14 Principal Component Analysis

We now turn to consider a form of **unsupervised** learning called Principal Component Analysis (PCA), a technique for dimensionality reduction. The goal of PCA, roughly speaking, is to find a low-dimensional representation of high dimensional data. It is an unsupervised in the sense that we are not given examples of such mappings (or embeddings). Rather, we are given a collection of high-dimsional data (e.g., images), and we need to learn a low-dimensional representation (or latent embedding) and the mapping from the observations space (i.e., the high-dimensional data) to the latent embedding space (i.e., the low-dimensional representtion). The goal of PCA specifically is to find the best possible linear mapping from data to a linear subspace of the observation space.

The fact that an effective low-dimensional representation should exist in the first place is based on the intuition that many high-dimensional data have low-dimensional underlying (physical) causes. In other words, the data often exhibit regularities or redundancy, such that data points can often be modelled well as a linear combination of other data points. PCA is one of the simplest approaches to dimensionality reduction. It is fast and effective, and widely used.

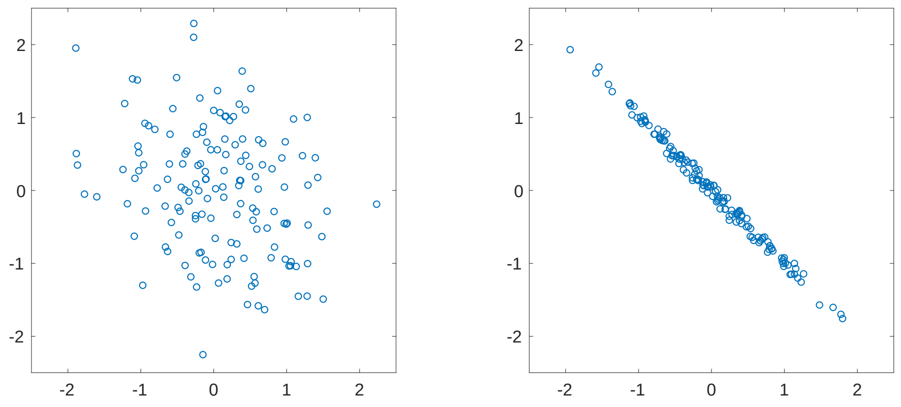

Figure 14.1: Here are two sets of 2D points $\mathbf{y} \in \mathbb{R}^2$. The points in the left scatter plot fill the 2D space while those on the right are well approximated by a 1D subspace (in this case, along the line $y_1 = y_2$. We might therefore approximate the data on the right by projecting it onto a 1D subspace. That is, if we represent each datapoint in terms of the closest point on the line $y_1 = y_2$, then we can represent each 2D point with just one value (i.e., where it is along the line). The error in such a 1D representation would be the distances between points and the line.

Indeed, PCA has been developed independently in several distinct research communities, from different perspectives, and for different tasks. As such, there are different perspectives on its motivation and use:

- **Visualization:** High-dimensional data are extremely hard to visualize, e.g., to see how disjoint two different categories of feature vectors might be, or to see how noisy some measurements are. PCA provides a way to project high-dimensional data onto two or three dimensions for purposes of easy visualization.

- **Pre-processing:** Learning regression and classification models from high-dimensional data is often difficult, in part because the number of model parameters often grows with the dimension of the feature space. This will be true with other techniques we discuss later in the notes (e.g., on clustering). As a consequence, learning is often very slow slow and prone to over-fitting. This is sometimes called the *curse of dimensionality*. PCA can be used to first map the data to a low-dimensional feature space before applying a more sophisticated algorithm to it. With PCA one can also *whiten* the representation, which rebalances the weights of the data to give better performance in some cases (e.g., with artificial neural networks)..
- **Prior Modeling:** Bayesian techniques often require generative models to serve as prior models of some process or class of observations (e.g., images of faces). Such models were used above in generative approaches to classification. A probabilistic form of PCA can be used to learn a low-dimensional Gaussian model for data.
- **Compression:** One of the early uses of PCA was data compression. If one can find a low-dimensional representation of a high-dimensional image, for example, then one can use such a representation to store and transmit data more efficiently.

For all these tasks, we want to find a low-dimensional representation of high-dimensional data that captures as much of the structure of the data as possible, but with relatively few parameters.

14.1 Modelling and Learning

In PCA we assume we are given N, d-dimensional data vectors $\{\mathbf{y}_i\}_{i=1}^N$, where $\mathbf{y}_i \in \mathbb{R}^d$. Our goal is to replace these vectors with lower-dimensional vectors $\{\mathbf{x}_i\}$, with dimensionality k, where $k \ll d$. In particular, let's further assume a simple mapping of the form

$$\mathbf{y} \;=\; \sum_{j=1}^{k} \mathbf{w}_j x_j + \mathbf{b} \;=\; \mathbf{W}\mathbf{x} + \mathbf{b}\,. \tag{14.1}$$

Here, the matrix $\mathbf{W}$ comprises the k basis vectors, $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_k]$. If we also assume Gaussian noise in the measurements, then this model is looks just like the linear regression model studied above, but note that in this case, both the matrix $\mathbf{W}$ and the $\mathbf{x}$'s are unknown, while in regression we were given training data comprising both $\mathbf{x}$ and $\mathbf{y}$. Since they are not known a priori, the $\mathbf{x}$ coordinates here are often called **latent** coordinates.

One way to learn the model is to solve the following constrained least-squares problem:

$$\arg\min_{\mathbf{W},\mathbf{b},\{\mathbf{x}_i\}} \quad \sum_i ||\,\mathbf{y}_i - (\mathbf{W}\mathbf{x}_i + \mathbf{b})\,||^2 \tag{14.2}$$

$$\text{subject to } \mathbf{W}^T\mathbf{W} = \mathbf{I}_k\,, \tag{14.3}$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix. The constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}_k$ requires that we obtain an orthonormal mapping $\mathbf{W}$;[1] i.e.,

$$\mathbf{w}_i^T \mathbf{w}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} . \tag{14.4}$$

This constraint is often used in the formation to resolve an ambiguity in the mapping. If we did not require $\mathbf{W}$ to be orthonormal, then the objective function is underconstrained (why?). Note that an ambiguity remains in the learning even with this constraint (which one?), but this ambiguity is not very important.

Stated here without proper derivation, the algorithm for minimizing the objective, to obtain the mapping, $\mathbf{W}$ and $\mathbf{b}$, and the latent coordinates of the training points, $\{\mathbf{x}_i\}_{i=1}^N$, comprises six main steps:

1. Let $\mathbf{b} = \frac{1}{N}\sum_i \mathbf{y}_i$
2. Compute the data covariance matrix: $\mathbf{C} = \frac{1}{N}\sum_i (\mathbf{y}_i - \mathbf{b})(\mathbf{y}_i - \mathbf{b})^T$
3. Let $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{C}$ be the eigenvector decomposition of $\mathbf{C}$.
   Here, $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues ($\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, ...\lambda_d)$), and $\mathbf{V} = [\mathbf{V}_1, ...\mathbf{V}_d]$ contains the orthonormal eigenvectors, $\mathbf{V}^T\mathbf{V} = \mathbf{I}_d$.
4. Assume the eigenvalues are sorted from largest to smallest ($\lambda_i \geq \lambda_{i+1}$). If this is not the case, then sort them along with their corresponding eigenvectors.
5. Let $\mathbf{W}$ be a matrix containing the first k eigenvectors: $\mathbf{W} = [\mathbf{V}_1, ...\mathbf{V}_k]$.
6. Let $\mathbf{x}_i = \mathbf{W}^T(\mathbf{y}_i - \mathbf{b})$, for all i.

14.2 Representation and Reconstruction of New Data

Suppose we have learned a PCA model, and are given a new $\mathbf{y}_{new}$ value. How do we estimate its corresponding $\mathbf{x}_{new}$? This can be done by minimizing

$$||\, \mathbf{y}_{new} - (\mathbf{W}\mathbf{x}_{new} + \mathbf{b})\, ||^2 \tag{14.5}$$

This is a linear least-squares problem, and can be solved with standard methods (in MATLAB, implemented by the backslash operator). However $\mathbf{W}$ is orthonormal, with a simple derivation you can show that the matrix pseudo-inverse is simply the transpose. As such, the solution simplifies to

$$\mathbf{x}_{new}^* = \mathbf{W}^T(\mathbf{y}_{new} - \mathbf{b}) . \tag{14.6}$$

[1]It turns out that this orthogonality constraint on $\mathbf{W}$ is not strictly necessary to derive PCA mathematically. By jointly maximizing the variance in a projection onto k dimensions, one can show that orthogonality emerges naturally in the solution. But, as is done in many textbooks, it is often easier to derive if one assumes orthogonality in the formulation.

Not surprisingly you will recognize this as the mapping used in Step 6 above to find the latent coordinates for the training data.

Given the latent repreesentation, $\mathbf{x}^*_{new}$, for the new point $\mathbf{y}_{new}$, we can use the forward model in Equation (14.1) to 'reconstruct' the approximation to $\mathbf{y}_{new}$ under the model. This approximation is $\mathbf{y}^*_{new} = \mathbf{W}\mathbf{x}^*_{new} + \mathbf{b}$, and the error in the reconstructed approximation is simply $||\,\mathbf{y}_{new} - \mathbf{y}^*_{new}||^2$.

14.3 Properties of PCA

Mean zero coefficients. One can show that the PCA coefficients (latent coordinates) that represent the training data, i.e., $\{\mathbf{x}_i\}_{i=1}^N$, are mean zero.

$$\begin{aligned} \text{Mean}(\mathbf{x}) \equiv \frac{1}{N}\sum_i \mathbf{x}_i &= \frac{1}{N}\sum_i \mathbf{W}^T(\mathbf{y}_i - \mathbf{b}) \\ &= \frac{1}{N}\mathbf{W}^T\left(\sum_i \mathbf{y}_i - N\mathbf{b}\right) \\ &= 0 \end{aligned} \tag{14.7}$$

since, in Step 1 above we set $\mathbf{b} = \frac{1}{N}\sum_i \mathbf{y}_i$.

Variance maximization. PCA can also be defined in the following way; in fact, this is the original definition of PCA, and the one that is often meant when people discuss PCA. However, this formulation is exactly equivalent to the one discussed above. In this goal, we wish to find the first principal component $\mathbf{w}_1$ to maximize the variance of the first coordinate of the data:

$$\text{Var}(x_1) = \frac{1}{N}\sum_i x_{1,i}^2 = \frac{1}{N}\sum_i (\mathbf{w}_1^T(\mathbf{y}_i - \mathbf{b}))^2 \tag{14.8}$$

such that $||\mathbf{w}_1||^2 = 1$. Then, we wish to choose the second principal component to be a unit vector and orthogonal to the first component, while maximizing the variance of $\mathbf{x}_2$. The remaining principle components are also defined in this recursive way, so that each component $\mathbf{w}_i$ is a unit vector, orthogonal to all previous basis vectors.

Uncorrelated coefficients. It is straightforward to show that the covariance matrix of the PCA latent coordinates is the just the upper left $k \times k$ submatrix of $\mathbf{\Lambda}$ (i.e., the diagonal matrix containing the k leading eigenvalues of $\mathbf{C}$.

$$\begin{aligned}
\mathrm{Cov}(\mathbf{x}) &\equiv \frac{1}{N}\sum_i \mathbf{x}_i\mathbf{x}_i^T \\
&= \frac{1}{N}\sum_i (\mathbf{W}^T(\mathbf{y}_i-\mathbf{b}))(\mathbf{W}^T(\mathbf{y}_i-\mathbf{b}))^T \\
&= \frac{1}{N}\mathbf{W}^T\left(\sum_i(\mathbf{y}_i-\mathbf{b})(\mathbf{y}_i-\mathbf{b})^T\right)\mathbf{W} \\
&= \mathbf{W}^T\mathbf{C}\mathbf{W} \\
&= \mathbf{W}^T\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T\mathbf{W} \\
&= \tilde{\mathbf{\Lambda}}
\end{aligned} \tag{14.9}$$

where $\tilde{\mathbf{\Lambda}}$ is the diagonal matrix containing the k leading eigenvalues in $\mathbf{\Lambda}$. This simple derivation also shows that the marginal variances of the PCA coefficients are given by the eigenvalues; i.e., $\mathrm{Var}(x_j) = \lambda_j$.

Out of subspace error. The total variance in the data is given by the sum of the eigenvalues of the sample covariance matrix $\mathbf{C}$. The variance captured by the PCA latent representation is the sum of the first k eigenvalues. The total amount of variance *lost* in the representation is given by the sum of the remaining eigenvalues. In fact, one can show that the least-squares error in the approximation to the original data provided by the optimal (ML) model parameters, $\mathbf{W}^*$, $\{\mathbf{x}_i^*\}$, and $\mathbf{b}^*$, is given by

$$\sum_i ||\,\mathbf{y}_i - (\mathbf{W}^*\mathbf{x}_i^* + \mathbf{b}^*)\,||^2 \;=\; \sum_{j=k+1}^{d} \lambda_j \;. \tag{14.10}$$

When learning a PCA model it is common to use the ratio of the total LS error and the total variance in the training data (i.e., the sum of all eigenvalues). One needs to choose k to be large enough that this ratio is small (often 0.1 or less).

Choosing the Subspace Dimension. Choosing the dimension of the subspace, i.e., the number of principal directions, k, for the low-dimensional approximation to the data is a challenging problem for which we do not have straightforward solutions. In essence, by choosing the dimension of the subspace we are effectively tryingn to identify the subspace that contains the signal of interest, assuming that the structure in the complementary subspace is predominantly noise.

There are two plots that are valuable as they sometimes reveal interesting structure in the data. The first is to plot the eigenvalues of the data covariance matrix, from largest to smallest along the x-axes. For some datasets the eigen-values decrease reasonably quickly and then flattern out to form a long tail. One rule of thumb is that as long the long flat tail is likely noise.

Another plot that is commonly used is to plot the fraction of variance captured within the subspace as a function of the subspace dimension.

$$h(k) \;=\; \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{d} \lambda_j}\,. \tag{14.11}$$

As the subspace dimension grows one captures more variance in the data. This plot visualizes clearly how much of the data variance is captured in the subspace. Many practitioners will choose the subspace dimension k such that some fraction of the data variance (e.g., 90% or 95%), is captured in the subspace.

14.4 Whitening

Whitening is a preprocess that replaces the data with a representation that has zero-mean and unit covariance, and is often useful as a data preprocessing step. Given measurements $\{\mathbf{y}_i\}$, we replace them with $\{\mathbf{z}_i\}$ given by

$$\mathbf{z}_i \;=\; \tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\mathbf{W}^T(\mathbf{y}_i - \mathbf{b}) \;=\; \tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\mathbf{x}_i \tag{14.12}$$

where $\tilde{\mathbf{\Lambda}}$ is a diagonal matrix containing the first k eigenvalues along the diagonal.

Then, one can show that sample mean of the $\mathbf{z}$'s is equal to 0:

$$\text{Mean}(\mathbf{z}) \;=\; \text{Mean}(\tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\mathbf{x}_i) \;=\; \tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\text{Mean}(\mathbf{x}) = 0 \tag{14.13}$$

To derive the sample covariance, we first compute the covariance of the untruncated values, i.e., $\tilde{\mathbf{z}} \equiv \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T(\mathbf{y} - \mathbf{b})$:

$$\begin{aligned}
\text{Cov}(\tilde{\mathbf{z}}) \;&\equiv\; \frac{1}{N}\sum_i \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T(\mathbf{y}_i - \mathbf{b})(\mathbf{y}_i - \mathbf{b})^T\mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} \\
&=\; \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T\left(\frac{1}{N}\sum_i (\mathbf{y}_i - \mathbf{b})(\mathbf{y}_i - \mathbf{b})^T\right)\mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} \\
&=\; \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T\mathbf{K}\mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} \\
&=\; \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T\mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} \\
&=\; \mathbf{I}_d
\end{aligned} \tag{14.14}$$

Since $\mathbf{z}$ is just the first k elements of $\tilde{\mathbf{z}}$, $\mathbf{z}$ has sample covariance $\mathbf{I}_k$.

14.5 Modeling

PCA is sometimes used to model the distribution of a collection of training data, e.g., we can use it as a form of a "prior". For example, suppose we have noisy measurements of some $\mathbf{y}$ values and wish to estimate their true values. If we parameterize the unknown $\mathbf{y}$ values by their corresponding $\mathbf{x}$ values instead, then we constrain the estimated values to lie in the low-dimensional subspace of the original data. However, this approach implies a uniform prior over $\mathbf{x}$ values, which may be inadequate, while being intolerant to deviations from the subspace. A better approach with an inherently probabilistic model is described below.

14.6 Probabilistic PCA

Probabilistic PCA is a way to estimate a Gaussian probability distribution over the observation space, $p(\mathbf{y})$. To that wend, we assume the following probability distribution:

$$\begin{aligned}
\mathbf{x} &\sim \mathcal{N}(0, \mathbf{I}) && (14.15)\\
\mathbf{y} &= \mathbf{W}\mathbf{x} + \mathbf{b} + \eta, \quad \eta \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) && (14.16)
\end{aligned}$$

where $\mathbf{x}$ and $\mathbf{n}$ are assumed to be statistically independent. The model says that the low-dimensional coordinates $\mathbf{x}$ (i.e., the underlying causes) come from a unit Gaussian distribution, and the $\mathbf{y}$ measurements are a linear function of these low-dimensioanl causes, plus Gaussian noise. Note that we do **not** require that $\mathbf{W}$ be orthonormal anymore (in part because we now constrain the magnitude of the $\mathbf{x}$ variables).

Since $\mathbf{x}$ and η are independent and Gaussian, and any linear transformation of a Gaussian variable is itself Gaussian, $\mathbf{y}$ must also be Gaussian. This distribution is:

$$\begin{aligned}
p(\mathbf{y}) &= \int p(\mathbf{x}, \mathbf{y})\, d\mathbf{x}\\
&= \int p(\mathbf{y}|\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}\\
&= \int G(\mathbf{y}; \mathbf{W}\mathbf{x} + \mathbf{b},\, \sigma^2 \mathbf{I})\, G(\mathbf{x}; 0,\, \mathbf{I})\, d\mathbf{x} && (14.17)
\end{aligned}$$

Evaluating this integral will give us $p(\mathbf{y})$, however, there is a simpler way to solve for the parameters of the Gaussian distribution.

Since we know $\mathbf{y}$ is Gaussian, all we need to do is derive its mean and covariance, which can be done as follows (using the fact that mathematical expectation is linear):

$$\begin{aligned}
\text{Mean}(\mathbf{y}) &\equiv E[\mathbf{y}] = E[\mathbf{W}\mathbf{x} + \mathbf{b} + \mathbf{n}]\\
&= \mathbf{W}E[\mathbf{x}] + \mathbf{b} + E[\mathbf{n}]\\
&= \mathbf{b} && (14.18)\\
\text{Cov}(\mathbf{y}) &\equiv E[(\mathbf{y} - \mathbf{b})(\mathbf{y} - \mathbf{b})^T]\\
&= E[(\mathbf{W}\mathbf{x} + \mathbf{b} + \mathbf{n} - \mathbf{b})(\mathbf{W}\mathbf{x} + \mathbf{b} + \mathbf{n} - \mathbf{b})^T]\\
&= E[(\mathbf{W}\mathbf{x} + \mathbf{n})(\mathbf{W}\mathbf{x} + \mathbf{n})^T]\\
&= E[\mathbf{W}\mathbf{x}\mathbf{x}^T\mathbf{W}^T] + E[\mathbf{W}\mathbf{x}\mathbf{n}^T] + E[\mathbf{n}\mathbf{x}^T\mathbf{W}^T] + E[\mathbf{n}\mathbf{n}^T]\\
&= \mathbf{W}E[\mathbf{x}\mathbf{x}^T]\mathbf{W}^T + \mathbf{W}E[\mathbf{x}]E[\mathbf{n}^T] + E[\mathbf{n}]E[\mathbf{x}^T]\mathbf{W}^T + \sigma^2\mathbf{I}\\
&= \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}\,. && (14.19)
\end{aligned}$$

As a consequence, we can now specify the distribution over $\mathbf{y}$ as

$$\mathbf{y} \sim \mathcal{N}(\mathbf{b}, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}) \qquad (14.20)$$

Learning a PPCA model is equivalent to learning a low-dimensional Gaussian distribution over the training data. This is illustrated in Figure 14.2. The PPCA model is not as general as learning a

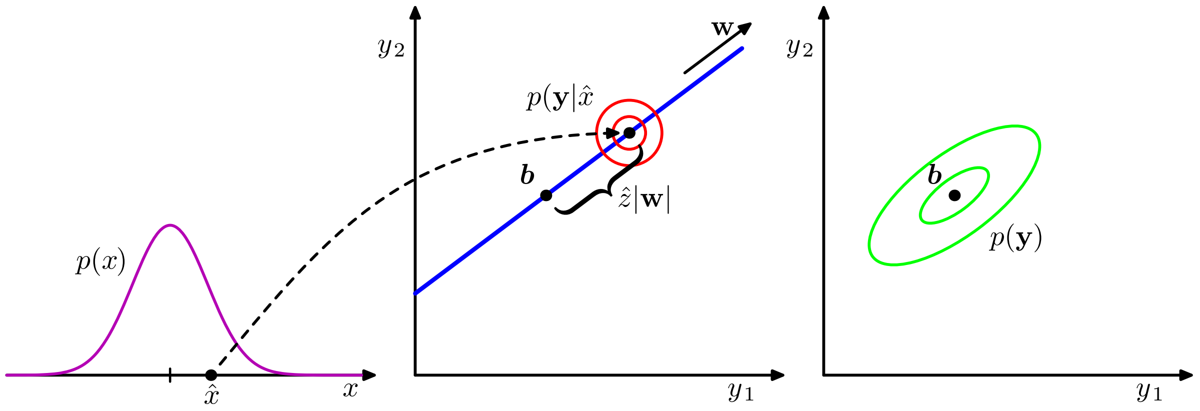

Figure 14.2: This is a visualization of the PPCA generative model, mapping from a 1D latent representation to a 2D observation space. In particular, a Gaussian in 1D is mapped to a line in 2D, and then blurred with 2D noise. (Figure from *Pattern Recognition and Machine Learning* by Chris Bishop.)

full Gaussian model with a $d \times d$ covariance matrix. However, it uses fewer numbers to represent the Gaussian ($k\,d + 1$ versus $d^2/2 + d/2$; why?). Because the representation is more compact, it can be estimated from smaller datasets, and requires less memory to store the model.

These differences will be significant when d is large; e.g., if $d = 100$, the full covariance matrix would requires 5050 parameters. One therefore needs hundreds of thousands of data points to estimate the covariance reliably. However, if the effective dimensionality is, say, 2 or 3, then the PPCA representation will only have a few hundred parameters, and, accordingly, many fewer measurements.

Learning. The PPCA model can be learned by Maximum Likelihood, i.e., by minimizing the negative log likliehood of the data:

$$\begin{aligned} L(\mathbf{W}, \mathbf{b}, \sigma^2) &= -\ln \prod_{i=1}^{N} G(\mathbf{y}_i;\, \mathbf{b},\, \mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I}) \\ &= \frac{1}{2} \sum_i (\mathbf{y}_i - \mathbf{b})^T (\mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I})^{-1} (\mathbf{y}_i - \mathbf{b}) + \frac{N}{2} \ln (2\pi)^d |\mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I}| \quad (14.21) \end{aligned}$$

Setting the gradient to zero, one can show that this can be optimized in closed form. Not surprisingly, the resulting algorithm is very similar to the conventional PCA case above:

1. Let $\mathbf{b} = \frac{1}{N}\sum_i \mathbf{y}_i$
2. Let $\mathbf{C} = \frac{1}{N}\sum_i (\mathbf{y}_i - \mathbf{b})(\mathbf{y}_i - \mathbf{b})^T$
3. Let $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{C}$ be the eigenvector decomposition of $\mathbf{C}$. $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues ($\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, ...\lambda_d)$). The matrix $\mathbf{V}$ contains the eigenvectors: $\mathbf{V} = [\mathbf{V}_1, ...\mathbf{V}_d]$ and is orthonormal $\mathbf{V}^T\mathbf{V} = \mathbf{I}_d$.
4. Assume that the eigenvalues are sorted from largest to smallest ($\lambda_i \geq \lambda_{i+1}$). If this is not the case, sort them (and their corresponding eigenvectors).
5. Let $\sigma^2 = \frac{1}{d-k}\sum_{j=k+1}^{d} \lambda_j$. In words, the estimated noise variance is equal to the average marginal data variance over all directions that are orthogonal to the k principal directions (i.e., this is the average variance (per dimension) of the data that is lost in the approximation of the data in the k dimensional subspace).
6. Let $\tilde{\mathbf{V}}$ be the matrix comprising the first k eigenvectors: $\tilde{\mathbf{V}} = [\mathbf{V}_1, ...\mathbf{V}_k]$, and let $\tilde{\mathbf{\Lambda}}$ be the diagonal matrix with the k leading eigenvalues: $\tilde{\mathbf{\Lambda}} = [\lambda_1, ...\lambda_k]$.
7. $\mathbf{W} = \tilde{\mathbf{V}}(\tilde{\mathbf{\Lambda}} - \sigma^2\mathbf{I})^{\frac{1}{2}}$.
8. Let $\mathbf{x}_i = \mathbf{W}^T(\mathbf{y}_i - \mathbf{b})$, for all i.

Note that this solution is similar to that in the conventional PCA case with whitening, except that (a) the noise variance is estimated, and (b) the noise is removed from the variances of the remaining eigenvalues.

An alternative optimization. In the above learning algorithm, we "marginalized out" $\mathbf{x}$ when estimating PPCA. In other words, we maximized

$$\begin{aligned} p(\mathbf{y}_{1:N} \,|\, \mathbf{W}, \mathbf{b}, \sigma^2) &= \int p(\mathbf{y}_{1:N}, \mathbf{x}_{1:N} \,|\, \mathbf{W}, \mathbf{b}, \sigma^2)\, d\mathbf{x}_{1:N} \\ &= \int p(\mathbf{y}_{1:N} \,|\, \mathbf{x}_{1:N}, \mathbf{W}, \mathbf{b}, \sigma^2)\, p(\mathbf{x}_{1:N})\, d\mathbf{x}_{1:N} \\ &= \prod_i \int p(\mathbf{y}_i \,|\, \mathbf{x}_i, \mathbf{W}, \mathbf{b}, \sigma^2)\, p(\mathbf{x}_i)\, d\mathbf{x}_i \,, \end{aligned} \tag{14.22}$$

instead of maximizing

$$\begin{aligned} p(\mathbf{y}_{1:N}, \mathbf{x}_{1:N} \,|\, \mathbf{W}, \mathbf{b}, \sigma^2) &= \prod_i p(\mathbf{y}_i, \mathbf{x}_i \,|\, \mathbf{W}, \mathbf{b}, \sigma^2) \\ &= \prod_i p(\mathbf{y}_i \,|\, \mathbf{x}_i, \mathbf{W}, \mathbf{b}, \sigma^2)\, p(\mathbf{x}_i) \end{aligned} \tag{14.23}$$

By integrating out $\mathbf{x}$, we are estimating fewer parameters and thus can get better estimates. Loosely speaking, doing so might be viewed as being "more Bayesian." Suppose we did instead try to estimate the $\mathbf{x}$'s together with the model parameters:

$$
\begin{aligned}
L(\mathbf{x}_{1:N}, \mathbf{W}, \mathbf{b}, \sigma^2) &= -\ln p(\mathbf{y}_{1:N}, \mathbf{x}_{1:N} \,|\, \mathbf{W}, \mathbf{b}, \sigma^2) && (14.24)\\
&= \sum_i \left(\frac{1}{2\sigma^2} ||\mathbf{y}_i - (\mathbf{W}\mathbf{x}_i + \mathbf{b})||^2 + \frac{1}{2}||\mathbf{x}_i||^2 \right) \\
&\quad + \frac{N\,d}{2} \ln \sigma^2 + N\,d \ln 2\pi\ . && (14.25)
\end{aligned}
$$

Now, suppose we are optimizing this objective function, and we have some estimates for $\mathbf{W}$ and $\mathbf{x}$. We can always reduce the objective function by replacing

$$
\begin{aligned}
\mathbf{W} &\leftarrow 2\mathbf{W} && (14.26)\\
\mathbf{x} &\leftarrow \mathbf{x}/2 && (14.27)
\end{aligned}
$$

By doing this replacement arbitrarily many times, we can get infinitesimal values for $\mathbf{x}$. This indicates that the objective function is degenerate. Using it will produce poor results.

Note that, however, this arises using *the same model* as before, but without marginalizing out $\mathbf{x}$. This illustrates a general principle: The more parameters you estimate (instead of marginalizing out), the greater the danger of biased and/or degenerate solutions.