Introduction to ML and Linear Regression

Class	Machine Learning
Created	@May 5, 2020 12:44 AM
Materials	
Reviewed	☐
Type	Lecture Notes

👉 **Notes by Hemanth Kotagiri**
💾 **GitHub:** https://github.com/hemanth-kotagiri/
🏢 **LinkedIn**: https://www.linkedin.com/in/hemanth-kotagiri/

Introduction to Machine Learning and Linear Regression Algorithm

What is Machine Learning?

> **Arthur Samuel** described it as: "the field of study that gives computers the ability to learn without being explicitly programmed."

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Example: playing checkers. E = the experience of playing many games of checkers T = the task of playing checkers. P = the probability that the program will win the next game. In general, any machine learning problem can be assigned to one of two broad classifications: Supervised learning and Unsupervised learning.

- **Supervised Learning**: = This is of two types, regression problems and classification problems. When the data set is given to you and the data points mean something, and learn from them, this comes under supervised learning.
  - Formal def = Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs
- **Unsupervised learning** = This is when you are given data with no labels, no nothing of what to do, we predict certain favorable outcome of our choice. For example, grouping the news articles which are similar under one card in the Google News. Or, trying classifying your employees under some category given their performance, punctuality and other features.
  - Formal Def = Unsupervised learning is a type of self-organized Hebbian learning that helps find previously unknown patterns in data set without pre-existing labels.

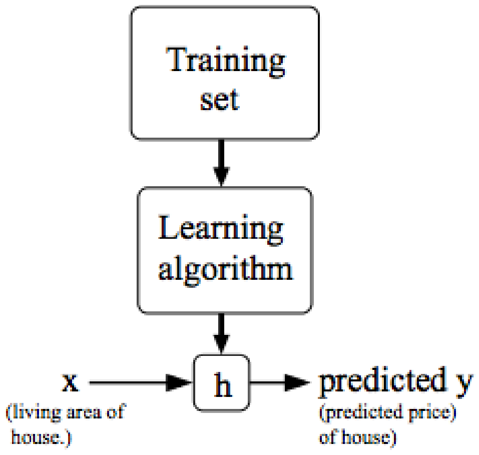
SUPERVISED LEARNING

- This is a type of machine learning model in which we have input data. This is of two types, regression and classification. So in regression type of problems we are trying to model a function to map the best fit to the given data to predict later. For example, predicting the housing prices given a data set of 100 houses sold at different prices. We model a linear function to it and use it to predict later. Where as in a classification problem, we have discrete output values of which we are predicting using the model based on the data give. For example, the classification that some one is suffering from cancer. Given a data set of malignant and benign patients, we classify the new input either anyone of the classification.

▼ MODEL AND COST FUNCTION

▼ HYPOTHESIS

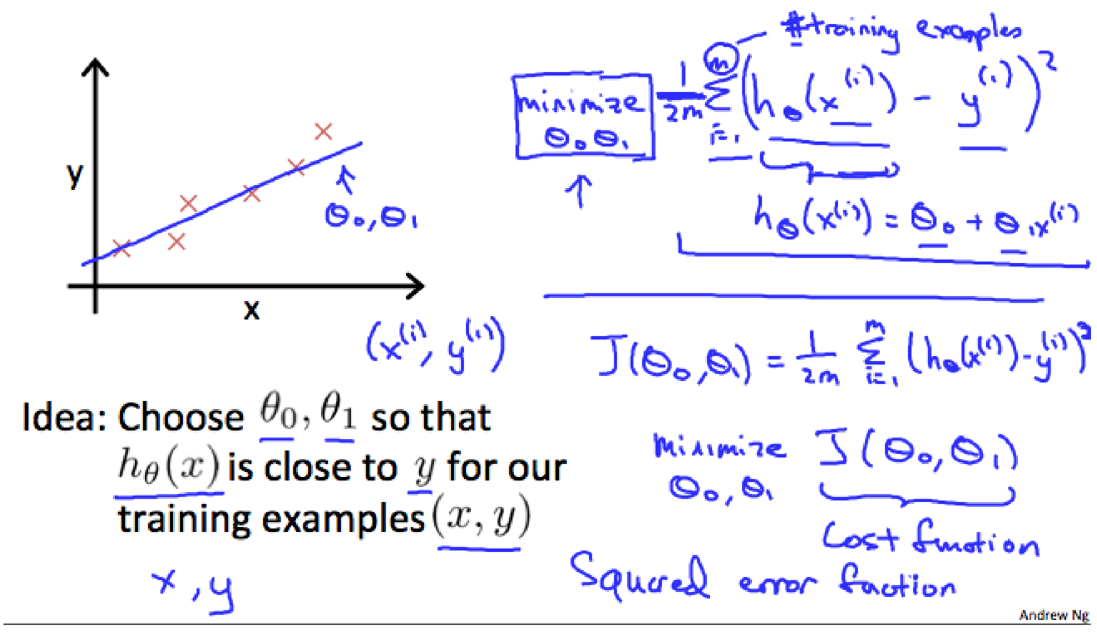
- So basically, we are given a data set which has input and output pairs. Some basic terminology related to this data set is this:
- m = total number of training examples. Let x denote a feature given in the set, and y denote the output. Then a pair (xi,yi) is called as a training example. There maybe multiple features, x1i ,x2i ,.xni which has only one output; yi. We now describe a hypothesis h(x) and we describe it so as the get a good predictor.



- The hypothesis takes in the parameters theta with which we compute the best fit line (linear regression). Hypothesis h(x) = theta0+theta1*x for 1 feature.

▼ COST FUNCTION

- A cost function is used to measure the accuracy of our hypothesis h. This takes the average results of the hypothesis subtracted with the original output values. Here is the representation of the cost function:

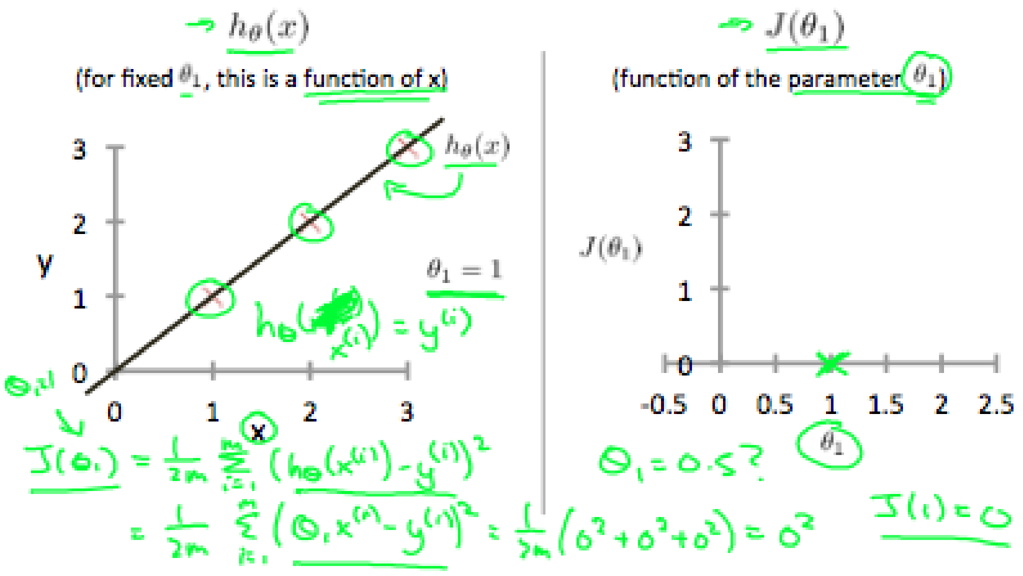


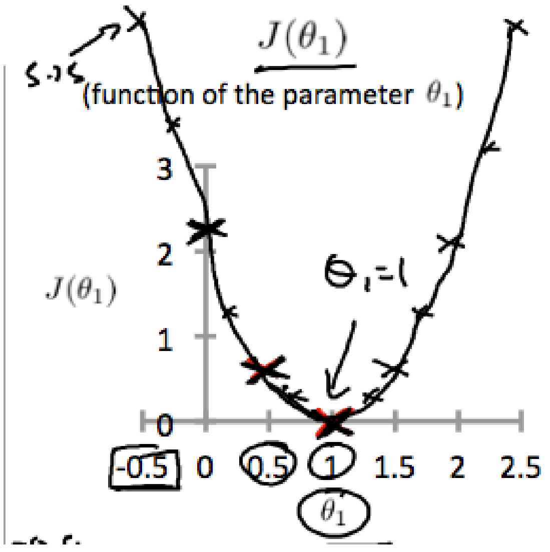
y
x
θ_0, θ_1
$(x^{(i)}, y^{(i)})$
# training examples
minimize $\theta_0 \theta_1$
$\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$
$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)
x, y
minimize θ_0, θ_1 $J(\theta_0,\theta_1)$
Cost function
Squared error function
Andrew Ng

- Our main goal in this is to **minimize the cost function** to get a very close hypothesis to the data set.

▼ COST FUNCTION INTUITION

- When we map the cost function with x-axis as the input to the hypothesis and y axis as the value of the cost function, then the resultant graph would be a parabola. As said earlier, we need to minimize the cost function.

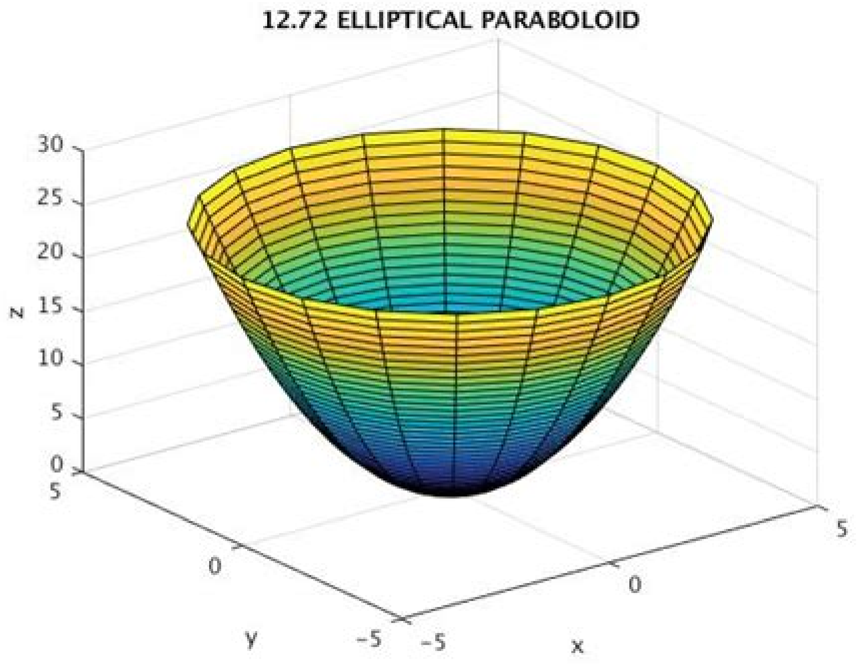




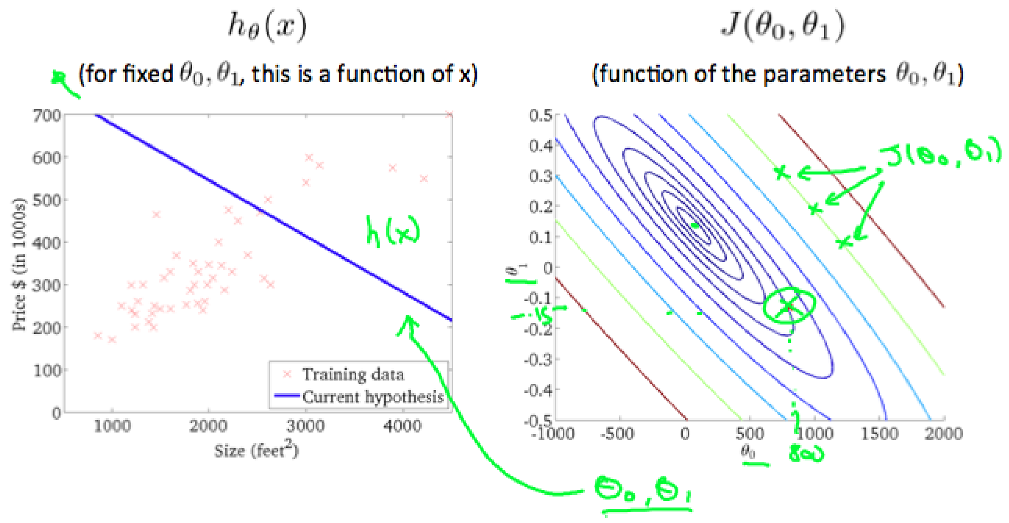
- For the data set of (0,0),(1,1),(2,2),(3,3) the theta0 = 0 and theta1 = 1 is the global minimum of the hypothesis and it fits perfectly which is not always the case.

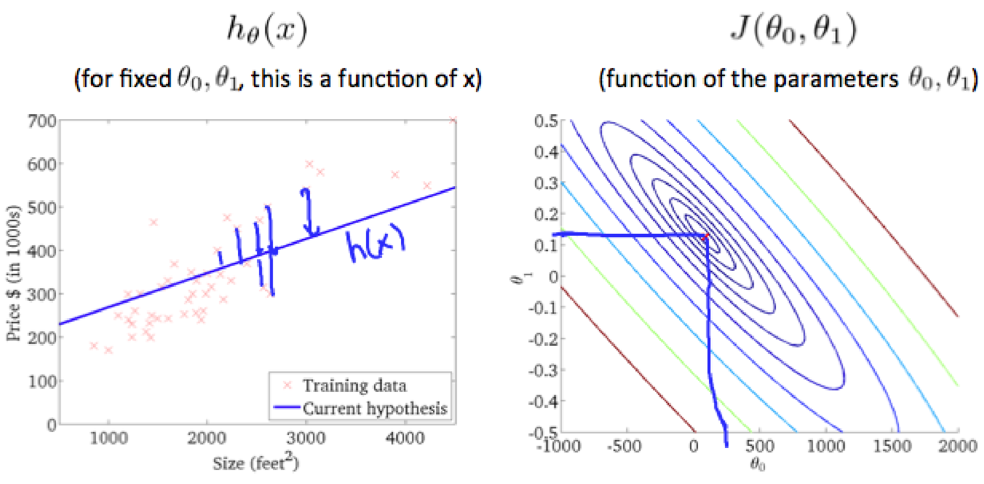
▼ **MORE COMPLICATED HYPOTHESIS FUNCTIONS**

- So, as the number of features of the data set increase, so does the complexity to visualize the graph. So, for two features, we can form a contour plot of the cost function. And for a linear regression problem, there always exists a global minimum.
- The plot of the 2 features cost function may look like this below which is a Elliptical Paraboloid.



- And the contour plot of it looks as below.





GRADIENT DESCENT: THE GATEWAY TO THE GLOBAL MINIMUM

ALGORITHM

- So, by using the cost function, the hypothesis, we have this gradient descent algorithm to slowly but surely get to the global minimum/local minimum of the given contour plot. The main aim of this is to:

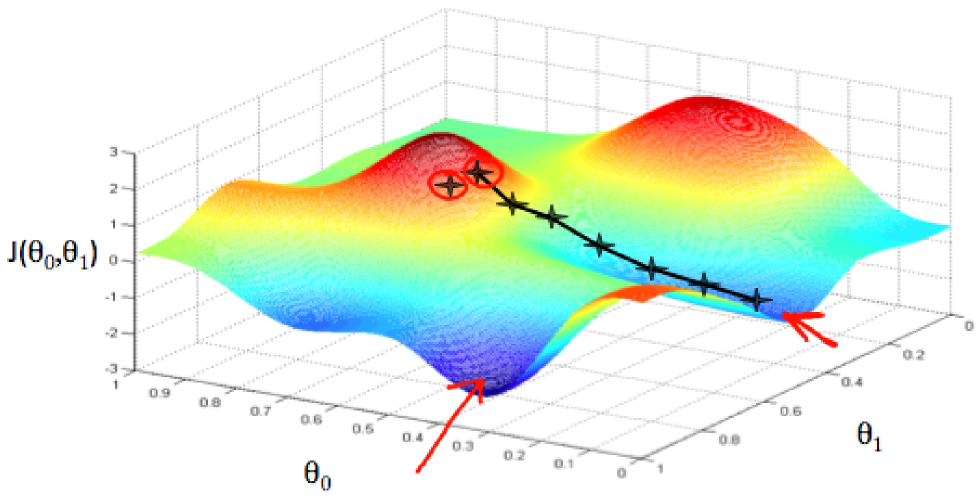
Repeat until convergence

{

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

}

- So, we have the inputs which are theta0,theta1,...thetaN. Here, we update these values according to the gradient descent. We have a learning rate alpha set to some tiny bit maybe like 0.001 which can be varied according to the data set and multiply it with the partial derivative of the cost function with respect to the thetai.(i=1,2,3... n).



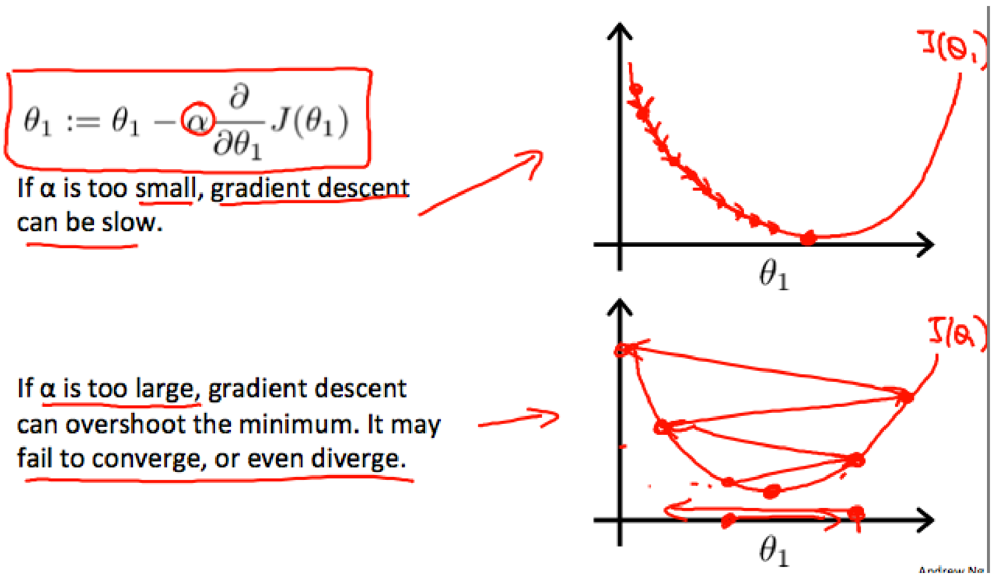
- We are taking tiny steps from the starting point as shown with the view of a direction which makes us go down the maximum.

▼ **There are some important constraints to keep in mind while doing this to achieve good gradient descent. They are:**

  - Choosing the right value of alpha. If the alpha is too small, it may take much time/many iterations to get to a global/local minimum. If the alpha value is too large, then the gradient descent may diverge and shoot off the graph.

▼ ALGORITHM INTUITION

- Taking into the consideration of having only one parameter say Theta1, and when it's cost function is plotted, we can clearly see the gradient descent in action.
- Our formula for a single parameter was to update theta 1 until convergence as stated in the algorithm above.
- There are two possible cases here
  - When the slope is negative, the value of theta 1 is eventually increased.
  - When the slope is negative, the value of theta 1 is decreased depending upon the learning rate / alpha.



$$\boxed{\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)}$$

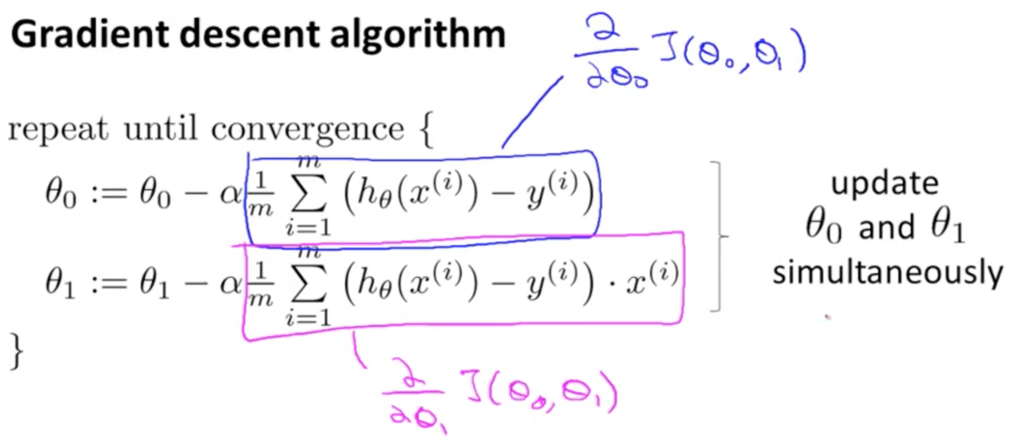
If α is too small, gradient descent can be slow.

If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

- The main point here to be noticed is that, as we get closer and closer to the global maximum, the value of the partial converges or tends to zero. Which possibly gets us the best parameter for the hypothesis that we choose.
- ⚠Gradient Descent can converge to a local minimum, even when the learning rate alpha is fixed.

▼ LINEAR REGRESSION IN ACTION

- So, the entire algorithm is based off on two main concepts here.
  1. The cost function and the hypothesis.
  2. Gradient Descent Algorithm
- So, finally applying all these to a data set, would be the linear regression in action.
- When the gradient descent finds the global/local min of the given hypothesis, we would have a closely fit theta values to predict. More the features, more the theta values. The entire gradient descent for 2 features is summarized below.



Gradient descent algorithm

repeat until convergence {

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

$$\theta_1 := \theta_1 - \alpha \underbrace{\boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\cdot x^{(i)}}}_{\frac{\partial}{\partial\theta_1} J(\theta_0,\theta_1)}$$

}

update θ_0 and θ_1 simultaneously

$$
\begin{aligned}
\frac{\partial}{\partial\theta_j} J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}\left(h_\theta(x) - y\right)^2 \\
&= 2\cdot\frac{1}{2}\left(h_\theta(x) - y\right)\cdot\frac{\partial}{\partial\theta_j}\left(h_\theta(x) - y\right) \\
&= \left(h_\theta(x) - y\right)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right) \\
&= \left(h_\theta(x) - y\right) x_j
\end{aligned}
$$