# Simple Linear Regression Example

In this lesson, we apply regression analysis to some fictitious data, and we show how to interpret the results of our analysis.

## Problem Statement

Last year, five randomly selected students took a math aptitude test before they began their statistics course. The Statistics Department has three questions.

- What linear regression equation best predicts statistics performance, based on math aptitude scores?
- If a student made an 80 on the aptitude test, what grade would we expect her to make in statistics?
- How well does the regression equation fit the data?

## How to Find the Regression Equation

In the table below, the $x_i$ column shows scores on the aptitude test. Similarly, the $y_i$ column shows statistics grades. The last two columns show deviations scores - the difference between the student's score and the average score on each test. The last two rows show sums and mean scores that we will use to conduct the regression analysis.

| Student | $x_i$ | $y_i$ | $(x_i-x)$ | $(y_i-y)$ |
|---|---|---|---|---|
| 1 | 95 | 85 | 17 | 8 |
| 2 | 85 | 95 | 7 | 18 |
| 3 | 80 | 70 | 2 | -7 |
| 4 | 70 | 65 | -8 | -12 |
| 5 | 60 | 70 | -18 | -7 |
| **Sum** | 390 | 385 | | |
| **Mean** | 78 | 77 | | |

And for each student, we also need to compute the squares of the deviation scores (the last two columns in the table below).

| Student | $x_i$ | $y_i$ | $(x_i-x)^2$ | $(y_i-y)^2$ |
|---|---|---|---|---|
| 1 | 95 | 85 | 289 | 64 |

| | | | | |
|---|---|---|---|---|
| 2 | 85 | 95 | 49 | 324 |
| 3 | 80 | 70 | 4 | 49 |
| 4 | 70 | 65 | 64 | 144 |
| 5 | 60 | 70 | 324 | 49 |
| **Sum** | 390 | 385 | 730 | 630 |
| **Mean** | 78 | 77 | | |

And finally, for each student, we need to compute the product of the deviation scores.

| Student | $x_i$ | $y_i$ | $(x_i-x)(y_i-y)$ |
|---|---|---|---|
| 1 | 95 | 85 | 136 |
| 2 | 85 | 95 | 126 |
| 3 | 80 | 70 | -14 |
| 4 | 70 | 65 | 96 |
| 5 | 60 | 70 | 126 |
| **Sum** | 390 | 385 | 470 |
| **Mean** | 78 | 77 | |

The regression equation is a linear equation of the form: $\hat{y} = b_0 + b_1 x$ . To conduct a regression analysis, we need to solve for $b_0$ and $b_1$. Computations are shown below. Notice that all of our inputs for the regression analysis come from the above three tables.

First, we solve for the regression coefficient ($b_1$):

$$b_1 = \Sigma [ (x_i - x)(y_i - y) ] / \Sigma [ (x_i - x)^2]$$

$$b_1 = 470/730$$

$$b_1 = 0.644$$

Once we know the value of the regression coefficient ($b_1$), we can solve for the regression slope ($b_0$):

$$b_0 = y - b_1 * x$$

$$b_0 = 77 - (0.644)(78)$$

$$b_0 = 26.768$$

Therefore, the regression equation is: $\hat{y} = 26.768 + 0.644x$ .

## How to Use the Regression Equation

Once you have the regression equation, using it is a snap. Choose a value for the independent variable (*x*), perform the computation, and you have an estimated value ($\hat{y}$) for the dependent variable.

In our example, the independent variable is the student's score on the aptitude test. The dependent variable is the student's statistics grade. If a student made an 80 on the aptitude test, the estimated statistics grade ($\hat{y}$) would be:

$$\hat{y} = b_0 + b_1x$$

$$\hat{y} = 26.768 + 0.644x = 26.768 + 0.644 * 80$$

$$\hat{y} = 26.768 + 51.52 = 78.288$$

## How to Find the Coefficient of Determination

Whenever you use a regression equation, you should ask how well the equation fits the data. One way to assess fit is to check the coefficient of determination, which can be computed from the following formula.

$$R^2 = \{ ( 1 / N ) * \Sigma [ (x_i - x) * (y_i - y) ] / (\sigma_x * \sigma_y ) \}^2$$

where N is the number of observations used to fit the model, $\Sigma$ is the summation symbol, $x_i$ is the x value for observation i, x is the mean x value, $y_i$ is the y value for observation i, y is the mean y value, $\sigma_x$ is the standard deviation of x, and $\sigma_y$ is the standard deviation of y.

Computations for the sample problem of this lesson are shown below. We begin by computing the standard deviation of x ($\sigma_x$):

$$\sigma_x = \text{sqrt} [ \Sigma ( x_i - x )^2 / N ]$$

$$\sigma_x = \text{sqrt}( 730/5 ) = \text{sqrt}(146) = 12.083$$

Next, we find the standard deviation of y, ($\sigma_y$):

$$\sigma_y = \text{sqrt} [ \Sigma ( y_i - y )^2 / N ]$$

$$\sigma_y = \text{sqrt}( 630/5 ) = \text{sqrt}(126) = 11.225$$

And finally, we compute the coefficient of determination ($R^2$):

$$R^2 = \{ ( 1 / N ) * \Sigma [ (x_i - x) * (y_i - y) ] / (\sigma_x * \sigma_y ) \}^2$$

$$R^2 = [ ( 1/5 ) * 470 / ( 12.083 * 11.225 ) ]^2$$

$$R^2 = ( 94 / 135.632 )^2 = ( 0.693 )^2 = 0.48$$

A coefficient of determination equal to 0.48 indicates that about 48% of the variation in statistics grades (the dependent variable) can be explained by the relationship to math aptitude scores (the independent variable). This would be considered a good fit to the data, in the sense that it would substantially improve an educator's ability to predict student performance in statistics class.

## Introduction to Multiple Regression

Simple linear regression is a technique for predicting the value of a dependent variable, based on the value of a single independent variable. Sometimes, you only need one relevant independent variable to make an accurate prediction.

Often, however, the prediction is better when you use two or more independent variables. Multiple regression is a technique for predicting the value of a dependent variable, based on the values of two or more independent variables.

### The Regression Equation

This is a tutorial about *linear* regression, so our focus is on *linear* relationships between variables. The regression equation that expresses the linear relationships between a single dependent variable and one or more independent variables is:

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + \ldots + b_{k-1}x_{k-1} + b_kx_k$$

In this equation, $\hat{y}$ is the *predicted* value of the dependent variable. Values of the *k* independent variables are denoted by $x_1, x_2, x_3, \ldots, x_k$.

And finally, we have the *b*'s - $b_0, b_1, b_2, \ldots, b_k$. The b's are constants, called regression coefficients. Values are assigned to the *b*'s based on the principle of least squares.

## What is the Principle of Least Squares?

In multiple regression, the deviation of the actual value for a dependent variable from its predicted value is called the residual. The residual (e) for a single observation *i* is:

$$e_i = y_i - \hat{y}_i = y_i - ( b_0 + b_1x_{1i} + b_2x_{2i} + \ldots + b_kx_{ki} )$$

Assume that the set of data consists of *n* observations. The principle of least squares requires that the sum of squared residuals for all *n* observations be minimized. That is, we want the following value to be as small as possible:

$$\Sigma [ y_i - ( b_0 + b_1x_{1i} + b_2x_{2i} + \ldots + b_kx_{ki} ]^2$$

Regression analysis requires that the values of $b_0, b_1, \ldots, b_k$ be defined to minimize the sum of the squared residuals. When we assign values to regression coefficients in this way, we are following the principle of least squares.

## Normal Equations for Simple Regression

Finding the right values for regression coefficients (i.e., values that satisfy a least squares criterion) involves solving a set of linear equations. These equations can be derived using calculus, and they are called normal equations.

To illustrate the use of normal equations, let's look at simple linear regression - regression with one dependent variable (y) and one independent variable (x). With simple linear regression, the regression equation is:

$$\hat{y} = b_0 + b_1x$$

The normal equations for simple linear regression are:

$$\Sigma y_i = nb_0 + b_1( \Sigma x_i )$$

$$\Sigma x_i y_i = b_0( \Sigma x_i ) + b_1( \Sigma x_i^2 )$$

Here, we have two equations with two unknowns. The unknowns are the regression coefficients $b_0$ and $b_1$. Using ordinary algebra, we can solve for $b_0$ and $b_1$. The result is:

$$b_1 = \Sigma [ (x_i - x)(y_i - y) ] / \Sigma [ (x_i - x)^2]$$

$$b_0 = y - b_1 * x$$

where x is the mean x score, and y is the mean y score. Note that these are the same equations that we presented in a previous lesson, when we introduced the topic of simple linear regression.

The use of normal equations to assign values to regression coefficients becomes more complicated when there are two or more independent variables.

## Regression Coefficients

With simple linear regression, there is one dependent variable and one independent variable. The regression equation is:

$$\hat{y} = b_0 + b_1 x$$

In the previous lesson, we developed a **least-squares solution** for the regression coefficients of simple linear regression:

$$b_1 = \Sigma [ (x_i - x)(y_i - y) ] / \Sigma [ (x_i - x)^2]$$

$$b_0 = y - b_1 * x$$

where $\hat{y}$ is the predicted value of the dependent variable, $b_0$ and $b_1$ are regression coefficients, $x_i$ is the value of the independent variable for observation *i*, $y_i$ is the value of the dependent variable for observation *i*, x is the mean x score, and y is the mean y score.

In this lesson, we describe a least-squares solution for the regression coefficients of *multiple* regression.

## The Multiple Regression Challenge

With **simple linear regression**, there are only two regression coefficients - $b_0$ and $b_1$. There are only two normal equations. Finding **a least-squares solution** involves solving two equations with two unknowns - a task that is easily managed with ordinary algebra.

With multiple regression, things get more complicated. There are *k* independent variables and *k* + 1 regression coefficients. There are *k* + 1 normal equations. Finding a least-squares solution involves solving *k* + 1 equations with *k* + 1 unknowns. This can be done with ordinary algebra, but it is unwieldy.

To handle the complications of multiple regression, we will use matrix algebra.

## Matrix Algebra

To follow the discussion on this page, you need to understand a little matrix algebra. Specifically, you should be familiar with matrix addition, matrix subtraction, and matrix multiplication. And you should know about matrix transposes and matrix inverses.

## The Regression Equation in Matrix Form

With multiple regression, there is one dependent variable and *k* independent variables. The regression equation is:

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + \ldots + b_{k-1}x_{k-1} + b_kx_k$$

where $\hat{y}$ is the predicted value of the dependent variable, $b_k$ are regression coefficients, and $x_k$ is the value of independent variable *k*.

To express the regression equation in matrix form, we need to define three matrices: **Y**, **b**, and **X**.

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ . \\ . \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} . \\ y_n \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} . \\ b_k \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} 1X_{1,1}X_{1,2}...X_{1,k} \\ 1X_{2,1}X_{2,2}...X_{2,k} \\ . \quad . \quad . \quad ... \quad . \\ . \quad . \quad . \quad ... \quad . \\ . \quad . \quad . \quad ... \quad . \\ 1X_{n,1}X_{n,2}...X_{n,k} \end{bmatrix}$$

Here, the dataset consists of *n* records. Each record includes scores for 1 dependent variable and *k* independent variables. **Y** is an *n* x *1* vector that holds predicted values of the dependent variable; and **b** is a *k + 1* x *1* vector that holds estimated regression coefficients. Matrix **X** has a column of 1's plus *k* columns of values for each independent variable in the regression equation.

Given these matrices, the multiple regression equation can be expressed concisely as:

$$\mathbf{Y} = \mathbf{Xb}$$

It is sort of cool that this simple expression describes the regression equation for 1, 2, 3, or *any* number of independent variables.

## Normal Equations in Matrix Form

Just as the regression equation can be expressed compactly in matrix form, so can the normal equations. The least squares normal equations can be expressed as:

$$\mathbf{X'Y} = \mathbf{X'Xb} \quad \text{or} \quad \mathbf{X'Xb} = \mathbf{X'Y}$$

Here, matrix **X'** is the transpose of matrix **X**. To solve for regression coefficients, simply pre-multiply by the inverse of **X'X**:

$$\mathbf{(X'X)^{-1}X'Xb} = \mathbf{(X'X)^{-1}X'Y}$$

$$\mathbf{b} = \mathbf{(X'X)^{-1}X'Y}$$

where $\mathbf{(X'X)^{-1}X'X = I}$, the identity matrix.

In the real world, you will probably never compute regression coefficients by hand.

## Test Your Understanding

### Problem 1

Consider the table below. It shows three performance measures for five students.

| Student | Test score | IQ | Study hours |
|---|---|---|---|
| 1 | 100 | 110 | 40 |
| 2 | 90 | 120 | 30 |
| 3 | 80 | 100 | 20 |
| 4 | 70 | 90 | 0 |
| 5 | 60 | 80 | 10 |

Using least squares regression, develop a regression equation to predict test score, based on (1) IQ and (2) the number of hours that the student studied.

### Solution

For this problem, we have some raw data; and we want to use this raw data to define a **least-squares regression equation:**

$$\hat{y} = b_0 + b_1x_1 + b_2x_2$$

where $\hat{y}$ is the predicted test score; $b_0$, $b_1$, and $b_2$ are regression coefficients; $x_1$ is an IQ score; and $x_2$ is the number of hours that the student studied.

On the right side of the equation, the only unknowns are the regression coefficients. To define the regression coefficients, we use the following equation:

$$\mathbf{b = (X'X)^{-1}X'Y}$$

To solve this equation, we need to complete the following steps:

- Define **X**.
- Define **X'**.
- Compute **X'X**.
- Find the inverse of **X'X**.
- Define **Y**.

Let's begin with matrix **X**. Matrix **X** has a column of 1's plus two columns of values for each independent variable. So, this is matrix **X** and its transpose **X'**:

$$\mathbf{X} = \begin{bmatrix} 1 & 1\,1\,0 & 40 \\ 1 & 120 & 30 \\ 1 & 100 & 20 \\ 1 & 90 & 0 \\ 1 & 80 & 10 \end{bmatrix}$$

$$\mathbf{X'} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 110 & 120 & 100 & 90 & 80 \\ 40 & 30 & 20 & 0 & 10 \end{bmatrix}$$

Given **X'** and **X**, it is a simple matter to compute **X'X**.

$$\mathbf{X'X} = \begin{bmatrix} 5 & 500 & 100 \\ 500 & 51,000 & 10,800 \\ 100 & 10,800 & 3,000 \end{bmatrix}$$

Ultimately, we find:

$$\mathbf{(X'X)^{-1}} = \begin{bmatrix} 101/5 & -7/30 & 1/6 \\ -7/30 & 1/360 & -1/450 \\ 1/6 & -1/450 & 1/360 \end{bmatrix}$$

Next, we define **Y**, the vector of dependent variable scores. For this problem, it is the vector of test scores.

$$\mathbf{Y} = \begin{bmatrix} 100 \\ 90 \\ 80 \\ 70 \\ 60 \end{bmatrix}$$

With all of the essential matrices defined, we are ready to compute the least squares regression coefficients.

$$\mathbf{b} = (\mathbf{X'X})^{-1}\mathbf{X'Y}$$

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 20 \\ 0.5 \\ 0.5 \end{bmatrix}$$

To conclude, here is our **least-squares regression equation:**

$$\hat{y} = 20 + 0.5x_1 + 0.5x_2$$

where $\hat{y}$ is the predicted test score; $x_1$ is an IQ score; and $x_2$ is the number of hours that the student studied. The regression coefficients are $b_0 = 20$, $b_1 = 0.5$, and $b_2 = 0.5$.

**Exercise 1.** The data regarding the production of wheat in tons (X) and the price of the kilo of flour in pesetas (Y ) in the decade of the 80's in Spain were:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Wheat production | 30 | 28 | 32 | 25 | 25 | 25 | 22 | 24 | 35 | 40 |
| Flour price | 25 | 30 | 27 | 40 | 42 | 40 | 50 | 45 | 30 | 25 |

Fit the Simple regression line using the method of least squares.

Compute the residual variance in Exercise 1

CHAPTER 3

# Multiple Linear Regression

A regression model that involves more than one regressor variable is called a **multiple regression model**. Fitting and analyzing these models is discussed in this chapter. The results are extensions of those in Chapter 2 for simple linear regression.

## 3.1 MULTIPLE REGRESSION MODELS

Suppose that the yield in pounds of conversion in a chemical process depends on temperature and the catalyst concentration. A multiple regression model that might describe this relationship is

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon \tag{3.1}$$

where $y$ denotes the yield, $x_1$ denotes the temperature, and $x_2$ denotes the catalyst concentration. This is a **multiple linear regression model** with two regressor variables. The term **linear** is used because Eq. (3.1) is a linear function of the unknown parameters $\beta_0$, $\beta_1$, and $\beta_2$.

The regression model in Eq. (3.1) describes a plane in the three-dimensional space of $y$, $x_1$, and $x_2$. Figure 3.1*a* shows this regression plane for the model

$$E(y) = 50 + 10x_1 + 7x_2$$

where we have assumed that the expected value of the error term $\varepsilon$ in Eq. (3.1) is zero. The parameter $\beta_0$ is the intercept of the regression plane. If the range of the data includes $x_1 = x_2 = 0$, then $\beta_0$ is the mean of $y$ when $x_1 = x_2 = 0$. Otherwise $\beta_0$ has no physical interpretation. The parameter $\beta_1$ indicates the expected change in response ($y$) per unit change in $x_1$ when $x_2$ is held constant. Similarly $\beta_2$ measures the expected change in $y$ per unit change in $x_2$ when $x_1$ is held constant. Figure 3.1*b* shows a **contour plot** of the regression model, that is, lines of

*Introduction to Linear Regression Analysis, Fourth Edition.*
By Douglas C. Montgomery, Elizabeth A. Peck, and Geoffrey Vining.

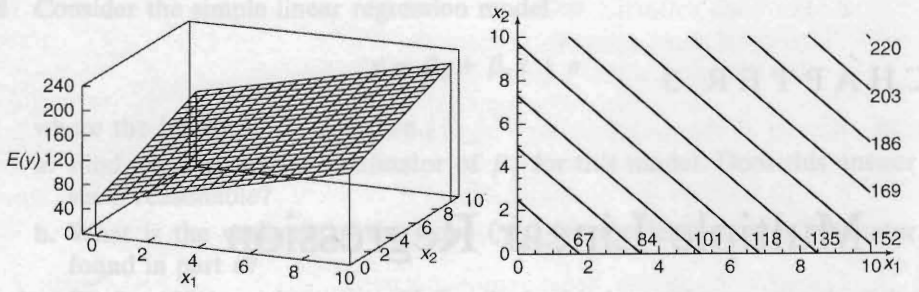



**Figure 3.1** (*a*) The regression plane for the model $E(y) = 50 + 10x_1 + 7x_2$. (*b*) The contour plot.

constant expected response $E(y)$ as a function of $x_1$ and $x_2$. Notice that the contour lines in this plot are parallel straight lines.

In general, the **response** $y$ may be related to $k$ **regressor** or **predictor variables**. The model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon \tag{3.2}$$

is called a **multiple linear regression model** with $k$ regressors. The parameters $\beta_j$, $j = 0, 1, \ldots, k$, are called the **regression coefficients**. This model describes a hyperplane in the $k$-dimensional space of the regressor variables $x_j$. The parameter $\beta_j$ represents the expected change in the response $y$ per unit change in $x_j$ **when all of the remaining regressor variables** $x_i (i \neq j)$ **are held constant**. For this reason the parameters $\beta_j$, $j = 1, 2, \ldots, k$, are often called **partial regression coefficients**.

Multiple linear regression models are often used as **empirical models** or approximating functions. That is, the true functional relationship between $y$ and $x_1, x_2, \ldots, x_k$ is unknown, but over certain ranges of the regressor variables the linear regression model is an adequate approximation to the true unknown function.

Models that are more complex in structure than Eq. (3.2) may often still be analyzed by multiple linear regression techniques. For example, consider the cubic polynomial model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \varepsilon \tag{3.3}$$

If we let $x_1 = x$, $x_2 = x^2$, and $x_3 = x^3$, then Eq. (3.3) can be written as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon \tag{3.4}$$

which is a multiple linear regression model with three regressor variables. Polynomial models will be discussed in more detail in Chapter 7.

Models that include **interaction effects** may also be analyzed by multiple linear regression methods. For example, suppose that the model is

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \varepsilon \tag{3.5}$$

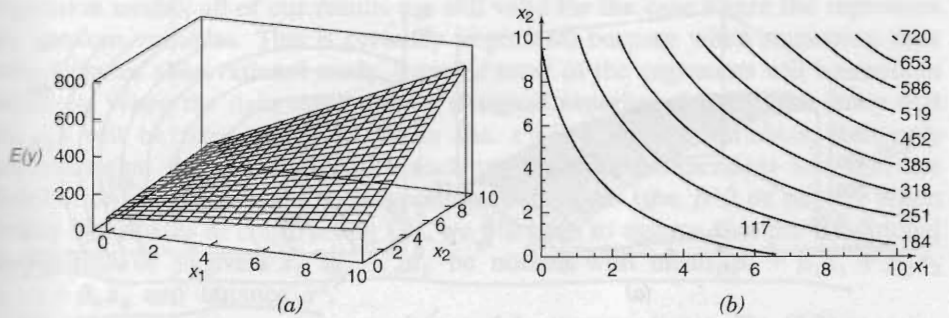



**Figure 3.2** (*a*) Three-dimensional plot of regression model $E(y) = 50 + 10x_1 + 7x_2 + 5x_1x_2$. (*b*) The contour plot.

If we let $x_3 = x_1x_2$ and $\beta_3 = \beta_{12}$, then Eq. (3.5) can be written as

$$y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_3x_3 + \varepsilon \tag{3.6}$$

which is a linear regression model.

Figure 3.2*a* shows the three-dimensional plot of the regression model

$$y = 50 + 10x_1 + 7x_2 + 5x_1x_2$$

and Figure 3.2*b* the corresponding two-dimensional contour plot. Notice that, although this model is a linear regression model, the shape of the surface that is generated by the model is not linear. In general, **any regression model that is linear in the parameters** (the $\beta$'s) **is a linear regression model, regardless of the shape of the surface that it generates.**

Figure 3.2 provides a nice graphical interpretation of an interaction. Generally, interaction implies that the effect produced by changing one variable ($x_1$, say) depends on the level of the other variable ($x_2$). For example, Figure 3.2 shows that changing $x_1$ from 2 to 8 produces a much smaller change in $E(y)$ when $x_2 = 2$ than when $x_2 = 10$. Interaction effects occur frequently in the study and analysis of real-world systems, and regression methods are one of the techniques that we can use to describe them.

As a final example, consider the **second-order model with interaction**

$$y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_{11}x_1^2 + \beta_{22}x_2^2 + \beta_{12}x_1x_2 + \varepsilon \tag{3.7}$$

If we let $x_3 = x_1^2$, $x_4 = x_2^2$, $x_5 = x_1x_2$, $\beta_3 = \beta_{11}$, $\beta_4 = \beta_{22}$, and $\beta_5 = \beta_{12}$, then Eq. (3.7) can be written as a multiple linear regression model as follows:

$$y = \beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_3x_3 + \beta_4x_4 + \beta_5x_5 + \varepsilon$$

Figure 3.3 shows the three-dimensional plot and the corresponding contour plot for

$$E(y) = 800 + 10x_1 + 7x_2 - 8.5x_1^2 - 5x_2^2 + 4x_1x_2$$

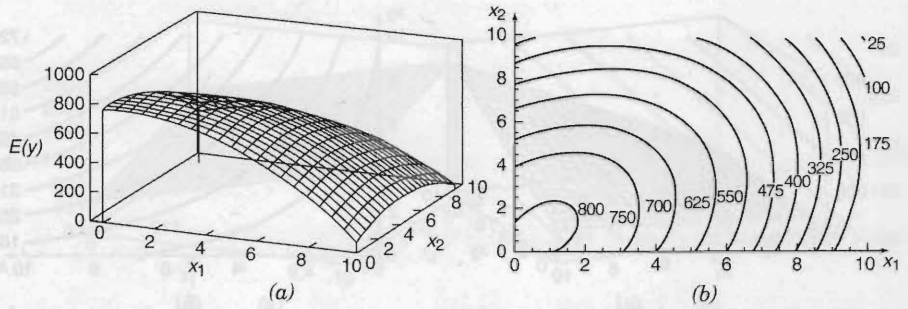



**Figure 3.3** (*a*) Three-dimensional plot of the regression model $E(y) = 800 + 10x_1 + 7x_2 - 8.5x_1^2 - 5x_2^2 + 4x_1x_2$. (*b*) The contour plot.

These plots indicate that the expected change in $y$ when $x_1$ is changed by one unit (say) is a function of **both** $x_1$ and $x_2$. The quadratic and interaction terms in this model produce a mound-shaped function. Depending on the values of the regression coefficients, the second-order model with interaction is capable of assuming a wide variety of shapes; thus, it is a very flexible regression model.

In most real-world problems, the values of the parameters (the regression coefficients $\beta_i$) and the error variance $\sigma^2$ will not be known, and they must be estimated from sample data. The fitted regression equation or model is typically used in prediction of future observations of the response variable $y$ or for estimating the mean response at particular levels of the $y$'s.

## 3.2 ESTIMATION OF THE MODEL PARAMETERS

### 3.2.1 Least-Squares Estimation of the Regression Coefficients

The **method of least squares** can be used to estimate the regression coefficients in Eq. (3.2). Suppose that $n > k$ observations are available, and let $y_i$ denote the $i$th observed response and $x_{ij}$ denote the $i$th observation or level of regressor $x_j$. The data will appear as in Table 3.1. We assume that the error term $\varepsilon$ in the model has $E(\varepsilon) = 0$, $\text{Var}(\varepsilon) = \sigma^2$, and that the errors are uncorrelated.

Throughout this chapter we will assume that the regressor variables $x_1, x_2, \ldots, x_k$ are fixed (i.e., mathematical or nonrandom) variables, measured without error. However, just as was discussed in Section 2.11 for the simple linear

**TABLE 3.1 Data for Multiple Linear Regression**

| Observation, $i$ | Response, $y$ | Regressors | | | |
|---|---|---|---|---|---|
| | | $x_1$ | $x_2$ | $\cdots$ | $x_k$ |
| 1 | $y_1$ | $x_{11}$ | $x_{12}$ | $\cdots$ | $x_{1k}$ |
| 2 | $y_2$ | $x_{21}$ | $x_{22}$ | $\cdots$ | $x_{2k}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ |
| $n$ | $y_n$ | $x_{n1}$ | $x_{n2}$ | $\cdots$ | $x_{nk}$ |

regression model, all of our results are still valid for the case where the regressors are random variables. This is certainly important, because when regression data arise from an **observational study**, some or most of the regressors will be random variables. When the data result from a **designed experiment**, it is more likely that the $x$'s will be fixed variables. When the $x$'s are random variables, it is only necessary that the observations on each regressor be independent and that the distribution not depend on the regression coefficients (the $\beta$'s) or on $\sigma^2$. When testing hypotheses or constructing CIs, we will have to assume that the conditional distribution of $y$ given $x_1, x_2, \ldots, x_k$ be normal with mean $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$ and variance $\sigma^2$.

We may write the sample regression model corresponding to Eq. (3.2) as

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + \varepsilon_i \\ &= \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} + \varepsilon_i, \quad i = 1, 2, \ldots, n \end{aligned} \tag{3.8}$$

The least-squares function is

$$S(\beta_0, \beta_1, \ldots, \beta_k) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{k} \beta_j x_{ij} \right)^2 \tag{3.9}$$

The function $S$ must be minimized with respect to $\beta_0, \beta_1, \ldots, \beta_k$. The least-squares estimators of $\beta_0, \beta_1, \ldots, \beta_k$ must satisfy

$$\left. \frac{\partial S}{\partial \beta_0} \right|_{\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k} = -2 \sum_{i=1}^{n} \left( y_i - \hat{\beta}_0 - \sum_{j=1}^{k} \hat{\beta}_j x_{ij} \right) = 0 \tag{3.10a}$$

and

$$\left. \frac{\partial S}{\partial \beta_j} \right|_{\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k} = -2 \sum_{i=1}^{n} \left( y_i - \hat{\beta}_0 - \sum_{j=1}^{k} \hat{\beta}_j x_{ij} \right) x_{ij} = 0, \quad j = 1, 2, \ldots, k \tag{3.10b}$$

Simplifying Eq. (3.10), we obtain the **least-squares normal equations**

$$\begin{aligned} n\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=1}^{n} x_{i1} + \hat{\beta}_2 \sum_{i=1}^{n} x_{i2} + \cdots + \hat{\beta}_k \sum_{i=1}^{n} x_{ik} &= \sum_{i=1}^{n} y_i \\ \hat{\beta}_0 \sum_{i=1}^{n} x_{i1} + \hat{\beta}_1 \sum_{i=1}^{n} x_{i1}^2 + \hat{\beta}_2 \sum_{i=1}^{n} x_{i1} x_{i2} + \cdots + \hat{\beta}_k \sum_{i=1}^{n} x_{i1} x_{ik} &= \sum_{i=1}^{n} x_{i1} y_i \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ \hat{\beta}_0 \sum_{i=1}^{n} x_{ik} + \hat{\beta}_1 \sum_{i=1}^{n} x_{ik} x_{i1} + \hat{\beta}_2 \sum_{i=1}^{n} x_{ik} x_{i2} + \cdots + \hat{\beta}_k \sum_{i=1}^{n} x_{ik}^2 &= \sum_{i=1}^{n} x_{ik} y_i \end{aligned} \tag{3.11}$$

Note that there are $p = k + 1$ normal equations, one for each of the unknown

regression coefficients. The solution to the normal equations will be the **least-squares estimators** $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k$.

It is more convenient to deal with multiple regression models if they are expressed in matrix notation. This allows a very compact display of the model, data, and results. In matrix notation, the model given by Eq. (3.8) is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

In general, $\mathbf{y}$ is an $n \times 1$ vector of the observations, $\mathbf{X}$ is an $n \times p$ matrix of the levels of the regressor variables, $\boldsymbol{\beta}$ is a $p \times 1$ vector of the regression coefficients, and $\boldsymbol{\varepsilon}$ is an $n \times 1$ vector of random errors.

We wish to find the vector of least-squares estimators, $\hat{\boldsymbol{\beta}}$, that minimizes

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} \varepsilon_i^2 = \boldsymbol{\varepsilon}'\boldsymbol{\varepsilon} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

Note that $S(\boldsymbol{\beta})$ may be expressed as

$$\begin{aligned} S(\boldsymbol{\beta}) &= \mathbf{y}'\mathbf{y} - \boldsymbol{\beta}'\mathbf{X}'\mathbf{y} - \mathbf{y}'\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta} \\ &= \mathbf{y}'\mathbf{y} - 2\boldsymbol{\beta}'\mathbf{X}'\mathbf{y} + \boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta} \end{aligned}$$

since $\boldsymbol{\beta}'\mathbf{X}'\mathbf{y}$ is a $1 \times 1$ matrix, or a scalar, and its transpose $(\boldsymbol{\beta}'\mathbf{X}'\mathbf{y})' = \mathbf{y}'\mathbf{X}\boldsymbol{\beta}$ is the same scalar. The least-squares estimators must satisfy

$$\left.\frac{\partial S}{\partial \boldsymbol{\beta}}\right|_{\hat{\boldsymbol{\beta}}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{0}$$

which simplifies to

$$\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y} \tag{3.12}$$

Equations (3.12) are the **least-squares normal equations**. They are the matrix analogue of the scalar presentation in (3.11).

To solve the normal equations, multiply both sides of (3.12) by the inverse of $\mathbf{X}'\mathbf{X}$. Thus, the **least-squares estimator** of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{3.13}$$

provided that the inverse matrix $(\mathbf{X}'\mathbf{X})^{-1}$ exists. The $(\mathbf{X}'\mathbf{X})^{-1}$ matrix will always exist if the regressors are **linearly independent**, that is, if no column of the $\mathbf{X}$ matrix is a linear combination of the other columns.

It is easy to see that the matrix form of the normal equations (3.12) is identical to the scalar form (3.11). Writing out (3.12) in detail, we obtain

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_{i1} & \sum_{i=1}^{n} x_{i2} & \cdots & \sum_{i=1}^{n} x_{ik} \\ \sum_{i=1}^{n} x_{i1} & \sum_{i=1}^{n} x_{i1}^2 & \sum_{i=1}^{n} x_{i1}x_{i2} & \cdots & \sum_{i=1}^{n} x_{i1}x_{ik} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum_{i=1}^{n} x_{ik} & \sum_{i=1}^{n} x_{ik}x_{i1} & \sum_{i=1}^{n} x_{ik}x_{i2} & \cdots & \sum_{i=1}^{n} x_{ik}^2 \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_{i1}y_i \\ \vdots \\ \sum_{i=1}^{n} x_{ik}y_i \end{bmatrix}$$

If the indicated matrix multiplication is performed, the scalar form of the normal equations (3.11) is obtained. In this display we see that $\mathbf{X}'\mathbf{X}$ is a $p \times p$ symmetric matrix and $\mathbf{X}'\mathbf{y}$ is a $p \times 1$ column vector. Note the special structure of the $\mathbf{X}'\mathbf{X}$ matrix. The diagonal elements of $\mathbf{X}'\mathbf{X}$ are the sums of squares of the elements in the columns of $\mathbf{X}$, and the off-diagonal elements are the sums of cross products of the elements in the columns of $\mathbf{X}$. Furthermore, note that the elements of $\mathbf{X}'\mathbf{y}$ are the sums of cross products of the columns of $\mathbf{X}$ and the observations $y_i$.

The fitted regression model corresponding to the levels of the regressor variables $\mathbf{x}' = [1, x_1, x_2, \ldots, x_k]$ is

$$\hat{y} = \mathbf{x}'\hat{\boldsymbol{\beta}} = \hat{\beta}_0 + \sum_{j=1}^{k} \hat{\beta}_j x_j$$

The vector of fitted values $\hat{y}_i$ corresponding to the observed values $y_i$ is

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} = \mathbf{H}\mathbf{y} \tag{3.14}$$

The $n \times n$ matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ is usually called the **hat matrix**. It maps the vector of observed values into a vector of fitted values. The hat matrix and its properties play a central role in regression analysis.

The difference between the observed value $y_i$ and the corresponding fitted value $\hat{y}_i$ is the **residual** $e_i = y_i - \hat{y}_i$. The $n$ residuals may be conveniently written in matrix notation as

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} \tag{3.15a}$$

There are several other ways to express the vector of residuals $\mathbf{e}$ that will prove useful, including

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{y} - \mathbf{H}\mathbf{y} = (\mathbf{I} - \mathbf{H})\mathbf{y} \tag{3.15b}$$

**Example 3.1 The Delivery Time Data**

A soft drink bottler is analyzing the vending machine service routes in his distribution system. He is interested in predicting the amount of time required by the route driver to service the vending machines in an outlet. This service activity includes stocking the machine with beverage products and minor maintenance or housekeeping. The industrial engineer responsible for the study has suggested that the two most important variables affecting the delivery time ($y$) are the number of cases of product stocked ($x_1$) and the distance walked by the route driver ($x_2$). The engineer has collected 25 observations on delivery time, which are shown in Table 3.2. (Note that this is an expansion of the data set used in Example 2.9.) We will fit the multiple linear regression model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

to the delivery time data in Table 3.2.

**TABLE 3.2 Delivery Time Data for Example 3.1**

| Observation Number | Delivery Time, $y$ (min) | Number of Cases, $x_1$ | Distance, $x_2$ (ft) |
|---|---|---|---|
| 1 | 16.68 | 7 | 560 |
| 2 | 11.50 | 3 | 220 |
| 3 | 12.03 | 3 | 340 |
| 4 | 14.88 | 4 | 80 |
| 5 | 13.75 | 6 | 150 |
| 6 | 18.11 | 7 | 330 |
| 7 | 8.00 | 2 | 110 |
| 8 | 17.83 | 7 | 210 |
| 9 | 79.24 | 30 | 1460 |
| 10 | 21.50 | 5 | 605 |
| 11 | 40.33 | 16 | 688 |
| 12 | 21.00 | 10 | 215 |
| 13 | 13.50 | 4 | 255 |
| 14 | 19.75 | 6 | 462 |
| 15 | 24.00 | 9 | 448 |
| 16 | 29.00 | 10 | 776 |
| 17 | 15.35 | 6 | 200 |
| 18 | 19.00 | 7 | 132 |
| 19 | 9.50 | 3 | 36 |
| 20 | 35.10 | 17 | 770 |
| 21 | 17.90 | 10 | 140 |
| 22 | 52.32 | 26 | 810 |
| 23 | 18.75 | 9 | 450 |
| 24 | 19.83 | 8 | 635 |
| 25 | 10.75 | 4 | 150 |

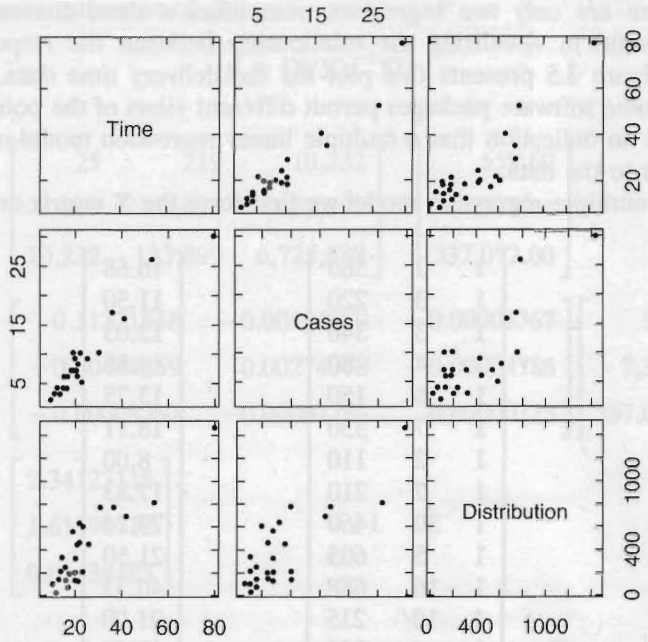



**Figure 3.4** Scatterplot matrix for the delivery time data from Example 3.1.

Graphics can be very useful in fitting multiple regression models. Figure 3.4 is a **scatterplot matrix** of the delivery time data. This is just a two-dimensional array of two-dimensional plots, where (except for the diagonal) each frame contains a scatter diagram. Thus, each plot is an attempt to shed light on the relationship between a pair of variables. This is often a better summary of the relationships than a numerical summary (such as displaying the correlation coefficients between each pair of variables) because it gives a sense of linearity or nonlinearity of the relationship and some awareness of how the individual data points are arranged over the region.

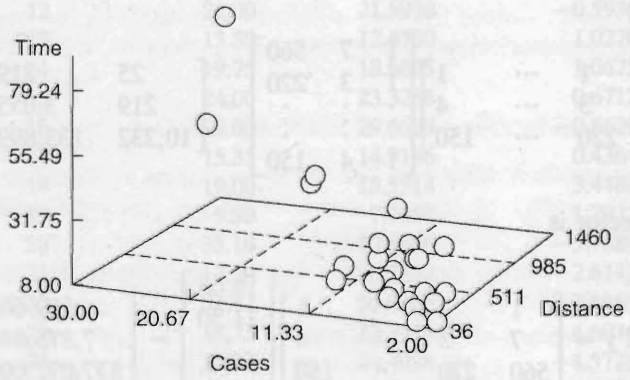



**Figure 3.5** Three-dimensional scatterplot of the delivery time data from Example 3.1.

When there are only two regressors, sometimes a three-dimensional scatter diagram is useful in visualizing the relationship between the response and the regressors. Figure 3.5 presents this plot for the delivery time data. By spinning these plots, some software packages permit different views of the point cloud. This view provides an indication that a multiple linear regression model may provide a reasonable fit to the data.

To fit the multiple regression model we first form the $\mathbf{X}$ matrix and $\mathbf{y}$ vector:

$$\mathbf{X} = \begin{bmatrix} 1 & 7 & 560 \\ 1 & 3 & 220 \\ 1 & 3 & 340 \\ 1 & 4 & 80 \\ 1 & 6 & 150 \\ 1 & 7 & 330 \\ 1 & 2 & 110 \\ 1 & 7 & 210 \\ 1 & 30 & 1460 \\ 1 & 5 & 605 \\ 1 & 16 & 688 \\ 1 & 10 & 215 \\ 1 & 4 & 255 \\ 1 & 6 & 462 \\ 1 & 9 & 448 \\ 1 & 10 & 776 \\ 1 & 6 & 200 \\ 1 & 7 & 132 \\ 1 & 3 & 36 \\ 1 & 17 & 770 \\ 1 & 10 & 140 \\ 1 & 26 & 810 \\ 1 & 9 & 450 \\ 1 & 8 & 635 \\ 1 & 4 & 150 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 16.68 \\ 11.50 \\ 12.03 \\ 14.88 \\ 13.75 \\ 18.11 \\ 8.00 \\ 17.83 \\ 79.24 \\ 21.50 \\ 40.33 \\ 21.00 \\ 13.50 \\ 19.75 \\ 24.00 \\ 29.00 \\ 15.35 \\ 19.00 \\ 9.50 \\ 35.10 \\ 17.90 \\ 52.32 \\ 18.75 \\ 19.83 \\ 10.75 \end{bmatrix}$$

The $\mathbf{X}'\mathbf{X}$ matrix is

$$\mathbf{X}'\mathbf{X} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 7 & 3 & \cdots & 4 \\ 560 & 220 & \cdots & 150 \end{bmatrix} \begin{bmatrix} 1 & 7 & 560 \\ 1 & 3 & 220 \\ \vdots & \vdots & \vdots \\ 1 & 4 & 150 \end{bmatrix} = \begin{bmatrix} 25 & 219 & 10{,}232 \\ 219 & 3{,}055 & 133{,}899 \\ 10{,}232 & 133{,}899 & 6{,}725{,}688 \end{bmatrix}$$

and the $\mathbf{X}'\mathbf{y}$ vector is

$$\mathbf{X}'\mathbf{y} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 7 & 3 & \cdots & 4 \\ 560 & 220 & \cdots & 150 \end{bmatrix} \begin{bmatrix} 16.68 \\ 11.50 \\ \vdots \\ 10.75 \end{bmatrix} = \begin{bmatrix} 559.60 \\ 7{,}375.44 \\ 337{,}072.00 \end{bmatrix}$$

The least-squares estimator of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

or

$$\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \begin{bmatrix} 25 & 219 & 10{,}232 \\ 219 & 3{,}055 & 133{,}899 \\ 10{,}232 & 133{,}899 & 6{,}725{,}688 \end{bmatrix}^{-1} \begin{bmatrix} 559.60 \\ 7{,}375.44 \\ 337{,}072.00 \end{bmatrix}$$

$$= \begin{bmatrix} 0.11321518 & -0.00444859 & -0.00008367 \\ -0.00444859 & 0.00274378 & -0.00004786 \\ -0.00008367 & -0.00004786 & 0.00000123 \end{bmatrix} \begin{bmatrix} 559.60 \\ 7{,}375.44 \\ 337{,}072.00 \end{bmatrix}$$

$$= \begin{bmatrix} 2.34123115 \\ 1.61590712 \\ 0.01438483 \end{bmatrix}$$

**TABLE 3.3 Observations, Fitted Values, and Residuals for Example 3.1**

| Observation Number | $y_i$ | $\hat{y}_i$ | $e_i = y_i - \hat{y}_i$ |
|---|---|---|---|
| 1 | 16.68 | 21.7081 | −5.0281 |
| 2 | 11.50 | 10.3536 | 1.1464 |
| 3 | 12.03 | 12.0798 | −0.0498 |
| 4 | 14.88 | 9.9556 | 4.9244 |
| 5 | 13.75 | 14.1944 | −0.4444 |
| 6 | 18.11 | 18.3996 | −0.2896 |
| 7 | 8.00 | 7.1554 | 0.8446 |
| 8 | 17.83 | 16.6734 | 1.1566 |
| 9 | 79.24 | 71.8203 | 7.4197 |
| 10 | 21.50 | 19.1236 | 2.3764 |
| 11 | 40.33 | 38.0925 | 2.2375 |
| 12 | 21.00 | 21.5930 | −0.5930 |
| 13 | 13.50 | 12.4730 | 1.0270 |
| 14 | 19.75 | 18.6825 | 1.0675 |
| 15 | 24.00 | 23.3288 | 0.6712 |
| 16 | 29.00 | 29.6629 | −0.6629 |
| 17 | 15.35 | 14.9136 | 0.4364 |
| 18 | 19.00 | 15.5514 | 3.4486 |
| 19 | 9.50 | 7.7068 | 1.7932 |
| 20 | 35.10 | 40.8880 | −5.7880 |
| 21 | 17.90 | 20.5142 | −2.6142 |
| 22 | 52.32 | 56.0065 | −3.6865 |
| 23 | 18.75 | 23.3576 | −4.6076 |
| 24 | 19.83 | 24.4028 | −4.5728 |
| 25 | 10.75 | 10.9626 | −0.2126 |

**TABLE 3.4 MINITAB Output for Soft Drink Time Data**

```
Regression Analysis: Time versus Cases, Distance

The regression equation is
Time = 2.34 + 1.62 cases + 0.0144 Distance

Predictor          Coef      SE Coef      T        P
Constant          2.341        1.097    2.13    0.044
Cases            1.6159       0.1707    9.46    0.000
Distance       0.014385     0.003613    3.98    0.001

S = 3.25947    R-Sq = 96.0%    R-Sq(adj) = 95.6%

Analysis of Variance

Source            DF          SS          MS        F        P
Regression         2      5550.8      2775.4   261.24    0.000
Residual Error    22       233.7        10.6
Total             24      5784.5

Source            DF      Seq SS
Cases              1      5382.4
Distance           1       168.4
```

The least-squares fit (with the regression coefficients reported to five decimals) is

$$\hat{y} = 2.34123 + 1.61591x_1 + 0.01438x_2$$

Table 3.3 shows the observations $y_i$ along with the corresponding fitted values $\hat{y}_i$ and the residuals $e_i$ from this model.

***Computer Output***

Table 3.4 presents a portion of the MINITAB output for the soft drink delivery time data in Example 3.1. While the output format differs from one computer program to another, this display contains the information typically generated. Most of the output in Table 3.4 is a straightforward extension to the multiple regression case of the computer output for simple linear regression. In the next few sections we will provide explanations of this output information.

### 3.2.2 A Geometrical Interpretation of Least Squares

An intuitive geometrical interpretation of least squares is sometimes helpful. We may think of the vector of observations $\mathbf{y}' = [y_1, y_2, \ldots, y_n]$ as defining a vector from the origin to the point $A$ in Figure 3.6. Note that $y_1, y_2, \ldots, y_n$ form the coordinates of an $n$-dimensional sample space. The sample space in Figure 3.6 is three-dimensional.

The $\mathbf{X}$ matrix consists of $p$ $(n \times 1)$ column vectors, for example, $\mathbf{1}$ (a column vector of 1's), $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$. Each of these columns defines a vector from the origin in the sample space. These $p$ vectors form a $p$-dimensional subspace called the

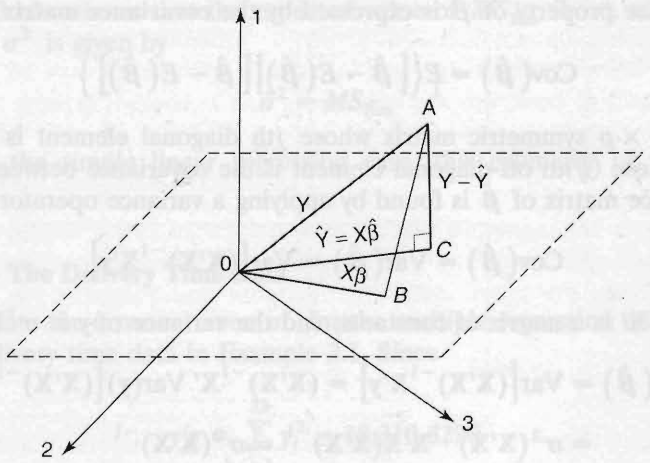



**Figure 3.6** A geometrical interpretation of least squares.

estimation space. The estimation space for $p = 2$ is shown in Figure 3.6. We may represent any point in this subspace by a linear combination of the vectors $\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_k$. Thus, any point in the estimation space is of the form $\mathbf{X}\boldsymbol{\beta}$. Let the vector $\mathbf{X}\boldsymbol{\beta}$ determine the point $B$ in Figure 3.6. The squared distance from $B$ to $A$ is just

$$S(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

Therefore, minimizing the squared distance of point $A$ defined by the observation vector $\mathbf{y}$ to the estimation space requires finding the point in the estimation space that is closest to $A$. The squared distance will be a minimum when the point in the estimation space is the foot of the line from $A$ normal (or perpendicular) to the estimation space. This is point $C$ in Figure 3.6. This point is defined by the vector $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$. Therefore, since $\mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$ is perpendicular to the estimation space, we may write

$$\mathbf{X}'(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = \mathbf{0} \qquad \text{or} \qquad \mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y}$$

which we recognize as the least-squares normal equations.

### 3.2.3 Properties of the Least-Squares Estimators

The statistical properties of the least-squares estimator $\hat{\boldsymbol{\beta}}$ may be easily demonstrated. Consider first bias:

$$\begin{aligned} E(\hat{\boldsymbol{\beta}}) &= E\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}\right] = E\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon})\right] \\ &= E\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}\boldsymbol{\beta} + (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\varepsilon}\right] = \boldsymbol{\beta} \end{aligned}$$

since $E(\boldsymbol{\varepsilon}) = \mathbf{0}$ and $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X} = \mathbf{I}$. Thus, $\hat{\boldsymbol{\beta}}$ is an **unbiased estimator** of $\boldsymbol{\beta}$.

The variance property of $\hat{\boldsymbol{\beta}}$ is expressed by the **covariance matrix**

$$\text{Cov}(\hat{\boldsymbol{\beta}}) = E\left\{[\hat{\boldsymbol{\beta}} - E(\hat{\boldsymbol{\beta}})][\hat{\boldsymbol{\beta}} - E(\hat{\boldsymbol{\beta}})]'\right\}$$

which is a $p \times p$ symmetric matrix whose $j$th diagonal element is the variance of $\hat{\beta}_j$ and whose $(ij)$th off-diagonal element is the covariance between $\hat{\beta}_i$ and $\hat{\beta}_j$. The covariance matrix of $\hat{\boldsymbol{\beta}}$ is found by applying a variance operator to $\hat{\boldsymbol{\beta}}$:

$$\text{Cov}(\hat{\boldsymbol{\beta}}) = \text{Var}(\hat{\boldsymbol{\beta}}) = \text{Var}\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}\right]$$

Now $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ is a matrix of constants, and the variance of $\mathbf{y}$ is $\sigma^2\mathbf{I}$, so

$$\begin{aligned}\text{Var}(\hat{\boldsymbol{\beta}}) &= \text{Var}\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}\right] = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\,\text{Var}(\mathbf{y})\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\right]' \\ &= \sigma^2(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1} = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}\end{aligned}$$

Therefore, if we let $\mathbf{C} = (\mathbf{X}'\mathbf{X})^{-1}$, the variance of $\hat{\beta}_j$ is $\sigma^2 C_{jj}$ and the covariance between $\hat{\beta}_i$ and $\hat{\beta}_j$ is $\sigma^2 C_{ij}$.

Appendix C.4 establishes that the least-squares estimator $\hat{\boldsymbol{\beta}}$ is the best linear unbiased estimator of $\boldsymbol{\beta}$ (the Gauss–Markov theorem). If we further assume that the errors $\varepsilon_i$ are normally distributed, then as we will see in Section 3.2.6, $\hat{\boldsymbol{\beta}}$ is also the maximum-likelihood estimator of $\boldsymbol{\beta}$. The maximum-likelihood estimator is the minimum variance unbiased estimator of $\boldsymbol{\beta}$.

### 3.2.4 Estimation of $\sigma^2$

As in simple linear regression, we may develop an estimator of $\sigma^2$ from the residual sum of squares

$$SS_{\text{Res}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} e_i^2 = \mathbf{e}'\mathbf{e}$$

Substituting $\mathbf{e} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$, we have

$$\begin{aligned}SS_{\text{Res}} &= (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})'(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) \\ &= \mathbf{y}'\mathbf{y} - \hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{y} - \mathbf{y}'\mathbf{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} \\ &= \mathbf{y}'\mathbf{y} - 2\hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{y} + \hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}}\end{aligned}$$

Since $\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y}$, this last equation becomes

$$SS_{\text{Res}} = \mathbf{y}'\mathbf{y} - \hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{y} \tag{3.16}$$

Appendix C.3 shows that the residual sum of squares has $n - p$ degrees of freedom associated with it since $p$ parameters are estimated in the regression model. The **residual mean square** is

$$MS_{\text{Res}} = \frac{SS_{\text{Res}}}{n - p} \tag{3.17}$$

Appendix C.3 also shows that the expected value of $MS_{\text{Res}}$ is $\sigma^2$, so an **unbiased estimator** of $\sigma^2$ is given by

$$\hat{\sigma}^2 = MS_{\text{Res}} \tag{3.18}$$

As noted in the simple linear regression case, this estimator of $\sigma^2$ is **model dependent**.

**Example 3.2 The Delivery Time Data**

We will estimate the error variance $\sigma^2$ for the multiple regression model fit to the soft drink delivery time data in Example 3.1. Since

$$\mathbf{y}'\mathbf{y} = \sum_{i=1}^{25} y_i^2 = 18{,}310.6290$$

and

$$\hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{y} = \begin{bmatrix} 2.34123115 & 1.61590721 & 0.01438483 \end{bmatrix} \begin{bmatrix} 559.60 \\ 7{,}375.44 \\ 337{,}072.00 \end{bmatrix}$$

$$= 18{,}076.90304$$

the residual sum of squares is

$$SS_{\text{Res}} = \mathbf{y}'\mathbf{y} - \hat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{y}$$

$$= 18{,}310.6290 - 18{,}076.9030 = 233.7260$$

Therefore, the estimate of $\sigma^2$ is the residual mean square

$$\hat{\sigma}^2 = \frac{SS_{\text{Res}}}{n-p} = \frac{233.7260}{25-3} = 10.6239$$

The MINITAB output in Table 3.4 reports the residual mean square as 10.6.

The model-dependent nature of this estimate $\sigma^2$ may be easily demonstrated. Figure 2.13 displays the computer output from a least-squares fit to the delivery time data using only one regressor, cases ($x_1$). The residual mean square for this model is 17.5, which is considerably larger than the result obtained above for the two-regressor model. Which estimate is "correct"? Both estimates are in a sense correct, but they depend heavily on the choice of model. Perhaps a better question is which **model** is correct? Since $\sigma^2$ is the variance of the errors (the unexplained noise about the regression line), we would usually prefer a model with a small residual mean square to a model with a large one.

### 3.2.5 Inadequacy of Scatter Diagrams in Multiple Regression

We saw in Chapter 2 that the scatter diagram is an important tool in analyzing the relationship between $y$ and $x$ in simple linear regression. We also saw in Example 3.1 that a **matrix of scatterplots** was useful in visualizing the relationship between

| $y$ | $x_1$ | $x_2$ |
|---|---|---|
| 10 | 2 | 1 |
| 17 | 3 | 2 |
| 48 | 4 | 5 |
| 27 | 1 | 2 |
| 55 | 5 | 6 |
| 26 | 6 | 4 |
| 9 | 7 | 3 |
| 16 | 8 | 4 |

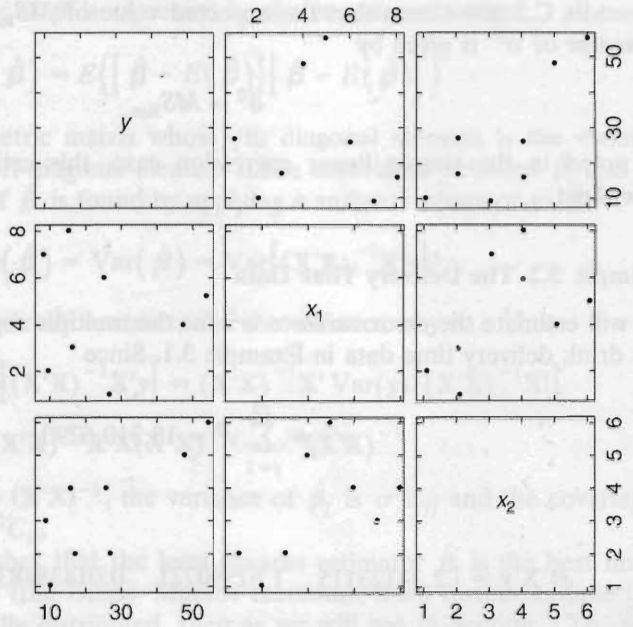



**Figure 3.7** A matrix of scatterplots.

$y$ and two regressors. It is tempting to conclude that this is a general concept; that is, examining scatter diagrams of $y$ versus $x_1$, $y$ versus $x_2, \ldots, y$ versus $x_k$ is always useful in assessing the relationships between $y$ and each of the regressors $x_1, x_2, \ldots, x_k$. Unfortunately, this is not true in general.

Following Daniel and Wood [1980], we illustrate the inadequacy of scatter diagrams for a problem with two regressors. Consider the data shown in Figure 3.7. These data were generated from the equation

$$y = 8 - 5x_1 + 12x_2$$

The matrix of scatterplots is shown in Figure 3.7. The $y$-versus-$x_1$ plot does not exhibit any apparent relationship between the two variables. The $y$-versus-$x_2$ plot indicates that a linear relationship exists, with a slope of approximately 8. Note that both scatter diagrams convey erroneous information. Since in this data set there are two pairs of points that have the same $x_2$ values ($x_2 = 2$ and $x_2 = 4$), we could measure the $x_1$ effect at fixed $x_2$ from both pairs. This gives, $\hat{\beta}_1 = (17 - 27)/(3 - 1) = -5$ for $x_2 = 2$ and $\hat{\beta}_1 = (26 - 16)/(6 - 8) = -5$ for $x_2 = 4$ the correct results. Knowing $\hat{\beta}_1$, we could now estimate the $x_2$ effect. This procedure is not generally useful, however, because many data sets do not have duplicate points.

This example illustrates that constructing scatter diagrams of $y$ versus $x_j$ ($j = 1, 2, \ldots, k$) can be misleading, even in the case of only two regressors operating in a perfectly additive fashion with no noise. A more realistic regression situation with several regressors and error in the $y$'s would confuse the situation even further. If there is only one (or a few) dominant regressor, or if the regressors

operate nearly independently, the matrix of scatterplots is most useful. However, when several important regressors are themselves interrelated, then these scatter diagrams can be very misleading. Analytical methods for sorting out the relationships between several regressors and a response are discussed in Chapter 9.

### 3.2.6 Maximum-Likelihood Estimation

Just as in the simple linear regression case, we can show that the maximum-likelihood estimators for the model parameters in multiple linear regression when the model errors are normally and independently distributed are also least-squares estimators. The model is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

and the errors are normally and independently distributed with constant variance $\sigma^2$, or $\boldsymbol{\varepsilon}$ is distributed as $N(\mathbf{0}, \sigma^2\mathbf{I})$. The normal density function for the errors is

$$f(\varepsilon_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}\varepsilon_i^2\right)$$

The likelihood function is the joint density of $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, or $\prod_{i=1}^{n} f(\varepsilon_i)$. Therefore, the likelihood function is

$$L(\boldsymbol{\varepsilon}, \boldsymbol{\beta}, \sigma^2) = \prod_{i=1}^{n} f(\varepsilon_i) = \frac{1}{(2\pi)^{n/2}\sigma^n} \exp\left(-\frac{1}{2\sigma^2}\boldsymbol{\varepsilon}'\boldsymbol{\varepsilon}\right)$$

Now since we can write $\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$, the likelihood function becomes

$$L(\mathbf{y}, \mathbf{X}, \boldsymbol{\beta}, \sigma^2) = \frac{1}{(2\pi)^{n/2}\sigma^n} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right)$$

As in the simple linear regression case, it is convenient to work with the log of the likelihood,

$$\ln L(\mathbf{y}, \mathbf{X}, \boldsymbol{\beta}, \sigma^2) = -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

It is clear that for a fixed value of $\sigma$ the log-likelihood is maximized when the term

$$(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

is minimized. Therefore, the maximum-likelihood estimator of $\boldsymbol{\beta}$ under normal errors is equivalent to the least-squares estimator $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$. The maximum-likelihood estimator of $\sigma^2$ is

$$\tilde{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})'(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})}{n}$$

These are multiple linear regression generalizations of the results given for simple linear regression in Section 2.10. The statistical properties of the maximum-likelihood estimators are summarized in Section 2.10.

## 3.3 HYPOTHESIS TESTING IN MULTIPLE LINEAR REGRESSION

Once we have estimated the parameters in the model, we face two immediate questions:

1. What is the overall adequacy of the model?
2. Which specific regressors seem important?

Several hypothesis testing procedures prove useful for addressing these questions. The formal tests require that our random errors be independent and follow a normal distribution with mean $E(\varepsilon_i) = 0$ and variance $\text{Var}(\varepsilon_i) = \sigma^2$.

### 3.3.1 Test for Significance of Regression

The test for **significance of regression** is a test to determine if there is a **linear relationship** between the response $y$ and any of the regressor variables $x_1, x_2, \ldots, x_k$. This procedure is often thought of as an overall or global test of model adequacy. The appropriate hypotheses are

$$H_0: \beta_1 = \beta_1 = \cdots = \beta_k = 0$$
$$H_1: \beta_j \neq 0 \quad \text{for at least one } j$$

Rejection of this null hypothesis implies that at least one of the regressors $x_1, x_2, \ldots, x_k$ contributes significantly to the model.

The test procedure is a generalization of the **analysis of variance** used in simple linear regression. The **total sum of squares** $SS_T$ is partitioned into a **sum of squares due to regression**, $SS_R$, and a **residual sum of squares**, $SS_{Res}$. Thus,

$$SS_T = SS_R + SS_{Res}$$

Appendix C.3 shows that if the null hypothesis is true, then $SS_R/\sigma^2$ follows a $\chi^2_k$ distribution, which has the same number of degrees of freedom as number of regressor variables in the model. Appendix C.3 also shows that $SS_{Res}/\sigma^2 \sim \chi^2_{n-k-1}$ and that $SS_{Res}$ and $SS_R$ are independent. By the definition of an $F$ statistic given in Appendix C.1,

$$F_0 = \frac{SS_R/k}{SS_{Res}/(n-k-1)} = \frac{MS_R}{MS_{Res}}$$

follows the $F_{k,n-k-1}$ distribution. Appendix C.3 shows that

$$E(MS_{Res}) = \sigma^2$$
$$E(MS_R) = \sigma^2 + \frac{\boldsymbol{\beta}^{*\prime}\mathbf{X}_c'\mathbf{X}_c\boldsymbol{\beta}^*}{k\sigma^2}$$

# Concept Learning

- Inducing general functions from specific training examples is a main issue of machine learning.
- **Concept Learning:** Acquiring the definition of a general category from given sample positive and negative training examples of the category.
- *Concept Learning* can seen as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples.
- The hypothesis space has a *general-to-specific ordering* of hypotheses, and the search can be efficiently organized by taking advantage of a naturally occurring structure over the hypothesis space.

# Concept Learning

- A Formal Definition for Concept Learning:

  **Inferring a boolean-valued function from training examples of its input and output.**

- An example for concept-learning is the learning of bird-concept from the given examples of birds (positive examples) and non-birds (negative examples).
- We are trying to learn the definition of a concept from given examples.

# A Concept Learning Task – Enjoy Sport
# Training Examples

| Example | **Sky** | **AirTemp** | **Humidity** | **Wind** | **Water** | **Forecast** | **EnjoySport** |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | **YES** |
| 2 | Sunny | Warm | High | Strong | Warm | Same | **YES** |
| 3 | Rainy | Cold | High | Strong | Warm | Change | **NO** |
| 4 | Sunny | Warm | High | Strong | Warm | Change | **YES** |

**ATTRIBUTES** **CONCEPT**

- A set of example days, and each is described by six attributes.
- The task is to learn to predict the value of EnjoySport for arbitrary day, based on the values of its attribute values.

# EnjoySport – Hypothesis Representation

- **Each hypothesis consists of a conjuction of constraints on the instance attributes**.
- Each hypothesis will be a vector of six constraints, specifying the values of the six attributes
  - (Sky, AirTemp, Humidity, Wind, Water, and Forecast).
- Each attribute will be:

  **?** - indicating any value is acceptable for the attribute (**don't care**)

  **single value** – specifying a single required value (ex. Warm) (**specific**)

  **0** - indicating no value is acceptable for the attribute (**no value**)

# Hypothesis Representation

- A hypothesis:

  *Sky* *AirTemp* *Humidity* *Wind* *Water* *Forecast*

  < Sunny, ? , ? , Strong , ? , Same >

- *The most general hypothesis* – that every day is a positive example

  <?, ?, ?, ?, ?, ?>

- *The most specific hypothesis* – that no day is a positive example

  <0, 0, 0, 0, 0, 0>

- *EnjoySport concept learning task* requires learning the sets of days for which EnjoySport=yes, describing this set by a conjunction of constraints over the instance attributes.

# EnjoySport Concept Learning Task

**Given**

- ***Instances X*** : set of all possible days, each described by the attributes
  - Sky – (values: Sunny, Cloudy, Rainy)
  - AirTemp – (values: Warm, Cold)
  - Humidity – (values: Normal, High)
  - Wind – (values: Strong, Weak)
  - Water – (values: Warm, Cold)
  - Forecast – (values: Same, Change)
- ***Target Concept (Function) c*** : EnjoySport : X → {0,1}
- ***Hypotheses H*** : Each hypothesis is described by a conjunction of constraints on the attributes.
- ***Training Examples D*** : positive and negative examples of the target function

**Determine**

- A hypothesis h in H such that h(x) = c(x) for all x in D.

# The Inductive Learning Hypothesis

- Although the learning task is to determine a hypothesis h identical to the target concept cover the entire set of instances X, the only information available about c is its value over the training examples.
  - Inductive learning algorithms can at best guarantee that the output hypothesis fits the target concept over the training data.
  - Lacking any further information, our assumption is that the best hypothesis regarding unseen instances is the hypothesis that best fits the observed training data. This is the fundamental assumption of inductive learning.

- **The Inductive Learning Hypothesis -** Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

# Concept Learning As Search

- Concept learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation.
- The goal of this search is to find the hypothesis that best fits the training examples.
- By selecting a hypothesis representation, the designer of the learning algorithm implicitly defines the space of all hypotheses that the program can ever represent and therefore can ever learn.

# Enjoy Sport - Hypothesis Space

- Sky has 3 possible values, and other 5 attributes have 2 possible values.
- There are 96 (= 3.2.2.2.2.2) distinct instances in X.
- There are 5120 (=5.4.4.4.4.4) *syntactically distinct hypotheses* in H.
  - Two more values for attributes: ? and 0
- Every hypothesis containing one or more 0 symbols represents the empty set of instances; that is, it classifies every instance as negative.
- There are 973 (= 1 + 4.3.3.3.3.3) *semantically distinct hypotheses* in H.
  - Only one more value for attributes: ?, and one hypothesis representing empty set of instances.
- Although EnjoySport has small, finite hypothesis space, most learning tasks have much larger (even infinite) hypothesis spaces.
  - We need efficient search algorithms on the hypothesis spaces.

# General-to-Specific Ordering of Hypotheses

- Many algorithms for concept learning organize the search through the hypothesis space by relying on a *general-to-specific ordering of hypotheses*.
- By taking advantage of this naturally occurring structure over the hypothesis space, we can design learning algorithms that exhaustively search even infinite hypothesis spaces without explicitly enumerating every hypothesis.

- Consider two hypotheses

  h1 = (Sunny, ?, ?, Strong, ?, ?)
  h2 = (Sunny, ?, ?, ?, ?, ?)

- Now consider the sets of instances that are classified positive by hl and by h2.
  - Because h2 imposes fewer constraints on the instance, it classifies more instancesas positive.
  - In fact, any instance classified positive by hl will also be classified positive by h2.
  - Therefore, we say that h2 is *more general than* hl.

# More-General-Than Relation

- For any instance x in X and hypothesis h in H, we say that x satisfies h if and only if h(x) = 1.

- **More-General-Than-Or-Equal Relation:**

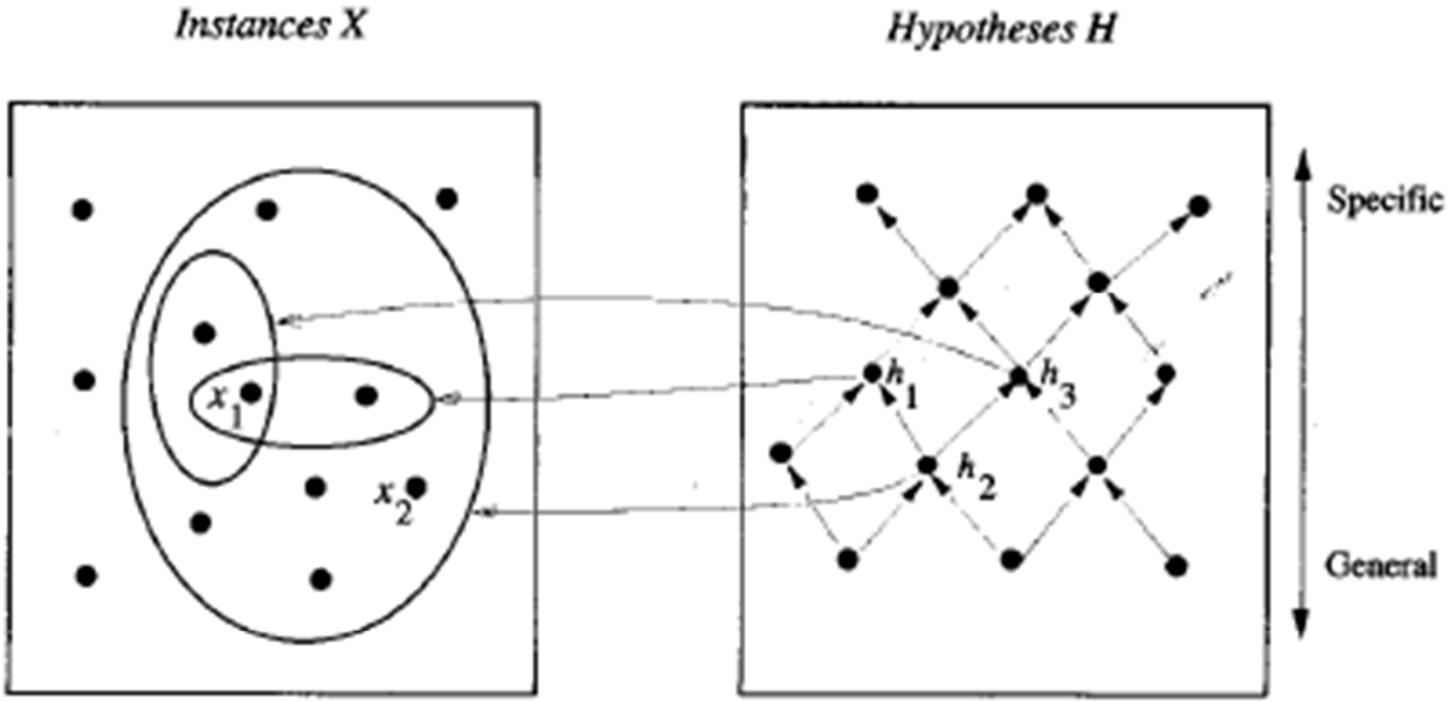
  Let h1 and h2 be two boolean-valued functions defined over X.

  Then h1 is ***more-general-than-or-equal-to*** h2 (written h1 $\geq$ h2)

  if and only if any instance that satisfies h2 also satisfies h1.

- h1 is ***more-general-than*** h2 ( h1 > h2) if and only if h1$\geq$h2 is true and h2$\geq$h1 is false. We also say h2 is ***more-specific-than*** h1.

# More-General-Relation




- h2 > h1 and h2 > h3
- But there is no more-general relation between h1 and h3

# FIND-S Algorithm

- FIND-S Algorithm starts from the most specific hypothesis and generalize it by considering only positive examples.
- FIND-S algorithm ignores negative examples.
  - As long as the hypothesis space contains a hypothesis that describes the true target concept, and the training data contains no errors, ignoring negative examples does not cause to any problem.
- FIND-S algorithm finds the most specific hypothesis within H that is consistent with the positive training examples.
  - The final hypothesis will also be consistent with negative examples if the correct target concept is in H, and the training examples are correct.

# FIND-S Algorithm

1. Initialize h to the most specific hypothesis in H

2. For each positive training instance x

   For each attribute constraint a, in h

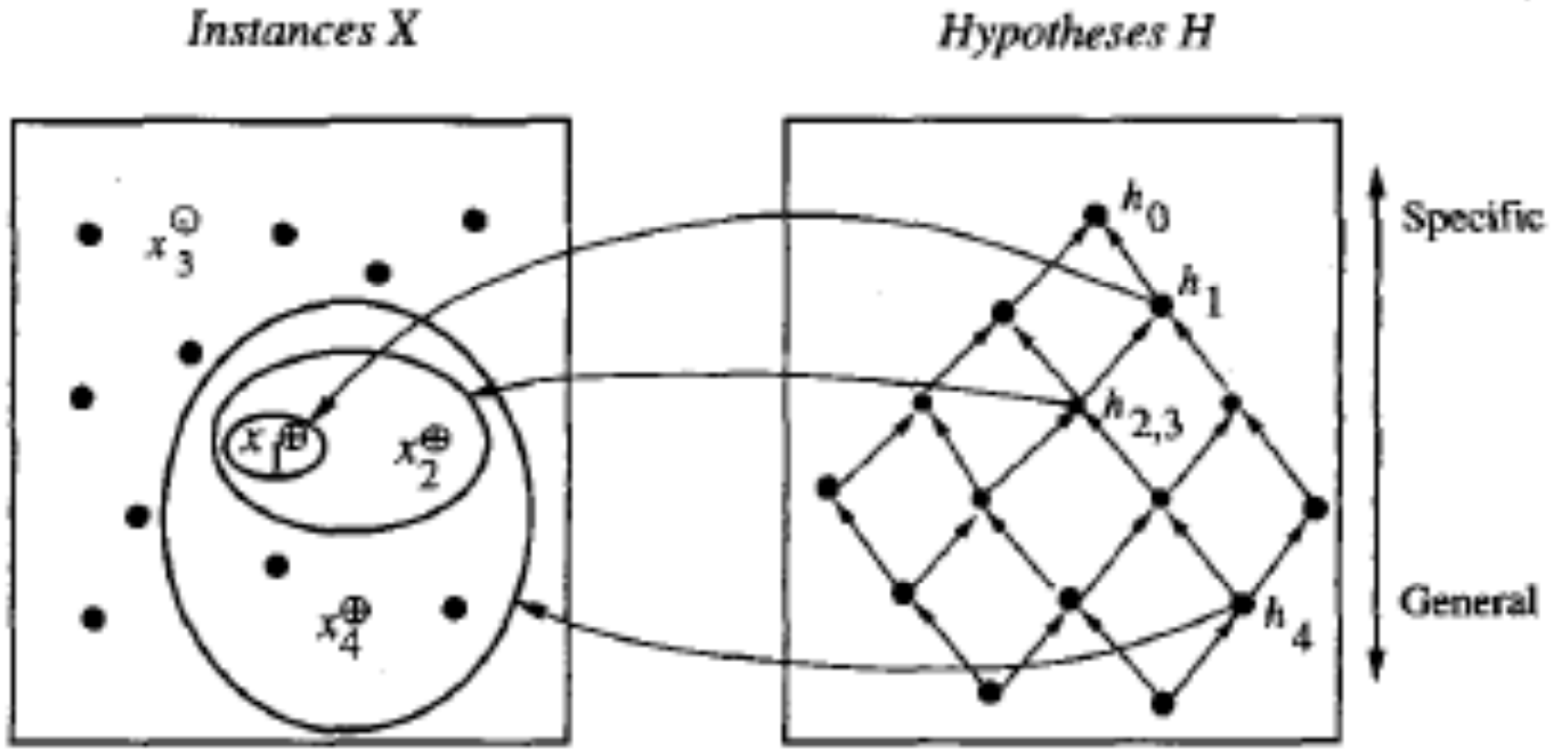
      If the constraint a, is satisfied by x

      Then do nothing

      Else replace a, in h by the next more general constraint that is satisfied by x

3. Output hypothesis h

# FIND-S Algorithm - Example




$x_1$ = <*Sunny Warm Normal Strong Warm Same*>, +

$x_2$ = <*Sunny Warm High Strong Warm Same*>, +

$x_3$ = <*Rainy Cold High Strong Warm Change*>, -

$x_4$ = <*Sunny Warm High Strong Cool Change*>, +

$h_0$ = <∅, ∅, ∅, ∅, ∅, ∅>

$h_1$ = <*Sunny Warm Normal Strong Warm Same*>

$h_2$ = <*Sunny Warm ? Strong Warm Same*>

$h_3$ = <*Sunny Warm ? Strong Warm Same*>

$h_4$ = <*Sunny Warm ? Strong ? ?*>

# Unanswered Questions by FIND-S Algorithm

- Has FIND-S converged to the correct target concept?
  - Although FIND-S will find a hypothesis consistent with the training data, it has no way to determine whether it has found the only hypothesis in H consistent with the data (i.e., the correct target concept), or whether there are many other consistent hypotheses as well.
  - We would prefer a learning algorithm that could determine whether it had converged and, if not, at least characterize its uncertainty regarding the true identity of the target concept.

- Why prefer the most specific hypothesis?
  - In case there are multiple hypotheses consistent with the training examples, FIND-S will find the most specific.
  - It is unclear whether we should prefer this hypothesis over, say, the most general, or some other hypothesis of intermediate generality.

# Unanswered Questions by FIND-S Algorithm

- Are the training examples consistent?
  - In most practical learning problems there is some chance that the training examples will contain at least some errors or noise.
  - Such inconsistent sets of training examples can severely mislead FIND-S, given the fact that it ignores negative examples.
  - We would prefer an algorithm that could at least detect when the training data is inconsistent and, preferably, accommodate such errors.
- What if there are several maximally specific consistent hypotheses?
  - In the hypothesis language H for the EnjoySport task, there is always a unique, most specific hypothesis consistent with any set of positive examples.
  - However, for other hypothesis spaces there can be several maximally specific hypotheses consistent with the data.
  - In this case, FIND-S must be extended to allow it to backtrack on its choices of how to generalize the hypothesis, to accommodate the possibility that the target concept lies along a different branch of the partial ordering than the branch it has selected.

# Candidate-Elimination Algorithm

- FIND-S outputs a hypothesis from H, that is consistent with the training examples, this is just one of many hypotheses from H that might fit the training data equally well.
- The key idea in the Candidate-Elimination algorithm is to output a description of the set of all hypotheses consistent with the training examples.
  - Candidate-Elimination algorithm computes the description of this set without explicitly enumerating all of its members.
  - This is accomplished by using the more-general-than partial ordering and maintaining a compact representation of the set of consistent hypotheses.

# Consistent Hypothesis

A hypothesis $h$ is **consistent** with a set of training examples $D$ of target concept $c$ if and only if $h(x) = c(x)$ for each training example $\langle x, c(x) \rangle$ in $D$.

$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D)\ h(x) = c(x)$$

- The key difference between this definition of ***consistent*** and ***satisfies.***
- An example *x* is said to ***satisfy*** hypothesis ***h*** when ***h(x)*** = 1, regardless of whether x is a positive or negative example of the target concept.
- However, whether such an example is ***consistent*** with ***h*** depends on the target concept, and in particular, whether ***h(x)*** = ***c(x).***

# Version Spaces

- The Candidate-Elimination algorithm represents the set of all hypotheses consistent with the observed training examples.
- This subset of all hypotheses is called the ***version space*** with respect to the hypothesis space H and the training examples D, because it contains all plausible versions of the target concept.

The **version space**, $VS_{H,D}$, with respect to hypothesis space $H$ and training examples $D$, is the subset of hypotheses from $H$ consistent with all training examples in $D$.

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

# List-Then-Eliminate Algorithm

- List-Then-Eliminate algorithm initializes the version space to contain all hypotheses in H, then eliminates any hypothesis found inconsistent with any training example.
- The version space of candidate hypotheses thus shrinks as more examples are observed, until ideally just one hypothesis remains that is consistent with all the observed examples.
  - Presumably, this is the desired target concept.
  - If insufficient data is available to narrow the version space to a single hypothesis, then the algorithm can output the entire set of hypotheses consistent with the observed data.
- List-Then-Eliminate algorithm can be applied whenever the hypothesis space H is finite.
  - It has many advantages, including the fact that it is guaranteed to output all hypotheses consistent with the training data.
  - Unfortunately, it requires exhaustively enumerating all hypotheses in H - an unrealistic requirement for all but the most trivial hypothesis spaces.

# List-Then-Eliminate Algorithm

1. $VersionSpace \leftarrow$ a list containing every hypothesis in $H$
2. For each training example, $\langle x, c(x) \rangle$

   remove from $VersionSpace$ any hypothesis $h$ for which $h(x) \neq c(x)$
3. Output the list of hypotheses in $VersionSpace$

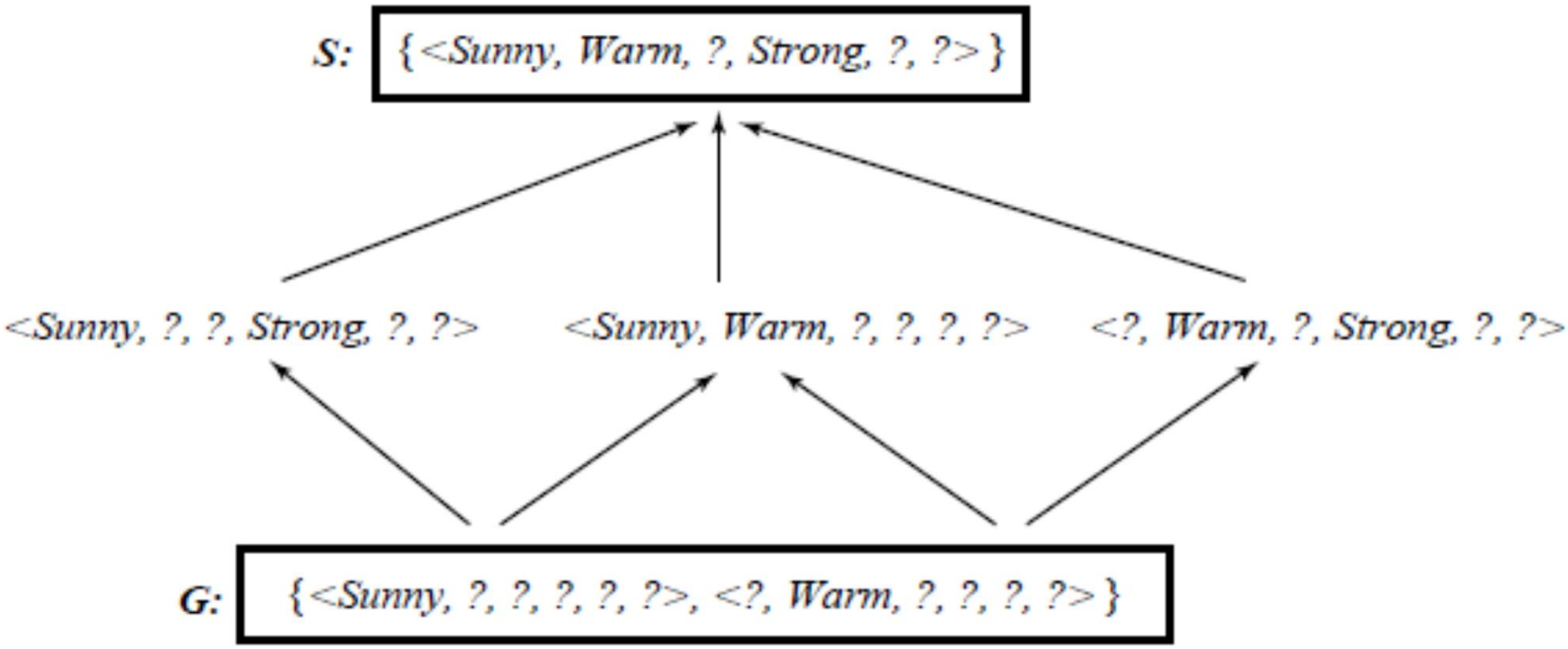
# Compact Representation of Version Spaces

- A version space can be represented with its *general* and *specific boundary sets*.
- The Candidate-Elimination algorithm represents the version space by storing only its most general members G and its most specific members S.
- Given only these two sets S and G, it is possible to enumerate all members of a version space by generating hypotheses that lie between these two sets in general-to-specific partial ordering over hypotheses.
- Every member of the version space lies between these boundaries

$$VS_{H,D} = \{h \in H | (\exists s \in S)(\exists g \in G)(g \geq h \geq s)\}$$

where x ≥y means x is more general or equal to y.

# Example Version Space




- A version space with its general and specific boundary sets.
- The version space includes all six hypotheses shown here, but can be represented more simply by S and ***G.***

# Candidate-Elimination Algorithm

- The Candidate-Elimination algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.
- It begins by initializing the version space to the set of all hypotheses in H; that is, by initializing the G boundary set to contain the most general hypothesis in H

  G0 ← { <?, ?, ?, ?, ?, ?> }

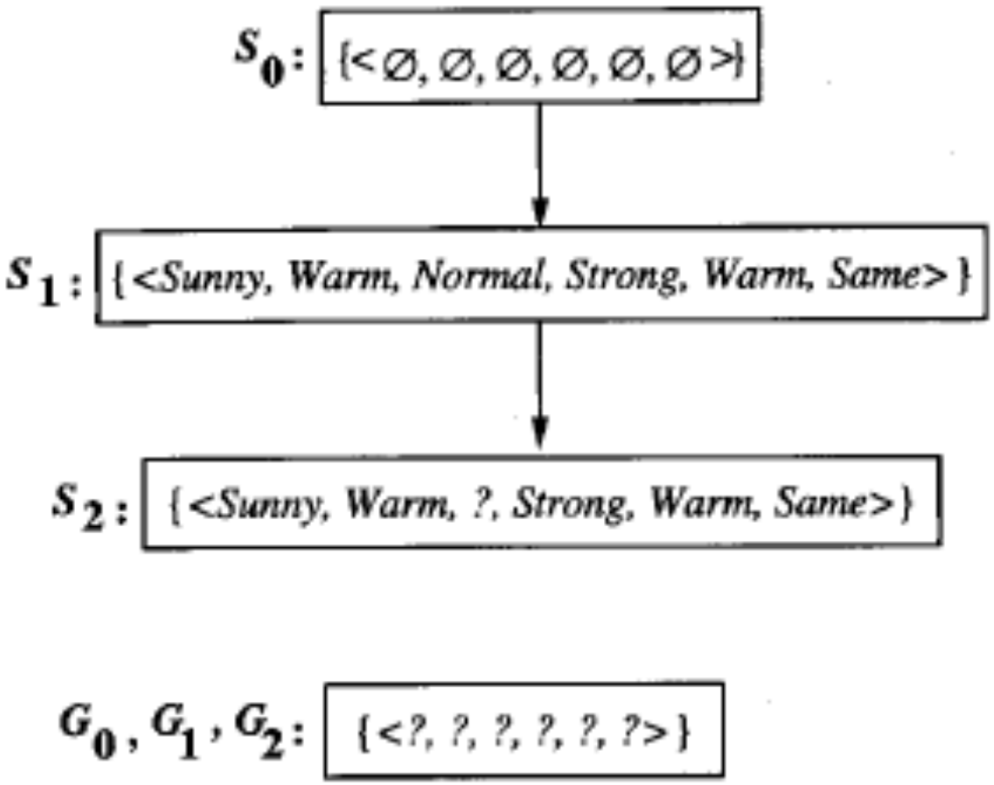
  and initializing the S boundary set to contain the most specific hypothesis

  S0 ← { <0, 0, 0, 0, 0, 0> }
- These two boundary sets delimit the entire hypothesis space, because every other hypothesis in H is both more general than S0 and more specific than G0.
- As each training example is considered, the S and G boundary sets are generalized and specialized, respectively, to eliminate from the version space any hypotheses found inconsistent with the new training example.
- After all examples have been processed, the computed version space contains all the hypotheses consistent with these examples and only these hypotheses.

# Candidate-Elimination Algorithm

- Initialize G to the set of maximally general hypotheses in H
- Initialize S to the set of maximally specific hypotheses in H
- For each training example d, do
  - If d is a positive example
    - Remove from G any hypothesis inconsistent with d ,
    - For each hypothesis s in S that is not consistent with d ,-
      - Remove s from S
      - Add to S all minimal generalizations h of s such that
        - » h is consistent with d, and some member of G is more general than h
      - Remove from S any hypothesis that is more general than another hypothesis in S
  - If d is a negative example
    - Remove from S any hypothesis inconsistent with d
    - For each hypothesis g in G that is not consistent with d
      - Remove g from G
      - Add to G all minimal specializations h of g such that
        - » h is consistent with d, and some member of S is more specific than h
      - Remove from G any hypothesis that is less general than another hypothesis in G

# Candidate-Elimination Algorithm - Example



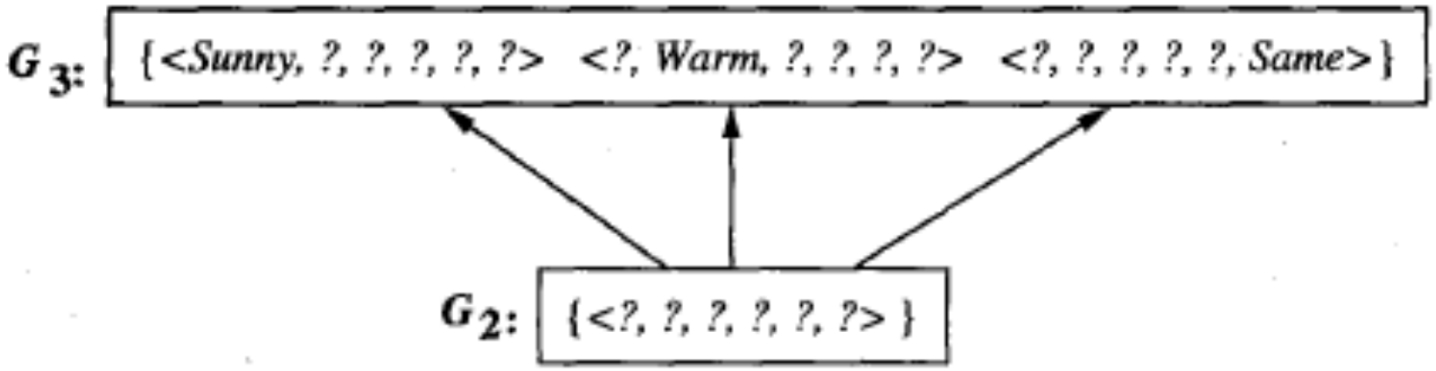

- *S0* and *G0* are the initial boundary sets corresponding to the most specific and most general hypotheses.
- Training examples 1 and 2 force the *S* boundary to become more general.
- They have no effect on the *G* boundary

Training examples:

1. *<Sunny, Warm, Normal, Strong, Warm, Same>, Enjoy Sport = Yes*
2. *<Sunny, Warm, High, Strong, Warm, Same>, Enjoy Sport = Yes*

# Candidate-Elimination Algorithm - Example

$S_2, S_3$: { <Sunny, Warm, ?, Strong, Warm, Same> }



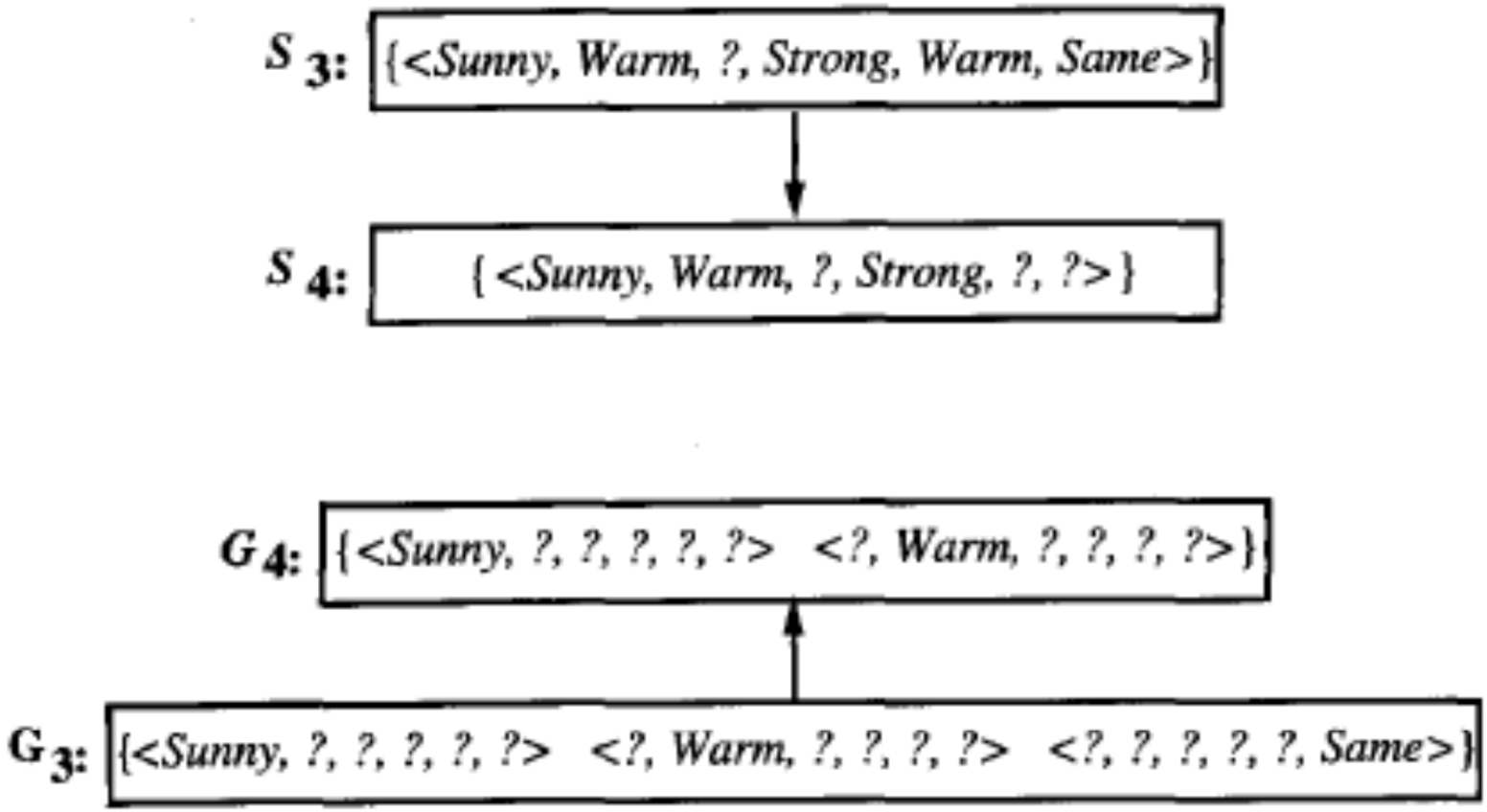

Training Example:

3. <Rainy, Cold, High, Strong, Warm, Change>, EnjoySport=No

# Candidate-Elimination Algorithm - Example

- Given that there are six attributes that could be specified to specialize **G2,** why are there only three new hypotheses in **G3?**
- For example, the hypothesis ***h*** = <?, ?, ***Normal,*** ?, ?, ?> is a minimal specialization of **G2** that correctly labels the new example as a negative example, but it is not included in **G3.**
  - The reason this hypothesis is excluded is that it is inconsistent with S2.
  - The algorithm determines this simply by noting that ***h*** is not more general than the current specific boundary, **S2.**
- In fact, the **S** boundary of the version space forms a summary of the previously encountered positive examples that can be used to determine whether any given hypothesis is consistent with these examples.
- The G boundary summarizes the information from previously encountered negative examples. Any hypothesis more specific than G is assured to be consistent with past negative examples

# Candidate-Elimination Algorithm - Example



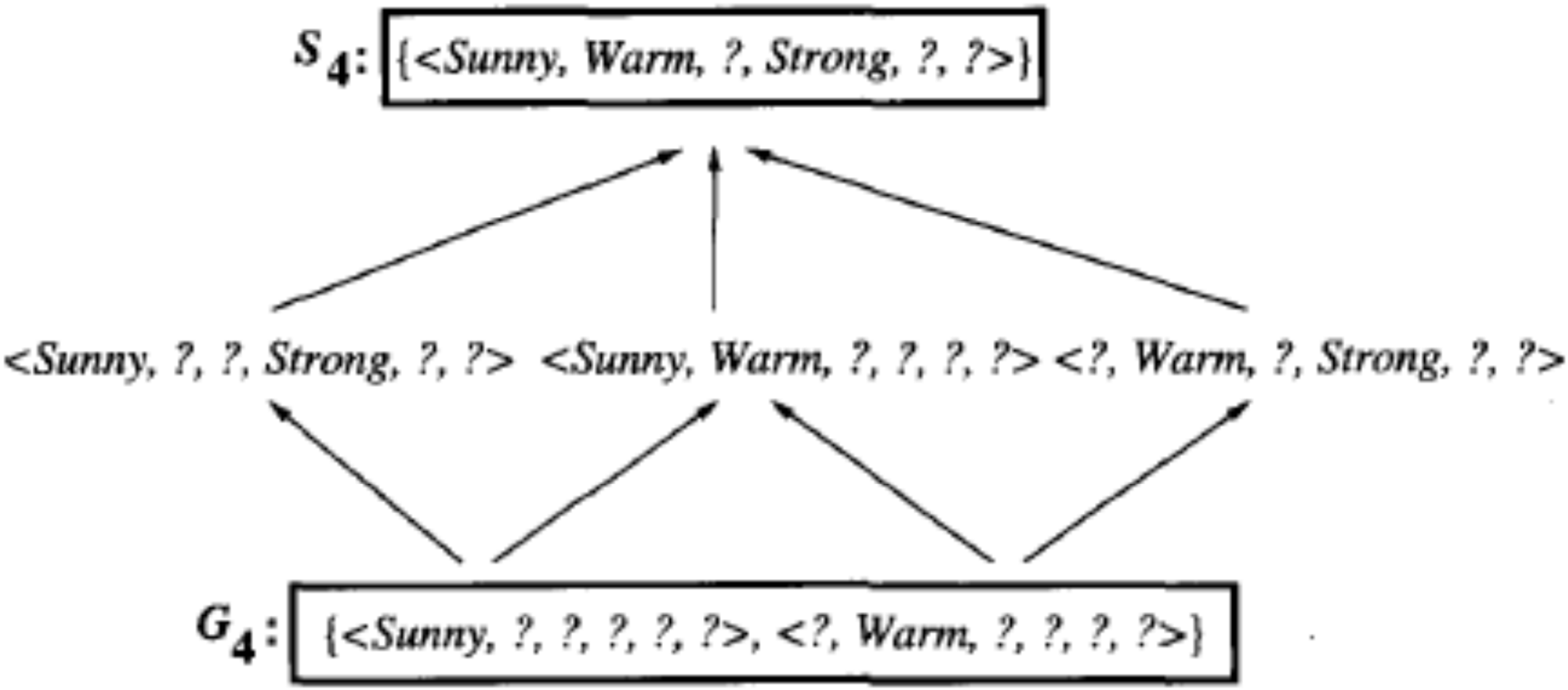

Training Example:

4. <*Sunny, Warm, High, Strong, Cool, Change*>, *EnjoySport* = *Yes*

# Candidate-Elimination Algorithm - Example

- The fourth training example further generalizes the S boundary of the version space.
- It also results in removing one member of the G boundary, because this member fails to cover the new positive example.
  - To understand the rationale for this step, it is useful to consider why the offending hypothesis must be removed from G.
  - Notice it cannot be specialized, because specializing it would not make it cover the new example.
  - It also cannot be generalized, because by the definition of G, any more general hypothesis will cover at least one negative training example.
  - Therefore, the hypothesis must be dropped from the G boundary, thereby removing an entire branch of the partial ordering from the version space of hypotheses remaining under consideration

# Candidate-Elimination Algorithm – Example
# Final Version Space



# Candidate-Elimination Algorithm – Example
# Final Version Space

- After processing these four examples, the boundary sets S4 and G4 delimit the version space of all hypotheses consistent with the set of incrementally observed training examples.
- This learned version space is independent of the sequence in which the training examples are presented (because in the end it contains all hypotheses consistent with the set of examples).
- As further training data is encountered, the S and G boundaries will move monotonically closer to each other, delimiting a smaller and smaller version space of candidate hypotheses.

# Will Candidate-Elimination Algorithm Converge to Correct Hypothesis?

- The version space learned by the Candidate-Elimination Algorithm will converge toward the hypothesis that correctly describes the target concept, provided
  - There are no errors in the training examples, and
  - there is some hypothesis in H that correctly describes the target concept.
- What will happen if the training data contains errors?
  - The algorithm removes the correct target concept from the version space.
  - S and G boundary sets eventually converge to an empty version space if sufficient additional training data is available.
  - Such an empty version space indicates that there is no hypothesis in H consistent with all observed training examples.
- A similar symptom will appear when the training examples are correct, but the target concept cannot be described in the hypothesis representation.
  - e.g., if the target concept is a disjunction of feature attributes and the hypothesis space supports only conjunctive descriptions

# What Training Example Should the Learner Request Next?

- We have assumed that training examples are provided to the learner by some external teacher.
- Suppose instead that the learner is allowed to conduct experiments in which it chooses the next instance, then obtains the correct classification for this instance from an external oracle (e.g., nature or a teacher).
  - This scenario covers situations in which the learner may conduct experiments in nature or in which a teacher is available to provide the correct classification.
  - We use the term query to refer to such instances constructed by the learner, which are then classified by an external oracle.
- Considering the version space learned from the four training examples of the EnjoySport concept.
  - What would be a good query for the learner to pose at this point?
  - What is a good query strategy in general?

# What Training Example Should the Learner Request Next?

- The learner should attempt to discriminate among the alternative competing hypotheses in its current version space.
  - Therefore, it should choose an instance that would be classified positive by some of these hypotheses, but negative by others.
  - One such instance is <Sunny, Warm, Normal, Light, Warm, Same>
  - This instance satisfies three of the six hypotheses in the current version space.
  - If the trainer classifies this instance as a positive example, the S boundary of the version space can then be generalized.
  - Alternatively, if the trainer indicates that this is a negative example, the G boundary can then be specialized.
- In general, the optimal query strategy for a concept learner is to generate instances that satisfy exactly half the hypotheses in the current version space.
- When this is possible, the size of the version space is reduced by half with each new example, and the correct target concept can therefore be found with only $\lceil \log_2 |VS| \rceil$ experiments.

# How Can Partially Learned Concepts Be Used?

- Even though the learned version space still contains multiple hypotheses, indicating that the target concept has not yet been fully learned, it is possible to classify certain examples with the same degree of confidence as if the target concept had been uniquely identified.

- Let us assume that the followings are new instances to be classified:

| Instance | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| A | Sunny | Warm | Normal | Strong | Cool | Change | ? |
| B | Rainy | Cold | Normal | Light | Warm | Same | ? |
| C | Sunny | Warm | Normal | Light | Warm | Same | ? |
| D | Sunny | Cold | Normal | Strong | Warm | Same | ? |

# How Can Partially Learned Concepts Be Used?

- ***Instance A*** was is classified as a positive instance by every hypothesis in the current version space.
- Because the hypotheses in the version space unanimously agree that this is a positive instance, the learner can classify instance A as positive with the same confidence it would have if it had already converged to the single, correct target concept.
- Regardless of which hypothesis in the version space is eventually found to be the correct target concept, it is already clear that it will classify instance A as a positive example.
- Notice furthermore that we need not enumerate every hypothesis in the version space in order to test whether each classifies the instance as positive.
  - This condition will be met if and only if the instance satisfies every member of S.
  - The reason is that every other hypothesis in the version space is at least as general as some member of S.
  - By our definition of more-general-than, if the new instance satisfies all members of S it must also satisfy each of these more general hypotheses.

# How Can Partially Learned Concepts Be Used?

- ***Instance B*** is classified as a negative instance by every hypothesis in the version space.
  - This instance can therefore be safely classified as negative, given the partially learned concept.
  - An efficient test for this condition is that the instance satisfies none of the members of G.
- Half of the version space hypotheses classify ***instance C*** as positive and half classify it as negative.
  - Thus, the learner cannot classify this example with confidence until further training examples are available.
- ***Instance D*** is classified as positive by two of the version space hypotheses and negative by the other four hypotheses.
  - In this case we have less confidence in the classification than in the unambiguous cases of instances A and B.
  - Still, the vote is in favor of a negative classification, and one approach we could take would be to output the majority vote, perhaps with a confidence rating indicating how close the vote was.

# Inductive Bias - Fundamental Questions for Inductive Inference

- The Candidate-Elimination Algorithm will converge toward the true target concept provided it is given accurate training examples and provided its initial hypothesis space contains the target concept.

- What if the target concept is not contained in the hypothesis space?
- Can we avoid this difficulty by using a hypothesis space that includes every possible hypothesis?
- How does the size of this hypothesis space influence the ability of the algorithm to generalize to unobserved instances?
- How does the size of the hypothesis space influence the number of training examples that must be observed?

# Inductive Bias - A Biased Hypothesis Space

- In EnjoySport example, we restricted the hypothesis space to include only conjunctions of attribute values.
  - Because of this restriction, the hypothesis space is unable to represent even simple disjunctive target concepts such as "Sky = Sunny or Sky = Cloudy."

| Example | *Sky* | *AirTemp* | *Humidity* | *Wind* | *Water* | *Forecast* | *EnjoySport* |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Warm | Normal | Strong | Cool | Change | Yes |
| 2 | Cloudy | Warm | Normal | Strong | Cool | Change | Yes |
| 3 | Rainy | Warm | Normal | Strong | Cool | Change | No |

- From first two examples ➔ S2 : <?, Warm, Normal, Strong, Cool, Change>
  - This is inconsistent with third examples, and there are no hypotheses consistent with these three examples

PROBLEM: We have biased the learner to consider only conjunctive hypotheses.
➔ We require a more expressive hypothesis space.

# Inductive Bias - An Unbiased Learner

- The obvious solution to the problem of assuring that the target concept is in the hypothesis space H is to provide a hypothesis space capable of representing every teachable concept.
  - Every possible subset of the instances X ➔ ***the power set of X.***

- What is the size of the hypothesis space H (the power set of X) ?
  - In EnjoySport, the size of the instance space X is 96.
  - The size of the power set of X is $2^{|X|}$ ➔ The size of H is $2^{96}$
  - Our conjunctive hypothesis space is able to represent only 973of these hypotheses.

    ➔ a very biased hypothesis space

# Inductive Bias - An Unbiased Learner : Problem

- Let the hypothesis space H to be the power set of X.
  - A hypothesis can be represented with disjunctions, conjunctions, and negations of our earlier hypotheses.
  - The target concept "Sky = Sunny or Sky = Cloudy" could then be described as

    <Sunny, ?, ?, ?, ?, ?> $\vee$ <Cloudy, ?, ?, ?, ?, ?>

**NEW PROBLEM**: our concept learning algorithm is now completely unable to generalize beyond the observed examples.

- three positive examples (xl,x2,x3) and two negative examples (x4,x5) to the learner.
- S : { x1 $\vee$ x2 $\vee$ x3 } and G : { $\neg$ (x4 $\vee$ x5) } ➔ NO GENERALIZATION
- Therefore, the only examples that will be unambiguously classified by S and G are the observed training examples themselves.

# Inductive Bias – Fundamental Property of Inductive Inference

- **A learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.**

- **Inductive Leap**: A learner should be able to generalize training data using prior assumptions in order to classify unseen instances.
- The generalization is known as **inductive leap** and our prior assumptions are the **inductive bias** of the learner.
- Inductive Bias (prior assumptions) of Candidate-Elimination Algorithm is that the target concept can be represented by a conjunction of attribute values, the target concept is contained in the hypothesis space and training examples are correct.

# Inductive Bias – Formal Definition

**Inductive Bias**:

Consider a concept learning algorithm ***L*** for the set of instances ***X***. Let ***c*** be an arbitrary concept defined over ***X***, and

let ***Dc = {<x , c(x)>}*** be an arbitrary set of training examples of ***c***.

Let ***L(xi, Dc)*** denote the classification assigned to the instance ***xi*** by ***L*** after training on the data ***Dc.***

The **inductive bias** of ***L*** is any minimal set of assertions ***B*** such that for any target concept ***c*** and corresponding training examples ***Dc*** the following formula holds.

$$(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$

# Inductive Bias – Three Learning Algorithms

**ROTE-LEARNER:** Learning corresponds simply to storing each observed training example in memory. Subsequent instances are classified by looking them up in memory. If the instance is found in memory, the stored classification is returned. Otherwise, the system refuses to classify the new instance.

***Inductive Bias:*** No inductive bias

**CANDIDATE-ELIMINATION:** New instances are classified only in the case where all members of the current version space agree on the classification. Otherwise, the system refuses to classify the new instance.

***Inductive Bias:*** the target concept can be represented in its hypothesis space.

**FIND-S:** This algorithm, described earlier, finds the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances.

***Inductive Bias:*** the target concept can be represented in its hypothesis space, and all instances are negative instances unless the opposite is entailed by its other know1edge.

# Concept Learning - Summary

- Concept learning can be seen as a problem of searching through a large predefined space of potential hypotheses.
- The general-to-specific partial ordering of hypotheses provides a useful structure for organizing the search through the hypothesis space.
- The FIND-S algorithm utilizes this general-to-specific ordering, performing a specific-to-general search through the hypothesis space along one branch of the partial ordering, to find the most specific hypothesis consistent with the training examples.
- The CANDIDATE-ELIMINATION algorithm utilizes this general-to-specific ordering to compute the version space (the set of all hypotheses consistent with the training data) by incrementally computing the sets of maximally specific (S) and maximally general (G) hypotheses.

# Concept Learning - Summary

- Because the S and G sets delimit the entire set of hypotheses consistent with the data, they provide the learner with a description of its uncertainty regarding the exact identity of the target concept. This version space of alternative hypotheses can be examined
  - to determine whether the learner has converged to the target concept,
  - to determine when the training data are inconsistent,
  - to generate informative queries to further refine the version space, and
  - to determine which unseen instances can be unambiguously classified based on the partially learned concept.
- The CANDIDATE-ELIMINATION algorithm is not robust to noisy data or to situations in which the unknown target concept is not expressible in the provided hypothesis space.

# Concept Learning - Summary

- Inductive learning algorithms are able to classify unseen examples only because of their implicit inductive bias for selecting one consistent hypothesis over another.
- If the hypothesis space is enriched to the point where there is a hypothesis corresponding to every possible subset of instances (the power set of the instances), this will remove any inductive bias from the CANDIDATE-ELIMINATION algorithm .
  - Unfortunately, this also removes the ability to classify any instance beyond the observed training examples.
  - An unbiased learner cannot make inductive leaps to classify unseen examples.