Stanford CS 229 - Machine Learning - Lecture 2

Giovani da Silva

June 2021

1 An Application of Supervised Learning - Autonomous Deriving

Some notations definitions for the rest of the course: m :training examples

x :input variables(features)

y :output variable/"target"

(x,y) : training example

(x_i,y_i) : ith training example

theta's : parameters

2 Supervised Learning

Given a training set, and we're going to feed our training set, comprising our M training example into a learning algorithm. Our algorithm then has an output function usually denoted by h, the hypothesis

What the hypothesis does is it takes this input, a new living area in square feet,for example, and output will estimates the price of this house So the hypothesis H maps from inputs X to outputs Y.

In order to design a learning algorithm, the first thing we have to decide is how we want to represent the hypothesis.

If we have only one feature:

$$h(x) = \theta_0 + \theta_1 x \tag{1}$$

With 2 features:

Example: x_1 = Size , x_2 = numbers of bedrooms

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{2}$$

for conciseness, defines x_0=1

n = number of features

$$h(x) = \sum_{n=0}^{n} \theta_i x_i = 1 \tag{3}$$

To chose the parameters so that our hypothesis h will make make accurate

predictions.We want to minimize over the parameters theta, so the squared area between the predicted price and the actual price.

$$J(\theta) = 1/2 \sum_{i=1}^{n} (h_\theta(x_i) - y_i)^2 \tag{4}$$

We want to minimize $J(\theta)$

There are some algorithms for performing that minimization over theta:

Search algorithm (Say $\theta = 0$):
Initialize vector θ to be the vector of all zeros. Keep changing theta to reduce J(θ) until we hopefully end up at the minimum with respect to theta of J of θ.

Gradient descent:
Take small steps to minimize J of theta as fasts as possible. Will go downhill in the gradient function, keep going until it gets to local minimum of this function. Gradient descent can sometimes depends on where you initialize your parameters.

Alpha is the parameter that control how big the steps are. Usually set by hand.

$$\theta_i := \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i} \tag{5}$$

Example: One training example

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{(h_\theta(x) - y)^2}{\partial \theta_i} \tag{6}$$

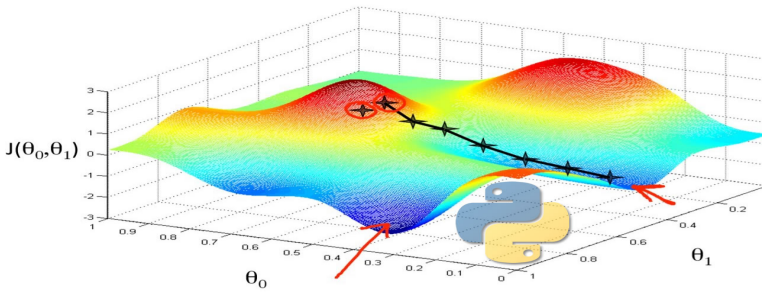
The derivation results:

$$(h_\theta(x) - y)\frac{\partial(\theta_0 x_0 + ...\theta_n x_n - y)}{\partial \theta_i} \tag{7}$$

$$There's only one ther m that depends of the tai, so := h(\theta_x - y)x_i \tag{8}$$

Finally:

$$\theta_i := \theta i - \alpha(h(x) - y)x_i \tag{9}$$



Batch Gradient Descent Algorithm:

The term batch refers to the fact that on every step of gradient descent you're going to look at your entire training set.
Repeat till convergence:

$$\theta_i := \theta i - \alpha \sum_{j=1}^{m} (h_\theta(x_j) - y_j)^2 x_i(j) \tag{10}$$

If m is a few million then if you're running batch rate and descent, this means that to perform every step of gradient descent you need to perform a sum from J equals one to a million. Which is not optimal. So we will need an alternative algorithm, the Stochastic gradient descent.

Stochastic gradient descent:

Repeat for j = i to n

$$\theta_j := \theta j - \alpha(h_\theta(x_j) - y_j)x_j$$

for all j

The advantage of this algorithm is that in order to start learning, in order to start modifying the parameters, you only need to look at your first training examples.

Uses the first training example and perform an update using the derivative of the error with respect to just your first training example, and then you look at your second training example and perform another update.It keeps adapting your parameters much more quickly without needing to scan over your database.

For large data sets, Stochastic gradient descent is often much faster and what happens is that constant gradient descent is that it won't actually converge to the global minimum exactly but it will contours the function. The parameters tends to wander to the region of global minimum.

Gradient descent can also be represented by:

$$\theta := \theta - \alpha\nabla_\theta J \tag{11}$$

also, after some linear algebra deductions and X being the vector of x's:

$$\theta = (X^T X)^{-1} X^T y \tag{12}$$