Lecture Notes: Principal Component Analysis

Ted Meeds[1,2]

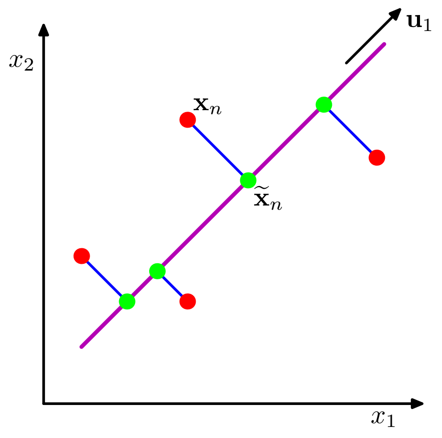
[1] Informatics Institute, University of Amsterdam
[2] The Centre for Integrative Bioinformatics, Vrije University
tmeeds@gmail.com

Abstract In this note we discuss a popular unsupervised learning technique call **Principal Component Analysis** (PCA). PCA has many useful applications, including dimensionality reduction, data exploration, data normalization, and lossy compression. PCA wil be derived from two perspectives, maximizing the variance of projections and minimizing construction error. Two of the applications are discusses, dimensionality reduction and a normalization technique called **sphering**. Finally, PCA in high-dimensional spaces is discussed.

1 Principal Component Analysis

Consider a dataset of digit images like MNIST. Each image is 28 x 28 = 784 dimensional vector. However, we might believe that the data exist on a much lower dimensional **manifold** since many of the pixels are redundant and highly correlated with each other. Using dimensionality reduction techniques we can find a lower dimensional representation of the data that preserves much of the variability in the original data set (i.e. has low reconstruction error). Principal Component Analysis (PCA) is a core unsupervised learning technique that can, among others, can be used for dimensionality reduction.

Another example with images is the application of a rotation operator. Each of these operations can take an original image and rotates it. In pixel space, the images are still 784 dimensions, but the intrinsic dimension is 1, the value of the rotation. We can smoothly rotate the image and by doing so we smoothly move along a 1D manifold. We can think of the unknown manifold as a continuous latent variable which we can then infer. PCA is the simplest continuous latent variable model.



2 Perspective: Maximum Variance Formulation

Goal: **Find a M-dim hyperplane with maximum variance projection**. We start with a single projection then move to M projections.

Let $\mathbf{u}_1$ be the first projection. Calculate the projection and its variance:

$$\begin{aligned}
y_{n1} &= \tilde{x}_n^1 && && \text{(1a)}\\
&= \mathbf{u}_1^T \mathbf{x}_n && \text{single data vector} && \text{(1b)}\\
\bar{y_1} &= \frac{1}{N}\sum_n \mathbf{u}_1^T \mathbf{x}_n && && \text{(1c)}\\
&= \mathbf{u}_1^T \bar{\mathbf{x}} && \text{mean projection} && \text{(1d)}\\
\mathrm{var}(y_1) &= \frac{1}{N}\sum_{n=1}^N (y_{n1} - \bar{y_1})^2 && \text{variance of projection} && \text{(1e)}\\
&= \frac{1}{N}\sum_{n=1}^N (\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}})^2 && && \text{(1f)}\\
&= \frac{1}{N}\sum_{n=1}^N (\mathbf{u}_1^T (\mathbf{x}_n - \bar{\mathbf{x}}))(\mathbf{u}_1^T (\mathbf{x}_n - \bar{\mathbf{x}}))^T && && \text{(1g)}\\
&= \frac{1}{N}\sum_{n=1}^N \mathbf{u}_1^T (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{u}_1 && && \text{(1h)}\\
&= \mathbf{u}_1^T \left(\underbrace{\sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T / N}_{\mathbf{S}\text{: covariance matrix}} \right) \mathbf{u}_1 && && \text{(1i)}\\
&= \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 && && \text{(1j)}
\end{aligned}$$

2.1 Optimizing the first projection

This gives us the objective function: maximize the variance $\mathrm{var}(y_1)$. The *constrained optimization function is*

$$\mathbf{u}_1 = \arg\max_{\mathbf{u}_1} \mathbf{u}_1^T \mathbf{S}\mathbf{u}_1 \qquad \text{subject to } \mathbf{u}_1^T\mathbf{u}_1 = 1 \tag{1k}$$

this constraint is important or the objective can be made arbitrarily large.

The unconstrained optimization function is

$$f(\mathbf{u}_1) = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 - \lambda_1(\mathbf{u}_1^T\mathbf{u}_1 - 1) \tag{1l}$$

We take derivative, set to zero, and find that its solution is the solution to an eigenvalue problem:

$$\frac{\partial f(\mathbf{u}_1)}{\partial \mathbf{u}_1} = 2\mathbf{S}\mathbf{u}_1 - 2\lambda_1\mathbf{u}_1) = 0 \tag{1m}$$

$$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1 \qquad \text{an eigenvalue problem} \tag{1n}$$

with the solution λ_1 and $\mathbf{u}_1$, we can substitute $\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1$ into the variance:

$$\begin{aligned} \mathrm{var}(y_1) &= \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 && (1o)\\ &= \mathbf{u}_1^T\lambda_1\mathbf{u}_1 && (1p)\\ &= \lambda_1\mathbf{u}_1^T\mathbf{u}_1 && (1q)\\ &= \lambda_1 && (1r) \end{aligned}$$

we can then maximize the variance of y_1 by choosing $\mathbf{u}_1$ to be the eigenvector of the largest eigenvalue λ_1.

2.2 Optimizing M projections

In general, we want projections $\mathbf{u}_1, \mathbf{u}_1, \ldots, \mathbf{u}_M$. Let

$$\mathbf{U} = [\mathbf{u}_1 \ \mathbf{u}_1 \ \ldots \ \mathbf{u}_M] \tag{2}$$

and

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_M \end{bmatrix} \tag{3}$$

Step 1: Set-up unconstrained optimization

Objective function:

$$\begin{aligned} J(\mathbf{U}) &= \sum_{i=1}^{M} \mathbf{u}_i^T\mathbf{S}\mathbf{u}_i - \sum_{i=1}^{M} \lambda_i(\mathbf{u}_i^T\mathbf{u}_i - 1) && (4a)\\ &= \mathrm{tr}(\mathbf{U}^T\mathbf{S}\mathbf{U}) - \mathrm{tr}(\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T) + \mathrm{tr}(\boldsymbol{\Lambda}) && (4b) \end{aligned}$$

(notice the transposes within the trace)

Step 2: Solve Eigenvalue Problem

By taking the derivative of J and setting to 0 (using matrix cookbook):

$$\frac{\partial J(\mathbf{U})}{\mathbf{U}} = 2\mathbf{S}\mathbf{U} - 2\mathbf{U}\boldsymbol{\Lambda} = 0 \tag{4c}$$

$$\mathbf{S}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda} \qquad \text{An eigenvalue problem} \tag{4d}$$

Using built in software libraries, we can solve for $\mathbf{U}$, the set of eigenvectors, and $\boldsymbol{\Lambda}$ the corresponding eigenvalues. I.e. we assume we solve for these.

Step 3: Substitute solution into objective

Note that the trace operator has the following property: $\mathrm{tr}(\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T) = \mathrm{tr}(\boldsymbol{\Lambda}\mathbf{U}^T\mathbf{U})$. In addition, since $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and by plugging in $\mathbf{S}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda}$, the objective can be re-written :

$$\begin{aligned} J(\mathbf{U}) &= \mathrm{tr}(\mathbf{U}^T\mathbf{U}\boldsymbol{\Lambda}) - \mathrm{tr}(\boldsymbol{\Lambda}\mathbf{U}^T\mathbf{U}) + \mathrm{tr}(\boldsymbol{\Lambda}) && \text{(4e)} \\ &= \mathrm{tr}(\mathbf{I}\boldsymbol{\Lambda}) - \mathrm{tr}(\boldsymbol{\Lambda}\mathbf{I}) + \mathrm{tr}(\boldsymbol{\Lambda}) && \text{(4f)} \\ &= \mathrm{tr}(\boldsymbol{\Lambda}) && \text{(4g)} \\ &= \sum_{i=1}^{M} \lambda_m && \text{(4h)} \end{aligned}$$

The objective function is sum over M eigenvalues, which is **maximized** by using the max valued eigenvalues; the corresponding eigenvectors $\mathbf{u}_1, \mathbf{u}_1, \ldots, \mathbf{u}_M$ are the M projections.

3 Perspective: Minimize reconstruction error

For this perspective we assume a full set of basis vectors $\{\mathbf{u}_i\}_{i=1}^D$ (i.e. there is the same number of basis vectors as there are coordinates in the original space). Further we assume the basis vectors are orthonormal $\mathbf{u}_i^T\mathbf{u}_j = \delta_{ij}$, where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. Because we have a complete coordinate system, we can write $\mathbf{x}_n$ **exactly** as

$$\mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni}\mathbf{u}_i \tag{5a}$$

where $\alpha_{nj} = \mathbf{x}_n^T \mathbf{u}_j$. To see this, compute $\mathbf{x}_n^T \mathbf{u}_j$

$$\begin{aligned}
\mathbf{x}_n^T \mathbf{u}_j &= \left(\sum_{i=1}^{D} \alpha_{ni} \mathbf{u}_i \right)^T \mathbf{u}_j && \text{(5b)} \\
&= \sum_{i=1}^{D} \alpha_{ni} \mathbf{u}_i^T \mathbf{u}_j && \text{(5c)} \\
&= \sum_{i=1}^{D} \alpha_{ni} \delta_{ij} \qquad \text{due to orthogonality} && \text{(5d)} \\
&= \alpha_{nj} && \text{(5e)}
\end{aligned}$$

Now consider a reconstruction of $\mathbf{x}_n$ using the a coordinate system using the first M basis functions and some shared offsets for the remaining dimensions:

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{M} z_{ni} \mathbf{u}_i + \sum_{i=M+1}^{D} b_i \mathbf{u}_i \tag{5f}$$

where z_{ni} is the ith coordinate for $\mathbf{x}_n$ on hyperplane and b_i is an offset common to all data. We will need to solve for $\{\mathbf{u}_i\}_{i=1}^{M}$, $\{b_i\}_{i=M+1}^{D}$, and $\{z_{ni}\}$. To do this we use the **distortion** cost function:

$$C = \frac{1}{N} \sum_{n=1}^{N} ||\mathbf{x}_n - \tilde{\mathbf{x}}_n||^2 \tag{5g}$$

Step 1: Form of z and b that minimize cost C

$$\begin{aligned}
\frac{\partial C}{\partial z_{nj}} &= \frac{1}{N} 2(\mathbf{x}_n - \tilde{\mathbf{x}}_n)^T \mathbf{u}_j = 0 && \text{(5h)} \\
\mathbf{x}_n^T \mathbf{u}_j &= \tilde{\mathbf{x}}_n^T \mathbf{u}_j && \text{(5i)} \\
&= \left(\sum_{i=1}^{M} z_{ni} \mathbf{u}_i + \sum_{i=M+1}^{D} b_i \mathbf{u}_i \right)^T \mathbf{u}_j \qquad \text{sub back and solve for z} && \text{(5j)} \\
&= \sum_{i=1}^{M} z_{ni} \mathbf{u}_i^T \mathbf{u}_j \qquad \text{same trick as above} && \text{(5k)} \\
&= z_{nj} && \text{(5l)}
\end{aligned}$$

$$\boxed{z_{nj} = \mathbf{x}_n^T \mathbf{u}_j} \tag{5m}$$

Note this assumes $j \leq M$.

$$\frac{\partial C}{\partial b_j} = \frac{1}{N}\sum_n 2(\mathbf{x}_n - \tilde{\mathbf{x}}_n)^T \mathbf{u}_j = 0 \tag{5n}$$

$$\underbrace{\frac{1}{N}\sum_n \mathbf{x}_n^T}_{\bar{\mathbf{x}}^T} \mathbf{u}_j = \frac{1}{N}\sum_n \tilde{\mathbf{x}}_n^T \mathbf{u}_j \tag{5o}$$

$$= \frac{1}{N}\sum_n \left(\sum_{i=M+1}^{D} b_i \mathbf{u}_i\right)^T \mathbf{u}_j \qquad \text{sub in and solve} \tag{5p}$$

$$= \frac{1}{N}\sum_n b_j = b_j \tag{5q}$$

$$\boxed{b_j = \bar{\mathbf{x}}^T \mathbf{u}_j} \tag{5r}$$

Step 2: Plug-in optimal z and b into cost

First note that

$$\mathbf{x}_n - \tilde{\mathbf{x}}_n = \sum_{i=1}^{D} \alpha_{ni}\mathbf{u}_i - \sum_{i=1}^{M} z_{ni}\mathbf{u}_i - \sum_{i=M+1}^{D} b_i \mathbf{u}_i \tag{6a}$$

$$= \underbrace{\sum_{i=1}^{D} (\mathbf{x}_n^T \mathbf{u}_i)\mathbf{u}_i - \sum_{i=1}^{M} (\mathbf{x}_n^T \mathbf{u}_i)\mathbf{u}_i}_{\text{first M terms cancel}} - \sum_{i=M+1}^{D} (\bar{\mathbf{x}}^T \mathbf{u}_i)\mathbf{u}_i \tag{6b}$$

$$= \sum_{i=M+1}^{D} (\mathbf{x}_n^T \mathbf{u}_i)\mathbf{u}_i - (\bar{\mathbf{x}}^T \mathbf{u}_i)\mathbf{u}_i \tag{6c}$$

$$= \sum_{i=M+1}^{D} \left((\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{u}_i\right) \mathbf{u}_i \tag{6d}$$

So the cost simplifies to:

$$
\begin{aligned}
C &= \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - \tilde{\mathbf{x}}_n||^2 && \text{(7a)}\\
&= \frac{1}{N}\sum_{n=1}^{N} || \sum_{i=M+1}^{D} \left((\mathbf{x}_n - \bar{\mathbf{x}})^T\mathbf{u}_i\right)\mathbf{u}_i||^2 && \text{(7b)}\\
&= \frac{1}{N}\sum_{n=1}^{N} \left(\sum_{i=M+1}^{D} \left((\mathbf{x}_n - \bar{\mathbf{x}})^T\mathbf{u}_i\right)\mathbf{u}_i\right)^T \left(\sum_{i=M+1}^{D} \left((\mathbf{x}_n - \bar{\mathbf{x}})^T\mathbf{u}_i\right)\mathbf{u}_i\right) && \text{(7c)}\\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D} \left(\left((\mathbf{x}_n - \bar{\mathbf{x}})^T\mathbf{u}_i\right)\mathbf{u}_i\right)^T \left(\left((\mathbf{x}_n - \bar{\mathbf{x}})^T\mathbf{u}_i\right)\mathbf{u}_i\right) && \text{(7d)}\\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D} \mathbf{u}_i^T(\mathbf{x}_n - \bar{\mathbf{x}})\mathbf{u}_i^T\mathbf{u}_i(\mathbf{x}_n - \bar{\mathbf{x}})^T\mathbf{u}_i && \text{(7e)}\\
&= \sum_{i=M+1}^{D} \mathbf{u}_i^T \sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T/N\mathbf{u}_i && \text{(7f)}\\
&= \sum_{i=M+1}^{D} \mathbf{u}_i^T\mathbf{S}\mathbf{u}_i && \text{(7g)}\\
&= \mathrm{tr}(\mathbf{U}^T\mathbf{S}\mathbf{U}) && \text{(7h)}
\end{aligned}
$$

here $\mathbf{U}$ is $K = M + 1 - D$ by D. Note in the above we used the orthonormal property to bring out the sum over i.

Step 3: Solve eigenvalue problem

Once again we need to contrain each vector to be positive:

$$
\begin{aligned}
C &= \mathrm{tr}(\mathbf{U}^T\mathbf{S}\mathbf{U}) - \mathrm{tr}(\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T) + \mathrm{tr}(\boldsymbol{\Lambda}) && \text{(7i)}\\
\frac{\partial C}{\partial \mathbf{U}} &= 2\mathbf{S}\mathbf{U} - 2\mathbf{U}\boldsymbol{\Lambda} = 0 && \text{(7j)}\\
\mathbf{S}\mathbf{U} &= \mathbf{U}\boldsymbol{\Lambda} \qquad \text{An eigenvalue problem} && \text{(7k)}
\end{aligned}
$$

Step 4: Substitute solution into objective

Again, since $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and by plugging in $\mathbf{S}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda}$, the objective can be re-written :

$$
\begin{aligned}
C &= \mathrm{tr}(\mathbf{U}^T\mathbf{U}\boldsymbol{\Lambda}) - \mathrm{tr}(\boldsymbol{\Lambda}\mathbf{U}^T\mathbf{U}) + \mathrm{tr}(\boldsymbol{\Lambda}) && \text{(7l)}\\
&= \mathrm{tr}(\mathbf{I}\boldsymbol{\Lambda}) - \mathrm{tr}(\boldsymbol{\Lambda}\mathbf{I}) + \mathrm{tr}(\boldsymbol{\Lambda}) && \text{(7m)}\\
&= \mathrm{tr}(\boldsymbol{\Lambda}) && \text{(7n)}\\
&= \sum_{i=M+1}^{D} \lambda_i && \text{(7o)}
\end{aligned}
$$

The objective function is sum over K eigenvalues, which is **minimized** by using the min valued eigenvalues. The M basis vectors are therefore the eigenvectors of the M max valued eigen values.

Note that above we kind of pushed under the carpet subbing in $\mathbf{U}\boldsymbol{\Lambda}$. The eigenvalue problem solves $\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$; if this is subbed into $\sum_{i=M+1}^{D} \mathbf{u}_i^T \mathbf{S}\mathbf{u}_i$, the result is $\sum_{i=M+1}^{D} \mathbf{u}_i^T \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T \mathbf{u}_i = \sum_{i=M+1}^{D} \lambda_i$.

4 Uses of PCA

In Python we can solve for the projections by:

1. `S = cov(X.T)` compute covariance
2. `U,W,V = svd(S)` singular value decomposition

4.1 Dimensionality Reduction

Now that we have the solution, we can project our data on to the new coordinate space. First we subtract the mean, then project:

$$\mathbf{y}_n = \mathbf{U}(\mathbf{x}_n - \bar{\mathbf{x}}) \tag{8}$$

I.e. the steps are 1) center the data by removing the mean, 2) rotate the data using $\mathbf{U}$. We can then make scatter plots of y_{n1} versus y_{n2}, for example.

4.2 Data Normalization

Standardizing the features of data sets is common practice. This can mean different things, but generally we mean that the data matrix has zero mean and standard deviation 1, for each column. With PCA we can go one step further and decorrelate the features by rotating them onto orthogonal basis vectors. Sphering is one further step from the above centering and rotating, it scales the projected values to have variance 1. Since the rotation is an orthogonal basis, the off-diagonal covariances are 0. This where the term **sphering** comes from, equal variances and no covariances.

$$\begin{aligned}
\tilde{y}_{in} &= \lambda_i^{-1/2}\mathbf{u}_i^T(\mathbf{x}_n - \bar{\mathbf{x}}) && \text{(9a)}\\
<\tilde{y}_i> &= \frac{1}{N}\sum_n \lambda_i^{-1/2}\mathbf{u}_i^T(\mathbf{x}_n - \bar{\mathbf{x}}) && \text{(9b)}\\
&= \lambda_i^{-1/2}\mathbf{u}_i^T\left(\frac{1}{N}\sum_n(\mathbf{x}_n - \bar{\mathbf{x}})\right) && \text{(9c)}\\
&= \lambda_i^{-1/2}\mathbf{u}_i^T(\bar{\mathbf{x}} - \bar{\mathbf{x}}) && \text{(9d)}\\
&= 0 && \text{(9e)}\\
\sigma_{ij} &= \frac{1}{N}\sum_n \tilde{y}_{in}\tilde{y}_{jn} && \text{(9f)}\\
&= \frac{1}{N}\sum_n \left(\lambda_i^{-1/2}\mathbf{u}_i^T(\bar{\mathbf{x}} - \bar{\mathbf{x}})\right)\left((\bar{\mathbf{x}} - \bar{\mathbf{x}})^T\mathbf{u}_j\lambda_j^{-1/2}\right) && \text{(9g)}\\
&= \lambda_i^{-1/2}\lambda_j^{-1/2}\mathbf{u}_i^T\left(\frac{1}{N}\sum_n(\bar{\mathbf{x}} - \bar{\mathbf{x}})(\bar{\mathbf{x}} - \bar{\mathbf{x}})^T\right)\mathbf{u}_j && \text{(9h)}\\
&= \lambda_i^{-1/2}\lambda_j^{-1/2}\underbrace{\mathbf{u}_i^T\mathbf{S}\mathbf{u}_j}_{=0 \text{ if } i\neq j \quad \lambda_i \text{ o.w.}} && \text{(9i)}\\
&= \lambda_i^{-1/2}\lambda_j^{-1/2}\lambda_i\delta_{ij} && \text{(9j)}\\
&= \lambda_i^{-1/2}\lambda_i^{-1/2}\lambda_i\delta_{ij} && \text{(9k)}\\
&= \delta_{ij} && \text{(9l)}
\end{aligned}$$

in other words $\{\tilde{y}_{in}\}$ are decorrelated; have diagonal covariance of 1.

Recap of whitening/sphering:

1. Center each data vector: $\mathbf{x}_n - \bar{\mathbf{x}}$.
2. Rotate centered data: $\mathbf{y}_n = U(\mathbf{x}_n - \bar{\mathbf{x}})$ ($\mathbf{y}$ is M dimensional).
3. Scale rotated data to have equal (unity) variance: $\tilde{\mathbf{y}}_n = \boldsymbol{\Lambda}^{-1/2}U(\mathbf{x}_n - \bar{\mathbf{x}})$

5 High-dimensional PCA

When $N < D$, then there are at most $N-1$ eigenvectors. Assume $\mathbf{X}$ is N by D and has been centered. Further we will assume $N \ll D$ and that D is very large.

The usual eigenvalue problem is too big: $\mathbf{S} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$ (D by D), implies an eigenvalue problem $\mathbf{SU} = \mathbf{U}\boldsymbol{\Lambda}$. We apply a simply trick to make this feasible (due to memory requirements, computational complexity).

$$\mathbf{S}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda} \tag{10a}$$

$$\frac{1}{N}\mathbf{X}^T\mathbf{X}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda} \tag{10b}$$

$$\frac{1}{N}\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{U} = \mathbf{X}\mathbf{U}\boldsymbol{\Lambda} \qquad \text{left multiply by } \mathbf{X} \tag{10c}$$

$$\frac{\mathbf{X}\mathbf{X}^T}{N}\underbrace{(\mathbf{X}\mathbf{U})}_{\mathbf{V}} = (\mathbf{X}\mathbf{U})\boldsymbol{\Lambda} \tag{10d}$$

$$\frac{\mathbf{X}\mathbf{X}^T}{N}\mathbf{V} = \mathbf{V}\boldsymbol{\Lambda} \tag{10e}$$

This gives an eigenvalue problem that is N by N instead.

We now need to work back from $\mathbf{V}$ to $\mathbf{U}$:

$$\frac{\mathbf{X}\mathbf{X}^T}{N}\mathbf{V} = \mathbf{V}\boldsymbol{\Lambda} \tag{10f}$$

$$\mathbf{X}^T\frac{\mathbf{X}\mathbf{X}^T}{N}\mathbf{V} = \mathbf{X}^T\mathbf{V}\boldsymbol{\Lambda} \qquad \text{left-multiply by } \mathbf{X}^T \tag{10g}$$

$$\frac{\mathbf{X}^T\mathbf{X}}{N}(\mathbf{X}^T\mathbf{V}) = (\mathbf{X}^T\mathbf{V})\boldsymbol{\Lambda} \tag{10h}$$

$$\frac{\mathbf{X}^T\mathbf{X}}{N}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda} \tag{10i}$$

So, we have that $\mathbf{U} = \mathbf{X}^T\mathbf{V}$. (solve $\mathbf{V}$, then multiply to get $\mathbf{U}$). Note these are unnormalized eigenvectors. Assuming $\mathbf{V}$ eigenvectors are normalized, is $\mathbf{U} = N^{-1/2}\boldsymbol{\Lambda}^{-1/2}\mathbf{X}^T\mathbf{V}$.