II. Linear Regression With One Variable

Training set of housing prices in Portland, OR

Size in feet^2 (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
⋮	⋮

Lecture 5: Model Representation

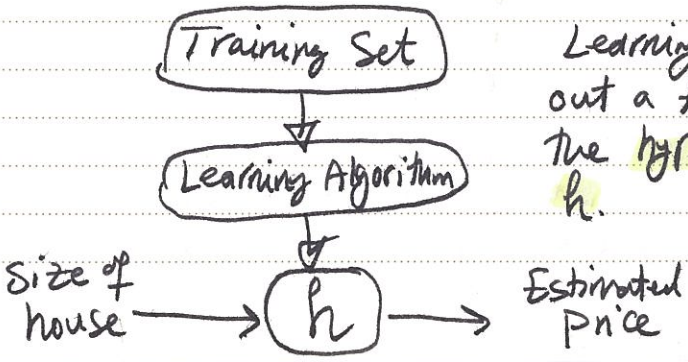
Notation:
m = Number of training examples
x's = "input" variable/feature
y's = "output" variable/"target" variable

*Our job: to learn from this data how to predict price of houses as a function of their size.

→ This is, of course, supervised learning.
→ It's, moreover, regression because the output variable is continuous.

More notation: $(x^{(i)}, y^{(i)})$ - i^{th} training example

Big Picture:



Learning algorithm spits out a function, called the hypothesis function h.

Question: How do we represent h?

→ We start with the simplest thing possible, & then consider more complex models later:

$$\boxed{h_\theta(x) = \theta_0 + \theta_1 x}$$ Univariate linear regression

which just means that we're assuming a linear relationship between housing prices & size of houses.

Lecture 6: Cost Function

What are the values of θ_0 & θ_1 which provide the best fit to our training set?

Notation: $\boxed{\theta_i\text{'s} = \text{parameters}}$

well, we would want $h_\theta(x^{(i)})$ to be as close to $y^{(i)}$ as possible; so we can do the following:

$$\boxed{J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}$$

Cost function

& then pick θ_0 & θ_1 which minimizes this cost function.

→ Why this cost function & not some other measure of model-data error? We will come back to this later, but as it turns out, this particular cost function works well for linear regression.

→ The most obvious obervation is that for $h_\theta(x) = \theta_0 + \theta_1 x$, $J(\theta_0, \theta_1)$ is a paraboloid, which has only one minimum, i.e the global minimum.

Lecture 7(8): Cost Function Intuition I (II)

Some nice videos & plots on why minimizing $J(\theta_0, \theta_1)$ leads to a good fit of data (which shows some liner $y-x$ relationship of course).

Lecture 9: Gradient Descent

→ Gradient Descent is an algorithm for minimizing the cost function $J(\theta_0, \theta_1)$. Of course, for the model $h_\theta(x) = \theta_0 + \theta_1 x$, this can be done analytically:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) = 0 \qquad (1)$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) x^{(i)} = 0 \qquad (2)$$

$$\theta_0 \sum_{i=1}^{m} 1 + \theta_1 \sum_{i=1}^{m} x^{(i)} = \sum_{i=1}^{m} y^{(i)}$$

$$\theta_0 \sum_{i=1}^{m} x^{(i)} + \theta_1 \sum_{i=1}^{m} x^{(i)} x^{(i)} = \sum_{i=1}^{m} x^{(i)} y^{(i)}$$

$$\boxed{\begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} 1 & \sum_{i=1}^{m} x^{(i)} \\ \sum_{i=1}^{m} x^{(i)} & \sum_{i=1}^{m} x^{(i)} x^{(i)} \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{m} y^{(i)} \\ \sum_{i=1}^{m} x^{(i)} y^{(i)} \end{bmatrix}}$$

→ We will later see that sometimes it is advantageous to use Gradient Descent in more generalized situations (e.g. when there are many features). Indeed, the reason why we were about to minimize J analytically is because it depends quadratically on θ_i, which in turn can be traced back to the fact our hypothesis function h is linear in θ_i.

→ Here will talk about using gradient descent to minimize an arbitrary function.

* Have some function $J(\theta_0, \theta_1)$

* Want $\min\limits_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

* To do: → start with some θ_0, θ_1
  → Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum.

* How would we change (θ_0, θ_1)?

One algorithm that makes intuitive sense is the following: start at some $\langle \theta_0, \theta_1 \rangle$. Then look to see in what direction J is decreasing the fastest at that point. Take a step in that direction. Repeat this at the new point, & so on, until you reach a (possibly local) minimum. The main question then becomes: what is the direction along which J decreases the fastest?

Answer: minus the direction of the gradient of J.

Let's see how we can prove this.

Proof

Consider a function $J(\theta_1, \theta_2, \ldots, \theta_n)$. Let's say we're sitting at the point $\vec{\theta}$. We pick a unit vector $\hat{n} = \frac{\vec{n}}{\sqrt{\vec{n}\cdot\vec{n}}}$. The rate of change of J at point $\vec{\theta}$ along $\hat{n}$ is:

$$\hat{n}^i \frac{\partial J}{\partial \theta^i}(\vec{\theta}).$$

We would like to find the unit direction along which J decreases the fastest at point $\vec{\theta}$. This is equivalent to minimizing the function $f(\vec{n}) \equiv \frac{n^i}{\sqrt{\vec{n}\cdot\vec{n}}} \frac{\partial J}{\partial \theta^i}(\vec{\theta})$.

Let's do so:

$$\frac{\partial f}{\partial n^k} = \frac{\partial J}{\partial \theta^i}(\vec{\theta}) \left[\frac{\delta^i_k}{\sqrt{\vec{n}\cdot\vec{n}}} - \frac{n^i n_k}{(\vec{n}\cdot\vec{n})^{3/2}} \right] = 0$$

claim: the solution is $n_k = \pm \frac{\partial J}{\partial \theta^k}(\vec{\theta})$ or $\boxed{\vec{n} = \pm \nabla J(\vec{\theta})}$

check: $\frac{\partial f}{\partial n^k} = \frac{1}{\sqrt{\nabla J \cdot \nabla J}} \frac{\partial J}{\partial \theta^k}(\vec{\theta}) - \frac{\nabla J \cdot \nabla J}{(\nabla J \cdot \nabla J)^{3/2}} \frac{\partial J}{\partial \theta^k}(\vec{\theta}) = 0$

Which one do we pick?

$$f(\pm \nabla J(\vec{\theta})) = \pm \sqrt{\nabla J(\vec{\theta}) \cdot \nabla J(\vec{\theta})}$$

Since we want the direction in which J decreases the fastest, we want

$$\boxed{\vec{n} = -\nabla J(\vec{\theta})} \qquad (\leftrightarrow \text{negative } f)$$

Then our algorithm becomes:

$$\boxed{\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}(\vec{\theta})}$$

Gradient Descent Algorithm

Learning Rate: determines how long a step we take along the $-\nabla J$ direction. $(\alpha > 0)$

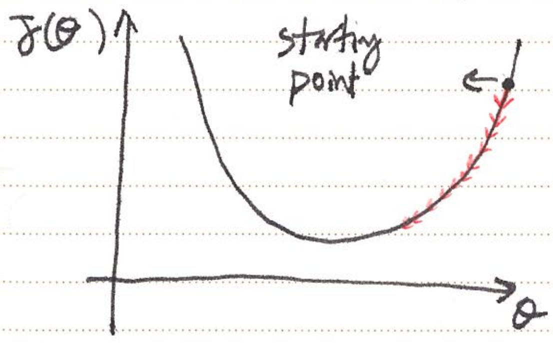
Lecture 10: Gradient Descent Intuition

We explained the logic behind the algorithm, but let's dig a little deeper into some of its feature.

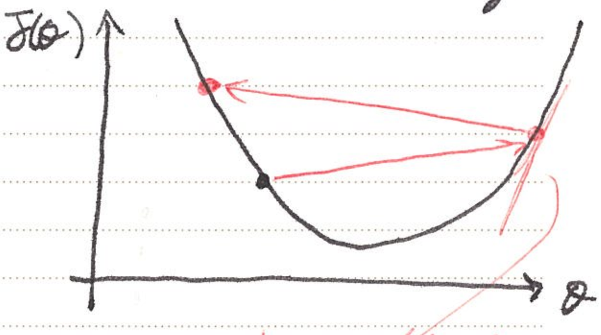
- If θ is already at a local minimum, i.e. $\nabla J(\theta) = 0$, then gradient descent does no update, which is what one wants.

- How big should α be?

Too small: slow convergence

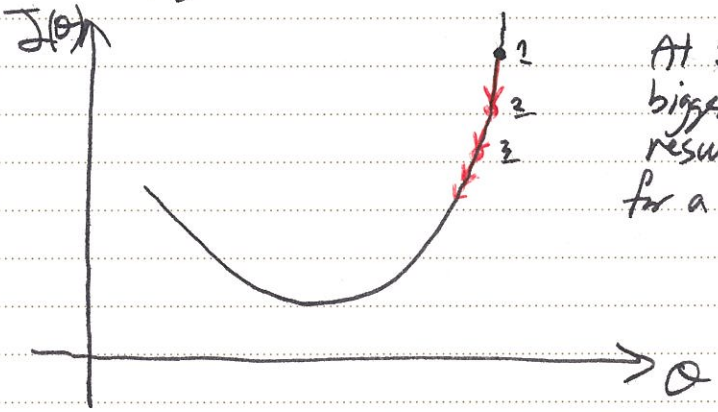


Too big: overshooting & even divergence:



Since the derivative line is larger, the next step becomes even bigger for a fixed value of α.

An interesting feature of gradient descent is that even for a fixed value of α, the steps become smaller & smaller as we approach the minimum, simply because the derivative term becomes smaller:



At 1, the derivative is bigger than 2, so the resulting step is bigger for a fixed α.

Lecture 11: Gradient descent for Linear Regression

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_0+\theta_1 x^{(i)} - y^{(i)}\right)^2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}\left(\theta_0+\theta_1 x^{(i)} - y^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m}\left(\theta_0+\theta_1 x^{(i)} - y^{(i)}\right)x^{(i)} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x^{(i)}$$

where $h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$

Gradient descent algorithm:

repeat until convergence {

$$\theta_0 := \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}\left(\theta_0+\theta_1 x^{(i)} - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \frac{\alpha}{m}\sum_{i=1}^{m}\left(\theta_0+\theta_1 x^{(i)} - y^{(i)}\right)x^{(i)}$$

}

2D for a complicated function with many local minima, gradient descent might end up at different minima depending on the starting point ←

→ Of course, this is not an issue for linear regression since the cost function only has one global minimum.

* Lecture 12 is an overview of what's to come & Lectures 13–18 are about reviewing basics of linear algebra. *