B Principal Components Analysis

Suppose we have a set of zero-mean n-dimensional data vectors whose elements are correlated (figure B.1). If the data is not zero-mean, we can make it so by subtracting the average value beforehand. Because of the correlation, the data is partially redundant; knowledge of one variable gives us approximate information about the values of other variables.

It might be more natural to describe the data in terms of its variation along directions $\mathbf{v}_1$ and $\mathbf{v}_2$ (figure B.1). Most of the positional information is conveyed by the distance along $\mathbf{v}_1$ with the distance along $\mathbf{v}_2$ adding only a small correction. For highly correlated data, the contribution from $\mathbf{v}_2$ will be small compared to that of $\mathbf{v}_1$, so we might choose to ignore $\mathbf{v}_2$ completely. This introduces some error but gives us a more compact description. Perfectly correlated data would lie along a line (or on a hyperplane, in general), so $\mathbf{v}_2$ would be zero and we would lose nothing by describing the data entirely in terms of the distance along $\mathbf{v}_1$.

Correlation can arise because a system has internal variables $v_1, v_2, \ldots v_n$ the effects of which are indirectly observed through intermediate variables $x_1, x_2, \ldots x_n$. Each observable variable is potentially affected by each internal variable. The observed variables may depend linearly on the internal variables, for example

$$x_1 = a_{11}v_1 + a_{12}v_2 + \ldots$$
$$x_2 = a_{21}v_1 + a_{22}v_2 + \ldots$$
$$\vdots$$

By observing correlations in the observed data, we may be able to identify the internal variables that control the system behavior. Knowing the correlations, we can then measure several observable variables and get good estimates of the internal state.

The purpose of principal components analysis (PCA) is to identify the important directions of variation of a data set. Singular value decomposition and the Karhunen-Loève transform have similar goals and are closely related techniques. The result may be a more natural coordinate system better aligned with the major axes of variation of the data. Sometimes these axes will correspond to natural features of the data.

Consider an arbitrary unit-length vector $\mathbf{v}$. The projection $\mathbf{x}_v$ of a point $\mathbf{x}$ onto $\mathbf{v}$ is the point on $\mathbf{v}$ that is closest to $\mathbf{x}$

$$\mathbf{x}_v = (\mathbf{x}^T\mathbf{v})\mathbf{v}. \tag{B.1}$$

This is a vector with magnitude $(\mathbf{x}^T\mathbf{v})$ extended along the unit vector $\mathbf{v}$. Note that the residual component, $\epsilon = \mathbf{x} - (\mathbf{x}^T\mathbf{v})\mathbf{v}$, is orthogonal to $\mathbf{v}$, that is, $\epsilon^T\mathbf{v} = \mathbf{x}^T\mathbf{v} - (\mathbf{x}^T\mathbf{v})\mathbf{v}^T\mathbf{v}$ $= \mathbf{x}^T\mathbf{v} - (\mathbf{x}^T\mathbf{v}) = 0$.

Obviously, the projected magnitude depends on the orientation of $\mathbf{v}$. Given a set of vectors $\{\mathbf{x}_i\}$, we can search for a unit vector $\mathbf{v}$ that maximizes the mean squared value of

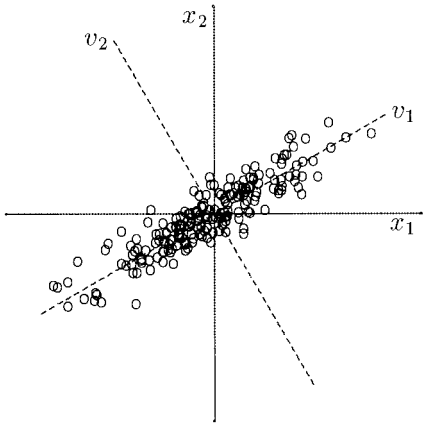

Figure B.1
Correlated data are partially redundant because knowledge of one variable gives approximate knowledge of the other variables. In two dimensions, perfectly correlated data lie on a straight line; in n dimensions, they lie on a lower dimensional subspace.

this projected distance. By definition, this yields the first principal component of the data set. That is, the first principal component of a set of zero-mean vectors $\{\mathbf{x}_i\}$, $i = 1 \ldots m$, $E[\mathbf{x}] = 0$, is the vector $\mathbf{v}_1$, which maximizes the variance of the projected magnitudes

$$E\left[(\mathbf{v}_1^T \mathbf{x})^2\right]. \tag{B.2}$$

After finding the first principal component, subtract the projection along that direction to get an $(n-1)$-dimensional data set lying in a subspace orthogonal to $\mathbf{v}_1$. Then search for a variance maximizing vector in this reduced space to obtain the second principal component, $\mathbf{v}_2$. Further repetition yields the remaining components, $\mathbf{v}_3, \mathbf{v}_4, \ldots$. Because the vectors $\mathbf{v}_i$ are unit-length and orthogonal, they form an orthonormal set

$$\mathbf{v}_i^T \mathbf{v}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{B.3}$$

The vectors $\mathbf{v}_i$ can be calculated as follows. Let $y = \mathbf{x}^T \mathbf{v}$ be the projected distance of $\mathbf{x}$ along $\mathbf{v}$. Since $E[\mathbf{x}] = 0$, $E\left[y\right] = 0$. The variance of y is then

$$\begin{aligned} \sigma_y^2 &= E\left[y^2\right] = E\left[y^T y\right] = E\left[\mathbf{v}^T \mathbf{x}\mathbf{x}^T \mathbf{v}\right] \\ &= \mathbf{v}^T E\left[\mathbf{x}\mathbf{x}^T\right] \mathbf{v} \\ &= \mathbf{v}^T \mathbf{R} \mathbf{v} \end{aligned} \tag{B.4}$$

where $\mathbf{R} = E\left[\mathbf{x}\mathbf{x}^T\right]$ is the covariance matrix of the vectors $\mathbf{x}$. $\mathbf{R}$ is an $n \times n$ symmetric matrix so its eigenvalues are real. Because it is a covariance matrix, its eigenvalues are nonnegative. To maximize the variance subject to the condition that $\|\mathbf{v}\| = 1$ we can use the cost function

$$\mathbf{v}^T\mathbf{R}\mathbf{v} - \mu(\mathbf{v}^T\mathbf{v} - 1) \tag{B.5}$$

where μ is a Lagrange multiplier. Taking the derivative with respect to $\mathbf{v}$ and setting equal to zero gives

$$2\mathbf{R}\mathbf{v} - 2\mu\mathbf{v} = 0 \tag{B.6}$$

$$\mathbf{R}\mathbf{v} = \mu\mathbf{v}. \tag{B.7}$$

This is an eigenvalue problem. For a nonnull solution to exist, μ must be chosen to satisfy the characteristic equation

$$\det(\mathbf{R} - \mu\mathbf{I}) = 0. \tag{B.8}$$

That is, μ must be an eigenvalue of $\mathbf{R}$ and $\mathbf{v}$ must be the corresponding eigenvector. Taking λ as the eigenvalue and substituting into (B.4), we have

$$\begin{aligned}\sigma_y^2 &= \mathbf{v}^T\mathbf{R}\mathbf{v} = \mathbf{v}^T(\lambda\mathbf{v}) = \lambda\|\mathbf{v}\|^2 \\ &= \lambda.\end{aligned} \tag{B.9}$$

In summary, the direction vector that maximizes the variance of the projection is given by the principal eigenvector $\mathbf{v}_1$ of $\mathbf{R}$, and the corresponding eigenvalue λ_1 measures the variance of the projection along that direction. Similarly, eigenvector $\mathbf{v}_2$ maximizes the variance of the projection in the residual space orthogonal to $\mathbf{v}_1$ and so on. Assuming the eigenvalues are distinct, we can number them in order of decreasing magnitude

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n \geq 0. \tag{B.10}$$

Assuming $\mathbf{R}$ has full rank, its eigenvectors form an alternate coordinate system with coordinate vectors numbered in order of their importance in explaining the variation of the data set. A vector $\mathbf{x}$ can be expressed as the sum of its projections along the orthogonal components $\mathbf{v}_i$

$$\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \tag{B.11}$$

where $\alpha_i = \mathbf{x}^T\mathbf{v}_i$ is the projection of $\mathbf{x}$ onto the i^{th} coordinate vector. The vector $\alpha = (\alpha_1, \alpha_2, \ldots \alpha_n)$ is the representation of $\mathbf{x}$ in the new coordinate system. If $\mathbf{V}$ denotes the

$n \times n$ matrix containing eigenvector $\mathbf{v}_i$ in column i then in matrix notation

$$\mathbf{x} = \mathbf{V}\alpha. \tag{B.12}$$

The eigenvectors are orthonormal so $\mathbf{V}$ is unitary

$$\mathbf{V}^T\mathbf{V} = \mathbf{I}. \tag{B.13}$$

Note that $\|\mathbf{x}\|^2 = \mathbf{x}^T\mathbf{x} = \alpha^T\mathbf{V}^T\mathbf{V}\alpha = \alpha^T\alpha = \|\alpha\|^2$.

As noted, the principal component vectors of a data set form an orthogonal coordinate system with coordinate vectors numbered in order of their importance in explaining the variation of the data set. If $\mathbf{R}$ has rank $r < n$, it is singular and $n - r$ of its eigenvalues are zero. Some elements of $\mathbf{x}$ are exactly predicted by linear combinations of other elements so the data lies on an r-dimensional linear subspace embedded in the n-dimensional space. This provides an opportunity for data compression because the data can be described by fewer numbers in the new coordinate system. There is no variation along $n - r$ dimensions so we could omit those elements in the representation and obtain a more compact description without loss of information. Even if $\mathbf{R}$ has full rank, some of its eigenvalues may be small in which case the data has little variation along the corresponding dimensions. Element i contributes $\alpha_i\mathbf{v}_i$ to the position information. The mean squared error introduced by omitting element i is $\langle\alpha_i^2\rangle = \lambda_i$. Obviously, if we omit any component, it should be the one with the smallest eigenvalue since this incurs the smallest error. Likewise, if we omit m components, they should be the elements with the m smallest eigenvalues. In general, the rank r linear projection of a data set, $r < n$, with the lowest mean squared error is the projection onto the first r principal components of the data.

In practice, measurement noise and numerical errors complicate the process of calculating the eigenvectors. Some of the estimated eigenvalues may be very small but not identically zero so some judgment is required to decide if they should be set to zero or not. Numerical analysis texts suggest singular value decomposition as a preferable method for obtaining the projection directions since formation of the covariance matrix tends to square the numerical errors.

As a side note, figure B.2 illustrates the effect of not removing the mean vector. For the zero-mean data plotted in (a), the eigenvectors of the data correlation matrix accurately reflect the axes of data variation and the eigenvalues (0.9595 and 0.0417) estimate the variance along those directions. In (b), the same data is offset by $\mathbf{m} = (1, 2)$; the resulting eigenvectors are rotated and the eigenvalues (5.7016 and 0.2968) are larger. In this case, the first eigenvalue is dominated by the length of the offset vector, $\|\mathbf{m}\| = 5$. As noted in appendix A the maximum stable learning rate for gradient descent is inversely proportional to the maximum eigenvalue of the data correlation matrix so smaller learning rates must

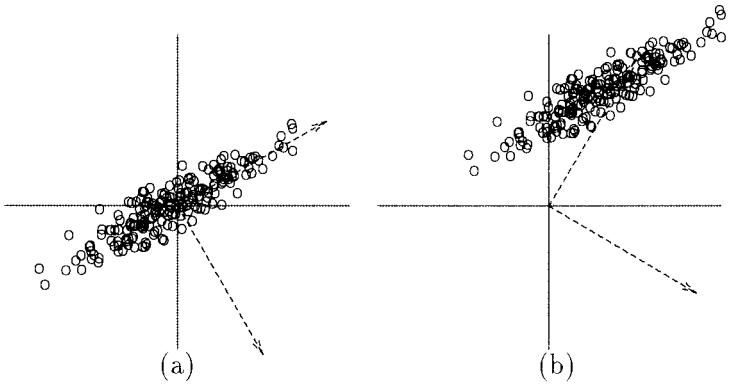

Figure B.2
Effect of nonzero-mean on the eigenvectors of the correlation matrix. (a) For zero-mean data, the eigenvectors of the correlation matrix indicate the main axes of data variation and the eigenvalues reflect the variance along each dimension. (b) For nonzero-mean data, the eigenvectors are rotated and the eigenvalues are influenced by the length of the offset vector.

be used to avoid stability problems and learning may take longer if the mean is not removed.

B.1 Autoencoder Networks and Principal Components

Consider a network with n inputs, $h < n$ *linear* hidden units, and n *linear* output units (figure B.3). What happens if we train the network to reproduce the input vector at its output? Given an input, the goal is to reproduce it at the output so the network acts as autoencoder, mapping an input pattern to itself.

This may seem pointless but note that $h < n$ so the hidden layer acts as a bottleneck that forces the network to form a compressed representation of the data. The hidden layer activities are a linear function of the inputs but the hidden layer is smaller than the input dimension so some information must necessarily be lost, in general. The best hidden layer representation will be one that preserves as much information about the input as possible. Ideally, it will ignore nonessential noise and reproduce only the most significant features of the input pattern.

Bourlard and Kamp [57] showed that the optimal (in a minimum mean squared error sense) hidden unit weights are determined by a set of vectors spanning the singular value decomposition of the input data. That is, the ideal representation formed at the hidden layer spans the same space as the h eigenvectors corresponding to the h largest eigenvalues of the covariance matrix of the training data. The network is linear so it can be collapsed

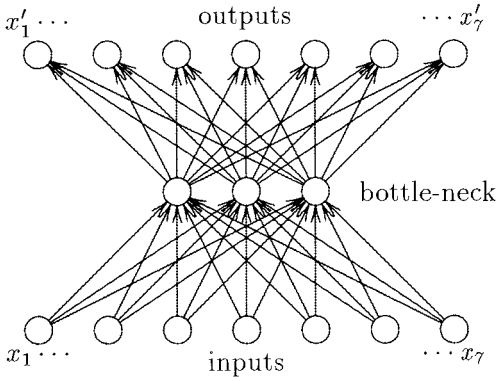

Figure B.3
An autoencoder network maps an input vector to itself. Given an input, the goal is to reproduce it at the output. A small hidden layer acts as a bottleneck and forces the network to find a compressed representation of the input pattern. With a linear network and a least squares error function, the ideal internal representation is related to the principal components of the training data.

into a single linear transformation $y = Fx$. The rank of F is limited in the preceding by h, the dimension of the hidden layer. From the principal components discussion, we know that the best rank h transformation is the projection onto the first h principal component directions. It turns out that linear hidden nodes are optimal in this case; nonlinear nodes cannot improve the approximation and only cause problems by introducing local minima that may confuse gradient descent optimizers. (If A is the hidden-to-output weight matrix and $\mathbf{h}(\mathbf{x})$ is the vector of hidden unit activities, the function to be minimized is $\langle \|\mathbf{x} - A\mathbf{h}\|^2 \rangle$. The output is a linear function of $\mathbf{h}$ so the optimal $\mathbf{h}(\mathbf{x})$ is a linear transform of the inputs $\mathbf{x}$.)

Baldi and Hornik [18] showed that the error function has a unique minimum at this solution. That is, for a linear network and a quadratic error function, the overall transformation F determined by the orthogonal projection onto the space spanned by the first h eigenvectors of $\mathbf{R}$ is a unique local and global minimum. Saddle points occur for solutions spanning other combinations of h or fewer eigenvectors of $\mathbf{R}$. In principle, this means the solution can be found by gradient descent methods such as back-propagation although conventional linear algebraic methods are generally more efficient.

In general, the solution obtained by training from random initial weights will not be identical to the principal components decomposition because it is only necessary that the hidden unit activities span the same space as the first h principal components. Let B and A be the optimal weight matrices determined by singular value decomposition. B is an $h \times n$ matrix of input-to-hidden weights and A is an $n \times h$ matrix of hidden-to-output

weights. The hidden layer computes $h = Bx$ and the output computes $y = Ah = ABx$. Equivalent results can be obtained by the weights $B' = CB$ and $A' = AC^{-1}$ where C is any invertible $h \times h$ matrix. In general, C will depend on the random initial weights. The minimum identified by Baldi and Hornik [18] is unique in terms of the overall function F, which can be achieved by many different combinations of weight matrices A' and B'.

Because C can be a rotation matrix, there will not be a neat correspondence between the hidden units and the principal components projections. Principal components analysis separates the data into orthogonal components ranked in order of importance. Deletion of the first component will cause a larger error than deletion of the second and so forth. In a linear network trained by gradient descent from random initial weights, the contribution from each hidden unit tends to be more nearly equal. The autoencoder extracts the first h principal components but the matrix C typically spreads their functions approximately equally across the h hidden units [18]. The activities of the hidden units will not necessarily be uncorrelated. This may favor fault tolerance, but it is not necessarily helpful in data compression applications where it is useful to be able to pick components serially in terms of their importance. With PCA, we can do one decomposition and inspect the eigenvalues to see how many components are needed to achieve a desired approximation error. In a trained autoencoder, the functions are mixed among the h hidden units, so this is not possible. To find an approximation spanning the first $h - 1$ components, a completely new network with $h - 1$ hidden units must be trained from scratch.

Many papers have been written on the links between neural networks and principal components analysis. An entire book on subject is [106]. Oja and others [289, 290, 288, 71] have investigated Hebbian learning rules that extract principal components. Biological feasibility and local computation are interesting features of these algorithms. Many are on-line methods requiring no data storage. (However, on-line versions of conventional algorithms also exist.) Others have investigated data compression applications, for example, [334, 333]. More general results for linear networks, including but not limited to PCA, are surveyed in [17].

It is worth noting that, although neural networks are often very nonlinear, analysis of linear networks is informative because networks initialized with small weights tend to compute approximately linear functions in the early stages of training and only become significantly nonlinear after the weights grow to larger values. The initial training dynamics are often dominated by approximately linear interactions.

It should also be noted that a bottleneck structure does not automatically imply that the network must implement a principal components solution. The PCA solution is optimal only for linear networks or single-hidden-layer networks with linear outputs. With several nonlinear layers before and after the bottleneck, the bottleneck units are not linear functions

of the input and cannot be interpreted in terms of principal components. In theory, anything could be transmitted through a bottleneck preceded by a sufficiently complex encoder and followed by a corresponding decoder. There are practical limits to this, of course.

B.2 Discriminant Analysis Projections

Although principal components is useful for data reduction, it does not always produce good directions for discriminating between output classes. As an unsupervised method, it sees only the input vectors and is blind to classification information. The problem is that large variations in the data do not always correspond to useful information; the variation could be due to noise or irrelevant signals from other processes. Ideally we would like to remove these sorts of irrelevant variations during preprocessing, but this is not always possible. Figure B.4 illustrates the problem. In figure B.4a, the main contribution to the variance of the input data comes from the separation between the class means so the directions found by PCA will be useful for discriminating between classes. In figure B.4b, however, the major axis of variation is along direction $\mathbf{v}_1$ but the classes are separated along a minor direction $\mathbf{v}_2$. If $\mathbf{v}_2$ were removed for data compression, a system presented with the reduced data would not be able to separate the classes based only on the information in $\mathbf{v}_1$.

Linear discriminant analysis (LDA), [130], for example, provides a way to reduce dimensionality in a supervised learning context. It has dimensionality reduction properties like principal components analysis, but also accounts for class information in forming the

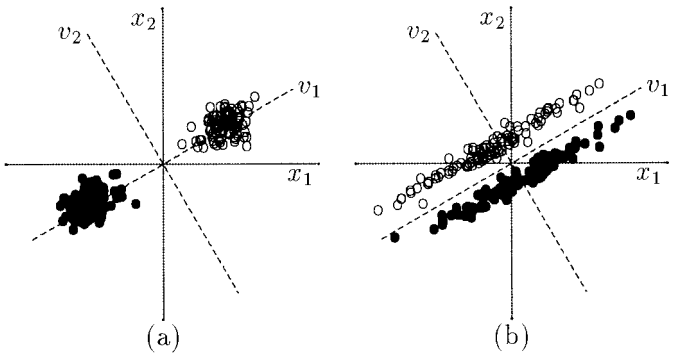

Figure B.4
Although principal components analysis sometimes produces directions useful for discriminating between classes, this is not always true. In (a) the main contribution to the variance of the input data comes from the separation between class means so the principal component directions are useful for discriminating between the classes. In (b) however, the major axis of variation is along direction $\mathbf{v}_1$ but the classes are separated along the minor direction $\mathbf{v}_2$.

projection. Given a set of Gaussian clusters corresponding to different target classes, the goal is to find a lower dimensional linear projection that maximizes the separation between class means and minimizes the spread of each cluster. Ideally, this minimizes overlap of the clusters in the projection and allows for unambiguous classification.

Following [130: chapter 10], suppose the data consists of m points $\mathbf{x}_i$, $i = 1 \ldots m$, grouped into K clusters which correspond to classes. Let $\mathbf{m}_k$, $k = 1 \ldots K$ be the mean vector of cluster k and $\mathbf{m}_o$ be the overall mean vector. The *within-class scatter matrix* $\mathbf{W}$ measures the covariance of the data points around the mean of their respective classes

$$\begin{aligned}\mathbf{W} &= \sum_{k=1}^{K} P_k E\left[(\mathbf{x}-\mathbf{m}_k)(\mathbf{x}-\mathbf{m}_k)^T \mid \mathbf{x} \in \text{class } k\right] \\ &= \sum_{k=1}^{K} P_k \Sigma_k \end{aligned} \tag{B.14}$$

where P_k is the probability that a randomly selected point belongs class k. The *between-class scatter matrix* $\mathbf{B}$ measures the covariance of the class means around the overall mean

$$\mathbf{B} = \sum_{k=1}^{K} P_k(\mathbf{m}_k - \mathbf{m}_o)(\mathbf{m}_k - \mathbf{m}_o)^T \tag{B.15}$$

where $\mathbf{m}_o = E[\mathbf{x}] = \sum_{k=1}^{K} P_k \mathbf{m}_k$. The *mixture scatter matrix* is the overall covariance matrix of all points, regardless of their class

$$\mathbf{M} = E\left[(\mathbf{x}-\mathbf{m}_o)(\mathbf{x}-\mathbf{m}_o)^T\right] = \mathbf{W} + \mathbf{B}. \tag{B.16}$$

These matrices are chosen to be invariant to coordinate shifts.

We would like to find a projection that maximizes the separation between class means and minimizes the sizes of the projected clusters. The ideal projection would yield small, widely separated clusters with no overlap. A number of criteria can be used to measure cluster separability for optimization purposes. These include [130]

- $J_1 = \text{Tr}(S_2^{-1} S_1)$
- $J_2 = \ln|S_2^{-1} S_1| = \ln|S_1| - \ln|S_2|$
- $J_3 = \text{Tr}(S_1) - \mu(\text{Tr}(S_2) - c)$
- $J_4 = \text{Tr}(S_1)/\text{Tr}(S_2)$

where S_1 and S_2 are one of $\mathbf{B}$, $\mathbf{W}$, or $\mathbf{M}$. Possible combinations for $\{S_1, S_2\}$ include $\{\mathbf{B}, \mathbf{W}\}$, $\{\mathbf{B}, \mathbf{M}\}$, and $\{\mathbf{W}, \mathbf{M}\}$. Remarks on these choices can be found in [130].

Consider the matrix $\mathbf{W}^{-1}\mathbf{B}$ using $S_1 = \mathbf{B}$ and $S_2 = \mathbf{W}$. $\mathbf{B}$ and $\mathbf{W}$ are covariance matrices describing the variation between cluster means and within clusters, respectively. The principal eigenvectors of $\mathbf{B}$ maximize the spread of the projected class means. Similarly, the principal eigenvectors of $\mathbf{W}^{-1}$ minimize the average size of the projected clusters (because $1/\lambda$ is an eigenvalue of $\mathbf{W}^{-1}$ if λ is an eigenvalue of matrix $\mathbf{W}$). Intuitively at least, the matrix $\mathbf{W}^{-1}\mathbf{B}$ seems like a reasonable compromise to accomplish both purposes.

It turns out [130] that the linear projection maximizing J_1 consists of projection onto the principal eigenvectors of $S_2^{-1}S_1$. Although $S_2^{-1}S_1$ is not necessarily symmetric, it is the product of matrices with real nonnegative eigenvalues so its eigenvalues will be

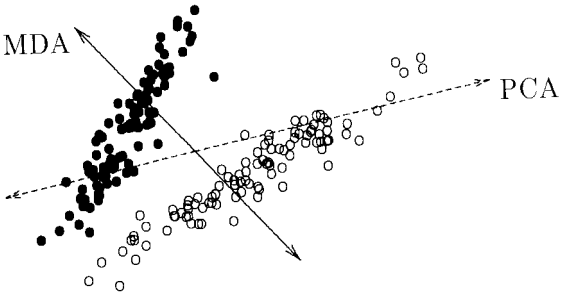

Figure B.5
The projection vectors found by discriminant analysis (MDA) and principal components analysis (PCA) for a simple data set.

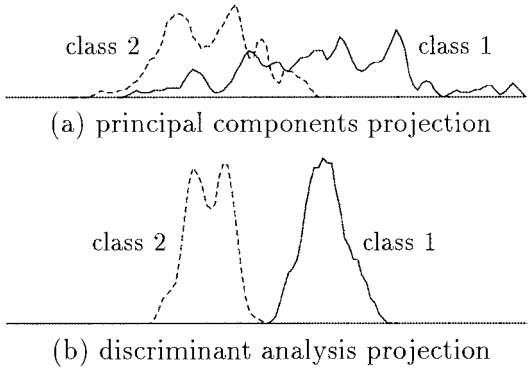

Figure B.6
Projection histograms for the MDA and PCA directions shown in figure B.5: (a) the principal components projection shows cluster overlap, and (b) the discriminant analysis projection separates the clusters well.

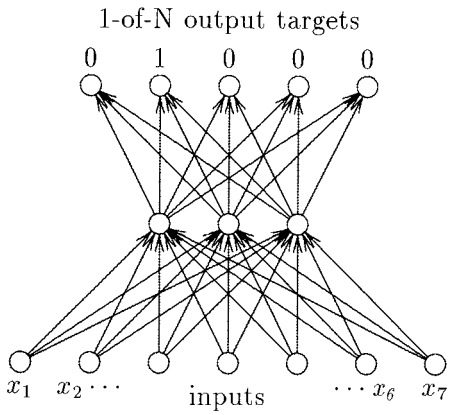

Figure B.7
A single-hidden-layer *linear* network trained to perform classification with a 1-of-N target representation implements a form of discriminant analysis [385, 134].

nonnegative real and its eigenvectors orthogonal. A p-dimensional projection is obtained by extracting the p principal eigenvectors of the matrix. $\mathbf{B}$ has rank $K - 1$, where K is the number of clusters, because it is the covariance matrix of K class means so the dimension of the projection is limited by the number of clusters.

Figure B.5 illustrates the difference between the directions found by discriminant analysis and principal components analysis on a simple problem. Figure B.6 shows histograms of the resulting projections. In this example, the discriminant analysis projection separates the clusters well and the principal components projection shows cluster overlap.

Remarks A basic assumption in this analysis is that the clusters are roundish blobs described entirely by their mean and covariance structure, that is, Gaussian clouds. Optimal projections will not necessarily be found for more complex shapes. Sometimes a class consists of several distinct clusters; the compound cluster does not have the required structure so nonoptimal projections will probably result. One remedy is to present the algorithm with cluster labels instead of class labels, but this, of course, requires knowledge of the cluster structure.

Like PCA, discriminant analysis can be used to reduce the dimensionality of the data presented to a network. This reduces network size and may speed up training significantly if necessary information is not lost. Discriminant analysis is used for this purpose in [309].

Like PCA, discriminant analysis is a linear projection method so it can fail where nonlinear transformations are necessary. Neither will be able to separate classes that are not

linearly separable, for example, but both techniques remain useful for neural network design in cases where the linear projection makes sense. Although discriminant analysis is a little more complex, it makes use of the class information and therefore gives better separation than principal components in certain cases. When the data have linear dependencies, both techniques can be useful for preprocessing.

Just as the linear autoencoder implements a form of PCA, it has been shown [385, 134] that a single-hidden-layer *linear* network trained to perform classification with a 1-of-N target representation implements a form of discriminant analysis (figure B.7).