Randomized Algorithms for Eigenvector Computation

Krishna Balasubramanian

March 31, 2020

1 Introduction

In this notes, we will study two (randomized) algorithm for computing eigenvectors. Eigenvalues and Eigenvectors are uniformly popular across almost all disciplines from probability, statistics, applied mathematics, computer science, etc. because of their wide applicability, for both theoretical and practical purposes. For simplicity, we will focus only on positive semidefinite matrices in this notes. Recall that a square matrix $\mathbf{A} \in \mathbb{R}^d$ is positive semidefinite, if $\forall\ \mathbf{b} \in \mathbb{R}^d$, we have the quadratic form satisfy $\mathbf{b}^\top \mathbf{A}\mathbf{b} \geq 0$. Furthermore, recall that the Frobenius norm could also be represented using the trace as follows:

$$\|\mathbf{A}\|_F^2 = \text{TR}(\mathbf{A}\mathbf{A}^\top).$$

2 Power Method

Power method is one of the oldest method to compute the eigenvectors. Let $\mathbf{A} \in \mathbb{R}^{d\times d}$ be a **deterministic** positive semi-definite matrix. That means that we have the following decomposition

$$\mathbf{A} = \sum_{i=1}^{d} \lambda_i \mathbf{v}_i \mathbf{v}_i^\top$$

for some (unit-norm) eigenvectors $\mathbf{v}_i \in \mathbb{R}^d$ and eigenvalues λ_i.

Fact 2.0.1. Eigenvectors $\mathbf{v}_i$, $i = 1, \ldots, n$ form a basis for the Euclidean space $\mathbb{R}^d$. [prove!]

We assume that $\lambda_1 > \lambda_i$ for all $i = 2, \ldots, d$. Such an eigenvalue is then an **unique/isolated** top eigenvalue and the corresponding eigenvector $\mathbf{v}_1$ is called as the top eigenvector. Note that it is not guaranteed that every positive semi-definite matrix has such a top eigenvector – **this is an assumption!** The problem, we consider is given a matrix $\mathbf{A}$ that has the above structure, find a vector $\mathbf{c} \in \mathbb{R}^n$ that is an approximation to $\mathbf{v}_1$.

Algorithm 1 Power Method

Input: Unit-vector $\mathbf{c}^{(0)} \in \mathbb{R}^d$ (Randomly generated) and number of iterations T.
for $t = 1, \ldots, T$ **do**
  $\mathbf{c}^{(t)} = \mathbf{A}\mathbf{c}^{(t-1)}$
  $\mathbf{c}^{(t)} = \frac{\mathbf{c}^{(t)}}{\|\mathbf{c}^{(t)}\|_2}$ (Normalization step)
end for

We also note the following alternative way of calculating eigenvector:

$$\begin{aligned}\mathbf{v}_1 &= \underset{\mathbf{v} \text{ SUCH THAT } \|\mathbf{v}\|_2=1}{\arg\min} \left\{\|\mathbf{A} - \lambda_1 \mathbf{v}\mathbf{v}^\top\|_F^2 = \|\mathbf{A}\|_F^2 + \|\lambda_1 \mathbf{v}\mathbf{v}^\top\|_F^2 - 2\lambda_1 \mathrm{TR}(\mathbf{A}\mathbf{v}\mathbf{v}^\top)\right\} \\ &= \underset{\mathbf{v} \text{ SUCH THAT } \|\mathbf{v}\|_2=1}{\arg\min} \|\mathbf{A}\|_F^2 + \|\lambda_1 \mathbf{v}\mathbf{v}^\top\|_F^2 - 2\lambda_1 \mathrm{TR}(\mathbf{v}^\top \mathbf{A}\mathbf{v}) \\ &= \underset{\mathbf{v} \text{ SUCH THAT } \|\mathbf{v}\|_2=1}{\arg\max} \mathrm{TR}(\mathbf{v}^\top \mathbf{A}\mathbf{v})\end{aligned}$$

The power method is described in Algorithm 1. Note that, if we ignore the normalization step, we have $\mathbf{c}^{(t)} = \mathbf{A}^t \mathbf{c}^{(0)}$. A consequence of the Fact 2.0.1 is that any vector could be written as a linear combination of the d eigenvectors $\mathbf{v}_i$. Specifically, the random vector $\mathbf{c}^{(0)}$ given as input to Algorithm 1 could be written as follows, for some constants $\gamma_1, \ldots, \gamma_d$.

$$\mathbf{c}^{(0)} = \gamma_1 \mathbf{v}_1 + \gamma_2 \mathbf{v}_2 + \ldots + \gamma_d \mathbf{v}_d$$

Now, look at the iterates of Algorithm 1, without normalization step. We then have

$$\begin{aligned}\mathbf{c}^{(1)} = \mathbf{A}\mathbf{c}^{(0)} &= \gamma_1 \mathbf{A}\mathbf{v}_1 + \gamma_2 \mathbf{A}\mathbf{v}_2 + \ldots + \gamma_d \mathbf{A}\mathbf{v}_d \\ &= \gamma_1 \lambda_1 \mathbf{v}_1 + \gamma_2 \lambda_2 \mathbf{v}_2 + \ldots + \gamma_d \lambda_d \mathbf{v}_d\end{aligned}$$

Similarly, we have

$$\begin{aligned}\mathbf{c}^{(2)} = \mathbf{A}\mathbf{c}^{(1)} &= \gamma_1 \lambda_1 \mathbf{A}\mathbf{v}_1 + \gamma_2 \lambda_2 \mathbf{A}\mathbf{v}_2 + \ldots + \gamma_d \lambda_d \mathbf{A}\mathbf{v}_d \\ &= \gamma_1 \lambda_1^2 \mathbf{v}_1 + \gamma_2 \lambda_2^2 \mathbf{v}_2 + \ldots + \gamma_d \lambda_d^2 \mathbf{v}_d.\end{aligned}$$

For the general case, we have

$$\mathbf{c}^{(t+1)} = \mathbf{A}c^{(t)} = \gamma_1 \lambda_1^{(t+1)} \mathbf{v}_1 + \gamma_2 \lambda_2^{(t+1)} \mathbf{v}_2 + \ldots + \gamma_d \lambda_d^{(t+1)} \mathbf{v}_d.$$

Recall that, we have assumed λ_1 is larger than other other eigenvalues. So, when it is raised to the power $(t+1)$, the vector $\mathbf{c}^{(t+1)}$ is most correlated (or aligned) in the direction of $\mathbf{v}_1$. But if λ_2 is very close to λ_1, then t might have to be really large to get $\mathbf{c}^{(t+1)}$ correlated with

Algorithm 2 Truncated Power Method

Input: Unit-vector $\mathbf{c}^{(0)} \in \mathbb{R}^d$ (Randomly generated) and number of iterations T.
for $t = 1, \ldots, T$ **do**
$\mathbf{c}^{(t)} = \mathbf{A}\mathbf{c}^{(t-1)}$
$\mathbf{c}^{(t)} = \frac{\mathbf{c}^{(t)}}{\|\mathbf{c}^{(t)}\|_2}$ (Normalization step)
Let $F^{(t)} \subset \{1, \ldots, d\}$ be the set of co-ordinates of $\mathbf{c}^{(t)}$ corresponding to top k absolute values.
Keep the values of $\mathbf{c}^{(t)}$ in the co-ordinates index by the set $F^{(t)}$ as such and set the rest of the coordinates to zero and re-normalize.
end for

$\mathbf{v}_1$. Now, suppose instead of normalizing with the $\|\mathbf{c}^{(t)}\|_2$, we normalize with λ_1. [Note: λ_1 is typically not known and might be hard to calculate. This is done, just for understanding purpose.] In this case, we have

$$\begin{aligned}\mathbf{c}^{(t+1)} &= \gamma_1 \left(\frac{\lambda_1}{\lambda_1}\right)^{(t+1)} \mathbf{v}_1 + \gamma_2 \left(\frac{\lambda_2}{\lambda_1}\right)^{(t+1)} \mathbf{v}_2 + \ldots + \gamma_n \left(\frac{\lambda_n}{\lambda_1}\right)^{(t+1)} \mathbf{v}_n \\ &= \gamma_1 \mathbf{v}_1 + \gamma_2 \left(\frac{\lambda_2}{\lambda_1}\right)^{(t+1)} \mathbf{v}_2 + \ldots + \gamma_n \left(\frac{\lambda_n}{\lambda_1}\right)^{(t+1)} \mathbf{v}_n\end{aligned}$$

Note that, in this case, $\mathbf{c}^{(t+1)}$ will be correlated with $\mathbf{v}_1$ even within a few iterations (small value of t). But obviously, we might not know λ_1. A compromise is to estimate just to direction of eigenvector $\mathbf{v}_1$. So, we assume that $\mathbf{v}_1$ is an unit vector, all the magnitude information is absorbed into the eigenvalue λ_1. Thus, in Algorithm 1, we just normalize with the magnitude of $\mathbf{c}$ vectors to estimate the direction.

In some applications, the eigenvector might by a sparse vector, i.e., only k of the d co-ordinates are non-zero. The rest of the co-ordinates are zero. The above power method in Algorithm 1 could then be easily modified to this situation, with a simple truncation argument in each step. The detailed algorithm is given in Algorithm 2.
For both the power method and truncated power method, the difference between λ_1 and λ_2 determines the rate of convergence. Furthermore, its crucial how the initialization vector is generated.

3 Statistical Application: PCA

Principal component analysis (PCA) is arguably the most popular dimensionality reduction technique in statistics. In linear PCA, we try to represent each data sample as linear

combination of some fixed basis vector. That is[1],

$$X^{(i)} = \sum_{j=1}^{p} \mathbf{v}_j \beta_j^{(i)} = \mathbf{V}\beta^{(i)}$$

where $\mathbf{V} \in \mathbb{R}^{d\times p}$ is the matrix with the basis vectors $\mathbf{v}_j$ as its columns and $\beta_j^{(i)}$ denote the j^{th} coordinate of the vector $\beta^{(i)}$. Note that the basis vectors are fixed and does not change with i and the coefficients change with i. The reasoning behind this is that, you want to represent each sample as a different linear combination of the same basis vector. For example, if the $X^{(i)}$ represent human faces, then $\mathbf{v}_j$ represents "template" human faces. Then PCA posits that every human face is indeed some linear combination of some template faces. This reasoning is motivated by empirical studies in human face anatomy. Furthermore, this reasoning also appears to be true for several different fields, which made PCA widely applicable in practice.

In the PCA model above, note that without any other assumptions on $\mathbf{V}$, the model is not unique. Hence we assume that the matrix $\mathbf{V}$ is such that $\mathbf{V}^\top\mathbf{V} = I$. That is, the basis vectors $\mathbf{v}_j$ are orthogonal. Given this, PCA boils down to the to task of estimating the vector $\{\hat{\beta}^{(i)}\}_{i=1}^n$ and the matrix $\hat{\mathbf{V}}$ that minimizes certain reconstruction error, given the data $\{X^{(i)}\}_{i=1}^n \in \mathbb{R}^d$. That is ,we have

$$\{\hat{\beta}^{(i)}\}_{i=1}^n, \hat{\mathbf{V}} = \underset{\substack{\mathbf{V} \text{ such that } \mathbf{V}^\top\mathbf{V}=I \\ \{\beta^{(i)}\}_{i=1}^n \in \mathbb{R}^p}}{\arg\min} \sum_{i=1}^{n} \|X^{(i)} - \mathbf{V}\beta^{(i)}\|_2^2$$

The above problem is a non-convex problem. Fortunately, one can use Eigenvalue decomposition to get the global solution. In order to see that, first assume that we want to find $\hat{\beta}^{(i)}$ for each i, given the optimal basis matrix $\hat{\mathbf{V}}$. In order to do that, we have

$$\hat{\beta}^{(i)} = \arg\min_{\beta} \|X^{(i)} - \hat{\mathbf{V}}\beta^{(i)}\|_2^2$$

for all $i = 1, \ldots, n$. This is because the objective function for finding $\hat{\beta}^{(i)}$ does not depend on any other terms involving $\hat{\beta}^{(k)}$, where $k \neq i$. Note that the above problem is a linear least-squares problem and so we have

$$\hat{\beta}^{(i)} = (\hat{\mathbf{V}}^\top\hat{\mathbf{V}})^{-1}\hat{\mathbf{V}}^\top X^{(i)} = \hat{\mathbf{V}}^\top X^{(i)},$$

as the optimal basis matrix is orthogonal. The main message here is that the optimal $\beta^{(i)}$ is going to have form $\hat{\mathbf{V}}^\top X^{(i)}$ for each i, if we know $\hat{\mathbf{V}}$. But we don't know $\hat{\mathbf{V}}$. In order to find $\hat{\mathbf{V}}$, substitute $\beta^{(i)} = \mathbf{V}^\top X^{(i)}$ in the original objective function to get:

$$\hat{\mathbf{V}} = \underset{\mathbf{V} \text{ such that } \mathbf{V}^\top V=I}{\arg\min} \sum_{i=1}^{n} \|X^{(i)} - \mathbf{V}\mathbf{V}^\top X^{(i)}\|_2^2$$

[1]Note that in the previous section, we used the superscript (t), to denote the number of iteration of the power method. Now, we use it to denote the samples.

The above problem could be expanded out as

$$\begin{aligned}
\hat{\mathbf{V}} = & \underset{\mathbf{V} \text{ such that } \mathbf{V}^\top V = I}{\arg\min} \quad \sum_{i=1}^{n} \left({X^{(i)}}^\top X^{(i)} - {X^{(i)}}^\top \mathbf{V}\mathbf{V}^\top X^{(i)} \right) \\
& \underset{\mathbf{V} \text{ such that } \mathbf{V}^\top V = I}{\arg\min} \quad \sum_{i=1}^{n} \left({X^{(i)}}^\top X^{(i)} - \text{trace}\left({X^{(i)}}^\top \mathbf{V}\mathbf{V}^\top X^{(i)} \right) \right)
\end{aligned}$$

Note that the above minimization problem is equivalent to the following maximization problem:

$$\begin{aligned}
\hat{\mathbf{V}} = & \underset{\mathbf{V} \text{ such that } \mathbf{V}^\top V = I}{\arg\max} \quad \sum_{i=1}^{n} \left(\text{trace}\left({X^{(i)}}^\top \mathbf{V}\mathbf{V}^\top X^{(i)} \right) \right) \\
& \underset{\mathbf{V} \text{ such that } \mathbf{V}^\top V = I}{\arg\max} \quad \sum_{i=1}^{n} \left(\text{trace}\left(X^{(i)} {X^{(i)}}^\top \mathbf{V}\mathbf{V}^\top \right) \right) \\
& \underset{\mathbf{V} \text{ such that } \mathbf{V}^\top V = I}{\arg\max} \quad \sum_{i=1}^{n} \left(\text{trace}\left(\mathbf{V}^\top X^{(i)} {X^{(i)}}^\top \mathbf{V} \right) \right) \\
& \underset{\mathbf{V} \text{ such that } \mathbf{V}^\top V = I}{\arg\max} \quad \left(n \cdot \text{trace}\left(\mathbf{V}^\top \left(\frac{1}{n} \sum_{i=1}^{n} X^{(i)} {X^{(i)}}^\top \right) \mathbf{V} \right) \right) \\
& \underset{\mathbf{V} \text{ such that } \mathbf{V}^\top V = I}{\arg\max} \quad \text{trace}\left(\mathbf{V}^\top \left(\hat{\Sigma}_n \right) \mathbf{V} \right)
\end{aligned}$$

In the above steps, we have used two properties of trace: (i) $\text{trace}(A + B) = \text{trace}(A) + \text{trace}(B)$ and (ii) $\text{trace}(ABC) = \text{trace}(CAB) = \text{trace}(BCA)$. Also, in the last step, we dropped n as it plays no part in the maximization. The above problem involves the sample covariance matrix $\hat{\Sigma}_n = \frac{1}{n} \sum_{i=1}^{n} X^{(i)} {X^{(i)}}^\top$.

3.1 Top Principal Component

Lets try to interpret the above problem. Assume that $p = 1$. Hence $\mathbf{V} = \mathbf{v}_1 \in \mathbb{R}^d$. In this case, the above problem becomes

$$\hat{\mathbf{v}}_1 = \underset{\mathbf{v} \text{ such that } \mathbf{v}^\top \mathbf{v} = 1}{\arg\max} \quad \text{trace}\left(\mathbf{v}^\top \left(\hat{\Sigma}_n \right) \mathbf{v} \right) = \underset{\mathbf{v} \in \mathbb{R}^\mathbf{d} \text{ such that } \mathbf{v}^\top \mathbf{v} = 1}{\arg\max} \quad \mathbf{v}^\top \left(\hat{\Sigma}_n \right) \mathbf{v}$$

We see that $\hat{\mathbf{v}}_1$ is nothing but the top eigenvector of the sample covariance matrix. We can use the power method in Algorithm 1 to calculate them. Specifically, we take $\mathbf{A} = \hat{\Sigma}_n$.

The above discussion of linear PCA implicitly that $n > d$. Only then, the sample covariance matrix will convergence to the true covariance matrix and as a consequence, the top eigenvectors of the sample covariance matrix will convergence correspondingly to the top

eigenvectors of the true covariance matrix. When $n < d$, we see that the sample covariance matrix is not full-rank. Hence it will not converge to the true covariance matrix and as a consequence the sample eigenvectors won't converge to the truth. So we enforce sparsity constraint on the $\mathbf{v}_1$ vector and use Algorithm 2 instead. Specifically, we have the following problem:

$$\hat{\mathbf{v}}_1 = \underset{\mathbf{v_1} \in \mathbb{R}^{\mathbf{d}} \text{ such that } \mathbf{v}_1^\top \mathbf{v}_1 = 1 \text{ and } \|\mathbf{v}_1\|_0 = s}{\arg\max} \mathbf{v}_1^\top \left(\hat{\Sigma}_n \right) \mathbf{v}_1$$

Here the constraint $\|\mathbf{v}_1\|_0 = s$ ensures that there are only s non-zero entries in the d dimensional vector $\mathbf{v}_1$. This is a NP-hard combinatorial optimization problem (a.k.a *bad* problem) and takes exponential time to be solved. It turns out that, if one is willing to spend the exponential time and solve the above problem, then $d > n > s \log(d)$ samples are required for the sample eigenvector to converge to the true eigenvector. But if we want an efficient algorithm (that is an algorithm that runs in polynomial time rather than exponential time), then any such algorithm **must** require $d > n > s^2 \log(d)$ to work. The truncated power method, outlined in 2 with $\mathbf{A} = \hat{\Sigma}_n$ achieves this sample complexity requirement. The intriguing phenomenon with sparse PCA is even if you use other *complicated algorithms*, they too require the same number of samples $n > s^2 \log(d)$ to work. This is often times the case in several problems.

Coming back to regular (non-sparse) PCA, recall that we use Algorithm 1 with $\mathbf{A} = \hat{\Sigma}_n$. In each iteration, we are re-using the set of n samples again and again. Can we do better ? It turns out that, we can formulate an algorithm that uses only 1 sample at a time. Hence, in this case, the number of iterations of the algorithm and the number of samples used by the algorithm becomes the same!

In order to provide the algorithm, first note that the population version of the PCA problem is given by

$$\mathbf{v}_1 = \underset{\mathbf{v} \in \mathbb{R}^{\mathbf{d}} \text{ such that } \mathbf{v}^\top \mathbf{v} = 1}{\arg\max} \mathbf{v}^\top (\Sigma) \mathbf{v} = \underset{\mathbf{v} \in \mathbb{R}^{\mathbf{d}} \text{ such that } \mathbf{v}^\top \mathbf{v} = 1}{\arg\max} f(\Sigma, \mathbf{v})$$

where $\Sigma = \mathbb{E}[XX^\top] = \int_{\mathbb{R}^d} XX^\top P(X)$. In practice, we don't observe $P(X)$ but we will be able to sample from $P(X)$. Furthermore, note that $\nabla_{\mathbf{v}} f(\Sigma, \mathbf{v}) = \Sigma \mathbf{v} \in \mathbb{R}^d$. The algorithm is then based on calculating the gradient using one sample in each iteration and is provided in Algorithm 3. Note that the algorithm is very fast to implement and uses only 1 sample in each iteration as opposed to the Algorithm 1 with $\mathbf{A} = \hat{\Sigma}_n$.

Finally, although we have not provided rates of convergence of the above algorithms, rigorous rates have been established in the recent past for these algorithms.

Algorithm 3 Stochastic/Online Algorithm for PCA

Input: Unit-vector $\mathbf{v}^{(0)} \in \mathbb{R}^d$ (Randomly generated) and step-size β.
for $t = 1, \ldots, T$ **do**
  Sample $X^{(t)}$ from the distribution P(X).
  Let $\mathbf{v}^{(t)} = \mathbf{v}^{(t-1)} + \beta X^{(t)} {X^{(t)}}^\top \mathbf{v}^{(t-1)}$
  $\mathbf{v}^{(t)} = \frac{\mathbf{v}^{(t)}}{\|\mathbf{v}^{(t)}\|_2}$
end for