Stanford University

Machine Learning

Personal Report

Presented:	January 29, 2017
Author:	Trung C. Nguyen
Prof.:	Andrew Ng.

Contents

1 Single variable linear Regression

1.1 Notation

Notation	**Meaning**
m	number of training example
x's	"input" variable/ "input"features
y's	"output" variable/ "target" variable
(x,y)	one training example
$(x^{(i)}, y^{(i)})$	training example i^{th}

Table 1.1: Notation table applied for whole course

Give an example: We have a training set of housing prices (Portland, OR)

Size in $feet^2$ (x)	**Price($) in 1000's (y)**
2140	460
1416	232
1534	315
852	178
...	...

$x^{(1)} = 2014$; $x^{(2)} = 1416$

1.2 Model Representation

To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function $h : X \to Y$ so that h(x) is a good predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis. Seen pictorially, the process is therefore like figure 1.1:

Another word, h is a function that maps from x's to y's When the target variable

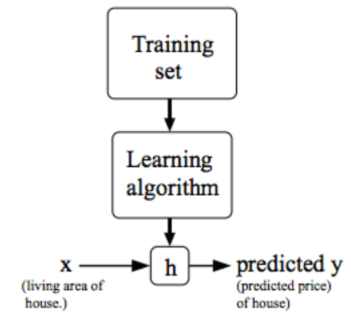


Figure 1.1: Process of machine learning

that were trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a classification problem.

If h is a function of one variable we called the model is **Univariate linear regression**

1.3 Cost function

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 \cdot x$
$\theta_i' s$: Parameters of the model
Cost function measures the accuracy of our hypothesis function. It is a function of parameters

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

This function is otherwise called the "Squared error function", or "Mean squared error". The mean is halved ($\frac{1}{2}$) as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the $\frac{1}{2}$ term. Now problem turns out of choosing value of θ_0 and θ_1. With each combination (θ_0, θ_1), we have a hypothesis (h) function.

Explain:
For each combination of (θ_0, θ_1), we have a value for hypothesis function. To find a closest $h(x)$ to actual data set means we have to find a combination of (θ_0, θ_1), such as $(h(x) - y)$ is minimum.

Lets consider whole training set, we have to find a combination of (θ_0, θ_1), such as the average of all difference: $\frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$ is minimum. Square the difference to cover the case hypothesis value is smaller than actual one.

If we call the difference above is cost function, the problem becomes find combination of (θ_0, θ_1) such as the cost function is smallest =¿ Minimization problem

1.3.1 Contour plot

A contour plot is a graph that contains many contour lines. A contour line of a two variable function has a constant value at all points of the same line. An example of such a graph is the one to the right below. Taking any color and going along the 'circle', one would expect to get the same value of the cost function. For example, the three green points found on the green line above have the same value for $J(\theta_0, \theta_1)$ and as a result, they are found along the same line.

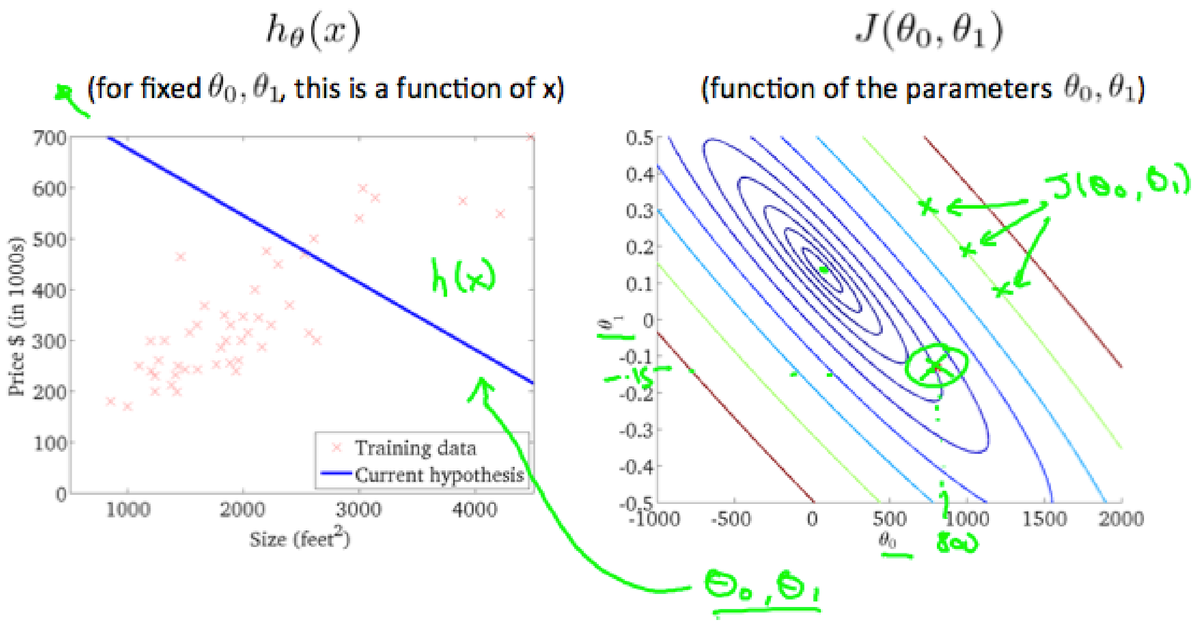


Figure 1.2: Contour plot

2 Algorithm gradient descent

Here is the gradient descent algorithm:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \tag{2.1}$$

where j=0,1 represents the feature index number.

Some important notes:

- One should simultaneously update the parameters $\theta_1, \theta_2, ..., \theta_n$. Updating a specific parameter prior to calculating another one on the $j^{(th)}$ iteration would yield to a wrong implementation
- Taking the derivative (the tangential line to a function) of our cost function $J(\theta_0, \theta_1)$. The slope of the tangent is the derivative at that point and it will give us a direction to move towards
- The size of each step is determined by the parameter α, which is called the learning rate. A smaller α would result in a smaller step and a larger α results in a larger step
- We make steps down the cost function in the direction with the steepest descent
- Depending on where one starts on the graph, one could end up at different points.

3 Multivariate linear regression

3.1 Multiple features - multiple variables

When an output does not depend only on ONE variable (input) but also on multiple variables. Each variable (input) is called as a feature. **Notation**

- n = number of features (in table 3.1 is number of column = 4)
- m = number of an training example (number of row = m)
- $x^{(i)}$ = input (features) of i^{th} training example
- x_j^i = value of feature j in i^{th} training example

Example: Let's say housing price depends on 4 features as shown in Tab. 3.1

Size	**# of bedrooms**	**# of floors**	**Age of home**	**Price**
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
123	1	3	32	232
3211	4	4	32	432
...	...	...	...	...

Table 3.1: An example of multiple features

Saying: $x^{(1)} = \begin{bmatrix} 2014 \\ 5 \\ 1 \\ 45 \\ 460 \end{bmatrix} \in \Re^n$

$x^{(1)}$ is the 1^{st} training set, it includes all inputs (features) of the 1^{st} training set.

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

3.2 Hypothesis

The hypothesis for multiple variables or multiple feature will be expressed as:

$$h_\theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... + \theta_n x_n$$

Lets define $x_0 = 1$ or ($x_0^{(i)} = 1$, which means all training example of feature 0 equal to 1. Hence,

$$h_\theta = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... + \theta_n x_n \tag{3.1}$$

In addition, substitue:

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \Re^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n+1} \end{bmatrix} \in \Re^{n} \tag{3.2}$$

Finally, **multivariate linear regression** is expressed as

$$\begin{aligned} h_\theta &= \theta^\intercal X \\ h_\theta &= [\theta_0, \theta_1, \theta_2, \cdots, \theta_{n+1}] \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \end{aligned} \tag{3.3}$$

where:

- X is a vector of inputs or features
- $\theta^\intercal$ is a transpose of the column matrix θ

3.3 Gradient Descent for multiple variables

3.3.1 Cost function

Renote: The **cost function** is a function of θ
The previous version for single variable:

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

where:

- $\hat{y}^{(i)}$ is the hypothesis value of output corresponding an input in the i^{th} training example
- $y^{(i)}$ is the actual output in the i^{th} training example
- $x^{(i)}$ is the input in the i^{th} training example
- θ_0, θ_1 are coefficients or parameters

The same function is applied for multivariate linear regression. But, we have multiple variables
=> MUST have multiple coefficients θ

The new version:

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

repeat until convergence: {

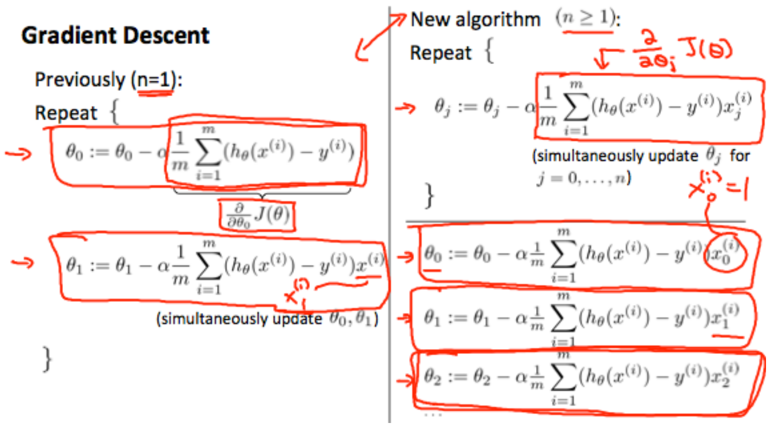
$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

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

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

...

}



Gradient Descent

Previously (n=1):

Repeat {

$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial\theta_0}J(\theta)}$

$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$ $x_1^{(i)}$

(simultaneously update θ_0, θ_1)

}

New algorithm $(n \geq 1)$:

Repeat {

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

}

$x_0^{(i)} = 1$

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

$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}$

$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}$

...

Figure 3.1: Difference between Univariate and Multivariate version

3.4 Notes to improve runtime of Gradient Descent

3.4.1 Scaling feature

Scaling features is a MUST before running algorithm because of the fact:

- The longer range of a feature, the slower the cost function converges

Putting all features in a same scale helps reducing runtimes of the algorithms.

What a contour look like if using feature with different scales?

- The thinner side of contour is the θ of the longer-range feature
- The wider side of contour is the θ of the shorter-range feature

Or we can say: *a longer range feature has a shorter range of* θ
Scaling feature makes contour more likes a circles

Example:
The housing price depends on 2 features: the area and the number of floors. Lets say

- Area varies from 50 m^2 - 200 m^2

- Number of floors varies from 1 - 5

Scaling feature means instead of using a feature with a range (50 - 200) and a feature with a range (1 - 5) as inputs of gradient descent algorithm. We uses a scaling feature with a range ($\frac{50}{200}$ - $\frac{200}{200}$) or (0.25 - 1) and a scaling feature with a range (0.2 - 1).

Each feature can have different scaling factors

RULES for scaled range

- $[-3, +3]$ is OK, any larger: BAD
- $\left[-\frac{1}{3}, +\frac{1}{3}\right]$ is OK, any smaller: BAD

3.4.2 Mean normalization

Aim of normalizing the mean is to bring the mean of "feature's range" close to zero.
Take a feature x_j and replace it by:

$$\frac{x_j - \mu_j}{\sigma_j}$$

where:

- μ_j is mean or average of values of feature j^{th}
- σ_j is the standard deviation of feature j^{th}, it is calculated as (max - min)

Example:
Let say size is a feature that affects the housing price. In training examples, its range is from 100 - 180. The mean of size feature is 140 and the standard deviation is 80.
The feature that we use in hypothesis is $\frac{x_j - 140}{80}$

3.4.3 Learning Rate α

This section explains how we choose a suitable α

Tool for checking the Gradient Descent working or not α
Using plot of $\min(J(\theta))$ vs No. of iteration

PRINCIPAL: J(θ) decreases after every iteration

- IF: decreasing slowly, then need to increase α
- IF: non-decreasing, then need to reduce α

Typically, a the process for alpha selection

- Try a range of alpha values
- Plot J(θ) vs number of iterations for each version of alpha
- Go for roughly threefold increases: 0.001, 0.003, 0.01, 0.03. 0.1, 0.3

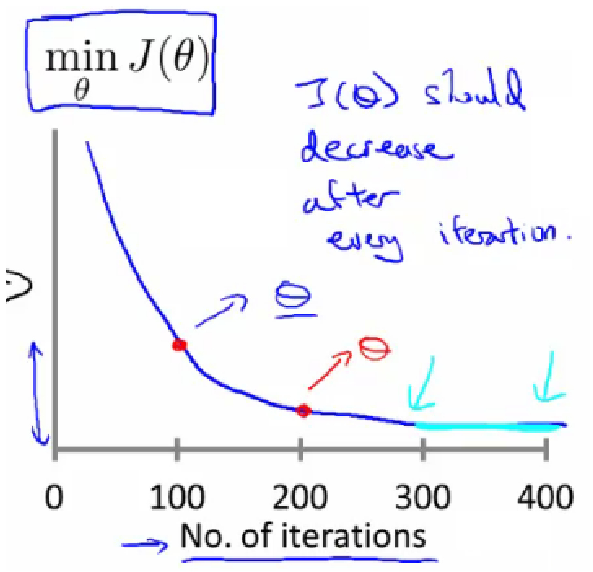


Figure 3.2: Plot for checking Gradient Descent

3.5 Features vs Polynomial Regression

- DO NOT need to use all features (since some features are correlated), there are algorithms to choose suitable features
- Sometimes, form of actual data (such as: price vs size) does not look like a linear funtion, we have to use POLYNOMIAL REGRESSION

3.5.1 Features

We can **create a new feature**

Example
In house price prediction, Two features:

- Frontage (x_1) - width of the plot of land along road
- Depth (x_2)- depth away from road

Might decide that an important feature is the land area.
So, create a new feature (may be a better indicator), area (x_3) = frontage * depth
Now, the new model is:

$$h(\theta) = \theta_0 + \theta_1 x_3$$

3.5.2 Polynomial Regression

How do we create the model to this data?
Having a set of:

$$x_1 = x$$
$$x_2 = x^2$$
$$x_3 = x^3$$

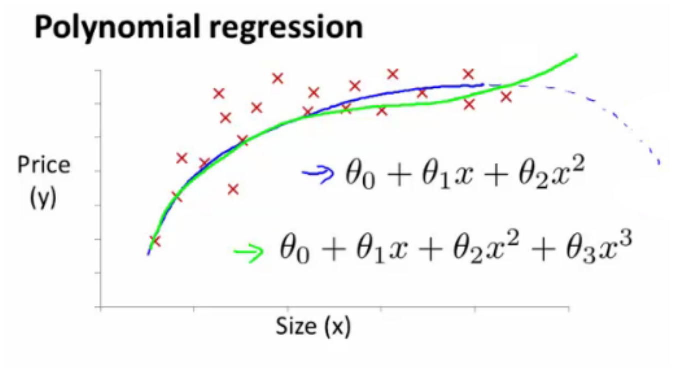


Figure 3.3: Choosing a polynomial to express actual relationship

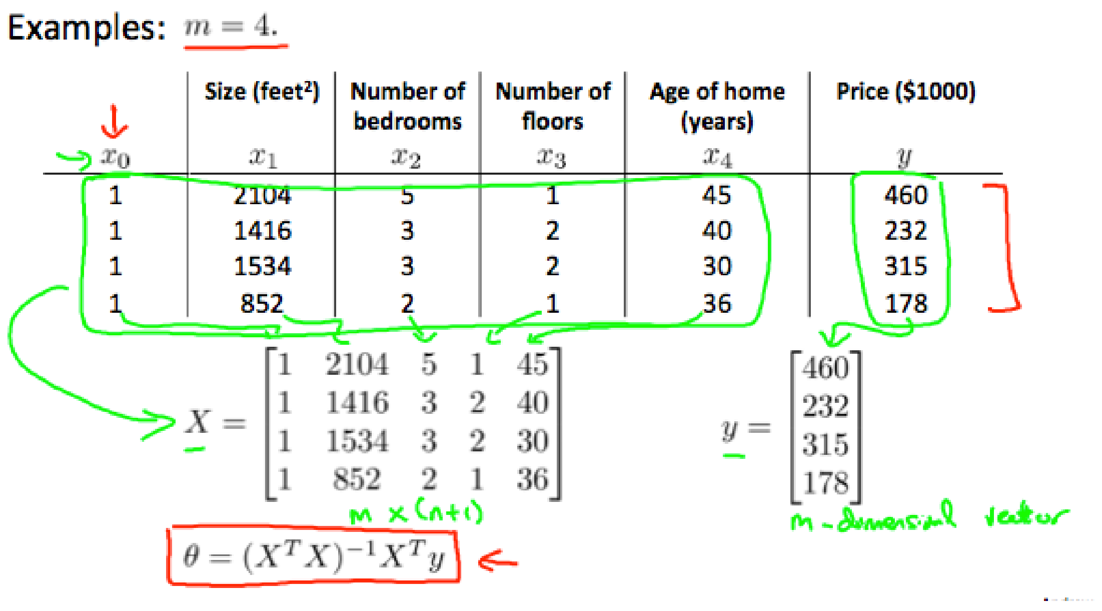
By selecting the features like this and applying the linear regression algorithms you can do polynomial regression.
Remember, feature scaling becomes even more important here.

4 NEW ALGORITHM: Analytically method to compute theta

4.1 Normal equation

Formula:

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



Examples: $m = 4.$

	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

m × (n+1)

m-dimensional vector

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

Gradient Descent: Finding value of θ for getting minimal value of the cost function by perform multiple iterations.
Normal equation: we will minimize J by explicitly taking its derivatives with respect to the $\theta^{(j)}$, and setting them to zero. This allows us to find the optimum

theta without iteration.

There is no need to do feature scaling with the normal equation
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^TX
Works well when n is large	Slow if n is very large

n: number of features
To be specific, n > 10000 => should go with an iterative approach

4.2 Normal Equation Noninvertibility

As be described in Eq. 4.1; we need to do inverse operation of $(X^TX)^{-1}$. But in some cases, inverting is impossible.
Lets find out what are the causes of this problem:

- Redundant features, where two features are very closely related (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).

NOTE:
In octave, using the 'pinv' function rather than 'inv.' Since the 'pinv' function will give you a value of θ even if X^TX is not invertible.

5 Practical part

5.1 Octave and Matlab

Lets check how gitignore works

6 Appendix

List of Figures

List of Tables

List of Listings

References