Week 2 Lecture Notes

[ML: Linear Regression With Multiple Variables]

Linear regression with multiple variables is also known as "multivariate linear regression".

We now introduce notation for equations where we can have any number of input variables.

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

$x^{(i)}$ = the ~~column~~ column vector of all the feature inputs of the i^{th} training example

m = the number of training examples

$n = |x^{(i)}|$; (the number of features)

Now define the multivariable form of the hypothesis function as follows, accommodating these multiple features:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

In order to develop intuition about this function, we can think about θ_0 as the basic price of a house, θ_1 as the price per square meter, θ_2 as the price per floor, etc. x_1 will be the number of square meters in the house, x_2 the number of floors, etc.

Using the definition of matrix multiplication, our ~~multiplication~~ multivariable hypothesis function can be concisely represented as:

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

This is a vectorization of our hypothesis function for one training example; see the lessons on vectorization to learn more.

Remark: Note that for convenience reasons in this course Mr. Ng assumes that $x_0^{(i)} = 1$ for $(i \in 1, \ldots, m)$

[Note: So that we can do matrix operations with theta and x, we will set $x_0^{(i)} = 1$, for all values of i. This makes the two vectors 'theta' and $x^{(i)}$ match each other element-wise (that is, have the same number of elements: n+1).]

The training examples are stored in X row-wise, like such:

$$X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} \\ x_0^{(2)} & x_1^{(2)} \\ x_0^{(3)} & x_1^{(3)} \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

We can calculate the hypothesis as a column vector of size (m x 1) with:

$$h_\theta(x) = X\theta$$

For the rest of these notes, and the lecture notes, X will represent a matrix of training examples $x^{(i)}$ stored row-wise.

Cost function:

For the parameter vector θ (of type $\mathbb{R}^{n+1}$ or in $\mathbb{R}^{(n+1)\times 1}$), the cost function is:

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

The vectorized version is:

$$J(\theta) \quad \frac{1}{2m} \left(X\theta - \vec{y} \right)^T \left(X\theta - \vec{y} \right)$$

Where $\vec{y}$ denotes the vector of all y values.

- Gradient Descent for Multiple Variables:

The gradient Descent equation itself is generally the same form; We just have to repeat it for our 'n' features:

Repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_1(x^{(i)}) - y^{(i)}\right) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_2(x^{(i)}) - y^{(i)}\right) \cdot x_2^{(i)}$$

...

}

In other words:

Repeat until convergence: {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x_j^{(i)} \quad \text{for } j := 0, ...$$

}

Matrix Notation

The gradient Descent rule can be expressed as:

$$\theta := \theta - \alpha \nabla J(\theta)$$

Where $\nabla J(\theta)$ is a column vector of the form:

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} & \frac{\partial J(\theta)}{\partial \theta_1} & \cdots & \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix}$$

The j-th component of the gradient is the summation of the product of two terms:

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

$$= \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} \cdot \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

Somewhere, the summation of the product of two terms can be represented as the product of two vectors.

Here, $x_j^{(i)}$, for $i = 1, \dots m$, represents the m elements of the j-th column, $\vec{x}_j$, of the training set X.

The other term $\left(h_\theta(x^{(i)}) - y^{(i)} \right)$ is the vector of the ~~deviations~~ deviations between the ~~pas~~ predictions $h_\theta(x^{(i)})$ and the true values $y^{(i)}$.

Re-Writing $\frac{\partial J(\theta)}{\partial \theta_j}$, we have:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \vec{x}_j^{\,T} (X\theta - \vec{y})$$

$$\nabla J(\theta) = \frac{1}{m} X^T (X\theta - \vec{y})$$

Finally, the matrix notation (vectorized) of the Gradient Descent rule is:

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

Feature normalization:

We can speed up gradient descent by having each of our input values in roughly the same range. This is because θ will descend quickly on small ranges and slowly on large ranges, and is inefficient if θ descends slowly down to the optimum when the variables are very unevenly designed.

The way to prevent this is to modify the ranges of our input variables so that they are all roughly the same. Ideally:

$$-1 \leq x_{(i)} \leq 1$$

or

$$-0.5 \leq x_{(i)} \leq 0.5$$

These aren't hard requirements; we are really trying to speed up things. The goal is to get all input variables into roughly one of these ranges, give or take a few.

Two techniques to help with this are feature scaling and mean normalization. Feature scaling involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variables, resulting in a new range of just 1. Mean normalization involves subtracting the average value for an input variable, resulting in a new average value for the input variable of just zero. To implement both of these techniques, adjust your input values as shown in this formula:

$$x_i := \frac{x_i - \mu_i}{s_i}$$

Where μ_i is the average of all the values of feature (i) and s_i is the range of values (max - min), or s_i is the standard deviation.

Note that dividing by the range, or dividing by the standard deviation,

give different results. The gradient in their cases was large. The Programming exercises was standard deviation.

Example: x_i is housing prices with range of 100 to 2000, with a mean value of 1000. Then, $x_i := \frac{\text{Price} - 1000}{1900}$.

- Gradient Descent Tips:

Debugging gradient descent. Make a plot with number of iterations on the x-axis. Now plot the cost function, $J(\theta)$ over the number of iterations of gradient descent. If $J(\theta)$ ~~over~~ increases then you probably have to decrease α.

Automatic Convergence Test. Declare convergence if $J(\theta)$ decreases by less than E in one iteration, where E is some small value such as 10^{-3}.

It has been proven that if learning rate α is sufficiently small then $J(\theta)$ will decrease on every iteration. Andrew Ng recommends decreasing α by multiples of 3.

- Features and Polynomial Regression:

We can improve our features and the form of our hypothesis function in a couple different ways.

We can <u>combine</u> multiple features into one. For example, we can combine X_1 and X_2 into a new feature X_3 by taking $X_1 \cdot X_2$.

Polynomial Regression

Our hypothesis function need not be linear (a straight line) if that does not fit the data well.

We can change the behavior or curve of our hypothesis function by making it a quadratic, cubic or square root function (or any other form).

For example, if our hypothesis function is $h_\theta(x) = \theta_0 + \theta_1 x_1$ then we can create additional features based on X_1, to get the quadratic function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$ or the cubic function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$

In the cubic version, we have created new features X_2 and X_3 where $X_2 = x_1^2$ and $X_3 = x_1^3$.

To make it square root function, we could do: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$

In that at 2:52 and through 6:22 in the "Features and Polynomial Regression" video, the curve that Prof Andrew Ng discusses about "don't we curve come back down" is in reference to the hypothesis function that uses the $\text{sqrt}()$ function (shown by the solid purple line), and not the one that uses size^2 (shown with the dotted blue line). The quadratic form of the hypothesis function would have the shape shown with the blue dotted line if θ_2 was negative.

One important thing to keep in mind is, if you choose your features this way then feature scaling becomes very important.

eg. If x_1 has range 1 - 1000 Then range of x_1^2 becomes 1 - 1000... and That of x_1^3 becomes 1 - 1000...].

Normal Equation:

The "normal Equation" is a method of finding the optimum θ for θ without iteration.

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

There is no need to do feature scaling with the normal equation.

A mathematical proof of the Normal equation requires knowledge of linear ~~Regression~~ algebra and is fairly involved. So you do not need to worry about the details.

The following is a comparison of gradient descent and the normal equation:

Gradient Descent	Normal Equation
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
$O(kn^2)$	$O(n^3)$, need to calculate inverse of $X^T X$
Works well when n is large	Slow if n is very large

With the normal equation, computing the inversion has complexity $O(n^3)$. So if we have very large number of features, the normal equation will be slow. In practice, when n exceeds 10,000 it might be a good time to go from a normal solution to an iterative process.

Normal Equation Noninvertibility

When implementing the normal equation in octave we want to use the 'pinv' function rather than 'inv'.

$X^T X$ may be noninvertible. The common causes are:

* Redundant features, where two features are highly correlated (i.e. They are linearly dependent)

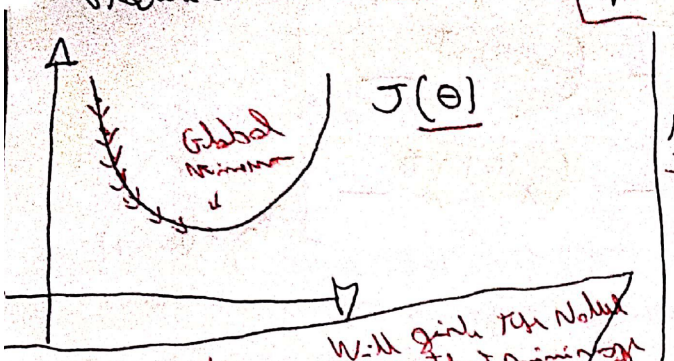
* Too many features (e.g. $m \leq n$). In this case, delete some features or use "regularization" (to be explained in a later lesson).

Solutions to the above problems include deleting a feature that is linearly dependent with another or deleting one or more features when there are too many features.

Idea: Normal Equation:

Gradient Descent

Note



Will give the Value of θ that minimizes $J(\theta)$ (for that function)

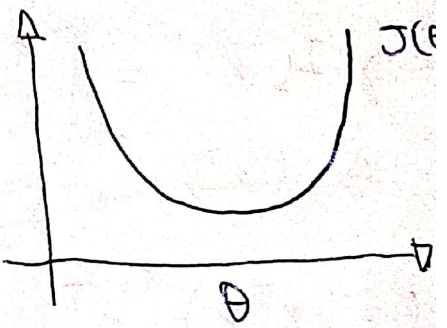
Normal equation: Method to solve for θ analytically

Intuition: If 1D ($\theta \in \mathbb{R}$)

$J(\theta) = a\theta^2 + b\theta + c$

$\frac{d}{d\theta} J(\theta) = \dots \overset{set}{=} 0$

Solve for θ



Ex. $m = 4$

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

$\theta \in \mathbb{R}^{n+1}$ $J(\theta_0, \theta_1 \dots \theta_n) =$

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

Partial derivative $\rightarrow \frac{\partial}{\partial \theta_j} J(\theta) = \dots \overset{set}{=} 0$ (for every j)

Solve for $\theta_0, \theta_1 \dots, \theta_n$.

Ex. $m = 4$.

Size (ft²)	N° bedrooms	N° floors	Age (years)	Price ($1000)
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

x_0

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \quad m \times (n+1)$$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \quad m\text{-dimensional vector}$$

All the features

m examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$, n features.

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

X (design matrix) $= \begin{bmatrix} - (x^{(1)})^T - \\ - (x^{(2)})^T - \\ \vdots \\ - (x^{(m)})^T - \end{bmatrix}$ $m \times (n+1)$

E.g. If $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

X (matrix) $= \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ \vdots & \\ 1 & x_1^{(m)} \end{bmatrix}$ $m \times 2$

y (vector) $= \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$

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

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

↳ is inverse of matrix $X^T X$

Set $A = X^T X$

$(X^T X)^{-1} = A^{-1}$

Octave: `pinv(X'*X)*X'*y`

$\theta = (X^T X)^{-1} X^T y$ $\min_\theta J(\theta)$

Feature Scaling

$-1 \le x_i \le 1$

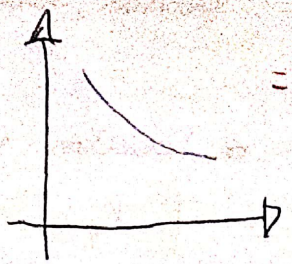
$0 \le x_i \le 100$

$0 \le x_i \le 10^{-5}$

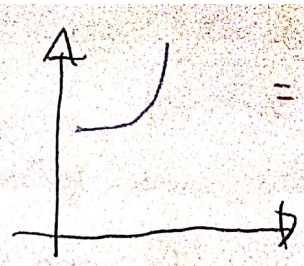
Gradient Descent X (Vs) Normal Equation

= Week 2 Lecture Note, Comparison

$= \alpha = 0.1$



$= \alpha = 0.01$



$= \alpha = 1$

$\alpha = 0.01$

$\alpha = 0.1$

$\alpha = 1$

In ~~graph~~ graph C, the cost function is increasing as the ~~for~~ learning rate is set too high. Both graphs A and B converge to an optimum of the cost function, but the graph B ~~has~~ is very slowly, as its learning rate is set too ~~low~~ small.

Graph A is better than the two.

Summarily:

- If α is too small: slow convergence.
- If α " " large: $J(\theta)$ may not decrease on every iteration; may not converge.

To choose α, try:

$\ldots, 0.001, \quad 0.003, \quad 0.01, \quad 0.03, \quad 0.1, \quad 0.3, \quad 1, \ldots$

$\ldots \quad \times 3 \quad \times 3 \quad \times 3 \quad \times 3 \quad \times 3 \ldots$

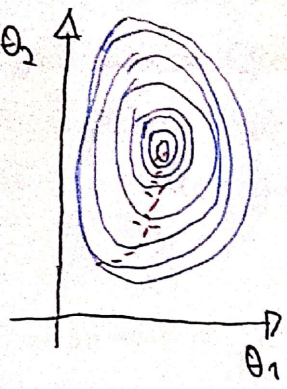
Video: Gradient Descent in Practice I - Feature Scaling!

- Feature Scaling

Idea: Make sure features are on a similar scale.

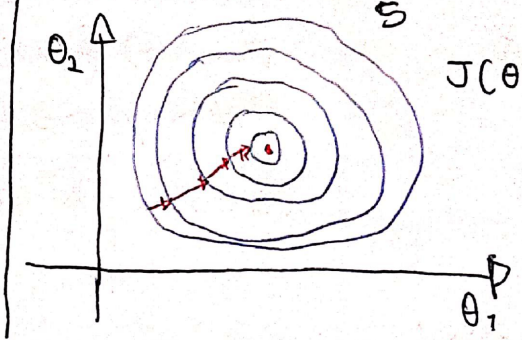
E.g. x_1 = size (0-2000 ft^2)

x_2 = number of bedrooms (1-5)



$x_1 = \frac{\text{size } (ft^2)}{2000}$

$x_2 = \frac{\text{number of bedrooms}}{5}$



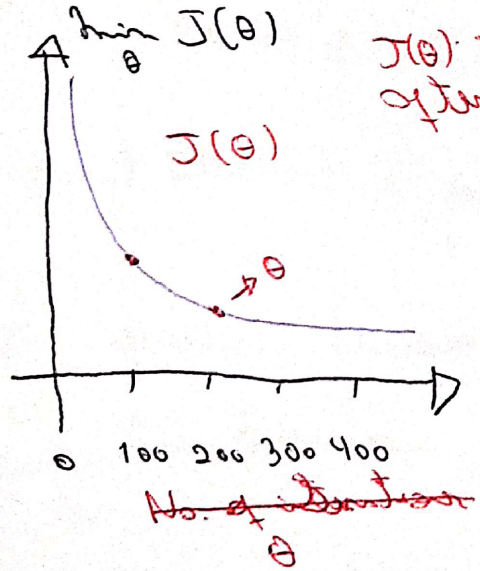
$0 \le x_1 \le 1$ $\quad 0 \le x_j \le 1$

Feature Scaling

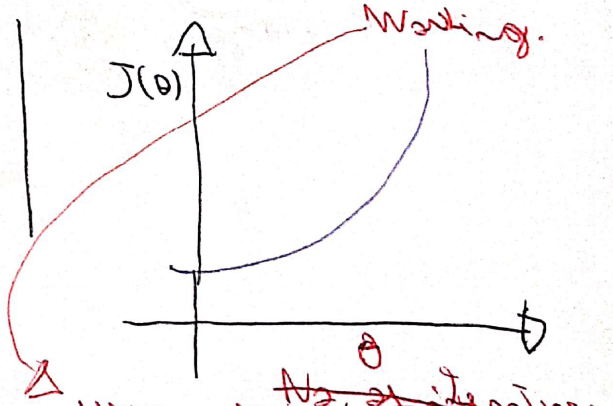
Get every feature into approximately a $-1 < x_i \le 1$ range.

G.D. in Practice II - Learning Rate: Video

Making sure gradient descent is working correctly.

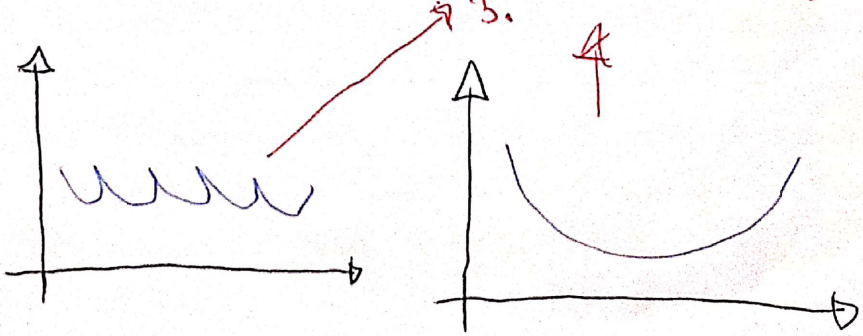


$J(\theta)$ should decrease after every iteration.



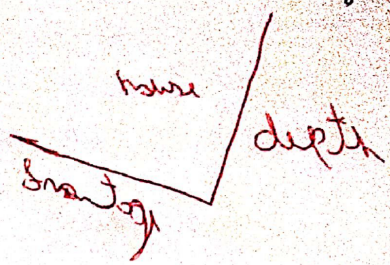
Use smaller α.

decreasing α by multi.



...ing price Prediction: Video → Features and Polynomial Regression

$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} + \theta_2 \times \underbrace{\text{depth}}_{x_2}$

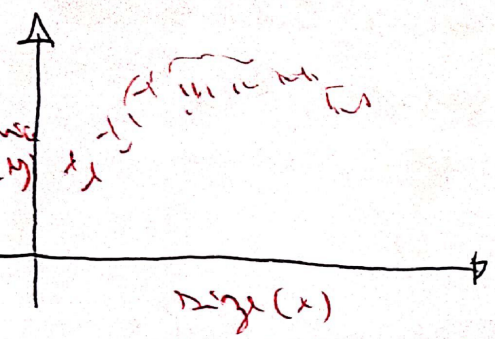


Area

$x = \text{frontage} * \text{depth}$

$h_\theta(x) = \theta_0 + \theta_1 x$ ← Land Area

Polynomial Regression



$\theta_0 + \theta_1 x + \theta_2 x^2$

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

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

$= \theta_0 + \theta_1 (\text{size}) + \theta_2 (\text{size})^2 + \theta_3 (\text{size})^3$

$x_1 = (\text{size})$

$x_2 = (\text{size})^2$

$x_3 = (\text{size})^3$

size: 1 - 1000

size^2: 1 - 1000,000

size^3: 1 - 10^9

...sion of features:

→ $h_\theta(x) = \theta_0 + \theta_1 (\text{size}) + \theta_2 \sqrt{(\text{size})}$

Normal equation

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

- What if $X^T X$ is non-invertible? (Singular/ degenerate)
- Octave: pinv(x' * x) * x' * y

θ

[pinv
 inv

- Redundant features (linearly dependent).

  Eg x_1 = size in $feet^2$ $\quad n = 3.28$ feet

  x_2 = size in m^2

  $x_1 = (3.28)^2 x_2$

- Too many features (e.g. $m \leq n$). $\quad n = 10$, $n = 100$

  - Delete some features, or use regularization

Quiz

1.

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

x_1 = hidden layer

x_2 = (hidden layer)^2

$h_\theta = \theta_0 + \theta_1 \cdot 89 + \theta_2 1921 \; 4$

x_1 94 − 69 = 25

x_2 = 8836 − 4161 = 4675

y = 90 − 74 = 22

$x_1^{(1)} = ?$

$x_i := \frac{x_i - \mu_i}{s_i}$

Scan — / 25

The order of x_1 in ~~66155~~ 19.15

and the range 94 − 69 = 25

$x_1^{(1)} = \frac{(89 - \sout{19.15})}{25} = 0.31$

	x_1	x_2
~~81~~	89	1921
~~19.15~~ ÷	12	5184
	94	8836
	64	4161

÷ 4 = 66155

2.

$\alpha = 0.3$