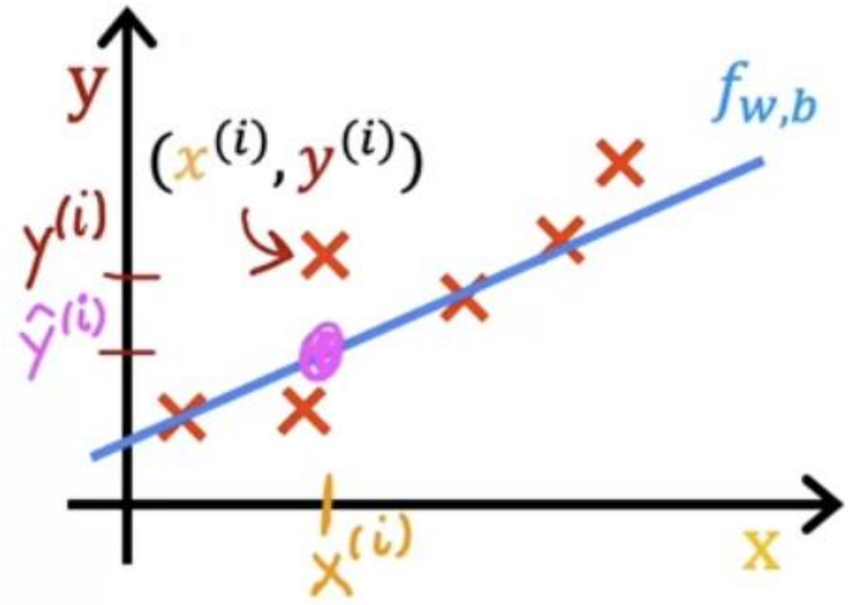


  - Linear regression with one variable (single feature x) – **univariate linear regression**

- ➔ Cost Function Formula
  - Going back to the linear regression function model (*$f_{w,b}(x) = wx+b$*)
    - *w,b* are parameters/coefficients/weights
      - More specifically *w* is known as the **slope** and the *b* value is the **y-intercept**
      - Variables you can adjust during training to improve the model
      - Examples:

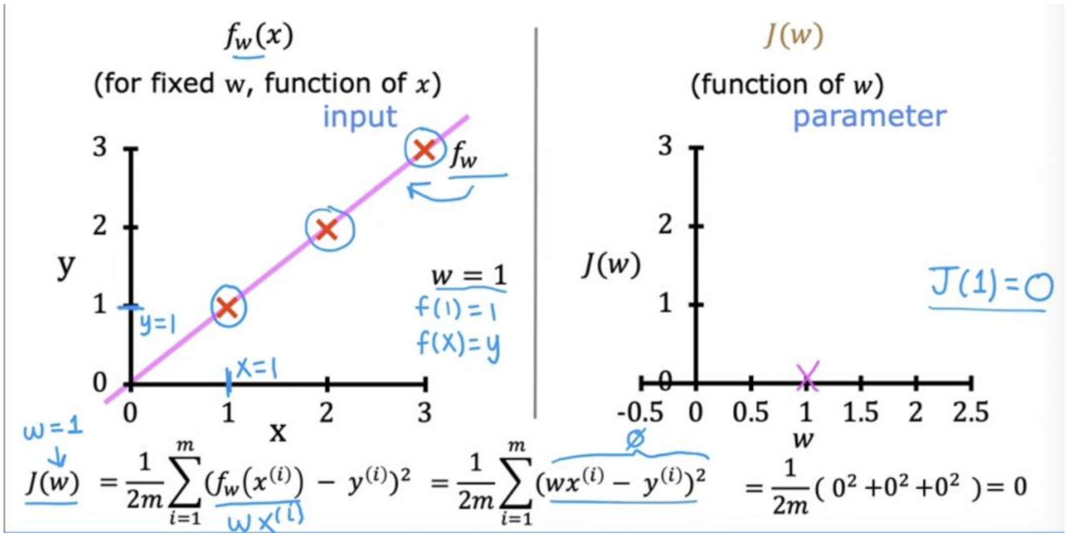


        - Example with training set:



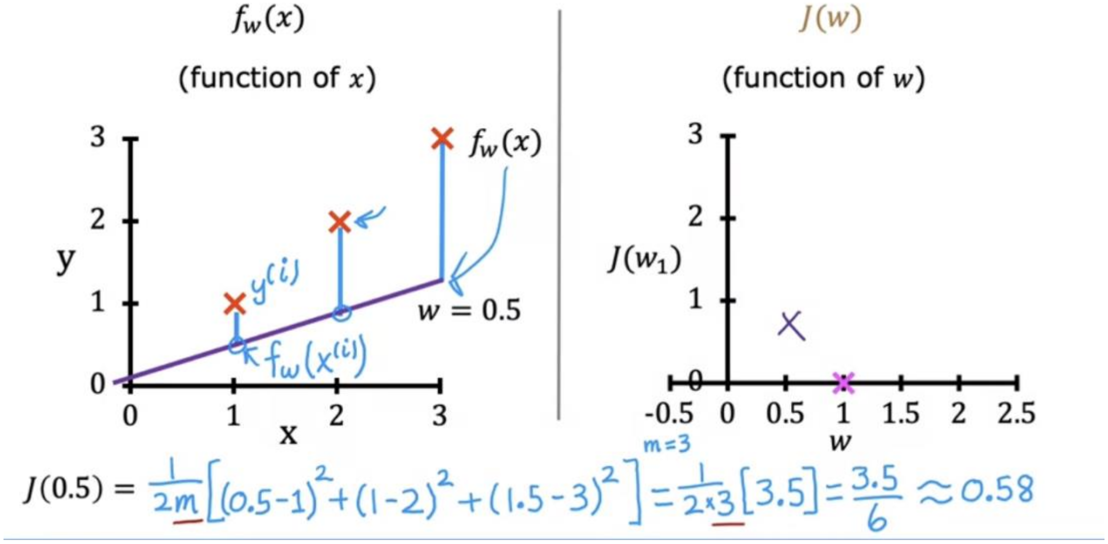
            - The line passing through the points in the data set is roughly passing through the training examples
            - To predict a ŷ value: $\hat{y}^{(i)} = f_{w,b}(x^{(i)})$ or $f_{w,b}(x^{(i)}) = wx^{(i)} + b$
            - To find the best fit line ŷ should be close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$
    - Cost function
        - $\hat{y} - y = error$ error → how far off from the target the predicted value is
        - $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m} \quad (\hat{y}^{(i)} - y^{(i)})^2 \text{ or } J(w,b) = \frac{1}{2m}\sum_{i=1}^{m} \quad (f_{w,b}(x^{(i)}) - y^{(i)})^2$
            - m = number of training examples
            - To prevent the summation function from getting bigger we divide it by the number of training examples (m)
                - In the function above you can see the summation is divided by 2m → this is because to make machine learning a little easier
- Cost function intuition
    - The goal of is to minimize *J(w,b)* by adjusting the values of *w* and *b*
    - Example: $f_w(x) \ = \ wx$ (in this case *b = 0*)
        - $\rightarrow Cost\ Function\colon J(w) \ = \ \frac{1}{2m}\sum_{i=1}^{m} \quad (f_w(x^{(i)}) \ - \ y^{(i)})^2$
            - $f_w(x^{(i)}) \ = \ wx^{(i)}$
            - Goal: Find value of *w* that minimizes *J(w)*

- Example: For w = 1



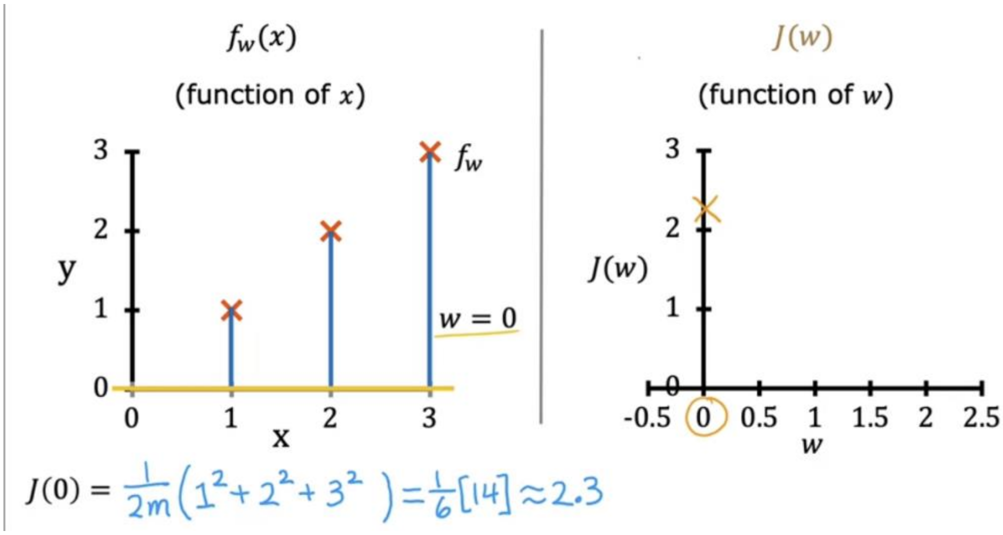
$$\overset{w=1}{\underline{J(w)}} = \frac{1}{2m}\sum_{i=1}^{m}(\underbrace{f_w(x^{(i)})}_{wx^{(i)}} - y^{(i)})^2 = \frac{1}{2m}\sum_{i=1}^{m}\overbrace{(wx^{(i)} - y^{(i)})^2}^{\emptyset} = \frac{1}{2m}(0^2 + 0^2 + 0^2) = 0$$

- Example: For w = 0.5



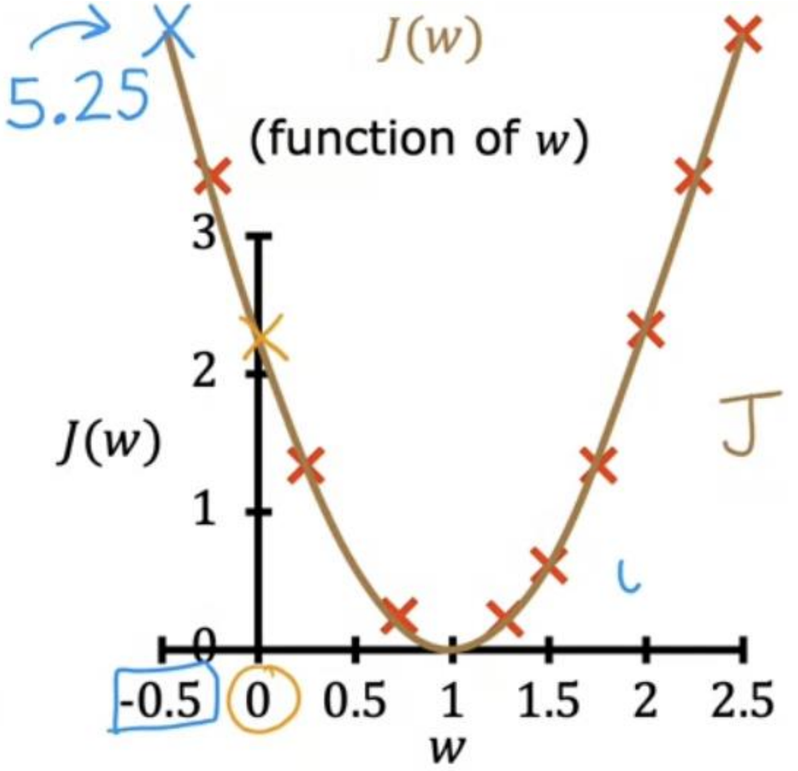
$$J(0.5) = \frac{1}{2\underline{m}}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right] = \frac{1}{2\times\overset{m=3}{\underline{3}}}[3.5] = \frac{3.5}{6} \approx 0.58$$

- Example: w = 0



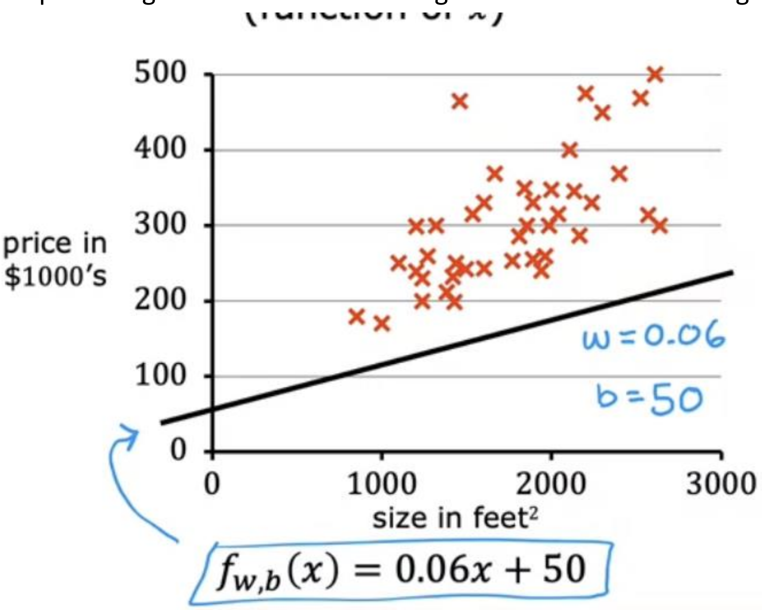
$$J(0) = \frac{1}{2m}(1^2+2^2+3^2) = \frac{1}{6}[14] \approx 2.3$$

- Example: Other w values

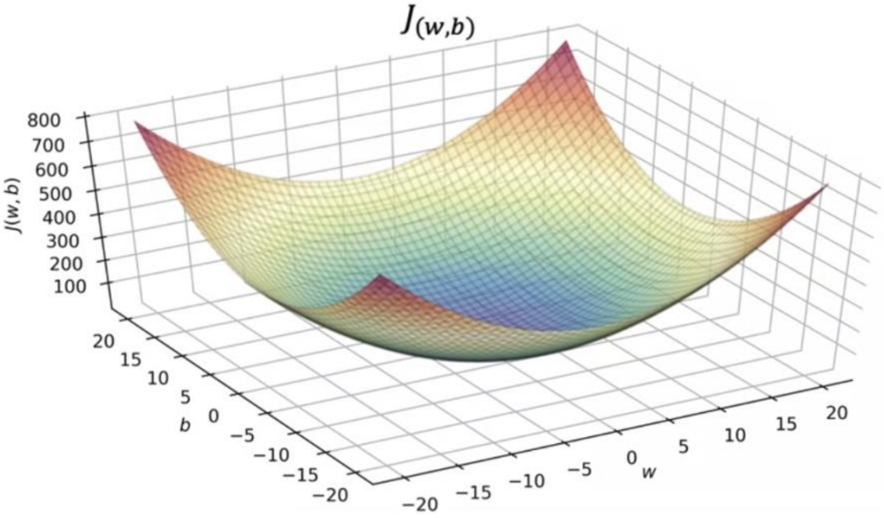


    - How to choose w?
        - Choose *w* to minimize *J(w)* to minimze the square errors
            - For this example you would choose w = 1
- Visualizing the Cost Function
    - We are going to now module *w* and *b*

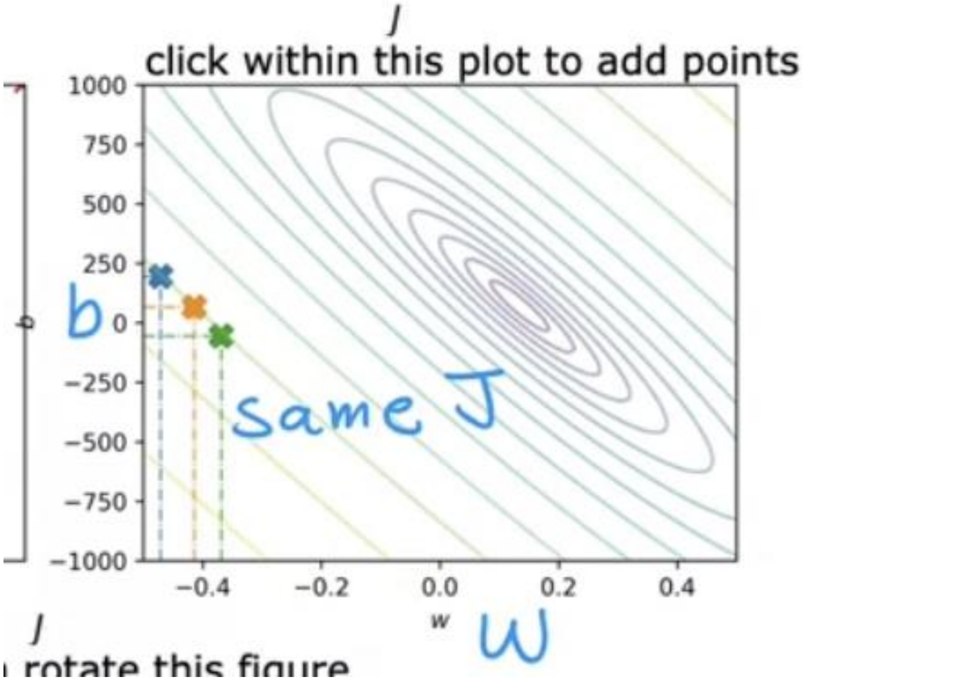
- Example: Using this data set and setting w = 0.06 and b = 50 we get the following line



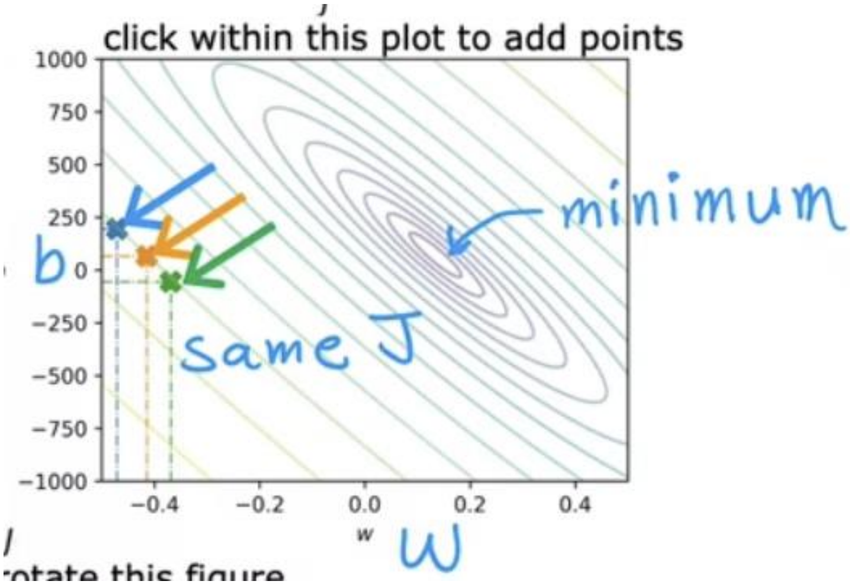
  - → We get this 3-dimensional graph (since we have to now plot w and b) for *J(w, b)*



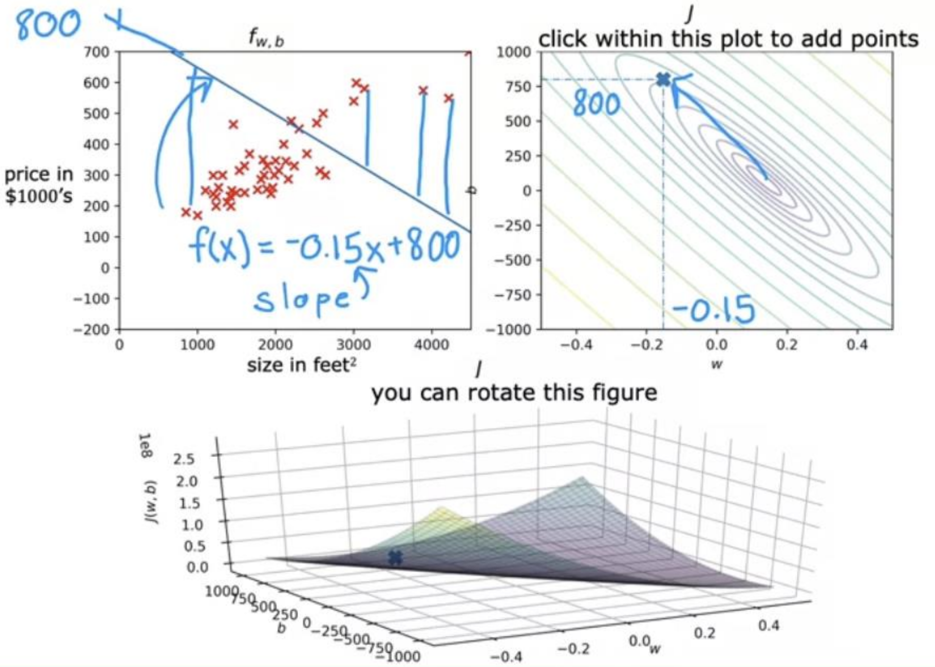
- While 3-D plots are nice, we can also use a contour plot to visualize



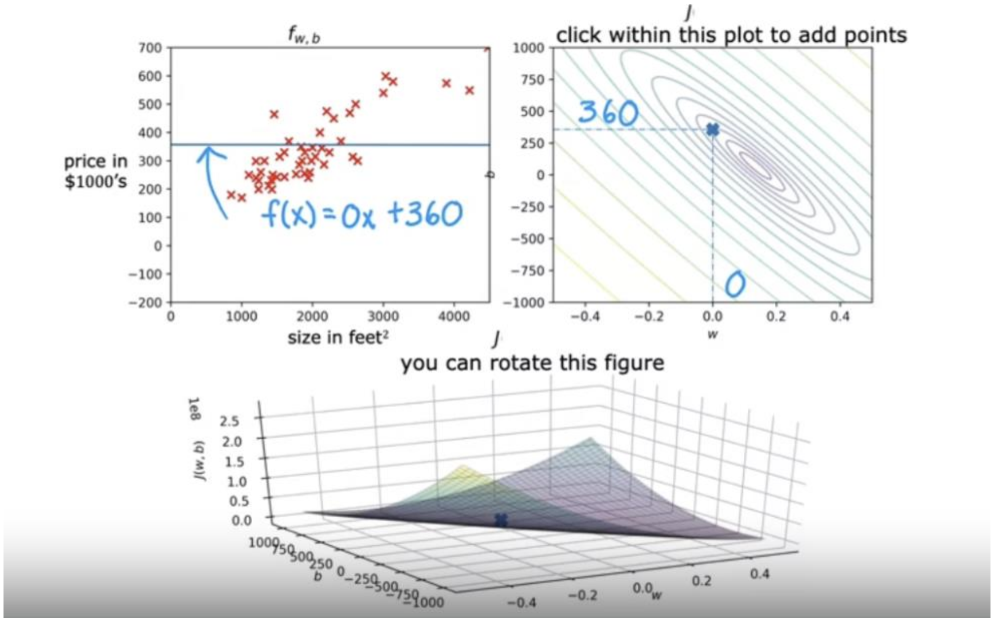
  - Y axis is b and X axis is w
  - Ovals show center points on 3D surface which are the same height or the same cost function (*J*)
    -
  - To get contour plot you have a 3-D plot and slice contours horizontally
    - This way you get all values with the same height
  - Bottom of the bowl where *J* is at a minimum, indicated ideal *w* and *b* values



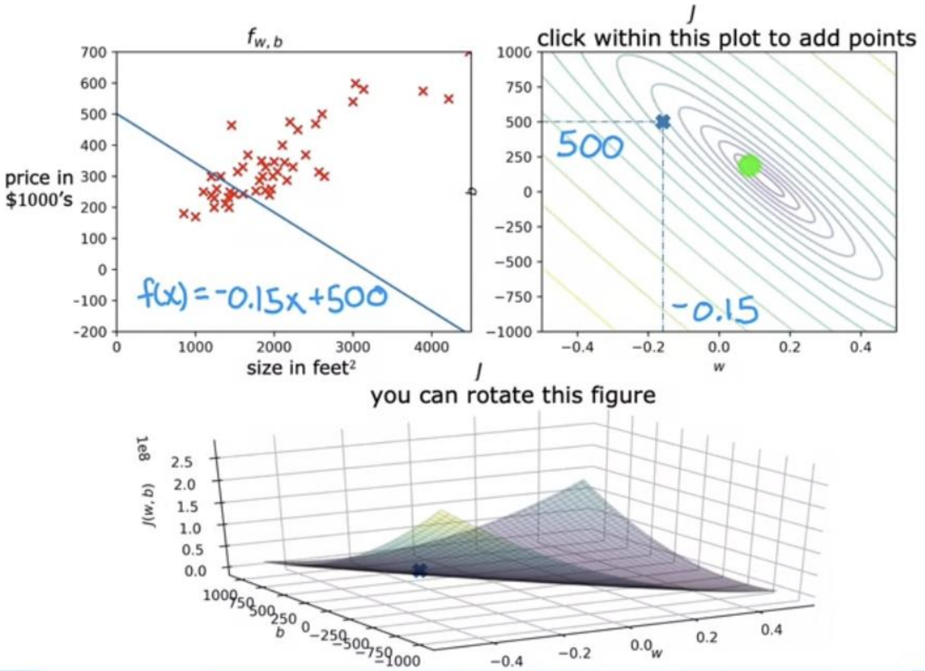
- Visualization Examples
  - Using the function, f(x) = -0.15x + 800, we can see there is a high error margin since the data points are far from the function we chose
    - w = -0.15
    - b = 800
    - → Get this contour plot



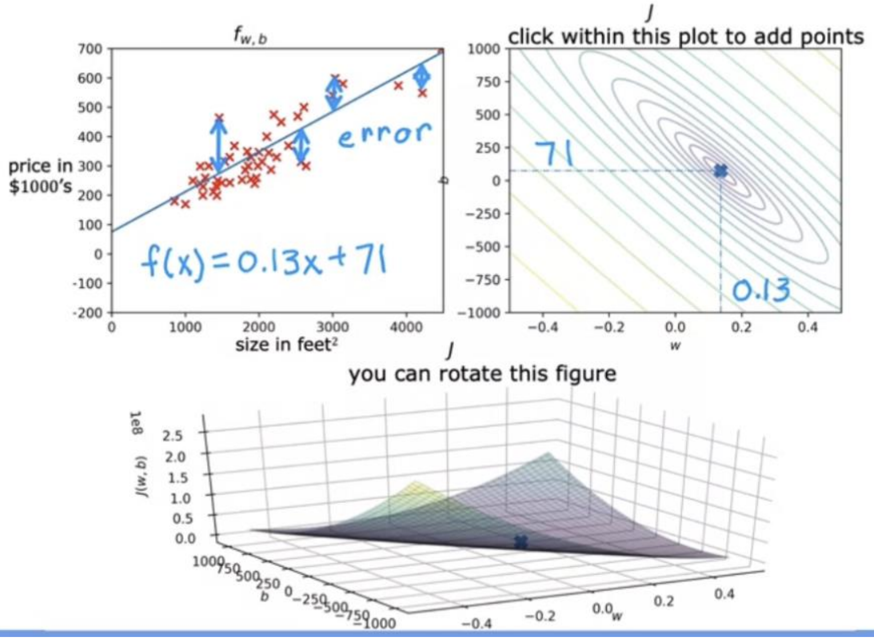
    - Poor since the value is far away from the minimum value
  - Another function, f(x) = 0x + 360



- And another function where f(x) = -0.15x + 500

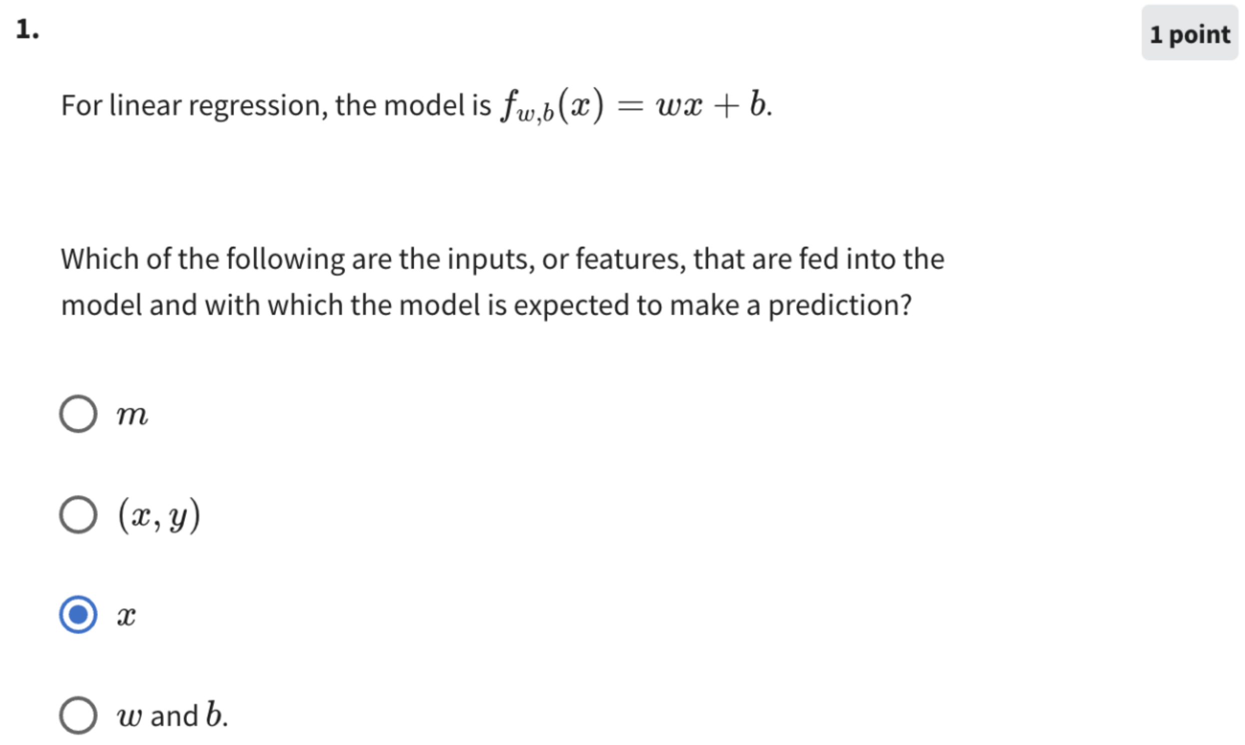


- And another where f(x) = 0.13x + 71



- Pretty good fit on contour graph with low error rate in regression plot

Practice Quiz: Regression Model



1.

1 point

For linear regression, the model is $f_{w,b}(x) = wx + b$.

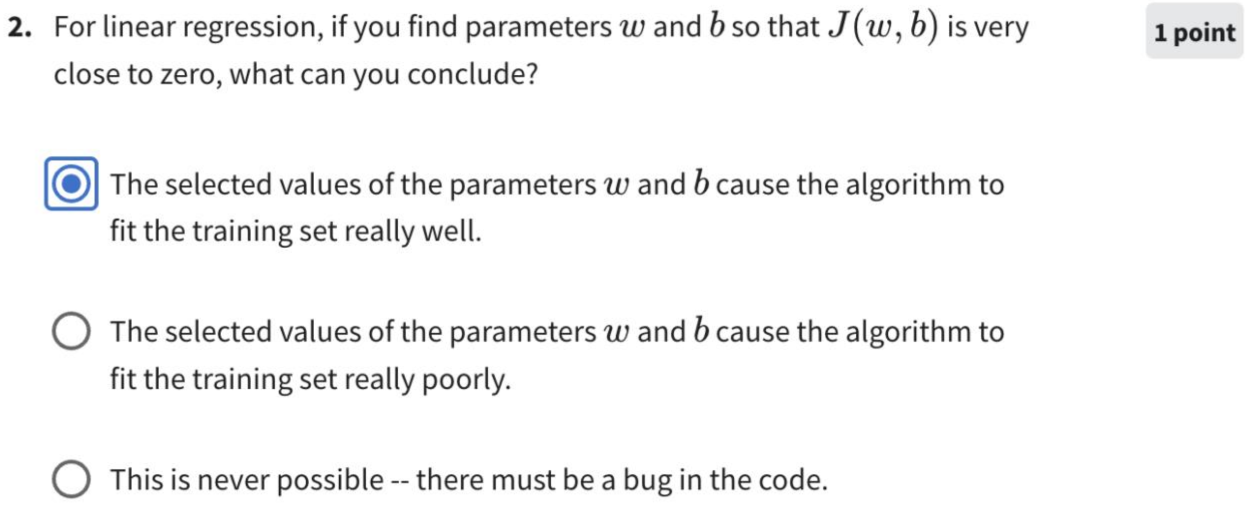
Which of the following are the inputs, or features, that are fed into the model and with which the model is expected to make a prediction?

○ m

○ (x, y)

● x

○ w and b.



2. For linear regression, if you find parameters w and b so that $J(w,b)$ is very close to zero, what can you conclude?

1 point

● The selected values of the parameters w and b cause the algorithm to fit the training set really well.

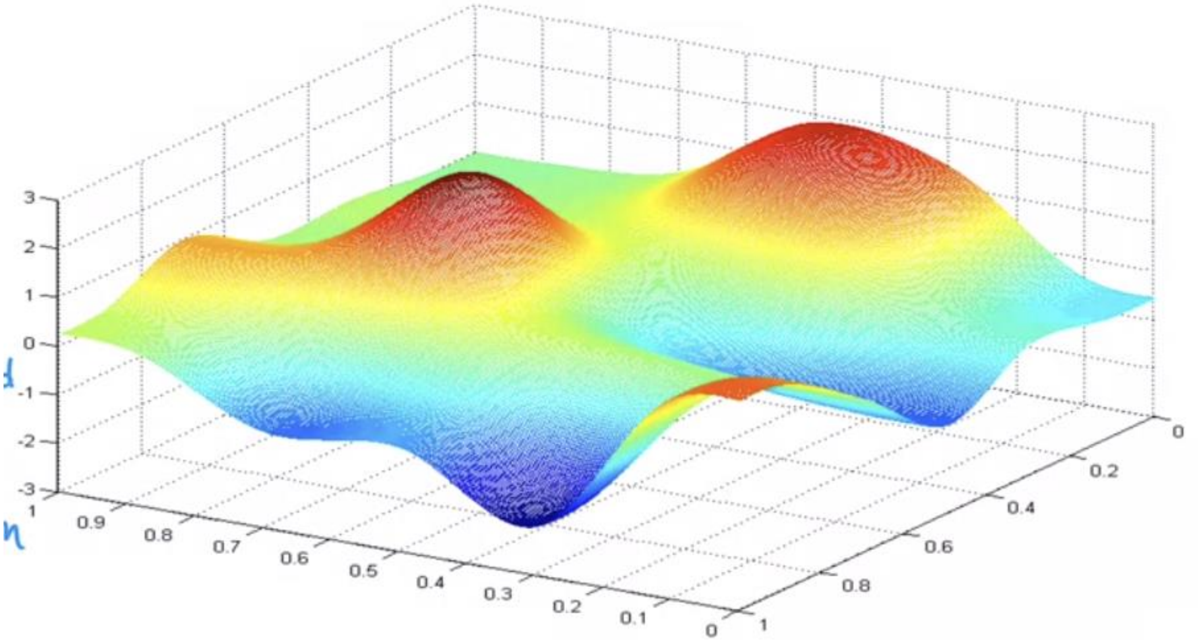
○ The selected values of the parameters w and b cause the algorithm to fit the training set really poorly.

○ This is never possible -- there must be a bug in the code.

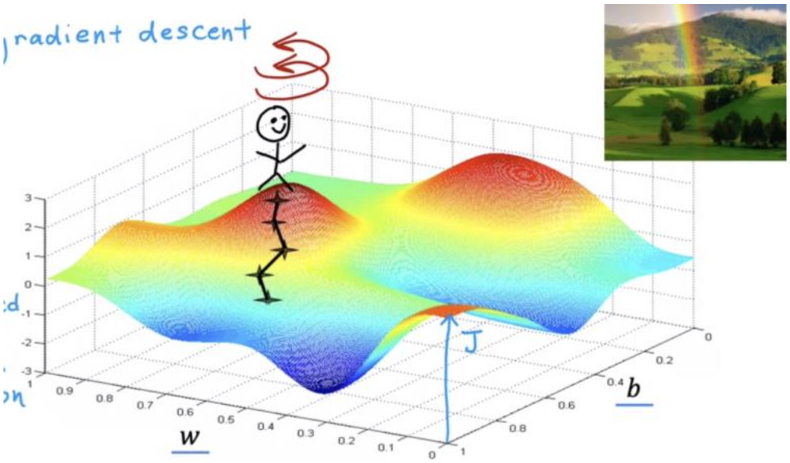
Train the Model with Gradient Descent

- ➔ Gradient Descent
  - ◆ You have a cost function *J(w, b)* (which could be used to describe any function and not just linear regression), and the goal is to find the $min\ J(w, b)$

    - Could also be used for more parameters ($J(w_{1,}, w_2, \ldots, w_n, b)$)
- Outline:
    - Start with some value for *w* and *b*
        - → Keep changing *w, b* to reduce *J(w, b)* until we style at or near a minimum value
            - Please note for non-parabolic cost function graphs, there can be more than one minimum value
- Example:

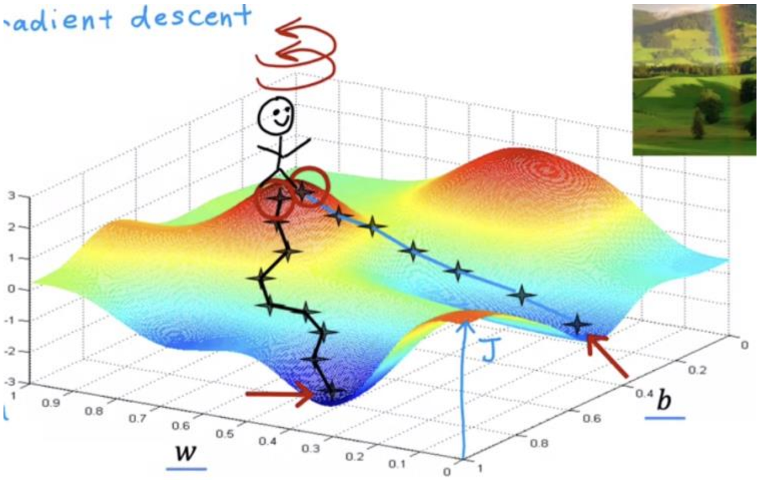


    - Not a squared error cost function and not a linear regression (in linear regression you always end up with a bowl shape)
        - Analogy:



            - Standing on top of a hill and your goal is to get down the hill to the minimum point as fast as possible → you make a 360° turn and evaluate which is the best way and you walk down a little bit → reevaluate your path again → walk down a little bit, etc.

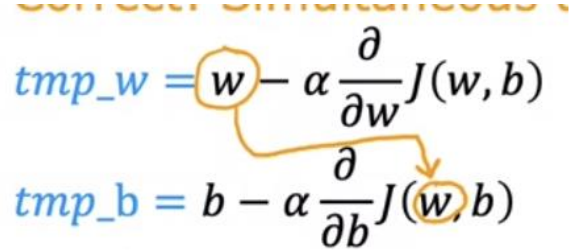
- This is the same thing the gradient descent algorithm is doing until you reach a local minimum
  - Another path can lead to another local minimum if the starting position is changed



➔ Implementing Gradient Descent

- In gradient descent, the value of *w* is updated at every step
  - Formula: $w = w\ (old\ value) - \alpha \times \frac{d}{dw}J(w, b)$
    - = sign in the equation above is the **assignment operator**
      - This is used in code
      - Conversely in truth assertion for math you can say a = c to mean a and c are of equal values, but you cannot say things like a = a + 1 since that does not make sense in mathematical terms
        - Written as “==” in code commonly
    - α is known as the learning rate
      - Usually between 0 to 1
      - Denotes how big of a step you take downhill to try and get to the minimum
        - The bigger the alpha value the bigger the steps you are going to be taking
    - $\frac{d}{dw}J(w, b)$ in simple terms is what direction you want to take your step
      - In combination with α it tells us how big and the direction of the step that we want to take downhill
- The b value is also updated at every step
  - Formula $b = b\ (old\ value) - \alpha \times \frac{d}{db}J(w, b)$
- **→ The goal is to repeat the steps for the new *b* value and new *w* value till convergence is reached**
  - Reach the point at a local point where *w* and *b* don’t change much

    - Both *w* and *b* are updated **simultaneously**
        - Formulas
            - $tmp\ w\ =\ w\ -\alpha\frac{\partial}{\partial w}J(w,b)$
            - $tmp\ b\ =\ b\ -\alpha\frac{\partial}{\partial w}J(w,b)$
            - Note the pre-derivative *w* goes into the function formula



$$tmp\_w = \boxed{w} - \alpha\frac{\partial}{\partial w}J(w,b)$$
$$tmp\_b = b - \alpha\frac{\partial}{\partial b}J(\boxed{w},b)$$

        - → Calculate both the *tmp w* and *tmp b* simultaneously and store the values → copy the value of *tmp w* into w (w = *tmp w*) and the value of *tmp b* into b (b = *tmp b*)

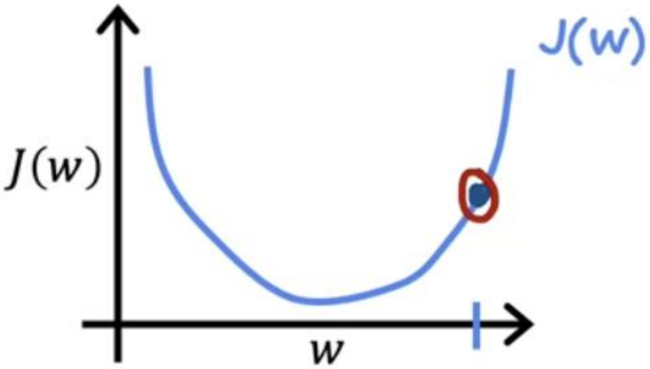
- ➔ Gradient Descent Intuition
    - Goal: repeat until convergence {

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

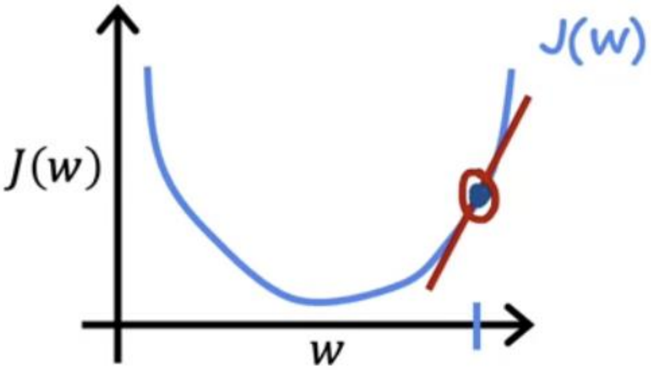
$$b\ =\ b\ -\alpha\frac{\partial}{\partial w}J(w,b)$$

        - $a\ =\ learning\ rate$
        - $\frac{\partial}{\partial w}J(w,b)\ =\ derivative$
            - What is the derivative?
                - You can draw a tangent line at a point on a line and touches the curve
        - Example: *J(w)* cost function
            - Gradient descent formula then would be $w\ =\ w - a\frac{\partial}{\partial w}J(w)$
                - Goal is to adjust w to get $min\ J(w)$
                    - Could look at two-dimensional graphs since b can be set to 0

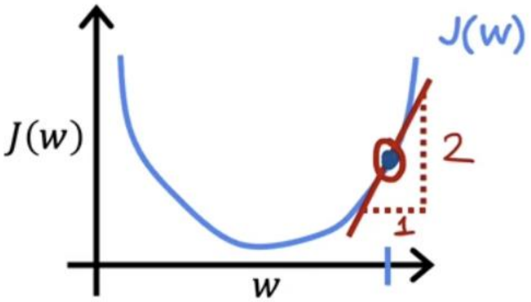
- →Pick an initial point



- → update using $w \ = \ w - a\frac{\partial}{\partial w}J(w)$
  - Derivative meaning ($\frac{\partial}{\partial w}J(w)$)
    - The slope of the line at a specific point in the derivative of the line

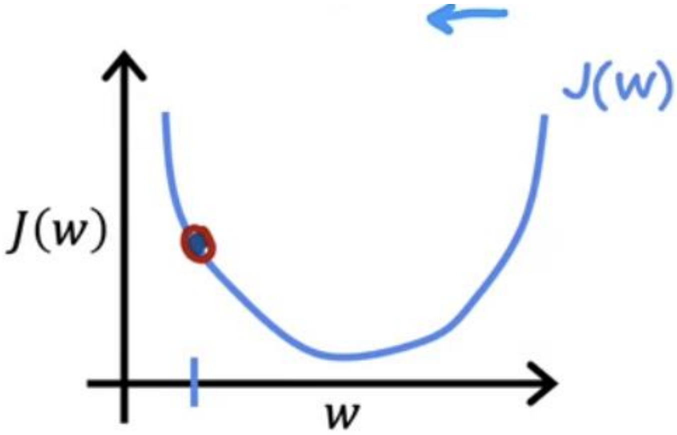


- To compute the slope, we can use a triangle and divide the height by the width (2/1)



- In this case the slope is also positive due to its direction, so the derivative is a positive number → $w \ = \ w \ - \alpha \ \cdot \ (positive \ number)$

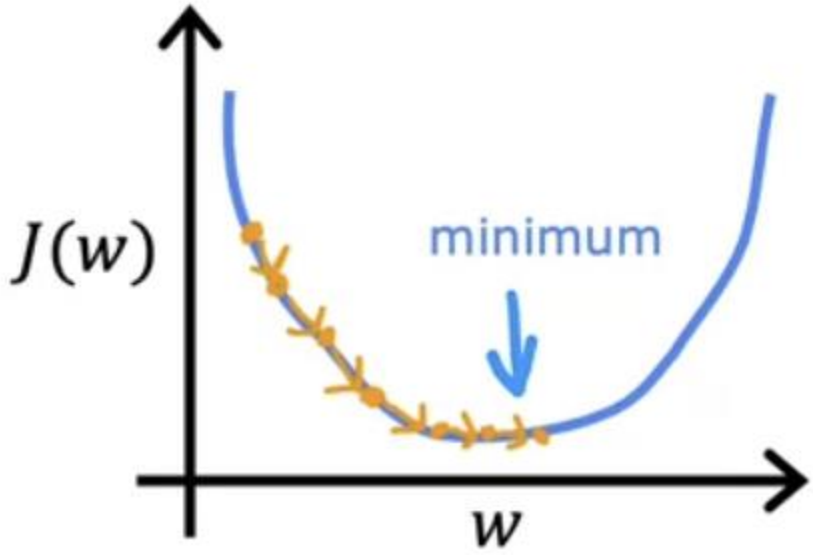
              - Learning rate is always positive → $w \ = \ w \ - \ (+)\,(+) \ = \ w -$ $(+)$ → new w value will always be smaller (moving to the left in the case of our example). This makes sense because moving left moves us closer to a minimum *J(w)* value
  - Another example with the same function (*J(w)* cost function)
    - → Starting with a different initial value for *w*



      - Tangent line is negative in this case because it's sloping down and to the right → derivative function is negative
        - $w \ = \ w \ - \alpha \ \cdot \ (negative\ number)$
          - Learning rate is always positive → $w \ = \ w \ - \ (+)\,(-) \ = \ w - \ (-) \ = \ w \ + \ (+)$
            - New *w* value in this case will be bigger (moving to the right on the graph). This makes sense because moving right moves us closer to a minimum *J(w)* value

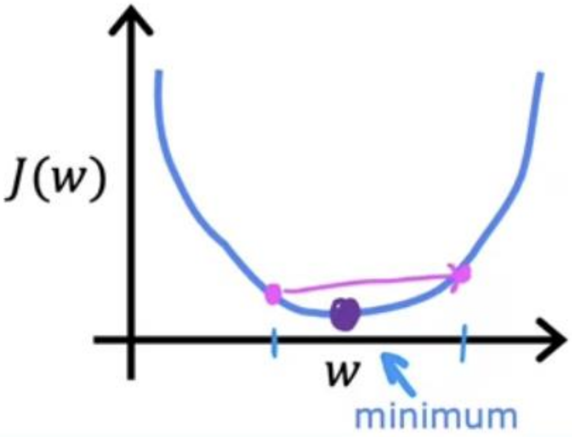
- Learning Rate
  - Going back to the formula: $w \ = \ w - a\frac{\partial}{\partial w}J(w)$
    - α is the learning rate

- If α is too small, you multiply the derivative term by a small number → take a really small step toward the minima



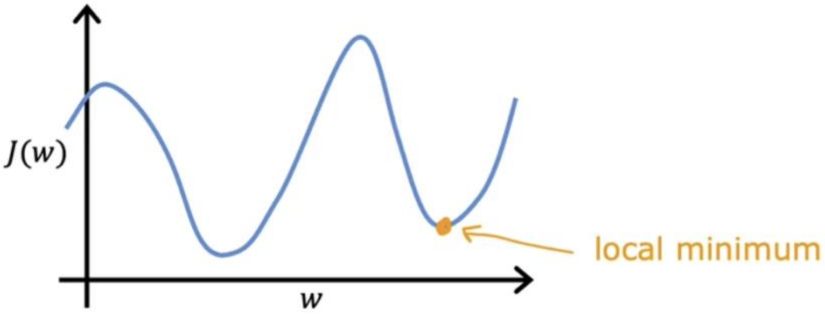
  - You eventually get to the minimum but in a lot of steps and really slowly

- If α is too big, you multiply the derivative term by a big number → could potentially overshoot the new w value corresponding to the minima that you get

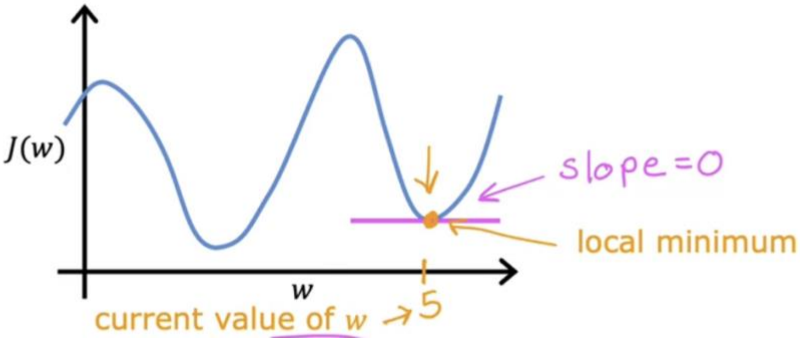


  - Cost has actually increased in this example which is not what you want → could never reach the minimum and fail to converge or even diverge

- Another example: *w* is already at a local minimum

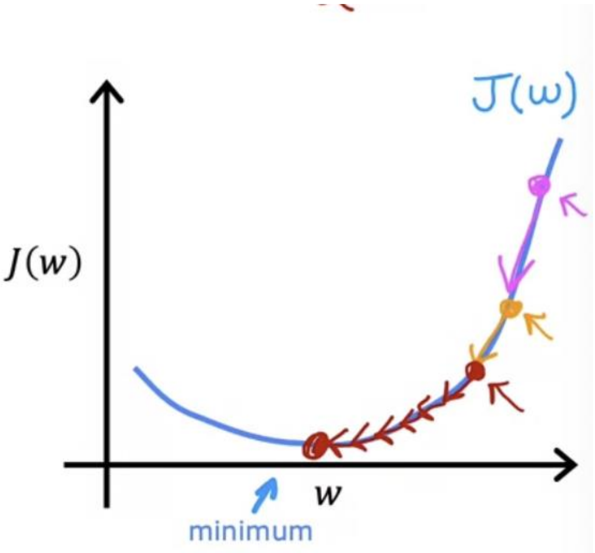


  - Not a square error cost function with two local minima
  - If you draw a tangent line to the local minima, the slope will equal 0 and the derivative function in $w = w - a\frac{\partial}{\partial w}J(w)$ will also equal 0



    - $\rightarrow w = w - a \cdot 0 \rightarrow w = w$
      - If you are at local minima, gradient descent will leave *w* unchanged
  - **If you reach a local minimum, gradient descent will not work**
    - Gradient descent can reach a local minimum with a fixed learning rate

- Example: $w\ =\ w - a\frac{\partial}{\partial w}J(w)$



  - If you start with the first value, there is bigger slope tangent to the point, derivative function will be large
  - → Going to the next value for *w*, the derivative will be smaller, and then smaller as we approach the minimum value, etc.

- ➔ Gradient Descent for Linear Regression
  - ◆ Linear Regression Model
    - Function: $f_{w,b}(x)\ =\ wx\ + b$
    - Cost Function: $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\quad (f_{w,b}(x^{(i)}) - y^{(i)})^2$
  - ◆ Gradient Descent Algorithm
    - repeat until convergence {

      $w\ =\ w\ - \alpha\frac{\partial}{\partial w}J(w,b)$ → $\frac{1}{m}\sum_{i=1}^{m}\blacksquare(f_{w,b}(x^{(i)}) - y^{(i)})x^{(i)}\blacksquare$(derivative of cost function with respect to w)

      $b\ =\ b\ - \alpha\frac{\partial}{\partial w}J(w,b)$ → $\frac{1}{m}\sum_{i=1}^{m}\blacksquare(f_{w,b}(x^{(i)}) - y^{(i)})\blacksquare$(derivative of cost function with respect to b)

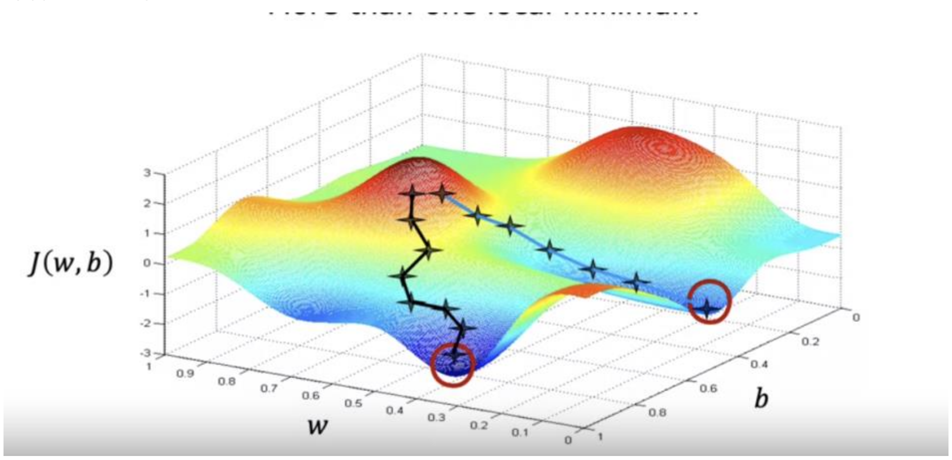
      }
    - Could implement gradient descent with the summation formulas
  - ◆ Gradient Descent Alogirithm for Linear Regression
    - repeat until convergence {

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

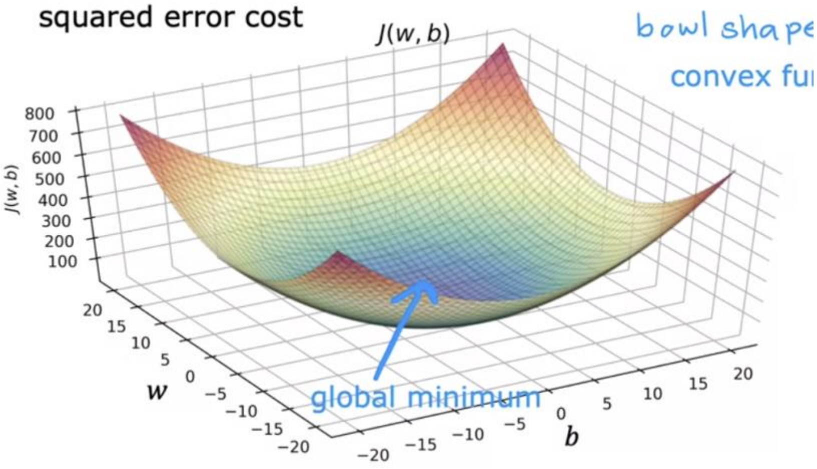
$$b\ =\ b - a\frac{1}{m}\sum_{i=1}^{m}\quad (f_{w,b}(x^{(i)}) - y^{(i)})$$

}

            - Want to update *w* and *b* simultaneously on each step
    - Gradient Descent Problems
        - Could lead to local minimum rather than global minimum
            - Where you initialize the parameters *w* and *b* could lead to different local minima

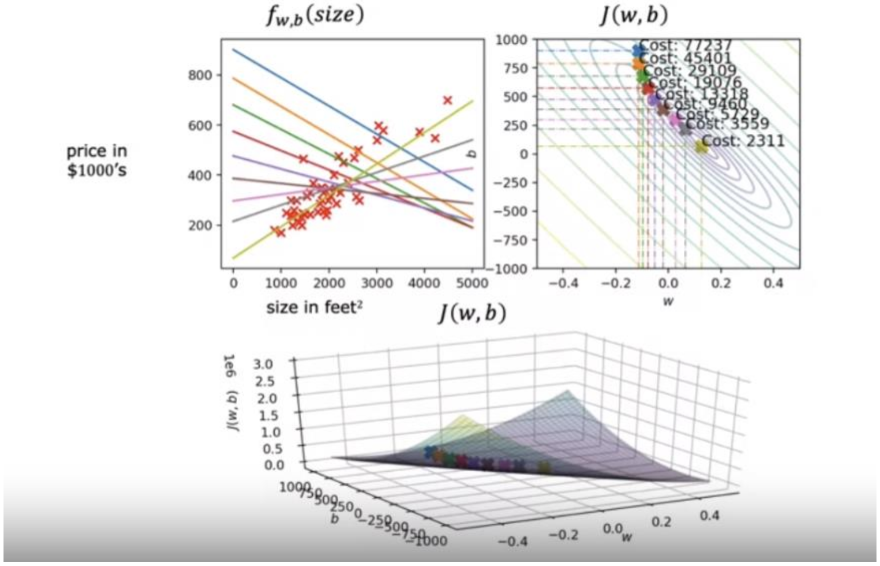


            - When using a squared error cost function with linear regression this is not a problem as there is only one global minima due to it being a convex function



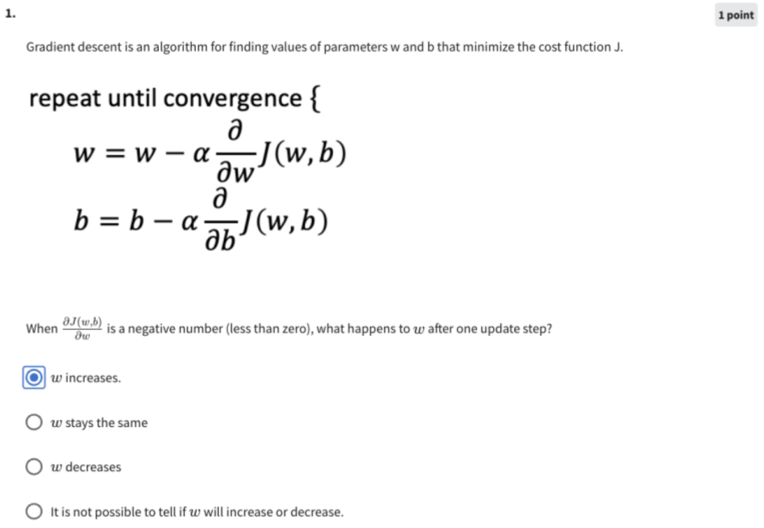
- Running Gradient Descent
    - Example: f(x) = -0.1x + 900 for initial starting for a data set

- Cost function slowly moves to decrease the cost as gradient descent continues



  - This gradient descent process is called **batch gradient descent**
    - This means that on every step of gradient descent we are using all the training examples rather than a subset of the training data

➔ Practice Quiz: Train the Model with Gradient Descent



1.

1 point

Gradient descent is an algorithm for finding values of parameters w and b that minimize the cost function J.

repeat until convergence {

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

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

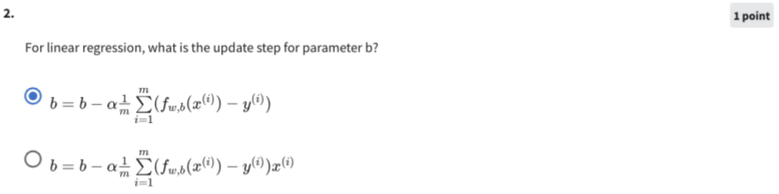
When $\frac{\partial J(w,b)}{\partial w}$ is a negative number (less than zero), what happens to w after one update step?

● w increases.

○ w stays the same

○ w decreases

○ It is not possible to tell if w will increase or decrease.



2.

1 point

For linear regression, what is the update step for parameter b?

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

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