Tutorial on Principle Component Analysis and Data Whitening

Zhihan Yang
Carleton College
yangz2@carleton.edu

May 2020

Contents

TODO:

- Add algorithm 2 for Whitening with option of PCA (done)
- Add a table of content with hyperlinks for easy navigation (done)
- Write section 3.6: the basic idea, first, using the elbow, second, for real world data, choose the number of components under 95 percent variance is captured
- re-write the introduction
- Read through sub-sections one at a time and tick them out
- change principle component analysis to PCA

1 Introduction

This document explains how Principle Component Analysis (PCA) works.

2 Descriptors of data

Let's consider a data set X as follows:

$$X = [\boldsymbol{x}^1, \cdots, \boldsymbol{x}^N]$$

where $\boldsymbol{x}^n$ is the n-th example (or data point) and $\boldsymbol{x}^n \in \mathbb{R}^d$. The j entry of $\boldsymbol{x}^n$ is denoted by x_j^n.

2.1 Mean

We may be interested in what a *typical* data point look like for X. Therefore, we define a mean vector $\boldsymbol{\mu}$ as follows:

$$\boldsymbol{\mu} \triangleq \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^n,$$

and consequently, μ_j, the j-component of $\boldsymbol{\mu}$, is defined as $\frac{1}{N}\sum_{n=1}^{N} x_j^n$.

2.2 Covariance matrix

We may also be interested in the covariance between every pair of dimensions, i.e., the degree to which two variables are linearly associated. The covariance matrix, Σ, is essentially a lookup table where the (i,j)-th entry tells us the covariance between dimension i and j. It is defined as follows:

$$\Sigma \triangleq \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T.$$

To further appreciate this definition, we may expand the matrix-algebra notation above as follows:

$$\begin{aligned}
\Sigma &\triangleq \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T \\
&\triangleq \frac{1}{N-1}\sum_{n=1}^{N} \underbrace{\begin{bmatrix} x_1^n - \mu_1 \\ \vdots \\ x_d^n - \mu_d \end{bmatrix}\begin{bmatrix} x_1^n - \mu_1 & \cdots & x_d^n - \mu_d \end{bmatrix}}_{\text{Outer product}} \\
&\triangleq \frac{1}{N-1}\sum_{n=1}^{N} \begin{bmatrix} (x_1^n - \mu_1)^2 & \cdots & (x_1^n - \mu_1)(x_d^n - \mu_d) \\ \vdots & \ddots & \vdots \\ (x_d^n - \mu_d)(x_1^n - \mu_1) & \cdots & (x_d^n - \mu_d)^2 \end{bmatrix} \\
&\triangleq \begin{bmatrix} \frac{1}{N-1}\sum_{n=1}^{N}(x_1^n - \mu_1)^2 & \cdots & \frac{1}{N-1}\sum_{n=1}^{N}(x_1^n - \mu_1)(x_d^n - \mu_d) \\ \vdots & \ddots & \vdots \\ \frac{1}{N-1}\sum_{n=1}^{N}(x_d^n - \mu_d)(x_1^n - \mu_1) & \cdots & \frac{1}{N-1}\sum_{n=1}^{N}(x_d^n - \mu_d)^2 \end{bmatrix}
\end{aligned}$$

where the (i,j)-th entry indeed tells us the covariance between dimension i and j.

3 Principle component analysis (PCA)

3.1 Linear projection of data to one dimension

Suppose we want to project our data set X from d dimensions to one dimension only. Formally, this can be done by projecting each example along some vector $\boldsymbol{v}$ as follows:

$$\boldsymbol{v}^T X = [\boldsymbol{v}^T\boldsymbol{x}^1 \cdots \boldsymbol{v}^T\boldsymbol{x}^N].$$

Projecting data from a large number of dimensions to a lower number is called *dimensionality reduction*, which helps ameliorate the curse of dimensionality. A lower number of dimensions usually implies a lower number of tunable parameters in machine-learning models, which is more likely to be appropriately determined by a finite data set. However, this operation reduces the amount of information held in data. To minimize this consequence, we want to carefully tune the direction of $\boldsymbol{v}$ so that the variance of $\boldsymbol{v}^T X$ is maximized. We also want to constraint the norm of $\boldsymbol{v}$ to be a constant so that any increase in the variance of $\boldsymbol{v}^T X$ is not due to simple scalings.

3.2 Variance of projected data

First, let us derive an expression for the variance of $\boldsymbol{v}^T X$. Before we proceed, note that the mean for $\boldsymbol{v}^T X$ is simply $\boldsymbol{v}^T \boldsymbol{\mu}$; this step is left as a mental exercise for the reader. Then, the variance of $\boldsymbol{v}^T X$, which we shall denote by σ^2, can be computed as follows:

$$\begin{aligned}
\sigma^2 &= \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{v}^T\boldsymbol{x}^n - \boldsymbol{v}^T\boldsymbol{\mu}) \\
&= \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{v}^T(\boldsymbol{x}^n - \boldsymbol{\mu}))^2 \\
&= \frac{1}{N-1}\sum_{n=1}^{N}\underbrace{(\boldsymbol{v}^T(\boldsymbol{x}^n - \boldsymbol{\mu})(\boldsymbol{x}^n - \boldsymbol{\mu})^T\boldsymbol{v})}_{\boldsymbol{v}\text{ is not dependent on }n} \\
&= \boldsymbol{v}^T\underbrace{\left(\frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{x}^n - \boldsymbol{\mu})(\boldsymbol{x}^n - \boldsymbol{\mu})^T\right)}_{\text{Covariance matrix, }\Sigma}\boldsymbol{v} \\
&= \boldsymbol{v}^T\Sigma\boldsymbol{v}
\end{aligned}$$

3.3 Maximize variance of projected data

Then, we deal with the following constrained optimization problem:

$$\begin{aligned}
&\text{maximize } f(\boldsymbol{v}) = \boldsymbol{v}^T\Sigma\boldsymbol{v} \\
&\text{subject to } ||\boldsymbol{v}||^2 = \boldsymbol{v}^T\boldsymbol{v} = 1
\end{aligned}$$

which can be solved using the method of Lagrange multipliers, i.e., we define $f = \boldsymbol{v}^T\Sigma\boldsymbol{v}$ and $g = ||\boldsymbol{v}||$ and solve for the system of equations:

$$\begin{aligned}
\nabla f|_{\boldsymbol{v}} &= \lambda\nabla g|_{\boldsymbol{v}} \\
\boldsymbol{v}^T\boldsymbol{v} &= 1
\end{aligned}$$

We substitute $\nabla f|_{\boldsymbol{v}} = 2\Sigma\boldsymbol{v}$ and $\nabla g|_{\boldsymbol{v}} = 2\boldsymbol{v}$ into the system of equations above and obtain:

$$\Sigma\boldsymbol{v} = \lambda\boldsymbol{v} \tag{1}$$

$$\boldsymbol{v}^T\boldsymbol{v} = 1 \tag{2}$$

Subtracting the RHS from both sides of (1) and factor, obtain:

$$(\Sigma - \lambda I)\boldsymbol{v} = 0 \tag{3}$$

Since we already know that $\boldsymbol{v} \neq \vec{0}$ from (2), $\Sigma - \lambda I$ is not invertible because it transforms a non-zero vector to the zero vector. Therefore, its determinant is equal to zero, i.e., $\det(\Sigma - \lambda I) = 0$. Solving this equation yields values of λ, each of which can be substituted back into (1) to obtain the *form* of its corresponding $\boldsymbol{v}$. The constraint imposed by (2) can then be used to determine the values of the entries of the $\boldsymbol{v}$'s.

For those who are familiar with eigenvalues and eigenvectors, it is obvious from (2) that λ is an eigenvalue of Σ and $\boldsymbol{v}$ is an eigenvector of Σ. The *principle axis* is simply the eigenvector of Σ with the largest eigenvalue, which we denote by $\boldsymbol{v}_{\max}$. This is because $\boldsymbol{v}_{\max}^T\Sigma\boldsymbol{v}_{\max} = \lambda_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{v}_{\max} = \lambda_{\max} \geq \lambda$.

3.4 Interpretation of other eigenvectors of Σ as non-primary principal components

Let us consider the original data but with its component along $\boldsymbol{v}_{\max}$ removed:

$$\tilde{X} = X - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T X$$

where $\boldsymbol{v}_{\max}^T X$ denotes the *projection* (a scalar) of X onto $\boldsymbol{v}_{\max}$ and correspondingly $\boldsymbol{v}\boldsymbol{v}_{\max}^T X$ denotes the *component* (a vector) of X along $\boldsymbol{v}_{\max}$. Intuitively, $\tilde{X}$ represents the part of X whose variance was not captured by $\boldsymbol{v}_{\max}$. Therefore, it is interesting to consider, what would be the direction of projection that captures the most variance of $\tilde{X}$?

Formally, the projection of $\tilde{X}$ along some direction $\boldsymbol{v}$ can be written as:

$$\boldsymbol{v}^T\tilde{X} = \boldsymbol{v}^T(X - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T X)$$

Again, we constrain the norm of $\boldsymbol{v}$ to 1 for aforementioned reasons.

We are interested in maximizing the variance of the projection, $\boldsymbol{v}^T\tilde{X}$. The variance of $\boldsymbol{v}^T\tilde{X}$, which we shall denote by $\tilde{\sigma^2}$, can be computed as follows. Note the mean vector of $\boldsymbol{v}^T\tilde{X}$ is assumed to be zero; practically, this can be easily accomplished through a demean operation.

$$\begin{aligned}
\tilde{\sigma^2} &= \frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{v}^T(\boldsymbol{x}^n - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n)\}^2 \\
&= \frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{v}^T(\boldsymbol{x}^n - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n)(\boldsymbol{x}^n - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n)^T\boldsymbol{v}\} \\
&= \boldsymbol{v}^T\left(\frac{1}{N-1}\sum_{n=1}^{N}\{(\boldsymbol{x}^n - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n)\underbrace{(\boldsymbol{x}^n - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n)^T}_{\text{Distribute the transpose.}}\}\right)\boldsymbol{v} \\
&= \boldsymbol{v}^T\left(\frac{1}{N-1}\sum_{n=1}^{N}\underbrace{\{(\boldsymbol{x}^n - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n)(\boldsymbol{x}^{nT} - \boldsymbol{x}^{nT}\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T)\}}_{\text{Simplify by removing the brackets.}}\right)\boldsymbol{v} \\
&= \boldsymbol{v}^T\left(\underbrace{\frac{1}{N-1}\sum_{n=1}^{N}}_{\text{Distribute.}}\{\boldsymbol{x}^n\boldsymbol{x}^{nT} - \boldsymbol{x}^n\boldsymbol{x}^{nT}\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n\boldsymbol{x}^{nT} + \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n\boldsymbol{x}^{nT}\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\}\right)\boldsymbol{v} \\
&= \boldsymbol{v}^T\left(\frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{x}^n\boldsymbol{x}^{nT}\} - \frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{x}^n\boldsymbol{x}^{nT}\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\}\right. \\
&\quad \left. - \frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n\boldsymbol{x}^{nT}\} + \frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\boldsymbol{x}^n\boldsymbol{x}^{nT}\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\}\right)\boldsymbol{v} \\
&= \boldsymbol{v}^T\left(\underbrace{\frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{x}^n\boldsymbol{x}^{nT}\}}_{\Sigma} - \underbrace{\frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{x}^n\boldsymbol{x}^{nT}\}}_{\Sigma}\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\right. \\
&\quad \left. - \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\underbrace{\frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{x}^n\boldsymbol{x}^{nT}\}}_{\Sigma} + \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\underbrace{\frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{x}^n\boldsymbol{x}^{nT}\}}_{\Sigma}\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\right)\boldsymbol{v} \\
&= \boldsymbol{v}^T\left(\Sigma - \underbrace{\Sigma\boldsymbol{v}_{\max}}_{\lambda_{\max}}\boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T - \boldsymbol{v}_{\max}\underbrace{\boldsymbol{v}_{\max}^T\Sigma}_{(\Sigma\boldsymbol{v}_{\max})^T=\lambda_{\max}\boldsymbol{v}^T} + \boldsymbol{v}_{\max}\boldsymbol{v}_{\max}^T\underbrace{\Sigma\boldsymbol{v}_{\max}}_{\lambda_{\max}\boldsymbol{v}}\boldsymbol{v}_{\max}^T\right)\boldsymbol{v}
\end{aligned}$$

$$
\begin{aligned}
&= \boldsymbol{v}^T \left(\Sigma - \lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T - \lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T + \lambda_{\max} \boldsymbol{v}_{\max} \underbrace{\boldsymbol{v}_{\max}^T \boldsymbol{v}_{\max}}_{1} \boldsymbol{v}_{\max}^T \right) \boldsymbol{v} \\
&= \boldsymbol{v}^T \left(\Sigma - \lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T - \lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T + \lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T \right) \boldsymbol{v} \\
&= \boldsymbol{v}^T \left(\Sigma - \lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T \right) \boldsymbol{v} \\
&= \boldsymbol{v}^T \Sigma^{\text{new}} \boldsymbol{v}
\end{aligned}
$$

where we have defined $\Sigma^{\text{new}} = \Sigma - \lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T$.

Similar to how we obtained equation (1), we obtain:

$$
\begin{aligned}
\Sigma^{\text{new}} \boldsymbol{v} &= \lambda \boldsymbol{v} \\
(\Sigma - \lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T) \boldsymbol{v} &= \lambda \boldsymbol{v} \\
\Sigma \boldsymbol{v} - \lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T \boldsymbol{v} &= \lambda \boldsymbol{v}
\end{aligned}
$$

where λ denotes the Lagrange Multiplier. Note that all eigenvectors of Σ are solutions of this equation. To see this clearly, consider two cases: (1) $\boldsymbol{v} \neq \boldsymbol{v}_{\max}$ and (2) $\boldsymbol{v} = \boldsymbol{v}_{\max}$. First, suppose $\boldsymbol{v} \neq \boldsymbol{v}_{\max}$. Notice how $\lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T \boldsymbol{v}$ becomes zero when $\boldsymbol{v} \neq \boldsymbol{v}_{\max}$ since the eigenvectors of a symmetric matrix (in the case, the covariance matrix) are orthogonal; all that's left is $\Sigma \boldsymbol{v} = \lambda \boldsymbol{v}$. Therefore, the eigenvalues for all $\boldsymbol{v} \neq \boldsymbol{v}_{\max}$ is untouched. Second, suppose $\boldsymbol{v} = \boldsymbol{v}_{\max}$. Notice how $\lambda_{\max} \boldsymbol{v}_{\max} \boldsymbol{v}_{\max}^T \boldsymbol{v}$ becomes $\lambda_{\max} \boldsymbol{v}_{\max}$; the LHS of equation of n becomes $\Sigma \lambda_{\max} - \lambda_{\max} \boldsymbol{v}_{\max} = \lambda_{\max} \boldsymbol{v}_{\max} - \lambda_{\max} \boldsymbol{v}_{\max} = 0$. Therefore, the new eigenvalue of $\boldsymbol{v}_{\max}$ is now zero.

Clearly, the direction of projection that leads to max variance would be the vector $v_{\max'}$ with the largest eigenvalue of Σ^{new}, which is the second largest largest eigenvalue of Σ. This is because $\boldsymbol{v}_{\max'}^T \Sigma^{\text{new}} \boldsymbol{v}_{\max'} = \lambda_{\max'} \boldsymbol{v}_{\max'}^T \boldsymbol{v}_{\max'} = \lambda_{\max'} \geq \lambda$.

TODO: clearly, what's done here can be repeated for the component along both primary and secondary principle component being reduced. Therefore, we conclude that, when we want to use k components to capture the maximum variance, we should choose the k components corresponding to the k largest eigenvalues.

3.5 Implementation of PCA

In the previous two sub-sections, we discussed the interpretation of eigenvectors of the covariance matrix derived from some data set. Here, we simply show how PCA can be implemented. The following implementation is not guaranteed to be the most efficient; however, it can be easily coded using numerical computing packages such as Numpy (a library in Python).

Algorithm 1: PCA

Input: data set X (shape: (d, N)) with zero mean and unit variance, number of components M
Output: projected data Y (shape: (M, N))

First, We define the following functions; they are already implemented in many packages so we won't bother repeating.

1) `cov(`X`)` $= XX^T$

* It computes the covariance matrix of X.

2) `eig(`Σ`)` $= (\Lambda, U)$

* It computes eigenvalues ($\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_i, \cdots, \lambda_d)$) and eigenvectors ($U$) of Σ. Note that the eigenvalue contained as the i-th diagonal element of Λ corresponds to the eigenvector contained as the i-th column vector of U.

Then, $\Sigma = \text{cov}(X)$; $\Lambda, U = \text{eig}(\Sigma)$.

Then, sort the eigenvalues such that the top-most eigenvalue is the largest eigenvalue; sort the eigenvectors accordingly.

Then, since we have decided to only project X along the first M principal components, we compute $Y = U[:, :M]^T X$, where $U[:, :M]^T$ has shape (M, d).

Finally, output Y.

3.6 Importance of normalizing input data

Real-world data sets tend to have features with wildly different scales. For example, in the UCI wine-quality (red wine) data set available at [UCI wine-quality dataset], feature ... have a variance of ... and feature ... have a variance of ... . Given this

combination, it is obvious that the direction of projection that would give the most variance is closer to the first feature.
Require more thought
First, scale does not change information. Conversely, we don't know
http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv
Changing the scale of a feature does not change the information it is describing.

3.7 Heuristics for picking the number of principal components

4 Data whitening

Given a data set containing d features, we can linearly transform the data set to obtain a identity covariance matrix, without loss of information. This technique is called *data whitening* because (1) the variance of each feature is one and (2) features are completely de-correlated. These properties are often beneficial to or assumed by downstream classifiers.

In my opinion, an excellent starting point is to consider the covariance matrix of M features outputted by PCA, where $M \leq d$. Recall that these M features are obtained by projecting the data set onto the M eigenvectors with the largest eigenvalues.

4.1 Covariance matrix of PCA output

We consider diagonal and off-diagonal elements separately.

Diagonal elements. Consider the i-th PCA feature. This feature contains the projection of data onto $\boldsymbol{v}_i$, where i denotes the fact that $\boldsymbol{v}_i$ has the i-th largest eigenvalue. Assuming that we have normalized each feature of the original data set to have zero mean and unit variance, all features of PCA output would have zero mean. Therefore, the variance of the i-th PCA feature can be computed as follows:

$$\begin{aligned}\sigma_i^2 &= \boldsymbol{v}_i^T \Sigma \boldsymbol{v}_i \\ &= \lambda_i \boldsymbol{v}_i^T \boldsymbol{v}_i \\ &= \lambda_i\end{aligned}$$

Off-diagonal elements. Consider the i-th and j-th PCA feature. The covariance between these two features can be computed as follows:

$$\begin{aligned}\sigma_{i,j}^2 &= \frac{1}{N-1}\sum_{n=1}^{N}\{(\boldsymbol{v}_i^T \boldsymbol{x}^n)(\boldsymbol{v}_j^T \boldsymbol{x}^n)^T\} \\ &= \frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{v}_i^T \boldsymbol{x}^n \boldsymbol{x}^{nT} \boldsymbol{v}_j\} \\ &= \boldsymbol{v}_i^T \frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{x}^n \boldsymbol{x}^{nT}\}\boldsymbol{v}_j \\ &= \boldsymbol{v}_i^T \Sigma \boldsymbol{v}_j \\ &= \lambda_j \underbrace{\boldsymbol{v}_i^T \boldsymbol{v}_j}_{\text{Orthogonal}} \\ &= 0\end{aligned}$$

Therefore, the covariance matrix of the PCA features is diagonal, with the i-th diagonal elements being the variance of the i-th PCA feature, which is λ_i. Recall that one of the goals of data whitening is to transform the data so that the covariance matrix is the identity matrix (diagonal and all diagonal elements are one). Here, by applying PCA, we have already completed the first part of this goal, that is, making the covariance matrix diagonal.

Importantly, the interpretation of the results above *depends on what we want to whiten.*

- If we want to whiten the PCA output, the good news is that we are already halfway done. This is true regardless whether $M < d$ or $M = d$.

- On the other hand, if we want to whiten the original data set from which the PCA output is derived, we would need to beforehand set the number of principal components kept by PCA to d, the dimensionality of the original data set so that no information is discarded by PCA. Since the original data lives in a d-dimensional space, each data point can be expressed in terms of a set of d orthonormal vectors, e.g.., the set of *all* eigenvectors of Σ that represents projection directions.

In both cases, the assumption of no information loss holds.

4.2 Normalize diagonal elements of covariance matrix to ones

In the previous sub-section, we saw that the covariance matrix derived from the output of PCA is diagonal. Now, let's consider what to do next to normalize its diagonal elements to one. Recall from the previous sub-section that the variance of the i-th feature, σ_i^2, is λ_i. We want the new variance of the i-th feature, which we shall denote by $\sigma_i'^2$, to be one. Since $\sigma_i^2/\lambda_i = 1 = \sigma_i'^2$, we have:

$$\begin{aligned}
\sigma_i'^2 &= \frac{\sigma_i^2}{\lambda_i} \\
&= \frac{\boldsymbol{v}_i^T \Sigma \boldsymbol{v}_i}{\lambda_i} \\
&= \boldsymbol{v}_i^T \frac{\Sigma}{\lambda_i} \boldsymbol{v}_i \\
&= \boldsymbol{v}_i^T \frac{\frac{1}{N-1}\sum_{n=1}^{N}\{\boldsymbol{x}^n \boldsymbol{x}^{nT}\}}{\lambda_i} \boldsymbol{v}_i \\
&= \boldsymbol{v}_i^T \frac{1}{N-1}\sum_{n=1}^{N}\{\lambda_i^{-1}\boldsymbol{x}^n \boldsymbol{x}^{nT}\}\boldsymbol{v}_i \\
&= \boldsymbol{v}_i^T \frac{1}{N-1}\sum_{n=1}^{N}\{(\lambda_i^{-1/2}\boldsymbol{x}^n)(\lambda_i^{-1/2}\boldsymbol{x}^{n^T})\}\boldsymbol{v}_i
\end{aligned}$$

which means to we should multiple the i-th feature by $\lambda_i^{-1/2}$.

Now, let's consider what this operation looks like in *matrix form*. The projections of original data set X along M principal components can be expressed as (Algorithm 1):

$$Y = U[:, :M]^T X$$

where Y denotes the output of PCA. As we've derived above, for the covariance of Y to be the identity matrix, we need to multiply each feature by its corresponding eigenvalue raised to $-1/2$ power:

$$Y = \Lambda^{-1/2}[:M, :M]U[:, :M]^T X$$

where $\Lambda^{-1/2}$ denotes the matrix whose i-th diagonal element is $\lambda_i^{-1/2}$ (Algorithm 1).

4.3 Implementation of data whitening

In two previous sub-sections, we discussed about thinking of data whitening as two distinct steps: PCA and normalization of diagonal elements. Since we can choose to either keep all principal components or only a part, this choice determines whether we are talking about data whitening for original data or whitening for projected data. A more detailed analysis of the relationship between PCA and data whitening is given in section 5. Here, we simply outline the steps required to implement data whitening.

Algorithm 2: Data whitening

Input: data set X (shape: (d, N)) with zero mean and unit variance, number of components M
Output:

- If $M = d$, we have whitened data $\tilde{Y}$ (shape: (d, N)).
- If $M < d$, we have whitened projected data $\tilde{Y}$ (shape: (M, N)).

First, run PCA (Algorithm 1) on **Input** and compute $Y = U[:, :M]^T X$ and additionally Λ as PCA output.

* Note that the covariance matrix of Y is now *diagonal* according to sub-section 4.1.

Then, compute $\tilde{Y} = \Lambda^{-1/2} Y$.

* Note that the covariance matrix of $\tilde{Y}$ is now *an identity matrix* according to sub-section 4.2.

Finally, output $\tilde{Y}$.

5 Summary of relationship between PCA and data whitening

Simply put, PCA helps us achieve the first step of data whitening. Nevertheless, it is helpful to consider the following 4 cases to better understand their relationship:

Case 1. $M < d$, $Y = U[:, : M]^T X$ (PCA).

We have (1) a diagonal covariance matrix of Y and (2) dimensionality reduction / information loss.

Case 2. $M < d$, $Y = \Lambda^{-1/2} U[:, : M]^T X$ (data whitening, which involves PCA).

We have (1) an identity covariance matrix of Y and (2) dimensionality reduction / information loss.

Case 3. $M = d$, $Y = U[:, : M]^T X$ (PCA).

We have (1) a diagonal covariance matrix of Y and (2) dimensionality is unchanged.

Case 4. $M < d$, $Y = \Lambda^{-1/2} U[:, : M]^T X$ (data whitening, which involves PCA).

We have (1) an identity covariance matrix of Y and (2) dimensionality is unchanged.