Gradient Descent in Practice I - Feature Scaling

Feature Scaling

Idea: Make sure features are on similar scale.

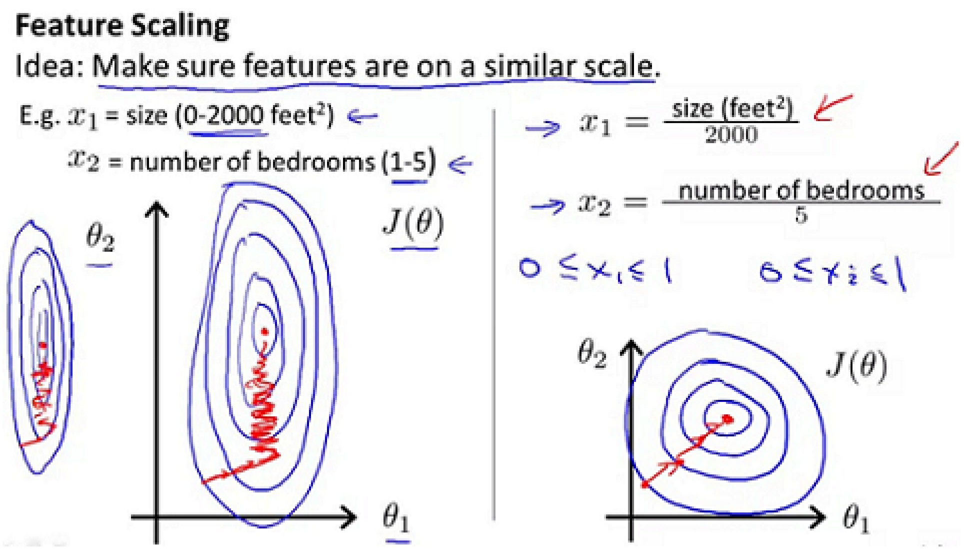
If we make sure that the features are on a similar scale (different features take on similar ranges of values), then gradient descents can converge more quickly. Concretely let's say we have a problem with two features where x_1 is the size of house and takes on values between 0 to 2000 and x_2 is the number of bedrooms, that takes on values between 0 and 5.

E.g. x_1 = size (0 - 2000 $feet^2$)

x_2 = number of bedrooms (1 - 5)

If we plot the contours of the cost function $J(\theta)$ (J of theta is a function of parameters θ_0, θ_1 and θ_2, but for this example we're going to ignore θ_0), then the contours may look like a very skewed elliptical shape, and with the so 2000 to 5 ratio, it can be even more skewed.

So, the contours may look like a very tall and skinny ellipses, that can form the contours of the cost function $J(\theta)$. And if we run gradient descents on the cost function, our gradients may end up taking a long time and can oscillate back and forth and take a long time before it can finally find its way to the global minimum.



Feature Scaling
Idea: Make sure features are on a similar scale.
E.g. x_1 = size (0-2000 feet²)
x_2 = number of bedrooms (1-5)
θ_2
$J(\theta)$
θ_1
$x_1 = \frac{\text{size (feet}^2)}{2000}$
$x_2 = \frac{\text{number of bedrooms}}{5}$
$0 \le x_1 \le 1$
$0 \le x_2 \le 1$
θ_2
$J(\theta)$
θ_1

In these settings, a useful thing to do is to scale the features. Concretely if we instead define the feature x_1 to be the size of the house divided by 2000, and define x_2 to be the number of bedrooms divided by five, then the count well as of the cost function J can become much less skewed so the contours may look more like circles, and we can wind up with an implementation of gradient descent.

Feature Scaling

Get every feature into approximately a $-1 \leq x_i \leq 1$ range

More generally, when we're performing feature scaling, what we often want to do is get every feature into approximately a -1 to $+1$ range and concretely, our feature x_0 is always equal to 1. So, that's already in that range.

Mean Normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g. $x_1 = \frac{size - 1000}{2000}$

$x_2 = \frac{\#bedrooms - 2}{5}$

In both of these cases, we therefore wind up with features x_1 and x_2. They can take on values roughly between:

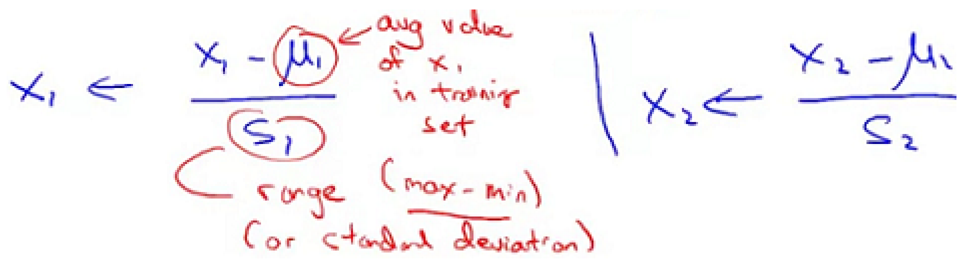
$$-0.5 \le x_1 \le 0.5, -0.5 \le x_2 \le 0.5$$

Video Question: Suppose you are using a learning algorithm to estimate the price of houses in a city. You want one of your features to capture the age of the house. In your training set, all of your houses have an age between 30 and 50 years, with an average age of 38 years. Which of the following would you use as features, assuming you use feature scaling and mean normalization?

- $x_i = ageofhouse$
- $x_i = \frac{ageofhouse}{50}$
- $x_i = \frac{ageofhouse - 38}{50}$

> $x_i = \frac{ageofhouse - 38}{20}$

In general:



$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1}$
avg value of x_1 in training set
range (max - min)
(or standard deviation)
$x_2 \leftarrow \frac{x_2 - \mu_1}{s_2}$

Summary

We can speed up gradient descent by having each of our input values in roughly the same range. This is because θ will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to the optimum when the variables are very uneven.

The way to prevent this is to modify the ranges of our input variables so that they are all roughly the same. Ideally:

$$-1 \le x_{(i)} \le 1$$

or

$$-0.5 \le x_{(i)} \le 0.5$$

These aren't exact requirements; we are only trying to speed things up. The goal is to get all input variables into roughly one of these ranges, give or take a few.

Two techniques to help with this are **feature scaling** and **mean normalization**. Feature scaling involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1. Mean normalization involves subtracting the average value for an input variable from the values for that input variable resulting in a new average value for the input variable of just zero. To implement both of these techniques, adjust your input values as shown in this formula:

$x_i := \frac{x_i - \mu_i}{s_i}$

Where μ_i is the average of all the values for feature (i) and s_i is the range of values (max - min), or s_i is the standard deviation. Note that dividing by the range, or dividing by the standard deviation, give different results.

For example, if x_i represents housing prices with a range of 100 to 2000 and a mean value of 1000, then, $x_i := \frac{price-1000}{1900}$