IV. Linear Regression With Multiple Variables

Lecture 19: Multiple Features

In the housing price example, our goal was to predict the price of a house as a function of its size. Of course, there are many more factors which could affect house prices:

Multiple features (variables)

Size x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
...	...	...	...	...

Notation:

$n \equiv$ number of features

$x_j^{(i)} \equiv$ value of j^{th} feature in i^{th} training example

$x^{(i)} \equiv$ input (features) of i^{th} training example

Example above: $n = 4$; $x_3^{(2)} = 2$; $x^{(3)} = \begin{bmatrix} 1534 \\ 3 \\ 2 \\ 30 \end{bmatrix}$

Our hypothesis now is:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Notation:

* $x_0 \equiv 1 \leftrightarrow x_0^{(i)} = 1$
* $x^{(i)} \equiv \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$
* $\theta \equiv \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$

so that $\boxed{h_\theta(x) = \theta^T x}$

Lecture 20: Gradient Descent for Multiple Variables

Cost function: $J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

where $h_\theta(x^{(i)}) = \theta^T x^{(i)} = [\theta_0 \; \theta_1 \cdots \theta_n] \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left[\sum_{j=1}^{n} \theta_j x_j^{(i)} - y^{(i)} \right]^2$$

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{2m} \sum_{i=1}^{m} 2 \left(\sum_{j=1}^{n} \theta_j x_j^{(i)} - y^{(i)} \right) x_j^{(i)} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x

Note that we can write the cost function as:

$$J(\theta) = \frac{1}{2m} (X\theta - Y)^T (X\theta - Y)$$

where $X_{ij} \equiv x_j^{(i)}$ & $Y_i = y^{(i)}$

Also, gradient descent can be written as

Repeat until converge {

$$\theta := \theta - \frac{\alpha}{m} X^T (X\theta - Y)$$

}

Of course, these only hold for linear regression.

Gradient Descent: repeat until convergence {

$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)}$$

} (simultaneously update θ_j for $j = 0, 1, \dots, n$)

Lecture 21: Gradient Descent in Practice I - Feature Scaling

→ Different features may have very different values.

Example) → Size of house ~ 1000 (feet^2)
→ Number of rooms ~ 1-5

This causes the cost function J to be a lot more stretched in certain directions than others. For instance, in the example above, taking $x_1 \equiv$ size of house & $x_2 \equiv$ number of units & the hypothesis function:

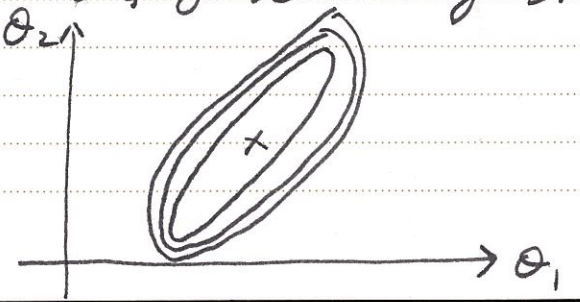
$$h_\theta(x) = \theta_1 x_1 + \theta_2 x_2$$

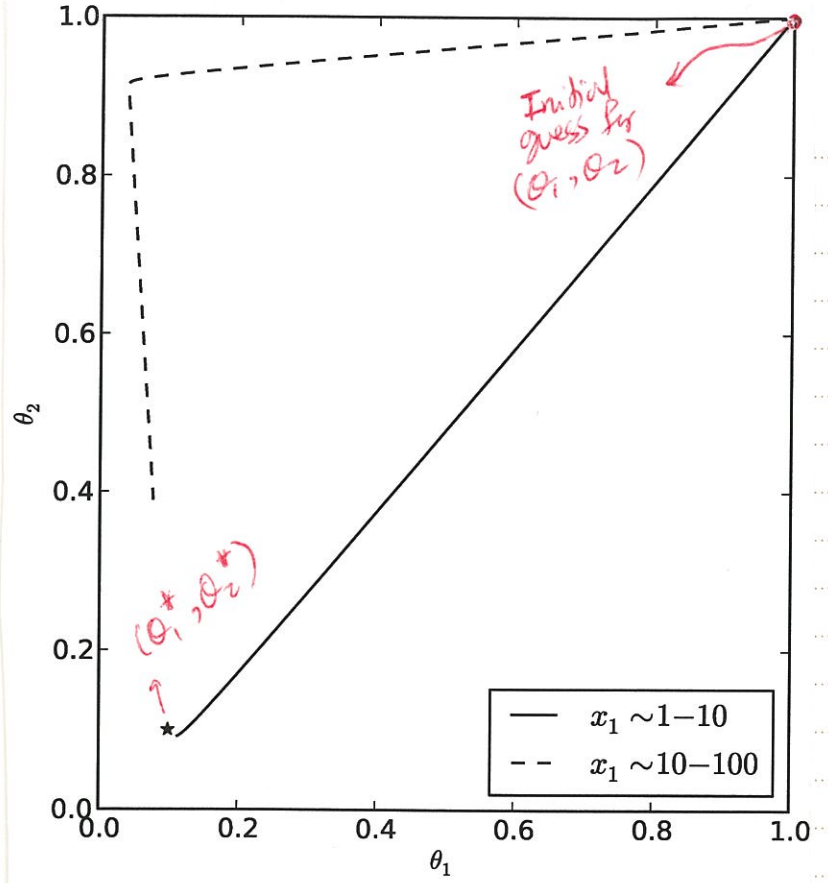
Then, an order of magnitude estimate of the cost function $J(\theta)$ would be:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} - y^{(i)})^2$$

$$\sim 10^6 \theta_1^2 + 10\, \theta_2^2 + \dots$$

So that the contours of J become very skewed:





To see the effect this could have on gradient descent, I did the following exercise:

① ⊛ Pick a target value θ_1^*, θ_2^*.

⊛ Pick m random numbers uniformly for both x_1 & x_2:

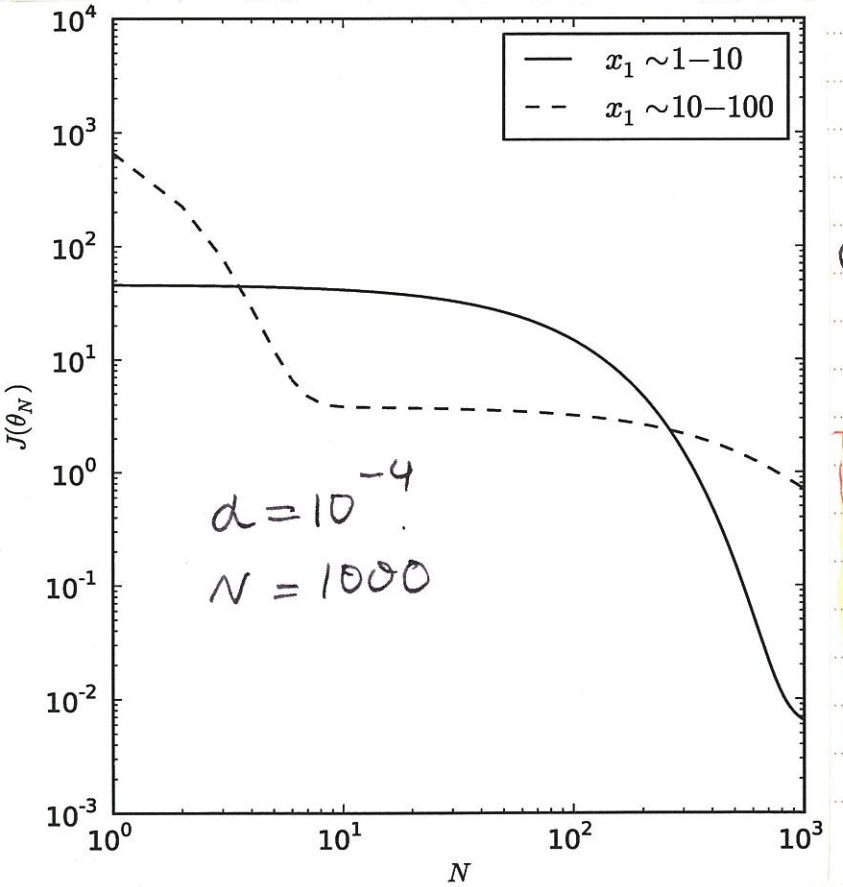
$x_1 \sim 1-10$

$x_2 \sim 1-10$

⊛ Construct $y^{(i)} = \theta_1^* x_1^{(i)} + \theta_2^* x_2^{(i)}$ & add a bit of noise to $y^{(i)}$

⊛ Run Gradient descent on the training set $(x_1, x_2; y)$ using the model $h_\theta(x) = \theta_1 x_1 + \theta_2 x_2$

② ⊛ Do exactly as above except pick x_1 from a range $x_1 \sim 10-100$.



Conclusion: Gradient Descent takes much longer to converge when the features are not of the same order!

To alleviate this problem, we do what is known as feature scaling:

→ Scale all features so that they are in the similar range $(-1, 1)$.

→ Some ways of doing this:

$$x_j^{(i)} \longrightarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

where μ_j & σ_j are the mean & standard deviation of the feature j.

$$x_j^{(i)} \longrightarrow \frac{x_j^{(i)} - \mu_j}{S_j}$$

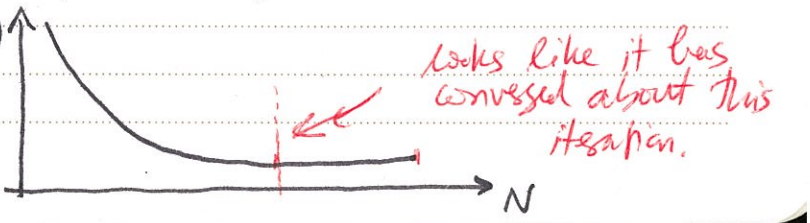
where S_j is the range of values of feature j, i.e $\max\{x_j^{(i)}\} - \min\{x_j^{(i)}\}$.

Lecture 22: Gradient Descent in Practice II - Learning rate

→ A good check of seeing if gradient descent is working correctly is to plot the value of the cost function as a function of the iterations in gradient descent. The bottom figure on the previous page is a plot of this sort. The important thing to note that J should decrease with every iteration. If this is not the case, something is wrong. (Of course, provided the learning rate is small enough)

→ If J is increasing with the number of iterations, it usually means that the learning rate α is too large.

→ This plot also helps us to see if gradient descent has converged or not.

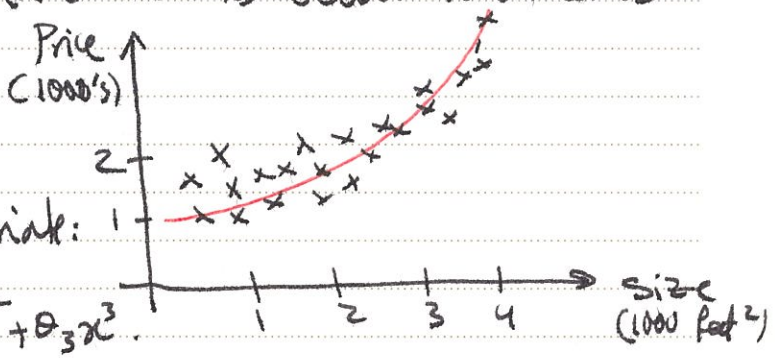


Choosing α: → ~~try~~ running gradient descent with different values of α & plot $J(\theta_N)$ - N.

(Example) $\alpha = 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \ldots$

Lecture 23: Features and Polynomial Regression

What if we have data which is better modeled as a polynomial fit?



Let's say we think the following model is appropriate:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3.$$

Well, we can then easily use our linear regression techniques by simply letting: $\begin{cases} x_1 = x \\ x_2 = x^2 \\ x_3 = x^3. \end{cases}$

As long as $h_\theta(x)$ is linear in θ, we can use linear regression! This is quite powerful.

⇒ Note that feature scaling becomes crucial in such cases. If we take x to be the size of the house in square-feet, then:

$$x_1 \sim 1 - 1000$$
$$x_2 \sim 1 - 10^6$$
$$x_3 \sim 1 - 10^9$$

⇦

How can we tell what the best fit to the data should be? Well, apparently there are algorithms which automatically find the best fit (e.g. if a 2nd degree polynomial is appropriate, a 3rd degree, a mixing of features, etc.)

2▷ My guess is that such an algorithm would try to find a fit which has as low a $J(\theta_{fit})$ as possible, while minimizing the number of features introduced.

2▷ Note that we can get pretty fancy: say we have a training set with input variables x_1, x_2 & output variable y. We can apply linear regression to:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \cdots$$

where $x_3 = \sqrt{x_1 x_2}$; $x_4 = e^{x_1 - x_2}$; ....

Lecture 24: Normal Equation

We can, of course, just minimize the cost function J analytically.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left[\sum_{j=0}^{n} \theta_j x_j^{(i)} - y^{(i)} \right]^2$$

$$\frac{\partial J}{\partial \theta_k} = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{j=0}^{n} \theta_j x_j^{(i)} - y^{(i)} \right) x_k^{(i)} = 0$$

$$\sum_{j=0}^{n} \theta_j \left(\sum_{i=1}^{m} x_j^{(i)} x_k^{(i)} \right) = \sum_{i=1}^{m} y^{(i)} x_k^{(i)} \quad (\#)$$

called the Design Matrix

Let $\boxed{X_{ij} \equiv x_j^{(i)} \quad \& \quad Y_i = y^{(i)}}$.

Then $\sum_{i=1}^{m} x_j^{(i)} x_k^{(i)} = \sum_{i=1}^{m} X_{ij} X_{ik} = [X^T X]_{kj}$

$\sum_{i=1}^{m} y^{(i)} x_k^{(i)} = \sum_{i=1}^{m} X_{ik} Y_i = [X^T Y]_k$

Then ⊕⊕ on previous page reads:

$$\sum_{j=0}^{n} [X^T X]_{kj} \theta_j = [X^T Y]_k$$

$$\Rightarrow [X^T X] \textcircled{H} = X^T Y$$

&

$$\boxed{\textcircled{H} = [X^T X]^{-1} X^T Y}$$

This exact solution is known as the normal equation.

Example)

x_0	size (feet^2) x_1	# of bedrooms x_2	Price ($1000) y
1	2104	5	460
1	1416	3	232
1	1534	3	315
1	852	2	178

$$X = \begin{bmatrix} 1 & 2104 & 5 \\ 1 & 1416 & 3 \\ 1 & 1534 & 3 \\ 1 & 852 & 2 \end{bmatrix} \quad \& \quad Y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

Note that X is $m \times (n+1)$ dimensional

In the language we established on pg 17:

$$X = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \quad \text{where } x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ \vdots \\ x_m^{(i)} \end{bmatrix}$$

2) Andrew claims that feature scaling is not necessary here. I can see why, but it would be nice if the elements of X are of the same order of magnitude...

2) Gradient Descent vs. Normal Equation

* Obviously, wherever there is an analytic solution, life becomes much easier. The only computational challenge in using the normal equation is inverting $(X^T X)_{n \times n}$

* Inversion is an order n^3 operation & for large n, gradient descent works much better.

* Rough rule: $\begin{cases} n \lesssim 10000 \rightarrow \text{Normal Equation} \\ n \gtrsim 10000 \rightarrow \text{Gradient Descent.} \end{cases}$

Remember: $n = \#$ of features!

* Indeed, gradient descent is used to solve large system of linear equations of the form $Ax = b$, when inverting A directly becomes very expensive.

The optimal learning rate for linear regression

→ This is not discussed in the course, but when we're using a model linear in the fitting parameters θ, there's an optimal value of α.

→ Say we start at $\hat{\theta} = \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \\ \vdots \\ \hat{\theta}_n \end{bmatrix}$. The gradient of $J(\theta)$ at $\hat{\theta}$ is $\hat{dJ} = \begin{bmatrix} \frac{\partial J}{\partial \theta_0}(\hat{\theta}) \\ \frac{\partial J}{\partial \theta_1}(\hat{\theta}) \\ \vdots \\ \frac{\partial J}{\partial \theta_n}(\hat{\theta}) \end{bmatrix}$. Our next move would be to $\hat{\theta} - \alpha \hat{dJ}$. What is the optimal α? Well, we can pick the α which would lead to the lowest value of J! In other word, consider $J(\hat{\theta} - \alpha \hat{dJ})$ & find the value of α which minimizes it: $\frac{d}{d\alpha} J(\hat{\theta} - \alpha \hat{dJ}) = 0$

$$J(\hat{\theta} - \alpha \hat{dJ}) = \frac{1}{2m} \sum_{i=1}^{m} \left[(\hat{\theta} - \alpha \hat{dJ})^T x^{(i)} - y^{(i)} \right]^2$$

$$\frac{d}{d\alpha} J(\hat{\theta} - \alpha \hat{dJ}) = \frac{1}{m} \sum_{i=1}^{m} \left[(\hat{\theta} - \alpha \hat{dJ})^T x^{(i)} - y^{(i)} \right] \times - \hat{dJ}^T x^{(i)}$$

$$= 0$$

$$\sum_{i=1}^{m} (\hat{\theta}^T x^{(i)})(\hat{dJ}^T x^{(i)}) = \sum_{j,k=1}^{n} \hat{\theta}_j \hat{dJ}_k \sum_{i=1}^{m} x^{(i)}_j x^{(i)}_k$$

$$= \hat{\theta}^T X^T X \hat{dJ} = (X\hat{\theta})^T X \hat{dJ}$$

$$\rightarrow (X\hat{\theta})^T X \hat{dJ} - \alpha (X \hat{dJ})^T X \hat{dJ} - \hat{dJ}^T X^T y = 0$$

$$\rightarrow \alpha = \frac{(X \hat{dJ})^T [X\hat{\theta} - y]}{(X \hat{dJ})^T (X \hat{dJ})}$$

It just remains to find $\widehat{dJ}$:

$$\frac{\partial J}{\partial \theta_k} = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})x_k^{(i)}$$

$$= \frac{1}{m}\left[(X^TX)\theta_0 - X^Ty\right]_k$$

$$dJ = \frac{1}{m}\left[X^TX\theta - X^Ty\right]$$

$$\rightarrow \alpha = \frac{\frac{1}{m}\left[XX^TX\hat{\theta} - XX^Ty\right]^T\left[X\hat{\theta} - y\right]}{\frac{1}{m^2}\left[XX^TX\hat{\theta} - XX^Ty\right]^T\left[XX^TX\hat{\theta} - XX^Ty\right]}$$

$$= m\,\frac{\left[X\hat{\theta} - y\right]^T XX^T\left[X\hat{\theta} - y\right]}{\left[X\hat{\theta} - y\right]^T XX^TXX^T\left[X\hat{\theta} - y\right]}$$

Let $M \equiv XX^T$ & $G(\theta) \equiv X\theta - y$

$$\rightarrow \alpha = m\frac{G(\theta)^T M\, G(\theta)}{G(\theta)^T M^2 G(\theta)}$$

Gradient Descent: repeat until convergence {

$$\theta := \theta - \frac{G(\theta)^T M G(\theta)}{G(\theta)^T M^2 G(\theta)} X^T G(\theta)$$

}

Gradient Descent for Linear Regression.

Note that M only has to be computed once!

Lecture 25: Normal Equation Noninvertibility

Could it be that $A \equiv X^T X$ is non-invertible?

Yes. This could have two reasons:

1. Redundant features

Say we have 2 feature x_1 & x_2 such that $x_2^{(i)} = \alpha x_1^{(i)}$. Then: $A_{11} = \sum_{i=1}^{m} (x_1^{(i)})^2$

$A_{12} = \alpha A_{11}$ & $A_{22} = \alpha^2 A_{11}$

Also assume we're not fitting for θ_0. Then

$$A = \sum_{i=1}^{m} (x_1^{(i)})^2 \begin{bmatrix} 1 & \alpha \\ \alpha & \alpha^2 \end{bmatrix} \text{ & } \det A = 0.$$

2. Too many features (i.e $m \leq n$)

This means that we're trying to fit for too many parameters given a small training set. Of course, this would cause degeneracies which translate into saying that A is not invertible.

* Lectures 26 - 32 are about using Octave.