An Exploration into Linear Algebra

Yash Money, Imran Iftikar

Idk man, I just work here

1 Basic Matrix Code

To begin, we have will create code that represents matrices and performs elementary matrix operations, such as computing the inverse, multiplying, and adding.

We will utilise python to do so. Python has a data structure known as a "list" or an "array." These are essentially a collection of indexed data than can be manipulated. Lists may contain sublists; it is in this way that we can represent a "matrix" in python, for indeed, a matrix is nothing but a collection of row/column vectors, which themselves can be represented as an individual python lists.

We have decided to represent a matrix as a collection of row vectors.

For example consider the following matrix, $A \in \mathbb{R}^{m \times n}$:

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ b_1 & b_2 & b_3 & \dots & b_n \\ \dots & \dots & \dots & \dots & \dots \\ m_1 & m_2 & m_3 & \dots & m_n \end{bmatrix}$$

with m rows and n columns. We choose to represent the same matrix, A pythonically in the following way:

1.1 Matrix Multiplication

```
def project_vector(vector, proj_mat, return_error=False):

'''
if using sub_space obj, will look like newvec =
    project_vector(vec, sub_space.proj_mat)
'''

projected = mp.multiply_matrix(proj_mat, [vector])

if return_error == False:
    return projected[0]
else:
    return mp.mround(projected)[0],
        euclidean_norm(mp.subtract_row(vector, projected[0]))
```

2 Representing Subspaces Pythonically

3 Representing Linear Mappings Pythonically

4 More Advanced Matrix Operations

4.1 Solving Homogeneous Matricies

4.2 Finding Eigenvalues and Eigenvectors Of A Matrix

4.3 Finding the Eiegendecomposition

4.4 Finding the Singular Value Decomposition