6.S966: Exam 2, Spring 2024

Do not tear exam booklet apart!

- This is a closed book exam. One page (8 1/2 in. by 11 in) of notes, front and back, are permitted. Calculators are not permitted.
- The total exam time is 1 hours and 20 minutes.
- The problems are not necessarily in any order of difficulty.
- Record all your answers in the places provided. If you run out of room for an answer, continue on a blank page and mark it clearly.
- If a question seems vague or under-specified to you, make an assumption, write it down, and solve the problem given your assumption.
- If you absolutely *have* to ask a question, come to the front.
- **Write your name on every piece of paper.**

Name: ______________________ MIT Email: ______________________

Question	Points	Score
1	50	
2	35	
3	15	
Total:	100	

Name: ______________________

Steerable Convolution on Hexagonal Images

1. (50 points) There are two regular lattices that tile the 2D plane, square lattices (left) and hexagonal lattices (right).

While it is more common to do convolutions over images made of square pixels, you can also do the same for hexagonal pixels. In this problem, we will determine a basis of hexagonal filters that transform as irreps, parameterize these filters using weights, and perform steerable convolution using tensor product decomposition.

If we neglect translations and center on a specific hexagon pixel, a hexagonal lattice has a point group symmetry of D_6 with six-fold (60 degree) rotations and mirrors across the edges and diagonals of the hexagon. The character table for D_6 is

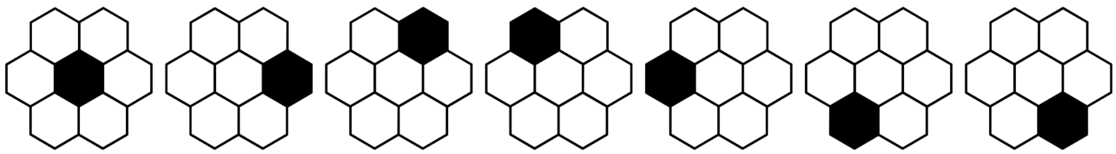
$\mathbf{D_6}$	E	$2C_6$	$2C_3$	C_2	3σ	$3\sigma'$
A_1	+1	+1	+1	+1	+1	+1
A_2	+1	+1	+1	+1	-1	-1
B_1	+1	-1	+1	-1	+1	-1
B_2	+1	-1	+1	-1	-1	+1
E_1	+2	+1	-1	-2	0	0
E_2	+2	-1	-1	+2	0	0

where the conjugacy classes are the columns and irreps are rows. For the conjugacy classes, E is the identity, C_n are rotations of $2\pi/n$, and σ are mirrors.

(a) Use the character table above to determine how many elements are in the point group D_6. What parts of the character table tell us how many elements a group has?

Name: ____________________

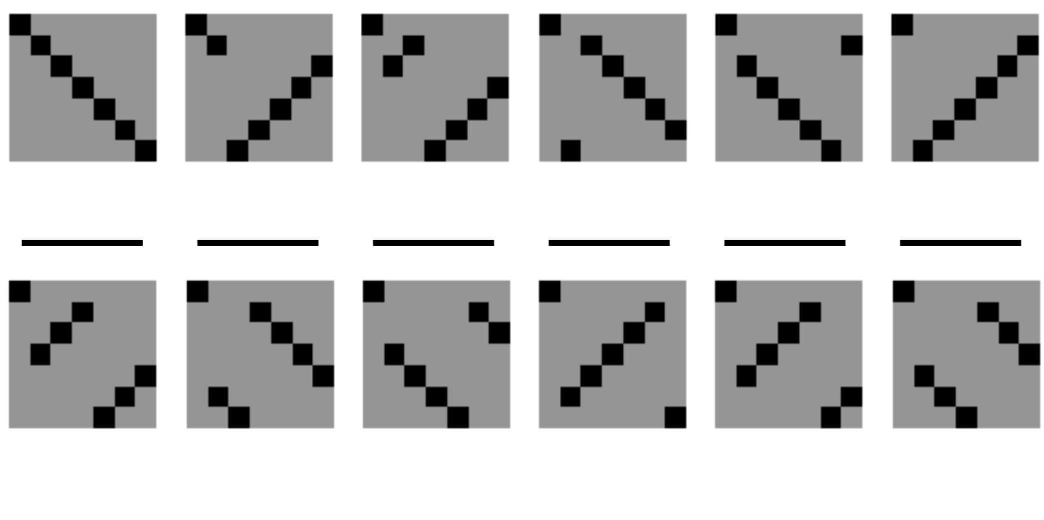
(b) A hexagonal filter up to 1st nearest neighbors (1NN) pixels is built from 7 hexagonal pixels. The single pixel basis and corresponding pixel coordinates for the hexagonal 1NN filter are



$(0,0)$ $(1,0)$ $\left(\frac{1}{2},\frac{\sqrt{3}}{2}\right)$ $\left(-\frac{1}{2},\frac{\sqrt{3}}{2}\right)$ $(-1,0)$ $\left(-\frac{1}{2},-\frac{\sqrt{3}}{2}\right)$ $\left(\frac{1}{2},-\frac{\sqrt{3}}{2}\right)$

Below, we provide the permutation matrices (grey= 0, black=1) that represent how D_6 acts on the single pixel basis (assume the columns are in the same order as the pixel basis).

Below each permutation matrix, fill in the D_6 conjugacy class that matrix belongs to. You may assume $3\sigma'$ is the conjugacy class of mirrors that pass through two edges of the central hexagon (leaves three pixels invariant), while 3σ is the conjugacy class of mirrors through pairs of vertices of the central hexagon.



(c) We define `perm_matrices` as the permutation matrices from above and execute the following code.

```
1 D6_table = groups.make_multiplication_table(perm_matrices)
2 D6_irreps = rep.infer_irreps(D6_table)
3 for i, ir in enumerate(D6_irreps):
4     print(i, linalg.infer_change_of_basis(ir, perm_matrices).shape)
5 > 0 (2, 1, 7)
6 > 1 (1, 1, 7)
7 > 2 (0, 1, 7)
8 > 3 (0, 1, 7)
9 > 4 (1, 2, 7)
10 > 5 (1, 2, 7)
```

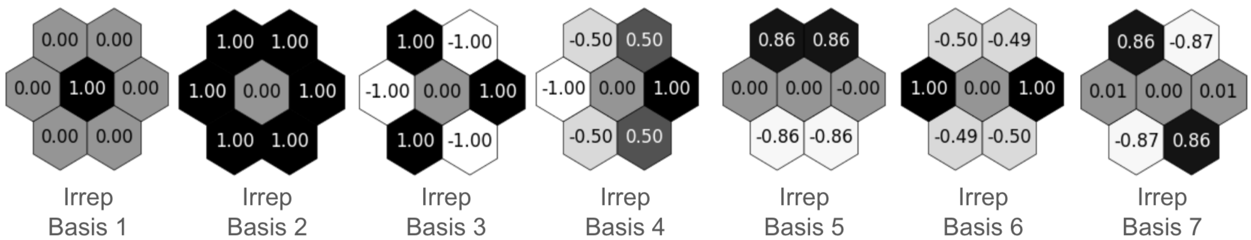
i. Describe what's happening in lines 1 and 2. In particular, how does `rep.infer_irreps` use its input to obtain irreps of the group?

ii. Describe what is happening in lines 3-4. In particular, what is the significance of the shape of the output of `linalg.infer_change_of_basis`? How many irreps are contained in the representation `perm_matrices`?

Name: ______________________________

(d) The outputs of `linalg.infer_change_of_basis` give us the change of basis between the single pixel basis and specific irreps. This means the change of basis gives coefficients indicating how much of each single pixel basis is contained in each irrep basis. We can plot these coefficients to visualize the 7 irrep basis functions below. These basis functions transform as a direct sum the following irreps of D_6:

$$\rho_{\text{Hex 1NN}} = 2A_1 \oplus B_2 \oplus E_1 \oplus E_2 \tag{1}$$



i. Irrep Basis 1, 2, and 3 correspond to the output from part (c) for $i = 0$ and $i = 1$. Using the D_6 character table, determine which of these basis functions transform as A_1 vs. B_2. Explain your reasoning.

ii. Given your answers in part (i), explain why there are two basis functions that transform as A_1? How are they similar? How are they different?

iii. Irrep Basis 4 and 5 correspond to the output from part (c) for $i = 4$. Using the D_6 character table, determine which 2D irrep these basis functions transforms as. Explain your reasoning.

iv. Irrep Basis 6 and 7 correspond to the output from part (c) for $i = 5$. Using the D_6 character table, determine which 2D irrep these basis functions transforms as. Explain your reasoning.

)

(e) Now, we want to add weights to our basis functions to parameterize convolutional kernels, $\psi(x, y) = WB(x, y)$. The weights matrix W for our kernel ψ is a linear map $W : \rho_{\text{basis}} \to \rho_{\text{filter}}$ and thus has shape $\rho_{\text{filter}} \times \rho_{\text{basis}}$, i.e. the rows span ρ_{filter} and the columns span ρ_{basis}.

To commute with group action W must satisfy the following,

$$W^{\rho_{\text{filter}} \times \rho_{\text{basis}}} D^{\rho_{\text{basis}}}(g) x^{\rho_{\text{basis}}} = D^{\rho_{\text{filter}}}(g) W^{\rho_{\text{filter}} \times \rho_{\text{basis}}} x^{\rho_{\text{basis}}}. \tag{2}$$

where D^{ρ} is the matrix representation for representation vector space ρ and is therefore a $\rho \times \rho$ matrix.

Assume $\rho_{\text{basis}} = \rho_{\text{filter}} = \rho_{\text{Hex 1NN}} = 2A_1 \oplus B_2 \oplus E_1 \oplus E_2$ (in that order). Below, fill in the weight matrix W such that it commutes with group action. Use lower case Latin letters $(a, b, \ldots, z)$ to label distinct weights. You may leave entries blank or use zeros to indicate zeros.

$\rho_{\text{basis}} \rightarrow$

$\rho_{\text{filter}} \downarrow$

(f) To perform steerable convolution, we perform an elementwise tensor product of our filter and the image patch the filter overlaps. The direct product table for the irreps of D_6 is given below. For example, using the table we see $E_1 \otimes B_2 = E_2$.

	$\mathbf{A_1}$	$\mathbf{A_2}$	$\mathbf{B_1}$	$\mathbf{B_2}$	$\mathbf{E_1}$	$\mathbf{E_2}$
$\mathbf{A_1}$	A_1	A_2	B_1	B_2	E_1	E_2
$\mathbf{A_2}$	A_2	A_1	B_2	B_1	E_1	E_2
$\mathbf{B_1}$	B_1	B_2	A_1	A_2	E_2	E_1
$\mathbf{B_2}$	B_2	B_1	A_2	A_1	E_2	E_1
$\mathbf{E_1}$	E_1	E_1	E_2	E_2	$A_1 \oplus [A_2] \oplus E_2$	$B_1 \oplus B_2 \oplus E_1$
$\mathbf{E_2}$	E_2	E_2	E_1	E_1	$B_1 \oplus B_2 \oplus E_1$	$A_1 \oplus [A_2] \oplus E_2$

If the input is a hexagonal image with features that transform as A_2, and we use $\rho_{\text{filter}} = 2A_1 \oplus B_2 \oplus E_1 \oplus E_2$, how will the outputs of tensor product transform? In other words, how does $A_2 \otimes (2A_1 \oplus B_2 \oplus E_1 \oplus E_2)$ decompose into irreps?

Name: ____________________

Planes of 4D Rotations

2. (35 points) In this question, we will investigate the Lie group of 4D rotations $SO(4)$.

   (a) Representations of Lie groups take the form of $e^{\sum_i \theta_i X_i}$ with parameters θ_i multiplying generators X_i, where $e^A = \sum_k \frac{1}{k!} A^k$. Like finite groups, Lie groups are closed under group multiplication. If X and Y are matrices that do not necessarily commute, how do we compute Z (or an approximation of Z) in $e^X e^Y = e^Z$? Explain your reasoning.

   (b) Orthogonal matrices have the property $R^T R = RR^T = I$ which means $R^T = R^{-1}$. Suppose we have an orthogonal matrix generated by A, i.e., $R = e^A$. We can use the definition of $e^A = \sum_k \frac{1}{k!} A^k$ to see $R^T = e^{A^T}$. What condition do we have on A^T if $e^A e^{A^T} = I = e^0$? What must the diagonals of these generators be?

   (c) There are 3 generators for $SO(3)$ and 6 generators for $SO(4)$. What would you expect the number of generators to be for $SO(5)$? You can derive this using the condition above for 5×5 matrices. Alternatively, you may use the fact that in n-dimensional space, there are n choose 2 (i.e., $\frac{n!}{2!(n-2)!}$) planes but in that case explain how are rotations are connected to planes. In case helpful, the generators for $SO(4)$ are given on the next page.

Name: ____________________

(d) Given the condition above, we can determine the generators for $SO(n)$ of any n. The six generators of $SO(4)$ can be written as:

$$L_1 = L_{(01)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad L_2 = L_{(02)} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad L_3 = L_{(03)} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}$$

$$L_4 = L_{(12)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad L_5 = L_{(13)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \quad L_6 = L_{(23)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

where L_i enumerates the 6 generators and $L_{(jk)}$ specifies the plane of rotation.

i. Compute the commutators $[L_{(01)}, L_{(12)}]$ and $[L_{(12)}, L_{(13)}]$

ii. Compute the commutators of $[L_{(01)}, L_{(23)}]$ and $[L_{(02)}, L_{(13)}]$.

(e) Given what you computed above, describe in words the following cases for the Lie algebra (commutator relationship) of $SO(4)$: Under what circumstances are the commutators zeros? Under what circumstances are the commutators non-zero? Don't forget to handle the case of the $[L_{(ij)}, L_{(ij)}]$. From a geometric perspective (thinking of how rotations are related to planes), does this make sense?

(f) We execute the following code.

```
1 so4_vec_vec = lie.tensor_product(so4_generators, so4_generators)
2 np.random.seed(42)
3 so3_vec_vec_irreps = lie.decompose_rep_into_irreps(so4_vec_vec)
```

In line 2, we are setting a random seed which makes the use of random numbers reproducible. How is randomness used in `lie.decompose_rep_into_irreps` to arrive at the irreps contained in the tensor product representation `so4_vec_vec`? Feel free to use an `einsum` to help give your explanation.

Name: ______________________________

Spherical Harmonic Identity

3. (15 points) From elementary trigonometry, we know that $\sin(\theta)^2 + \cos(\theta)^2 = 1$ for any θ. In this question, we prove a generalized version of this identity for spherical harmonics. Let $Y_{\ell,m}(\xi)$ denote the spherical harmonics for $\ell = 0, 1, \ldots$ and $\ell \leq m \leq \ell$. Here, $\xi = (x, y, z)^T \in \mathbb{R}^3$ is a vector. Also, assume that we normalized spherical harmonics such that

$$\int_{S^2} \left|Y_{\ell,m}(\xi)\right|^2 d\xi = 1 \tag{9}$$

for each ℓ, m, where the integral is over the sphere $S^2 = \left\{\xi \in \mathbb{R}^3 : |\xi|^2 = 1\right\}$.

In this problem, we want to show that for any ℓ and any $\xi \in S^2$,

$$\sum_{m=-\ell}^{\ell} \left|Y_{\ell,m}(\xi)\right|^2 = \frac{2\ell+1}{4\pi}. \tag{10}$$

(a) First, define the following function:

$$f(\xi_1, \xi_2) = \sum_{m=-\ell}^{\ell} Y_{\ell,m}(\xi_1) Y_{\ell,m}(\xi_2). \tag{11}$$

Use the properties of spherical harmonics under 3D rotation

$$Y_{\ell,m}(D^\xi(g)\xi) = \sum_{m'=-\ell}^{\ell} D^\ell_{m,m'} Y_{\ell,m'}(\xi) \tag{12}$$

and the property of orthogonal matrices $D^\ell_{ij}(g) = D^\ell_{ji}(g^{-1})$ to show that $f(\xi_1, \xi_2)$ is invariant under rotation, i.e. $f(D^\xi(g)\xi_1, D^\xi(g)\xi_2) = f(\xi_1, \xi_2)$.

Name: ____________________

(b) Note, that because $f(\xi_1, \xi_2)$ is invariant for any ξ_1 and ξ_2, for $\xi_1 = \xi_2 = \xi$, $f(\xi, \xi) = \sum_{m=-\ell}^{\ell} \left|Y_{\ell,m}(\xi)\right|^2$ is still invariant. Prove that $\sum_{m=-\ell}^{\ell} \left|Y_{\ell,m}(\xi)\right|^2 = \frac{2\ell+1}{4\pi}$.
Hint. Integrate the left-hand side of the above identity. You may also find Eqn. 9 helpful. Also, $\int_{S^2} d\xi = 4\pi$.

Name: ____________________

symm4ml Docstring listing

Modules listed in order: `groups, linalg, rep, lie`

```
groups.make_multiplication_table:
Makes multiplication table for group.
    Input:
        matrices: np.array of shape [n, d, d], n matrices of dimension d that form
             a group under matrix multiplication.
        tol: float numberical tolerance
    Output:
        Group multiplication table.
        np.array of shape [n, n] where entries correspond to indices of first dim
           of matrices.

linalg.infer_change_of_basis:
Compute the change of basis matrix from X1 to X2.
    tip: Use the function nullspace
    Input:
        X1: an (n, d1, d1) array of n (d1, d1) matrices
        X2: an (n, d2, d2) array of n (d2, d2) matrices
    Output:
        Sols: An (m, d1, d2) array of m solutions.
        Each solution is a (d1, d2) matrix that satisfies X1 @ S = S @ X2.

rep.decompose_rep_into_irreps:
Decomposes representation into irreducible representations.
    Input:
        rep: np.array [n, d, d] representation of group. rep[g] is a matrix that
           represents g-th element of group.
    Output:
        Irreducible representations. List of np.array [n, d_i, d_i] where d_i is a
             dimension of i-th irrep.

rep.infer_irreps:
Infers irreducible representations of group represented by multiplication table.
    Input:
        table: np.array [n, n] where table[i, j] = k means i * j = k.
    Output:
        Irreducible representations. List of np.array [n, d, d] where d is a
           dimension of irrep.

lie.decompose_rep_into_irreps:
Decomposes representation into irreducible representations.
    Input:
        X: np.array [lie_dim, d, d] - generators of a representation.
    Output:
```

```
        Ys: List[np.array] - list of generators of irreducible representations.

lie.tensor_product:
Tensor product of two representations of a Lie group.
    Input:
        X1: np.array [lie_dim, d1, d1] - generators of a representation.
        X2: np.array [lie_dim, d2, d2] - generators of a representation.
    Output:
        X: np.array [lie_dim, d1*d2, d1*d2] - tensor product of the
            representations.
```

Name: ______________________________

Work space

Name: ______________________________

Work space