# Dimensionality Reduction for Machine Learning

Neil D. Lawrence

June 26, 2012

# Contents

# Notation and Symbols

## General Comments

Any background on notational choice here.

### 0.0.1 Reading Notation

The use of the design matrix convention means that the sample covariance matrix is given as $\mathbf{S} = n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$.

$$\mathrm{cov}\left(\mathbf{Y}\right) = \frac{1}{n}\sum_{i=1}^{n}\hat{\mathbf{y}}_{i,:}\hat{\mathbf{y}}_{i,:}^\top = n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$$

whilst the centered inner product matrix is given by $\mathbf{K} = \hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$

$$\mathbf{K} = \left(k_{i,j}\right)_{i,j}, \qquad k_{i,j} = \hat{\mathbf{y}}_{i,:}^\top\hat{\mathbf{y}}_{j,:}$$

**General**

| | |
|---|---|
| $q$ | dimension of latent/embedded space |
| $p$ | dimension of data space |
| $n$ | number of data points |
| $\mathbf{Y} = [\mathbf{y}_{1,:}, \ldots, \mathbf{y}_{n,:}]^\top = [\mathbf{y}_{:,1}, \ldots, \mathbf{y}_{:,p}] \in \Re^{n \times p}$ | data matrix |
| $\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_{1,:}, \ldots, \hat{\mathbf{y}}_{n,:}]^\top = [\hat{\mathbf{y}}_{:,1}, \ldots, \hat{\mathbf{y}}_{:,p}] \in \Re^{n \times p}$ | *centered* data matrix |
| $\mathbf{X} = [\mathbf{x}_{1,:}, \ldots, \mathbf{x}_{n,:}]^\top = [\mathbf{x}_{:,1}, \ldots, \mathbf{x}_{:,q}] \in \Re^{n \times q}$ | latent variables |
| $\mathbf{W} \in \Re^{p \times q}$ | mapping matrix |
| $\mathbf{H} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}} \in \Re^{n \times n}$ | centering matrix |
| $\sigma$ | standard deviation of a data set |
| $\ell$ | length scale of a kernel matrix |

**Vectors, Matrices and Norms**

| | |
|---|---|
| $\mathbf{1}$ | vector with all entries equal to one |
| $\mathbf{0}$ | vector with all entries equal to zero |
| $\mathbf{I}$ | identity matrix |
| $\mathbf{A}^\top$ | transposed matrix (or vector) |
| $\mathbf{A}^{-1}$ | inverse matrix (in some cases, pseudo-inverse) |
| $\mathrm{tr}\,(A)$ | trace of a matrix |
| $\lvert A \rvert$ | determinant of a matrix |
| $\lVert \cdot \rVert$ | 2-norm, $\lVert \mathbf{x} \rVert := \sqrt{\mathbf{x}^\top \mathbf{x}}$ |
| $\mathbf{a}_{i,:}$ | a column vector from the $i$th row of a given matrix $\mathbf{A}$ |
| $\mathbf{a}_{:,j}$ | a column vector from the $j$th column of a given matrix $\mathbf{A}$ |

**Probability**

| | |
|---|---|
| $P(C)$ | probability of an event $C$ |
| $p(x)$ | density evaluated at $x$ |
| $p(x\|y)$ | density evaluated at $x$ conditioned on $y$ |
| $q(x)$ | approximating distribution (often variational) |
| $\langle \cdot \rangle$ | expectation of a random variable |
| $\langle \cdot \rangle_{p(\cdot)}$ | expectation of a random variable under the density $p(\cdot)$ |
| $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ | multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\Sigma$ |
| $\mathcal{N}(\mathbf{z}\|\boldsymbol{\mu}, \Sigma)$ | multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\Sigma$, defined over $\mathbf{z}$ |
| $\mathcal{G}(a, b)$ | gamma distribution with shape parameter $a$ and rate parameter $b$ |
| $\mathcal{G}(z\|a, b)$ | gamma distribution with shape parameter $a$ and rate parameter $b$, defined over $z$ |
| $\chi^2_1(x)$ | Chi-squared density with one degree of freedom. |

**Miscellaneous**

| | |
|---|---|
| $\Re$ | the set of real numbers |
| $\log$ | logarithm to base $e$ |
| $\log_2$ | logarithm to base 2 |
| $\delta_{ij}$ | Kronecker $\delta$ ($\delta_{ij} = 1$ if $i = j, 0$ otherwise) |
| $\delta_x$ | Dirac $\delta_x$ satisfying $\int \delta_x(y) f(y) dy = f(x)$ |

Chapter 1

# Thinking in High Dimensions

To make full use of the computing power available to us today we need to interface the real world to the computer. To represent a real world object for processing by a computer, we are typically required to summarize that object by a series of *features* represented by numbers. As a representation of an object becomes more complex, the number of features required typically increases. Examples of objects we might want to process in this way include:

**a customer in a data base,** where the features might include their purchase history, where they live, their sex, and age;

**a digitized photograph,** where the features include the pixel intensities, time, date, and location of the photograph;

**human motion capture data for the movie and games industries,** where features consist of a time series of angles at each joint;

**human speech,** where the features consist of the energy at different frequencies (or across the cepstrum) as a time series;

**a webpage or other document,** where the features may consist of frequencies of given words as they appear in a set of documents and linkage information between documents;

**gene expression data,** where the features consist of the level of expression of thousands of genes, perhaps across a time series, or under different disease conditions.

As the complexity of the representation increases, a more accurate representation of the underlying object is realized. However, simultaneously we need to deal with more and more features. Data of this type is known as high dimensional data.

Data is the raw material of machine learning, models are the tools with which we can process the raw material to make interpretations of the data: for example categorization of a document's subject, transcription of spoken language, or classification of disease. Matching the right model to the data is a crucial first step in approaching a learning task. In this book we are interested in *dimensionality reduction.* This is the process of taking data with a lot of features, or dimensions, and summarizing it with a lower dimensional representation. It is not the only machine learning approach available, but it is often used either as a *preprocessing* step or as a way of visualizing a data set. In the first part of the book we will try and motivate why this might be a sensible approach to dealing with data. Our starting point will be to develop our understanding of *why* high dimensional data is special. It is only by understanding its characteristics that we can then really think about how to model it. In a successful application of machine learning the data and the model should work in partnership.

Machine learning is already a very technical subject, and is perhaps becoming more so. However, in this book we will try and rely more on intuitions and analogy, with technical material for back up. Reliance on intuition can be a dangerous thing. We are often fooled by our particular perspective on a problem: our intuitions can fail. This doesn't mean we should reject them. Rather, our approach will be to try and understand when and how they are wrong. However, we should always proceed with a little caution and bear in mind that "Proof by analogy is fraud" [Stroustrup, 2000, page 692].

A common approach to getting an intuition across is to use a "toy" data set to illustrate how an algorithm works. By focusing on toy data we can get a much better idea of a particular facet of a learning problem. Such toy data sets can be useful, giving intuitions about the learning scenario that generalize well to real world problems. However, such toy examples can also mislead. They can present an overly simple perspective on a particular learning problem. One of the approaches to understanding dimensionality reduction we will use in this book is to explore these toy problems and to use them to think about when, how and if they fail. Through better understanding of the data we hope to develop better models and ultimately improved performance of our learning algorithms.

## 1.1 Clustering

A recurring plot that is shown in talks on machine learning, particularly those on clustering, is a configuration of data points in two dimensions such as those

**boxfloat 1.1** Probability notation

n this book we will make extensive use of probabilistic modeling to represent data. In this box we give a short introduction to the probabilistic notation we use.

A probability distribution is defined over a set of a discrete number of possible outcomes for a variable, $S$. For example if we were considering a die roll we would have $S = 1, 2, 3, 4, 5, 6$. The distribution is a function that maps from an observed state for $S$ to a probability. If the observed state is defined to be $s$ then we write $P(S = s)$. The probability distribution is defined to be positive, and normalized, so that the sum across all possible states, $\sum_S P(S) = 1$.

A probability density is defined over a continuous space, $X$. For example, we might consider a density of the heights of computer science students. The value of a density at a given height is given by $p(X = x)$, which is defined to be a positive number. It is important to note though that this is *not* a probability. A probability density is defined to integrate to 1 over the range of values in $X$ which are valid,

$$\int_X p(X = x)\mathrm{d}x = 1.$$

We can interrogate the density about any range of values for $X$ for which the answer is a discrete outcome. We can express that answer in the form of a probability. For example, we define $S = 0, 1$ where 0 represents a height being less than 2m and 1 indicates it is greater or equal. We can recover a distribution from the density as follows

$$P(S = (X \geq 2)) = \int_2^\infty p(X = x)\mathrm{d}x.$$

In practice we normally use a shorthand notation for distributions and densities, we tend to drop the explicit notation of the event,

$$p(X = x) \equiv p(x).$$

This avoids "crowding" of the notation, but it can lead to confusion. We use this convention throughout this book. Until you are comfortable with it, you may find it useful to rewrite formulae with the full notation to remind yourself of their meaning.

shown in figure 1.7(a). At first glance, the data appears quite realistic. The density of the data points varies considerably as we move around the plot. The data seems to be clustered, but not in a uniform manner: some clusters are tighter than others. The clusters also seem to be somewhat randomly distributed around the plot. At some point during the talk, a slide containing a fit to the data, such as that in figure 1.7(b) is shown. This figure shows the means and variances of a mixture of multivariate (two dimensional) Gaussian densities [McLachlan and Basford, 1988]. The fit seems to be a good approximation to the data.

Models of this type can also be justified by appeals to our intuition. The way I like to think of these mixture models is as a summary of the data by a series of prototypes (represented by the means of the Gaussian components of the mixture density). These prototypes are subject to distortions. The density associated with each component represents the range of distortions that the data can undergo. In the case of the mixture of Gaussians, the distortions are of the form of adding zero mean, Gaussian distributed, random noise with a particular covariance matrix. ***How does the presence of prototype Gaussian densities affect the distance from the mean—how does it affect the density of interpoint squared distances. Need to cover this.***

Such a model seems intuitively appealing. When the *expectation maximization* approach to optimizing such powerful models was first described by Dempster et al. [1977] the statistics community it must have seemed to some that the main remaining challenge for density estimation was principally computational. There are, of course, applications for which for which a mixture of Gaussians is an appropriate model (*e.g.* low dimensional data). Such applications are not the focus of *this* book. We are more interested in the failings of the Gaussian mixture. There are two foundation assumptions which underpin this model. We described them both in our early appeal to intuition. The first assumption is that the data is generated through prototypes. We will not consider this assumption further. We will focus on the second assumption: how the prototypes are corrupted to obtain the data we observe. For the Gaussian mixture model the prototypes are corrupted by Gaussian noise with a particular covariance. It is this second assumption that we would like to consider first. In particular, we are going to examine the special case where the covariance is given by a constant diagonal matrix. We will see that the behavior of a Gaussian in low dimensional space can be quite misleading when we develop intuitions as to how these densities behave in higher dimensions. By higher dimensionality we can think of dimensionality that is difficult to visualize directly, *i.e.* dimensionality greater than $p = 3$.

### 1.1.1 Dimensionality Greater than Three

In higher dimensions *models* that seem reasonable in two or three dimensions can fail dramatically. Note the emphasis on models. In this section, we

**boxfloat 1.2** The Gaussian Density

he Gaussian density, or strictly speaking, the multi-variate Gaussian density will dominate our discussions in this book. The Gaussian density appeared, independently, as an approximation to the binomial de Moivre [1733] and as an approximation to the posterior density in a Bayesian analysis of biased coins Laplace [1774]. Both derived it by considering series expansions about the mode of a distribution of interest. The density is more commonly associated with Gauss due to his explicit use of it for modeling the "distribution of error" to justify least squares approaches Gauss.
The multivariate generalization of the Gaussian density takes the form

$$\mathcal{N}\left(\mathbf{y}|\boldsymbol{\mu},\mathbf{C}\right)=\frac{1}{\left(2\pi\right)^{\frac{p}{2}}\left|\mathbf{C}\right|^{\frac{1}{2}}}=\exp\left(-\frac{1}{2}\left(\mathbf{y}-\boldsymbol{\mu}\right)^{\top}\mathbf{C}^{-1}\left(\mathbf{y}-\boldsymbol{\mu}\right)\right),$$

where we refer to $\boldsymbol{\mu}$ as the mean and $\mathbf{C}$ as the covariance matrix. The notation $|\cdot|$ represents the determinant of a matrix. In the multivariate Gaussian it is part of the normalization term ensuring the density integrates to unity. Determinants often arise in association with volumes: see box 1.3 for the intuition behind this. Our density is defined over over $\mathbf{y}$ which is a $p$ dimensional vector, $\mathbf{y}\in\mathbb{R}^{p\times 1}$. The mean is the first moment of the density,

$$\boldsymbol{\mu}=\left\langle\mathbf{y}\right\rangle_{\mathcal{N}(\mathbf{y}|\boldsymbol{\mu},\mathbf{C})}=\int\mathbf{y}\mathcal{N}\left(\mathbf{y}|\boldsymbol{\mu},\mathbf{C}\right)\mathrm{d}\mathbf{y},$$

whilst the covariance gives the expected *squared deviation* about the mean,

$$\mathbf{C}=\left\langle\left(\mathbf{y}-\boldsymbol{\mu}\right)\left(\mathbf{y}-\boldsymbol{\mu}\right)^{\top}\right\rangle_{\mathcal{N}(\mathbf{y}|\boldsymbol{\mu},\mathbf{C})}=\int\left(\mathbf{y}-\boldsymbol{\mu}\right)\left(\mathbf{y}-\boldsymbol{\mu}\right)^{\top}\mathcal{N}\left(\mathbf{y}|\boldsymbol{\mu},\mathbf{C}\right)\mathrm{d}\mathbf{y}.$$

The covariance matrix is constrained to be positive definite.
The Gaussian density has some interesting properties, the sum of two Gaussian variables is also Gaussian. And, under certain conditions, the sum of very many non-Gaussian variables tends to be Gaussian. This is known as the central limit theorem, an early version of which was proposed by Laplace [1810], see box 1.5.
**Properties of the Gaussian Density**

- A multivariate Gaussian is fully specified by its mean vector and covariance matrix.
- All the marginal densities of a Gaussian are also Gaussian.
- All the conditional densities of a Gaussian are also Gaussian.
- Sum of two samples from multivariate Gaussians is also Gaussian distributed. The sum of the samples has a covariance given by the sum of the original covariances and a mean given by the sum of the original means.
- A Gaussian sample scaled by matrix $\mathbf{W}$ is also Gaussian distributed with covariance $\mathbf{W}\mathbf{C}\mathbf{W}^{\top}$ and mean $\mathbf{W}\boldsymbol{\mu}$.
- A Gaussian sample shifted by vector $\mathbf{m}$ is Gaussian distributed with mean $\boldsymbol{\mu}+\mathbf{m}$.
- ***Central limit theorem?***
- ***Marginalization property?***

**boxfloat 1.3** The Determinant of a Matrix

he determinant of a matrix will arise various times throughout this book. A full course in linear algebra is beyond the scope of this text but it is certainly worth attempting to give some intuitions about important concepts. The intuition we want to get across is the determinant as an assessment of volume.

The area of a square is its width multiplied by length. The volume of a cube is width multiplied by length multiplied by height. In general hyper cubes have a volume equal to the product of their dimensions. Lets represent each dimension of a hypercube by $\lambda_i$. The volume of a $p$-dimensional hypercube is then given by

$$V = \prod_{i=1}^{p} \lambda_i.$$

We can think of the vector $\boldsymbol{\lambda}$ as set of values on orthogonal (right angled) axes which describes the hypercube, we can define the coordinates of the vectors associated with this basis by introducing $\boldsymbol{\Lambda}$. The diagonal elements of $\boldsymbol{\Lambda}$ are given by $\boldsymbol{\Lambda}_{i,i} = \lambda_i$. The matrix thereby defines a set of $p$ axis aligned vectors, each one with a length of $\lambda_i$. The determinant of this matrix, $|\boldsymbol{\Lambda}|$ is the volume of the hypercube, $V = \prod_{i=1}^{p} \lambda_i = |\boldsymbol{\Lambda}|$. We can rotate this basis to rotate the cube using multiplication by a rotation matrix, $\mathbf{U}$, where $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ (i.e. it has orthonormal vectors). So we have a new basis for the cube, $\mathbf{W} = \mathbf{U}\boldsymbol{\Lambda}$. Pure rotation of the cube doesn't affect its volume, and the determinant of this matrix still gives the volume of the hypercube: $|\mathbf{W}| = \prod_{i=1}^{p} \lambda_i$.

As is required by the fact that it represents volumes the determinant is insensitive to pre-multiplication and post-multiplication rotation by an orthonormal basis. A common linear algebraic decomposition of a matrix, $\mathbf{W}$, is the singular value decomposition (SVD) (see box 2.12) which involves a further rotation of $\mathbf{W}$,

$$\mathbf{W} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{R}^\top$$

where $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$. The insensitivity to rotation means that $\left|\mathbf{U}\boldsymbol{\Lambda}\mathbf{R}^\top\right| = \prod_{i=1}^{p} \lambda_i$.

A slight problem with the above approach is that if an odd number of $\lambda_i$s are less than zero, then the volume will be returned as negative. This can be fixed by squaring and square-rooting the system. So we have $\sqrt{\left(\prod_{i=1}^{p} \lambda_i\right)^2 = \left(\prod_{i=1}^{p} \lambda_i^2\right)}$ and we the volume of the hypercube is given by $\left|\Lambda^2\right|^{\frac{1}{2}}$. Pre-multiplying and post-multiplying by $\mathbf{U}$ we can see the volume is also given by $\left|\mathbf{U}\Lambda^2\mathbf{U}^\top\right|^{\frac{1}{2}}$.

The form $\mathbf{U}\Lambda^2\mathbf{U}$ is *positive definite*, due to the squaring of $\Lambda$. Any valid covariance matrix can be written in this form, $\mathbf{C} = \mathbf{U}\Lambda^2\mathbf{U}$, so we see that for the multivariate Gaussian (box 1.2) $|\mathbf{C}|^{\frac{1}{2}}$ is the volume of the basis given by the diagonal of $\Lambda$. The factor of $(2\pi)^{p2}$ that appears in the normalization constant for the multivariate Gaussian deals with the fact that the volume is the integral under the exponentiated negative quadratic, not that of a hypercube.

are not making any statements about how a 'realistic' data set behaves in higher dimensions, we are making statements about modeling failures in higher dimensions. In particular we will focus on what assumptions the model makes about where the data is distributed in higher dimensions. We will illustrate the ideas through introducing the Gaussian egg. The Gaussian egg is a hard boiled egg that consists of three parts: the yolk of the egg, the white of the egg and a thin boundary shell between the yolk and the white. In an over boiled egg the boundary layer is often green from iron sulfide formed by reactions between the white and the yolk, giving a bad taste. We therefore refer to this as the "green" of the egg. We will consider spherical Gaussian distributions which have covariance matrices of the form $\sigma^2\mathbf{I}$. We define the volume of the density associated with the yolk as part of the egg that is within $0.95\sigma$ of the mean. The green is defined to be the region from $0.95\sigma$ to $1.05\sigma$. Finally, the yolk is all the density beyond $1.05\sigma$. We assume the density of the egg varies according to the Gaussian density, so that the density at the center of the egg is highest, and density falls off with the exponentiated negative squared Euclidean distance from the mean. We take the overall mass of our egg to be one. The question we now ask is how much of our egg's mass is now taken up by the three component parts: the green, the white and the yolk. The proportions of the mass are dependent on the dimensionality of our egg.

The answer is found through integrating over the Gaussian. For a one dimensional egg we can find the portion of the Gaussian that sits inside 0.95 of a standard deviation's distance from the mean as

$$\int_{-0.95\sigma}^{0.95\sigma} \mathcal{N}\left(y|0,\sigma^2\right) \mathrm{d}y.$$

The other portions can be found similarly. For higher dimensional Gaussians the integral is slightly more difficult. To compute it, we will switch from considering the Gaussian density that governs the data directly to the density over squared distances from the mean that the Gaussian implies. Before we introduce that approach, we show three low dimensional Gaussian eggs in figure 1.1–1.3 indicating their associated masses for the yolk, the green and the white.

### 1.1.2 Distribution of Mass against Dimensionality

For the three low dimensional Gaussians, we note that the allocation of the egg's mass to the three zones changes as the dimensionality increases. The green and the white increase in mass and the yolk decreases in mass. Of greater interest to us is the behavior of this distribution of mass in higher dimensions. It turns out we can compute this through the cumulative distribution function of the gamma density.

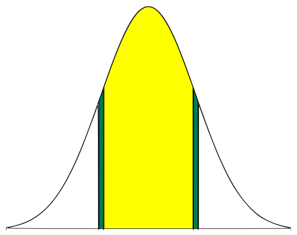


Figure 1.1: Volumes associated with the one dimensional Gaussian egg. Here the yolk has 65.8%, the green has 4.8% and the white has 29.4% of the mass.

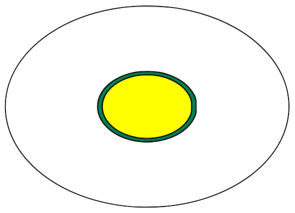


Figure 1.2: Volumes associated with the regions in the two dimensional Gaussian egg. The yolk contains 59.4%, the green contains 7.4% and the white 33.2%.

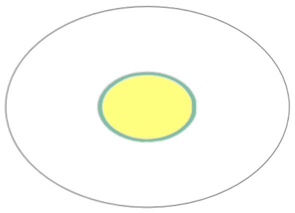


Figure 1.3: Volumes associated with the regions in the three dimensional Gaussian egg. Here the yolk has 56.1% the green has 9.2% the white has 34.7%.

We will compute the distribution of the density mass in the three regions by assuming each data point is sampled independently from our spherical covariance Gaussian density,

$$\mathbf{y}_{i,:} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$$

where $\mathbf{y}_{i,:}$ is the $i$th data point. Independence across features also means we can consider the density associated with the $k$th feature of the $i$th data point, $y_{i,k}$,

$$y_{i,k} \sim \mathcal{N}\left(0, \sigma^2\right).$$

We are interested in the squared distance of any given sample from the mean. Our choice of a zero mean Gaussian density means that the squared distance of each feature from the mean is easily computed as $y_{i,k}^2$. We can exploit a useful characteristic of the Gaussian to describe the density of these squared distances. The squares of a Gaussian distributed random variable are known to be distributed according the *chi-squared density*,

$$y_{i,k}^2 \sim \sigma^2\chi_1^2,$$

The chi squared density is itself a special case of the gamma density (see box 1.4) with shape parameter $a = \frac{1}{2}$ and rate parameter $b = \frac{1}{2\sigma^2}$,

$$\mathcal{G}\left(x|a,b\right) = \frac{b^a}{\Gamma\left(a\right)}x^{a-1}e^{-bx}.$$

So we have the squared distance from the mean for a single feature being given by

$$y_{i,k}^2 \sim \mathcal{G}\left(\frac{1}{2}, \frac{1}{2\sigma^2}\right).$$

Of course, we are interested in the distance from the mean of the data point given by $\mathbf{y}_{i,:} = \left[y_{i,1}, \dots, y_{i,p}\right]^\top$. The *squared* distance from the mean of the $i$th data point will be given by $\sum_{k=1}^p y_{i,k}^2$. Fortunately, the properties of the gamma density (see box 1.4) mean we can also compute the density of the resulting random variable, in particular we have

$$\sum_{k=1}^p y_{i,k}^2 \sim \mathcal{G}\left(\frac{p}{2}, \frac{1}{2\sigma^2}\right).$$

We can compute the mean and standard deviation of the squared distance for each point from the mean,

$$\left\langle \sum_{k=1}^p y_{i,k}^2 \right\rangle = p\sigma^2,$$

**boxfloat 1.4** The Gamma Density

The Gamma is a density over positive numbers. It has the form

$$\mathcal{G}\left(x|a,b\right) = \frac{b^a}{\Gamma\left(a\right)} x^{a-1} e^{-bx},$$

where $a$ is known as the shape parameter and $b$ is known as a rate parameter. The function $\Gamma\left(a\right)$ is known as the gamma function and is defined through the following indefinite integral,

$$\Gamma\left(a\right) = \int_0^\infty x^{a-1} e^{-x} \mathrm{d}x.$$

The mean of the gamma density is given by

$$\left\langle x\right\rangle_{\mathcal{G}(x|a,b)} = \frac{a}{b}$$

and the variance is given by

$$\mathrm{var}_{\mathcal{G}(x|a,b)}\left(x\right) = \frac{a}{b^2}.$$

Sometimes the density is defined in terms of a scale parameter, $\beta = b^{-1}$, instead of a rate. Confusingly, this parameter is also often denoted by "$b$". For example, the statistics toolbox in MATLAB defines things this way. The gamma density generalizes several important special cases including the exponential density with rate $b$, $\left\langle x\right\rangle_b$, which is the specific case where the shape parameter is taken to be $a = 1$. The chi-squared density with one degree of freedom, denoted $\chi_1^2\left(x\right)$, is the special case where the shape parameter is taken to be $a = \frac{1}{2}$ and the rate parameter is $b = \frac{1}{2}$.
The gamma density is the conjugate density for the inverse variance (precision) of a Gaussian density. See box 2.5 for more on conjugacy in Bayesian inference. Gamma random variables have the property that, if multiple gamma variates are sampled from a density with the same rate, $b$, and shape parameters $\left\{a_k\right\}_{k=1}^p$, then the sum of those variates is also Gamma distributed with rate parameter $b$ and shape parameter $a' = \sum_{k=1}^p a_k$.

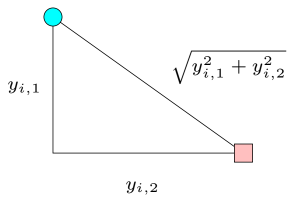


Figure 1.4: Distance from mean of the density (circle) to a given data point (square).

which, we note, scales linearly with the dimensionality of the data. The average squared distance of each feature will be distributed as follows,

$$\frac{1}{p}\sum_{k=1}^{p} y_{i,k}^2 \sim \mathcal{G}\left(\frac{p}{2}, \frac{p}{2\sigma^2}\right)$$

the mean for which is simply the variance of the underlying Gaussian density,

$$\left\langle \frac{1}{p}\sum_{k=1}^{p} y_{i,k}^2 \right\rangle = \sigma^2.$$

We can use this gamma density to work out how much of the mass of the Gaussian egg is in the different zones. The cumulative distribution function for the gamma density,

$$\mathcal{GAMMACDF}\left(z|a,b\right) = \int_0^x \mathcal{G}\left(z|a,b\right) \mathrm{d}z,$$

doesn't have a nice analytical form, but implementations of it are provided for many programming languages including R, Matlab, Octave and Python. We can use the cumulative distribution to give the probability of the squared distance falling under a certain value. For the data point to be in the yolk, we expect its squared distance from the mean to be under $(0.95\sigma)^2$. For the data point to be in the green we expect its squared distance from the mean to be between $(0.95/\sigma)^2$ and $(1.05/\sigma)^2$. Data in the white will have a squared distance from the mean greater than $(1.05/\sigma)^2$.

### 1.1.3 Looking at Gaussian Samples

The theory has shown us that data sampled from a Gaussian density in very high dimensions will live in a shell around one standard deviation from the mean of the Gaussian. This is perhaps surprising because we are used to thinking of Gaussians being distributions where most of the data is near the mean of the

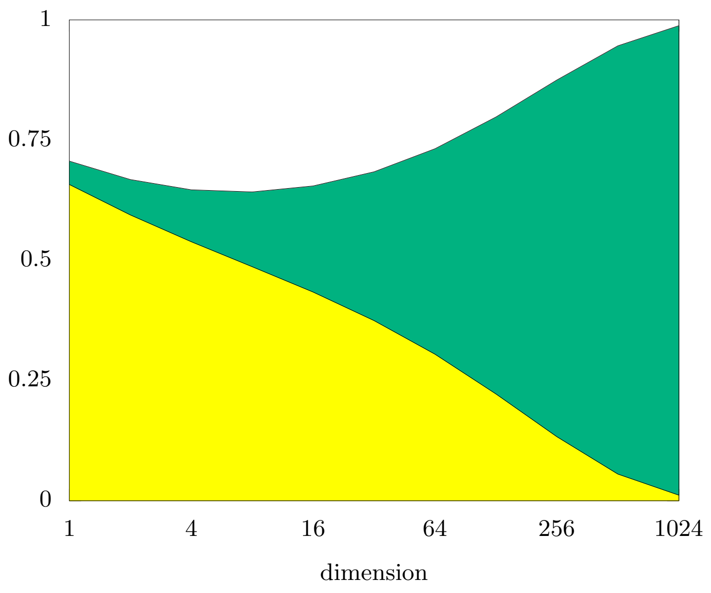


Figure 1.5: Plot of probability mass versus dimension. Plot shows the volume of density inside 0.95 of a standard deviation (yellow), between 0.95 and 1.05 standard deviations (green), over 1.05 and standard deviations (white).

density. But this is because we are used to looking at low dimensional Gaussian densities. In this regime most of the data is near the mean. One useful property of the Gaussian density is that all the marginal densities are also Gaussian. This means that when we see a two dimensional Gaussian we can think of it as a three dimensional Gaussian with one dimension marginalized. The effect of this marginalization is to project the third dimension down onto the other two by summing across it. This means when we see a two dimensional Gaussian on a page such as that in figure 1.6, we can think of a corresponding three dimensional Gaussian which has this two dimensional Gaussian as a marginal. That three dimensional Gaussian would have data going into and coming out of the page. The projection down to two dimensions makes that data look like it is close to the mean, but when we expand up to the three dimensional Gaussian some of that data moves away from the mean. Following this line of argument, we can ask what if the plot is the two dimensional marginal of a truly four dimensional Gaussian. Now some more of the data that appears close to the mean in the two dimensional Gaussian (and perhaps was close to the mean in the three dimensional Gaussian) is also projected away. We can continue applying this argument until we can be certain that there is no data left near the mean. So the gist of the argument is as follows. When you are looking at a low dimensional projection of a high dimensional Gaussian, it does appear that there is a large amount of data close to the mean. But, when we consider that the data near the mean has been projected down from potentially many dimensions we understand that in fact there is very little data near the mean.

### 1.1.4 High Dimensional Gaussians and Interpoint Distances

Our analysis above showed that for spherical Gaussians in high dimensions the density of squared distances from the mean collapses around the expected value of one standard deviation. Another related effect is the distribution of *interpoint* squared distances. It turns out that in very high dimensions, the interpoint squared distances become equal. This has knock on effects for many learning algorithms: for example in nearest neighbors classification algorithms a test data point is assigned a label based on the labels of the $k$ nearest neighbors from the training data. If all interpoint squared distances were close to equal then the stability of these $k$ neighbors with respect to small data perturbations would be poor.

***Something wrong here*** Can show this for Gaussians with a similar proof to the above,

$$y_{i,k} \sim \mathcal{N}\left(0, \sigma_k^2\right) \qquad y_{j,k} \sim \mathcal{N}\left(0, \sigma_k^2\right)$$

$$y_{i,k} - y_{j,k} \sim \mathcal{N}\left(0, 2\sigma_k^2\right)]$$

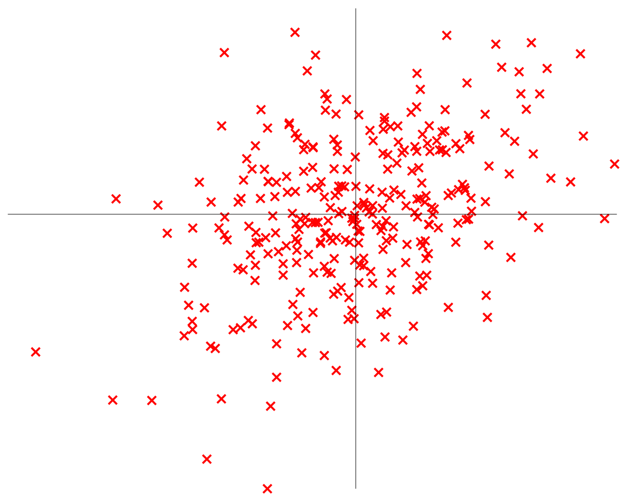


Figure 1.6: Looking at a projected Gaussian. This plot shows, in two dimensions, samples from a potentially very high dimensional Gaussian density. The mean of the Gaussian is at the origin. There appears to be a lot of data near the mean, but when we bear in mind that the original data was sampled from a much higher dimensional Gaussian we realize that the data has been projected down to the mean from those other dimensions that we are not visualizing.

$$(y_{i,k} - y_{j,k})^2 \sim \mathcal{G}\left(\frac{1}{2}, \frac{1}{4\sigma_k^2}\right)$$

Once again we can consider the specific case where the data is spherical, $\sigma_k^2 = \sigma^2$, as we can always individually rescale the data input dimensions to ensure this is the case

$$\sum_{k=1}^{p} (y_{i,k} - y_{j,k})^2 \sim \mathcal{G}\left(\frac{p}{2}, \frac{1}{4\sigma^2}\right)$$

$$\frac{1}{p}\sum_{k=1}^{p} (y_{i,k} - y_{j,k})^2 \sim \mathcal{G}\left(\frac{p}{2}, \frac{D}{4\sigma^2}\right)$$

The dimension normalized squared distance between points is Gamma distributed. Mean is $2\sigma^2$. Variance is $\frac{8\sigma^2}{D}$.

## 1.2 Central Limit Theorem and Non-Gaussian Case

So far we have entirely focused on data which is generated according to a Gaussian density. For data generated according to spherical Gaussian densities we can compute the corresponding densities of squared distances from the mean and interpoint squared distances *analytically*. However, our conclusions are not limited to this case. In general, for data where the features (the columns of $\mathbf{Y}$) are sampled independently the *central limit theorem* (see box 1.5) applies. ***Need to analyze in detail how the central limit theorem applies.***

- The mean squared distance in high dimensional space is the mean of the variances.
- The variance about the mean scales as $p^{-1}$.

### 1.2.1 Independent Features: Summary

So far, our analysis has focused on data sets which are assumed to have arisen through independently generated features. We have seen that a model which assumes that a high dimensional data set is made up by independently sampling each feature behaves rather counter intuitively. All the data falls exactly one standard deviation away from the mean. If we think of a mixture model representation of the data, where we justify the mixture representation by the assumption that our data is made up of prototypes (the mixture centres) which

**boxfloat 1.5** The Central Limit Theorem

he "central limit theorem" underpins much of statistics. Indeed it is named "central" due to its importance as a foundation of much of statistics. It states that the sum of a number of independent random variables, $\sum_{i=1}^{n} y_i$ , with finite variance, will, as the number of variables increases, appear to be drawn from Gaussian density. The mean of the Gaussian is given by the sum of the means of the independent variables, $\sum_{i=1}^{n} \mu_i$, and the variance is given as the sum of the variances, $\sum_{i=1}^{n} \sigma_i^2$ so we have

$$p\left(\sum_{i=1}^{n} y_i\right) \approx \mathcal{N}\left(\sum_{i=1}^{n} y_i | \sum_{i=1}^{n} \mu_i, \sum_{i=1}^{n} \sigma_i^2\right)$$

for large $n$.

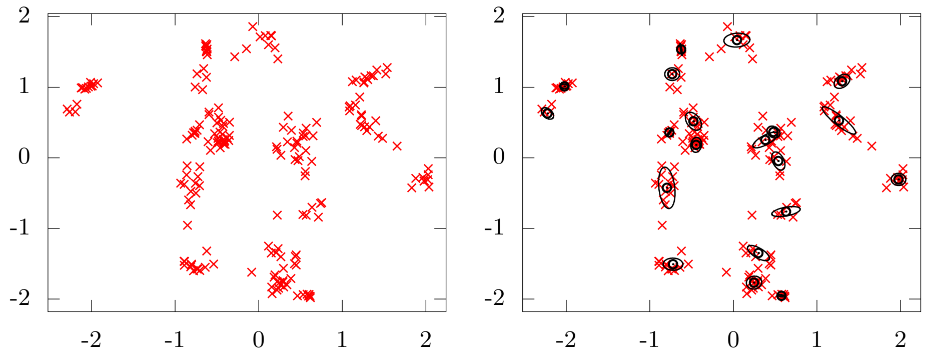


Figure 1.7: A two dimensional data set and the fit of a mixture of Gaussians to the data. A mixture of Gaussians appears to be a very powerful model as it seems to fit a data set with a fairly complex structure.

are corrupted in some way, we find that our prototypes have vanishingly small probability as dimensions increase.

For the Gaussian case the data are distributed uniformly across the hypersphere. If we were able to sit at the center of the Gaussian looking out, the view of the data would be like looking at the night sky, although all the stars would be one standard deviation away.

***expand on the central limit theorem approach.*** For the super Gaussian case the data would be clustered along the axes. For the sub Gaussian case the data would be clustered at points rotated 45° away from the axes.

## 1.3 The Curse of Dimensionality?

***Add in references to material about consistency of models which consider features to be independent ... in this case dimensionality is a blessing ... in the context of maximum likelihood it allows parameters to be well determined. This can be a way of introducing GPLVM and spectral methods. Question: for GPLVM the number of parameters is known, but for spectral methods, how many parameters are we using, if , for example we do LLE? k*N. This is odd, because as data we only have d*N data points and sometimes k>d.***

When I first started working in machine learning, the 'curse of dimensionality' was often raised. The theoretical problems of high dimensional spaces were well understood. When their implications are considered in the context of a given algorithm, it is often clear that the algorithm's behavior will deteriorate. For example, many algorithms are based on measuring the distance between points and assuming that points which are close together behave in a similar way. If all data points were approximately equidistant we would be unable to differentiate between different data points through this mechanism. Now we will explore the question of whether or not this is an accurate representation for real high dimensional data sets. We will start, though, by considering an artificial data set, one actually sampled independently across its features from Gaussian distributions.

## 1.4 Gaussian Samples in High Dimensions

As a sanity check we will first sample from a Gaussian distribution. We'll then histogram inter-point squared distances and plot them against the theoretical distribution we derived in the last chapter. We will take the squared distance

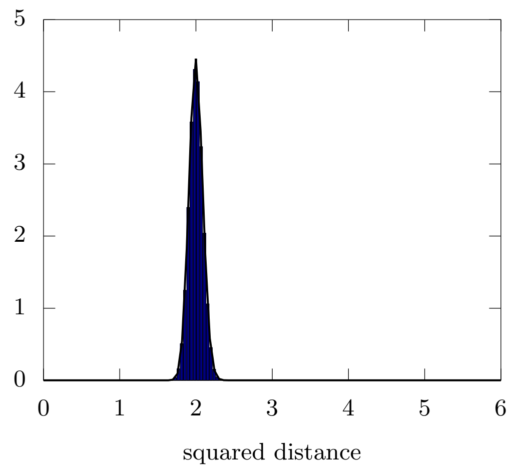


Figure 1.8: Histogram of inter-point squared distances between points sampled from the 1000 point, 1000 dimensional Gaussian. A good match betwen theory and the samples for a 1000 dimensional Gaussian distribution.

distribution for samples from a Gaussian distribution with 1000 features, $p = 1000$, and 1000 data points, $n = 1000$. Figure 1.8 shows the results from plotting the histogram of inter-point squared distances against the theoretical curve.

Having performed an empirical validation of the theoretical squared distance distribution on a known underlying density, we now introduce a series of real world data sets of differing characteristics. These data sets have been designed to reflect the diversity of data we might expect to encounter when using dimensionality reduction in practice. We will refer to them as the **oil data**, the **stick man data**, the **Spellman data**, the **grid speech data** and the **Netflix data**.

We now turn to a real data set, one that is commonly employed to demonstrate dimensionality reduction algorithms. The data set consists of simulated measurements from an oil pipeline. The model is as follows: oil, water and gas flow together through an oil pipeline. Depending on the relative proportions they either flow a

## 1.5 Oil Data

The "oil data" consists of 12 simulated measurements of a combination of oil, water, and gas flowing in a pipeline Bishop and James [1993]. The flow has three

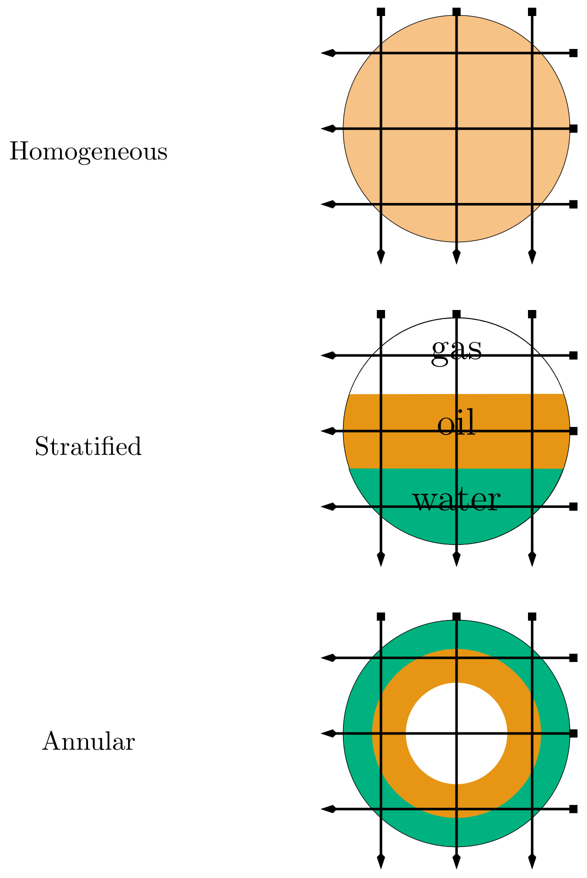


Figure 1.9: The “oil data”. The data set is artificially generated by modeling the manner in which a gamma ray’s intensity falls when it passes through a different density materials.

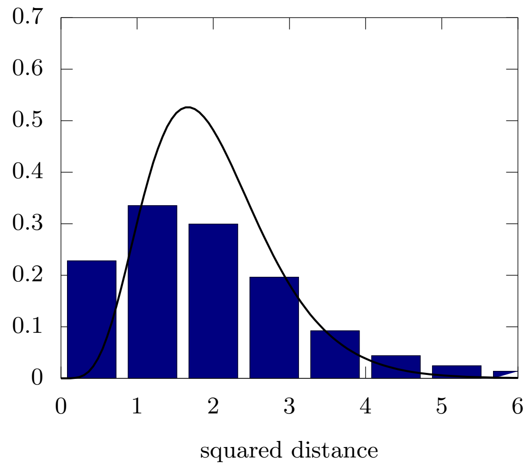


Figure 1.10: Interpoint squared distance distribution for oil data with $p = 12$ (variance of squared distances is 1.98 vs predicted 0.667).

different regimes: homogeneous, stratified, or annular. In the homogeneous regime the oil, water, and gas is distributed relatively uniformly in a cross section of the pipe. In the other two regimes it either flows in layers: water on the bottom, oil in the middle, and gas on top. Or it flows as annular rings of oil, water, and gas.

The flow regimes are simulated and the data consists of simulated measurements from gamma ray densitometry probes which measure the density between two points across the pipeline. There are 12 probes and they are arranged approximately as shown in figure 1.9. The measurement from each probe is taken to be a feature. The data consists of 1000 measurements in total for different proportions of oil, water, and gas.

The density of squared distances between the measurements is shown as a histogram in figure 1.10.

## 1.6 Stick Man Data

Motion capture rigs are commonly used in the games and movie industries for capturing realistic motion. The subject wears a special suit with reflective balls attached at key locations (typically the subject's joints). Several cameras are used to record the subject's motion and geometric techniques are used to reconstruct the location of each joint given the video recordings. High sampling rates are used (typically 120 frames per second) and missing markers are filled in through interpolation or and smoothing techniques.

The data set we have selected for analysis is from Ohio State University's Motion Capture Lab.[1] It consists of a man breaking into a run from a crouched position. As the run progresses, the subject becomes more upright: a video provided with the data shows that the runner is operating in an enclosed space, so it seems that towards the end of the run the runner's fairly extreme upright position is associated with the need to stop. We downsampled the data from the original 120 frames to 30 frames per second to reduce the amount of available data.

The raw motion capture data is recovered as a three dimensional point cloud: each joint is associated with an $x$, $y$ and $z$ position. By convention in graphics $y$ is taken to be the vertical axis and $x$ and $z$ represent the horizontal plane. Kinematic constraints are normally imposed on the data by the construction of an underlying skeleton and the conversion of the $x$, $y$, $z$ positions to a series of associated joint angles. This is important in animation as it prevents character configurations associated with physical impossible maneuvers (such as an extension of the leg). However, for our purposes, we will consider the raw point cloud data. This will ensure, when we generate data from the model, that we are not relying on the constraints imposed by the skeleton to generate realistic data. The subject in the data set consisted of 34 joints[2], leading to $p = 102$ dimensional data for each frame. The down-sampled data consisted of $n = 55$ frames. The data contains approximately three strides from the run and is therefore pseudo periodic (it is not truly periodic due to the changing angle of run).

Some frames from the data are visualized in figure 1.11. The histogram of interpoint squared distances for the "stick man" data is given in figure 1.12

## 1.7 Robot WiFi Data

This data consists of a time series of WiFi signal strength recordings from a robot as it traces a square path in a building. The robot records the strength of WiFi signals in an attempt to localize its position [see Ferris et al., 2007, for an application]. Since the robot moves only in two dimensions, the inherent dimensionality of the data should be two: the reduced dimensional space should reflect the robots moves. The WiFi signals are very noisy relative to (e.g.) motion capture data. The robot completes a single circuit after entering from a separate corridor, so it is expected to exhibit "loop closure" in the resulting map. The data consists of 215 frames of measurement, each frame consists of the WiFi signal strength of up to 30 access points.

[1]See `http://accad.osu.edu/research/mocap/mocap_data.htm`, data is "Figure Run 1".

[2]We obtained the data from `http://accad.osu.edu/research/mocap/mocap_data.htm`

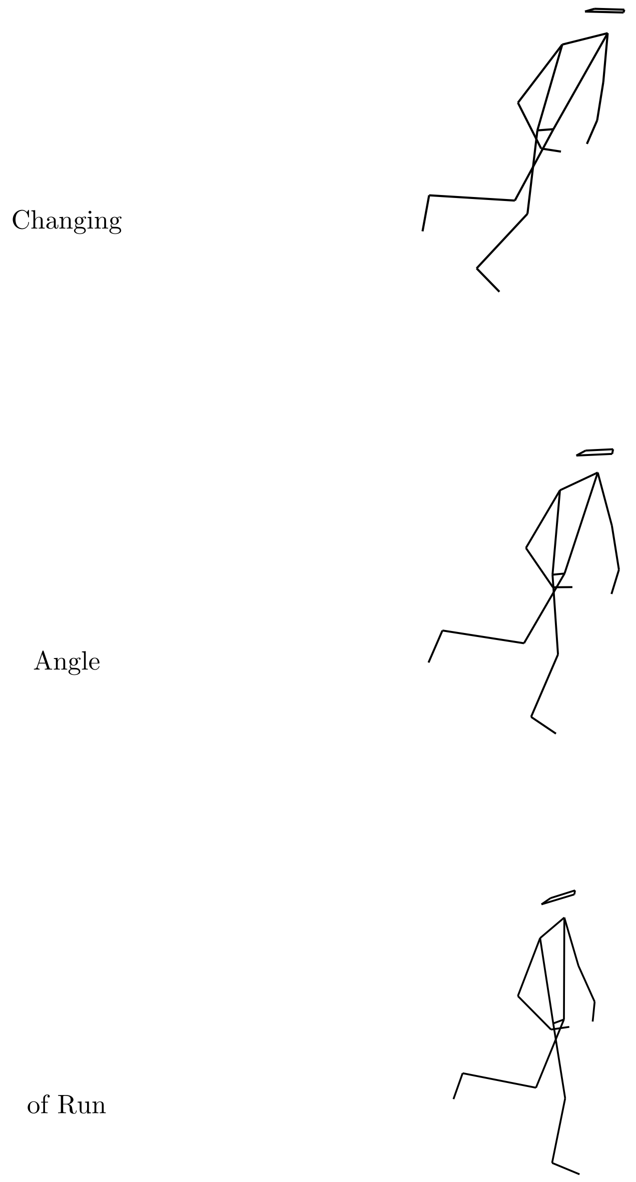


Figure 1.11: The “stick man” data set. We have illustrated frames that show the changing angle of run associated with the data.

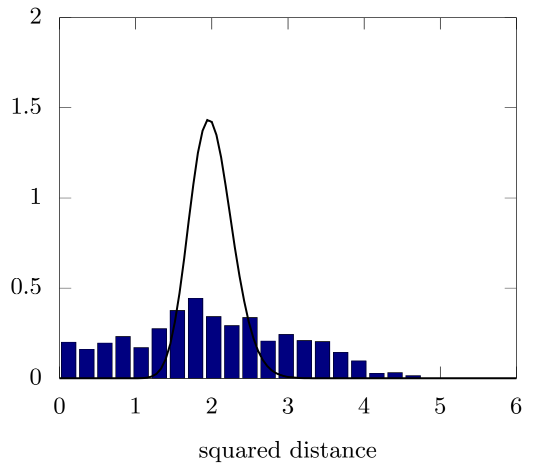


Figure 1.12: Interpoint squared distance distribution for stick man data with $p = 102$ (variance of squared distances is 1.09 vs predicted 0.0784).

## 1.8 Yeast Cell Cycle Data

Microarray measurements are designed to quantify the amount of messenger RNA (mRNA) being expressed by the genome at any given time. This is done by extracting RNA from a population of cells and reverse transcribing the mRNA to cDNA. This cDNA is fluorescently tagged and then bound to a set of probes on a slide. The slide is then scanned to measure the amount of fluorescence associated with each probe. This acts as a proxy for the amount of mRNA extraced from the biological sample. Some of these approaches, including many "spotted arrays" bind two samples with different excitation levels for the fluorescence process two samples at the same time. This allows the ratio of the fluorescence to be taken as a proxy for the ratio of expression in the original sample. One of the first publicly available DNA microarray data sets was published by Spellman et al. [1998] and concerns the yeast cell cycle. The arrays quantify the expression level of 24,721 different genes measured at 24 time points in a time series[3]

## 1.9 Grid Speech Data

The "grid corpus" of speech data was collected by the Speech and Hearing (SpandH) Group at the University of Sheffield. The objective was to acquire

[3]Data available from `http://genome-www.stanford.edu/cellcycle/data/rawdata/individual.htm`.

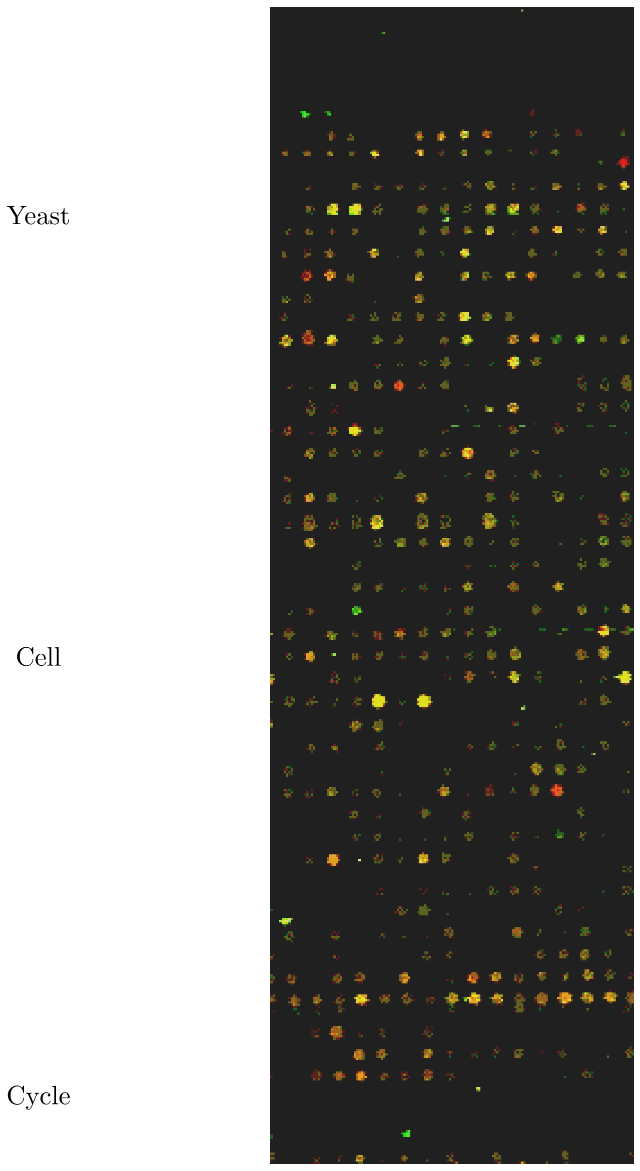


Figure 1.13: The “Spellman” data. The data consists of gene expression measurements from yeast to explore which genes are involved in the cell cycle.

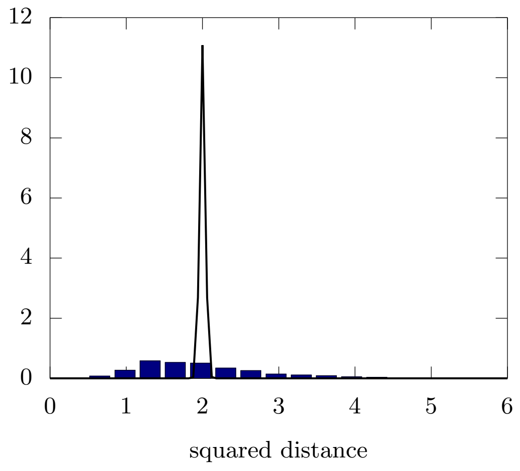


Figure 1.14: Interpoint squared distance distribution for Spellman microarray data with $p = 6178$ (variance of squared distances is 0.694 vs predicted 0.00129).

a speech and vision data set with a restricted language model under controlled conditions for the study of robust speech recognition. The data set consists of 34 subjects. Jon Barker of the Sheffield group has created a set of synthesis models for this data based on hidden Markov models (HMM). The raw data modeled is the subject's 25 dimensional spectrogram. The actual value of the spectrogram along with the "velocities" and "accelerations" are modeled for each frame. Thirty-six different phones are modeled. To partially deal with coarticulation affects several phones (for example iy, eh and ih) have multiple models. For our purposes we can consdier the HMM used to contain seven states: five states for each phone and a start and stop state. Each state has a single component Gaussian density associated with it. This density has a diagonal covariance (so each feature is modeled independently). Need to cite the guy who did this originally **?**. ***details of data size and source for speech synthesis***.

## 1.10 Netflix Data

The final data set we will consider is the Netflix movie recommender data. This data was released for the "Netflix Prize". The data consists of about 100 million ratings from over 400 thousand users for over 17 thousand potential DVD rentals. The ratings are between 1 and 5 stars. The matrix of ratings is highly sparse as a typical user may rate only a few hundred films. Prediction of the missing ratings using other users ratings is known as collaborative filtering.

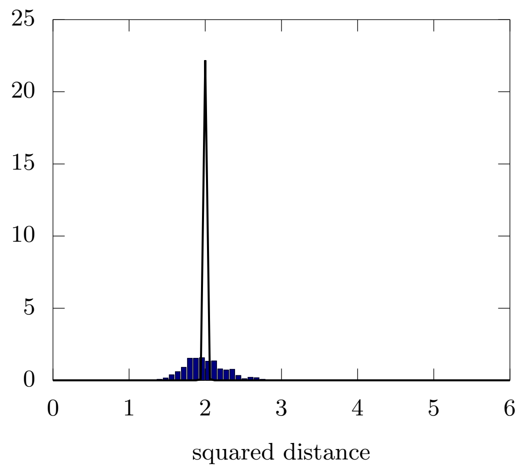


Figure 1.15: Interpoint squared distance distribution for Grid corpus vowel data with $p = 24750$ (variance of squared distances is 0.07 vs predicted 0.000323).

This type of prediction of user preference is a challenging problem for internet retailers who can raise their turnover by accurate prediction of user tastes. Whilst this book isn't about collaborative filtering, a sensible approach to this data is to try and reduce its dimensionality. The fact that the data set is large and has many missing elements also presents us with particular challenges that we might often expect to find in real world data.

***Maybe include a movielens data for exploration here?***

## 1.11 Another Look at Gaussian Samples

It is clear that for many data sets the squared distance density does not conform to the theoretical distribution that we described. Where does practice depart from our theory? In practice we find that the variance of the squared distance densities is much larger than we might expect. To explore how this occurs, let's consider another Gaussian density with $p = 1000$. Somehow the real data seems to have a smaller effective dimension. In figure 1.16 we show inter point squared distances computed from another 1000 dimensional Gaussian density. Here though, the covariance of the Gaussian density has a particular structure.

The theory no longer matches the practice for this histogram. Indeed, the for a two dimensional Gaussian has a much closer match to the empirically computed inter-point squared distances. The density we used exhibits correlations

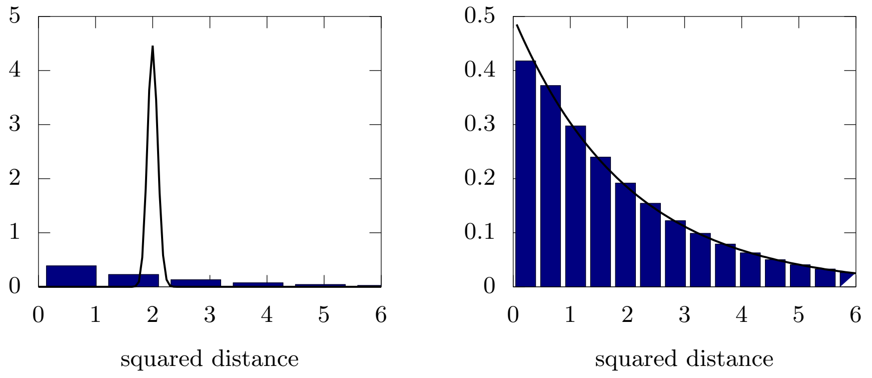


Figure 1.16: Interpoint squared distance density for Gaussian with $p = 1000$. The Gaussian density now has a specific low rank covariance matrix $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$. Data generated by taking $\sigma^2 = 1e-2$ and sample $\mathbf{W} \in \Re^{1000\times 2}$ elements independently from $\mathcal{N}(0,1)$. On the left the theoretical curve for 1,000 independent dimensions is imposed on the data, on the right the theoretical curve for 2 independent dimensions used. The fit is much better for 2 dimensions that 1,000.

between all data points, and these correlations are such that the underlying dimensionality of the Gaussian appears closer to two. Structured covariance Gaussians such as this one appear to be a more appropriate model for data than the independent Gaussian on which our analysis was based. In the next chapter we will explore a latent variable model explanation for this density. This explanation will reveal why this data is structured differently. We will see that whilst the apparent dimensionality of the data is a thousand, the underlying dimensionality is only two.

***Not sure if we need exchangeability here, if we do we need a box on it! Need to explain why the marginal is then the same ...***

# Chapter 2

# A Linear Model for Dimensionality Reduction

We've used the first chapter of the book to introduce some intuitions about what might happen in high dimensions, backing up some basic assumptions with theoretical analysis. For independently sampled features, we saw how data has the characteristic that all points end up at a fixed distance from the center of the density. For Gaussian densities we saw that all sampled points become uniformly distributed across a sphere, rather like a stars scattered uniformly across the sky: but with the distances to all those stars fixed and equal to the standard deviation of the Gaussian. For non-Gaussian densities data becomes clustered at points which are either axis aligned (for super Gaussian sources) or rotated 45 degrees from the axes. The distance between those clusters and the origin is the standard deviation of the generating density.

To explore this effect in a range of real data sets we measured the squared distances for a small cross section of different data. We saw that there was a serious mismatch between the theory we'd developed and the reality and the reality. As a sanity check, we considered features sampled independently from a spherical Gaussian density in high dimensions and saw that the theory worked well. We finished the by showing that for covariance matrices of a particular form,

$$\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$$

we could recover the mismatch.

In this chapter we will develop the probabilistic model behind this covariance matrix. This will allow us to introduce several concepts that will also prove useful in future derivations. Because the covariance matrix is non-diagonal, it describes correlations between the features. This is where the mismatch between the theory and practice occurs. When $\mathbf{W} \in \Re^{p\times q}$, with $q < p$ we can think of this covariance matrix as a combination of a reduced rank portion ($\mathbf{W}\mathbf{W}^\top$, which has a rank of $q$) with a diagonal term, $\sigma^2\mathbf{I}$ that ensures positive definitiveness of the covariance. If $\sigma$ is small relative to the magnitude of the columns of $\mathbf{W}$, $\mathbf{w}_{:,i}$, then we can think of the structure of the covariance as being inherently low dimensional. In this chapter we will introduce another interpretation of that low rank structure: we will show it can arise through a dimensionality reduction. This comes about through a probabilistic interpretation of principal component analysis (PCA).

## 2.1 Probabilistic Linear Dimensionality Reduction

The term "principal components" was introduced by Harold Hotelling, he was interested in summarizing multivariate data with a reduced number of variables. Quoting from Hotelling, 1933, page 417:

> Consider $p$ variables attaching to each individual of a population. These statistical variables $y_1$, $y_2$, ... , $y_p$ might for example be scores made by school children in tests of speed and skill in solving arithmetical problems or in reading; or they might be various physical properties of telephone poles, or the rates of exchange among various currencies. The $y$'s will ordinarily be correlated. It is natural to ask whether some more fundamental set of independent variables exists, perhaps fewer in number than the $y$'s, which determine the values the $y$'s will take. If $x_1$, $x_2$, ... are such variables, we shall then have a set of relations of the form
>
> $$y_i = f(x_1, x_2, ...) \quad (i = 1, 2, ..., p) \quad (1)$$
>
> Quantities such as the $x$'s have been called mental factors in recent psychological literature. However in view of the prospect of application of these ideas outside of psychology, and the conflicting usage attaching to the word "factor" in mathematics, it will be better simply to call the $x$'s components of the complex depicted by the tests.

Where in the quote we have substituted Hotelling's notation with ours for ease of integration with the book.

Hotelling went on to assume that the components, $\mathbf{x}$, (which we refer to as latent variables) were drawn from a Gaussian density, and he restricted the relationship between them and the data to be linear. It turns out that there is a rotational identifiability issue in such models. To resolve this he suggested selecting, in turn, orthogonal components that describe the largest portion of the data variance. His approach of seeking components to explain the largest variance was motivated mainly by analogy. He called these the principal components.

Principal component analysis is one of the earliest approaches to dimensionality reduction. It also underlies many other approaches to dimensionality reduction. As we shall see, it can be viewed from different perspectives, but it seems appropriate to introduce it first from a perspective that is close to Hotelling's own motivation.

Hotelling's definition of principal components would be described in the machine learning literature as a *generative model*. He describes a generating process for the data: that it arises from a set of uncorrelated factors and is then mapped to a higher dimensional data space.

A modern probabilistic perspective on PCA also addresses it from a generative modeling perspective. By introducing *noise* in the generative model, PCA has been reformulated giving a model which does not need to recourse to analogies to formulate the associated optimization algorithm. By introducing a model of the noise in the system and optimizing the model through *maximum likelihood* we recover a fully probabilistic interpretation for PCA. In this guise the model is known as either "sensible PCA" or "probabilistic PCA".

Probabilistic PCA Tipping and Bishop [1999b], Roweis [1998] is a *latent variable* representation of data. The basic idea is the same as Hotelling's model: the "true" manifestation of the data lies in a low, $q$-dimensional latent space. However, we observe a data point which is mapped from the low dimensional space to a, $p$-dimensional space. The main difference in the probabilistic PCA model is that we include corrupting noise that corrupts the observed location of the data point.

## 2.2 The Probabilistic PCA Model

For our notation, we express the $i$th data point's position in the underlying latent space by the vector $\mathbf{x}_{i,:}$. The relationship between the latent space and the data space in the form of a linear mapping,

$$\mathbf{y}_{i,:} = \mathbf{W}\mathbf{x}_{i,:} + \boldsymbol{\mu} + \boldsymbol{\epsilon}_{i,:}$$

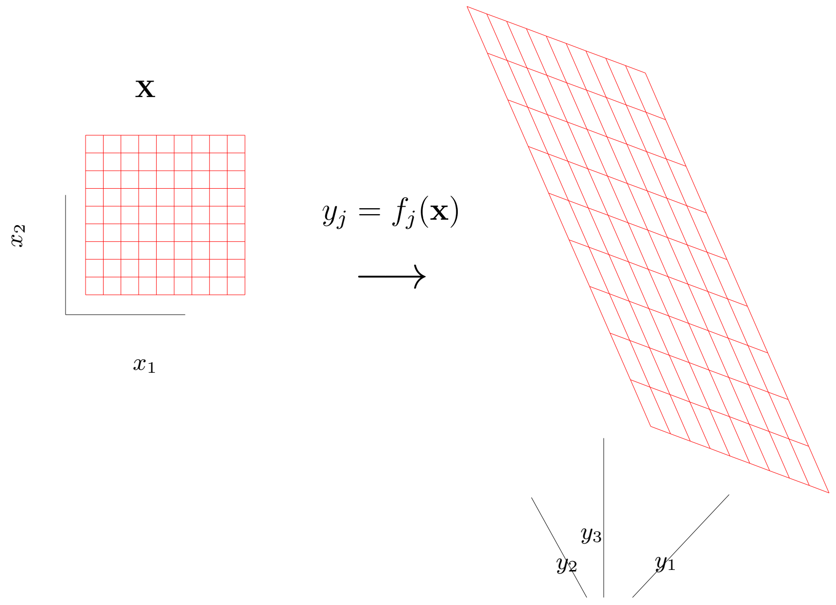


Figure 2.1: Mapping a two dimensional plane to a higher dimensional space in a linear way. Data are generated by corrupting points on the plane with noise.

where $\mathbf{W} \in \Re^{p,q}$ is a mapping matrix which encodes the relationship between the latent space and the data space. The vector $\boldsymbol{\epsilon}$ represents the corrupting noise. We will assume the noise is independently sampled from a Gaussian density,

$$\boldsymbol{\epsilon}_{i,:} \sim \mathcal{N}\left(0, \sigma^2\mathbf{I}\right),$$

where $\sigma$ is the standard deviation of the corrupting noise.

Linear dimensionality reduction can be thought of as finding a lower dimensional plane embedded in a higher dimensional space. The plane is described by the matrix $\mathbf{W}$. The direction of the vectors in this matrix define the orientation of the plane. Figure 2.1 shows how this works when the latent space is two dimensional and the data space is three dimensional.

We represent the entire data set in the form of a *design matrix*, $\mathbf{Y} \in \Re^{n \times p}$.

The next step is to incorporate these assumptions into a probability density. This is done by using a Gaussian density with a *mean function*. The basic idea

**boxfloat 2.1** Joint Probabilities

joint probability expresses the distribution for two simultaneously occurring events. We can think of it as being the probability of both $S = s$ and $Z = z$. It is expressed with commas, so we have the probability $P(S = s, Z = z)$.
Conditional probabilities represent the influence of one event upon another, so if we know that the value of $Z$ is dependent on $S$ we can write the distribution as $P(S = s|Z = z)$. If the outcome of $S$ is unaffected by the value of $Z$ then the two events are said to be independent and we can write $P(S = s|Z = z) = P(S = s)$.
The product rule of probability states that the joint distribution is related to the conditional distribution in the following way,

$$P(S = s, Z = z) = P(S = s|Z = z)P(Z = z).$$

is to specify the mean of a Gaussian density by a function that is dependent on the latent variables. If the variance of this density is taken to be the variance of the corrupting Gaussian noise then we have a simple Gaussian model for a data point, $\mathbf{y}_{i,:}$ given the mapping matrix, $\mathbf{W}$, the latent point, $\mathbf{x}_{i,:}$, and the noise variance, $\sigma^2$. If the noise is independently added to each feature, and has the same variance for each feature, then the mean for the $j$th feature of the $i$th data point will be given by $\mathbf{w}_{j,:}^\top \mathbf{x}_{i,:} + \mu_j$ and the corresponding variance will be $\sigma^2$. The Gaussian density governing the data point will then be

$$y_{i,j} \sim \mathcal{N}\left(\mathbf{w}_{j,:}^\top \mathbf{x}_{i,:} + \mu_j, \sigma^2\right).$$

If we assume that the noise is independent across both features and data, then can we write down the joint probability density for the entire data set as a product of the marginal densities for each data point and each feature given in (**??**). Note that we are not saying that each data point and feature is completely independent, we are saying that given the parameters of the model and the latent variables each feature of each data point is independent. In other words, the noise that corrupts the features of each data point is independent. Actual data points may be strongly correlated, both across observations and features (through the mapping matrix and the latent variables respectively).

Given this independence assumption the joint distribution over the observed data, given the parameters and latent variables, is then

$$\mathbf{Y} \sim \prod_{i=1}^{n} \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{w}_{j,:}^\top \mathbf{x}_{i,:} + \mu_j, \sigma^2\right).$$

We can rewrite this density, incorporating the feature independence in a multivariate Gaussian density with a *spherical covariance matrix matrix*, $\sigma^2\mathbf{I}$, as a product over observations:

$$p\left(\mathbf{Y}|\mathbf{X}, \mathbf{W}\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:} + \boldsymbol{\mu}, \sigma^2\mathbf{I}\right).$$

**boxfloat 2.2** Marginalization

he sum rule of probability relates the distribution over one variable, $P(S = s)$ to the joint distribution:

$$P(S = s) = \sum_{Z} P(S = s, Z = z),$$

this process is known as marginalization (presumably because the variable $Z$ disappears into the "margin"). The distribution $P(S = s)$ is known as the marginal distribution. For continuous variables and probability densities marginalization is achieved through integration,

$$p(X = x) = \int_{Y} p(X = x, Y = y)\mathrm{d}y$$

We will be using the shorthand described in box 1.1 which allows us to write

$$p(x, y) \equiv p(X = x, Y = y).$$

Note here that probability notation differs fundamentally from standard function notation. For function notation, $f(x, y)$ does not typically equal $f(y, x)$. It will do so only if the function is symmetric. This is because in function notation position is significant (as it is in many programming languages like C, Java and Matlab). However, probability notation is not the same as functional notation. When using probability notation, position isn't significant: there is an implicity "name" associated with each argument that we have dropped in our shorthand (this is like "named parameter" functions in programming languages such as Python, Ada and R). When names are provided ordering isn't important. In the shorthand for probability notation the names are implicit so $p(x, y) = p(y, x)$ even if the density function is not symmetric because we are really writing $p(X = x, Y = y) = p(Y = y, X = x)$.

**boxfloat 2.3** Independence over observations in likelihood and prior.

he fact that both are independent over data points (given the parameters) means that the joint density over latent variables and data is also independent over observations,

$$p(\mathbf{Y}, \mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = \prod_{i=1}^{n} p(\mathbf{y}_{i,:}, \mathbf{x}_{i,:}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$$

This means that the marginal likelihood for $\mathbf{Y}$ will also be independent over observations.

$$p(\mathbf{Y}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = \int \prod_{i=1}^{n} p(\mathbf{y}_{i,:}, \mathbf{x}_{i,:}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)\mathrm{d}\mathbf{x}_{i,:} \qquad = \prod_{i=1}^{n} p(\mathbf{y}_{i,:}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2).$$

This is a very typical set up for latent variable models. It should be contrasted with a Bayesian treatment of the parameters, which typically don't use prior densities which are independent over observations. Such priors induce dependencies between observations in the data set.

There are several unknowns in this likelihood: the parameters ($\mathbf{W}$, $\boldsymbol{\mu}$, and $\sigma^2$) and the latent variables, $\mathbf{X}$.

The latent variables, $\mathbf{X}$, are sometimes known as "nuisance parameters". This reflects the idea that if we were only interested in the values of the mapping matrix (which defines the embedded plane) we may not be interested in the positions of the latent variables. For PCA it will turn out that we are often interested in both. However, the nuisance parameters need to be dealt with. Hotelling took the approach of assuming that these parameters are sampled from another Gaussian density, with zero mean and unit variance,

$$x_{i,j} \sim \mathcal{N}(0, 1).$$

So the joint density for the components can be written

$$p(\mathbf{X}) = \prod_{i=1}^{n} \mathcal{N}(\mathbf{x}_{i,:}|\mathbf{0}, \mathbf{I}).$$

This density is often known as a *prior distribution.* Once we have defined the prior density we can compute the *marginal likelihood* of the data given the parameters, $p(\mathbf{Y}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$ by integration over the latent variables,

$$p(\mathbf{Y}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = \int p(\mathbf{Y}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)p(\mathbf{X})\mathrm{d}\mathbf{X}.$$

This integration can be done analytically for the case where both the likelihood, $p(\mathbf{Y}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$, and the prior, $p(\mathbf{X})$ are Gaussian. We will go through it in detail

**tipfloat 2.1** Integration Tricks for Probabilities

ntegration is a non-trivial operation: when dealing with integration over probability densities there are a couple of tricks that can help. The primary trick is that we know that any probability density must be normalized. By being familiar with the form of densities you effectively have knowledge of indefinite integrals. For example if we wish to know the indefinite integral of an exponentiated quadratic, $\int \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{b})^\top \mathbf{A}(\mathbf{x}-\mathbf{b})\right) \mathrm{d}\mathbf{x}$, we can use the fact that

$$\mathcal{N}(\mathbf{x}|\mathbf{b},\mathbf{A}) = \frac{1}{|2\pi\mathbf{A}|^{-\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{b})^\top \mathbf{A}^{-1}(\mathbf{x}-\mathbf{b})\right)$$

and the normalization property of a density, $\int \mathcal{N}(\mathbf{x}|\mathbf{b},\mathbf{A})\,\mathrm{d}\mathbf{x} = 1$, to derive

$$\int \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{b})^\top \mathbf{A}^{-1}(\mathbf{x}-\mathbf{b})\right) \mathrm{d}\mathbf{x} = |2\pi\mathbf{A}|^{-\frac{1}{2}}.$$

This trick is very useful in speeding up calculations. We can also apply it for the gamma density (box 1.4) to recover, $\int_0^\infty \exp(-bx)x^{a-1}\mathrm{d}x = \frac{\Gamma(a)}{b^a}$.
The normalization property leads to an often used shorthand for probabilities,

$$p(x|a,b) \propto \exp(-bx)x^{a-1},$$

where the implication here is that $x|a,b$ has a gamma density. This shorthand is often used when exploiting *conjugacy* to extract the posterior (see box 2.5).

now. The first step is to exploit the independence across data points in both the prior and the likelihood (see box 2.3 for details),

$$p(\mathbf{Y}|\mathbf{W},\boldsymbol{\mu},\sigma^2) = \prod_{i=1}^n \prod_{i=1}^n p(\mathbf{y}_{i,:}|\mathbf{W},\boldsymbol{\mu},\sigma^2)\mathrm{d}\mathbf{x}_{i,:}.$$

so we can analyze the density for each observation independently,

$$p(\mathbf{y}_{i,:}|\mathbf{W},\boldsymbol{\mu},\sigma^2) = \int p(\mathbf{y}_{i,:}|\mathbf{x}_{i,:},\mathbf{W},\boldsymbol{\mu},\sigma^2)p(\mathbf{x}_{i,:})\mathrm{d}\mathbf{x}_{i,:}.$$

We can perform this integral analytically: the Gaussian prior is conjugate to the latent variables in the likelihood (see box 2.5). The first step is to write down the joint density,

$$p(\mathbf{y}_{i,:},\mathbf{x}_{i,:}) \propto \exp\left(-\frac{1}{2\sigma^2}\left(\mathbf{y}_{i,:}-\mathbf{W}\mathbf{x}_{i,:}-\boldsymbol{\mu}\right)^\top\left(\mathbf{y}_{i,:}-\mathbf{W}\mathbf{x}_{i,:}-\boldsymbol{\mu}\right) - \frac{1}{2}\mathbf{x}_{i,:}^\top\mathbf{x}_{i,:}\right).$$

To marginalize the latent variables we collect terms involving $\mathbf{x}_{i,:}$,

$$p(\mathbf{y}_{i,:},\mathbf{x}_{i,:}) \propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y}_{i,:}-\boldsymbol{\mu})^\top(\mathbf{y}_{i,:}-\boldsymbol{\mu}) - \frac{1}{2}\mathbf{x}_{i,:}^\top\boldsymbol{\Sigma}_x^{-1}\mathbf{x}_{i,:} + \sigma^{-2}(\mathbf{y}_{i,:}-\boldsymbol{\mu})^\top\mathbf{W}\mathbf{x}_{i,:}\right).$$

**boxfloat 2.4** Bayes' Rule

For two random variables, $x$ and $y$, Bayes' rule relates the two possible conditional probabilities through the marginal distributions. It arises first through the product rule for probability,

$$p(x, y) = p(x|y)p(y)$$

Recall from box 1.1 that for probabilities $p(x, y) = p(y, x)$. The joint density can also be expressed by

$$p(x, y) = p(y|x)p(x).$$

So we can write $p(x|y)p(y) = p(y|x)p(x)$. This can be re-expressed as

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}.$$

This is known as Bayes' rule. If $x$ is a parameter or latent variable and $y$ is an observation then the formula has the following interpretation: The density $p(x)$ represents our knowledge about the latent variable before we saw the observation. This is known as the *prior* density. The density $p(y|x)$ represents the relationship between the latent variable (or parameter) and our observation. This is known as the *likelihood*. The density $p(x|y)$ is known as the *posterior*. It represents our updated understanding of $x$ arising from the observation of the data, $y$, and the relationship we specify between the data and $x$ through the likelihood. Finally $p(y)$ is known as the marginal likelihood. It represents the probability of the data under the marginalized model. The marginal likelihood is also a consequence of our definition of prior and likelihood because it is recovered via marginalization,

$$p(y) = \int p(y|x)p(x)\mathrm{d}x.$$

This marginalization is often the stumbling block for Bayesian methods as for complicated combinations of prior and likelihood the integral is generally not analytically tractable.

**boxfloat 2.5** Conjugate Priors

As we saw in box 2.4 Bayes' rule allows us to combine an observation (through a likelihood) with a prior density to recover a posterior density. However, the operation is often intractable due to the integral required to compute the marginal likelihood. Cojugacy is used to describe a particular class of likelihood and prior pairings that leads to tractable models. Conjugacy is used in the sense of conjoined. A prior applies to a particular parameter of the likelihood. Let's refer to it as $x$. If the prior is conjugate to the likelihood, then the functional form of the posterior density after multiplying by the prior will be identical to the prior.

For example, if we place a gamma prior, $p(x) \propto x^{a-1}\exp(-bx)$ over the precision (which is the inverse variance) of a zero mean Gaussian density, $p(y) \propto x^{\frac{1}{2}}\exp\left(-\frac{x}{2}y^2\right)$, we recover a conjugate relationship when we compute the posterior density,

$$p(x|y) \propto p(y|x)p(x) \propto x^{a-1}\exp(-bx)x^{\frac{1}{2}}\exp\left(-\frac{x}{2}y^2\right)$$

$$p(x|y) \propto x^{a+\frac{1}{2}-1}\exp(-(b+y^22)x),$$

which using the marginalization trick from tip 2.1 we can write down as a gamma,

$$p(x|y) = \mathcal{G}\left(x|a+\tfrac{1}{2}, b+\tfrac{y^2}{2}\right).$$

Other useful conjugate relationships include a Gaussian prior over the mean of a Gaussian density, a gamma prior over the rate parameter of a Poisson distribution, a beta prior over the parameters of a binomial and a Dirichlet prior over the parameters of a multinomial. For more on Bayesian modeling and conjugacy see Gelman et al. [1995].

where we have introduced $\boldsymbol{\Sigma}_x = (\sigma^{-2}\mathbf{W}^\top\mathbf{W} + \mathbf{I})^{-1}$. We now complete the square[1] for $\mathbf{x}_{i,:}$,

$$p(\mathbf{y}_{i,:}, \mathbf{x}_{i,:}) \propto \exp\Bigg( -\frac{1}{2}(\mathbf{y}_{i,:} - \boldsymbol{\mu})^\top(\sigma^{-2}\mathbf{I} - \sigma^{-4}\mathbf{W}\boldsymbol{\Sigma}_x\mathbf{W}^\top)(\mathbf{y}_{i,:} - \boldsymbol{\mu}) \\ -\frac{1}{2}\left(\mathbf{x}_{i,:} - \boldsymbol{\Sigma}_x\sigma^{-2}(\mathbf{y}_{i,:} - \boldsymbol{\mu})\right)^\top \boldsymbol{\Sigma}_x^{-1}\left(\mathbf{x}_{i,:} - \boldsymbol{\Sigma}_x\sigma^{-2}(\mathbf{y}_{i,:} - \boldsymbol{\mu})\right)\Bigg).$$

This process of completing the matrix square is a very common operation when manipulating Gaussians. Here it has had the effect of refactorizing the joint density into the form $p(\mathbf{y}_{i,:}, \mathbf{x}_{i,:}) = p(\mathbf{x}_{i,:}|\mathbf{y}_{i,:})p(\mathbf{y}_{i,:})$, i.e. the product of the posterior and the marginal likelihood. We recognize the first term in the exponent of (**??**) as belonging to the marginal likelihood,

$$p\left(\mathbf{y}_{i,:}\right) \propto \exp\left(-\frac{1}{2}(\mathbf{y}_{i,:} - \boldsymbol{\mu})^\top(\sigma^{-2}\mathbf{I} - \sigma^{-4}\mathbf{W}\boldsymbol{\Sigma}_x\mathbf{W}^\top)(\mathbf{y}_{i,:} - \boldsymbol{\mu})\right)$$

and the second as being from the posterior of the latent variables given the data,

$$p\left(\mathbf{x}_{i,:}|\mathbf{y}_{i,:}\right) \propto \exp\left(-\frac{1}{2}\left(\mathbf{x}_{i,:} - \boldsymbol{\Sigma}_x\sigma^{-2}(\mathbf{y}_{i,:} - \boldsymbol{\mu})\right)^\top \boldsymbol{\Sigma}_x^{-1}\left(\mathbf{x}_{i,:} - \boldsymbol{\Sigma}_x\sigma^{-2}(\mathbf{y}_{i,:} - \boldsymbol{\mu})\right)\right).$$

Both these densities are also in the form of Gaussians.

We will return to the posterior density later, for the moment though we will focus on the marginal likelihood. As we mentioned the marginal likelihood is in the form of a multivariate Gaussian. By inspection we can identify that the mean of that Gaussian is given by $\boldsymbol{\mu}$ and the covariance is given by $(\sigma^{-2}\mathbf{I} - \sigma^{-4}\mathbf{W}\boldsymbol{\Sigma}_x\mathbf{W}^\top)^{-1}$. We can use the matrix inversion lemma (see box 2.6) to rewrite this as

$$(\sigma^{-2}\mathbf{I} - \sigma^{-4}\mathbf{W}\boldsymbol{\Sigma}_x\mathbf{W}^\top)^{-1} = \sigma^2\mathbf{I} + \mathbf{W}\mathbf{W}^\top.$$

So the final marginal likelihood has the form

$$\mathbf{y}_{i,:} \sim \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{C}\right).$$

where the covariance matrix is given by $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$.

### 2.2.1 Constrained Gaussian Model

The marginal likelihood we have derived is in the same form of that we introduced at the beginning of the chapter. The covariance is composed of a

[1]Completing the square for a matrix quadratic form is similar to the univariate case. We first look to express the quadratic form as a matrix inner product and we then compensate for the constant term that arises by adding it back in.

low rank term, $\mathbf{W}\mathbf{W}^\top$ which is of rank, at most, $q$, and a noise term, $\sigma^2\mathbf{I}$. The probabilistic model described by the marginal likelihood is a Gaussian density with this covariance. Before proceeding, let's just deal with one potential question. Given that we've ended up with a Gaussian model applied independently to the data, what is the advantage to this approach over just fitting a multivariate Gaussian model to the data with a general covariance matrix, $\mathbf{C}$. The maximum likelihood solution in this case would cause us to set the covariance matrix to the sample based covariance, $\mathbf{C} = n^n\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$. However, underlying this matrix there is no implicit low rank structure such as that we've derived by considering the latent variable model. This low rank structure reflects our representation of the data as having a lower implicit dimensionality, $q$, than the observed dimensionality, $p$. The covariance matrix in the probabilistic PCA model results in a constrained form of Gaussian model. As we increase the latent dimensionality, we place less constraints on the form of the Gaussian. If we set $q = p - 1$, then the Gaussian becomes unconstrained. ***why $p-1$ and not just $p$.***

### 2.2.2 Alternative Derivation of Marginal Likelihood

We derived the marginal likelihood for the model by computation with Bayes' rule. However, because we specified each component of our model is Gaussian there is also another approach to deriving the model using properties of multivariate Gaussians (see box 1.2).

The dimensionality reduction model we specified works as follows: each data observation is assumed to have been generated by first sampling a $q$-dimensional vector from a zero mean Gaussian with covariance $\mathbf{I}$,

$$\mathbf{x}_{i,:} \sim \mathcal{N}(0, \mathbf{I}).$$

This vector is then multiplied by the mapping matrix $\mathbf{W}$, to produce a $p$-dimensional vector which is then offset by a mean vector, $\boldsymbol{\mu}$, and corrupted with Gaussian noise, $\boldsymbol{\epsilon}$. This is summarized as

$$\hat{\mathbf{y}}_{i,:} = \mathbf{W}\mathbf{x}_{i,:} + \boldsymbol{\mu} + \boldsymbol{\epsilon}_{i,:}.$$

Since the noise and the $\mathbf{x}_{i,:}$ are both Gaussian distributed and the other vectors are non-stochastic, this is enough information to specify both the marginal density for $\mathbf{y}_{i,:}$ and its covariance matrix. Any Gaussian sample multiplied by a scalar value simply scales the standard deviation of that density, so a unit variance Gaussian scaled by the value $w$ becomes a Gaussian with variance $w^2$. Similarly, a vector sampled from a zero mean unit covariance Gaussian distribution multiplied by a matrix, $\mathbf{W}$ leads to a Gaussian density with *covariance* $\mathbf{W}\mathbf{W}^\top$. The effect of offsetting the variable with $\boldsymbol{\mu}$ merely shifts the mean of the Gaussian from $\mathbf{0}$ to $\boldsymbol{\mu}$. Finally adding the corrupting noise is

**boxfloat 2.6** The Matrix Inversion Lemma

he matrix inversion gives an approach to rewriting matrix inverses of the form $(\mathbf{A}+\mathbf{BCD})^{-1}$, where $\mathbf{A}\in\Re^{n\times n}$ and $\mathbf{C}\in\Re^{m\times m}$ are both invertible matrices. The matrix $\mathbf{B}\in\Re n\times m$ and $\mathbf{D}\in\Re m\times n$. The lemma states in general terms that this inverse can be rewritten as

$$(\mathbf{A}+\mathbf{BCD})^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{C}^{-1}+\mathbf{D}^{\top}\mathbf{A}^{-1}\mathbf{B}^{\top}\right)^{-1}\mathbf{DA}^{-1}.$$

It can be particularly useful when the inverse of $\mathbf{A}$ is quick to compute, for example if it is diagonal. In these cases it allows us to move between computing an inverse of an $m\times m$ matrix and an $n\times n$ matrix which can lead to computational savings. For example we might be interested in an inverse of the form

$$(a\mathbf{I}+\mathbf{BB}^{\top})^{-1}=a^{-1}\mathbf{I}-a^{-2}\mathbf{B}\left(\mathbf{I}+a^{-1}\mathbf{B}^{\top}\mathbf{B}\right)\mathbf{B}^{\top}.$$

The formula is very useful for algebraic manipulation, but in implementation some care must be taken in its application: computing the left hand side is more numerically stable than computing the right hand side.

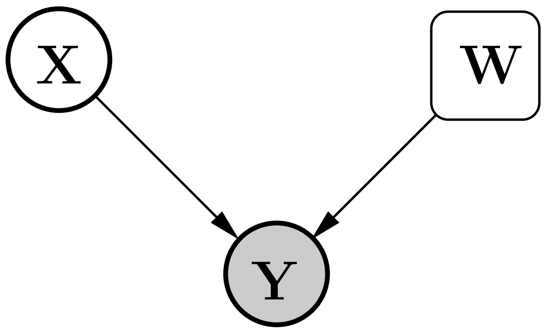


Figure 2.2: What goes here.

equivalent to summing two Gaussian random variables: the resulting variable's covariance is the sum of the two covariances. So if the noise has covariance $\sigma^2$ then we have

$$\mathbf{y}_{i,:}\sim\mathcal{N}\left(\boldsymbol{\mu},\mathbf{WW}^{\top}+\sigma^2\mathbf{I}\right).$$

The covariance of the Gaussian model we have derived is recognized as the same covariance we used to produce the samples in Section 1.11. There we used a probabilistic PCA model with a latent dimensionality of $q=2$ (for data of dimensionality $p=1000$). The squared distance histogram we obtained was far closer to the theory for two dimensional data than that for 1000 dimensional data.

## 2.3 Optimization of Probabilistic PCA

***define this as feature consistency, when independence across data points occurs***

### 2.3.1 Maximum Likelihood and Blessing of Dimensionality

When we consider maximum likelihood consistency arguments [see e.g. Wasserman, 2003, pg 126] we see that this model isn't consistent as we increase the number of data points, $n$, for fixed data dimensionality, $p$, but it *is* consistent as we increase data dimensionality, $p$, for a fixed number of data points $n$. This is a clear *blessing of dimensionality* that emerges when we assume correlation between data points in the model.

Our objective is to find the parameters of the model. We do this through maximum likelihood. The idea is to find the parameters which make the data more likely than any other. Maximum likelihood is often justified through consistency arguments. These arguments run as follows: if the data was really generated by a variant of the model you are considering, then as the size of the data set you have approaches infinity then you will recover the original model. We can show this consistency by introducing the Kullback-Leibler (KL) divergence (box **??**). Let's assume that the true data generating density is given by $p(\mathbf{Y})$. The inclusive KL-divergence between the true data generating density and our approximation is given by

$$\mathrm{KL}\left(p(\mathbf{Y})\,\|\,p(\mathbf{Y}|\mathbf{W},\sigma^2,\boldsymbol{\mu})\right) = \int p(\mathbf{Y})\log\frac{p(\mathbf{Y})}{p(\mathbf{Y}|\mathbf{W},\sigma^2,\boldsymbol{\mu})}\mathrm{d}\mathbf{Y}.$$

we rewrite this in the form of two expectations

$$\mathrm{KL}\left(p(\mathbf{Y})\,\|\,p(\mathbf{Y}|\mathbf{W},\sigma^2,\boldsymbol{\mu})\right) = \left\langle\log p(\mathbf{Y})\right\rangle_{p(\mathbf{Y})} - \left\langle\log p(\mathbf{Y}|\mathbf{W},\sigma^2,\boldsymbol{\mu})\right\rangle_{p(\mathbf{Y})}.$$

The first expectation is simply the negative entropy of the true data generating density. It is not affected by the parameters of the model. The second term is the expectation of the log likelihood of the data under the true generating density, $p(\mathbf{Y})$. If the log likelihood of the data factorizes across the data points,

$$p(\mathbf{Y}|\mathbf{W},\sigma^2,\boldsymbol{\mu}) = \prod_{i=1}^{n} p(\mathbf{y}_{:,i}|\mathbf{W},\sigma^2,\boldsymbol{\mu}),$$

then we can write that expectation as

$$\left\langle\log p(\mathbf{Y}|\mathbf{W},\sigma^2,\boldsymbol{\mu})\right\rangle_{p(\mathbf{Y})} = \sum_{i=1}^{n}\left\langle\log p(\mathbf{y}_{:,i}|\mathbf{W},\sigma^2,\boldsymbol{\mu})\right\rangle_{p(\mathbf{y}_{:,i})},$$

**boxfloat 2.7** The Kullback-Leibler Divergence

he Kullback-Leibler divergence Kullback and Leibler [1951], also known as the relative entropy ***AND WHAT ELSE*** is an information theoretic assessment of the dissimilarity between two distributions or densities. It is the expected value of the log ratio of the two densities. The asymmetry arises as you must choose one of the densities for the expectation. Here, for example, we compare two densities $p(x)$ and $q(x)$ and take the expectation under $p(x)$

$$\mathrm{KL}\left(p(x)\,\|\,q(x)\right) = \int p(x)\log\frac{p(x)}{q(x)}\mathrm{d}x.$$

The log ratio is always taken such that whichever density is used for the expectation is always the numerator in the ratio. *Jensen's inequality* can be used to show that the KL-divergence is always positive unless $q(x) = p(x)$ when it is zero. This allows the KL-divergence to be used when approximating one density with another. We can minimize the KL-divergence with respect to $q(x)$ to find an approximation for $p(x)$. When performing these optimizations we must typically make a choice of which KL-divergence to use. If the expectation is taken under $p(x)$ we call this as the *inclusive* KL-divergence. If it is taken under $q(x)$ we call this the *exclusive* KL-divergence. The term inclusive reflects the fact that when we wish to approximate $p(x)$ and we take the expectation under that density we *include* in that expectation all the actual areas of high density. The exclusive form only considers areas of high density under the approximating density.

If the samples are *exchangeable* then the marginal likelihood for each data point will be the same and we can write this as

$$\left\langle \log p(\mathbf{Y}|\mathbf{W},\sigma^2,\boldsymbol{\mu})\right\rangle_{p(\mathbf{Y})} = n\left\langle \log p(\mathbf{y}|\mathbf{W},\sigma^2,\boldsymbol{\mu})\right\rangle_{p(\mathbf{y})},$$

where we are using $p(\mathbf{y})$ to marginal probability for any single data point.

We don't have the true marginal, $p(\mathbf{y})$, but we do have samples from it: these are our data points. We therefore make use of a sample based approximation (see box 6.1) to the integral,

$$\left\langle \log p(\mathbf{y}|\mathbf{W},\sigma^2,\boldsymbol{\mu})\right\rangle_{p(\mathbf{y})} = \frac{1}{n}\sum_{i=1}^{n}\log p(\mathbf{y}_i|\mathbf{W},\sigma^2,\boldsymbol{\mu}).$$

## 2.4 Maximizing the Log Likelihood

In practice of course, we are given data and want to determine the parameters of the model from that data. The parameters in probabilistic PCA are $\mathbf{W}$, $\boldsymbol{\mu}$ and $\sigma^2$.The log likelihood of the model is given by

$$\log p(\mathbf{Y}|\mathbf{W},\boldsymbol{\mu},\sigma^2) = -\frac{np}{2}\log 2\pi - \frac{n}{2}\log|\mathbf{C}| - \frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_{i,:}-\boldsymbol{\mu})^{\top}\mathbf{C}^{-1}(\mathbf{y}_{i,:}-\boldsymbol{\mu}).$$

**boxfloat 2.8** Sample Approximations to Integrals

sample based approximation to an integral is a discrete sum approximation. The approximation has the form

$$\int f(x)p(x)\mathrm{d}x \approx \frac{1}{S}\sum_{i=1}^{S} f(x_i)$$

if $\{x_i\}_{i=1}^{S}$ are a set of samples from $p(x)$. As the number of samples approaches infinity, this approximation becomes exact.

This is our objective function. We want to find the global maximum of this function with respect to the parameters. The standard approach for finding a stationary point such as a maximum is to differentiate the function with respect to the parameters and find where the gradient is zero. Any maximization problem havs an equivalent minimization problem: simply place a minus sign in front of the objective. By convention, the optimization community prefers to discuss minimization rather than maximization. Optimization forms a very important component of machine learning. There is a strong case for the argument that machine learning is nothing more than model formulation (writing down an objective function) and optimization. The approach to model formulation we are taking in this chapter is a probabilistic one. In an acknowledgement of the importance of optimization for machine learning, we follow their convention and consider minimizing the negative log likelihood:

$$E(\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = \frac{n}{2}\log|\mathbf{C}| + \frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_{i,:} - \boldsymbol{\mu})^{\top}\mathbf{C}^{-1}(\mathbf{y}_{i,:} - \boldsymbol{\mu}), \qquad (2.1)$$

where we have dropped the term which is constant in the parameters. This objective function is sometimes known as an *error function.* But we can think of it as an approximation to the KL-divergence we are trying to minimize.

## 2.5 Optimizing With Respect to the Mean Vector

We will first look for the minimum of (2.1) with respect to the vector $\boldsymbol{\mu}$. This involves looking for the gradient of the error function with respect to each element of the vector $\boldsymbol{\mu}$. We can think of this as multivariate calculus: the usual rules of calculus apply, but here they are being applied to matrix operations. However, matrix operations are merely a short-hand for a series of operations being placed on a collection of parameters. Matrix differentiation has a generalized set of rules. The term of the error function involving the mean is the last term,

$$E(\boldsymbol{\mu}) = -\frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_{i,:} - \boldsymbol{\mu})^{\top}\mathbf{C}^{-1}(\mathbf{y}_{i,:} - \boldsymbol{\mu})$$

which is recognized as the matrix equivalent of a quadratic: a quadratic form. The gradient of this form with respect to the mean vector is

$$\frac{\mathrm{d}E(\boldsymbol{\mu})}{\mathrm{d}\boldsymbol{\mu}} = \sum_{i=1}^{n}\mathbf{C}^{-1}(\mathbf{y}_{i,:} - \boldsymbol{\mu})$$

which can be rewritten

$$\frac{\mathrm{d}E(\boldsymbol{\mu})}{\mathrm{d}\boldsymbol{\mu}} = \mathbf{C}^{-1}\left(\sum_{i=1}^{n}\mathbf{y}_{i,:} - n\boldsymbol{\mu}\right).$$

To find the minimum, we look for a point where the gradients of the function are all zero,

$$\mathbf{0} = \mathbf{C}^{-1}\left(\sum_{i=1}^{n}\mathbf{y}_{i,:} - n\boldsymbol{\mu}\right)$$

which implies that

$$\mathbf{C}^{-1}\boldsymbol{\mu} = \mathbf{C}^{-1}\frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_{i,:}$$
$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_{i,:}.$$

So the gradients are zero when $\boldsymbol{\mu}$ is given by the sample mean of the data. We can double check that this is a minimum by looking at the curvature of the error function. The curvature is found by the second derivative of the error function. In multivariate systems, the curvature consists of a matrix (it includes all possible second derivatives between the parameters), it is also known as the Hessian. The Hessian for this system is easily derived,

$$\frac{\mathrm{d}^2E(\boldsymbol{\mu})}{\mathrm{d}\boldsymbol{\mu}\boldsymbol{\mu}^{\top}} = \mathbf{C}^{-1},$$

it is given by the inverse covariance matrix: also known as the *precision* or *information matrix.* This matrix is known to be positive definite if the Gaussian is to be normalizable. The Hessian being positive definite at a given point is also a sufficient condition to confirm that this is indeed a minimum (equivalent in the univariate case to the second derivative being positive). This implies that the solution for $\boldsymbol{\mu}$ is a minimum. We also note that the curvature is not dependent on the value of $\boldsymbol{\mu}$. In other words the Hessian is always positive-definite. This implies that this multivariate function is convex in $\boldsymbol{\mu}$, which in turn implies that the minimum is unique and therefore global.

**tipfloat 2.2** Centered Data Sets

t is very often convenient (for notational reasons) to deal with a centered data set. As we have seen in the main text, the maximum likelihood solution for the mean vector of a Gaussian is that it should equal the sample mean. A centered data set is one where the sample mean of the data has been subtracted, when working with a centered data and Gaussian densities, we can assume that we have already found the maximum likelihood solution for the mean.

We can center a data set stored in a design matrix with the following operation,

$$\hat{\mathbf{Y}} = \mathbf{Y} - \mathbf{1}\boldsymbol{\mu}^\top,$$

where $\boldsymbol{\mu}$ is the *sample mean*. This operation can also be carried out through multiplication by a *centering matrix*,

$$\mathbf{H} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^\top \in \Re^{n\times n},$$

so we have $\hat{\mathbf{Y}} = \mathbf{H}\mathbf{Y}$. When modeling a centered data set with a Gaussian density we can set the mean of the Gaussian to be zero and we know we have the maximum likelihood solution.

In other words, the maximum likelihood solution for the mean vector is given by the sample mean, the fact that the optimization surface is convex, and that the solution is not dependent on the covariance matrix, means that this solution is a global minimum of the objective function. It can be computed before either $\mathbf{W}$ and $\sigma^2$ are known. For this reason, we often consider the data to be *centered* and we ignore the mean. In tip 2.2 we show how this operation can be applied. We follow this approach when finding the rest of the maximum likelihood solution.

## 2.6 Optimizing With Respect to the Covariance

A common trick to reduce the notational complexity of equations is to work with the centered data set. This is useful for making equations shorter, but it is worth bearing in mind that it only works because the maximum likelihood solution for the mean is the sample mean. Given this solution, we can center the data set and work with $\hat{\mathbf{Y}} = \mathbf{Y} - \mathbf{1}\boldsymbol{\mu}^\top$. Substituting this form into the likelihood results in a new "likelihood" over the centered data,

$$p\left(\hat{\mathbf{Y}}|\mathbf{W}\right) = \prod_{j=1}^{p} \mathcal{N}\left(\hat{\mathbf{y}}_{i,:}|\mathbf{0},\mathbf{C}\right),$$

where we are using the particular form for the covariance matrix, $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$. Tipping and Bishop [1999b] showed that the global maximum likelihood solution for the parameters $\mathbf{W}$ and $\sigma^2$ can be found through an eigenvalue problem. Before we review how that is done, we will first show what the maximum likelihood solution for an unconstrained covariance matrix would be, i.e. for the moment we will ignore the fact that $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$ and allow $\mathbf{C}$ to be any positive definite matrix.

Up to constant terms the negative log likelihood[2] of the data is

$$E(\mathbf{W}, \sigma^2) = \frac{np}{2}\log 2\pi + \frac{n}{2}\log|\mathbf{C}| + \frac{1}{2}\sum_{i=1}^{n}\hat{\mathbf{y}}_{i,:}^\top\mathbf{C}^{-1}\hat{\mathbf{y}}_{i,:}.$$

We are going to look for the minimum through matrix calculus. This means that it is much more convenient to use matrix notation for all terms in our objective instead of the sum sign. To do this we can introduce the matrix trace (see box 2.9). The matrix trace (which is the sum of the diagonal terms of a matrix) allows us to write

$$E(\mathbf{W}, \sigma^2) = \frac{np}{2}\log 2\pi + \frac{n}{2}\log|\mathbf{C}| + \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\right).$$

Now we have our objective function in the form of matrix operations, we can compute the gradient of the objective with respect to the coovariance matrix using results from matrix calculus (see also appendix **??**). Many of these results will also prove useful in future chapters, so we will go through them in a little more detail here. First of all we can compute the gradient with respect to the covariance matrix, $\mathbf{C}$. This involves a couple of matrix derivatives. Firstly, we need the gradient of the log determinant,

$$\frac{\mathrm{d}\log|\mathbf{C}|}{\mathrm{d}\mathbf{C}} = \mathbf{C}^{-1}.$$

The gradient of the log determinant has a nice simple form: determinants are associated with volumes and therefore dependent on products across dimensions. The logarithms of products lead to sums, typically leading to simpler derivatives than the gradient of the product directly (see intuition 2.1).

Secondly, we need to be able to compute the gradient of the trace term. This is slightly more involved as the $\mathbf{C}$ appears in inverse form. We will therefore need to make use of the following relationship,

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{C}} = -\mathbf{C}^{-1}\frac{\mathrm{d}f}{\mathrm{d}\mathbf{C}^{-1}}\mathbf{C}^{-1},$$

which is a multivariate generalization of

$$\frac{\mathrm{d}f}{\mathrm{d}\lambda} = -\lambda^{-2}\frac{\mathrm{d}f}{\mathrm{d}\lambda^{-1}}$$

**boxfloat 2.9** The Trace of a Matrix

he trace of a matrix is merely the sum of its diagonal elements,

$$\mathrm{tr}\left(\mathbf{A}\right) = \sum_i a_{i,i}.$$

However, it has some important characteristics. Whilst in normal matrix multiplication, the ordering is important, within a trace the ordering becomes less important, so for example

$$\mathrm{tr}\left(\mathbf{AB}\right) = \mathrm{tr}\left(\mathbf{BA}\right)$$

if $\mathbf{A}$ and $\mathbf{B}$ are both square matrices. This is very useful when computing expectations of matrix quadratic forms. We can write a sum as matrix inner products as $\sum_{i=1}^n \mathbf{y}_{i,:}^\top \mathbf{C}^{-1}\mathbf{y}_{i,:}$ Consider the following matrix, $\mathbf{A} = \mathbf{Y}\mathbf{C}^{-1}\mathbf{Y}^\top$, it has dimensionality $n \times n$ (see tip **??**). The element from the $i$th row and $j$th of $\mathbf{A}$ is given by

$$a_{i,j} = \mathbf{y}_{i,:}^\top \mathbf{C}^{-1}\mathbf{y}_{j,:}.$$

The trace of $\mathbf{A}$ will be the sum of the diagonal elements,

$$\mathrm{tr}\left(\mathbf{Y}\mathbf{C}^{-1}\mathbf{Y}^\top\right) = \sum_{i=1}^n \mathbf{y}_{i,:}^\top \mathbf{C}^{-1}\mathbf{y}_{i,:},$$

recovering the sum of matrix inner products. The ordering property of the trace means that

$$\mathrm{tr}\left(\mathbf{Y}\mathbf{C}^{-1}\mathbf{Y}^\top\right) = \mathrm{tr}\left(\mathbf{C}^{-1}\mathbf{Y}^\top\mathbf{Y}\right).$$

This reformulation can be very useful when computing expectations of quadratic forms,

$$\sum_{i=1}^n \left\langle \mathbf{y}_{i,:}^\top \mathbf{C}^{-1}\mathbf{y}_{i,:}\right\rangle_{p(\mathbf{Y})} = \mathrm{tr}\left(\mathbf{C}^{-1}\left\langle \mathbf{Y}^\top\mathbf{Y}\right\rangle_{p(\mathbf{Y})}\right).$$

*Important note*: although we make use of the notation $\mathrm{tr}\left(\mathbf{Y}\mathbf{C}^{-1}\mathbf{Y}^\top\right)$ you shouldn't normally compute it that way. For example in Octave/Matlab you might be tempted to write

```
trace(Y*Cinv*Y')
```

but this will use matrix multiplications to create an $n \times n$ matrix, and then only the diagonal elements are needed in the resulting sum. More efficient would be

```
sum(sum(Y.*(Y*Cinv)))
```

where $.*$ is Matlab/Octave notation for the element by element product ($\odot$).

**tipfloat 2.3** Matrix Dimensions

his is perhaps an obvious trick, but a very useful one so worth reinforcing. We will be making a lot of use of matrix notation in this book. When using matrix multiplication it is easy to get confused about where transposes should be and what the ordering should be. One important sanity check is a dimensionality check on the matrices. We have defined the data matrix to be $\mathbf{Y} \in \Re^{n\times p}$ and the inverse covariance matrix is $\mathbf{C}^{-1} \in \Re^{p\times p}$. When writing down a multiplication of these matrices try writing the dimensionalities below,

$$\begin{array}{cccl} \mathbf{Y} & \mathbf{C}^{-1} & \mathbf{Y}^\top & \text{matrix multiplication} \\ n\times p & p\times p & p\times n & = n\times n \text{ dimensionalities.} \end{array}$$

At the interface between the multiplications, the dimensionalities should match. The final dimensionality is found by "canceling" these interface dimensionalities. You should get in the habit of performing this sanity check when writing down equations.
If $\mathbf{K} \in \Re^{n\times n}$ and recalling that $\mathbf{X} \in \Re^{n\times q}$ and $\mathbf{W} \in \Re^{p\times q}$ confirm for yourself the dimensionality of the results from the following operations:

$$\begin{aligned} \mathbf{Y}^\top\mathbf{K}\mathbf{Y} \\ \mathbf{X}\mathbf{X}^\top\mathbf{K} \\ \mathbf{X}\mathbf{W}^\top - \mathbf{Y} \\ \mathbf{X}\mathbf{X}^\top\mathbf{K}^{-1}\mathbf{Y} \end{aligned}$$

A final note with regard to matrices and vectors. Whilst it may seem logical to consider a row extracted from a matrix (for which we use the Matlab-like notation, $\mathbf{a}_{i,:}$) as a row vector, this makes parsing of equations harder. In this book all vectors are column vectors, this means that $\mathbf{a}^\top\mathbf{a}$ is easily spotted as an inner product and quadratic forms always appear in the following manner $\mathbf{a}^\top\mathbf{B}\mathbf{a}$.

**intfloat 2.1** Gradient of the Log Determinant

eterminants represent volumes (see box 1.3). When we take the gradient of the determinants logarithm we are seeing how the log volume changes with respect to elements of the matrix. This is perhaps easier to see if we first consider a diagonal matrix, $\mathbf{C} = \mathbf{\Lambda}^2$ where $\mathbf{\Lambda}$ is a diagonal matrix. Now the log determinant is simply given by

$$\log\left|\mathbf{\Lambda}^2\right| = \sum_i^p \log \lambda_i^2.$$

The gradient of this sum with respect to an off diagonal element of $\mathbf{\Lambda}$ is zero. The gradient with respect to the $i$th diagonal element is,

$$\frac{\mathrm{d}\sum_{j=1}^p \log \lambda_j^2}{\mathrm{d}\lambda_i^2} = \lambda_i^{-2}$$

so we have

$$\frac{\mathrm{d}\log\left|\mathbf{\Lambda}^2\right|}{\mathrm{d}\mathbf{\Lambda}^2} = \mathbf{\Lambda}^{-2}.$$

What this is basically saying is that the largest changes are made to the log volume by changes in the smallest values of $\lambda_i$. When applied to positive definite matrices, similar intuitions apply but the basis $\mathbf{\Lambda}^2$ is rotated,

$$\frac{\mathrm{d}\log\left|\mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^\top\right|}{\mathrm{d}\mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^\top} = \mathbf{U}\mathbf{\Lambda}^{-2}\mathbf{U}^\top,$$

since any symmetric positive definite matrix can be represented by $\mathbf{C} = \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^\top$ and $\mathbf{C}^{-1} = \mathbf{U}\mathbf{\Lambda}^{-2}\mathbf{U}^\top$ because $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ we recover

$$\frac{\mathrm{d}\log\left|\mathbf{C}\right|}{\mathrm{d}\mathbf{C}} = \mathbf{C}^{-1}$$

this result is a multivariate generalization of the univariate derivative,

$$\frac{\mathrm{d}\log\lambda}{\mathrm{d}\lambda} = \frac{1}{\lambda}.$$

**intfloat 2.2** Derivative of a Trace

he trace of the matrix is the sum of the diagonal values. So for any matrix we have,

$$\text{tr}\left(\mathbf{A}\right) = \sum_{i=1}^{p} a_{i,i},$$

where $a_{i,i}$ is the $i$th diagonal element of the matrix. We can easily differentiate this expression with respect to all elements of the matrix giving the identity,

$$\frac{\text{dtr}\left(\mathbf{A}\right)}{\text{d}\mathbf{A}} = \mathbf{I},$$

as a result. This is because the gradient of the diagonal sum with respect to each off diagonal element is zero whilst the gradient with respect to each diagonal element is unity.
For a trace of a product of matrices,

$$\text{tr}\left(\mathbf{AB}\right) = \sum_{i=1}^{p} \mathbf{a}_{i,:}^{\top}\mathbf{b}_{:,i} = \sum_{i=1}\sum_{j=1} a_{i,j}b_{j,i}$$

so the gradients are given by

$$\frac{\text{dtr}\left(\mathbf{AB}\right)}{\text{d}\mathbf{A}} = \mathbf{B}^{\top}.$$

which can be derived through the chain rule.

That leaves us with computation of the matrix derivative of a trace (see intuition 2.2). The direct derivative of a trace is given by the identity matrix,

$$\frac{\text{dtr}\left(\mathbf{A}\right)}{\text{d}\mathbf{A}} = \mathbf{I}.$$

If the matrix is part of a product within the trace then the result is

$$\frac{\text{dtr}\left(\mathbf{AB}\right)}{\text{d}\mathbf{A}} = \mathbf{B}^{\top}.$$

so for the trace term that appears in the log likelihood we can make use of the following result,

$$\frac{\text{dtr}\left(\mathbf{C}^{-1}\hat{\mathbf{Y}}^{\top}\hat{\mathbf{Y}}\right)}{\text{d}\mathbf{C}^{-1}} = \hat{\mathbf{Y}}^{\top}\hat{\mathbf{Y}}.$$

Combining it with with equation (2.6)–(2.6) we can write down the gradient of the negative log likelihood with respect to the covariance matrix,

$$\frac{\text{d}E(\mathbf{W},\sigma^2)}{\text{d}\mathbf{C}} = \frac{n}{2}\mathbf{C}^{-1} - \frac{1}{2}\mathbf{C}^{-1}\hat{\mathbf{Y}}^{\top}\hat{\mathbf{Y}}\mathbf{C}^{-1}. \tag{2.2}$$

[2]The equivalence between this expression and the negative log likelihood we used in the previous section can easily be seen if you set $\hat{\mathbf{y}}_{i,:} = \mathbf{y}_{i,:} - \boldsymbol{\mu}$ and recall that at the minimum $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_{i,:}$.

**boxfloat 2.10** The Exponential Family

he exponential family is a class of distributions and densities that can be written in the form

$$p(\mathbf{y}|\boldsymbol{\theta}) = h(\mathbf{y}) \exp\left(\eta(\boldsymbol{\theta})^\top T(\mathbf{y}) - A(\boldsymbol{\theta})\right)$$

where $\boldsymbol{\theta}$ is a vector of parameters and $\mathbf{y}$ is the space over which the density is defined. The function $T(\mathbf{y})$ is known as the sufficient statistic of the distribution. The form is dependent on four functions: $h(\mathbf{y})$ and $A(\boldsymbol{\theta})$ which are scalar functions of the data and parameters respectively; $T(\mathbf{y})$ and $\eta(\boldsymbol{\theta})$ which are vector output functions
***sufficient statistics***
***natural parameters***
***normalization factor***

The covariance matrix, $\mathbf{C}$, has a particular structure for this probabilistic PCA model, so it is not legitimate to optimize directly with respect to the covariance matrix. We need to find the gradients with respect to the mapping matrix, $\mathbf{W}$ and the noise variance $\sigma^2$. However, while we are here it seems legitimate to take a slight detour and remind ourselves what the maximum likelihood solution for the covariance matrix is. It will prove useful when we contrast with the solution for probabilistic PCA.

Setting the gradient of (2.2) to zero and rearranging we find that at a stationary point we have

$$\mathbf{C} = n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}.$$

This can also be written as

$$\mathbf{C} = n^{-1}\sum_{i=1}^{n}\hat{\mathbf{y}}_{i,:}\hat{\mathbf{y}}_{i,:}^\top = n^{-1}\sum_{i=1}^{n}\left(\mathbf{y}_{i,:}-\boldsymbol{\mu}\right)\left(\mathbf{y}_{i,:}-\boldsymbol{\mu}\right)^\top.$$

In statistics this is known as the sample covariance (it is the sample based approximation to the covariance). So if the covariance matrix of the Gaussian model is unconstrained then a stationary point in the likelihood solution is given by setting the covariance to the sample covariance of the data, $\mathbf{C} = n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$. Further analysis can be used to show that this stationary point is a unique maximum of the likelihood. In other words, for the multivariate Gaussian density the maximum likelihood solution corresponds to setting the mean to the sample mean and the covariance to the sample covariance. This actually corresponds to a technique known as moment matching—match the first moment of the density (the mean) to the sample based approximation of the first moment. Then match the second moment (given by $\left\langle\mathbf{Y}^\top\mathbf{Y}\right\rangle$) of the density to the empirical value. This is an approach to fitting densities that Karl Pearson suggested. However, it is not always the case that the maximum likelihood solution corresponds to moment matching: the Gaussian is a special case where the first two moments are also the sufficient statistics. (see box 2.10).

Of course, our multivariate Gaussian density has the special form $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$. We do not have the freedom to set the covariance matrix equal to the sample covariance.

**boxfloat 2.11** The Multivariate Chain Rule

he chain rule is perhaps the mainstay of differential calculus,

$$\frac{\mathrm{d}x}{\mathrm{d}z} = \frac{\mathrm{d}x}{\mathrm{d}y} \times \frac{\mathrm{d}y}{\mathrm{d}z}.$$

where $x$ is assumed to a function of $y$ and $y$ is a function of $z$. The multivariate chain rule is applied when a function has several parameters, each of which is dependent on some shared parameter. Imagine $x$ is a function of a set of $y$s, $\{y_i\}$, and each $y_i$ depends on $z$, then we have

$$\frac{\mathrm{d}x}{\mathrm{d}z} = \sum_i \frac{\partial x}{\partial y_i} \times \frac{\mathrm{d}y_i}{\mathrm{d}z},$$

where $\frac{\partial \cdot}{\partial \cdot}$ represents the partial derivative. In matrix calculus the derivative $\mathrm{d}E/\mathrm{d}\mathbf{C}$ represents the partial derivatives of $E$ with respect to each element of $\mathbf{C}$. The matrix $\mathrm{d}\mathbf{C}/\mathrm{d}\sigma^2$ is the matrix formed by the derivative of each element of $\mathbf{C}$ with respect to $\sigma^2$. Applying the multivariate chain rule to this system gives

$$\frac{\mathrm{d}E}{\mathrm{d}\sigma^2} = \mathrm{tr}\left(\frac{\mathrm{d}E}{\mathrm{d}\mathbf{C}}\frac{\mathrm{d}\mathbf{C}}{\mathrm{d}\sigma^2}\right)$$

where the two matrices of derivatives are being multiplied together inside the trace.

### 2.6.1 Optimizing With Respect to W and $\sigma^2$

To find the optimal mapping matrix and noise variance we need to make use of the chain rule for matrix derivatives. Let's consider the noise variance first. This is a scalar quantity. We are interested in $\mathrm{d}E(\mathbf{W},\sigma^2)/\mathrm{d}\sigma^2$ and we have $\mathrm{d}E(\mathbf{W},\sigma^2)/\mathrm{d}\mathbf{C}$: the gradient with respect to each element of the covariance matrix. We can use the multivariate chain rule to compute $\mathrm{d}E(\mathbf{W},\sigma^2)/\mathrm{d}\sigma^2$: we need to find how each element of the covariance matrix varies with $\sigma^2$, $\mathrm{d}\mathbf{C}/\mathrm{d}\sigma^2$. We then multiply this matrix elementwise by $\mathrm{d}E(\mathbf{W},\sigma^2)/\mathrm{d}\mathbf{C}$ and sum all the elements to get the result. Computation of the gradient of the covariance matrix with respect to the noise variance is relatively trivial,

$$\frac{\mathrm{d}\mathbf{C}}{\mathrm{d}\sigma^2} = \mathbf{I}.$$

Multiplying these gradients elementwise and summing across the elements simply leads to the trace,

$$\frac{\mathrm{d}E(\mathbf{W},\sigma^2)}{\mathrm{d}\sigma^2} = \mathrm{tr}\left(\frac{\mathrm{d}E(\mathbf{W},\sigma^2)}{\mathrm{d}\mathbf{C}}\right).$$

Computing the gradients with respect to the mapping matrix is slightly more involved. To develop them through the chain rule we need to introduce an approach for computing the derivative of one matrix, $\mathbf{C}$ with respect to another, $\mathbf{W}$. Details of how this is achieved are given in appendix **??**. Here we will simply consider the result,

$$\frac{\mathrm{d}E(\mathbf{W},\sigma^2)}{\mathrm{d}\mathbf{W}} = \frac{n}{2}\mathbf{C}^{-1}\mathbf{W} - \frac{1}{2}\mathbf{C}^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\mathbf{C}^{-1}\mathbf{W}.$$

We again need to look for stationary points of the system which implies that

$$\mathbf{W} = n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\mathbf{C}^{-1}\mathbf{W}.$$

Note that we can identify the sample based covariance matrix in this equation, for convenience we represent it by $\mathbf{S} = n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$. Applying the matrix inversion lemma to $\mathbf{C}$ then gives,

$$\mathbf{W} = \mathbf{S}\left[\sigma^{-2}\mathbf{W} - \sigma^{-4}\mathbf{W}\left(\mathbf{I} + \sigma^{-2}\mathbf{W}^\top\mathbf{W}\right)^{-1}\mathbf{W}^\top\mathbf{W}\right]. \qquad (2.3)$$

Tipping and Bishop, 1999b showed how we can solve this system for the mapping matrix. The first step of the solution is to consider the singular value decomposition of the mapping matrix (see box 2.12),

$$\mathbf{W} = \mathbf{U}_q\mathbf{L}_q\mathbf{R}^\top,$$

substituting into (2.3) we can write

$$\mathbf{U}_q\mathbf{L}_q\mathbf{R}^\top = \mathbf{S}\mathbf{U}_q\left[\sigma^2\mathbf{I} + \mathbf{L}^2\right]^{-1}\mathbf{L}_q\mathbf{R}^\top.$$

post-multiplying both sides by $\mathbf{R}\mathbf{L}_q^{-1}\left[\sigma^2\mathbf{I} + \mathbf{L}_q^2\right]$ we recover

$$\mathbf{U}_q\left[\sigma^2\mathbf{I} + \mathbf{L}_q^2\right] = \mathbf{S}\mathbf{U}_q$$

this is recognized as the matrix form of an eigenvalue problem (see box **??**). It implies that stationary points are found when $\mathbf{U}_q$ are eigenvectors of the sample based covariance matrix, $\mathbf{S}$. The associated eigenvalues are then given by $\sigma^2\mathbf{I} + \mathbf{L}_q^2$. We cannot recover the rotation matrix as all its appearances have canceled. This makes sense as $\mathbf{W}$ only appears in the likelihood in the form $\mathbf{W}\mathbf{W}^\top = \mathbf{U}_q\mathbf{L}_q^2\mathbf{U}_q^\top$. So the likelihood is insensitive to our choice of $\mathbf{R}$. It is common practice to set $\mathbf{R} = \mathbf{I}$, but it should be borne in mind that the solution is valid for any rotation matrix.

The eigenvalue decomposition of the sample covariance matrix has the form

$$\mathbf{U}_q\mathbf{\Lambda} = \mathbf{S}\mathbf{U}_q,$$

which implies that $\ell_{i,i}^2 + \sigma^2 = \lambda_{i,i}$, implying that $\ell_{i,i} = \sqrt{\lambda_{i,i} - \sigma^2}$ can be derived from the eigenvalues of the sample covariance and the noise variance, $\sigma^2$. Let's now return to the estimation of the noise variance.

## 2.7 Fixed Point Equation for $\sigma^2$

The stationary points of $\mathbf{W}$ are derived through the eigenvectors and eigenvalues of the sample covariance matrix. Any set of $q$ eigenvector/value pairings will be a stationary point but which $q$ eigenvectors should we choose to retain? We can order the eigenvector/value pairs so that those that we retain are the first $q$. Then we can index the retained pairings by $i = 1 \ldots q$. To determine which pairings these are we first need to return to estimation of the noise variance, $\sigma^2$.

Substituting in the The gradient with respect to $\sigma^2$ was expressed in terms of the sample covariance, $\mathbf{S}$, and the mapping matrix, $\mathbf{W}$. Specifically it was given by the trace of the gradient with respect to the covariance matrix, as given by (2.2). First let's derive an alternative representation of the covariance matrix. Substituting the solution for $\mathbf{W}$, we can write the covariance as

$$\mathbf{C} = \mathbf{U}_q \mathbf{L}_q^2 \mathbf{U}_q + \sigma^2 \mathbf{I}.$$

We now define a new matrix $\mathbf{L} \in \Re^{p \times p}$. We set the first $q$ diagonal elements of $\mathbf{L}$ to be given by, $\{\ell_{i,i}\}_{i=1}^{q}$. The remaining diagonal elements are set to zero. This allows us to introduce the $p \times p$ matrix of eigenvectors of $\mathbf{S}$, $\mathbf{U}$. We can then write $\mathbf{U}_q \mathbf{L}_q^2 \mathbf{U}_q^\top = \mathbf{U}\mathbf{L}^2\mathbf{U}^\top$. Now, since $\mathbf{U}\mathbf{U}^\top = \mathbf{I}$ then we can write,

$$\mathbf{C} = \mathbf{U}\left(\mathbf{L}^2 + \sigma^2 \mathbf{I}\right)\mathbf{U}^\top, \tag{2.4}$$

and the corresponding inverse is given by

$$\mathbf{C}^{-1} = \mathbf{U}\left(\mathbf{L}^2 + \sigma^2 \mathbf{I}\right)^{-1}\mathbf{U}^\top.$$

Combining this representation with $\hat{\mathbf{Y}}^\top \hat{\mathbf{Y}}$ we can express the gradient of the trace component of the error function as

$$\mathbf{C}^{-1}\hat{\mathbf{Y}}^\top \hat{\mathbf{Y}}\mathbf{C}^{-1} = n\mathbf{U}\left(\mathbf{L}^2 + \sigma^2 \mathbf{I}\right)^{-2}\mathbf{\Lambda}\mathbf{U}^\top$$

by substituting the sample covariance matrix, $n^{-1}\hat{\mathbf{Y}}^\top \hat{\mathbf{Y}}$ with its eigenvalue decomposition, $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ we can write the trace of this matrix as

$$\mathrm{tr}\left(\mathbf{C}^{-1}\hat{\mathbf{Y}}^\top \hat{\mathbf{Y}}\mathbf{C}^{-1}\right) = \frac{n}{2}\sum_{i=1}^{q}\frac{\lambda_{i,i}}{(\ell_{i,i}^2 + \sigma^2)^2} + \frac{n}{2}\sum_{i=q+1}^{p}\frac{\lambda_{i,i}}{\sigma^4}.$$

This needs to be combined with the trace of $\mathbf{C}^{-1}$ to get the gradient with respect to $\sigma^2$. So we need

$$\frac{n}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\right) = \frac{n}{2}\sum_{i=1}^{q}\frac{1}{\ell_{i,i}^2 + \sigma^2} + \frac{n}{2}\frac{p-q}{\sigma^2}.$$

Within these trace terms we first substitute the solution for $\ell_{i,i} = \sqrt{\lambda_{i,i} - \sigma^2}$. Then we can write down the resulting gradient with respect to $\sigma^2$,

$$\frac{\mathrm{d}E(\mathbf{W}, \sigma^2)}{\mathrm{d}\sigma^2} = \frac{n}{2}\frac{p-q}{\sigma^2} - \frac{n}{2}\sum_{i=q+1}^{p}\frac{\lambda_{i,i}}{\sigma^4}.$$

A stationary point is given when

$$\sigma^2 = \frac{1}{p-q}\sum_{i=q+1}^{p}\lambda_{i,i}.$$

In other words the stationary point is when $\sigma^2$ is the average of the discarded eigenvectors.

## 2.8 The Global Optimum

We now have stationary points for all the constituent parts of the covariance matrix. However, we still haven't determined what type of stationary points they are. Tipping and Bishop, 1999b substituted the stationary points into the log likelihood. Using the same representation of $\mathbf{C}$ as we described in (2.4) we can write the negative log likelihood at these stationary points. First though, we add an additional term: the log determinant of the sample covariance matrix. This has no effect on the optimum with respect to our parameters, but it will lead to a nice interpretation of the objective,

$$E = -\frac{n}{2}\log|\mathbf{C}| + \frac{n}{2}\log|\mathbf{S}| + \frac{n}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\mathbf{S}\right)$$

This can be rewritten as

$$E = -\frac{n}{2}\log\left|\mathbf{C}^{-1}\mathbf{S}\right| + \frac{n}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\mathbf{S}\right).$$

Using our decomposition of the covariance matrix at the stationary points we have

$$\mathbf{C}^{-1}\mathbf{S} = \mathbf{U}(\mathbf{L}^2 + \sigma^2\mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{U}^\top.$$

The first $q$ eigenvalues of this matrix are therefore given by $\frac{\lambda_{i,i}}{\ell_{i,i}^2+\sigma^2}$ which at the stationary points we have derived is equal to 1. The remaining eigenvalues are given by $\frac{\lambda_{i,i}}{\sigma^2}$ which at the stationary point we derived for $\sigma^2$ are equal to $(p-q)\frac{\lambda_{i,i}}{\sum_{j=q+1}^{p}\lambda_{i,i}}$. In other words this matrix has eigenvalues given by 1 for the first $q$ dimensions, and by $\frac{(p-q)\lambda_i}{\sum_{i=q+1}^{p}\lambda_i}$ for the remaining eigenvalues. The error function can be written entirely in terms of these eigenvalues because both the trace and the log determinant can be expressed as a sum of eigenvalues and log eigenvalues respectively.

$$
\begin{aligned}
E =& \frac{n(p-q)}{2} \log \left( \prod_{i=q+1}^{p} \frac{\lambda_i}{\sum_{j=q+1}^{p} \lambda_j} \right)^{\frac{1}{p-q}} + \frac{n(p-q)}{2} \sum_{i=q+1}^{p} \frac{\lambda_i}{\sum_{j=q+1}^{p} \lambda_j} \\
&+ \frac{n(p-q)}{2} \log(p-q)
\end{aligned}
$$

$$
E() = -\frac{1}{2} \sum_{i=1}^{q} \log \lambda_i - \frac{p-q}{2} \log \frac{\sum_{i=q+1}^{p} \lambda_i}{p-q} + \frac{q}{2} + \frac{(p-q)}{2} \sum_{i=q+1}^{p} \frac{\lambda_i}{\sum_{j=q+1}^{p} \lambda_j}
$$

where we have used the fact that the log determinant of a matrix is the sum of the log-eigenvalues and the trace of a matrix is the sum of its eigenvalues.

***Describe why it is a global maximum.***

## 2.9 Posterior Density for the Latent Positions

Given the maximum likelihood solution, we can compute the posterior density for given data point, $\mathbf{y}_{i,:}$, in the latent space, $p(\mathbf{x}_{i,:}|\mathbf{y}_{i,:}, \mathbf{W}, \boldsymbol{\mu}, \sigma^2)$. Incorporating the maximum likelihood solution for $\boldsymbol{\mu}$ with the data to center it, we write this as $p(\mathbf{x}_{i,:}|\hat{\mathbf{y}}_{i,:}, \mathbf{W}, \sigma^2)$. We compute this density through Bayes' rule,

$$
p(\mathbf{x}_{i,:}|\hat{\mathbf{y}}_{i,:}, \mathbf{W}, \sigma^2) = \frac{p(\hat{\mathbf{y}}_{i,:}|\mathbf{x}_{i,:}, \mathbf{W}, \sigma^2) p(\mathbf{x}_{i,:})}{p(\hat{\mathbf{y}}_{i,:}|\mathbf{W}, \sigma^2)}.
$$

The prior is Gaussian and conjugate to the likelihood (box 2.5), we can use the integration trick given in tip 2.1 to compute the posterior, this involves us collecting all terms which depenend on $\mathbf{x}_{i,:}$. Up to a constant the log posterior is then given by,

$$
\log p(\mathbf{x}_{i,:}|\hat{\mathbf{y}}_{i,:}, \mathbf{W}, \sigma^2) = \sigma^{-2} \hat{\mathbf{y}}_{i,:}^\top \mathbf{W} \mathbf{x}_{i,:} - \frac{1}{2} \mathbf{x}_{i,:}^\top (\sigma^{-2} \mathbf{W}^\top \mathbf{W} + \mathbf{I}) \mathbf{x}_{i,:} + \text{const},
$$

which is a quadratic form. We can find the full posterior distribution by completing the square of the quadratic form, this shows us that the posterior for $\mathbf{x}_{i,:}$ is Gaussian,

$$
p(\mathbf{x}_{i,:}|\hat{\mathbf{y}}_{i,:}, \mathbf{W}, \sigma^2) = \mathcal{N}\left(\mathbf{x}_{i,:}|\boldsymbol{\mu}_{x_i}, \mathbf{C}_x\right), \tag{2.5}
$$

**boxfloat 2.12** The Singular Value Decomposition

he singular value decomposition of a matrix decomposes it in the following form,

$$\mathbf{W} = \mathbf{U}_q \mathbf{L}_q \mathbf{R}^\top$$

where the matrix we are decomposing is assumed to be $\mathbf{W} \in \Re^{p\times q}$ with $p \geq q$. The matrix $\mathbf{U}_q \in \Re^{p\times q}$ is constrained so that $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ and $\mathbf{L} \in \Re^{q\times q}$ is a diagonal matrix of the "singular values". Finally $\mathbf{R} \in \Re^{q\times q}$ is an orthonormal matrix which we can think of as a rotation, $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$.
We can think of the singular values, $\mathbf{L}$, as an axis aligned $q$-dimensional basis which is rotated into higher dimensions, $p$, by the matrix $\mathbf{U}_q$. Within the $q$-dimensional space, we can also rotate the basis by $\mathbf{R}$.
The singular value decomposition can represent arbitrary matrices, but there is an important relationship between it and the eigendecomposition of a positive semi-definite matrices in the following way. A positive semi-definite matrix may be formed by the product of two unconstrained matrices,

$$\mathbf{C} = \mathbf{W}\mathbf{W}^\top$$

substituting the singular value decomposition of $\mathbf{W}$ we have

$$\mathbf{C} = \mathbf{U}_q \mathbf{L}^2 \mathbf{U}_q^\top.$$

The rotation matrix, $\mathbf{R}$, disappears from this operation. This can be seen as an eigenvalue decomposition of the positive definite matrix (see box 2.13). We could also construct a positive definite matrix the other way around,

$$\boldsymbol{\Sigma} = \mathbf{W}^\top\mathbf{W} = \mathbf{R}^\top \mathbf{L}^2 \mathbf{R}$$

which is full rank and has eliminated $\mathbf{U}_q$ instead of $\mathbf{R}$.

**boxfloat 2.13** The Eigenvalue Decomposition of a Positive Definite Matrix

**boxfloat 2.14** Pseudoinverse of a Matrix

he inverse of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^{-1}$ and ensures that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$. The inverse only exists if the matrix is *square*. For rectangular matrices we make use of a *pseudoinverse*. If the matrix has $m$ rows and $n$ columns, where $m \geq n$ then the pseudoinverse is given by

$$\mathbf{A}^{+} = (\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}.$$

This matrix has the property that $\mathbf{A}^{+}\mathbf{A} = \mathbf{I}$.

with a covariance matrix given by $\mathbf{C}_x = (\sigma^{-2}\mathbf{W}^{\top}\mathbf{W} + \mathbf{I})^{-1}$, and mean vector given by $\boldsymbol{\mu}_{x_i} = \sigma^{-2}\mathbf{C}_x\mathbf{W}^{\top}\mathbf{y}_{i,:}$.

To visualize the data projected into the latent space, we can look at the mean projection into the latent space. Substituting the form of the covariance matrix into our expression for the mean we recover the projection for each point as

$$\boldsymbol{\mu}_{x_i} = (\mathbf{W}^{\top}\mathbf{W} + \sigma^2\mathbf{I})^{-1}\mathbf{W}^{\top}\mathbf{y}_{i,:}.$$

Note that as the noise variance goes towards zero, $\sigma^2 \to 0$, this mapping simply involves projecting to the latent space using the *pseudoinverse* (box 2.14) of $\mathbf{W}$,

$$\lim_{\sigma^2 \to 0} \boldsymbol{\mu}_{x_i} = (\mathbf{W}^{\top}\mathbf{W})^{-1}\mathbf{W}^{\top}\mathbf{y}_{i,:}.$$

So once the principal components are computed we can visualize the data set, $\hat{\mathbf{Y}}$ by projecting it on to the expected latent positions,

$$\langle \mathbf{X} \rangle = \hat{\mathbf{Y}}\mathbf{W}(\mathbf{W}^{\top}\mathbf{W} + \sigma^2\mathbf{I}).$$

Of course, since $\mathbf{W}$ is recovered only up to a rotation, $\mathbf{R}$, the latent positions can also only be recovered up to a rotation.

## 2.10 Summary of Probabilistic PCA

1. You have a data set with $n$ points and $p$ features in a matrix $\mathbf{Y} \in \Re^{n \times p}$.
2. Decide on a latent dimensionality, $q$.
3. Center your data set, $\hat{\mathbf{Y}} = \mathbf{Y}\mathbf{H}$.
4. Compute the empirical covariance from the centred data $\mathbf{S} = \frac{1}{n}\hat{\mathbf{Y}}^{\top}\hat{\mathbf{Y}}$.
5. Compute the eigenvalues, $\boldsymbol{\Lambda}$, and corresponding eigenvectors, $\mathbf{U}$ of the covariance, $\mathbf{C}$.
6. Discard the columns of the eigenvector matrix associated with the smallest $p-q$ eigenvalues, leaving a rectangular matrix $\mathbf{U}_q \in \Re^{p \times q}$ of eigenvectors.

7. Set $\sigma^2$ to the average of the discarded eigenvalues,
$$\sigma^2 = \frac{1}{p-q}\sum_{i=q+1}^{p}\lambda_i.$$
8. Set $\mathbf{W} = \mathbf{U}_q(\mathbf{\Lambda} - \sigma^2\mathbf{I})^{-\frac{1}{2}}$.
9. Visualize the data set by plotting the posterior expectation,
$$\langle\mathbf{X}\rangle = \hat{\mathbf{Y}}\mathbf{W}(\mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I})^{-1}.$$

## 2.11 Why is a Probabilistic Interpretation Important?

We have presented a probabilistic interpretation of principal component analysis. The motivation for the model is very close to Hotelling's own description of principal component analysis, but why should we place such stock on the probabilistic interpretation of the model? Probabilistic models clarify the assumptions we are making for the generating process for the data. For probabilistic PCA we are assuming that low dimensional latent variables, $\mathbf{X}$, are sampled from a Gaussian density and linearly mapped into a higher dimensional space through a matrix, $\mathbf{W}$, before being corrupted by isotropic Gaussian noise. Any of these assumptions can be changed. For example, suggesting that non-Gaussian densities are used for the latent variables, but maintaining independence across the latent dimensions, leads to independent component analysis (ICA) [MacKay, 1996, Hyvärinen et al., 2001]. If the Gaussian noise has a different variance for each output dimension then factor analysis is recovered [Roweis and Ghahramani, 1999]. In later chapters we will consider nonlinear mappings between the latent variables and the data (see chapter 6–**??**. Probabilistic models provide a principled approach to dealing with missing data [Tipping and Bishop, 1999b]. They can also be easily be expanded in a number of formulaic ways: for example Bayesian treatments of parameters [Bishop, 1999a,b, Minka, 2001] and mixture models [Tipping and Bishop, 1999a, Ghahramani and Hinton, 1997, Ghahramani and Beal, 2000]. With the exception of ICA we will exploit all these characteristics of the model. First though we will apply standard probabilistic PCA to some of the data sets we introduced in the last chapter.

### 2.11.1 Expectation Maximization Algorithms

The probabilistic interpretation of principal component analysis was introduced simultaneously by Roweis [1998] and Tipping and Bishop [1999b]. Roweis, 1998 noted that the probabilistic interpretation leads to an expectation maximization algorithm (EM algorithm) for fitting the principal components. Expectation maximization [Dempster et al., 1977] is an approach to maximum likelihood when there are marginalized variables in the model. In the case of probabilistic PCA, the marginalized variables are the latent positions, $\mathbf{X}$. The model is defined through a joint distribution,

$$p(\mathbf{Y}|\boldsymbol{\theta}) = \int p(\mathbf{Y}, \mathbf{X}|\boldsymbol{\theta})\mathrm{d}\mathbf{X},$$

which in turn is given by the product of the likelihood and the prior,

$$p(\mathbf{Y}, \mathbf{X}|\boldsymbol{\theta}) = p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta})p(\mathbf{X})$$

where in the above we have collected all the parameters into $\boldsymbol{\theta} = \{\mathbf{W}, \sigma^2, \boldsymbol{\mu}\}$ to keep the notation simple.

EM algorithms decompose the log likelihood in the following way:

$$\log p(\mathbf{Y}|\boldsymbol{\theta}) = \int q(\mathbf{X}) \log \frac{p(\mathbf{Y}, \mathbf{X}|\boldsymbol{\theta})}{q(\mathbf{X})}\mathrm{d}\mathbf{X} + \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta})}\mathrm{d}\mathbf{X}.$$

where $q(\mathbf{X})$ is a probability density over $\mathbf{X}$. It is known as a variational density. We define the first term to be

$$\mathcal{L} = \int q(\mathbf{X}) \log \frac{p(\mathbf{Y}, \mathbf{X}|\boldsymbol{\theta})}{q(\mathbf{X})}\mathrm{d}\mathbf{X} \tag{2.6}$$

which is known as the *variational lower bound* on the log likelihood. The second term is recognized as the Kullback Leibler divergence (see box 2.7) between the density $q(\mathbf{X})$ and the posterior density $p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta})$.

$$\mathrm{KL}(q(\mathbf{X})|p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta})) = \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta})}\mathrm{d}\mathbf{X}.$$

The KL divergence is positive unless $q(\mathbf{X}) = p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta})$ when it is zero. In this case we can substitute $p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta})$ into (**??**) and recover the true likelihood,

$$\begin{aligned}\mathcal{L} &= \int p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta}) \log \frac{p(\mathbf{Y}, \mathbf{X}|\boldsymbol{\theta})}{p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta})}\mathrm{d}\mathbf{X} \\ &= \int p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta}) \log p(\mathbf{Y}|\boldsymbol{\theta})\mathrm{d}\mathbf{X} \\ &= \log p(\mathbf{Y}|\boldsymbol{\theta}).\end{aligned}$$

Setting $q(\mathbf{X}) = p(\mathbf{X}|\mathbf{Y}, \boldsymbol{\theta})$ maximizes $\mathcal{L}$ with respect to $q(\mathbf{X})$ and forces it to be equal to the true log likelihood. The EM algorithm exploits this relationship to perform a two stage optimization of the log likelihood. It does this by working with $\mathcal{L}$ instead of the likelihood directly. Firstly, $\mathcal{L}$ is maximized with respect to $q(\mathbf{X})$ by setting it equal to the posterior density for $\mathbf{X}$. Then $\mathcal{L}$ is then maximized with respect to the parameters, $\boldsymbol{\theta}$.

The EM algorithm for probabilistic PCA starts by setting $q(\mathbf{X})$ equal to the true posterior,

$$q(\mathbf{X}) = \prod_{i=1}^{n} p(\mathbf{x}_{i,:}|\hat{\mathbf{y}}_{i,:}, \mathbf{W}, \sigma^2).$$

For probabilistic PCA, this density factorizes across the data points and was computed for an individual data point in (2.5). We then maximize the variational lower bound, $\mathcal{L}$, with respect to the parameters, $\boldsymbol{\theta}$. The variational lower bound can be decomposed into two terms,

$$\mathcal{L} = \int q(\mathbf{X}) \log p(\mathbf{Y}, \mathbf{X}|\boldsymbol{\theta}) \mathrm{d}\mathbf{X} - \int q(\mathbf{X}) \log q(\mathbf{X}) \mathrm{d}\mathbf{X},$$

the second term is the *entropy* of the variational density $q(\mathbf{X})$, it doesn't depend on $\boldsymbol{\theta}$. The first term is known as the *variational free energy*, it does depend on $\boldsymbol{\theta}$. Maximization of the variational lower bound requires that we compute this first term, which is the expectation of the log of the joint density under the variational density,

$$\mathcal{L}(\boldsymbol{\theta}) = \int q(\mathbf{X}) \log p(\mathbf{Y}, \mathbf{X}|\boldsymbol{\theta}) \mathrm{d}\mathbf{X} + \text{const.}$$

For probabilistic PCA, the expectation of the log joint density is

$$\langle \log p(\mathbf{Y}, \mathbf{X}|\boldsymbol{\theta}) \rangle = -\frac{1}{2\sigma^2} \mathrm{tr} \left( \hat{\mathbf{Y}}^\top \hat{\mathbf{Y}} - 2\hat{\mathbf{Y}}\mathbf{W} \left\langle \mathbf{X}^\top \right\rangle + (\sigma^2 \mathbf{I} + \mathbf{W}^\top \mathbf{W}) \left\langle \mathbf{X}^\top \mathbf{X} \right\rangle \right) - \frac{np}{2} \log \sigma^2 + \text{const.}$$

where the expectations under the variational density can be computed as

$$\begin{aligned} \langle \mathbf{X} \rangle &= \hat{\mathbf{Y}}\mathbf{W}(\mathbf{W}^\top \mathbf{W} + \sigma^2 \mathbf{I}) \\ \left\langle \mathbf{X}^\top \mathbf{X} \right\rangle &= n(\mathbf{W}^\top \mathbf{W} + \sigma^2 \mathbf{I})^{-1} + \langle \mathbf{X} \rangle^\top \langle \mathbf{X} \rangle . \end{aligned}$$

Optimizing with respect to $\mathbf{W}$ we recover the following update

$$\mathbf{W} = \hat{\mathbf{Y}}^\top \langle \mathbf{X} \rangle \left\langle \mathbf{X}^\top \mathbf{X} \right\rangle^{-1}$$

similarly we can optimize with respect to $\sigma^2$ recovering

$$\sigma^2 = \frac{1}{np} \mathrm{tr} \left( \hat{\mathbf{Y}}^\top \hat{\mathbf{Y}} - 2\hat{\mathbf{Y}}\mathbf{W} \left\langle \mathbf{X}^\top \right\rangle + \mathbf{W}^\top \mathbf{W} \left\langle \mathbf{X}^\top \mathbf{X} \right\rangle \right)$$

which is more interpretable if we write it out as

$$\sigma^2 = \frac{1}{np} \left\langle \sum_{i=1}^{n} \sum_{j=1}^{p} \hat{\mathbf{y}}_{i,j} - \mathbf{w}_{j,:}^\top \mathbf{x}_{i,j} \right\rangle,$$

where we can see that it is the expected average error across each output.

Figure 2.3: The EM algorithm visualized. Plot shows the log likelihood with respect to one parameter, $w_{1,1}$, of the probabilistic PCA algorithm. The steps of the EM algorithm are visualized by plotting the lower bound, $\mathcal{L}$ to the log likelihood at each step, and showing where parameter $w_{1,1}$ moves to. The difference between the true likelihood and the lower bound at each point is the KL divergence between the true posterior and the approximating variational density.

## 2.12 Example Applications of Probabilistic PCA

### 2.12.1 Robot Wireless Data

### 2.12.2 Stick Man Data

### 2.12.3 Spellman Data

***Use to describe the dual version of the eigenvalue problem.***

### 2.12.4 Oil Data

#### 2.12.4.1 Missing Values

One advantage of the dual approach to probabilistic PCA is it leads immediately to a one shot principled approach to dealing with missing data in principal component analysis.

The likelihood of the model is given by

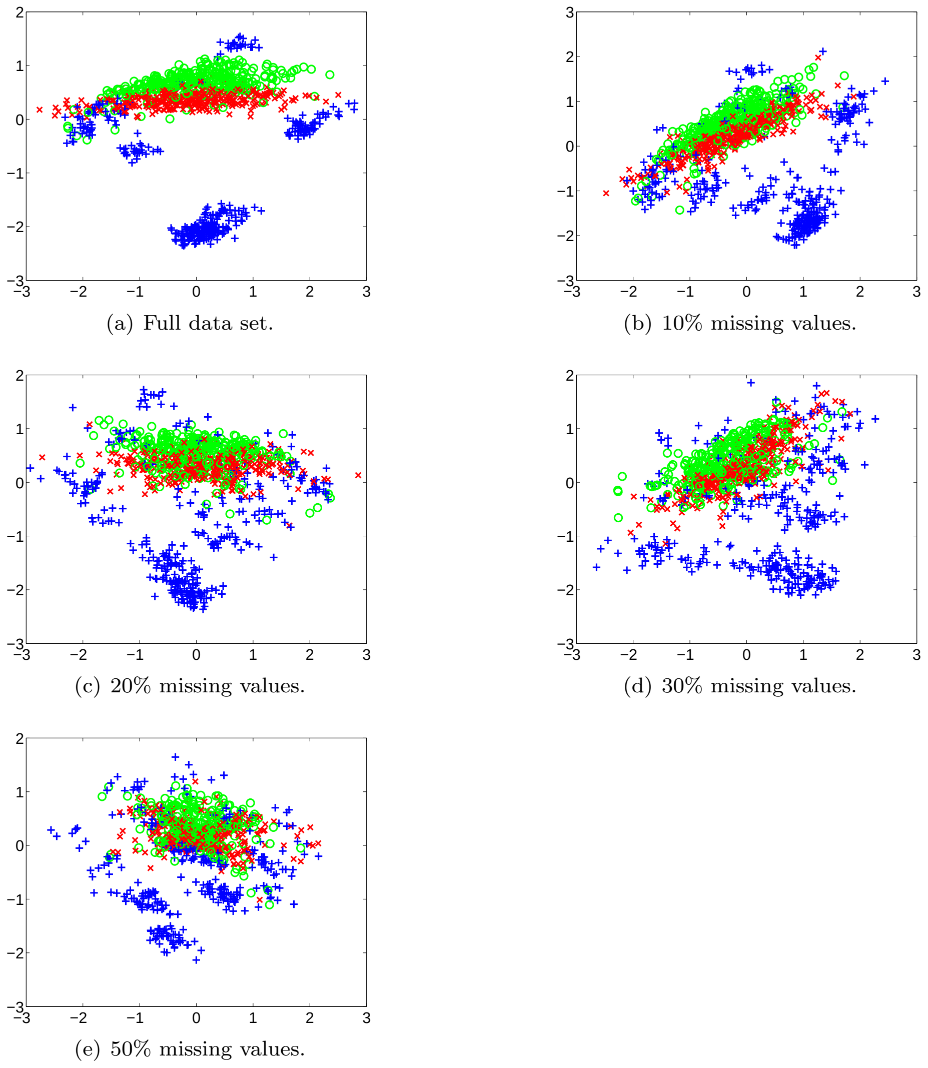
$$p(\mathbf{Y}|\mathbf{X}) = \frac{p}{2}\log|\mathbf{J}| - \sum_{j=1}^{p} \mathrm{tr}\left(\mathbf{y}_{:,j}\mathbf{Y}_{:,j}^{\top}\mathbf{J}_j\right)$$

where $\mathbf{J}_j = \sigma^{-2}\mathbf{I} - \sigma^{-2}\mathbf{X}_j(\mathbf{X}_j^{\top}\mathbf{X}_j + \sigma^2)^{-1}\mathbf{X}_j$ is the inverse of $\mathbf{X}\mathbf{X}^{\top} + \sigma^2\mathbf{I}$ with the missing data columns removed.

## 2.13 Factor Analysis

***Mention that factor Analysis leads to $\mathbf{C} = \mathbf{W}\mathbf{W}^{\top} + \mathbf{D}$, $\mathbf{D} = diag(\mathbf{d})$ — not discussed further.***

Derivation of factor analysis from RCA point of view and EM.



(a) Full data set.

(b) 10% missing values.

(c) 20% missing values.

(d) 30% missing values.

(e) 50% missing values.

Figure 2.4: Projection of the oil data set on to $q = 2$ latent dimensions using the probabilistic PCA model. Different plots show various proportions of missing values. All values are missing at random from the design matrix $\mathbf{Y}$.

## 2.14 Mixtures of Linear Models

Local minima issues

introduce through prototypes with linear pertubations.

Derive EM algorithm.

Mention global coordination issues.

## 2.15 Bayesian Principal Components

Variational approximation to the model

### 2.15.1 Probabilistic Matrix Factorization

Massively missing data.

## 2.16 Conclusions

So far we have shown that data can behave very non-intuitively in high dimensions. Fortunately, most data is not intrinsically high dimensional. We showed that for particular low rank structures of covariance matrix, data sampled from a Gaussian is not inherently high dimensional. The probabilistic PCA model underlies this low rank structure, when fitted to data it can exploit the linear low dimensional structure in the data. However, the linear constraint is a strong one. In the next section we will consider an alternative approach to dimensionality reduction. It involves distance matching between inter-point distances in the latent space and those in the data space. In statistics this approach is known as multidimensional scaling. As a first step we will show how the model can be related to principal component analysis. Then, in the next chapter, we shall use the resulting framework: known as principal coordinate analysis, to motivate non-linear extensions to linear dimensionality reduction models.

# Chapter 3

# Dimensionality Reduction through Distance Matching

In the last chapter we introduced principal component analysis through a generative interpretation of the model. The approach had much in common with Hotelling's original model, but through the addition of a noise term it led to a fully probabilistic interpretation for PCA. We've related this motivation to factor analysis where the assumption is that a set of low dimensional factors are responsible for explaining the data. In this chapter we switch tacks. We consider a different approach to dimensionality reduction: distance matching.

## 3.1 Inter-point Squared Distance Matching

The probabilistic interpretation associated with probabilistic PCA brings many associated advantages. It allows the model to be extended within the probabilistic framework, by for example mixture models [Tipping and Bishop, 1997, Ghahramani and Hinton, 1997] and Bayesian approaches [Bishop, 1999a,b, Minka, 2001]. We also saw above how a probabilistic interpretation facilitates the problem of missing data. However, the probabilistic model we described in the previous chapter requires us to consider explicitly the mapping between the low dimensional data and our observations, we constrained this mapping to be linear. In this chapter we will first highlight the need for approaches which allow for nonlinear relationships between the latent and data space. We will consider a class of approaches known as multidimensional scaling for performing this dimensionality reduction.

In chapter **??** we described how the dimensionality of data could be reduced through explicitly modeling a linear relationship between a low dimensional latent space and the observed data space. We referred to this as the generative approach to dimensionality reduction. A range of related approaches to dimensionality reduction such as independent component analysis and factor analysis can be seen as generative models, but our main focus was on the probabilistic interpretation of principal components analysis. In this chapter we will consider the alternative approach of inter-point squared distance matching. The basic idea is to compute the inter-point squared distances of the data set, $\mathbf{D}$. By computing the squared distances of the latent configurations, $\boldsymbol{\Delta}$, and moving the latent points, $\mathbf{X}$, to ensure these distances match the data distances we can find a reduced dimensional representation of the data. To motivate this approach we will consider an artificial example based around a handwritten six.

## 3.2 Rotated Sixes

In chapter **??** we discussed the use of mixtures of Gaussian densities for modelling data. The motivation was that the center of each Gaussian density represented a "prototype" data point that was then corrupted through various transformations. The corruption of the points was modeled by Gaussian densities. Here we would like to consider a more realistic corruption of a prototype.

Let's consider an image of the handwritten digit 6 (image br1561_6.3.pgm), taken from the ***USPS ??*** data. The image has 64 rows by 57 columns, giving a data point which is living in a $p = 3,648$ dimensional space. Let's consider a simple model for handwritten 6s. We model each pixel independently with a binomial distribution (box **??**). Each pixel is the result of a single binomial trial.

$$p(\mathbf{y}) = \prod_{j=1}^{p} \pi^{y_j}(1-\pi)^{(1-y_j)},$$

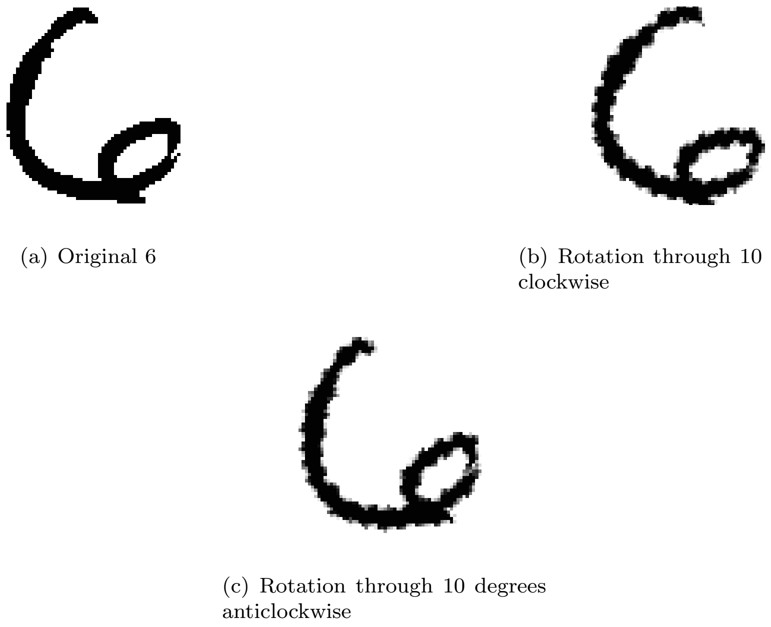
where the probability across pixels are taken to be independent and are governed by the same probability of being on, $\pi$. The maximum likelihood solution for this parameter is given by the number of pixels that are on in the data point divided by the total number of pixels (see box 3.1):

$$\pi = \frac{849}{3,648}.$$

The probability of recovering the original six in any given sample is then given by[1] ***Add number of ones***

$$p(\mathbf{y} = \text{given image}) = \pi^{849}(1-\pi)^{3,648-849} = 2.67 \times 10^{-860}.$$

[1]Note this is not the binomial distribution which gives probabilities for total number of successes in a series of binary trials, here we don't only need a precise number of successes, we also need the successes to occur at the right pixels.


(a) Original 6

(b) Rotation through 10 degrees clockwise

(c) Rotation through 10 degrees anticlockwise

Figure 3.1: The prototype 6 taken from the ***USPS*** data along with two example rotations of the digit. The original image has been converted to 64×64 to match dimensionality of the rotated images.

This implies that even if we sampled from this model every nanosecond between now and the end of the universe[2] we would still be highly unlikely to see the original six. Our expected waiting time would be $10^{217}$ years and there is only a $fill$ probability of seeing it before the universe ends[3]. This is clearly a very poor generative model of the six. The independence assumption across features (just as in chapter **??**) is a very poor one. Let's consider an alternative model for handwritten sixes. We generate a data set in the following way: we take the prototype 6 and rotate the image 360 times, by one degree for reach rotation. This corruption of the prototype 6 is designed to reflect the sort of corruptions that real prototypes might undergo. A rotated 6 is still a 6, and although it is not realistic to consider such drastic rotations as we have applied, it will be helpful in visualizing the corruption of the data. In figure 3.1 we show the original digit and rotations of 10 degrees clockwise and anticlockwise.

In figure **??** we take all 360 examples of the rotated digits and visualize these data using principal component analysis. and linearly project them down to a two dimensional space (using the posterior given in Section 2.9). Selected

[2]We follow MacKay [2003] in assuming that will be in approximately *number* years.
[3]This can be computed with the geometric distribution.

**boxfloat 3.1** The Binomial Distribution

he binomial distribution is the special case of the multinomial where the outcome of the trials is binary (success or failure). It represents the distribution for a given number of successes in a set number of trials when the probability of success is constant. For a probability of success given by $\pi$ the binomial distribution has the form

$$p(n|N,\pi) = \frac{N!}{n!(N-n)!}\pi^{n}(1-\pi)^{N-n}$$

where $n$ is the number of successes and $N$ is the total number of trials. The term $\frac{N!}{n!(N-n)!}$ is known as the binomial coefficient and represents the number of ways in which $n$ successes can occur in $N$ trials (for example 1 success in 2 trials can occur either by success then failure or by failure then success. That's two ways: $\frac{2!}{(1!\times(2-1)!)} = 2$). A special case of the binomial is the when $N=1$. Then it is sometimes known as the Bernoulli distribution,

$$p(s|\pi) = \pi^{s}(1-\pi)^{1-s}.$$

For a model of binary pixels in an image that assumes each pixel is independent we have

$$p(\mathbf{y}|\pi) = \prod_{j=1}^{p}\pi^{y_j}(1-\pi)^{1-y_j}$$

where $\mathbf{y}$ is a vector of binary pixel values from the image and the image has $p$ pixels. The negative log likelihood is

$$E(\pi) = -\sum_{j=1}^{p} y_j \log \pi - \sum_{j=1}^{p}(1-y_j)\log(1-\pi)$$

and gradients with respect to $\pi$ can be computed as

$$\frac{\mathrm{d}E(\pi)}{\mathrm{d}\pi} = \frac{1}{\pi}\sum_{j=1}^{p} y_j - \frac{1}{1-\pi}\sum_{j=1}^{p}(1-y_j)$$

which can be re-expressed using the fact that the number of pixels that are on is $p_1 = \sum_{j=1}^{p} y_j$ and the number of off pixels is $p - p_1 = \sum_{j=1}^{p}(1-y_j)$,

$$\frac{\mathrm{d}E(\pi)}{\mathrm{d}\pi} = \frac{p_1}{\pi} - \frac{p-p_1}{1-\pi}$$

setting to zero and rearranging we recover

$$\pi = \frac{p_1}{p-p_1}1-\pi$$

at a stationary point. Further rearrangement leads to

$$\pi = \frac{\frac{p_1}{p-p_1}}{\left(1+\frac{p_1}{p-p_1}\right)} = \frac{p_1}{p}$$

which turns out to be a global minimum of the negative log likelihood. To see this we simply compute the second derivative of the negative log likelihood with respect to $\pi$,

$$\frac{\mathrm{d}^2E(\pi)}{\mathrm{d}\pi^2} = -\frac{p_1}{\pi^2} - \frac{p-p_1}{(1-\pi)^2},$$

which will always be negative if $0 \leq \pi \leq 1$ as it must be. This implies the negative log likelihood is convex and the stationary point is a global minimum.

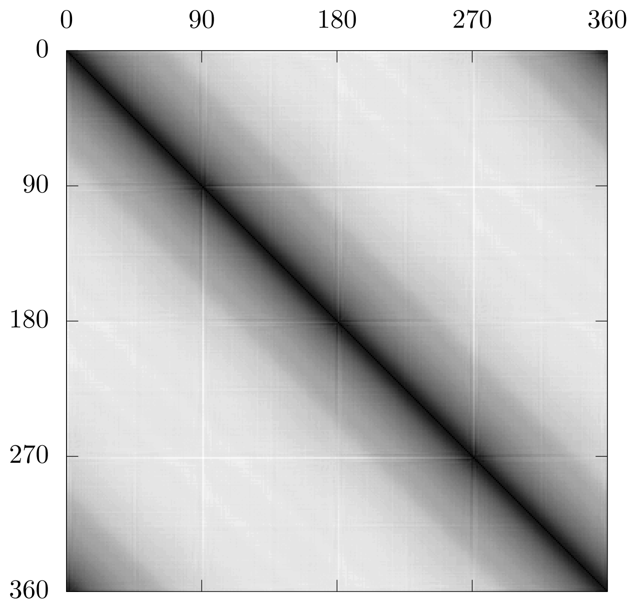


Figure 3.2: Inter-point squared distances for the rotated digits data. Much of the data structure can be seen in the matrix. All points are ordered by angle of rotation. We can see that the distance between two points with similar angle of rotation is small (note in the upper right and lower left corners the low distances associated with 6s rotated by roughly 360 degrees and unrotated 6s.

examples from the data are also placed on the plot near their corresponding projected position. What we see makes intuitive sense. The data, projected into the two dimensional space, proscribes a circle. The data set is inherently one dimensional. The dimension of the data is associated with the rotation transformation. A pure rotation would lead to a pure circle. In practice rotation of images requires some interpolation and this leads to small corruptions of the latent projections away from the circle.

This inspires an alternative approach to dimensionality reduction: can we find a configuration of points, $\mathbf{X}$, such that the normalized squared distance between each latent point,

$$\frac{\delta_{i,j}}{q} = \frac{1}{q} \left\|\mathbf{x}_{i,:} - \mathbf{x}_{j,:}\right\|_2^2$$

closely matches the corresponding normalized squared distance $\frac{d_{i,j}}{p}$ in the data space? We can do this by defining an objective function that depends on the

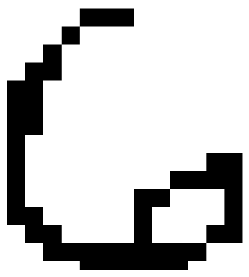


Figure 3.3: Subsampling the digit six. Here we have subsampled the digit to 1/16th of the size. We took every fourth row and every fourth column from the original digit to generate this image. The form of the six is still clearly visible. The new image has 16 rows and 15 columns.

matrix of squared distances in the latent space, $\mathbf{\Delta} = (\delta_{i,j})_{i,j}$, and the matrix of squared distances in the data space, $\mathbf{D} = (d_{i,j})_{i,j}$.

## 3.3 Feature Selection

Let's start by considering a very simple approach to dimensionality reduction. Given a $p$-dimensional data set, $\mathbf{Y}$, we will just select a $q$-dimensional subset of the columns of $\mathbf{Y}$ and retain these for our latent matrix, $\mathbf{X}$. To motivate this approach, let's uniformly sample our handwritten six and visualize the result. In figure 3.3 we show the six sub-sampled down to an image with 16 rows and 15 columns ($p = 240$).

We can see from the figure that the main elements of a handwritten six are still clearly visible. For this example, we sub-sampled both the columns of our image and the rows of the image uniformly (every 4th value), mainly so that we can view the resulting image. In practice though, we would want to select the columns of $\mathbf{Y}$ which preserve the structure of the original data set. Next we will show how, for a particular distance based objective function, the best features to retain are those with the largest variance.

To decide which $q$ features to retain we define an objective function that measures the absolute error between the squared distances in the latent space and those in the data space. We represent this through an entrywise $L_1$ norm on average difference between squared distances

$$E(\mathbf{X}) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \left\| \frac{d_{i,j}}{p} - \frac{\delta_{i,j}}{q} \right\|_1 .$$

The pre-factor $\frac{1}{n(n-1)}$ accounts for the number of non-zero entries in the squared distance matrices that are being compared: there are $n^2$ entries, but the $n$ diagonal entries will always be zero. For a given number of retained features, $q$, we will look to minimize this error. We can do this by selecting for $\mathbf{X}$, in turn, the column from $\mathbf{Y}$. We can repeat this process until we have the desired number of features for our latent representation.

It will turn out that to minimize $E(\mathbf{Y})$ we need to compose $\mathbf{X}$ by extracting the columns of $\mathbf{Y}$ which have the largest variance.

The squared distance in the data space can be expressed as

$$d_{i,j} = \sum_{k=1}^{p} \left(y_{i,k} - y_{j,k}\right)^2.$$

Since we can re-order the columns of $\mathbf{Y}$ without affecting the distances we choose an ordering which is such that the first $q$ columns of $\mathbf{Y}$ are the those that will best represent the distance matrix. Each selected feature will also be scaled by a factor given by $\ell_k$. This allows us to replace $\mathbf{X}$ with columns from $\mathbf{Y}$. We perform the substitution $\mathbf{x}_{:,k} = \ell_k \mathbf{y}_{:,k}$ for all $k = 1 \ldots q$. This means we can express the squared distances in latent space using

$$\delta_{i,j} = \sum_{k=1}^{q} \left(x_{i,k} - x_{j,k}\right)^2 = \sum_{k=1}^{q} \ell_k^2 \left(y_{i,k} - y_{j,k}\right)^2$$

Using this form for the squared latent distances we can rewrite the objective function as

$$\begin{aligned} E(\mathbf{X}) =& \frac{1}{n(n-1)} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{q} \left\| \left(\frac{1}{p} - \frac{\ell_k^2}{q}\right) \left(y_{i,k} - y_{j,k}\right)^2 \right\|_1 \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=q+1}^{p} \frac{1}{p} \left(y_{i,k} - y_{j,k}\right)^2. \end{aligned} \tag{3.1}$$

First of all we optimize with respect to the scale parameter, $\ell_k$. This scale parameter only appears in the first term of (3.1). Because of the L1 norm around this term, its minimum must be non-negative. We can see by inspection that this term can be set to zero by setting the scale parameter to $\ell_k = \sqrt{\frac{q}{p}}$ minimizing the first term. This leaves us with

$$E(\mathbf{X}) = \frac{1}{pn(n-1)} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=q+1}^{p} \left(y_{i,k} - y_{j,k}\right)^2.$$

In other words, the minimum of the objective is dependent on the squared distance between points that is associated with the columns that we will choose to discard. We need to minimize this quantity. To do this we first introduce the mean of each dimension, $\mu_k = \frac{1}{n}\sum_{i=1}^{n} y_{i,k}$, and center each data point. This can be done without effecting the overall objective function,

$$E(\mathbf{X}) = \frac{1}{pn(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=q+1}^{p} \left((y_{i,k} - \mu_k) - (y_{j,k} - \mu_k)\right)^2 .$$

Expanding the brackets gives

$$E(\mathbf{X}) = \frac{1}{pn(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=q+1}^{p} \Big( (y_{i,k} - \mu_k)^2 + (y_{j,k} - \mu_k)^2 \\ - 2(y_{j,k} - \mu_k)(y_{i,k} - \mu_k) \Big).$$

Bringing the sums over the data points inside the sum over the discarded dimensions we have

$$E(\mathbf{X}) = \frac{1}{p(n-1)} \sum_{k=q+1}^{p} \Big( \sum_{i=1}^{n} (y_{i,k} - \mu_k)^2 + \sum_{j=1}^{n} (y_{j,k} - \mu_k)^2 \\ - \frac{2}{n} \sum_{j=1}^{n} (y_{j,k} - \mu_k) \sum_{i=1}^{n} (y_{i,k} - \mu_k) \Big).$$

Using the fact that $\sum_{j=1}^{n} y_{j,k} = \sum_{j=1}^{n} \mu_k$ we can remove the last term inside the brackets

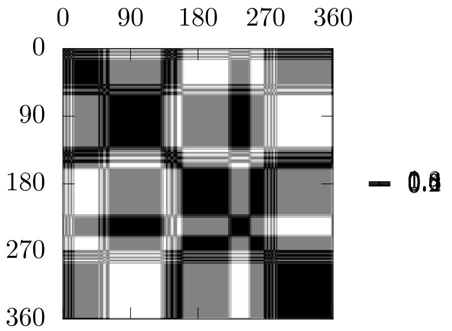
$$E(\mathbf{X}) = \frac{1}{p(n-1)} \sum_{k=q+1}^{p} \left( \sum_{i=1}^{n} (y_{i,k} - \mu_k)^2 + \sum_{j=1}^{n} (y_{j,k} - \mu_k)^2 \right).$$

If we estimate the variance[4] of the $k$th column of $\mathbf{Y}$ to be $\varsigma_k^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_{i,k} - \mu_k)^2$ we can re-express the error as a function of the sum of the sample variance for each discarded column of $\mathbf{Y}$,

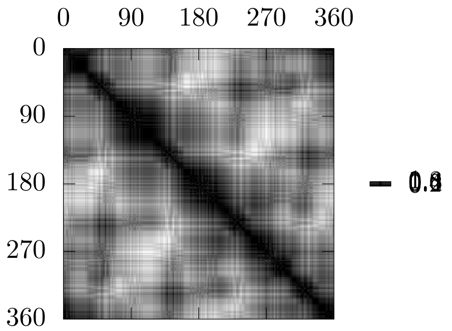
$$E(\mathbf{X}) = \frac{2}{p} \sum_{k=q+1}^{p} \varsigma_k^2.$$

This error can be minimized by discarding the $p - q$ columns with the smallest sample variance: in other words we need to retain the $q$ columns with the largest sample variance. These columns need to be retained and scaled by $\sqrt{\frac{q}{p}}$ to compose the matrix $\mathbf{X}$.

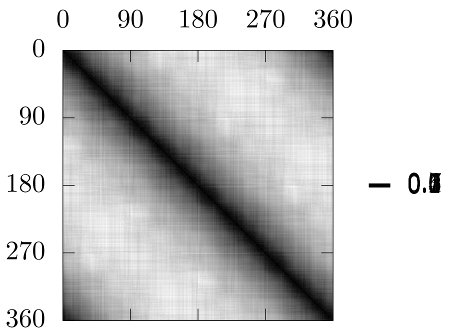
[4]Here, for convenience we are using the "unbiased" estimator of the variance which uses $n - 1$ in the denominator. Generally we will use the maximum likelihood estimator, which uses $n$ alone.



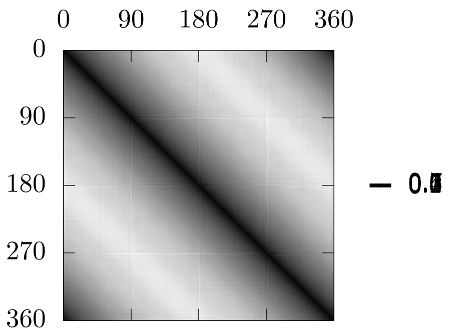
(a) Distances reconstructed with two dimensions. MAE: 0.215.



(b) Distances reconstructed with 10 dimensions. MAE: 0.214.



(c) Distances reconstructed with 100 dimensions. MAE: 0.203.



(d) Distances reconstructed with 1000 dimensions. MAE: 0.109.

Figure 3.4: Reconstruction of inter-point squared distances for the rotated six data where feature selection is used to reduce the dimensionality.

### 3.3.1 Feature Selection for Rotated Sixes

Our feature selection algorithm is quite simple: compute the variance of each feature (column) of $\mathbf{Y}$. Retain the $q$ features with the largest variance. Applying this to the rotated six example, let's see how good a job the features we select do when representing the data space distances. In figure 3.4 we've imaged $\boldsymbol{\Delta}$ for different values of $q$. We can see that it isn't until we select a considerable number of features (around $q = 1000$) that the finer features of the distance matrix are recreated. In particular the grid lines at 90, 180, and 270 degrees are not visible until 1000 features are selected. We also show the mean absolute error between the true distances and the latent distances in the caption of each plot. It drops relatively slowly from 0.215 to 0.109.

As an approach to dimensionality reduction, feature selection doesn't seem very encouraging. To reflect the finer structure of the original data we seem to need to retain around 1000 dimensions: for this many dimensions our latent dimensional space is itself very high dimensional. In the next section we will further develop the approach to distance matching by rotating the features before we select them. This implies a linear transformation of the features to the latent space: rotation followed by scaling and selection. The added flexibility of the rotation may allow us to make a much better approximation to the squared distances with far fewer features.

## 3.4 Feature Extraction

To motivate rotations of the data set, we first note that rotating the data has no effect on the inter-point squared distance matrix. Consider a new data set $\mathbf{Y}' = \mathbf{Y}\mathbf{R}^\top$ where $\mathbf{R}$ is a $p \times p$ orthogonal matrix such that $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$. The inter-point squared distances for the original data can be written in matrix form as,

$$\mathbf{D} = \text{diag}\left(\mathbf{Y}\mathbf{Y}^\top\right)\mathbf{1}^\top - 2\mathbf{Y}\mathbf{Y}^\top + \mathbf{1}\text{diag}\left(\mathbf{Y}\mathbf{Y}^\top\right)$$

where diag $(\mathbf{A})$ is an operator that extracts the diagonal of matrix $\mathbf{A}$ as a column vector. Computation of the distances for the rotated data set gives

$$\mathbf{D}' = \text{diag}\left(\mathbf{Y}\mathbf{R}^\top\mathbf{R}\mathbf{Y}^\top\right)\mathbf{1}^\top - 2\mathbf{Y}\mathbf{R}^\top\mathbf{R}\mathbf{Y}^\top + \mathbf{1}\text{diag}\left(\mathbf{Y}\mathbf{R}^\top\mathbf{R}\mathbf{Y}^\top\right)^\top .$$

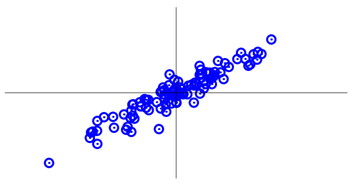
Using the fact that $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$ we can see that $\mathbf{D}' = \mathbf{D}$. So rotations of the features have no effect on the inter-point squared distances.[5] We take this as a license to perform any rotation we want before selecting features. Our plan is simple: rotate the data in such a way that the retained features for $\mathbf{X}$ best approximate the data's inter-point squared distance matrix, $\mathbf{D}$, through the latent matrix $\boldsymbol{\Delta}$,

$$\boldsymbol{\Delta} = \text{diag}\left(\mathbf{X}\mathbf{X}^\top\right)\mathbf{1}^\top - 2\mathbf{X}\mathbf{X}^\top + \mathbf{1}\text{diag}\left(\mathbf{X}\mathbf{X}^\top\right)^\top .$$

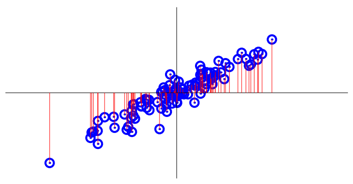
In the last section we consider dimensionality reduction by simply extracting features directly from the data for our latent variables. Ignoring the scale factor of $\sqrt{\frac{q}{p}}$ this is equivalent to projecting the data directly onto the features we retain. This is shown for a simple two dimensional data set projected onto one feature in figure 3.5.

The fact that we can rotate the data without changing the inter-point squared distances suggests the following alternative strategy: rotate the data to increase the variance associated with the retained dimensions, so that the discarded variance is smaller. This works because the *total variance*: the sum of the variances of the features remains constant under rotation. If more variance can be associated with retained features, less variance will be associated with discarded features. The general idea is shown in figure 3.6.

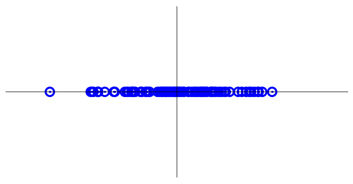
[5]Clearly this is also be true for latent squared distances when the latent variables are rotated. $\mathbf{X}' = \mathbf{X}\mathbf{R}^\top \rightarrow \boldsymbol{\Delta}' = \boldsymbol{\Delta}$. So any solution we find for $\mathbf{X}$ will be rotation invariant.



(a) Data with two features ($p = 2$) shown as a scatter plot.

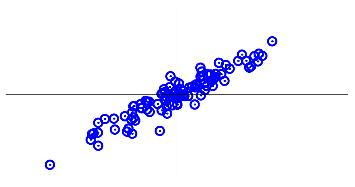


(b) Selecting the feature with the largest variance is equivalent to projecting the data onto the $x$-axis.

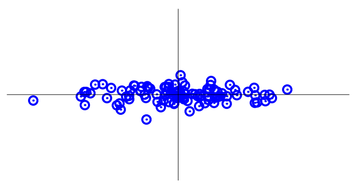


(c) The new representation of the data has only one feature.

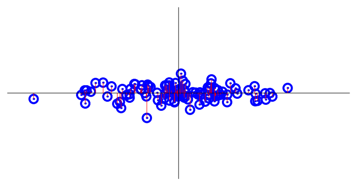
Figure 3.5: Feature selection via distance preservation. Data is having its dimensionality reduced through selecting the feature with the largest variance. Here data containing two features is projected down to data containing one feature by selecting the feature with the largest variance to represent the data.



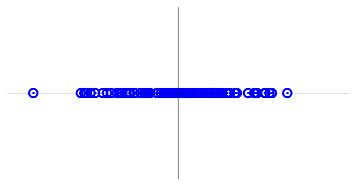
(a) Original data in two dimensions.



(b) Data rotated so that direction of maximum variance is now axis aligned.



(c) Lines show projections of data onto the principal axis.



(d) Data is projected down to a single dimension.

Figure 3.6: Rotation of the data features better preserves the inter-point squared distances. Here data is rotated onto its direction of maximum variations and then the data dimension is reduced to one.

### 3.4.1 Which Rotation?

More formally we are interested in rotating the data set so we can minimize the residual error associated with retaining features of the data. We can use our algorithm for discarding features, we showed in the last section that we should retain features with the maximum variance, the error is then given by the sum of residual variances,

$$E(\mathbf{X}) = \frac{2}{p}\sum_{k=q+1}^{p} \varsigma_k^2.$$

The total variance of a data set is defined as the sum of the variances of each feature,

$$\varsigma_{\text{total}}^2 = \sum_{k=1}^{p} \varsigma_k^2 = (n-1)^{-1}\text{tr}\left(\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\right)$$

where $\hat{\mathbf{Y}}$ is the centered data matrix so that the maximum likelihood estimate of the sample covariance is given by $\mathbf{S} = n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$. From the properties of the trace (see box 2.9)we know that rotations of the data have no effect on the total variance

$$\text{tr}\left(\mathbf{R}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\mathbf{R}\right) = \text{tr}\left(\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\right).$$

If we are looking to discard the directions associated with the smallest variance then we need to retain the directions associated with the maximum variance as the total variance will be conserved under rotation. We will now develop the algorithm that rotates the data to find the directions of maximum variance. We will operate directly on the sample covariance matrix $\mathbf{S} = n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$ of the data and will extract each direction in turn.

### 3.4.2 Maximum Variance Rotations

We want to rotate the data such that our first extracted dimension $\mathbf{x}_{:,1} = \sqrt{\frac{q}{p}}\hat{\mathbf{Y}}\mathbf{r}_{:,1}$, has the maximum variance. Where $\mathbf{r}_{:,1}$ is the first column of the rotation matrix, $\mathbf{R}$. We can find the maximum variance direction by looking for

$$\mathbf{r}_{:,1} = \text{argmax}_{\mathbf{r}_{:,1}}\text{var}\left(\hat{\mathbf{Y}}\mathbf{r}_{:,\mathbf{1}}\right).$$

Since $\mathbf{r}_{:,1}$ is a vector from the rotation matrix we must also impose the constraint that it has unit length, in other words

$$\mathbf{r}_{:,1}^\top\mathbf{r}_{:,1} = 1.$$

We can write the variance of the latent variable in terms of the sample covariance. Since $\hat{\mathbf{Y}}$ is centered, then the latent direction $\mathbf{x}_{:,1}$ will also be centered. Its variance is given therefore given by $\text{var}(\mathbf{x}_{:,1}) = n^{-1}\mathbf{x}_{:,1}^\top\mathbf{x}_{:,1}$. This

**boxfloat 3.2** Lagrange Multipliers

Lagrange multipliers allow us to perform the optimization of a function under constraints. They do this through a clever trick. Let's say we are optimizing an error function, $E(\boldsymbol{\theta})$ under some constraint on the parameters. For example we might constrain the parameters such that $g(\boldsymbol{\theta}) = b$. We can force the solution to respect the constraint by introducing a new 'parameter' $\lambda$. We then write the objective function as

$$L(\boldsymbol{\theta}) = E(\boldsymbol{\theta}) + \lambda(g(\boldsymbol{\theta}) - b).$$

This function is known as a Lagrangian and the new parameter, $\lambda<$ is known as a Lagrange multiplier.The effect of this modified objective can be seen if we take the gradient with respect to $\lambda$,

$$\frac{\mathrm{d}L(\boldsymbol{\theta})}{\mathrm{d}\lambda} = g(\boldsymbol{\theta}) - b$$

looking for a fixed point where the gradient of the Lagrangian is zero gives us the following equation:

$$g(\boldsymbol{\theta}) = b$$

so the fixed points of the augmented system will impose the constraint we required. If multiple constraints are required further terms can be added,

$$L(\boldsymbol{\theta}) = E(\boldsymbol{\theta}) + \sum_i \lambda_i (g_i(\boldsymbol{\theta}) - b_i).$$

***Sometimes constraints are also imposed in the form of inequalities,*** $g_i(\boldsymbol{\theta}) \geq b$***. In this case the Karush Kahn Tucker conditions must be used.***

can be rewritten in terms of the centered data matrix by substituting with $\mathbf{x}_{:,1} = \sqrt{\frac{q}{p}}\hat{\mathbf{Y}}\mathbf{r}_{:,1}$

$$\mathrm{var}\left(\mathbf{x}_{:,1}\right) = n^{-1}\frac{q}{p}\left(\hat{\mathbf{Y}}\mathbf{r}_{:,1}\right)^{\top}\hat{\mathbf{Y}}\mathbf{r}_{:,1} = \frac{q}{p}\mathbf{r}_{:,1}^{\top}\underbrace{\left(n^{-1}\hat{\mathbf{Y}}^{\top}\hat{\mathbf{Y}}\right)}_{\text{sample covariance}}\mathbf{r}_{:,1} = \frac{q}{p}\mathbf{r}_{:,1}^{\top}\mathbf{S}\mathbf{r}_{:,1}.$$

Finding the direction of maximum variance in the data therefore consists of maximizing $\mathbf{r}_{:,1}^{\top}\mathbf{S}\mathbf{r}_{:,1}$ with respect to $\mathbf{r}_{:,1}$ under the constraint that the length of $\mathbf{r}_{:,1}$ is one. This requires that we introduce a Lagrange multiplier, $\lambda_1$, to enforce the constraint (see box 3.2). giving a *Lagrangian* of the form

$$L\left(\mathbf{r}_{:,1}, \lambda_1\right) = \mathbf{r}_{:,1}^{\top}\mathbf{S}\mathbf{r}_{:,1} + \lambda_1\left(1 - \mathbf{r}_{:,1}^{\top}\mathbf{r}_{:,1}\right).$$

The gradient of the Lagrangian with respect to $\mathbf{r}_{:,1}$ is given by

$$\frac{\mathrm{d}L\left(\mathbf{r}_{:,1}, \lambda_1\right)}{\mathrm{d}\mathbf{r}_{:,1}} = 2\mathbf{S}\mathbf{r}_{:,1} - 2\lambda_1\mathbf{r}_{:,1}$$

and a stationary point of this objective may be found by rearranging the equation to give

$$\mathbf{S}\mathbf{r}_{:,1} = \lambda_1\mathbf{r}_{:,1}. \tag{3.2}$$

**boxfloat 3.3** Eigenvalue Problems

n eigenvalue problem is a matrix equation of the form

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

where $\mathbf{A} \in \Re^{k\times k}$, $\lambda$ is a scalar and $\mathbf{u} \in \Re^{k\times 1}$ and is constrained to have unit length, $\mathbf{u}^\top\mathbf{u} = 1$. The word comes from the German mathematician: "eigen" in this sense may be thought to mean "characteristic". In other words these are characteristic vectors and values of the matrix $\mathbf{A}$. . The aim in an eigenvalue problem is to find that vector. If the matrix $\mathbf{A} \in \Re^{k\times k}$ then there will be up to $k$ different solutions for this eigenvalue problem. re are typically be several vectors in other words the idea is to find a vector, $\mathbf{u}$, which when multiplied by a matrix If the matrix is real and symmetric then the eigenvalues of the system will also be real.

This equation has the form of an eigenvalue problem (see box 3.3). By pre-multiplying the eigenvalue problem by $\mathbf{r}_{:,1}^\top$ and using the constraint that $\mathbf{r}_{:,1}^\top\mathbf{r}_{:,1} = 1$ we can see that

$$\lambda_1 = \mathbf{r}_{:,1}^\top\mathbf{S}\mathbf{r}_{:,1}.$$

So the variance of our rotated data is given by a scaled eigenvalue of $\mathbf{S}$: $\mathrm{var}\left(\mathbf{x}_{:,1}\right) = \frac{q}{p}\lambda_1$. The maximum variance must be given by the largest eigenvalue of $\mathbf{S}$. The associated eigenvector is then given by $\mathbf{r}_{:,1}$. This is the single direction that will best preserve the inter-point squared distances under the absolute error criterion.

Having extracted a new feature, $\mathbf{x}_{:,1}$ from the data that best preserves the inter-point squared distance in the absolute sense, we now need to find the next feature, given by $\mathbf{x}_{:,2} = \sqrt{\frac{q}{p}}\hat{\mathbf{Y}}\mathbf{r}_2$. Since we are rotating the data space, the vector $\mathbf{r}_{:,2}$ must be of unit length *and* orthogonal to $\mathbf{r}_{:,1}$ such that $\mathbf{r}_{:,1}^\top\mathbf{r}_2 = 0$. In fact, for all future features we extract, we must ensure that their associated directions, $\mathbf{r}_k$, are orthogonal to all the directions we have extracted before. In other words we need to ensure that for $j < k$ we have

$$\mathbf{r}_{:,j}^\top\mathbf{r}_{:,k} = \mathbf{0} \qquad \mathbf{r}_{:,k}^\top\mathbf{r}_{:,k} = 1.$$

This ensures that the full rotation matrix is orthonormal so $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$. To find these directions, we use a Lagrangian of the form

$$L\left(\mathbf{r}_{:,k}, \lambda_k, \boldsymbol{\gamma}\right) = \mathbf{r}_{:,k}^\top\mathbf{S}\mathbf{r}_{:,k} + \lambda_k\left(1 - \mathbf{r}_{:,k}^\top\mathbf{r}_{:,k}\right) + \sum_{j=1}^{k-1}\gamma_j\mathbf{r}_{:,j}^\top\mathbf{r}_{:,k}$$

where $\gamma_j$s are Lagrange multipliers that enforce the constraint that the $k$th direction is orthogonal to all the preceding directions. Gradients with respect to the $k$th direction can then be found as,

$$\frac{\mathrm{d}L\left(\mathbf{r}_{:,k}, \lambda_k\right)}{\mathrm{d}\mathbf{r}_{:,k}} = 2\mathbf{S}\mathbf{r}_{:,k} - 2\lambda_k\mathbf{r}_{:,k} + \sum_{j=1}^{k-1}\gamma_j\mathbf{r}_{:,j}$$

and a stationary point is recovered when this equation is set to zero,

$$\mathbf{0} = 2\mathbf{S}\mathbf{r}_{:,k} - 2\lambda_k \mathbf{r}_{:,k} + \sum_{j=1}^{k-1} \gamma_j \mathbf{r}_{:,j}.$$

Premultiplying this equation by $\mathbf{r}_{:,i}^{\top}$, where $i < k$, gives

$$\gamma_i = -2\mathbf{r}_{:,i}^{\top}\mathbf{S}\mathbf{r}_{:,k}. \tag{3.3}$$

If $i = 1$, then from the transpose of (3.2) we already know that $\mathbf{r}_{:,1}^{\top}\mathbf{S} = \lambda_1 \mathbf{r}_{:,1}^{\top}$ which we can substitute into (3.3) giving

$$\gamma_1 = -2\mathbf{r}_{:,1}^{\top}\mathbf{r}_{:,k} = 0$$

because $\mathbf{r}_{:,1}^{\top}\mathbf{r}_{:,k}$ is constrained to be zero. Given the fact that $\gamma_1$ is zero we can consider the stationary point when $k = 2$,

$$\mathbf{S}\mathbf{r}_{:,2} = \lambda_2 \mathbf{r}_{:,2},$$

which also has the form of an eigenvalue problem. Since the variance associated with the discarded directions must be minimized, and maximum variance direction has already been extracted, the eigenvector associated with the second direction must be that associated with the second highest eigenvalue.

This result can also substitute this result into (3.3) to recover that $\gamma_2 = 0$. We can then proceed to extract the direction associated with the third feature, $\mathbf{r}_{:,3}$. Accordingly we find that the third direction is associated with the 3rd highest eigenvalue. This process can be repeated until all $q$ directions are recovered with the result that the directions associated with $q$ highest eigenvalues should be used as features. We refer to these as the first $q$ *principal eigenvectors*. Premultiplying (**??**) by $\mathbf{r}_k^{\top}$ shows that the eigenvalues give the associated variances

$$\lambda_k = \mathbf{r}_{:,k}^{\top}\mathbf{S}\mathbf{r}_{:,k}.$$

The full matrix of latent variables is given by $\mathbf{X} = \sqrt{\frac{q}{p}}\hat{\mathbf{Y}}\mathbf{R}_q$ where $\mathbf{R}_q$ is the first $q$ principal eigenvectors of $\mathbf{S}$ in a $p \times q$ matrix. The resulting variances of the latent representation can then be given as a diagonal matrix in the form $\frac{q}{p(n-1)}\mathbf{R}_q^{\top}\hat{\mathbf{Y}}^{\top}\hat{\mathbf{Y}}\mathbf{R}_q = \frac{q}{p}\mathbf{\Lambda}_q$ where $\mathbf{\Lambda}_q$ is a diagonal matrix containing the first $q$ principal eigenvalues of $\mathbf{S}$. The mean absolute error between the latent squared distances and the data squared distances is then given by

$$E\left(\mathbf{X}\right) = \frac{2n}{p(n-1)} \sum_{k=q+1}^{p} \lambda_k,$$

where we have converted between the maximum likelihood estimate of the variance and the unbiased estimate of the variance with the factor $\frac{n}{n-1}$ which approaches unity as the number of data becomes large. Since the sum of all eigenvalues is given by the total variance and the total variance is given by the trace of the sample covariance matrix: $\varsigma^2_{\text{total}} = \frac{n}{n-1}\text{tr}\left(\mathbf{S}\right) = \frac{1}{n-1}\text{tr}\left(\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\right)$, then we can write the error in terms of the retained eigenvalues,

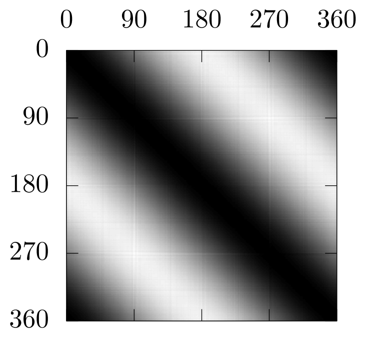
$$E\left(\mathbf{X}\right) = \frac{2n}{p(n-1)}\left(\text{tr}\left(\mathbf{S}\right) - \sum_{k=1}^{q}\lambda_k\right).$$

### 3.4.3 Principal Component Analysis

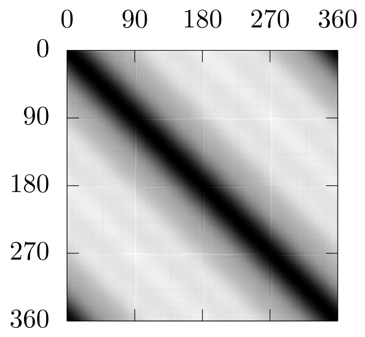
We saw in the last chapter that the process of performing an eigendecomposition on the sample covariance matrix of the data is known as principal component analysis: in the last chapter we had a probabilistic interpretation of the approach as a generative model. In this chapter the same solution arose from an exercise in matching the mean absolute error between squared distances in the latent space and the data space. Principal component analysis is the backbone of this book. Our aim is to relate approaches to dimensionality reduction to each other through how they relate to principal component analysis. As we've seen in this chapter and the last: two seemingly very different approaches to dimensionality reduction actually both lead to principal component analysis. In the next chapter we will start to consider nonlinear extensions of PCA, but before we do that we'll revisit the rotated sixes example and show how PCA can be applied in classical multidimensional scaling.

### 3.4.4 Rotation Reconstruction from Latent Space

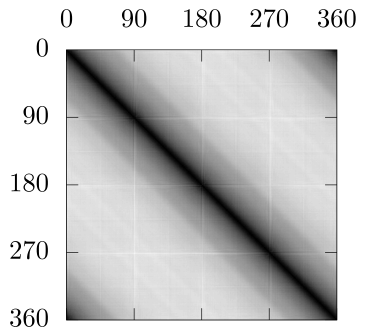
We now look at the rotated sixes data using the feature extraction techniques we've developed. Basically here we are finding the principal components of the data and computing the latent square distance matrices for different latent dimensions. These squared distance matrices are shown in figure 3.7. The reconstructions are much better than those achieved using feature selection. Even for two latent dimensions (which are in fact those plotted in figure **??** when introducing the example) we have a much more powerful representation of the data than 1000 dimensions of feature selection (note the grayscale is covering a much smaller range in these plots). The approach has already extracted salient features like the grid pattern at 90, 180 and 270 degrees and the cyclic nature of the data set. Note that we are unable to extract more than $q = 360$ dimensions because while the sample covariance matrix $\mathbf{S}$ is $p \times p$ its rank is upper bounded by the number of data, $n$, which is 360 for this example. This means that extracting 360 dimensions ***More detail on this***



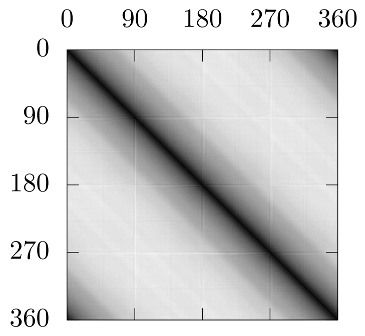
(a) Distances reconstructed with two dimensions. MAE: $3.30 \times 10^{-5}$.



(b) Distances reconstructed with 10 dimensions. MAE: $1.52 \times 10^{-5}$.



(c) Distances reconstructed with 100 dimensions. MAE: $3.85 \times 10^{-6}$.



(d) Distances reconstructed with 360 dimensions. MAE: 0000.

Figure 3.7: Reconstructed distance matrix for the artificial data set generated from latent variables by rotating the features before extracting them.

## 3.4.5 Principal Coordinates Analysis

We motivated distance matching as an approach to dimensionality reduction, and we introduced rotations of the data set to extract features of the largest variance to construct our latent representation. So far though, we have relied on direct access to the data matrix, $\mathbf{Y}$, to construct our latent representations through rotations, $\mathbf{X} = \sqrt{\frac{q}{p}}\mathbf{Y}\mathbf{R}_q$. However, there are situations where such access is not possible. In multidimensional scaling the assumption is that we only have access to the proximity information (such as a squared distance matrix) and not the actual data. A classical example is trying to determine the configuration of a given set of cities given road distances between the cities. We will look closely at such an example in Section 3.5.1, but first we need to find an approach to computing the eigenvectors of the sample covariance when the sample covariance is not available.

### 3.4.5.1 An Alternative Formalism

The matrix form of the eigenvalue problem for the first $q$ eigenvectors of the sample covariance is written in the form

$$\mathbf{S}\mathbf{R}_q = \mathbf{R}_q\boldsymbol{\Lambda}_q, \tag{3.4}$$

where $\mathbf{R}_q$ is a matrix in $\Re^{p\times q}$ for which $\mathbf{R}_q^\top\mathbf{R}_q = \mathbf{I}$. We can express the sample based covariance matrix in terms of the centered data matrices, $\hat{\mathbf{Y}}$, using the fact that $\mathbf{S} = \frac{1}{p}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$ as

$$\frac{1}{p}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\mathbf{R}_q = \mathbf{R}_q\boldsymbol{\Lambda}_q.$$

We now we premultiply both sides of this equation by the centered data matrix, giving

$$\frac{1}{p}\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\mathbf{R}_q = \hat{\mathbf{Y}}\mathbf{R}_q\boldsymbol{\Lambda}_q.$$

Postmultiplying both sides by the inverted square root of the eigenvalue matrix, $\boldsymbol{\Lambda}_q^{-\frac{1}{2}}$, gives

$$\frac{1}{p}\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\mathbf{R}_q\boldsymbol{\Lambda}_q^{-\frac{1}{2}} = \hat{\mathbf{Y}}\mathbf{R}_q\boldsymbol{\Lambda}_q^{-\frac{1}{2}}\boldsymbol{\Lambda}_q.$$

We now introduce the matrix $\mathbf{U}_q$ which we define to be equal to $\mathbf{U}_q = \hat{\mathbf{Y}}\mathbf{R}_q\boldsymbol{\Lambda}_q^{-\frac{1}{2}}$. This allows us to write

$$\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top\mathbf{U}_q = \mathbf{U}_q p\boldsymbol{\Lambda}_q \tag{3.5}$$

which again has the form of a matrix eigenvalue problem. If we can show that the matrix $\mathbf{U}_q$ *diagonalizes* the centered inner product matrix, $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$, then we know that $\mathbf{U}_q$ are the eigenvectors of $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$. To show this diagonalization, we pre- and postmultiply $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ by $\mathbf{U}_q^\top$ and $\mathbf{U}_q$ and substitute our definition for $\mathbf{U}_q$ giving

$$\mathbf{U}_q^\top\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top\mathbf{U}_q = \boldsymbol{\Lambda}_q^{-\frac{1}{2}}\mathbf{R}_q^\top\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\mathbf{R}_q\boldsymbol{\Lambda}_q^{-\frac{1}{2}}.$$

The central term can be rewritten using its eigenvalue decomposition. Because the full eigendecomposition is $\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}} = \mathbf{R}\boldsymbol{\Lambda}\mathbf{R}^\top$ and $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$ we can write $\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}} = p^2\mathbf{R}\mathbf{U}^2\mathbf{R}^\top$ we can write

$$\mathbf{U}_q^\top\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top\mathbf{U}_q = p^2\boldsymbol{\Lambda}_q^{-\frac{1}{2}}\mathbf{R}_q^\top\mathbf{R}\boldsymbol{\Lambda}^2\mathbf{R}^\top\mathbf{R}_q\boldsymbol{\Lambda}_q^{-\frac{1}{2}}.$$

which can be reexpressed as

$$\mathbf{U}_q^\top\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top\mathbf{U}_q = p^2\boldsymbol{\Lambda}_q^2$$

because the product of the first $q$ eigenvectors with the rest is given by

$$\mathbf{R}^\top\mathbf{R}_q = \begin{bmatrix} \mathbf{I}_q \\ \mathbf{0} \end{bmatrix} \in \Re^{p\times q}$$

where we have used $\mathbf{I}_q$ to denote a $q \times q$ identity matrix. When multiplied by the eigenvalues, $\boldsymbol{\Lambda}$, this matrix extracts the first $q$ eigenvalues,

$$\boldsymbol{\Lambda}\mathbf{R}^\top\mathbf{R}_q = \begin{bmatrix} \boldsymbol{\Lambda}_q \\ \mathbf{0} \end{bmatrix}.$$

This is expression is then multiplied by itself transposed giving

$$\mathbf{R}_q^\top\mathbf{R}\boldsymbol{\Lambda}^2\mathbf{R}^\top\mathbf{R}_q = \boldsymbol{\Lambda}_q^2.$$

Because $\mathbf{U}_q$ diagonalizes $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ giving a $q\times q$ diagonal matrix $p^2\boldsymbol{\Lambda}_q^2$ then we know that the first $q$ eigenvalues of $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ are $p\boldsymbol{\Lambda}_q$ and the corresponding principal eigenvectors are given by

$$\mathbf{U}_q = \hat{\mathbf{Y}}\mathbf{R}_q\boldsymbol{\Lambda}_q^{-\frac{1}{2}}.$$

By taking $q = p$ we can see that the two eigenvalue problems for $\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$ and $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ are fully equivalent.[6] One solves for the rotation that needs to be applied to the data space to find the latent space, $\mathbf{R}_q$ the other gives a solution related to the location of these data points once they are in latent space. Using the fact that $\mathbf{X} = \sqrt{\frac{q}{p}}\hat{\mathbf{Y}}\mathbf{R}_q$ we have

$$\mathbf{U}_q = \sqrt{\frac{p}{q}}\mathbf{X}\boldsymbol{\Lambda}_q^{-\frac{1}{2}}$$

[6]In this situation we have $\mathbf{U} = \hat{\mathbf{Y}}\mathbf{R}\boldsymbol{\Lambda}^{-\frac{1}{2}}$.

showing that $\mathbf{U}_q$ is equal to $\mathbf{X}$ with the columns scaled (since $\mathbf{\Lambda}_q^{-\frac{1}{2}}$ is diagonal it only serves to scale the columns of $\mathbf{X}$). In general when performing these eigenvalue problems, if $p < n$ it is more computationally efficient to solve for the rotation, $\mathbf{R}_q$. But when $p > n$ we solve for the latent variables. This is known as principal coordinate analysis: it can be seen as a dual form of principal component analysis. Note that the eigenvalues we obtain in each case are simply scaled versions of one another. This explains whey we could not extract more than 360 dimensions from the rotated six data we considered earlier in the chapter: the maximum number of non zero eigenvalues is given by the smaller of $n$ or $p$.

## 3.5 The Standard Transformation

In multidimensional scaling we may not know $\mathbf{Y}$ and so cannot compute $\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$, as we are given only a interpoint squared distance matrix. Can we compute $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ instead? It turns out we can, and this is known as the *standard transformation* between a similarity and a distance matrix. Given a similarity matrix, $\mathbf{K}$, the standard transformation relates the similarity matrix to a squared distance matrix in the following way,

$$d_{i,j} = k_{i,i} + k_{j,j} - 2k_{i,j}.$$

This is the squared distance between the two points according to the similarity matrix. This transformation is known as the standard transformation between a similarity and a distance [Mardia et al., 1979, pg 402]. As we will see later in the book it also has an interpretation as squared distance in a feature space or an expected distance between two points sampled from a correlated Gaussian.

If the similarity matrix is taken to be of the form $\mathbf{K} = \mathbf{Y}\mathbf{Y}^\top$ then $k_{i,j} = \mathbf{y}_{i,:}^\top\mathbf{y}_{j,:}$ and

$$d_{i,j} = \mathbf{y}_{i,:}^\top\mathbf{y}_{i,:} + \mathbf{y}_{j,:}^\top\mathbf{y}_{j,:} - 2\mathbf{y}_{i,:}^\top\mathbf{y}_{j,:} = \left\|\mathbf{y}_{i,:} - \mathbf{y}_{j,:}\right\|_2^2.$$

and we see that we recover standard squared Euclidean distance. To perform classical multidimensional scaling we take the squared distance matrix,

$$\mathbf{D} = \mathbf{1}\mathrm{diag}\left(\mathbf{Y}\mathbf{Y}^\top\right) - 2\mathbf{Y}\mathbf{Y}^\top + \mathrm{diag}\left(\mathbf{Y}\mathbf{Y}^\top\right)\mathbf{1}^\top$$

and center it giving

$$\mathbf{HDH} = -2\mathbf{HYYH} = -2\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top.$$

Multiplying by $-\frac{1}{2}$ then gives

$$-\frac{1}{2}\mathbf{HDH} = \hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$$

which is the inner product matrix formed from the centered data matrix. Since we know the relationship between the eigenvectors of this matrix and those of $\hat{\mathbf{Y}}^{\top}\hat{\mathbf{Y}}$ we can now perform principal component analysis multidimensional scaling by working directly with the squared Euclidean distance matrix, $\mathbf{D}$. The procedure is:

1. Compute the centered matrix $\mathbf{B} = -\frac{1}{2}\mathbf{H}\mathbf{D}\mathbf{H}$.
2. Compute the first $q$ principal eigenvalues, $\mathbf{\Lambda}_q$, of $\mathbf{B}$ and associated eigenvectors, $\mathbf{U}_q$.
3. Set the latent representation to be $\mathbf{X} = \sqrt{\frac{q}{p}}\mathbf{U}_q\mathbf{\Lambda}_q^{\frac{1}{2}}$
4. This is the configuration of points $\mathbf{X}$ that minimizes (3.1) under the optimal linear transformation from the original space.

For any other squared distance matrix we can follow the same procedure to recover a configuration of points that minimizes the same function. As long as the eigenvalues of the resulting $\mathbf{B}$ are all non-negative, the distances can be seen to have been computed in a Euclidean space.

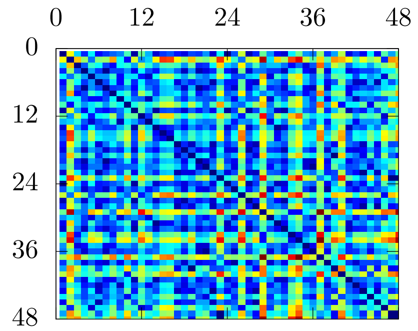
### 3.5.1 Example: Road Distances with Classical MDS

To illustrate the operation of classical multidimensional scaling we turn to a standard example given in many statistical texts Mardia et al. [see e.g. 1979]. We recreate this example by considering road distances between 53 cities in Europe. The data consists of the a distance matrix containing the distances between each of the cities. Our approach to the data is to first convert it to a similarity matrix using the standard transformation and perform an eigendecomposition on the resulting similarity matrix.

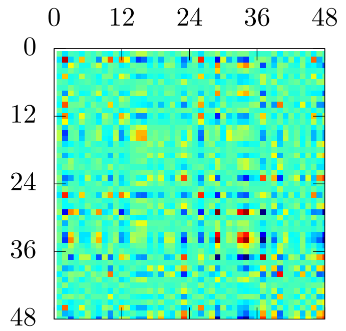
***Detailed description of the example.***

***Need to describe procrustes rotations***

In figure 3.10 we plot the eigenvalues of the similarity matrix. We note that in this case some of the eigenvalues are negative. This indicates that the distances in figure **??** cannot be embedded in an Euclidean space. There are two effects at play here. If we were to use shortest distances between the cities the curvature of the earth dictates that we wouldn't be able to flatten all distances onto a two dimensional plane. Secondly, the use of road distances prevents ...



(a) Road distances between European cities.



(b) Centered similarity matrix derived from the distance matrix.

Figure 3.8: Road distances can be converted to a similarity matrix using the "standard transformation". Our interpretation is that the similarity matrix is a covariance matrix from which the locations of the cities were sampled.

## 3.6 Summary

In summary multidimensional scaling is a statistical approach to creating a low dimensional representation of a data set through matching distances in the low dimensional space to distances in a high dimensional space. If our objective function is to minimize the absolute error between the squared distances in the two spaces, we can produce a low dimensional representation through an eigenvalue problem. This is known as classical multidimensional scaling. If the distances as computed in the data space are simply the Euclidean distance between each data point, $\mathbf{y}_{i,:}$, then classical MDS is equivalent to principle component analysis. This algorithm is generally referred to as principal coordinate analysis.

We also considered the classical from the statistical scaling literature: that of embedding a set of cities in a two dimensional space using the distances by road between the cities. This highlighted an issue with *ad hoc* specification of inter point distances: there is no guarantee that they can be embedded in a Euclidean space. Sets of distances that cannot be embedded in Euclidean spaces are associated with negative eigenvalues in their associated similarity matrices ***Need to check what Mardia says on this***.

We have given very little attention to how we derive the distance matrix, in the examples we gave we either considered Euclidean distances or road distances. The main focus of the developments in spectral approaches to dimensionality reduction have been in clever ways of specifying the distance/similarity matrix. We shall cover these in Chapter **??**. We have also paid scant regard to the choice

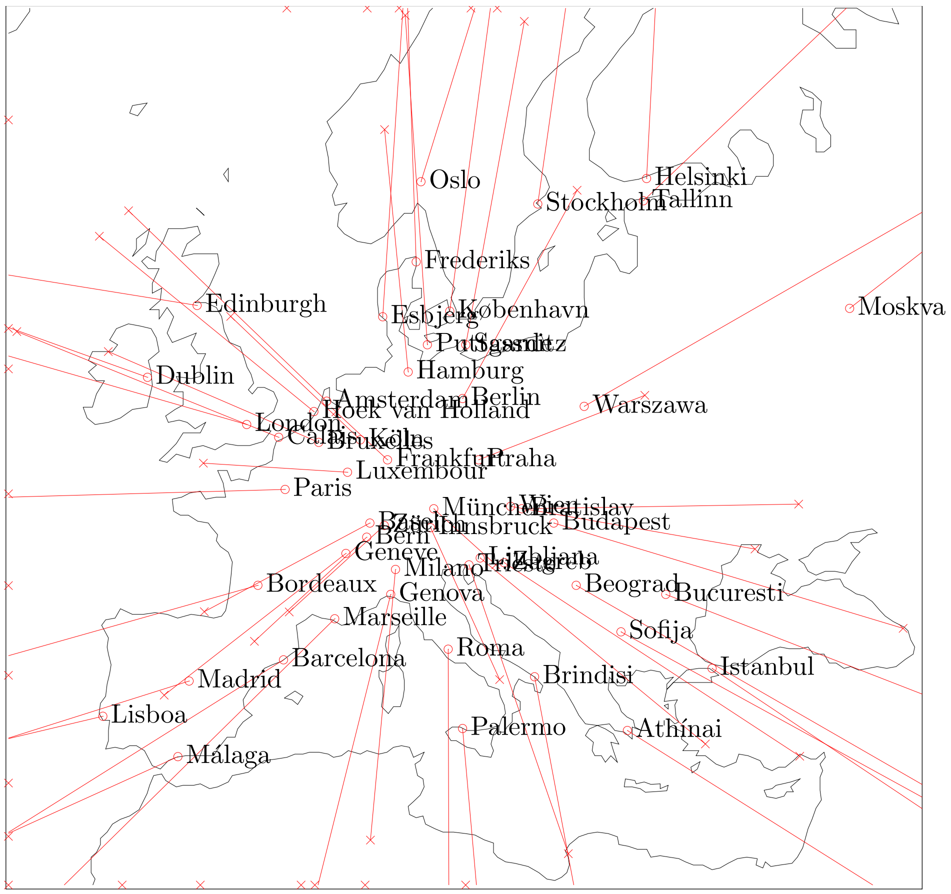


Figure 3.9: Reconstructed locations projected onto true map using Procrustes rotations.

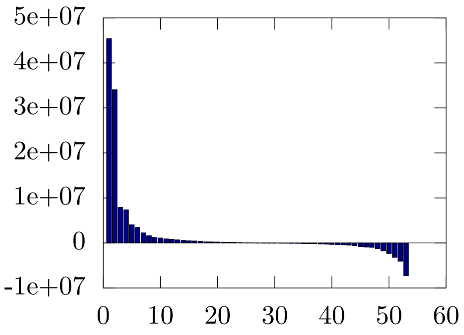
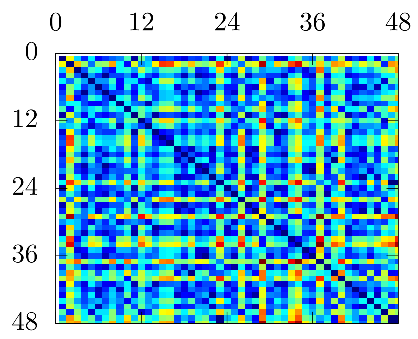
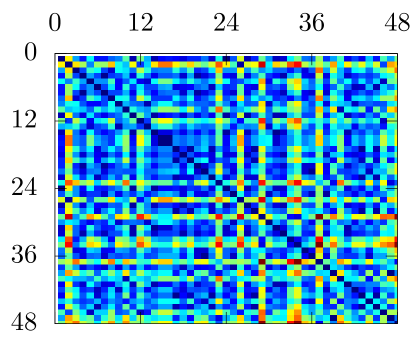


Figure 3.10: Eigenvalues of the similarity matrix are negative in this case.



(a) Original distance matrix.



(b) Reconstructed distance matrix.

Figure 3.11: Comparison between the two distance matrices, the original inter city distance matrix and the reconstructed distance matrix provided by the latent configuration of the data.

of objective function for distance matching. We will return to the objective function in Chapter **??**.

***relate to pca in previous chapter***

## 3.7 Notes

Multidimensional scaling: preserve a distance matrix.

#### 3.7.0.1 Classical MDS

A particular objective function

For Classical MDS distance matching is equivalent to maximum variance

Spectral decomposition of the similarity matrix

For Euclidean distances in $\mathbf{Y}$ space classical MDS is equivalent to PCA.

known as principal coordinate analysis (PCO)

Haven't discussed choice of distance matrix.

# Chapter 4

# Kernel PCA

**TODO: Chris's connection paper on metric MDS and kernel PCA should be discussed it may give further insight into why kernel PCA is performing dimensionality expansion and on the relation between classical MDS in a non-linear distance space and metric MDS in a linear distance space**.

In the last chapter we saw how classical multidimensional scaling can be applied to a data set consisting of distances or similarities. The classical scaling algorithm consists of three steps:

1. Square the distances in the inter point distance matrix to give a matrix $\mathbf{D}$.
2. Center the distance matrix by pre and postmultiplying by the centering matrix. Set the matrix $\mathbf{B}$ to negative 1/2 of the result. $\mathbf{B} = -\frac{1}{2}\mathbf{HDH}$.
3. Extract the principal $q$ eigenvectors from $\mathbf{B}$. Use these (appropriately scaled) to form the latent representation $\mathbf{X}$.

This algorithm gives the optimal linear dimensionality reduction of the data set under a criterion that measures the discrepancy in latent space interpoint distances with those in data space,

$$E(\mathbf{X}) = \sum_{i=1}^{n}\sum_{j=1}^{i-1}\left\|\frac{\delta_{i,j}}{q} - \frac{d_{i,j}}{p}\right\|_1$$

where $d_{i,j}$ and $\delta_{i,j}$ are the squared distances between data points $i$ and $j$ in data space and latent space respectively.

A weakness with the algorithm is it only gives a linear mapping from the space where the distances are computed to the latent space. Thus if we compute squared Euclidean distances directly in the data space, $d_{i,j} = \left\Vert \mathbf{y}_{i,:} - \mathbf{y}_{j,:} \right\Vert_2^2$, the algorithm provides a linear dimensionality reduction: indeed as we saw in the last chapter it is equivalent to principal component analysis and is often known as principal coordinate analysis.

A natural approach to ensuring we can recover a $\mathbf{X}$ which is nonlinearly related to $\mathbf{Y}$ is to compute squared distances in a space that is nonlinearly related to $\mathbf{Y}$. This can be achieved through the use of a set of basis functions.

We will make regular use of basis functions for constructing nonlinear algorithms. A brief review of basis functions is given in box 4.1.

Rather than computing interpoint squared distances directly in the data space, we can first map them to a nonlinear space and compute the distance there. We consider $M$ basis functions, for the sake of argument let's take them to be exponentiated quadratics with different location parameters. We can compute our basis functions as

$$\phi_j(\mathbf{y}_{j,:}) = \exp\left(-\frac{1}{\ell^2}\left\Vert \mathbf{y}_{j,:} - \boldsymbol{\mu}_i \right\Vert_2^2\right)$$

and by taking $\phi_{i,j} = \phi_j(\mathbf{y}_{i,:})$ we can form a set of basis we can convert them to a set of basis vectors, $\boldsymbol{\phi}_{i,:}$, which we represent in a matrix as $\boldsymbol{\Phi} = [\boldsymbol{\phi}_{1,:} \dots \boldsymbol{\phi}_{n,:}]^\top \in \Re^{n\times M}$. We choose this form of representation to keep the basis matrix in the form of a design matrix: i.e. a form with rows equal to the number of data points, $n$.

Just as the squared distance in the data space was given by $d_{i,j} = \left\Vert \mathbf{y}_{i,:} - \mathbf{y}_{j,:} \right\Vert_2^2 = \mathbf{y}_{i,:}^\top \mathbf{y}_{i,:} - 2\mathbf{y}_{i,:}^\top \mathbf{y}_{j,:} + \mathbf{y}_{j,:}^\top \mathbf{y}_{j,:}$, the squared distance in the space given by the basis functions can be computed as

$$d_{i,j} = \left\Vert \boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:} \right\Vert_2^2 = \boldsymbol{\phi}_{i,:}^\top \boldsymbol{\phi}_{i,:} - 2\boldsymbol{\phi}_{i,:}^\top \boldsymbol{\phi}_{j,:} + \boldsymbol{\phi}_{j,:}^\top \boldsymbol{\phi}_{j,:}$$

so we can compute a matrix of squared distances using

$$\mathbf{D} = \mathrm{diag}\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)\mathbf{1}^\top - 2\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \mathbf{1}\mathrm{diag}\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)^\top.$$

Classical scaling on this matrix leads us to compute the eigenvectors of

$$\mathbf{B} = -\frac{1}{2}\mathbf{H}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\mathbf{H}.$$

So we are actually dealing with a similarity matrix, $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$, which has been centered, $\mathbf{B} = \hat{\boldsymbol{\Phi}}\hat{\boldsymbol{\Phi}}^\top$.

In the last chapter (Section 3.4.5.1) we already saw the equivalence between eigenvalue problems on matrices of the form $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ and those on matrices such as $\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$. In particular can show that if the eigenvalues of $\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$ are $\boldsymbol{\Lambda}$ and the eigenvectors are $\mathbf{R}$ then the eigenvalues of $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ are also $\boldsymbol{\Lambda}$ with eigenvectors given by $\mathbf{U} = \hat{\mathbf{Y}}\mathbf{R}\boldsymbol{\Lambda}^{-\frac{1}{2}}$. The same applies if we use the centered basis functions, $\hat{\boldsymbol{\Phi}}$, instead of the centered data, $\hat{\mathbf{Y}}$.

**boxfloat 4.1** Basis Functions

A basis is a set of "seed" vectors which can be added together to form a vector. A basis "spans" the set of vectors that it can create. The usual basis for a 3-dimensional vector would be the set of vectors $\boldsymbol{\phi}_1 = [1 \quad 0 \quad 0]^\top$, $\boldsymbol{\phi}_2 = [0 \quad 1 \quad 0]^\top$ and $\boldsymbol{\phi}_3 = [0 \quad 0 \quad 1]$. These three vectors can be added together with different weights to form any 3-dimensional vector we require. For example $[3 \quad 4 \quad 5] = 4\boldsymbol{\phi}_1 + 4\boldsymbol{\phi}_2 + 5\boldsymbol{\phi}_3$. The basis we have given allows us to represent any 3 dimensional vector. This is known as its vector space. The space that the basis vectors can represent is known as the space that they span.

Sometimes a basis is chosen to be orthogonal, merely meaning that each basis vector is at right angles to one another, so $\boldsymbol{\phi}_i^\top \boldsymbol{\phi}_j = 0$ for $i \neq j$. Each vector can also be normalized to unit length giving $\boldsymbol{\phi}_i^\top \boldsymbol{\phi}_i = 1$. This then implies that the matrix of basis vectors, $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \ldots \boldsymbol{\phi}_p]^\top$, is an orthogonal matrix giving $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \mathbf{I}$. The eigenvectors of a symmetric matrix, for example, have this property. Indeed finding the eigenvectors of a covariance matrix can be seen as finding the basis for which the data becomes uncorrelated, although in the matrix of eigenvectors each basis vector is a column, whereas we will typically place basis vectors along rows of our basis matrices. A basis where each vector is normalized and at right angles to the other vectors is known as orthonormal. A basis needn't be orthonormal, but any basis can be represented by an orthonormal basis which spans the same space. Note, confusingly, an orthogonal matrix is composed of basis vectors that are *orthonormal*, not just orthogonal.

Basis functions are the functional equivalent of a basis. There is a duality between a function and vectors: a vector can be seen as a "look up table" representation of a function. The input to the function is the required element of the vector and the output is the value of that element. A one dimensional function can be thought of as the continuous generalization of this system: for example if $f(x) = x^2$ we can generate a vector of length 3, $\mathbf{f} = [1\ 4\ 9]^\top$, from the function. Given this duality between functions and vectors we can think of basis functions as sets of functions that can be added together to form a new function. So we might represent a function as $f(\mathbf{x}) = w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + w_3\phi_3(\mathbf{x})$.

**TODO: what is spanning a function space — the functions that the basis can represent??**

A common choice in machine learning for basis functions is an exponentiated quadratic form,

$$\phi_i(\mathbf{x}) = \exp\left(-\frac{1}{\ell^2}\left\|\mathbf{x} - \boldsymbol{\mu}_i\right\|_2^2\right),$$

which is variously known as the radial basis function, the Gaussian basis. A widely used basis in signal processing is the Fourier basis which uses sines and cosines. The weights in the Fourier space give a frequency based representation of the system.

Note that if we use only a finite number of basis functions we will not span the space of all functions. For example the Fourier basis would lead to periodic functions and the exponentiated square basis leads to smooth functions with a lengthscale given by $\ell$. However, this is not necessarily a bad characteristic. There may be certain functions we don't wish to represent: ones that have unusual characteristics such as a large number of discontinuities. Our choice of basis restricts the functions we consider to those that our model spans. Prior knowledge can often be very useful in informing this choice.

## 4.1 A Further Intuition on Distances in Feature Space

Instead of computing distances in data space, $\mathbf{Y}$, we are now computing distances in a feature space, $\boldsymbol{\Phi}$. Such distances are nonlinearly related to the original data. Now we would like to develop further intuitions about these distances by considering them from the perspective of basis function based mappings.

We can make use of a weighted set of basis functions to represent a function, as we saw in box 4.1,

$$f(\mathbf{y}_{i,:}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{y}_{i,:}),$$

or using our basis vector notation we can write

$$f(\mathbf{y}_{i,:}) = \mathbf{w}^\top \boldsymbol{\phi}_{i,:} = f_i.$$

To generate random functions, we introduce a probability density that governs the vector $p(\mathbf{w})$. We can now ask the question, what would the expected squared distance be between two different function locations, $f(\mathbf{y}_{i,:}) \equiv f_i$ and $f(\mathbf{y}_{j,:}) \equiv f_j$. If we knew the value of $\mathbf{w}$ this can be directly calculated,

$$(f_i - f_j)^2 = (\boldsymbol{\phi}_{i,:}^\top \mathbf{w} - \boldsymbol{\phi}_{j,:}^\top \mathbf{w})^2$$

we can rewrite this as

$$(f_i - f_j)^2 = (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:})^\top \mathbf{w}\mathbf{w}^\top (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:}).$$

Taking the expectation under the sampling density for $\mathbf{w}$ then gives,

$$\left\langle (f_i - f_j)^2 \right\rangle = (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:})^\top \left\langle \mathbf{w}\mathbf{w}^\top \right\rangle_{p(\mathbf{w})} (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:}).$$

We now see that if the second moment of $p(\mathbf{w})$ is given by the identity matrix, $\left\langle \mathbf{w}\mathbf{w}^\top \right\rangle_{p(\mathbf{w})} = \mathbf{I}$ then the expected squared distance between any two points on the function is given by

$$\left\langle (f_i - f_j)^2 \right\rangle = (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:})^\top (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:}).$$

So we can see this squared distance in feature space arising from the expected distance between any two points on a random function that spans the given basis where the weights of the basis functions are sampled with zero mean, $\langle \mathbf{w} \rangle_{p(\mathbf{w})} = \mathbf{0}$, and unit covariance, $\operatorname{cov}(\mathbf{w}) = \mathbf{I}$. In other words when generating random functions no individual basis function is being scaled larger (on average) than another and the scalings for each basis are independent.

We considered the simple set of exponentiated quadratic basis functions shown in figure 4.1. To illustrate how the expected squared distance is being computed between two points taken from the random functions we show a series of random functions in figure 4.2 along with the distance between two points on those random functions.

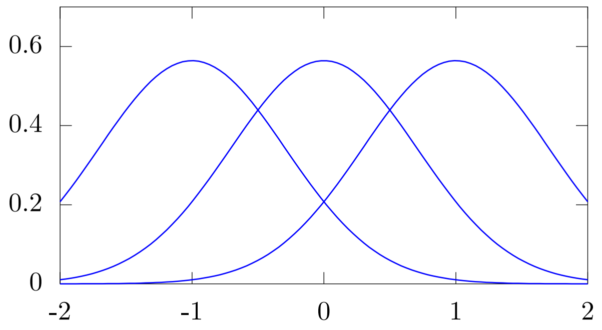


Figure 4.1: A small set of exponentiated quadratic basis functions with centers at $-1$, 0, and 1. The lengthscale of the basis functions is given by $\ell = 1$.

### 4.1.1 Number and Location of Basis

So far we have considered the use of a fixed number of basis functions. The use of exponentiated quadratic basis however highlights a problem with the approach. The use of the random function analogy to see how the distances are derived from the basis functions highlights a problem. In figure 4.3 we show the basis set we have been using across a broader section of the input space. We note that as we get more distance from the centers of the basis functions, their contribution all fades to zero. This has a side effect on distances computed between points in these zones. In figure 4.4 we show distances between points on the randomly sampled functions where the input locations are in the regions not well covered by the basis functions. We see that the distances are very small, despite the differences between the $y$ values being large. This is a side effect of bad basis function placement. However, there is an elegant solution for the exponentiated quadratic basis function that involves placing basis all across the $y$ space. The approach leads to a particular class of basis function methods often known as kernel methods.

## 4.2 Kernel Methods for Basis Functions

We introduced basis functions as a way of nonlinearizing the relationship between our data points and the Euclidean distances we compute between those data points. However, a danger of this approach is that if the basis functions don't cover the entire data space then the distances computed in the feature space can be misleadingly small.

One answer to this problem is to distribute the basis functions all across the data space. We will introduce this approach by considering a one dimensional data space, but it generalizes to high dimensional input spaces.

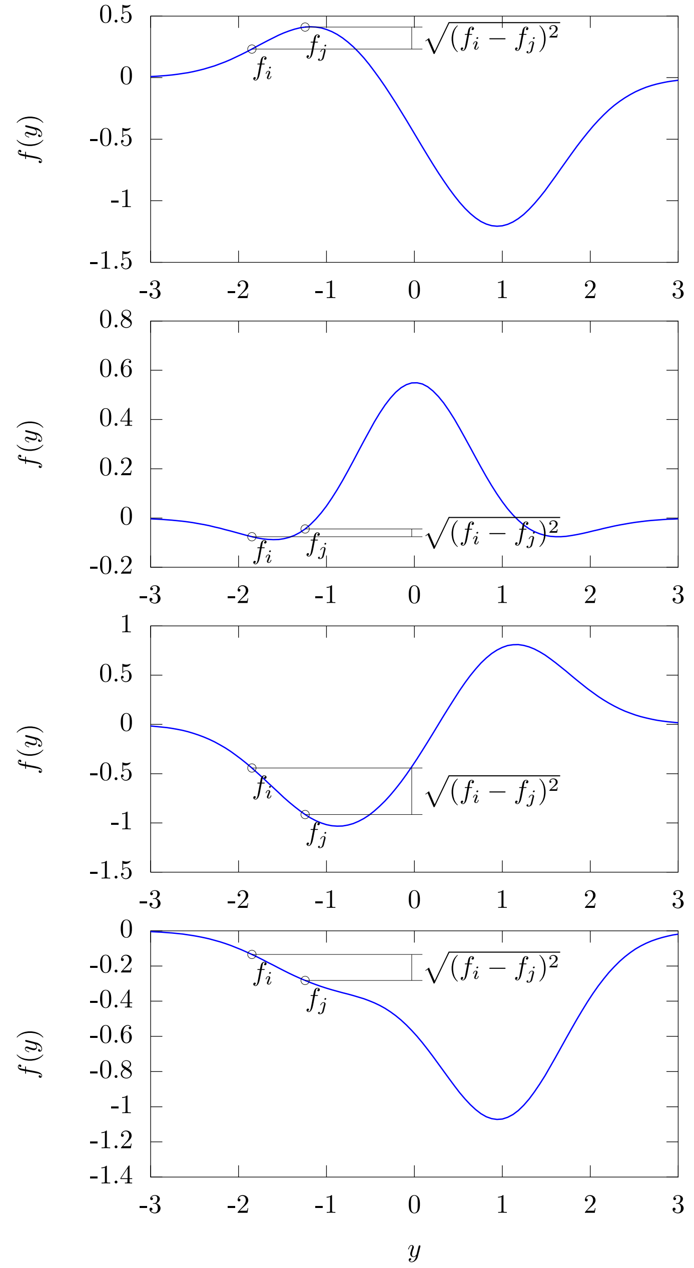


Figure 4.2: Distance between two points in the function $f(y)$. The mapping function is constructed from three exponentiated quadratic basis functions located at $-1$, 0, and 1. The width of the basis functions is given by $\ell = 1$. A 3 dimensional vector, $\mathbf{w}$, is sampled from a Gaussian with zero mean and unit covariance. This vector is used to weight the different basis functions producing the random function shown. Random functions and associated distances are shown for four different samples of $\mathbf{w}$.

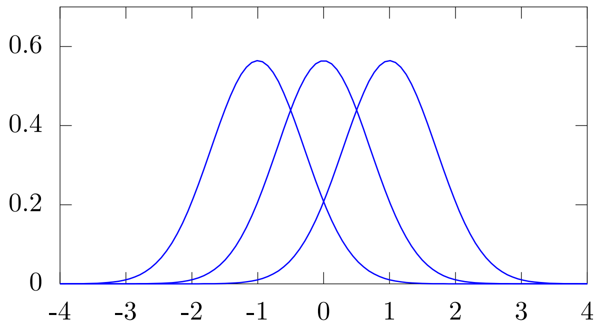


Figure 4.3: The exponentiated quadratic basis functions with centers at $-1$, 0, and 1. As we move away from the centers of the basis functions they decay towards zero.

### 4.2.1 An Infinite Basis

One design question is how to choose the number of basis functions, $M$, and the location of these basis functions, $\{\mu_k\}_{k=1}^{M}$. First we will explore what happens when we place the basis functions at uniform intervals between two values, $a$ and $b$, over a one dimensional space so we have functions of the form

$$f(y) = \sum_{k=1}^{M} w_k \exp\left(-\frac{(y - a - k\Delta\mu)^2}{\ell^2}\right),$$

where we have set the location parameter of each $\phi_k(y)$ to

$$\mu_k = a + k\Delta\mu.$$

The distances in feature space are entirely dependent on the inner product between basis vectors at the different locations, which we write

$$\phi_k(y)\phi_k(y') = \exp\left(-\frac{y^2 + {y'}^2 - 2\left(a + \Delta\mu \cdot k\right)\left(y + y'\right) + 2\left(a + \Delta\mu \cdot k\right)^2}{\ell^2}\right).$$

More basis functions allow greater flexibility in our model of the mapping function. We can increase the number of basis functions we use in the interval between $a$ and $b$ by decreasing the interval $\Delta\mu$. However, to do this without increasing the expected variance of the resulting TF concentration we need to scale down the variance of the prior distribution for $\mathbf{w}$. This can be done by setting the variance parameter to be proportional to the interval, so we take $\gamma = \alpha\Delta\mu$. Now we have basis functions where the location of the leftmost basis function, $k = 1$, is $\mu_1 = a + \Delta\mu$ and the rightmost basis is $\mu_M = b$ so that

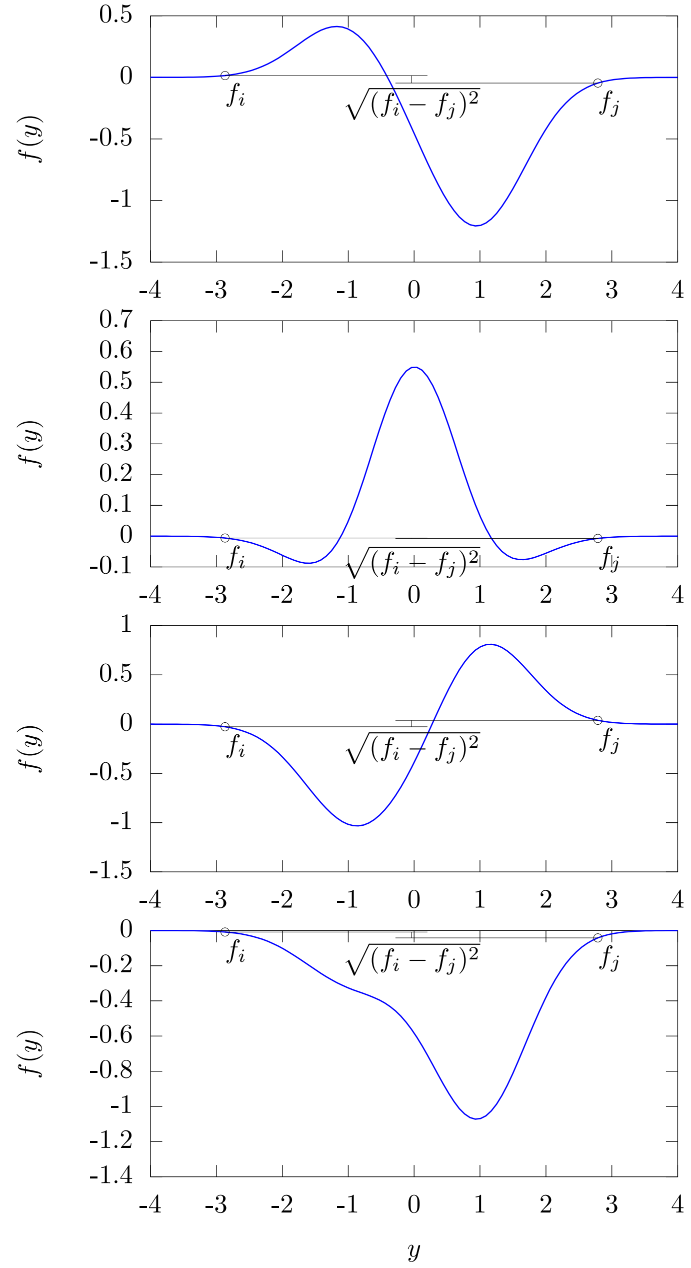


Figure 4.4: Distance between two points in the function $f(y)$. Now the locations are far apart in $y$. However, since they are both in regions where the response from the basis set is small, the distance between the points after mapping through the function, $f(y)$, is small.

$b = a + \Delta\mu \cdot M$. The fixed interval distance between $a$ and $b$ is therefore given by $b - a = (M-1)\Delta\mu$.

We are going to increase the number of basis functions by taking the limit as $\Delta\mu \to 0$. This will take us from a discrete system to a continuous system. In this limit the number of basis functions becomes infinite because we have $M = \lim_{\Delta\mu\to 0} \frac{b-a}{\Delta\mu} + 1$. In other words we are moving from a fixed number of basis functions to infinite basis functions. The inner product between the basis functions becomes

$$k(y, y') = \frac{\alpha}{\pi\ell^2} \int_a^b \exp\left(-\frac{y^2 + {y'}^2 - 2\mu(y + y') + 2\mu^2}{\ell^2}\right) \mathrm{d}\mu,$$

where we have used $\mu = a + \Delta\mu$. Completing the square gives

$$k(y, y') = \frac{\alpha}{\pi\ell^2} \int_a^b \exp\left(-\frac{y^2 + {y'}^2 + 2\left(\mu - \frac{1}{2}(y + y')\right)^2 - \frac{1}{2}(y + y')^2}{\ell^2}\right) \mathrm{d}\mu$$

which we rewrite in the form

$$k(y, y') = \frac{\alpha}{\sqrt{2\pi\ell^2}} \exp\left(-\frac{(y - y')^2}{2\ell^2}\right) \sqrt{\frac{2}{\pi\ell^2}} \int_a^b \exp\left(-\frac{2}{\ell^2}\left(\mu - \frac{y + y'}{2}\right)^2\right) \mathrm{d}\mu,$$

allowing us to integrate over $\mu$ to give

$$k(y, y') = \frac{\alpha}{\sqrt{2\pi\ell^2}} \exp\left(-\frac{(y - y')^2}{2\ell^2}\right) \frac{1}{2}\left[1 + \mathrm{erf}\left(\sqrt{\frac{2}{\ell^2}}\left(\mu - \frac{y + y'}{2}\right)\right)\right]_a^b,$$

which can be rewritten as

$$k(y, y') = \frac{\alpha}{\sqrt{2\pi\ell^2}} \exp\left(-\frac{(y - y')^2}{2\ell^2}\right) \frac{1}{2}\left[\mathrm{erf}\left(\sqrt{\frac{2}{\ell^2}}\left(b - \frac{y + y'}{2}\right)\right) - \mathrm{erf}\left(\sqrt{\frac{2}{\ell^2}}\left(a - \frac{y + y'}{2}\right)\right)\right].$$

Finally, we wish to spread our basis functions across the entire real line so there are no zones where they are inactive. This can be done by taking the limit where the boundaries of the locations head towards infinity, so we take the limit as $a \to -\infty$ and $b \to \infty$. This gives a model where we have infinite basis functions distributed across the entire real line. In this case the term inside the square brackets becomes two, leaving only the exponentiated quadratic term

$$k(y, y') = \frac{\alpha}{\sqrt{2\pi\ell^2}} \exp\left(-\frac{(y - y')^2}{2\ell^2}\right).$$

This analysis above shows that if we take a one dimensional fixed basis function model, we can increase the number of basis functions to infinity and distribute them evenly across the real line. We no longer have to specify the number or location of individual basis functions and there will no longer be zones where basis functions are zero. Note that we can no longer compute the basis, $\boldsymbol{\phi}(y)$, explicitly as it is infinite dimensional, however inner products of the basis can be computed, we have $\boldsymbol{\phi}(y)^\top \boldsymbol{\phi}(y') = k(y, y') = \exp\left(-\frac{(y-y')^2}{2\ell^2}\right)$.

This procedure for moving from inner products, $\boldsymbol{\phi}(y)^\top \boldsymbol{\phi}(y')$, to covariance functions, $k(y, y')$, is sometimes known as kernelization Schölkopf and Smola [2001] due to the fact that $k(y, y')$ has the properties of a Mercer kernel. Specifically if a symmetric matrix of values of $k(y, y')$ is computed for a vector of times $\mathbf{y}$ it should always be positive semi-definite. In other words if $k_{i,j} = k(y_i, y_j)$ is the element from the $i$th row and $j$th column of $\mathbf{K}$ and $y_i$ is the $i$th element from $\mathbf{y}$, we should have that $\mathbf{K}$ is a positive definite matrix for any vector of inputs $\mathbf{y}$. This same property allows $k(y, y')$ to be used as a *covariance function*: a function that can generate a covariance matrix. Covariances must be positive definite: the matrix $\mathbf{K}$ specifies the covariance between instantiations of the function $f(y)$ at the times given by $\mathbf{y}$. Mercer's theorem says that underlying all such positive definite functions there is always a (possibly infinite) feature space, $\phi(y)$, which can be used to construct the covariance function. For our example the relationship between the feature space and this covariance function emerges naturally through considering a Bayesian approach to a fixed basis function model. The resulting model is known as a Gaussian process [O'Hagan, 1978, Williams, Rasmussen and Williams, 2006] (see chapter **??**).

We can sample random functions as before.

## 4.3 Salt Taffy Effect ? Need to check what John was referring to!

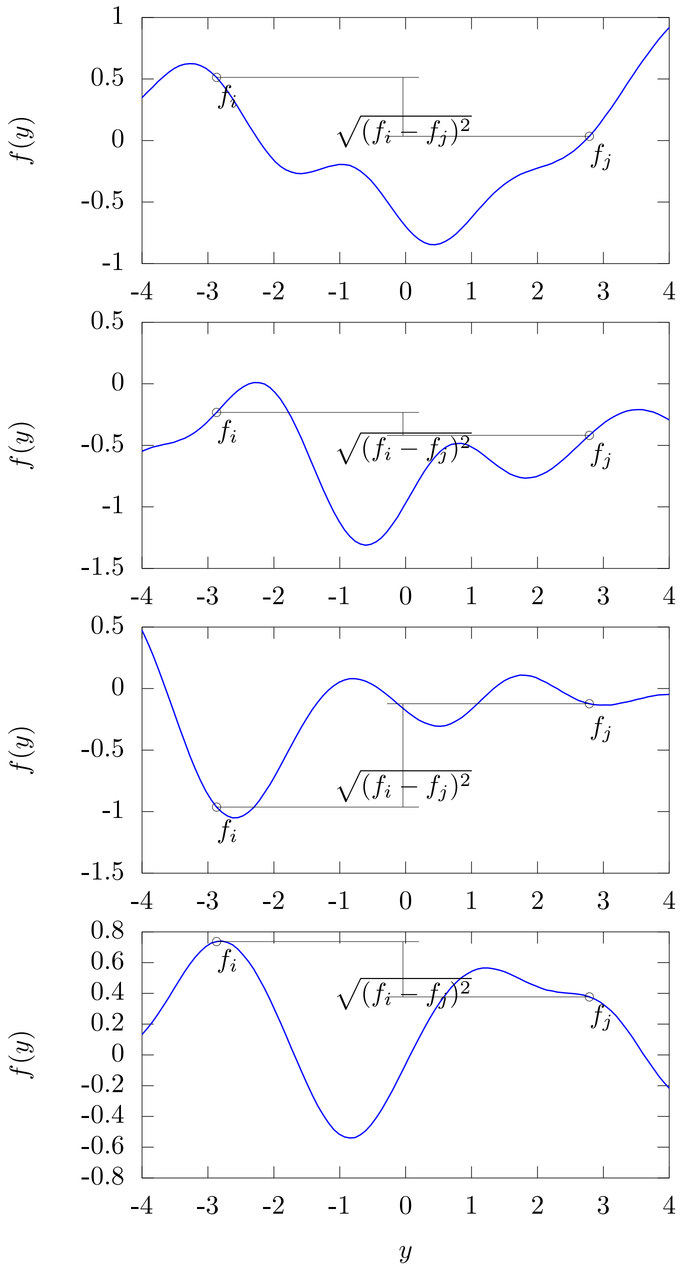
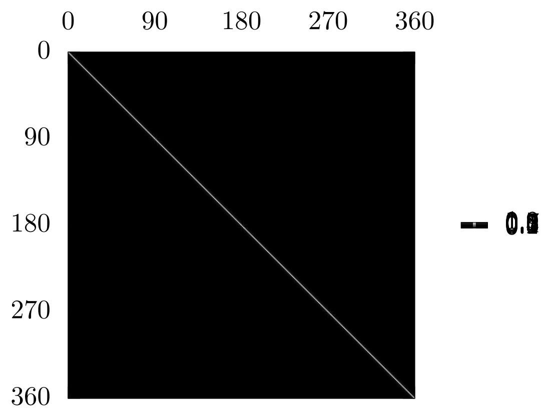
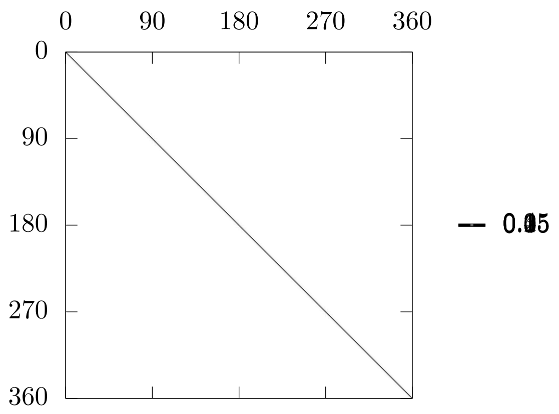


Figure 4.5: Distance between two points in the function $f(y)$. The locations are again far apart in $y$ but now we are using an infinite basis set so the distance between the two points in feature space is large.

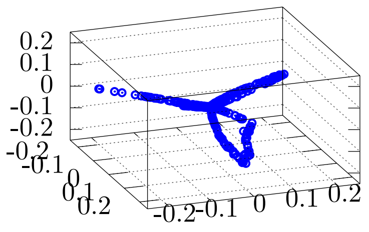


(a)

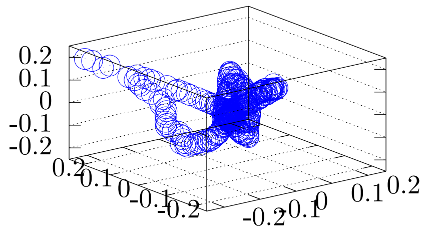


(b)

Figure 4.6: (a) similarity matrix for RBF kernel on rotated sixes. (b) implied distance matrix for kernel on rotated sixes. Note that most of the distances are set to $\sqrt{2} \approx 1.41$.



(a)



(b)

Figure 4.7: Visualization of different dimensions of the rotated six data using kernel PCA. Only close neighbors in the rotated data are similar under the exponentiated quadratic kernel. Other points are spread out along axes so that the dissimilar points are always $\sqrt{2}$ apart in the latent space.

# Chapter 5

# Spectral Dimensionality Reduction

## 5.1 Introduction

In this chapter we review spectral approaches to dimensionality reduction. The main innovation in these approaches over the scaling approaches we reviewed in the last chapter are in how the interpoint distances are computed. We mainly focus on four different algorithms: locally linear embeddings [Roweis and Saul, 2000], Laplacian eigenmaps [Belkin and Niyogi, 2003], isomap **?** and maximum variance unfolding [Weinberger et al., 2004]. These approaches are all closely related to the classical multidimensional scaling we introduced in the last chapter. The main difference between these different approaches in in the distance matrix they imply. We will present each algorithm, but also relate them through a unifying perspective derived from Gaussian random field models Lawrence [2010]. We will first consider the maximum variance unfolding (MVU) algorithm.

## 5.2 Maximum Variance Unfolding

Classical multidimensional scaling provides only a linear transformation of space in which the squared distances are expressed. The novelty of modern spectral approaches is distance computation in spaces which are nonlinearly related to the data. This gives a nonlinear algorithm. This can be seen clearest for kernel

PCA. In kernel PCA the squared distances are embedded in a Hilbert space and related to the original data through a kernel function,

$$d_{i,j} = k(\mathbf{y}_{i,:}, \mathbf{y}_{i,:}) - 2k(\mathbf{y}_{i,:}, \mathbf{y}_{j,:}) - k(\mathbf{y}_{j,:}, \mathbf{y}_{j,:}) \tag{5.1}$$

which is recognized as the squared distance in "feature space" [see Ham et al., 2004]. In CMDS this is known as the *standard transformation* between a similarity and distance [Mardia et al., 1979]. Kernel PCA (KPCA) recovers an $\mathbf{x}_{i,:}$ for each data point and a mapping from the data space to the $\mathbf{X}$ space. Under the CMDS procedure the eigenvalue problem is performed on the centered kernel matrix, $\mathbf{B} = \mathbf{HKH}$ where $\mathbf{K} = [k(\mathbf{y}_{i,:}, \mathbf{y}_{j,:})]_{i,j}$. This matches the KPCA algorithm [Schölkopf et al., 1998]. However, for the commonly used exponentiated square kernel, $k(y_{i,:}, y_{j,:}) = \exp(-\gamma \left\Vert \mathbf{y}_{i,:} - \mathbf{y}_{j,:} \right\Vert_2^2)$, KPCA actually expands the feature space rather than reducing the dimension [see Weinberger et al., 2004, for some examples of this].

The observation that KPCA expands the feature space motivated the maximum variance unfolding algorithm [Weinberger et al., 2004]. The idea in MVU is to learn a kernel matrix that will allow for dimensionality reduction. This is achieved by only considering *local relationships* in the data. A set of neighbors is defined (e.g. by $k$-nearest neighbors) and only distances between neighboring data points are respected. These distances are specified as constraints, and the other elements of the kernel matrix are filled in by maximizing the trace of the kernel matrix, $\text{tr}(\mathbf{K})$, i.e. the *total variance* of the data in feature space, while respecting the distance constraints and keeping the resulting matrix centered. Maximizing $\text{tr}(\mathbf{K})$ in turn maximizes the interpoint squared distances for all points that are unconnected in the neighborhood graph, thereby unravelling the manifold.

**TODO: Prove that inverse of the covariance is sparse???**

## 5.3 Maximum Entropy Unfolding

The maximum variance unfolding was motivated by considering data points as a set of nodes in an interlinked graph. The idea was to reduce the dimensionality by forcing those nodes apart, but constraining connected nodes in the graph to be a fixed distance apart. The distance chosen was the distance in "feature space" proscribed by the kernel function that was learnt. A visualization is then obtained by computing the eigenvectors of the resulting graph. We will introduce further spectral approaches like this, but to give a unifying perspective, we introduce a slightly different approach to MVU: rather than maximizing the variance of the system, we consider maximizing the *entropy*, the entropy $p(x)$ is t The entropy is related to the variance (see box 5.1), but maximum entropy has a side effect: it is equivalent to maximum likelihood and results in a *probabilistic* model. As we saw for probabilistic PCA in chapter **??** Probabilistic interpretations of a model can be useful as they allow the full calculus of probabilities to be applied to the system. This brings significant advantages, such as the ability to deal with missing data, placing the model within a wider probabilistic system (such as a mixture model) and applying Bayesian treatments.

Since entropy and variance both relate to uncertainty: as variance increases we become more uncertain about the likely outcome of our model, and in information theory entropy is the measure of uncertainty, we might expect maximizing entropy leads to a similar algorithm to MVU. Furthermore, it allows us to bring to bear the full *maximum entropy principal* framework, developed by Jaynes, 1986.

In the maximum entropy formalism [see e.g. Jaynes, 1986, and box 5.2], we maximise the entropy of a distribution subject to constraints on the moments of that distribution. Here those constraints will be the expectations of the squared distances between two data points sampled from the model. Constraints will only apply to points that are defined to be "neighbors". For continuous data, maximum entropy can only be defined relative to a base distribution. We follow a common choice and take the base distribution to be a spherical Gaussian with covariance $\gamma^{-1}\mathbf{I}$. The maximum entropy distribution is then given by

$$p(\mathbf{Y}) \propto \exp\left(-\frac{1}{2}\mathrm{tr}\left(\gamma\mathbf{Y}\mathbf{Y}^\top\right)\right)\exp\left(-\frac{1}{2}\sum_i\sum_{j\in\mathcal{N}(i)}\lambda_{i,j}d_{i,j}\right),$$

where $\mathcal{N}(i)$ represents the set of neighbors of data point $i$, and $\mathbf{Y} = [\mathbf{y}_{1,:},\dots,\mathbf{y}_{n,:}]^\top \in \Re^{n\times p}$ is the *design matrix* containing our data and we've introduced , $\{\lambda_{i,j}\}$, which are a set of Lagrange multipliers (see box 3.2) which enforce the moment constraints. Compared to the standard maximum entropy formalism we've introduced a factor of $-1/2$ in front of our Lagrange multipliers which will prove notationally convenient later on. Also for convenience, we define a sparse matrix of Lagrange multipliers, $\mathbf{\Lambda}$ which contains $\lambda_{i,j}$ if $i$ is a neighbor of $j$ and zero otherwise. This allows us to write the distribution as

$$p(\mathbf{Y}) \propto \exp\left(-\frac{1}{2}\mathrm{tr}\left(\gamma\mathbf{Y}\mathbf{Y}^\top\right) - \frac{1}{4}\mathrm{tr}\left(\mathbf{\Lambda}\mathbf{D}\right)\right).$$

We now introduce a matrix $\mathbf{L}$. This matrix is going to allow us to rewrite the distribution in terms of $\mathbf{Y}$ instead of $\mathbf{D}$. The matrix $\mathbf{L}$ is symmetric and constrained to have a null space in the constant vector, i.e. $\mathbf{L}\mathbf{1} = \mathbf{0}$. Its off diagonal elements are given by $-\mathbf{\Lambda}$. We then force the null space constraint by setting its diagonal elements to be the sum of the off diagonal in the corresponding row (or column since it is symmetric) giving $\ell_{i,i} = \sum_{j\in\mathcal{N}(i)}\lambda_{i,j}$. This enables us to write

$$\mathrm{tr}\left(\mathbf{L}\mathbf{D}\right) = -\mathrm{tr}\left(\mathbf{\Lambda}\mathbf{D}\right)$$

**boxfloat 5.1** Entropy and Information Theory

The entropy of a distribution is the expectation of the negative log likelihood of the distribution under that distribution,

$$P(x) = -\left\langle \log P(x) \right\rangle_{P(x)} = -\sum_{x} P(x) \log P(x).$$

Negative log likelihood has an interpretation as the "error" when fitting probabilistic models to data. This means that the entropy has an interpretation as being the average error we expect across all data sets sampled from the model itself (since the expectation is under the distribution itself, $p(x)$). What governs this average error? Principally it is the number of different samples (or data sets) that $p(x)$ can generate. If the model $p(x)$ is deterministic then when sampled it will only ever give one result, and the probability of this result will be 1, giving an error of 0. If a model is defined over a space containing $M$ different possible values for $x$ and each of these values is equally likely, then the entropy is given by $\log M$. This is the maximum possible entropy. This model has the most possible outcomes and is associated with the largest amount of uncertainty. The idea here is that even if we have the right model for a data set, if the entropy is high, we don't still expect a large error.
Continuous Systems
Entropy is well defined for discrete systems, because the number of possible outcomes for a model is countable. For continuous systems the number of possible outcomes is infinite, so it is not well defined. By taking the limit of a discrete system as it becomes continuous we find that it becomes,

$$E(p(x)) = -\int p(x) \log p(x) \mathrm{d}x - O(log \mathrm{d}x).$$

The second term is unbounded, so in practise we only really answer questions about the difference between entropies for continuous distributions (where that term cancels, this also happens in Kullback-Leibler divergences, see box 2.7), however we often (sloppily) think of the entropy as being given by

$$E(p(x)) = -\int p(x) \log p(x) \mathrm{d}x.$$

and problems then only occur when we ask questions like what is the entropy of delta function (which we expect to be zero since it is deterministic) and we find that it is infinite.

**boxfloat 5.2** The Maximum Entropy Principal

The maximum entropy principal was developed by Ed Jaynes as a way of defining distributions. The principal is as follows: we wish to develop a distribution given only some constraint on the moments of the distributions. For example, we might know that the mean of the distribution is 1, and the variance is 1. Which class of distributions should we choose? The idea of maximum entropy is to choose the class which has the largest entropy under those constraints. In other words we want the model that will have the *largest expected error* for any samples from it. Where error is defined to be the negative log likelihood. This seems a little counter intuitive at first: why should we want the distribution with the largest expected error? Bear in mind that this expected error is for future (test) data, any information we have currently is being included by the moment constraints. The largest expected error equates to the maximum uncertainty about samples from the distribution. The principal is that by maximizing the expected error we cover more possible data sets.
The framework is extremely elegant, as it turns out that a variational maximization may be performed across all possible distributions, and that the resulting distribution always has the form

$$P(x) \propto \exp(\sum_i \lambda_i f_i(x)$$

***NEEDS FIXING***

This is possible because the distance matrix, $\mathbf{D}$, is zero along its diagonal. The trace of a matrix product is equivalent to multiplying each matrix together elementwise and then summing over all the elements of the result (see also box 2.9). Since the distance matrix is zero along the diagonal then this trace is unaffected by what the diagonal elements of $\mathbf{L}$ are, we can set them as we please without changing the value of $\mathrm{tr}\left(\mathbf{LD}\right)$. Our choice to set them as the sum of the off diagonals gives the $\mathbf{L}$ that null space in the constant vector. Recalling the matrix representation of the squared distance matrix,

$$\mathbf{D} = \mathbf{1}\mathrm{diag}\left(\mathbf{YY}^\top\right)^\top - 2\mathbf{YY}^\top + \mathrm{diag}\left(\mathbf{YY}^\top\right)\mathbf{1}^\top,$$

where the operator $\mathrm{diag}\left(\mathbf{A}\right)$ forms a vector from the diagonal of $\mathbf{A}$. We now see the benefit of incorporating that null space,

$$-\mathrm{tr}\left(\mathbf{\Lambda D}\right) = \mathrm{tr}\left(\mathbf{LD}\right) = \mathrm{tr}\left(\mathbf{L1}\mathrm{diag}\left(\mathbf{YY}^\top\right)^\top - 2\mathbf{LYY}^\top + \mathrm{diag}\left(\mathbf{YY}^\top\right)\mathbf{1}^\top\mathbf{L}\right) = -2\mathrm{tr}\left(\mathbf{LYY}^\top\right),$$

where we have used the properties of the trace (box 2.9). This in turn allows us to recover

$$p(\mathbf{Y}) = \frac{\left|\mathbf{L} + \gamma\mathbf{I}\right|^{\frac{1}{2}}}{(2\pi)^{\frac{np}{2}}}\exp\left(-\frac{1}{2}\mathrm{tr}\left((\mathbf{L}+\gamma\mathbf{I})\mathbf{YY}^\top\right)\right). \tag{5.2}$$

which is recognized as a *Gaussian random field* (see box 5.3).

**boxfloat 5.3** Gaussian Random Field

A Gaussian random field is a particular type of Gaussian distribution where there is an underlying neighborhood structure. Each variable governed by the Gaussian distribution is conditioned only on its neighbors in the field. There is a sparse structure to the This implies we can write

We can now write down the full form of this probability distribution: it is a *Gaussian random field.* It can be written as

$$p(\mathbf{Y}) = \prod_{j=1}^{p} \frac{|\mathbf{L} + \gamma\mathbf{I}|^{\frac{1}{2}}}{(2\pi)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}\mathbf{y}_{:,j}^{\top}(\mathbf{L} + \gamma\mathbf{I})\mathbf{y}_{:,j}\right),$$

which clarifies the fact that the GRF is being expressed independently across data features (each vector $\mathbf{y}_{:,j}$ contains the $j$th feature from all data points). The model therefore falls into the class of models we defined in Section 2.3 as *feature consistency*, so they are appropriate in the large $p$ small $n$ domain which has traditionally been thought to be so problematic. This contrasts with most applications of Gaussian models that are applied independently across data points. Notable exceptions include the dual form of probabilistic PCA we saw in chapter **??**, the Gaussian process latent variable model we will see in chapter **??** and work on semisupervised learning [Zhu et al., 2003] and human learning [Kemp and Tenenbaum, 2008].

***Why is this so, say either in box or here*** As with all maximum entropy methods, maximum likelihood for this model is equivalent to finding the correct setting of the Lagrange multipliers. We can find the parameters $\mathbf{\Lambda}$ through maximum likelihood on this distribution. Some algebra shows that the gradient of each Lagrange multiplier is given by,

$$\frac{\mathrm{d}\log p(\mathbf{Y})}{\mathrm{d}\lambda_{i,j}} = \frac{1}{2}\left\langle d_{i,j}\right\rangle_{p(\mathbf{Y})} - \frac{1}{2}d_{i,j},$$

where $\langle\rangle_{p(\cdot)}$ represents an expectation under the distribution $p(\cdot)$. This result is expected given our maximum entropy formulation. To compute gradients we need the expectation of the squared distance given by $\langle d_{i,j}\rangle = \left\langle y_{i,:}^{\top}y_{i,:}\right\rangle - 2\left\langle y_{i,:}^{\top}y_{j,:}\right\rangle + \left\langle y_{j,:}^{\top}y_{j,:}\right\rangle$, which we can compute directly from the covariance matrix of the GRF, $\mathbf{K} = (\mathbf{L} + \gamma\mathbf{I})^{-1}$,

$$\langle d_{i,j}\rangle = \frac{p}{2}\left(k_{i,i} - 2k_{i,j} + k_{j,j}\right).$$

This is immediately recognized as a scaled version of the standard transformation between distances and similarities (see chapter 3). This relationship arises naturally in the probablistic model. Every Gaussian random field has an implied associated distance matrix. It is this matrix that is being used in classical multidimensional scaling. In chapter 3 we also interpreted this as the distance in "feature space" defined by the kernel function. Now, however, and also in MVU, each individual element of the kernel matrix *cannot* in general be represented only as a function of the corresponding two data points (i.e. we can't represent them as $k_{i,j} = k(\mathbf{y}_{i,:}, \mathbf{y}_{j,:})$). Given this we feel it is more correct to think of this matrix as a covariance matrix induced by our specification of the random field rather than a true Mercer kernel.

If $K$ neighbors are used for each data point there are $O(Kn)$ parameters in the model, so the model is nonparametric in the sense that the number of parameters increases with the number of data. For the parameters to be well determined we require a large number of features, $p$, for each data point, otherwise we would need to look to regularize the model.***reference ahead if we do this, but probably we won't***

Once the maximum likelihood solution is recovered the data can be visualized, as for MVU and kernel PCA, by looking at the eigenvectors of the centered covariance matrix $\mathbf{HKH}$. The resulting algorithm has been called maximum entropy unfolding [MEU Lawrence, 2010].

Note that the entropy of a Gaussian is related to the determinant of the covariance matrix which can be re-expressed as the sum of the log of the eigenvalues of $\mathbf{K}$, $\log|\mathbf{K}| = \sum_{i=1}^{n} \log \lambda_i$. In contrast MVU looks to maximize the trace of the covariance matrix $\text{tr}(\mathbf{K}) = \sum_{i=1}^{n} \lambda_i$, subject to distance constraints.

When optimizing in MVU and MEU we need to ensure that the covariance matrix is positive definite. In MVU this is ensured through a semidefinite program. In MEU the objective is not linear in $\mathbf{K}$ so we need to use other approaches. Possibilities include exploiting the fact that if the Lagrange multipliers are constrained to be positive the system is "attractive" and this guarantees a valid covariance [see e.g. Koller and Friedman, 2009, pg 255]. Although now (as in a suggested variant of the MVU) the distance constraints would be inequalities. Another alternative would be to constrain $\mathbf{L}$ to be diagonally dominant through adjusting $\gamma$. We will also consider two further approaches in Section 5.5 and Section **??**.

Finally, we note that for MEU and MVU, as we increase the neighborhood size to $K = n - 1$, we recover principal component analysis. In this limit all expected squared distances, implied by the GRF model, are required to match the observed squared distances and $\mathbf{L}$ becomes non-sparse. Classical multidimensional scaling on the resulting squared distance matrix is known as principal coordinate analysis and is equivalent to principal component analysis [see Mardia et al., 1979].

## 5.4 Laplacian Eigenmaps

Laplacian eigenmaps is a fast and powerful approach to dimensionality reduction introduced by [Belkin and Niyogi, 2003].

### 5.4.1 Relation to Maximum Entropy Unfolding

From the eigendecomposition of $\mathbf{K} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ it is easy to show that $\mathbf{L} = \mathbf{U}\left(\boldsymbol{\Lambda}^{-1} - \gamma\mathbf{I}\right)\mathbf{U}^\top$ is the eigendecomposition of $\mathbf{L}$. So in other words, the principal eigenvalues of $\mathbf{K}$ will be the smallest eigenvalues of $\mathbf{L}$. The very smallest eigenvalue of $\mathbf{L}$ is zero and associated with the constant eigenvector. However, in CMDS this would be removed by the centering operation and in LE it is discarded. So we see that once the parameters of the Laplacian have been set CMDS is being performed to recover the latent variables in Laplacian eigenmaps. However, since the moment constraints are not being imposed in Laplacian eigenmaps, the squared distance matrix used for CMDS will not preserve the inter-neighbor distances as it will for MVU and MEU. In fact since the covariance matrix is never explicitly computed it is not possible to make specific statements about what these distances will be in the general case. However, LE gains significant computational advantage by not representing the covariance matrix explicitly. No matrix inverses are required in the algorithm and the resulting eigenvalue problem is sparse. This means that LE can be applied to much larger data sets than would be possible for MEU or MVU.

## 5.5 Locally Linear Embedding

The locally linear embedding, along with isomap, triggered a revolution in spectral methods in the machine learning community.

### 5.5.1 Relation to Maximum Entropy Unfolding

When introducing MEU we discussed how it is necessary to constrain the Laplacian matrix to be positive semidefinite. A further way of doing this is to factorize the Laplacian as $\mathbf{L} = \mathbf{M}\mathbf{M}^\top$ where $\mathbf{M}$ is non-symmetric. If $\mathbf{M}$ is constrained so that $\mathbf{M}^\top\mathbf{1} = \mathbf{0}$ then we will also have $\mathbf{L}\mathbf{1} = \mathbf{0}$. We can achieve this constraint by setting the diagonal elements $m_{i,i} = -\sum_{j\in\mathcal{N}(i)} m_{j,i}$. Then if we force $m_{j,i} = 0$ if $j \notin \mathcal{N}(i)$ we will have a Laplacian matrix which is positive semidefinite without need for any further constraint on $\mathbf{M}$. Note that the sparsity pattern of $\mathbf{L}$ will be different from the pattern of $\mathbf{M}$. The entry for $\ell_{i,j}$ will only be zero if there are no shared neighbors between $i$ and $j$.

Locally linear embeddings [Roweis and Saul, 2000] are then a specific case of this random field model where

1. The diagonal sums, $m_{i,i}$, are further constrained to unity.

2. The parameters of the model are optimized by maximizing the pseudolikelihood of the resulting GRF.

To see the first point, we note that if the diagonals were constrained to unity then we can write $\mathbf{M} = \mathbf{I} - \mathbf{W}$. Here the sparsity pattern of $\mathbf{W}$ matches $\mathbf{M}$, apart from the diagonal which is set to zero. These constraints mean that $(\mathbf{I} - \mathbf{W})^\top \mathbf{1} = \mathbf{0}$. LLE proscribes that the smallest eigenvectors of $(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W})^\top = \mathbf{M}\mathbf{M}^\top = \mathbf{L}$ are used with the constant eigenvector associated with the eigenvalue of 0 being discarded. As for the Laplacian eigenmaps this is equivalent to CMDS on the Gaussian random field described by $\mathbf{L}$.

For the second point we consider the following. The pseudolikelihood approximation [see e.g. Koller and Friedman, 2009, pg 970] to the joint density in a graphical model is the product of the conditional densities: $p(\mathbf{Y}) \approx \prod_{i=1}^n p(\mathbf{y}_{i,:}|\mathbf{Y}_{\backslash i})$, where $\mathbf{Y}_{\backslash i}$ represents all that data other than the $i$th point. The true joint likelihood is proportional to the product of conditional densities, but it requires renormalization. In pseudolikelihood this normalization is ignored. To see how this decomposition applies we first factorize the model by noting that $\mathrm{tr}\left(\mathbf{Y}\mathbf{Y}^\top\mathbf{M}\mathbf{M}^\top\right) = \sum_{i=1}^n \mathbf{m}_{:,i}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{m}_{:,i}$ so we have $\exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{Y}\mathbf{Y}^\top\mathbf{M}\mathbf{M}^\top\right)\right) = \prod_{i=1}^n \exp\left(-\frac{1}{2}\mathbf{m}_{i,:}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{m}_{i,:}\right)$. This provides the necessary factorization for the conditionals which can be rewritten as

$$p(\mathbf{y}_{i,:}|\mathbf{Y}_{\backslash i}) = \left(\frac{m_{i,i}^2}{2\pi}\right)^{\frac{p}{2}} \exp\left(-\frac{m_{i,i}^2}{2}\left\|\mathbf{y}_{i,:} - \sum_{j\in\mathcal{N}(i)} \frac{w_{j,i}}{m_{i,i}}\mathbf{y}_{j,:}\right\|_2^2\right).$$

Optimizing the pseudolikelihood is equivalent to optimizing $\log p(\mathbf{Y}) \approx \sum_{i=1}^n \log p(\mathbf{y}_{i,:}|\mathbf{Y}_{\backslash i})$ which is equivalent to solving $n$ independent regression problems with a constraint on the regression weights that they sum to one. This is exactly what is done to estimate the parameters in LLE [Roweis and Saul, 2000]. The constraint arises because the regression weights are constrained to be $w_{j,i}/m_{i,i}$ and $m_{i,i} = \sum_{j\in\mathcal{N}(i)} w_{j,i}$. Effectively in LLE a further constraint is placed that $m_{i,i} = 1$ which implies none of these regression problems should be solved to a greater precision than another. However, the algorithm also works if this further constraint isn't imposed.

Locally linear embeddings are therefore an approximation to maximum likelihood on the Gaussian random field. They have a neat way of constraining the Laplacian to be positive semidefinite by assuming a factorized form. The pseudolikelihood also allows for relatively quick parameter estimation by ignoring the partition function from the actual likelihood. This again removes the need to invert to recover the covariance matrix and means that LLE can be applied to larger data sets than MEU or MVU. However, the sparsity pattern in the Laplacian for LLE will not match that used in the Laplacian for the other algorithms due to the factorized representation.

**boxfloat 5.4** Shortest Path Algorithms

LLE is motivated by considering local linear embeddings of the data, although interestingly, as we increase the neighborhood size to $K = n - 1$ we do not recover PCA which is known to be the optimal linear embedding of the data under linear Gaussian constraints. The fact that LLE is optimizing the pseudolikelihood makes it clear why this is the case. In contrast the MEU algorithm, which LLE approximates, does recover PCA when $K = n - 1$.

## 5.6 Isomap

Isomap is perhaps one of the most intuitive dimensionality reduction approaches, from the perspective of classical multidimensional scaling, isomap works to ensure that the distances embedded genuinely do reflect the distances along the mainifold.

### 5.6.1 Relation to Other Spectral Approaches

Isomap more directly follows the CMDS framework. In isomap [Tenenbaum et al., 2000] a sparse graph of distances is created between all points considered to be neighbors. This graph is then filled in for all non-neighboring points by finding the shortest distance between any two neighboring points in the graph (along the edges specified by the neighbors). The resulting matrix is then element-wise squared to give a matrix of square distances which is then processed in the usual manner (centering and multiplying by -0.5) to provide a similarity matrix for multidimensional scaling. Compare this to the situation for MVU and MEU. Both MVU and MEU can be thought of as starting with a sparse graph of (squared) distances. The other distances are then filled in by either maximizing the trace of the associated covariance or maximizing the entropy of the associated Gaussian distribution. Importantly, though, the interneighbor distances in this graph are preserved (through constraints imposed by Lagrange multipliers) just like in isomap. For both MVU and MEU the covariance matrix, $\mathbf{K}$, is guaranteed positive semidefinite because the distances are implied by an underlying covariance matrix that is constrained positive definite. For isomap the shortest path algorithm is effectively approximating the distances between non-neighboring points. This can lead to an implied covariance matrix which has negative eigenvalues (see [Weinberger et al., 2004]). The algorithm is still slower than LLE and LE because it requires a dense eigenvalue problem and the application of a shortest path algorithm to the graph provided by the neighbors.

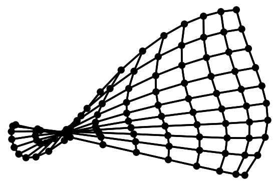


Figure 5.1: Finite element model of a piece of paper. The paper is divided into a grid of points. Each point is connected to its neighbors through springs.

## 5.7 Physical Interpretation of the System

The maximum entropy model we introduced has a physical interpretation that may provide some intuition when understanding the process of dimensionality reduction.

### 5.7.1 The Mattress Model

An often used example to introduce the problem of dimensionality reduction is that of a piece of (two dimensional) paper is crumpled so that it lives in a three dimensional space. In this analogy, the process of dimensionality reduction is the flattening out of the three dimensional crumpled paper to a two dimensional space. The analogy can also be extended to consider data spaces of higher dimensionality, $p$, and higher dimensional, $q$, pieces of paper. However, for the moment let's consider a two dimensional sheet of paper crumpled in a $p$-dimensional space.

One approach to modeling a form such as a sheet of paper is to consider a finite element model (effectively a discretization of the continuous surface). A finite element model of a piece of paper could consist of a grid of points, each point connected to its nearest 4 neighbors through a spring,[1] like an old fashioned mattress. We denote the spring constant connecting point[2] $i$ to point $j$ by $\kappa_{i,j}$. If two points are unconnected we can take $\kappa_{i,j} = 0$. We represent all springs in a matrix, $\boldsymbol{\mathcal{K}}$. Such a grid representing paper folded in three dimensions is shown in Figure 5.1.

[1]For higher dimensional paper, more neighbors would be considered, for lower dimensional paper (a piece of string) two neighbors would be considered.

[2]The spring constant could be negative (indicating a repulsive spring).

When the paper is deformed in high dimensions, we denote the location of the $i$th point from the grid by $\mathbf{y}_{i,:}$. We represent all points from the grid in a *design matrix*, $\mathbf{Y} \in^{n\times p}$. Using Hooke's law, we can compute the potential energy of the system for a particular deformed configuration of the points, $\mathbf{Y}$,

$$E(\mathbf{Y}, \boldsymbol{\mathcal{K}}) = \frac{1}{2}\sum_{i,j} \kappa_{i,j}(\mathbf{y}_{i,:} - \mathbf{y}_{j,;})^\top (\mathbf{y}_{i,:} - \mathbf{y}_{j,:})$$

To model paper of uniform density the nonzero spring constants would all be fixed to the same value. However, in our derivations we will consider the general case where they can vary in value as it will prove useful when we extend our analogy.

The potential energy of this spring network is given by the sum of the squared displacement of the springs, multiplied by the spring constants. We can rewrite this energy in matrix form as

$$E(\mathbf{Y}, \boldsymbol{\mathcal{K}}) = \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\mathcal{K}}\mathbf{D}\right).$$

Such spring systems are commonly represented in the form of a *stiffness matrix*, which has the same form as our Laplacian, $\mathbf{L}$. The elements of this matrix are given by

$$\ell_{i,j} = -\frac{1}{2}(\kappa_{i,j} + \kappa_{j,i}) \text{ if } i \neq j \quad (5.3)$$

$$\ell_{i,i} = \frac{1}{2}\sum_{j\neq i}(\kappa_{i,j} + \kappa_{j,i}). \quad (5.4)$$

Since the squared distance matrix is symmetric with diagonal elements set to zero we can substitute the $\boldsymbol{\mathcal{K}}$ directly with the negative Laplacian form in our expression of the potential energy,

$$E(\mathbf{Y}, \boldsymbol{\mathcal{K}}) = \mathrm{tr}\left(\mathbf{Y}\mathbf{Y}^\top\mathbf{L}\right),$$

As for the maximum entropy formulation, this effect is achieved by construction—each diagonal element in the stiffness matrix is set to be the negative sum of the corresponding row/column. Recall that the reason that we can set these diagonal elements arbritarily is that their contribution to the trace comes through multiplication by the diagonal elements of $\mathbf{D}$. Since these elements are all zero, the diagonal can be set arbitrarily. The Laplacian form is convenient as it allows us to switch between expressing the energy in terms of the data point positions and the interpoint squared distances.

A probability density can be formed from the energy by exponentiating the negative energy and normalizing[3],

$$p(\mathbf{Y}) \propto \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{L}\mathbf{Y}\mathbf{Y}^\top\right)\right).$$

[3]Where we have multiplied by a factor of half to match the standard Gaussian form.

The result matches our maximum entropy formulation in the limit as $\gamma \to 0$. Expressing the spring energy in the form of a Gaussian random field is equivalent to assuming that the lattice of springs has been submerged in a heat bath and points in the lattice are being randomly disturbed by collisions with thermally excited particles. A sample from the distribution is equivalent to a sample from the stationary distribution of this system. Note also the relationship with the Gaussian process latent variable model [Lawrence, 2005]. That model also specifies a Gaussian random field across the data points, but one that is derived by considering Gaussian process mappings from a set of latent variables to the data. This contrasts with our random field which was derived by assuming the latent points live on a lattice connected by springs.

Chapter 6

# Nonlinear Probabilistic Dimensionality Reduction

In chapter **??** we showed how linear dimensionality reduction models can be constructed using a probabilistic model, where our reduced dimensional representation, $\mathbf{X}$, is take to be *latent variables*. We assumed a linear relationship between these latent variables and our data,

$$y_{i,j} = \mathbf{w}_{j,:}^{\top}\mathbf{x}_{i,:} + \boldsymbol{\mu} + \epsilon_{i,j},$$

implying a Gaussian likelihood of the form

$$p(\mathbf{y}_{i,:}|\mathbf{W}, \mathbf{x}_{i,:}, \sigma^2) = \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:}, \sigma^2\mathbf{I}\right)$$

which can be written in matrix form as

$$\mathbf{Y} = \mathbf{W}\mathbf{X} + \boldsymbol{\mu}\mathbf{1}^{\top} + \boldsymbol{E}$$

For probabilistic principal component analysis the latent variables were assumed to be drawn from a spherical Gaussian prior distribution,

$$x_{i,j} \sim \mathcal{N}\left(0, 1\right),$$

that allows the nuisance variables, $\mathbf{X} = \{\mathbf{x}_{i,:}\}_{i=1}^{n}$ to be marginalized analytically (Section **??**) obtaining

$$p(\mathbf{Y}|\mathbf{W}) = \mathcal{N}\left(\mathbf{y}_{i,:}|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^{\top} + \sigma^2\mathbf{I}\right),$$

which is the marginal likelihood.

However, this constrained our data to be related to our latent variables in a linear way. This constraint is very strong, and is clearly often violated in practise. Nonlinear dimensionality reduction involves assuming a nonlinear relationship between the latent variables and each observed data dimension. So we have

$$y_{i,j} = f_j(\mathbf{x}_{i,:}) + \epsilon_{i,j}$$

where $\epsilon_{i,j}$ is some corrupting noise, typically independently and identically distributed. The linear special case of assuming

$$f_j(\mathbf{x}) = \mathbf{w}_{j,:}^\top \mathbf{x}_{i,:} + \boldsymbol{\mu}$$

that we discussed in chapter **??** combined with a Gaussian assumption for the noise variables,

$$\epsilon_{i,j} \sim \mathcal{N}\left(0, \sigma^2\right)$$

Consider, for example, the rotated sixes example we introduced in chapter 3. They were projected onto their principal axes and a clear one dimensional nonlinear pattern, in the form of a circle was observed. The data are one dimensional in terms of latent representation: they were generated by a rotation of the original images which relied on a singled parameter, the angle of rotation, but because we represented them with a linear low dimensional space we required more dimensions to represent the data. This motivates the need for nonlinear low dimensional representations, before we consider this further though, we first introduce a standard approach to representing nonlinear functions through a *basis.*

## 6.1 Basis Function Representations

A common approach to regression is to specify that a function is given by a linear sum over a fixed basis set,

$$f\left(\mathbf{x}_{i,:}; \mathbf{w}\right) = \sum_{k=1}^{M} w_k \phi_k\left(\mathbf{x}_{i,:}\right), \tag{6.1}$$

In this equation there are $M$ basis functions, and the $k$th basis function is represented by $\phi_k(\cdot)$ and $\mathbf{w} = \left[w_1, \dots, w_M\right]^\top$. Basis functions can take several forms, the idea of the basis function is to map the data into a feature space, from which a linear sum over the basis leads to a non linear function. A common *local basis* is the *radial basis function* where

$$\phi_k\left(\mathbf{x}_i\right) = \exp\left(-\frac{\left|\mathbf{x}_i - \boldsymbol{\mu}_k\right|^2}{2\ell^2}\right),$$

where each basis function is centered at $\boldsymbol{\mu}_k$, and has radius of $\ell$. A set of such basis functions is visualized in figure 6.1 is the center associated with the $k$th

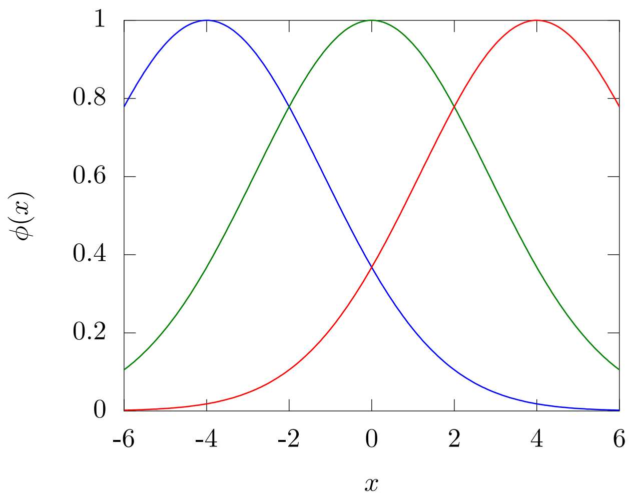


Figure 6.1: A set of radial basis functions with width $\ell = 2$ and location parameters $\boldsymbol{\mu} = [-4 \;\; 0 \;\; 4]^\top$.

basis function. Weighting these basis functions using a matrix $\mathbf{W}$,

$$f_j(\mathbf{x}_i) = \sum_{k=1}^{M} w_{j,k}\phi_k(\mathbf{x}_i)$$

provides us with a function. To demonstrate the type of basis functions shown in figure 6.1 we can sample the matrix $\mathbf{W}$ from a Gaussian density,

$$w_{j,k} \sim \mathcal{N}(0, \alpha),$$

to give us the parameters. We can then visualize the resulting functions by plotting them as in figure 6.2

We can make use of these functions to map, for example, to map from a $q = 2$ to a $p = 3$ dimensional space.

By the probabilistic model

can then be specified by adding noise,

$$y(\mathbf{x}_i) = f(\mathbf{x}_i; \mathbf{w}) + \epsilon_i,$$

where $\epsilon_i$ is the noise associated with the $i$th data point. If the noise is taken to be Gaussian distributed with variance $\sigma^2$,

$$\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right),$$

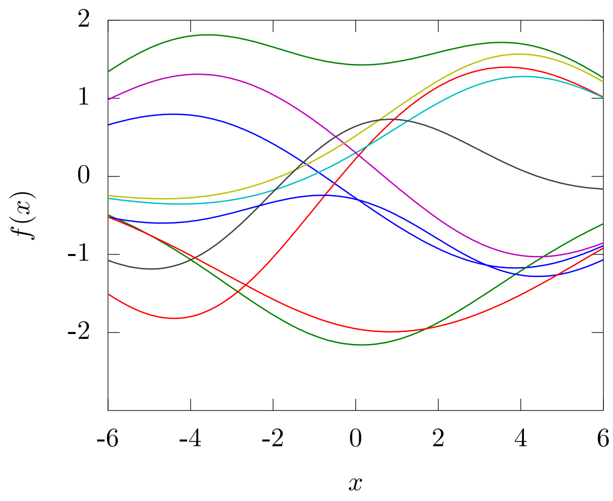


Figure 6.2: Functions sampled using the basis set from figure 6.1. Each line is a separate sample, generated by a weighted sum of the basis set. The weights, $\mathbf{w}$ are sampled from a Gaussian density with variance $\alpha = 1$.

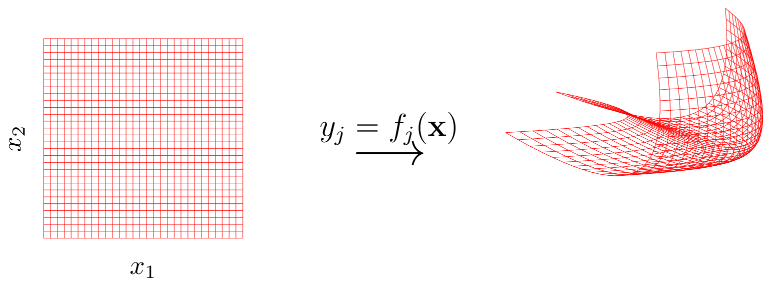


Figure 6.3: A three dimensional manifold formed by mapping from a two dimensional space to a three dimensional space.

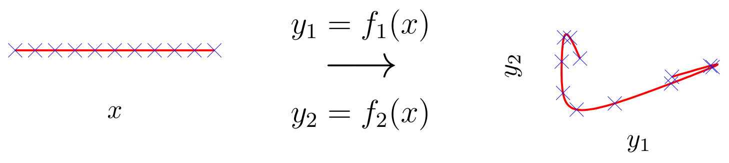


Figure 6.4: A string in two dimensions, formed by mapping from one dimension, $x$, line to a two dimensional space, $[y_1,\ y_2]$ using nonlinear functions $f_1(\cdot)$ and $f_2(\cdot)$.

then we can write down the following probabilistic representation for our data,

$$p\left(\mathbf{y}|\mathbf{X},\mathbf{w},\sigma^2\right) = \prod_{i=1}^{M} \mathcal{N}\left(y_i|w_i,\sigma^2\right),$$

where the mean of the Gaussian distributions in the product is given by the evaluations of our function at the training points, $w_i = f\left(\mathbf{x}_i;\mathbf{w}\right)$.

This looks very similar to the representation we used in Section **??**. However, there is one important difference. We have made use of a parameter vector in the representation above. To determine the parameters, we might want to use Bayesian inference, for computational convenience placing a zero mean Gaussian prior over $\mathbf{w}$ with covariance matrix $\gamma'\mathbf{I}$,

$$p\left(\mathbf{w}\right) = \mathcal{N}\left(\mathbf{w}|\mathbf{0},\gamma'\mathbf{I}\right).$$

By constructing a design matrix from our basis functions, $\mathbf{\Phi} = [\boldsymbol{\phi}_1,\dots,\boldsymbol{\phi}_m]$, where $\boldsymbol{\phi}_j = \left[\phi_j\left(\mathbf{x}_1\right),\dots,\phi_j\left(\mathbf{x}_n\right)\right]^\top$ is the vector containing the values of the $j$th basis function at all input data points, we can rewrite (6.1) for the training data in matrix vector form,

$$\mathbf{f} = \mathbf{\Phi}\mathbf{w}.$$

## 6.2 Nonlinear Probabilistic Approaches and Normalization

The difficulty for probabilistic approaches to dimensionality reduction is that the generative model the proscribe is difficult to normalize. If we assume that the prior distribution is Gaussian,

$$\mathbf{x}_{i,:} \sim \mathcal{N}\left(\mathbf{0},\mathbf{I}\right)$$

and that the mapping for each data point is given by

$$y_{i,j} = f_j(\mathbf{x}_{i,:}) + \epsilon_{i,j}$$

with a standard Gaussian noise assumption,

$$\epsilon_{i,j} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\right)$$

then the likelihood of the data given the latent variables is given by

$$p(\mathbf{y}_{i,:}|\mathbf{x}_{i,:}) = \prod_{j=1}^{p} \mathcal{N}\left(y_{i,j}|f_j(\mathbf{x}_{i,:}), \sigma^2\mathbf{I}\right).$$

The marginalized likelihood of the data then has the form

$$\begin{aligned} p(\mathbf{y}_{i,:}) = & \int p(\mathbf{y}_{i,:}, \mathbf{x}_{i,:})\mathrm{d}\mathbf{x}_{i,:} \\ & \int p(\mathbf{y}_{i,:}|\mathbf{x}_{i,:})p(\mathbf{x}_{i,:})\mathrm{d}\mathbf{x}_{i,:} \\ & \int \prod_{j=1}^{p} \mathcal{N}\left(y_{i,j}|f_j(\mathbf{x}_{i,:}), \sigma^2\right) \mathcal{N}\left(\mathbf{x}_{i,:}|\mathbf{0}, \mathbf{I}\right) \mathrm{d}\mathbf{x}_{i,:}. \end{aligned}$$

The key component of this integral is in the exponent of the joint density. Taking the logarithm we can see that the exponent has the form

$$\log p(\mathbf{y}_{i,:}, \mathbf{x}_{i,:}) = -\frac{1}{2}\sum_{j=1}^{p}(y_{i,j} - f_j(\mathbf{x}_{i,:}))^2 - \frac{1}{2}\mathbf{x}_{i,:}^{\top}\mathbf{x}_{i,:} + \text{const}$$

where we have introduced a constant term that does not depend on $\mathbf{x}_{i,:}$. For the linear case, where we have $f_j(\mathbf{x}) = \mathbf{w}_{j,:}^{\top}\mathbf{x}$, we can rewrite this exponent as

$$\log p(\mathbf{y}_{i,:}, \mathbf{x}_{i,:}) = -\frac{1}{2}\mathbf{y}_{i,:}^{\top}\mathbf{y}_{i,:} + \mathbf{y}_{i,:}^{\top}\mathbf{W}\mathbf{x}_{i,:} - \frac{1}{2}\mathbf{x}_{i,:}^{\top}\mathbf{W}^{\top}\mathbf{W}\mathbf{x}_{i,:} - \frac{1}{2}\mathbf{x}_{i,:}^{\top}\mathbf{x}_{i,:} + \text{const}$$

which is a quadratic form implying that integrating over $\mathbf{x}_{i,:}$ to find the marginal likelihood $p(\mathbf{y}_{i,:})$ *is* tractable. This is the integral that is performed in probabilistic PCA and factor analysis (see chapter **??**). However, for more general nonlinear functions,[1] $f(\cdot)$, this integral is *not* tractable. The motivation for using general nonlinear functions is that they do lead to a much richer class of probability densities, for example, even in the special case where we don't perform any dimensionality reduction, for example $p = 1$ and $q = 1$, we can plot the distribution over $x$ and make a numerical estimate of the corresponding density over $y$. This is done in figure 6.5. Placing the density through the nonlinear function leads to a more complex multimodal density. However, the price we pay for the greater representational power of the model is the inability to express the marginal likelihood for $p(\mathbf{y}_{i,:})$ in a closed form, to fit such models we need to look for approximations.

[1]There are some specific nonlinear functions which keep the integral tractable. For example if $y = \exp(x)$ we can perform the integral and the resulting density over $y$ is known as the *log normal*.

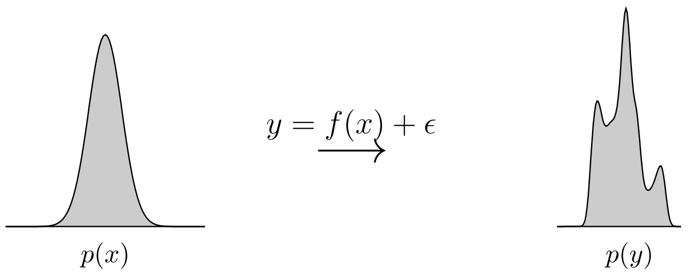


Figure 6.5: A Gaussian distribution propagated through a non-linear mapping. We define a one dimensional Gaussian distribution in $x$ (left). Then the relationship between $y$ and $x$ is defined as $y_i = f(x_i) + \epsilon_i$ where $\epsilon$ is Gaussian random noise with standard deviation $\sigma = 0.2$ and $f(\cdot)$ is computed using an RBF basis with 100 centres uniformly distributed between -4 and 4 and basis function widths $\ell = 0.1$. The new distribution over $y$ (right) is multimodal and difficult to normalize. It is this normalization problem that provides problems for models based on such nonlinear functions.

## 6.3 Density Networks

Density networks were introduced by MacKay [1995] as an approximation for fitting such models. Density networks make use of a sample based approximation to the marginal likelihood to represent the data. Making explicit dependence of the mapping function, $f_j(\mathbf{x}_{i,:};\ \boldsymbol{\theta})$, on its parameters, $\boldsymbol{\theta}$, we have

$$p\left(\mathbf{Y}|\mathbf{X}\right) = \prod_{i=1}^{n}\prod_{j=1}^{p}\mathcal{N}\left(y_{i,j}|f_j(\mathbf{x}_{i,:};\ \boldsymbol{\theta}), \sigma^2\right)$$

and typically we would chose a Gaussian prior for the latent space (although we can choose any distribution here from which we can sample),

$$p\left(\mathbf{X}\right) = \mathcal{N}\left(\mathbf{x}_{i,:}|\mathbf{0}, \mathbf{I}\right)$$

### 6.3.1 Sample Based Approximation

The sample based approximation involves replacing the continuous integral over the density with a discrete sum of samples from the density. Given a set of $m$ samples, $\{\hat{\mathbf{x}}_{s,:}\}_{s=1}^{m}$ drawn from $p(\mathbf{x}_{i,:})$ we can write down the sample based

**boxfloat 6.1** Sample Based Approximations

approximation to the marginal likelihood for the $i$th data point as

$$p(\mathbf{y}_{i,:}) = \int \prod_{j=1}^{p} p\left(y_{i,j}|\mathbf{x}_{i,:},\boldsymbol{\theta}\right) p(\mathbf{x}_{i,:})\mathrm{d}\mathbf{x}_{i,:}$$
$$\approx \frac{1}{m}\sum_{s=1}^{m}\prod_{j=1}^{p} p\left(y_{i,j}|\hat{\mathbf{x}}_{s,:},\boldsymbol{\theta}\right)$$

where we have been explicit about including the parameter vector, $\boldsymbol{\theta}$, in the likelihood and the approximation becomes more accurate as our sample size, $m$, goes towards infinity, see box 6.1 for more details on sample based approximations. The joint likelihood of the entire data set is given by a product over the likelihoods for each data point,

$$p(\mathbf{Y}|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(\mathbf{y}_{i,:}|\boldsymbol{\theta})$$

reflecting an assumption that the data is independent and identically distributed *given* the parameters, $\boldsymbol{\theta}$. So to compute the likelihood of the entire data set, $\mathbf{Y}$, we need to consider sample based approximations for every data point. The key innovation in density networks is to use the *same set* of samples for each data point. This turns out to significantly reduce the computational demands. Taking the logarithm and substituting with our approximation we have

$$\log p\left(\mathbf{Y}|\boldsymbol{\theta}\right) = \sum_{i=1}^{n} \log \frac{1}{m}\sum_{s=1}^{m} p\left(\mathbf{y}_{i,:}|\hat{\mathbf{x}}_{s,:},\boldsymbol{\theta}\right).$$

### 6.3.2 Maximizing the Approximate Likelihood

We can directly maximize the approximate likelihood by taking its gradients with respect to the parameters, $\boldsymbol{\theta}$,

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log p\left(\mathbf{y}_{i,:}|\boldsymbol{\theta}\right) = \sum_{s=1}^{m}\frac{p\left(\mathbf{y}_{i,:}|\hat{\mathbf{x}}_{s,:},\boldsymbol{\theta},\right)}{\sum_{s'=1}^{m} p\left(\mathbf{y}_{i,:}|\hat{\mathbf{x}}_{s',:},\boldsymbol{\theta}\right)}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log p\left(\mathbf{y}_{i,:}|\hat{\mathbf{x}}_{s,:},\boldsymbol{\theta}\right).$$

For convenience we define,

$$r_{i,s} = \frac{p\left(\mathbf{y}_{i,:}|\boldsymbol{\theta},\hat{\mathbf{x}}_{s,:}\right)}{\sum_{s'=1}^{m} p\left(\mathbf{y}_{i,:}|\boldsymbol{\theta},\hat{\mathbf{x}}_{s',:}\right)},$$

which, as we will see has multiple interpretations depending on the context. These include the posterior probability of mixture component membership

**boxfloat 6.2** Importance Sampling

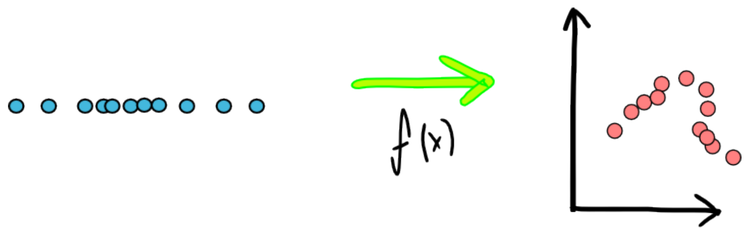


Figure 6.6: One dimensional Gaussian mapped to two dimensions.

(sometimes referred to as responsibility) and importance sampling weights, see box 6.2. This allows us to write the gradient of the log likelihood as a weighted sum of gradients of the likelihood as follows

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log p\left(\mathbf{y}_{i,:}|\boldsymbol{\theta}\right) = \sum_{s=1}^{m} r_{i,s} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log p\left(\mathbf{y}_{i,:}|\boldsymbol{\theta}, \hat{\mathbf{x}}_{s,:}\right).$$

### 6.3.3 Density Network Mappings

The formalism for density networks allows any general $f(\cdot)$ but at the time of publication *multi-layer perceptron* models were particularly popular so it probably seemed natural to consider functions based on these models. These can be seen as basis function models where each basis functions is given by

$$\phi_k(\mathbf{x}_{i,:}; \mathbf{v}_{k,:}) = \frac{1}{1 + \exp(-\mathbf{v}_{k,:}^\top \mathbf{x}_{i,:})}.$$

The parameters of the basis functions, $\mathbf{V} = \{\mathbf{v}_{k,:}\}_{k=1}^{M}$, can be optimized along with the matrix $\mathbf{W}$.

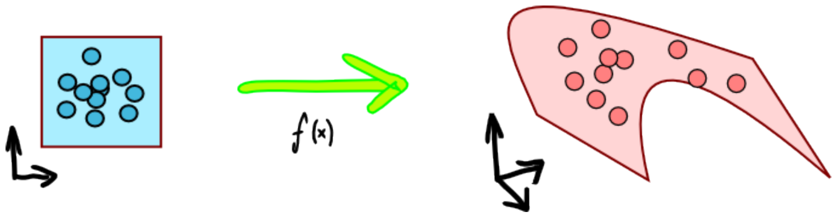


Figure 6.7: Two dimensional Gaussian mapped to three dimensions.

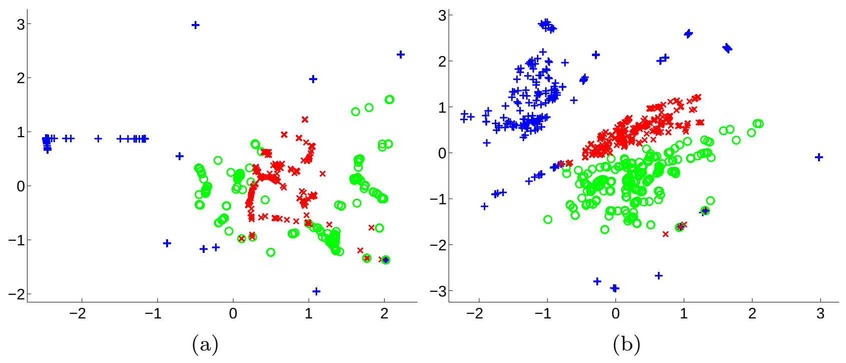


Figure 6.8: Oil data visualised with a density network using an MLP network with (a) 100 (b) 400 points in the sample. Nearest neighbour errors: (a) 22 (b)16. Code can be run with (a) `demOilDnet4` (b) `demOilDnet5`

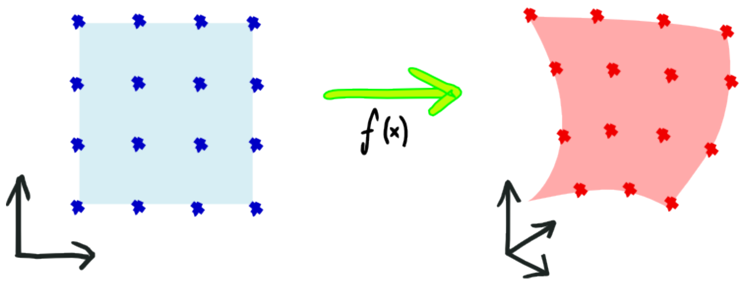


Figure 6.9: One dimensional Gaussian mapped to two dimensions.

## 6.4 Likelihood Optimisation

### 6.4.1 Example: Oil Data

## 6.5 Generative Topographic Mapping

Generative Topographic Mapping (GTM) **?**

Key idea: Lay points out on a *grid*.

Constrained mixture of Gaussians.

### 6.5.1 GTM Prior

Prior distribution is a mixture model in a latent space.

$$p(\mathbf{X}) = \prod_{i=1}^{n} p(\mathbf{x}_{i,:})$$

$$p(\mathbf{x}_{i,:}) = \frac{1}{m}\sum_{s=1}^{m} \delta(\mathbf{x}_{i,:} - \hat{\mathbf{x}}_{s,:})$$

The $\hat{\mathbf{x}}_{s,:}$ are laid out on a regular grid.

### 6.5.2 Mapping and E-Step

Likelihood is a Gaussian with non-linear mapping from latent space to data space for the mean

$$p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{n}\prod_{j=1}^{p} \mathcal{N}\left(y_{i,j}|w_j(\mathbf{x}_{i,:}; \mathbf{W}, \ell), \sigma^2\right)$$

In the original paper Bishop et al. [1998] an RBF network was suggested,

In the E-step, posterior distribution over $k$ is given by

$$r_{i,k} = \frac{\prod_{j=1}^{p} \mathcal{N}\left(y_{i,j}|f_j(\hat{\mathbf{x}}_k; \mathbf{W}, \ell), \sigma^2\right)}{\sum_{s=1}^{m}\prod_{j=1}^{p} \mathcal{N}\left(y_{i,j}|f_j(\hat{\mathbf{x}}_s; \mathbf{W}, \ell), \sigma^2\right)}$$

sometimes called the "responsibility of component $k$ for data point $i$".

### 6.5.3 Likelihood Optimisation

We then maximise the lower bound on the log likelihood,

$$\log p(\mathbf{y}_{i,:}|\boldsymbol{\theta}) \geq \left\langle \log p(\mathbf{y}_{i,:}, \hat{\mathbf{x}}_{s,:}|\boldsymbol{\theta})\right\rangle_{q(s)} - \left\langle \log q(s)\right\rangle_{q(s)},$$

Free energy part of bound

$$\left\langle \log p(\mathbf{y}_{i,:}, \hat{\mathbf{x}}_{s,:}|\boldsymbol{\theta})\right\rangle = \sum_{s=1}^{m} r_{i,s} \log p(\mathbf{y}_{i,:}|\hat{\mathbf{x}}_{s,:}, \boldsymbol{\theta}) + \text{const}$$

When optimising parameters in EM, we ignore dependence of $r_{i,k}$ on parameters. So we have

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left\langle \log p(\mathbf{y}_{i,:}, \hat{\mathbf{x}}_{s,:}|\boldsymbol{\theta})\right\rangle = \sum_{s=1}^{m} r_{i,s}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log p(\mathbf{y}_{i,:}|\hat{\mathbf{x}}_{s,:}, \boldsymbol{\theta})$$

which is very similar to density network result! Interpretation of posterior is slightly different.

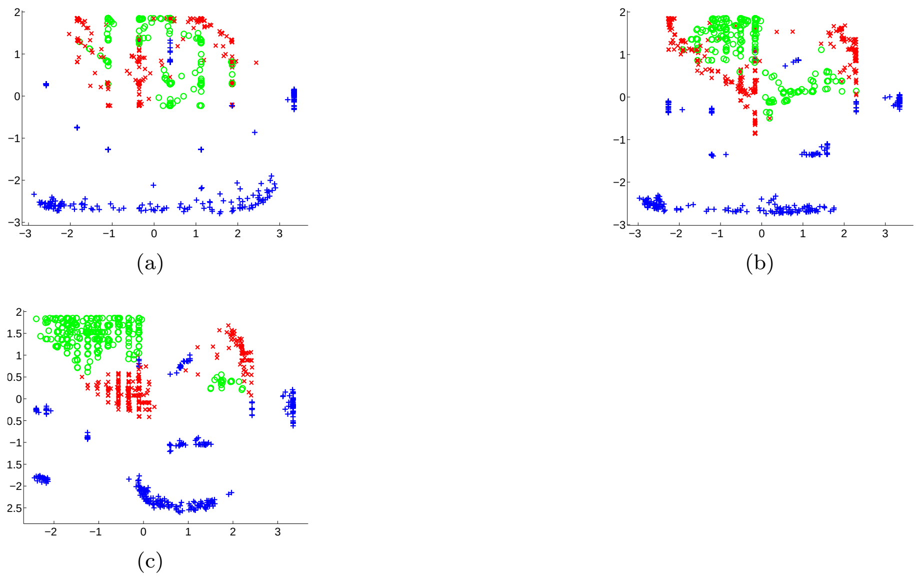


Figure 6.10: Oil data visualised with the GTM using an RBF network with (a) 10×10 (b) 20 × 20 and (c) 30 × 30 points in the grid. Nearest neighbour errors: (a) 74 (b) 44 (c) 11. These experiments can be recreated with (a) `demOilDnet1` (b) `demOilDnet2` (c) `demOilDnet3`.

### 6.5.4 Oil Data

### 6.5.5 Magnification Factors

**?**

### 6.5.6 Stick Man Data

### 6.5.7 Separated Means: Bubblewrap Effect

### 6.5.8 Equivalence of GTM and Density Networks

GTM and Density Networks have the same origin. Bishop et al. [1996], MacKay [1995].

In original Density Networks paper MacKay suggested Importance Sampling MacKay [1995].

Early work on GTM also used importance sampling.

Main innovation in GTM was to lay points out on a grid (inspired by Self Organizing Maps Kohnonen [2001].

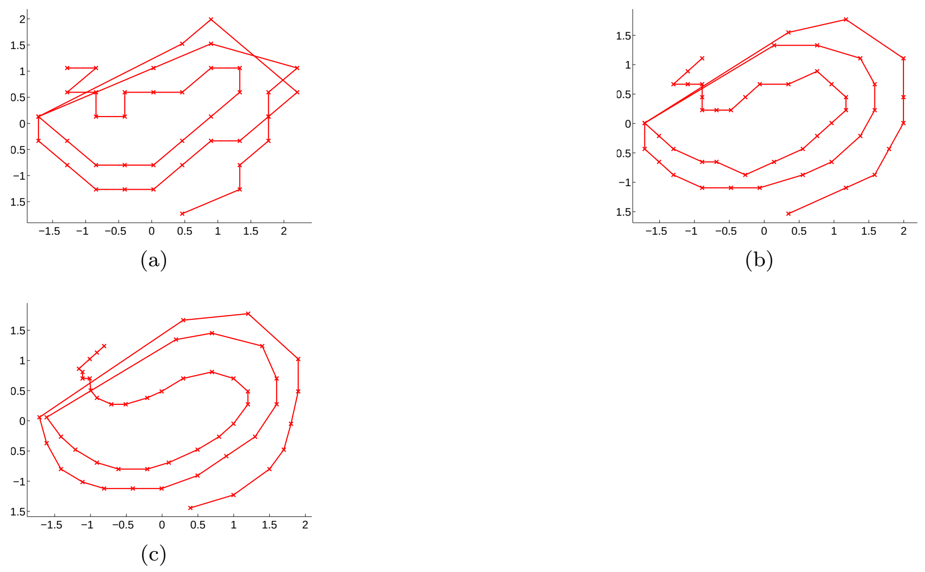


Figure 6.11: Oil data visualised with the GTM using an RBF network with (a) 10×10 (b) 20 × 20 (c) 30 × 30 points in the grid. Experiments can be recreated with (a) `demStickDnet1` (b) `demStickDnet2` (c) `demStickDnet3`.

## 6.6 Non Linear Factor Analysis

Variational approach to dimensionality reduction.

Combine Gaussian prior over latent space with neural network Honkela and Valpola [2005]

Assume variational prior separates. Optimise with respect to variational distributions.

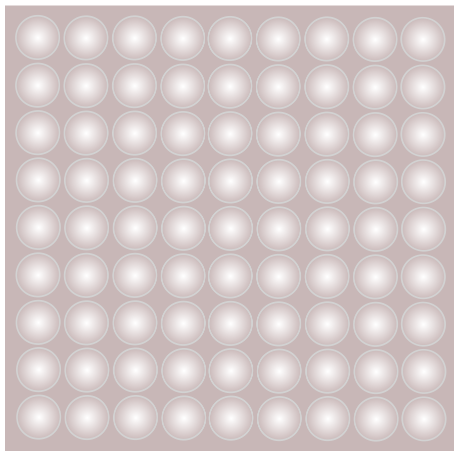
### 6.6.1 Summary

Two point based approaches to dimensionality reduction.

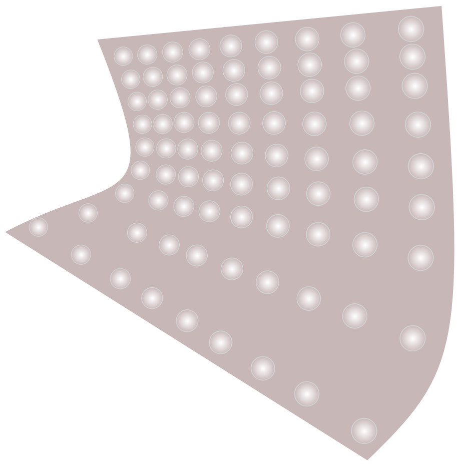
Approaches seem to generalise well even when dimensions of data is greater than number of points.

Approaches are difficult to extend to higher dimensional latent spaces: number of samples/centres required increases exponentially with dimension.

Next we will explore a different probabilistic interpretation of PCA and extend that to non-linear models.

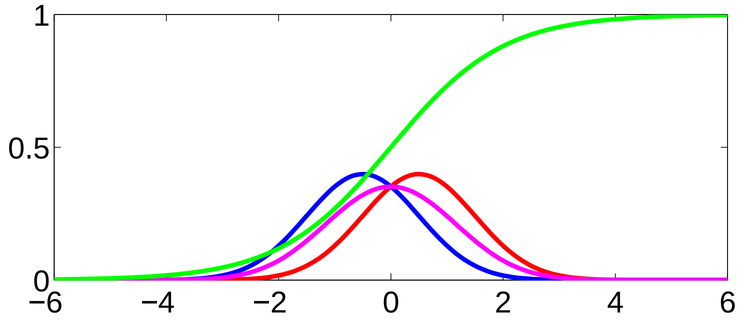


(a)

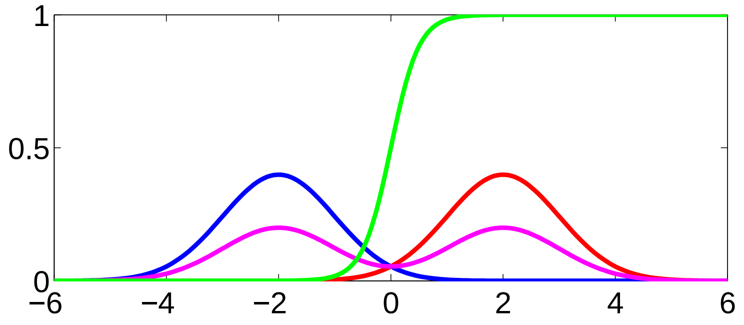


(b)

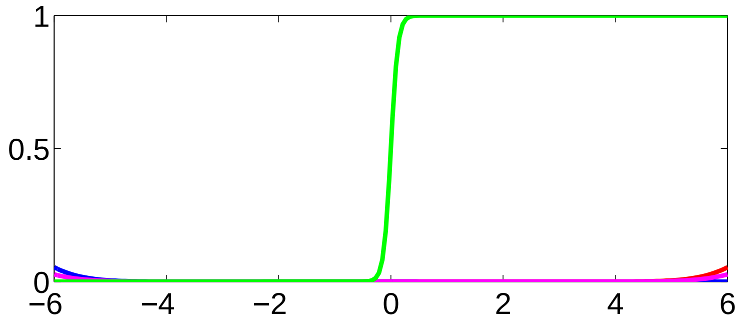
Figure 6.12: The manifold is more like bubblewrap than a piece of paper.



(a)



(b)



(c)

Figure 6.13: As Gaussians become further apart the posterior probability becomes more abrupt. (a) 1 (b) 4 (c) 16 standard deviations apart.

# Glossary

**centering matrix** the centering matrix, denoted in this book by $\mathbf{H}$, is a matrix which, when premultiplying a design matrix returns a centered version of the data, where the center is defined by the sample mean of the data set. 56

**design matrix** The design matrix is the standard way of representing a data set in statistics. It involves placing the data in a matrix, where each row of the matrix is a separate data point. Different features are then given in each column of the design matrix. 44

**diagonalize** In matrix algebra, diagonalization is the process of converting a square matrix into a diagonal matrix, typically through rotation. 89

**entropy** The entropy of a distribution is given by the expectation of its negative log likelihood $E(p(x)) = -\left\langle \log p(x) \right\rangle_{p(x)}$. 112

**error function** In machine learning error function is often used to refer to the objective function that is being minimized. ***How does this relate to Gauss and Laplace's use of the word error function and the erf?***. 54

**features** Features in machine learning are characteristics of the data, often represented by numbers. For a human being some features might be height (in centimeters), weight (in kg) and eye color. For eye color a one of-$n$ (or nominal) encoding might be used. 13

**Gaussian random field** a Gaussian distribution where there is typically a neighborhood (often spatial) relationship between the variables governed by the random field. The conditional relationships for each variable are defined only in terms of its neighbors, leading to a sparse inverse covariance for the Gaussian distribution. 114

**generative model** Generative models in machine learning are models which explicitly attempt to model a generation process for the data. Often, it will not be a close approximation to the actual generating process, but would typically be composed of tractable probability densities, perhaps formed in a hierarchical manner, from which data are sampled. For example in speech processing, the cepstral coefficients are often modeled as being generated from a discrete Markov chain which selects Gaussian components from which the coefficients are sampled. Generative models are often contrasted with *discriminative models* (such as the support vector machine) which do not seek to model the generating process directly, but may provide a mapping from features directly to a categorization for a data point. 43

**Jensen's inequality** ***More detail*** is an inequality between the sum of a convex function and the convex function of a sum. In probability it is perhaps most used to relate the sum of logarithms with the logarithm of a sum,

$$\sum_i \log f_i(x) \leq \log \sum_i f_i(x).$$

. 53

**Lagrangian** The Lagrangian is the name given to an objective function composed of an original objective function with additional terms that include Lagrange multipliers (see box 3.2) associated with any constraints imposed on the system. See box 3.2. 85

**latent variable** A latent variable is a variable in a model that is not observed. Such a latent variable can either be real, or simply explanatory. For example, body mass index is a latent variable which explains the relationship between height and weight of an individual. This is an explanatory latent variable. Early work on principal component analysis and factor analysis sought to summarize intelligence through a reduced number of explanatory latent variables. Real latent variables occur in, for example, state space models. In a state space model the latent variables are often physical properties of the system: such as the speed, velocity, and acceleration of an aircraft. 43

**likelihood** The likelihood of a data set is the probability density that relates the data to the parameters and potentially the latent variables. In Bayesian inference the likelihood is combined with the prior to find the posterior density and the marginal likelihood of the data. 49

**marginal likelihood** The marginal likelihood is a term used to refer to a likelihood from which one or more of the parameters have been marginalized. It is also called the evidence Jaynes [2003]. The marginal likelihood is often used in model selection, in this context ratios between marginal likelihoods are sometimes known as Bayes factors Raftery [1996]. Maximization of the marginal likelihood with respect to its parameters is sometimes called empirical Bayes, but it can also just be seen as another variant of maximum likelihood. 46

**maximum entropy principal** Maximum entropy is a framework for deriving probability distributions given some constraints on the moments of those distributions. The basic idea is to look for a class of distributions which respects those moment constraints, but has the largest entropy of all distributions which respect those constraints. 113

**maximum likelihood** Maximum likelihood is a technique for fitting a probabilistic model to a data set. To perform maximum likelihood, the probability of the data given the parameters is first written down. This is known as the likelihood. This likelihood is then maximized with respect to the parameters. Normally, rather than maximizing the likelihood directly, we maximize the logarithm of the likelihood (the log likelihood). Maximizing the log likelihood is equivalent to maximizing the likelihood because logarithm is a *monotonic* transformation. Very often the log likelihood has a slightly simpler form than the likelihood (e.g. in the case of a Gaussian likelihood it is quadratic in the data, instead of an exponentiated quadratic). 43

**noise** Noise is a catch all term with origins in signal processing. It was originally used to describe unwanted perturbation to the signal. In a generative model, it is typically used to represent aspects of the data that we are not explicitly modeling with our generating process. 43

**posterior** The posterior is a density or distribution which reflects our knowledge about a parameter or variable having observed some data. It is computed through Bayes' rule and depends on the form of the likelihood and the prior. 49

**preprocessing** is a term used to describe operations applied to data before a further algorithm is applied. Dimensionality reduction is often used as a preprocessing step for classification or regression.. 14

**principal eigenvectors** The principal eigenvectors of a matrix are those associated with the highest eigenvalues of the matrix. 86

**prior** The prior is a density or distribution which encapsulates our assumptions about a variable or parameter before we have seen the data. 49

**prior distribution** The prior distribution (or density) encapsulates our believes about the values of a parameter, or variable, before we have seen the data. For statisticians the use of the prior density is one of the more controversial aspects of Bayesian inference as it can introduce subjectivity into statistical analysis. However, the use of prior distributions can be very powerful as it can allow us to include assumptions (such as smoothness) in the model. 46

**sample mean** A sample mean is the value obtained when the data points are summed together and divided by the total number of data points. It is a sample based approximation of the true mean, which is the expected value of a data point. 56

**spherical covariance matrix** A spherical covariance matrix is a covariance matrix (typically for a Gaussian density) where all data points sampled from the density are independent (i.e. off diagonal terms are zero) and share the same variance (i.e. on diagonal terms are constant). It is termed spherical because contours of equal likelihood from the associated Gaussian density are in the form of circles (for 2D Gaussians), spheres (for 3D Gaussians), and hyper-spheres (for higher dimensionality). Gaussians with other forms of covariance matrix have elliptical contours of equal likelihood. 45

**standard transformation** in classical multidimensional scaling is a way of transforming a distance matrix into a similarity matrix. If the similarity is a positive definite matrix it is equivalent to the distance in the "feature space" that the similarity matrix implies. Alternatively it can be thought of as the square root of the expected squared distance under a Gaussian random field governed by the similarity as a covariance. 91

**total variance** The total variance associated with a covariance matrix is the sum of the diagonal variance elements. It is given by the trace of the covariance matrix and it is invariant to rotation of the data set. 81

# Bibliography

Raymond C. Archibald. A rare pamphlet of Moivre and some of his discoveries. *Isis*, 8:671–676, 1926. Cited on p. 151.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. doi: 10.1162/089976603321780317. Cited on pp. 117 and 123.

Christopher M. Bishop. Bayesian PCA. In Michael J. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 482–388, Cambridge, MA, 1999a. MIT Press. Cited on pp. 70 and 77.

Christopher M. Bishop. Variational principal components. In *Proceedings Ninth International Conference on Artificial Neural Networks, ICANN'99*, volume 1, pages 509–514, 1999b. Cited on pp. 70 and 77.

Christopher M. Bishop and Gwilym D. James. Analysis of multiphase flows using dual-energy gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research*, A327:580–593, 1993. doi: 10.1016/0168-9002(93)90728-Z. Cited on p. 28.

Christopher M. Bishop, Marcus Svensén, and Christopher K. I. Williams. A fast EM algorithm for latent variable density models. In David Touretzky, Michael Mozer, and Mark Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 465–471, Cambridge, MA, 1996. MIT Press. Cited on p. 142.

Christopher M. Bishop, Marcus Svensén, and Christopher K. I. Williams. GTM: the Generative Topographic Mapping. *Neural Computation*, 10(1):215–234, 1998. doi: 10.1162/089976698300017953. Cited on p. 141.

Abraham de Moivre. Approximatio ad summam terminorum binomii $(a+b)^n$ in seriem expansi. Reprinted in Archibald [1926], 1733. Cited on p. 15.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977. Cited on pp. 14 and 71.

Brian D. Ferris, Dieter Fox, and Neil D. Lawrence. WiFi-SLAM using Gaussian process latent variable models. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2480–2485, 2007. Cited on p. 31.

Carl Friedrich Gauss. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solum Ambientium*. Perthes et Besser, Hamburg. Cited on p. 15.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, 1995. Cited on p. 48.

Zoubin Ghahramani and Matthew J. Beal. Variational inference for Bayesian mixtures of factor analysers. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 831–864, Cambridge, MA, 2000. MIT Press. Cited on p. 70.

Zoubin Ghahramani and Geoffrey E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, The University of Toronto, 1997. Cited on pp. 70 and 77.

Russell Greiner and Dale Schuurmans, editors. *Proceedings of the International Conference in Machine Learning*, volume 21, 2004. Omnipress. Cited on pp. 152 and 155.

Jihun Ham, Daniel D. Lee, Sebastian Mika, and Bernhard Schölkopf. A kernel view of dimensionality reduction of manifolds. In Greiner and Schuurmans [2004]. Cited on p. 118.

Antti Honkela and Harri Valpola. Unsupervised variational Bayesian learning of nonlinear models. In Lawrence Saul, Yair Weiss, and Léon Bouttou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 593–600, Cambridge, MA, 2005. MIT Press. Cited on p. 143.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933. Cited on p. 40.

Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. John Wiley and Sons, 2001. ISBN 978-0-471-40540-5. Cited on p. 70.

Edwin T. Jaynes. Bayesian methods: General background. In J. H. Justice, editor, *Maximum Entropy and Bayesian Methods in Applied Statistics*, pages 1–25. Cambridge University Press, 1986. Cited on p. 119.

Edwin T. Jaynes. *Probability Theory: The Logic of Science.* Cambridge University Press, Cambridge, U.K., 2003. ISBN 0-521-59271-2. Cited on p. 148.

Charles Kemp and Joshua B. Tenenbaum. The discovery of structural form. *Proc. Natl. Acad. Sci. USA*, 105(31), 2008. Cited on p. 122.

Teuvo Kohnonen. *Self-Organizing Maps*, volume 30 of *Information Sciences.* Springer-Verlag, Berlin, 3rd edition, 2001. ISBN 3-540-67921-9. Cited on p. 142.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009. ISBN 978-0-262-01319-2. Cited on pp. 123 and 125.

Solomon Kullback and Richard A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951. Cited on p. 53.

Pierre Simon Laplace. Mémoire sur la probabilité des causes par les évènemens. In *Mémoires de mathèmatique et de physique, presentés à lAcadémie Royale des Sciences, par divers savans, & lù dans ses assemblées 6*, pages 621–656, 1774. Translated in Stigler [1986]. Cited on p. 15.

Pierre Simon Laplace. Mémoire sur les approximations des formules qui sont fonctions de tres̀ grands nombres et sur leur application aux probabilités. In *Mémoires de l'Académie des sciences de Paris*, pages 353–415, 1810. Cited on p. 15.

Neil D. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 11 2005. Cited on p. 129.

Neil D. Lawrence. A unifying probabilistic perspective for spectral dimensionality reduction. Technical report, University of Sheffield, 2010. Cited on pp. 117 and 123.

David J. C. MacKay. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research, A*, 354(1):73–80, 1995. doi: 10.1016/0168-9002(94)00931-7. Cited on pp. 137 and 142.

David J. C. MacKay. Maximum likelihood and covariant algorithms for independent component analysis. Unpublished manuscript, available from `http://wol.ra.phy.cam.ac.uk/mackay/homepage.html`, 1996. Cited on p. 70.

David J. C. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, Cambridge, U.K., 2003. ISBN 0-52164-298-1. Cited on p. 79.

Kantilal V. Mardia, John T. Kent, and John M. Bibby. *Multivariate analysis*. Academic Press, London, 1979. ISBN 0-12-471252-5. Cited on pp. 97, 98, 118, and 123.

Geoffrey J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988. Cited on p. 14.

Thomas P. Minka. Automatic choice of dimensionality for PCA. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 598–604, Cambridge, MA, 2001. MIT Press. Cited on pp. 70 and 77.

Anthony O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society, B*, 40:1–42, 1978. Cited on p. 112.

Adrian E. Raftery. *Hypothesis Testing and Model Selection*, chapter 10, pages 164–187. Chapman and Hall, 1996. Cited on p. 149.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006. ISBN 0-262-18253-X. Cited on p. 112.

Sam T. Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 626–632, Cambridge, MA, 1998. MIT Press. Cited on pp. 41 and 70.

Sam T. Roweis and Zoubin Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345, 1999. Cited on p. 70.

Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. doi: 10.1126/science.290.5500.2323. Cited on pp. 117, 124, and 125.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2001. Cited on p. 112.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10: 1299–1319, 1998. doi: 10.1162/089976698300017467. Cited on p. 118.

Paul T. Spellman, Gavin Sherlock, Michael Q. Zhang, Vishwanath R. Iyer, Kirk Anders, Michael B. Eisen, Patrick O. Brown, David Botstein, and Bruce Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, 1998. URL `http://www.molbiolcell.org/cgi/content/full/9/12/3273`. Cited on p. 33.

Stephen M. Stigler. Laplace's 1774 memoir on inverse probability. *Statistical Science*, 1:359–378, 1986. Cited on p. 153.

Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, Boston, MA, USA, 3rd edition, 2000. ISBN 0201700735. Cited on p. 12.

Joshua B. Tenenbaum, Virginia de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290 (5500):2319–2323, 2000. doi: 10.1126/science.290.5500.2319. Cited on p. 126.

Michael E. Tipping and Christopher M. Bishop. Mixtures of principal component analysers. In *Proceedings IEE Fifth International Conference on Artificial Neural Networks, Cambridge, U.K., July.*, pages 13–18, 1997. Cited on p. 77.

Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999a. Cited on p. 70.

Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, B*, 6(3):611–622, 1999b. doi: doi:10.1111/1467-9868.00196. Cited on pp. 41, 56, 64, 66, and 70.

Larry A. Wasserman. *All of Statistics*. Springer-Verlag, New York, 2003. ISBN 9780387402727. Cited on p. 52.

Kilian Q. Weinberger, Fei Sha, and Lawrence K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In Greiner and Schuurmans [2004], pages 839–846. Cited on pp. 117, 118, and 126.

Christopher K. I. Williams. Regression with Gaussian processes. Paper presented at the *Mathematics of Neural Networks and Applications conference*, Oxford, UK, July 1995. Cited on p. 112.

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Semi-supervised learning: From Gaussian fields to Gaussian processes. Technical Report CMU-CS-03-175, Carnegie Mellon University, 2003. Cited on p. 122.