6.S966: Practice 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	20	
2	20	
3	20	
Total:	60	

Name: _____

A not-so-fully-connected neural network

1. (20 points) In this problem, you will determine how we should parameterize weights to make neural network operations equivariant, i.e. commute with group action.

   In a fully-connected (or dense) neural network, layers are built from two operations: a matrix acts on the input vector and then an activation function is applied to the resulting vector.

   (a) Let's first consider the linear operation (matrix acting on the input vector). A weights matrix W is a linear map from $W : \rho_{\text{in}} \to \rho_{\text{out}}$ and thus has shape $\rho_{\text{out}} \times \rho_{\text{in}}$, i.e. the rows span ρ_{out} and the columns span ρ_{in}.

   To commute with group action W must satisfy the following,

   $$W^{\rho_{\text{out}} \times \rho_{\text{in}}} D^{\rho_{\text{in}}}(g) x^{\rho_{\text{in}}} = D^{\rho_{\text{out}}}(g) W^{\rho_{\text{out}} \times \rho_{\text{in}}} x^{\rho_{\text{in}}}. \tag{1}$$

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

   Schur's Lemma tell us the conditions under which a matrix W commutes with a group representation $D(g)$

   i. Suppose ρ_{in} and ρ_{out} are equivalent irreducible representation vector spaces in the same basis. What must W be for the above equation to hold? Explain your reasoning.

   ii. Suppose ρ_{in} and ρ_{out} are inequivalent irreducible representations. What must W be for the above equation to hold? Explain your reasoning.

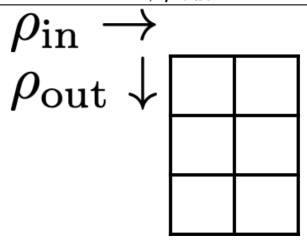
Name: _____

(b) In each of the following subparts, we will give you the representations of ρ_{in} and ρ_{in} as direct sums of irreps labeled A, B, J, or K.

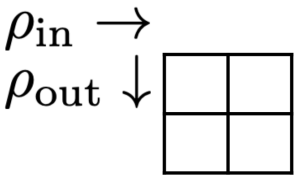
- A and B are distinct one-dimensional irreps
- J and K are distinct two-dimensional irreps

Use lower case Latin letters $(a, b, \ldots, z)$ to label distinct weights. You may leave entries blank or use zeros to indicate zeros.

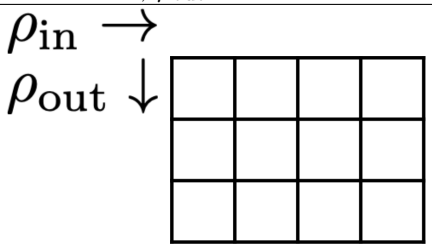
i. $\rho_{\text{in}} = A \oplus B$, $\rho_{\text{out}} = A \oplus A \oplus B$



ii. $\rho_{\text{in}} = J$, $\rho_{\text{out}} = K$



iii. $\rho_{\text{in}} = J \oplus J$, $\rho_{\text{out}} = A \oplus J$



(c) Now, let's think about what we need to do to make activation functions equivariant. For simplicity, let's assume we want to be equivariant to C_4 (90 degree rotations) is a 2D vector which does not transform as the trivial representation. To be equivariant, our activation function σ must satisfy the following property.

$$\sigma(D^{\rho_{\text{out}}} y^{\rho_{\text{out}}}) = D^{\rho_{\text{out}}} \sigma(y^{\rho_{\text{out}}}) \tag{2}$$

A common activation function is the Rectified Linear Unit (ReLU), $\text{ReLU}(x) = \max(0, x)$. Is this activation function equivariant (satisfying Eqn. 2)? You may find it useful to consider the 2D vector $(1, 0)$ and it's rotations under C_4, $(0, 1)$, $(-1, 0)$, and $(0, -1)$.

(d) What property of the 2D vector could we apply any activation function to as long as we are considering groups that can be represented with unitary transformations?

Name: _____

Cartesian Tensors

2. (20 points) In 3D Euclidean space, the standard basis is

$$\hat{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \hat{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad \hat{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{3}$$

A Cartesian Tensor is a tensor has indices over the standard basis. For example

- A 3D vector is a Cartesian Tensor with one index $v_i \to \vec{v} = (v_x, v_y, v_z)$
- A 3x3 matrix is a Cartesian Tensor with two indices M_{ij}.
- a 3x3x3 matrix is a Cartesian Tensor with three indices M_{ijk}.

Cartesian Tensors of more than one index can be thought of as a tensor product representation of vector representations, i.e. $\rho^M = \rho^{\vec{v}} \otimes \rho^{\vec{v}}$. We can decompose these tensor product representation to understand what irreps these Cartesian Tensors are made of.

A valid set of generators for $SO(3)$ on the (irreducible) vector representation, i.e. $L = 1$, is

```
so3_vec = np.array([
    [[0, 0, 0.],
     [0, 0, -1],
     [0, 1, 0]],
    [[0, 0, 1],
     [0, 0, 0],
     [-1, 0, 0]],
    [[0, -1, 0],
     [1, 0, 0],
     [0, 0, 0]],
])
```

We can infer the first 5 irreps of $SO(3)$ using

```
so3_irreps = lie.infer_irreps_from_tensor_products(so3_vec, n=5)
```

We can compute the tensor product representations of 2, 3, and 4 index Cartesian Tensors using

```
so3_vec_vec = lie.tensor_product(so3_vec, so3_vec)
so3_vec_vec_vec = lie.tensor_product(so3_vec_vec, so3_vec)
so3_vec_vec_vec_vec = lie.tensor_product(so3_vec_vec_vec, so3_vec)
```

Name: _____

(a) We execute the following code

```
for i in range(5):
  print(i, linalg.infer_change_of_basis(so3_irreps[i], so3_vec_vec).shape)
> 0 (1, 1, 9)
> 1 (1, 3, 9)
> 2 (1, 5, 9)
> 3 (0, 7, 9)
> 4 (0, 9, 9)
```

What do the outputs of `linalg.infer_change_of_basis` indicate when one input is an irreducible representation like `so3_irreps[i]` and the other input is a reducible representation like `so3_vec_vec`?

i. What is the significance of the first value of the shape (`shape[0]`) of the outputs of `linalg.infer_change_of_basis`?

ii. What is the significance of the second value of the shape (`shape[1]`) of the outputs of `linalg.infer_change_of_basis`?

iii. What is the significance of the third value of the shape (`shape[2]`) of the outputs of `linalg.infer_change_of_basis`?

(b) What do these outputs tell us about the tensor product representation of two index Cartesian Tensors (3×3 matrices)?

(c) Does decomposition of the reducible tensor product representation into a direct sum over irreps preserve dimension? Use the outputs from above to check this. Explain your reasoning.

(d) We execute the following code

```
for i in range(5):
  print(i, linalg.infer_change_of_basis(so3_irreps[i], so3_vec_vec_vec).shape)
> 0 (1, 1, 27)
> 1 (3, 3, 27)
> 2 (2, 5, 27)
> 3 (1, 7, 27)
> 4 (0, 9, 27)
```

What do these outputs tell us about the tensor product representation of three index Cartesian Tensors ($3 \times 3 \times 3$ tensors)?

(e) We execute the following code

```
for i in range(5):
  print(i, linalg.infer_change_of_basis(so3_irreps[i], so3_vec_vec_vec_vec).shape)
> 0 (3, 1, 81)
> 1 (6, 3, 81)
> 2 (6, 5, 81)
> 3 (3, 7, 81)
> 4 (1, 9, 81)
```

What do these outputs tell us about the tensor product representation of three index Cartesian Tensors ($3 \times 3 \times 3$ tensors)?

Name: _____

(f) The selection rules for tensor products decompositions of irreps of $SO(3)$, $l_1 \otimes l_2 \to l_3$, can be expressed as $|l_1 - l_2| \leq l_3 \leq l_1 + l_2$. Given this, what is the maximum irrep contained in an n-index Cartesian Tensor. Explain your reasoning.

(g) The irreps of a group are often reducible under the symmetries of a subgroup. For example, under spherical symmetry, only the irrep $L = 0$ is trivial. If a tensor has spherical symmetry, only components that transform as $L = 0$ can be nonzero.

Note: The symmetry of the tensor is different from the symmetry of the representations for how a tensor transforms, this is identical to the distinction between the symmetry of coordinate systems, versus the symmetry of "objects" in 3D space. The "symmetry of a tensor" is the later. The indices of the tensor still transform a representation of $SO(3)$ regardless of the tensor's symmetry

Under octahedral symmetry O_h (a subgroup of $L = 0$), $L = 0$ is still trivial but so are components of $L = 4$ (and components of higher Ls). What is the minimum number of indices a Cartesian tensor needs to be able to distinguish whether a tensor has spherical versus octahedral symmetry.

Name: _____

The Representations of $SU(2)$

3. (20 points) The special unitary group in 2 dimensions, $SU(2)$, comprises complex 2×2 matrices with determinant 1 and satisfy $U^\dagger U = 1$ where $U^\dagger$ is the conjugate transpose.

   $SU(2)$ is a Lie group and a valid set of generators of this group are

$$\sigma_x = \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} \qquad \sigma_y = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_z = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} \tag{4}$$

   (a) These generators are irreducible. How can we tell? Explain your reasoning.

   (b) Compute the following commutators of the generators: $[\sigma_x, \sigma_y]$, $[\sigma_y, \sigma_z]$, and $[\sigma_z, \sigma_x]$.

   (c) Given what you computed above, express the Lie algebra for this group. You may find it handy to use the Levi-Civita symbol ϵ_{ijk}, where $\epsilon_{ijk} = 0$ if ijk contain repeated indices e.g. xxz, $\epsilon_{ijk} = 1$ for ijk equal to even permutations of xyz, e.g. zxy, and $\epsilon_{ijk} = -1$ ijk equal to odd permutations of xyz, e.g. yxz. Does this Lie algebra look familiar? Why or why not?

(d) We can perform tensor product decompositions of the generators to find irreps of the group $SU(2)$. We will define the generators for $SU(2)$ the same as above except we will add a factor of $\frac{1}{2}$ which we will explain later.

```
su2_generators = np.array([
    [[0, 1j],
     [1j, 0]],
    [[0, -1],
     [1, 0]],
    [[1j, 0],
     [0, -1j]],
]) / 2.
```

Using the definition of `so3_generators` from the previous problem, we execute the following code

```
so3_irreps = lie.infer_irreps_from_tensor_products(so3_generators, n=4)
for ir in so3_irreps:
  print(ir.shape)
> (3, 1, 1)
> (3, 3, 3)
> (3, 5, 5)
> (3, 7, 7)
```

And similarly for $SU(2)$

```
su2_irreps = lie.infer_irreps_from_tensor_products(su2_generators, n=7)
for ir in su2_irreps:
  print(ir.shape)
> (3, 1, 1)
> (3, 2, 2)
> (3, 3, 3)
> (3, 4, 4)
> (3, 5, 5)
> (3, 6, 6)
> (3, 7, 7)
```

Name: _____

i. Given that the dimensions of some of the irreps look similar, we try the following

```
lie.are_isomorphic(su2_irreps[2], so3_irreps[1])
> True

lie.are_isomorphic(su2_irreps[4], so3_irreps[2])
> True

lie.are_isomorphic(su2_irreps[6], so3_irreps[3])
> True
```

What does this tell us about the representations of $SU(2)$ and $SO(3)$?

ii. Suppose we had not added the factor of $\frac{1}{2}$ to the SU(2) generators, or equivalently we multiply our generators by 2.

```
lie.are_isomorphic(su2_irreps[2] * 2, so3_irreps[1])
> False

lie.are_isomorphic(su2_irreps[4] * 2, so3_irreps[2])
> False

lie.are_isomorphic(su2_irreps[6] * 2, so3_irreps[3])
> False
```

Why are these representations no longer isomorphic? How does this relate to how we compute isomorphism of representations?

Name: _____

Work space

Name: _____

Work space