A

Linear Algebra

This appendix provides a brief introduction to linear algebra with a focus on topics that are relevant to the material in this text. We begin by describing vectors, which can be used to represent both data objects and attributes. We then discuss matrices, which can be used both to represent data sets and to describe transformations on them.

A.1 Vectors

A.1.1 Definition

In Euclidean space, such as the ordinary two- and three-dimensional space with which we are familiar, a **vector** is a quantity that has **magnitude** and **direction**. It is traditionally represented as an arrow that has a length equal to its magnitude and an orientation given by its direction. Figure A.1(a) shows two vectors: vector $\mathbf{u}$, which has a length of 1 and is parallel to the y axis, and vector $\mathbf{v}$, which has a length of 2 and a direction of 45° with respect to the x axis. (We shall use lowercase bold letters, such as $\mathbf{u}$ and $\mathbf{v}$, to represent vectors. They are often also represented by italic lowercase letters, such as u and v.) Since a point can be regarded as a displacement from the origin in a particular direction, it can be represented by a vector from the origin to the point.

A.1.2 Vector Addition and Multiplication by a Scalar

Various operations can be performed on vectors. (In what follows, we assume that the vectors are all from the same space, i.e., have the same dimensionality.) For instance, vectors can be added and subtracted. This is best illustrated

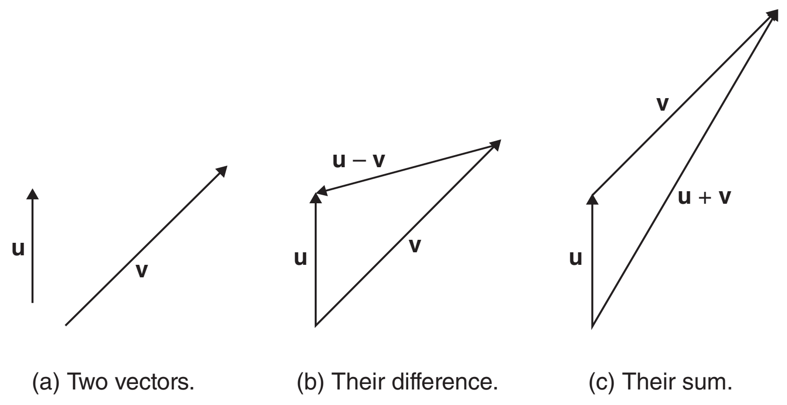


Figure A.1. Two vectors and their sum and difference.

graphically, and vector subtraction and addition are shown in Figures A.1(b) and A.1(c), respectively. Like the addition of numbers, vector addition has some familiar properties. If **u**, **v**, and **w** are three vectors, then these properties can be described as follows:

- **Commutativity of vector addition.** The order of addition does not matter. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$.
- **Associativity of vector addition.** The grouping of vectors during addition does not matter. $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$.
- **Existence of an identity element for vector addition.** There exists a **zero vector**, simply denoted as **0**, which is the identity element. For any vector **u**, $\mathbf{u} + \mathbf{0} = \mathbf{u}$.
- **Existence of additive inverses for vector addition.** For every vector **u**, there is an inverse vector $-\mathbf{u}$ such that $\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$.

Another important operation is the multiplication of a vector by a number, which, in the context of linear algebra, is typically called a **scalar**. Scalar multiplication changes the magnitude of the vector; the direction is unchanged if the scalar is positive and is reversed if the scalar is negative. If **u** and **v** are vectors and α and β are scalars (numbers), then the properties of the scalar multiplication of vectors can be described as follows:

- **Associativity of scalar multiplication.** The order of multiplication by two scalars does not matter. $\alpha(\beta u) = (\alpha\beta)u$.
- **Distributivity of scalar addition over multiplication of a scalar by a vector.** Adding two scalars and then multiplying the resulting sum by a vector is the same as multiplying each scalar by the vector and then adding the two resultant vectors. $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$.
- **Distributivity of scalar multiplication over vector addition.** Adding two vectors and then multiplying the sum by a scalar is the same as multiplying each vector by the scalar and then adding. $\alpha(\mathbf{u} + \mathbf{v}) = \alpha\mathbf{u} + \alpha\mathbf{v}$.
- **Existence of scalar identity.** If $\alpha = 1$, then for any vector $\mathbf{u}$, $\alpha\mathbf{u} = \mathbf{u}$.

A.1.3 Vector Spaces

A **vector space** is a set of vectors, along with an associated set of scalars (e.g., the real numbers) that satisfies the properties given above and that is closed under vector addition and multiplication by a scalar. (By closed, we mean that every result of vector addition and/or scalar multiplication results in a vector in the original set.) Vector spaces have the property that any vector can be represented as a **linear combination** of a small set of vectors, which are known as a **basis**. More specifically, if $\mathbf{u}_1, \ldots, \mathbf{u}_n$ are the basis vectors, then we can find a set of n scalars $\{\alpha_1, \ldots, \alpha_n\}$ for any vector $\mathbf{v}$, so that $\mathbf{v} = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i$. We say that the basis vectors **span** the vector space. The **dimension** of a vector space is the minimum number of vectors that are necessary to form a basis. Typically, the basis vectors are taken to have unit length.

The basis vectors are usually **orthogonal**. The orthogonality of vectors is an extension of the two-dimensional notion of perpendicular lines and will be defined more precisely later on. Conceptually, orthogonal vectors are unrelated or independent. If basis vectors are mutually orthogonal, then expressing a vector as a linear combination of basis vectors effectively decomposes the vector into a number of **independent components**.

Thus, a vector in an n-dimensional space can be considered to be an n-tuple of scalars (numbers). To provide a concrete illustration, consider two-dimensional Euclidean space, where each point is associated with a vector that represents the displacement of the point from the origin. The displacement vector to any point can be written as the sum of a displacement in the x

direction and a displacement in the y direction, which are, respectively, the x and y coordinates of the point.

We will refer to the components of a vector $\mathbf{v}$ by using the notation $\mathbf{v} = (v_1, v_2, \ldots, v_{n-1}, v_n)$. (With reference to the equation, $\mathbf{v} = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i$, $v_i = \alpha_i$.) Note that v_i is a component of $\mathbf{v}$, while $\mathbf{v}_i$ is one of a set of vectors.

With a component view of vectors, the addition of vectors becomes simple to understand; to add two vectors, we simply add corresponding components. For example, (2,3) + (4,2) = (6,5). To multiply a vector by a scalar, we multiply each component by the scalar, e.g., 3 * (2,3) = (6,9).

A.1.4 The Dot Product, Orthogonality, and Orthogonal Projections

We now define what it means for two vectors to be orthogonal. For simplicity, we restrict ourselves to Euclidean vector spaces, although the definitions and results are easily generalized. We begin by defining the **dot product** of two vectors.

Definition A.1 (Dot Product)**.** The dot product $\mathbf{u} \cdot \mathbf{v}$ of two vectors, $\mathbf{u}$ and $\mathbf{v}$, is given by the following equation:

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i. \tag{A.1}$$

In words, the dot product of two vectors is computed by multiplying corresponding components of a vector and then adding the resulting products. For instance, $(2,3) \cdot (4,1) = 2 * 4 + 3 * 1 = 11$.

In Euclidean space it can be shown that the dot product of two (non-zero) vectors is 0 if and only if they are perpendicular. Geometrically, two vectors define a plane, and their dot product is 0 if and only if the angle (in the plane) between the two vectors is 90°. We say that such vectors are **orthogonal**.

The dot product can also be used to compute the length of a vector in Euclidean space, namely, length($\mathbf{u}$) $= \sqrt{\mathbf{u} \cdot \mathbf{u}}$. The length of a vector is also known as its L_2 **norm** and is written as $||\mathbf{u}||$. Given a vector $\mathbf{u}$, we can find a vector that is pointing in the same direction as $\mathbf{u}$, but is of unit length, by dividing each component of $\mathbf{u}$ by its length; i.e., by computing $\mathbf{u}/||\mathbf{u}||$. We say that we have normalized the vector to have an L_2 norm of 1.

Given the notation for the norm of a vector, the dot product of a vector can be written as

$$\mathbf{u} \cdot \mathbf{v} = ||\mathbf{u}||\ ||\mathbf{v}||\ cos(\theta), \tag{A.2}$$

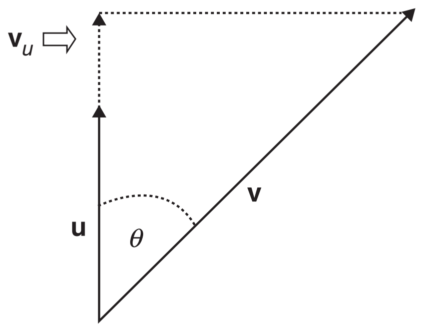


Figure A.2. Orthogonal projection of vector $\mathbf{v}$ in the direction of vector $\mathbf{u}$.

where θ is the angle between the two vectors. By grouping terms and reordering, this can be rewritten as

$$\mathbf{u} \cdot \mathbf{v} = (||\mathbf{v}||\ cos(\theta))\ ||\mathbf{u}|| = \mathbf{v}_u\ ||\mathbf{u}||, \tag{A.3}$$

where $\mathbf{v}_u = ||\mathbf{v}||\ cos(\theta)$ represents the length of $\mathbf{v}$ in the direction of $\mathbf{u}$ as illustrated in Figure A.2. If $\mathbf{u}$ is a unit vector, then the dot product is the component of $\mathbf{v}$ in the direction of $\mathbf{u}$. We refer to this as the **orthogonal projection** of $\mathbf{v}$ onto $\mathbf{u}$. Of course, it is also true that if $\mathbf{v}$ is a unit vector, then the dot product is the projection of $\mathbf{u}$ in the direction of $\mathbf{v}$.

An important consequence of this is that, given a set of orthogonal vectors of norm 1 that form a basis of a vector space, we can find the components of any vector with respect to that basis by taking the dot product of the vector with each basis vector.

A concept that is closely related to that of orthogonality is the notion of **linear independence**.

Definition A.2 (Linear Independence)**.** A set of vectors is linearly independent if no vector in the set can be written as a linear combination of the other vectors in another set.

If a set of vectors is not linearly independent, then they are **linearly dependent**. Note that we want our basis to consist of a set of vectors such that no vector is linearly dependent with respect to the remaining basis vectors, because if this were so, then we could eliminate that vector and still have a set of vectors that span the entire vector space. If we choose our basis vectors to be mutually orthogonal (independent), then we automatically obtain a linearly independent set since any two vectors that are orthogonal are linearly independent.

A.1.5 Vectors and Data Analysis

Although vectors were originally introduced to deal with quantities such as force, velocity, and acceleration, they have proven useful for representing and understanding many other kinds of data. In particular, we can often regard a data object or an attribute as a vector. For example, Chapter 2 described a data set that consisted of 150 Iris flowers that were characterized by four attributes: sepal length, sepal width, petal length, and petal width. Each flower can be regarded as a four dimensional vector, and each attribute can be regarded as a 150 dimensional vector. As another example, a document can be represented as a vector, where each component corresponds to a term (word) and the value of each component is the number of times the term appears in the document. This yields a very sparse, high-dimensional vector, where by sparse, we mean that most of the entries of the vector are 0.

Once we have represented our data objects as vectors, we can perform various operations on the data that derive from a vector viewpoint. For example, using various vector operations, we can compute the similarity or distance of two vectors. In particular, the cosine similarity of two vectors is defined as

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}}{||\mathbf{u}||} \cdot \frac{\mathbf{v}}{||\mathbf{v}||}. \tag{A.4}$$

This similarity measure does not take into account the magnitude (length) of the vectors, but is only concerned with the degree to which two vectors point in the same direction. In terms of documents, this means that two documents are the same if they contain the same terms in the same proportion. Terms that do not appear in both documents play no role in computing similarity.

We can also simply define the distance between two vectors (points). If **u** and **v** are vectors, then the Euclidean distance between the two vectors (points) is simply

$$dist(\mathbf{u}, \mathbf{v}) = \sqrt{(\mathbf{u} - \mathbf{v}) \cdot (\mathbf{u} - \mathbf{v})}. \tag{A.5}$$

This type of measure is more appropriate for the Iris data, since the magnitude of the various components of the vectors does make a difference in whether they are considered to be similar.

Also, for vector data, it is meaningful to compute the mean of the set of vectors, which is accomplished by computing the mean of each component. Indeed, some clustering approaches, such as K-means (Chapter 7) work by dividing the data objects into groups (clusters) and characterizing each cluster by the mean of the data objects (data vectors). The idea is that a good cluster is one in which the data objects in the cluster are close to the mean, where closeness is measured by Euclidean distance for data like the Iris data and by cosine similarity for data like document data.

Other common operations that are performed on data can also be thought of as operations on vectors. Consider dimensionality reduction. In the simplest approach, some of the components of the data vector are eliminated, while leaving the others unchanged. Other dimensionality reduction techniques produce a new set of components (attributes) for the data vector that are linear combinations of the previous components. Still other methods change the vectors in more complicated ways. Dimensionality reduction is discussed further in Appendix B.

For certain areas of data analysis, such as statistics, the analysis techniques are expressed mathematically in terms of operations on data vectors and the data matrices that contain these data vectors. Thus, a vector representation brings with it powerful mathematical tools that can be used to represent, transform, and analyze the data.

In the remainder of this appendix, we will complete the story, by discussing matrices.

A.2 Matrices

A.2.1 Matrices: Definitions

A **matrix** is a tabular representation of a set of numbers as a collection of rows and columns. We will use uppercase bold letters, such as $\mathbf{A}$, to represent matrices. (Uppercase italic letters, such as A, are also used.) The term "m by n matrix" is commonly used to refer to a matrix with m rows and n columns. For example, the matrix $\mathbf{A}$, shown below, is a 2 by 3 matrix. If $m = n$, we say that the matrix is a **square matrix**. The transpose of $\mathbf{A}$ is written as $\mathbf{A}^T$ and is produced by interchanging the rows and columns of $\mathbf{A}$.

$$\mathbf{A} = \begin{bmatrix} 2 & 6 & 1 \\ 7 & 5 & 2 \end{bmatrix} \qquad \mathbf{A}^T = \begin{bmatrix} 2 & 7 \\ 6 & 5 \\ 1 & 2 \end{bmatrix}$$

The **matrix entries** are represented by subscripted, lowercase letters. For matrix **A**, for example, a_{ij} is the entry in the i^{th} row and j^{th} column. Rows are numbered from top to bottom and columns from left to right. As a specific illustration, $a_{21} = 7$ is the entry in the second row and first column of **A**.

Each row or column of a matrix defines a vector. For a matrix **A**, the i^{th} **row vector** can be represented using the notation $\mathbf{a}_{i*}$ and the j^{th} **column vector** using the notation $\mathbf{a}_{*j}$. Using the previous example, $\mathbf{a}_{2*} = [7\ 5\ 2]$, while $\mathbf{a}_{*3} = [1\ 2]^T$. Notice that row and column vectors are matrices and must be distinguished; i.e., a row vector and column vector that have the same number of entries and the same values represent different matrices.

A.2.2 Matrices: Addition and Multiplication by a Scalar

Like vectors, matrices can be added by adding their corresponding entries (components). (Here we are assuming that the matrices have the same number of rows and columns.) More specifically, if **A** and **B** are two matrices having dimensions m by n, then the sum of **A** and **B** is defined as follows:

Definition A.3 (Matrix Addition)**.** The sum of two m by n matrices, **A** and **B**, is an m by n matrix C, whose entries are given by the following equation:

$$c_{ij} = a_{ij} + b_{ij}. \tag{A.6}$$

For example,

$$\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 5 & 4 \\ 2 & 9 \end{bmatrix} = \begin{bmatrix} 8 & 5 \\ 3 & 11 \end{bmatrix}.$$

Matrix addition has the following properties:

- **Commutativity of matrix addition.** The order of addition does not matter. $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$.

- **Associativity of matrix addition.** The grouping of matrices during addition does not matter. $(\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C})$.

- **Existence of an identity element for matrix addition.** There exists a **zero matrix**, having all 0 entries and simply denoted as $\mathbf{0}$, which is the identity element. For any matrix **A**, $\mathbf{A} + \mathbf{0} = \mathbf{A}$.

- **Existence of additive inverses for matrix addition.** For every matrix **A** there is a matrix $-\mathbf{A}$ such that $\mathbf{A} + (-\mathbf{A}) = \mathbf{0}$. The entries of $-\mathbf{A}$ are $-a_{ij}$.

As with vectors, we can also multiply a matrix by a scalar.

Definition A.4 (Scalar Multiplication of a Matrix)**.** The product of a scalar α and a matrix $\mathbf{A}$ is the matrix $B = \alpha A$, whose entries are given by the following equation.

$$b_{ij} = \alpha\, a_{ij} \tag{A.7}$$

Scalar multiplication has properties that are very similar to those of multiplying a vector by a scalar.

- **Associativity of scalar multiplication.** The order of multiplication by two scalars does not matter. $\alpha(\beta\mathbf{A}) = (\alpha\beta)\mathbf{A}$.
- **Distributivity of scalar addition over multiplication of a scalar by a matrix.** Adding two scalars and then multiplying the sum by a matrix is the same as multiplying each scalar times the matrix and then adding the two resultant matrices. $(\alpha + \beta)\mathbf{A} = \alpha\mathbf{A} + \beta\mathbf{A}$.
- **Distributivity of scalar multiplication over matrix addition.** Adding two matrices and then multiplying the sum by a scalar is the same as multiplying each matrix by the scalar and then adding. $\alpha(\mathbf{A} + \mathbf{B}) = \alpha\mathbf{A} + \alpha\mathbf{B}$.
- **Existence of scalar identity.** If $\alpha = 1$, then for any matrix $\mathbf{A}$, $\alpha\mathbf{A} = \mathbf{A}$.

None of the previous properties should be surprising since we can think of a matrix as being composed of row or column vectors, and hence, matrix addition or the multiplication by a scalar amounts to adding corresponding row or column vectors or multiplying them by a scalar.

A.2.3 Matrices: Multiplication

We can define a multiplication operation for matrices. We begin by defining multiplication between a matrix and a vector.

Definition A.5 (Multiplication of a Matrix by a Column Vector)**.** The product of an m by n matrix $\mathbf{A}$ and an n by 1 column matrix $\mathbf{u}$ is the m by 1 column matrix $\mathbf{v} = \mathbf{Au}$, whose entries are given by the following equation.

$$v_i = \mathbf{a}_{i*} \cdot \mathbf{u} \tag{A.8}$$

In other words, we take the dot product of the transpose of **u** with each row vector of **A**. In the following example, notice that the number of rows in **u** must be the same as the number of columns of **A**.

$$\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \end{bmatrix} = \begin{bmatrix} 17 \\ 9 \end{bmatrix}$$

We can similarly define the multiplication of a matrix by a row vector on the left side.

Definition A.6 (Multiplication of a Matrix by a Row Vector)**.** The product of a 1 by m row matrix **u** and an m by n matrix **A** is the 1 by n row matrix $\mathbf{v} = \mathbf{uA}$, whose entries are given by the following equation.

$$v_i = \mathbf{u} \cdot (\mathbf{a}_{*j})^T \tag{A.9}$$

In other words, we take the dot product of the row vector with the transpose of each column vector of **A**. An example is given below.

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 5 & 4 \\ 2 & 9 \end{bmatrix} = \begin{bmatrix} 9 & 22 \end{bmatrix}$$

We define the product of two matrices as an extension to the above idea.

Definition A.7. The product of an m by n matrix **A** and an n by p matrix **B** is the m by p matrix $\mathbf{C} = \mathbf{AB}$, whose entries are given by the equation

$$c_{ij} = \mathbf{a}_{i*} \cdot (\mathbf{b}_{*j})^T \tag{A.10}$$

In words, the ij^{th} entry of **C** is the dot product of the i^{th} row vector of **A** and the transpose of the j^{th} column vector of **B**.

$$\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 5 & 4 \\ 2 & 9 \end{bmatrix} = \begin{bmatrix} 17 & 21 \\ 9 & 22 \end{bmatrix}$$

Matrix multiplication has the following properties:

- **Associativity of matrix multiplication.** The order of multiplication of matrices does not matter. $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$.
- **Distributivity of matrix multiplication.** Matrix multiplication is distributive with respect to matrix addition. $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$ and $(\mathbf{B} + \mathbf{C})\mathbf{A} = \mathbf{BA} + \mathbf{CA}$.

- **Existence of an identity element for matrix multiplication.** If $\mathbf{I}_p$ is the p by p matrix with 1's only on the diagonal and 0 elsewhere, then for any m by n matrix $\mathbf{A}$, $\mathbf{AI}_n = \mathbf{A}$ and $\mathbf{I}_m\mathbf{A} = \mathbf{A}$. (Note that the identity matrix is an example of a **diagonal matrix**, which is a matrix whose off diagonal entries are all 0, i.e., $a_{ij} = 0$, if $i \neq j$.)

In general, matrix multiplication is not commutative, i.e., $\mathbf{AB} \neq \mathbf{BA}$.

A.2.4 Linear Transformations and Inverse Matrices

If we have an n by 1 column vector $\mathbf{u}$, then we can view the multiplication of an m by n matrix $\mathbf{A}$ by this vector on the right as a transformation of $\mathbf{u}$ into an m-dimensional column vector $\mathbf{v} = \mathbf{Au}$. Similarly, if we multiply $\mathbf{A}$ by a (row) vector $\mathbf{u} = [u_1, \ldots, u_m]$ on the left, then we can view this as a transformation of $\mathbf{u}$ into an n-dimensional row vector $\mathbf{v} = \mathbf{uA}$. Thus, we can view any m by n matrix $\mathbf{A}$ as a function that maps one vector space onto another.

In many cases, the transformation (matrix) can be described in easily understood terms.

- A **scaling matrix** leaves the direction of the vector unchanged, but changes its length. This is equivalent to multiplying by a matrix that is the identity matrix multiplied by a scalar.

- A **rotation matrix** changes the direction of a vector, but leaves the magnitude of the vector unchanged. This amounts to a change of coordinate system.

- A **reflection matrix** reflects a vector across one or more coordinate axes. This would be equivalent to multiplying some of the entries of the vector by -1, while leaving the other entries unchanged.

- A **projection** matrix takes vectors into a lower dimensional subspace. The simplest example is the modified identity matrix where one or more of the 1's on the diagonal have been changed into 0's. Such a matrix eliminates the vector components corresponding to those zero entries, while preserving all others.

Of course, a single matrix can do two kinds of transformations at once, e.g., scaling and rotation.

Following are a few properties of matrices when viewed as functions that map vectors from one vector space to another.

- Matrices are **linear transformations**, i.e., $\mathbf{A}(\alpha\mathbf{u}+\beta\mathbf{v}) = \alpha\mathbf{A}\mathbf{u}+\beta\mathbf{A}\mathbf{v}$ and $(\alpha\mathbf{u}+\beta\mathbf{v})\mathbf{A} = \alpha\mathbf{u}\mathbf{A}+\beta\mathbf{v}\mathbf{A}$.

- The set of all transformed row vectors of a matrix $\mathbf{A}$ is called the **row space** of $\mathbf{A}$ because the row vectors of the matrix, or some subset of them, form a basis for the subspace of transformed row vectors. This is evident from the following equation, which expresses the product of a 1 by m row vector $\mathbf{u} = [u_1, \ldots, u_m]$ and an m by n matrix $\mathbf{A}$ as a linear combination of the rows of the matrix.

$$\mathbf{v} = \mathbf{u}\mathbf{A} = \sum_{i=1}^{m} u_i \mathbf{a}_{i*} \qquad \text{(A.11)}$$

  The dimension of the row space tells us the number of linearly independent rows of $\mathbf{A}$.

- The set of all transformed column vectors is called the **column space** of $\mathbf{A}$. The column vectors of the matrix, or some subset of them, form a basis for the subspace of transformed column vectors. This is clear from the following equation, which expresses the product of an n by 1 column vector $\mathbf{u} = [u_1, \ldots, u_n]^T$ and an m by n matrix $\mathbf{A}$ as a linear combination of the columns of the matrix.

$$\mathbf{v} = \mathbf{A}\mathbf{u} = \sum_{j=1}^{n} u_j \mathbf{a}_{*j} \qquad \text{(A.12)}$$

  The dimension of the column space tells us the number of linearly independent columns of $\mathbf{A}$.

- The **left nullspace** is the set of row vectors that the matrix maps to 0.

- The **right nullspace** (or more commonly, just nullspace) is the set of column vectors that the matrix maps to 0.

Note that the **rank of a matrix** is the minimum of the dimensionality of the row space and column space and is often used to characterize a matrix. For instance, if we take a single 1 by n row vector and duplicate it m times to create an m by n matrix, we would only have a matrix of rank 1.

A question of practical and theoretical importance is whether matrices, like real numbers, have multiplicative inverses. First, we note that because of the nature of matrix multiplication (i.e., dimensions have to match), a matrix must be square if it is to have an **inverse matrix**. Thus, for an m by m matrix $\mathbf{A}$, we are asking if we can find a matrix $\mathbf{A}^{-1}$ such that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_m$. The answer is that some square matrices have inverses and some do not.

More abstractly, an m by m matrix has an inverse only if both of its null spaces contain only the 0 vector, or if, equivalently, the row and column spaces are both of dimension m. (This is equivalent to the rank of the matrix being m.) Conceptually, an m by m matrix has an inverse if and only if it uniquely maps every non-zero m-dimensional row (column) vector onto a unique, non-zero m-dimensional row (column) vector.

The existence of an inverse matrix is important when solving various matrix equations.

A.2.5 Eigenvalue and Singular Value Decomposition

We now discuss a very important area of linear algebra: eigenvalues and eigenvectors. Eigenvalues and eigenvectors, along with the related concept of singular values and singular vectors, capture the structure of matrices by allowing us to factor or decompose matrices and express them in a standard format. For that reason, these concepts are useful in the solution of mathematical equations and for dimensionality and noise reduction. We begin with the definition of eigenvalues and eigenvectors.

Definition A.8 (Eigenvectors and Eigenvalues)**.** The eigenvalues and eigenvectors of an n by n matrix $\mathbf{A}$ are, respectively, the scalar values λ and the vectors $\mathbf{u}$ that are solutions to the following equation.

$$A\mathbf{u} = \lambda\mathbf{u} \tag{A.13}$$

In other words, **eigenvectors** are the vectors that are unchanged, except for magnitude, when multiplied by $\mathbf{A}$. The **eigenvalues** are the scaling factors. This equation can also be written as $(\mathbf{A} - \lambda\mathbf{I})\mathbf{u} = \mathbf{0}$.

For square matrices, it is possible to decompose the matrix using eigenvalues and eigenvectors.

Theorem A.1. *Assume that $\mathbf{A}$ is an n by n matrix with n independent (orthogonal) eigenvectors, $u_1, \ldots, u_n$ and n corresponding eigenvalues, $\lambda_1, \ldots, \lambda_n$. Let $\mathbf{U}$ be the matrix whose columns are these eigenvectors, i.e., $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$ and let $\mathbf{\Lambda}$ be a diagonal matrix, whose diagonal entries are the λ_i, $1 \leq i \leq n$. Then $\mathbf{A}$ can be expressed as*

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}. \tag{A.14}$$

Thus, $\mathbf{A}$ can be decomposed into a product of three matrices. $\mathbf{u}$ is known as the **eigenvector matrix** and $\mathbf{\Lambda}$ as the **eigenvalue matrix**.

More generally, an arbitrary matrix can be decomposed in a similar way. Specifically, any m by n matrix $\mathbf{A}$ can be factored into the product of three matrices as described by the following theorem.

Theorem A.2. *Assume that* $\mathbf{A}$ *is an* m *by* n *matrix. Then* $\mathbf{A}$ *can be expressed as follows*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{A.15}$$

Where $\mathbf{U}$ *is* m *by* m*,* Σ *is* m *by* n*, and* $\mathbf{V}$ *is* n *by* n*.* $\mathbf{U}$ *and* $\mathbf{V}$ *are orthonormal matrices, i.e., their columns are of unit length and are mutually orthogonal. Thus,* $\mathbf{U}\mathbf{U}^T = \mathbf{I}_m$ *and* $\mathbf{V}\mathbf{V}^T = \mathbf{I}_n$*.* $\mathbf{\Sigma}$ *is a diagonal matrix whose diagonal entries are non-negative and are sorted so that the larger entries appear first, i.e.,* $\sigma_{i,i} \geq \sigma_{i+1,i+1}$

The column vectors of $\mathbf{V}$, $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are the **right singular vectors**, while the columns of $\mathbf{U}$ are the **left singular vectors**. The diagonal elements of $\mathbf{\Sigma}$, the **singular value matrix**, are typically written as $\sigma_1, \ldots, \sigma_n$ and are called the **singular values** of $\mathbf{A}$. (This use of σ should not be confused with the use of σ to represent the standard deviation of a variable.) There are at most $rank(A) \leq \min(m, n)$ non-zero singular values.

It can be shown that the eigenvectors of $\mathbf{A}^T\mathbf{A}$ are the right singular vectors (i.e., the columns of $\mathbf{V}$), while the eigenvectors of $\mathbf{A}\mathbf{A}^T$ are the left singular vectors (i.e., the columns of $\mathbf{U}$). The non-zero eigenvalues of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ are the σ_i^2, i.e., the squares of the singular values. Indeed, the eigenvalue decomposition of a square matrix can be regarded as a special case of singular value decomposition.

The singular value decomposition (SVD) of a matrix can also be expressed with the following equation. Note that while $\mathbf{u}_i\mathbf{v}_i^T$ might look like a dot product, it is not, and the result is a rank 1 m by n matrix.

$$\mathbf{A} = \sum_{i=1}^{rank(\mathbf{A})} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \tag{A.16}$$

The importance of the above representation is that every matrix can be expressed as a sum of rank 1 matrices that are weighted by singular values. Since singular values, which are sorted in non-increasing order, often decline rapidly in magnitude, it is possible to obtain a good approximation of a matrix by using only a few singular values and singular vectors. This is useful for dimensionality reduction and will be discussed further in Appendix B.

A.2.6 Matrices and Data Analysis

We can represent a data set as a data matrix, where each row is a data object and each column is an attribute. (We can, with equal validity, have attributes as rows and objects as columns.) Matrix representation provides a compact and well-structured representation for our data and permits the easy manipulation of the objects or attributes of the data through various matrix operations.

Systems of linear equations are one very common example of the usefulness of the matrix representation of data. A system of linear equations can be written as the matrix equation $\mathbf{Ax} = \mathbf{b}$ and solved using matrix operations.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m \end{aligned}$$

In particular, if $\mathbf{A}$ has an inverse, the system of equations has a solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. If not, then the system of equations has either no solution or an infinite number of solutions. Note that in this case, our rows (data objects) were equations and our columns were variables (attributes).

For many statistical and data analysis problems, we want to solve linear systems of equations, but these equations cannot be solved in the manner just described. For example, we may have a data matrix where the rows represent patients and the columns represent characteristics of the patients—height, weight, and age—and their response to a particular medication, e.g., a change in blood pressure. We want to express blood pressure (the independent variable) as a linear function of the other (dependent) variables, and we can write a matrix equation in much the same way as above. However, if we have more patients than variables—the usual case—the inverse of the matrix does not exist.

In this case, we still want to find the best solution for the set of equations. This means that we want to find the best linear combination of the independent variables for predicting the dependent variable. Using linear algebra terminology, we want to find the vector $\mathbf{Ax}$ that is as close to $\mathbf{B}$ as possible; in other words, we want to minimize $||\mathbf{b} - \mathbf{Ax}||$, which is the length of the vector $\mathbf{b} - \mathbf{Ax}$. This is known as the **least squares** problem. Many statistical techniques (e.g., **linear regression**, which is discussed in Appendix D) require the solution of a least squares problem. It can be shown that the least squares solution of the equation $\mathbf{Ax} = \mathbf{b}$ is $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$.

Singular value and eigenvalue decomposition are also very useful in analyzing data, particularly in the area of dimensionality reduction, which is discussed in Appendix B. Note that noise reduction can also occur as a side effect of dimensionality reduction.

While we have given a few examples of the application of linear algebra, we have omitted many more. Examples of other areas where linear algebra is important in the formulation and solution of problems include solving systems of differential equations, optimization problems (such as linear programming), and graph partitioning.

A.3 Bibliographic Notes

There are many books that provide good coverage of linear algebra, including those by Demmel [758], Golub and Van Loan [759], and Strang [760].

Bibliography

[758] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM Press, September 1997.

[759] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, November 1996.

[760] G. Strang. *Linear Algebra and Its Applications*. Harcourt Brace & Company, Orlando, FL, 3rd edition, 1986.

B

Dimensionality Reduction

This appendix considers various techniques for dimensionality reduction. The goal is to expose the reader to the issues involved and to describe some of the more common approaches. We begin with a discussion of Principal Components Analysis (PCA) and Singular Value Decomposition (SVD). These methods are described in some detail since they are among the most commonly used approaches and we can build on the discussion of linear algebra in Appendix A. However, there are many other approaches that are also employed for dimensionality reduction, and thus, we provide a quick overview of several other techniques. We conclude with a short review of important issues.

B.1 PCA and SVD

PCA and SVD are two closely related techniques. For PCA, the mean of the data is removed, while for SVD, it is not. These techniques have been widely used for decades in a number of fields. In the following discussion, we will assume that the reader is familiar with linear algebra at the level presented in Appendix A.

B.1.1 Principal Components Analysis (PCA)

The goal of PCA is to find a new set of dimensions (attributes) that better captures the variability of the data. More specifically, the first dimension is chosen to capture as much of the variability as possible. The second dimension is orthogonal to the first, and, subject to that constraint, captures as much of the remaining variability as possible, and so on.

PCA has several appealing characteristics. First, it tends to identify the strongest patterns in the data. Hence, PCA can be used as a pattern-finding technique. Second, often most of the variability of the data can be captured by a small fraction of the total set of dimensions. As a result, dimensionality reduction using PCA can result in relatively low-dimensional data and it may be possible to apply techniques that don't work well with high-dimensional data. Third, since the noise in the data is (hopefully) weaker than the patterns, dimensionality reduction can eliminate much of the noise. This is beneficial both for data mining and other data analysis algorithms.

We briefly describe the mathematical basis of PCA and then present an example.

Mathematical Details

Statisticians summarize the variability of a collection of multivariate data; i.e., data that has multiple continuous attributes, by computing the covariance matrix **S** of the data.

Definition B.1. Given an m by n data matrix $\mathbf{D}$, whose m rows are data objects and whose n columns are attributes, the covariance matrix of $\mathbf{D}$ is the matrix $\mathbf{S}$, which has entries s_{ij} defined as

$$s_{ij} = covariance(\mathbf{d}_{*i}, \mathbf{d}_{*j}). \tag{B.1}$$

In words, s_{ij} is the covariance of the i^{th} and j^{th} attributes (columns) of the data.

The covariance of two attributes is defined in Appendix C, and is a measure of how strongly the attributes vary together. If $i = j$, i.e., the attributes are the same, then the covariance is the variance of the attribute. If the data matrix $\mathbf{D}$ is preprocessed so that the mean of each attribute is 0, then $\mathbf{S} = \mathbf{D}^T\mathbf{D}$.

A goal of PCA is to find a transformation of the data that satisfies the following properties:

1. Each pair of new attributes has 0 covariance (for distinct attributes).
2. The attributes are ordered with respect to how much of the variance of the data each attribute captures.
3. The first attribute captures as much of the variance of the data as possible.

4. Subject to the orthogonality requirement, each successive attribute captures as much of the remaining variance as possible.

A transformation of the data that has these properties can be obtained by using eigenvalue analysis of the covariance matrix. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $\mathbf{S}$. The eigenvalues are all non-negative and can be ordered such that $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_{m-1} \geq \lambda_m$. (Covariance matrices are examples of what are called **positive semidefinite matrices**, which, among other properties, have non-negative eigenvalues.) Let $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$ be the matrix of eigenvectors of $\mathbf{S}$. These eigenvectors are ordered so that the i^{th} eigenvector corresponds to the i^{th} largest eigenvalue. Finally, assume that data matrix $\mathbf{D}$ has been preprocessed so that the mean of each attribute (column) is 0. We can make the following statements.

- The data matrix $\mathbf{D}' = \mathbf{DU}$ is the set of transformed data that satisfies the conditions posed above.
- Each new attribute is a linear combination of the original attributes. Specifically, the weights of the linear combination for the i^{th} attribute are the components of the i^{th} eigenvector. This follows from the fact that the j^{th} column of $\mathbf{D}'$ is given by $\mathbf{Du}_j$ and the definition of matrix-vector multiplication given in Equation A.12.
- The variance of the i^{th} new attribute is λ_i.
- The sum of the variance of the original attributes is equal to the sum of the variance of the new attributes.
- The new attributes are called **principal components**; i.e., the first new attribute is the first principal component, the second new attribute is the second principal component, and so on.

The eigenvector associated with the largest eigenvalue indicates the direction in which the data has the most variance. In other words, if all of the data vectors are projected onto the line defined by this vector, the resulting values would have the maximum variance with respect to all possible directions. The eigenvector associated with the second largest eigenvalue is the direction (orthogonal to that of the first eigenvector) in which the data has the largest remaining variance.

The eigenvectors of $\mathbf{S}$ define a new set of axes. Indeed, PCA can be viewed as a rotation of the original coordinate axes to a new set of axes that are aligned with the variability in the data. The total variability of the data is preserved, but the new attributes are now uncorrelated.

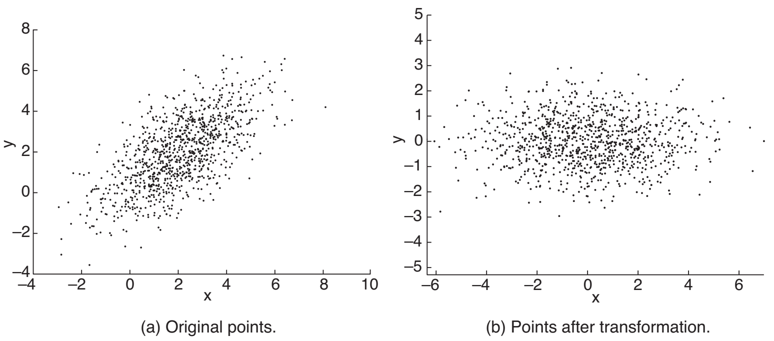
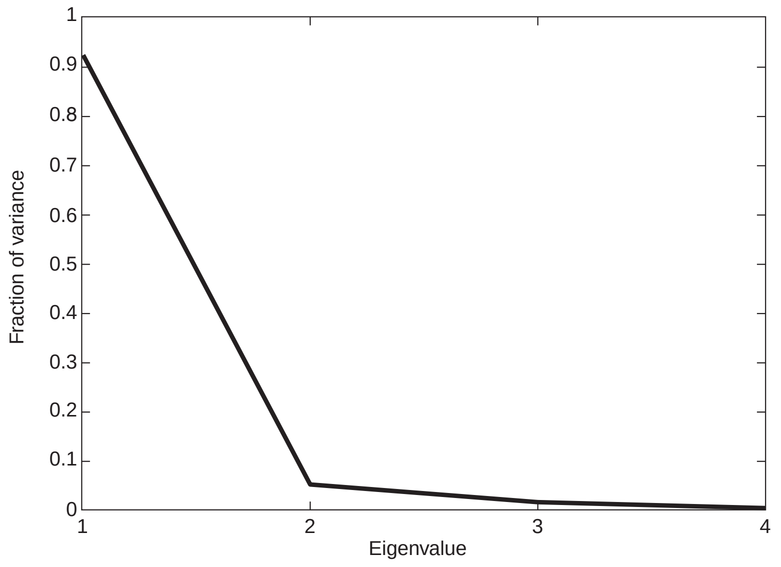


Figure B.1. Using PCA to transform the data.

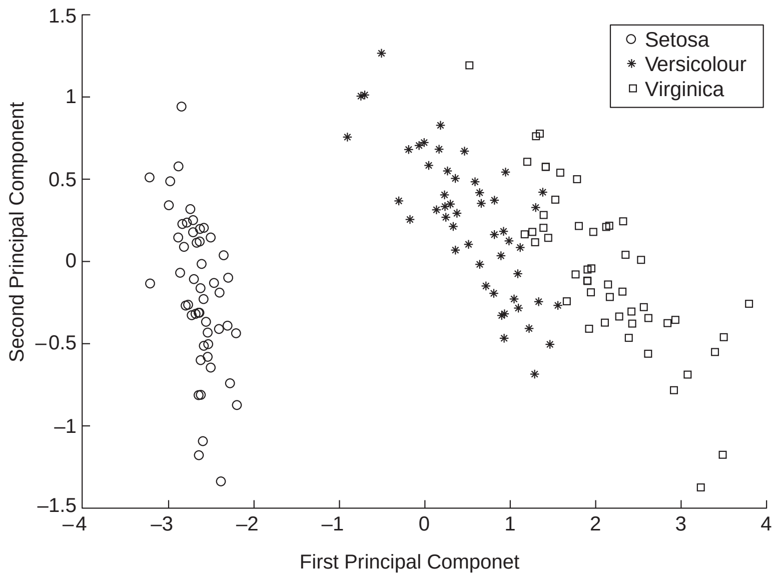
Example B.1 (Two-Dimensional Data). We illustrate the use of PCA for aligning the axes in the directions of the maximum variability of the data. Figure B.1 shows a set of 1000 two-dimensional data points, before and after a PCA transformation. The total variance for the original set of points is the sum of the variance of the x and y attributes, which is equal to 2.84 + 2.95 = 5.79. After transformation, the variance is 4.81 + 0.98 = 5.79. ■

Example B.2 (Iris Data). This example uses the Iris data set to demonstrate the use of PCA for dimensionality reduction. This data set contains 150 data objects (flowers); there are 50 flowers from each of three different Iris species: Setosa, Versicolour, and Virginica. Each flower is described by four attributes: sepal length, sepal width, petal length, and petal width.

Figure B.2(a) shows a plot of the fraction of the overall variance accounted for by each eigenvalue (principal component) of the covariance matrix. This type of plot is known as a **scree plot** and is useful for determining how many principal components need to be kept to capture most of the variability of the data. For the Iris data, the first principal component accounts for most of the variation (92.5%), the second for only 5.3%, and the last two components for just 2.2%. Thus, keeping only the first two principal components preserves most of the variability in the data set. Figure B.2(b) shows a scatter plot of the Iris data based on the first two principal components. Note that the Setosa flowers are well separated from the Versicolour and Virginica flowers. The latter two sets of flowers, while much closer to each other, are still relatively well separated.



(a) Fraction of variance accounted for by each principal component.



(b) Plot of first two principal components of Iris data.

Figure B.2. PCA applied to the Iris data set.

B.1.2 SVD

PCA is equivalent to an SVD analysis of the data matrix, once the mean of each variable has been removed. Nonetheless, it is informative to look at dimensionality reduction from the SVD point of view, since it is not always desirable to remove the mean from data, especially if the data is relatively sparse.

Mathematical Details

From Appendix A, we know that an m by n matrix A can be written as

$$\mathbf{A} = \sum_{i=1}^{rank(A)} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T. \tag{B.2}$$

where σ_i is the i^{th} singular value of $\mathbf{A}$ (the i^{th} diagonal entry of $\boldsymbol{\Sigma}$), $\mathbf{u}_i$ is the i^{th} left singular vector of $\mathbf{A}$ (the i^{th} column of $\mathbf{U}$), and the $\mathbf{v}_i$ is the i^{th} right singular vector of $\mathbf{A}$ (the i^{th} column of $\mathbf{V}$). (See Section A.2.5.) An SVD decomposition of a data matrix has the following properties.

- Patterns among the attributes are captured by the right singular vectors, i.e., the columns of $\mathbf{V}$.
- Patterns among the objects are captured by the left singular vectors, i.e., the columns of $\mathbf{U}$.
- A matrix $\mathbf{A}$ can be successively approximated in an optimal manner by taking, in order, the terms of Equation B.2. We do not explain what we mean by optimal, but refer the reader to the Bibliographic Notes. Informally, the larger a singular value, the larger the fraction of a matrix that is accounted for by the singular value and its associated singular vectors.
- To obtain a new data matrix with k attributes, we compute the matrix $\mathbf{D}' = \mathbf{D} * [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$. It might seem from the previous discussion that we would take the matrix that results from the first k terms of Equation A.12. However, while the resulting matrix is of rank k, it still has n columns (attributes).

Example B.3 (Document Data)**.** SVD decomposition can be used to analyze document data. The data for this example consists of 3204 newspaper articles from the *Los Angeles Times*. These articles come from 6 different sections: Entertainment, Financial, Foreign, Metro, National, and Sports. The data matrix is a document-term matrix, where each row represents a document and each column is a term (word). The value of the ij^{th} entry is the number of times the j^{th} term occurs in the i^{th} document. The data was processed using standard techniques to remove common words, to adjust for the different frequencies with which terms appear, and to adjust for the different lengths of documents. (See Section 2.3.7 for more details.)

An SVD analysis of the data was performed to find the first 100 singular values and vectors. (For many data sets, it is too expensive to find a full SVD or PCA decomposition and often pointless since relatively few of the singular values or eigenvalues are required to capture the structure of the matrix.) The largest singular value is associated with common terms that are frequent, but not eliminated by the preprocessing. (It can happen that the strongest patterns represent noise or uninteresting patterns.)

However, the patterns associated with other singular values were more interesting. For example, the following are the top 10 terms (words) associated with the strongest components in the second right singular vector:
`game, score, lead, team, play, rebound, season, coach, league, goal`
These are all terms associated with sports. Not surprisingly, the documents associated with the strongest components of the second left singular vector are predominantly from the Sports section.

The top 10 terms associated with the strongest components in the third right singular vector are the following:
`earn, million, quarter, bank, rose, billion, stock, company, corporation, revenue`
These are all financial terms, and, not surprisingly, the documents associated with the strongest components in the third left singular vector are predominantly from the Financial section.

We reduced the dimensionality of the data using the second and third singular vectors, i.e., $\mathbf{D}' = \mathbf{D} * [\mathbf{v}_2, \mathbf{v}_3]$. In other words, all documents were expressed in terms of two attributes, one relating to Sports and one relating to Finance. A scatter plot of documents is given by Figure B.3. For clarity, non-Sports, non-Financial documents have been eliminated. The Sports documents are shown in a lighter shade of gray, while the Financial documents are a darker gray. The two different categories of documents are well separated for the most part. Indeed, the Sports documents do not vary much with respect to the Financial variable (component 3) and the Financial documents do not vary much with respect to the Sports variable (component 2).

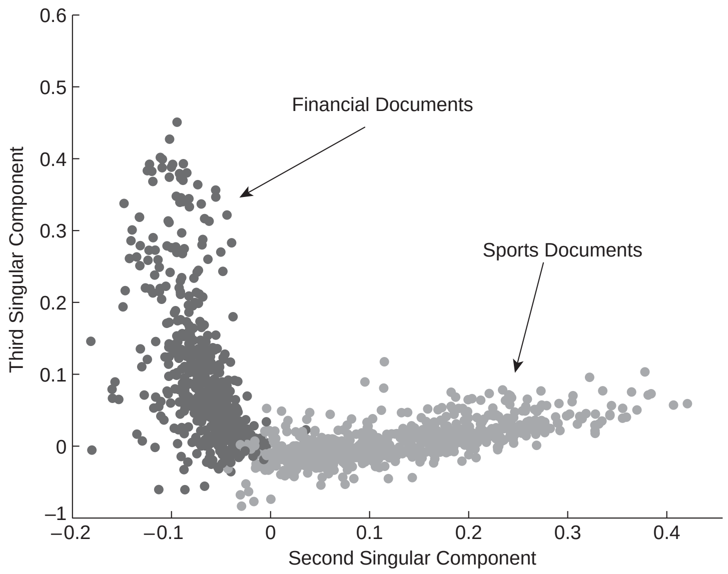


Figure B.3. Plot of Sports and Financial documents from the *LA Times* using the second and third singular values.

B.2 Other Dimensionality Reduction Techniques

In this section, we review a few other dimensionality reduction techniques. These techniques will be discussed more briefly, with a focus on their general motivation and approach.

B.2.1 Factor Analysis

For PCA and SVD, the new attributes that are produced are linear combinations of the original variables. With factor analysis, the goal is to express the original variables as linear combinations of a small number of **hidden** or **latent attributes**. The motivation is based on the following observation. Often there are characteristics of data objects that are hard to measure directly, but that seem to be related to measurable characteristics. One common example is intelligence and performance on various types of IQ tests. Another common example is the connection between performance in various athletic events and an athlete's speed and strength. If a small number of attributes can be found that group and summarize the original attributes, then we will have achieved both a reduction in dimensionality and an increase in our understanding of the data.

The motivation for factor analysis is sometimes also explained in terms of the covariance or correlation matrix of the data. Suppose that a group of attributes are not very highly correlated to other attributes, but are strongly correlated to one another, perhaps because they measure the same underlying quantity. In this case, it would seem desirable to develop techniques that could find a single underlying attribute that summarizes each such group.

For example, consider a data set that records the performance of a group of athletes in the ten separate events that comprise the decathlon. We might find that athletes tend to show the same performance in all events that emphasize speed; i.e., slow athletes are consistently slow and fast athletes are consistently fast. Likewise, we might find that an athlete's behavior in an event that requires strength indicates how he or she will perform in another event that emphasizes strength. Hence, we might hypothesize that an athlete's performance in any given event is really determined by the nature of the event and two underlying factors: speed and strength. Factor analysis attempts to discover such relationships.

More formally, let $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_p$ be the **latent factors**, i.e., the underlying or hidden attributes. Note that these are the new attributes and have a value for each object. If the original data matrix is $\mathbf{D}$, an m by n matrix, then the new data matrix is $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_p]$, which is an m by p matrix. (Note that $\mathbf{f}_{*j} = \mathbf{f}_j$.) The ij^{th} entry of $\mathbf{F}$ is f_{ij}, the j^{th} component of $\mathbf{f}_i$.

Assume that the mean of each attribute is 0. If $\mathbf{d}_{i*}$ is the i^{th} row of the original data matrix $\mathbf{D}$, then $\mathbf{f}_{i*}$ is the corresponding row of the new data matrix, $\mathbf{F}$. The standard factor analysis model assumes the following relationship between the old and new data objects:

$$\mathbf{d}_{i*}^T = \mathbf{\Lambda}\mathbf{f}_{i*}^T + \boldsymbol{\epsilon} \tag{B.3}$$

or equivalently by

$$d_{ij} = \lambda_{j1}f_{i1} + \lambda_{j2}f_{i2}, \ldots, \lambda_{jp}f_{ip} + \epsilon_i. \tag{B.4}$$

$\mathbf{\Lambda}$, which has entries λ_{kl}, is an n by p matrix of **factor loadings** that indicate, for each of the original attributes, how the original value depends on the latent factors, i.e., the new attributes. To illustrate, in the decathlon example, there would be two latent factors: speed and strength. These correspond to columns of $\mathbf{F}$. Each athlete would be represented by a row of $\mathbf{F}$ with entries recording the athlete's speed and strength. Each column of $\mathbf{D}$ would correspond to one of the ten events of the decathlon, while each row again corresponds to an athlete. The ij^{th} entry of $\mathbf{D}$ is the performance of the i^{th} athlete in the j^{th} event. $\mathbf{\Lambda}$ would be a 10 by 2 matrix. If the first column of $\mathbf{D}$ records the performance of the athletes on the 100-meter dash, then the performance of athlete i in the 100-meter dash is written as $d_{i1} = \lambda_{11} f_{i1} + \lambda_{12} f_{i2}$, where f_{i1} is a value indicating the speed of athlete i and f_{i2} is a value indicating the strength of athlete i. λ_{11} and λ_{12} indicate how an athlete's speed and strength, respectively, should be weighted to predict an athlete's performance in the 100 meter dash. We would expect that λ_{11} would be relatively large compared to λ_{12}. Note that these weights are the same across all objects (athletes).

Since all latent factors are involved in the determination of the value of any original attribute, they are known as **common factors**. $\boldsymbol{\epsilon}$ is an error term that accounts for the portion of the attributes that is not accounted for by the common factors, and hence, the components of $\boldsymbol{\epsilon}$ are known as the **specific factors**.

Example B.4 (Factor Analysis of Iris Data)**.** This example is based on the Iris data set. For this data, only a single factor could be found. The flowers in the Iris data set are organized so that the first 50 flowers are of species Setosa, the second 50 are Versicolour, and the last 50 are Virginica. This single factor (attribute) is plotted against flower as shown in Figure B.4. This factor seems to capture the distinction among the three species.

B.2.2 Locally Linear Embedding (LLE)

LLE is a technique for dimensionality reduction based on the idea of analyzing overlapping local neighborhoods in order to determine the local structure. The LLE algorithm is given below.

Algorithm B.1 LLE algorithm.

```
1: Find the nearest neighbors of each data point.
2: Express each point $\mathbf{x}_i$ as a linear combination of the other points, i.e., $\mathbf{x}_i = \sum_j w_{ij}\mathbf{x}_j$, where $\sum_j w_{ij} = 1$ and $w_{ij} = 0$ if $\mathbf{x}_j$ is not a near neighbor of $\mathbf{x}_i$.
3: Find the coordinates of each point in lower-dimensional space of specified dimension $p$ by using the weights found in step 2.
```

In step 2, the weight matrix $\mathbf{W}$, whose entries are w_{ij}, is found by minimizing the squared approximation error as measured by the following equation. W can be found by solving a least squares problem. (Such problems were

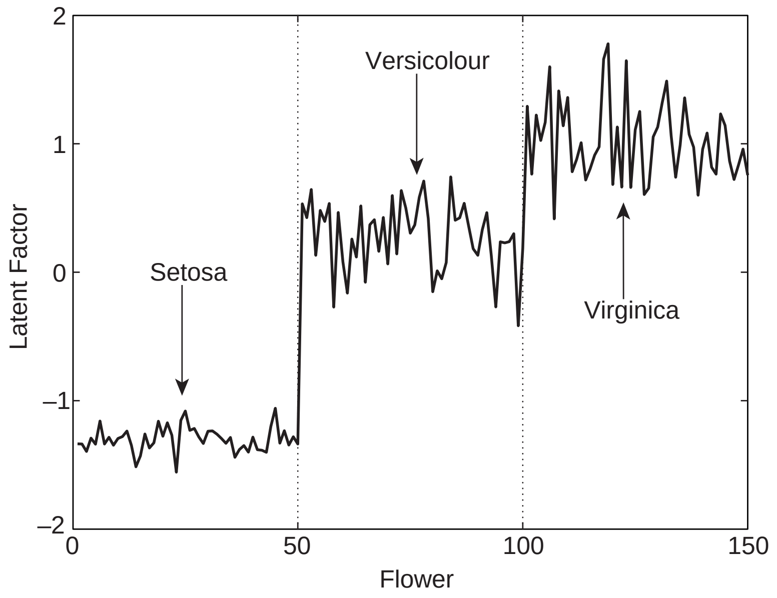


Figure B.4. Plot of the flower of the Iris data set versus the single latent factor.

discussed in Appendix A.)

$$error(\mathbf{W}) = \sum_i \left(\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j \right)^2 \tag{B.5}$$

Step 3 performs the actual dimensionality reduction. Given the weight matrix and a number of dimensions, p, specified by the user, the algorithm constructs a "neighborhood preserving embedding" of the data into the lower-dimensional space. If $\mathbf{y}_i$ is the vector in the lower-dimensional space that corresponds to $\mathbf{x}_i$ and $\mathbf{Y}$ is the new data matrix whose i^{th} row is $\mathbf{y}_i$, then this can be accomplished by finding a $\mathbf{Y}$ that minimizes the following equation.

$$error(\mathbf{Y}) = \sum_i \left(\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j \right)^2 \tag{B.6}$$

Example B.5. the use of LLE for dimensionality reduction is illustrated using the Iris data set. Specifically, the data was projected to two dimensions. A neighborhood of 30 points was used. A scatter plot of the projected data

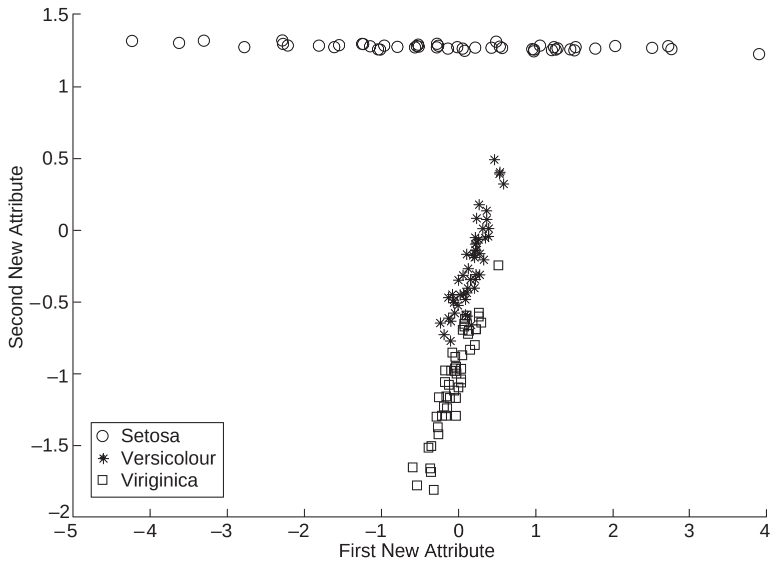


Figure B.5. Plot of the flowers of the Iris data set based on two new attributes from LLE.

is shown in Figure B.5. The data can also be projected to one dimension. In that case, it looks much like Figure B.4.

B.2.3 Multidimensional Scaling, FastMap, and ISOMAP

Multidimensional scaling is a technique that is often used for dimensionality reduction. A number of variations of this technique have been proposed, but the general strategy of these techniques is the same: Find a projection of the data to a lower-dimensional space that preserves pairwise distances as well as possible, as measured by an objective function. Because of this strategy, MDS starts from a dissimilarity matrix, and thus, can be used even for data that does not originally have a vector space representation, e.g., strings.

Standard MDS Techniques

We begin by describing the classical MDS approach for projecting data to a p-dimensional space. Assume that we are given a distance matrix $\mathbf{D}$, where the entry d_{ij} is the distance between the i^{th} and j^{th} objects. Let d'_{ij} be the distance between the objects after they have been transformed. Classical MDS tries to assign each object to a p-dimensional point such that a quantity called

stress is minimized, where stress is defined as

$$stress = \sqrt{\frac{\sum_{ij}\left(d'_{ij} - d_{ij}\right)^2}{\sum_{ij} d_{ij}^2}}. \tag{B.7}$$

The classical version of MDS is an example of **metric MDS** techniques, which assume that the dissimilarities are continuous variables (interval or ration). **Non-metric MDS** techniques assume that the data is categorical (at best ordinal). We will not discuss the details of these algorithms, except to say that the typical approach is to initially assign objects to p-dimensional points in some manner and then try to modify the points to reduce the stress.

When classical MDS or some of the other standard variants of MDS are applied to the Iris data set, they yield almost the same results as shown in Figure B.2. Indeed, classical MDS for Euclidean distance is equivalent to PCA.

FastMap

A recent development in the area of MDS is the algorithm FastMap. It has the same goal as other MDS techniques, but has two important differences.

- It is faster—linear complexity.
- It can operate incrementally.

The FastMap algorithm identifies a pair of objects and then computes the distance of each remaining object in this direction. This can be accomplished using only pairwise distances by employing certain facts of geometry, namely, the law of cosines. This distance is taken as the value of the first attribute. The objects are then projected onto an $(n-1)$-dimensional subspace. Again, this can be performed using only pairwise distances. The process is then repeated.

The FastMap algorithm is initially applied to an entire data set. However, if we keep track of the pairs of objects that are chosen at each step, then we can incrementally apply FastMap to a new object. The only information needed is the distance of the new object to the selected pairs.

ISOMAP

MDS and PCA are not good at dimensionality reduction when the points have a complicated, nonlinear relationship to one another. (An exceptions is kernel

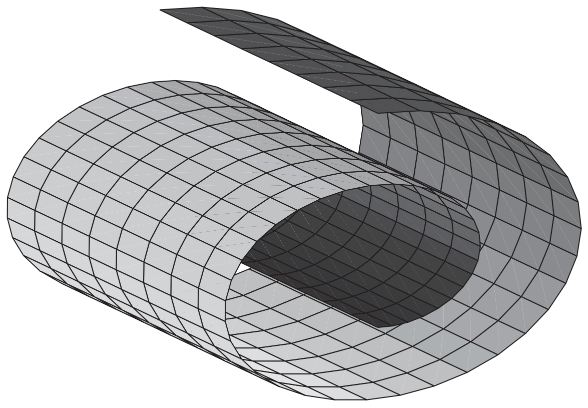


Figure B.6. Plot of Swiss roll data set.

PCA—see Bibliographic Notes.) ISOMAP, which is an extension of traditional MDS, was developed to handle such data sets. An example of the type of data set that it can handle is given in Figure B.6, which shows a plot of the "Swiss roll" surface. A data set with this structure constitutes a two-dimensional set of data in a three-dimensional space, but one that cannot be successfully handled by PCA or MDS. However, ISOMAP can successfully analyze this data set.

Algorithm B.2 outlines the basic ISOMAP algorithm. Nearest neighbors

Algorithm B.2 ISOMAP Algorithm.

```
1: Find the nearest neighbors of each data point and create a weighted graph by
   connecting a point to its nearest neighbors. The nodes are the data points and
   the weights of the links are the distances between points.
2: Redefine the distances between points to be the length of the shortest path
   between the two points in the neighborhood graph.
3: Apply classical MDS to the new distance matrix.
```

can be defined, either by taking the k-nearest points, where k is a parameter, or by taking all points within a specified radius of the point. The purpose of step 2 is to compute the geodesic distance; i.e., the distance between two points that stays on the surface, rather than the Euclidean distance. As an example, the Euclidean distance between two cities on opposite sides of the Earth is the length of a line segment that passes through the Earth, while the geodesic distance between two cities is the length of the shortest arc on the surface of the Earth.

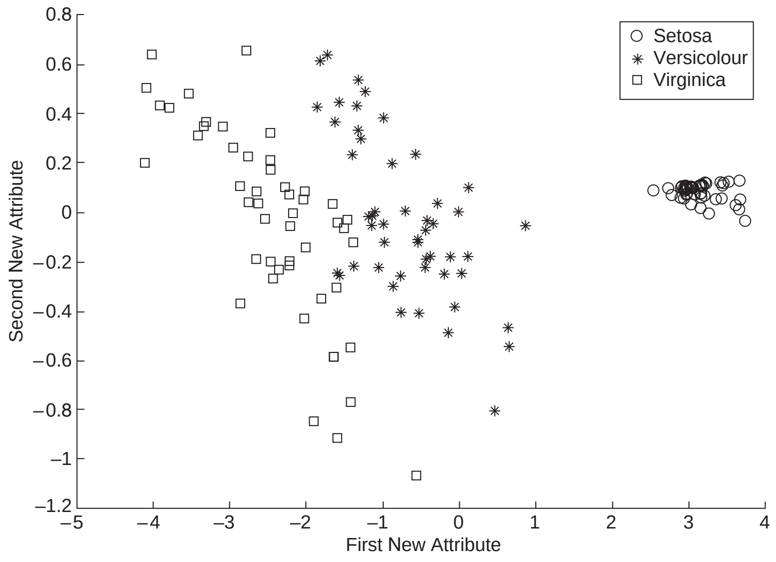


Figure B.7. Plot of the flower of the Iris data set based on two new attributes from ISOMAP.

Example B.6. ISODATA was used to project the Iris data into two dimensions. See Figure B.7. The result is similar to previous techniques.

B.2.4 Common Issues

As with other data analysis techniques, we can distinguish between different dimensionality techniques in a number of areas. One key issue is the quality of the result: Can a technique produce a reasonably faithful representation of the data in a lower-dimensional space? Does this representation capture the characteristics of the data that are important to the intended application (e.g., clusters), while eliminating aspects that are irrelevant or even detrimental (e.g., noise)?

To a large extent, the answer depends on the kind of data and data distributions that can be analyzed by the dimensionality reduction approach. Techniques such as PCA, SVD, and factor analysis assume that there is a linear relationship between the old and new sets of attributes. Although this may be approximately true in many cases, there are many cases where a nonlinear approach is necessary. In particular, algorithms such as ISOMAP and LLE have been developed to deal with nonlinear relationships.

The time and space complexity of dimensionality reduction algorithms is a key issue. Most of the algorithms that we have discussed have time and/or space complexity of $O(m^2)$ or higher, where m is the number of objects. This limits their applicability to larger data sets, although sampling can sometimes be used quite effectively. FastMap is the only algorithm presented here that has linear time and space complexity.

Another important aspect of dimensionality reduction algorithms is whether they produce the same answer every time they are run. PCA, SVD, and LLE do. Factor analysis and the MDS techniques can produce different answers on different runs. Many of the techniques that we did not discuss also have this characteristic because they try to optimize some objective, and this requires a search that may become trapped in a local minimum. Search-based approaches can also have poor time complexity.

Finally, a key issue is determining the number of dimensions for the dimensionality reduction. The techniques that we have considered can typically perform a dimensionality reduction to almost any number of dimensions. The goodness of the reduction is typically measured by some quantity that can be plotted, as in a scree plot. In some cases, this curve provides a clear indication of the intrinsic dimensionality. In many other situations, a choice needs to be made between a smaller number of dimensions and a larger approximation error, and a smaller approximation error and more dimensions.

B.3 Bibliographic Notes

Dimensionality reduction is a broad topic, and the relevant references are scattered across many fields. A comprehensive discussion of PCA can be found in the book by Jolliffe [768], while an introduction to SVD is given by Demmel [764] and other linear algebra texts. Kernel PCA is described by Schölkopf et al. [771]. Many books on multivariate statistical analysis, such as that by Anderson [761], also include discussions on PCA, as well as factor analysis. More details on MDS can be found in the book by Kruskal and Wish [769]. The FastMap algorithm was proposed by Faloutsos and Lin [766]. The papers for LLE (Roweis and Saul [772]) and ISOMAP (Tenenbaum et al. [770]) appeared in the same issue of *Science*. MATLAB code for the ISOMAP and LLE algorithms is available on the Web. Other articles that may be of interest include those by M. Belkin and P. Niyogi [762], Donoho and Grimes [765], and Ye et al. [773, 774]

There are many other techniques that are often used for dimensionality reduction or are strongly related to it. These areas include principal curves and surfaces, nonlinear PCA (including neural network approaches), vector quantization, random projections, Independent Components Analysis (ICA), Self-Organizing Maps (SOM), projection pursuit, regression-based approaches, genetic algorithms, and optimization-based approaches such as simulated or deterministic annealing. Descriptions of these areas and additional references can be found in two surveys on dimensionality reduction by Fodor [767] and Carreira-Perpinan [763]. SOM is discussed in Section 8.2.3.

Bibliography

[761] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis.* Wiley, 2nd edition, July 2003.

[762] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. Technical Report TR 2002-01, Department of Computer Science and Statistics, University of Chicago, January 2002.

[763] M. A. Carreira-Perpinan. A Review of Dimension Reduction Techniques. Technical Report CS–96–09, Dept. of Computer Science, University of Sheffield, January 1997.

[764] J. W. Demmel. *Applied Numerical Linear Algebra.* SIAM Press, September 1997.

[765] D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *PNAS*, 100(10):5591–5596, 2003.

[766] C. Faloutsos and K.-I. Lin. FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In *Proc. of the 1995 ACM SIGMOD Intl. Conf. on Management of Data*, pages 163–174, San Jose, California, June 1995.

[767] I. K. Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, LLNL, June 2002.

[768] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, 2nd edition, October 2002.

[769] J. B. Kruskal and M. Wish. *Multidimensional Scaling.* SAGE Publications, January 1978.

[770] S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

[771] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998.

[772] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

[773] J. Ye, R. Janardan, and Q. Li. GPCA: an efficient dimension reduction scheme for image compression and retrieval. In *Proc. of the 10th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 354–363, Seattle, Washington, August 2004. ACM.

[774] J. Ye, Q. Li, H. Xiong, H. Park, R. Janardan, and V. Kumar. IDR/QR: an incremental dimension reduction algorithm via QR decomposition. In *Proc. of the 10th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 364–373, Seattle, Washington, 2004. ACM.

C

Probability and Statistics

This appendix presents some of the basic concepts in probability and statistics used throughout this book.

C.1 Probability

A **random experiment** is the act of measuring a process whose outcome is uncertain. Examples include rolling a die, drawing from a deck of cards, and monitoring the types of traffic across a network router. The set of all possible outcomes of a random experiment is known as the **sample space**, Ω. For example, $\Omega = \{1, 2, 3, 4, 5, 6\}$ is the sample space for rolling a die. An **event** E corresponds to a subset of these outcomes, i.e., $E \subseteq \Omega$. For example $E = \{2, 4, 6\}$ is the event of observing an even number when rolling a die.

A probability P is a real-valued function defined on the sample space Ω that satisfies the following properties:

1. For any event $E \subseteq \Omega$, $0 \leq P(E) \leq 1$.
2. $P(\Omega) = 1$.
3. For any set of disjoint events, $E_1, E_2, \ldots, E_k \in \Omega$,

$$P(\bigcup_{i=1}^{k} E_i) = \sum_{i=1}^{k} P(E_i).$$

The probability of an event E, which is written as $P(E)$, is the fraction of times event E is observed in a potentially unlimited number of experiments.

In a random experiment, there is often a quantity of interest we want to measure; e.g., counting the number of times a tail turns up when tossing a coin fifty times or measuring the height of a person taking a roller coaster ride at a theme park. Since the value of the quantity depends on the outcome of a random experiment, the quantity of interest is known as a **random variable**. The value of a random variable can be discrete or continuous. A Bernoulli random variable, for example, is a discrete random variable whose only possible values are 0 and 1.

For a discrete random variable X, the probability X takes on a particular value ν is given by the total probability of all outcomes e in which $X(e) = \nu$:

$$P(X = \nu) = P(E = \{e|e \in \Omega, X(e) = \nu\}). \tag{C.1}$$

The probability distribution of a discrete random variable X is also known as its **probability mass function**.

Example C.1. Consider a random experiment where a fair coin is tossed four times. There are 16 possible outcomes of this experiment: HHHH, HHHT, HHTH, HTHH, THHH, HHTT, HTHT, THHT, HTTH, THTH, TTHH, HTTT, THTT, TTHT, TTTH, and TTTT, where H (T) indicates that a head (tail) is observed. Let X be a random variable that measures the number of times a tail is observed in the experiment. The five possible values for X are 0, 1, 2, 3, and 4. The probability mass function for X is given by the following table:

X	0	1	2	3	4
P(X)	1/16	4/16	6/16	4/16	1/16

For example, $P(X = 2) = 6/16$ because there are six outcomes in which the tail is observed twice during the four tosses. ■

On the other hand, if X is a continuous random variable, then the probability that X has a value between a and b is

$$P(a < x < b) = \int_a^b f(x)dx \tag{C.2}$$

The function $f(x)$ is known as the **probability density function** (pdf). Because f is a continuous distribution, the probability that X takes a particular value x is always zero.

Table C.1 shows some of the well-known discrete and continuous probability functions. The notion of a probability (mass or density) function can

Table C.1. Examples of probability functions. ($\Gamma(n+1) = n\Gamma(n)$ and $\Gamma(1) = 1$)

	Probability Function	Parameters
Gaussian	$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$	μ, σ
Binomial	$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$	n, p
Poisson	$p(x) = \frac{1}{x!}\theta^x \exp^{-\theta}$	θ
Exponential	$p(x) = \theta \exp^{-\theta x}$	θ
Gamma	$p(x) = \frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp^{-\lambda x}$	λ, α
Chi-square	$p(x) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} \exp^{-x/2}$	k

be extended to more than one random variable. For example, if X and Y are random variables, then $p(X, Y)$ denotes their **joint** probability function. The random variables are **independent** of each other if $P(X, Y) = P(X) \times P(Y)$. If two random variables are independent, it means that the value for one variable has no impact on the value for the other.

Conditional probability is another useful concept for understanding the dependencies among random variables. The conditional probability for variable Y given X, denoted as $P(Y|X)$, is defined as

$$P(Y|X) = \frac{P(X, Y)}{P(X)}. \tag{C.3}$$

If X and Y are independent, then $P(Y|X) = P(Y)$. The conditional probabilities $P(Y|X)$ and $P(X|Y)$ can be expressed in terms of one another using a formula known as the **Bayes theorem**:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}. \tag{C.4}$$

If $\{X_1, X_2, \ldots, X_k\}$ is the set of mutually exclusive and exhaustive outcomes of a random variable X, then the denominator of the above equation can be expressed as follows:

$$P(X) = \sum_{i=1}^{k} P(X, Y_i) = \sum_{i=1}^{k} P(X|Y_i)P(Y_i). \tag{C.5}$$

Equation C.5 is called the **law of total probability**.

C.1.1 Expected Values

The **expected value** of a function g of a random variable X, denoted as $E[g(X)]$, is the weighted-average value of $g(X)$, where the weights are given by the probability function for X. If X is a discrete random variable, then the expected value can be computed as follows:

$$E[g(X] = \sum_i g(x_i)P(X = x_i). \tag{C.6}$$

On the other hand, if X is a continuous random variable,

$$E[g(X)] = \int_{-\infty}^{\infty} g(X)f(X)dX, \tag{C.7}$$

where $f(X)$ is the probability density function for X. The remainder of this section considers only the expected values for discrete random variables. The corresponding expected values for continuous random variables are obtained by replacing the summation with an integral.

There are several particularly useful expected values in probability theory. First, if $g(X) = X$, then

$$\mu_X = E[X] = \sum_i x_i \; P(X = x_i). \tag{C.8}$$

This expected value corresponds to the **mean** value of the random variable X. Another useful expected value is when $g(X) = (X - \mu_X)$. The expected value of this function is

$$\sigma_X^2 = E[(X - \mu_X)^2] = \sum_i (x_i - \mu_X)^2 \; P(X = x_i). \tag{C.9}$$

This expected value corresponds to the **variance** of the random variable X. The square root of the variance corresponds to the **standard deviation** of the random variable X.

Example C.2. Consider the random experiment described in Example C.1. The average number of tails expected to show up when a fair coin is tossed four times is

$$\mu_X = 0 \times 1/16 + 1 \times 4/16 + 2 \times 6/16 + 3 \times 4/16 + 4 \times 1/16 = 2. \tag{C.10}$$

The variance for the number of tails expected to show up is

$$\begin{aligned}\sigma_X^2 &= (0-2)^2 \times 1/16 + (1-2)^2 \times 4/16 + (2-2)^2 \times 6/16 \\ &\quad +(3-2)^2 \times 4/16 + (4-2)^2 \times 1/16 = 1.\end{aligned}$$

■

For pairs of random variables, a useful expected value to compute is the **covariance** function, Cov, which is defined as follows:

$$Cov(X,Y) = E[(X-\mu_X)(Y-\mu_Y)] \tag{C.11}$$

Note that the variance of a random variable X is equivalent $Cov(X,X)$. The expected value of a function also has the following properties:

1. $E[a] = a$, if a is a constant.
2. $E[aX] = aE[X]$.
3. $E[aX + bY] = aE[X] + bE[Y]$.

Based on these properties, Equations C.9 and C.11 can be rewritten as follows:

$$\sigma_X^2 = E[(X-\mu_X)^2] = E[X^2] - E[X]^2 \tag{C.12}$$

$$Cov(X,Y) = E[XY] - E[X]E[Y] \tag{C.13}$$

C.2 Statistics

To draw conclusions about a population, it is generally not feasible to gather data from the entire population. Instead, we must make reasonable conclusions about the population based on evidence gathered from sampled data. The process of drawing reliable conclusions about the population based on sampled data is known as **statistical inference**.

C.2.1 Point Estimation

The term **statistic** refers to a numeric quantity derived from sampled data. Two examples of useful statistics include the sample mean ($\overline{x}$) and the sample variance (s_X^2):

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{C.14}$$

$$s_X^2 = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \overline{x})^2 \tag{C.15}$$

The process of estimating the parameters of a population using sample statistics is known as **point estimation**.

Example C.3. Let $X_1, X_2, \ldots, X_N$ be a random sample of N independent and identically distributed observations drawn from a population with mean μ_X and variance σ_X^2. Let $\overline{x}$ be the sample mean. Then

$$E[\overline{X}] = E\left[\frac{1}{N}\sum_i X_i\right] = \frac{1}{N}\sum_i E[X_i] = \frac{1}{N} \times N\mu_X = \mu_X, \tag{C.16}$$

where $E[X_i] = \mu_X$ since all the observations come from the same distribution with mean μ_X. This result suggests that the sample mean $\overline{x}$ approaches the population mean μ_X, especially when N is sufficiently large. In statistical terms, the sample mean is called an **unbiased** estimator of the population mean. It is possible to show that the variance of the sample mean is

$$E\left[\left(\overline{x} - E[\overline{x}]\right)^2\right] = \sigma_X^2/N. \tag{C.17}$$

Because the population variance is usually unknown, the variance of the sample mean is often approximated by replacing σ_X^2 with the sample variance s_X^2. The quantity $s_X/\sqrt{N}$ is known as the **standard error** of the mean. ■

C.2.2 Central Limit Theorem

The normal distribution is perhaps one of the most widely-used probability distributions because there are many random phenomena that can be modeled using this distribution. This is a consequence of a statistical principle known as the **central limit theorem**.

Theorem C.1 (Central Limit Theorem)**.** *Consider a random sample of size N drawn from a probability distribution with mean μ_X and variance σ_X^2. If $\overline{x}$ is the sample mean, then the distribution of $\overline{x}$ approaches a normal distribution with mean μ_X and variance σ_X^2/N when the sample size is large.*

The central limit theorem holds regardless of the distribution from which the random variable is drawn. For example, suppose we randomly draw N independent examples from a data set with an unknown distribution. Let X_i be a random variable that denotes whether the i^{th} example is predicted correctly by a given classifier; i.e., $X_i = 1$ if the example is classified correctly, and 0 otherwise. The sample mean, $\overline{X}$, denotes the expected accuracy of the classifier. The central limit theorem suggests that the expected accuracy (i.e., the sample mean) tends to be normally distributed even though the distribution from which the examples are drawn may not be normally distributed.

C.2.3 Interval Estimation

When estimating the parameters of a population, it is useful to indicate the reliability of the estimate. For example, suppose we are interested in estimating the population mean μ_X from a set of randomly drawn observations. Using a point estimate such as the sample mean, $\overline{x}$, may not be sufficient, especially when the sample size is small. Instead, it may be useful to provide an interval that contains the population mean with high probability. The task of estimating an interval in which the population parameter can be found is termed interval estimation. Let θ be the population parameter to be estimated. If

$$P(\theta_1 < \theta < \theta_2) = 1 - \alpha, \tag{C.18}$$

then (θ_1, θ_2) is the confidence interval for θ at a **confidence level** of $1 - \alpha$. Figure C.1 shows the 95% confidence interval for a parameter derived from a normal distribution with mean 0 and variance 1. The shaded region under the normal distribution has an area equal to 0.95. In other words, if we generate a sample from this distribution, there is a 95% chance that the estimated parameter falls between -2 and $+2$.

Consider a sequence of randomly drawn observations, $X_1, X_2, \ldots, X_N$. We would like to estimate the population mean, μ_X, based upon the sample mean, $\overline{x}$, at 68% confidence interval. According to the central limit theorem, $\overline{x}$ approaches a normal distribution with mean μ_X and variance σ_X^2/N when N is sufficiently large. Such a distribution can be transformed into a standard normal distribution (i.e., a normal distribution with mean 0 and variance 1) in the following way:

$$Z = \frac{\overline{x} - \mu_X}{\sigma_X/\sqrt{N}} \approx \frac{\overline{x} - \mu}{s_X/\sqrt{N}} \approx \aleph(0, 1), \tag{C.19}$$

where the population standard deviation is approximated by the standard error of the sample mean. From the probability table of a standard normal

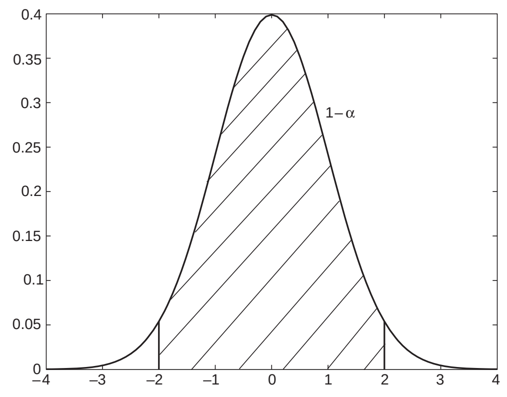


Figure C.1. Confidence interval of a parameter.

distribution, $P(-1 < Z < 1) = 0.68$. The probability can be rewritten in the following way:

$$P(-s_X/\sqrt{N} < \overline{x} - \mu_X < s_X/\sqrt{N}) = 0.68,$$

or equivalently,

$$P(\overline{x} - s_X/\sqrt{N} < \mu_X < \overline{x} + s_X/\sqrt{N}) = 0.68.$$

Therefore, the 68% confidence interval for μ_X is $\overline{x} \pm s_X/\sqrt{N}$.

C.3 Hypothesis Testing

Hypothesis testing is a statistical inference procedure to determine whether a conjecture or hypothesis should be accepted or rejected based on the evidence gathered from data. Examples of hypothesis tests include verifying the quality of patterns extracted by data mining algorithms and validating the significance of the performance difference between two classification models.

In hypothesis testing, we are usually presented with two contrasting hypotheses, which are known, respectively, as the **null hypothesis** and the **alternative hypothesis**. The general procedure for hypothesis testing consists of the following four steps:

1. Formulate the null and alternative hypotheses to be tested.
2. Define a test statistic θ that determines whether the null hypothesis should be accepted or rejected. The probability distribution associated with the test statistic should be known.

3. Compute the value of θ from the observed data. Use the knowledge of the probability distribution to determine a quantity known as the p-value.

4. Define a **significance level**, α, which controls the range of θ values in which the null hypothesis should be rejected. The range of values for θ is known as the **rejection region**.

Consider an association pattern X derived using the algorithms presented in Chapter 5. Suppose we are interested in evaluating the quality of the pattern from a statistical perspective. The criterion for judging whether the pattern is interesting depends on a quantity known as the support of the pattern (see Equation 5.1), $s(X)$. Support measures the fraction of records in which the pattern is actually observed. X is considered interesting if $s(X) > minsup$, where *minsup* is a user-specified minimum threshold.

The problem can be formulated into the hypothesis testing framework in the following way. To validate the pattern X, we need to decide whether to accept the null hypothesis, $H_0 : s(X) = minsup$, or the alternative hypothesis $H_1 : s(X) > minsup$. If the null hypothesis is rejected, then X is considered an interesting pattern. To perform the test, the probability distribution for $s(X)$ must also be known. We can apply the binomial distribution to model this problem because determining the number of times pattern X appears in N records is analogous to determining the number of heads that shows up when tossing N coins. The former can be described by a binomial distribution with mean $s(X)$ and variance $s(X) \times (1 - s(X))/N$. The binomial distribution can be further approximated using a normal distribution if N is sufficiently large, which is typically the case in most market basket analysis problems.

Under the null hypothesis, $s(X)$ is assumed to be normally distributed with mean *minsup* and variance $minsup \times (1 - minsup)/N$. To test whether the null hypothesis should be accepted or rejected, the following Z-statistic can be used:

$$Z = \frac{s(X) - minsup}{\sqrt{minsup \times (1 - minsup)/N}} \tag{C.20}$$

Z has a standard normal distribution with mean 0 and variance 1. The statistic essentially measures the difference between the observed support $s(X)$ and the *minsup* threshold in units of standard deviations. Let $N = 10000$, $s(X) = 11\%$, and $minsup = 10\%$. The Z-statistic under the null hypothesis is $Z = (0.11 - 0.1)/\sqrt{0.09/10000} = 3.33$. From the probability table of a standard normal distribution, a one-sided test with $Z = 3.33$ corresponds to a p-value of 4.34×10^{-4}.

Suppose $\alpha = 0.001$ is the desired significance level. α controls the probability of falsely rejecting the null hypothesis even though the hypothesis is true (in the statistics literature, this is known as the **Type 1** error). For example, an α value of 0.01 suggests that there is one in a hundred chance the discovered pattern is spurious. At each significance level α, there is a corresponding threshold Z_α, such that when the Z value of a pattern exceeds the threshold, the pattern is considered statistically significant. The threshold Z_α can be looked up in a probability table for the standard normal distribution. For example, the choice of $\alpha = 0.001$ sets up a rejection region with $Z_\alpha = 3.09$. Since $p < \alpha$, or equivalently, $Z > Z_\alpha$, the null hypothesis is rejected and the pattern is considered statistically interesting.

D

Regression

Regression is a predictive modeling technique where the target variable to be estimated is continuous. Examples of applications of regression include predicting a stock market index using other economic indicators, forecasting the amount of precipitation in a region based on characteristics of the jet stream, projecting the total sales of a company based on the amount spent for advertising, and estimating the age of a fossil according to the amount of carbon-14 left in the organic material.

D.1 Preliminaries

Let D denote a data set that contains N observations,

$$D = \{(\mathbf{x}_i, y_i)|\ i = 1, 2, \ldots, N\}.$$

Each $\mathbf{x}_i$ corresponds to the set of attributes of the ith observation (also known as the **explanatory variables**) and y_i corresponds to the **target** (or response) **variable**. The explanatory attributes of a regression task can be either discrete or continuous.

Definition D.1 (Regression)**.** Regression is the task of learning a **target function** f that maps each attribute set $\mathbf{x}$ into a continuous-valued output y.

The goal of regression is to find a target function that can fit the input data with minimum error. The **error function** for a regression task can be expressed in terms of the sum of absolute or squared error:

$$\text{Absolute Error} = \sum_i |y_i - f(\mathbf{x}_i)| \tag{D.1}$$

$$\text{Squared Error} = \sum_i (y_i - f(\mathbf{x}_i))^2 \tag{D.2}$$

D.2 Simple Linear Regression

Consider the physiological data shown in Figure D.1. The data corresponds to measurements of heat flux and skin temperature of a person during sleep. Suppose we are interested in predicting the skin temperature of a person based on the heat flux measurements generated by a heat sensor. The two-dimensional scatter plot shows that there is a strong linear relationship between the two variables.

Heat Flux	Skin Temperature
10.858	31.002
10.617	31.021
10.183	31.058
9.7003	31.095
9.652	31.133
10.086	31.188
9.459	31.226
8.3972	31.263
7.6251	31.319
7.1907	31.356
7.046	31.412
6.9494	31.468
6.7081	31.524
6.3221	31.581
6.0325	31.618
5.7429	31.674
5.5016	31.712
5.2603	31.768
5.1638	31.825
5.0673	31.862
4.9708	31.919
4.8743	31.975
4.7777	32.013
4.7295	32.07
4.633	32.126
4.4882	32.164
4.3917	32.221
4.2951	32.259
4.2469	32.296
4.0056	32.334
3.716	32.391
3.523	32.448
3.4265	32.505
3.3782	32.543
3.4265	32.6
3.3782	32.657
3.3299	32.696
3.3299	32.753
3.4265	32.791

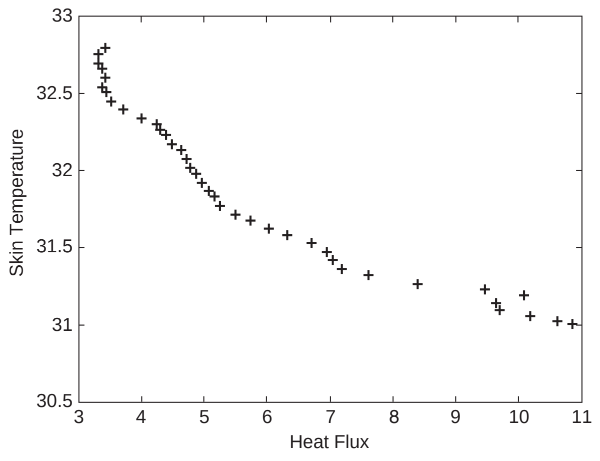


Figure D.1. Measurements of heat flux and skin temperature of a person.

D.2.1 Least Square Method

Suppose we wish to fit the following linear model to the observed data:

$$f(x) = \omega_1 x + \omega_0, \tag{D.3}$$

where ω_0 and ω_1 are parameters of the model and are called the **regression coefficients**. A standard approach for doing this is to apply the **method of least squares**, which attempts to find the parameters (ω_0, ω_1) that minimize the sum of the squared error

$$SSE = \sum_{i=1}^{N} [y_i - f(x_i)]^2 = \sum_{i=1}^{N} [y_i - \omega_1 x - \omega_0]^2, \tag{D.4}$$

which is also known as the **residual sum of squares**.

This optimization problem can be solved by taking the partial derivative of E with respect to ω_0 and ω_1, setting them to zero, and solving the corresponding system of linear equations.

$$\begin{aligned} \frac{\partial E}{\partial \omega_0} &= -2\sum_{i=1}^{N} [y_i - \omega_1 x_i - \omega_0] = 0 \\ \frac{\partial E}{\partial \omega_1} &= -2\sum_{i=1}^{N} [y_i - \omega_1 x_i - \omega_0] x_i = 0 \end{aligned} \tag{D.5}$$

These equations can be summarized by the following matrix equation, which is also known as the **normal equation**:

$$\begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix} \begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}. \tag{D.6}$$

Since $\sum_i x_i = 229.9$, $\sum_i x_i^2 = 1569.2$, $\sum_i y_i = 1242.9$, and $\sum_i x_i y_i = 7279.7$, the normal equations can be solved to obtain the following estimates for the parameters.

$$\begin{aligned} \begin{pmatrix} \hat{\omega}_0 \\ \hat{\omega}_1 \end{pmatrix} &= \begin{pmatrix} 39 & 229.9 \\ 229.9 & 1569.2 \end{pmatrix}^{-1} \begin{pmatrix} 1242.9 \\ 7279.7 \end{pmatrix} \\ &= \begin{pmatrix} 0.1881 & -0.0276 \\ -0.0276 & 0.0047 \end{pmatrix} \begin{pmatrix} 1242.9 \\ 7279.7 \end{pmatrix} \\ &= \begin{pmatrix} 33.1699 \\ -0.2208 \end{pmatrix} \end{aligned}$$

Thus, the linear model that best fits the data in terms of minimizing the SSE is

$$f(x) = 33.17 - 0.22x.$$

Figure D.2 shows the line corresponding to this model.

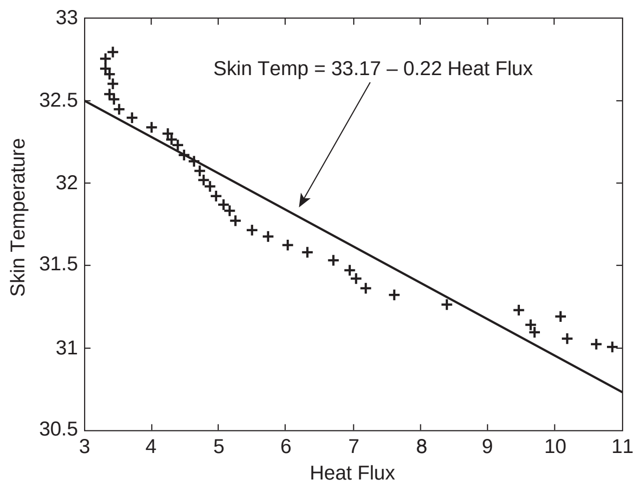


Figure D.2. A linear model that fits the data given in Figure D.1.

We can show that the general solution to the normal equations given in D.6 can be expressed as follow:

$$\begin{aligned} \hat{\omega}_0 &= \overline{y} - \hat{\omega}_1 \overline{x} \\ \hat{\omega}_1 &= \frac{\sigma_{xy}}{\sigma_{xx}} \end{aligned} \qquad \text{(D.7)}$$

where $\overline{x} = \sum_i x_i/N$, $\overline{y} = \sum_i y_i/N$, and

$$\sigma_{xy} = \sum_i (x_i - \overline{x})(y_i - \overline{y}) \qquad \text{(D.8)}$$

$$\sigma_{xx} = \sum_i (x_i - \overline{x})^2 \qquad \text{(D.9)}$$

$$\sigma_{yy} = \sum_i (y_i - \overline{y})^2 \qquad \text{(D.10)}$$

Thus, linear model that results in the minimum squared error is given by

$$f(x) = \overline{y} + \frac{\sigma_{xy}}{\sigma_{xx}}[x - \overline{x}]. \qquad \text{(D.11)}$$

In summary, the least squares method is a systematic approach to fit a linear model to the response variable y by minimizing the squared error between the true and estimated value of y. Although the model is relatively simple, it seems to provide a reasonably accurate approximation because a linear model is the first-order Taylor series approximation for any function with continuous derivatives.

D.2.2 Analyzing Regression Errors

Some data sets may contain errors in their measurements of $\mathbf{x}$ and y. In addition, there may exist confounding factors that affect the response variable y, but are not included in the model specification. Because of this, the response variable y in regression tasks can be non-deterministic, i.e., it may produce a different value even though the same attribute set $\mathbf{x}$ is provided.

We can model this type of situation using a probabilistic approach, where y is treated as a random variable:

$$\begin{aligned} y &= f(\mathbf{x}) + [y - f(\mathbf{x})] \\ &= f(\mathbf{x}) + \epsilon. \end{aligned} \tag{D.12}$$

Both measurement errors and errors in model specification have been absorbed into a random noise term, ϵ. The random noise present in data is typically assumed to be independent and follow a certain probability distribution.

For example, if the random noise comes from a normal distribution with zero mean and variance σ^2, then

$$P(\epsilon|\mathbf{x},\Omega) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{[y-f(\mathbf{x},\Omega)]^2}{2\sigma^2}} \tag{D.13}$$

$$\log[P(\epsilon|\mathbf{x},\Omega)] = -\frac{1}{2}(y - f(\mathbf{x},\Omega))^2 + \text{constant} \tag{D.14}$$

This analysis shows that minimizing the SSE, $[y-f(\mathbf{x},\Omega]^2$, implicitly assumes that the random noise follows a normal distribution. Furthermore, it can be shown that the constant model, $f(\mathbf{x},\Omega) = c$, that best minimizes this type of error is the mean, i.e., $c = \overline{y}$.

Another typical probability model for noise uses the Laplacian distribution:

$$P(\epsilon|\mathbf{x},\Omega) = c\exp^{-c|y-f(\mathbf{x},\Omega)|} \tag{D.15}$$

$$\log[P(\epsilon|\mathbf{x},\Omega)] = -c|y - f(\mathbf{x},\Omega)| + \text{constant} \tag{D.16}$$

This suggests that minimizing the absolute error $|y - f(\mathbf{x}, \Omega)|$ implicitly assumes that the random noise follows a Laplacian distribution. The best constant model for this case corresponds to $f(\mathbf{x}, \Omega) = \tilde{y}$, the median value of y.

Besides the SSE given in Equation D.4, we can also define two other types of errors:

$$\begin{aligned} SST &= \sum_i (y_i - \overline{y})^2 && \text{(D.17)} \\ SSM &= \sum_i (f(x_i) - \overline{y})^2 && \text{(D.18)} \end{aligned}$$

where SST is known as the total sum of squares and SSM is known as the regression sum of squares. SST represents the prediction error when the average value $\overline{y}$ is used as an estimate for the response variable. SSM, on the other hand, represents the amount of error in the regression model. The relationship among SST, SSE, and SSM is derived as follows:

$$\begin{aligned} SSE &= \sum_i [y_i - \overline{y} + \overline{y} - f(x_i)]^2 \\ &= \sum_i [y_i - \overline{y}]^2 + \sum_i [f(x_i) - \overline{y}]^2 + 2\sum_i (y_i - \overline{y})(\overline{y} - f(x_i)) \\ &= \sum_i [y_i - \overline{y}]^2 + \sum_i [f(x_i) - \overline{y}]^2 - 2\sum_i (y_i - \overline{y})\omega_1(x_i - \overline{x}) \\ &= \sum_i [y_i - \overline{y}]^2 + \sum_i [f(x_i) - \overline{y}]^2 - 2\sum_i \omega_1^2(x_i - \overline{x})^2 \\ &= \sum_i [y_i - \overline{y}]^2 - \sum_i [f(x_i) - \overline{y}]^2 \\ &= SST - SSM && \text{(D.19)} \end{aligned}$$

where we have applied the following relationships:

$$\begin{aligned} \overline{y} - f(x_i) &= -\omega_1(x_i - \overline{x}) \\ \sum_i [y_i - \overline{y}][x_i - \overline{x}] &= \sigma_{xy} = \omega_1 \sigma_{xx} = \omega_1 \sum_i [x_i - \overline{x}]^2. \end{aligned}$$

Thus, we can write $SST = SSE + SSM$.

D.2.3 Analyzing Goodness of Fit

One way to measure the goodness of the fit is by computing the following measure:

$$R^2 = \frac{SSM}{SST} = \frac{\sum_i [f(x_i) - \overline{y}]^2}{\sum_i [y_i - \overline{y}]^2} \tag{D.20}$$

The R^2 (or *coefficient of determination*) for a regression model may range between 0 and 1. Its value is close to 1 if most of the variability observed in the response variable can be explained by the regression model.

R^2 is also related to the correlation coefficient, r, which measures the strength of the linear relationship between the explanatory and response variables

$$r = \frac{\sigma_{xy}}{\sqrt{\sigma_{xx}\sigma_{xy}}}. \tag{D.21}$$

From Equations D.9, D.10, and D.11, we can write

$$\begin{aligned} R^2 &= \frac{\sum_i [f(x_i) - \overline{y}]^2}{\sum_i [y_i - \overline{y}]^2} \\ &= \frac{\sum_i [\frac{\sigma_{xy}}{\sigma_{xx}}(x_i - \overline{x})]^2}{\sigma_{yy}} \\ &= \frac{\sigma_{xy}^2}{\sigma_{xx}^2 \sigma_{yy}} \sum_i (x_i - \overline{x})^2 \\ &= \frac{\sigma_{xy}^2}{\sigma_{xx}^2 \sigma_{yy}} \sigma_{xx} \\ &= \frac{\sigma_{xy}^2}{\sigma_{xx}\sigma_{yy}}. \end{aligned} \tag{D.22}$$

The above analysis shows that the correlation coefficient is equivalent to the square root of the coefficient of determination (except for its sign, which depends on the direction of the relationship, whether positive or negative).

It is worth noting that R^2 increases as we add more explanatory variables into the model. One way to correct for the number of explanatory variables added to the model is by using the following adjusted R^2 measure:

$$\text{Adjusted } R^2 = 1 - \left(\frac{N-1}{N-d}\right)(1 - R^2), \tag{D.23}$$

where N is the number of data points and $d + 1$ is the number of parameters of the regression model.

D.3 Multivariate Linear Regression

The normal equations can be written in a more compact form using the following matrix notation. Let $\mathbf{X} = (\mathbf{1}\ \mathbf{x})$, where $\mathbf{1} = (1, 1, 1, \ldots)^T$ and $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$. Then, we can show that

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} \mathbf{1}^T\mathbf{1} & \mathbf{1}^T\mathbf{x} \\ \mathbf{x}^T\mathbf{1} & \mathbf{x}^T\mathbf{x} \end{pmatrix} = \begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix}, \tag{D.24}$$

which is equivalent to the left-hand side matrix of the normal equation. Similarly, if $\mathbf{y} = (y_1, y_2, \ldots, y_N)^T$, we can show that

$$\begin{pmatrix} \mathbf{1} & \mathbf{x} \end{pmatrix}^T \mathbf{y} = \begin{pmatrix} \mathbf{1}^T\mathbf{y} \\ \mathbf{x}^T\mathbf{y} \end{pmatrix} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}, \tag{D.25}$$

which is equivalent to the right-hand side matrix of the normal equation. Substituting Equations D.24 and D.25 into Equation D.6 we obtain the following equation:

$$\mathbf{X}^T\mathbf{X}\Omega = \mathbf{X}^T\mathbf{y}, \tag{D.26}$$

where $\Omega = (\omega_0, \omega_1)^T$. We can solve for the parameters in Ω can as follows:

$$\Omega = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \tag{D.27}$$

The above notation is useful because it allows us to extend the linear regression method to the multivariate case. More specifically, if the attribute set consists of d explanatory attributes $(x_1, x_2, \ldots, x_d)$, $\mathbf{X}$ becomes an $N \times d$ **design matrix**:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1d} \\ 1 & x_{21} & x_{22} & \ldots & x_{2d} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_{N1} & x_{N2} & \ldots & x_{Nd} \end{pmatrix}, \tag{D.28}$$

while $\Omega = (\omega_0, \omega_1, \ldots, \omega_{d-1})^T$ is a d-dimensional vector. The parameters can be computed by solving the matrix equation given in Equation D.26.

D.4 Alternative Least-Square Regression Methods

The least squares method can also be used to find other types of regression models that minimize the SSE. More specifically, if the regression model is

$$\begin{aligned} y &= f(\mathbf{x}, \Omega) + \epsilon && \text{(D.29)} \\ &= \omega_0 + \sum_i \omega_i g_i(\mathbf{x}) + \epsilon, && \text{(D.30)} \end{aligned}$$

and the random noise is normally distributed, then we can apply the same methodology as before to determine the parameter vector Ω. The g_i's can be any type of basis functions, including polynomial, kernel, and other nonlinear functions.

For example, suppose $\mathbf{x}$ is a two-dimensional feature vector and the regression model is a polynomial function of degree 2

$$f(x_1, x_2, \Omega) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_1 x_2 + \omega_4 x_1^2 + \omega_5 x_2^2. \tag{D.31}$$

If we create the following design matrix:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & x_{11}x_{12} & x_{11}^2 & x_{22}^2 \\ 1 & x_{21} & x_{22} & x_{21}x_{22} & x_{21}^2 & x_{22}^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{N1} & x_{N2} & x_{N1}x_{N2} & x_{N1}^2 & x_{N2}^2 \end{pmatrix}, \tag{D.32}$$

where x_{ij} is the jth attribute of the ith observation, then the regression problem becomes equivalent to solving Equation D.26. The least-square solution to the parameter vector Ω is given by Equation D.27. By choosing the appropriate design matrix, we can extend this method to any type of basis functions.

E

Optimization

Optimization is a methodology for finding the maximum or minimum value of a function. It is an important topic in data mining because there are many data mining tasks that can be cast as optimization problems. For example, the K-means clustering algorithm described in Section 7.2.1 seeks to find a set of clusters that minimizes the sum of the squared error (SSE). Similarly, the method of least squares presented in Section D.2.1 is designed to learn the regression coefficients that minimize the SSE of the model. This section presents a brief overview of the various techniques used to solve optimization problems.

E.1 Unconstrained Optimization

Suppose $f(x)$ is a univariate function with continuous first-order and second-order derivatives. In an unconstrained optimization problem, the task is to locate the solution x^* that maximizes or minimizes $f(x)$ without imposing any constraints on x^*. The solution x^*, which is known as a **stationary point**, can be found by taking the first derivative of f and setting it to zero:

$$\left.\frac{df}{dx}\right|_{x=x^*} = 0.$$

$f(x^*)$ can take a maximum or minimum value depending on the second-order derivative of the function:

- x^* is a maximum stationary point if $\frac{d^2f}{dx^2} < 0$ at $x = x^*$.
- x^* is a minimum stationary point if $\frac{d^2f}{dx^2} > 0$ at $x = x^*$.

- x^* is a point of inflection when $\frac{d^2 f}{dx^2} = 0$ at $x = x^*$.

Figure E.1 illustrates an example of a function that contains all three stationary points (maximum, minimum, and point of inflection).

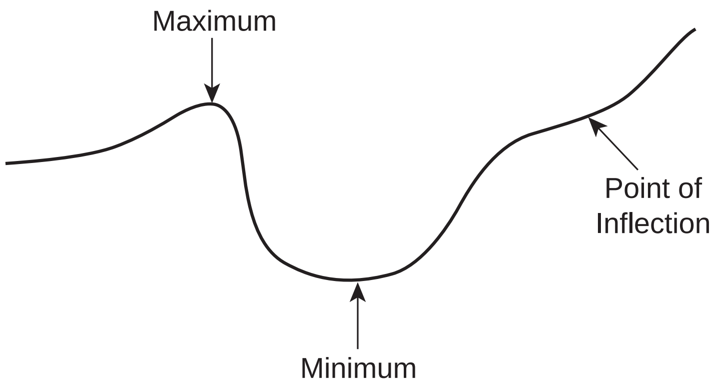


Figure E.1. Stationary points of a function.

This definition can be extended to a multivariate function, $f(x_1, x_2, \ldots, x_d)$, where the condition for finding a stationary point $\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_d^*]^T$ is

$$\left.\frac{\partial f}{\partial x_i}\right|_{x_i = x_i^*} = 0, \ \forall i = 1, 2, \ldots, d. \tag{E.1}$$

However, unlike univariate functions, it is more difficult to determine whether $\mathbf{x}^*$ corresponds to a maximum or minimum stationary point. The difficulty arises because we need to consider the partial derivatives $\frac{\partial^2 f}{dx_i dx_j}$ for all possible pairs of i and j. The complete set of second-order partial derivatives is given by the Hessian matrix

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots \frac{\partial^2 f}{\partial x_d \partial x_d} \end{bmatrix}. \tag{E.2}$$

- A Hessian matrix $\mathbf{H}$ is positive definite if and only if $\mathbf{x}^T\mathbf{H}\mathbf{x} > 0$ for any non-zero vector $\mathbf{x}$. If $\mathbf{H}(\mathbf{x}^*)$ is positive definite, then $\mathbf{x}^*$ is a minimum stationary point.

- A Hessian is negative definite if and only if $\mathbf{x}^T\mathbf{H}\mathbf{x} < 0$ for any non-zero vector $\mathbf{x}$. If $\mathbf{H}(\mathbf{x}^*)$ is negative definite, then $\mathbf{x}^*$ is a maximum stationary point.
- A Hessian is indefinite if $\mathbf{x}^T\mathbf{H}\mathbf{x}$ is positive for some value of $\mathbf{x}$ and negative for others. A stationary point with indefinite Hessian is a **saddle point**, which can have a minimum value in one direction, and a maximum value in another.

Example E.1. Suppose $f(x,y) = 3x^2 + 2y^3 - 2xy$. Figure E.2 shows a plot of this function. The conditions for finding the stationary points of this function are

$$\begin{aligned} \frac{\partial f}{\partial x} &= 6x - 2y = 0 \\ \frac{\partial f}{\partial y} &= 6y^2 - 2x = 0 \end{aligned} \tag{E.3}$$

whose solutions are $x^* = y^* = 0$ or $x^* = 1/27$, $y^* = 1/9$.

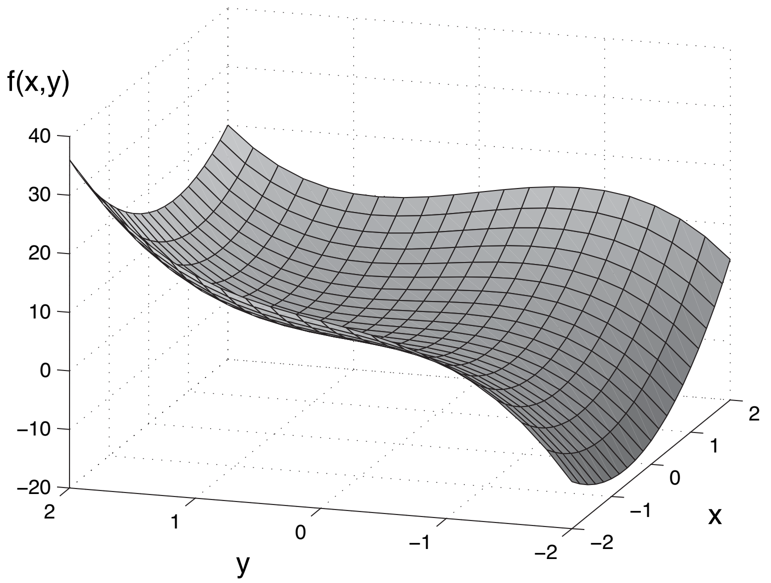


Figure E.2. Plot for the function $f(x,y) = 3x^2 + 2y^3 - 2xy$.

The Hessian of f is

$$\mathbf{H}(x,y) = \begin{bmatrix} 6 & -2 \\ -2 & 12y \end{bmatrix}.$$

At $x = y = 0$,

$$\mathbf{H}(0,0) = \begin{bmatrix} 6 & -2 \\ -2 & 0 \end{bmatrix}.$$

Since $[x\ y]\ H(0,0)\ [x\ y]^T = 6x^2 - 4xy = 2x(3x-2y)$, which can be either positive or negative, the Hessian is indefinite and $(0,0)$ is a saddle point.

At $x = 1/27$, $y = 1/9$,

$$\mathbf{H}(1/27, 1/9) = \begin{bmatrix} 6 & -2 \\ -2 & 12/9 \end{bmatrix}.$$

Since $[x\ y]\ \mathbf{H}(1/27,1/9)\ [x\ y]^T = 4x^2 - 2xy + 4y^2/3 = 4(x-y/4)^2 + 13y^2/4 > 0$ for non-zero x and y, the Hessian is positive definite. Therefore, $(1/27, 1/9)$ is a minimum stationary point. The minimum value of f is -0.0014. ■

E.1.1 Numerical Methods

The preceding approach works if Equation E.1 can be solved analytically for $\mathbf{x}^*$. In many cases, finding analytical solutions is a very difficult problem, thus necessitating the use of numerical methods to find approximate solutions. Some of the numerical methods for finding the minimum value of a function include golden search, Newton's method, and gradient descent search. While the techniques presented here are used to minimize the objective function $f(\mathbf{x})$, they are also applicable to maximization problems because a maximization problem can be easily turned into a minimization problem by converting the function $f(\mathbf{x})$ to $-f(\mathbf{x})$.

Golden Search Consider the unimodal distribution illustrated in Figure E.3, where the minimum value is bracketed between a and b. The golden search method iteratively finds successively smaller brackets that contain the minimum value until the interval width is small enough to approximate the stationary point. To determine the smaller brackets, two additional points, c and d, are chosen so that the intervals (a, c, d) and (c, d, b) have equal width. Let $c - a = b - d = \alpha(b - a)$ and $d - c = \beta \times (b - a)$. Therefore,

$$1 = \frac{(b-d) + (d-c) + (c-a)}{b-a} = \alpha + \beta + \alpha,$$

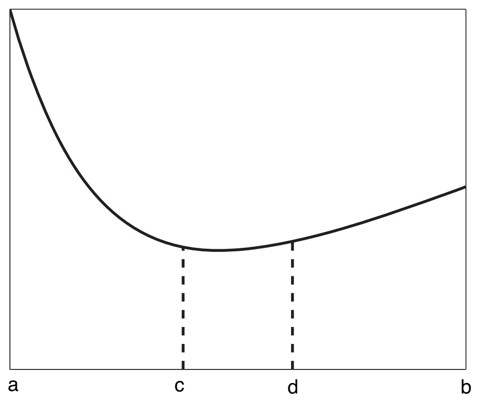


Figure E.3. Example of a unimodal function.

or equivalently,

$$\beta = 1 - 2\alpha. \tag{E.4}$$

The widths are also chosen to obey the following condition so that a recursive procedure can be applied:

$$\frac{d-c}{b-c} = \frac{c-a}{b-a},$$

or equivalently,

$$\frac{\beta}{1-\alpha} = \alpha. \tag{E.5}$$

Together, Equations E.4 and E.5 can be solved to yield $\alpha = 0.382$ and $\beta = 0.236$. By comparing $f(c)$ with $f(d)$, it is possible to detect whether the minimum value occurs in the interval (a, c, d) or (c, d, b). The interval that contains the minimum value is then recursively partitioned until the interval width is small enough to approximate the minimum value, as shown in Algorithm E.1.

The golden search method makes no assumption about the function, other than it must be continuous and unimodal within the initial bracket $[a, b]$. It converges linearly to the solution for the minimum value.

Newton's Method Newton's method is based on using a quadratic approximation to the function $f(x)$. By using a Taylor series expansion of f around

Algorithm E.1 Golden search algorithm.

```
1: $c = a + 0.382(b - a)$.
2: while $b - a > \epsilon$ do
3:     $d = b - 0.382(b - a)$.
4:     if $f(d) > f(c)$ then
5:         $b = d$.
6:     else
7:         $a = c$, $c = d$.
8:     end if
9: end while
10: return $c$.
```

x_0, the following expression is obtained:

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2} f''(x_0). \tag{E.6}$$

Taking the derivative of the function with respect to x and setting it to zero leads to the following equation:

$$\begin{aligned} f'(x) &= f'(x_0) + (x - x_0)f''(x_0) = 0 \\ x &= x_0 - \frac{f'(x_0)}{f''(x_0)}. \end{aligned} \tag{E.7}$$

Equation E.7 can be used to update x until it converges to the location of the minimum value. It can be shown that Newton's method has quadratic convergence, although it may fail to converge in some cases, especially when the initial point x_0 is located far away from the minimum point. A summary of this method is given in Algorithm E.2

Algorithm E.2 Newton's method.

```
1: Let $x_0$ be the initial point.
2: while $|f'(x_0)| > \epsilon$ do
3:     $x = x_0 - \frac{f'(x_0)}{f''(x_0)}$.
4:     $x_0 = x$.
5: end while
6: return $x$.
```

Newton's method can be extended to multivariate data by replacing the first order derivative $f'(x)$ with the gradient operator $\nabla f(\mathbf{x})$ and the second order derivative $f''(x)$ with the Hessian matrix $\mathbf{H}$:

$$\mathbf{x} = \mathbf{x} - \mathbf{H}^{-1}\nabla f(\mathbf{x}).$$

However, instead of computing the inverse of the Hessian matrix, it is easier to solve the following equation:

$$\mathbf{Hz} = -\nabla f(\mathbf{x})$$

to obtain the vector $\mathbf{z}$. The iterative formula for finding the stationary point is modified to $\mathbf{x} = \mathbf{x} + \mathbf{z}$.

Gradient Descent Method Newton's method is one of several incremental methods to progressively locate the stationary point of a function using the following update formula:

$$\mathbf{x} = \mathbf{x} + \lambda g(\mathbf{x})), \tag{E.8}$$

The function $g(\mathbf{x})$ determines the direction in which the search should proceed and λ determines the step size.

The gradient descent method assumes that the function $f(\mathbf{x})$ is differentiable and computes the stationary point as follows:

$$\mathbf{x} = \mathbf{x} - \lambda\nabla f(\mathbf{x}), \tag{E.9}$$

In this method, the location of $\mathbf{x}$ is updated in the direction of the steepest descent, which means that $\mathbf{x}$ is moved toward the decreasing value of f. Section 4.7.2 described how the gradient descent method can be used to learn the weight parameters of an artificial neural network. A summary of this method is given in Algorithm E.3. Notice that the algorithm looks very similar to Algorithm E.2, except for the update formula.

Algorithm E.3 Gradient descent method.

```
1: Let $\mathbf{x}_0$ be the initial point.
2: while $\|\nabla f(\mathbf{x}_0)\| > \epsilon$ do
3:     $x = x_0 - \lambda\nabla f(\mathbf{x})$.
4:     $x_0 = x$.
5: end while
6: return $x$.
```

E.2 Constrained Optimization

This section examines how to solve an optimization problem when the variables are subjected to various types of constraints.

E.2.1 Equality Constraints

Consider the problem of finding the minimum value of $f(x_1, x_2, \ldots, x_d)$ subjected to equality constraints of the form

$$g_i(\mathbf{x}) = 0,\ i = 1, 2, \ldots, p.$$

A method known as Lagrange multipliers can be used to solve the constrained optimization problem. This method involves the following steps:

1. Define the Lagrangian, $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^{p} \lambda_i g_i(\mathbf{x})$, where λ_i is a dummy variable called the **Lagrange multiplier**.

2. Set the first-order derivatives of the Lagrangian with respect to $\mathbf{x}$ and the Lagrange multipliers to zero,

$$\frac{\partial L}{\partial x_i} = 0,\ \forall i = 1, 2, \ldots, d$$

and

$$\frac{\partial L}{\partial \lambda_i} = 0,\ \forall i = 1, 2, \ldots, p.$$

3. Solve the $(d + p)$ equations in step 2 to obtain the stationary point $\mathbf{x}^*$ and the corresponding values for λ_i's.

The following example illustrates how the Lagrange multiplier method works.

Example E.2. Let $f(x, y) = x + 2y$. Suppose we want to minimize the function $f(x, y)$ subject to the constraint $x^2 + y^2 - 4 = 0$. The Lagrange multiplier method can be used to solve this constrained optimization problem in the following way.

First, we introduce the Lagrangian

$$L(x, y, \lambda) = x + 2y + \lambda(x^2 + y^2 - 4),$$

where λ is the Lagrange multiplier. To determine its minimum value, we need to differentiate the Lagrangian with respect to its parameters:

$$\begin{aligned}
\frac{\partial L}{\partial x} &= 1 + 2\lambda x = 0 & \text{(E.10)}\\
\frac{\partial L}{\partial y} &= 2 + 2\lambda y = 0 & \text{(E.11)}\\
\frac{\partial L}{\partial \lambda} &= x^2 + y^2 - 4 = 0
\end{aligned}$$

Solving these equations yields $\lambda = \pm\sqrt{5}/4$, $x = \mp 2/\sqrt{5}$, and $y = \mp 4/\sqrt{5}$. When $\lambda = \sqrt{5}/4$, $f(-2/\sqrt{5}, -4/\sqrt{5}) = -10/\sqrt{5}$. Similarly, when $\lambda = -\sqrt{5}/4$, $f(2/\sqrt{5}, 4/\sqrt{5}) = 10/\sqrt{5}$. Thus, the function $f(x, y)$ has its minimum value at $x = -2/\sqrt{5}$ and $y = -4/\sqrt{5}$. ■

E.2.2 Inequality Constraints

Consider the problem of finding the minimum value of $f(x_1, x_2, \ldots, x_d)$ subjected to inequality constraints of the form

$$h_i(\mathbf{x}) \leq 0, \; i = 1, 2, \ldots, q.$$

The method for solving this problem is quite similar to the Lagrange method described above. However, the inequality constraints impose additional conditions to the optimization problem. Specifically, the optimization problem stated above leads to the following Lagrangian

$$L = f(\mathbf{x}) + \sum_{i=1}^{q} \lambda_i h_i(\mathbf{x}), \tag{E.12}$$

and constraints known as the Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned}
\frac{\partial L}{\partial x_i} &= 0, \; \forall i = 1, 2, \ldots, d & \text{(E.13)}\\
h_i(\mathbf{x}) &\leq 0, \; \forall i = 1, 2, \ldots, q & \text{(E.14)}\\
\lambda_i &\geq 0, \; \forall i = 1, 2, \ldots, q & \text{(E.15)}\\
\lambda_i h_i(\mathbf{x}) &= 0, \; \forall i = 1, 2, \ldots, q. & \text{(E.16)}
\end{aligned}$$

Notice that the Lagrange multipliers are no longer unbounded in the presence of inequality constraints.

Example E.3. Suppose we want to minimize the function $f(x,y) = (x-1)^2 + (y-3)^2$ subject to the following constraints:

$$x + y \leq 2, \quad \text{and} \quad y \geq x.$$

The Lagrangian for this problem is $L = (x-1)^2 + (y-3)^2 + \lambda_1(x+y-2) + \lambda_2(x-y)$ subjected to the following KKT constraints:

$$\frac{\partial L}{\partial x} = 2(x-1) + \lambda_1 + \lambda_2 = 0 \tag{E.17}$$
$$\frac{\partial L}{\partial y} = 2(y-3) + \lambda_1 - \lambda_2 = 0 \tag{E.18}$$
$$\lambda_1(x+y-2) = 0 \tag{E.19}$$
$$\lambda_2(x-y) = 0 \tag{E.20}$$
$$\lambda_1 \geq 0,\ \lambda_2 \geq 0,\ x+y \leq 2,\ y \geq x \tag{E.21}$$

To solve the above equations, we need to examine all the possible cases of Equations E.19 and E.20.

Case 1: $\lambda_1 = 0$, $\lambda_2 = 0$. In this case, we obtain the following equations:

$$2(x-1) = 0 \quad \text{and} \quad 2(y-3) = 0,$$

whose solution is given by $x = 1$ and $y = 3$. Since $x + y = 4$, this is not a feasible solution because it violates the constraint $x + y \leq 2$.

Case 2: $\lambda_1 = 0$, $\lambda_2 \neq 0$. In this case, we obtain the following equations:

$$x - y = 0,\ 2(x-1) + \lambda_2 = 0,\ 2(y-3) - \lambda_2 = 0,$$

whose solution is given by $x = 2$, $y = 2$, and $\lambda_2 = -2$, which is not a feasible solution because it violates the conditions $\lambda_2 \geq 0$ and $x+y \leq 2$.

Case 3: $\lambda_1 \neq 0$, $\lambda_2 = 0$. In this case, we obtain the following equations:

$$x + y - 2 = 0,\ 2(x-1) + \lambda_1 = 0,\ -2(x+1) + \lambda_1 = 0,$$

whose solution is given by $x = 0$, $y = 2$, and $\lambda_1 = 2$, which is a feasible solution.

Case 4: $\lambda_1 \neq 0$, $\lambda_2 \neq 0$. In this case, we obtain the following equations:

$$x + y - 2 = 0,\ x - y = 0,\ 2(x - 1) + \lambda_1 + \lambda_2 = 0,\ 2(y - 3) + \lambda_1 - \lambda_2 = 0,$$

whose solution is $x = 1$, $y = 1$, $\lambda_1 = 2$, and $\lambda_2 = -2$, which is not a feasible solution.

Therefore, the solution for this problem is $x = 0$ and $y = 2$. ■

Solving the KKT conditions can be quite a laborious task especially if the number of constraining inequalities is large. In such cases, finding a closed-form solution is no longer possible and it is necessary to use numerical techniques such as linear and quadratic programming.

F

Big Data: Scaling Up Data Mining Algorithms

In this appendix, we review some of the common techniques used for scaling up data mining algorithms. As mentioned in Chapter 1, scalability is an important challenge as even simple counting problems such as finding the top-k most popular items is computationally expensive when applied to massive data sets. To illustrate this, consider the problem of detecting trending topics in social media postings. Suppose each topic corresponds to a hashtag associated with a social media content posted by users. Figure F.1(a) shows a continuous stream of hashtags generated by various users. Our goal is to automatically determine the most popular hashtags from the data stream.

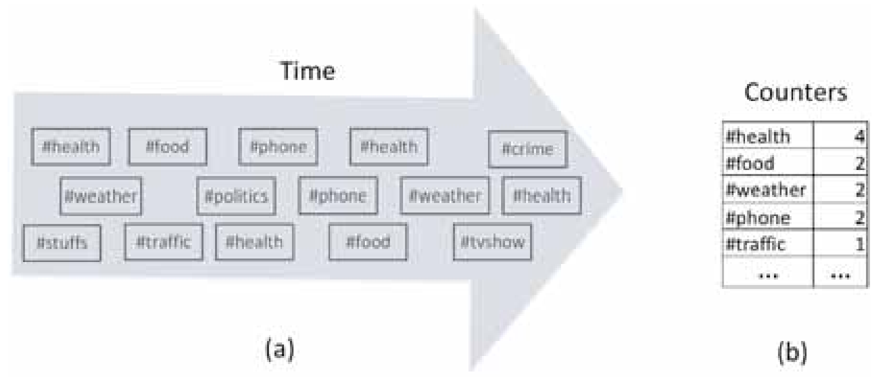


Figure F.1. (a) A data stream of user-generated hashtags. (b) Counters for hashtag frequencies.

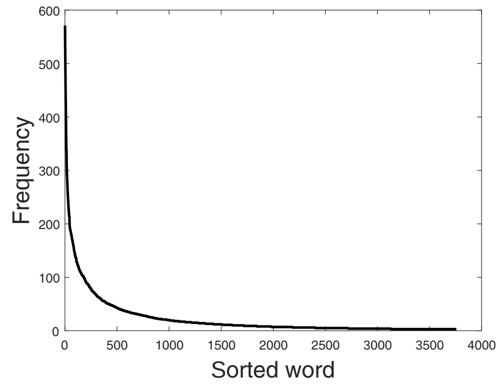


Figure F.2. A Zipf distribution of words that appear in a document corpus.

A simple solution to the problem is to maintain a set of counters for the hashtag frequencies, as shown in Figure F.1(b). Specifically, the counter is incremented each time its corresponding hashtag is observed in the data stream. However, managing a large number of counters and sorting them to identify the most popular hashtags can be a very expensive operation. This appendix presents some of the general strategies that can be used to address this problem. Although these strategies are presented in the context of the hashtag popularity problem, they are generally applicable to many of the data mining techniques presented in this book.

F.1 Sampling-based Approach

A sampling-based approach would estimate the popular hashtags based on a small, random sample of the data. For example, assuming the data is uniformly distributed, one could periodically sample every k-th hashtag in the data stream and count their respective frequencies. Alternatively, one could generate a random number p for each hashtag and add the hashtag to the sample if the random number exceeds a pre-defined threshold. Assuming the hashtags follow a Zipf distribution such as the one shown in Figure F.2, both sampling procedures can reduce the number of counters significantly as most of the rare hashtags are likely to be omitted from the sample. While such simple sampling strategies are sufficient for the hashtag popularity problem, they may not be effective for more sophisticated data mining tasks that require a finite sample size to fit in memory at all times. This is because the sample size grows linearly with increasing length of the data stream for both sampling strategies. For example, if N is the length of the data stream, then the sample size would be N/k for periodic sampling and $N(1-p)$ for random sampling.

The linear growth in sample size may not be suitable for many data mining algorithms that require the input data to reside in main memory. For such algorithms, we would like to create a random sample that fits the amount of memory available. While the sample composition may change dynamically as new instances arrive in the data stream, the sample size needs to fixed. This would allow the data mining algorithm to be periodically applied to the sampled data. Let n be the desired sample size and N be the size of the input data. To ensure uniform sampling, every instance must have the same probability, n/N, to be included in the sample. If N is known, we can create such a uniform sample by generating a random number p for each instance and keep the instance in the sample if $p \geq n/N$. However, if N is unknown or varies dynamically (e.g., for a data stream), it would be impossible to determine the probability by which an instance should be added to the sample data.

To overcome this problem, a procedure known as reservoir sampling [777] can be used. The procedure ensures that every instance encountered so far in the data stream has an equal probability of being included in the sample, regardless of the length of the data stream. If n is the desired sample size and N is the current length of the data stream, then the probability an instance is kept in the sample would be:

$$p = \begin{cases} 1, & \text{if } n \geq N; \\ \frac{n}{N}, & \text{otherwise.} \end{cases}$$

The reservoir sampling procedure works as follows. Assuming the instances in the data stream arrive one at a time, we add the first n instances to the sample. In this situation, every instance has an equal probability of 1 to be included in the sample when the length of the data stream N is less than or equal to n. Next, for each i-th arriving instance, where $i > n$, we generate a random integer r between 1 and i. If $r \leq n$, we replace the r-th instance within the current sample with the new instance. Otherwise, the new instance will be discarded. This ensures that the i-th arriving instance has a probability of n/i to be included the sample. More importantly, it can be shown that each instance currently in the sample also has the same probability of n/i to remain in the sample. We can prove this by induction. Let S_i be the set of sampled instances after encountering the first i instances. The probability that an existing instance $x \in S_{i-1}$ remains in S_i is

$$P(x \in S_i) = P(x \in S_{i-1}) \times P(i \not\rightarrow x),$$

where the first term, $P(x \in S_{i-1})$, refers to the probability x belongs to the sample S_{i-1} whereas the second term $P(i \not\rightarrow x)$ refers to the probability that x is not chosen to be replaced by the i-th instance. By induction, we assume $P(x \in S_{i-1}) = n/(i-1)$. Furthermore, $P(i \not\rightarrow x) = 1 - P(i \rightarrow x)$, where $P(i \rightarrow x)$ is the probability x is replaced by the i-th instance. The latter happens when the following two conditions are satisfied:

1. The random number r associated with the i-th instance is less than or equal to n.
2. The random number r associated with the i-th instance is the same as the position occupied by x in the current sample S_{i-1}.

The probability for the first condition is given by n/i whereas the probability for the second condition is given by $1/n$. Putting it together, we have

$$P(i \not\rightarrow x) = 1 - P(i \rightarrow x) = 1 - \frac{n}{i} \times \frac{1}{n} = 1 - \frac{1}{i}$$

Thus, the probability that the item x remains in the sample S_i is

$$P(x \in S_i) = P(x \in S_{i-1}) \times P(i \not\rightarrow x) = \frac{n}{i-1}\left(1 - \frac{1}{i}\right) = \frac{n}{i},$$

which completes the proof.

Sampling-based approaches are useful when the patterns to be discovered appear frequently in the data. The random sample must be representative of the underlying characteristics of the input data. These approaches may not be effective at detecting infrequent events such as anomalies or for modeling rare classes in the data.

F.2 Parallel/Distributed Approach

Another way to scale up existing data mining algorithms is to allow multiple processing nodes in a cluster of machines to analyze the input data. This strategy is useful when the computational resources needed to apply the data mining algorithm far exceeds the capacity provided by a single machine. The massive data set can be initially split into smaller chunks. Each data chunk is then assigned to one of the nodes in the cluster for local processing. The partial results generated by all the processing nodes are then aggregated to obtain a global solution to the computational problem. Such a strategy is employed by

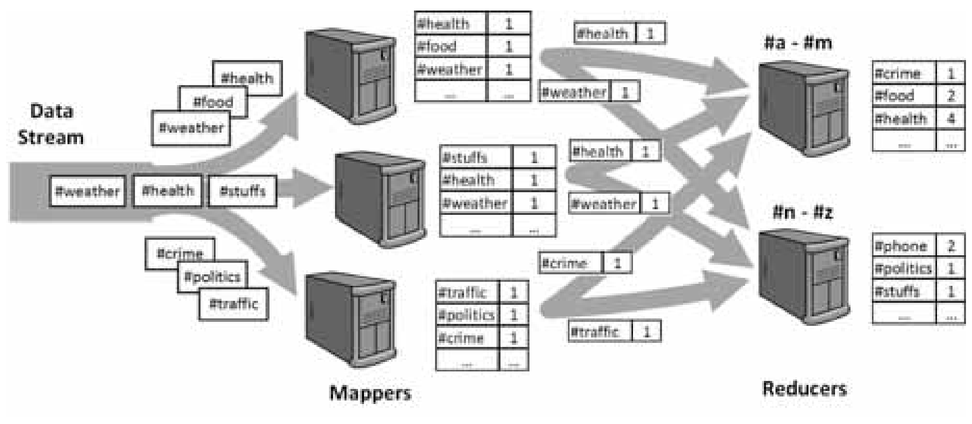


Figure F.3. Frequency counting of hashtags using a MapReduce framework.

the MapReduce framework [775], which is a distributed computational model for mining big data. The framework decomposes a large-scale computational problem into two phases. During the map phase, each processing node in the cluster would perform a local computation on the data chunks it owns. This will be followed by a reduce phase, where the local results are aggregated to obtain the final solution.

Figure F.3 illustrates an example application of MapReduce to the hashtag popularity problem. The framework is implemented on a cluster of processing nodes, which can be configured to serve as mappers or reducers (or both) of the computational task. The advantage of using a distributed programming model such as MapReduce is that it hides many of the implementation details, such as, process synchronization, load balancing, fault tolerance, and so on, from programmers, thus simplifying the process of writing distributed programs. Instead, the programmers need to focus only on how to decompose the computational problem into a chain of map and reduce tasks.

For the hashtag popularity problem, the input stream can be split among the mappers. Each mapper is responsible for counting the frequencies of hashtags assigned to the mapper. The mapper will then output a list of key-value pairs, in which the key would correspond to a hashtag and the value corresponds to its local frequency count. The key-value pairs are then distributed to the reducers based on their key values. For example, with 2 reducers, all hashtags that begin with #a through #m can be assigned to the first reducer while those that begin with #n through #z can be assigned to the second reducer (see Figure F.3). The reducers are responsible for summing up the frequencies for each hashtag it receives from the mappers. The reducers will

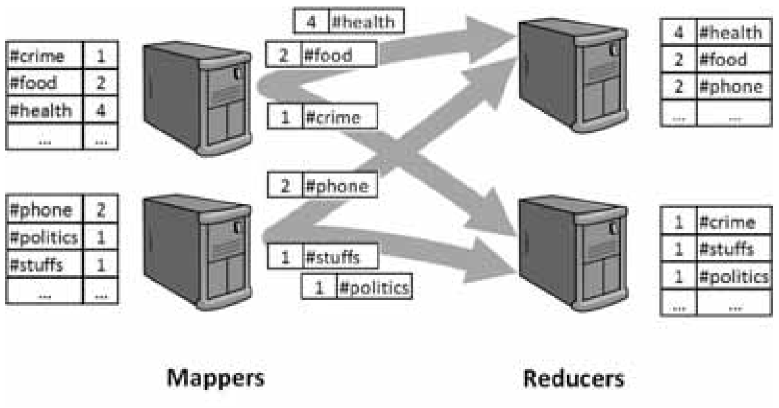


Figure F.4. A second MapReduce job for sorting the frequencies of keyword tags.

then output the hashtags along with their aggregated frequencies. Since we are interested in identifying the most popular hashtags, a second MapReduce job is needed, whereby each mapper will receive one of the reducer's output from the first job as its input data. The mapper transmits the frequency as its key and the hashtag associated with the given frequency as its value. The reducers will then collect all the hashtags with the same frequency, sort them, and generate the sorted frequencies along with their list of hashtags as output (see Figure F.4).

The MapReduce framework is generally applicable to more complex problems such as matrix computations and gradient-based optimization by designing the appropriate mapper and reducer tasks. It is highly scalable especially for *embarrassingly parallel* problems, where there are few dependencies among the mapper or reducer tasks.

F.3 Online/Incremental Learning Approach

A third strategy for scaling up a data mining technique is to develop an online implementation of the algorithm. Similar to the sampling-based approach, online learning does not require the entire data to fit into main memory. Instead, the algorithm derives an initial model from a subset of the input data and then update the model incrementally with the remaining instances. This distinguishes online learning from sampling-based approaches as the latter completely ignores instances that were excluded from the data sample.

To illustrate the online learning approach, we consider the problem of identifying popular hashtags from social media postings. Let d be the number of unique hashtags in the data and $n \ll d$ is the size of memory buffer available to store the counters for the hashtag frequencies. Due to the limited memory size, it is not possible to store a counter for each hashtag. An online learning algorithm such as Misra-Gries [776] can be applied to identify popular hashtags despite the memory limitation.

For each newly arriving instance in the data stream, the algorithm processes the new instance as follows:

1. Examine the hashtag of the instance. If the hashtag is already stored in the memory buffer, go to Step 2. Otherwise, go to Step 3.
2. Increment the counter associated with the hashtag by 1. Go to Step 5.
3. If the memory buffer is not full, create a new counter for the hashtag and initialize its count to 1. Go to Step 5. Otherwise, if the memory buffer is full, go to Step 4.
4. Decrement the counter of every hashtag in the memory buffer by 1. If the frequency reduces to 0, remove the hashtag and its counter from the memory buffer. Go to Step 5.
5. Fetch the next instance from the data stream.

For example, consider the data stream shown in Figure F.5. Assume the buffer can store the frequency counts for up to 2 hashtags only. Thus, the buffer becomes full after observing the first two data instances. When the third instance is processed, its hashtag, `#health`, already exists in the buffer. So its frequency is incremented by 1, as shown in Figure F.5(c). When the fourth instance is encountered, its hashtag, `#weather`, is not available in the memory buffer. Therefore, we reduce the frequencies for all the hashtags in memory by 1. One of the existing keyword tags, `#stuffs`, is removed from the buffer since its frequency is equal to 0. This frees up the buffer to store the frequency for the next new hashtag encountered in the data stream. After processing the fifth instance, the state of the buffer is shown in Figure F.5(e). As can be seen from the figure, the most popular keyword tag, `#health`, remains in the memory buffer.

It can be theoretically proven that all popular hashtags whose frequencies exceed $\lfloor N/(n+1) \rfloor$ are guaranteed to be in the memory buffer, where N is the length of the data stream that has been processed so far. For the example shown in Figure F.5, $n = 2$ and $N = 5$. Thus, all popular keyword tags whose frequencies are above $\lfloor 5/3 \rfloor = 1$ will be in the memory buffer.

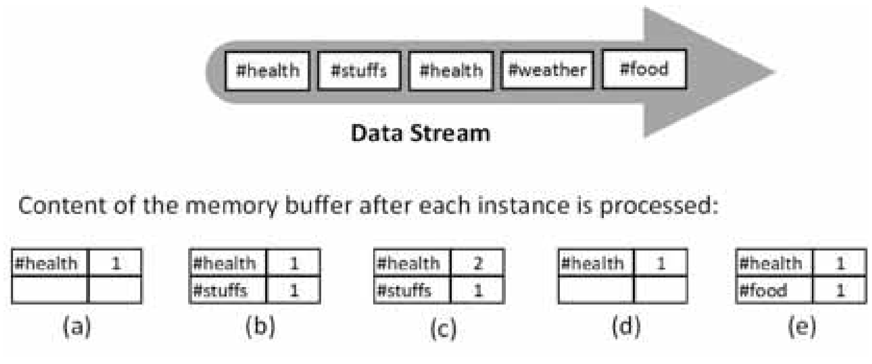


Figure F.5. Example application of the Misra-Gries algorithm for finding popular hashtags.

The advantage of using online learning is that it is a single-pass algorithm, which makes it scalable to processing massive data streams. Unlike the sampling-based approach, online learning algorithms would process all the instances in the input data stream, instead of a random sample from the stream. Nevertheless, it provides only an approximate solution to the computational problem, which means, its solution may include erroneous results. For instance, not all hashtags that remain in the buffer after applying the Misra-Gries algorithm to the data stream shown in Figure F.5 correspond to popular hashtags (i.e., appear more than once).

Bibliography

[775] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proc. of OSDI*, pages 137–150, San Francisco, California, 2004.

[776] J. Misra and D. Gries. Finding Repeated Elements. *Sci. Comput. Program.*, 2(2): 143–152, 1982.

[777] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.