6 Dimensionality Reduction

The complexity of any classifier or regressor depends on the number of inputs. This determines both the time and space complexity and the necessary number of training examples to train such a classifier or regressor. In this chapter, we discuss various methods for decreasing input dimensionality without losing accuracy.

6.1 Introduction

In an application, whether it is classification or regression, observational data that we believe contain information are taken as inputs and fed to the system for decision making. Ideally, we should not need feature selection or extraction as a separate process; the classifier (or regressor) should be able to use whichever features are necessary, discarding the irrelevant. However, there are several reasons why we are interested in reducing dimensionality as a separate preprocessing step:

- In most learning algorithms, the complexity depends on the number of input dimensions, $d$, as well as on the size of the data sample, $N$, and for reduced memory and computation, we are interested in reducing the dimensionality of the problem. Decreasing $d$ also decreases the complexity of the inference algorithm during testing.

- When an input is decided to be unnecessary, we save the cost of extracting it.

- Simpler models are more robust on small datasets. Simpler models have less variance, that is, they vary less depending on the particulars of a sample, including noise, outliers, and so forth.
- When data can be explained with fewer features, we get a better idea about the process that underlies the data, which allows knowledge extraction.

- When data can be represented in a few dimensions without loss of information, it can be plotted and analyzed visually for structure and outliers.

There are two main methods for reducing dimensionality: feature selection and feature extraction. In feature selection, we are interested in finding $k$ of the $d$ dimensions that give us the most information and we discard the other $(d - k)$ dimensions. We are going to discuss subset selection as a feature selection method.

In feature extraction, we are interested in finding a new set of $k$ dimensions that are the combination of the original $d$ dimensions. These methods may be supervised or unsupervised depending on whether or not they use the output information. The best known and most widely used feature extraction methods are Principal Components Analysis (PCA) and Linear Discriminant Analysis (LDA), which are both linear projection methods, unsupervised and supervised respectively. PCA bears much similarity to two other unsupervised linear projection methods, which we also discuss—namely, Factor Analysis (FA) and Multidimensional Scaling (MDS).

### 6.2 Subset Selection

In subset selection, we are interested in finding the best subset of the set of features. The best subset contains the least number of dimensions that most contribute to accuracy. We discard the remaining, unimportant dimensions. Using a suitable error function, this can be used in both regression and classification problems. There are $2^d$ possible subsets of $d$ variables, but we cannot test for all of them unless $d$ is small and we employ heuristics to get a reasonable (but not optimal) solution in reasonable (polynomial) time.

There are two approaches: In forward selection, we start with no variables and add them one by one, at each step adding the one that decreases the error the most, until any further addition does not decrease the error (or decreases it only slightly). In backward selection, we start with all variables and remove them one by one, at each step removing
6.2 Subset Selection

the one that decreases the error the most (or increases it only slightly),
until any further removal increases the error significantly. In either case,
checking the error should be done on a validation set distinct from the
training set because we want to test the generalization accuracy. With
more features, generally we have lower training error, but not necessarily
lower validation error.

Let us denote by \( F \), a feature set of input dimensions, \( x_i, i = 1, \ldots, d \).
\( E(F) \) denotes the error incurred on the validation sample when only the
inputs in \( F \) are used. Depending on the application, the error is either the
mean square error or misclassification error.

In sequential forward selection, we start with no features: \( F = \emptyset \). At
each step, for all possible \( x_i \), we train our model and calculate \( E(F \cup x_i) \)
on the validation set. Then, we choose that input \( x_j \) that causes the least
error

\[
(6.1) \quad j = \arg \min_i E(F \cup x_i)
\]

and we

\[
(6.2) \quad \text{add } x_j \text{ to } F \text{ if } E(F \cup x_j) < E(F)
\]

We stop if adding any feature does not decrease \( E \). We may even
decide to stop earlier if the decrease in error is too small, where there is a
user-defined threshold that depends on the application constraints, trading
off the importance of error and complexity. Adding another feature
introduces the cost of observing the feature, as well as making the clas-
sifier/regressor more complex.

This process may be costly because to decrease the dimensions from \( d \)
to \( k \), we need to train and test the system \( d + (d-1) + (d-2) + \cdots + (d-k) \)
times, which is \( O(d^2) \). This is a local search procedure and does not
guarantee finding the optimal subset, namely, the minimal subset causing
the smallest error. For example, \( x_i \) and \( x_j \) by themselves may not be good
but together may decrease the error a lot, but because this algorithm is
greedy and adds attributes one by one, it may not be able to detect this.
It is possible to generalize and add \( m \) features at a time, instead of one,
at the expense of more computation. We can also backtrack and check
which previously added feature can be removed after a current addition,
thereby increasing the search space but this increases complexity. In
floating search methods (Pudil, Novovičová, and Kittler 1994), the number
of added features and removed features can also change at each step.
In **sequential backward selection**, we start with $F$ containing all features and do a similar process except that we remove one attribute from $F$ as opposed to adding to it, and we remove the one that causes the least error

\[(6.3) \quad j = \arg \min_i E(F - x_i)\]

and we

\[(6.4) \quad \text{remove } x_j \text{ from } F \text{ if } E(F - x_j) < E(F)\]

We stop if removing a feature does not decrease the error. To decrease complexity, we may decide to remove a feature if its removal causes only a slight increase in error.

All the variants possible for forward search are also possible for backward search. The complexity of backward search has the same order of complexity as forward search except that training a system with more features is more costly than training a system with fewer features, and forward search may be preferable especially if we expect many useless features.

Subset selection is supervised in that outputs are used by the regressor or classifier to calculate the error, but it can be used with any regression or classification method. In the particular case of multivariate normals for classification, remember that if the original $d$-dimensional class densities are multivariate normal, then any subset is also multivariate normal and parametric classification can still be used with the advantage of $k \times k$ covariance matrices instead of $d \times d$.

In an application like face recognition, feature selection is not a good method for dimensionality reduction because individual pixels by themselves do not carry much discriminative information; it is the combination of values of several pixels together that carry information about the face identity. This is done by feature extraction methods which we discuss next.

### 6.3 Principal Components Analysis

In projection methods, we are interested in finding a mapping from the inputs in the original $d$-dimensional space to a new $(k < d)$-dimensional
space, with minimum loss of information. The projection of \( x \) on the
direction of \( w \) is

\[
(6.5) \quad z = w^T x
\]

**Principal components analysis** (PCA) is an unsupervised method in that
it does not use the output information; the criterion to be maximized is
the variance. The principal component is \( w_1 \) such that the sample, after
projection on to \( w_1 \), is most spread out so that the difference between
the sample points becomes most apparent. For a unique solution and to
make the direction the important factor, we require \( \|w_1\| = 1 \). We know
from equation 5.14 that if \( z_1 = w_1^T x \) with \( \text{Cov}(x) = \Sigma \), then

\[
\text{Var}(z_1) = w_1^T \Sigma w_1
\]

We seek \( w_1 \) such that \( \text{Var}(z_1) \) is maximized subject to the constraint
that \( w_1^T w_1 = 1 \). Writing this as a Lagrange problem, we have

\[
(6.6) \quad \max_{w_1} w_1^T \Sigma w_1 - \alpha (w_1^T w_1 - 1)
\]

Taking the derivative with respect to \( w_1 \) and setting it equal to 0, we have

\[
2 \Sigma w_1 - 2 \alpha w_1 = 0, \quad \text{and therefore} \quad \Sigma w_1 = \alpha w_1
\]

which holds if \( w_1 \) is an eigenvector of \( \Sigma \) and \( \alpha \) the corresponding eigen-
value. Because we have

\[
w_1^T \Sigma w_1 = \alpha w_1^T w_1 = \alpha
\]

we choose the eigenvector with the largest eigenvalue for the variance
to be maximum. Therefore the principal component is the eigenvector
of the covariance matrix of the input sample with the largest eigenvalue,
\( \lambda_1 = \alpha \).

The second principal component, \( w_2 \), should also maximize variance,
be of unit length, and be orthogonal to \( w_1 \). This latter requirement is so
that after projection \( z_2 = w_2^T x \) is uncorrelated with \( z_1 \). For the second
principal component, we have

\[
(6.7) \quad \max_{w_2} w_2^T \Sigma w_2 - \alpha (w_2^T w_2 - 1) - \beta (w_2^T w_1 - 0)
\]

Taking the derivative with respect to \( w_2 \) and setting it equal to 0, we have

\[
(6.8) \quad 2 \Sigma w_2 - 2 \alpha w_2 - \beta w_1 = 0
\]
Premultiply by $w_2^T$ and we get

$$2w_2^T \Sigma w_2 - 2\alpha w_2^T w_2 - \beta w_2^T w_1 = 0$$

Note that $w_1^T w_2 = 0$. $w_2^T \Sigma w_2$ is a scalar, equal to its transpose $w_2^T \Sigma w_1$ where, because $w_1$ is the leading eigenvector of $\Sigma$, $\Sigma w_1 = \lambda_1 w_1$. Therefore

$$w_2^T \Sigma w_2 = w_2^T \Sigma w_1 = \lambda_1 w_2^T w_1 = 0$$

Then $\beta = 0$ and equation 6.8 reduces to

$$\Sigma w_2 = \alpha w_2$$

which implies that $w_2$ should be the eigenvector of $\Sigma$ with the second largest eigenvalue, $\lambda_2 = \alpha$. Similarly, we can show that the other dimensions are given by the eigenvectors with decreasing eigenvalues.

Because $\Sigma$ is symmetric, for two different eigenvalues, the eigenvectors are orthogonal. If $\Sigma$ is positive definite ($x^T \Sigma x > 0$, for all nonnull $x$), then all its eigenvalues are positive. If $\Sigma$ is singular, then its rank, the effective dimensionality, is $k$ with $k < d$ and $\lambda_i, i = k + 1, \ldots, d$ are 0 ($\lambda_1$ are sorted in descending order). The $k$ eigenvectors with nonzero eigenvalues are the dimensions of the reduced space. The first eigenvector (the one with the largest eigenvalue), $w_1$, namely, the principal component, explains the largest part of the variance; the second explains the second largest; and so on.

We define

$$(6.9) \quad z = W^T(x - m)$$

where the $k$ columns of $W$ are the $k$ leading eigenvectors of $S$, the estimator to $\Sigma$. We subtract the sample mean $m$ from $x$ before projection to center the data on the origin. After this linear transformation, we get to a $k$-dimensional space whose dimensions are the eigenvectors, and the variances over these new dimensions are equal to the eigenvalues (see figure 6.1). To normalize variances, we can divide by the square roots of the eigenvalues.

Let us see another derivation: We want to find a matrix $W$ such that when we have $z = W^T x$ (assume without loss of generality that $x$ are already centered), we will get $\text{Cov}(z) = D'$ where $D'$ is any diagonal matrix, that is, we would like to get uncorrelated $z_i$. 
6.3 Principal Components Analysis

If we form a \((d \times d)\) matrix \(C\) whose \(i\)th column is the normalized eigenvector \(c_i\) of \(S\), then \(C^T C = I\) and

\[
S = S C C^T \\
= S(c_1, c_2, \ldots, c_d) C^T \\
= (Sc_1, Sc_2, \ldots, Sc_d) C^T \\
= (\lambda_1 c_1, \lambda_2 c_2, \ldots, \lambda_d c_d) C^T \\
= \lambda_1 c_1 c_1^T + \cdots + \lambda_d c_d c_d^T \\
= CDC^T
\]

where \(D\) is a diagonal matrix whose diagonal elements are the eigenvalues, \(\lambda_1, \ldots, \lambda_d\). This is called the spectral decomposition of \(S\). Since \(C\) is orthogonal and \(C C^T = C^T C = I\), we can multiply on the left by \(C^T\) and on the right by \(C\) to obtain

\[
C^T S C = D
\]

We know that if \(z = W^T x\), then \(\text{Cov}(z) = W^T S W\), which we would like to be equal to a diagonal matrix. Then from equation 6.11, we see that we can set \(W = C\).

Let us see an example to get some intuition (Rencher 1995): Assume we are given a class of students with grades on five courses and we want to order these students. That is, we want to project the data onto one
dimension, such that the difference between the data points become most apparent. We can use PCA. The eigenvector with the highest eigenvalue is the direction that has the highest variance, that is, the direction on which the students are most spread out. This works better than taking the average because we take into account correlations and differences in variances.

In practice even if all eigenvalues are greater than zero, if $|S|$ is small, remembering that $|S| = \prod_{i=1}^{d} \lambda_i$, we understand that some eigenvalues have little contribution to variance and may be discarded. Then, we take into account the leading $k$ components that explain more than, for example, 90 percent, of the variance. When $\lambda_i$ are sorted in descending order, the proportion of variance explained by the $k$ principal components is

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

If the dimensions are highly correlated, there will be a small number of eigenvectors with large eigenvalues and $k$ will be much smaller than $d$ and a large reduction in dimensionality may be attained. This is typically the case in many image and speech processing tasks where nearby inputs (in space or time) are highly correlated. If the dimensions are not correlated, $k$ will be as large as $d$ and there is no gain through PCA.

Scree graph is the plot of variance explained as a function of the number of eigenvectors kept (see figure 6.2). By visually analyzing it, one can also decide on $k$. At the "elbow," adding another eigenvector does not significantly increase the variance explained.

Another possibility is to ignore the eigenvectors whose eigenvalues are less than the average input variance. Given that $\sum_i \lambda_i = \sum_i s_i^2$ (equal to the trace of $S$, denoted as $\text{tr}(S)$), the average eigenvalue is equal to the average input variance. When we keep only the eigenvectors with eigenvalues greater than the average eigenvalue, we keep only those which have variance higher than the average input variance.

If the variances of the original $x_i$ dimensions vary considerably, they affect the direction of the principal components more than the correlations, so a common procedure is to preprocess the data so that each dimension has 0 mean and unit variance, before using PCA. Or, one may use the eigenvectors of the correlation matrix, $R$, instead of the covariance matrix, $S$, for the correlations to be effective and not the individual variances.
6.3 Principal Components Analysis

Figure 6.2: (a) Scree graph. (b) Proportion of variance explained is given for the Optdigits dataset from the UCI Repository. This is a handwritten digit dataset with ten classes and sixty-four dimensional inputs. The first twenty eigenvectors explain 90 percent of the variance.

PCA explains variance and is sensitive to outliers: A few points distant from the center would have a large effect on the variances and thus the eigenvectors. Robust estimation methods allow calculating parameters in the presence of outliers. A simple method is to calculate the Mahalanobis distance of the data points, discarding the isolated data points that are far away.

If the first two principal components explain a large percentage of the variance, we can do visual analysis: We can plot the data in this two-dimensional space (figure 6.3) and search visually for structure, groups, outliers, normality, and so forth. This plot gives a better pictorial description of the sample than a plot of any two of the original variables.
By looking at the dimensions of the principal components, we can also try to recover meaningful underlying variables that describe the data. For example, in image applications where the inputs are images, the eigenvectors can also be displayed as images and can be seen as templates for important features; they are typically named "eigenfaces," "eigendigits," and so forth (Turk and Pentland 1991).

When \( d \) is large, calculating, storing, and processing \( S \) may be tedious. It is possible to calculate the eigenvectors and eigenvalues directly from data without explicitly calculating the covariance matrix (Chatfield and Collins 1980).

We know from equation 5.15 that if \( x \sim \mathcal{N}_d(\mu, \Sigma) \), then after projection \( W^T x \sim \mathcal{N}_k(W^T \mu, W^T \Sigma W) \). If the sample contains \( d \)-variate normals, then
6.3 Principal Components Analysis

it projects to $k$-variate normals allowing us to do parametric discrimination in this hopefully much lower dimensional space. Because $z_j$ are uncorrelated, the new covariance matrices will be diagonal, and if they are normalized to have unit variance, Euclidean distance can be used in this new space, leading to a simple classifier.

Instance $x^t$ is projected to the $z$-space as

$$z^t = W^T(x^t - \mu)$$

When $W$ is an orthogonal matrix such that $WW^T = I$, it can be backprojected to the original space as

$$\hat{x}^t = Wz^t + \mu$$

$\hat{x}^t$ is the reconstruction of $x^t$ from its representation in the $z$-space.

It is known that among all orthogonal linear projections, PCA minimizes the reconstruction error, which is the distance between the instance and its reconstruction from the lower dimensional space:

$$\sum_t \| \hat{x}^t - x^t \|^2$$

The reconstruction error depends on how many of the leading components are taken into account. In a visual recognition application—for example, face recognition—displaying $\hat{x}^t$ allows a visual check for information loss during PCA.

PCA is unsupervised and does not use output information. It is a one-group procedure. However, in the case of classification, there are multiple groups. Karhunen-Loève expansion allows using class information; for example, instead of using the covariance matrix of the whole sample, we can estimate separate class covariance matrices, take their average (weighted by the priors) as the covariance matrix, and use its eigenvectors.

In common principal components (Flury 1988), we assume that the principal components are the same for each class whereas the variances of these components differ for different classes:

$$S_t = C D_t C^T$$

This allows pooling data and is a regularization method whose complexity is less than that of a common covariance matrix for all classes, while still allowing differentiation of $S_t$. A related approach is flexible discriminant analysis (Hastie, Tibshirani, and Buja 1994), which does a linear projection to a lower-dimensional space where all features are uncorrelated and then uses a minimum distance classifier.