Chapter 5
Principal Component Analysis and SVD

5.1 Decoder Based Data Summarization

5.1.1 Structure of a Summarization Problem with Decoder

Summarization as a concept covers many activities from data compression to labeling a dataset with a phrase like "Archeology finds indicate no King David Palace at the time of King David". Principal component analysis lies somewhere between these two to summarize the observed features in somewhat sharper structures. In contrast to a correlation problem, the features are not divided here into those belonging to input or output of the phenomenon under consideration. It is a different situation. One may think of this as that all features available are target features so that those to be constructed as a summary are in fact "hidden input features".

In this way, the structure of a summarization problem may be likened to that of a correlation problem if a rule is provided to predict all the original features from the summary. That is, the original data in the summarization problem act as target data in the correlation problem. That implies that there should be two rules involved in a summarization problem: one for building the summary, the other to provide a feedback from the summary to the observed data.

Unlike in the correlation problem, though, here the feedback rule must be prespecified so that the focus is on building a summarization rule rather than on using the summary for prediction; this is why we refer to the feedback rule as a "decoder" rather than a "predictor". In the machine learning literature, the issue of data summarization has not been given yet that attention it deserves; this is why the problem is usually considered somewhat simplistically as just deriving a summary from data without any feedback (see Fig. 5.1c).

A proper consideration of the structure of a summarization problem should rely on the existence of a decoder to provide the feedback from a summary back to the data and make the summarization process more or less similar to that of the correlation process (see Fig. 5.1a versus 5.1b). More exactly, a decoder is a device that translates the summary representation encoded in the chosen summarization rule back into the original data format. This allows us to utilize the same criterion of minimization of the difference between the original data and those output by

B. Mirkin, *Core Concepts in Data Analysis: Summarization, Correlation and Visualization*, Undergraduate Topics in Computer Science,
DOI 10.1007/978-0-85729-287-2_5, © Springer-Verlag London Limited 2011

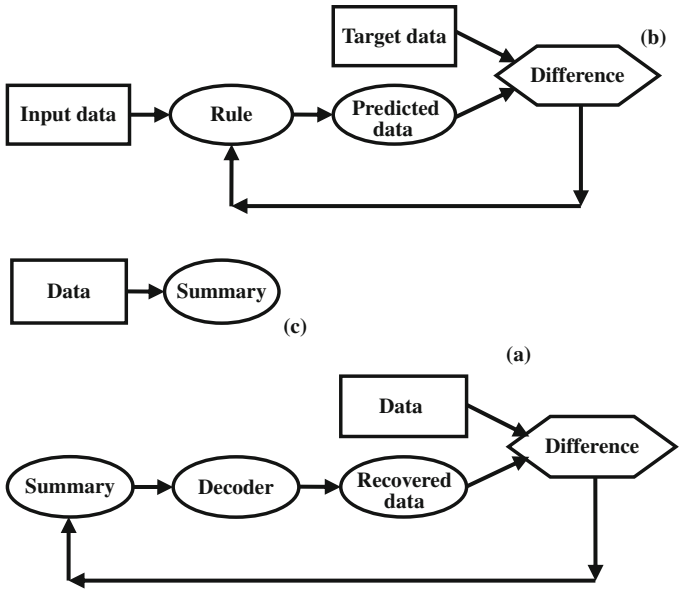

Fig. 5.1 A diagram for coder/decoder data summarization (**a**) versus learning input-target correlation (**b**) or summarization with no decoder (**c**). *Rectangles* are for observed data, *ovals* for computational constructions, *hexagons* for feedback comparisons

the decoder: the less the difference, the better. This text is largely concerned with methods that can be expressed in terms of decoder based criteria as presented on Fig. 5.1a.

P5.1.2 Data Recovery Criterion: Presentation

The data recovery approach in data summarization is based on the assumption that there is a regular structure in the phenomenon of which the observed dataset informs. This regular structure A is the summary to be found. When A is determined, this can feed back to the observed data Y in the format of the decoded data $F(A)$ that should coincide with Y up to residuals, that are due to possible flaws in any or all of the following three aspects:

(a) bias in entity sampling,
(b) selecting and measuring features,
(c) adequacy of the set of admissible A structures to the phenomenon in question.

Each of these three can drastically affect results. However, so far only the simplest of the aspects, (a) sampling bias, has been addressed scientifically, in statistics – as a random bias, due to the probabilistic nature of the data. The other two

are subjects of much effort in specific domains but not in the general computational data analysis framework as yet. Rather than focusing on accounting for the causes of errors, let us consider the underlying equation in which the errors are looked at as a whole:

$$Observed\_Data\ Y = Model\_Data\ F(A) + Residuals\ E \tag{5.1}$$

This equation brings in an inherent data recovery criterion for the assessment of the quality of the model A in recovering data Y – according to the level of residuals E: the smaller the residuals, the better the model. Since a data model typically involves unknown parameters, this naturally leads to the idea of fitting these parameters to the data in such a way that the residuals become as small as possible.

In many cases this principle can be rather easily implemented as the least squares principle because of an extension of the Pythagoras theorem relating the square lengths of the hypotenuse and two other sides in a right-angle triangle connecting "points" Y, $F(A)$ and 0 (see Fig. 5.2). The least squares criterion requires fitting the model A by minimizing the sum of the squared residuals. Geometrically, it often means an orthogonal projection of the data set considered as a multidimensional point onto the space of all possible models represented by the x axis on Fig. 5.2. In such a case the dataset (pentagram), its projection (rectangle) and the origin (0) form a right-angle triangle for which a multidimensional extension of the Pythagoras' theorem holds. The theorem states that the squared length of the hypotenuse is equal to the sum of squares of two other sides. The squared hypotenuse translates into the data scatter, that is, the sum of all the data entries squared, being decomposed in two parts, the part explained by the summary model A, that is, the contribution of the line between 0 and rectangle, and the part left unexplained by A. The latter part is the contribution of the residuals E expressed as the sum of squared residuals, which is exactly the least squares criterion. This very decomposition was employed in the problems of linear and non-linear regression in Sections 3.2 and 4.3, classification trees in Section 4.5, and it will be used again in further described methods: Principal component analysis and K-Means clustering, as well as additive clustering.

When the data can be considered as a random sample from a multivariate Gaussian distribution, the least squares principle can be derived, under some simplifying assumptions, from a major statistical principle, that of maximum likelihood. In the data analysis framework, the data do not necessarily come from a probabilistic

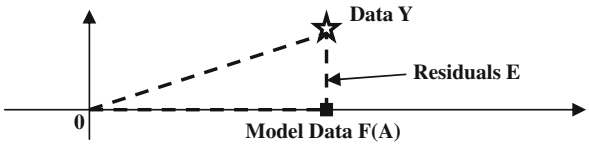

Fig. 5.2 Geometric relation between the observed data (*pentagram*), the fitted model data (*black rectangle*), and the residuals (*connecting line*)

population. Still, the least squares framework frequently provides for solutions that are both practically relevant and theoretically sound. The least squares will be the only criterion utilized in this text.

F5.1.3 Data Recovery Criterion: Formulation

A decoder based summarization problem can be stated as follows. Given N vectors forming a matrix $Y = \{(y_i)\}$ with rows $y_i = (y_{i1}, \ldots, y_{iV})$ of V features observed at entities $i = 1, 2, \ldots, N$ and a set of admissible summary structures $\boldsymbol{A}$ with decoder $D: \boldsymbol{A} \Rightarrow R^p$, build a summary

$$A = F(Y),\ A \in \boldsymbol{A}$$

such that the error, which is the difference between the decoded data $D(A)$ and observed data Y, is minimal over the class of admissible rules F. More explicitly, one assumes that

$$Y = D(A) + E \tag{5.2}$$

where E is matrix of residual values, or errors: the smaller the errors, the better the summarization A. According to the least-squares approach, the errors are minimized by minimizing the summary, or average, squared error:

$$E^2 = <Y - D(A),\ Y - D(A)> = <Y - D(F(Y)),\ Y - D(F(Y))> \tag{5.3}$$

with respect to all admissible summarization rules F.

Expression (5.3) can be further decomposed into

$$E^2 = <Y,\ Y> - 2<Y,\ D(A)> + <D(A),\ D(A)>.$$

In many data summarization methods, such as the Principal component analysis and K-Means clustering described later in Sections 5.2 and 6.2, the set of all possible decodings $D(F(Y))$ forms a linear subspace. In this case, the data matrices Y and $D(A)$, considered as multidimensional points, form a "right-angle triangle" around the origin 0, as presented on Fig. 5.2 above. In such a case $<Y,\ D(A)> = <D(A),\ D(A)>$ and the square error (5.3) becomes part of a multivariate analogue to the Pythagoras equation relating the squares of the "hypotenuse", Y, and the "sides", $D(A)$ and E:

$$<Y,\ Y> = <D(A),\ D(A)> + E^2 \tag{5.4}$$

or on the level of matrix entries,

$$\sum_{i\in I}\sum_{v\in V} {y_{iv}}^2 = \sum_{i\in I}\sum_{v\in V} {d_{iv}}^2 + \sum_{i\in I}\sum_{v\in V} {e_{iv}}^2 \qquad (5.4')$$

The data is an $N\times V$ matrix $Y = (y_{iv})$ that can be considered as either set of rows/entities y_i ($i = 1, \ldots, N$) or set of columns/features y_v ($v = 1, \ldots, V$) or both. The item on the left in (5.4′) is usually referred to as the data scatter and denoted by *T(Y)*,

$$T(Y) = \sum_{i\in I}\sum_{v\in V} {y_{iv}}^2 \qquad (5.5)$$

Why is this termed "scatter"? Indeed, *T(Y)* is the sum of Euclidean squared distances from 0 to all entities, thus a measure of scattering them around 0. In fact, *T(Y)* has a dual interpretation. On the one hand, *T(Y)* is the sum of row-based entity contributions, the squared distances $d(y_i, 0)(i = 1, \ldots, N)$. On the other hand, *T(Y)* is the sum of column-based feature contributions $t_v = \Sigma_{i\in I} {y_{iv}}^2$. In the case when the average c_v has been subtracted from all values of the column v, the summary contribution t_v is N times the variance, $t_v = N{\sigma_v}^2$.

Q.5.1. Prove that the summary contribution t_v is N times the variance, $t_v = N{\sigma_v}^2$ if feature v is centered. **A.** Indeed, $t_v = \Sigma_{i\in I} {y_{iv}}^2 = \Sigma_{i\in I}(y_{iv} - c_v)^2 = N[\Sigma_{i\in I}(y_{iv} - c_v)^2/N] = N{\sigma_v}^2$, where $c_v = 0$ is the mean of feature v.

5.1.4 Data Standardization

The least-squares solutions highly depend on the feature scales and may be highly affected by the scale changes, as decomposition (5.4′) clearly demonstrates. This is not exactly the case in correlation problems, at least in those with only one target feature, because the least squares there are, in fact, just that feature's errors, thus all expressed in the same scale. The data standardization problem, which is rather marginal at learning correlations, is of a great importance in data summarization. The problem of data standardization can be reformulated as the issue of defining the relative relevance, or importance, among the features. The greater the range of v, the greater the contribution t_v, thus the greater the relevance of v. There can be no universal answer to the issue of feature importance, because the answer always depends on the goal of summarization.

The assumption of equal importance of features currently underlies all the efforts and makes the entire edifice of data analysis somewhat crippled – but there is nothing new in this. As the history of science clearly demonstrates, any breakthrough in the sciences starts with a rather shaky base.

To balance contributions of features to the data scatter, one conventionally applies the operation of data standardization comprising two transformations, shift of the origin and rescaling.

We have already encountered standardization while studying multivariate classifiers, decision trees and neural networks. In neural networks as well as in

Support Vector Machine, the standardization involves the scale shift to the midrange and rescaling by normalizing the feature values by the half-range. These parameters are distribution independent.

Another, much more popular, choice is the feature's mean for the scale shift and normalizing by the standard deviation for rescaling. This standardization is a cornerstone in mathematical statistics and it works very well if observations come from a Gaussian distribution, because the distribution becomes parameter-free if standardized by subtracting the mean followed by dividing over the standard deviation. In statistics, this transformation is frequently referred to as z-scoring. In the context of data analysis, though, distributions are rarely Gaussian and rarely of any popular family at all; moreover, observations are not necessarily random or independent. In these circumstances, the choice of shifting and rescaling needs a rethink.

First of all, there is no need in linking the two operations together: shifting the origin has nothing to do with balancing feature weights. The goal of the shifting is to position the data against a backdrop of a "norm" which is put to the origin by the shift. In this way, the analysis involves the differences of the data and the norm. The experimental evidence accumulated in the ever growing body of data analysis research suggests that it is much easier to find meaningful structures if the "normal background" has been removed from data. According to the least squares criterion, it is the mean that approximates the overall "norm" the best. Since this criterion underlies all the methods considered in this text, the mean – sometimes referred to as grand mean, to point out its position over the entire entity set – will be the choice for the origin.

The normalization seems to be better if done by half-range or, equivalently, the range, indeed. On the first glance, there is no advantage in normalization by the range. Z-scoring seems a better choice, especially since z-scoring satisfies the intuitively appealing equality principle – all features contribute to the data scatter equally after they have been divided by their standard deviations.

This view is, however, overly simplistic. In fact, the feature's contribution to the data scatter is affected by two unrelated factors: (a) the feature scale range and (b) the distribution shape. While reducing the effect of the former, normalization should not suppress the effect of the latter because the distribution shape is an important indicator of the data structure. But the standard deviation involves both and thus mixes them up. Take a look, for example, at distributions of two features presented on Fig. 5.3. One of them has one mode only (a), whereas the other has two modes (b). Since the features have the same range, the standard deviation is greater for the distribution (b), which means that its relative contribution to the data scatter decreases under z-scoring standardization. This means that its clear cut discrimination between two parts of the distribution will be stretched in while the unimodal structure, which is hiding the two-part structure, will be stretched out. This is not exactly what we want of data standardization. Data standardization should help in revealing the data structure rather than concealing it. Thus, normalization by the range helps in bringing forward multimodal features by assigning them relatively larger weights proportional to their variances.

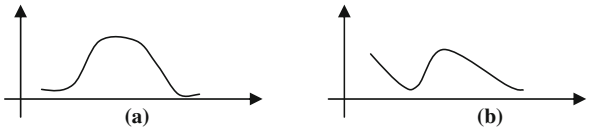

Fig. 5.3 One-modal distribution shape on (**a**) versus a two-modal distribution shape on (**b**): the standard deviation of the latter is greater, thus making it less significant under the z-scoring standardization

Therefore, in contrast to conventional wisdom, z-scoring standardization should be avoided unless there is a strong indication that the data come from a Gaussian distribution indeed. Any index related to the scale range can be used for normalization. In this text, the range is universally accepted. If, however, there is a strong indication that the range may be subject to outlier effects and, thus, unstable and random, more stable indexes could be used for normalization such as, for example, the distance between upper and lower 1% quantiles.

Worked example 5.1. Standardizing Iris dataset

Consider Iris dataset in Table 1.3. Its grand mean and midrange are presented in Table 5.1, along with its range and standard deviations.

These have been found by using the following MatLab commands:

```
≫ iris=load('Data\iris.dat');
≫ m=mean(iris); % grand mean
≫ ma=max(iris);% maximum
≫ ma=min(iris);% minimum
≫ mr=(ma+mi)/2; % midrange
≫ s=std(iris);% standard deviation
≫ ra=ma-mi;% range
```

In Table 5.1, midrange more or less follows the grand mean, but there are some discrepancies between the range and standard deviation. For example, ranges of w2 and w4 are the same, whereas standard deviations differ by almost 100%.

Table 5.1 Characteristics of Iris dataset

	Features			
Characteristics	w1	w2	w3	w4
Mean m	5.84	3.06	3.76	1.20
Midrange mr	6.10	3.20	3.95	1.30
Standard deviation s	0.83	0.44	1.77	0.76
Range ra	3.60	2.40	5.90	2.40

Let us take three different standardizations:

A – range related, y⇐(x-mr)/ra;
B – mean/range standardization, y⇐(x-m)/ra;
C – z-scoring, y⇐(x-m)/s;

and evaluate feature contributions to the data scatter after each of them (Table 5.2):

Feature contributions under A and B are similar, because both involve division by the range. According to these standardizations features w3 and w4 contribute most, because they are bimodal (see Q.2.24 and Fig. 2.19) and, thus play important role in further summarization methods, both Principal component analysis and cluster analysis. This concurs with the botanists' view that it is these sizes that determine the belongingness of an Iris specimen to a specific taxon (see references in Mirkin 2005). Moreover, at building a classification tree over Iris dataset, it was feature w4 that was involved in the splits according to three goodness criteria (see Fig. 4.11 in Section 4.5). In contrast, the first line assigns contributions according to feature values so that the lengths w1 and w3 get much larger contributions than the widths w2 and w4. And z-scoring (standardization C) makes all features contribute similarly, even in spite of the fact that two of them are bimodal.

The problem of standardization can be addressed by the user if they know the type of the distribution behind the observed data – the parameters of the distribution typically lead to a reasonable standardization. For example, the data should be standardized by z-scoring if the data is generated by independent one-dimensional Gaussian distributions. According to the formula for Gaussian density, a z-scored feature column would then fit the conventional $N(0,1)$ distribution making all features comparable to each other A similar strategy applies if the data is generated from a multivariate Gaussian density, just the data first needs to be transformed into mutually orthogonal singular vectors or, equivalently, principal components.

If no reasonable distribution can be assumed in the data, then there is no universal advice on standardization. However, with the summarization problems that we are going to address, the principal component analysis and clustering, some advice can be given in terms of the data scatter.

Table 5.2 Iris feature contributions to data scatter at different standardizations, per cent to the data scatter value

	Features			
Standardization	w1	w2	w3	w4
No standardization	54.76	15.00	27.07	3.17
A: midrange/range	20.61	12.70	31.48	35.66
B: mean/range	19.15	11.94	32.40	36.51
C: mean/std	25.00	25.00	25.00	25.00

The data transformation effected by the standardization can be expressed as

$$y_{iv} = (x_{iv} - a_v)/b_v \tag{5.6}$$

where $X = (x_{iv})$ stands for the original and $Y = (y_{iv})$ for standardized data, whereas $i \in I$ denotes an entity and $v \in V$ a feature. Parameter a_v stands for the shift of the origin and b_v for normalizing factor at each feature $v \in V$. In other words, one may say that the transformation (5.6), first, shifts the data origin into the point $a = (a_v)$, after which each feature v is rescaled separately by dividing its values over b_v.

The position of the space's origin, zero point $0 = (0, 0, \ldots, .0)$, at the standardized data Y is unique because any linear transformation of the data, that is, matrix product AY, for any A, can be expressed as a set of rotations of the coordinate axes around the origin, so that the origin itself is invariant. The principal component analysis can be expressed mathematically as a set of linear transformations of the data features as becomes clear in Section 5.2, which means that all the action in this method occurs around the origin. Metaphorically, the origin can be likened to the eye through which data points are looked at by the methods below. Therefore, for the purposes of data analysis, the origin should be put somewhere in the center of the data set, for which the gravity center, the point of all within-feature averages, is a best candidate. What is nice about it is that the feature contributions to the scatter of the center-of-gravity standardized data (5.5) above are equal to $t_v = \Sigma_{i \in I} {y_{iv}}^2 (v \in V)$, which means that they are proportional to the feature variances. Indeed, after the average c_v has been subtracted from all values of the column v, the summary contribution satisfies equation $t_v = N{\sigma_v}^2$ so that t_v is N times the variance. Even nicer properties of the gravity center as the origin have been derived in the framework of the simultaneous analysis of the categorical and quantitative data, see in Sections 4.5 and 6.3.

As to the normalizing coefficients, b_v, their choice is underlied by the idea of balancing the features weights. A most straightforward expression of the principle of feature equal importance is the use of the standard deviations as the normalizing coefficients, $b_v = \sigma_v$. This standardization makes the variances of all the variables $v \in V$ equal to 1 so that all the feature contributions become equal to $t_v = N$, which is seen at Table 5.5.

A very popular way to take into account the relative importance of different features is by using weight coefficients of features in computing the distances. This, in fact, is equivalent to and can be achieved with a proper standardization. Take, for instance, the weighted squared Euclidean distance between arbitrary entities $x = (x_1, x_2, x_M)$ and $y = (y_1,\ y_2,\ y_M)$ which is defined as

$$D_w(x, y) = w_1(x_1 - y_1)^2 + w_2(x_2 - y_2)^2 + \ldots + w_M(x_M - y_M)^2$$

where w_v are a pre-specified weights of features $v \in V$. Let us define (additional) normalizing parameters $b_v = 1/\sqrt{w_v}$ $(v \in E)$ to transform x and y into ${x'}_v = x_v/b_v$ and ${y'}_v = y_v/b_v$. It is rather obvious that

$$D_w(x, y) = d(x', y')$$

where d is the unweighted Euclidean squared distance.

That is, the following fact holds: for the Euclidean squared distance, the feature weighting is equivalent to an appropriate normalization as described above.

Q.5.2. Is it true that the sum of feature values standardized by subtracting the mean is zero? **A.** Yes, because the sum is proportional to the mean which is zero after centering.

Q.5.3. Consider a reversal of the operations in standardizing data: the scaling to be followed by the scale shift. Is it that different from the conventional standardization? **A.** Denote the scale shift and rescaling factor by a and b. Then the conventional standardization produces $y = (x-a)/b = x/b - a/b$ from x, whereas that suggested gives $z = x/b - a$. These differ at $a \neq 0$. To make them equal, the scale shift in the latter case must be a/b.

C5.1.5 Data Standardization: Computation

For the N×V data set X, its V-dimensional arrays of averages, standard deviations and ranges can be found in MatLab with respective operations

≫ av=mean(X);

≫ st=std(X,1); % here 1 indicates that divisor at sigmas is N rather than N−1

≫ ra=max(X)−min(X);

To properly standardize X, these V-dimensional rows must be converted to the format of $N \times V$ matrices, which can be done with the operation repmat(x,m,n) that expands a $p \times q$ array x into an $mp \times nq$ array by replicating it n times horizontally and m times vertically as follows:

≫avm=repmat(av, N,1);

≫stm=repmat(st, N,1);

≫ram=repmat(ra, N,1);

These are $N \times V$ arrays, with the same lines in each of them – feature averages in avm, standard deviations in stm, and ranges in ram.

To range-standardize the data, one can use a non-conventional MatLab operation of the entry-wise division of arrays:

≫Y=(X−avm)./ram;

Project 5.1. Standardization of mixed scale data and its effect

Pr5.1.A Data table and its quantification

Consider the Company dataset described in Section 1.2 and copied here in Table 5.3. Let us convert it into a quantitative format. The table contains two categorical variables, EC, with categories Yes/No, and Sector, with categories Utility, Industrial

Table 5.3 Data of eight companies producing goods A, B, or C related to the initial symbol of company's name

Company	Income	SharP	NSup	EC	Sector
Aversi	19.0	43.7	2	No	Utility
Antyos	29.4	36.0	3	No	Utility
Astonite	23.9	38.0	3	No	Industrial
Bayermart	18.4	27.9	2	Yes	Utility
Breaks	25.7	22.3	3	Yes	Industrial
Bumchist	12.1	16.9	2	Yes	Industrial
Civok	23.9	30.2	4	Yes	Retail
Cyberdam	27.2	58.0	5	Yes	Retail

and Retail. The former feature, EC, in fact represents just one category, "Using E-Commerce" and can be recoded as such by substituting 1 for Yes and 0 for No. The other feature, Sector, has three categories. To be able to treat them in a quantitative way, one should substitute each by a dummy variable.

Specifically, the three category features are:

(i) Utility: Is it Utility sector?
(ii) Industry: Is it Industrial sector?
(iii) Retail: Is it Retail sector?

Each of them admits Yes or No values, respectively substituted by 1 and 0. In this way, the original heterogeneous table will be transformed into a quantitative matrix in Table 5.4.

The first two features, Income and SharP, dominate the data in Table 5.4, especially with regard to the data scatter, that is, the sum of all the data entries squared, equal to 14833. As shown in Table 5.5, the two of them contribute more than 99% to the data scatter. To balance the contributions, features should be rescaled. Another

Table 5.4 Quantitatively recoded Company data table, along with summary characteristics

Company	Income	SharP	NSup	EC	Util	Indu	Reta
Aversi	19.0	43.7	2	0	1	0	0
Antyos	29.4	36.0	3	0	1	0	0
Astonite	23.9	38.0	3	0	0	1	0
Bayermart	18.4	27.9	2	1	1	0	0
Breaks	25.7	22.3	3	1	0	1	0
Bumchist	12.1	16.9	2	1	0	1	0
Civok	23.9	30.2	4	1	0	0	1
Cyberdam	27.2	58.0	5	1	0	0	1
Average	22.45	34.12	3.0	5/8	3/8	3/8	1/4
St deviation	5.26	12.10	1.0	0.48	0.48	0.48	0.43
Midrange	20.75	37.45	3.5	0.5	0.5	0.5	0.5
Range	17.3	41.1	3.0	1.0	1.0	1.0	1.0

Table 5.5 Within-column sums of the entries squared in Table 5.4

Contribution	Income	SharP	NSup	EC	Util	Ind	Retail	Data scatter
Absolute	4253	10487	80	5	3	3	2	14833
Per cent	28.67	70.70	0.54	0.03	0.02	0.02	0.01	100.00

important transformation of the data is the shift of the origin, because it affects the value of the data scatter and the decomposition of it in the explained and unexplained parts, as can be seen on Fig. 5.2.

Pr5.1.B Visualization of the data unnormalized

One can take a look at the effects of different standardization options. Table 5.6 contains data of Table 5.4 standardized by the scale shifting only: in each column, the within-column average has been subtracted from the column entries. Such standardization is referred to as centering.

The relative configuration of the 7-dimensional row-vectors in Table 5.6 can be captured by projecting them onto a plane, which is a two-dimensional, in an optimal way; this is provided by the two first singular values and corresponding singular vectors, as will be explained later in Section 5.2. This visualization is presented on Fig. 5.4 at which different product companies are shown with different shapes: squares (for A), triangles (for B) and circles (for C). As expected, this bears too much on features 2 and 1 that contribute 83.2% and 15.7%, respectively, here; a slight change from the original 70.7% and 23.7% according to Table 5.5. The features seem not related to products at all – the products are randomly intermingled with each other on the picture.

Table 5.6 The data in Table 5.4 standardized by the shift scale only, with the within-column averages subtracted. The values are rounded to the nearest two-digit decimal part, choosing the even number when two are the nearest. The bottom rows represent contributions of the columns to the data scatter as they are and per cent

Ave	−3.45	9.58	−1	−0.62	0.62	−0.38	−0.25
Ant	6.95	1.88	0	−0.62	0.62	−0.38	−0.25
Ast	1.45	3.88	0	−0.62	−0.38	0.62	−0.25
Bay	−4.05	−6.22	−1	0.38	0.62	−0.38	−0.25
Bre	3.25	−11.82	0	0.38	−0.38	0.62	−0.25
Bum	−10.4	−17.22	−1	0.38	−0.38	0.62	−0.25
Civ	1.45	−3.92	1	0.38	−0.38	−0.38	0.75
Cyb	4.75	23.88	2	0.38	−0.38	−0.38	0.75
Cnt	221.1	1170.9	8	1.9	1.9	1.9	1.5
Cnt %	15.7	83.2	0.6	0.1	0.1	0.1	0.1

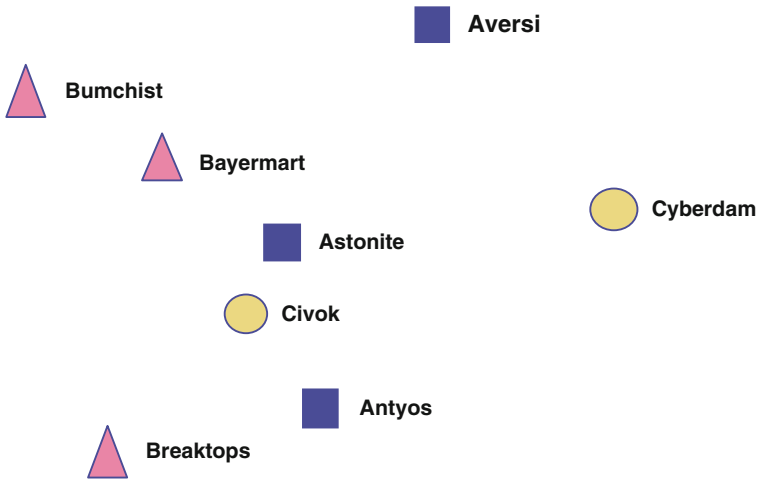

Fig. 5.4 Visualization of the entities in Companies data, centered only

Pr5.1.C Standardization by z-scoring

Consider now a more balanced standardization involving not only feature centering but also feature normalization over the standard deviations – z-scoring, as presented in Table 5.7.

An interesting property of this standardization is that contributions of all features to the data scatter are equal to each other, and moreover, to the number of entities, 8! This is not a coincidence but a property of z-scoring standardization.

The data in Table 5.7 projected on to the plane of two first singular vectors better reflect the products – on Fig. 5.5, C companies are clear-cut separated from the others; yet A and B are still intertwined.

Table 5.7 The data in Table 5.4 standardized by z-scoring. The bottom rows represent contributions of the columns to the data scatter as they are and per cent

Ave	−0.66	0.79	−1.00	−1.29	1.29	−0.77	−0.58
Ant	1.32	0.15	0	−1.29	1.29	−0.77	−0.58
Ast	0.28	0.32	0	−1.29	−0.77	1.29	−0.58
Bay	−0.77	−0.51	−1.00	0.77	1.29	−0.77	−0.58
Bre	0.62	−0.98	0	0.77	−0.77	1.29	−0.58
Bum	−1.97	−1.42	−1.00	0.77	−0.77	1.29	−0.58
Civ	0.28	−0.32	1.00	0.77	−0.77	−0.77	1.73
Cyb	0.90	1.97	2.00	0.77	−0.77	−0.77	1.73
Cnt	8	8	8	8	8	8	8
Cnt, %	14.3	14.3	14.3	14.3	14.3	14.3	14.3

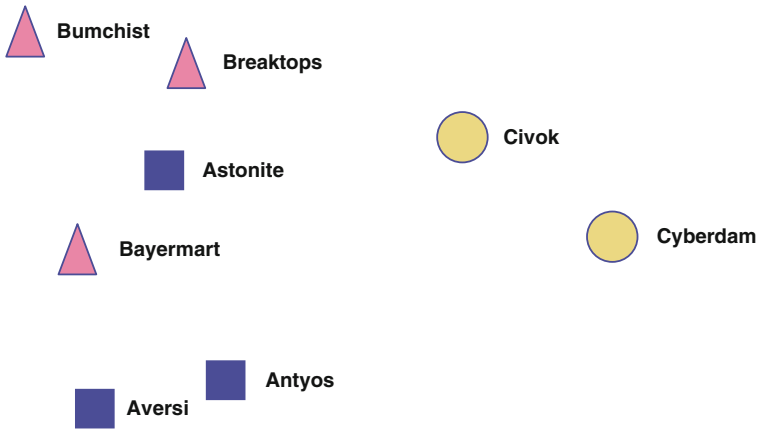

Fig. 5.5 Visualization of the entities in Companies data after z-scoring (Table 5.7)

Pr5.1.D Range normalization and rescaling of dummy features

We would like now to standardize the data in a mixed way by: (a) shifting the scales to the averages, as in z-scoring, but (b) dividing the results not by the feature's standard deviations but rather their ranges. However, when using both categorical and quantitative features, there is a catch here: each of the categories represented by dummy binary variables will have a greater variance than any of the quantitative counterparts after dividing by the ranges. Table 5.8 represents contributions of the range-standardized columns of Table 5.4.

Binary variables contribute much greater than the quantitative variables according to this standardization. The total contribution of the three categories of the original variable Sector looks especially odd – it is more than 55% of the data scatter, by far greater than should be assigned to one of the five original variables. This is partly because that variable Sector in the original table has been enveloped into three variables corresponding to its categories,, thus blowing out the contribution accordingly. To make up for this, the summary contribution of the three dummies should be decreased back three times. This can be done by making further normalization of them by dividing the normalized values by the square root of their number – 3 in our case. Why the square root is used, not just 3? Because contribution to the data scatter involves not the entries themselves but their squared values.

Table 5.8 Within-column sums of the entries squared in the data of Table 5.4 standardized by subtracting the averages and dividing the results by the ranges

Contribution	Income	SharP	NSup	EC	Util	Ind	Retail	Data scatter
Absolute	0.739	0.693	0.889	1.875	1.875	1.875	1.500	9.446
Per cent	7.82	7.34	9.41	19.85	19.85	19.85	15.88	100.00

The data table after additionally dividing entries in the three right-most columns over the square root of 3 is presented in Table 5.9. One can see that the contributions of the last three features did decrease threefold from those in Table 5.8, though the relative contributions changed less. Now the most contributing feature is the binary EC that divides the sample along the product based lines. This probably has contributed to the structure visualized on Fig. 5.6. The product defined clusters, much blurred on the previous figures, are clearly seen here, which shows that the original features indeed are informative of the products – just a proper standardization has to be carried out.

Q.5.4. How to do a z-scoring in MatLab? **A.** Take [n,v]=size(X) where X is the data matrix. Then define Y=(X-repmat(mean(X,n,1))./ repmat(std(X,n,1).

Q.5.5. What are the feature contributions after z-scoring? **A.** They all are equal to the same value, the data scatter related to V, the number of features.

Table 5.9 The data in Table 5.4 standardized by: (i) shifting to the within-column averages, (ii) dividing by the within-column ranges, and (iii) further dividing the category based three columns by 3. The values are rounded to the nearest two-digit decimal part

Av	−0.20	0.23	−0.33	−0.63	0.36	−0.22	−0.14
An	0.40	0.05	0	−0.63	0.36	−0.22	−0.14
As	0.08	0.09	0	−0.63	−0.22	0.36	−0.14
Ba	−0.23	−0.15	−0.33	0.38	0.36	−0.22	−0.14
Br	0.19	−0.29	0	0.38	−0.22	0.36	−0.14
Bu	−0.60	−0.42	−0.33	0.38	−0.22	0.36	−0.14
Ci	0.08	−0.10	0.33	0.38	−0.22	−0.22	−0.14
Cy	0.27	0.58	0.67	0.38	−0.22	−0.22	0.43
Cnt	0.74	0.69	0.89	1.88	0.62	0.62	0.50
Cnt %	12.42	11.66	14.95	31.54	10.51	10.51	8.41

Note: only two different values stand in each of the four columns on the right – why? Moreover, the entries within every column sum to 0 (see Q.5.2).

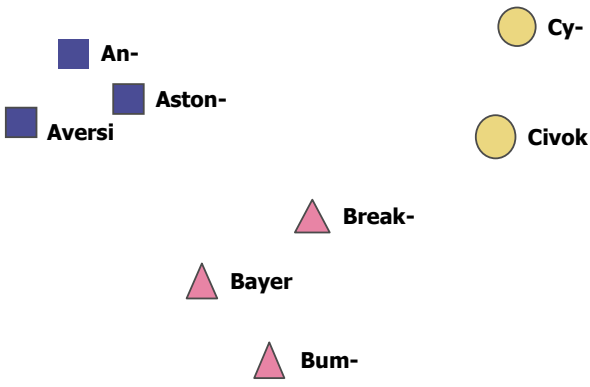

Fig. 5.6 Visualization of data in Table 5.9 standardized by the ranges with further subdividing the binary category features by the square root of 3

Q.5.6. How distances are affected if a different set of scale shifts is applied? **A.** Since the coordinates of two points are subtracted from each other in the distance, the scale shifts cancel each other and have no effect on the distances: the distances do not depend on the location of the origin of the space.

Q.5.7. How to do a distribution-free standardization by shifting to mid-range and normalizing by half-ranges? **A.** See Worked example 5.1, p. 179.

5.2 Principal Component Analysis: Model, Method, Usage

P5.2.1 SVD Based PCA and Its Usage: Presentation

The method of principal component analysis (PCA) has emerged in the research of "inherited talent" undertaken on the verge of nineteenth and twentieth centuries by F. Galton and K. Pearson, first of all to measure talent. For the time being, it has become one of the most popular methods for data summarization and visualization. The mathematical structure and properties of the method are based on the so-called singular value decomposition of data matrices (SVD); this is why in some publications terms PCA and SVD are used as synonymous. In the UK and USA, though, the term PCA frequently refers only to a technique for the analysis of inter-feature covariance or correlation matrix by extracting most contributing linear combinations of features, which utilizes no specific data models and is considered as purely heuristic. However, this method can be related to a genuine decoder based data summarization model that is underlied by the SVD equations – in the case when the data matrix has been centered beforehand. But the centering can hardly make a big difference to the method as such; this is why I refer to the method, even when the data matrix is not centered, as PCA.

There are many motivations for this method, of which we consider the following:

1. *Scoring a hidden factor*
2. *Data visualization*
3. *Feature space reduction*

P5.2.1.1 Scoring a Hidden Factor

Hidden Factor with a Multiplicative Decoder

Consider the following problem. Given student's marks at different subjects, can we derive from this their score at a hidden factor of talent that is supposedly reflected in the marks? Take a look, for example, at the first six students' marks over the three subjects in Table 5.10 extracted from Students data, Table 1.5.

To judge of the relative strength of a student, the average mark is used in practice. This ignores the relative work load that different subjects may impose on a student – can you see that CI mark is greater than SEn mark for each of the six students? – and

Table 5.10 Marks at three subjects for six students from Students data Table 1.5

#	SEn	OOP	CI	Average
1	41	66	90	65.7
2	57	56	60	57.7
3	61	72	79	70.7
4	69	73	72	71.3
5	63	52	88	67.7
6	62	83	80	75.0

in fact, is purely empirical and does not allow much theoretical speculation. Let us assume that there is a hidden factor, not measurable straightforwardly, the talent, that is manifested in the marks. Suppose that another factor manifested in the marks is subject load, and, most importantly, assume that these factors multiply to make a mark, so that a student's mark over a subject is the product of the subject's loading and the student's talent:

$$\text{Mark(Student, Subject)}=\text{Talent\_Score(Student)}*\text{Loading(Subject)}$$

One may point out two issues related to this model – one internal, the other external.

The external issue is that the mark, as observed, depends on many other factors differently affecting different students – the weather, a sleepless night or malady, level of interest in the subject, etc., which make the model as is overly simplistic and prone to errors. Well, a proponent would say, sure the model is simplistic – it takes on only most important factors. The others will cause errors indeed, but these can be tackled by minimizing them: the idea is that the hidden talent and loading factors can be found by minimizing the differences between the real marks and those derived from the model. The PCA method is based on the least-squares approach so that it is the sum of squared differences between the observed and computed marks that is minimized in PCA.

The internal issue is that the model as is admits no unique solution because it is the product of mark by loading that matters, not their individual values – if one multiples all the talent scores by a number, say Talent_Score(Student)*5, and simultaneously divides all the subject loadings by the same number, Loading(Subject)/5, the product will not change. How one is supposed to compute something which admits no definite representation? To make a solution unique, conventionally, a constant norm of one or both of the items is assumed so that one more item into the product is admitted – that expressing the product's magnitude. Then, as stated in the formulation part of this section, there is a unique solution indeed, with the magnitude expressed by the so-called maximum singular value of the data matrix with the score and load factors being its corresponding normed singular vectors.

Specifically, the maximum singular value of matrix in Table 5.10 is 291.4, and the corresponding normed singular vectors are $z = (0.40,\ 0.34,\ 0.42,\ 0.42,\ 0.41,\ 0.45)$, for the talent score, and $c = (0.50,\ 0.57,\ 0.66)$, for the loadings. That means that every mark in the matrix is product of three items. For example, to compute

the model SEn value for student 6, one takes 291.4*0.45*0.50 = 65.6, which is not that far from the observed mark of 62. Yet our model involves the product of two items only. To get back to that, we need to distribute the singular value between the vectors. There is only one way to do it complying with the singular value equations (5.12) – by multiplying each of the vectors by the same value, the square root of the singular value which is 17.1. Thus, the denormalized talent score and subject loading vectors will be $z' = (6.85,\ 5.83,\ 7.21,\ 7.20,\ 6.95,\ 7.64)$ and $c' = (8.45,\ 9.67,\ 11.25)$. According to the model, the score of student 3 over subject SEn is the product of the talent score, 7.21, and the loading, 8.45, which is 60.9, quite close to the observed mark 61. Similarly, product 5.83*9.67 = 56.4 is close to 56, student 2's mark over OOP. The differences can be greater though: product 5.83*8.45 is 49.3, which is rather far away from the observed mark 57 for student 2 over SEn.

In matrix terms, the model can be represented by the following equation

$$\begin{bmatrix} 6.85 \\ 5.83 \\ 7.21 \\ 7.20 \\ 6.95 \\ 7.64 \end{bmatrix} * \begin{bmatrix} 8.45 & 9.67 & 11.25 \end{bmatrix} = \begin{bmatrix} 57.88 & 66.29 & 77.07 \\ 49.22 & 56.37 & 65.53 \\ 60.88 & 69.72 & 81.05 \\ 60.83 & 69.67 & 81.00 \\ 58.69 & 67.22 & 78.15 \\ 64.53 & 73.91 & 85.92 \end{bmatrix} \tag{5.7}$$

whereas its relation to the observed data matrix, by equation

$$\begin{bmatrix} 41 & 66 & 90 \\ 57 & 56 & 90 \\ 61 & 72 & 79 \\ 69 & 73 & 72 \\ 63 & 52 & 88 \\ 62 & 83 & 80 \end{bmatrix} = \begin{bmatrix} 57.88 & 66.29 & 77.07 \\ 49.22 & 56.37 & 65.53 \\ 60.88 & 69.72 & 81.05 \\ 60.83 & 69.67 & 81.00 \\ 58.69 & 67.22 & 78.15 \\ 64.53 & 73.91 & 85.92 \end{bmatrix} + \begin{bmatrix} -16.88 & -0.29 & 12.93 \\ 7.78 & -0.37 & -5.53 \\ 0.12 & 2.28 & -2.05 \\ 8.17 & 3.33 & -9.00 \\ 4.31 & -15.22 & 9.85 \\ -2.53 & 9.09 & -5.92 \end{bmatrix} \tag{5.8}$$

where the left-hand item is the observed mark matrix; that in the middle, the model-computed estimations of the marks; and the right-hand item comprises the differences between the real and decoded marks.

Error of the Model

Among questions that arise with respect to the matrix equation such as that in (5.8) are the following:

(i) Why are the differences appearing at all?
(ii) How can the overall level of differences be assessed?
(iii) Can any better fitting estimates for the talent be found?

We address them in turn.

(i) Differences between real and model-derived marks

The differences emerge because the model imposes significant constraints on the model-derived estimates of marks. They are generated as products of components of just two vectors, the talent score and the subject loadings. This means that every row in the model-based matrix (5.7) is proportional to the vector of subject loadings, and every column, to the vector of talent scores. Therefore, the rows are mutually proportional as well as the columns. Real marks, generally speaking, do not satisfy such a property: mark rows or columns are typically not proportional to each other. More formally, this can be expressed in the following way: 6 talent scores and 3 subject loadings together can generate not more than $6+3=9$ independent estimates. (One more degree of freedom may go because the norms of these two vectors are the same.) The number of marks however, is the product of these, $6*3=18$. The greater the size of the data matrix, $M\times V$, the smaller the proportion of the independent values, $M+V$, that can be generated from the model.

In other words, matrix (5.7) is one-dimensional. It is well recognized in mathematics in the concept of matrix rank which corresponds to the "inner" dimension bore by a matrix – matrices that are products of two vectors are referred to as matrices of rank 1.

(ii) Assessment of the level of differences

A conventional measure of the level of error of the model is the ratio of the scatters of the model derived matrix and the observed data matrix in (5.8). The scatter of matrix A, $T(A)$, is the sum of the squares of all of A-entries or, which is the same, the sum of the diagonal entries in matrix A^*A^T, the $trace(A^*A^T)$.

Worked example 5.2. Explained proportion of data scatter in Equation (5.8)

Consider scatters of three matrices in (5.8) in Table 5.11. The residual data scatter is rather small and accounts for only $\varepsilon^2=1183.2/86092=0.0137$, that is, 1.37%, of the original data scatter. Its complement to unity, 98.63%, is the proportion of the data scatter explained by the multiplicative model. This also can be straightforwardly derived from the singular value, 291.4: its square shows the part of the data scatter explained by the model, $291.4^2/86092=0.9863$.

Table 5.11 Scatters of matrices in Equation (5.8)

	Data matrix	Scatter of Model matrix	Residual matrix
Absolute	86092	84908.80	1183.20
Proportion	100	98.63	1.37

Q.5.8. In spite of the fact that some errors in (5.8) are rather high, the overall squared error is quite small, just about one per cent of the data scatter. Why is that?
A. Because the data values are far away from 0 – see Q.5.20 explaining the effect mathematically.

(iii) The singular vector estimates are the best

The squared error is the criterion optimized by the estimates of talent scores and subject loadings. No other estimates can give a smaller value to the error for data matrix in Table 5.10 than ε^2=1.37%.

Formulaic Expression of the Hidden Factor Through the Data

The relations between singular vectors (see Eqs. (5.12) in Section 5.2.2.1 provide us with a conventional expression of the talent score as a weighted average of marks at different subjects. The weights are proportional to the subject loadings $c' = (8.45,\ 9.67,\ 11.25)$: weight vector w is the result of dividing of all entries in c' by the singular value, $w = c'/291.4 = (0.029,\ 0.033,\ 0.039)$. For example, the talent score for student 1 is the w weighted average of their marks, 0.029*41 + 0.033*66 + 0.039*90 = 6.88.

The model-derived averaging allows one also to score the talent of other students, those not belonging to the sample being analyzed. If marks of a student over the three subjects are (50,50,70), their talent score will be the w-weighted average: 0.029*50 + 0.033*50 + 0.039*70 = 5.83.

A final touch to the hidden factor scoring can be given by rescaling it in a way conforming to the application domain. Specifically, one may wish to express the talent scores in a 0-100 scale resembling that of the original mark scales. That means that the score vector z′ has to be transformed into $z'' = \alpha^* z' + \beta$, where α and β are the scaling factor and shift coefficients, that can be found from two natural conditions: (a) z'' is 0 when all the marks are 0 and (b) z'' is 100 when all the marks are 100. Condition (a) means that $\beta = 0$, and condition (b) calls for calculation of the talent score of a student with all top marks. Summing three 100 marks with weights from w leads to the value $zM = 0.029^*100 + 0.033^*100 + 0.039^*100 = 10.10$ which implies that the rescaling coefficient α must be $100/zM = 9.92$ or, equivalently, weights must be rescaled as $w' = 9.92^* w(0.29,\ 0.33,\ 0.38)$. Talent scores found with these weights are presented in the right column of Table 5.12 – hardly a great difference from the average scores, except that the talent scores are slightly higher, due to a greater weight assigned to the mark-earning CI subject.

In spite of the fact that the original model does not assume any averaging of the marks, the optimal scoring is a form of averaging indeed. However, one should note that it is the model that provides us with both the weights, which are the optimal subject loadings, and the error – these are entirely out of the picture at the empirical averaging.

This line of thinking can be applied to any other hidden performance measures such as quality of life in different cities using scorings over its different aspects

Table 5.12 Marks and talent scores for six students

#	SEn	OOP	CI	Average	Talent
1	41	66	90	65.7	68.0
2	57	56	60	57.7	57.8
3	61	72	79	70.7	71.5
4	69	73	72	71.3	71.5
5	63	52	88	67.7	69.0
6	62	83	80	75.0	75.8

(housing, transportation, catering, pollution, etc.) or performance of different management sections in a big company or government.

Sensitivity of the Hidden Factor to Data Standardization

One big issue related to the multiplicative hidden factor model is its instability with respect to data standardization that has been clearly seen at different data normalization options in Project 5.1. Here is another example.

Worked example 5.3. Principal components after feature centering

Consider now the data set in Table 5.10 analyzed above. Take the means of marks over different disciplines in this table, 58.8 for SEn, 67.0 for OOP, and 78.2 for CI, and subtract them from the marks, to shift the data to the mean point (see Table 5.13). This would not much change the average scores presented in Tables 5.10, 5.12 – just shifting them back by the average of the means, $(58.8 + 67.0 + 78.2)/3 = 68$.

Everything changes, though, in the multiplicative model, starting from the data scatter, which is now 1729.7 – a 50 times reduction from the case of uncentered data. The maximum singular value of the feature centered matrix in Table 5.13,

Table 5.13 Centered marks for six students and corresponding talent scores, first, as found as explained in A.1, and, second, that rescaled to produce extreme values 0 and 100 if all subject marks are 0 or 100, respectively

#	SEn	OOP	CI	Average	Talent score	Talent rescaled
1	−17.8	−1.0	11.8	−2.3	−3.71	13.69
2	−1.8	−11.0	−18.2	−10.3	1.48	17.60
3	2.2	5.0	0.8	2.7	0.49	16.85
4	10.2	6.0	−6.2	3.3	2.42	18.31
5	4.2	−15.0	9.8	−0.3	−1.94	15.02
6	3.2	16.0	1.8	7.0	1.25	17.42

$$\begin{bmatrix} -17.8 & -1.0 & 11.8 \\ -1.8 & -11.0 & -18.2 \\ 2.2 & 5.0 & 0.8 \\ 10.2 & 6.0 & -6.2 \\ 4.2 & -15.0 & 9.8 \\ 3.2 & 16.0 & 1.8 \end{bmatrix} = \begin{bmatrix} -11.51 & -7.24 & 13.83 \\ 4.60 & 2.90 & -5.53 \\ 1.52 & 0.96 & -1.82 \\ 7.52 & 4.73 & -9.04 \\ -6.02 & -3.79 & 7.23 \\ 3.88 & 2.44 & -4.67 \end{bmatrix} + \begin{bmatrix} -6.33 & 6.24 & -2.00 \\ -6.44 & -13.90 & -12.63 \\ 0.65 & 4.05 & 2.66 \\ 2.65 & 1.27 & 2.87 \\ 10.18 & -11.21 & 2.60 \\ -0.72 & 13.56 & 6.50 \end{bmatrix} \quad (5.9)$$

is 27.37 so that the multiplicative model now accounts for only $27.37^2/1729.7 = 0.433 = 43.3\%$ of the data scatter. This goes in line with the idea that much of the data structure can be seen from the "grand" mean (see Fig. 5.10 on p. 204 illustrating the point), however, this also greatly increases the error. In fact, the relative order of errors does not change that much, as can be seen in formula (5.10) decomposing the centered data (in the box on the left) in the model-based item, the first on the right, and the residual errors in the right-hand item. What changes is the denominator. The model-based estimates have been calculated in the same way as those in formula (5.7) – by multiplying every entry of the new talent score vector $z^* = (-3.71,\ 1.48,\ 0.49,\ 2.42,\ -1.94,\ 1.25)$ over every entry of the new subject loading vector $c^* = (3.10,\ 1.95,\ -3.73)$.

Worked example 5.4. Rescaling the talent score from Worked example 5.3

Let us determine rescaling parameters α and β that should be applied to z^*, or to the weights c^*, in Worked example 5.3 so that at 0 marks over all three subjects the talent score would be 0 and at all 100 marks the talent score would be 100.

As in the previous section, we first determine what scores correspond to these situations in the current setting. All-zero marks, after centering, become minus the average marks, −58.8 for SEn, −67.0 for OOP, and −78.2 for CI. Averaged according to the loadings c from Worked example 5.3, they produce $3.10^*(-58.8) + 1.95^*(-67) - 3.73^*(-78.2) = -21.24$. Analogously, all-hundred marks, after centering, become 41.2 for SEn, 33.0 for OOP, and 21.8 for CI to produce the score $3.10^*(41.2) + 1.95^*(33) - 3.73^*(21.8) = 110.8$. The difference between these, $110.8 - (-21.2) = 132.0$ divides 100 to produce the rescaling coefficient $a = 100/132 = 0.75$, after which shift value is determined from the all-0 score as $b = -0.75^*(-21.24) = 16.48$. Thus rescaled talent scores are in the last column of Table 5.13. These are much less related to the average scoring than it was the case at the original data. One can see some drastic changes such as, for instance, the formerly worst student 2 becoming second best, since their deficiency over CI has been converted to an advantage because of the negative loading at CI.

For a student with marks (50,50,70) that becomes (−8.8, −17.0, −8.2) after centering, the rescaled talent score comes from the adjusted weighting vector $w = a^*c = 0.75^*(3.10,\ 1.95,\ -3.73) = (2.34,\ 1.47,\ -2.81)$ as the weighted

average $2.34^*(-8.8) + 1.47^*(-17) - 2.81^*(-8.2) = -22.73$ plus the shift value $b = 16.48$ so that the result is, paradoxically, -6.25 – less than at all zeros! This is again a result of the negative loading at CI.

This example illustrates not only the idea of a great sensitivity of the multiplicative model, but, also, that there should be no mark centering when evaluating performances.

P5.2.1.2 Data Visualization

For the purposes of *visualization* of the data entities on a 2D plane, the data set is usually first centered to put it against the backdrop of the center – we mentioned already that more structure in the dataset can be seen when looking at it from the center of gravity, that is, the grand mean location. What has been disastrous for the purposes of scoring in Worked example 5.4 is beneficial for the purposes of structuring. Solutions to the multiple factor model, that is, the hidden factor scoring vectors which are singular vectors of the data matrix, in this case, are referred to as principal components (PCs). Two principal components corresponding to the maximal singular values are needed for a 2D representation.

What is warranted in this arrangement is that the PC plane approximates the data, as well as the between-feature covariances and between-entity similarities, in the best possible way. The coordinates provided by the singular vectors/ principal components are not unique, though, and can be changed by rotating the axes, but they do hold a unique property that each next component maximally contributes to the residual data scatter.

Worked example 5.5. Visualization of a fragment of Students dataset

Consider four features in the Students dataset – the Age and marks for SEn, OOP and CI subjects. Let us center it by subtracting the mean vector $a = (33.68, 58.39, 61.65, 55.35)$ from all the rows, and normalize the features by their ranges $r = (31, 56, 67, 69)$. The latter operation seems a necessity because the Age, expressed in years, and subject marks, per cent, are not exactly comparable. Characteristics of all the four singular vectors, that are principal components (PCs), of these data for feature loadings are presented in Table 5.14.

The summary contribution of the first two PCs to the data scatter is $42.34 + 29.77 = 72.11\%$, which is not that bad for educational data and warrants a close representation of the entities on the principal components plane. The two principal components are found by multiplication of each of the corresponding left singular vectors z_1 and z_2 by the square root of the corresponding singular value. Each entity $i = 1, 2, \ldots, N$ is represented on the PC plane by the pair of the first and second PC values (z_{1i}^*, z_{2i}^*) (see Fig. 5.7). The left part is just the data with no labels. On the right part, two occupational categories are visualized using triangles (IT)

Table 5.14 Components of the normed loading parts of principal components for the standardized part of Student data set; corresponding singular values, along with their squares expressed both per cent to their total, the data scatter, and in real

Singular value	Singular value squared	Contribution, per cent	Singular vector			
			Age	SEn	OOP	CI
3.33	11.12	42.34	−0.59	0.03	0.59	0.55
2.80	7.82	29.77	0.53	0.73	0.10	0.42
2.03	4.11	15.67	−0.51	0.68	−0.08	−0.51
1.79	3.21	12.22	−0.32	0.05	−0.80	0.51

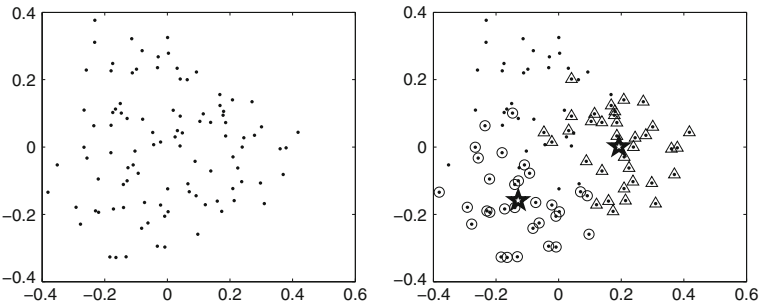

Fig. 5.7 Scatter plot of the student 4D data (Age and marks over SP, OO, CI) row points on the plane of two first principal components, after they have been centered and rescaled. *Pentagrams* represent the mean points of the occupation categories AN and IT

and circles (AN); remaining dots relate to category BA. In spite of the fact that the occupation has not been involved in building the PC space, its categories appear to occupy different parts of the plane, which will be explained later, in Worked example 5.6.

P5.2.1.3 Feature Space Reduction: Criteria of Contribution and Interpretability

The principal components provide for the best possible least-squares approximation of the data in a low dimension space. The quality of such a data compression is usually judged over (i) the proportion of the data scatter taken into account by the reduced dimension space and (ii) interpretability of the factors supplied by the PCs.

Contribution of the PCA model to the data scatter is reflected in the sum of squared singular values corresponding to the principal components in the reduced data. This sum should be related to the data scatter or, equivalently, to the total of all singular values squared, to see the impact. For example, the 2D representation of 4D student data on Fig. 5.7 contributes 72.11% to the data scatter, as found in

Worked example 5.5. In the example of marks for six students in Worked examples 5.2 and 5.4, the talent scoring factor contributed 98.6% to the data scatter at the original data and 43.3% after centering the data. Does that mean that marks should not be centered at all, to get a better approximation? Not necessarily. When all data entries are not negative, the large contribution of a principal component is an artifact of the very remoteness of the data set from the origin – the farther away you move the data from the origin, by adding a positive number to all the entries for example, the greater the contribution. This phenomenon follows a known property of positive fractions: if $0 < a/b < 1$, then adding a positive c to both the numerator and denominator may only increase the ratio; the greater the c, the greater the increase, so that $(a+c)/(b+c)$ converges to 1 when c tends to infinity (see Q.5.18). The analogy becomes clear if we consider b the data scatter and a, the principal component's contribution.

This example shows that the contribution, in spite of its firm mathematical footing, can be rather shaky an argument when data is not centered. One more criterion, of interpretability, gives a different perspective.

To interpret PCA results one should use the feature loadings according to the singular vectors related to features. These straightforwardly show how much a principal component is affected by a feature: the larger the value the greater the correlation. Features with relatively high positive or negative coefficients are used to interpret the component, as illustrated in the worked example below.

Worked example 5.6. Interpretation of principal components at the standardized Student data

Take a look at the first singular vector in Table 5.14 on p. 196 corresponding to the maximum singular value 3.33 at the standardized Students data. (Please note this 100×4 data differs from the 6×3 data of six students analyzed in the beginning.) One can see that the first component positively relates to marks over all subjects, perhaps except SEn at which the loading is almost zero, and negatively to the Age. That means that on average, the first factor is greater when a student gets better marks and is younger. Thus, the first component can be interpreted as the "Age-related Computer Science proficiency". The second component (the second line in Table 5.14) is positively related to all of the features, especially SEn marks, which can be interpreted as "Age defying inclination towards Software Engineering".Then the triangle and circle patterns on the right of Fig. 5.7 show that AN laborers are on the minimum side of the age-related CS proficiency, whereas IT occupations are high on that – all of which seem rather reasonable. Both are rather low on the second component, though, in contrast to students represented by dots, thus belonging to BA occupation category, that get the maximum values on it.

In the early days of the development of factor analysis, yet within the psychology community, researchers were trying to explore the possibility of achieving a more interpretable solution by rotating the axes of the PC space. The goal was to find

a simple structure of the loadings, in which most of the loading elements are zero with a few non-zero values that should be as close to either 1 or -1 as possible. This goal, however, is subject to too much of arbitrariness and remains an open issue. Keeping singular vectors as they are, not rotated, has the advantage that each of them contributes to the residual data scatter as much as possible. This relates to frequently occurring real world situations in which factors underlying the phenomenon of interest contribute to it differently. The PCA factors express such a structure formally: that one contributing the most is followed by the second best contributing, then by the third best contributing, etc.

Q.5.9. Prove that the condition of statistical independence for a contingency data table can be equivalently reformulated as the contingency table being of rank 1. **A.** Indeed an $N \times V$ matrix of rank 1 is a matrix whose elements are products of components of two vectors, N- and V-dimensional ones. In the case of a relative contingency table, $P = (p_{kl})$, the statistical independence condition, $p_{kl} = p_{k+}p_{+l}$ for all rows k and columns l, shows exactly that: all elements of matrix P are products of components of two vectors, (p_{k+}) and (p_{+l}), which proves the statement.

Q.5.10. What could be a purpose to aggregate the features in the Market towns' data? **A.** Since all the features relate to the extent of development of a town, the aggregate feature perhaps would express the extent of the town's development.

F5.2.2 Mathematical Model of PCA-SVD and Its Properties: Formulation

F5.2.2.1 A Multiplicative Decoder

Let us consider a data matrix X with entries x_{iv} and standardize it into $Y = (y_{iv})$ ($i = 1, 2, \ldots, N$; $v = 1, 2, \ldots, V$). The PCA model assumes hidden factor scores z_i^* and feature loadings c_v^* such that their product $z_i^* c_v^*$ is the decoder for y_{iv}, which can be explicated, by using additive residuals e_{iv}, as

$$y_{iv} = c_v^* z_i^* + e_{iv} \tag{5.10}$$

where the residuals are to be minimized using the least squares criterion

$$L^2 = \sum_{i \in I} \sum_{v \in V} e^2_{iv} = \sum_{i \in I} \sum_{v \in V} (y_{iv} - c_v^* z_i^*)^2 \tag{5.11}$$

The decoder in (5.10), as a mathematical model for deriving z_i^* and c_v^*, has a flaw from the technical point of view: its solution cannot be defined uniquely! Indeed, assume that we have got the talent score z_i^* for student i and the loading c_v^* at subject v, to produce $z_i^* c_v^*$ as the estimate for the student's mark at the subject. However, the same estimate will be produced if we halve the talent score vector and

simultaneously double the loading vector: $z_i{}^* c_v{}^* = (z_i{}^*/2)(2c_v{}^*)$. Any other real taken as the divisor/multiplier would, obviously, do the same.

A conventional remedy to this is following: specify the norms of vectors z^*, c^* to unities, and treat the multiplicative effect of the two as a real $\mu \geq 0$. Then put the product $\mu z_i c_v$ in (5.10) and (5.11) instead of $z_i{}^* c_v{}^*$ where z and c are normed versions of z* and c*, and μ is their multiplicative effect, the product of norms of z^* and c^*. The (Euclidean) norm $\|x\|$ of vector $x = (x_1, \ldots, x_N)$ is defined as its length, that is, the square root of $\|x\|^2 = x^T x = x_1{}^2 + x_2{}^2 + \ldots + x_N{}^2$. Thus a vector is referred to as normed if its length is 1, $\|x\| = 1$. After μ, z and c minimizing (5.11) are determined, return to the talent score vector z^* and loading vector c^* with formulas: $z^* = \mu^{1/2} z$, $c^* = \mu^{1/2} c$. It should be pointed out that a different norming condition such as say $|x_1| + |x_2| + \ldots + |x_N| = 1$ would lead to a different than the singular triplet solution – it seems no one ever explored such an opportunity.

The first-order optimality conditions to a triplet (μ, z, c) be the least-squares solution to (5.10) imply that $\mu = z^T Y c$ is maximum value satisfying equations

$$Y^T z = \mu c \quad \text{and} \quad Yc = \mu z \tag{5.12}$$

These equations for the optimal scores give the transformation of the data leading to the summaries z^* and c^*. The transformation, denoted by *F(Y)* in (5.1), appears to be linear, and combines optimal c and z so that each determines the other. It appears, this type of summarization is well known in linear algebra.

A triplet (μ, z, c) consisting of a non-negative μ and two vectors, c (size $M \times 1$) and z (size $N \times 1$) is referred to as to a singular triplet for Y if it satisfies (5.12); μ is referred to as a singular value and z, c the corresponding singular vectors. What can be proven immediately is the following:

Property 1 For any singular triplet (μ, z,c) satisfying (5.12) at $\mu \neq 0$ vectors z and c have the same norm.

Indeed, by multiplying the left-side equation in (5.12) by c^T, and the right-side equation by z^T, both from the left, one arrives at equations $c^T Y^T z = \mu c^T c$ and $z^T Yc = \mu z^T z$. Since $c^T Y^T z = (z^T Yc)^T$ and both are just real numbers, the equation $c^T c = z^T z$ holds because $\mu \neq 0$. Typically, the norms of c and z are taken to be unities. However, at the Principal components in (5.10), they are equal to the square root of the singular value μ, which proves the statement.

Any matrix Y can have only a finite number of singular values, and the number is equal to the rank of Y. Singular vectors z corresponding to different singular values are necessarily mutually orthogonal, as well as singular vectors c. When two or more singular values coincide, their singular vectors form a linear subspace and can be chosen to be orthogonal, which is the case in computational packages such as MatLab.

Therefore, $z^* = \mu^{1/2} z$ and $c^* = \mu^{1/2} c$ is a solution to the model (5.10) minimizing (5.11) defined by the maximum singular value of matrix Y and the corresponding normed singular vectors. Vectors z^* and c^* obviously also satisfy (5.12). This leads to other nice mathematical properties.

Property 2 The score vector z^* is a linear combination of columns of Y weighted by c^*'s components: c^*'s components are feature weights in the score z^*

Equations (5.12) allow mapping additional features or entities onto the other part of the hidden factor model. Consider, for example an additional N-dimemsional feature vector y standardized same way as Y. Its loading $c^*(y)$ is determined as $c^*(y) = <z^*, y>/\mu$ for the talent score z*. Similarly, an additional standardized V-dimensional entity point h has its hidden factor score defined according to the other part of (5.12), $z^*(h) = <c^*, h>/\mu$.

Property 3 Pythagorean decomposition of the data scatter T(Y) relating the least squares criterion (5.11) and the singular value holds as follows:

$$T(Y) = \mu^2 + L^2 \tag{5.13}$$

This implies that the squared singular value μ^2 expresses the proportion of the data scatter explained by the principal component z^*.

F5.2.2.2 Extension of the PC Decoder to the Case of Many Factors

It is well known by now that there is not just one talent behind the human efforts but a range of them. Assume a relatively small number K of different hidden factors z^*_k and corresponding feature loading vectors $c^*_k (k = 1, 2, \ldots, K;\ K < V)$, with students and subjects differently scored over them so that the observed marks, after standardization, are sums of those over the different talents:

$$y_{iv} = \sum_{k=1}^{K} c^*_{kv}\, z^*_{ik} + e_{iv}, \tag{5.14}$$

This is again a decoder that can be used for deriving a summary from the standardized marks matrix $Y = (y_{iv})$ so that the hidden score and loading vectors z^*_k and c^*_k are found by minimizing residuals, e_{iv}. To eliminate the mathematical ambiguity, we again assume that $z^*_k = \mu^{1/2} z_k$ and $c^*_k = \mu^{1/2} c_k$, where z_k and c_k are normed vectors.

Assume that the rank of Y is r and $K < r$. Assume that the singular values of Y are sorted so that $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_r$. It can be proven that the least-squares solution to (5.14) is provided by the maximal singular values μ_k and corresponding normed singular vectors z_k and c_k $(k = 1, 2, \ldots, K)$.

The underlying mathematical property is that any matrix Y can be decomposed over its singular values and vectors,

$$y_{iv} = \sum_{k=1}^{r} \mu_k\, c_{kv}\, z_{ik}, \tag{5.15}$$

which is referred to as the singular value decomposition (SVD). In matrix terms, SVD can be expressed as

$$Y = \sum_{k=1}^{r} \mu_k z_k c^T{}_k = ZMC^T \tag{5.15'}$$

where the right-hand item Z is $N \times r$ matrix with columns z_k and C is $M \times r$ matrix with columns c_k and M is an $r \times r$ diagonal matrix with entries μ_k on the diagonal and all other entries zero.

Equation (5.15′) implies, because the singular vectors are mutually orthogonal, that the scatter of matrix Y is decomposed into the sum of the squared singular values:

$$T(Y) = \mu_1{}^2 + \mu_2{}^2 + \ldots + \mu_r{}^2 \tag{5.16}$$

This implies that the least-squares fitting of the PCA model in Equation (5.14) decomposes the data scatter into the sum of contributions of individual singular vectors and the least-squares criterion $L^2 = \Sigma_{i,\, v} e_{iv}{}^2$:

$$T(Y) = \mu_1{}^2 + \mu_2{}^2 + \ldots + \mu_K{}^2 + L^2 \tag{5.17}$$

This provides for the evaluation of the relative contribution of the model (5.14) to the data scatter as $(\mu_1{}^2 + \mu_2{}^2 + \ldots + \mu_K{}^2)/T(Y)$.

In particular, this part of decomposition (5.15) is used at 2D visualization:

$$y_{iv}{}^* \approx z_{ii}{}^* c_{1v}{}^* + z_{i2}{}^* c_{2v}{}^*$$

where the elements on the right come from the two first principal components. This equation holds not 100% but $100^*(\mu_1{}^2 + \mu_2{}^2)/T(Y)$ percent. Every entity $i \in I$ is represented on a 2D Cartesian plane by pair $(z_{ii}{}^*,\ z_{i2}{}^*)$. Moreover, because of the symmetry, every feature v can be represented, on the same plane by pair $(c_{1v}{}^*,\ c_{2v}{}^*)$. Such a simultaneous representation of both entities and features is referred to as a joint display or a biplot. As a matter of fact, features are presented on a biplot by not just the corresponding points, but by lines joining them to 0. This reflects the fact, that projections of points, representing the entities (entity markers), to these lines are meaningful. For a variable v, the length and direction of the projection of an entity marker to the corresponding line reflects the value of v on the entity.

F5.2.2.3 Conventional Formulation of PCA Using Covariance Matrix

In the English-written literature, PCA is conventionally introduced in a different way: not via the decoder based model (5.10) or (5.14), but rather as a heuristic technique to build up most contributing linear combinations of features with the help of the data covariance matrix (see, for example, Kendall and Stewart 1973, Hair et al. 2010).

The covariance matrix is defined as $V \times V$ matrix $C = Y^T Y/N$, where Y is a centered version of the data matrix X, so that all its columns are centered. The (v',v'')-entry in the covariance matrix is the covariance coefficient between features v' and v''; and the diagonal elements are variances of the corresponding features. The covariance matrix is referred to as the correlation matrix if Y has been z-score standardized, that is, after shifting each column to its mean, it was further normalized by dividing by its standard deviation. In this case, elements of C are correlation coefficients between corresponding variables. (Note how a bivariate concept is carried through to multivariate data by using matrix multiplication.)

The conventional PCA problem formulation goes like this. Given a centered $N \times V$ data matrix Y, find a normed V-dimensional vector $c = (c_v)$ such that the sum of Y columns weighted by c, $f = Yc$, has the largest variance possible. This vector is the principal component, termed so because it shows the direction of the maximum variance in data. Vector f is centered for any c, since Y is centered. Therefore, its variance is $s^2 = <f,f>/N = f^T f/N$. The last equation comes under the convention that a V-dimensional vector is a $V \times 1$ matrix, that is, a column. By substituting Yc for f, this leads to equation $s^2 = c^T Y^T Yc/N$. Maximizing this with respect to all vectors c that are normed, that is, satisfy condition $c^T c = 1$, is equivalent to unconditionally maximizing the ratio

$$q(c) = \frac{c^T Y^T Yc}{c^T c} \tag{5.18}$$

over all V-dimensional vectors c. Expression (5.18) is well known in linear algebra as the Rayleigh quotient for matrix $NC = Y^T Y$ which is proportional to the covariance matrix of course. The maximum of Rayleigh quotient is reached at c being an eigen-vector, also termed latent vector, for matrix C, corresponding to its maximum eigenvalue (latent value) $q(c)$ (5.18).

Vector a is referred to as an eigenvector for a square matrix B if $Ba = \lambda a$ for some, possibly complex, number λ which is referred to as the eigenvalue corresponding to a. In the case of a covariance matrix all eigenvalues are not only real but non-negative as well. The number of eigenvalues of B is equal to the rank of B, and eigenvectors corresponding to different eigenvalues are orthogonal to each other.

Therefore, the first principal component, in the conventional definition, is vector $f = Yc$ defined by the eigenvector of the covariance matrix A corresponding to its maximum eigenvalue. The second principal component is conventionally defined as another linear combination of columns of Y, which maximizes its variance under the condition that it is orthogonal to the first principal component. It is defined, of course, by the second eigenvalue and corresponding eigenvector. Other principal components are defined similarly, in a recursive manner, under the condition that they are orthogonal to all preceding principal components; this implies that they correspond to other eigenvalues, in the descending order, and the corresponding eigenvectors.

This construction seems rather remote from how the principal components are introduced above. However, it is not difficult to prove that the two definitions are computationally equivalent. Indeed, take equation $Yc = \mu z$ from (5.12), express z from this as $z = Yc/\mu$, and substitute this z into the other Equation (5.12): $Y^T z = \mu c$ so that $Y^T Yc/\mu = \mu c$. This implies that μ^2 and c, defined by the multiplicative decoder model, satisfy equation

$$Y^T Yc = \mu^2 c, \tag{5.19}$$

that is, c is an eigenvector of square matrix $Y^T Y$, corresponding to its maximum eigenvalue $\lambda = \mu^2$. Matrix $Y^T Y$, in the case when Y is centred, is the covariance matrix C up to the constant factor $1/N$. Therefore, c in (5.19) is an egenvector of C corresponding to its maximum eigenvalue. This proves that the two definitions are equivalent when the data matrix Y is centered. The given proof also establishes a simple relation between the eigen values λ of C and singular values μ of Y: $\lambda = \mu^2$.

In spite of the computational equivalence, there are some conceptual differences between the two definitions. In contrast to the definition in 5.2.2.1 based on the multiplicative decoder, the conventional definition is purely heuristic, assuming no underlying model whatsoever. It makes sense only for centered data because of its reliance on the concept of covariance. Moreover, the fact that the principal components are linear combinations of features is postulated in the conventional definition, whereas this is a derived property of the optimal solution to the multiplicative decoder model which involves no assumptions on a linear or nonlinear relation between features and hidden factors.

Q.5.11. Can you write equations defining μ^2 and z as an eigenvalue and corresponding eigenvector of matrix YY^T. Does this square matrix have any meaning of its own? **A.** By multiplying the left-side equation in (5.12) by Y on the left, we obtain $YY^T z = \mu Yc = \mu^2 z$, the latter equation following from the right-hand equation in (5.12). Elements of matrix YY^T are inner products of rows of matrix Y to express similarities between corresponding entities.

F5.2.2.4 Geometric Interpretation of Principal Components

Take all talent score points $z = (z_1, \ldots, z_N)$ that are normed, that is, satisfy equation $<z, z> = 1$ or $z^T z = 1$ or $z_1^2 + \ldots + z_N{}^2 = 1$: they form a sphere of radius 1 in the N-dimensional "entity" space (Fig. 5.8a). The image of these points in the feature space, $Y^T z$, forms a skewed sphere, an ellipsoid in the feature space, consisting of points μc where c is normed. The longest axis of this ellipsoid corresponds to the maximum μ, that is the first singular value of Y. [Indeed, the first singular value μ_1 and corresponding normed singular vectors c_1, z_1 satisfy equation $Yc_1 = \mu z_1$ and, thus, its transpose, ${c_1}^T Y^T = \mu_1 z^T$. Multiplying the latter by the former, one gets equation ${c_1}^T Y^T Y c_1 = {\mu_1}^2$, because $z^T z = 1$.]

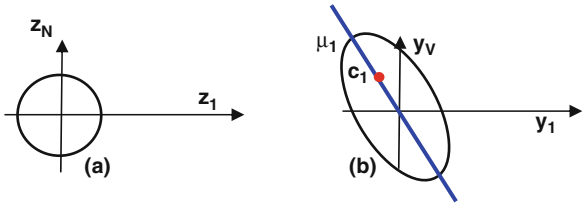

Fig. 5.8 Sphere $z^T z = 1$ in the entity space (**a**) and its image, ellipsoid $c^T Y^T Y c = {\mu_1}^2$, in the feature space. The first component, c_1, corresponds to the maximal axis of the ellipsoid with its length equal to $2\mu_1$

What the longest axis has to do with the data? This is exactly the direction which is looked for in the conventional definition of PCA. The direction of the longest axis of the data ellipsoid makes minimum of the summary distances (Euclidean squared) from data points to their projections on the line (see Fig. 5.9), because of the least-squares optimality of the decoder in (5.10) so that this axis is the best possible 1D representation of the data.

This property extends to all subspaces generated in the order of extraction of principal components: the first two PCs make a plane that is the best two-dimensional approximation of the dataset; the first three make a 3D space best approximating the dataset, etc.

One more illustration concerns the difference between SVD for the raw data and that after centering; see Fig. 5.10, which is basically extension of the previous Fig. 5.9 to the cases. One can see that at the raw data, Fig. 5.10a, the longest axis follows the direction between the origin and the points cloud, whereas after centering it follows the structure of the dataset, Fig. 5.10b.

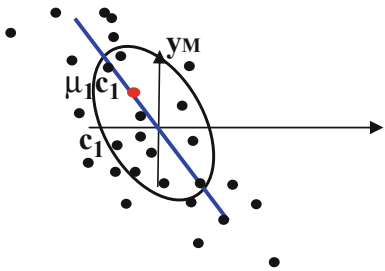

Fig. 5.9 The direction of the longest axis of the data ellipsoid makes minimum the summary distances (Euclidean squared) from data points to their projections onto the line

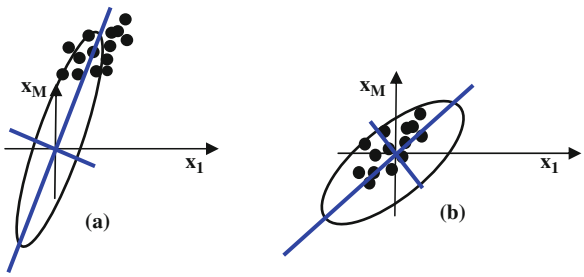

Fig. 5.10 (**a**) PCA at data not centered, (**b**) PCA at the data after centering

C5.2.3 Computing Principal Components

The SVD decomposition is found with MatLab's svd.m function

```
[Z, M, C]=svd(Y);
```

where Y is $N \times V$ data matrix after standardization whose rank is r. Typically, if all data entries come from observation, the rank $r = \min(N, V)$.

The output consists of three matrices:

Z – $N \times N$ matrix of which only r columns are meaningful, r factor score normed columns;
C – $V \times V$ matrix of corresponding feature loading columns (normed) of which only r are meaningful;
M – $N \times V$ matrix with $r \times r$ diagonal submatrix of corresponding singular values sorted in the descending order on the top left, the part below and to the right is all zeros.

Worked example 5.7. SVD for Six Students dataset

For the centered 6×3 matrix in Table 5.13 the SVD matrices are as follows:

```
      −0.7086    0.1783    0.4534    0.4659   0.1888    0.0888
       0.2836   −0.6934    0.4706    0.0552   0.2786    0.3697
z=     0.0935    0.1841   −0.0486   −0.1870   0.9048   −0.3184
       0.4629    0.0931   −0.1916    0.8513   0.0604   −0.1092
      −0.3705   −0.3374   −0.7293    0.1083   0.2279    0.3916
       0.2391    0.5753    0.0455   −0.0922   0.1116    0.7673

      −0.6566    0.0846    0.7495             27.37       0        0
c=    −0.2514    0.9123   −0.3232      M=        0       26.13     0
       0.7111    0.4006    0.5778               0         0      17.26
```

Worked example 5.8. Standardized Student data visualized

To produce the scatter-plot of Fig. 5.7 with 100×4 Student data matrix Y, the following MatLab commands can be used:

```
≫ [z,m,c]=svd(y);
≫ z1*=z(:,1)*sqrt(m(1,1));
≫ z2*=z(:,2)*sqrt(m(2,2));
% z1*, z2* are first PCs defined on p. 201
≫ subplot(1,2,1); plot(z1*,z2*,'k.'); %Fig. 5.7, picture on the left
≫ subplot(1,2,2);
≫ plot(z1*, z2*,'k.', z1*(1:35),z2*(1:35),'k^',. . .z1*(70:100),z2*(70:100),'ko',ad1,ad2,'kp');
```

In the last command, there are several items to be shown on the same plot:

(i) z1*, z2*, 'k.' – these are black dot markers for all 100 entities exactly as on the plot on the left;
(ii) z1*(1:35), z2*(1:35),'k^' – these are triangles to represent entities 1 to 35 – those in category IT;
(iii) z1*(70:100), z2*(70:100),'ko' – these are circles to represent entities 70 to 100 – those in category AN;
(iv) ad1, ad2, 'kp' – ad1 is a 2×1 vector of the averages of z1* over entities 1 to 35 (category IT) and 70 to 100 (category AN), and ad2 a similar vector of within-category averages of z2*. These are represented by pentagrams.

Q.5.12. Assume that a category covers subset S of entities and y(S) represents the feature mean vector over S. Prove that the supplementary introduction of y(S) onto the plain of singular vectors z via equation $z^* = \sqrt{\mu}z = Y^*y(S)/\sqrt{\mu}$ from (5.12) onto the 2D PCA display is equivalent to representing the category by the averages of the 2D points z_{1i}^* and z_{2i}^* over $i \in S$. **A.** Indeed, the operation of averaging involves but addition and dividing by a number, which are not affected by a linear operation of matrix multiplication.

Worked example 5.9. Evaluation of the quality of visualization of the standardized Student data

To evaluate how well the data are approximated by the PC plane such as that on Fig. 5.7, according to Equation (5.17), one needs to assess the summary contribution of the first two singular values squared in the total data scatter. To get the squares one can multiply the diagonal matrix of singular values by itself and then see the proportion of the first two values in the total:

≫ mu=m(1:4,:); %no need in 4×100 matrix output, have a square size 4×4
≫ la=diag(mu*mu);% make squares and put them as a vector
≫ lar=la*100/sum(sum(la)) % vector of the relative contributions of each PC
≫ lar(1)+lar(2) % contribution of the 2 first components

This prints to the screen:

```
lar =
   42.3426
   29.7719
   15.6664
   12.2191

ans =
   72.1145
```

The latter is the sum of two first elements of the former – the proportion of the data scatter taken into account by the 2D visualization on Fig. 5.7.

5.3 Application: Latent Semantic Analysis

The number of papers applying PCA to various problems – image analysis, information retrieval, gene expression interpretation, complex data storage, etc. – makes many hundreds published annually. Some of the applications are well established techniques of their own. We present two such techniques: Latent semantic indexing (analysis) in this section and Correspondence analysis, in the next section.

P5.3.1 Latent Semantic Analysis: Presentation

Latent semantic analysis is an application of PCA to document analysis – information retrieval, first of all, using document-to-keyword data (see Deerwester et al. 1990).

Information retrieval is an application that no computational data analysis may skip: given a set of records or documents stored, find out those related to a specific query expressed by a set of keywords. Initially, at the dawn of computer era, when all the documents were stored in the same database, the problem was treated in a hard manner – only documents containing the query words were to be given to the user. Currently, this is a much softer problem that is being constantly and efficiently solved by various search engines such as Google, for millions of World Wide Web users (see Manning et al. 2008).

In its generic format, the problem can be illustrated with data in Table 5.15, already utilized as Table 4.1 in Section 4.2. It refers to a number of newspaper

Table 5.15 Database of 12 newspaper articles along with 10 keywords and the conventional coding of term frequencies. The articles are labeled F for Feminism, E for Entertainment and H for Household. One line holds document frequencies of terms (df) and the other, inverse document frequency weights (idf)

	Keyword									
Article	Drink	Equal	Fuel	Play	Popular	Price	Relief	Talent	Tax	Woman
F1	1	2	0	1	2	0	0	0	0	2
F2	0	0	0	1	0	1	0	2	0	2
F3	0	2	0	0	0	0	0	1	0	2
F4	2	1	0	0	0	2	0	2	0	1
E1	2	0	1	2	2	0	0	1	0	0
E2	0	1	0	3	2	1	2	0	0	0
E3	1	0	2	0	1	1	0	3	1	1
E4	0	1	0	1	1	0	1	1	0	0
H1	0	0	2	0	1	2	0	0	2	0
H2	1	0	2	2	0	2	2	0	0	0
H3	0	0	1	1	2	1	1	0	2	0
H4	0	0	1	0	0	2	2	0	2	0
df	5	5	6	7	7	8	5	6	4	5
idf	0.88	0.88	0.69	0.54	0.54	0.41	0.88	0.69	1.10	0.88

articles related to subjects such as entertainment, feminism and households, conveniently coded with letters E, F and H, respectively. Columns correspond to keywords, or terms, listed in the first line of the table, and entries refer to term frequency in the articles, according to a conventional coding scheme:

0 – no occurrence,
1 – occurs once,
2 – occurs twice or more.

The user may wish to retrieve all the articles on the subject of households, but they are subjected to inquire by using the listed keywords only. For example, query "fuel" will retrieve all four of the household related articles, and, in fact more than that – E1 and E3 will show up too; query "tax" will get four items, three – H1, H3, and H4 – on the subjects of household and one – E3 – on the subject of entertainment. No combination of these two can improve the result.

This is very much a class description problem; just the decision rules, the queries, must be combinations of keywords. The error of such a query is characterized by two characteristics, precision and recall (see Section 4.2.3). For example, "fuel" query's precision is $4/6 = 2/3$ since only four of six are relevant and recall is 1 because all of the relevant documents have been returned. Similarly, for "tax" query both precision and recall are $3/4$.

The rigidity of the query format does not fit well into the polysemy of natural language – such words as "fuel" or "play" have more than one meaning – thus leading to impossibility of exact information retrieval in many cases.

The method of latent semantic indexing (LSI) utilizes the SVD decomposition of the document-to-term data to soften and thus improve the query system by embedding both documents and terms into a subspace of singular vectors of the data matrix.

Before proceeding to SVD, the data table sometimes is pre-processed, typically, with what is referred to Term-Frequency-Inverse-Document-Frequency (tf-idf) normalization. This procedure gives a different weight to any keyword according to the number of documents it occurs at (document frequency df). The intuition is that the greater the document frequency, the more common and thus less informative is the word. The idf weighting assigns each keyword with a weight inversely proportional to the logarithm of its document frequency. For Table 5.15, df and idf weights are in its last lines.

The term frequency, tf, value is that of the corresponding data matrix entry referring to the frequency of the occurrence of the column word in the row document related to the document size.

After the SVD of the data matrix is obtained, the documents are considered points of the subspace of a few first singular vectors. The dimension of the space is not very important here, though it still should be much smaller than the original dimension. Good practical results have been reported at the dimension of about 100-200 when the number of documents in tens and hundred thousands and the number of keywords in thousands. A query is also represented as a point in the same space. The

principal components, in general, are considered as "orthogonal" concepts underlying the meaning of terms. This however, should not be taken too literally as the singular vectors can be quite difficult to interpret. Also, the representation of documents and queries as points in a Euclidean space is referred to sometimes as the vector space model in information retrieval.

The Euclidean space format allows to measure similarity between items using the inner product or even what is called cosine - the inner product between rows that have been pre-normalized. Then a query would return the set of documents whose similarity to the query point is greater than a threshold. This tool may provide for a better resolution in the problem of information retrieval, because it well separates different meanings of synonyms.

Worked example 5.10. Latent semantic space for article-to-term data

Let us illustrate the LSI at the data in Table 5.15. To apply tf-idf normalization, we assume that all the documents have the same length so that the absolute term frequencies in Table 5.15 can be used as tf estimates. Then the tf-idf coding of any entry is equal to the entry value multiplied by the corresponding idf value (the last line in Table 5.15).

Consider the combination fuel-price-relief-tax as a query Q that should relate to Household category. Table 5.16 contains data that are necessary for computing coordinates of a query Q in the concept space. The query is represented by 1/0 vector in the last line of Table 5.16. The first coordinate of Q image on the map is computed by summing all the corresponding components of the first left singular vector and dividing the result by the square root of the first singular value: $u1 = (-0.34-0.42-0.29-0.24)/8.6^{1/2} = -0.44$. The second coordinate is computed similarly from the second singular vector and value: $u2 = (-0.25-0.22-0.35-0.33)5.3^{1/2} = -0.48$. These correspond to the pentagram on the left part of Fig. 5.11. One can think that query Q corresponds to an additional row in the data table which is equal to the last line in Table 5.16, so that the visualization algorithm of PCA applies to that line.

Figure 5.11 represents the documents in the space of two first principal components; left part corresponds to the original term frequency codes and the right part to the data tf-idf normalized.

Table 5.16 Two first singular vectors of term frequency data in Table 5.15

	Ei.value	Contrib.,%	Left singular vectors normed									
1st comp.	8.6	46.9	−0.25	−0.19	−0.34	−0.40	−0.39	−0.42	−0.29	−0.32	−0.24	−0.22
2nd comp.	5.3	17.8	0.22	0.34	−0.25	−0.07	0.01	−0.22	0.35	0.48	−0.33	0.51
Query Q			0	0	1	0	0	1	1	0	1	0

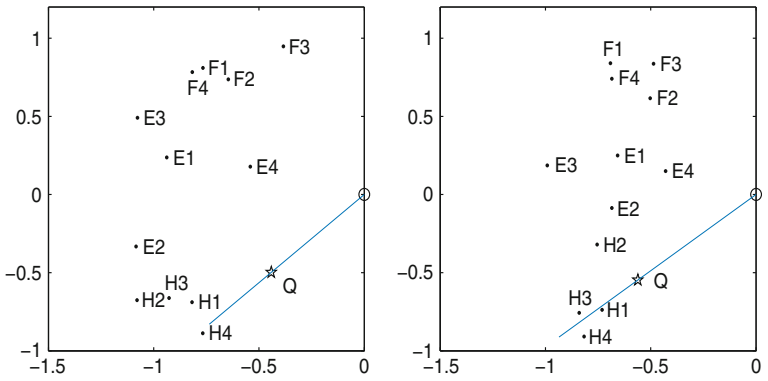

Fig. 5.11 Two first principal components plane for data in Table 5.15, both in the original format (*left*) and after tf-idf normalization (*right*). Query Q combining fuel-price- relief-tax keywords is mapped to the pentagram; the line connects it with the origin 0

As one can see, both representations separate the three subjects, F, E and H, more or less similarly, and provide the query Q with a rather good resolution.Taking into account the position of the origin of the concept space – the circle in the middle of the right boundary, the four H items are indeed have very good angular similarity to the pentagram representing the query Q.

The SVD representation of documents is utilized in other applications such as text mining, web search, text categorization, software engineering, etc.

F5.3.2 Latent Semantic Analysis: Formulation

The full SVD of data matrix F leads to equation $F = ZMC^T$ where Z and C are matrices whose columns are right and left normed singular vectors of F and M is a diagonal matrix with the corresponding singular values of F on the diagonal. By leaving only K columns in these matrices, we substitute matrix F by matrix $Z_K M_K {C_K}^T$ so that the entities are represented in the K-dimensional concept space by the rows of matrix $Z_K {M_K}^{1/2}$.

To translate a query presented as a vector q in the V-dimensional space into the corresponding point u in the K-dimensional concept space, one needs to take the product $g = {C_K}^T q$, which is equal to $g = z{M_K}^{1/2}$ according to the definition of singular values, after which z is found as $z = g{M_K}^{-1/2}$. Specifically, k-th coordinate of vector z is calculated as $z_k = <c_k, q> / {\mu_k}^{1/2} (k = 1, 2, \ldots, K)$.

The similarities between rows (documents), corresponding to row-to-row inner products in the concept space are computed as $Z_K {M_K}^2 {Z_K}^T$ and, similarly, the similarities between columns (keywords) are computed according to the dual formula $C_K {M_K}^2 {C_K}^T$. Applying this to the case of the K-dimensional point z representing the original V-dimensional vector q, its similarities to N original entities are computed as $z{M_K}^2 {Z_K}^T$.

C5.3.3 Latent Semantic Analysis: Computation

Let X be $N \times V$ array representing the original frequency data. To convert that to the conventional coding, in which all the entries larger than 1 are coded by 2, one can use this operation:

≫ Y=min(X,2*ones(N,V));

Computing vector *df* of document frequencies over matrix *Y* can be done with this line:

≫df=zeros(1,V); for k=1:V;df(k)=length(find(Y(:,k)>0));end;

and converting *df* to the inverse-document-frequency weights, with this:

≫ idf=log(N./df);

After that, it-idf normalization can be made by using command

≫YI=Y.*repmat(idf, N,1);

Given term frequency matrix *Y*, its *K*-dimensional concept space is created with commands:

≫ [z,m,c]=svd(Y);
≫zK=z(:, [1:K]); cK=c(:, [1:K]); mK=m([1:K], [1:K]);

Worked example 5.11. Drawing Figure 5.11

Consider that z is the matrix of normed document score singular vectors, c the matrix of normed keyword loading vectors, and m the matrix of singular values of the data in Table 5.15 as they are.

To draw the left part of Fig. 5.11, one can define the coordinates with vectors z1 and z2:

≫ z1=z(:,1)*sqrt(m(1,1)); %first coordinates of N entities in the concept space
≫ z2=z(:,2)*sqrt(m(2,2)); %second coordinates of N entities in the concept space

Then prepare the query vector and its translation to the concept space:
≫ q=[0 0 1 0 0 1 1 0 1 0]; % "fuel, price, relief, tax" query vector
≫ d1=q*c(:,1)/sqrt(m(1,1)); %first coordinate of query q in the concept space
≫ d2=q*c(:,2)/sqrt(m(2,2)); %second coordinate of query q in the concept space

After this, an auxiliary text data should be put according to MatLab requirements:
≫ tt={'E1','E2',. . ., 'H4'}; % cell of 12 names of the items in data matrix
≫ll=[0:.04:1.5]; zd1=d1*ll; zd2=d2*ll;
% pair zd1, zd2 will draw a line through origin and point (d1,d2)

Now we are ready for plotting the left drawing on Fig. 5.11:
≫ subplot(1,2,1);
≫ plot(u1,u2,'k.',d1,d2,'kp',0,0,'ko',ud1,ud2);text(u1,u2,tt);

≫ text(d1,d2,′ Q′);
≫ axis([−1.5 0 −1 1.2]);

The arguments of plot command here are:
u1,u2,′k.′ – black dots corresponding to the original entities;
d1,d2,′kp′ – black pentagram corresponding to query q;
0,0,′ko′ – black circle corresponding to the space origin;
ud1,ud2 – line through the query and the origin.

Command *text* provides for string labels at corresponding points. Command *axis* specifies the boundaries of the Cartesian plane box on the figure, which can be used for making different plot boxes uniform.

The plot on the right of Fig. 5.11 is coded similarly by using SVD of tf-idf matrix *YI* rather than *Y*.

5.4 Application: Correspondence Analysis

P5.4.1 Correspondence Analysis: Presentation

Correspondence Analysis is an extension of PCA to contingency tables taking into account the specifics of co-occurrence data: they are not only comparable across the table but also can be meaningfully summed up across the table. This leads to a unique way of standardization of such data – by using the Quetelet coefficients rather than the original frequencies, which is an advantage over the common situations in which the data standardization is rather arbitrary.

Correspondence Analysis (CA) is a method for visually displaying both row and column categories of a contingency table $P = (p_{ij}), i \in I, j \in J$, in such a way that distances between the presenting points reflect the patterns of co-occurrences in P. This method is usually introduced as a set of dual heuristics applied simultaneously to rows and columns of the contingency table (see, for example, Lebart, Morineau and Piron 1995). Yet there is a way for introducing CA as a decoder based data recovery technique similar to that used for introducing PCA above. According to this perspective (Mirkin 1996), CA is a version of PCA differing from PCA, due to the specifics of contingency data, in the following aspects:

(i) CA decoder applies to the relative Quetelet coefficients rather than to the original frequency data;
(ii) Both rows and columns are assigned with weights equal to the marginal frequencies – these weights are used in the least-squares criterion as well as the orthogonality conditions;
(iii) Both rows and columns are visualized on the same display in such a way that the geometric distances between the representing them points reflect the so-called chi-square distances between row and column conditional frequency profiles (see (5.24) in 5.4.2);
(iv) Data scatter is measured by the Pearson chi-square association coefficient rather than just the sum of squares of Quetelet coefficients.

Worked example 5.12. Correspondence analysis of Protocol/Attack contingency table

Consider Table 5.17, a copy of Table 3.6 representing the distribution of protocol types and attack types according to Intrusion data in Section 1.2: totals on its margins show separate summary distributions of protocols and attacks.

To apply the method to data in Table 5.17, we first transform it into Quetelet coefficients (see Table 5.18).

This standardization does make the data structure somewhat sharper, as has been explained in Section 3.4. One can see, for example, that $q = 9\ (= 900\%)$ for the equivalent pair *(Icmp, surf)*. But the transformation $p \Rightarrow q$ does not work alone in CA. It is coupled with the weighting of each row and column by its corresponding marginal probability so that the squared errors in the criterion are weighted by products of the marginal probabilities. Moreover, the vector norm is weighted by them too.

Figure 5.12 represents a CA visualization of Table 5.17 derived as described in Section 5.4.3, which shows indeed that the equivalent Smurf and Icmp fall in the same place; Norm is much associated to Udp, and Apache and Saint are between Tcp and Icmp protocols. The Norm category slightly falls out: all points representing columns should be within the convex closure of the three protocol points because of Equation (5.23) relating row points and column points.

F5.4.2 Correspondence Analysis: Formulation

Correspondence Analysis (CA) is a method for visually displaying both row and column categories of a contingency table $P = (p_{ij})$, $i \in I$, $j \in J$, in such a way that distances between the presenting points reflect the pattern of co-occurrences in P.

Table 5.17 Protocol/attack contingency table for intrusion data

Category	Apache	Saint	Smurf	Norm	Total
Tcp	23	11	0	30	64
Udp	0	0	0	26	26
Icmp	0	0	10	0	10
Total	23	11	10	56	100

Table 5.18 Quetelet indexes, per cent, for the Protocol/Attack contingency Table 5.17

Category	Apache	Saint	Smurf	Norm
Tcp	**56.25**	**56.25**	−100.00	−16.29
Udp	−100.00	−100.00	−100.00	**78.57**
Icmp	−100.00	−100.00	**900.00**	−100.00

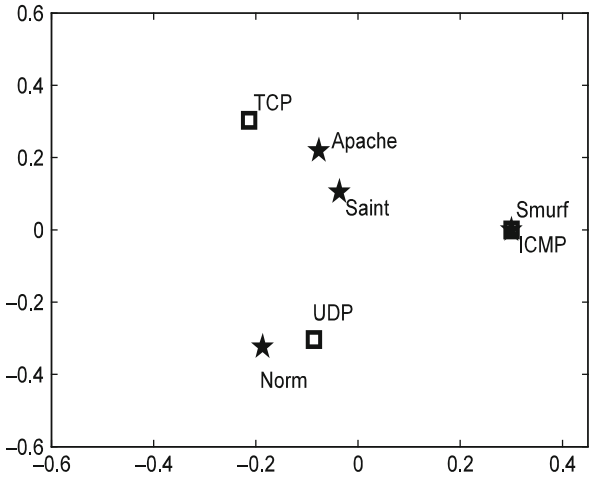

Fig. 5.12 Visualization of Protocol/Attack contingency data in Table 5.17 using Correspondence Analysis. *Squares* stand for protocol types and *stars* for attack categories

To be specific, let us take on the issue of visualization of P on a 2D plane so that we are looking for just two approximating factors, $u_1 = (v_1,\ w_1)$ where $v_1 = (v_1(i))$ and $w_1 = (w_1(j))$ and $u_2 = (v_2,\ w_2)$ where $v_2 = (v_2(i))$ and $w_2 = (w_2(j))$, with $I \cup J$ as their domain, such that each row $i \in I$ is displayed as point $u(i) = (v_1(i),\ v_2(i))$ and each column $j \in J$ as point $u(j) = (w_1(j),\ w_2(j))$ on the plane as shown in Fig. 5.12.

The $|I|$-dimensional vectors v_t and $|J|$-dimensional vectors w_t constituting the $u_t (t = 1, 2)$ are calculated to approximate the relative Quetelet coefficients $q_{ij} = p_{ij}/(p_{i+}p_{+j}) - 1$ rather than the co-occurences p_{ij} themselves, according to equations:

$$q_{ij} = \mu_1 v_1(i) w_1(j) + \mu_2 v_2(i) w_2(j) + e_{ij} \tag{5.20}$$

where μ_1 and μ_2 are positive reals, by minimizing the weighted least-squares criterion

$$E^2 = \sum_{i \in I} \sum_{j \in J} p_{i+} p_{+j} {e_{ij}}^2 \tag{5.21}$$

with regard to μ_t, v_t, w_t, subject to conditions of weighted orthonormality:

$$\sum_{i \in I} p_{i+} v_t(i) v_{t'}(i) = \sum_{j \in J} p_{+j} w_t(j) w_{t'}(j) = \begin{cases} 1, & \text{if } t = t' \\ 0, & \text{if not} \end{cases} \tag{5.22}$$

where $t,\ t' = 1, 2$.

The weighted criterion E^2 is equivalent to the unweighted least-squares criterion L^2 applied to the matrix R that has Pearson indexes $r_{ij} = q_{ij}(p_{i+}p_{+i})^{1/2} = (p_{ij} - p_{i+}p_{+i})/(p_{i+}p_{+i})^{1/2}$ as its entries. This implies that the factors v and w are

determined by the singular-value decomposition of matrix $R = (r_{ij})$. More explicitly, the two maximal singular values μ_t and corresponding singular vectors $f_t = (f_{it})$ and $g_t = (g_{it})$ of matrix R, defined by equations $Rg_t = \mu_t f_t$, $R^T f_t = \mu_t g_t$ $(t = 1, 2)$ determine the optimal values μ_t and optimal solutions to the problem of minimization of (5.21)–(5.22). Indeed, these singular triplets relate to the optimal solution according to equations $v_t(i) = f_{it}/(p_{i+}{}^{1/2})$ and $w_t(j) = g_{jt}/(p_{+j}{}^{1/2})$. The proof follows from the first-order optimality conditions for the Lagrange function of the problem (5.21)–(5.22).

The singular triplet equations can be rewritten in terms of v_t and w_t, as follows:

$$\sum_{j\in J} \frac{p_{ij}}{p_{i+}} w_t(j) = \mu_t v_t(i), \quad \sum_{i\in I} \frac{p_{ij}}{p_{+j}} v_t(i) = \mu_t w_t(j) \tag{5.23}$$

To prove the left-hand equation, take equation $Rg_t = \mu_t f_t$ in its component-wise form, $\sum_{j\in J} r_{ij} g_j = \mu f_i$ (index t omitted for the sake of convenience) and substitute by vectors v and w defined above: $\sum_{j\in J} r_{ij} (\frac{p_{+j}}{p_{i+}})^{1/2} w(j) = \mu v(i)$. This is equivalent to $\sum_{j\in J} (\frac{p_{ij}}{p_{i+}} - p_{+j}) w(j) = \mu v(i)$. To complete the proof, equation $\sum_j p_{+j} w(j) = 0$ is to be proven. To do that, let us first prove that vector g_0 whose components are $p_{+j}{}^{1/2}$ is a singular vector of R corresponding to singular value 0 (the other singular vector is equal to $f_0 = (p_{i+}{}^{1/2})$). Indeed, $\sum_{j\in J} r_{ij} p_{+j}{}^{1/2} = (1/p_{i+}{}^{1/2}) \sum_{j\in J} (p_{ij} - p_{i+} p_{+j}) = (1/p_{i+}{}^{1/2})(p_{i+} - p_{i+}) = 0$. Then the equation $\sum_j p_{+j} w(j) = 0$ follows from the fact that all the singular vectors are mutually orthogonal so that singular vector g corresponding to w is orthogonal to g_0, which proves the statement. The right-hand equation can be proven in a similar way, from equation $R^T f_t = \mu_t g_t$.

Equations (5.23) are referred to as transition equations and considered to justify the joint display of rows and columns because the row-points $v_t(i)$ appear to be averaged column-points $w_t(j)$ and, vice versa, the column-points appear to be averaged versions of the row-points, up to the singular value of μ_t course.

The mutual location of the row-points is considered as justified by the fact that between-row-point squared Euclidean distances $d^2(u(i), u(i'))$ approximate the chi-square distances between corresponding rows of the contingency table. Specifically, chi-square distance is defined a weighted squared Euclidean distance:

$$\chi^2(i, i') = \sum\nolimits_{j\in J} p_{+j} (q_{ij} - q_{i'j})^2 = \sum\nolimits_{j\in J} (p_{ij}/p_{i+} - p_{i'j}/p_{i'+})^2 / p_{+j}. \tag{5.24}$$

Here $u(i) = (v_1(i), v_2(i))$ for v_1 and v_2 rescaled in such a way that their norms are equal to μ_1 and μ_2, respectively. A similar property holds for columns j, j'. In fact, it is the right-hand item in (5.24) which is used to define the chi-squared distance (Lebart et al. 1995), but the definition in terms of Quetelet coefficients in the middle of (5.24) (Mirkin 1996) looks more natural. The distance is dubbed chi-square distance because of its links to the chi-square coefficient for table P. First of all if we take the weighted chi-square summary distance to 0, $\sum_{i\in I} p_{i+} \chi^2(i, 0)$ where 0 is put instead of $q_{i'j}$ in (5.24), it is easy to see that this is the Pearson

chi-squared coefficient, without the factor N of course, which is simultaneously the expression for the data scatter according to criterion E^2 in (5.21):

$$\sum_{i\in I} p_{i+}\chi^2(i,0) = \sum_{i\in I}\sum_{j\in J} p_{i+}p_{+j}e_{ij}^2 = X^2/N \tag{5.25}$$

The weighted data scatter is equal to the scatter of R, the sum of its squared entries $T(R)$, which can be easily calculated from the definition of R. Indeed, $\sum_{i\in I}\sum_{j\in J}(p_{ij} - p_{i+}p_{+j})^2/(p_{i+}p_{+j}) = X^2/N$. This implies that

$$X^2/N = \mu_1^2 + \mu_2^2 + E^2 \tag{5.26}$$

which can be seen as a decomposition of the contingency data scatter, expressed by X^2, into contributions of the individual factors, μ_1^2 and μ_2^2, and unexplained residuals, E^2. (Only two factors are considered here, but the number of factors to be found can be raised up to the rank of matrix R with no other changes).

In a common situation, the first two singular values account for a major part of X^2, thus justifying the use of the plane of the first two factors for visualization of the interrelations between I and J.

C5.4.3 Correspondence Analysis: Computation

Given a contingency table P, the computation of correspondence analysis factors can go in three steps: (a) computing Pearson index matrix R, (b) finding the singular decomposition of R and the two first correspondence analysis factors, and (c) visualization of the joint display of rows and columns of P. Here are MatLab commands for these.

(a) Computing Pearson index matrix R

```
≫ Pc=sum(P); Pr=sum(P′); total=sum(Pc);
≫ P=P/total; %relative frequencies
≫ Pc=Pc/total; %column relative frequencies
≫ Pr=Pr/total; %row relative frequencies
≫ Prod=Pr′*Pc; % matrix of products
≫ rProd=Prod.^(0.5); % square roots of products
≫ r=(P−Prod)./rProd; % Pearson index matrix
```

(b) Finding the correspondence analysis factors:

```
≫ [a,mu,b]=svd(r);
≫ % finding first factor
≫ x1=(a(:,1).*sqrt(Pr′))*sqrt(mu(1,1));
≫ y1=(b(:,1).*sqrt(Pc′))*sqrt(mu(1,1));
≫ % finding second factor
≫ x2=(a(:,2).*sqrt(Pr′))*sqrt(mu(2,2));
≫ y2=(b(:,2).*sqrt(Pc′))*sqrt(mu(2,2));
```

As a bonus, one can estimate the proportion of data scatter, the chi-squared, taken into account by the factors, and display it on the screen:

```
≫ yy=r.*r; chi=sum(sum(yy))% data scatter
≫ ccn=(mu(1,1)^2+mu(2,2)^2)*100/chi;
   %contribution of the first two
≫ disp('Contribution of the solution:'); ccn
```

(c) Visualization of the joint display of rows and columns of P. The plot is easy to do with command

```
≫ plot(x1,x2,'ks', y1,y2,'kp');
```

Yet to make the points annotated with row and column names, which are to be available in a string cell termed say "names", the joint set of rows and columns should get their x-coordinate and y-coordinate vectors, z1 and z2 below:

```
≫ z1=[x1' y1']; z2=[x2' y2']; text(z1,z2,names);
≫ v=axis; axis(1.5*v);
```

The last line is to make the plot to look tighter by extending its boundaries.

Q.5.13. In many situations, (a) the first singular vector all positive and (b) the second singular vector half negative. Why can be that? **A.** A typical situation: (a) All features are positively correlated which implies that the first eigenvector is positive; (b) the second must be orthogonal to the first, to make 0 their inner product.

Q.5.14. For the data in Table 5.10, as well as many others, svd function in MatLab produces first singular vectors z and c negative, which contradicts the meaning of them as talent scores and subject loadings. Can anything be done about that? **A.** Yes, they can be changed to $-z$ and $-c$ without compromising their singular vector status.

Q.5.15. Is matrix

$$\begin{matrix} 1 & 2 \\ 2 & 1 \end{matrix}$$

of rank 1 or not? **A.** The rows are not proportional to each other, thus not.

Q.5.16. Prove that if matrix Y is symmetric then its eigenvalues and vectors (λ, z) are simultaneously its singular triplets $(\lambda,\ z,\ z)$.

Q.5.17. Find a matrix of rank 1 that is the nearest to matrix in Q.5.15 according to the least-squares criterion. **A.** The solution is given by the first singular value and corresponding singular vectors which are the same as the first eigenvalue and and corresponding eigen-vector, $\lambda = 3$ and $z = (1/\sqrt{2},\ 1/\sqrt{2})$, thus leading to matrix

$$\begin{matrix} 3/2 & 3/2 \\ 3/2 & 3/2 \end{matrix}$$

as the solution.

Q.5.18. For positive a and b, inequality $a < b$ can be equivalently expressed as $a/b < 1$. The difference between a and b does not change if $c > 0$ is added to both of them, but the ratio does. Prove that for any $c > 0$, $(a + c)/(b + c) > a/b$ – this would illustrate that the further away the positive data are from zero, the greater the contribution of the first principal component.

Q.5.19. There is another representation of a singular value problem as an eigenvalue problem. Given a rectangular $N \times V$ matrix Y, consider a $(N + V) \times (N + V)$ matrix Y^* that consists of four blocks, two of which, the diagonal $N \times N$ and $V \times V$ blocks, are all zeros:

$$Y^* = \begin{pmatrix} 0 & Y \\ Y^T & 0 \end{pmatrix}.$$

Prove that a triplet $(\mu,\ z,\ c)$ is singular for Y if and only if μ is eigenvalue for Y^* corresponding to eigenvector $y = (z, c)$ in which first N components are taken by z and the remaining V components, by c. **A.** Consider an arbitrary eigenvalue μ and corresponding eigenvector y of matrix Y^* and denote the vector of its first N component by z, and the rest by c so that $y = (z, c)$. The product Y^*y will have its first N components equal to $0z + Yc = Yc$ and the next V components equal to $Y^Tz + 0c = Y^Tz$. Since $Y^*y = \mu y$, that means that $Yc = \mu z$ and $Y^Tz = \mu c$, which proves the statement.

Q.5.20. Prove that if an eigenvector $y = (z, c)$ of Y^* is normed, then its components z and c both have norms of $0.5^{1/2}$. **A.** Indeed, $\|y\|^2 = \|z\|^2 + \|c\|^2$ because these are just sums of the squared components. On the other hand, as proven in Q.5.19, z and c are singular vectors of Y so that they must have equal norms because of Property 1 in Section 5.2.2. This leads to equation $\|z\|^2 = \|c\|^2 = 1/2$, which proves the statement.

Q.5.21. Prove that if μ is an eigenvalue of Y^* corresponding to its eigenvector $y = (z, c)$ then so is its negation $-\mu$ corresponding to eigenvector $y = (-z, c)$. **A.** Indeed, equations $Yc = \mu z$ and $Y^Tz = \mu c$ hold if and only if $Yc = (-\mu)(-z)$ and $Y(-z) = (-\mu)c$.

5.5 Summary

This chapter introduces the concept of data summarization as a coder-decoder pair and describes the method of principal components (PCA) as a data-driven model in this framework. Luckily, this model is underlied with a well developed mathematical theory of singular value decomposition (SVD) for rectangular matrices. Unlike the conventional formulation of PCA, this model does not require to postulate that the principal components are to be linear combinations of features. This property is rather derived from the model. Yet the PCA model itself is rather simplistic

and suggests that further thinking on better data summarization models should be undertaken.

Three applications of PCA – scoring hidden factors, data visualization, and feature space reduction are illustrated with further instructions and worked examples. Two more distant applications, Latent semantic analysis (for disambiguation in document retrieval) and Correspondence analysis (for visualization of contingency tables), are explained in full, too.

References

Deerwester, S., Dumais, S., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis, J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)

Hair, J.F., Black, W.C., Babin, B.J., Anderson, R.E.: Multivariate Data Analysis, 7th edn, Prentice Hall, ISBN-10: 0-13-813263-1 (2010)

Kendall, M.G., Stewart, A.: Advanced Statistics: Inference and Relationship, 3rd edn. Griffin, London, ISBN: 0852642156 (1973)

Lebart, L., Morineau, A., Piron, M.: Statistique Exploratoire Multidimensionelle. Dunod, Paris, ISBN 2-10-002886-3 (1995)

Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge, Cambridge University Press (2008)

Mirkin, B.: Mathematical Classification and Clustering. Dordrecht, Kluwer Academic Press (1996)

Mirkin, B.: Clustering for Data Mining: A Data Recovery Approach. London, Chapman & Hall/CRC, ISBN 1-58488-534-3 (2005)