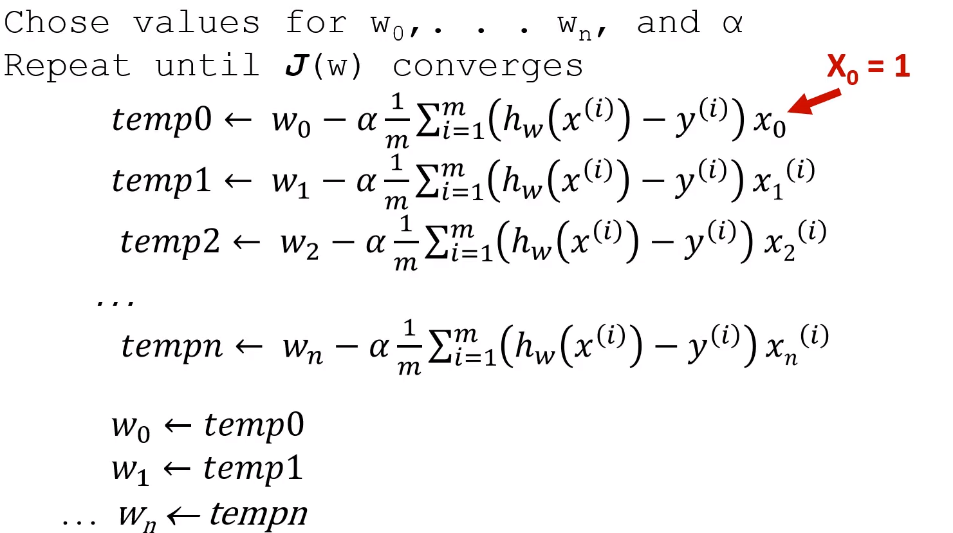
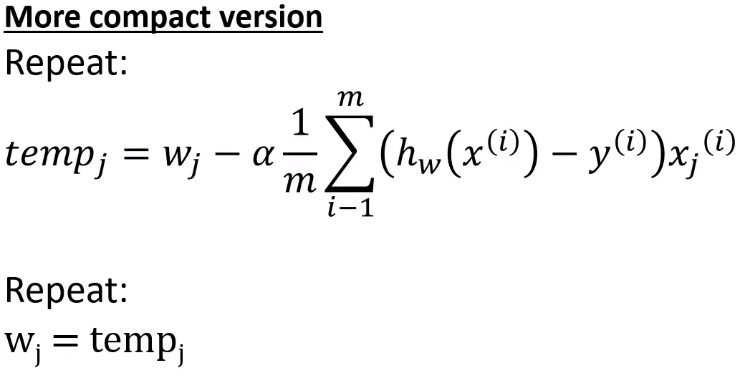
Before, one feature, x, used to predict y.

* hw(x) = w0 + w1x

Now

* hw(x) = w0 + w1x + w2x2 + w3+x3….wn+xn
* n = number of features
* m = number of training examples
* xi = input variables
* y = depdent variable

# Learning Algorithm for Linear Regression with Multiple Features



* Squared error cost function is still the same.
* Same way to determine if gradient descent is working

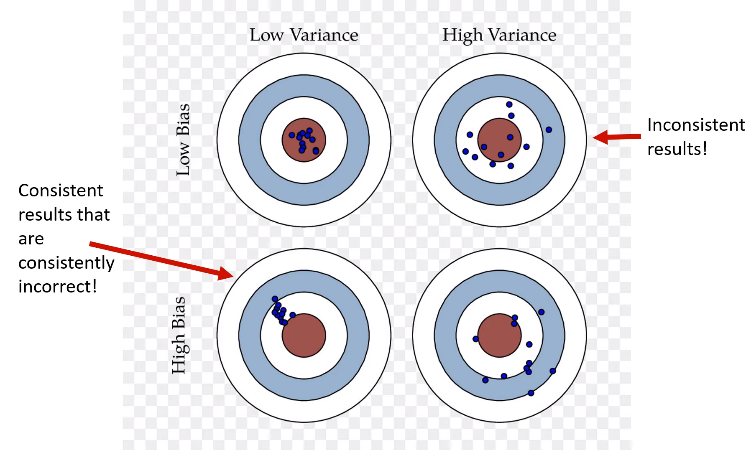
## Feature Scaling

Make sure features on similar scale

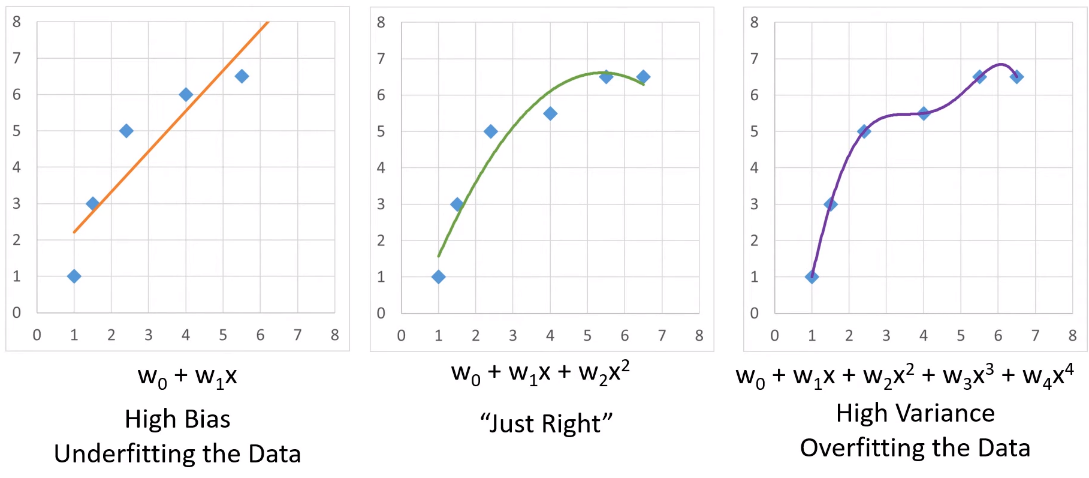
* Failure to do so will make gradient descent inefficient computationally
* Scale so each type has similar range of values
* Can scale every feature so that are approximately between -3 and 3
* Mean normalization
  + Replace xi with xi-ui to make features have approximately zero mean
  + Divide by either range of values or standard deviation

# Basic Evaluation

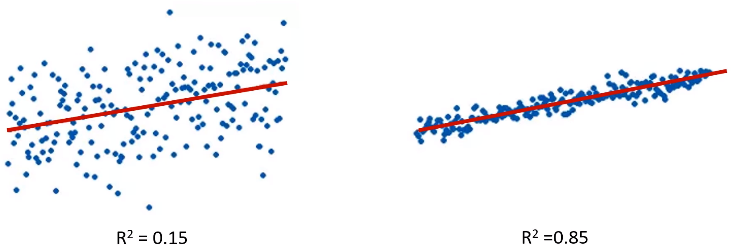
Evaluating hypothesis function:

* Just because it has low training error does not function will generalize to new examples that were not in training set
* We always want to test our system on data other than the data it was trained on
* A standard way to evaluate your hypothesis is to randomly divide your data into a training set (~70%) and a test set (~30%)
  + Training set to used to learn hypothesis that minimizes training error
  + Use that function to compute error on test set
* Potential problems:
  + Overfitting
    - If we have too many, not very important features, the learned hypothesis fits training set well but fails to generalize to new examples (High variance)
  + Underfitting
    - Hypothesis is too simple and does not capture all the important aspects of the data

Example for regression



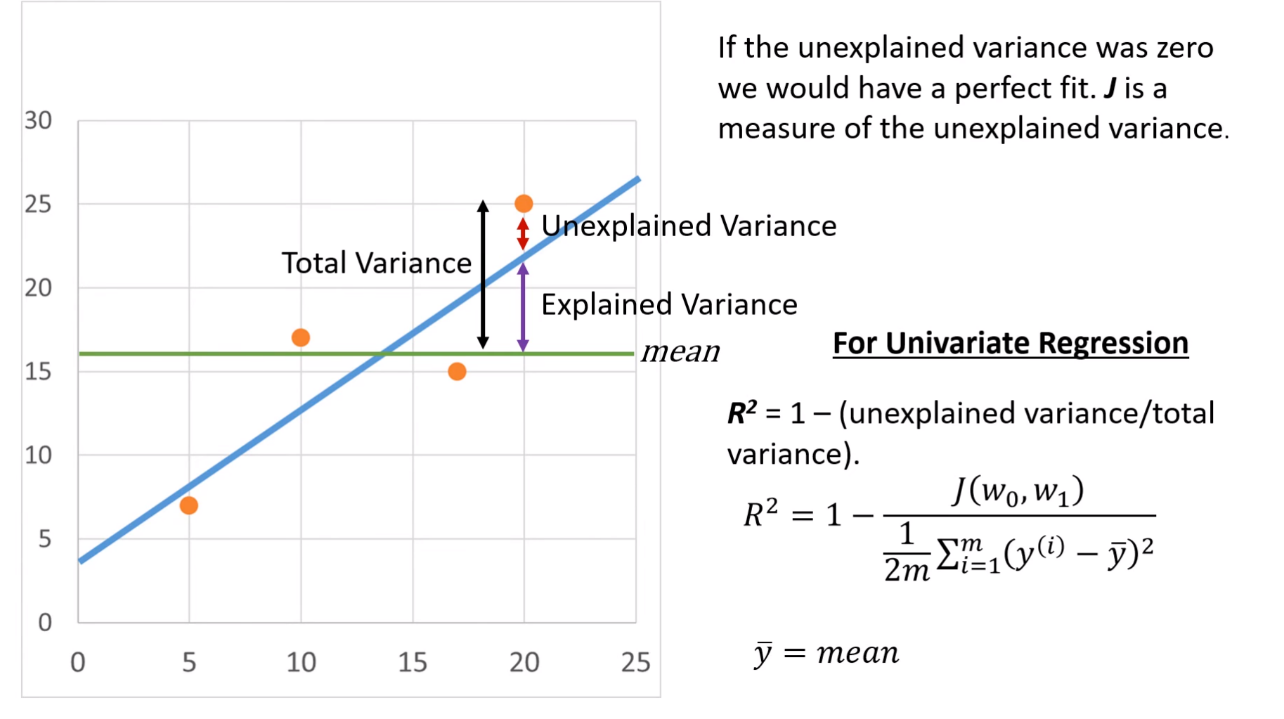
# Coefficient of Determination’

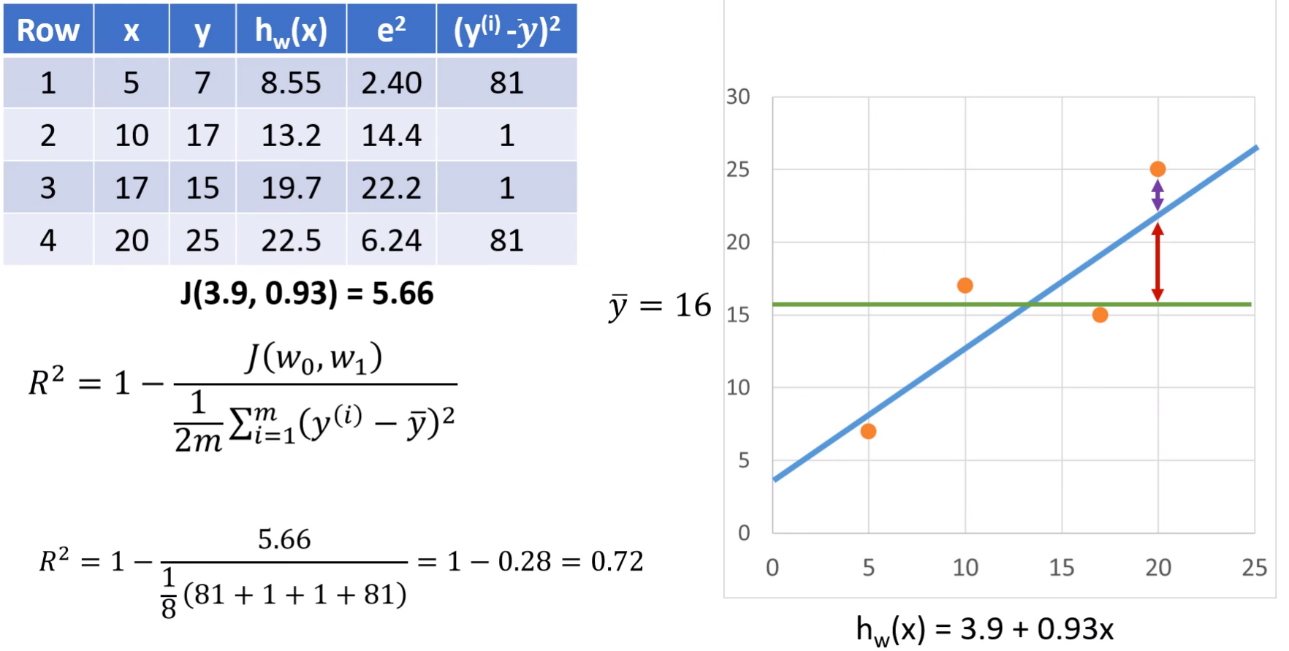
J doesn’t tell us if linear regression is a good model to predict y

Measure of how well regression fits the data is 0 <= R2 <= 1

* Coefficient of determination

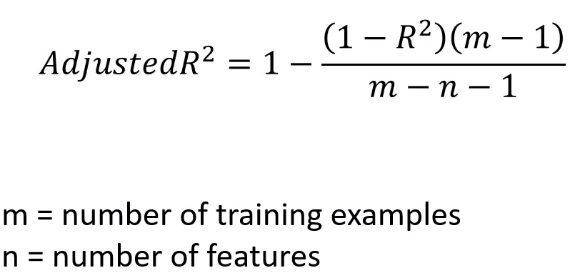
For univariate regression:





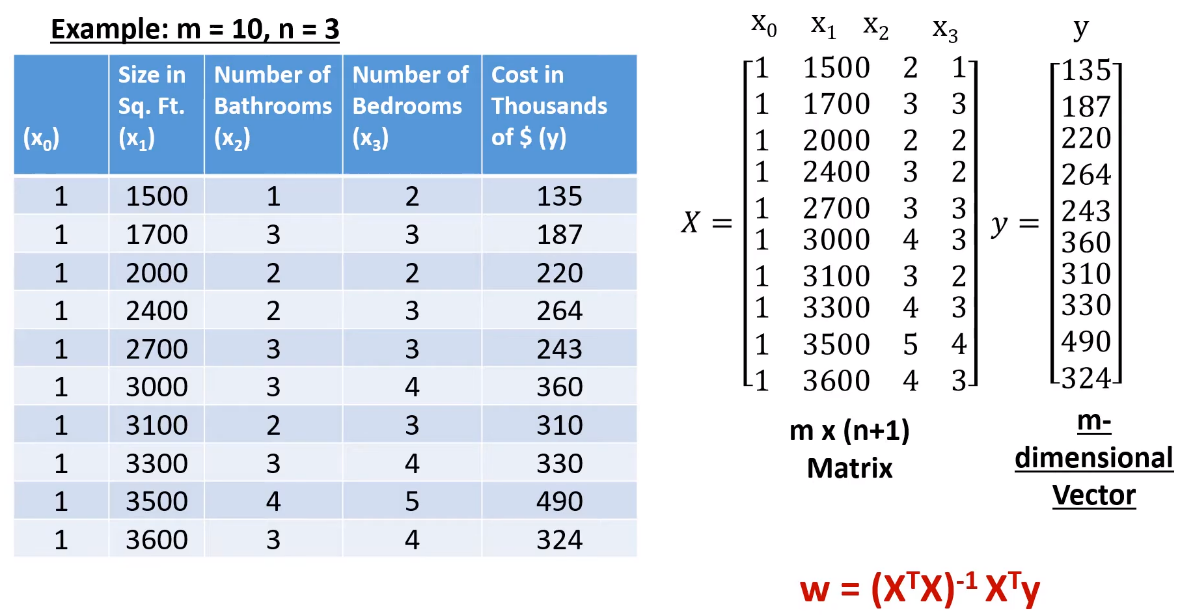
f

For multivariate:

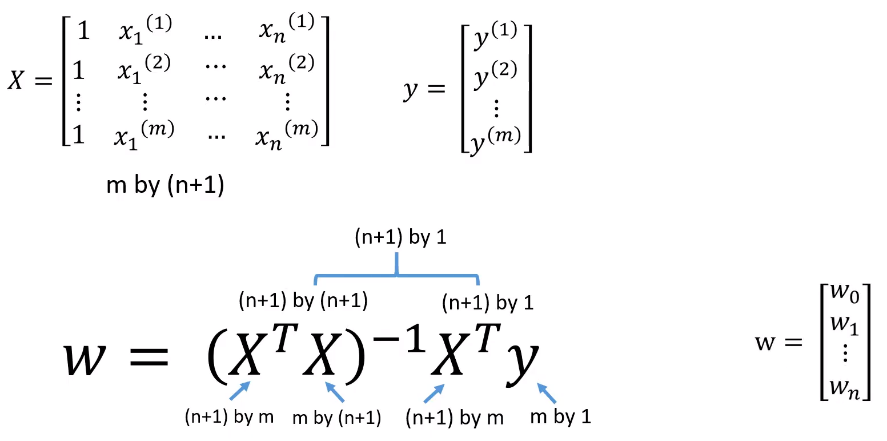
* Need to use Adjusted R2
* Measures proportion of variation explained by only those independent variables that really help in explaining dependent variable. Penalizes you for adding independent variable that do not help in predicting dependent variable

# The Normal Equation

Another idea from basic calculus

Example

In general for training examples and n features

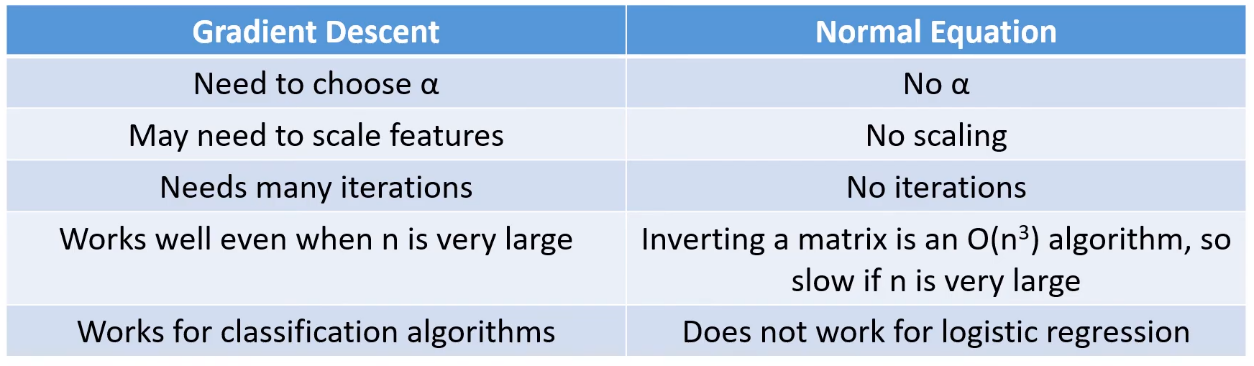


w = (XTX)-1XTy In numpy:

(XTX)-1 = np.linalg.pinv(np.dot(X.T, x))

XTy = np.dot(X.T, y)

therefore, w = np.dot(np.linalg.pinv(np.dot(X.T, X)), np.dot(X.T, y))



# Creating New Features

Don’t have to use features exactly as given, can define new from data.

Straight line not necessary, can do polynomial regression