Week 1

August 6, 2021

1. Model Representation

August 5, 2021

1 Notation

- $x^{(i)}$ - denotes the input variable
- $y^{(i)}$ - denotes the output
- $(x^{(i)}, y^{(i)})$ - denotes a training example
- $(x^{(i)}, y^{(i)}); i = 1, ..., m$ - denotes a training set
- X - the space of input values
- Y - the space of output values

2 Mathematical Definition of Supervised Learning

Given a training set to learn a function $h : X \to Y$ so that $h(x)$ is a good predictor of the corresponding y

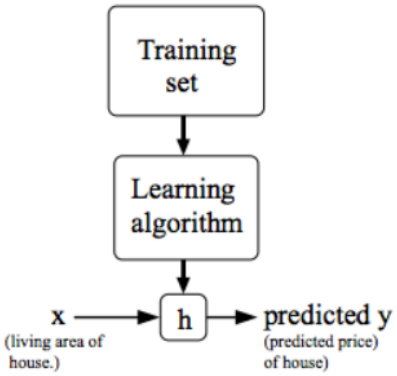


Figure 1: Graphical illustration of the mathematical representation of supervised learning - a training set goes through a learning algorithm which produces a function able to translate input of x to an output of y

The function h is called the hypothesis

When the target (i.e. the predicted y value)we are trying to predict is continuous the learning problem is known as a regression problem, when the target takes on a discrete value it is called a classification problem

2. Cost Function

August 5, 2021

1 Hypothesis

The hypothesis function will generally be linear (for simplicity) and thus takes on the following form $h_\theta(x) = \theta_0 + \theta_1 \times x$ where θ is know as a parameter.

The objective in defining a hypothesis function is to find values for θ so that the straight line fits the data well, where the data represents the inputs and outputs the function is suppose to predict.

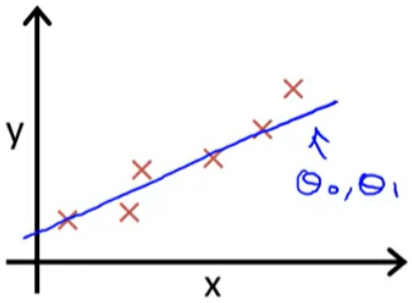


Figure 1: Linear function that approximately fits the input y against the output y

To find values for the parameters that is a good fit to the data, we choose values so that $h_\theta(x)$ is close to y for our training examples (x, y), this boils down to a minimization problem (to minimize the error between the hypothesis function and the training examples)

2 Cost Function

The accuracy of the the hypothesis function can be measured using a cost function

The cost function takes the average difference of all the results of the hypothesis given the input of x's and the expected outputs (y's from the training

set). Put in mathematical terms

$$\begin{aligned} J(\theta_0, \theta_1) &= \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 \\ &= \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x) - y_i)^2 \end{aligned} \tag{1}$$

The square is taken to obtain positive differences, the division by m is in line with obtaining the average of something and the halving of the mean is for convenience when finding the gradient descent of the cost function where the derivative of $(h_\theta(x) - y_i)^2$ cancels with $1_{\overline{2-gradientdescentdiscussionstilltofollow.}}$

3 Cost Function Intuition 1

Notice that from the previous section that our hypothesis function $h_\theta(x)$ is a function of x for fixed values of the parameters θ_0 and θ_1

Whereas our cost function $J(\theta_0, \theta_1)$ is a function of our parameters θ_0 and θ_1

Consider an example. Given our hypothesis function $h_\theta(x) = \theta_0 + \theta_1 x$ has $\theta_0 = 0$ for simplicity, and it only fits 3 values $(1,1), (2,2), (3,3)$ our cost function for different values of θ_1 gives us:

$$\begin{aligned} J(\theta_0, \theta_1) &= \frac{1}{2\times 3}\sum_{i=1}^{3}((x^i) - y^i)^2 \\ &= \frac{1}{6}[(1-1)^2 + (2-2)^2 + (3-3)^2] \\ &= \frac{1}{6} \times 0 \\ &= 0 \end{aligned} \tag{2}$$

for $\theta_1 = 1$

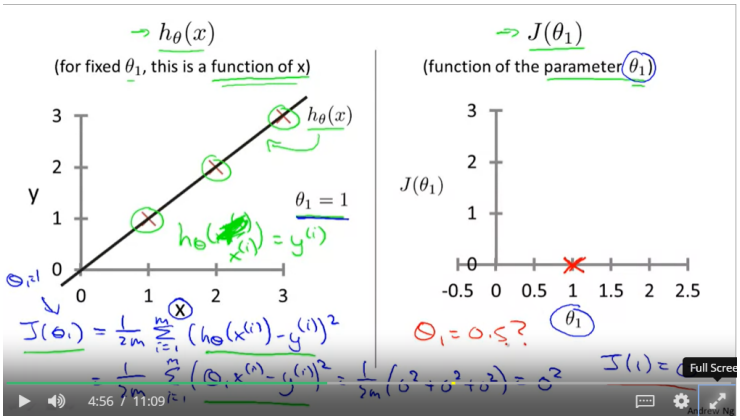


Figure 2: Cost is 0 with $\theta_1 = 1$

setting $\theta_1 = 0.5$

$$\begin{aligned} J(\theta_0, \theta_1) &= \frac{1}{2\times 3}\sum_{i=1}^{3}((x^i) - y^i)^2 \\ &= \frac{1}{6}[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2] \\ &= \frac{1}{6} \times 3.5 \\ &= 0.53 \end{aligned} \tag{3}$$

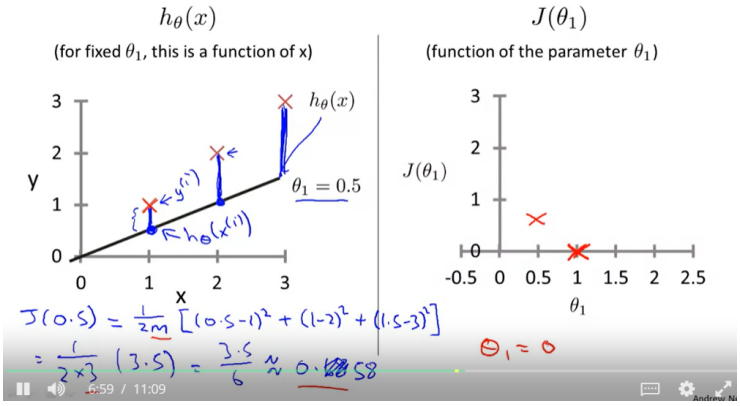


Figure 3: Cost is 0 with $\theta_1 = 0.5$

setting $\theta_1 = 0$

$$\begin{aligned} J(\theta_0, \theta_1) &= \frac{1}{2\times 3}\sum_{i=1}^{3}((x^i) - y^i)^2 \\ &= \frac{1}{6}[(0-1)^2 + (0-2)^2 + (0-3)^2] \\ &= \frac{1}{6} \times 15 \\ &= 2.5 \end{aligned} \tag{4}$$

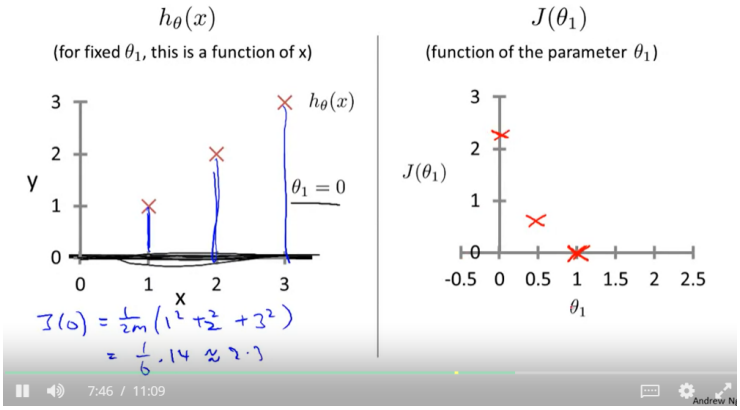


Figure 4: Cost is 0 with $\theta_1 = 0$

carrying on in this manner we can see that $\theta_1 = 1$ is the global minimum of the cost function $J(\theta_1)$

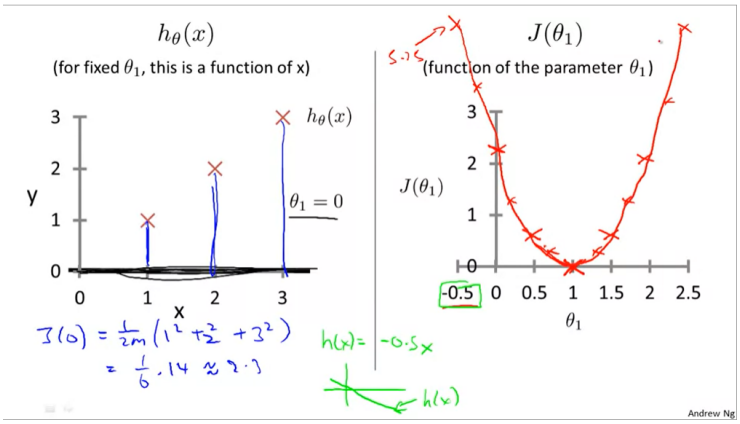


Figure 5: Cost function plot of different values for θ_1

4 Cost Function Intuition 2

The previous intuition section looked at the scenario where θ_0 was set to 0, this made our cost function 2 dimensional. Consider now that we have a non-zero θ_0 value, meaning our cost function is 3D

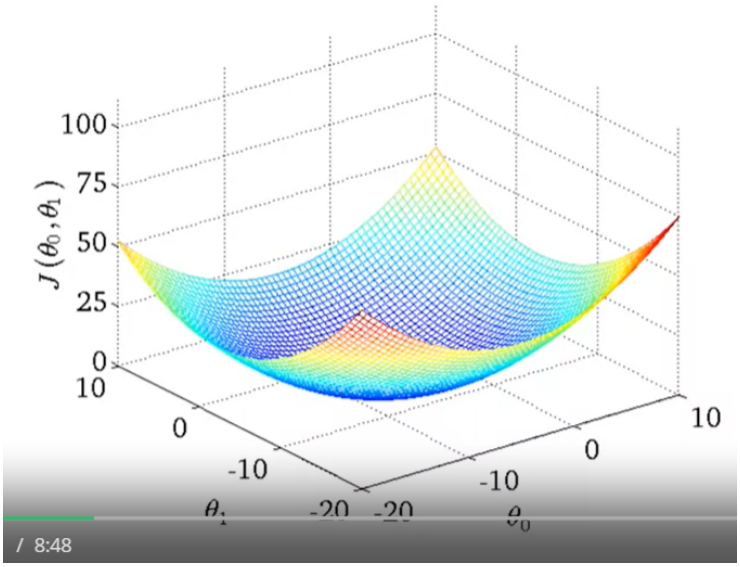
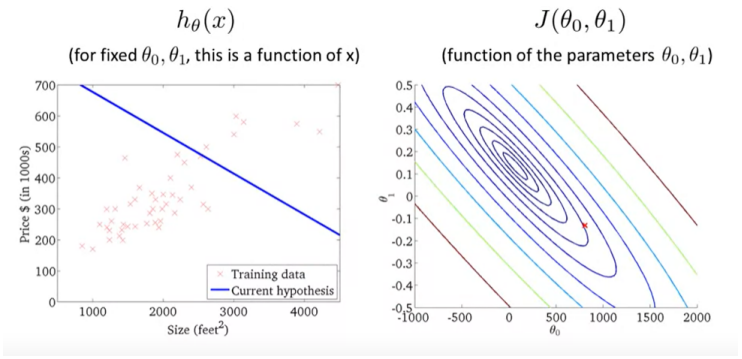
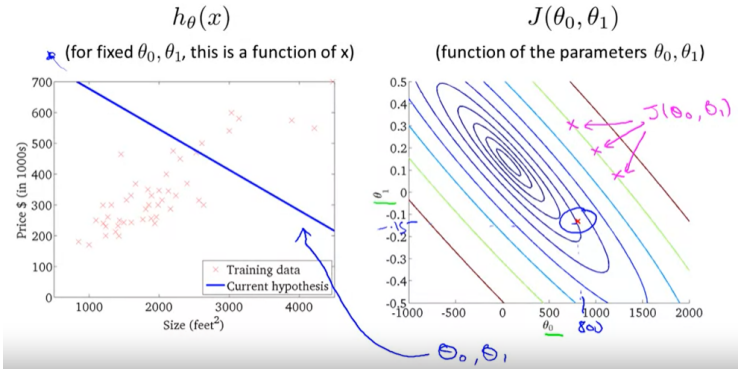
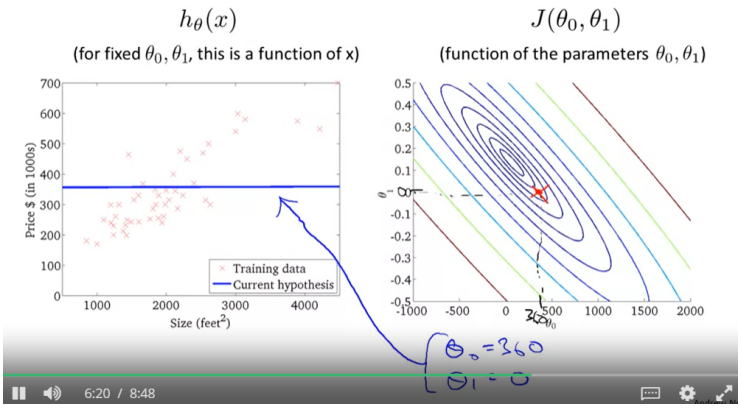


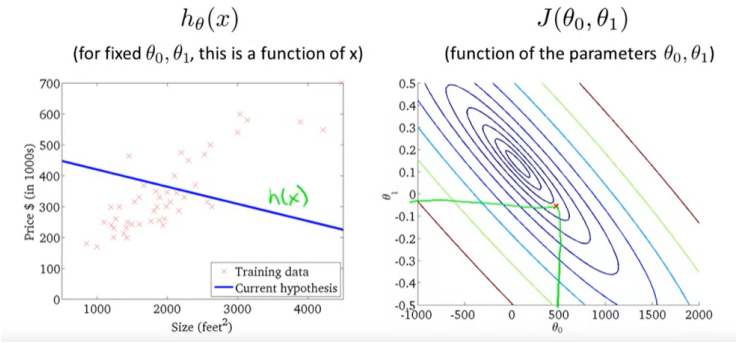
Figure 6: Illustration of the cost function with non zero theta values

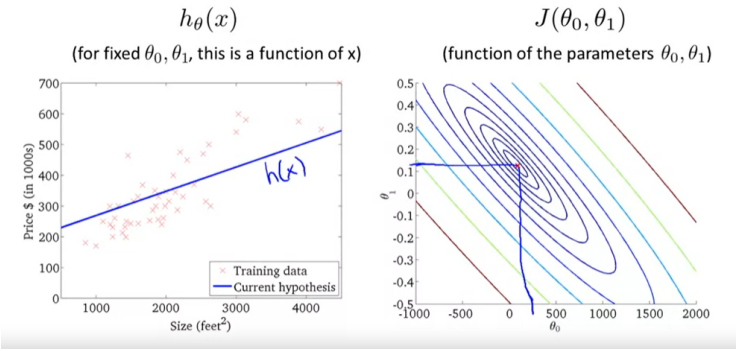
Now, illustrating the effect of different theta values and the effect it has on the hypothesis function but more importantly the cost function we will be making use of contour plots (it is a plane section of the three-dimensional graph - from wikipedia). Each contour corresponds to a different hypothesis function











Notice that we obtain a minimum when the values for θ_0 and θ_1 are approximately $(250, 0.15)$

5 Conclusion

The cost function is a function that tells us how far off our hypothesis function is from predicting the output y of some input x.

The idea with the 2 intuition sections was to show that there exists values for θ_0 and θ_1 defining the hypothesis function such that the cost function is minimized (i.e. the predicted value y given x is close to the real value). Our task is to find these parameter values. We are also not restricted to only 2 parameters but to multiple.

3. Gradient Descent

August 5, 2021

1 What is Gradient Descent

Gradient descent is an algorithm used to minimize a cost function. Although previously we considered only linear regression problems, the gradient descent algorithm can be applied to other functions as well.

The algorithm works like this (again considering only 2 parameter values):

1. Start with an initial guess of the parameters
2. Keep changing the parameter values with a small change (or step) to reduce the cost function $J(\theta_0, \theta_1)$ until we arrive at a global minimum

We can depict the above steps visually

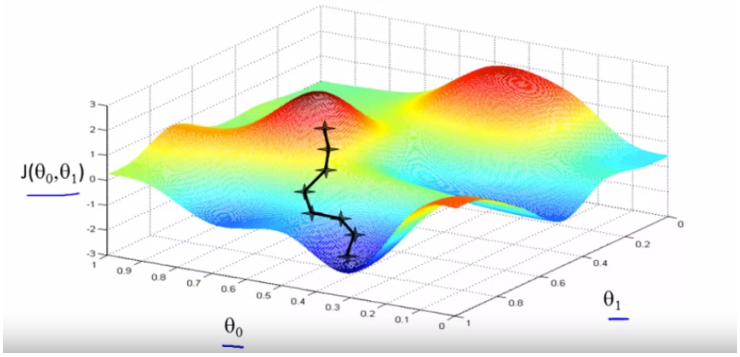


Figure 1: We start at some arbitrary values for θ_0 and θ_1, change our guess repeatedly until the cost function yields the smallest value it can possibly take on

Our initial value for (θ_0, θ_1) is not always going to guarantee we arrive at the global minimum. In the figure below, the initial guess of the parameters lead us to find a local minima

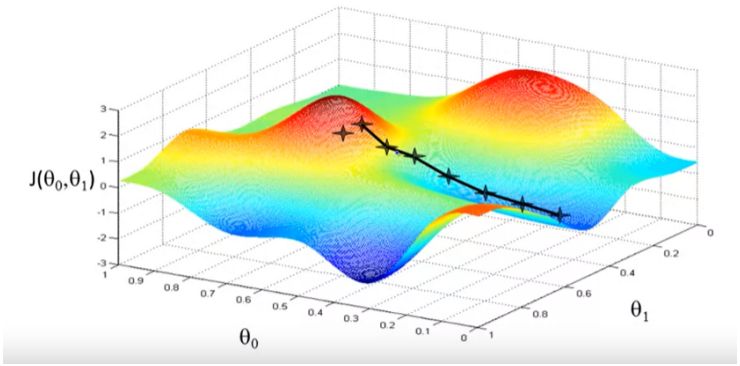


Figure 2: Inital guess of parameters lead us to find a local minima rather than the desired global minimum

The property of the gradient descent algorithm to lead to different minimas based on the initial guess of the parameters is a property we will explore in section 2.1

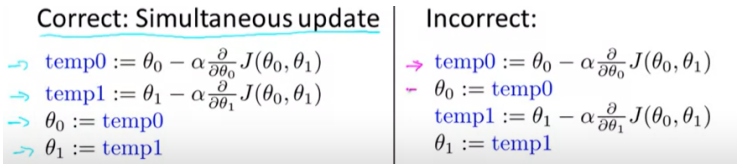
The mathematical definition of the gradient descent algorithm is given below

$$\theta_j \quad := \quad \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1); (j = 0, j = 1) \tag{1}$$

repeated until convergence
where:

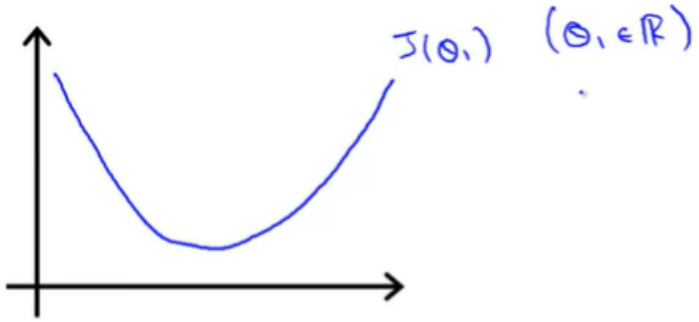
1. := is the assignment operator
2. α is called the learning rate and controls how big a step we take to find the minimum (larger α means biger steps, conversely, smaller α means we take smaller steps)
3. The derivative of the cost function will be explained later

Keep in mind that the mathematical definition of the gradient descent function says to simultaneously update the parameter values

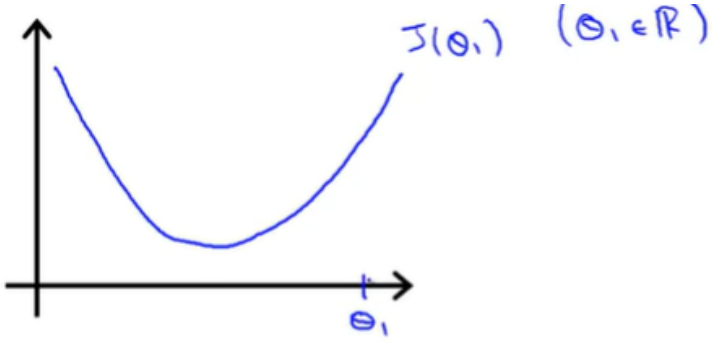


2 Gradient Descent Intuition 1

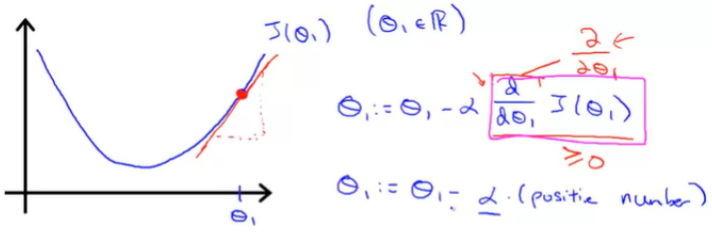
Consider a cost function that contains only 1 parameter θ_1



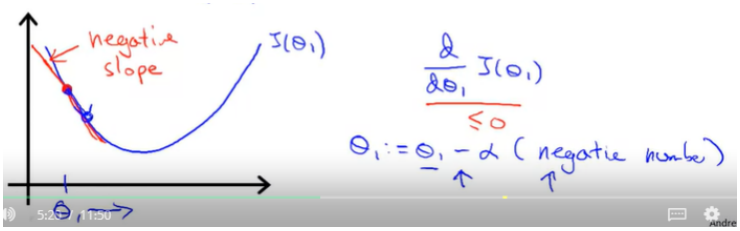
Now imagine we have an initial value of θ_1 as shown in the figure below



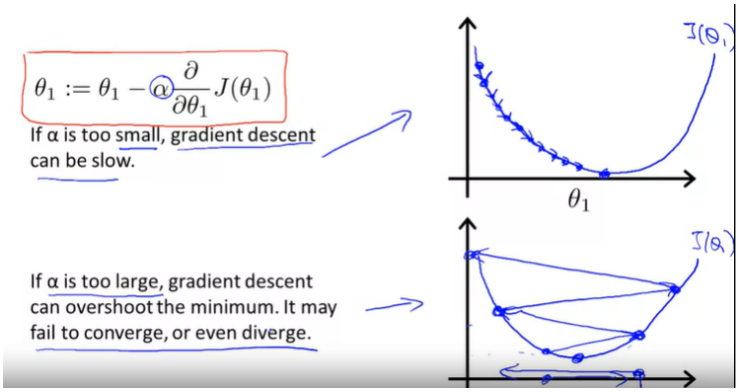
For the 1 variable cost function, the gradient descent algorithm becomes a simple derivative. Finding the derivative at the initial point will yield us the slope/gradient of a tangent to the point on the graph (yields a positive value). Thus the new value for θ_1 will tend towards the minimum of the cost function.



If the initial value was to the left of the minimum the gradient would be negative and we would still be yielded a new value that tends towards the minimum of the cost function

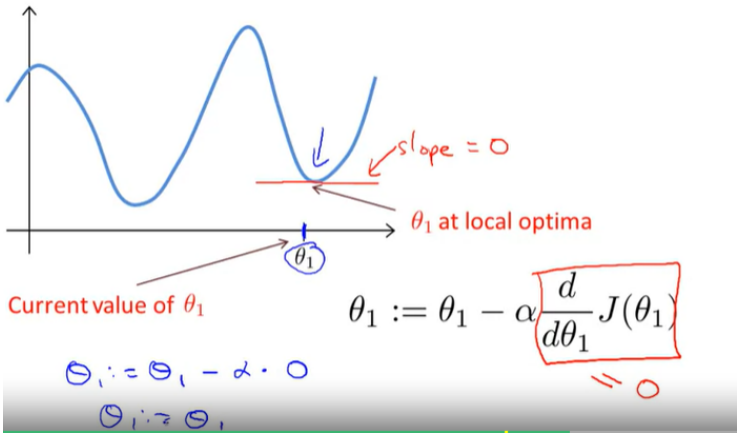


Looking at the learning rate α we can see that the smaller it is, the more changes need to be made to θ_1 to get to the minimum and the bigger it is the greater the change in θ_1 would be leading to the possibility of overshooting our minimum.



2.1 Initial Value of Parameters

Recall previously we said that an initial value of the parameters determine whether we achieve a global min or a local min. Here we explain why. Consider we set the initial value of θ_1 to a value to the right of the local minimum which is the current value of θ_1. Gradient descent would take us to the current local minimum and it would stop (it will not achieve global min). This is because the gradient at this point is 0, the gradient descent algorithm will want to keep this value as is.



If my memory serves me correctly, I believe finding the 2nd derivative of the function determines whether we are at a local min or a global min (more on this in sections to come that look at finding global min)

3 Gradient Descent for Linear Regression

Recall for linear regression the hypothesis function looks as follow $h_\theta(x) = \theta_0 + \theta_1 \times x$ and the cost function is given as $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$

Now to find gradient descent for linear regression we substitute in the hypothesis and cost function

$$\begin{aligned} \theta_j &= \theta_j - \alpha \tfrac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\ &= \theta_j - \alpha \tfrac{\partial}{\partial \theta_j} [\tfrac{1}{2m} \textstyle\sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2] \\ &= \theta_j - \tfrac{\alpha}{2m} \textstyle\sum_{i=1}^{m} \tfrac{\partial}{\partial \theta_j} (\theta_0 + \theta_1 x^i - y^i)^2 \end{aligned} \tag{2}$$

$$\begin{aligned} \therefore \theta_0 &= \theta_0 - \tfrac{\alpha}{2m} \textstyle\sum_{i=1}^{m} (2)(1)(\theta_0 + \theta_1 x^i - y^i) \\ &= \theta_0 - \tfrac{\alpha}{m} \textstyle\sum_{i=1}^{m} (\theta_0 + \theta_1 x^i - y^i) \end{aligned} \tag{3}$$

and

$$\begin{aligned} \therefore \theta_1 &= \theta_1 - \tfrac{\alpha}{2m} \textstyle\sum_{i=1}^{m} (2)(x^i)(\theta_0 + \theta_1 x^i - y^i) \\ &= \theta_1 - \tfrac{\alpha}{m} \textstyle\sum_{i=1}^{m} (\theta_0 + \theta_1 x^i - y^i) x^i \end{aligned} \tag{4}$$

And here we see the need for the linear regression cost function to have been divided by 2 as the derivatives of the cost function lead to its cancellation with a numerator 2.

3.1 Batch Gradient Descent with Linear Regression

The summing terms of equation 3 and equation 4 says that we use all the training examples in determining the parameters per gradient descent step, this gradient descent algorithm is known as batch gradient descent

There are other kinds of gradient descent algorithms that we will explore later on

Week 2

August 6, 2021

1. Multiple Features

August 6, 2021

1 Introduction

Previously, we only considered linear regression with a single feature/variable x, now however we will consider more than 1 feature that can be used to predict something. As an example we can have features such as the size of a house, the number of bedrooms, the age of the house and the number of floors to predict the price of the house

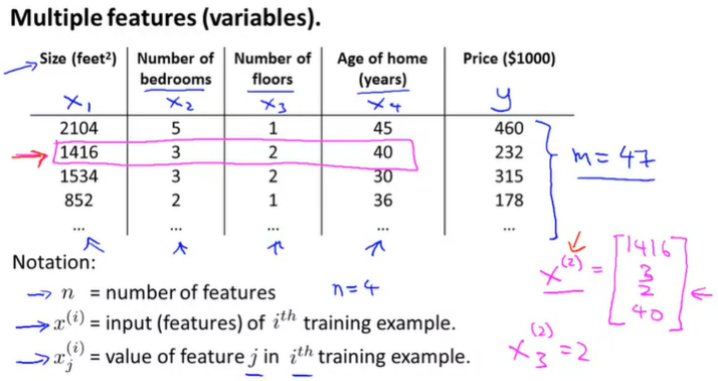


Figure 1: Notation of multiple variables

2 Hypothesis Function with Multiple Features

Our hypothesis function now changes from a function of 1 variable to a function of multiple

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{1}$$

for convenience lets define a 0 feature x_0 which will always have the value 1, then our hypothesis function becomes

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{2}$$

Lets define a feature vector X given as

$$X = \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} \in \Re^{n+1} \tag{3}$$

And a parameter vector defined as

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} \in \Re^{n+1} \tag{4}$$

Our hypothesis function can then be written in matrix form as

$$\begin{aligned} h_\theta(x) &= \Theta^T \times X \\ &= \begin{bmatrix} \theta_0 & \theta_1 & . & . & . & \theta_n \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} \end{aligned} \tag{5}$$

2. Gradient Descent for Multiple Variables

August 6, 2021

1 Cost function and Gradient Descent with Multple Features

From the previous section we derived a matrix expression of the hypothesis function for multiple features. Given $h_\theta(x) = \Theta^T \times X$ where Θ is given as $\begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}$ (the parameters of the features) and X is given as $\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix}$

Now our cost function can be given in terms of the feature vector Θ as below

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

Substituting this into the gradient descent algorithm, we get

$$\begin{aligned} \{\theta_j &= \theta_j - \alpha \times \frac{\partial}{\partial\theta_j}J(\Theta) \\ &= \theta_j - \frac{\alpha}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial\theta_j}(h_\theta(x^i) - y^i)^2 \\ &= \theta_j - \frac{\alpha}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial\theta_j}(h_\theta(x^i) - y^i)^2 \\ &= \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i; for j = 1, ..., n\} \end{aligned} \tag{2}$$

because we defined the feature x_0 in the previous section and the vector X contains it, we can arrive at the general definition of the gradient descent function defined in equation 2

3. Gradient Descent in Practice

August 6, 2021

1 Introduction

2 ways of making gradient descent more efficient in practice (to make gradient descent converge faster towards global min) is using something called feature scaling and a specific learning rate.

2 Feature Scaling

Feature scaling is the practice of making features lie within the same range of values (because by nature features wont lie in the same range, the number of rooms in a house will have a different range of values than the distance of the house from sea-level). Applying feature scaling we can be assured that gradient descent will converge faster.

2.1 Example of Feature Scaling

Consider 2 features of a house, its size in feet ^{2} (in the range of $0 - 20000$) and the number of bedrooms (in the range $1 - 5$).

In finding values for the parameters θ_1 and θ_2, the gradient descent function will converge very slowly towards the global minimum of the cost function.

The cost function for the different ranges of x_1 and x_2 will be elliptical as depicted below

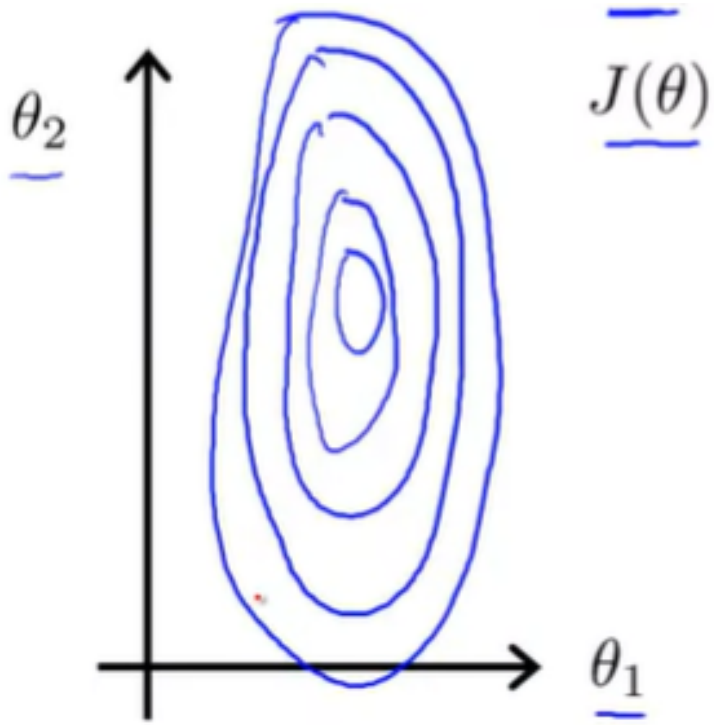


Figure 1: An elliptical shaped cost function for features that are not in the same range - with such a cost function, gradients will oscillate back and forth in search of the global min resulting in a longer time to converge.

If we apply feature scaling to the 2 variables according to the equation below, we ensure that both x_1 and x_2 lie in the range $0 \leq x_1, x_2 \leq 1$

$$\begin{aligned} x_1 &= \frac{size(feet^2)}{2000} \\ x_2 &= \frac{x_2}{5} \end{aligned} \tag{1}$$

With the feature scaling applied as above we get a cost function as shown below

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

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

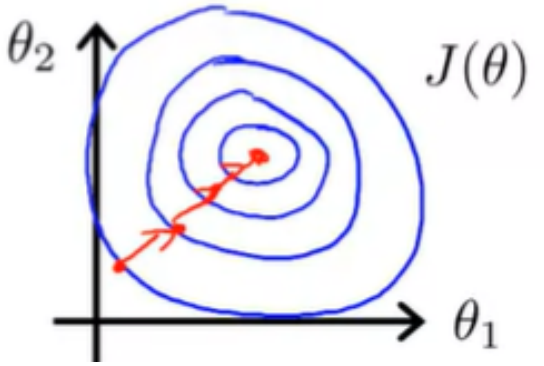


Figure 2: A circular shaped cost function arises from features that are in range

2.2 More on Feature Scaling

We are not restricted to making features lie between 0 and 1, so long as the range is not too far away from -1 and +1. Examples of valid ranges include but are not limited to:

$$\begin{aligned} -1 \leq x_1, x_2 \leq 1 \\ -2 \leq x_1, x_2 \leq 1 \\ -5 \leq x_1, x_2 \leq 5 \end{aligned} \tag{2}$$

Ranges that would constitute being too far from -1 and +1 include:

$$\begin{aligned} -100 \leq x_1, x_2 \leq 100 \\ -0.00001 \leq x_1, x_2 \leq 0.00001 \end{aligned} \tag{3}$$

2.3 Other ways of Feature Scaling

The way we applied feature scaling previously was to divide the values of x_1 and x_2 by the max value of each respective variable's previous scale. This is not the only way to scale the features however, another way is by using mean normalization which is given as

$$x \leftarrow \frac{x-\mu}{\sigma} \tag{4}$$

where x is the value of the feature to be scalled to the new scale, μ is the training set average and σ is the standard deviation or the max initial range minus the min initial range.

3 Learning Rate

We can make sure gradient descent works (converges) by plotting the cost function versus the number of iterations performed in finding the optimal parameter values.

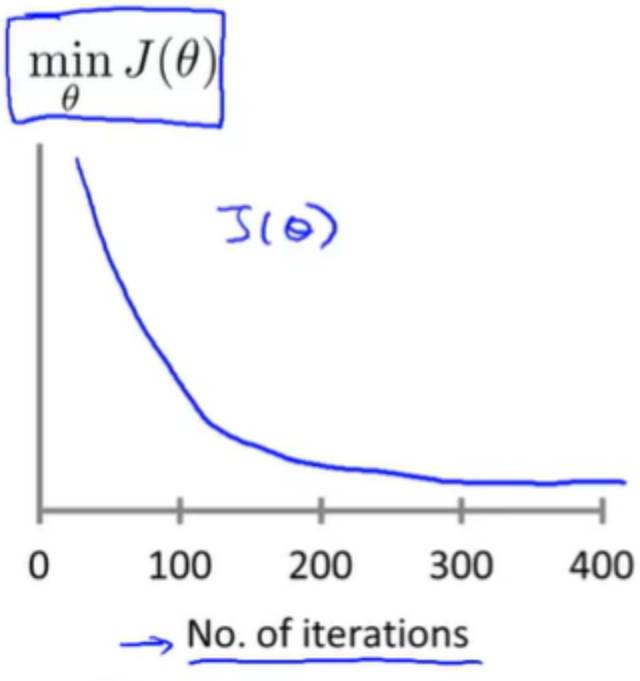
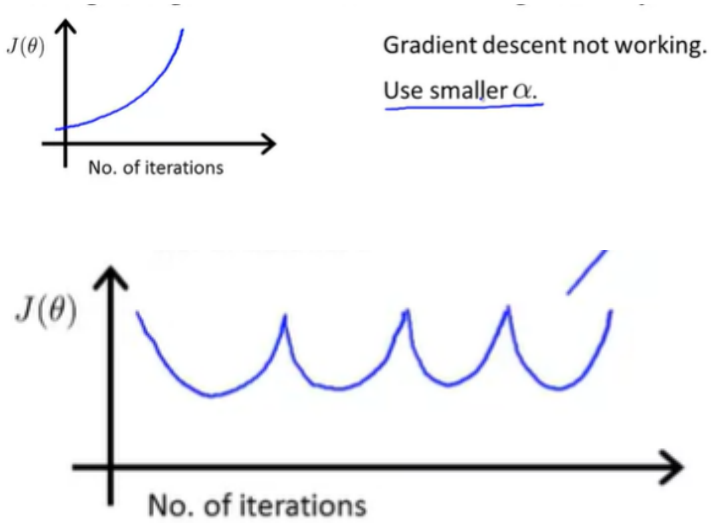


Figure 3: Plot of cost function versus the number of iterations performed in finding parameter values resulting in the smallest cost

From the plot in figure 3 we can also tell after how many iterations gradient descent converges (where the graph starts to become constant, between 300 and 400 iterations).

The graph also tells us whether our value for the learning rate is too large. For too large a learning rate we could get graphs that look like any of the below plots:



With both figures above, too large a learning rate means we are overshooting our minimum for cost.

So looking at the extreme (that α can be too large), what if α is too small ? A proof exists that shows that if α is too small we are guaranteed that $J()$ will still converge but at a slow rate.

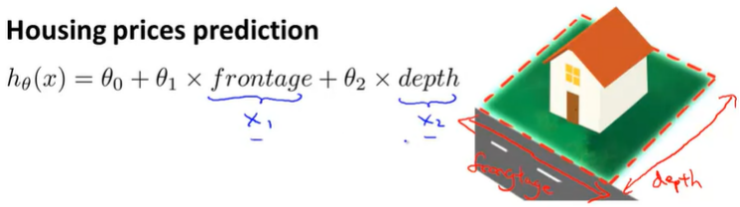
We can choose a learning rate starting at 0.0001 and increase it to 3 times the previous value to find convergence at a reasonable rate - at least this is a suggestion from Andrew Ng.

4. Polynomial Regression

August 6, 2021

1 Choosing Features

We are not restricted to what we choose as features for our hypothesis functions. Consider the example of predicting the price of a house. We can have the frontage and depth of a house as 2 features to include in our hypothesis function.



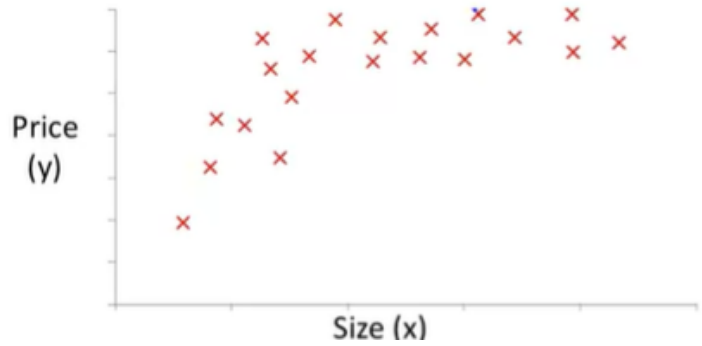
Upon learning our model to predict the price of a house we might find that our model is not as good at predicting housing prices as we hoped. We can then try and revisit the features we defined and conclude that only 1 feature (the area of the house rather than the frontage and depth) might result in a model better at predicting housing prices. This is possible.

The thought experiment above begs the question, "How will we know whether one set of features will allow our model to perform better than another ?". To answer this question we introduce Polynomial regression

2 Polynomial Regression

Our hypothesis function need not be linear!

Consider the example training set below



A number of models can be fit to this training set

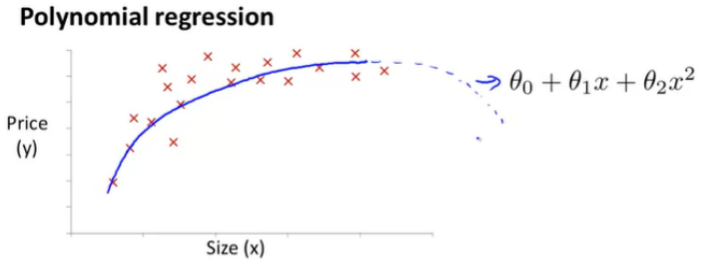


Figure 1: Quadratic model

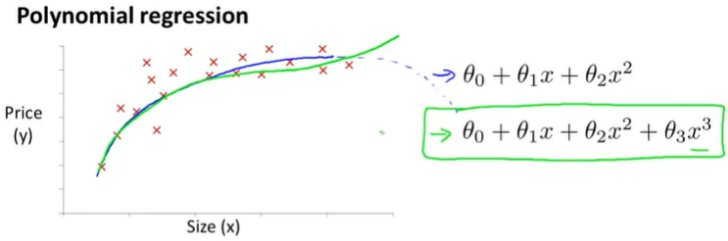


Figure 2: Cubic model

From both the above models we can see that possible "good" features to choose are the size of the house plus the square of the size of the house OR the

size of the house plus the square of the size of the house plus the cube of the size of the house.

Now we have narrowed down the possibility of 2 sets of features to choose from but which one of the 2 will be the best one for our model ? There exist algorithms to determine this.

Computing Parameters Analytically - Normal Equation

August 6, 2021

1 Introduction

So far we have been using the gradient descent algorithm to find values for the parameters. The normal equation is a better way to solve for parameters of some linear regression problems.

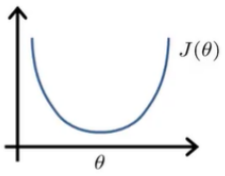
The normal equation solves for parameters analytically (in one step) as opposed to the iterative way of the gradient descent algorithm.

2 Normal Equation

Consider the parabolic cost function shown below

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

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



To minimize such a function requires that we find the gradient function and set it equal to 0 and find the value for θ.

For more complex cost functions we perform the same but we use partial derivatives to arrive at the parameter in question that will minimize the cost function

$\theta \in \mathbb{R}^{n+1}$ $\quad J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

$\frac{\partial}{\partial \theta_j} J(\theta) = \cdots = 0$ (for every j)

Solve for $\theta_0, \theta_1, \ldots, \theta_n$

The normal equation uses this principle of partial derivatives and minimizing the cost function as part of its proof, however the proof will be skipped here and only the result of the normal equation is presented.

The normal equation relies on the x_0 feature and the construction of matrices for the input and output of the training set as well as for the parameters. Consider the example training set and corresponding features below

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) 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

constructing matrices for the input and output of this set below

Examples: $m = 4$.

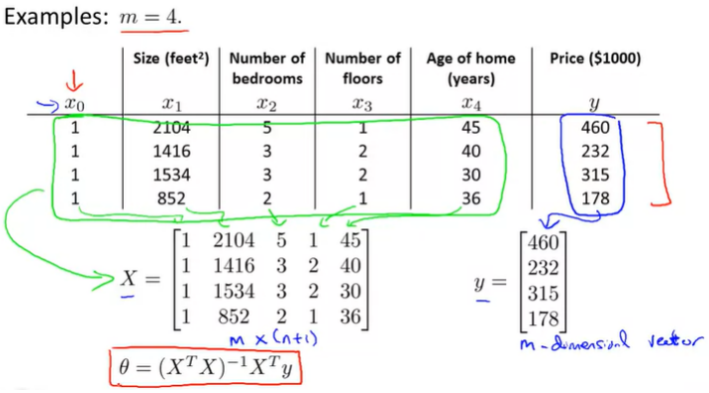
x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

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

m x (n+1)

m-dimensional vector

The normal equation used in finding the parameters resulting in a minimized cost function is given below



The beauty of using the normal equation to find the parameters of the hypothesis function is that it does not require feature scaling, don't need to define an arbitrary learning rate and it is not iterative (we arrive at our solution at once!). But of-course there are also some caveats with the normal equation. Obviously the matrices to be inverted need to be invertible (a matrix might not be invertible if there are redundant features - linear dependency, or if there are too many features - number of features are more than the number of training examples, although this can be solved via regularization or reducing the number of features but more on these in sections to come). Also using the normal equation can be slow for a large number of features (typically anything more than 10000 features from practice).

So the question now is, when to use what. Use what is more in favour of which algorithm as depicted below.