LINEAR ALGEBRA: ARRAY INVERSE

i) Could do it with factors and determinant but it's slow.

ii) Solve a linear system instead:

\[ A \mathbf{x} = \mathbf{1} \Rightarrow \mathbf{x} = A^{-1} \]

Use the methods we saw already, to solve \( n \) linear systems, where \( n \) is the number of columns in \( \mathbf{x} \). Implemented in numpy finally as \( \mathbf{x} = \text{inv}(A) \).

---

EIGENVALUES AND EIGENVECTORS

i) For symmetric \( A \):

\[ A \mathbf{v} = \lambda \mathbf{v} \rightarrow \text{eigenvector} \]

\[ \downarrow \text{eigenvalue} \]

If \( A \) is \( N \times N \), then we have \( N \) eigenvalues and \( N \) eigenvectors. Eigenvectors are orthogonal:

\[ \mathbf{v}_i \cdot \mathbf{v}_j = 0 \]

We also normalize them:

\[ \mathbf{v}_i \cdot \mathbf{v}_i = \| \mathbf{v}_i \|^2 = 1 \]

ii) Consider all the eigenvectors and put them into an array \( \mathbf{V} \)

\[ \mathbf{V} = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{pmatrix} \]

Note that \( \mathbf{V} \) is by definition an orthogonal matrix.
\[ \to \quad V^TV = 1 \]

Now if we also build the diagonal array of eigenvectors \( D = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{pmatrix} \), we can simultaneously get write the eigenvector equations for all \( N \) eigenvectors as:

\[ AV = VD, \quad \text{or equivalently} \quad V^TAV = D \]

We search for an efficient algorithm to compute \( V \) and \( D \).

This is based on the \( QR \) decomposition; write \( A \) as:

\[ A = Q'R' \rightarrow \text{upper triangular} \]

\[ \text{orthogonal} \]

It is possible to show (deterministic) that this is possible for any \( A \). In (iii) we see how to use this decomposition recursively to get \( V \) and \( D \) for any \( A \). In (iii) we see how to use this decomposition recursively to get \( V \) and \( D \) for any \( A \).

(iii) Assume we have found \( A = QR \), now build:

1) \[ A_1 = R_1A_1 \quad \text{(that is, "invert" the order of } Q_1 \text{ and } R_1) \]

It is immediate to see that \( R_1 = Q_1^TA \) (because \( Q_1^TQ_1 = I \))

Traceless:

\[ A_1 = R_1Q_1 = Q_1^TAQ_1 \]

2) Now build the \( QR \) decomposition of \( A_1 \):

\[ A_2 = R_2Q_2 \]

and swap again:

\[ A_3 = R_3Q_3 \]

We see that:

\[ R_2 = Q_2^TA_1 = Q_2^TQ_1^TAQ_1 \]

Now replace \( R_2 = Q_2^TQ_1^TAQ_1 \) in \( A_3 = R_2Q_3 \) to get:

\[ A_3 = Q_2^TQ_1^TAQ_1Q_3 \]
3) Go on recursively. At step $k$ you build $A_k = Q_k P_k$ and you know that

$$A_k = (Q_k^T Q_{k-1}^T \ldots Q_1^T) A (Q_1 \ldots Q_k)$$

Then $A_{k+1} = R_k A_k$ and so on...

Note that the product of orthogonal matrices is also orthogonal. Therefore $Q_1 \ldots Q_k$ is an orthogonal matrix

4) It can be shown that, for $k$ large enough, $\prod_{i=1}^k Q_i$ converges to $V$, where $V$ is the eigenvector matrix. Therefore:

$$\lim_{k \to \infty} \prod_{i=1}^k Q_i = V$$

$$\lim_{k \to \infty} \left( \prod_{i=1}^k Q_i^T \right) A \left( \prod_{i=1}^k Q_i \right) = \lim_{k \to \infty} A_k = W^T V A V = D$$

The final algorithm to get $V$ and $D$, for every $A$, is:

- a) Initialize $V$ to $I$
- b) Decompose $A = Q R$
- c) Swap to get $A_r = R Q$
- d) Set $A_i = V$ and compute $D = V^T A V$.
- e) If $D$ is diagonal (in the sense that off-diagonal elements are smaller than some $E$), output $D$ and $V$. Otherwise, go back to step b), finding the QR decomposition of $A_i$.

The missing step is showing how to build a QR decomposition of a given array (symmetric) $A$.

Start by thinking of $A$ as a sequence of $N$ column vectors of length $N$. —
\[ A = \begin{pmatrix} \hat{a}_0 & \hat{a}_1 & \hat{a}_2 & \cdots & \hat{a}_n \end{pmatrix} \quad (A \text{ is a } N \times N \text{ symmetric array}) \]

We want to build an orthonormal basis, using the vectors \( \{\hat{a}_0, \cdots, \hat{a}_n\} \). We can do this via a Gram-Schmidt orthonormalization procedure.

If we call \( \{\hat{\varphi}_0, \cdots, \hat{\varphi}_n\} \) the final basis, we can build it like this:

\[
\begin{align*}
\hat{\varphi}_0 &= \frac{\hat{a}_0}{\|\hat{a}_0\|} \\
\hat{\varphi}_1 &= \frac{\hat{a}_1 - (\hat{a}_1 \cdot \hat{\varphi}_0) \hat{\varphi}_0}{\| \hat{a}_1 - (\hat{a}_1 \cdot \hat{\varphi}_0) \hat{\varphi}_0 \|} \\
\hat{\varphi}_2 &= \frac{\hat{a}_2 - (\hat{a}_2 \cdot \hat{\varphi}_0) \hat{\varphi}_0 - (\hat{a}_2 \cdot \hat{\varphi}_1) \hat{\varphi}_1}{\| \hat{a}_2 - (\hat{a}_2 \cdot \hat{\varphi}_0) \hat{\varphi}_0 - (\hat{a}_2 \cdot \hat{\varphi}_1) \hat{\varphi}_1 \|} \\
&\quad \vdots \\
\hat{\varphi}_i &= \frac{\hat{a}_i - \sum_{k=0}^{i-1} (\hat{a}_i \cdot \hat{\varphi}_k) \hat{\varphi}_k}{\| \hat{a}_i - \sum_{k=0}^{i-1} (\hat{a}_i \cdot \hat{\varphi}_k) \hat{\varphi}_k \|}
\end{align*}
\]

The idea is that we project each vector on the previous basis vectors and remove the components along those directions to orthonormalize. We then divide by the modulus to orth normalize. Graphically:

1) Start
\[ \hat{\varphi}_1 = \frac{\hat{a}_1}{\|\hat{a}_1\|}, \quad \hat{\varphi}_0 = \hat{a}_0 \]

2) Since \( \{\hat{\varphi}_0, \cdots, \hat{\varphi}_n\} \) is a basis, we re-expand \( \hat{a}_0, \cdots, \hat{a}_n \) in this basis to get (invert the Gram-Schmidt above)

\[
\begin{align*}
\hat{a}_0 &= |\hat{\varphi}_0| \hat{\varphi}_0 \\
\hat{a}_1 &= |\hat{\varphi}_1| \hat{\varphi}_1 + (\hat{\varphi}_0 \cdot \hat{a}_1) \hat{\varphi}_0 \\
\hat{a}_2 &= |\hat{\varphi}_2| \hat{\varphi}_2 + (\hat{\varphi}_0 \cdot \hat{a}_2) \hat{\varphi}_0 + (\hat{\varphi}_1 \cdot \hat{a}_2) \hat{\varphi}_1 \\
&\quad \vdots
\end{align*}
\]

We can write this last set of equations in matrix form:

\[ A = \begin{pmatrix} a_0 & a_1 & \ldots & a_n \end{pmatrix} = \begin{pmatrix} \vec{a}_0 & \vec{a}_1 & \ldots & \vec{a}_n \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ \mathbf{R} \end{pmatrix} \]

\[ \mathbf{A} \text{ orthogonal} \]

\[ \mathbf{R} \text{ upper triangular} \]

This is the QR decomposition we looked for.