Linear Regression

Md. Mohsin Uddin

East West University

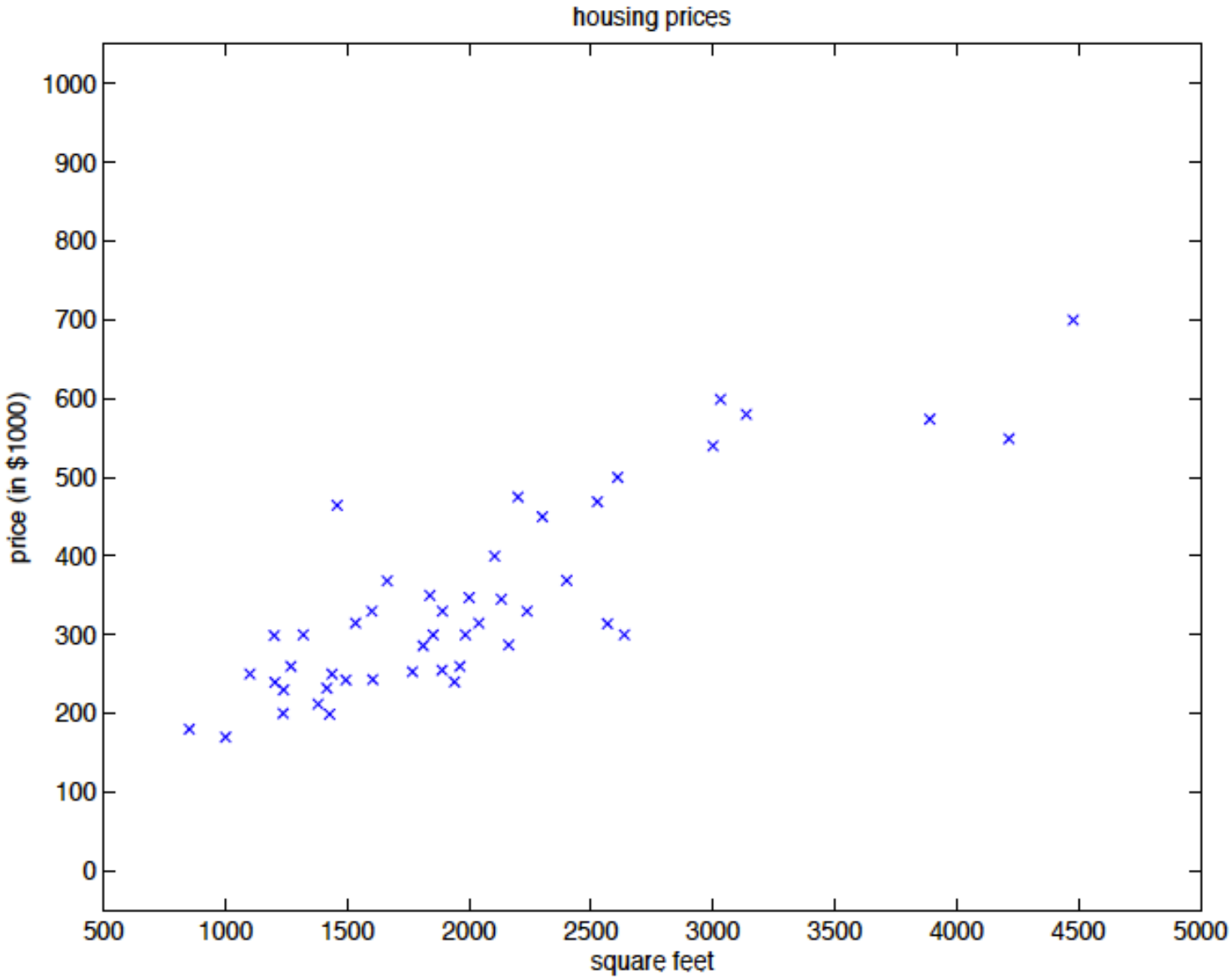
mmuddin@ewubd.edu

May 19, 2019

Supervised Learning (An Example)

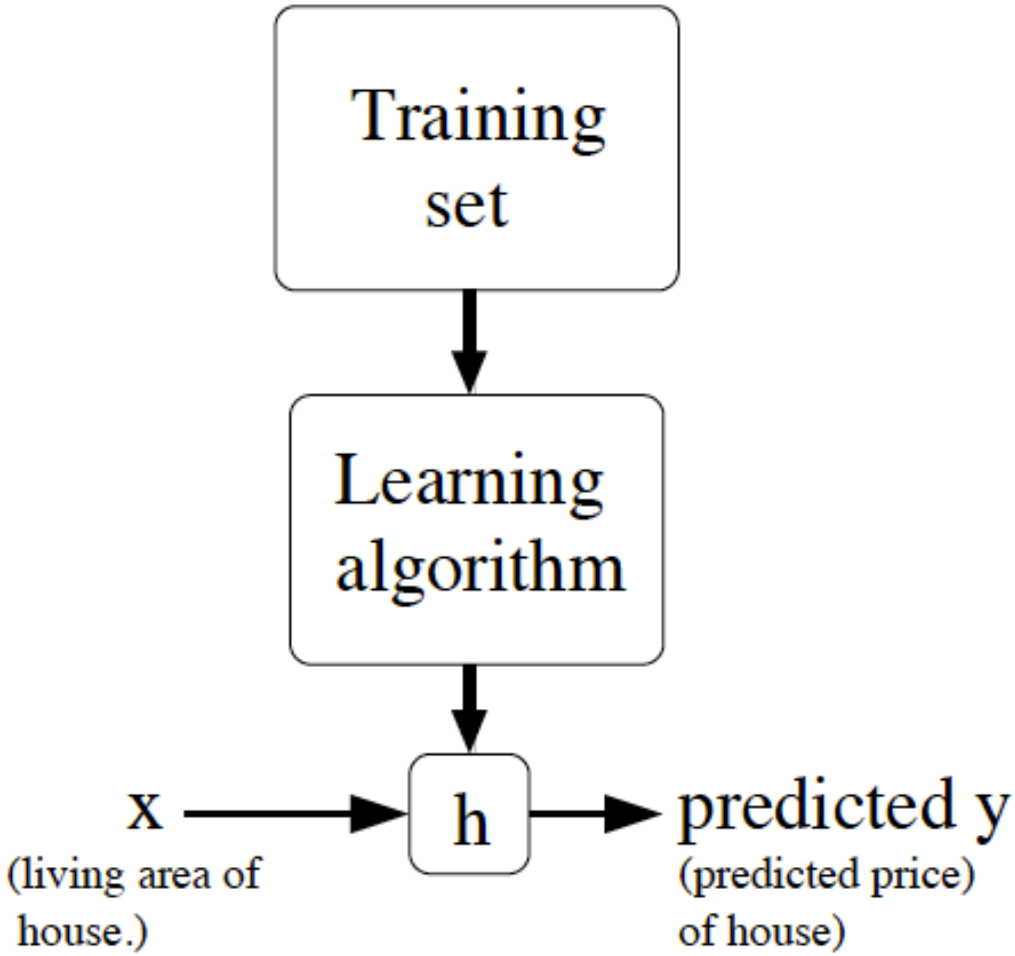
- Suppose, we have a dataset giving the living areas and prices of 47 houses from Portland, Oregon.

Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
$\vdots$	$\vdots$


housing prices
price (in $1000)
square feet
0
100
200
300
400
500
600
700
800
900
1000
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

Supervised Learning (Prediction?)

- Given data like this, how can we learn to predict the prices of other houses in Portland, as a function of the size of their living areas?



h is Hypothesis and $h_\theta(x) = \theta_0 + \theta_1 x$, here θ_i: θ_0, θ_1 are parameters (also called weights)

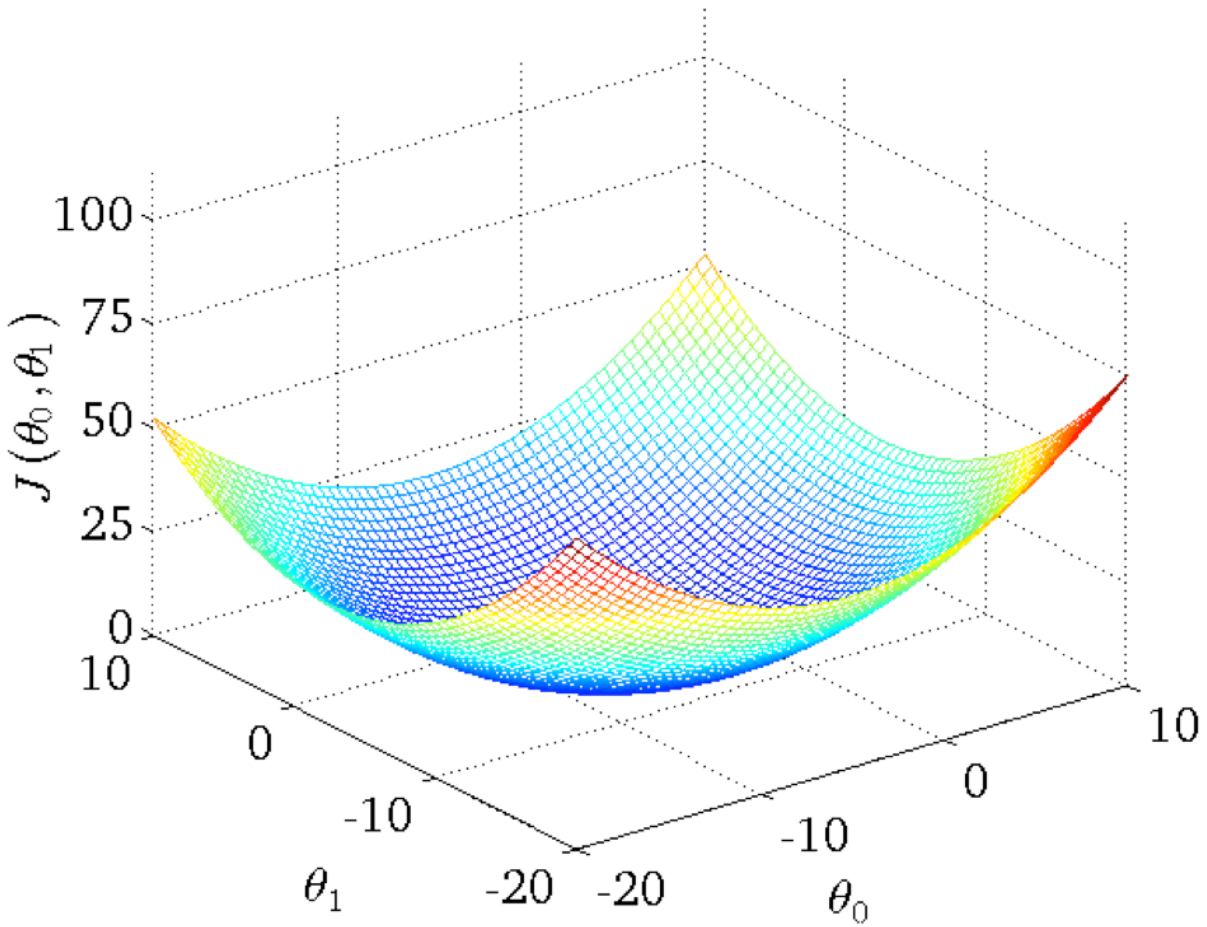
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

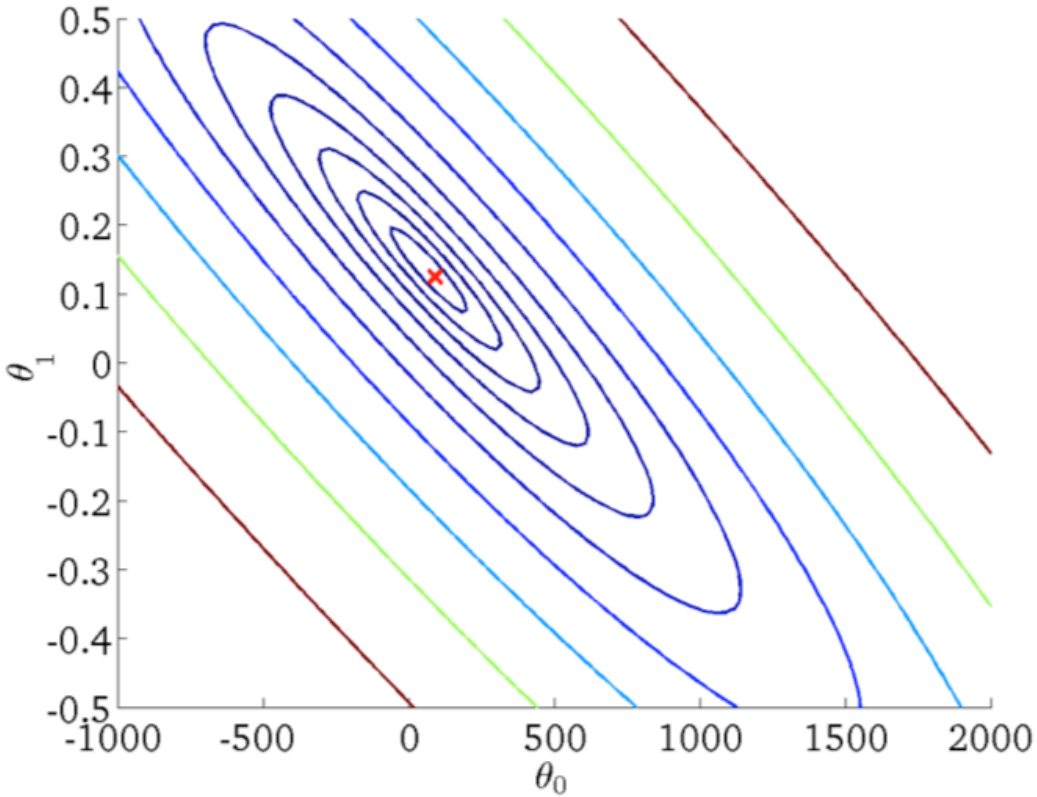
Least Square Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
here, m is # of training instances or examples (# of rows in a dataset)

Goal: $Minimize_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

3D Plot



Contour Plot



A contour plot is a graphical technique for representing a 3-dimensional surface by plotting constant z slices, called contours, on a 2-dimensional format. That is, given a value for z, lines are drawn for connecting the (x,y) coordinates where that z value occurs.

Linear Regression with Multiple features/variables

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
$\vdots$	$\vdots$	$\vdots$

Hypothesis, $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$, here θ_i: θ_0, θ_1, θ_2 are parameters (also called weights)

Linear Regression with Multiple features/variables

Hypothesis: $h_\theta(x) = h(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1, θ_2

Least Square Cost Function: $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
here, m is # of training instances or examples (# of rows in a dataset)

Goal: $Minimize_\theta J(\theta)$

Linear Regression with Multiple features

Hypothesis, $h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$

Here, n is the number of input variables

θ and x both are vectors

Least Mean Square (LMS) update rule and Gradient Descent

- Choose θ and minimize $J(\theta)$
- Use a search algorithm that starts with some initial guess for θ
- Repeatedly change θ to make $J(\theta)$ smaller until we converge to a value of θ that minimizes $J(\theta)$.

Gradient Descent

- Starts with some initial θ
- Repeatedly performs the update: $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$
  Here, α is called the learning rate.
  The update is simultaneously performed for all values of $j = 0, \ldots, n$

- Gradient descent repeatedly takes a step in the direction of steepest decrease of J.

Gradient Descent: Partial Derivative

Let's consider we have only one training example (x, y), so that we can neglect the sum in the definition of J.

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

Least Mean Square (LMS) update rule

For a single training example, this gives the update rule
$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j{}^(i)$

This rule is called LMS update rule or Widrow-Hoff learning rule

For training set with more that one example, the update rule is

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

This method looks at every example in the entire training set on every step. It's called batch gradient descent.

Stochastic Gradient Descent (SGD) or Incremental Gradient Descent

Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

}

In this algorithm, we repeatedly run through the training set, and each time we encounter a training example, we update the parameters according to the gradient of the error with respect to that single training example only.

SGD VS Batch GD

- Batch gradient descent has to scan through the entire training set before taking a single step which is a costly operation if m (# of training examples) is large.
- Stochastic gradient descent can start making progress right away, and continues to make progress with each example it looks at.
- Often, stochastic gradient descent gets close to the minimum much faster than batch gradient descent.
- Particularly, when the training set is large, stochastic gradient descent is often preferred over batch gradient descent.

References


Christopher M. Bishop, Pattern recognition and Machine learning. Springer, 2006.


Tom Mitchell, Machine learning. McGraw-Hill, 1997


Lecture Notes of Andrew Ng