Module II Regression Based Learning

1 Introduction

In Figure 1-1 is illustrated the layout of a typical machine learning environment. The three main concepts involved in the representation of the environment are the data, which in the machine learning jargon is called the training set or training data, a machine learning algorithms whose task is to derive a model of the data contained in the training set, and the model itself. The training set is typically in the form of a table with data organized into columns and rows. The data may be labeled or unlabeled depending on the type of learning used (*as supervised* or *unsupervised learning*). The machine learning algorithm component may be one of the wide range of algorithms available in literature. The model produced by the algorithm is an abstract representation of the data in the training set. It may be in the form of a mathematical equation or formula or a set of association rules. In this learning module, we focus on a supervised machine learning methodology called regression. In particular, we would discuss two common regression learning methods, i.e., linear and logistic regressions at length. We would also introduce the gradient descent algorithm and least mean square (LMS) rule, two important concepts in error-correction based learning. In the subsequent section, we would introduce the notation used in these lecture notes whose understanding would be useful in grasping the explanations later.

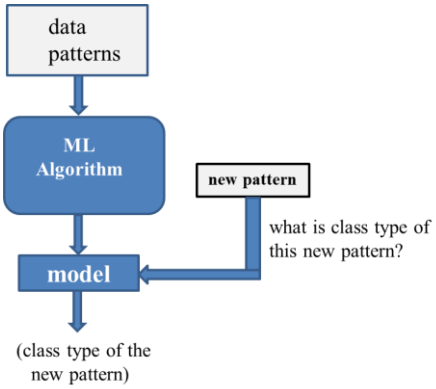

Figure 0-1 A Machine learning environment

1.1 The training set and its notation

In most machine learning settings, a training set is a rectangular arrangement or table of values. Each row in the table is called a training example or a training sample. If the learning mode is supervised, a training example comprises of two components; a vector of input values called the feature vector; and one or more output values called the target or observed values. Each feature vector is a finite list of ordered values and each value is a measure of some feature of the pattern under observation. The output value is either a real number or some discrete value representing a category or class (depending on whether we are dealing with the regression or classification problem). Table 1 in Figure 0-1 gives an illustration of the structure of the training set.

The training data in Table 1 has m training examples numbered 1 through m. Each training example consists of a $x^{(i)}$ component, which represents the feature vector and a $y^{(i)}$ component, which is the target or output value for that feature vector. The index i ($i = 1, 2, \dots, m$) in the superscript position denotes the training example number. For example, $x^{(i)}$ means the feature vector of the ithtraining example and $y^{(i)}$ the output value for the ithtraining example. Each feature vector has n values and each value is denoted by the notation $x_j^{(i)}$ with $j = 1, 2, \dots, n$ in the subscript position representing the feature number. See Figure 0-1.

Table 0-1

training example#	$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	..	$x_n^{(i)}$	$y^{(i)}$
1	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$		$x_n^{(1)}$	$y^{(1)}$
2	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$		$x_n^{(2)}$	$y^{(2)}$
3	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$		$x_n^{(3)}$	$y^{(3)}$
…	…	…	…		…	…
m	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$		$x_n^{(m)}$	$y^{(m)}$

$$x^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \\ \vdots \\ x_n^{(1)} \end{bmatrix} \qquad x^{(3)} = \begin{bmatrix} x_1^{(3)} \\ x_2^{(3)} \\ x_3^{(3)} \\ \vdots \\ x_n^{(3)} \end{bmatrix}$$

$x_1^{(1)}$ means feature 1 of feature vector 1, $x_2^{(1)}$ feature 2 of feature vector 1, and so on. $y^{(1)}$means the target or output for $x^{(1)}$

$x_1^{(3)}$ means feature 1 of feature vector 3, $x_2^{(3)}$ feature 2 of feature vector 3, and so on.$y^{(3)}$means the target or output for $x^{(3)}$.

Figure 0-2 Structure of the training data

In the figure above, the vectors $x^{(1)}$ and $x^{(2)}$ have been given in column notation. When inside text, vectors $x^{(1)}$ and $x^{(2)}$ can also be written as row vectors. For example, $x^{(1)}$ can be written as $x^{(1)} = [x_1^{(1)}\ x_2^{(1)}\ \ldots\ x_n^{(1)}]^T$and $x^{(2)}$ as $x^{(2)} = [x_1^{(2)}\ x_2^{(2)}\ \ldots\ x_n^{(2)}]^T$. The row vector notation has the advantage that contrary to column vector notation, vectors in row notation do not stretch over multiple lines of text and, thus, the required amount of spacing between lines is preserved. Having introduced the basic notation above, we are now in a position to introduce the formal notation for specifying training the training set in these lecture notes.

A training data in these notes would be formally specified as follows.

- Let $\tau = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, $x^{(i)} \in \mathbb{R}^n$, $y^{(i)} \in \mathbb{R}$

The above notation means, there are m training examples, each comprising of a feature vector of n real values and a target of one real value. In mathematical jargon, we say the vector $x^{(i)}$ takes its value from a domain that is the Cartesian product of n real sets. For $y^{(i)}$, we say that it takes its values from the real number set. We would also be using the notation, e.g., we will use $X \in \mathbb{R}^{m \times n}$ to denote the matrix of feature vectors in the training data and $Y \in \mathbb{R}^m$ the vector of target values. See Figure 0-2.

Table 0-2

training example#	$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	…	$x_n^{(i)}$	$y^{(i)}$
1	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$		$x_n^{(1)}$	$y^{(1)}$
2	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$		$x_n^{(2)}$	$y^{(2)}$
3	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$		$x_n^{(3)}$	$y^{(3)}$
…	…	…	…		…	…
m	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$		$x_n^{(m)}$	$y^{(m)}$

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \cdots & \vdots \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

Example 1. The Breast Cancer Example

Consider a doctor or consortium of doctors examining patients for the diagnoses of breast cancer. The cancer arises as a tumor but not all tumors are cancers and the doctor has to carefully examine the physical attributes of a tumor to classify it as either malignant (cancer-causing and harmless) or benign (non-cancer causing and harmless). Assume that the doctor uses five attributes of the tumor to decide whether the tumor is malignant or benign. These attributes are the mean radius, mean texture, mean area, mean diameter, and fractal dimension of a tumor. Assume performing diagnostic on 6 patients, the doctor comes to have the following table of data (Table 2, Figure 0-3) along with the diagnostic. The data has more formally been represented in Table 3. The symbol x_1 denotes the feature "mean radius" of the tumor, x_2, the "mean texture", and so on. $x^{(1)} = [x_1^{(1)}\ x_2^{(1)} x_3^{(1)} x_4^{(1)} x_5^{(1)}]^T = [17.90\ 10.4\ 12.3\ 26.54\ 0.12]^T$ is the data related to patient 1 represented as a row vector, $x^{(2)} = [x_1^{(2)}\ x_2^{(2)} x_3^{(2)} x_4^{(2)} x_5^{(2)}]^T = [20.56\ 17.8\ 13.3\ 18.6\ 0.08]^T$ data related to patient 2, and so

on. $y^{(i)}$ represents the diagnostic result for a patient with "0" denoting a "benign" tumor and "1" a "malignant" tumor. For example, $y^{(1)}$ denotes the diagnostic result for patient 1, which is 0, $y^{(3)}$ denotes the diagnostic result of patient 3, which is 1 and so on.

We will denote this training data as $\tau = \{(x^{(i)}, y^{(i)})\}_{i=1}^{6}$, $x^{(i)} \in \mathbb{R}^5$, $y^{(i)} \in \{0,1\}$, which means there are 6 training examples and each training example is comprised of a feature vector and a target value.

Table 0-3

patient #	Mean radius	Mean texture	Mean area	Mean perimeter	Fractal dimension	Diagnostic
1	17.90	10.4	12.3	26.54	0.12	benign
2	20.56	17.8	13.3	18.6	0.08	benign
3	19.60	21.2	13.0	24.3	0.08	malignant
4	15.50	18.6	13.0	23.4	0.09	benign
5	7.70	24.5	47.8	0.00	0.07	malignant
6	8.60	20.1	10.5	12.7	013	benign

Table 0-4

Training example	$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	$x_4^{(i)}$	$x_5^{(i)}$	$y^{(i)}$
$(x^{(1)}, y^{(1)})$	17.90	10.4	12.3	26.54	0.12	0
$(x^{(2)}, y^{(2)})$	20.56	17.8	13.3	18.6	0.08	0
$(x^{(3)}, y^{(3)})$	19.60	21.2	13.0	24.3	0.08	1
$(x^{(4)}, y^{(4)})$	15.50	18.6	13.0	2 .4	0.09	0
$(x^{(5)}, y^{5})$	7.70	24.5	47.8	0.00	0.07	1
$(x^{(6)}, y^{(6)})$	8.60	20.1	10.5	12.7	0.13	0

Figure 0-3

The notation $x^{(i)} \in \mathbb{R}^5$ means that a vector is a 5-tuple of real numbers (a 5-tuple is an ordered list of five values). The expression $y^{(i)} \in \{0,1\}$ means that the target value can either be 1 or 0. The vectors can also be listed as column vectors. For example, the column-format representation of vectors $x^{(4)}$ and $x^{(6)}$ is as below.

$$x^{(4)} = \begin{bmatrix} 15.50 \\ 18.60 \\ 13.00 \\ 23.40 \\ 0.09 \end{bmatrix} \qquad x^{(6)} = \begin{bmatrix} 8.60 \\ 20.10 \\ 10.50 \\ 12.70 \\ 0.13 \end{bmatrix}$$

Example 2: The Shoe Size Example

Assume that you are interested to find whether shoes size of people can be predicted from their heights and chest size. You take measurement of a few people and record the findings in the following table. The number of training examples (i.e., m) for this problem is 4, (i.e., $m = 4$). The number of features, i.e., n, is 4 (i.e.,$n = 4$).

Table 0-5

Person#	Height	Chest	Shoe size
1	72.1	38.0	10.0
2	69.0	37.6	8.20
3	70.3	37.3	8.5
4	72.2	38.1	9.5

Table 0-6

Training example	$x_1^{(i)}$	$x_2^{(i)}$	$y^{(i)}$
$(x^{(1)}, y^{(1)})$	72.1	38.0	10.0
$(x^{(2)}, y^{(2)})$	69.0	37.6	8.20
$(x^{(3)}, y^{(3)})$	70.3	37.3	8.5
$(x^{(4)}, y^{(4)})$	72.2	38.1	9.5

Figure 0-4

The notation for this data would be,

- The training set, $\tau = \{(x^{(i)}, y^{(i)})\}_{i=1}^{4}$, $x^{(i)} \in \mathbb{R}^2$, $y \in \mathbb{R}$
- The vectors are,

$$x^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix} = \begin{bmatrix} 72.1 \\ 38.0 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \end{bmatrix} = \begin{bmatrix} 69.0 \\ 37.6 \end{bmatrix} \qquad x^{(3)} = \begin{bmatrix} x_1^{(3)} \\ x_2^{(3)} \end{bmatrix} = \begin{bmatrix} 70.3 \\ 37.3 \end{bmatrix} \qquad x^{(4)} = \begin{bmatrix} x_1^{(4)} \\ x_2^{(4)} \end{bmatrix} = \begin{bmatrix} 72.2 \\ 38.1 \end{bmatrix}$$

- The target values are: $y^{(1)} = 10.0, y^{(2)} = 8.20, y^{(3)} = 8.5, y^{(4)} = 9.5$
- The matrix of feature vectors is:

$$X^T = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ x_1^{(3)} & x_2^{(3)} \\ x_1^{(4)} & x_2^{(4)} \end{bmatrix} = \begin{bmatrix} 72.1 & 38.0 \\ 69.0 & 37.6 \\ 70.3 & 37.3 \\ 72.2 & 38.1 \end{bmatrix}^T$$

- The target vector is

$$Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \end{bmatrix} = \begin{bmatrix} 10.0 \\ 8.20 \\ 8.50 \\ 9.50 \end{bmatrix}$$

Practice Problems

(Problem 1): Write mathematical notation for the training sets in the following tables. List vectors 2nd and 4th in each case.

Table 0-7

$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	$x_4^{(i)}$	$y(i)$
$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$y^{(1)}$
$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$y^{(2)}$
$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$y^{(3)}$
$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	$y^{(4)}$
$x_1^{(5)}$	$x_2^{(5)}$	$x_3^{(5)}$	$x_4^{(5)}$	$y^{(5)}$

Table 0-8

$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	$x_4^{(i)}$	$x_5^{(i)}$	$x_6^{(i)}$	$y^{(i)}$
$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$	$x_6^{(1)}$	$y^{(1)}$
$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$	$x_6^{(2)}$	$y^{(2)}$
$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$	$x_6^{(3)}$	$y^{(3)}$
$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	$x_5^{(4)}$	$x_6^{(4)}$	$y^{(4)}$

(Problem 2): Express the following notation training data in the form of tables.

(a) $\tau = \{(x^{(k)}, y^{(k)})\}_{k=1}^{l}, x^{(i)} \in \mathbb{R}^N, y^{(k)} \in \mathbb{R}$

(b) $\tau = \{(x^{(i)}, y^{(i)})\}_{i=1}^{6}, x^{(i)} \in \mathbb{R}^4, y^{(i)} \in [0,1]$

(c) $\tau = \{(x^{(i)}, y^{(i)})\}_{i=1}^{7}, x^{(i)} \in \mathbb{R}^2, y^{(i)} \in \mathbb{R}$

(Problem 3): Following is data related to a kind of follower called iris. Iris has three categories, the iris-setosa, iris-versicolor, iris-verginica. The categorization is based on the difference in length, width of sepal and petal of the flower. The data would be fed to a machine learning algorithm and that algorithm would produce that would be used a new instance of the flower in one of the three categories. Write the formal notation (as in question 2) for the training.

Table 0-9

Sample #	***Sepal Length (Cm)***	***Sepal Width (Cm)***	***Petal Length (Cm)***	***Petal Width (Cm)***	***Species***
1	7	3.2	4.7	1.4	Iris-versicolor
2	5.1	3.5	1.4	0.2	Iris-setosa

3	4.9	3	1.4	0.2	Iris-setosa
4	5.8	2.7	5.1	1.9	Iris-virginica
5	4.7	3.2	1.3	0.2	Iris-setosa
6	6.4	3.2	4.5	1.5	Iris-versicolor
7	6.9	3.1	4.9	1.5	Iris-versicolor
8	5	3.6	1.4	0.2	Iris-setosa
9	5.4	3.9	1.7	0.4	Iris-setosa
10	4.6	3.4	1.4	0.3	Iris-setosa

(Problem 4): List the feature vectors 5, 7, and 10 of the training data in problem 3 both in column and row formats. Also list the values of $x_1^{(4)}$, $x_3^{(5)}$, $x_2^{(1)}$, $x_3^{(8)}$, $y^{(8)}$, and $y^{(4)}$ from the table.

---

2 Regression based learning

Having introduced the notation and some basic terminology, we are in a position to delve deeper into the topic of regression learning. We will discuss two regression learning methods; *linear regression* and *logistic regression*. In Section 2.1, we discuss linear regression. Section 2.2 contains a discussion of logistic regression.

2.1 Linear Regression

In linear regression, we are given some data related to some independent variables (these variables represent features) together with one or more dependent variables (these variables represent the target). For example, in the Shoe-size example, height and chest size are dependent variables (features) and shoe-size is the dependent variable because our feelings are that shoes size of a person can be determined/predicted from height and chest-size of the person. In the terminology of statistics, the independent variables are called explanatory variables and the dependent variable is called the response variable. If the number of explanatory variables is one, the regression is called *simple linear regression* and if it is more than one, the regression is called *multiple linear regression* In case, there are more than one response variables in the model, the regression is called *multivariate regression.*

Linear regression is the problem of finding a linear relationship between the explanatory and response variable (i.e., fitting a linear model to the data). Stated more formally,

- Simple linear regression is the problem of finding a linear function $f: \mathbb{R} \to \mathbb{R}, x \mapsto f(x, w_0, w_1)$
- Multiple repression is the problem of finding a linear function, $f: \mathbb{R}^n \to \mathbb{R}$, $x \mapsto f(x_1, x_2, \dots, x_n, w_0, w_1, w_2, \dots, w_n)$
- Multivariate regression is the problem of finding a linear model, $f: \mathbb{R}^n \to \mathbb{R}^m$, $x \mapsto [f_1 \ f_2 \ \dots f_m]^T$ where
    - $f_1 \Longrightarrow f_1(x_1, \dots, x_n, w_{10}, w_{11}, \dots, w_{1n})$
    - $f_2 \Longrightarrow f_2(x_1, \dots, x_n, w_{20}, w_{21}, \dots, w_{2n})$
    - and so on.

The aim is to find a model such that it is the best-fit to the given data. The symbols, w_0, w_1, w_{01}, … are called the parameters of the model. The actual task in the regression problem consists in finding appropriate values for the model parameters, w_0, w_1, …

2.1.1 Solving a multiple linear regression problem

In mathematical terms, simple linear regression is equivalent to finding a function $f: \mathbb{R}^n \to \mathbb{R}$.

- Let our dataset be $\tau = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, $x^{(i)} \in \mathbb{R}^n$, $y^i \in \mathbb{R}$
- We want to derive from the dataset a predictor of the form $\hat{y}^{(i)} = w_0 + w_1 x_1^{(i)} + w^2 x_2^{(i)} + \cdots + w_n x_n^{(i)} = w_0 + \sum_{j=1}^{n} w_j x_j^{(j)}$ that describes the data as accurately as possible. This means that given an $x^{(i)} \in \tau$, $i = 1,2, \dots, m$, the predicted value of $\hat{y}^{(i)}$ is as close to the target value $y^{(i)}$as possible.

The actual task in deriving the model $\hat{y}^{(i)}$ is learning the parameters $w_0, w_1, \ldots, w_n$. This can be done in two ways. In the analytical method, values of the parameters (i.e., $w_0, w_1, \ldots, w_n$) are determined algebraically. This involves solving a system of simultaneous equations. The problem with this method is that a large number of parameters would result in a large number of equations and solving so many equations becomes unwieldy. Alternatively, the gradient-descent method is used. The gradient-descent method is an iterative method that finds values for parameters numerically. Both methods make use of a rule called least mean square (LMS) heuristic. We focus on the gradient-descent based method here. First, we discuss the application of this method in a generic setting, which would then be followed by a concrete example.

The training set τ in table form is given in Figure 2-1. To simplify notation, we introduce the variable $x_0^{(i)}$ in the dataset (see Table on the right) and set its value to the constant 1. This enables us to write $\hat{y}^{(i)} = w_0 + \sum_{j=1}^{n} w_j x_j^{(i)}$ as $\hat{y}^{(i)} = \sum_{j=0}^{n} w_j x_j^{(i)}$. We proceed as follows.

Table 2-1

#	$x_0^{(i)}$	$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	...	$x_n^{(i)}$	$y^{(i)}$
1	1	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$		$x_n^{(1)}$	$y^{(1)}$
2	1	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$		$x_n^{(2)}$	$y^{(2)}$
3	1	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$		$x_n^{(3)}$	$y^{(3)}$
…	…	…	…	…		…	…
m	1	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$		$x_n^{(m)}$	$y^{(m)}$

Table 2-2

#	$x_0^{(i)}$	Height $x_1^{(i)}$	Chest $x_2^{(i)}$	Shoe size $y^{(i)}$
1	1	72.1	38.0	10.0
2	1	69.0	37.6	8.20
3	1	70.3	37.3	8.5
4	1	72.2	38.1	9.5

Figure 2-1

(1) **The predictor:** We want to derive from the given training data a predictor of the form

$$\hat{y}^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + \cdots + w_n x_n^{(i)} \qquad (2\text{-}1)$$

Using linear algebra notation, the above equation can be written more compactly as the dot product of the vectors,

$$w = [w_0 \ w_1 \ldots \ w_n] \text{ and } x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \cdots \\ x_n^{(i)} \end{bmatrix}$$

$$\hat{y}^{(i)} = [w_0 \ w_1 \ldots w_n] \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \cdots \\ x_n^{(i)} \end{bmatrix} = w^T . x^{(i)} \qquad (2\text{-}2)$$

(1) The predictor in this case is,

$$\hat{y}^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x^{(i)}$$

Using linear algebra notation,

$$w = [w_0 \ w_1 \ w_2] \text{ and } x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$$

$$\hat{y}^{(i)} = [w_0 \ w_1 \ w_2] \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \end{bmatrix} = w^T . x^{(i)}$$

(2) **Defining the error of prediction:** For any vector $x^{(i)}$, the actual or observed value of the target is $y^{(i)}$ and the predicted value is $\hat{y}^{(i)}$ (calculated from equation (2.1) or (2.2). We want $\hat{y}^{(i)}$ to be as close to $y^{(i)}$ as possible. The discrepancy between $y^{(i)}$ and $\hat{y}^{(i)}$ is called the empirical loss or error of prediction and is denoted as $l(y^{(i)}, \hat{y}^{(i)})$. There are many ways to measure it. One common way is to define it as the squared difference between $y^{(i)}$ and $\hat{y}^{(i)}$, i.e.,

$$l(y^{(i)}, \hat{y}^{(i)}) = (y^{(i)} - \hat{y}^{(i)})^2.$$

This is summed over all the training examples in the dataset and a mean of the squared error is defined as,

$$E(w) = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$$

$E(w)$ is also called the error function or the loss function. Here, the total loss would be defined as,

$$E(w) = \frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2 \qquad (2\text{-}3)$$

(2) The squared error in this case is,

$$E(w) = \frac{1}{2}\sum_{i=1}^{4}(y^{(i)} - \hat{y}^{(i)})^2$$
$$= \frac{1}{2}\begin{bmatrix}(y^{(1)} - \hat{y}^{(1)}) + (y^{(2)} - \hat{y}^{(2)}) \\ +(y^{(3)} - \hat{y}^{(3)}) + (y^{(4)} - \hat{y}^{(4)})\end{bmatrix}$$

where

$$\begin{bmatrix}y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)}\end{bmatrix} = \begin{bmatrix}10.0 \\ 8.20 \\ 8.5 \\ 9.5\end{bmatrix}$$

$\hat{y}^{(1)}, \hat{y}^{(2)}, \hat{y}^{(3)}$, and $\hat{y}^{(4)}$ are calculated from

$$\hat{y}^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x^{(2)}$$

For example,

$$\hat{y}^{(1)} = [w_0\ w_1\ w_2]\begin{bmatrix}1 \\ 72.1 \\ 38.0\end{bmatrix}$$
$$= w_0 + 72.1w_1 + 38w_2$$

Of course, values of w_0, w_1, and w_3 have to be determined first.

(3) **Minimizing the error of prediction:** $E(w)$ is called the error of prediction or empirical risk and represents the error of prediction incurred for all the training examples in the dataset. For the predictor $\hat{y}^{(i)} = w^T.x^{(i)}$ to be correct, $E(w)$ should be minimum because $\hat{y}^{(i)}$ and eventually $E(w)$ depends on w, we should choose w such that $E(w)$ is minimized. Thus we formulate the problem of finding values for w as,

$$\min_{w^*} E(w) = \frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2 \qquad (2\text{-}4)$$

where w^* is the set of optimal weights. Optimal weights are weights that correspond to minimum $E(w)$.

(3) Formulation of the problem for this training set would become,

$$\min_{w^*} E(w) = \frac{1}{2}\sum_{i=1}^{4}(y^{(i)} - \hat{y}^{(i)})^2$$

(4) **Finding w^*:** The next step is to determine the set of optimal weights. The question is how do we determine it? By defining the loss $E(w)$ to be a quadratic function of weights and as a quadratic function has one global minimum or maximum, if we find the minimum of $E(w)$, the weights corresponding to the minimum would be optimal. Minima and maxima of functions occur at the point of concavity, and it is the point where the slope of the tangent to the graph of the function or differential of the function becomes equal to zero. This gives us a way to determine $w^* = [w_0^* \; w_1^* \; \dots \; w_n^*]$.

We can do this in two ways; analytically, by finding the differential of $E(w)$ with respect to the weight vector, setting the differentials equal to zero, and solving the resulting system of simultaneous equations for w^*; iteratively, by the method of gradient descent, in which we make use of the information about the gradient of the function to find w^*. The aim here is to explain the method of gradient descent.

The gradient of a multivariate function is the collection of its differential with respect to each independent variable. The differentials are called partial differentials. The gradient of $E(w)$ would be given by,

$$\begin{aligned}\nabla_w E(w) &= \frac{dE(w)}{dw} \\ &= \frac{dE(w)}{d\begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_n \end{bmatrix}} \\ &= \left[\frac{\partial E(w)}{\partial w_0} \;\; \frac{\partial E(w)}{\partial w_1} \dots \frac{\partial E(w)}{\partial w_n}\right]\end{aligned} \qquad (2\text{-}5)$$

(4) The gradient of the loss function for the given data would be as follows.

$$\begin{aligned}\nabla_w E(w) &= \frac{dE(w)}{dw} \\ &= \frac{dE(w)}{d\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}} \\ &= \left[\frac{\partial E(w)}{\partial w_0} \;\; \frac{\partial E(w)}{\partial w_1} \;\; \frac{\partial E(w)}{\partial w_2}\right]\end{aligned}$$

See the figure below. Assume our model were, $\hat{y}^{(i)} = w_0 + w_1 x_1^{(i)}$, then E would be dependent on w_0 and w_1. Its minimum would occur at the bottom point of the parabola shape where $w_0 = w_0^*$ and $w_1 = w_1^*$.

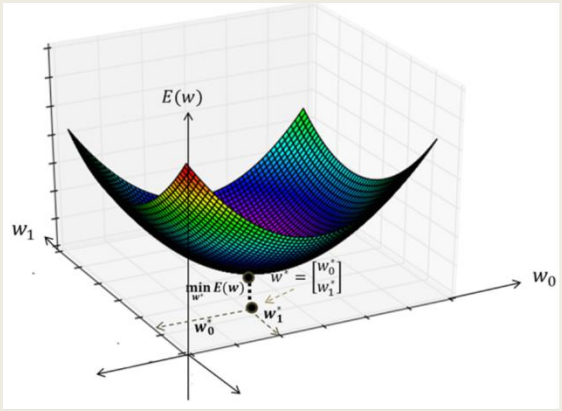

Figure 2-2

(5) The method of gradient descent: The gradient descent method is guided by the insight that $E(w)$ being a quadratic function has a graph that opens up and is a parabola. Its minimum occurs when $w = w^*$. If we move away from w^*, the gradient increases and if we move towards w^*, the gradient (i.e., slope) decreases. Thus if we start with some random values for w but update w such that it takes us towards w^*, we would find w^*, optimal values for w where $E(w)$ is minimum. This enables us to use a weight update rule to find w^*.

$$w^{Next} = w^{prev} - \eta \nabla E_w(w)$$

where w^{Next} is the value of the weight vector in the next iteration and w^{prev} is its value in the previous iteration. We can write the above in the expanded form as,

(5) The gradients $\frac{\partial E(w)}{\partial w_0}, \frac{\partial E(w)}{\partial w_1}, \frac{\partial E(w)}{\partial w_2}$ are found by applying the chain rule of differentiation. The process is explained for $\frac{\partial E(w)}{\partial w_0}$. For the other two differentials, the same process is applied.

$$\begin{aligned}\frac{\partial E(w)}{\partial w_0} &= \frac{\partial}{\partial w_0}\left[\frac{1}{2}\sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)^2\right] \\ &= \frac{1}{2} \times 2 \sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)^{2-1} \frac{\partial}{\partial w_0}\left[y^{(i)} - \hat{y}^{(i)}\right] \\ &\quad \text{Because, } \hat{y}^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)} \\ &= \sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right) \frac{\partial}{\partial w_0}\left[-w_0 x_0^{(i)} - w_1 x_1^{(i)} - w_2 x_2^{(i)}\right]\end{aligned}$$

$$\begin{bmatrix} w_0^{next} \\ w_1^{next} \\ \dots \\ w_n^{next} \end{bmatrix} = \begin{bmatrix} w_0^{prev} \\ w_1^{prve} \\ \dots \\ w_n^{prev} \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial E(w)}{\partial w_0} \\ \frac{\partial E(w)}{\partial w_1} \\ \dots \\ \frac{\partial E(w)}{\partial w_n} \end{bmatrix}$$

In the form of individual equations, we can write the above as,

$$w_0^{next} = w_0^{prev} - \eta \frac{\partial E(w)}{\partial w_0}$$

$$w_1^{next} = w_1^{prev} - \eta \frac{\partial E(w)}{\partial w_1}$$

$$\dots$$

$$w_n^{next} = w_n^{prev} - \eta \frac{\partial E(w)}{\partial w_n}$$

The individual differentials can be found as follows.

$$\frac{\partial E(w)}{\partial w_0} = \frac{\partial}{\partial w_0}\left[\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2\right]$$

Applying the chain rule of differentiation,

$$\frac{\partial E(w)}{\partial w_0} = -\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})x_0^{(i)}$$

Likewise,

$$\frac{\partial E(w)}{\partial w_1} = -\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})x_1^{(i)}$$

$$\dots$$

$$\frac{\partial E(w)}{\partial w_n} = -\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})x_n^{(i)}$$

Putting in the above equations, our weight update rules are

$$w_0^{next} = w_0^{prev} + \eta \sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})x_0^{(i)}$$

$$w_1^{next} = w_1^{prev} + \eta \sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})x_1^{(i)}$$

$$\dots$$

$$w_n^{next} = w_n^{prev} + \eta \sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})x_n^{(i)}$$

$$\frac{\partial E(w)}{\partial w_0} = -\sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)x_0^{(i)}$$

It should be noted that $\frac{\partial}{\partial w_0}\left[w_0 x_0^{(i)} + w_1 x_1^{(i)} + w_2 x_2^{(i)}\right] = x_0^{(i)}$ because we are performing differentiation with respect to w_0. In similar way,

$$\frac{\partial E(w)}{\partial w_1} = -\sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)x_1^{(i)}$$

$$\frac{\partial E(w)}{\partial w_2} = -\sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)x_2^{(i)}$$

Our weight update rules take the following form.

$$w_0^{next} = w_0^{prev} + \eta \sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)x_0^{(i)}$$

$$w_1^{next} = w_1^{prev} + \eta \sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)x_1^{(i)}$$

$$w_2^{next} = w_2^{prev} + \eta \sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)x_2^{(i)}$$

In vector form, this can be written as,

$$\begin{bmatrix} w_0^{next} \\ w_1^{next} \\ w_2^{next} \end{bmatrix} = \begin{bmatrix} w_0^{prev} \\ w_1^{prve} \\ w_2^{prev} \end{bmatrix} + \eta \begin{bmatrix} \sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)x_0^{(i)} \\ \sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)x_1^{(i)} \\ \sum_{i=1}^{4}\left(y^{(i)} - \hat{y}^{(i)}\right)x_2^{(i)} \end{bmatrix}$$

2.2 The gradient descent algorithm

Having developed the theory, we are now in a position to give the gradient descent algorithm. The algorithm given here is a variant of gradient descent algorithm called the stochastic gradient descent. Another variant of gradient descent algorithm is the batch gradient descent that would be discussed later.

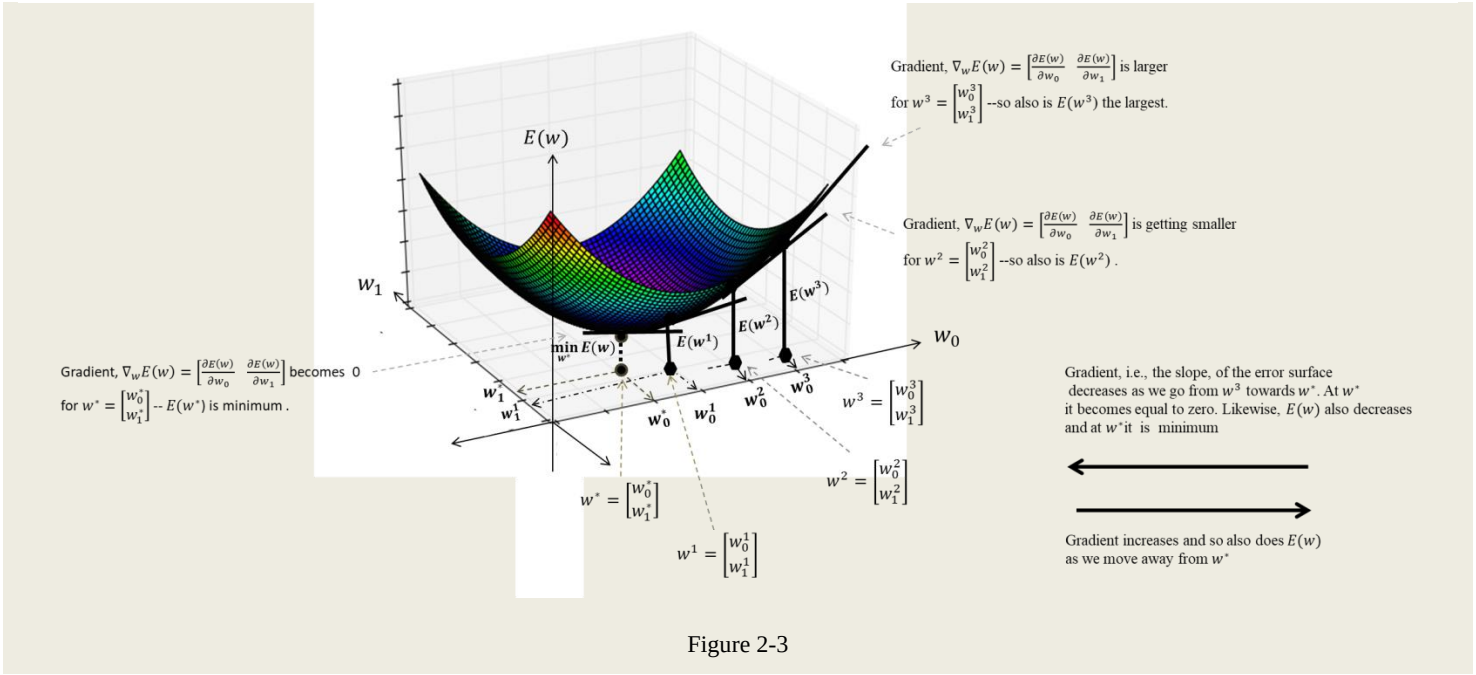

Figure 2-3

Algorithm: The Gradient Descent Algorithm

Inputs The training set; X, the feature vector, Y, the vector of target values

Outputs The weights, $w_0, w_1, \ldots, w_n$

Initialize weights, $w_0, w_1, \ldots, w_2$ and learning rate η
Set the someThresholdValue

WHILE (gradient > someThresholdValue) **DO**

FOR each training example **DO**

Compute $\hat{y}^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + \cdots + x_n^{(i)}$

Compute the gradients $\frac{\partial E(w)}{\partial w_0}, \frac{\partial E(w)}{\partial w_1}, \ldots, \frac{\partial E(w)}{\partial w_n}$

Update each weight, $w_j^{next} = w_j^{prev} - \eta \frac{\partial E(w)}{\partial w_j}, j = 1,2, \ldots, n$

Compute $\nabla_w E(w)$ and *aggregate* over the training examples

END FOR

END WHILE

Output $w_0, w_1, \ldots, w_n$

RETURN

The weights are and learning rates are initialized to some random values. The value of learning rate is typically set at some value between 0 and 1.

The while loop is called the training loop. It may be executed a fixed number of times or may be terminated when a certain condition, e.g., when the gradient approaches zero or the error of prediction reaches a threshold value. Each execution of the while loop is called an epoch.

Let 0.1, 0.5, 0.23 be the initial values of weights and 0.1 be the learning rate, then for the 1st training example of our shoe size prediction problem, $\hat{y}^{(i)}$ is calculated as follows.

$$\hat{y}^{(1)} = 0.1 \times 1 + 0.5 \times 72 + 0.25 \times 37$$

Calculations of other $\hat{y}^{(i)}$, $i = 2, 3, 4$ is done similarly. These are used in the calculation o gradients, $\frac{\partial E(w)}{\partial w_j}, j = 1, \ldots 4$, which are used to update weights.

In gradient descent algorithm, gradient of the error function (i.e. $E(w)$) is found using the chain rule of differentiation. The chain rule of differentiation works for functions of the form, $z = f(u)$ and $u = g(x_1, x_2, \dots, x_n)$. In this, the function f depends on the variable u and $u = g(x_1, x_2, \dots, x_n)$ in turn depends on variables x_1, x_2,..., and x_n. g is called the inner function and f, the outer function. The differentiation rule is as follows.

$$\nabla_{[x]} z = \frac{dz}{du}.\frac{du}{dx} = \frac{dz}{du}.\frac{du}{d[x_1 \quad x_2 \quad \dots \quad x_n]^T} = \frac{dz}{du}.\left[\frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} + \cdots + \frac{\partial u}{\partial x_n}\right] \tag{2-6}$$

Example 1: Consider the following function.

$$(x_1^2 + x_2^2 + x_3^2)^{\frac{1}{2}}$$

We want to differentiate this with respect to $x = [x_1 \quad x_2 \quad \cdots \quad x_3]^T$. To apply the chain rule, let $u = g([x_1 \quad x_2 \quad \cdots \quad x_3]^T) = x_1^2 + x_2^2 + x_3^2$. Then $z = u^{\frac{1}{2}}$. Now we apply the chain rule as in (2-6).

$$\begin{aligned}
\nabla_{[x]} z &= \frac{dz}{du}.\frac{du}{dx} = \frac{dz}{du}.\frac{du}{d[x_1 \quad x_2 \quad \dots \quad x_n]^T} = \frac{dz}{du}.\left[\frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} + \frac{\partial u}{\partial x_3}\right] \\
&= \frac{d}{du}\left[u^{\frac{1}{2}}\right].\left[\frac{\partial}{\partial x_1}(x_1^2 + x_2^2 + x_3^2) + \frac{\partial}{\partial x_2}(x_1^2 + x_2^2 + x_3^2) + \frac{\partial}{\partial x_3}(x_1^2 + x_2^2 + x_3^2)\right] \\
&= \frac{1}{2}u^{-\frac{1}{2}}.[(2x_1 + 0 + 0) + 0 + (0 + 2x_2 + 0) + (0 + 0 + 2x_3)] \\
&= \frac{(x_1 + x_2 + x_3)}{\sqrt{x_1^2 + x_2^2 + x_3^2}}
\end{aligned}$$

Problem 1: Perform the following differentiation.

(1) $(\sqrt{x_1} + x_2^3)^3$, differentiate wrt $x_1$and x_2

(2) $\sin u^{\frac{1}{2}}$, where $u = w_1^{\frac{3}{2}} + w_2^2 + \frac{1}{2}w_3 + 3w^2$, differentiate wrt $w = [w_1 \; w_2 \; w_3 \; w_4]^T$

(3) $\cos(\sin(\theta_1 + \frac{1}{2}\theta_2))$, differentiate wrt $\theta = \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix}$

(4) e^{2u} where $u = x_1 + 5\sqrt{x_2}$, differentiate wrt $x = \begin{bmatrix}x_1 \\ x_2\end{bmatrix}$

Problem 1: In linear, the form of the error function is $E(w) = \frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$. We found its gradient by differentiating with respect to the weight vector. This can also be written as $E\left(\hat{y}^{(i)}\right) = \frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$ because $\hat{y}^{(i)} = w_0 x_0^{(i)} + w_1 x_1^{(i)} + \cdots + w_n x_n^{(i)}$ where $x_0^{(i)} = 1$. Then differentiating $E\left(\hat{y}^{(i)}\right)$ with respect to the weight vector $w = [w_0 \, w_1 \, \dots w_n]^T$ can be represented as $\frac{dE}{dw} = \frac{dE}{d\hat{y}^{(i)}}\frac{d\hat{y}^{(i)}}{dw}$. There are other definitions of the error function as well. Some are listed below. Find the gradient of these functions with respect to weights assuming $\hat{y}^{(i)} = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_3 x_3^{(i)}$.

(1) $E\left(y^{(i)}, \hat{y}^{(i)}\right) = \frac{1}{\ln 2}\sum_{i=1}^{m}\ln\left(1 + e^{-y^{(i)}\hat{y}^{(i)}}\right)$

(2) $E\left(y^{(i)}, \hat{y}^{(i)}\right) = \sum_{i=1}^{m} e^{-y^{(i)}\hat{y}^{(i)}}$

2.2.1 Stochastic gradient descent verses batch gradient

The algorithm given in Figure 2-3 is a variant of gradient descent called the stochastic gradient descent algorithm (abbreviated as SGD). Another variant of gradient descent algorithm is the batch gradient descent algorithm (BGD). The difference between the two

varieties of the gradient descent is in the mode of computing the gradient and the time of weight update. In stochastic gradient descent, the weight-update is performed per training example, which means that the weights are updated immediately after the gradient is calculated. In the batch mode, the gradient values for all the training examples are aggregated first and then the weight-update is performed. Therefore, there is one weight-update operation per training cycle. This is indicated by the inclusion of the weight update operation inside the inner loop (i.e., the for loop) in Figure 2-3 and the outer loop (i.e., the while loop) in Figure 2-4. Another, version of gradient descent algorithm that is popular in neural network applications is the mini-batch gradient-descent algorithm, which divides the training set into fixed-sized groups of training examples called mini-batches and applies the weight-update on a mini-batch basis. In later section, we would describe the strengths and weaknesses of these various approaches to implementing the gradient algorithm.

…
// $j = 0, 1, 2, \ldots, n$

WHILE (gradient > someThresholdValue) **Do**

    …

    $gSum_j = 0$

    FOR each training example **DO**

        …

        $gSum_j = gSum_j + \frac{\partial E(w)}{\partial w_j}$

    END FOR

    $w_j^{next} = w_j^{prev} - \eta \times gSum_j$

END WHILE

It should be noted that the gradient is aggregated for each weight, w_0, w_1, w_2,…,w_n, separately as indicated by the use of index $j = 0,1,..,n$.

Likewise, the update is applied to each weight individually.

Figure 2-4