13 Principal Components Analysis

We now discuss an **unsupervised** learning algorithm, called Principal Components Analysis, or PCA. The method is unsupervised because we are learning a mapping without any examples of what the mapping looks like; all we see are the outputs, and we want to estimate both the mapping *and* the inputs.

PCA is primarily a tool for dealing with high-dimensional data. If our measurements are 17-dimensional, or 30-dimensional, or 10,000-dimensional, manipulating the data can be extremely difficult. Quite often, the actual data can be described by a much lower-dimensional representation that captures all of the structure of the data. PCA is perhaps the simplest approach for finding such a representation, and yet is it also very fast and effective, resulting in it being very widely used.

There are several ways in which PCA can help:

- **Visualization:** PCA provides a way to visualize the data, by projecting the data down to two or three dimensions that you can plot, in order to get a better sense of the data. Furthermore, the principal component vectors sometimes provide insight as to the nature of the data as well.
- **Preprocessing:** Learning complex models of high-dimensional data is often very slow, and also prone to overfitting — the number of parameters in a model is usually exponential in the number of dimensions, meaning that very large data sets are required for higher-dimensional models. This problem is generally called **the curse of dimensionality.** PCA can be used to first map the data to a low-dimensional representation 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.
- **Modeling:** PCA learns a representation that is sometimes used as an entire model, e.g., a prior distribution for new data.
- **Compression:** PCA can be used to compress data, by replacing data with its low-dimensional representation.

13.1 The model and learning

In PCA, we assume we are given N data vectors $\{\mathbf{y}_i\}$, where each vector is D-dimensional: $\mathbf{y}_i \in \mathbb{R}^D$. Our goal is to replace these vectors with lower-dimensional vectors $\{\mathbf{x}_i\}$ with dimensionality C, where $C < D$. We assume that they are related by a linear transformation:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} = \sum_{j=1}^{C} \mathbf{w}_j x_j + \mathbf{b} \tag{1}$$

The matrix $\mathbf{W}$ can be viewed as a containing a set of C basis vectors $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_C]$. If we also assume Gaussian noise in the measurements, this model is the same as the linear regression model studied earlier, but now the $\mathbf{x}$'s are unknown in addition to the linear parameters.

To learn the model, we solve the following constrained least-squares problem:

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

$$\text{subject to } \mathbf{W}^T\mathbf{W} = \mathbf{I} \tag{3}$$

The constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ requires that we obtain an orthonormal mapping $\mathbf{W}$; it is equivalent to saying that

$$\mathbf{w}_i^T\mathbf{w}_j \;=\; \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{4}$$

This constraint is required 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.

The $\mathbf{x}$ coordinates are often called **latent** coordinates.

The algorithm for minimizing this objective function is as follows:

1. Let $\mathbf{b} = \frac{1}{N}\sum_i \mathbf{y}_i$
2. Let $\mathbf{K} = \frac{1}{N}\sum_i (\mathbf{y}_i - \mathbf{b})(\mathbf{y}_i - \mathbf{b})^T$
3. Let $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{K}$ be the eigenvector decomposition of $\mathbf{K}$. $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues ($\mathbf{\Lambda} = \text{diag}(\lambda_1, ...\lambda_D)$). The matrix $\mathbf{V}$ contains the eigenvectors: $\mathbf{V} = [\mathbf{V}_1, ...\mathbf{V}_D]$ and is orthonormal $\mathbf{V}^T\mathbf{V} = \mathbf{I}$.
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 $\mathbf{W}$ be a matrix of the first C eigenvectors: $\mathbf{W} = [\mathbf{V}_1, ...\mathbf{V}_C]$.
6. Let $\mathbf{x}_i = \mathbf{W}^T(\mathbf{y}_i - \mathbf{b})$, for all i.

13.2 Reconstruction

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

$$||\mathbf{y}_{new} - (\mathbf{W}\mathbf{x}_{new} + \mathbf{b})||^2 \tag{5}$$

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, and thus its transpose is the pseudoinverse, so the solution is given simply by:

$$\mathbf{x}^*_{new} = \mathbf{W}^T(\mathbf{y}_{new} - \mathbf{b}) \tag{6}$$

13.3 Properties of PCA

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

$$\begin{aligned} \text{mean}(\mathbf{x}) \equiv \frac{1}{N}\sum_i \mathbf{x}_i &= \frac{1}{N}\sum_i \mathbf{W}^T(\mathbf{y}_i - \mathbf{b}) && (7)\\ &= \frac{1}{N}\mathbf{W}^T\left(\sum_i \mathbf{y}_i - N\mathbf{b}\right) && (8)\\ &= 0 && (9) \end{aligned}$$

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:

$$\text{var}(x_1) = \frac{1}{N}\sum_i x_{1,i}^2 = \frac{1}{N}\sum_i (\mathbf{w}_1^T(\mathbf{y}_i - \mathbf{b}))^2 \tag{10}$$

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 coefficients is the just the upper left $C \times C$ submatrix of $\mathbf{\Lambda}$ (i.e., the diagonal matrix containing the C leading eigenvalues of $\mathbf{K}$.

$$\begin{aligned} \text{cov}(\mathbf{x}) &\equiv \frac{1}{N}\sum_i (\mathbf{W}^T(\mathbf{y}_i - \mathbf{b}))(\mathbf{W}^T(\mathbf{y}_i - \mathbf{b}))^T && (11)\\ &= \frac{1}{N}\mathbf{W}^T\left(\sum_i (\mathbf{y}_i - \mathbf{b})(\mathbf{y}_i - \mathbf{b})^T\right)\mathbf{W} && (12)\\ &= \mathbf{W}^T\mathbf{K}\mathbf{W} && (13)\\ &= \mathbf{W}^T\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T\mathbf{W} && (14)\\ &= \tilde{\mathbf{\Lambda}} && (15) \end{aligned}$$

where $\tilde{\mathbf{\Lambda}}$ is the diagonal matrix containing the C 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., $\text{var}(x_j) = \lambda_j$.

Out of Subspace Error. The total variance in the data is given by the sum of the eigenvalues of the sample covariance matrix $\mathbf{K}$. The variance captured by the PCA subspace representation is the sum of the first C 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

$$\sum_i ||\mathbf{y}_i - (\mathbf{W}^*\mathbf{x}_i^* + \mathbf{b}^*)||^2 = \sum_{j=C+1}^{D} \lambda_j \,. \tag{16}$$

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 C to be large enough that this ratio is small (often 0.1 or less).

13.4 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

$$\mathbf{z}_i = \tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\mathbf{W}^T(\mathbf{y}_i - \mathbf{b}) = \tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\mathbf{x}_i \tag{17}$$

where $\tilde{\mathbf{\Lambda}}$ is a diagonal matrix of the first C eigenvalues.

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

$$\text{mean}(\mathbf{z}) = \text{mean}(\tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\mathbf{x}_i) = \tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\text{mean}(\mathbf{x}) = 0 \tag{18}$$

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

$$\begin{aligned}
\text{cov}(\tilde{\mathbf{z}}) &\equiv \frac{1}{N}\sum_i \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T(\mathbf{y}_i - \mathbf{b})(\mathbf{y}_i - \mathbf{b})^T\mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} && (19)\\
&= \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T\left(\frac{1}{N}\sum_i (\mathbf{y}_i - \mathbf{b})(\mathbf{y}_i - \mathbf{b})^T\right)\mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} && (20)\\
&= \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T\mathbf{K}\mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} && (21)\\
&= \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T\mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} && (22)\\
&= \mathbf{I} && (23)
\end{aligned}$$

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

13.5 Modeling

PCA is sometimes used to model data likelihood, 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.

13.6 Probabilistic PCA

Probabilistic PCA is a way to estimate a probability distribution $p(\mathbf{y})$; in fact, it is a form of Gaussian distribution. In particular, we assume the following probability distribution:

$$\mathbf{x} \sim \mathcal{N}(0, \mathbf{I}) \tag{24}$$

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} + \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(0, \sigma^2\mathbf{I}) \tag{25}$$

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 any linear transformation of a Gaussian variable is itself Gaussian, $\mathbf{y}$ must also be Gaussian. This distribution is:

$$p(\mathbf{y}) = \int p(\mathbf{x}, \mathbf{y}) d\mathbf{x} = \int p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \int G(\mathbf{y}; \mathbf{W}\mathbf{x} + \mathbf{b},\ \sigma^2\mathbf{I})\, G(\mathbf{x}; 0,\ \mathbf{I})\, d\mathbf{x} \tag{26}$$

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

Since we know that $\mathbf{y}$ 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):

$$\begin{aligned}
\text{mean}(\mathbf{y}) &= E[\mathbf{y}] = E[\mathbf{W}\mathbf{x} + \mathbf{b} + \mathbf{n}] && (27)\\
&= \mathbf{W}E[\mathbf{x}] + \mathbf{b} + E[\mathbf{n}] && (28)\\
&= \mathbf{b} && (29)\\
\text{cov}(\mathbf{y}) &= E[(\mathbf{y} - \mathbf{b})(\mathbf{y} - \mathbf{b})^T] && (30)\\
&= E[(\mathbf{W}\mathbf{x} + \mathbf{b} + \mathbf{n} - \mathbf{b})(\mathbf{W}\mathbf{x} + \mathbf{b} + \mathbf{n} - \mathbf{b})^T] && (31)\\
&= E[(\mathbf{W}\mathbf{x} + \mathbf{n})(\mathbf{W}\mathbf{x} + \mathbf{n})^T] && (32)\\
&= E[\mathbf{W}\mathbf{x}\mathbf{x}^T\mathbf{W}^T] + E[\mathbf{W}\mathbf{x}\mathbf{n}^T] + E[\mathbf{n}\mathbf{x}^T\mathbf{W}^T] + E[\mathbf{n}\mathbf{n}^T] && (33)\\
&= \mathbf{W}E[\mathbf{x}\mathbf{x}^T]\mathbf{W}^T + \mathbf{W}E[\mathbf{x}]E[\mathbf{n}^T] + E[\mathbf{n}]E[\mathbf{x}^T]\mathbf{W}^T + \sigma^2\mathbf{I} && (34)\\
&= \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I} && (35)
\end{aligned}$$

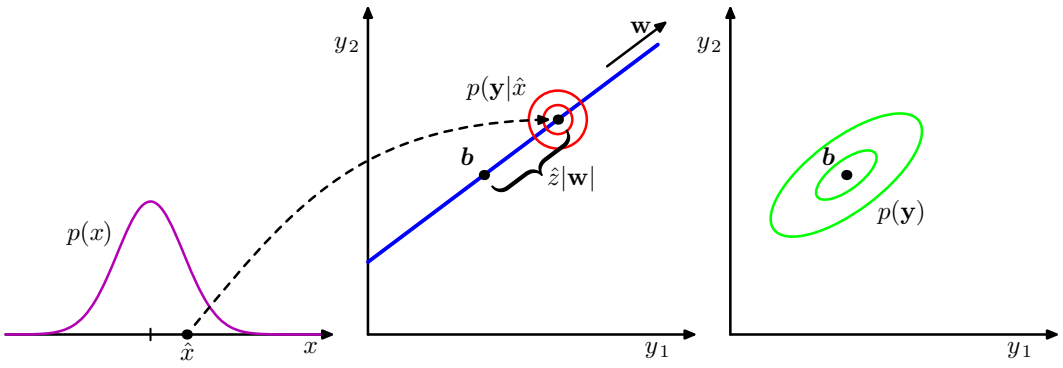

Figure 1: Visualization of PPCA mapping for a 1D to 2D model. A Gaussian in 1D is mapped to a line, and then blurred with 2D noise. (Figure from *Pattern Recognition and Machine Learning* by Chris Bishop.)

Hence

$$\mathbf{y} \sim \mathcal{N}(\mathbf{b}, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}) \tag{36}$$

In other words, learning a PPCA model is equivalent to learning a particular form of a Gaussian distribution. This is illustrated in Figure 1. The PPCA model is not as general as learning a full Gaussian model with a $D \times D$ covariance matrix; however, it uses fewer numbers to represent the Gaussian ($CD+1$ 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 require 5050 parameters and thus require hundreds of thousands of data points to estimate reliably. However, if the effective dimensionality is, say, 2 or 3, then the PPCA representation will only have a few hundred parameters and many fewer measurements.

Learning. The PPCA model can be learned by Maximum Likelihood, i.e., by minimizing:

$$L(\mathbf{W}, \mathbf{b}, \sigma^2) = -\ln \prod_{i=1}^{N} G(\mathbf{y}_i;\, \mathbf{b},\, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}) \tag{37}$$

$$= \frac{1}{2}\sum_i (\mathbf{y}_i - \mathbf{b})^T(\mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I})^{-1}(\mathbf{y}_i - \mathbf{b}) + \frac{N}{2}\ln(2\pi)^D|\mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}| \tag{38}$$

This can be optimized in closed form. The solution is very similar to the conventional PCA case:

1. Let $\mathbf{b} = \frac{1}{N}\sum_i \mathbf{y}_i$
2. Let $\mathbf{K} = \frac{1}{N}\sum_i (\mathbf{y}_i - \mathbf{b})(\mathbf{y}_i - \mathbf{b})^T$

3. Let $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{K}$ be the eigenvector decomposition of $\mathbf{K}$. $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues ($\mathbf{\Lambda} = \text{diag}(\lambda_1, ...\lambda_D)$). The matrix $\mathbf{V}$ contains the eigenvectors: $\mathbf{V} = [\mathbf{V}_1, ...\mathbf{V}_D]$ and is orthonormal $\mathbf{V}^T\mathbf{V} = \mathbf{I}$.

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 $\sigma^2 = \frac{1}{D-C}\sum_{j=C+1}^{D} \lambda_j$. In words, the estimated noise variance is equal to the average marginal data variance over all directions that are orthogonal to the C 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 C dimensional subspace).

6. Let $\tilde{\mathbf{V}}$ be the matrix comprising the first C eigenvectors: $\tilde{\mathbf{V}} = [\mathbf{V}_1, ...\mathbf{V}_C]$, and let $\tilde{\mathbf{\Lambda}}$ be the diagonal matrix with the C leading eigenvalues: $\tilde{\mathbf{\Lambda}} = [\lambda_1, ...\lambda_C]$.

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

$$p(\mathbf{y}_{1:N}|\mathbf{W}, \mathbf{b}, \sigma^2) = \int p(\mathbf{y}_{1:N}, \mathbf{x}_{1:N}|\mathbf{W}, \mathbf{b}, \sigma^2) d\mathbf{x}_{1:N} \quad (39)$$

$$= \int p(\mathbf{y}_{1:N}|\mathbf{x}_{1:N}, \mathbf{W}, \mathbf{b}, \sigma^2) p(\mathbf{x}_{1:N}) d\mathbf{x}_{1:N} \quad (40)$$

$$= \prod_i \int p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{W}, \mathbf{b}, \sigma^2) p(\mathbf{x}_i) d\mathbf{x}_i \quad (41)$$

instead of maximizing

$$p(\mathbf{y}_{1:N}, \mathbf{x}_{1:N}|\mathbf{W}, \mathbf{b}, \sigma^2) = \prod_i p(\mathbf{y}_i, \mathbf{x}_i|\mathbf{W}, \mathbf{b}, \sigma^2) \quad (42)$$

$$= \prod_i p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{W}, \mathbf{b}, \sigma^2) p(\mathbf{x}_i) \quad (43)$$

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:

$$\begin{aligned} L(\mathbf{x}_{1:N}, \mathbf{W}, \mathbf{b}, \sigma^2) &= -\ln p(\mathbf{y}_{1:N}, \mathbf{x}_{1:N} | \mathbf{W}, \mathbf{b}, \sigma^2) && (44) \\ &= \sum_i \left(\frac{1}{2\sigma^2} ||\mathbf{y}_i - (\mathbf{W}\mathbf{x}_i + \mathbf{b})||^2 + \frac{1}{2} ||\mathbf{x}_i||^2 \right) \\ &\quad + \frac{ND}{2} \ln \sigma^2 + ND \ln 2\pi && (45) \end{aligned}$$

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

$$\begin{aligned} \mathbf{W} &\leftarrow 2\mathbf{W} && (46) \\ \mathbf{x} &\leftarrow \mathbf{x}/2 && (47) \end{aligned}$$

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 yield to very 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.

