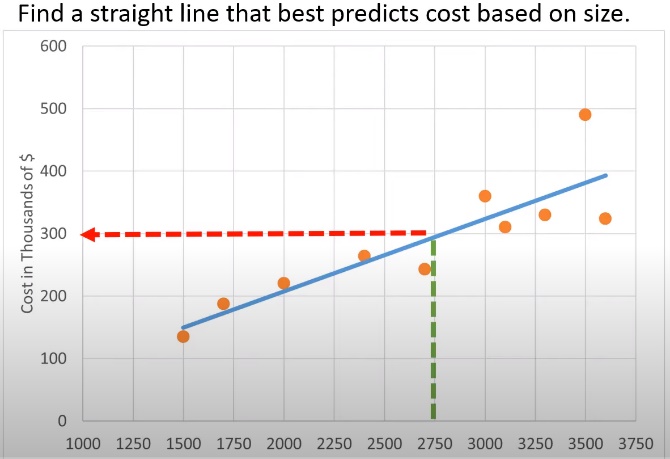
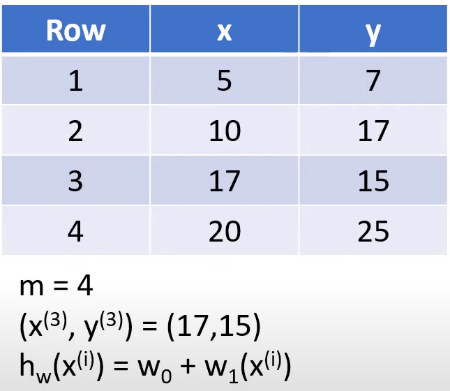
# Univariate Linear Regression

Predicts a real-valued output from one feature  
Supervised

Example: Predict cost of house based on its size in square feet

* Hw(x) is a hypothesis functions that predicts a y given x
* Represented as a linear function
* Hw(x) = w0 + w1x
  + W0 and w1 are weights (or parameters)

Basic Idea

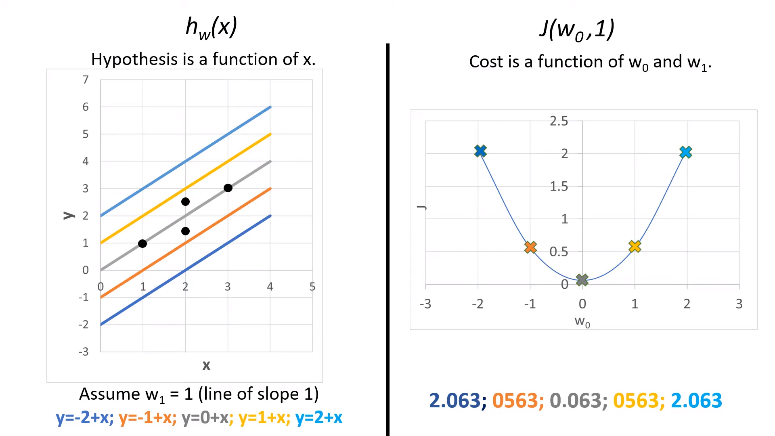
* Find a straight line that best fits observed data
* Best fit needs a definition!
  + Define error at each point (x, y) as the squared distance between y and hw(x)
  + Error = (h­w(x) – y)2
  + Why squared? Less emphasis on small errors, more emphasis on large errors
* To measure overall error:
  + Squared error cost function
  + J(w0, w1) = wxi – yi)2
    - M = number of training examples
    - X = input variable (feature)
    - Y = dependent variable
    - (Xi, yi) = ith training example

Best ways to find w0 and w1

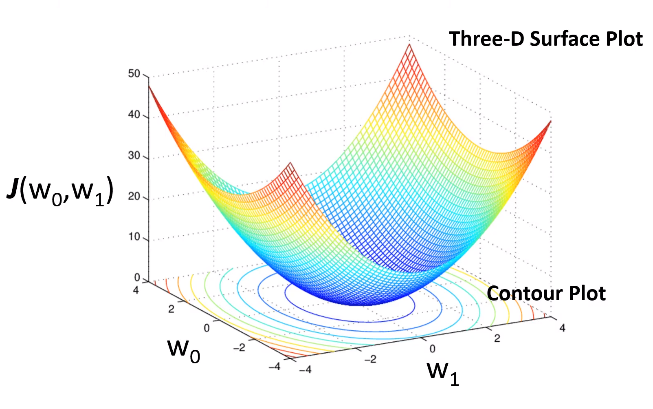
* Algebraic method with closed form:
  + Limited usefulness! Forget it
* Iterated method known as gradient descent
  + Need to know for logistic regression and neural networks
* Linear algebra method with matrix multiplication
  + Usually the best method for Regression

# Gradient Descent

Example:

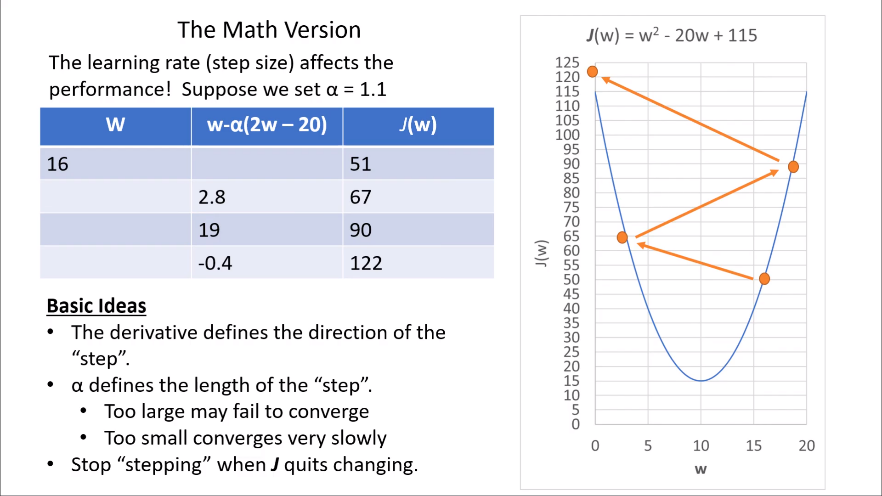
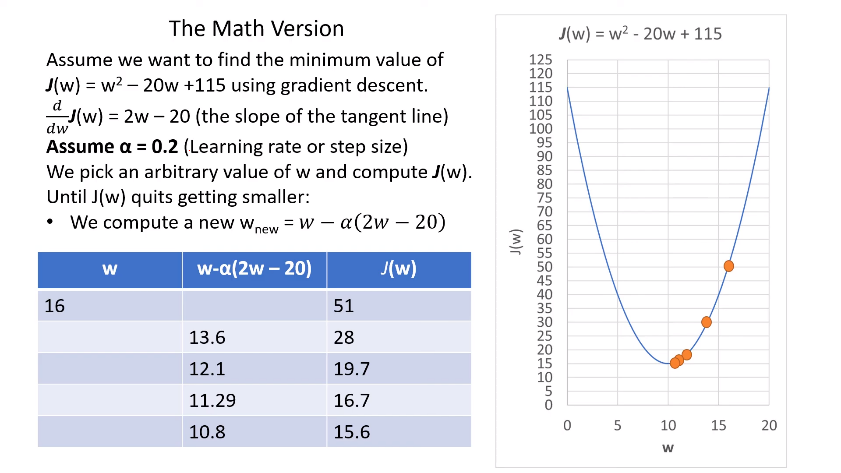
* J(w0, w1) = wxi – yi)2
* Iterates through each combination of potential weights to find lowest error
  + J(0, w1) = Iterates through all w1s with w0 as 0
  + J(w0, 1) = Iterates through all w0s with w0 as 1
* But we want to look at all combinations
  + Actually, a 3d surface plot
  + Want the function that has lowest value of J

Process

* We want to minimize J
* Start with some W0, W1, etc
* Keep changing Ws to reduce J until we end up at a minimum
* Note: General Descent is a general algorithm that can be used for more than just two variables

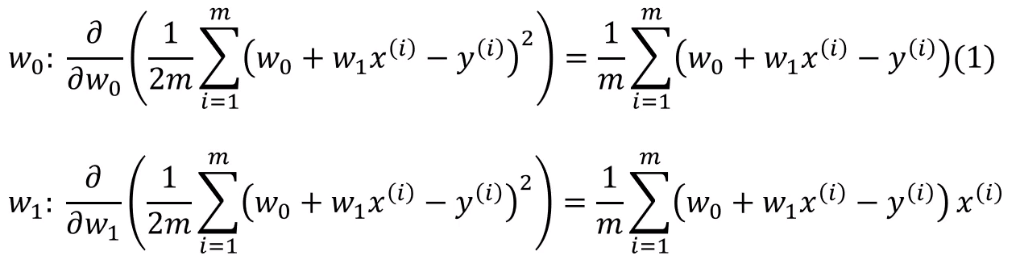
One variable example

* Assume we want to find minimum value of J(w) where J is a function of one variable
* J`(w) is slope of tangent line at W
* α is the learning rate
* Pick value of w and compute J(w)
* Repeat until J(w) reaches a minimum value
  + We compute new wnew = w – (α)J`(w)
  + Compute J(wnew)
* If derivative is positive then subtracting it from w moves new w to the left. Α is always positive and scales how far wnew moves.

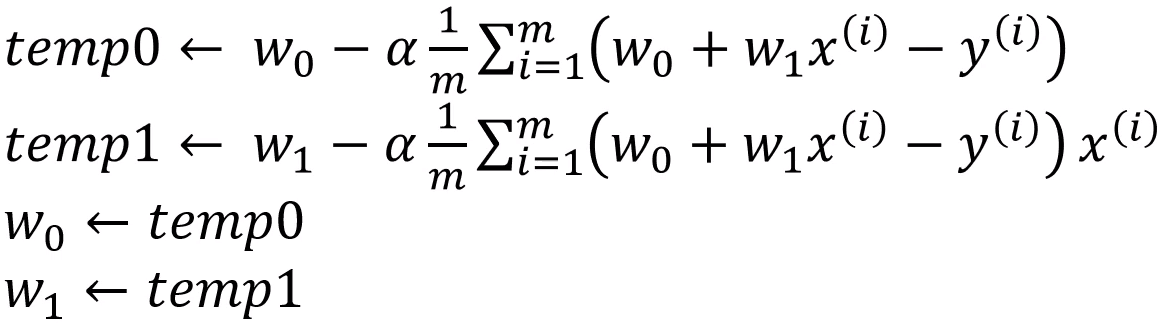


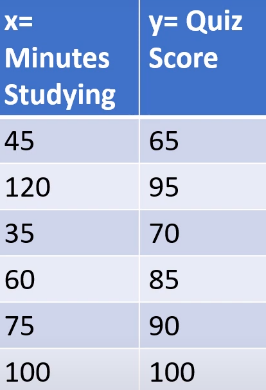
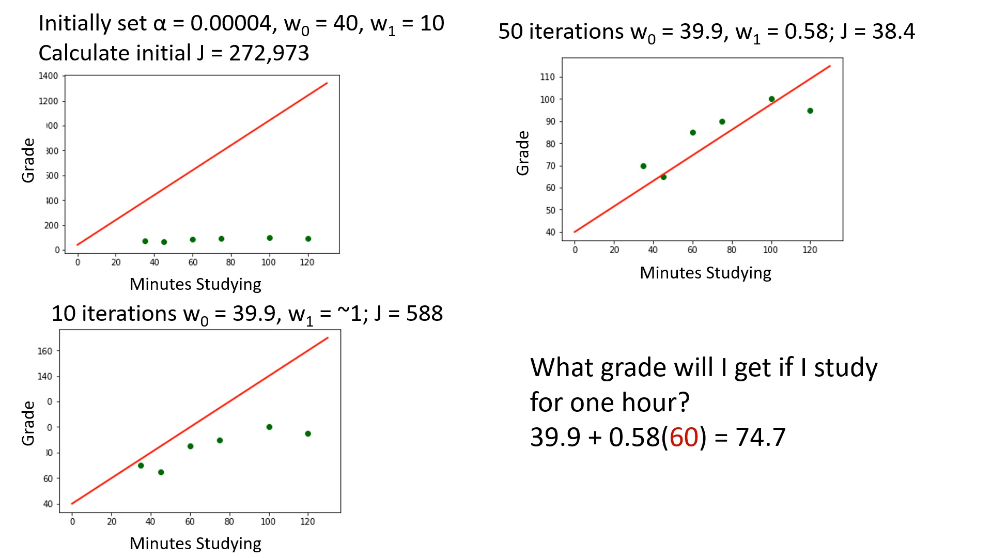
# Applying Gradient Descent

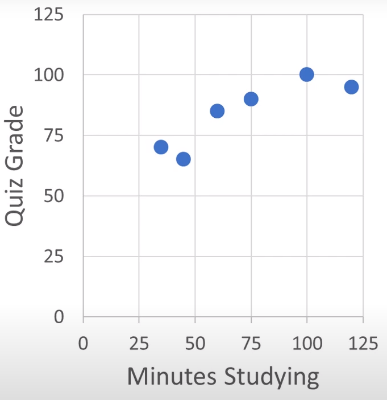
Reminder: Partial Derivatives and Chain Rule

* Hw(x) = w0 + w1x is our hypothesis function that our ML algorithm will use to predict y given x
* J(w0, w1) = wxi – yi)2 is our squared error cost function that varies with whatever w0, w1 we use in hw
* We need to compute partial derivative of J(w0, w1) with respect to w0 and W1

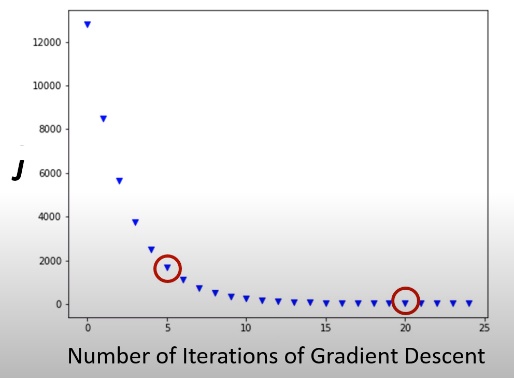
The Learning Algorithm for Linear Regression with One Variable is

* (If supervised) Read in training data and put into an (x, y) array
* Choose values for W0, W1, α
* Repeat until J converges
  + 
* The final values of w0, w1 are the best fit solution for your hypothesis function
  + Hw(x) = w0 + w1x
* (If supervised), use above function to predict on new data

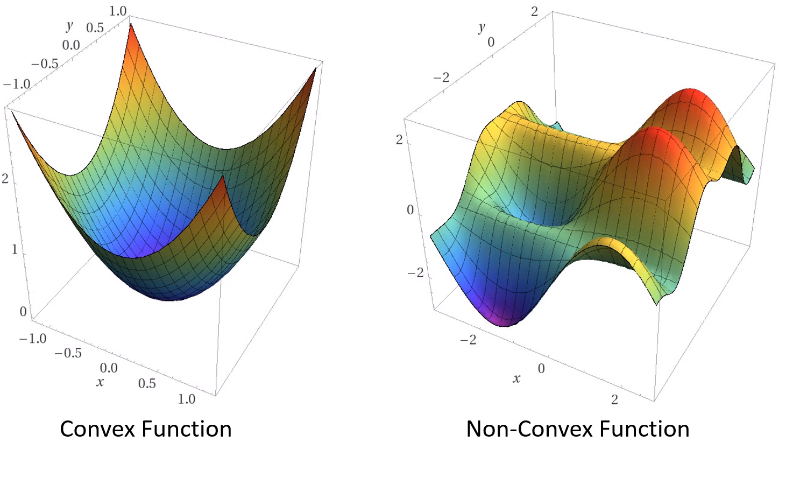
Tiny Example



How to determine if Gradient Descent is working and when to stop

* Each time you update weight, compute cost function
* Cost function should decrease with each iteration
* At some point, J is changing by tiny amounts, you can declare convergence
* Number of iterations needed can vary a lot
* Plot J vs number of iterations to declare convergence
* Can automatically declare convergence
  + Setting a rule such as if J decreases by less than small value (not recommended)

Choosing learning rate: α

* J(w) should be decreasing function with respect to number of ireations
* If J(w) is increasing or bouncing back and forth, learning rate too large
* If α is too small, computationally expensive

Problem with gradient descent to find minimum value

* If descent not uniform, and there are local minimums, you may declare convergence prematurely
* In this course, will only use convex functions