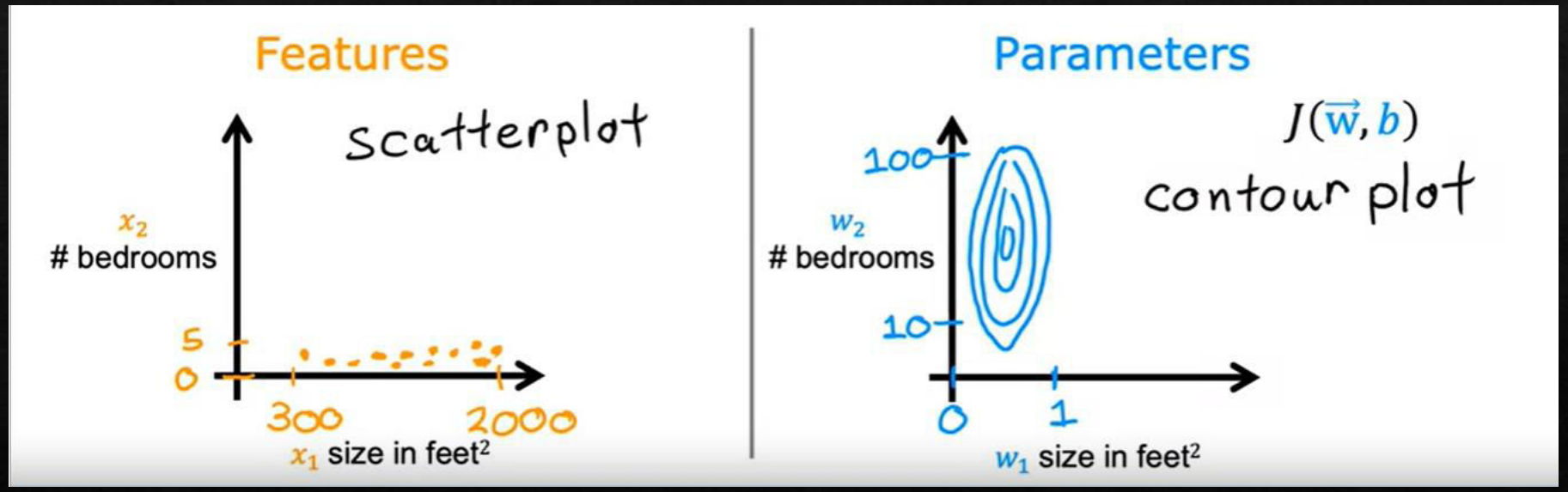
Data Normalization and Standardization

A023119820032

Sunidhi.singh1@s.amity.edu

Feature Scaling is a data preprocessing technique .

Normalization helps to scale down your feature between 0 and 1 , standardization will help you to scale down your feature based on standard normal distribution, which is where mean is 0 and standard deviation is 1 . This is the basic difference between standardization and normalization.



Data Normalisation techniques

Normalization (min-max Normalization)

In this approach we will scale down the values of the features between 0 to 1.

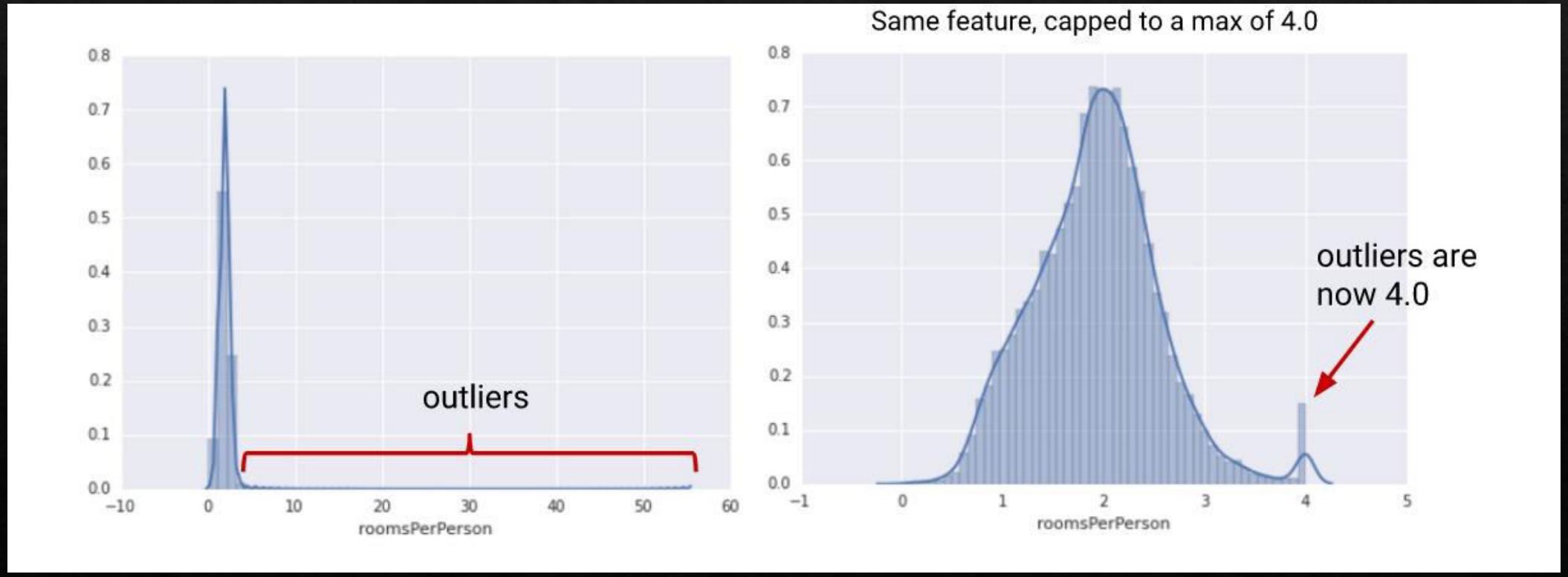
$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Scaling to a range is a good choice when both of the following conditions are met:
•You know the approximate upper and lower bounds on your data with few or no outliers.
•Your data is approximately uniformly distributed across that range.

A good example is age. Most age values falls between 0 and 90, and every part of the range has a substantial number of people.
In contrast, you would *not* use scaling on income, because only a few people have very high incomes. The upper bound of the linear scale for income would be very high, and most people would be squeezed into a small part of the scale.

Feature Clipping

If your data set contains extreme outliers, you might try feature clipping, which caps all feature values above (or below) a certain value to fixed value. For example, you could clip all temperature values above 40 to be exactly 40. You may apply feature clipping before or after other normalizations.



Log Scaling

Log scaling computes the log of your values to compress a wide range to a narrow range. Log scaling is helpful when a handful of your values have many points, while most other values have few points. This data distribution is known as the power law distribution. Movie ratings are a good example. In the chart below, most movies have very few ratings (the data in the tail), while a few have lots of ratings (the data in the head). Log scaling changes the distribution, helping to improve linear model performance.

Z-score normalization or standardisation is a variation of scaling that represents the number of standard deviations away from the mean. You would use *z*-score to ensure your feature distributions have mean = 0 and std = 1. It's useful when there are a few outliers, but not so extreme that you need clipping.

z-score normalization

After z-score normalization, all features will have a mean of 0 and a standard deviation of 1.

To implement z-score normalization, adjust your input values as shown in this formula:

$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

where j selects a feature or a column in the $\mathbf{X}$ matrix. μ_j is the mean of all the values for feature (j) and σ_j is the standard deviation of feature (j).

$$\mu_j = \frac{1}{m} \sum_{i=0}^{m-1} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=0}^{m-1} (x_j^{(i)} - \mu_j)^2$$

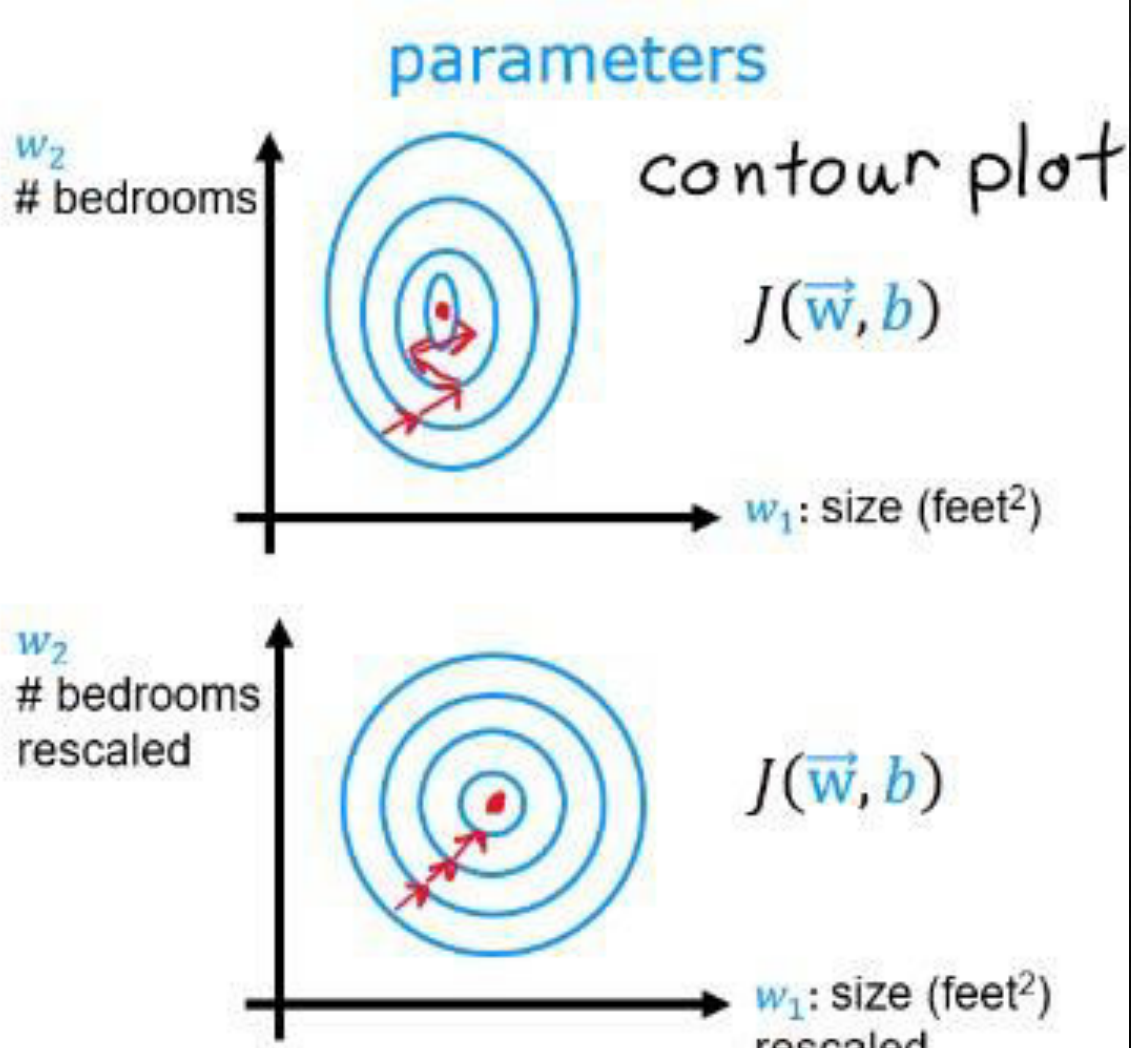
Max-Min Normalisation typically allows us to transform the data with varying scales so that no specific dimension will dominate the statistics, and it does not require making a very strong assumption about the distribution of the data

However, Normalisation does not treat outliers very well. On the contrary, standardisation allows users to better handle the outliers and facilitate convergence for some computational algorithms like gradient descent.

Another way to view feature scaling is in terms of the cost contours. When feature scales do not match, the plot of cost versus parameters in a contour plot is asymmetric.

The plot is so asymmetric, the curves completing the contours are not visible. In contrast, when the features are normalized, the cost contour is much more symmetric. The result is that updates to parameters during gradient descent can make equal progress for each parameter.

Cost Contours



Another importance of Feature Scaling is that gradient descent runs smoothly with scaled features rather than with unscaled values which also depends on whether the need to scaling arises or not .

In [4]:
```
#set alpha to 9.9e-7
_, _, hist = run_gradient_descent(X_train, y_train, 100, alpha = 9.9e-7)
```

```
Iteration Cost          w0       w1       w2       w3       b       djdw0    djdw1    djdw2    djdw3    djdb
---------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
        0 9.55884e+04  5.5e-01  1.0e-03  5.1e-04  1.2e-02  3.6e-04 -5.5e+05 -1.0e+03 -5.2e+02 -1.2e+04 -3.6e+02
       10 1.85075e+06  1.6e+00  2.8e-03  1.4e-03  3.0e-02  1.1e-03 -2.4e+06 -4.5e+03 -2.3e+03 -5.7e+04 -1.6e+03
       20 3.63756e+07  6.0e+00  1.1e-02  5.6e-03  1.3e-01  4.0e-03 -1.1e+07 -2.0e+04 -1.0e+04 -2.6e+05 -7.1e+03
       30 7.15493e+08  2.6e+01  4.7e-02  2.4e-02  5.9e-01  1.7e-02 -4.8e+07 -8.8e+04 -4.6e+04 -1.1e+06 -3.1e+04
       40 1.40740e+10  1.1e+02  2.1e-01  1.1e-01  2.7e+00  7.4e-02 -2.1e+08 -3.9e+05 -2.0e+05 -5.1e+06 -1.4e+05
       50 2.76841e+11  5.0e+02  9.2e-01  4.8e-01  1.2e+01  3.3e-01 -9.5e+08 -1.7e+06 -9.0e+05 -2.2e+07 -6.2e+05
       60 5.44556e+12  2.2e+03  4.1e+00  2.1e+00  5.3e+01  1.5e+00 -4.2e+09 -7.7e+06 -4.0e+06 -1.0e+08 -2.7e+06
       70 1.07116e+14  9.9e+03  1.8e+01  9.5e+00  2.3e+02  6.4e+00 -1.9e+10 -3.4e+07 -1.8e+07 -4.4e+08 -1.2e+07
       80 2.10702e+15  4.4e+04  8.1e+01  4.2e+01  1.0e+03  2.9e+01 -8.3e+10 -1.5e+08 -7.9e+07 -2.0e+09 -5.4e+07
       90 4.14458e+16  1.9e+05  3.6e+02  1.9e+02  4.6e+03  1.3e+02 -3.7e+11 -6.7e+08 -3.5e+08 -8.7e+09 -2.4e+08
w,b found by gradient descent: w: [-7.45e+05 -1.37e+03 -7.11e+02 -1.76e+04], b: -484.18
```

In [8]:
```
#set alpha to 1e-7
_,_,hist = run_gradient_descent(X_train, y_train, 100, alpha = 1e-7)
```

```
Iteration Cost          w0       w1       w2       w3       b       djdw0    djdw1    djdw2    djdw3    djdb
---------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
        0 4.42313e+04  5.5e-02  1.0e-04  5.2e-05  1.2e-03  3.6e-05 -5.5e+05 -1.0e+03 -5.2e+02 -1.2e+04 -3.6e+02
       10 1.89006e+03  2.4e-01  4.3e-04  2.2e-04  4.9e-03  1.6e-04 -4.7e+04 -8.0e+01 -3.7e+01 -4.4e+02 -3.5e+01
       20 1.58187e+03  2.5e-01  4.5e-04  2.2e-04  4.5e-03  1.7e-04 -4.0e+03 -7.9e-01  4.5e+00  5.8e+02 -6.9e+00
       30 1.57918e+03  2.5e-01  4.4e-04  2.2e-04  3.9e-03  1.8e-04 -3.6e+02  5.9e+00  8.0e+00  6.6e+02 -4.6e+00
       40 1.57872e+03  2.5e-01  4.4e-04  2.1e-04  3.2e-03  1.8e-04 -4.5e+01  6.5e+00  8.3e+00  6.7e+02 -4.4e+00
       50 1.57827e+03  2.5e-01  4.3e-04  2.0e-04  2.6e-03  1.9e-04 -1.8e+01  6.5e+00  8.3e+00  6.7e+02 -4.3e+00
       60 1.57782e+03  2.5e-01  4.2e-04  1.9e-04  1.9e-03  1.9e-04 -1.6e+01  6.5e+00  8.3e+00  6.7e+02 -4.3e+00
       70 1.57738e+03  2.5e-01  4.2e-04  1.8e-04  1.2e-03  2.0e-04 -1.6e+01  6.5e+00  8.3e+00  6.7e+02 -4.4e+00
       80 1.57693e+03  2.5e-01  4.1e-04  1.7e-04  5.7e-04  2.0e-04 -1.6e+01  6.5e+00  8.3e+00  6.7e+02 -4.4e+00
       90 1.57648e+03  2.5e-01  4.0e-04  1.7e-04 -1.0e-04  2.0e-04 -1.6e+01  6.5e+00  8.3e+00  6.7e+02 -4.4e+00
w,b found by gradient descent: w: [ 2.53e-01  3.98e-04  1.58e-04 -7.01e-04], b: 0.00
```

```
In [14]: w_norm, b_norm, hist = run_gradient_descent(X_norm, y_train, 100, 1.0e-1, )
```

Iteration	Cost	w0	w1	w2	w3	b	djdw0	djdw1	djdw2	djdw3	djdb
0	5.76170e+04	8.9e+00	3.0e+00	3.3e+00	-6.0e+00	3.6e+01	-8.9e+01	-3.0e+01	-3.3e+01	6.0e+01	-3.6e+02
10	7.69462e+03	5.2e+01	4.8e+00	3.7e+00	-3.4e+01	2.5e+02	-2.5e+01	6.9e+00	1.0e+01	1.3e+01	-1.3e+02
20	1.46259e+03	7.0e+01	-2.4e+00	-6.8e+00	-4.0e+01	3.2e+02	-1.4e+01	6.6e+00	9.6e+00	3.0e+00	-4.4e+01
30	5.39273e+02	8.2e+01	-7.9e+00	-1.5e+01	-4.1e+01	3.5e+02	-9.9e+00	4.6e+00	6.7e+00	2.2e-01	-1.5e+01
40	3.43315e+02	9.0e+01	-1.2e+01	-2.0e+01	-4.1e+01	3.6e+02	-7.0e+00	3.2e+00	4.6e+00	-5.2e-01	-5.4e+00
50	2.77339e+02	9.6e+01	-1.4e+01	-2.4e+01	-4.0e+01	3.6e+02	-5.0e+00	2.3e+00	3.2e+00	-6.3e-01	-1.9e+00
60	2.48040e+02	1.0e+02	-1.6e+01	-2.7e+01	-4.0e+01	3.6e+02	-3.6e+00	1.7e+00	2.2e+00	-5.5e-01	-6.5e-01
70	2.33724e+02	1.0e+02	-1.8e+01	-2.8e+01	-3.9e+01	3.6e+02	-2.5e+00	1.2e+00	1.5e+00	-4.3e-01	-2.3e-01
80	2.26544e+02	1.1e+02	-1.9e+01	-3.0e+01	-3.9e+01	3.6e+02	-1.8e+00	8.7e-01	1.1e+00	-3.2e-01	-7.9e-02
90	2.22919e+02	1.1e+02	-1.9e+01	-3.1e+01	-3.9e+01	3.6e+02	-1.3e+00	6.2e-01	7.5e-01	-2.4e-01	-2.8e-02

```
w,b found by gradient descent: w: [107.83 -19.91 -31.15 -38.49], b: 363.15
```

The gradient of each parameter is tiny by the end of this fairly short run. A learning rate of 0.1 is a good start for regression with normalized features.