APPLIED MACHINE LEARNING

An Introduction

Kishan Jainandunsing, PhD

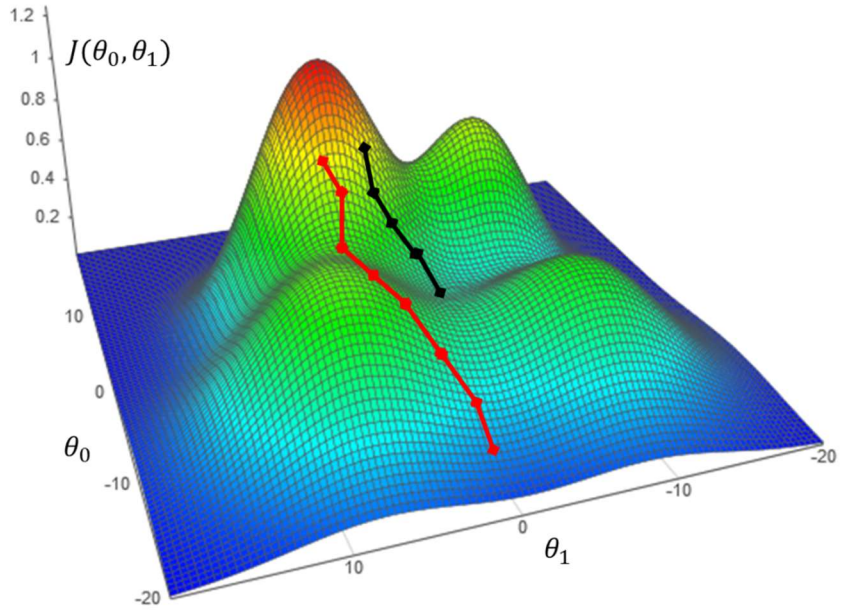

Table of Contents

List of Figures

List of Tables

1 Preface

This book is an introduction to Machine Learning (ML) and artificial neural networks (ANNs). It provides the novice to ML and ANNs with a solid understanding of ML and ANNs and veterans in these fields with a useful reference book and refresher. The material in this book not only covers the mathematics which lays at the foundation of ML and ANNs but it also provides numerous examples of real world applications, henceforth, the title. A good understanding of the foundational mathematics is important and therefore the reader is encouraged to not skip through it.

The material in this book covers the broad field that machine learning is, spanning linear regression, logistic regression, stochastic gradient descent, mini-batch gradient descent, regularization, neural networks, clustering, Principal Component Analysis, Support Vector Machines, Gaussian Kernels, stochastic gradient descent, and mini-batch gradient descent. The material also covers important analytical tools to evaluate and debug machine learning algorithms and their performance, such as bias, variance, and learning curve analysis, and provides insightful examples of how these tools are used effectively.

In addition the book covers design methodologies and best practices for the design of large-scale machine learning systems, to focus resources where the biggest improvements in system performance are to be gained. Lastly, the material covers several important machine learning applications, such as anomaly detection, recommender systems, online learning, and photo optical character recognition. The reader will find ample examples that illustrate the concepts and the theory along the way.

The rest of this book is organized as follows:

- Chapter 2 provides a refresher on the linear algebra and calculus used in the book.
- Chapter 3 covers the theory and application of univariate and multivariate linear regression in depth, including gradient descent, hypotheses, cost functions and cost function minimization using iterative and direct methods.
- Chapter 4 covers the theory and application of logistic regression in depth, including complex decision boundaries, hypotheses, cost functions and cost function minimization.
- Chapter 5 covers the theory and application of regularization for linear and logistic regression, including the concepts of under- and over-fitting.

- Chapter 6 covers the theory and application of neural networks as self-learning implementations for single and multi-class classification applications, including backpropagation, scalar and vectorized implementations, and neural network architectures with single and multiple hidden layers.
- Chapter 7 covers tuning strategies for machine learning algorithms, including hypothesis evaluation, model selection, bias and variance, and the usefulness of learning curves.
- Chapter 8 covers best practices in machine learning system design, including error analyses, error metrics, precision and recall, and collection of training examples.
- Chapter 9 covers the theory and application of support vector machines, including the concept of large margin classification and the use of kernels and landmarks.
- Chapter 10 covers the theory and application of clustering for unsupervised learning, including the K-Means algorithm, Agglomerative Hierarchical Clustering, Mean-Shift Clustering, Expectation Maximizing Clustering, DBSCAN and Self-Organizing Maps.
- Chapter 0 covers the theory and application of data compression or dimensionality reduction using principal component analysis.
- Chapter 12 covers the theory and application of anomaly detection using probability densities and kernels.
- Chapter 13 covers the special application of an online recommender system for product or service recommendations, including the use of collaborative filtering.
- Chapter 14 covers the design of large-scale machine learning systems, using stochastic and mini-batch gradient descent, as well as MapReduce.
- Chapter 15 covers an example application for photo OCR to illustrate best practices to optimize resources for improving the performance of a machine learning system under development.

The material in this book is largely based on the Coursera Machine Learning course taught by Prof. Andrew Ng. Novices to ML and ANNs are encouraged to take up the course. Not only is Prof. Ng an excellent teacher and instructor, but the course also offers many hands-on programming exercises.

This page left intentionally blank.

2 Math Refresher

2.1 Linear Algebra Refresher

The reader is encouraged to read through this chapter if not thoroughly familiar with the linear algebra that underlies the material covered in this book.

Dimension of a vector:

Vectors are represented as a row or a column in this book. The notation used to denote a row vector of n elements is: $v \in \mathbb{R}^{1\times n}$ (i.e., $1\ by\ n$). The notation used to denote a column vector of n elements is: $v \in \mathbb{R}^{n\times 1}$ (i.e., $n\ by\ 1$).

Dimension of a matrix:

A matrix of m rows and n columns is denoted as: $A \in \mathbb{R}^{m\times n}$ (i.e., $by\ n$). The element in the ith row and the jth row is denoted as a_{ij}.

The transpose operator:

The transpose of a vector v is written as v^T and turns a row vector into a column vector and vice versa. This means that if $v \in \mathbb{R}^{1\times n}$, then $v^T \in \mathbb{R}^{n\times 1}$.

The transpose of a matrix $A \in \mathbb{R}^{m\times n}$ transposes each column of the matrix to a row. Hence, $A^T \in \mathbb{R}^{n\times m}$.

Diagonal matrix:

A diagonal matrix $A \in \mathbb{R}^{n\times n}$ has all its off-diagonal elements equal to zero and has equal number of rows and columns:

$$A = \begin{bmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{nn} \end{bmatrix}$$

Identity matrix:

An Identity matrix $I \in \mathbb{R}^{n\times n}$ is a special type of diagonal matrix which has all its diagonal elements equal to 1:

$$I = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$$

Vector-vector addition:

The addition of a vector $v \in \mathbb{R}^{1\times n}$, and a vector $w \in \mathbb{R}^{1\times n}$ is again a vector $y \in \mathbb{R}^{1\times n}$:

$$v + w = y$$

Matrix-matrix addition:

The addition of matrix $A \in \mathbb{R}^{n\times n}$ and matrix $B \in \mathbb{R}^{n\times n}$ is again a matrix $C \in \mathbb{R}^{n\times n}$:

$$A + B = C$$

Scalar-vector multiplication:

Multiplication of a vector v by a scalar s is the element-wise multiplication of the vector by the scalar:

$$s \times v = v \times s = s \times [v_1 \dots v_n] = [s \times v_1 \dots s \times v_n]$$

Scalar-matrix multiplication:

Multiplication of a matrix M by a scalar s is the element-wise multiplication of the matrix by the scalar:

$$A \times s = s \times A = s \times \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \cdots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} s \times a_{11} & \cdots & s \times a_{1n} \\ \vdots & \cdots & \vdots \\ s \times a_{m1} & \cdots & s \times a_{mn} \end{bmatrix}$$

Inner-product:

The inner-product of two vectors $v, w \in \mathbb{R}^{1\times n}$ is the sum product of the elements of the vectors:

$$v \times w^T = \sum_{i=1}^{n} v_i \times w_i = s \in \mathbb{R}$$

Vector-matrix multiplication:

Multiplying a matrix $A \in \mathbb{R}^{m\times n}$ by a vector $v \in \mathbb{R}^{n\times 1}$ is performed by taking the inner-product of each row $a_i \in \mathbb{R}^{1\times n}$ of the matrix with the vector v:

$$A \times v = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \times v = \begin{bmatrix} a_1 \times v \\ \vdots \\ a_m \times v \end{bmatrix} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix} = w \in \mathbb{R}^{m\times 1}$$

Matrix-matrix multiplication:

Multiplying a matrix $A \in \mathbb{R}^{m\times n}$ by a matrix $B \in \mathbb{R}^{n\times p}$ is performed by taking the inner-product of each row $a_i \in \mathbb{R}^{1\times n}$ of the matrix A with each column $b_j \in \mathbb{R}^{n\times 1}$ of the matrix B:

$$A \times B = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \times [b_1 \dots b_p] = \begin{bmatrix} a_1 \times b_1 & \cdots & a_1 \times b_p \\ \vdots & \cdots & \vdots \\ a_m \times b_1 & \cdots & a_m \times b_p \end{bmatrix} = \begin{bmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \cdots & \vdots \\ c_{m1} & \cdots & c_{mp} \end{bmatrix} = C \in \mathbb{R}^{m \times p}$$

A special case of a matrix-matrix multiplication is the multiplication of a matrix $A \in \mathbb{R}^{m \times n}$ with an identity matrix $I \in \mathbb{R}^{n \times n}$:

$$A \times I = A$$

Inverse of a matrix:

A matrix $A \in \mathbb{R}^{n \times n}$ has an inverse $A^{-1} \in \mathbb{R}^{n \times n}$, so that $A \times A^{-1} = I \in \mathbb{R}^{n \times n}$, if and only if the matrix has full rank.

Note: Several other properties apply to an invertible matrix, such as its determinant being nonzero, and being decomposable in the product of a lower and upper triangular matrix.

Full rank:

A matrix $A \in \mathbb{R}^{n \times n}$ has full rank if all columns and rows of the matrix are linearly independent, i.e., none of the rows of the matrix can be expressed as a linear combination of any set of the rows of the matrix, except of themselves, and none of the columns of the matrix can be expressed as a linear combination of any set of the columns of the matrix, except of themselves.

2.2 Calculus Refresher

The reader is encouraged to read through this chapter if not thoroughly familiar with the calculus that underlies the material covered in this book.

Derivative of the sum or difference of two functions:

$$h = f \pm g$$

$$h' = f' \pm g' \tag{1}$$

Derivative of the product of two functions:

$$h = f \times g$$

$$h' = f' \times g + f \times g' \tag{2}$$

Derivative of the quotient of two functions:

$$h = \frac{f}{g}$$

$$h' = (f' \times g - f \times g')/g^2 \quad (3)$$

The chain rule for derivatives:

$$h = g(f)$$

$$h' = g'(f) \times f' \quad (4)$$

Derivative of $f(x) = e^x$:

$$f' = e^x \quad (5)$$

Derivative of the sigmoid function:

$$f(x) = \frac{1}{(1 + e^{-x})}$$

Using the quotient rule with $g(x) = (1 + e^{-x})$ and $h(x) = 1$, and the chain rule with $k(x) = -x$ and $l(k(x)) = e^{k(x)}$:

$$f'(x) = -\frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})} - \frac{1}{(1+e^{-x})^2} = f(x)(1 - f(x)) \quad (6)$$

3 Linear Regression

3.1 Single Variable Linear Regression

3.1.1 Model Representation

Consider the following problem. We are interested in creating a model that can predict housing prices in Portland, Oregon. We collected a set of data points that link the square footage of a house with its price and tabulated these data points in Table 1.

i	Size (sqft): $x^{(i)}$	Price ($K): $y^{(i)}$
1	2104	460
2	1416	232
3	1534	315
4	852	178
5	1530	300
6	1451	235
7	2051	405
8	901	185
9	2181	475
10	567	131

Table 1. House prices (y) by square footage (x)

Let's introduce the following terminology:

1. m = the number of data point pairs $(x^{(i)}, y^{(i)})$ or *training examples*
2. $x^{(i)}$= the *i*th *input variable* or *feature*
3. $y^{(i)}$= the *i*th *output variable* or *target variable*

Therefore, for the dataset in Table 1 we get:

1. The 1st example pair is $(x^{(1)}, y^{(1)}) = (2104, 460)$
2. The 2nd training example is $(x^{(2)}, y^{(2)}) = (1416, 232)$
3. The last training example is $(x^{(10)}, y^{(10)}) = (567, 131)$
4. And $m = 10$.

What we are interested in is an algorithm that learns from the training example set and that predicts the price of any size house fed into the algorithm. See Figure 1 .

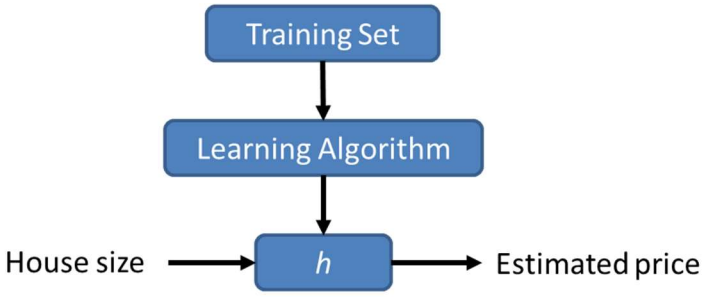

Figure 1. The house price predictor

We want to discover the relationship or *hypothesis h* between the *input variable* or *feature x*, and the *output variable* or *target variable y*. Plotting $y^{(i)}$ against $x^{(i)}$ can provide us with a clue. In Figure 2 we plotted the example pairs from Table 1 and drew a straight line to represent the *hypothesis:*

$$h\left(x^{(i)}\right) = y^{(i)} \tag{7}$$

Given that the model is linear, we can re-write equation (7) as follows:

$$h_\theta\left(x^{(i)}\right) = \theta_0 + \theta_1 x^{(i)} = y^{(i)} \tag{8}$$

The challenge is to find $\theta_0$and θ_1 such that $h_\theta(x^{(i)})$ predicts $y^{(i)}$ as best as possible for the complete training example set. This is the subject of the next section.

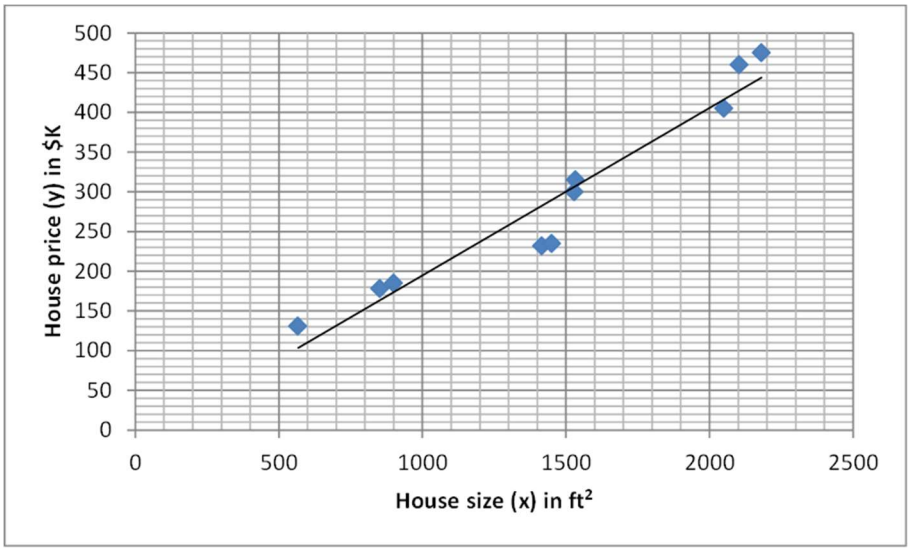

Figure 2. Scatter plot of example pairs $(x^{(i)}, y^{(i)})$

3.1.2 The Cost Function

Given the hypothesis $h_\theta\left(x^{(i)}\right)$ in equation (8), the challenge is to choose the parameters θ_0 and $\theta_1$such that we minimize the error between $h_\theta\left(x^{(i)}\right)$ and $y^{(i)}$ for the complete set of 10 training examples ($m = 10$).

One way of doing this is to minimize the sum of the squares of the differences between the pairs $(h_\theta\left(x^{(i)}\right), y^{(i)})$ for the complete set of example pairs, $(x^{(i)}, y^{(i)})$, $i \in \{1, \dots, m\}$. I.e., we are looking to compute the minimum of $J(\theta)$:

$$\min_{\theta_{0,}\theta_1} J(\theta) \tag{9}$$

We can write $J(\theta)$ as follows:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2, for\ \theta = (\theta_0, \theta_1) \tag{10}$$

This is the classical and well understood Least Squares Minimization (LSM) problem, where $J(\theta)$ is the Least Square Error (LSE) or the cost associated with the choice of parameter θ. A graphical interpretation is shown in Figure 3.

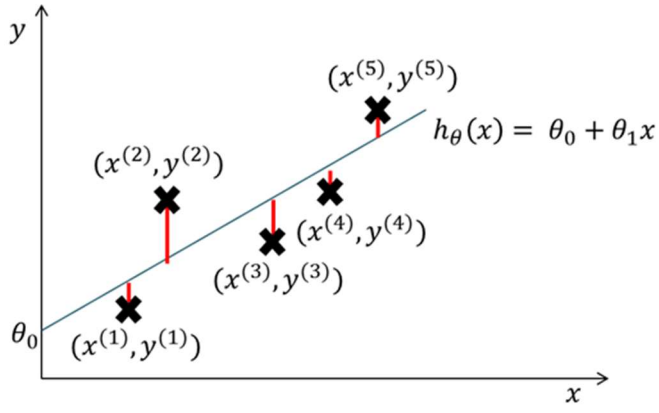

Figure 3. Errors $\boldsymbol{\varepsilon^{(i)} = h_\theta\left(x^{(i)}\right) - y^{(i)}}$

Solving equation (9)$\min_{\theta_{0,}\theta_1} J(\theta)$ requires finding the zeros for the partial derivatives in θ_0 and θ_1 of $J(\theta)$:

$$\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)=0 \tag{11}$$

$$\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1)=0 \tag{12}$$

Before stating the solution, we first develop an intuitive understanding how to approach a solution. Let:

(1) $\theta_0=0$

(2) $h_\theta\left(x^{(i)}\right)=\theta_1 x$ for $\theta_0=0$

(3) $J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2=\frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x^{(i)}-y^{(i)})^2$

(4) $\min_{\theta_1} J(\theta_1)$

Assume for a moment that we have the situation in Figure 4 for example pairs $(x^{(i)},y^{(i)})$.

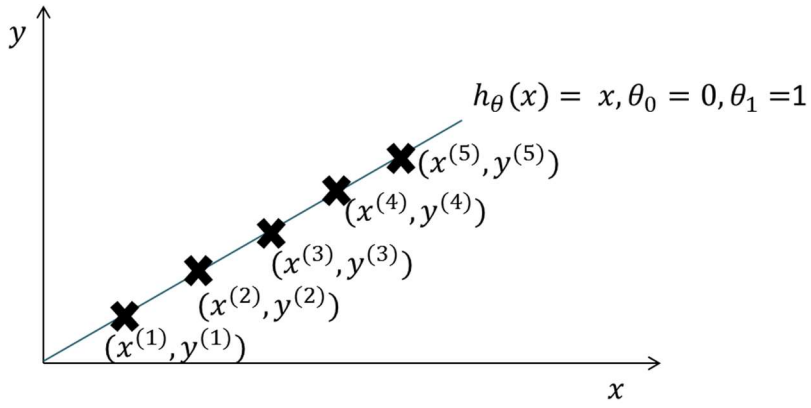

Figure 4. Hypothesis with cost or LSE equal to 0 for the example set

That is:

(1) $\theta_0=0$

(2) $\theta_1=1$

(3) $J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2=\frac{1}{2m}\sum_{i=1}^{m}(y^{(i)}-y^{(i)})^2=0$

Hence, in this case $\min_{\theta} J(\theta)$ is obtained for $\theta_0=0$ and $\theta_1=1$. I.e., $J(\theta)=0$.

If we let θ_1 vary, $J(\theta)$ becomes a function of θ_1. Figure 5 shows how $h_\theta(x)$ varies with different choices of θ_1, where $\theta_0=0$.

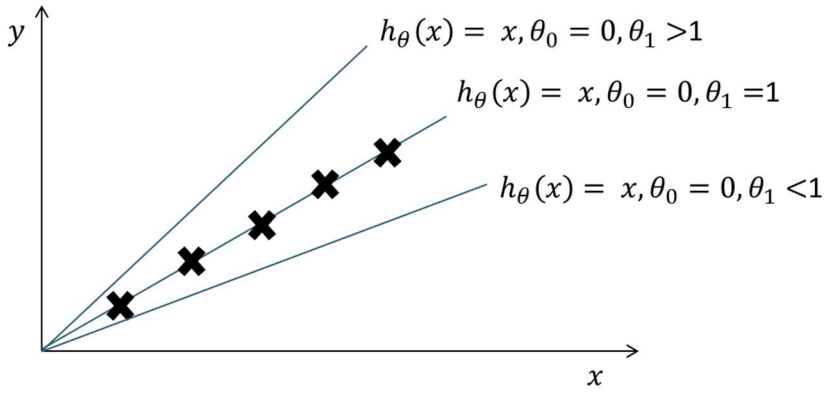

Figure 5. Letting $\boldsymbol{\theta_1}$ vary around 1 where the cost $\boldsymbol{J(\theta) = 0}$.

From equation (10) it follows that the cost function, $J(\theta)$, is a quadratic function:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x) - y)^2 = \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x - y)^2 \tag{13}$$

The values of $J(\theta)$ in equation (13) are tabulated against the values of $\theta = (\theta_0 = 0, \theta_1)$ in Table 2 for a few values of θ_1 around 1, for an example set of $m = 3$ and $x \in \{1,2,3\}$.

θ_1	$J(\theta)$
1.75	1.3
1.5	0.6
1.25	0.1
1	0.0
0.75	0.1
0.5	0.6
0.25	1.3

Table 2. $\boldsymbol{J(\theta)}$ as a function of $\boldsymbol{\theta_1}$, where $\boldsymbol{\theta_0 = 0}$

The graph of $J(\theta_1)$ is shown in Figure 6.

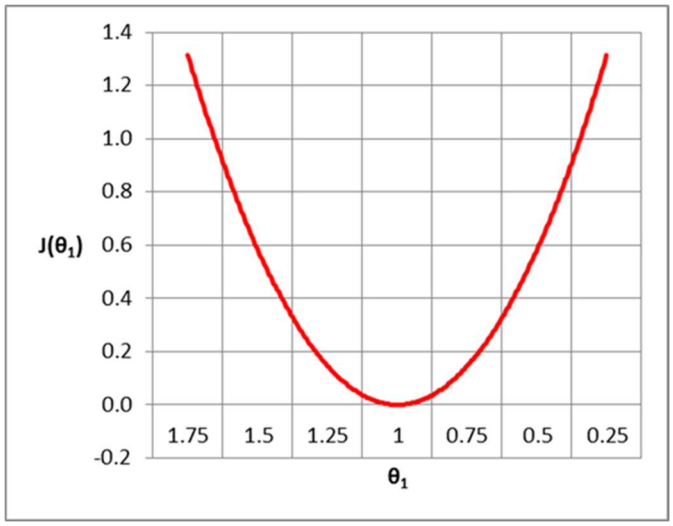

Figure 6. Graph of $J(\theta_1)$ in equation (6) as a function of θ_1

The key take away from this graph is that the cost function $J(\theta_1)$ is a hyperbolic function with global minimum at $\theta_1 = 1$, for $\theta_0 = 0$. Indeed, using calculus:

$J(\theta_0) = \frac{1}{6}((\theta_0 - 1)^2 + (\theta_0 - 2)^2 + (\theta_0 - 3)^2)$, hence, $\frac{\partial}{\partial \theta_0} J(\theta_0) = \theta_0 - 2 = 0$, for $\theta_0 = 2$

Similarly, from $(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\theta_0 - y)^2$, for $\theta_1 = 0$, we can find that the cost function $J(\theta_0)$ is a hyperbolic function with global minimum at $\theta_0 = 2$. Indeed, using calculus:

$J(\theta_1) = \frac{1}{6}((\theta_1 - 1)^2 + (2\theta_1 - 2)^2 + (3\theta_1 - 3)^2)$, hence, $\frac{\partial}{\partial \theta_1} J(\theta_1) = \frac{5}{3}(\theta_1 - 1) = 0$, for $\theta_1 = 1$

3.1.3 Gradient Descent

We can solve for the global minimum of the cost function using calculus as shown previously, or we can use an iterative method. Figure 5 and Figure 6 give us a clue how to construct an iterative method. The key in Figure 5 and Figure 6 is that we let θ_1 change in the direction that reduces $J(\theta_1)$ with each step. The following algorithm accomplishes this and is called gradient descent:

Repeat until convergence for $\alpha \geq 0$ *{* (14)

$\theta_0 \coloneqq \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$,

$\theta_1 \coloneqq \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$,

}

The partial derivative term $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ provides the direction in which θ_j must be changed for each step, until $J(\theta_0, \theta_1)$ has reached its minimum. See Figure 7. For the example here above this occurs for $\theta_0 = 2$ and $\theta_1 = 1$.

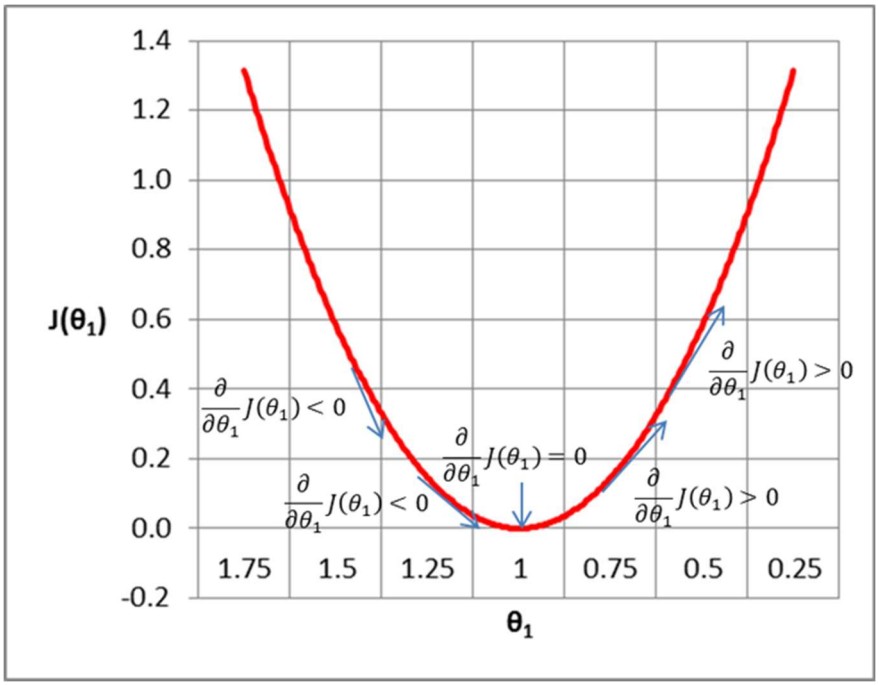

Figure 7. Using the partial derivative in θ_1 to minimize $J(\theta_1)$ iteratively

The term α is called the *learning rate*. Its magnitude determines how fast $J(\theta_0, \theta_1)$ reaches its minimum. Note from Figure 7 that if α is too small, $J(\theta_0, \theta_1)$ will take a long time before reaching its minimum, whereas if α is too large, $J(\theta_0, \theta_1)$ keeps overshooting the minimum and never convergence to the minimum.

Important notes:

1. When updating θ_j, it is important to update $J(\theta_0, \theta_1)$ only after *all* new θ_j variables have been updated.
2. It's not guaranteed that $J(\theta_0, \theta_1)$ converges to a global minimum. Depending on the starting values of θ_j , $J(\theta_0, \theta_1)$ can end up in a local minimum. See example in Figure 8.
3. It's not necessary to change the value of α while $J(\theta_0, \theta_1)$ approaches a minimum, since the derivative term $\frac{\partial}{\partial \theta_j} J(\theta_j)$ continues to shrink, provided that α is not too large. See Figure 9.

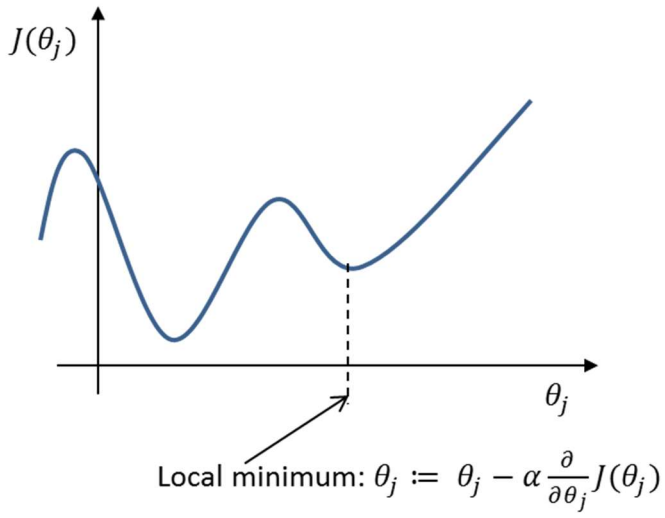

Figure 8. Convergence to a local instead of the global minimum

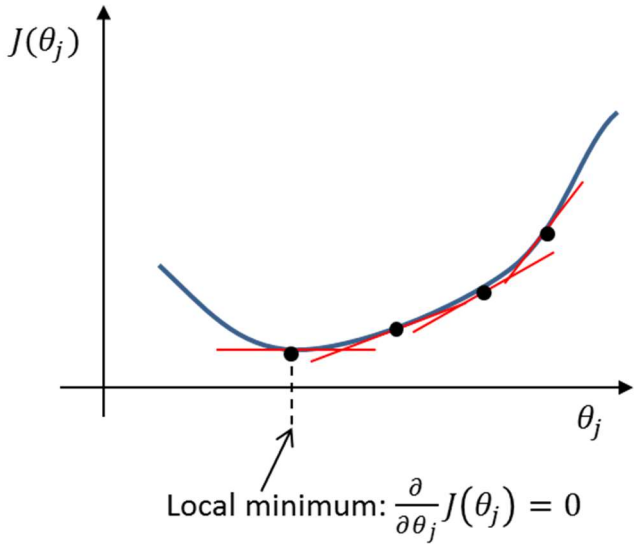

Figure 9. Shrinking derivative $\frac{\partial}{\partial \theta_j} J(\theta_j)$ as θ_j approaches the minimum

Next we compute the derivative term $\frac{\partial}{\partial \theta_j} J(\theta_j)$ for θ_j.

Remember that:

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

(15.a)

and

$$h_\theta\left(x^{(i)}\right) = \theta_0 + \theta_1 x^{(i)} \tag{15.b}$$

Using the fact that the derivative of a function $f(x) = (ax + b)^2$ is $f'(x) = 2a(ax + b)$, it is easy to see that:

$$\frac{\partial}{\partial \theta_0} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta\left(x^{(i)}\right) - y^{(i)}) \tag{16.a}$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta\left(x^{(i)}\right) - y^{(i)})\, x^{(i)} \tag{16.b}$$

Hence, the gradient descent algorithm in (14) can be rewritten as:

Repeat until convergence for $\alpha \geq 0$ *{* (17)

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

$$\theta_1 \coloneqq \theta_1 - \alpha(\frac{1}{m} \sum_{i=1}^{m} (h_\theta\left(x^{(i)}\right) - y^{(i)})\, x^{(i)}),$$

}

A visualization of the gradient descent steps is provided in Figure 10 for two different starting points.

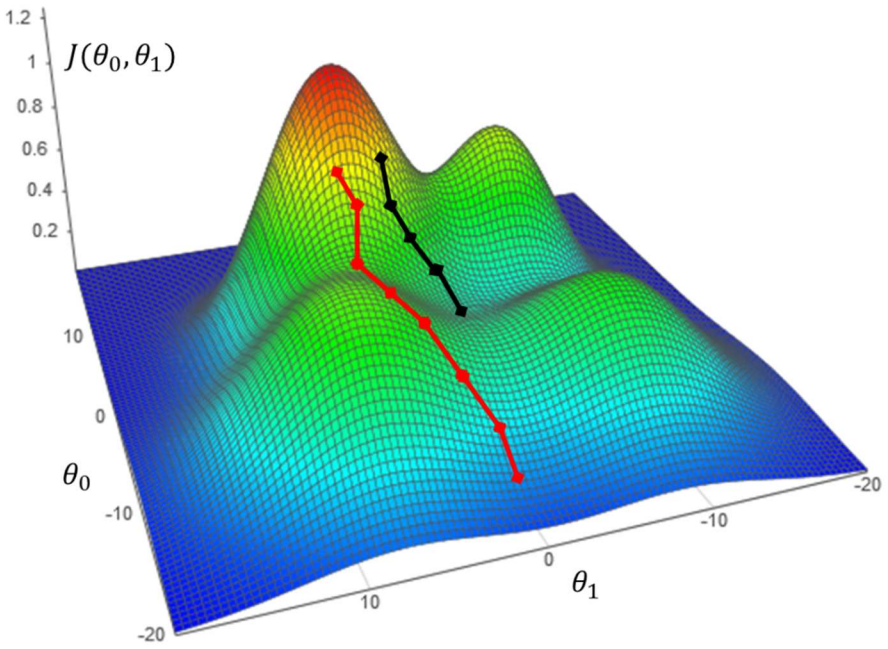

Figure 10. Visualization of the gradient descent algorithm in (17)

In case $J(\theta_0, \theta_1)$ is not a concave function, as shown in Figure 11, there can be multiple local minima, and as noted earlier, depending on the starting values of θ_0 and θ_1, gradient descent can end up in a local minimum (the red path in Figure 10).

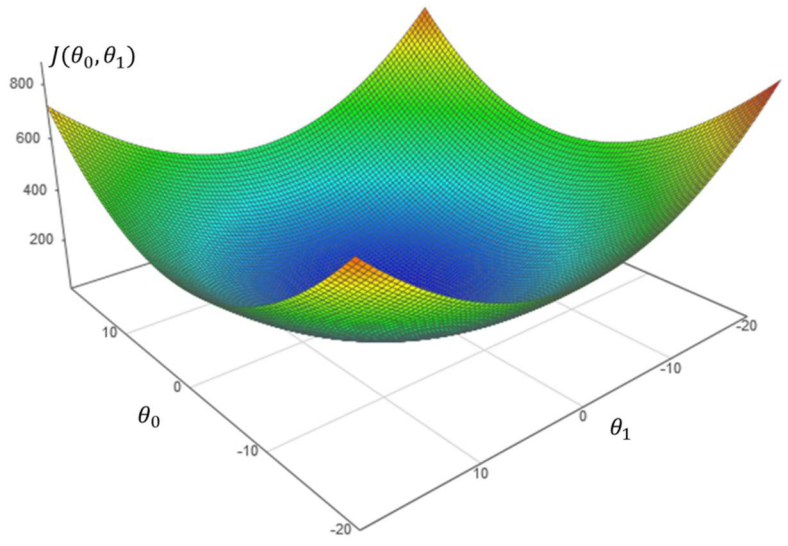

Figure 11. $J(\theta)$ as a concave function

An alternative and easier way of visualizing the progression of the gradient descent is through a contour map. The sequence in Figure 12 shows the gradient descent path in the contour map of the cost function $J(\theta)$ side by side with the change in the hypothesis $h_\theta(x)$.

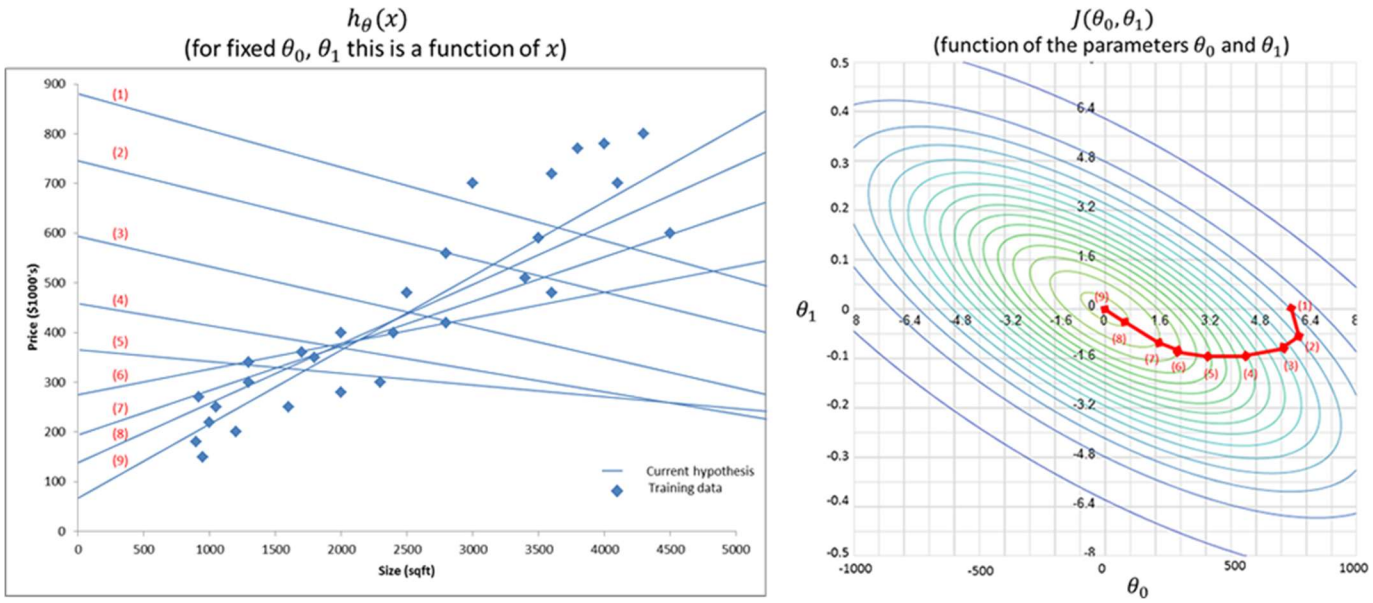

Figure 12. Hypothesis $h_\theta(x)$ changes with corresponding gradient descent steps for cost $J(\theta)$

As a final note, the gradient descent algorithm in (17) is also called *batch gradient descent* since each step uses all the training examples.

3.2 Multivariate Linear Regression

In the previous section we considered single-variable linear regression, or univariate linear regression. I.e., there was only one variable in the training example set as shown in Table 1 for square feet vs. price. Note that the hypothesis for univariate linear regression $h_\theta\left(x^{(i)}\right) = \theta_0 + \theta_1 x^{(i)}$ has two variables though, θ_0 and θ_1. However, univariate refers to single feature and not the number of parameters in the hypothesis.

In this section we expand univariate linear regression to multivariate linear regression. Consider augmenting square feet with number of bedrooms, number of floors and age of the home. See Table 3.