LINEAR ALGEBRA: ARRAY INVERSE

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

ii) Solve a linear system instead:

$$A X = \mathbb{1} \Rightarrow X = A^{-1}$$

Use the methods we saw already, to solve n linear systems, where n is the number of columns in X. Implemented in numpy linalg as $X = \mathrm{inv}(A)$.

~~Eigen~~

EIGENVALUES AND EIGENVECTORS

i) For symmetric A:

$$A\vec{V} = \underbrace{\lambda}_{\text{eigenvalue}} \vec{V} \rightarrow \text{eigenvector}$$

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

$$\vec{V}_i \cdot \vec{V}_j = 0$$

We also normalize them: $\vec{V}_i \cdot \vec{V}_i = \| V_i \|^2 = 1$

ii) Consider all the eigenvectors and put them into an array V

$$V \equiv \begin{pmatrix} \vdots & \vdots & & \vdots \\ \vec{V}_1 & \vec{V}_2 & \cdots & \vec{V}_N \\ \vdots & \vdots & & \vdots \end{pmatrix}$$

Note that V is by definition an orthogonal matrix:

$\rightarrow \quad V^T V = \mathbb{1}$

Now if we also build the diagonal array of eigenvectors $D \equiv \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{pmatrix}$, we can simultaneously ~~get~~ write the eigenvector equations, for all N eigenvectors, as:

$$\boxed{AV = VD}, \text{ or, equivalently, } \boxed{V^T A V = D}$$

We search for an efficient algorithm to compute V and D. This is based on the QR decomposition; write A as

$$A = \underbrace{Q}_{\text{orthogonal}} \; R \rightarrow \text{upper triangular.}$$

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

iii) Assume we found $\boxed{A = Q_1 R_1}$. Now build:

1) $\boxed{A_1 = R_1 Q_1}$ (that is, "invert" the order of Q_1 and R_1)

It is immediate to see that $R_1 = Q_1^T A$ (because $Q_1^T Q_1 = \mathbb{1}$)

Therefore:

$$A_1 = R_1 Q_1 = Q_1^T A Q_1$$

2) Now build the QR decomposition of A_1: $\boxed{A_1 = Q_2 R_2}$

and swap again: $\boxed{A_2 = R_2 Q_2}$

We see that:

$$R_2 = Q_2^T A_1 = Q_2^T Q_1^T A Q_1$$

Now replace $R_2 = Q_2^T Q_1^T A Q_1$ in $A_3 = R_2 Q_2$ to get

$$A_2 = Q_2^T Q_1^T A Q_1 Q_2$$

3) Go on recursively. At step k you build $A_k = Q_k R_k$ and you know that

$$A_k = \left(Q_k^T Q_{k-1}^T \dots Q_1^T\right) A \left(Q_1 Q_2 \dots Q_k\right)$$

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

Note that the product of orthogonal matrices is also orthogonal. Therefore $Q_1 \dots Q_k$ 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:

$$\begin{cases} \lim_{k\to\infty} \prod_{i=1}^{k} Q_i = V \\ \lim_{k\to\infty} \left(\prod_{i=1}^{k} Q_i^T\right) A \left(\prod_{i=1}^{k} Q_i\right) \equiv \lim_{k\to\infty} A_k = \cancel{V} V^T A V = D \end{cases}$$

The final algorithm to get V and D_k, for array A, is

a) Initialize V to $\mathbb{1}$

b) Decompose $A = Q_1 R_1$

c) Swap to get $A_1 = R_1 Q_1$

d) Set $A_1 = V$ and compute $D = V^T A V$.

e) If D is diagonal (in the sense that off-diagonal elements are smaller than some ε), output D and ~~[illegible]~~ V. Otherwise, go back to step b), finding the QR decomposition of A_1.

iv) 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 vector of length N: →

$\rightarrow \quad A = \begin{pmatrix} \vdots & \vdots & \vdots & & \vdots \\ \vec{a}_0 & \vec{a}_1 & \vec{a}_2 & \cdots & \vec{a}_N \\ \vdots & \vdots & \vdots & & \vdots \end{pmatrix}$ (A is a N×N symmetic array)

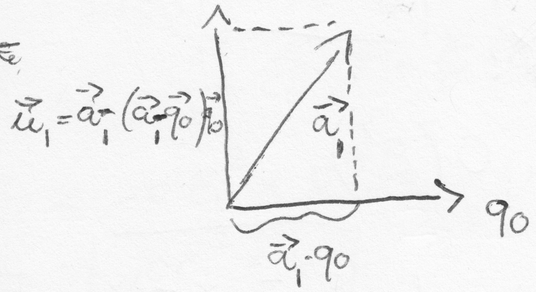
•) We want to build an orthonormal basis, using the vectors $\{\vec{a}_0 \dots \vec{a}_N\}$. We can do this via Gram-Schmidt orthonormalization procedure,

If we call $\{\vec{q}_0, \dots, \vec{q}_N\}$ the final basis, we can build it like this:

$$\begin{cases} \vec{u}_0 = \vec{a}_0 \, , & \vec{q}_0 = \dfrac{\vec{u}_0}{|\vec{u}_0|} \\ \vec{u}_1 = \vec{a}_1 - (\vec{a}_1 \cdot \vec{q}_0)\vec{q}_0 \, , & \vec{q}_1 = \dfrac{\vec{u}_1}{|\vec{u}_1|} \\ \vec{u}_2 = \vec{a}_2 - (\vec{a}_2 \cdot \vec{q}_0)\vec{q}_0 - (\vec{a}_2 \cdot \vec{q}_1)\vec{q}_1 \, , & \vec{q}_2 = \dfrac{\vec{u}_2}{|\vec{u}_2|} \\ \vdots & \\ \boxed{\vec{u}_i = \vec{a}_i - \sum_{k=0}^{i-1} (\vec{a}_i \cdot \vec{q}_k)\vec{q}_k \, , \quad \vec{q}_i = \dfrac{\vec{u}_i}{|\vec{u}_i|}} & \end{cases}$$

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

-) ~~Since~~



•) Since $\{\vec{q}_0 \cdots \vec{q}_N\}$ is a basis, we re-expand $\{\vec{a}_0, \dots, \vec{a}_N\}$ in this basis to get (invert the Gram-Schmidt above)

$$\begin{cases} \vec{a}_0 = |u_0| \, \vec{q}_0 \\ \vec{a}_1 = |\vec{u}_1| \vec{q}_1 + (\vec{q}_0 \cdot \vec{a}_1)\vec{q}_0 \\ \vdots \\ \vec{a}_i = |\vec{u}_i| \vec{q}_i + \sum_{k=1}^{i-1} (\vec{q}_{i-1} \cdot \vec{a}_i)\vec{q}_{i-1} \end{cases}$$

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

$$A = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \vec{a}_0 & \vec{a}_1 & \cdots & \vec{a}_N \\ \vdots & \vdots & & \vdots \end{pmatrix} = \underbrace{\overbrace{\begin{pmatrix} \vdots & \vdots & & \vdots \\ \vec{q}_0 & \vec{q}_1 & \cdots & \vec{q}_N \\ \vdots & \vdots & & \vdots \end{pmatrix}}^{Q}}_{\text{orthogonal}} \underbrace{\overbrace{\begin{pmatrix} |\vec{u}_0| & \vec{q}_0 \cdot \vec{a}_1 & \vec{q}_0 \cdot \vec{a}_2 & \cdots & \vec{q}_0 \cdot \vec{a}_N \\ 0 & |\vec{u}_1| & \vec{q}_1 \cdot \vec{a}_2 & \cdots & \vec{q}_1 \cdot \vec{a}_N \\ 0 & 0 & |\vec{u}_2| & \cdots & \vec{q}_2 \cdot \vec{a}_N \\ & & & \ddots & \vdots \\ & & & & |\vec{u}_N| \end{pmatrix}}^{R}}_{\text{upper triangular}}$$

This is the QR decomposition we looked for.