CS.479/679, Fall 2015, Representation Learning Lecture 4 - 09/10/2015

Subspace Learning

Lecturer: Dr. Raman Arora *Scribe: Corbin Rosset*

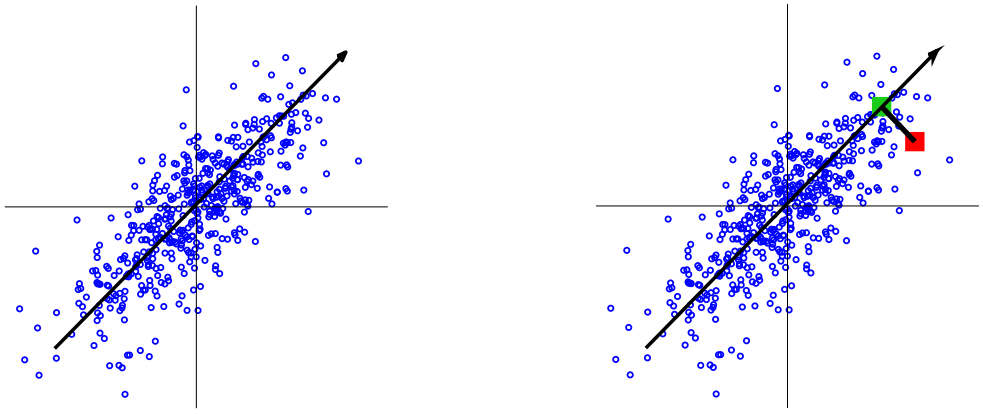
1 What is Subspace Learning

Many raw data sets in machine learning capture high dimensional representations of complex natural phenomena. Yet it is often the case that natural processes are largely controlled by finitely many simpler mechanisms that operate in lower dimensions. For example, a pixel-by-pixel image representation of a data set comprised of rotations of a single object rotated in 3-space under constant light conditions may appear to be high dimesional: shadows cast by surfaces on the face, as well as the different perspectives, make the images appear highly varied, even though the only real parameter in this scenario is the angle of rotation of the object. Instead of representing each image by an array of pixels, each image is merely a linear transformation of a source image. Such transformations of data are common and they only provide a few degrees of freedom with respect to the dimensionality of the dataset. Our task is to seek a low dimensional representation of the data that captures these latent variables.

Figure 1: Seemingly high dimesional variance caused by the rotation of a single object

2 Problem Specification

We assume that the data can be represented in a substantially lower dimensional space such that when the data is transformed back ("reconstructed") into its original space, the loss of information is minimal. In this way, subspace learning is intuitively a version of lossy compression. Concretely, say the original space is $\mathbb{R}^2$ and the data therein is plotted in Figure 2:



(a) Lower dimensional subspace superimposed on the original data

(b) The learned subspace minimizes squared error of projection

Figure 2: The goal of subspace learning is to find a low dimensional linear subspace that captures the most information or variance in the data. It is not to be confused with linear regression.

Given n data points in $\mathbb{R}^d$, find a projection matrix $U \in \mathbb{R}^{d \times k}$ such that the projection $\hat{\mathrm{x}} \in \mathbb{R}^k$ of a data vector $x \in \mathbb{R}^d$ given by $\hat{\mathrm{x}} = \mathrm{U}\mathrm{U}^\top \mathrm{x}$ minimizes the empirical reconstruction error:

$$error = \frac{1}{n}\sum_{i=1}^{n} \|\mathrm{x}_i - \mathrm{U}\mathrm{U}^\top \mathrm{x}_i\|_2^2 = \frac{1}{n}\sum_{i=1}^{n} \left(\|\mathrm{x}\|_2^2 - \|\mathrm{U}^\top \mathrm{x}_i\|_2^2\right) \tag{1}$$

The above equality is derived:

$$\begin{aligned}
\|\mathrm{x}_i - \mathrm{U}\mathrm{U}^\top \mathrm{x}_i\|_2^2 &= (\mathrm{x}_i - \mathrm{U}\mathrm{U}^\top \mathrm{x}_i)^\top (\mathrm{x}_i - \mathrm{U}\mathrm{U}^\top \mathrm{x}_i) \\
&= \mathrm{x}_i^\top \mathrm{x}_i - \mathrm{x}_i^\top \mathrm{U}\mathrm{U}^\top \mathrm{x}_i - \mathrm{x}_i^\top \mathrm{U}\mathrm{U}^\top \mathrm{x}_i + \mathrm{x}_i^\top \mathrm{U}^\top \mathrm{U}\mathrm{U}\mathrm{U}^\top \mathrm{x}_i \\
&= \mathrm{x}_i^\top \mathrm{x}_i - \mathrm{x}_i^\top \mathrm{U}\mathrm{U}^\top \mathrm{x}_i && \text{recall } \mathrm{U}^\top \mathrm{U} = \mathrm{I} \\
&= \|\mathrm{x}_i\|_2^2 - \|\mathrm{U}^\top \mathrm{x}_i\|_2^2 \\
&= \|\mathrm{x}\|_2^2 - \|\mathrm{U}^\top \mathrm{x}_i\|_2^2
\end{aligned}$$

In Equation 1, the term $\frac{1}{n}\sum_{i=1}^{n} \|\mathrm{x}_i\|_2^2$ is constant given any data set X. So in order to minimize the reconstruction, or projection, error, we must maximize $\frac{1}{n}\sum_{i=1}^{n} \|\mathrm{U}^\top \mathrm{x}_i\|_2^2$. We will show that this is equivalent to maximizing the variance captured by the subspace, that is, maximizing the trace of the empirical covariance matrix of X:

$$\underbrace{\left[\frac{1}{n}\sum_{i=1}^{n}\|\mathrm{U}^\top \mathrm{x}_i\|_2^2\right]}_{\text{Variance captured by U}} = \left[\frac{1}{n}\sum_{i=1}^{n}\mathrm{x}_i^\top \mathrm{U}\mathrm{U}^\top \mathrm{x}_i\right] = \text{trace}\left(\frac{1}{2}\sum_{i=1}^{n}\mathrm{U}^\top \mathrm{x}_i\mathrm{x}_i^\top \mathrm{U}\right) = \text{trace}\left(\mathrm{U}^\top \underbrace{\left[\frac{1}{n}\sum_{i=1}^{n}\mathrm{x}_i\mathrm{x}_i^\top\right]}_{\text{Emp. Covariance}} \mathrm{U}\right) \tag{2}$$

Since U is unique, minimizing the error is equal to maximizing the trace of the empirical covariance matrix of X; the diagonal of any covariance matrix contains the variance of each data point.

3 Purpose and Applications

As we will see, PCA is a type of subspace learning. PCA is ubiquitous in machine learning, data analysis, information retrieval, and is expanding into data intensive fields like bioinformatics and even drug discovery. The purpose of PCA is two fold. Firstly, to reduce the dimensionality of the feature space: many machine machine learning models such as clustering have sample complexities that depend exponentially on the dimension of the data. High dimensionality can then lead to poor generalization. The second objective is to learn a unifying and governing structure behind noisy data, specifically, learning uncorrelated components of data. Lower dimensional data can also make the structure more interpretable, allowing causal relationships to be more easily identified. Moreover, independent components of influence behind data makes a system more robust to corruption and noise.

4 PCA

In principle component analysis, we attempt to find a subspace comprised of k normalized linearly independent and uncorrelated basis vectors $\{u_i\}$, $i = 1, ..., k$ where the i'th component y_i of x is $y_i = u_i^\top x$ and $Var(y_1) \geq Var(y_2) \geq ... \geq Var(y_k)$. These principle components in fact yield from the top k eigenvectors of the empirical covariance matrix of X: $C_{XX} = \mathrm{E}[\mathrm{x}\mathrm{x}^\top]$. If a covariance matrix is not at hand, then it suffices to use singular value decomposition of a data matrix normalized and centered for each attribute.

We inspect the mean and variance of y_i:

$$\mathrm{E}[y_i] = \mathrm{E}[u_i^\top \mathrm{x}_i] = u_i^\top \mathrm{E}[x_i] = 0 \tag{3}$$

since the data is centered, $\mathrm{E}[x_i] = 0$. This simplifies the value of the variance:

$$Var[y_i] = \mathrm{E}[(u_i^\top \mathrm{x}_i)^2] - (\mathrm{E}[u_i^\top \mathrm{x}_i])^2 = u_i^\top \mathrm{E}[x_i] = \mathrm{E}[(u_i^\top \mathrm{x}_i)^2] - 0 \tag{4}$$

4.1 Finding the First Principle Component

The first principal component should capture a largest amount of variance contained in the data vector x. From the definition above we solve the optimization problem:

$$\begin{aligned} \underset{\mathrm{u}_1 \in \mathbb{R}^d}{\text{maximize}} \quad & \mathrm{Var}(\mathrm{u}_1^\top \mathrm{x}) \\ \text{subject to} \quad & \mathrm{u}_1^\top \mathrm{u}_1 = 1 \end{aligned}$$

where $\mathrm{Var}(\mathrm{u}_1^\top \mathrm{x}) = \mathrm{E}[(\mathrm{u}_1^\top \mathrm{x})^2] = \mathrm{E}[\mathrm{u}_1^\top \mathrm{x}\mathrm{x}^\top \mathrm{u}_1] = \mathrm{u}_1 \mathrm{E}[\mathrm{x}\mathrm{x}^\top]\mathrm{u}_1 = \mathrm{u}_1^\top C_{\mathrm{xx}} \mathrm{u}_1$. We solve the maximization problem by defining the Lagrangian: $\mathcal{L} = \mathrm{u}_1^\top C_{\mathrm{xx}} \mathrm{u}_1 + \lambda_1(1 - \mathrm{u}_1^\top \mathrm{u}_1)$, for $\lambda_1 \in \mathbb{R}$. Setting $\partial\mathcal{L}/\partial \mathrm{u}_1 = 2C_{\mathrm{xx}}\mathrm{u}_1 - 2\lambda \mathrm{u}_1 = 0$ yields $C_{\mathrm{xx}}\mathrm{u}_1 = \lambda_1 \mathrm{u}_1$. Notice u_1 meets the definition of being an eigenvector of the covariance matrix, and the corresponding eigenvalue $\lambda_1 = \lambda_1 \mathrm{u}_1^\top \mathrm{u}_1 = \mathrm{u}_1^\top C_{\mathrm{xx}} \mathrm{u}_1 = \mathrm{Var}(\mathrm{u}_1^\top \mathrm{x})$ is the maximum eigenvalue of C_{xx}, which satisfies the optimization problem.

4.2 Subsequent Principal Components

The second principal component should capture the largest variance in the data set given it is linearly uncorrelated and orthogonal from the first. We would suspect the second principle component to be the second eigenvector of C_{xx}, and eigenvectors corresponding to unique eigenvalues of a symmetric matrix (such as the covariance matrix) are perpindicular.

For y_1 and y_2 to be uncorrelated, their covariance must be zero, i.e. $\mathrm{E}[y_1 y_2] - \mathrm{E}[y_1]\mathrm{E}[y_2] = 0$. This amounts to $\mathrm{E}[y_1 y_2] = 0$ by Equation 3. Expanding, $\mathrm{E}[y_1 y_2] = \mathrm{E}[(\mathrm{u}_1^\top \mathrm{x})(\mathrm{u}_2^\top \mathrm{x})] = \mathrm{u}_1^\top C_{\mathrm{xx}} \mathrm{u}_2 = \lambda_1 \mathrm{u}_1^\top \mathrm{u}_2 = 0$, which gives $\mathrm{u}_1^\top \mathrm{u}_2 = 0$. Conveniently, the constraint necessary for principle components to be uncorrelated is merely that they are orthogonal (and unit normalized). We can now find the second principle component:

$$\begin{aligned} \underset{\mathrm{u}_2 \in \mathbb{R}^d}{\text{maximize}} \quad & \mathrm{Var}(\mathrm{u}_2^\top \mathrm{x}) \\ \text{subject to} \quad & \mathrm{u}_2^\top \mathrm{u}_2 = 1, \mathrm{u}_1^\top \mathrm{u}_2 = 0 \end{aligned}$$

We proceed as before, defining the Lagrangian as $\mathcal{L} = \mathrm{u}_2^\top C_{\mathrm{xx}} \mathrm{u}_2 + \lambda_2(1 - \mathrm{u}_2^\top \mathrm{u}_2) + \lambda_{12} \mathrm{u}_1^\top \mathrm{u}_2$. Setting $\partial \mathcal{L} / \partial \mathrm{u}_2 = 2C_{\mathrm{xx}} \mathrm{u}_2 - 2\lambda \mathrm{u}_2 + \lambda_{12} \mathrm{u}_1 = 0$ yields $C_{\mathrm{xx}} \mathrm{u}_2 + \frac{\lambda_{12}}{2} \mathrm{u}_1 = \lambda_2 \mathrm{u}_2$. Multiplying on left by $\mathrm{u}_1^\top$: $\mathrm{u}_1^\top C_{\mathrm{xx}} \mathrm{u}_2 + \frac{\lambda_{12}}{2} \mathrm{u}_1^\top \mathrm{u}_1 = \lambda_2 \mathrm{u}_1^\top \mathrm{u}_2$ yields $\lambda_{12} = 0$. We get finally we get $C_{\mathrm{xx}} \mathrm{u}_2 = \lambda_2 \mathrm{u}_2$, then $\mathrm{Var}(\mathrm{u}_2^\top \mathrm{x}) = \mathrm{u}_2^\top C_{\mathrm{xx}} \mathrm{u}_2 = \lambda_2$, and u_2 is the eigenvector corresponding to the second largest eigenvalue. Note in this proof we could have also taken the derivative of the Lagrangian with respect to u_1.

By induction, the variance of the projected data is maximized by the top k eigenvectors of the covariance matrix of data.

4.3 Solving for the Projection Matrix

We saw that the principal components are given as top-k singular vectors of $\mathrm{X} = [\mathrm{x}_1, \ldots, \mathrm{x}_n]$, or given as top-$k$ eigenvectors of $\mathrm{X}\mathrm{X}^\top$. We must find a projection $\mathrm{U}\mathrm{U}^\top$ to minimize the reconstruction error. Then U will provide a geometric description of the dominant subspace structure for all data points. In this section we will solve for U.

Theorem 1 (Principal subspace)**.** *Given a set of points $\{\mathrm{x}_j\}_{j=1}^n$ in $\mathbb{R}^d$, the k-dimensional subspace $\mathrm{S} \subseteq \mathbb{R}^d$ that best fits these points, i.e. minimizes the residual error*

$$\begin{aligned} \underset{U \in \mathbb{R}^{d \times k},\ y_j \in \mathbb{R}^k}{\textit{minimize}} \quad & \sum_{j=1}^n \|x_j - U y_j\|_2^2 \\ \textit{subject to} \quad & U^\top U = I_k \end{aligned}$$

is spanned by the top-k eigenvectors of $\mathrm{X}\mathrm{X}^\top$, where $\mathrm{X} \in \mathbb{R}^{d \times n} = [\mathrm{x}_1, \ldots, \mathrm{x}_n]$.

As alluded to in Figure 2, if U were known, this would be a least squares regression problem. Since it is not known, PCA is a joint optimization problem. We solve for U by again defining the lagrangian:

$$\mathcal{L} = \sum_{j=1}^n \|\mathrm{x}_j - \mathrm{U}\mathrm{y}_j\|_2^2 + \mathrm{trace}\Big(\Big(\mathrm{I} - \mathrm{U}^\top \mathrm{U} \Big) \Lambda \Big)$$

for $\Lambda = \Lambda^\top$. Setting $\frac{\partial \mathcal{L}}{\partial \mathrm{y}_j} = -2\mathrm{U}^\top(\mathrm{x}_j - \mathrm{U}\mathrm{y}_j) = 0$ yields $\mathrm{y}_j = \mathrm{U}^\top \mathrm{x}_j$. Upon substitution, notice that

$$\begin{aligned}\sum_{j=1}^{n} \|\mathrm{x}_j - \mathrm{U}\mathrm{y}_j\|_2^2 &= \sum_{j=1}^{n} (\mathrm{x}_j - \mathrm{U}\mathrm{U}^\top \mathrm{x}_j)^\top (\mathrm{x}_j - \mathrm{U}\mathrm{U}^\top \mathrm{x}_j) \\ &= \sum_{j=1}^{n} (\mathrm{x}_j^\top \mathrm{x}_j - \mathrm{x}_j^\top \mathrm{U}\mathrm{U}^\top \mathrm{x}_j) \\ &= \mathrm{trace}(\mathrm{X}^\top \mathrm{X} - \mathrm{X}^\top \mathrm{U}\mathrm{U}^\top \mathrm{X})\end{aligned}$$

Upon inspection, minimizing, $\mathrm{trace}(\mathrm{X}^\top \mathrm{X} - \mathrm{X}^\top \mathrm{U}\mathrm{U}^\top \mathrm{X})$ is equivalent to maximizing $\mathrm{trace}(\mathrm{X}^\top \mathrm{U}\mathrm{U}^\top \mathrm{X})$ over all $\mathrm{U} \in \mathbb{R}^{d \times k}$ since X is given. We now have a new objective function: $\mathrm{trace}(\mathrm{X}^\top \mathrm{X} - \mathrm{X}^\top \mathrm{U}\mathrm{U}^\top \mathrm{X})$, so we rewrite

$$\begin{aligned} \underset{\mathrm{U} \in \mathbb{R}^{d \times k}}{\text{maximize}} \quad & \mathrm{trace}(\mathrm{X}^\top \mathrm{X} - \mathrm{X}^\top \mathrm{U}\mathrm{U}^\top \mathrm{X}) \\ \text{subject to} \quad & \mathrm{U}^\top \mathrm{U} = \mathrm{I}_k \end{aligned} \tag{5}$$

The Lagrangian is given by:

$$\mathcal{L} = \mathrm{trace}(\mathrm{X}^\top \mathrm{U}\mathrm{U}^\top \mathrm{X}) + \mathrm{trace}\Big(\Big(\mathrm{I}_k - \mathrm{U}^\top \mathrm{U} \Big) \Lambda \Big)$$

for a $\Lambda = \Lambda^\top$. Setting $\partial \mathrm{L} / \partial \mathrm{U} = 2\mathrm{X}\mathrm{X}^\top \mathrm{U} - 2\mathrm{U}\Lambda = 0$ yields $\mathrm{X}\mathrm{X}^\top \mathrm{U} = \mathrm{U}\Lambda$.

There is one small problem here: Λ is not guaranteed to be diagonal. Write $\Lambda = \mathrm{U}^\top \mathrm{X}\mathrm{X}^\top \mathrm{U}$ a singular value decomposition for $\mathrm{X}\mathrm{X}^\top$. The objective for Equation 5 is then $\mathrm{trace}(\Lambda)$ since $\mathrm{trace}(\mathrm{U}^\top \mathrm{X}\mathrm{X}^\top \mathrm{U}) = \mathrm{trace}(\mathrm{X}^\top \mathrm{U}\mathrm{U}^\top \mathrm{X})$. If U is a solution to Equation 5 then so is $\tilde{\mathrm{U}} = \mathrm{U}\mathrm{V}$ for any orthogonal V. Choose V to be eigenvectors of $\Lambda = \mathrm{V}\tilde{\Lambda}\mathrm{V}^\top$. Then $\mathrm{X}\mathrm{X}^\top \tilde{\mathrm{U}} = (\mathrm{X}\mathrm{X}^\top \mathrm{U})\mathrm{V} = (\mathrm{U}^\top)^{-1}\Lambda\mathrm{V} = \mathrm{U}\mathrm{V}\tilde{\Lambda}\mathrm{V}^\top \mathrm{V} = (\mathrm{U}\mathrm{V})\tilde{\Lambda} = \tilde{\mathrm{U}}\tilde{\Lambda}$ is the solution to Equation 5 with objective function $\mathrm{trace}(\tilde{\Lambda})$. Notice, the columns of $\tilde{\mathrm{U}}$ are eigenvectors of $\mathrm{X}\mathrm{X}^\top$ with corresponding eigenvalues in $\tilde{\Lambda}$.

Since objective is $\mathrm{trace}(\tilde{\Lambda})$, $\tilde{\mathrm{U}}$ must be the top-k eigenvectors. Also, the top k eigenvectors of $\mathrm{X}\mathrm{X}^\top \equiv$ top k left singular vectors of X, meaning that SVD also gives an optimal solution to PCA. We can further write that $\mathrm{Y} = \mathrm{U}_k^\top \mathrm{X} = [\mathrm{y}_1, \ldots, \mathrm{y}_n] \in \mathbb{R}^{k \times n}$ is a more compact representation for X as typically $k \ll d$.

5 PCA Algorithms

5.1 Exact Solution: When the Covariance Matrix is computable

Below is the algorithm to compute the optimal projection matrix for PCA. The input is a data matrix $\mathrm{X} \in \mathbb{R}^{d \times n}$ for n examples each with dimension d.

$\texttt{pca}\left(k : \mathbb{N}, \mathrm{X} = [\mathrm{x}_1, \ldots, \mathrm{x}_n] : \mathbb{R}^{d \times n}\right)$

1 $\quad \mathrm{X} \leftarrow \mathrm{X} - \mathrm{X}\mathbf{1}_n \mathbf{1}_n^\top$;

2 $\quad \hat{C}_{\mathrm{xx}} \leftarrow \frac{1}{n-1}\mathrm{X}\mathrm{X}^\top$;

3 $\quad \mathrm{U}, \lambda \leftarrow \texttt{eig}(\hat{C}_{\mathrm{xx}})$;

4 $\quad \texttt{return}\ \mathrm{U}_{1:k}, \lambda_{1:k}$;

Algorithm 1: Principal component analysis.

The computational cost of computing the eigendecomposition above of a matrix of size $d \times d$ is $O(d^3)$. Furthermore, covariance matrix of $X \in \mathbb{R}^{d \times k}$ is $\frac{1}{n}\mathrm{X}\mathrm{X}^\top \in \mathbb{R}^{d \times d}$. Usually, if PCA is applied to a dataset, it means d is quite large, so the exact computation of U can be prohibitive. For this reason, efficient approximation algorithms for U are given below.

5.2 Approximation 1: Power Iteration Method

Given a real matrix $A \in \mathbb{R}^{d \times d}$, such that $A^\top = A$, such that

- A has d linearly independent eigenvectors, $\{\mathrm{v}_1, \ldots, \mathrm{v}_d\}$
- The eigenvalues can be ordered in magnitude as $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_d| > 0$

---

input symmetric matrix $A \in \mathbb{R}^{d \times d}$; number of iterations m

1. Sample $\mathrm{v}^{(0)} \in \mathbb{R}^d$ randomly
2. **for** $i = 1, 2, \ldots, m$ **do**
3. $\quad \tilde{\mathrm{v}}^{(i)} = A\mathrm{v}^{(i-1)}$
4. $\quad \mathrm{v}^{(i)} = \frac{\mathrm{v}^{(i)}}{\|\mathrm{v}^{(i)}\|_2}$
5. **end for**

output $\hat{\mathrm{v}}_1 = \mathrm{v}^{(m)}$, the approximate eigenvector

---

The algorithm returns an approximation for the first principal component. We prove this by first noting that any vector $\mathrm{v}^{(0)} \in \mathbb{R}^d$ can be written as $\mathrm{v}^{(0)} = c_1\mathrm{v}_1 + c_2\mathrm{v}_2 + \cdots + c_d\mathrm{v}_d$ where $\{\mathrm{v}_1, \ldots, \mathrm{v}_d\}$ is the set of linearly independent eigenvectors of A. Then

$$\begin{aligned} A\mathrm{v}^{(0)} &= c_1\lambda_1\mathrm{v}_1 + \cdots + c_d\lambda_d\mathrm{v}_d \\ A^2\mathrm{v}^{(0)} &= c_1\lambda_1^2\mathrm{v}_1 + \cdots + c_d\lambda_d^2\mathrm{v}_d \\ \cdots &\quad \cdots \\ A^m\mathrm{v}^{(0)} &= c_1\lambda_1^m\mathrm{v}_1 + \cdots + c_d\lambda_d^m\mathrm{v}_d \end{aligned}$$

Dividing by λ_1^m gives

$$\frac{1}{\lambda_1^m}A^m\mathrm{v}^{(0)} = c_1\mathrm{v}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^m \cdots + c_d\left(\frac{\lambda_d}{\lambda_1}\right)^m$$

As $m \to \infty$, $\frac{\lambda_j}{\lambda_1} \to 0$ for all $j > 1$ and only the term c_1v_1 survives for large m: $\frac{1}{\lambda_1^m}A^m\mathrm{v}^{(0)} = c_1\mathrm{v}_1$ where $\hat{\mathrm{v}}_1 = A^m\mathrm{v}^{(0)}$ is approximately an eigenvector corresponding to λ_1. If we choose $\mathrm{v}^{(0)} \not\perp \mathrm{v}_1 \Rightarrow c_1 \neq 0$, then when convergence is reached, for any $\mathrm{y} \not\perp \mathrm{v}_1$, we can approximate λ_1 as the ratio of two terms:

$$\begin{aligned} \frac{1}{\lambda_1^m}\left\langle A^m\mathrm{v}^{(0)}, \mathrm{y}\right\rangle &= c_1\mathrm{v}_1^\top\mathrm{y} \\ \frac{1}{\lambda_1^{m+1}}\left\langle A^{m+1}\mathrm{v}^{(0)}, \mathrm{y}\right\rangle &= c_1\mathrm{v}_1^\top\mathrm{y} \end{aligned}$$

which is

$$\frac{\mathrm{y}^\top A^{m+1}\mathrm{v}^{(0)}}{\mathrm{y}^\top A^m\mathrm{v}^{(0)} \cdot \mathrm{y}} = \frac{\lambda_1^{m+1}}{\lambda_1^m} = \lambda_1$$

We stop iterating when relative change in estimated eigenvalue is small. To find subsequent eigenvectors, run the power iteration method on $A = A - \lambda_1 \hat{\mathrm{v}}_1 \hat{\mathrm{v}}_1^\top$; recall $A = \mathrm{U}\Lambda\mathrm{U}^\top = \sum_{i=1}^n \lambda_i \mathrm{v}_i \mathrm{v}_i^\top$

Note: the power iteration method converges with fewer iterations if there is a large eigengap, meaning $\lambda_i \gg \lambda_j$ for any $i > j$. Typically, the first few eigenvectors are discovered readily, while it takes exponentially more iterations to converge to less dominant eigenvalues/eigenvectors. The other consideration to note is that if $\lambda_i = \lambda_j$ for $i > j$, then the returned eigenvector approximations $\hat{v_1}$, $\hat{v_2}$ will span the space of the true eigenvectors v_1 and v_2, that is, the algorithm may not recover the true eigenvectors, but rather a linear combination thereof.

5.3 Approximation 2: Orthogonal Iteration Method

Below is an algorithm to approximate the top-k eigenvectors simultaneously.

---

input symmetric matrix $A \in \mathbb{R}^{d \times d}$; number of iterations m

1. Sample $\mathrm{U}_0 \in \mathbb{R}^{d \times k}$ such that $\mathrm{U}_0^\top \mathrm{U}_0 = \mathrm{I}_k$
2. **for** $i = 1, 2, \ldots, m$ **do**
3. $\quad \tilde{\mathrm{U}}^{(i)} = A\mathrm{U}^{(i-1)}$.
4. $\quad \mathrm{U}^{(i)} =$ [matrix of orthonormal basis vectors for range $(\tilde{\mathrm{U}}^{(i)})$
5. **end for**

output $\mathrm{U} = \mathrm{U}^{(m)}$

---

where U_0 can be constructed from any randomly sampled matrix by performing Gram-Schmidt.

6 Dimensionality Considerations

6.1 Not Enough Samples

What if the dimension d of any vector in the space spanned by the data is much larger than n the number of examples? [2] This might occur, say, with having only a few hundred very high resolution pictures of millions of pixels. Trivially, it makes no sense to find k principle components for $k > n - 1$ because at least $d - n + 1$ eigenvalues of the data matrix are zero.

As before, define the empiral covariance matrix to be $C = \frac{1}{n}\mathrm{X}\mathrm{X}^\top$, an eigenvector u_i of which satisfies $\frac{1}{n}\mathrm{X}\mathrm{X}^\top \mathrm{u}_i = \lambda_i \mathrm{u}_i$. If we premultiply by X to obtain:

$$\frac{1}{n}\mathrm{X}\mathrm{X}^\top(\mathrm{X}\mathrm{u}_i) = \lambda_i(\mathrm{X}\mathrm{u}_i)$$

and define $v_i = \mathrm{X}\mathrm{u}_i$ an eigenvector for $\mathrm{X}\mathrm{X}^\top \in \mathbb{R}^{n \times n}$, then we can perform the eigendecomposition in $O(n^3)$ time rather than $O(d^3)$. To determine the eigenvectors, again premultipy the above by $\mathrm{X}^\top$ to give:

$$\frac{1}{n}(\mathrm{X}^\top\mathrm{X})(\mathrm{X}^\top \mathrm{v}_i) = \lambda_i(\mathrm{X}^\top \mathrm{v}_i)$$

and notice that $\mathrm{X}^\top \mathrm{v}_i$ is an eigenvector for C with eigenvalue λ_i. The only drawback is that $\mathrm{X}^\top \mathrm{v}_i$ may not be unit norm, so normalize. It is remarkable that to calculate PCA in this case, only the inner product

was used. This sets the foundation for kernel PCA algorithms, which operate in specialized inner product spaces.

6.2 Prior Assumption of Sparsity

Compressed sensing is a form of dimensionality reduction with the assumption that the data is sparse in some well-chosen basis. This is often the case, in fact, JPEG-2000 compression uses the assumption that images are sparse in a wavelet-basis [4]

References

[1] Bengio, Yoshua and Courville, Aaron and Vincent, Pascal, "Representation learning: A review and new perspectives",*Cambridge University Press*, 2006.

[2] Bishop, Christopher M. ¡i¿Pattern Recognition and Machine Learning¡/i¿. New York: Springer, 2006. Print.

[3] Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar. "Foundations of Machine Learning." MIT Press, 2012

[4] Shalev-Shwartz, Shai, and Shai Ben-David. Understanding Machine Learning. New York, NY: Cambridge UP, 2014. Print.