

Welcome

- ✓ **Video:** Welcome to Machine Learning!
1 min
- ✓ **Reading:** Machine Learning Honor Code
8 min

Introduction

- ✓ **Video:** Welcome
6 min
- ✓ **Video:** What is Machine Learning?
7 min
- ✓ **Reading:** What is Machine Learning?
5 min
- ✓ **Reading:** How to Use Discussion Forums
4 min
- ✓ **Video:** Supervised Learning
12 min
- ✓ **Reading:** Supervised Learning
4 min
- ✓ **Video:** Unsupervised Learning
14 min
- ✓ **Reading:** Unsupervised Learning
3 min
- ✓ **Reading:** Who are Mentors?
3 min
- ✓ **Reading:** Get to Know Your Classmates
8 min
- ✓ **Reading:** Frequently Asked Questions
11 min

Review

- ✓ **Reading:** Lecture Slides
20 min
- ✓ **Quiz:** Introduction
5 questions

Model and Cost Function

- ✓ **Video:** Model Representation
8 min
- ✓ **Reading:** Model Representation
~

Gradient Descent For Linear Regression

Note: [At 6:15 "h(x) = -900 - 0.1x" should be "h(x) = 900 - 0.1x"]

When specifically applied to the case of linear regression, a new form of the gradient descent equation can be derived. We can substitute our actual cost function and our actual hypothesis function and modify the equation to :

repeat until convergence: {

$$\begin{aligned}\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i) \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m ((h_{\theta}(x_i) - y_i)x_i) \\ &\end{aligned}$$

where m is the size of the training set, θ_0 a constant that will be changing simultaneously with θ_1 and x_i, y_i are values of the given training set (data).

Note that we have separated out the two cases for θ_j into separate equations for θ_0 and θ_1 ; and that for θ_1 we are multiplying x_i at the end due to the derivative. The following is a derivation of $\frac{\partial}{\partial \theta_j} J(\theta)$ for a single example :

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_{\theta}(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_{\theta}(x) - y) \\ &= (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i - y \right) \\ &= (h_{\theta}(x) - y) x_j\end{aligned}$$

The point of all this is that if we start with a guess for θ our hypothesis and then repeatedly apply these gradient descent equations, our hypothesis will become more and more accurate.

So, this is simply gradient descent on the original cost function J. This method looks at every example in the entire training set on every step, and is called **batch gradient descent**. Note that, while gradient descent can be susceptible to local minima in general, the optimization problem we have posed here for linear regression has only one global, and no other local, optima; thus gradient descent always converges (assuming the learning rate α is not too large) to the global minimum. Indeed, J is a convex quadratic function. Here is an example of gradient descent as it is run to minimize a quadratic function.

