Table 5.2

The SPEED and AGILITY ratings for 20 college athletes and whether they were drafted by a professional team.

ID	SPEED	AGILITY	DRAFT	ID	SPEED	AGILITY	DRAFT
1	2.50	6.00	no	11	2.00	2.00	no
2	3.75	8.00	no	12	5.00	2.50	no
3	2.25	5.50	no	13	8.25	8.50	no
4	3.25	8.25	no	14	5.75	8.75	yes
5	2.75	7.50	no	15	4.75	6.25	yes
6	4.50	5.00	no	16	5.50	6.75	yes
7	3.50	5.25	no	17	5.25	9.50	yes
8	3.00	3.25	no	18	7.00	4.25	yes
9	4.00	4.00	no	19	7.50	8.00	yes
10	4.25	3.75	no	20	7.25	5.75	yes

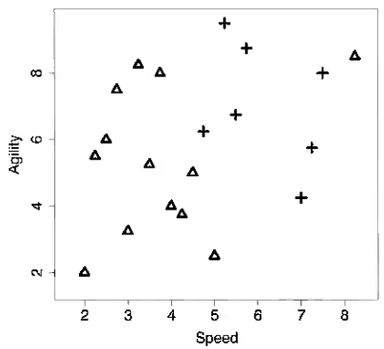

Figure 5.1

A feature space plot of the college athlete data in Table 5.2[182].

the feature space. Also, as the differences between the values of the descriptive features of two instances grows, so too does the distance between the points in the feature space that represent these instances. So the distance between two points in the feature space is a useful measure of the similarity of the descriptive features of the two instances.

5.2.2 Measuring Similarity Using Distance Metrics

The simplest way to measure the similarity between two instances, $\mathbf{a}$ and $\mathbf{b}$, in a dataset is to measure the distance between the instances in a feature space. We can use a **distance metric** to do this: $metric(\mathbf{a}, \mathbf{b})$ is a function that returns the distance between two instances $\mathbf{a}$ and $\mathbf{b}$. Mathematically, a **metric** must conform to the following four criteria:

1. **Non-negativity**: $metric(\mathbf{a}, \mathbf{b}) \geqslant 0$
2. **Identity**: $metric(\mathbf{a}, \mathbf{b}) = 0 \Longleftrightarrow \mathbf{a} = \mathbf{b}$
3. **Symmetry**: $metric(\mathbf{a}, \mathbf{b}) = metric(\mathbf{b}, \mathbf{a})$
4. **Triangular Inequality**: $metric(\mathbf{a}, \mathbf{b}) \leqslant metric(\mathbf{a}, \mathbf{c}) + metric(\mathbf{b}, \mathbf{c})$

One of the best known distance metrics is **Euclidean distance**, which computes the length of the straight line between two points. Euclidean distance between two instances $\mathbf{a}$ and $\mathbf{b}$ in an m-dimensional feature space is defined as

$$Euclidean(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{m} (\mathbf{a}[i] - \mathbf{b}[i])^2} \tag{5.1}$$

The descriptive features in the college athlete dataset are both continuous, which means that the feature space representing this data is technically known as a **Euclidean coordinate space**, and we can compute the distance between instances in it using Euclidean distance. For example, the Euclidean distance between instances $\mathbf{d}_{12}$ (SPEED = 5.00, AGILITY = 2.50) and $\mathbf{d}_5$ (SPEED = 2.75, AGILITY = 7.50) from Table 5.2[182] is

$$\begin{aligned} Euclidean(\mathbf{d}_{12}, \mathbf{d}_5) &= \sqrt{(5.00 - 2.75)^2 + (2.50 - 7.50)^2} \\ &= \sqrt{30.0625} = 5.4829 \end{aligned}$$

Another, less well-known, distance metric is the **Manhattan distance**.[2] The Manhattan distance between two instances $\mathbf{a}$ and $\mathbf{b}$ in a feature space with m dimensions is defined as

$$Manhattan(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{m} abs(\mathbf{a}[i] - \mathbf{b}[i]) \tag{5.2}$$

2 The Manhattan distance, or **taxi-cab distance**, is so called because it is the distance that a taxi driver would have to cover if going from one point to another on a road system that is laid out in blocks, like the Manhattan road system.

where the *abs*() function returns the absolute value. For example, the Manhattan distance between instances $\mathbf{d}_{12}$ (SPEED = 5.00, AGILITY = 2.50) and $\mathbf{d}_5$ (SPEED = 2.75, AGILITY = 7.50) in Table 5.2[182] is

$$\begin{aligned} Manhattan(\mathbf{d}_{12}, \mathbf{d}_5) &= abs(5.00 - 2.75) + abs(2.5 - 7.5) \\ &= 2.25 + 5 = 7.25 \end{aligned}$$

Figure 5.2(a)[184] illustrates the difference between the Manhattan and Euclidean distances between two points in a two-dimensional feature space. If we compare Equation (5.1)[183] and Equation (5.2)[183], we can see that both distance metrics are essentially functions of the differences between the values of the features. Indeed, the Euclidean and Manhattan distances are special cases of the **Minkowski distance**, which defines a family of distance metrics based on differences between features.

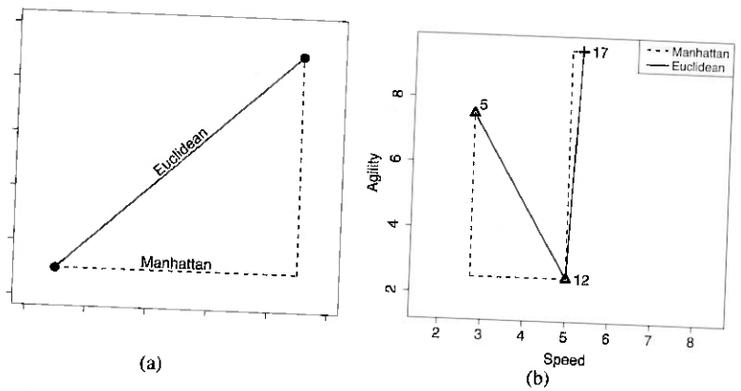

Figure 5.2

(a) A generalized illustration of the Manhattan and Euclidean distances between two points; (b) a plot of the Manhattan and Euclidean distances between instances $\mathbf{d}_{12}$ and $\mathbf{d}_5$ and between $\mathbf{d}_{12}$ and $\mathbf{d}_{17}$ from Table 5.2[182].

The **Minkowski distance** between two instances $\mathbf{a}$ and $\mathbf{b}$ in a feature space with m descriptive features is defined as

$$Minkowski(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^{m} abs(\mathbf{a}[i] - \mathbf{b}[i])^p \right)^{\frac{1}{p}} \tag{5.3}$$

where the parameter p is typically set to a positive value and defines the behavior of the distance metric. Different distance metrics result from adjusting the

value of p. For example, the Minkowski distance with $p = 1$ is the Manhattan distance, and with $p = 2$ is the Euclidean distance. Continuing in this manner, we can define an infinite number of distance metrics.

The fact that we can define an infinite number of distance metrics is not merely an academic curiosity. In fact, the predictions produced by a similarity-based model will change depending on the exact Minkowski distance used (i.e., $p = 1, 2, \ldots, \infty$). Larger values of p place more emphasis on large differences between feature values than smaller values of p because all differences are raised to the power of p. Consequently, the Euclidean distance (with $p = 2$) is more strongly influenced by a single large difference in one feature than the Manhattan distance (with $p = 1$).[3]

We can see this if we compare the Euclidean and Manhattan distances between instances $\mathbf{d}_{12}$ and $\mathbf{d}_5$ with the Euclidean and Manhattan distances between instances $\mathbf{d}_{12}$ and $\mathbf{d}_{17}$ (SPEED = 5.25, AGILITY = 9.50). Figure 5.2(b)[184] plots the Manhattan and Euclidean distances between these pairs of instances.

The Manhattan distances between both pairs of instances are the same: 7.25. It is striking, however, that the Euclidean distance between $\mathbf{d}_{12}$ and $\mathbf{d}_{17}$ is 8.25, which is greater than the Euclidean distance between $\mathbf{d}_{12}$ and $\mathbf{d}_5$, which is just 5.48. This is because the maximum difference between $\mathbf{d}_{12}$ and $\mathbf{d}_{17}$ for any single feature is 7 units (for AGILITY), whereas the maximum difference between $\mathbf{d}_{12}$ and $\mathbf{d}_5$ on any single feature is just 5 units (for AGILITY). Because these differences are squared in the Euclidean distance calculation, the larger maximum single difference between $\mathbf{d}_{12}$ and $\mathbf{d}_{17}$ results in a larger overall distance being calculated for this pair of instances. Overall the Euclidean distance weights features with larger differences in values more than features with smaller differences in values. This means that the Euclidean difference is more influenced by a single large difference in one feature rather than a lot of small differences across a set of features, whereas the opposite is true of Manhattan distance.

Although we have an infinite number of Minkowski-based distance metrics to choose from, Euclidean distance and Manhattan distance are the most commonly used of these. The question of which is the best one to use, however, still remains. From a computational perspective, the Manhattan distance has a

3 In the extreme case with $p = \infty$ the Minkowski metric simply returns the maximum difference between any of the features. This is known as the **Chebyshev distance** but is also sometimes called the **chessboard distance** because it is the number of moves a king must make in chess to go from one square on the board to any other square.

similarity in similarity-based models that do not satisfy all four of these criteria. We refer to measures of similarity of this type as **indexes**. Most of the time the technical distinction between a metric and an index is not that important; we simply focus on choosing the right measure of similarity for the type of instances we are comparing. It is important, however, to know if a measure is a metric or an index as there are some similarity-based techniques that strictly require measures of similarity to be metrics. For example, the ***k-d* trees** described in Section 5.4.2[195] require that the measure of similarity used be a metric (in particular that the measure conform to the triangular inequality constraint).

5.4.5.1 Similarity Indexes for Binary Descriptive Features

There are lots of datasets that contain binary descriptive features—categorical features that have only two levels. For example, a dataset may record whether or not someone liked a movie, a customer bought a product, or someone visited a particular webpage. If the descriptive features in a dataset are binary, it is often a good idea to use a **similarity index** that defines similarity between instances specifically in terms of **co-presence** or **co-absence** of features, rather than an index based on distance.

To illustrate a series of similarity indexes for binary descriptive features, we will use an example of predicting **upsell** in an online service. A common business model for online services is to allow users a free trial period after which time they have to *sign up* to a paid account to continue using the service. These businesses often try to predict the likelihood that users coming to the end of the trial period will accept the upsell offer to move to the paid service. This insight into the likely future behavior of a customer can help a marketing department decide which customers coming close to the end of their trial period the department should contact to promote the benefits of signup to the paid service.

Table 5.11[215] lists a small binary dataset that a nearest neighbor model could use to make predictions for this scenario. The descriptive features in this dataset are all binary and record the following information about the behavior of past customers:

- PROFILE: Did the user complete the profile form when registering for the free trial?
- FAQ: Did the user read the frequently asked questions page?
- HELPFORUM: Did the user post a question on the help forum?

Table 5.11
A binary dataset listing the behavior of two individuals on a website during a trial period and whether they subsequently signed up for the website.

ID	PROFILE	FAQ	HELPFORUM	NEWSLETTER	LIKED	SIGNUP
1	true	true	true	false	true	yes
2	true	false	false	false	false	no

- NEWSLETTER: Did the user sign up for the weekly newsletter?
- LIKED: Did the user *Like* the website on Facebook?

The target feature, SIGNUP, indicates whether the customers ultimately signed up to the paid service or not (*yes* or *no*).

The business has decided to use a nearest neighbor model to predict whether a current trial user whose free trial period is about the end is likely to sign up for the paid service. The query instance, **q**, describing this user is:

PROFILE = *true*, FAQ = *false*, HELPFORUM = *true*,
NEWSLETTER = *false*, LIKED = *false*

Table 5.12[216] presents a pairwise analysis of similarity between the current trial user, **q**, and the two customers in the dataset in Table 5.11[215] in terms of

- **co-presence** (CP), how often a true value occurred for the same feature in both the query data **q** and the data for the comparison user ($\mathbf{d}_1$ or $\mathbf{d}_2$)
- **co-absence** (CA), how often a false value occurred for the same feature in both the query data **q** and the data for the comparison user ($\mathbf{d}_1$ or $\mathbf{d}_2$)
- **presence-absence** (PA), how often a true value occurred in the query data **q** when a false value occurred in the data for the comparison user ($\mathbf{d}_1$ or $\mathbf{d}_2$) for the same feature
- **absence-presence** (AP), how often a false value occurred in the query data **q** when a true value occurred in the data for the comparison user ($\mathbf{d}_1$ or $\mathbf{d}_2$) for the same feature

One way of judging similarity is to focus solely on co-presence. For example, in an online retail setting, co-presence could capture what two users jointly viewed, liked, or bought. The **Russel-Rao** similarity index focuses on this and is measured in terms of the ratio between the number of co-presences and the

Table 5.12
The similarity between the current trial user, $\mathbf{q}$, and the two users in the dataset, $\mathbf{d}_1$ and $\mathbf{d}_2$, in terms of co-presence (CP), co-absence (CA), presence-absence (PA), and absence-presence (AP).

		$\mathbf{q}$	
		Pres.	Abs.
$\mathbf{d}_1$	Pres.	CP = 2	PA = 0
	Abs.	AP = 2	CA = 1

		$\mathbf{q}$	
		Pres.	Abs.
$\mathbf{d}_2$	Pres.	CP = 1	PA = 1
	Abs.	AP = 0	CA = 3

total number of binary features considered:

$$sim_{RR}(\mathbf{q},\mathbf{d}) = \frac{CP(\mathbf{q},\mathbf{d})}{|\mathbf{q}|} \tag{5.9}$$

where $\mathbf{q}$ and $\mathbf{d}$ are two instances, $|\mathbf{q}|$ is the total number of features in the dataset, and $CP(\mathbf{q},\mathbf{d})$ measures the total number of co-presences between $\mathbf{q}$ and $\mathbf{d}$. Using Russel-Rao, $\mathbf{q}$ has a higher similarity to $\mathbf{d}_1$ than to $\mathbf{d}_2$:

$$sim_{RR}(\mathbf{q},\mathbf{d}_1) = \frac{2}{5} = 0.4$$
$$sim_{RR}(\mathbf{q},\mathbf{d}_2) = \frac{1}{5} = 0.2$$

This means that the current trial user is judged to be more similar to the customer represented by instance $\mathbf{d}_1$ than the customer represented by instance $\mathbf{d}_2$.

In some domains co-absence is important. For example, in a medical domain when judging the similarity between two patients, it may be as important to capture the fact that neither patient had a particular symptom as it is to capture the symptoms that the patients have in common. The **Sokal-Michener** similarity index takes this into account and is defined as the ratio between the total number of co-presences and co-absences and the total number of binary features considered:

$$sim_{SM}(\mathbf{q},\mathbf{d}) = \frac{CP(\mathbf{q},\mathbf{d}) + CA(\mathbf{q},\mathbf{d})}{|\mathbf{q}|} \tag{5.10}$$

Using Sokal-Michener for our online services example $\mathbf{q}$, is judged to be more similar to instance $\mathbf{d}_2$ than instance $\mathbf{d}_1$:

$$sim_{SM}(\mathbf{q},\mathbf{d}_1) = \frac{3}{5} = 0.6$$
$$sim_{SM}(\mathbf{q},\mathbf{d}_2) = \frac{4}{5} = 0.8$$

Sometimes, however, co-absences aren't that meaningful. For example, we may be in a retail domain in which there are so many items that most people haven't seen, listened to, bought, or visited the vast majority of them, and as a result, the majority of features will be co-absences. The technical term to describe a dataset in which most of the features have zero values is **sparse data**. In these situations we should use a metric that ignores co-absences. The **Jaccard** similarity index is often used in these contexts. This index ignores co-absences and is defined as the ratio between the number of co-presences and the total number of features, excluding those that record a co-absence between a pair of instances:[18]

$$sim_J(\mathbf{q},\mathbf{d}) = \frac{CP(\mathbf{q},\mathbf{d})}{CP(\mathbf{q},\mathbf{d}) + PA(\mathbf{q},\mathbf{d}) + AP(\mathbf{q},\mathbf{d})} \tag{5.11}$$

Using Jaccard similarity, the current trial user in the online retail example is judged to be equally similar to instance $\mathbf{d}_1$ and $\mathbf{d}_2$:

$$sim_J(\mathbf{q},\mathbf{d}_1) = \frac{2}{4} = 0.5$$
$$sim_J(\mathbf{q},\mathbf{d}_2) = \frac{1}{2} = 0.5$$

The fact that the judgment of similarity between current trial user and the other users in the dataset changed dramatically depending on which similarity index was employed illustrates the importance of choosing the correct index for the task. Unfortunately, beyond highlighting that the Jaccard index is useful for sparse binary data, we cannot give a hard and fast rule for how to choose between these indexes. As is so often the case in predictive analytics, making the right choice requires an understanding of the requirements of the task that

18 One note of caution. The Jaccard similarity index is undefined for pairs of instances where all the features manifest co-absence as this leads to a division by zero.

we are trying to accomplish and matching these requirements with the features we want to emphasize in our model.

5.4.5.2 Cosine Similarity

Cosine similarity is an index that can be used as a measure of the similarity between instances with continuous descriptive features. The cosine similarity between two instances is the **cosine** of the inner angle between the two **vectors** that extend from the origin of a feature space to each instance. Figure 5.14(a)[220] illustrates the inner angle, θ, between the vector from the origin to two instances in a feature space defined by two descriptive features, SMS and VOICE.

Cosine similarity is an especially useful measure of similarity when the descriptive features describing instances in a dataset are related to each other. For example, in a mobile telecoms scenario, we could represent customers with just two descriptive features: the average number of SMS messages a customer sends per month, and the average number of VOICE calls a customer makes per month. In this scenario it is interesting to take a perspective on the similarity between customers that focuses on the mix of these two types of services they use, rather than the volumes of the services they use. Cosine similarity allows us to do this. The instances shown in Figure 5.14(a)[220] are based on this mobile telecoms scenario. The descriptive feature values for $\mathbf{d}_1$ are SMS = 97 and VOICE = 21, and for $\mathbf{d}_2$ are SMS = 181 and VOICE = 184.

We compute the cosine similarity between two instances as the normalized **dot product** of the descriptive feature values of the instances. The dot product is normalized by the product of the lengths of the descriptive feature value vectors.[19] The dot product of two instances, $\mathbf{a}$ and $\mathbf{b}$, defined by m descriptive features is

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{m} (\mathbf{a}[i] \times \mathbf{b}[i]) = (\mathbf{a}[1] \times \mathbf{b}[1]) + \cdots + (\mathbf{a}[m] \times \mathbf{b}[m]) \quad (5.12)$$

19 The length of a vector, $|\mathbf{a}|$, is computed as the square root of the sum of the elements of the vector squared: $|\mathbf{a}| = \sqrt{\sum_{i=1}^{m} \mathbf{a}[i]^2}$.

Geometrically, the dot product can be interpreted as equivalent to the cosine of the angle between the two vectors multiplied by the length of the two vectors:

$$\mathbf{a} \cdot \mathbf{b} = \sqrt{\sum_{i=1}^{m} \mathbf{a}[i]^2} \times \sqrt{\sum_{i=1}^{m} \mathbf{b}[i]^2} \times cos(\theta) \quad (5.13)$$

We can rearrange Equation (5.13)[219] to calculate the cosine of the inner angle between two vectors as the normalized dot product:

$$\frac{\mathbf{a} \cdot \mathbf{b}}{\sqrt{\sum_{i=1}^{m} \mathbf{a}[i]^2} \times \sqrt{\sum_{i=1}^{m} \mathbf{b}[i]^2}} = cos(\theta) \quad (5.14)$$

So, in an m-dimensional feature space, the cosine similarity between two instances $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$\begin{aligned} sim_{COSINE}(\mathbf{a}, \mathbf{b}) &= \frac{\mathbf{a} \cdot \mathbf{b}}{\sqrt{\sum_{i=1}^{m} \mathbf{a}[i]^2} \times \sqrt{\sum_{i=1}^{m} \mathbf{b}[i]^2}} \\ &= \frac{\sum_{i=1}^{m} (\mathbf{a}[i] \times \mathbf{b}[i])}{\sqrt{\sum_{i=1}^{m} \mathbf{a}[i]^2} \times \sqrt{\sum_{i=1}^{m} \mathbf{b}[i]^2}} \end{aligned} \quad (5.15)$$

The cosine similarity between instances will be in the range $[0, 1]$, where 1 indicates maximum similarity and 0 indicates maximum dissimilarity.[20] We can calculate the cosine similarity between $\mathbf{d}_1$ and $\mathbf{d}_2$ from Figure 5.14(a)[220] as

$$\begin{aligned} sim_{COSINE}(\mathbf{d}_1, \mathbf{d}_1) &= \frac{(97 \times 181) + (21 \times 184)}{\sqrt{97^2 + 21^2} \times \sqrt{181^2 + 184^2}} \\ &= 0.8362 \end{aligned}$$

20 If either vector used to calculate a cosine similarity contains negative feature values, then the cosine similarity will actually be in the range $[-1, 1]$. As before, 1 indicates high similarity, and 0 indicates dissimilarity, but it can be difficult to interpret negative similarity scores. Negative similarity values can be avoided, however, if we use range normalization (see Section 3.6.1[92]) to ensure that descriptive feature values always remain positive.

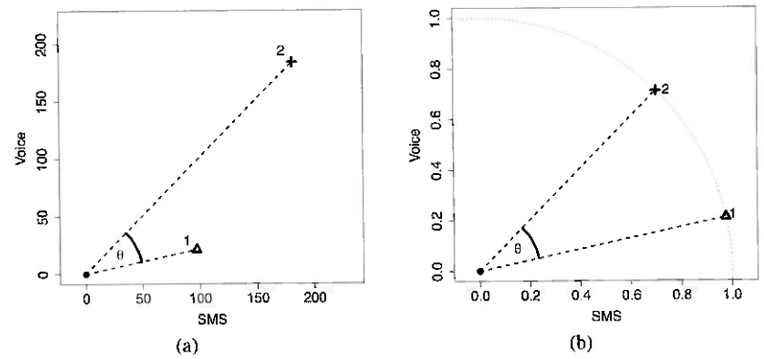

Figure 5.14

(a) The θ represents the inner angle between the vector emanating from the origin to instance $\mathbf{d}_1$ and the vector emanating from the origin to instance $\mathbf{d}_2$; (b) shows $\mathbf{d}_1$ and $\mathbf{d}_2$ normalized to the unit circle.

Figure 5.14(b)[220] highlights the normalization of descriptive feature values that takes place as part of calculating cosine similarity. This is different from the normalization we have looked at elsewhere in this chapter as it takes place within an instance rather than across all the values of a feature. All instances are normalized so as to lie on a **hypersphere** of radius 1.0 with its center at the origin of the feature space. This normalization is what makes cosine similarity so useful in scenarios in which we are interested in the relative spread of values across a set of descriptive features rather than the magnitudes of the values themselves. For example, if we have a third instance with SMS = 194 and VOICE = 42, the cosine similarity between this instance and $\mathbf{d}_1$ will be 1.0, because even though the magnitudes of their feature values are different, the relationship between the feature values for both instances is the same: both customers use about four times as many SMS messages as VOICE calls. Cosine similarity is also an appropriate similarity index for sparse data with non-binary features (i.e., datasets with lots of zero values) because the dot product will essentially ignore co-absences in its computation ($0 \times 0 = 0$).

5.4.5.3 Mahalanobis Distance

The final measure of similarity that we will introduce is the **Mahalanobis distance**, which is a metric that can be used to measure the similarity between instances with continuous descriptive features. The Mahalanobis distance is different from the other distance metrics we have looked at because it allows us to take into account how spread out the instances in a dataset are when judging similarities. Figure 5.15[221] illustrates why this is important. This figure shows scatter plots for three bivariate datasets that have the same central tendency, marked A and located in the feature space at $(50, 50)$, but whose instances are spread out differently across the feature space. In all three cases the question we would like to answer is, are instance B, located at at $(30, 70)$, and instance C, located at $(70, 70)$, likely to be from the same population from which the dataset has been sampled? In all three figures, B and C are equidistant from A based on Euclidean distance.

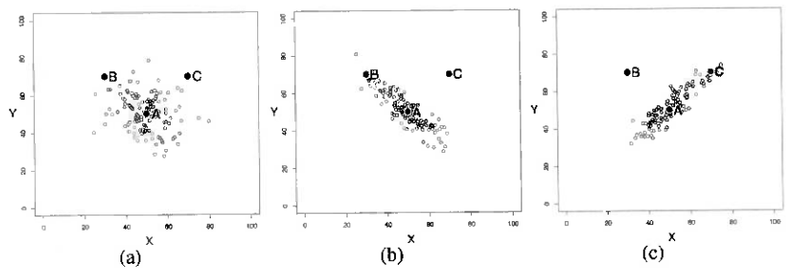

Figure 5.15

Scatter plots of three bivariate datasets with the same center point A and two queries B and C both equidistant from A; (a) a dataset uniformly spread around the center point; (b) a dataset with negative covariance; and (c) a dataset with positive covariance.

The dataset in Figure 5.15(a)[221] is equally distributed in all directions around A, and as a result, we can say that B and C are equally likely to be from the same population as the dataset. The dataset in Figure 5.15(b)[221], however, demonstrates a strong negative **covariance**[21] between the features. In this context, instance B is much more likely to be a member of the dataset than instance C. Figure 5.15(c)[221] shows a dataset with a strong positive covariance, and for this dataset, instance C is much more likely to be a member than instance B.

21 Covariance between features means that knowing the value of one feature tells us something about the value of the other feature. See Section 3.5.2[80] for more information.

What these examples demonstrate is that when we are trying to decide whether a query belongs to a group, we need to consider not only the central tendency of the group, but also how spread out the members in a group are. These examples also highlight that covariance is one way of measuring the spread of a dataset.

The Mahalanobis distance uses covariance to scale distances so that distances along a direction where the dataset is very spread out are scaled down, and distances along directions where the dataset is tightly packed are scaled up. For example, in Figure 5.15(b)[221] the Mahalanobis distance between B and A will be less than the Mahalanobis distance between C and A, whereas in Figure 5.15(c)[221] the opposite will be true. The Mahalanobis distance is defined as

$$Mahalanobis(\mathbf{a},\mathbf{b}) = \sqrt{\left[\mathbf{a}[1]-\mathbf{b}[1],\ldots,\mathbf{a}[m]-\mathbf{b}[m]\right] \times \sum^{-1} \times \begin{bmatrix} \mathbf{a}[1]-\mathbf{b}[1] \\ \vdots \\ \mathbf{a}[m]-\mathbf{b}[m] \end{bmatrix}} \quad (5.16)$$

Let's step through Equation (5.16)[222] bit by bit. First, this equation computes a distance between two instances $\mathbf{a}$ and $\mathbf{b}$, each with m descriptive features. The first big term we come to in the equation is $[\mathbf{a}[1]-\mathbf{b}[1],\ldots,\mathbf{a}[m]-\mathbf{b}[m]]$. This is a row vector that is created by subtracting each descriptive feature value of instance $\mathbf{b}$ from the corresponding feature values of $\mathbf{a}$. The next term in the equation, $\sum^{-1}$, represents the **inverse covariance matrix**[22] computed across all instances in the dataset. Multiplying the difference in feature values by the inverse covariance matrix has two effects. First, the larger the **variance** of a feature, the less weight the difference between the values for that feature will contribute to the distance calculation. Second, the larger the correlation between two features, the less weight they contribute to the distance. The final

22 We explain **covariance matrices** in Section 3.5.2[86]. The **inverse covariance matrix** is the matrix such that when the covariance matrix is multiplied by its inverse, the result is the **identity matrix**: $\sum \times \sum^{-1} = \mathbb{I}$. The identity matrix is a square matrix in which all the elements of the main diagonal are 1, and all other elements are 0. Multiplying any matrix by the identity matrix leaves the original matrix unchanged—this is the equivalent of multiplying by 1 for real numbers. So the effect of multiplying feature values be an inverse covariance matrix is to rescale the variances of all features to 1 and to set the covariance between all feature pairs to 0. Calculating the inverse of a matrix involves solving systems of linear equations and requires the use of techniques from linear algebra such as **Gauss-Jordan elimination** or **LU decomposition**. We do not cover these techniques here, but they are covered in most standard linear algebra textbooks such as Anton and Rorres (2010).

element of the equation is a column vector that is created in the same way as the row vector at the beginning of the equation—by subtracting each feature value from $\mathbf{b}$ from the corresponding feature value from $\mathbf{a}$. The motivation for using a row vector to hold one copy of the feature differences and a column vector to hold the second copy of the features differences is to facilitate matrix multiplication. Now that we know that the row and column vector both contain the difference between the feature values of the two instances, it should be clear that, similar to Euclidean distance, the Mahalanobis distance squares the differences of the features. The Mahalanobis distance, however, also rescales the differences between feature values (using the inverse covariance matrix) so that all the features have unit variance, and the effects of covariance are removed.

The Mahalanobis distance can be understood as defining an orthonormal coordinate system with (1) an origin at the instance we are calculating the distance from ($\mathbf{a}$ in Equation (5.16)[222]); (2) a primary axis aligned with the direction of the greatest spread in the dataset; and (3) the units of all the axes scaled so that the dataset has unit variance along each axis. The rotation and scaling of the axes are the result of the multiplication by the inverse covariance matrix of the dataset ($\sum^{-1}$). So, if the inverse covariance matrix is the identity matrix $\mathbb{I}$, then no scaling or rotation occurs. This is why for datasets such as the one depicted in Figure 5.15(a)[221], where there is no covariance between the features, the Mahalanobis distance is simply the Euclidean distance.[23]

Figure 5.16[224] illustrates how the Mahalanobis distance defines this coordinate system, which is translated, rotated, and scaled with respect to the standard coordinates of a feature space. The three scatter plots in this image are of the dataset in Figure 5.15(c)[221]. In each case we have overlaid the coordinate system defined by the Mahalanobis distance from a different origin. The origins used for the figures were (a) $(50,50)$, (b) $(63,71)$, and (c) $(42,35)$. The dashed lines plot the axes of the coordinate system, and the ellipses plot the 1, 3, and 5 unit distance contours. Notice how the orientation of the axes and the scaling of the distance contours are consistent across the figures. This is because the same inverse covariance matrix based on the entire dataset was used in each case.

23 The inverse of the identity matrix $\mathbb{I}$ is $\mathbb{I}$. So, if there is no covariance between the features, both the covariance and the inverse covariance matrix will be equal to $\mathbb{I}$.

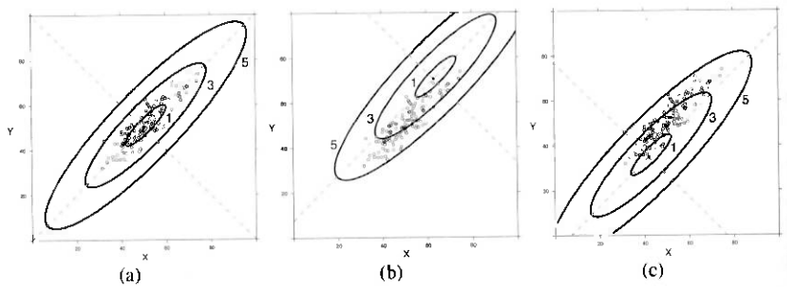

Figure 5.16

The coordinate systems defined by the Mahalanobis distance using the co-variance matrix for the dataset in Figure 5.15(c)[221] using three different origins: (a) (50, 50), (b) (63, 71), (c) (42, 35). The ellipses in each figure plot the 1, 3, and 5 unit distance contours.

Let's return to the original question depicted in Figure 5.15[221]: Are B and C likely to be from the same population from which the dataset has been sampled? Focusing on Figure 5.15(c)[221], for this dataset it appears reasonable to conclude that instance C is a member of the dataset but that B is probably not. To confirm this intuition we can calculate the Mahalanobis distance between A and B and A and C using Equation (5.16)[222] as

$$\begin{aligned} &Mahalanobis(A,B) \\ &= \sqrt{[50-30, 50-70] \times \begin{bmatrix} 0.059 & -0.521 \\ -0.521 & 0.0578 \end{bmatrix} \times \begin{bmatrix} 50-30 \\ 50-70 \end{bmatrix}} \\ &= 9.4049 \end{aligned}$$

$$\begin{aligned} &Mahalanobis(A,C) \\ &= \sqrt{[50-70, 50-70] \times \begin{bmatrix} 0.059 & -0.521 \\ -0.521 & 0.0578 \end{bmatrix} \times \begin{bmatrix} 50-70 \\ 50-70 \end{bmatrix}} \\ &= 2.2540 \end{aligned}$$

where the inverse covariance matrix used in the calculations is based on the covariance matrix[24] calculated directly from the dataset: $\begin{bmatrix} 82.39 & 74.26 \\ 74.26 & 84.22 \end{bmatrix}$

Figure 5.17[225] shows a contour plot of these Mahalanobis distances. In this figure, A indicates the central tendency of the dataset in Figure 5.15(c)[221], and the ellipses plot the Mahalanobis distance contours that the distances from A to the instances B and C lie on. These distance contours were calculated using the inverse covariance matrix for the dataset and point A as the origin. The result is that instance C is much closer to A than B and so should be considered a member of the same population as this dataset.

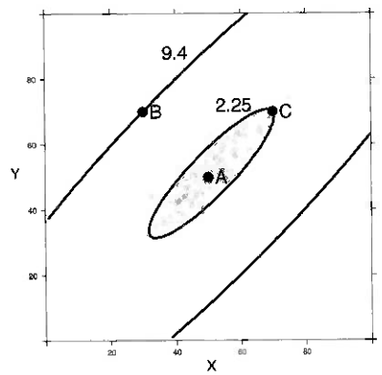

Figure 5.17

The effect of using a Mahalanobis versus Euclidean distance. A marks the central tendency of the dataset in Figure 5.15(c)[221]. The ellipses plot the Mahalanobis distance contours from A that B and C lie on. In Euclidean terms, B and C are equidistant from A; however, using the Mahalanobis distance, C is much closer to A than B.

To use Mahalanobis distance in a nearest neighbor model, we simply use the model in exactly the same way as described previously but substitute Mahalanobis distance for Euclidean distance.

5.4.5.4 Summary

In this section we have introduced a number of commonly used metrics and indexes for judging similarity between instances in a feature space. These are

24 Section 3.5.2[86] describes the calculation of covariance matrices. The inverse covariance matrix was calculated using the *solve* function from the **R** programming language.