Math 6108 Homework 8

Jacob Hauck

October 25, 2024

Question 1.

Let $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, and let $f(\lambda) = \lambda^2 - (a+d)\lambda + ad - bc = 0$ be the characteristic polynomial of A. Then

$$\begin{aligned} f(A) = A^2 - (a+d)A + (ad-bc)I &= \begin{bmatrix} a^2+bc & (a+d)b \\ (a+d)c & d^2+bc \end{bmatrix} - (a+d)\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} ad-bc & 0 \\ 0 & ad-bc \end{bmatrix} \\ &= \begin{bmatrix} a^2+bc-a(a+d)+ad-bc & (a+d)b-(a+d)b \\ (a+d)c-(a+d)c & d^2+bc-(a+d)d+ad-bc \end{bmatrix} \\ &= \begin{bmatrix} a^2+bc-a^2-ad+ad-bc & 0 \\ 0 & d^2+bc-ad-d^2+ad-bc \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \end{aligned}$$

Question 2.

Let $A \in \mathbb{R}^{n\times n}$ be invertible. Then $A^{-1} = g(A)$, where g is a polynomial over $\mathbb{R}$ of degree $n-1$.

Proof. Let f be the characteristic polynomial of A. Then $f(x) = a_n x^n + \cdots + a_0 I$ for some $a_0, a_1, \ldots a_n \in \mathbb{R}$. Suppose that $a_0 = 0$; then 0 is a root of f, which implies that 0 is an eigenvalue of A, which implies that $A\mathbf{v} = 0\mathbf{v} = \mathbf{0}$ for some $\mathbf{v} \neq \mathbf{0}$, contradicting the invertibility of A. Hence, $a_0 \neq 0$.

By the Cayley-Hamilton theorem, we must have

$$0 = a_n A^n + \cdots + a_0 I \implies I = -\frac{a_n}{a_0}A^n - \cdots - \frac{a_1}{a_0}A.$$

Multiplying by A^{-1} on both sides, we get

$$A^{-1} = -\frac{a_n}{a_0}A^{n-1} - \cdots - \frac{a_1}{a_0}I = g(A),$$

where $g(x) = -\frac{a_n}{a_n}x^{n-1} - \cdots - \frac{a_1}{a_0}$ is a polynomial of degree $n-1$. □

Question 3.

Let $A \in \mathbb{C}^{n\times n}$, and let f be a polynomial over $\mathbb{C}$. Then c is an eigenvalue of $f(A)$ if and only if $c = f(\lambda)$, where λ is an eigenvalue of A.

Proof. Suppose that $\lambda \in \mathbb{C}$ is an eigenvalue of A with corresponding eigenvector $\mathbf{v}$. There exists $f_0, f_1, \ldots, f_k \in \mathbb{C}$ such that $f(x) = f_k x^k + f_{k-1}x^{k-1} + \cdots + f_0$. Then

$$f(A)\mathbf{v} = f_k A^k \mathbf{v} + f_{k-1}A^{k-1}\mathbf{v} + \cdots + f_0\mathbf{v} = f_k\lambda^k\mathbf{v} + f_{k-1}\lambda^{k-1}\mathbf{v} + f_0\mathbf{v} = f(\lambda)\mathbf{v} = c\mathbf{v}.$$

This shows that c is an eigenvalue of $f(A)$.

Conversely, suppose that c is an eigenvalue of $f(A)$ with corresponding eigenvector $\mathbf{v}$. Let $A = UTU^*$ be the Schur decomposition of A, where U is unitary and T is upper-triangular, with diagonal entries equal to eigenvalues $\lambda_1, \dots, \lambda_n$ of A. Then

$$f(A)\mathbf{v} = c\mathbf{v} \implies f_k A^k \mathbf{v} + \cdots + f_0 \mathbf{v} = c\mathbf{v} \implies f_k UT^kU^*\mathbf{v} + f_{k-1}UT^{k-1}U^*\mathbf{v} + \cdots + f_0\mathbf{v} = c\mathbf{v}.$$

Let $\mathbf{w} = U^*\mathbf{v}$. Then $\mathbf{v} \neq 0 \implies \mathbf{w} \neq 0$ because U is unitary, and, multiplying by U^* on both sides of the last equation above, we have

$$f_k T^k \mathbf{w} + f_{k-1}T^{k-1}\mathbf{w} + \cdots + f_0\mathbf{w} = c\mathbf{w}.$$

We can show by induction that

$$T^j = \begin{bmatrix} \lambda_1^j & * & \dots & * \\ 0 & \lambda_2^j & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^j \end{bmatrix}, \qquad j = 1, \dots, k, \tag{1}$$

where $*$ is a placeholder for any complex number (the exact values are irrelevant to our proof). The base case T^1 follows from the Schur decomposition: the diagonal elements of $T = T^1$ are eigenvalues of A. Suppose that (1) holds for some $1 \leq j \leq k-1$. Then

$$T^{j+1} = T^jT = \begin{bmatrix} \lambda_1^j & * & \dots & * \\ 0 & \lambda_2^j & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^j \end{bmatrix} \begin{bmatrix} \lambda_1 & * & \dots & * \\ 0 & \lambda_2 & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} = \begin{bmatrix} \lambda_1^{j+1} & * & \dots & * \\ 0 & \lambda_2^{j+1} & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^{j+1} \end{bmatrix},$$

which completes the proof by induction. Thus, (1) holds for all $j = 1, 2, \dots, k$. Then we have

$$f_k \begin{bmatrix} \lambda_1^k & * & \dots & * \\ 0 & \lambda_2^k & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^k \end{bmatrix} \mathbf{w} + f_{k-1} \begin{bmatrix} \lambda_1^{k-1} & * & \dots & * \\ 0 & \lambda_2^{k-1} & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^{k-1} \end{bmatrix} \mathbf{w} + \cdots + f_0\mathbf{w} = c\mathbf{w}.$$

Since $\mathbf{w} \neq 0$, at least one component of $\mathbf{w} = (w_1, \dots, w_n)^T$ is nonzero. Let w_p be the last nonzero component of $\mathbf{w}$, so that $w_q = 0$ if $q > p$. Then looking at the pth component of the above equation gives

$$f_k\lambda_p^k w_p + f_{k-1}\lambda_p^{k-1}w_p + \cdots + f_0 w_p = cw_p.$$

Dividing both sides by $w_p \neq 0$ implies that $c = f(\lambda_p)$. □

Question 4.

Let $A = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & -1 & 1 \\ 1 & -1 & 0 & 1 \\ -1 & 1 & 1 & 0 \end{bmatrix}$.

To find an orthogonal matrix P such that $P^TAP = D$, where D is diagonal, we should find linearly independent unit eigenvectors of A. This matrix looks like it has some symmetries, so I think that I can guess the eigenvectors fairly easily.

My first guess is $\mathbf{v}_1^T = \begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix}$. We have

$$A\mathbf{v}_1 = \begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & -1 & 1\\ 1 & -1 & 0 & 1\\ -1 & 1 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\0\\0\\-1\end{bmatrix} = \begin{bmatrix}1\\0\\0\\-1\end{bmatrix} = 1\mathbf{v}_1,$$

so $\mathbf{v}_1$ is an eigenvector of A with eigenvalue $\lambda_1 = 1$. This suggests a second guess by permuting some of the rows and columns of A; let $\mathbf{v}_2^T = \begin{bmatrix}0 & 1 & -1 & 0\end{bmatrix}$. Then

$$A\mathbf{v}_2 = \begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & -1 & 1\\ 1 & -1 & 0 & 1\\ -1 & 1 & 1 & 0\end{bmatrix}\begin{bmatrix}0\\1\\-1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\-1\\0\end{bmatrix} = 1\mathbf{v}_2,$$

so $\mathbf{v}_2$ is an eigenvector of A with eigenvalue $\lambda_2 = 1$. To be orthogonal to the first two eigenvectors, let $\mathbf{v}_3^T = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$. Then

$$A\mathbf{v}_3 = \begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & -1 & 1\\ 1 & -1 & 0 & 1\\ -1 & 1 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\1\\1\\1\end{bmatrix} = 1\mathbf{v}_3,$$

so $\mathbf{v}_3$ is an eigenvector of A with eigenvalue $\lambda_3 = 1$. Lastly, I will guess $\mathbf{v}_4^T = \begin{bmatrix}-1 & 1 & 1 & -1\end{bmatrix}$ to be orthogonal to the first three guesses. Then

$$A\mathbf{v}_4 = \begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & -1 & 1\\ 1 & -1 & 0 & 1\\ -1 & 1 & 1 & 0\end{bmatrix}\begin{bmatrix}-1\\1\\1\\-1\end{bmatrix} = \begin{bmatrix}3\\-3\\-3\\3\end{bmatrix} = -3\mathbf{v}_4,$$

so $\mathbf{v}_4$ is an eigenvector A with eigenvalue $\lambda_4 = -3$. Noting that

$$\begin{aligned}&\langle \mathbf{v}_1, \mathbf{v}_2\rangle = 0, \quad \langle \mathbf{v}_1, \mathbf{v}_3\rangle = 0, \quad \langle \mathbf{v}_1, \mathbf{v}_4\rangle = 0,\\ &\langle \mathbf{v}_2, \mathbf{v}_3\rangle = 0, \quad \langle \mathbf{v}_2, \mathbf{v}_4\rangle = 0,\\ &\langle \mathbf{v}_3, \mathbf{v}_4\rangle = 0,\end{aligned}$$

we see that $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$ is an orthogonal set. Normalizing these vectors to

$$\begin{aligned}\mathbf{u}_1 &= \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 0 & -1\end{bmatrix}^T,\\ \mathbf{u}_2 &= \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|} = \frac{1}{\sqrt{2}}\begin{bmatrix}0 & 1 & -1 & 0\end{bmatrix}^T,\\ \mathbf{u}_3 &= \frac{\mathbf{v}_3}{\|\mathbf{v}_3\|} = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}^T,\\ \mathbf{u}_4 &= \frac{\mathbf{v}_4}{\|\mathbf{v}_4\|} = \frac{1}{2}\begin{bmatrix}-1 & 1 & 1 & -1\end{bmatrix}^T,\end{aligned}$$

we see that $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4\}$ is an orthonormal set. Then $P = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{u}_4\end{bmatrix}$ is an orthogonal matrix, and

$$AP = \begin{bmatrix}A\mathbf{u}_1 & A\mathbf{u}_2 & A\mathbf{u}_3 & A\mathbf{u}_4\end{bmatrix} = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & -3\mathbf{u}_4\end{bmatrix} = P\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -3\end{bmatrix} = PD,$$

where

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -3 \end{bmatrix}.$$

Thus, $P^T AP = D$, as desired.

Question 5.

We recall the orthogonal projection formula. Let $S \subset \mathbb{R}^n$ be a subspace spanned by the orthonormal set $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k\}$. Define the matrix $U = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_k \end{bmatrix}$. Then the projection $\mathbf{z}$ of a vector $\mathbf{x}$ onto the subspace S is given by

$$\mathbf{z} = UU^T\mathbf{x}.$$

To implement this formula when $S = \operatorname{span}\{\mathbf{x}_1, \mathbf{x}_2\}$ we need to find the matrix U whose columns are orthonormal vectors that span S. We can use our Gram-Schmidt code from Homework 6 to do this. Recall that we had a subroutine `gram_schmidt` that took a matrix A and found a matrix U whose columns were orthonormal and spanned the column space of A. It also detected if the columns of A were linearly independent.

Thus, we can implement the projection formula by applying `gram_schmidt` to obtain U. If `gram_schmidt` indicates that the columns of A are linearly dependent, then S is one-dimensional, so U is just one column, which is the normalization of $\mathbf{x}_1$ or $\mathbf{x}_2$, whichever is nonzero (if both are zero, we report an error). These considerations lead us to Algorithm 1.

Algorithm 1: Orthogonal projection onto a subspace spanned by 2 vectors

Input: $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, not both zero
Input: $\mathbf{x} \in \mathbb{R}^n$
Output: $\mathbf{z} \in \mathbb{R}^n$, the orthogonal projection of $\mathbf{x}$ onto $\operatorname{span}\{\mathbf{x}_1, \mathbf{x}_2\}$

$A \leftarrow \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 \end{bmatrix}$;
$U \leftarrow$ `gram_schmidt`(A);
if `gram_schmidt` *returned error* **then**
  $\mathbf{w} \leftarrow (\mathbf{x}_1 \neq \mathbf{0})\ ?\ \mathbf{x}_1 : \mathbf{x}_2$; `// ? is the ternary operator`
  $U \leftarrow \frac{\mathbf{w}}{\|\mathbf{w}\|}$;
end
$\mathbf{z} \leftarrow UU^T\mathbf{x}$;

This algorithm is implemented in Python in Listing 1. The result of running the algorithm with

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \qquad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

is given in Listing 2.

Listing 1: Projection onto a subspace spanned by 2 vectors

```python
"""Projection onto a subspace spanned by two vectors
"""
import numpy as np

import gs


```

```python
class ZeroVectorsError(BaseException):
    """Error to be raised when zero vectors are provided to proj2
    """


def proj2(x1, x2, x, eps_d=1e-10):
    """
    Find the orthogonal projection of a vector onto the subspace spanned
    by two given vectors.

    :param x1: (n) vector
    :param x2: (n) vector. x1 and x2 span the projection subspace. At least
        one of x1, x2 must be nonzero
    :param x: (n) vector to project onto the subspace
    :param eps_d: tolerance for detecing linear dependence and nonzero
        condition for x1 and x2
    :return: projection of x onto the subspace spanned by x1 and x2. Raises
        ZeroVectorsError if x1 and x2 are both 0 (or within tolerance eps_d
        of 0 in norm)
    """
    # ==== Input validation =====
    x1, x2 = np.array(x1, dtype=float).flatten(), np.array(x2, dtype=float).flatten()
    x = np.array(x, dtype=float).flatten()
    assert len(x1) == len(x2) == len(x)

    # ==== Construct u ====
    # Build input matrix a for gram_schmidt by stacking columnwise (axis=1)
    a = np.stack([x1, x2], axis=1)

    # Try to run gram_schmidt, catching possible LinearDependenceError
    try:
        u = gs.gram_schmidt(a, eps_d=eps_d)
    except gs.LinearDependenceError as e:
        # x1 and x2 are linearly dependent
        n1 = np.linalg.norm(x1)
        if n1 > eps_d:
            # Use u = x1 / norm(x1) if x1 is nonzero
            u = x1 / n1
        else:
            # Try to use u = x2 / norm(x2)
            n2 = np.linalg.norm(x2)
            if n2 > eps_d:
                u = x2 / n2
            else:
                raise ZeroVectorsError('Input vectors x1 and x2 were both (almost) 0')

    # ==== Projection formula ====
    return u @ u.T @ x


# Test example
```

```python
if __name__ == '__main__':
    x1 = (1, 0, 1)
    x2 = (1, 1, 0)
    x = (1, 2, 3)
    print('Test with')
    print(f'x1 = {x1}')
    print(f'x2 = {x2}')
    print(f'x = {x}')
    z = proj2(x1, x2, x)
    print(f'Projection = {z}')
```

Listing 2: Output of test of projection code

```
>python -m proj2
Test with
x1 = (1, 0, 1)
x2 = (1, 1, 0)
x = (1, 2, 3)
Projection = [2.33333333 0.66666667 1.66666667]
```