7. Gradient Descent in Practice.

I. feature scaling.

feature and parameter values.

$\hat{price} = w_1 x_1 + w_2 x_2 + b$.

x_1: size $(ft)^2$ — range 300 – 2000 — large.

x_2: bedrooms — range 0 – 5 — small.

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

One training example.

size of parameters w_1, w_2?

If we set $w_1 = 50$, $w_2 = 0.1$, $b = 50$. } not good choices.

$\hat{price} = 50 \times 2000 + 0.1 \times 5 + 50 = 100,050.5$ k. } very far.

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

$\hat{price} = 0.1 \times 2000 + 50 \times 5 + 50 = 500$ k } more reasonable.

When feature value is large, parameter value will be small. If feature value is small, param value might be large.

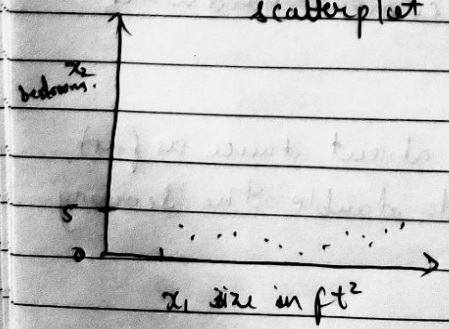
size of featur x_j — size of parameter w_j

size in ft^2

bedrooms.

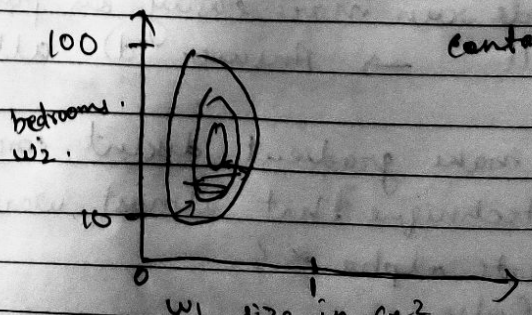
Features.

scatterplot



Parameters.

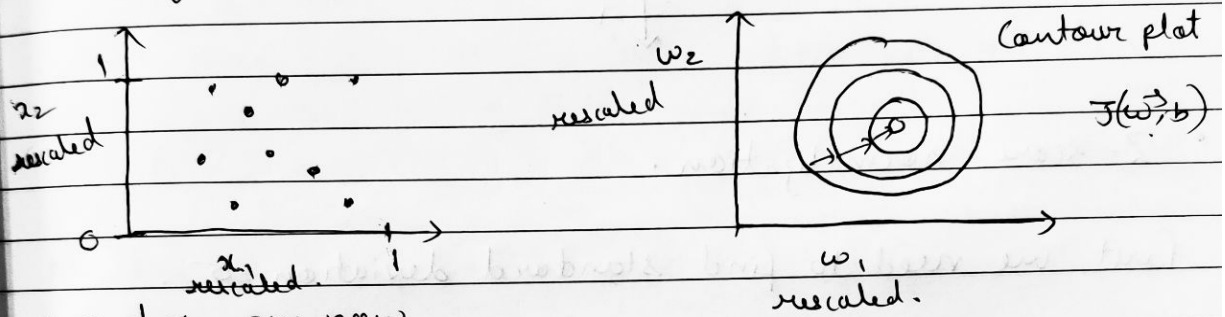
contour plot.



We have shorter on 1 side, longer on another. This is because of value impact.

If we go normal route, our Gradient Decent will go through all circles before reaching global minima.

To rectify this, we should transform our data.



x_1 and x_2 are now take comparable values.

More circular plot. Gradient descent can find a much more direct path to global minimum.

II. How to scale features?

if $300 \leq x_1 \leq 2000$, $0 \leq x_2 \leq 5$

we can divide x_1 by max value, 2000.

x_1 scaled $= \frac{x_1}{2000}$ x_2 scaled $= \frac{x_2}{5}$

Now, $0.15 \leq x_1$ scaled ≤ 1, $0 \leq x_2$ scaled ≤ 1.

- Mean Normalization.

$300 \leq x_1 \leq 2000$, $0 \leq x_2 \leq 5$

they can have both negative and positive values.

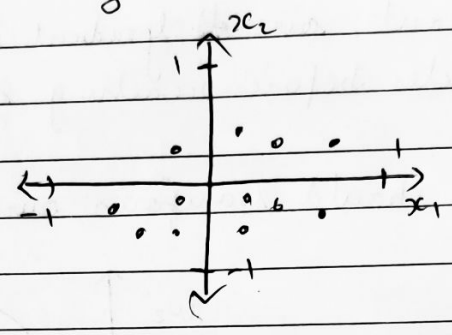
let μ_1 be the mean of x_1.

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

max - min. max min

So, $-0.18 \leq x_1 \leq 0.82$, $-0.46 \leq x_2 \leq 0.54$.

Using mean normalized x_1 and x_2, it will look like this.



- Z-score normalization.

First, we need to find standard deviation σ.

We first find μ_1, and the find σ.

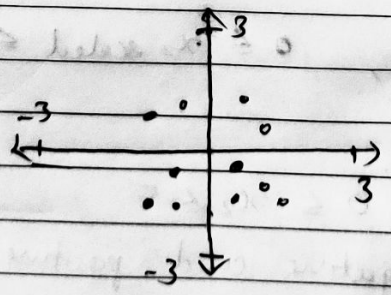
If $\sigma_1 = 450$, $\mu_1 = 600$ $\mu_2 = 2.3$, $\sigma_2 = 1.4$

then $x_1 = \frac{x_1 - \mu_1}{\sigma_1}$ $x_2 = \frac{x_2 - \mu_2}{\sigma_2}$.

$-0.67 \leq x \leq 3.1$

$-1.6 \leq x_2 \leq 1.9$

So, the normalized x_1 and x_2 will look like this:



- feature scaling

aim for about $-1 \leq x_j \leq 1$ for each feature x_j.

$-3 \leq x_j \leq 3$
$-0.3 \leq x_j \leq 0.3$
} acceptable ranges.

also, $0 \leq x_1 \leq 3$ no rescaling

$-2 \leq x_2 \leq 0.5$ no rescaling.

$-100 \leq x_3 \leq 100$ → too large, we need to rescale.

$-0.001 \leq x_4 \leq 0.001$ → too small, rescale.

$98.6 \leq x_5 \leq 105$ → too large, rescale.

No harm in carrying out feature rescaling.

III. Checking gradient Descent for convergence.

Gradient descent:

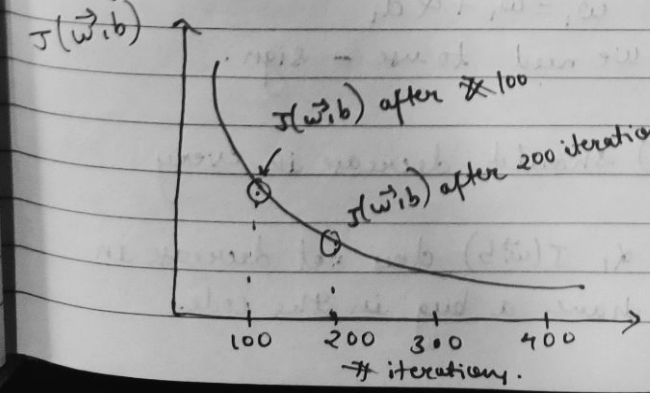
$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

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

a. Make sure gradient descent is working well.

objective: $\min_{\vec{w}, b} J(\vec{w}, b)$, for this, we plot the value of $J(\vec{w}, b)$ w.r.to. no. of iterations we run so far.

our graph ~~soha~~ should look like this:



This graph is also called learning curve.

$J(\vec{w}, b)$ should decrease after every iteration.

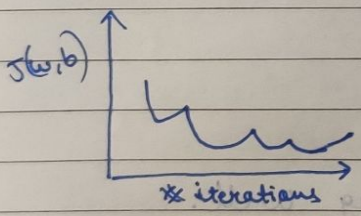
at 300, 400 our $J(w, b)$ is almost constant. Our gradient descent is almost converged.

The iterations may vary for each use case.
We can plot this graph to test this our.

b. Automatic convergence test.
Let ε "epsilon" be 10^{-3}
If $J(\vec{w}, b)$ decreases by $\leq \varepsilon$ in one iteration, declare convergence. (found parameters $\vec{w}, b$ to get close to global minimum).

IV. Choosing a learning rate.

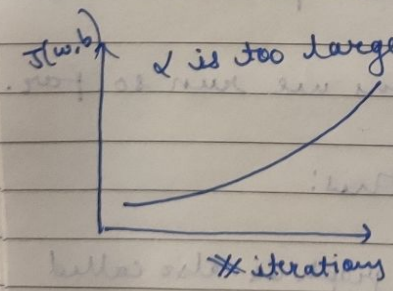
a. Identify problems with gradient



some issue with our code or learning

we need to adjust the learning rate α.

Use a smaller α.



Use smaller α or possible bug in the code.
e.g. $w_1 = w_1 + \alpha d_1$
we need to use $-$ sign.

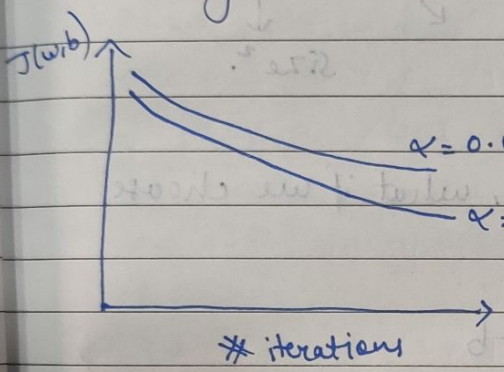
With a small α, $J(\vec{w}, b)$ should decrease in every iteration.
If ~~on~~ even with small α, $J(\vec{w}, b)$ does not decrease in every iteration, then we have a bug in the code.

If α is too small, it will take a lot of time to converge.

Values of α to try:

0.001 0.01 0.1 1.

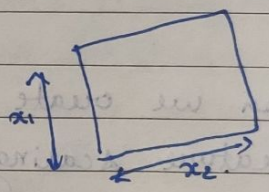
We need to plot cost function for these values, and select the value of α where $J(\vec{w}, b)$ decreases rapidly and consistently.



We need to get the largest decent value of α.

V. Feature Engineering.

Say we have x_1 as frontage, x_2 as depth.



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

Now, area = frontage × depth.

$x_3 = x_1 x_2$. (new feature).

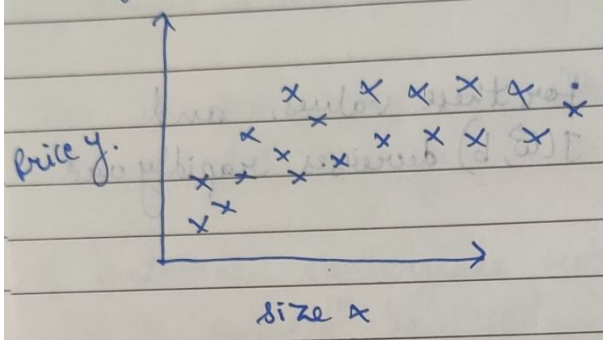
$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b.$$

Here, we created a new feature out of existing features. This is called feature engineering.

Using intuition to design new features, by transforming or combining original features.

VI. Polynomial Regression.

a. Polynomial Regression.



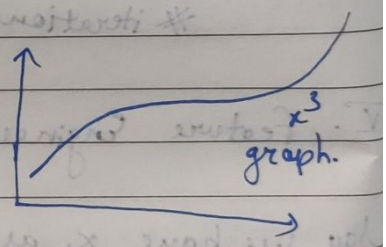
Rather than linear, maybe this function would work.

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

size size²

$$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + b$$

size size² size³.



When we create x^2 or x^3 values of our input features, feature scaling becomes extremely important.

b. Choices of features.

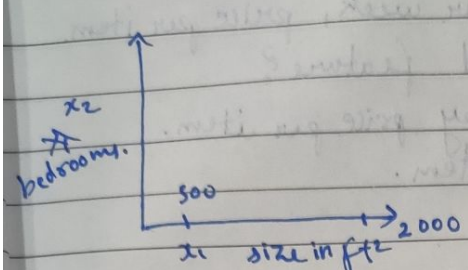
We can also use $\sqrt{x}$, so our model can look like this.

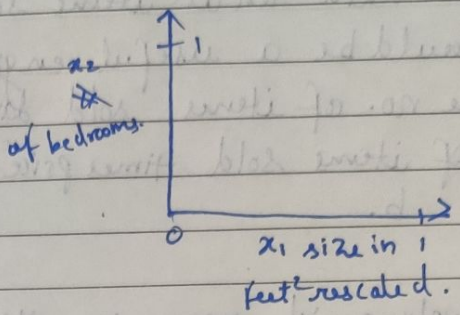
$$f_{\vec{w},b}(x) = w_1 x + w_2 \sqrt{x} + b.$$

size $\sqrt{\text{size}}$

What features to use?

VII. Quiz.





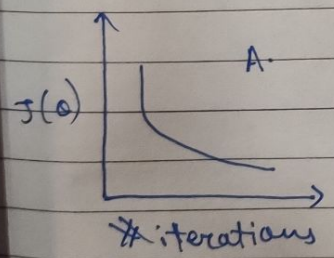
1. Which of the following is a valid step used during feature scaling?

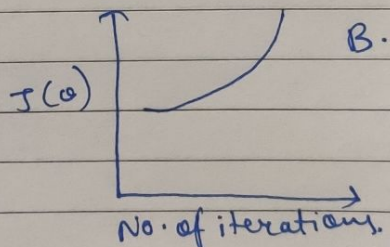
a) Add the mean from each value and then divide by the (max - min)

b) Subtract the mean from each value and then divide by (max - min).

Answer - b.

2. Suppose a friend ran gradient three separate times with three choices of the learning rate α, and plotted the learning curves for each (cost J for each iteration).





For which case, α was too large.

Answer - For B.

3. Of the circumstances below, for which one is feature scaling particularly helpful?

a). When a feature is much larger or smaller than others.

b) When all features in the original data range from 0 to 1.

Answer - a.

4. You are helping a grocery store predict its revenue, and have data on its items sold per week, price per item. What could be a useful engineered feature?

a) The no. of items sold divided by price per item.

b) No. of items sold times price per item.

Answer- b.

5. With polynomial regression, the predicted values $f_{\vec{w},b}(x)$ does not necessarily have to be a straight line (or linear) function of the input feature?

→ True.