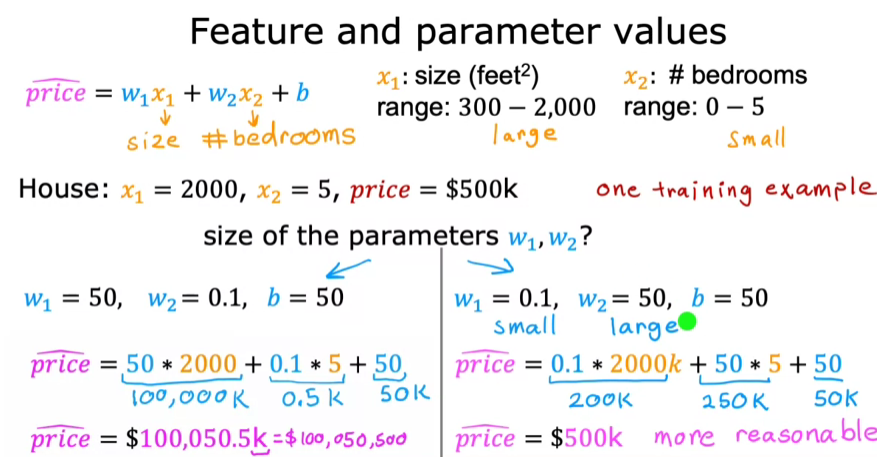
**GRADIENT DESCENT IN PRACTICE**

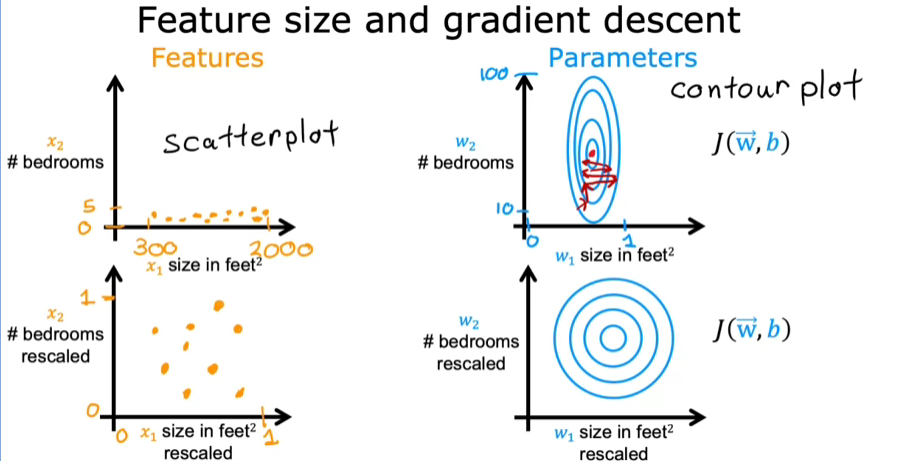
**FEATURE SCALING:**

* **Feature scaling is essential when features have different ranges, as it can significantly impact the performance of gradient descent. For example, the size of a house (in square feet) and the number of bedrooms can vary greatly in their numerical ranges.**
* **When features are not scaled, the parameters associated with larger features may dominate the learning process, leading to inefficient convergence.**

****

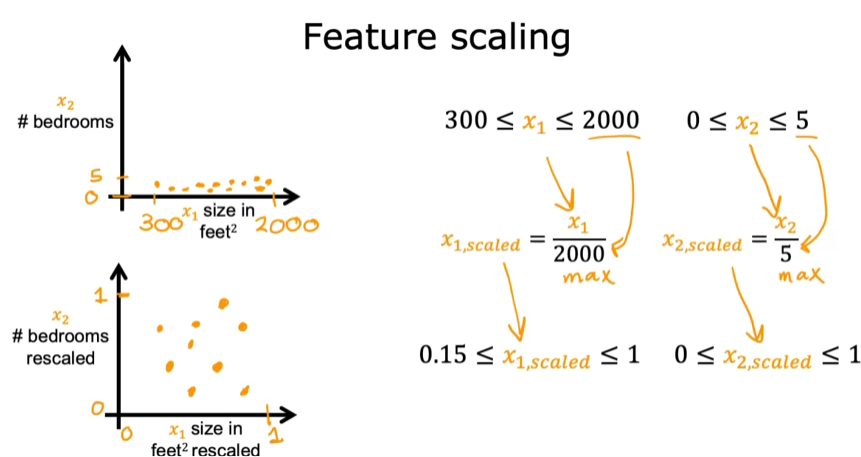
**Feature Size and Gradient Size**

* **If features are on different scales, the cost function may appear elongated, causing gradient descent to take a longer, more convoluted path to reach the global minimum.**
* **By scaling features to a similar range (e.g., 0 to 1), the contours of the cost function become more circular, allowing gradient descent to converge more quickly and efficiently.**

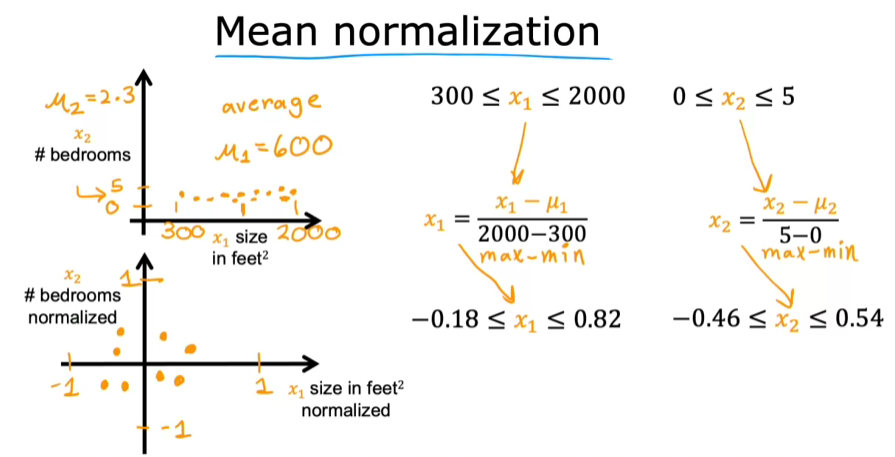
****

**Types of Feature Scaling**

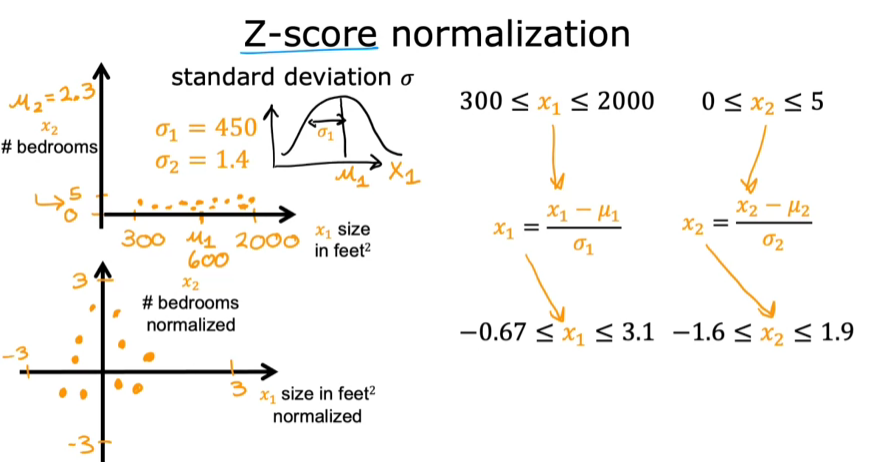
* ***Min-Max Scaling:* This method involves dividing each feature value by the maximum value of that feature, resulting in a scaled range from 0 to 1. For example, if a feature ranges from 3 to 2,000, dividing by 2,000 will scale it appropriately.**

****

* ***Mean Normalization:* This technique centers the data around zero by subtracting the mean of the feature and dividing by the range (max - min). This results in values that can be both negative and positive, typically between -1 and 1.**

****

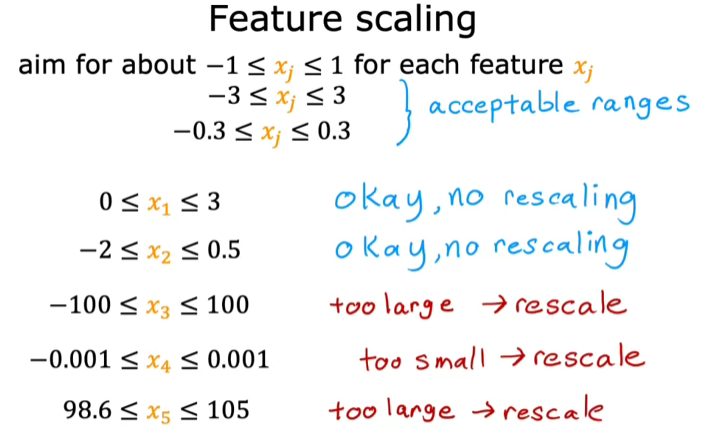
***Z-Score Normalization:* This method standardizes features by calculating the mean and standard deviation. Each feature value is adjusted by subtracting the mean and dividing by the standard deviation, allowing for a distribution centered around zero.**

****

* **The resulting values can vary widely, but they often fall within a range that reflects how far each value is from the mean in terms of standard deviations.**

**Re-scaling Considerations:**

* **It's generally beneficial to scale features, especially when they have significantly different ranges, as this can improve the performance of algorithms like gradient descent.**
* **While aiming for a range of -1 to 1 is a good rule of thumb, it's acceptable to have features that fall outside this range, depending on the context and the specific data characteristics.**

****

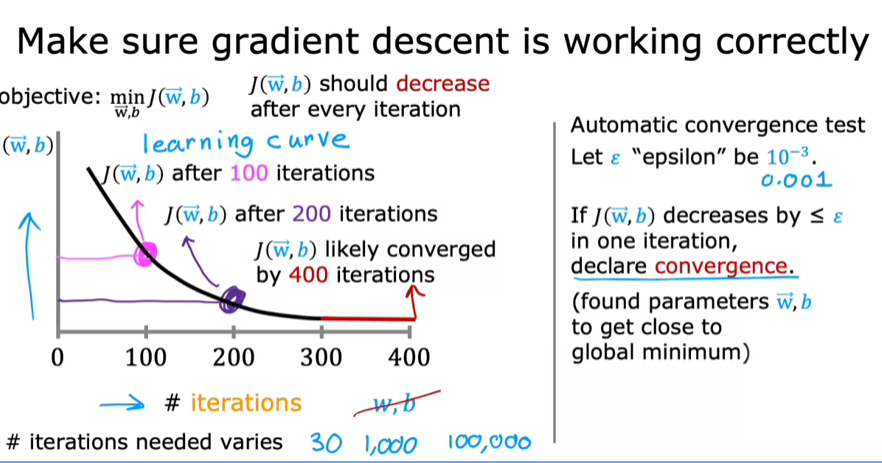
**CHECKING GRADIENT DESCENT FOR CONVERGENCE**

**Recognizing Convergence in Gradient Descent**

* **To assess convergence, plot the cost function J against the number of iterations of gradient descent. A well-functioning gradient descent should show a decreasing cost J with each iteration.**
* **If the cost J increases at any point, it may indicate a poorly chosen learning rate (Alpha) or a bug in the code.**

**Learning Curves and Iterations**

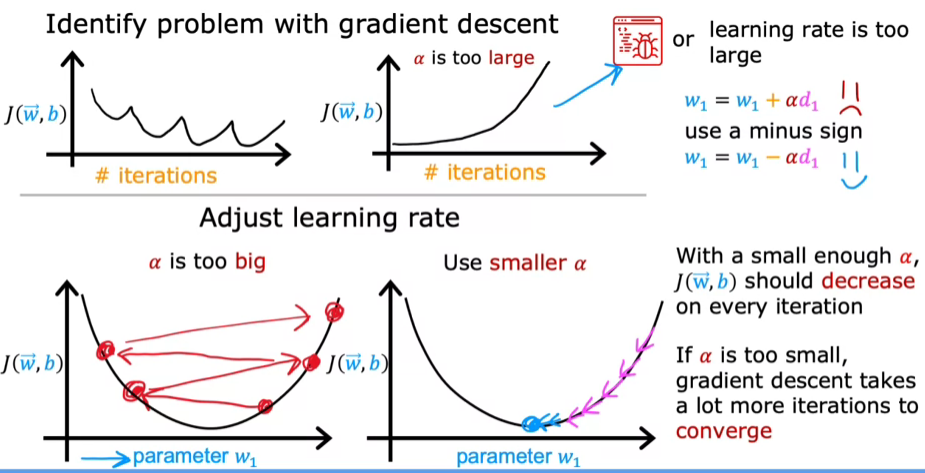
* **The learning curve illustrates how the cost J changes over iterations, helping to visualize when gradient descent is converging.**
* **The number of iterations required for convergence can vary significantly between different applications, making it essential to monitor the learning curve.**

****

**Automatic Convergence Tests**

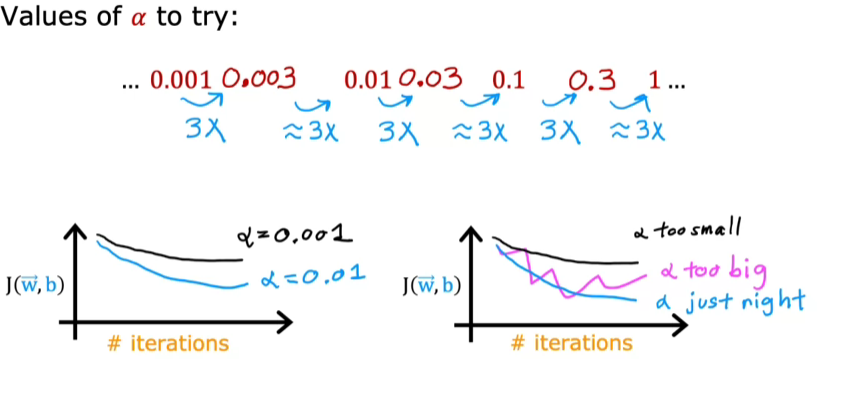
* **An automatic convergence test can be implemented using a small threshold (epsilon). If the cost J decreases by less than epsilon, it suggests that gradient descent has likely converged.**
* **Choosing the right threshold for epsilon can be challenging, so visualizing the learning curve is often more effective for monitoring convergence.**

**CHOOSING THE LEARNING RATE**

****

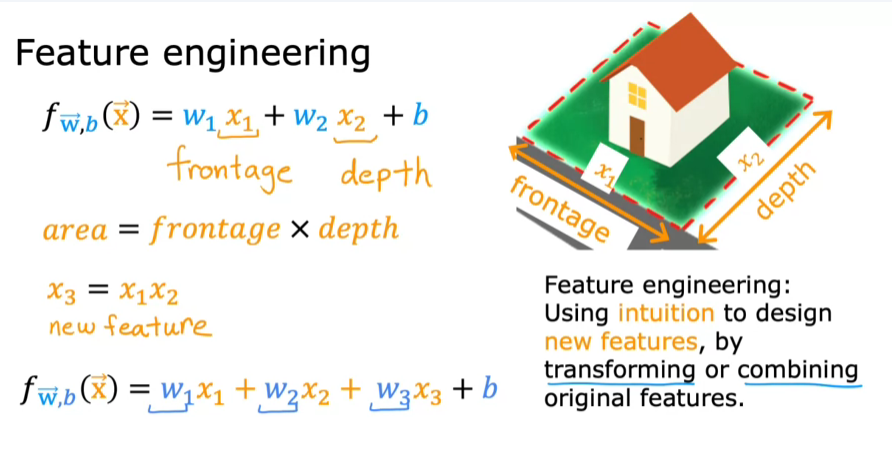
**Choosing Learning Rates:**

* + **Set a small learning rate to check if the cost decreases consistently. If it doesn’t, there may be a bug in the implementation.**
  + **Experiment with a range of learning rates (e.g., 0.001, 0.01, 0.1) to find the optimal value that decreases the cost effectively.**
  + **Gradually increase the learning rate to find the largest value that still allows for convergence.**

****

**FEATURE ENGINEERING**

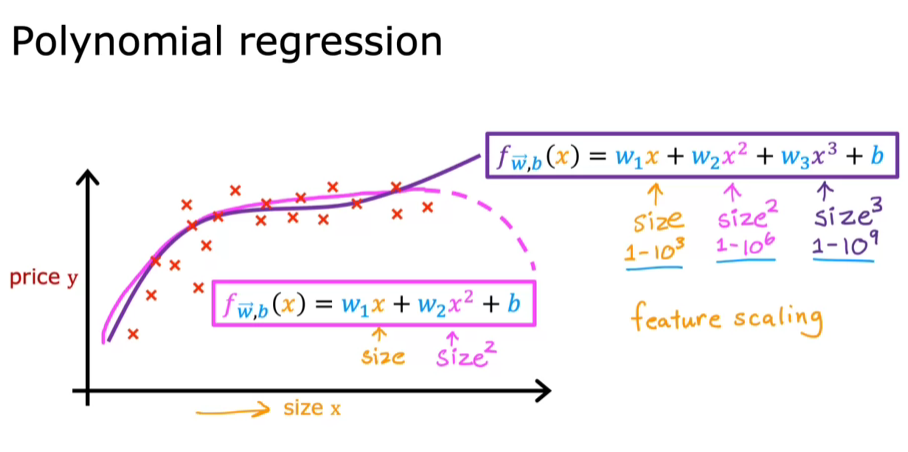
* **The choice of features significantly impacts the performance of a learning algorithm, making it crucial to select the right ones.**
* **Using original features alone may not yield the best results; combining or transforming them can lead to improved models.**

****

* **In predicting house prices, two initial features are the width (x1) and depth (x2) of the lot.**
* **By creating a new feature (x3) that represents the area of the lot (x1 times x2), the model can potentially make more accurate predictions.**

**POLYNOMIAL REGRESSION**

* **Polynomial regression extends linear regression by incorporating polynomial terms, such as size squared (x²) and size cubed (x³), to better-fit curves in data, like housing prices based on size.**
* **Different polynomial degrees can be used; for instance, a quadratic function may not be suitable if it suggests prices decrease with size, while a cubic function can provide a more realistic upward trend.**

****

* **Feature engineering involves creating new features from existing ones, such as using the square root of size, which can also improve model performance.**
* **When using polynomial features, feature scaling becomes crucial, as the ranges of values can differ significantly (e.g., x ranges from 1-1000, x² from 1-1,000,000).**