Unit III

CHAPTER 3

Gaussian Elimination, Inner Product and Orthogonalization

Syllabus Topics

Gaussian elimination : Echelon form, Gaussian elimination over GF(2), Solving a matrix-vector equation using Gaussian elimination, Finding a basis for the null space, Factoring integers.

Inner Product : The inner product for vectors over the reals, Orthogonality,

Orthogonalization : Projection orthogonal to multiple vectors, Projecting orthogonal to mutually orthogonal vectors, Building an orthogonal set of generators, Orthogonal complement, Eigenvector : Modeling discrete dynamic processes, Diagonalization of the Fibonacci matrix, Eigenvalues and eigenvectors, Coordinate representation in terms of eigenvectors, The Internet worm, Existence of eigenvalues, Markov chains, Modeling a web surfer: PageRank.

Syllabus Topic : Gaussian Elimination

We have already defined row echelon form of a matrix, which helped us t understand the rank of a matrix.

Here with help of row echelon form definition we make a note that a squa matrix which is upper triangular is also in a row echelon form.

i.e. Example $\begin{bmatrix} 2 & 4 & -1 \\ 0 & 6 & -7 \\ 0 & 0 & 9 \end{bmatrix}$

Gaussian elimination is most often applied to solving a system of lin equations.

It is generally applied to matrices over field $\mathbb{R}$. When it is actually carried o on computer using floating point arithmetic there are some other subtle involved ensuring that the outputs are accurate. Here we shall focus on Gauss elimination method on matrices over $\mathbb{R}$ and GF (2).

Now we explain the typical procedure of Gaussian elimination method.

We first clear idea about the row transformation $R_i \to R_i + kR_j$ means that the change happens in i^{th} row and the change is that the i^{th} row elements are going to be added with k times the corresponding elements of j^{th} row (in short every elements of i^{th} row will now be $r_i + k\, r_j$).

$R_i \leftrightarrow R_j$ means that the i^{th} row and the j^{th} row have interchanged and $R_i \to k\, R_j$, means that the i^{th} row elements are multiplied by the scalar k.

3.1 Gaussian Elimination Method

Consider the linear system of n equation and n unknown in matrix notation is written as

$$AX = B$$

$$\text{Where } A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

By performing sequence of row transformations we convert the above system to a equivalent system.

$\tilde{A}\, X = \tilde{B}$ where $\tilde{A}$ is a triangular matrix i.e.

Triangular matrix

$$\tilde{A} = \begin{bmatrix} \tilde{a_{11}} & \tilde{a_{12}} & \ldots & \tilde{a_{1n}} \\ 0 & \tilde{a_{22}} & \ldots & \tilde{a_{2n}} \\ 0 & 0 & \tilde{a_{33}} \ldots & \tilde{a_{3n}} \\ \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \tilde{a_{nn}} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} \tilde{b_1} \\ \tilde{b_2} \\ \vdots \\ \tilde{b_n} \end{bmatrix}$$

Then we solve for $x_1, x_2, \ldots, x_n$ by backward substitution.

i.e. we get first value for x_n then x_{n-1}, x_{n-2}, and so on ... x_1.

Gaussian elimination operation counts

Gaussian elimination method with back substitution applied on n × n system requires $\frac{n^3}{3} + n^2 - \frac{n}{3}$ multiplications or divisions and $\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}$ additions o subtractions.

As n grows, the $\frac{n^3}{3}$ term dominates each of these expressions. Thus, th important thing to remember is that Gaussian elimination with back substitutio on an n × n system requires about $\frac{n^3}{3}$ multiplication / divisions and about the sam number of additions / subtraction.

Example

Solve the following system using Gaussian elimination method.

$$\begin{aligned} v - w &= 3 \\ -2u + 4v - w &= 1 \\ -2u + 5v - 4w &= -2 \end{aligned}$$

$$\therefore \begin{bmatrix} 0 & 1 & -1 \\ -2 & 4 & -1 \\ -2 & 5 & -4 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ -2 \end{bmatrix}$$

Perform $R_1 \leftrightarrow R_2$

$$\begin{bmatrix} -2 & 4 & -1 \\ 0 & 1 & -1 \\ -2 & 5 & -4 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

$R_3 \rightarrow R_3 - R_1$

$$\begin{bmatrix} -2 & 4 & -1 \\ 0 & 1 & -1 \\ 0 & 1 & -3 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ -3 \end{bmatrix}$$

$R_3 \rightarrow R_3 - R_2$

$$\begin{bmatrix} -2 & 4 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ -6 \end{bmatrix}$$

$$\therefore -2w = -6 \Rightarrow w = 3$$

$$v = 3 + w = 3 + 3 = 6$$

$$-2u + 4v - w = 1 \Rightarrow u = \frac{1}{-2}(1 - 4v + w)$$

$$\Rightarrow \frac{1}{-2}(1 - 24 + 3) = 10$$

Questions for practice using Gaussian elimination.

(1) $x_1 + x_2 + x_3 = 1$
$x_1 + 2x_2 + 2x_3 = 1$
$x_1 + 2x_2 + 3x_3 = 1$

(2) $4x_2 - 3x_3 = 3$
$-x_1 + 7x_2 - 5x_3 = 4$
$-x_1 + 8x_2 - 6x_3 = 5$

The following python code shows how to find the determinant of the matrix. Here we have used Sympy library and the code is run on Sympy shell.

```
>>> from sympy import *
>>> M = Matrix([[1, 0, 1], [2, -1, 3], [4, 3, 2]])
>>> M
```

$$\begin{bmatrix} 1 & 0 & 1 \\ 2 & -1 & 3 \\ 4 & 3 & 2 \end{bmatrix}$$

```
>>> M.det()
```

```
-1
```

The reduced row echelon we find as follows

```
>>> M = Matrix([[1, 0, 1, 3], [2, 3, 4, 7], [-1, -3, -3, -4]])
>>> M
```

$$\begin{bmatrix} 1 & 0 & 1 & 3 \\ 2 & 3 & 4 & 7 \\ -1 & -3 & -3 & -4 \end{bmatrix}$$

```
>>> M.rref()
```

$$\left(\begin{bmatrix} 1 & 0 & 1 & 3 \\ 0 & 1 & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad [0, \ 1] \right)$$

The Null space is find as follows by using the Python.

```
>>> M = Matrix([[1, 2, 3, 0, 0], [4, 10, 0, 0, 1]])
>>> M
```

$$\begin{bmatrix} 1 & 2 & 3 & 0 & 0 \\ 4 & 10 & 0 & 0 & 1 \end{bmatrix}$$

```
>>> M.nullspace()
```

$$\left[\begin{bmatrix} -15 \\ 6 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -\frac{1}{2} \\ 0 \\ 0 \\ 1 \end{bmatrix} \right]$$

The Gaussian elimination code using the **gauss_jordan_solve()** is given as follows using Sympy.


```
>>> from sympy import Matrix
>>> A = Matrix([[1, 2, 1, 1], [1, 2, 2, -1], [2, 4, 0, 6]])
>>> b = Matrix([7, 12, 4])
>>> sol, params = A.gauss_jordan_solve(b)
>>> sol
```

$$\begin{bmatrix} -2\tau_0 - 3\tau_1 + 2 \\ \tau_0 \\ 2\tau_1 + 5 \\ \tau_1 \end{bmatrix}$$

```
>>> params
```

$$\begin{bmatrix} \tau_0 \\ \tau_1 \end{bmatrix}$$

Syllabus Topic : Echelon form

3.1.1 From Echelon form to a Basis for Row Space

Q. Why it is good to have a matrix in row echelon form?

Lemma 3.1.1 : If a matrix is in echelon form, the nonzero rows form a basis for the row space.

Solution : For example, a basis for the row space of

$$\begin{bmatrix} 0 & 2 & 3 & 0 & 5 & 6 \\ 0 & 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

is {[0 2 3 0 5 6], [0 0 1 0 3 4]}.

In particular, if every row is nonzero, as in each of the matrices

$$\begin{bmatrix} 0 & 2 & 3 & 0 & 5 & 6 \\ 0 & 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 9 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 & 4 & 1 & 3 & 9 & 7 \\ 0 & 6 & 0 & 1 & 3 & 0 & 4 & 1 \\ 0 & 0 & 0 & 0 & 2 & 1 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 1 & 3 & 0 \\ 0 & 3 & 0 & 1 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 9 \end{bmatrix}$$

Then the rows form a basis of the row space.

To prove Lemma 3.1.1, note that it is obvious that the nonzero rows span the row space; we need only show that these vectors are linearly independent.

Before giving the formal argument, I illustrate it using the matrix

$$\begin{bmatrix} 4 & 1 & 3 & 0 \\ 0 & 3 & 0 & 1 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 9 \end{bmatrix}$$

Recall the grow algorithm

```
def Grow (V)
S = φ
Repeat while possible:
Find a vector v in V that is not is span S, and put it in S.
```

We imagine the Grow algorithm adding to S each of the rows of the matrix, in reverse order

☞ **Gaussian Elimination**

- Initially $S = \phi$
- Since space ϕ does not include [0 0 0 0 9], the algorithm adds this vector to S.
- Now S = {[0, 0, 0, 9]}. Since every vector in span S has zeroes in the first three positions, span S does not contain [0, 0, 1, 7], so the algorithm adds this vector to S.
- Now, S = {[0, 0, 0, 9], [0, 0, 1, 7]}. Since every vector in span S has zeroes in the first two positions, span S does not contain [0, 3, 0, 1], so the algorithm adds this vector to S.

  Now, S = {[0, 0, 0, 9], [0, 0, 1, 7], [0, 3, 0, 1]}. Since every vector in Span S has a zero in the first position. Span S does not contain [4, 1, 3, 0], so the algorithm adds this vector to S and we are done.

By the Grow-Algorithm corollary, the set S is linearly independent. Now we present the same argument more formally:

☞ **Proof**

Let $a_1, \ldots\ldots, a_m$ be the vectors of a matrix in echelon form, in order from top to bottom. To show that these vectors are linearly independent, imagine we run the Grow algorithm on Span $\{a_1, \ldots\ldots, a_m\}$. We direct the Grow algorithm to add the a_m, a_{m-1}, ..., a_2, a_1 to S, in that order.

After the Grow algorithm has added a_m, a_{m-1}, a_i to S, we want it to add a_{i-1}. How can we be sure a_{i-1} is not already in span(S)? Suppose the first nonzero entry of a_{i-1} is in the k+ 1st position. Then, by the definition of echelon form, the first k entries of a_{i-1} are zero, so the first k +1 entries of a_i, a_{i+1}, ... a_m are zero. Therefore the first k + 1 entries of every vector in span S are zero. Since the $k + 1^{th}$ entry of a_{i-1} is nonzero, this vector is not in span S, so the algorithm is allowed to add it.

3.1.2 Rowlist in Echelon Form

- Since echelon form has a lot to do with rows, it is convenient in working with echelon form to represent a matrix not as an instance of Mat but as a row list a list of vectors.
- Since we want to handle vectors over arbitrary finite domains D rather than just sets of the form {0, 1, 2, ..., n – 1}, we have to decide on an ordering of the labels (the column-labels of the matrix).

  For this purpose, we use

  col_label_list= sorted (rowlist [0] . D, key = hash)

  Which sorts the labels.

3.1.3 Sorting Rows by Position of the Leftmost Nonzero

- Of course, not every rowlist is in echelon form. Our goal is to develop an algorithm that, given a matrix represented by a row-list, transforms the matrix into one in echelon form. We will later see exactly what kind of transformations are permitted.
- To start, let's simply find a reordering of the vectors in rowlist that has a chance of being in echelon form. The definition of echelon form implies that the shows should be ordered according to the positions of their leftmost nonzero entires. We will use a naive sorting algorithm: try to find a row with a nonzero in the first column, then a row with a nonzero in the second column,

and so on. The algorithm will accumulate the rows found in a list new_ rowlist, initially empty :

```
new_rowlist = [ ]
```

- The algorithm maintains the set of indices of rows remaining to be sorted, rows left, initially consisting of all the row indices :

```
row _left = set (range(len (rowlist)))
```

- It iterates through the column labels in order, finding a list of indices of the remaining rows that have nonzero entires in the current column. It takes one of these rows, adds it to new_ rowlist, and removes its index from rows_left.

```
for c in col_label_list :
    rows_with_nonzero = [r for r in row_left if rowlist [r] [c] ! = 0]
    pivot = rows_with_nonzero [0]
    now_rowlist.append (rowlist [pivot] )
    rows_left.remove (pivot)
```

- The row added to new_rowlist is called the pivot row, and the element of the pivot row in column c is called the pivot element.

Okay, let's try out our algorithm with the matrix

$$\begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 9 \end{bmatrix}$$

Things start out okay. After the iterations c = 0 c = 1, new_rowlist is

[[1 2 3 4 5], [0 2 3 4 5]]and rows_left is {1, 2, 4}.

- The algorithm runs into trouble in iteration c = 2 since none of the remaining rows have a nonzero in column 2.
- The code above raises a list index out of range exception. How can we correct this flaw ?
- When none of the remaining rows have a nonzero in the current column, the algorithm should just move on to the next column without changing new_rowlist or rows_left. We amend the code accordingly :


```
for c in col_label_list :
rows_with_nonzero = [r for r in rows_left if rowlist [r] [c] = [= 0] ]
if rows_with_nonzero ! = [ ] :
pivot = rows_with_nonzero [0]
new_rowlist. append (rowlist[pivot])
row_left.remove (pivot)
```

- With this change, the code does not raise an exception but at termination new_rowlist is

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 6 & 7 \end{bmatrix}$$

- Which violates the definition of echelon form : the fourth row's first nonzero entry occurs in the fourth position, which means that every previous row's first nonzero entry must be strictly to the left of the third position but the third row's first nonzero entry is in the fourth position.

3.1.4 Elementary Row-Addition Operations

There is a way to repair the algorithm. However, if, in the iteration corresponding to some column-label c, there is a row other than the pivot row that has a nonzero element in the corresponding column then the algorithm must perform an elementary row-addition operation to make that element into a zero.

For example, given the matrix

$$\begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix}$$

- In the iteration corresponding to the fourth column, the algorithm subtracts twice the second row

2 [0 0 0 3 2]

From the fourth

[0 0 0 6 7]

Getting new fourth row

[0 0 0 6 7] – 2 [0 0 0 3 2] = [0 0 0 6 – 6 7 – 4] = [0 0 0 0 3]

During the same iteration, the algorithm also subtracts thrice the second row

3 [0 0 0 3 2]

From the fifth

[0 0 0 9 9]

Getting new fifth row

[0 0 0 9 9] – 3 [0 0 0 3 2] = [0 0 0 0 3]

The resulting matrix is

$$\begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

- In the iteration corresponding to the fifth column, the algorithm selects [0 0 0 3] as the pivot row, and adds it to new_rowlist. Next, the algorithm subtracts two-thirds times the fourth row from the fifth row, getting new fifth row.

$$[0\ 0\ 0\ 0\ 2] - \frac{2}{3}[0\ 0\ 0\ 0\ 3] = [0\ 0\ 0\ 0\ 0]$$

- There are no more columns, and the algorithms stops. At this point, new_rowlist is

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

The code for the procedure is,

```
for c in col_label_list :
rows_with_nonzero = [r for r in row_left if rowlist [r] [c] ! = 0]
if rows_with_nonzero ! = [] :
pivot = rows_with_nonzero [0]
```



```
rows_left.remove (pivot)
new_rowlist. append (rowlist [pivot])
⇒ for r in row_with_nonzero [1 : ] :
⇒ multiplier = rowlist [r] [c] / rowlist [pivot] [c]
⇒ rowlist [r] = multiplier * rowlist [pivot]
```

- The only change is the addition of a loop in which the appropriate multiple of the pivot row is subtracted from the other remaining rows.
- We will prove that, when the algorithm completes, new_rowlist is a basis for the row space of the original matrix.

3.1.5 Multiplying by an Elementary Row-Addition Matrix

Subtracting a multiple of one row from another can be performed by multiplying the matrix by a elementary row-addition matrix.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 6 & 7 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

As we know, such a matrix is invertible:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{bmatrix} \text{ are inverses.}$$

3.1.6 Row-Addition Operations Preserve Row Space

- Our nominal goal in transforming a matrix into echelon form is to obtain a basis for the row space of the matrix.
- We will prove that row-addition operations do not change the row space.
- Therefore a basis for the row space of the transformed matrix is a basis for the original matrix.

Lemma 3.1.2 : For matrices A and N, Row NA $\subseteq$ Row A.

Proof :

Let v be any vector in Row NA. That is, v is a linear combination of the rows of NA. By the linear combinations definition of vector-matrix multiplication, there is a vector u such that

$v = [u^T]([N][A])$

$= ([u^T][N])[A]$ by associativity.

Which shows that v can be written as a linear combination of the rows of A.

Corollary 3.1.3 : For matrices A and M, if M is invertible the Row MA = Row A.

Proof:

By applying Lemma 3.1.3 with N = M, we obtain Row MA $\subseteq$ Row A.

Let B = MA. Since M is invertible, it has an inverse M^{-1}. Applying the lemma with $N = M^{-1}$.

We obtain row M^{-1} B $\subseteq$ Row B.

Since $M^{-1}B = M^{-1}(MA) = (M^{-1}M)A = IA = A$,

This proves Row A $\subseteq$ Row MA.

Example 3.1.4 : We return to the example in section 3.1.4. Let $A = \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix}$

and let $M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$.

Multiplying M by A yields $MA = \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix}$.

Solution : We will use the argument of Lemma 3.1.2 to show that row MA $\subseteq$ row A and row A $\subseteq$ row MA. Every vector v in row MA can be written as

$$v = [u_1\ u_2\ u_3\ u_4]\ MA$$

$$= [u_1\ u_2\ u_3\ u_4] \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix}$$

$$= [u_1\ u_2\ u_3\ u_4] \left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix} \right)$$

$$= \left([u_1\ u_2\ u_3\ u_4] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix} \right)$$

Showing that v can be written as a vector times the matrix A. This shows that v is row A. Since every vector in row MA is also in Row A, we have shown row MA $\subseteq$ Row A.

We also need to show that row A $\subseteq$ Row MA. Since $A = M^{-1}MA$, it suffices to show that Row M^{-1} MA $\subseteq$ Row MA.

Every vector v in Row M^{-1} MA can be written as,

$$v = [u_1\ u_2\ u_3\ u_4]\ M^{-1}MA$$

$$\left([u_1\ u_2\ u_3\ u_4] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix} \right)$$

$$= \left([u_1\ u_2\ u_3\ u_4] \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&2&1&0\\ 0&0&0&0&1 \end{bmatrix} \right) \begin{bmatrix} 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&-2&1&0\\ 0&0&0&0&1 \end{bmatrix} \begin{bmatrix} 0&2&3&4&5\\ 0&0&0&3&2\\ 1&2&3&4&5\\ 0&0&0&6&7\\ 0&0&0&9&8 \end{bmatrix}$$

Showing that v can be written as a vector times the matrix MA. this shows that v is in Row MA.

3.1.7 Basis, Rank, and Linear Independence through Gaussian Elimination

The program, we have written has been incorporated into a procedure row_reduce (rowlist) that, given a list rowlist of vectors, mutates the list performing the row-addition operations, and returns a list of vectors in echelon form with the same span as rowlist. The list of vectors returned includes no zero vectors, so is a basis for the span of rowlist.

Now that we have a procedure for finding a basis for the span of given vectors we can easily write procedures for rank and linear independence. But are they correct ?

☞ When Gaussian Elimination Fails

We have shown that the algorithm for obtaining a basis is mathematically correct. However, Python carries out its computations using floating-point numbers, and arithmetic operations are only approximately correct. As a consequence, it can be tricky to use the result to decide on the rank of a set of vectors.

Consider the Example

$$A = \begin{bmatrix} 10^{-20} & 0 & 1 \\ 1 & 10^{20} & 1 \\ 0 & 1 & -1 \end{bmatrix}$$

The rows of A are linearly independent. However, when we call row_reduce on these rows, the result is just two rows, which might lead us to conclude that the row rank is two.

First, for column $c = 0$, the algorithm selects the first row, $[10^{-20}\ 0\ 1]$, as the pivot row. It subtracts 10^{20} times this row from the second row, $[1\ 10^{20}\ 1]$, which should result in

$[1\ 10^{20}\ 1] - 10^{20}[10^{-20}\ 0\ 1] = [0\ 10^{20}\ 1 - 10^{20}]$

However, let's see how python computes the last entry :

```
>>> 1 - 1e + 20
- 1 e + 20
```

The 1 is swamped by the – 1e + 20, and is lost. Thus, according to python, the matrix after the row-addition operation is

$$\begin{bmatrix} 10^{-20} & 0 & 1 \\ 1 & 10^{20} & -10^{20} \\ 0 & 1 & -1 \end{bmatrix}$$

Next, for column $c = 1$, the algorithm selects the second row $[0\ 10^{20} - 10^{20}]$ as the pivot row, and subtracts 10^{20} times this row from the third row, resulting in the matrix.

$$\begin{bmatrix} 10^{-20} & 0 & 1 \\ 1 & 10^{20} & -10^{20} \\ 0 & 0 & 0 \end{bmatrix}$$

The only remaining row, the third row, has a zero in column c = 2, so no pivot row is selected, and the algorithm completes.

3.1.8 Pivoting and Numerical Analysis

While errors in calculation cannot be avoided when using in exact floating-point arithmetic, disastrous scenarios can be avoided by modifying Gaussian elimination. Pivoting refers to careful selection of the pivot element. Two strategies are employed:

- **Partial Pivoting :** Among rows with nonzero entries in column c, choose row with entry having largest absolute value.
- **Complete Pivoting :** Instead of selecting order of columns beforehand, choose each column on the fly to maximize pivot element.

Usually partial pivoting is used in practice because it is easy to implement and it runs quickly, but theoretically it can still get things disastrously bad for big matrices. Complete pivoting keeps those errors under control. The field of numerical analysis provides tools for the mathematical analysis of errors resulting from using algorithms such as Gaussian elimination with in exact arithmetic.

I don't cover numerical analysis in this text; I merely want the reader to b aware of the pitfalls of numerical algorithms and of the fact that mathematic analysis can help to guide the development of algorithms that side step thes pitfalls.

Using inexact arithmetic to compute the rank of a matrix is notoriously trick The accepted approach uses the singular value decomposition of the matrix concept covered a little later.

Syllabus Topic : Gaussian Elimination Over GF (2)

3.2 Gaussian Elimination Over GF (2)

– Gaussian elimination can be carried out on vectors over GF (2), and in th case all the arithmetic is exact, so no numerical issues arise.

Here is an example. We start with the matrix

	A	B	C	D
1	0	0	one	one
2	one	0	one	one
3	one	0	0	one
4	one	one	one	one

The algorithm iterates through the columns in the order A, B, C, D. F column A, the algorithm selects row 2 as the pivot row.

Since rows 3 and 4 also have non-zeroes in column A, the algorithm perfo row-addition operations to add row 2 to rows 3 and 4, obtaining the matrix.

	A	B	C	D
1	0	0	one	one
2	one	0	one	one
3	0	0	one	0
4	0	one	0	0

Now the algorithm handles column B. The algorithm selects row 4 as the p row. Since the other remaining rows (1 and 3) have zeroes in column B, no operations need be performed for this iteration, so the matrix does not change

Now the algorithm handles column c. It selects row 1 as the pivot row. The only other remaining row is row 3, and the algorithm performs a row addition operation to add row 1 to row 3 obtaining the matrix.

	A	B	C	D
1	0	0	one	one
2	one	0	one	one
3	0	0	0	one
4	0	one	0	0

Finally, the algorithm handles column D. The only remaining row is row 3, the algorithm selects it as the pivot row. There are no other rows, so no row-addition operations need to be performed. We have completed all the iterations for all columns. The matrix represented by new_rowlist is

	A	B	C	D
1	one	0	one	one
2	0	one	0	0
3	0	0	one	one
4	0	0	0	one

You can find a couple of example matrixes in the file

gaussian_examples.py

3.3 Using Gaussian Elimination for other Problems

We have learned that the nonzero rows of a matrix in echelon form are a basis for the row space of the matrix. We have learned how to use Gaussian elimination to transform a matrix into echelon form without changing the row space. This given us an algorithm for finding a basis of the row space of a matrix.

However, Gaussian elimination can be used to solve other problems as well :

(i) Solving linear systems, and

(ii) Finding a basis for the null space.

Over GF(2), the algorithm for solving a linear system can be used, for example, to solve an instance of lights out. It can be used by Eve to find the secret password used in the simple authentication scheme. More seriously, it can even be

used to predict the next random numbers coming from Python's random-numb
generator random. Over GF(2), finding a basis for the null space can be used
find a way to corrupt a file that will not be detected by our naive check
function. More seriously, it can be used to help factor integers, a notorious
difficult computational problem whose difficulty is at the heart of
cryptographic scheme, RSA, commonly used in protecting credit-card numb
transmitted via web browsers.

There is an Invertible matrix M such that MA is in Echelon form

The key idea in using Gaussian elimination to solve these other problems is
keep track of the elementary row-addition operations used to bring the inp
matrix into echelon form.

Remember that you can apply an elementary row-addition operation t
matrix by multiplying an elementary row-addition matrix M times the matr
Starting with the matrix A.

- The algorithm performs one row-addition operation, resulting in the mat
  M_1A.
- Then performs another row-addition operation on that matrix, resulting in
  matrix $M_2 M_1 A$.

:

and so on, resulting in the end in the matrix

$$M_k M_{k-1} \ldots M_2 M_1 A$$

If k is the total number of row-addition operations. Let $\overline{M}$ be the product of
through M_1. Then the final matrix resulting form applying Gaussian elimina
to A is $\overline{M}A$.

In our code, the final matrix resulting is not in echelon form because its r
are not in correct order. By reordering the rows of $\overline{M}$, we can obtain a matrix
such that MA is a matrix in echelon form.

Moreover, since each of the matries M_k through M_1 is invertible, so is
product $\overline{M}$. Thus $\overline{M}$ is square are its rows are linearly independent. Since
obtained from $\overline{M}$ by reordering rows, M is also square and its rows are lin
independent. Informally, we have proved the following :

Proposition 3.3.1 : For any matrix A, there is an invertible matrix M such that MA is in echelon form.

3.3.1 Computing M Without Matrix Multiplications

Actually comptuing M does not require all these matrix multiplicaitons however. There is a much slicker approach. The procedure maintains two matrices, each represented by a row-list.

- The matrix undergoing the transformation, represented in our code by rowlist, and
- The transforming matrix, which we will represent in code by M_rowlist.

The algorihtm maintains the invariant that the trasnforming matrix times the input matrix equals the matrix represented by rowlist.

M_rowlist (initial matrix) = rowlist ...(3.3.1)

Performing the i^{th} row-addition operation consists in subtracting some multiple of one row of rowlist from another. This is equivalent to multiplying the matrix rowlist by a row-adition matrix M_i. To maintain the invariant (Equation 3.1), we multiply both sides of the equation by M_i.

M_i (M_rowlist) (intial matrix) = M_i (rowlist)

On the right-hand side, the procedure carries out, the oepration by subtracting a mltiple of the pivot row from another row.

What about on the left-hand side ? To update M_rowlist to be the product of M_i with M_rowlist, the procedure similarly carries out the row-addition operation on M_rowlist, subtracitng the same multiple of the corresponding row from the corresponding row.

Example 3.3.2 : Let's run through an example using the matrix.

$$A = \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix}$$

Ans. :

Initially, rowlist consist of the rows of A. To make the invariant

(Equation 3.3.1) true, the algorithm initializes M_rowlist to be the identity matrix. Now we have,

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix}$$

The first row-addition operation is to subtract twice the second row from th fourth row. The algorithm applies this operation to the transforming matrix (th first matrix on the left-hand side) and the matrix being transformed (the matri on the right hand side), resulting in :

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & -2 & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix}$$

Since the same matrix M_1 has multiplied the left-hand side and the rig hand side, the invariant is still true. The next row-addition operation is subtract three times the second row from the fifth row. The algorithm applies th operation to the transforming matrix and the matrix being transformed, resulti in :

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & -2 & & 1 & \\ & -3 & & & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

The third and final row-addition is to subtract two-thirds times the fourth from the fifth row. The algorithm must apply this operation to the transform matrix and the matrix being transformed. The fourth row of the transform matrix is [0 – 2 0 1 0], and two-thirds of this row is $\begin{bmatrix} 0 & -1\frac{1}{3} & 0 & \frac{2}{3} & 0 \end{bmatrix}$. fifth row of the transforming matrix is [0 – 3 0 0 1] and subtracting two-thi the fourth row yields $\begin{bmatrix} 0 & -1\frac{1}{3} & 0 & -\frac{2}{3} & 0 \end{bmatrix}$. Thus the equation becomes

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & -2 & & 1 & \\ & -1\frac{1}{3} & & \left(\frac{2}{3}\right) & \end{bmatrix} \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \\ 0 & 0 & 0 & 9 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad ...(3.3.2)$$

To incorporate this strategy into our code, we need to make two changes :

- Intialize the variable M_rowlist to represent the identity matrix (using 1 of GF2. one as appropriate) :

```
M_rowlist = [Vec(row_labels, {row_label_list[i] : one}) for i in range (m)]
```

- And, whenever a row-addition operation is performed on rowlist, perform the same row-adition operation on M-rowlist.
- Here's the main loop, which shows the second change :

```
for c in sorted (col_labels, key= hash) :
rows_with_nonzero = [r for r in rows_left if rowlist [r] [c] ! = 0]
if rows_with_nonzero ! = [ ]:
pivot = rows_with_nonzero [0]
rows_left, remove (pivot)
for r in rows_with_nonzero [1 : ] :
multiplier = rowlist [r] [c]/ rowlist [pivot] [c]
rowlist[r]–= multiplier* rowlist [pivot]
M_rowlist[r]- = multiplier * M_rolwist [pivot]
```

- To make this useful for solving the other problems mentioned in section 3.3, we need to finally produce the matrix M such that multiplying M by the input matrix gives a matrix in echelon form.
- Note that in Equaiton 3.3.2, the matrix in the right-hand side is not in echelon form because the rows are in the wrong order. Thus the matrix.

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ -2 & & & 1 \\ -1\frac{1}{3} & & & -\frac{2}{3} \end{bmatrix}$$

Represented by M-rowlist is not quite M. It has the correct rows but those rows need to be reordered.

Here is a simple way to get the rows in the correct order. Recall out initial efforts in sorting rows by position of the leftmost nonzero (section 3.1.3). There we accumulated the pivot rows in a list called new_rowlist. We use the same idea but this time, instead for accumulating the pivot rows, we accumulate the corresponding rows of M_rowlist in a list called new_M_rowlist :

```
new_M_rowlist = [ ]
for c in sorted (col_labels, key = hash) :
rows_with_nonzero = [r for r in rows_left if rowlist [r] [c] ! = 0]
if rows_with__nonzero [!] = [ ] :
pivot = rows _with_non zero [ 0 ]
rows_left.remove (pivot)
```

```
new_M_rowlist.append (M_rowlist [pivot] )
for r in rows_with_nonzero [1 : ] :
multiplier = rowlist [r] [c]/rowlist [pivot] [c]
rowlist [r] - = multiplier * rowlist [pivot]
M_rowlist [r] - = multiplier *M_rolwist [pivot]
```

One problem with this approach : it fails to append the rows of M_rowlist corresponding to zero rows of rowlist since no zero row becomes a pivot row. I therefore put another loop at the end to append these rows to new_M_rowlist :


```
for c in sorted (col_lables, key = hash) :
rows_with_nonzero = [r for r in rows_left if rowlist [r] [c] ! = 0]
if rows_with_nonzero ! [ ] :
pivot  = rows_with_nonzero [0]
rows_left. Remove (pivot)
new_M_rowlist. append (M_rowlist [pivot] )
for r in rows_with_nonzero[1 :] :
multiplier = rowlist [r][c] / rowlist [pivot] [c]
rowlist [r] – = multiplier* rowlist [pivot]
M_rowlist[r] – = multiplier*M_rowlist[pivot]
⇒ for r in rows_left : new_M-rowlist. Append (M_rowlist[r])
```

The module echelon contains a procedure transformation (A) that returns an invertible matrix M such that MA is in echelon form. It uses the above code.

Example 3.3.3 : Here is another example of maintaining the transforming matrix.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & -2.5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -6 & -6 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ .5 & -2 & 1 & 0 \\ 0 & -2.5 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 4 & 1 & 2 & 4 & 2 \\ 5 & 0 & 0 & 2 & 8 \end{bmatrix} \begin{bmatrix} 0 & 2 & 4 & 2 & 8 \\ 2 & 1 & 0 & 5 & 4 \\ 0 & -1 & 2 & -5 & -2 \\ 0 & -2.5 & 0 & -10.5 & -2 \end{bmatrix}$$

Syllabus Topic : Solving a Matrix-Vector Equation Using Gaussian Elimination

3.4 Solving a Matrix-Vector Equation Using Gaussian Elimination

- Suppose you want to solve a matrix-vector equation

$$Ax = b \quad \ldots(3.4.1)$$

- Compute an invertible matrix M such that MA is a matrix U in echelon form, and multiply both sides of Equation 3.3.3 by M, obtaining the quation.

$$MAx = Mb \quad \ldots(3.4.2)$$

- This shows that if the original equation (Equation 3.4.1) has solution u, the same solution satisfies the new Equation (3.4.2). Conversely, suppose u is a solution to the new Equation.
- We then have MAu = Mb. Multiplying both sides by the inverse M^{-1}, we obtain $M^{-1}MAu = M^{-1}Mb$, which implies Au = b, showing that u is therefore a solution to the original equation.
- The new equation M Ax = Mb is easier to solve than the original equation because the matrix MA on the left-hand is in echelon form.

3.4.1 Solving a Matrix-vector Equation when the Matrix is in Echelon form-the Invertible Case

Can we give an algorithm to solve a matrix-vector equation Ux = b where U is in echelon form?

Consider first the case in which U is an invertible matrix. In the case, U is square and its diagonal elements are nonzero. It is upper triangular, and we can solve the equation Ux = $\overline{b}$ using backward substitution, the alrogithm described and embodied in the procedure triangular_solve (A, b) define in the module triangular.

3.4.2 Coping with Zero Rows

Now consider the general case. There are two ways in which U can fail to be triangular:

(i) There can be rows that are all zero, and

(ii) There can be columns of U for which no row has its leftmost nonzero entry in this column.

The first issue is easy to cope with : just ignore zero rows.

Consider an equation $a_i x = b_i$ where $a_i = 0$

- If $b_i = 0$ then the equation is true regardless of the choice of x.
- If $b_i \neq 0$ then the equation is false regardless of the choice of x.

Thus the only disadvantage of ignoring the rows that are zero is that the algorithm will not notice if the equations cannot be solved.

3.4.3 Coping with Irrelevant Columns

Assume therefore that U has no zero rows. Consider the following example

	A	B	C	D	E
0	1		1		
1		2		3	
2				1	9

$$* [x_a\ x_b\ x_d\ x_e] = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

There are no nonzero rows. Every row therefore has leftmost nonzero entry. Discard every column c such that no row has a leftmost nonzero in that column. (In the example, we discard columns C and E). The resulting system looks like this:

	A	B	D
0	1		
1		2	3
2			1

$$* [x_a\ x_b\ x_d] = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

This system is triangular, so can be solved using backward substitution. The solution assigns numbers to the variables x_a, x_b, x_d. What about the variables x_c and x_e corresponding to discarded columns? We just set these to zero.

Using the linear-combinations definition of matrix-vector multiplication, the effect is that the discarded columns contribute nothing to the linear combination. This shows this assignment to the variables remains a solution when the discarded columns are reinserted.

In problems, you will write a procedure to try to find a solution to a matrix-vector equation where the matrix is in echelon form. A straightforward but somewhat cumbersome approach is to form a new matrix by deleting the zero rows and irrelevant columns and to then use triangular_ solve.

There is also a simpler, shorter, and more elegant solution; the code is a slightly modified version of that for triangular_solve.

3.4.4 Attacking the Simple Authentication Scheme, and improving it

The simple authentication scheme

(i) The password is an n-vector $\hat{x}$ over GF(2).

(ii) As a challenge, computer sends random n-vector a.

(iii) As the response, human sends back a. $\hat{x}$.

(iv) The challenge-response interaction is repeated until computer is convinced that Human knowns password x.

Eve cavesdrops on communication, and learns m, pairs $a_1, b_1, ..., a_m, b_m$ such that b_i is the correct response to challenge a_i. Then the password $\hat{x}$ is a solution to once rank A reaches n, the solution is unique, and Eve can use Gaussian elimination to find it, obtaining the password.

$$\underbrace{\begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix}}_{A} \begin{bmatrix} x \end{bmatrix} = \underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}}_{b}$$

Making the scheme more secure by introducing mistakes

The way to make the scheme more secure is to introduce mistakes.

- In about 1/6 of the rounds, randomly, Human sends the wrong dot-product.
- Computer is convinced if Human gets the right answers 75% of the time.

Even if Eve knows that Human is making mistakes, She doesn't know which rounds involve mistakes. Gaussian elimination does not find the solution when some of the right-hand side values b_i are wrong. In fact, we don't know any efficient algorithm Eve can use to find the solution, even if Eve observes many, many rounds. Finding an "approximate" solution to a large matrix-vector equation over GF (2) is considered a difficult computational problem.

In contrast, in the next couple of chapters we will learn how to find approximate solutions to matrix-vector equations over $\mathbb{R}$.

Syllabus Topic : Finding a Basis for the Null Space

3.5 Finding a Basis for the Null Space

Given a matrix A, we describe an algorithm to find a basis for the vector space $\{u : u * A = 0\}$. This is the null space of A^T.

The first step is to find an invertible matrix M such that MA = U is in echelon form. In order to use the vector-matrix definition of matrix-matrix multiplication, interpret M and U as consisting of rows.

$$\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \begin{bmatrix} A \end{bmatrix} = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix}$$

For each row u_i of U that is a zero vector, the corresponding row b_i of M has the property that $b_i * A = 0$.

Example 3.5.1 : Suppose A is the following matrix over GF (2) :

A =	A	B	C	D	E
a	0	0	0	one	0
b	0	0	0	one	one
c	one	0	0	one	0
d	one	0	0	0	one
e	one	0	0	0	0

Using transformation (A), we obtain the transforming matrix M such that

MA = U:

	a	b	c	D	e
0	0	0	one	0	0
1	one	0	0	0	0
2	one	one	0	0	0
3	0	one	one	One	0
4	one	0	one	0	one

*

	A	B	C	D	E
a	0	0	0	one	0
b	0	0	0	one	one
c	one	0	0	one	0
d	one	0	0	0	one
e	one	0	0	0	0

	A	B	C	D	E
0	One	0	0	one	0
1	0	0	0	one	0
2	0	0	0	0	one
3	0	0	0	0	0
4	0	0	0	0	0

Since rows 3 and 4 of the right-hand side matrix are zero vectors, rows 3 and 4 of M, the first matrix on the left-hand side, belong to the vector space {v = v * A = 0}. We will show that in fact these two vectors form a basis for that vector space.

Thus the second step of our algorithm is to find out which rows of U are zero, and select the corresponding rows of M.

To show that the selected rows of M are a basis for the vector space {u = u * A = 0}, we must prove two things:

(i) They are linearly independent, and

(ii) They span the vector space.

Since M is an invertible it is square and its columns are linearly independent. Therefore its rank equals the number of columns, which is the same as the number of rows. Therefore its rows are linearly independent.

Therefore any subset of its rows is linearly independent.

To show that the selected rows span the vector space {v : v * A = 0}, we take an oblique approach. Let s be the number of selected rows. These rows belong to the vector space and so their span is a subspace.

It we can show that the rank of the selected rows equals the dimension of the vector space, the Dimension principle, will show that the span of the selected rows in fact equals the vector space. Because the selected rows are linearly independent, their rank equals s.

Let m be the number of rows of A. Note that U has the same number of rows. U has two kinds of rows : nonzero rows and zero rows. We saw in section 3.1.1 that the nonzero rows form a basis for Row A, so the number of nonzero rows is rank A.

$$\begin{aligned} m &= (\text{number of non zero rows of U }) + (\text{ number of zero rows of U}) \\ &= \text{rank(A)} + s \end{aligned}$$

By the Rank-Nullity theorem (the matrix version of the kernel-Image theorem).

$$m = \text{rank A} + \text{nullity } A^T$$

Therefore $s = \text{nullity } A^T$.

Syllabus Topic : Factoring Integers

3.6 Factoring Integers

We begin with a equation from Gauss, writing more than two hundred years ago. The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic.

It has engaged the industry and wisdom of ancient and modern geometers to such an extent that it would be superfluous to discuss the problem at length. Further, the dignity of the science itself seems to require solution of a problem so elegant and so celebrated. (Carl Friedrich Gauss, Disquisitiones Arithmeticae, 1801)

Recall that a prime number is an integer greater than 1 whose only divisors are 1 and itself. A composite number is an integer greater than 1 that is not prime, i.e. a positive integer that has a divisor greater than one. A fundamental theorem of number theory is this.

Theorem 3.6.1 : (Prime Factorization theorem) : For every positive integer N, there is a unique set of primes whose product is N and this representation is unique.

Solution : For example, 75 is the product of the elements in the bag {3, 5, 5}, and 126 is the product of the elements in the bag {2, 3, 3, 7}, and 23 is the product of the elements in the bag {23}. All the elements in a bag must be prime. If N is itself prime, the bag for N is just {N}.

Factoring a number N means finding the corresponding bag of primes. Gauss really spoke of two problems: (1) Distinguishing prime numbers from composite numbers, and (2) factoring integers. The fist problem has been solved. The second, factoring, has not, although there has been tremendous progress in algorithms for factoring since Gauss's time.

In Gauss's day, the problems were of mathematical interest. In our day, primality and factorization lie at the heart of the RSA cryptosystem, which we use every day to securely transfer credit-card numbers and other secrets. In your web browser, when you navigate to a secure website.

The browser communicates with the server using the protocol HTTPS (Secure HTTP), which is based on RSA. To quote from a book by current-day expert, Bill Gates:

Because both the system's privacy and the security of digital money depend on encryption, a breakthrough in mathematics or computer science that defeats the cryptographic system could be a disaster. The obvious mathematical breakthrough would be the development of an easy way to factor large prime numbers (Bill Gates, The Road Ahead, 1995).

Okay, Bill got it slightly wrong. Factoring a large prime number is easy. Don't worry – this was corrected in the next release of his book.

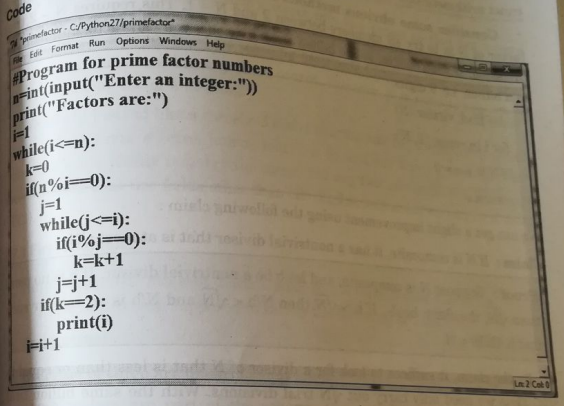
Factoring a composite number N into primes isn't the hard part. Suppose you had an algorithm factor(N) that, given a composite number N, found any integers a and b bigger than 1 such that N = ab. You could then obtain the prime factorization by recursively factoring a and b :

```
def prime_factorize (N) :
if is_prime (N) :
return [N]
a, b = factor (N)
return prime_factorize (a) + prime_factorize (b)
```

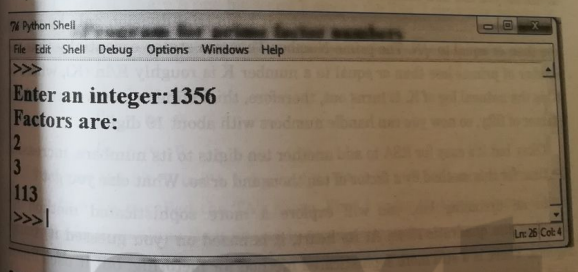

the challenge is implementing factor (N) to run quickly.

The Python code for prime factorization is as follows

Code



Output



3.6.1 Fist Attempt at Factoring

Let's consider algorithm that involve trial divisions. A trial division is testing, for a particular integer b, whether N is divisible by b. Trial division is not a trivial

operation – it's much slower than operations on floats because it has to be done to exact arithmetic-but it's not too bad.

Consider some obvious methods : the most obvious method of finding a factor of N is to try all members between 2 and N – 1. This requires N – 2 trial divisions. If your budget, is one billion tiral divisions, you can factor numbers up to a billion, i.e. 9 digits.

```
def find_visiosr (N)
for i in range (2, N):
if N % i = = 0 ;
return i
```

We can get a slight improvement using the following claim :

Claim : If N is composite, it has a nontrivial divisor that is at most $\sqrt{N}$.

Proof : Suppose N is composite, and let b be a nontrivial divisor. If b is no more than $\sqrt{N}$, the claim holds. If $b > \sqrt{N}$ then $N/b < \sqrt{N}$ and N/b is an integer such that b. (N/b) = N.

By the claim, it suffices to look for a divisor of N that is less than or equal to $\sqrt{N}$. Thus we need only carry out $\sqrt{N}$ trial divisions. With the same billion trial divisions, you can now handle numbers up to a billion squared, i.e. 18 digits.

The next refinement you might consider it to do trial division only by primes less than or equal to $\sqrt{N}$. The prime Number Theorem states essentially that the number of primes less than or equal to a number K is roughly K/ln (K), where ln (K) is the natural log of K. It turns out, therefore, this refinement saves you about a factor of fifty, so now you can handle numbers with about 19 digits.

Okay but it's easy for RSA to add another ten digits to its numbers, increasing the time for this method by a factor of ten thousand or so. What else you got ?

In an upcoming lab, you will explore a more sophisticated method for factoring, the quadratic sieve. At its heart, it is based on (you guessed it) linear algebra. There is a still more sophisticated method for factoring, but it uses linear algebra in a similar way.

3.7 Lab : Threshold Secrete-Sharing

We had a method for splitting a secret into two pieces so that both were required to recover the secret. The method used GF(2). We could generalize this to split the secret among, say four teaching assistants (TAs), so that jointly they could recover the secret but any three cannot. However, it is risky to rely on all four TAs showing up for a meeting.

We would instead like a threshold secret sharing scheme, a scheme by which, say, we could share a secret among four TAs so that any three TAs so that any three TAs could jointly recovery the secret, but any two TAs could not. There are such schemes that use fields other than GF (2), but let's see if we can do it using GF(2).

3.7.1 First Attempt

Here's (doomed) attempt I work with five 3-vectros over GF(2) : a_0, a_1, a_2, a_3, a_4. These vectors are supposed to satisfy the following requirement:

Requirement

Every set of three are linearly independent.

These vectors are part of the scheme; they are known to everybody. Now suppose I want to share a one-bit secret s among the TAs. I randomly select a 3-vector u such that $a_0 \cdot u = s$.

$$\beta_1 = a_1 \cdot u$$
$$\beta_2 = a_2 \cdot u$$
$$\beta_3 = a_3 \cdot u$$
$$\beta_4 = a_4 \cdot u$$

Now I give the bit β_1 to TA 1, I give β_2 to TA 2, β_3 to TA 3, and β_4 to TA 4. The bit given to a TA is called the TA's share. First I argue that this scheme allows any three TAs to combine their shares to recover the secret.

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

The three TAs know the right-hand side bits, so can construct this matrix-vector equation.

Since the vectors a_1, a_2, a_3 are linearly independent, the rank of the matrix is three, so the columns are also linearly independent. The matrix is square and its columns are linearly independent, so it is invertible, so there is a unique solution. The solution must therefore be the secret vector u. The TAs use to solve the recover u, and take the dot-product with a_0 to get the secret s.

Similarly, any three TAs can combine their shares to recover the secret vector u and thereby get the secret.

Now suppose two rogue TAs, TA1 and TA 2, decide they want to obtain the secret without involving either of the other TAs. They know β_1 and β_2. Can they use these to get the secret s ?

The answer is no: their information is consistent with both s = 0 and s = 0ne: since the matrix

$$\begin{bmatrix} a_0 \\ \hline a_1 \\ \hline a_2 \end{bmatrix}$$

is invertible, each of the two matrix equations

$$\begin{bmatrix} a_0 \\ \hline a_1 \\ \hline a_2 \end{bmatrix} \begin{bmatrix} x_0 \\ xvec_1 \\ xvec_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ \hline a_1 \\ \hline a_2 \end{bmatrix} \begin{bmatrix} x_0 \\ xvec_1 \\ xvec_2 \end{bmatrix} = \begin{bmatrix} one \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

Has a unique solution. The solution to the fist equation is a vector v such that $a_0 \cdot u = 0$, and the solution to the second equation is a vector v such that $a_0 \cdot v = one$.

3.7.2 Scheme that Works

So the scheme seems to work. What's the trouble?

The trouble is that there are no five 3-vectors satisfying the requirement. There are just not enough 3-vectors over GF (2) to make it work.

Instead, we go to bigger vectors. We will seek ten 6-vectors a_0, b_0, a_1, b_1, a_2, b_2, a_3 b_3, a_4 b_4 over GF(2). We think of them as forming five pairs:

(i) Pair 0 consists of a_1 and b_0, (ii) Pair 1 consits of a_1 ad b_1,

(iii) Pair 2 consits of a_2 ad b_2, (iv) Pair 3 consits of a_3 ad b_3, and

(v) Pair 4 consits of a_4 ad b_4.

The requirement is as follows :

Requirents

For any three paris, the corresponding six vectros are linearly indepenent.

To use this scheme to share two bits s and t, I choose a secret 6-vector u such that $a_0 \cdot u = 3$ and $b_0 \cdot u = t$. I then give TA 1 the two bits $\beta_1 = a_1 u$ and $\gamma_1 = b_1 u$, I give TA 2 the two bits $\beta_2 = a_2 \cdot u$ and $\gamma_2 = b_2 \cdot u$, and so on. Each TA's share thus consist of a pair of bits.

Syllabus Topic : Inner Product

3.8 Inner Product

3.8.1 The Inner Product for Vectors Over the Reals

We first deal with basic definitions needed to understand inner product.

Consider a vector $\bar{v} = (v_1, v_2, ..., v_n)$ in $\mathbb{R}^n$.

Then we define norm of vector $\bar{v}$ as

$$||v|| = \sqrt{v_1^2 + v_2^2 + ... + v_n^2}$$

We read $||v||$ as norm v.

Properties of $\|\cdot\|$

1. For any vector v, $||v||$ is a nonnegative real number.
   $\therefore ||v|| \geq 0$
2. For any vector v, $||v|| = 0$ iff $v = 0$
3. if α is any scalar $||\alpha v|| = |\alpha| \cdot ||v||$
4. If u and v are any vectors,
   $||u + v|| \leq ||u|| + ||v||$
   (This inequality is called triangle inequality)

If $v = (v_1, v_2, ..., v_n)$, often we

Use $||v||^2 = v_1^2 + v_2^2 + ... + v_n^2$ form of the definition.

We know define inner product

Syllabus Topic : Orthogonality

3.8.2 Orthogonality

We all know Pythagoras theorem for a right angled triangle: if a, b, c are the sides and c is the hypotenuse then

$$c^2 = a^2 + b^2$$

In this case we say a is perpendicular to be or b is perpendicular to a.

Now in terms of vectors we shall see the same notion as orthoganality.

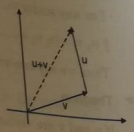


Fig. 3.8.1

We find the condition of (perpendicularity i.e. orthoganality for vector u and v.

$$\| u + v \|^2 = < u + v,\ u + v >$$
$$= < u + v, u > + < u + v, v >$$
$$= < u, u > + <v, u> + <u, v> + <v + v>$$
$$= <u, u> + < u, v > + < u, v > + < v, v >$$
$$= \| u \|^2 + 2 < u, v > + \| v \|^2$$

Thus, $\| u + v \|^2 = \| u \|^2 + \| v \|^2$ iff $2 < u, v > = 0$

i.e. $< u, v > = 0$

Lemma

If u and v are orthogonal vectors then for α, β any scalar we have

$$\| \alpha u + \beta v \|^2 = \alpha^2 \| u \|^2 + \beta^2 \| v \|^2$$

Proof

Consider, $\| \alpha u + \beta v \|^2 = < \alpha u + \beta v,\ \alpha u + \beta v >$

$$= < \alpha u,\ \alpha u + \beta v > + <\beta v,\ \alpha u + \beta v >$$
$$= < \alpha u, \alpha u > + <\alpha u, \beta v > + <\beta v, \alpha u > + <\beta v, \beta v >$$
$$= \alpha^2 \| u \|^2 + \alpha\beta < u, v> + \beta < u, v > + \beta^2 \| v \|^2$$

Now, u, v are orthogonal

$\therefore < u, v> = 0 = < v, u>$

Inner product

It is a rule that takes every pair of vectors to a real number. The rule is denoted as $< \cdot ; \cdot >$ (in mathematics)

It satisfies the following properties.

If u, v, w are vectors over $\mathbb{R}$ and α is any scalar i.e. $\alpha \in \mathbb{R}$.

Then

(i) $< u , u > \geq 0$ (Non-negativity)

(ii) $< u, u > = 0$ iff $u = 0$

(iii) $< u, v> = < v, u >$ (symmetry)

(iv) $< u + w, v > = < u, v > + < w, v >$

(v) $<u, w + v > = < u, w > + < u, v >$

(iv) & (v) are linearity property of inner product.

(vi) $< \alpha u, v > = \alpha < u, v >$ (homogeneity)

(Now the set of vectors on which the inner product rule is defined is known as inner product space). For vectors over $\mathbb{R}$, the standard inner product is often called as the dot product or scalar product..

i.e. $< u, v > = u \cdot v$

$\therefore \| v \| = \sqrt{< v, v >} \Rightarrow \| v \|^2 = < v, v > = v \cdot v$

Note there are various inner products that are defined as per the set of vectors we choose. For Example if we choose our vectors as matrices of order n whose entries are real number then we define inner product in this case as $< A, B > = \text{trace } (A^t B)$

In higher dimensions the norm notion acts like distance between two vectors. Distance between two vectors u and v.

$$= \| u - v \|$$
$$= < u - v,\ u - v >$$
$$= [(u - v) \cdot (u - v)]^{1/2}$$
$$= \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \ldots\ldots + (u_n - v_n)^2}$$

$\therefore \| \alpha u + \beta v \|^2 = \alpha^2 \| u \|^2 + {}^2 \| v \|^2$

☞ **Result**

Suppose $v_1, v_2, \ldots\ldots, v_n$ are mutually orthogonal

i.e. $< v_i, v_j > = 0$ for $i \neq j$

For any scalar $\alpha_1, \alpha_2, \ldots\ldots, \alpha_n$

$\| \alpha_1 v_1 + \alpha_2 v_2 + \ldots\ldots + \alpha_n u_n \|^2 = \alpha_1^2 \| v_1 \|^2 + \alpha_2^2 \| v_2 \|^2 + \ldots\ldots + \alpha_n \| v_n \|^2$

Thus we say two vectors u and v are orthogonal iff $< u, v> = 0$

☞ **Theorem**

If u and v are vectors over real numbers and they are orthogonal then

$\| u + v \|^2 = \| u \|^2 + v \|^2$

3.8.3 Properties of Orthogonality

1. If $< u, v> = 0$ then for any scalar $\alpha \in \mathbb{R}$, $<\alpha u, \alpha v> = 0$

☞ **Proof**

Consider, $<\alpha u, \alpha v>$

$= \alpha < u, \alpha v >$

$= \alpha < \alpha v , u>$ $\quad (\because < . \ ; \ . > \text{ is symmetric})$

$= (\alpha) (\alpha) < v, u>$

$= \alpha^2 < u, v >$

$= \alpha^2 (0) = 0$

$\therefore \ <\alpha u, \ \alpha v> = 0$

$\therefore$ αu and αv are orthogonal.

2. If u and v are both orthogonal to w then u + v is orthogonal to w.

☞ **Proof**

Given $< u , w> = 0$ and $< v, w > = 0$ to show $< u + v, w > = 0$

Consider, $< u + v, w> = < u, w> + <v, w> = 0 + 0 = 0$

☞ **Decomposition of a vector into parallel and perpendicular components**

Let us first understand to concept.

Let w be an element of v such that $\| w \| \neq 0$.

For any vector v there exist a unique number c

Such that v – cw is perpendicular to w.

i.e. $< v - cw , \ w > = 0$

$\therefore \ < v, w> + <- cw, w > = 0$

$< v, w > - \ c < w, w > = 0$

$\therefore \ < v, w > = c < w, w > \Rightarrow \ c = \frac{< v , w >}{< w, w >}$

The c is called the component of v along w and cw is the projection of v along w (Sometimes c is also known as Fourier coefficient).

This picture represents it.

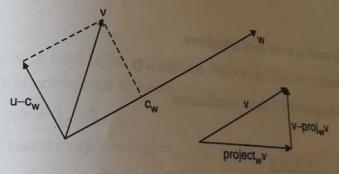


Fig. 3.8.2

Now we define

For any vector b and any vector v, $b^{\|v}$ and $b^{\perp v}$ to be the projection of b along v and the projection of b orthogonal to v. respectively

If $b = b^{\|v} + b^{\perp v}$

and for some scalar $\sigma \in \mathbb{R}$, $b^{\|v} = \sigma v$ and $b^{\perp v}$ is orthogonal to u.

Let, $b^{\|v} = X$ and $b^{\perp v} = Y$

then $b = X + Y$

Now, compare it with above concept the vector cw is the $b^{\|v}$ and $v \cdot cw$ is $b^{\perp v}$.

☞ **Lemma**

Let b and v be vectors. The point in span { v } closest to b is $b^{\|v}$ and the distance is $\| b^{\perp v} \|$.

Finding the projection and the closest point

We noted that, $< v - cw,\ w > = 0$

Thus converting it 0 notation of definition we get, $< b - b^{\|v},\ v > = 0$

Also, $b^{\|v} = \sigma u$

$\therefore \quad < b - \sigma v,\ v > \ = 0$

$\therefore \quad < b, v > - \sigma < v, v > = 0$

$$\sigma = \frac{< b, v >}{< v, v >}$$

If $\| v \| = 1$, $\sigma = < b, v >$

Thus,

Lemma

For any vector b and u over real numbers.

(1) There is a scalar σ such that $< b - \sigma v, v > = 0$

(2) The point p in span { v } that minimizes

$\| b - p \|$ is σ v

(3) The value of σ is $\frac{< b, v >}{< v, v >}$.

– We explain the above concept using two important theorems in mathematics.

Consider, $v_1, v_2, \ldots\ldots, v_n$ mutually perpendicular vectors

Let c_i be the component of v along v_i then $v - c_i v_i - \ldots\ldots\ldots\ldots - c_n v_n$ is perpendicular to $v_1, v_2, \ldots\ldots, v_n$

To see this, we take the inner product with v_j for any v. Now all terms involving $< v_i, v_j > = 0$ if $i \neq j$

Only terms remaining will be $< v, v_j > - c_j < v_j, v_j >$

Which cancel. Thus subtracting linear combinations as above orthogonalizes v w.r.t. $v_1, v_2, \ldots\ldots, v_n$. The next theorem shows that $c_1 v_1 + c_2 v_2 + \ldots\ldots + c_n v_n$ gives the closest approximation to v as a linear combination on of $v_1, v_2, \ldots\ldots, v_n$.

Theorem

Let $v_1, v_2, \ldots\ldots, v_n$ be vector which are mutually perpendicular such that $\| v_i \| \neq 0$ for all i. Let v be any vector and c_i be the component of v along v_i. Let $a_1, \ldots\ldots, a_n$ be scalars then

$$\left\| v - \sum_{k=1}^{n} c_k v_k \right\| \leq \left\| v - \sum_{k=1}^{n} a_k v_k \right\|$$

Theorem (Bessel's inequality)

If, $v_1, v_2, \ldots\ldots, v_n$ are mutually perpendicular unit vectors and if c_i is the component of v along v_i then $\sum_{i=1}^{n} c_i^2 \leq \| v \|^2$. (without proof)

Definition

(i) A vector v is said to be orthogonal to set of vectors $V = \{v_1, v_2, \ldots\ldots, v_n, \ldots\ldots\}$ if it is orthogonal to every vector $v_i \in V$.

i.e. $< v, v_i > = 0$, $\forall i$

(ii) A set of vectors say $S = \{w_1, w_2, \ldots\ldots\ldots, w_n, \ldots\ldots\ldots\}$ is said to orthogonal set if

$< w_i, w_j > = 0$ for $i \neq j$ and

$< w_i, w_j > \neq 0$ for $i = j$

Lemma

A vector v is orthogonal to each of the vectors $a_1, a_2, \ldots\ldots, a_n$ if and only if it is orthogonal to every vector in span $\{a_1, a_2, \ldots\ldots\ldots, a_n\}$.

Proof

Let $w \in$ span $\{ a_1, a_2, \ldots\ldots, a_n \}$

$\therefore \quad w = \alpha_1 a_1 + \alpha_2 a_2 + \ldots\ldots\ldots + \alpha_n a_n$, for $\alpha_i \in \mathbb{R}$

Now, since v is orthogonal to each a_i

$\therefore \quad < v, a_i > \ 0$, $i = 1, 2, \ldots\ldots, n$

Now, consider $< v, w >$

$$= < v, \alpha_1 a_1 + \alpha_2 a_2 + \ldots\ldots\ldots + \alpha_n a_n >$$

$$= < v_1, \alpha_1 a_1 > + < v_1 \alpha_2 a_2 > + \ldots\ldots + < v, \alpha_n a_n >$$

$= \alpha_1 < v, a_1 > + \alpha_2 < v, a_2 > + \ldots\ldots + \alpha_n < v, a_n >$

$= \alpha_1 \cdot 0 + \alpha_2 \cdot 0 + \ldots\ldots + \alpha_n \cdot 0 = 0$

$\therefore$ $< v, w > = 0 \Rightarrow$ v and w are orthogonal.

Now, w is any arbitrary vector $\in$ Span $\{ a_1, \ldots\ldots, a_n \}$

$\therefore$ C suppose v is orthogonal to every vector in span $\{ a_1, a_2, \ldots\ldots, a_n \}$

Since the span includes $a_1, a_2, \ldots\ldots, a_n$

We infer that v is orthogonal to $a_1, a_2, \ldots\ldots a_n$.

Lemma

Let V be a vector space over F and let b be a vector. The point in V closest to be b is $b^{\|v}$ and the distance is $\| b^{\perp v} \|$.

Proof Distance between b and $b^{\|v}$ is

$$\| b - b^{\|v} \| = \| b^{\perp v} \|$$

Let P be any point in V.

We show that P is no closer to b than $b^{\|v}$.

Consider, $b - p = b - b^{\|v} + b^{\|v} - P$

$$b - p = b^{\perp v} + b^{\|v} - P$$

Now by Pythagoras theorem,

$$\| b - p \|^2 = \| b^{\perp v} \|^2 + \| b^{\|v} - p \|^2$$

$\Rightarrow$ $\| b - p \|^2 > \| b^{\perp v} \|$

$\therefore$ $\| b - p \|^2 > \| b - b^{\|v} \|^2$

$\therefore$ $\| b - p \| > \| b - b^{\|v} \|$ if $p \neq b^{\|v}$

- First attempt at projecting orthogonal vectors to a list of vectors

Lemma

- (Loop invariant for project_orthogonal)

Let, k = len (v list). For i = 0, ..., k

Let b_i be the value of the variable b after the iterations. Then

b_i is the orthogonal to the first vector of v list

$b\text{-}b_i$ is in the span of the first i vectors of v list.

Proof

The proof is by induction principle on i

For i = 0,

the claim is true,

b_0 is orthogonal to each of the first 0 vectors and $b - b_0$ is in the span of the first 0 vectors because $b - b_0$ is the zero vector.

Assume that the claim holds for i – 1 iterations.

We prove it holds for i iterations

We write v list as $[v_1, v_2, \ldots\ldots, v_k]$

In the i^{th} iteration, the procedure computes

$b_i = b_{i-1}$ – project – along (b_{i-1}, v_i)

We can write $b_i = b_{i-1} - \alpha_i v_i$

Where, $\alpha_i = \dfrac{< b_{i-1}, v_i >}{< v_i, v_i >}$

The induction hypothesis stages that b_{i-1} is the projection of b_0 orthogonal to the first i – 1 vectors. We need to prove that b_i is orthogonal to each vector in $[v_1, v_2, \ldots\ldots, v_{i-1}, v_i]$

Now the choice of α_i ensures that b_i is orthogonal to v_i. We prove that $<b_i, v_j> = 0$ for $j < i$

$$<b_i, v_j> = < b_{i-1} - \alpha_i v_i, v_j >$$

$$= <b_{i-1}, v_j> - \alpha_i < v_i, v_j >$$

$$= 0 - \alpha_i < v_i, v_j >$$

$$= 0 - \alpha_i 0 = 0$$

We also need to show that $b_0 - b_i$ is in the span of the first i vectors of v list. By the inductive hypothesis $b_0 - b_i - 1$ is in the span of the first i – 1 vector.

i.e. $b_i - b_i = b_0 - (b_{i-1} - \alpha_i v_i)$

$$= (b_0 - b_{i-1}) + \alpha_i v_i$$

Now $b_0 - b_i$ is a vector in the span of first i – 1 vectors.

$\therefore$ $b_0 - b_i$ is a vector in the span of first i vectors

Hence, the claim.

Syllabus Topic : Building an Orthogonal set of Generators

3.9 Building an Orthogonal set of Generators

In this case we consider the input of $[v_1, v_2, \ldots\ldots, v_n]$ vectors over reals and an output of vectors will be $\{ v_1^*, v_2^*, \ldots\ldots, v_n^* \}$

Which are mutually orthogonal.

- This procedure is same as Gram schmidt orthogalization method.
- Projection of vector b_1 along vector v_1 is given by the formula,

$$\text{Proj}_{v_1}(b_1) = \frac{< b_1, v_1 >}{< v_1, v_1 >} \cdot v_1$$

$b_1 = [1, 1]$, $v_1 = [1, 0]$

Examples

1. Find projection of [1, 1] along [1, 0]

$$\text{Proj}_{v_1}(b_1) = \frac{< b_1, v_1 >}{< v_1, v_1 >} \cdot v_1$$

$$= \frac{((1, 1) \cdot (1, 0))}{1^2 + 0^2}(1, 0) = \frac{1}{1} \cdot (1, 0) = (1, 0)$$

Now graphically if we see.

It is nothing but the shadow of the vector (1, 1) on X-axis.

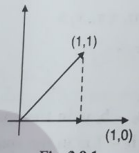


Fig. 3.9.1

2. Find the projection of [0, 1] along $\left[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right]$

Let, $b_1 = [0, 1]$, $v_1 = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$

$$\text{Proj}_{v_1}(b_1) = \frac{< b_1, v_1 >}{< v_1, v_1 >} \cdot v_1$$

$$= \frac{< (0, 1), \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) >}{< \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) >} \cdot \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$$

$$= \frac{\left(\frac{\sqrt{2}}{2}\right)}{\left(\frac{2}{4} + \frac{2}{4}\right)}\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) = \frac{\left(\frac{\sqrt{2}}{2}\right)}{\left(\frac{4}{4}\right)} \cdot \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$$

$$= \left(\frac{2}{4}, \frac{2}{4}\right) = \left(\frac{1}{2}, \frac{1}{2}\right)$$

Mutually orthogonal vectors

Suppose $\{ v_1, v_2, \ldots\ldots, v_n \}$ are vectors. We wish to construct orthogonal set of vectors for the vectors space v. Let that set of orthogonal vectors be $\{ w_1, w_2, \ldots\ldots w_n \}$ then,

Step 1 : Set $w_1 = v_1$

Step 2 : $w_2 = v_2 - \frac{< v_2, w_1 >}{< w_1, w_1 >} \cdot w_1$

Step 3 : $w_3 = v_3 - \frac{< v_3, w_1 >}{< w_1, w_1 >} \cdot w_1 - \frac{< v_3, w_2 >}{< w_2, w_2 >} w_2$

:

:

Step n : $w_n = v_n - \sum_{i=1}^{n-1} \cdot \frac{< v_n, w_i >}{< w_i, w_i >} \cdot w_i$

This process gives us n orthogonal vectors for v if $\{ v_1, v_2, \ldots\ldots, v_n \}$ are basis vectors then. $\{ w_1, w_2, \ldots\ldots, w_n \}$ so obtained would be orthogonal basis vectors.

(In linear Algebra, this process is known as Gram Schmidt Orthogonalisation process / Method) GSO Method

Python code for Gram Schmidt Orthogonalisation process.

Code

Gram Schmidt.py

```
import numpy as np
def gs(A):
    m = np.shape(A)[0]
    n = np.shape(A)[1]
    Q =  np.zeros((m, m))
    R =  np.zeros((n, n))
    print (m,n,Q,R)
    for j in range(n):
        v = A[:,j]
        for i in range(j):
            R[i,j] =  np.dot(Q[:,i].T , A[:,j])
            v = v.squeeze() - (R[i,j] * Q[:,i])
        R[j,j] =  np.linalg.norm(v)
        Q[:,j] = (v / R[j,j]).squeeze()
    return Q, R

As= np.random.rand(2,3,3)
print (As)

for A in As:
    print (gs(A))
```

Output

```
In [10]: %run "C:/Users/Administrator/Downloads/Gram Schmidt.py"
[[[ 0.13786     0.46691465  0.04459223]
  [ 0.1460094   0.27090044  0.59119834]
  [ 0.43313392  0.41828851  0.46005729]]

 [[ 0.24555824  0.55867011  0.72918265]
  [ 0.46118183  0.6444158   0.76522528]
  [ 0.78546914  0.87643908  0.18551725]]]
3 3 [[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]] [[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]
(array([[ 0.28876089,  0.88886686, -0.35571456],
       [ 0.38583057,  0.26643733,  0.9140453 ],
       [ 0.90724022, -0.37272892, -0.19490598]]), array([[ 0.47741923,  0.59716448,  0.61106549],
       [ 0.        ,  0.33129473,  0.0256772 ],
       [ 0.        ,  0.        ,  0.43485204]]))
3 3 [[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]] [[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]
(array([[ 0.26029863,  0.88559976, -0.3846527 ],
       [ 0.48886568,  0.22266233,  0.84346419],
       [ 0.83261934, -0.40759608, -0.37498063]]), array([[ 0.94337124,  1.19019397,  0.71836287],
       [ 0.        ,  0.28101211,  0.74053472],
       [ 0.        ,  0.        ,  0.29539267]]))
```


Example

1. Consider vectors $v_1 = (1, 1, 1, 1)$
$v_2 = (1, 2, 4, 5)$
$v_3 = (1, -3, -4, -2)$

Construct an orthogonal set of generators for subspace of $\mathbb{R}^4$ whose generators are v_1, v_2, v_3}

Ans. :

$$w_1 = v_1 = (1, 1, 1, 1)$$

$$w_2 = v_2 - \frac{< v_2, w_1 >}{< w_1, w_1 >} \cdot w_1$$

$$w_2 = v_2 - \frac{12}{4} w_1 = (-2, -1, 1, 2)$$

$$w_3 = v_3 - \frac{< v_3, w_1 >}{< w_1, w_1 >} \cdot w_1 - \frac{< v_3, w_2 >}{< w_2, w_2 >} \cdot w_2$$

$$= v_3 - \frac{(-8)}{4} w_1 - \frac{(-7)}{10} w_2$$

$$w_3 = \left(\frac{8}{5}, \frac{-17}{10}, \frac{-13}{10}, \frac{7}{5} \right)$$

Solving Closest point in the span of many vectors

Find the vector in span $\{ v_1, v_2, \ldots v_n \}$ that is closest to b.

We know that $b^{\| v}$ the projection of b onto v is the span $\{v_1, \ldots, v_n\}$ which is $b - b^{\perp v}$ where $b^{\perp v}$ is the projection of b orthogonal to v.

There are two equivalent ways to find $b^{\perp v}$.

One method

First apply orthogonalize to $v_1, v_2, \ldots, v_n$ and obtain $v_1^*, v_2^* \ldots, v_n^*$.

Secondly call,

Project orthogonal (b, $[v_1^*, v_2^*, \ldots, v_n^*]$) and obtain $b^{\perp v}$ as the result.

Second method

Exactly the same computations take place when orthogonalize applied to $[v_1, v_2, ..., v_n, b]$ to obtain $[v_1^*, v_2^*, ..., v_n^*, b^*]$. In last iteration of orthogenalize the vector b^* is obtained by projecting b orthogonal to $v_1^*, v_2^*, ..., v_n^*$. Thus $b^* = b^{\perp v}$.

Then $b^{/\!/v} = b - b^{\perp v}$ is close to b in span $\{v_1, v_2, ..., v_n\}$

Consider the question of finding a vector p which closest to vector b = [5, – 5, 2] from span $\{v_1, v_2\}$, where $v_1 = [8, -2, 2]$ $v_2 = [4, 2, 4]$

Consider the set $\{v_1, v_2\}$,

$$\langle v_1, v_2 \rangle = v_1 \cdot v_2 = (8)(4) + (-2)(2) + (2)(4) \neq 0$$

$\therefore$ v_1 and v_2 are not orthogonal.

We find orthogonal set from v_1 and v_2 using GSOM

Let $w_1 = v_1 = (8, -2, 2)$

$$w_2 = v_2 - \frac{\langle v_2, w_1 \rangle}{\langle w_1, w_1 \rangle} \cdot w1$$

$$w_2 = (4, 2, 4) - \frac{(32 - 4 + 8)}{64 + 4 + 4}(8, -2, 2)$$

$$= (4, 2, 4) - \frac{36}{72}(8, -2, 2) = (4, 2, 4) - (4, -1, 1)$$

$$w_2 = (0, 3, 3)$$

Now calculate

$$c_1 = \frac{\langle b, w_1 \rangle}{\langle w_1, w_1 \rangle} = \frac{40 + 10 + 4}{64 + 4 + 4} = \frac{54}{72} = \frac{3}{4}$$

$$c_2 = \frac{\langle b, w_2 \rangle}{\langle w_2, w_2 \rangle} = \frac{0 - 15 + 6}{9 + 9} = \frac{-9}{18} = \frac{-1}{2}$$

$$P = b^v = c_1 w_1 + c_2 w_2 = \frac{3}{4}(8, -2, 2) - \frac{1}{2}(0, 3, 3) = (6, -3, 0)$$

Syllabus Topic : Orthogonal Complement

3.10 Orthogonal Complement

Let v be a (inner product space) vector spaced over R. Let S be a subspace of V. We define set $S^\perp$ is the orthogonal complement of S as a collection of all those vector in V which are orthogonal to every j vector is S.

i.e. $S^\perp = \{v \in V \mid \langle v, s \rangle = 0, \forall s \in S\}$

Note that $S^\perp$ is a sub space of V.

Proof : Now $0 \in S^\perp$ because $\langle 0, s \rangle = 0, \forall s \in S$.

Let $v_1, v_2 \in S^\perp$,

Consider let S be arbitrary

$$\langle v_1 + \alpha v_2, s \rangle = \langle v_1, s \rangle + \langle \alpha v_2, s \rangle$$

$$= \langle v_1, s \rangle + \alpha \langle v_2, s \rangle$$

$$= 0 + 0 \quad (\because v_1, v_2 \in S)$$

$\therefore \langle v_1 + \alpha v_2, S \rangle = 0$

Hence $S^\perp$ is a subspace.

We state important theorem.

Theorem :

If $S^\perp$ is the orthogonal complement of S where S is subspace of V. Then

$$V = S \oplus S^\perp \text{ and } S \cap S^\perp = \{0\}$$

We fist prove $S \cap S^\perp = \{0\}$

Proof $S^\perp = \{v \in V \mid \langle v, s \rangle = 0, \forall s \in S\}$ P

Now $S \subseteq V \Rightarrow \langle s, s \rangle = 0$

$$\Leftrightarrow s = 0$$

$$\therefore S \cap S^\perp = \{0\}$$

$$S \oplus S^\perp = \{s + \tilde{s} \mid s \in S \text{ and } \tilde{s} \in S^\perp\}$$

Let $S \oplus S^\perp \subseteq V$ is obvious

Because $S \subseteq V$ and $S^\perp \subseteq V$ also.

$s + \tilde{s} \in S \oplus S^\perp$ is also in V.

$\therefore S \oplus S^\perp \subseteq V$

Now for any b in V, $b = b^{/\!/}s + b^{\perp s}$

$b^{/\!/} \in S$ and $b^{\perp s} \in S^\perp$.

$\therefore b \in S \oplus S^\perp$

$\therefore V \subseteq S \oplus S^\perp$

Hence, $V = S \oplus S^{\perp}$

3.10.1 Computing the Orthogonal Complement Algorithm

Definition find orthogonal – complement (S_basis, V_basis) "Given a basis S_basis for S and V_ basis for V.

Return a basis for orthogonal complement of S w.r.t V.

$[s_1^*, s_2^*, ..., s_k^*, v_1^*, v_2^*, ... v_n^*]$= othroganalize (S_basis, V_basis)

Return [v_i for c in {1, ..., n}] if w_i^* is not the zero vector.

Syllabus Topic : Eigen Vector - Modeling Discrete Dynamic Processes

3.11 Eigen Vector & Eigen values

Before we begin this topic we recall that for a square matrix $A_{n \times n}$, we define trace (A) = $\sum_{i=1}^{n} a_{ii}$,i.e. the total of diagonal elements and det (A) is the determinant of matrix A.

Let $AX = \lambda X$, where $A_{n \times n}$ and $X_{n \times 1}$ and $\lambda \in \mathbb{F}$.

Then $AX - \lambda X = 0$,

$(A - \lambda I) X = 0$ where I is identity matrix

Now $X \neq 0$ $\quad \therefore \det (A - \lambda I) = 0$

This is known as characteristic equation which of degree 'n' solving this equation we get n roots which are known as eigen values / characteristic roots/ latent roots.

For each eigen value λ_i, we solve $(A - \lambda_i I) X_i = 0$ and obtain x_i which is its corresponding eigen vector. The set of eigen vector for matrix A along with zero vector is known as eigen space. Eigen space is a vector space, generally denoted as E_c .(Thus we can find eigen values and eigen vectors only when we are dealing with square matrices.)

We explain the entire concept using an Example

Consider $A = \begin{bmatrix} 12 & -51 \\ 2 & -11 \end{bmatrix}$

Let $AX = \lambda X, \quad X \neq 0$

Now $\det (A - \lambda I) = 0$

$$\begin{vmatrix} 12-\lambda & -51 \\ 2 & -11-\lambda \end{vmatrix} = 0$$

$\lambda^2 - \lambda - 30 = 0$ this is the characteristic equation

$(\lambda - 6)(\lambda + 5) = 0$

$\therefore \quad \lambda = 6, -5$ these are the eigen value

Now to find Eigen vector for each λ, we use

$AX = \lambda X$

Put $\lambda = 6$

$\therefore \quad AX = 6X$

$\therefore \quad (A - 6I) X = 0$

$$\begin{bmatrix} 12-6 & -51 \\ 2 & -11-6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 6 & -51 \\ 2 & -17 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Use $R_1 \rightarrow R_1 - 3 R_2$

$$\begin{bmatrix} 0 & 0 \\ 2 & -17 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Thus we obtain

$$\begin{bmatrix} 2 & -17 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\therefore \quad 2x_1 - 17x_2 = 0$

$\therefore \quad x_1 = \dfrac{17x_2}{2}$

$\therefore \quad X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{17x_2}{2} \\ x_2 \end{bmatrix}$

Put $x_2 = 2$ (We cannot put $x_2 = 0$ because eigen vector X is always non zero vector.)

We get, $X = \begin{bmatrix} 17 \\ 2 \end{bmatrix}$

$\therefore$ Eigen Vector $X_1 = \begin{bmatrix} 17 \\ 2 \end{bmatrix}$ corresponds to eigen value $\lambda_1 = 6$

Now For $\lambda = -5$

$$AX = -5X$$

$$(A + 5I)X = 0$$

$$\begin{bmatrix} 12+5 & -51 \\ 2 & -11+5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 17 & -51 \\ 2 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\therefore$ $R_1 \to R_1/17$ and $R_2 \to R_2/2$

$$\begin{bmatrix} 1 & -3 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x_1 - 3x_2 = 0 \Rightarrow x_1 = 3x_2$$

Now put $x_1 = 3x_2$ in the vector

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\therefore \quad X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3x_2 \\ x_2 \end{bmatrix}$$

Put $x_2 = 1$

$$\therefore \quad X = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

$\therefore$ Eigen vector $X_2 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$ corresponds to the eigen value $\lambda = -5$

Note that here eigen values are distinct

Case 1 : When eigen values are repeated

Find eigen values and eigen vector for matrix

$$A = \begin{bmatrix} 3 & -1 & 1 \\ -1 & 3 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

Solution:- Let $AX = \lambda X$, $X \neq 0$

$\therefore$ $\det(A - \lambda I) = 0$

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

On solving above determinant we obtain

$$\lambda^3 - 9\lambda^2 + 24\lambda - 20 = 0$$

$$(\lambda - 2)(\lambda - 2)(\lambda - 5) = 0$$

$$\therefore \quad \lambda = 2, 2, 5$$

Now the $\lambda_1 = 5$, corresponding eigen Vector $X_1 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = ?$

Consider $AX = 5X$

$\therefore$ $[A - 5I]X = 0$

$$\begin{bmatrix} -2 & -1 & 1 \\ -1 & -2 & -1 \\ 1 & -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Perform $R_1 \leftrightarrow R_3$

$$\begin{bmatrix} 1 & -1 & -2 \\ -1 & -2 & -1 \\ 2 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$R_2 \to R_2 + R_1$ & $R_3 \to R_3 + 2R_1$

$$\begin{bmatrix} 1 & -1 & -2 \\ 0 & -3 & -3 \\ 0 & -3 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$R_3 \to R_3 - R_2$

$$\begin{bmatrix} 1 & -1 & -2 \\ 0 & -3 & -3 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \ldots(3.11.1)$$

$x_1 - x_2 - 2x_3 = 0$

$-3x_2 - 3x_3 = 0 \Rightarrow x_2 = -x_3$

$\therefore x_1 + x_3 - 2x_3 = 0 \Rightarrow x_1 = x_3$

$$\therefore X_1 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_3 \\ -x_3 \\ x_3 \end{bmatrix} = x_3 \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Put $x_3 = 1$

$$\therefore X_1 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Note: That (1) the matrix $\begin{bmatrix} 1 & -1 & -2 \\ 0 & -3 & -3 \\ 0 & 0 & 0 \end{bmatrix}$ is in row echelon form and the rank is 2

Now, To find eigen vector for $\lambda_2 = 2$

Put $\lambda = 2$ in $AX = \lambda X, X \neq 0$

$\therefore AX = 2X$

$[A - 2I]X = 0$

$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$\therefore R_2 \to R_2 + R_1$ and $R_3 \to R_3 - R_1$

$$\begin{bmatrix} 1 & -1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$x_1 - x_2 + x_3 = 0$$

$$\therefore x_2 = x_1 + x_3$$

Now the trick is to be played to obtain two vectors

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 + x_3 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 + 0x_3 \\ x_1 + x_3 \\ 0x_2 + x_3 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ x_1 \\ 0x_1 \end{bmatrix} + \begin{bmatrix} 0x_3 \\ x_3 \\ x_3 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + x_3 \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Thus Choose $X_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, X_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$

(Note that matrix $\begin{bmatrix} 1 & -1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ in (1) is in row echelon form and its rank is 1)

Also we note that the matrix is symmetric and

$< X_1, X_3 > = X_1 \cdot X_3 = 0$

$< X_1, X_2 > = X_1 \cdot X_2 = 0$

The python code for eigen values and eigen vector is as follows

Code

```
1 import numpy as np
2 A = np.mat("3 -2;1 0")
3 print ("A\n", A)
4 print( "Eigenvalues", np.linalg.eigvals(A))
5
6 eigenvalues, eigenvectors = np.linalg.eig(A)
7 print ("First tuple of eig", eigenvalues)
8 print( "Second tuple of eig\n", eigenvectors)
9
10 for i in range(len(eigenvalues)):
11     print ("Left", np.dot(A, eigenvectors[:,i]))
12     print(...)
```

Output

```
A
 [[ 3 -2]
 [ 1  0]]
Eigenvalues [ 2.  1.]
First tuple of eig [ 2.  1.]
Second tuple of eig
 [[ 0.89442719  0.70710678]
 [ 0.4472136   0.70710678]]
Left [[ 1.78885438]
 [ 0.89442719]]
Ellipsis
Left [[ 0.70710678]
 [ 0.70710678]]
Ellipsis
```

by using the Sympy the code for eigenvalue is as follows

```
>>> M = Matrix([[3, -2,  4, -2], [5,  3, -3, -2], [5, -2,  2, -2], [5, -2, -3,  3]])
>>> M
```

$$\begin{bmatrix} 3 & -2 & 4 & -2 \\ 5 & 3 & -3 & -2 \\ 5 & -2 & 2 & -2 \\ 5 & -2 & -3 & 3 \end{bmatrix}$$

```
>>> M.eigenvals()
```

$$\{-2:1,\ \ 3:1,\ \ 5:2\}$$

The code for Eigenvector using Sympy is as follows :

```
>>> M = Matrix([[3, -2,  4, -2], [5,  3, -3, -2], [5, -2,  2, -2], [5, -2, -3,  3]])
>>> M
```

$$\begin{bmatrix} 3 & -2 & 4 & -2 \\ 5 & 3 & -3 & -2 \\ 5 & -2 & 2 & -2 \\ 5 & -2 & -3 & 3 \end{bmatrix}$$

```
>>> M.eigenvects()
```

$$\left[\left(-2,\ 1,\ \left[\begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}\right]\right),\ \left(3,\ 1,\ \left[\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}\right]\right),\ \left(5,\ 2,\ \left[\begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix},\ \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}\right]\right)\right]$$

Some results that are helpful while solving the problems related to eigen v...

If $A_{n\times n}$ matrix and λ is its eigen value then

(1) λ is the eigen value for A^t

(2) $\frac{1}{\lambda}$ is the eigen value for A^{-1}, if $|A| \neq 0$

(3) λ^k is the eigen value for A^k

(4) If $A_{n\times n}$ is upper triangular matrix then all its diagonal values are the eigen values.

(5) Eigen values of a real symmetric matrix are real.

(6) For any $A_{n\times n}$, $\exists$ a unitary matrix Q such that $Q^{-1}AQ$ is triangular matrix

(7) For any matrix $A_{3\times 3}$, its characteristic equation can be seen as

$\lambda^3 - \text{trace}(A)\,\lambda^2 + (P_{11} + P_{22} + P_{33})\,\lambda - \det(A) = 0$

Where P_{ii} is the determinant of the matrix obtained by deleting the i^{th} row and the i^{th} column.

For $A_{2\times 2}$; characteristic equation is $\lambda^2 - \text{trace}(A)\,\lambda + \det(A) = 0$

(8) Also for matrix $A_{n\times n}$, if $\lambda_1, \lambda_2, \ldots, \lambda_n$ are eigen values then

$\text{trace}(A) = \lambda_1 + \lambda_2 + \ldots\ldots + \lambda_n$ = Total of eigen values and

$\det(A) = \lambda_1, \lambda_2, \ldots, \lambda_n$ = Product of all eigen values.

Similar matrices

We say two square matrices A and B are similar if there exist a non singular matrix S such that

$$A = P^{-1}BP$$

OR

$$B = PAP^{-1}$$

(Now non singular matrix P is a matrix whose determinant is non zero i.e. $\det(P) \neq 0$. Thus non-singular matrix P is an invertible matrix).

Theorem : Similar matrices have same eigen values

Proof :

Let A and B be similar matrices. By definition there exist b a matrix P $\det(P) \neq 0$, such that

$$A = P B P^{-1}$$

We need to prove that eigen values of A and B are same.

i.e. to show their characteristic equations are same i.e. to show $\det(A - \lambda I) = \det(B - \lambda I)$

Note : $\det(AB) = \det(A) \cdot \det(B)$, if $A_{n\times n}$, $B_{n\times n}$

Consider, $\det(A - \lambda I)$

$$= \det(PBP^{-1} - \lambda I)$$
$$= \det(PBP^{-1} - \lambda P \cdot P^{-1}) \quad (\because P \cdot P^{-1} = I)$$
$$= \det(P(B - \lambda I) \cdot P^{-1})$$
$$= \det(P) \cdot \det(B - \lambda I) \cdot \det(P^{-1})$$
$$= \det(P) \det(P^{-1}) \det(B - \lambda I)$$
$$= \det(P \cdot P^{-1}) \cdot \det(B - \lambda I)$$
$$= \det(I) \cdot \det(B - \lambda I)$$
$$= 1 \cdot \det(B - \lambda I)$$
$$= \det(B - \lambda I)$$

Hence the claim holds

Diagonalization

- We say a matrix $A_{n\times n}$ is similar to a diagonal matrix D if there exist an invertible matrix P such that $P^{-1} A P = D$
- If the above condition is satisfied then A is said to be diagonalizable matrix.
- There are various results and theorems that can be helpful to us.

Theorem :

A is an n × n matrix is diagonalizable iff it has n linearly independent eigen vectors. We know move ahead with diagonalization of the matrix. Eigen values for matrix $A_{n\times n}$ are $\lambda_1, \lambda_2, \ldots, \lambda_n$ and the corresponding eigen vector for $X_1, X_2, \ldots\ldots, X_n$.

We now form a matrix say (P) made up of these eigen vectors

$$P = [X_1 \mid X_2 \mid \ldots \mid X_n]$$

This matrix is known as **transforming matrix** and it is invertible.

Moreover, $A = P D P^{-1}$, where D is a diagonal matrix whose diagonal entries are $\lambda_1, \lambda_2, \ldots, \lambda_n$

$$\text{i.e. } D = \begin{bmatrix} \lambda_2 & & O \\ & \lambda_2 & \\ O & & \lambda_n \end{bmatrix}$$

For the matrix $A = \begin{bmatrix} 12 & -51 \\ 2 & -11 \end{bmatrix}$,

$$P = [X_1 \mid X_2] = \begin{bmatrix} 17 & 3 \\ 2 & 1 \end{bmatrix},$$

$$P^{-1} = \frac{1}{11}\begin{bmatrix} 1 & -3 \\ -2 & 17 \end{bmatrix}$$

$$\therefore \text{Consider } P^{-1} A P = \frac{1}{11}\begin{bmatrix} 1 & -3 \\ -2 & 17 \end{bmatrix}\begin{bmatrix} 12 & -51 \\ 2 & -11 \end{bmatrix}\begin{bmatrix} 17 & 3 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 0 \\ 0 & -5 \end{bmatrix} = D$$

$$\therefore P^{-1} A P = D$$

$$\therefore A = P D P^{-1}$$

Note : If a matrix $A_{n\times n}$ has distinct eigen values then the matrix is diagonalizable.

The following code is to diagnolize the matrix and it uses diagonalize. Diagonalize returns a tuple (P,D), where D is diagonal and $M=PDP^{-1}$

```
>>> P, D = M.diagonalize()
>>> P
```

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

```
>>> D
```

$$\begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

```
>>> P*D*P**-1
```

$$\begin{bmatrix} 3 & -2 & 4 & -2 \\ 5 & 3 & -3 & -2 \\ 5 & -2 & 2 & -2 \\ 5 & -2 & -3 & 3 \end{bmatrix}$$

```
>>> P*D*P**-1==M
```

True

Markov process and Markov chains

Before we enter the topic of Markov Chains and process there are some basic concepts that one needs to know so that one can model the dynamic situation to mathematical or programming language.

Probability Vectors :

A row vector $u = (u_1, u_2, \ldots, u_n)$ is called a probability vector if $u_1, u_2, \ldots, u_n$ are non negative and their total sum is 1.

Example

$w = \left(\frac{1}{4}, \frac{1}{4}, 0, \frac{1}{2}\right)$ is probability vector.

Remark

Since the sum of the components of a probability vector is one, an arbitrary probability vector with n components can be represented in terms of n – 1 unknowns as follows.

$(x_1, x_2, \ldots, x_{n-1}, 1 - x_1 - x_2, \ldots, x_{n-1})$

In particular, arbitrary probability vectors with 2 and 3 components can be represented in the form (x, 1 – x) and (x, y, 1 – x – y) respectively.

Stochastic and Regular Stochastic matrices

A square matrix $P = (p_{ij})$ is called a stochastic matrix if each of its rows is a probability vector i.e. if each entry of matrix P is non-negative and the sum of the entries in every row is 1.

Example

$$P = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2} & \frac{1}{6} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & 0 \end{bmatrix}$$

Theorem : If A and B are stochastic matrices then AB is a Stochastic matrix. Therefore in particular, all powers A^n are stochastic matrices.

Regular stochastic Matrix : A matrix P is said to be regular stochastic if all the entries of some power p^m are positive.

Example $P = \begin{bmatrix} 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$

$$P^2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{3}{4} \end{bmatrix}$$, every entry is positive

$\therefore$ P is regular stochastic

Fixed points of a square matrix : A non zero row vector $u = (u_1, u_2, \ldots, u_n)$ is called a fixed point of a square matrix A if u is left fixed, i.e.

$$uA = u$$

Example :

Consider, $A = \begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix}$, $u = (2, -1)$

Then $uA = (2, -1)\begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix} = (2, -1)$

Results :

If u is a fixed vector for matrix A then for any real number $\lambda \neq 0$, the scalar multiple λu is also a fixed vector of A

3.11.1 Relationship between Fixed Points and Regular Stochastic Matrices

☞ **Theorem : (without proof)**

Let P be a regular stochastic matrix. Then

(i) P has unique fixed probability vector v, and the components of v are all positive.

(ii) The sequence of P, P^2, P^3 ..... of powers of P approaches a matrix V whose rows are each the fixed point v.

(iii) If p is an probability vector, then the sequence of vectors pP, pP^2, pP^3, ... approaches the fixed point v.

☞ **Example :** To explain above theorem

Consider regular stochastic matrix $P = \begin{pmatrix} 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$ the probability vector

$t = (x, 1-x)$

Such that $tp = t$

$$(x, 1-x)\begin{pmatrix} 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} = (x, 1-x)$$

$$\therefore \frac{1}{2} - \frac{1}{2}x = x$$

$$\frac{1}{2} + \frac{1}{2}x = 1 - x$$

$$\therefore \quad x = \frac{1}{3}$$

$$t = \left(\frac{1}{3}, 1-\frac{1}{3}\right) = \left(\frac{1}{3}, \frac{2}{3}\right)$$

t is unique fixed probability vector of P.

$T = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}$ is the matrix to which the sequence P, P^2, P^3, P^4 approaches.

$$T = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} = \begin{pmatrix} 0.33 & 0.67 \\ 0.33 & 0.67 \end{pmatrix}$$

So some powers of P are shown below for understanding the concept.

$$P^2 = \begin{pmatrix} 0.5 & 0.5 \\ 0.25 & 0.75 \end{pmatrix}, \quad P^3 = \begin{pmatrix} 0.25 & 0.75 \\ 0.37 & 0.63 \end{pmatrix}$$

$$P^4 = \begin{pmatrix} 0.37 & 0.63 \\ 0.31 & 0.69 \end{pmatrix}, \quad P^5 = \begin{pmatrix} 0.31 & 0.69 \\ 0.34 & 0.66 \end{pmatrix}$$

☞ **Markov Chains**

We now consider a sequence of trials whose outcomes say $X_1, X_2, \ldots$ satisfy the following two properties.

(i) Each outcome belongs to a finite set of outcomes $(a_1, a_2, \ldots, a_m)$ called the state space of the system; if the outcome on the 'n' th trial is a_i, then we say that the system is in state a_i at time n or at the n^{th} step.

(ii) The outcome of any trial depends at most upon the outcome of immediately preceding trial and not upon any other previous outcome; with each pair of states (a_i, a_j) there is given probability P_{ij} that a_j occurs immediately after a_i occurs. Such a stochastic process is called a (finite) Markov chain. The number P_{ij} is called the transition probability and it can be arranged in a matrix.

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & & \vdots \\ p_{m1} & p_{m2} & & p_{mm} \end{pmatrix}$$

This matrix is called transition matrix.

It is a stochastic matrix.

Also with each state a_i there corresponds the i^{th} row $(p_{i1}, p_{i2}, \ldots, p_{im})$ of the transition matrix P, if the system is in the state a_i, then this row vector represents the probabilities of all the possible outcomes of the next trial and so it is a probability vector.

Example : Sunny or Cloudy

- A meteorologist studying the weather in a region decides to classify each day sunny or cloudy. After analyzing several years of weather records he finds :
- The day after a sunny day is sunny 80% of the time and cloudy 20% of the time.
- The day after cloudy day is sunny 60% of time and cloudy 40% of time.

We can set up a markov chain model to model this process. There are just two states S_1 = sunny S_2 = cloudy. The transition diagram is

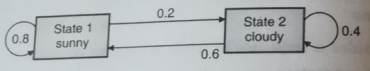


Fig. 3.11.1

$\therefore$ The transition matrix is $P = \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix}$

$P = \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix}$ has all entries positive.

Thus P is regular stochastic matrix.

To find long term probabilities of sunny and cloudy days we must find the eigenvector of P associated to eigenvalue $\lambda = 1$.

The probability vector v has the sum of the entries 1.

Consider $Pv = v$

$\therefore (P - I)v = 0$

$$\therefore \begin{bmatrix} 0.8-1 & 0.6 \\ 0.2 & 0.4-1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -0.2 & 0.6 \\ 0.2 & -0.6 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\therefore R_1 \to R_2 + R_1$

$$\begin{bmatrix} -0.2 & 0.6 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$\therefore -0.2\, v_1 + 0.6\, v_2 = 0$

$\therefore v_1 = 3v_2$

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 3v_2 \\ v_2 \end{bmatrix} = v_2 \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Also $v_1 + v_2 = 1 \Rightarrow 3v_2 + v_2 = 1 \Rightarrow 4v_2 = 1$

$\Rightarrow v_2 = \frac{1}{4}$

$$\therefore v = \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \end{bmatrix}$$

This vector $v = \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \end{bmatrix}$ tells us that in long run, the probability is $\frac{3}{4}$ that the process will be in state 1 and $\frac{1}{4}$ that the process will be in state 2. In other words in long run 75% of the days are sunny and 25% days are cloudy.

Syllabus Topic : Markov Chains

3.12 Markov Chains

In this section, we'll learn about a kind of probabilistic model, a Markov chain. Our first example of a Markov chain comes from computer architecture but we'll first disgulse it as a kind of population problem.

3.12.1 Modeling Population Movement

Imagine a dance club. Some people are on the dance floor and some are standing on the side. If you are standing on the side and a song starts that appeals to you at that moment, you go onto the dance floor and start dancing. Once you are on the dance floor, you are more likely to stay there, even if the song playing is not your favourite.

At the beginning of each song, 56% of the people standing on the side go onto the dance floor, and 12% of the people on the dance floor leave it and go stand on the side. By representing this transition rule by a matrix, we can study the long-

...n evolution of the proportion of people on the dance floor versus the proportion standing on the side.

Assume that nobody enters the club and nobody leaves. Let $x^{(t)} = \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix}$ be the vector representing the state of the system after t songs have played : $x_1^{(t)}$ is the number of people standing on the side, and $x_2^{(t)}$ is the number of people on the dance floor. The transition rule gives rise to an equation that resembles the one for adult juvenile rabbit populations:

$$\begin{bmatrix} x_1^{(t+1)} \\ x_2^{(t+1)} \end{bmatrix} = \begin{bmatrix} .44 & .12 \\ .56 & .88 \end{bmatrix} \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix}$$

One key difference between this system and the rabbit system is that here the overall population remains unchanged; no new people enter the system (and none leave). This is reflected by the fact that the entries in each column add up to exactly 1.

We can use diagonalization to study the long-term trends in proportion of people in each location. The matrix $A = \begin{bmatrix} 0.44 & 0.12 \\ 0.56 & 0.88 \end{bmatrix}$ has two eigenvalues 1 and 0.32. Since this 2×2 matrix has two distinct eigenvalues. Lemma 12.3.14 guarantees that it is diagonalisable: that there is a matrix S such that $S^{-1} AS = A$ where $A = \begin{bmatrix} 1 & 0 \\ 0 & 0.32 \end{bmatrix}$ is a diagonal matrix. One such matrix is $S = \begin{bmatrix} 0.209529 & -1 \\ 0.977802 & 1 \end{bmatrix}$. Writing $A = SAS^{-1}$, we can obtain a formula for $x^{(t)}$, the populations of the two locations after t songs, in terms of $x^{(0)}$, the initial populations :

$$\begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix} = (SAS^{-1})^t \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = SA^tS^{-1} \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix}$$

$$= \begin{bmatrix} 0.21 & -1 \\ 0.98 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & .32 \end{bmatrix}^t \begin{bmatrix} 0.84 & 0.84 \\ -0.82 & 0.18 \end{bmatrix} \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix}$$

$$= \begin{bmatrix} 0.21 & -1 \\ 0.98 & 1 \end{bmatrix} \begin{bmatrix} 1^t & 0 \\ 0 & .32^t \end{bmatrix} \begin{bmatrix} 0.84 & 0.84 \\ -0.82 & 0.18 \end{bmatrix} \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix}$$

$$= 1^t (0.84x_1^{(0)} + 0.84x_2^{(0)}) \begin{bmatrix} 0.21 \\ 0.98 \end{bmatrix} + (0.32)^t (-0.82x_1^{(0)} + 0.18x_2^{(0)}) \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$= 1^t (x_1^{(0)} + x_2^{(0)}) \begin{bmatrix} 0.18 \\ 0.82 \end{bmatrix} + (0.32)^t (-0.82x_1^{(0)} + 0.18x_2^{(0)}) \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad ...(3.12.1)$$

Although the numbers of people in the two locations after t songs depend on the initial numbers of people in the two locations, the dependency grows weaker as the number of songs increases $(0.32)^t$ gets smaller and smaller, so the second term in the sum matters less and less. After ten songs, $(0.32)^t$ is about 0.00001. After twenty songs, it is about 0.0000000001. The first term in the sum is $\begin{bmatrix} 0.18 \\ 0.82 \end{bmatrix}$ times the total number of people. This shows that, as the number of songs increases the proportion of people on the dance floor gets closer and closer to 82%.

12.2 Modeling Randy

Now, without changing the math, we switch interpretations. Instead of modeling whole populations, we model one guy, Randy. Randy moves randomly onto and off the dance floor. When he is off the dance floor (state S1), the probability is 0.56 that he goes onto the dance floor (state S2) when the next song starts; thus the probability is 0.44 that he stays off the floor. Once on the dance floor, when a new song starts, Randy stays on the dance floor with probability is 0.88 thus the probability is 0.12 that he leaves the dance floor. These are called transition probabilities. Randy's behavior is captured in the following diagram.

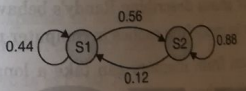


Fig. 3.12.1

Suppose we know whether Randy starts on or off the dance floor. Since Randy's behavior is random, we cannot hope for a formula specifying where he is after t songs.

However, there is a formula that specifies the probability distribution for his location after t songs. Let $x_1^{(t)}$ be the probability that Randy is standing on the side after t songs, and let $x_2^{(t)}$ be the probability that he is on the dance floor after t songs. The probabilities in a probability distribution must sum to one,

so $x_1^{(t)} + x_2^{(t)} = 1$. The transition probabilities imply that the equation.

$$\begin{bmatrix} x_1^{(t+1)} \\ x_2^{(t+1)} \end{bmatrix} = \begin{bmatrix} .44 & .12 \\ .56 & .88 \end{bmatrix} \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix}$$ still holds, so the analysis of section 12.8.1 still

applies : by Equation 3.12.1, as the number t of songs played increases $\begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix}$ very

quickly gets close to $\begin{bmatrix} 0.18 \\ 0.82 \end{bmatrix}$, regardless of where Randy starts out.

3.12.3 Markov Chain Definitions

- A matrix of nonnegative numbers each of whose columns adds up to one is called a stochastic matrix (sometimes a column-stochastic matrix).

  An n-state Markov chain is a discrete time random process such that.

  At each time, the system is in one of n states, say 1, ...n, and
- There is a matrix A such that, if at some time t the system is in state j then for i = 1,...,n, the probability that the system is in state i at time t + 1 is A[i, j].
- That is, A[i, j] is the probability of transitioning from j to i, the j → i transition probability. Randy's location is described by a two-state Markov chain.

3.12.4 Modeling Spatial Locality in Memory Fetches.

- The two-state Markov chain describing Randy's behavior actually comes from a problem that arises in modeling caches in computer memory.
- Since fetching a datum from memory can take a long time (high latency), a computer system uses caches to improve performance; basically the central processing unit (CPU) has its own, small memory (its cache) in which it temporarily stores values it has fetched from memory so that subsequent requests to the same memory location can be handled more quickly.
- If at time t the CPU requests the data at address a, it is rather likely that at time t + 1 the CPU will request the data at address a + 1.

This is true of instruction fetches because unless the CPU executes a branch instruction (Example resulting from an if statement or a loop), the instruction to be executed at time t + 1 is stored immediately after the instruction to be executed at time t. It is also true of data fetches because often a program involves iterating through all the elements of an array (Example Python list).

For this reason, the cache is often designed so that, when the CPU requests the value stored at a location, a whole block of data (consisting of may be sixteen locations) will be brought in and stored in the cache.

If the CPU's next address is within this block (Example the very next location), the CPU does not have to wait so long to get the value.

In order to help computer architects make design decisions, it is helpful to have a mathematical model for predicting whether memory requests that are consecutive in time are to consecutive memory addresses.

A very simple model would be a single (biased) coin: in each timestep.

Probability [address requested at time t + 1 is 1 + address requested at time t] = .6

However, this is too simple a model. Once consecutive addresses have been requested in timesteps t and t + 1, it is very likely that the address requested in timestep t + 2 is also consecutive.

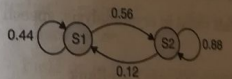


Fig. 3.12.2

The two-state Markov chain is a much more accurate model. Which state the system is in corresponds to whether the CPU is in the process of reading a consecutive sequence of memory locations or is reading unrelated locations.

Suppose the system is in State S1.

This corresponds to the CPU requesting an address. Next, the system follows one of the two arrows from S1, choosing among those arrows with the probability indicated in the diagram. One arrow leads back to S1. This corresponds to the CPU requesting another address that is unrelated to the first.

- ther arrow leads to state S2. This corresponds to the CPU requesting the next address in sequence. Once in state S2, the system stays in S2 for the next timestep with probability 0.88 (issuing a request for another consecutive address) and returns to S1 with probability 0.12 (issuing a request for an unrelated address).
- The analysis of Section 12.8.2 shows that, regardless of where the system starts, after a large number of steps, the probability distribution is approximately $\begin{bmatrix} 0.18 \\ 0.82 \end{bmatrix}$.
- Being in state S1 means that the CPU is issuing the first of a run (possibly of length one) of consecutive addresses. Since the system is in state S1 roughly 18% of the time, the average length of such a run is 1/0.18, which is 5.55.
- This analysis can be extended to make other predictions and help guide computer architects in choosing cache size, block size, and other such parameters.

3.12.5 Modeling Lots of other Stuff

Markov chains are hugely useful in computer science:

- Analyze use of system resources
- Markov chain Monte Carlo
- Hidden Markov models (used in cryptanalysis, speech recognition, AI, finance, biology).

We'll see another example : Google PageRank

In addition, there is work on augmented Markov models, such as Markov decision processes. The important part is that the system has no memory other than that implied by the state – all you need to know to predict a system's future state is its current state, not its history. This is the Markov assumption, and it turns out to be remarkably useful.

3.12.6 Stationary Distributions of Markov Chains

- Perhaps the most important concept in Markov chains is that of the stationary distribution. This is a probability distribution on the states of the Markov chain that is invariant in time.

- That is, if the probability distribution of Randy's state is a stationary distribution at some time t, then after any number of steps the probability distribution will remain the same.
- This is not the same as Randy not moving; of course he changes location many times. It's a statement about the probability distribution of a random variable, not about the value of that random variable.
- Under what circumstances does a Markov chain have a stationary distribution, and how can we find it?
- We saw that the probability distribution at time t, $x^{(t)}$, and the probability distribution at time $t + 1$, $x^{(t+1)}$, are related by the equation $x^{(t+1)} = Ax^{(t)}$.
- Suppose v is a probability distribution on the states of the Markov chain with transition matrix A. It follows that v being a stationary distribution is equivalent to v satisfying the equation.

$$v = Av \qquad \ldots(3.12.2)$$

- This equation in turn means that 1 is an eigenvalue of A, with corresponding eigenvector v.

3.12.7 Sufficient Condition for Existence of a Stationary Distribution

- When should we expect a Markov chain to have a stationary distribution?
- Let A be a column stochastic matrix. Every column sum is 1, so the rows as vectors add up to the all-ones vector.
- Hence the rows of A – I add up to the all zeroes vector. This shows that A – I is singular, and therefore there is a nontrivial linear combination v of its columns that equals the all-zeroes vector. This shows that 1 is an eigenvalue, and that v is a corresponding eigenvector.
- However, this does not show that v is a probability distribution; it might have negative entries. (We can always scale v so that its entries sum to 1)
- There are theorems that guarantee the existence of a nonnegative eigenvector. Here we give a simple condition that pertains to the application:

Theorem

If every entry of the stochastic matrix is positive, then there is a nonnegative eigenvector corresponding to the eigenvalue 1, and also (and we'll see why this is important) every other eigenvalue is smaller in absolute value than 1.

Syllabus Topic : Diagonalization of Fibonacci Matrix

3.13 Diagonalization of Fibonacci Matrix

We first explain the Fibonacci problem which was originally posed by Leonardo di Pisa also known as Fibonacci, in the thirteenth century on his book Liber abaci.

A young pair of rabbits (one of each sex) is placed on an island. A pair of rabbits does not breed until they are two months old. After two months, each pair produces another pair each month. Consider the following diagram

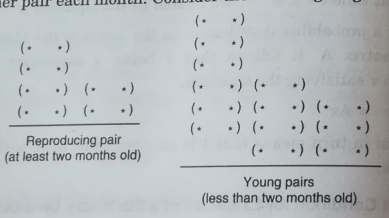


Fig. 3.13.1

We see the following pattern

Month	1	2	3	4	5	6	7
Reproducing pairs	0	0	1	1	2	3	5
Young pairs	1	1	1	2	3	5	8
Total pairs	1	1	2	3	5	8	13

Thus we have recurrence relation for the above sequence 1, 1, 2, 3, 5, 8, 13,... defined as $f_n = f_{n-1} + f_{n-2}$, $n \geq 3$ $f_1 = 1$, $f_2 = 1$

This entire model can represented using matrices as follows:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Which represent current population x_1 is the number of reproducing pairs and x_2 is the number of young pairs. Suppose $x^{(t)}$ is the population after t months.

Then the population at time $t+1$, $x^{(t+1)}$ is related via matrix multiplication to the population at time t, $x^{(t)}$:

$$x^{(t+1)} = A\,x^{(t)}$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Because the number of reproducing pairs at a time $t+1$ is the number of reproducing pair at time t.

$$\therefore \quad a_{11} = 1, \; a_{12} = 1$$

The number of young pairs at time $t+1$ is the number of adults at time t

$$\therefore \quad a_{21} = 1 \quad , \quad a_{22} = 0$$

So we solve the matrix A to obtain its eigen values.

$$dt\,[A - \lambda I] = 0$$

$$\begin{bmatrix} 1-\lambda & 1 \\ 1 & -\lambda \end{bmatrix} = 0$$

$$-\lambda + \lambda^2 - 1 = 0$$

$$\lambda^2 - \lambda - 1 = 0$$

$$\lambda = \frac{-(-1) \pm \sqrt{1+4}}{2}$$

$$\lambda = \frac{1 \pm \sqrt{5}}{2}$$

$$\therefore \quad \lambda_1 = \frac{1+\sqrt{5}}{2} \text{ and } \lambda_2 = \frac{1-\sqrt{5}}{2}$$

Note $\lambda_1 + \lambda_2 = 1$, $\lambda_1 \lambda_2 = -1$ and eigen vector corresponding $\lambda_1 = \frac{1+\sqrt{5}}{2}$ will be $Ax = \lambda_1 x$

$$\begin{bmatrix} 1 - \left(\frac{1+\sqrt{5}}{2}\right) & 1 \\ 1 & -\left(\frac{1+\sqrt{5}}{2}\right) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \lambda_2 & 1 \\ 1 & -\lambda_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\therefore \quad \lambda_2 x_1 + x_2 = 0 \Rightarrow x_2 = -\lambda_2 x_1$$

$$x_1 - \lambda_1 x_2 = 0 \Rightarrow x_1 = \lambda_1 x_2$$

$$\therefore \quad x_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} \lambda_1 x_2 \\ x_2 \end{bmatrix}$$

Put $x_2 = 1$, $\quad x_1 = \begin{bmatrix} \lambda_1 \\ 1 \end{bmatrix}$

For $\lambda_2 = \dfrac{1-\sqrt{5}}{2}$

$$\begin{bmatrix} 1-\left(\frac{1-\sqrt{5}}{2}\right) & 1 \\ 1 & \lambda_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \lambda_1 & 1 \\ 1 & -\lambda_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\lambda_1 x_1 + x_2 = 0 \Rightarrow$$

$$x_1 - \lambda_2 x_2 = 0 \Rightarrow x_1 = \lambda_2 x_2$$

$\therefore$ From $\lambda_1 x_1 + x_2 = 0$

$$\Rightarrow x_2 = -\lambda_1 x_1$$

$$\Rightarrow x_1 = \frac{-1}{\lambda_1} x_2$$

$$x_1 = \lambda_2 x_2$$

Hence both equations are same,

$$\therefore \quad x_2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} \lambda_2 x_2 \\ x_2 \end{bmatrix}$$

Put $x_2 = 1$, $\quad x_2 = \begin{bmatrix} \lambda_2 \\ 1 \end{bmatrix}$

$$\therefore \text{ The Matrix } P = \begin{bmatrix} \lambda_1 & \lambda_2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1+\sqrt{5}}{2} & \frac{1-\sqrt{5}}{2} \\ 1 & 1 \end{bmatrix}$$

Now diagonal matrix D will be

$$D = \begin{bmatrix} 1-\frac{1+\sqrt{5}}{2} & 0 \\ 0 & \frac{1-\sqrt{5}}{2} \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

$$P^{-1} = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & -\lambda_2 \\ -1 & \lambda_1 \end{bmatrix}$$

$$\therefore \quad P^{-1} AP = D$$

More over $P^{-1} A^k P = D^k$

Now $\lambda_1 = \dfrac{1+\sqrt{5}}{2} > \lambda_2 = \dfrac{1+\sqrt{5}}{2}$

Also $|\lambda_1| > |\lambda_2|$, the entries grow roughly like $(\lambda_1)^k$

For any given starting vector $x^{(0)}$, there are numbers a_1, a_2, b_1, b_2 such that for $i = 1.\ 2.$

Entry i of $x^{(t)} = a_i \lambda_1^{(t)} + a_i \lambda_2^{(t)}$

Let $\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = P^{-1} x^{(0)}$ then $A^t P^{-1} x^{(0)} = \begin{bmatrix} c_1 \lambda_1^t \\ c_2 \lambda_2^t \end{bmatrix}$

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \text{ then } P D^t P^{-1} x^{(0)} = P \begin{bmatrix} c_1 \lambda_1^t \\ c_2 \lambda_2^t \end{bmatrix}$$

$$PA^t P^{-1} x^{(0)} = \begin{bmatrix} p_{11} c_1 \lambda_1^t + p_{12} c_2 \lambda_2^t \\ p_{22} c_1 \lambda_2 + p_{22} c_2 \lambda_2^t \end{bmatrix}$$

Thus define $x^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. This corresponds to postulating that initially there is an pair of reproducity adult and no young pair of rabbits.

After a month there is one reproducing pair and one young rabbit's pair.

So $x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $\quad \therefore$ We get $a_1 \lambda_1^1 + b_1 \lambda_2^1 = 1$

$$a_2 \lambda_1^1 + b_1 \lambda_2^1 = 0$$

After 2 months, there are two reproducing pairs and one.

$$x^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$a_1 \lambda_1^2 + b_1 \lambda_2^2 = 1$$

$$a_2 \lambda_1^2 + b_2 \lambda_2^2 = 1$$

Thus we obtain

$$\begin{bmatrix} \lambda_1 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_1 & \lambda_2 \\ \lambda_1^2 & \lambda_2^2 & 0 & 0 \\ 0 & 0 & \lambda_1^2 & \lambda_2^2 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} \frac{5+\sqrt{5}}{10} \\ \frac{5-\sqrt{5}}{0} \\ \frac{1}{\sqrt{5}} \\ -\frac{1}{\sqrt{5}} \end{bmatrix}$$

- Based on this calculation, the number reproducing rabbits pair.

$$x^{(t)} = \left(\frac{5+\sqrt{5}}{10}\right)\left(\frac{1+\sqrt{5}}{2}\right)^t + \left(\frac{5-\sqrt{5}}{10}\right)\left(\frac{1-\sqrt{5}}{10}\right)^t$$

For Example Put t = 3, 4, 5, 6, ..., we get

2, 3, 5, 8, 13, ...

The matrix vectors of P are

$$v_1 = \begin{bmatrix} \frac{1+\sqrt{5}}{2} \\ 1 \end{bmatrix} \qquad v_2 = \begin{bmatrix} \frac{1-\sqrt{5}}{2} \\ 1 \end{bmatrix}$$

- Let $u^{(t)}$ be the co-ordinate representation of $x^{(t)}$ in terms of v_1 and v_2, we will derive an equation relating $u^{(t+1)}$ to $u^{(t)}$.
- (rep 2 vec) to convert from representation $u^{(t)}$ of $x^{(t)}$ to the vector $x^{(t)}$ itself, we multiply $u^{(t)}$ by P.

3.13.1 Co-ordinate representation in terms of page rank

Let $A_{n \times n}$ and $x^{(t)} = A^t x^{(0)}$ for t = 1, 2, ..., now suppose that A is diagonalizable i.e. ∃, P, det(P) ≠ 0 and $P^{-1} AP = D$.

Let $\lambda_1, \lambda_1, ..., \lambda_n$ be eigen values of A and $v_1, v_2, ..., v_n$ be the corresponding eigen vectors which are columns of P. Let $u^{(t)}$ be co-ordinate representation of $x^{(t)}$ in terms of eigen vectors. The equation,

$$x^{(t)} = A^t \cdot x^{(0)}$$ gives rise to

$$[U^{(t)}] = \begin{bmatrix} \lambda_1^t & & 0 \\ & \lambda_2^t & \\ 0 & & \lambda_n^t \end{bmatrix} [U^{(0)}]$$

As the power increases i.e. t increases and if $|\lambda_i| >> |\lambda_j|$ for all j then λ_i^t will dominate.

In particular $|\lambda_1|$ is largest value than for $t >> 0$, $A^t x \approx \alpha_1 \lambda_1^t v_1$

The terms corresponding to eigenvalues with absolute value strictly less than one will actually get smaller as t grows.

Syllabus Topic : The Internet Worm

3.14 The internet worm

- Consider the worm launched on the internet in 1988. A worm is a program that reproduces through the network; an instance of the programming running on one computer tries to break into neighbouring computers and spawn copies of itself on these computers.
- The 1988 worm did not damage but if essentially took over the significant proportion of the computers on internet; the computer was spending all of their cycles running on the worms. The reason is that each computer was running many independent instances of the program.
- The author Robert T. Morris had made some effort to prevent this. The program seems to have been designed so that each worm would check whether there was another worm running on the computer; if so one of them would set a flag indicating it was supposed to die.
- However, with probability 1/7 instead of doing the check, the worm would designate itself immortal. An immortal worm would not do any checks.
- As a consequence, it seems, each computer ends up running many copies of the worm, until the computers whole capacity is used up running worms.

Let us see an easy example.

Consider

Let C_1, C_2 and C_3 be 3 computers connected. In each iteration, each worm has probability of spanning a child worm on each neighbouring computer. Then if it is a mortal worm, with probability 1/T it becomes immortal otherwise dies.

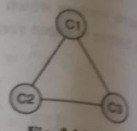


Fig. 3.14.1

The randomners in the models does not allow us to calculate the number of worms after a number of iteration but we can calculate expected (average) number of worms.

Let $x = (x_1, y_1, x_2, y_2, x_3, y_3)$

for $i = 1, 2, 3$, x_i is the expected number of mortal worms at computer i and y_i is the number of immortal worms at computer i.

For $t = 0, 1, 2, \ldots$

Let $x^{(t)} = \left(x_1^t, y_1^t, x_2^t, y_2^t, x_3^t, y_3^t\right)$

As per the model, any mortal worm at C_1 is child of a worm at C_2 or C_3. Thus the expected number of mortal worms at computer after t + 1 iterations is $\frac{1}{10}$ (expected number of worms at C_2 or C_3 after t iterations).

$$\therefore\ x_1^{(t+1)} = \frac{1}{10}\left(x_2^t + y_2^t + x_3^t + y_3^t\right)$$

P(A mortal worm at C_1 becomes immortal) = $\frac{1}{7}$ and those whose are immortal remain immortal.

$$\therefore\ y_1^{(t+1)} = \frac{1}{7}x_1^{(t)} + y_1^{(t)}$$

$\therefore$ The equations for $x_2^{(t+1)}$ and $y_2^{(t+1)}$ and $x_3^{(t+1)}$ and $y_3^{(t+1)}$ are similar

$$\therefore\ x^{(t+1)} = A\,x^{(t)}$$

Where $A = \begin{bmatrix} 0 & 0 & 1/10 & 1/10 & 1/10 & 1/10 \\ 1/7 & 1 & 0 & 0 & 0 & 0 \\ 1/10 & 1/10 & 0 & 0 & 1/10 & 1/10 \\ 0 & 0 & 1/7 & 1 & 0 & 0 \\ 1/10 & 1/10 & 1/10 & 1/10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/7 & 1 \end{bmatrix}$

The $A_{6\times 6}$, its largest eigen value is 1.0054 and it has linearly independent eigen vectors because the eigen values $\lambda_1 = 1.0054$ is the largest we conclude that the number of worms will grow exponentially with number of iteration.

The largest eigen value of A^t is about (1.0054)^t. To get a sense of magnitude for $t = 100$ this number is mere 29, for $t = 200$, the number is = 841.

Since our example A is small enough, the expected number of worms can be computed. Suppose that we start at computer 1 this corresponds to the vector $x^{(0)} = (1, 0, 0, 0, 0, 0)$. In this case the expected number of worms after 900 iterations is about 120 million.

Positive and Positive Semidefinite Matrices

We say a matrix $A_{n\times n}$ is positive definite if all its eigen values are positive and.

We say a matrix $A_{n\times n}$ is positive semidefinite if its all eigen values are nonnegative i.e. all eigen values $\lambda \geq 0$.

Theorem

Every square matrix over C has an eigen value.

Singular Value Decomposition.(SVD)

Let $A_{m\times n}$ invertible and U and V be unitary matrices over R.

Let $A = UDV^*$, D is diagonal matrix

$$\therefore\ A^* = VD^*U^* = VDU^*$$

$$\therefore\ A^*\cdot A = VDU^*\cdot UDV^*$$

Since, U is unitary; $U^*U = I$

$$\therefore\ A^*A = VD^2V^*$$

$\Rightarrow V^*(A^*A)V = D^2$

$\Rightarrow A^*A$ is diagonalizable and eigen values of A^*A are real and positive.

Power Method

Let A be a diagonalizable matrix and thus $\lambda_1, \lambda_2, \ldots, \lambda_n$ are distinct eigen values.

Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$ and $v_1, v_2, \ldots v_n$ the corresponding eigen vectors are linearly independent.

Note if $\lambda \in \mathbb{C}$, and $\lambda = x + i_y$, then $|\lambda| = \sqrt{x^2 + y^2}$)

For an random vector x_0, and a non negative integer

Let $x_t = A_t x_0$

Let $x_0 = \sum_{i=1}^{n} \alpha_i v_i$

$$\therefore x_t = \sum_{i=1}^{n} \alpha_i \lambda_i^t v_i$$

Suppose $\alpha_1 \neq 0$ and $|\lambda_1| >> |\lambda_2|$

Then the coefficient of v_1 grows faster than all other coefficients and eventually it dominates.

That $x_t = \alpha_1 \lambda_1^t v_1 + \text{error}$

As t grow $x_t \approx \alpha_1 \lambda_1^t v_i$

Further we can estimate the corresponding eigen value λ_1 from x_t, because Ax_t will be close to $\lambda_1 x_1$ similarly if the top of eigen values are identical or even very close. The $(q + 1^{st})$ eigen value has smaller absolute value, x_t will be close to a linear combination of the first of eigen vectors and will be an approximate eigen vector.

Syllabus Topic : Modeling a Web Surfer : PageRank

3.15 Modeling a Web Surfer : PageRank

PageRank, the score by which Google ranks pages (or used to, anyway), is based on the idea of a random web surfer, whom we will call Randy. Randy starts at some random web page, and chooses the next page as follows:

- With probability 0.85, Randy selects one of the links from his current web page, and follows it.
- With probability 0.15, Randy jumps to a random web page (never mind how Randy finds a random web page).
- Because of the second item, for every pair *i, j* of web pages, if Randy is currently viewing page *j*, there is a positive probability that the next page he

views is page *i*. Because of that, the theorem applies: there is a stationary distribution, and the power method will find it.

The stationary distribution assigns a probability to each web page. PageRank is this probability. Higher-probability pages are considered better. So the theoretical Google search algorithm is : when the user submits a query consisting of a set of words, Google presents the web pages containing these words, in descending order of probability.

Conveniently for Google, the PageRank vector (the stationary distribution) does not depend on any particular query, so it can be computed once and then used for all subsequent queries. (Of course, Google periodically recomputed it to take into account changes in the web.)

3.16 The Determinant

In this section, we informally discuss determinants. Determinants are very helpful in mathematical arguments.

We give one example of a computational technique based on determinants of 2×2 matrices, computing the area of a polygon.

3.16.1 Areas of Parallelograms

Ex 3.16.1 : Let A be a 2×2 matrix whose columns a_1, a_2 are orthogonal. What is the area of the following rectangle?

$$\{\alpha_1 a_1 + \alpha_2 a_2 : 0 \leq \alpha_1 \alpha_2 \leq 1\} \quad (3.16.1)$$

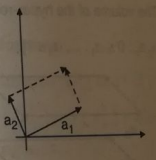


Fig. P 3.16.1

Soln. : The area of a rectangle is the product of the lengths of the two sides, $||a_1|| \, ||a_2||$.

Ex. 3.16.2 : If A is diagonal, Example $A = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$, the rectangle determined by its columns has area equal to the product of the absolute values of the diagonal elements, i.e. 6.

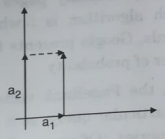


Fig. P 3.16.2

Ex. 3.16.3 : Let $A = \begin{bmatrix} \sqrt{2} & -\sqrt{9/2} \\ \sqrt{2} & \sqrt{9/2} \end{bmatrix}$. Then the columns of A are orthogonal, and their lengths are 2 and 3, so the area is again 6.

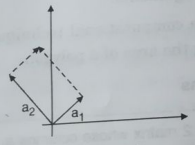


Fig. P 3.16.3

Ex. 3.16.4 : More generally, let A be a n × n matrix whose columns $a_1, ..., a_n$ are orthogonal. The volume of the hyper-rectangle.

$$\{\alpha_1 a_1 + ... + \alpha_n a_n : 0 \le \alpha_1, ... \alpha_n \le 1\} \quad (3.16.2)$$

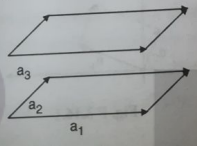


Fig. P 3.16.4

is the product of the lengths of the n sides, so $|| a_1 ||\ || a_2 || ... || a_n ||$.

Ex. 3.16.5 : Now we remove the assumption that a_1, a_2 are orthogonal. The set (3.16. becomes a parallelogram.

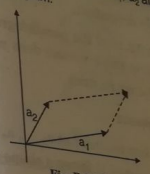


Fig. P 3.16.5

What is its area? You might remember from elementary geometry that the area of a parallelogram is the length of the base times the length of the height.

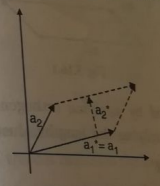


Fig. P 3.16.5(a)

Let $a_1^* = a_1$, and let a_2^* be the projection of a_2 orthogonal to a_1^*. We take a_1 to be the base of the parallelogram. The height is the projection. Then the area is $|| a_1^* ||\ || a_2^* ||$.

Properties of areas of parallelograms

- If a_1 and a_2 are orthogonal, the area is $|| a_1 ||\ || a_2 ||$
- More generally, the area is $|| a_1^* ||\ || a_2^* ||$ where a_1^*, a_2^* are the vectors resulting from orthogonalizing a_1, a_2.
- Multiplying any single vector a_i (i = 1 or i = 2) by a scalar α has the effect of multiplying a_i^* by α, which in turn multiplies the area by $|\alpha|$.
- Adding any scalar multiple of a_1 to a_2 does not change a_2^*, and therefore does not change the area of the parallelogram defined by a_1 and a_2.

- If a_2^* is a zero vector, the area is zero. This shows that the area is zero if the vectors a_1, a_2 are linearly dependent.
- The algebraic definition 12.11 of the parallelogram is symmetric with respect to a_1 and a_2, so exchanging these vectors does not change the parallelogram, and therefore does not change its area.

3.16.2 Volumes of Parallelepipeds

We can do the same in n dimensions. Let $a_1,\ldots, a_n$ be n-vectors. The set $\{\alpha_1 a_1 + \ldots + \alpha_n a_n : 0 \le \alpha_1, \ldots, \alpha_n \le 1\}$ forms a shape called a parallelepiped.

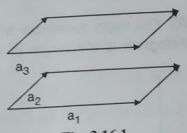


Fig. 3.16.1

Its volume can be found by applying orthogonalization to the columns to obtain $a_1^*, \ldots, a_n^*$ and multiplying the lengths. Just as in the two-dimensional case, we observe the following.

☛ **Properties of volumes of parallelepipeds**

- If $a_1, \ldots, a_n$ are orthogonal, the volume is
  $||a_1|| \, ||a_2|| \ldots ||a_n||$
- In general, the volume is
  $||a_1^*|| \, ||a_2^*|| \ldots ||a_n^*||$
  Where $a_1^*, a_2^* \ldots a_n^*$ are the vectors resulting from the orthogonalization of $a_1, a_2, \ldots a_n$.
- Multiplying any single vector a_i by a scalar α has the effect of multiplying a_i^* by α, which in turn multiplies the volume by $|\alpha|$.
- For any $i < j$, adding a multiple of a_i to a_j does not change a_j^*, and therefore does not change the volume.
- If the vectors $a_1, \ldots, a_n$ are linearly dependent, the volume is zero.
- Reordering the vectors does not change the volume.

3.16.3 Expressing the Area of a Polygon in Terms of Areas of Parallelograms

Fig. 3.16.2

We consider a computational problem arising in graphics, computing the area of a simple polygon.

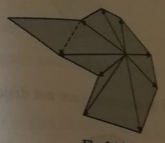


Fig. 3.16.3

Let $a_0, \ldots, a_{n-1}$ be the locations of the vertices of the polygon, expressed as (x, y) pairs. In the Fig. **, the dot indicates the location of the origin.

We can express the area of the polygon as the area of n triangles:

- The triangle formed by the origin with a_0 and a_1,
- With a_1 and a_2,
- ⋮
- With a_{n-2} and a_{n-1} and
- With a_{n-1} and a_0.

Duplicate and reflect the triangle formed by the origin with a_0 and a_1, and attach it to the original; the result is the parallelogram

$\{\alpha_0 a_0 + \alpha_1 a_1 : 0 \le \alpha_0, \alpha_1 \le 1\}$

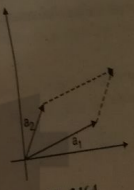


Fig. 3.16.4

- Randy is our random surfer. In each iteration, Randy selects an outgoing link from his current web page and follows that link. (If the current web page has no outgoing link, Randy stays put.)
- To see this rudimentary PageRank in action, let's consider a small example. We call it the Thimble-Wide Web. It consists of only six WebPages :

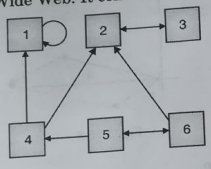


Fig. 3.17.1

- Here is the transition Markov chain :

$$A_1 = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & & & \frac{1}{2} & & \\ 2 & & & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{2} \\ 3 & & 1 & & & & \\ 4 & & & & & \frac{1}{3} & \\ 5 & & & & & & \frac{1}{2} \\ 6 & & & & & \frac{1}{3} & \end{array}$$

- Column j gives the probabilities that a surfer viewing page j transitions to pages 1 through 6. If page j has no outgoing links, the surfer stays at page j with probability 1.
- Otherwise, each of the pages linked to has equal probability; if page j has d links, the surfer transitions to each of the linked-to pages with probability 1/d.
- The probability is zero that the surfer transitions from page j to a page that page j do not link to. (In other words, cell A_{ij}) contains the probability that, at page j, the surfer will transition to page i.

- For example, page 5 links to pages 2, 4, and 6 so a surfer at page 5 transitions to each of these pages with probability 1/3. You should check that the above matrix is a stochastic matrix (every column sum is 1), and so it really describes a Markov chain.
- According to this Markov chain, how likely is Randy to be at each page after many iterations? What are the most likely pages? The answer depends on where he starts and how many steps he takes :
  - If he starts at page 6 and takes an even number of iterations, he has about probability 7 of being at page 3, probability 2 of being at page 2, and probability 1 of being at page 1.
  - If he starts at 6 and takes an odd number of iterations, the probability distribution is about the same except that the probabilities of nodes 2 and 3 are swapped.
  - If he starts at page 4, the probability is about 5 times that he is at page 1 and about 5 that he is at page 3 (if an even number of iterations) or page 2 if at odd number of iterations).

From the point of view of computing definitive pageranks using the power method, these are two things wrong with this Markov chain:

(1) There are multiple clusters in which Randy gets stuck. One cluster is page 2, page 3 and the other cluster is page 1.

(2) There is a part of the Markov chain that induces periodic behaviour. Once Randy enters the cluster page 2, page 3, the probability distribution changes in each iteration.

- The first property implies that there are multiple stationary distributions. The second property means that the power method might not converge.
- We want a Markov chain with a unique stationary distribution so we can use the stationary distribution as an assignment of importance weights to web pages.
- We also want to be able to compute it with the power method. We apparently cannot work with the Markov chain in which Randy simply chooses a random outgoing link in each step.

Therefore the area of the triangle is half the area of this parallelogram. Summing over all the triangles, we get that the area of the polygon is

$$\frac{1}{2}(\text{area } (a_0, a_1) + \text{area } (a_1, a_2) + ... + \text{area } (a_{n-1}, a_0)) \quad (3.16.3)$$

However, this approach fails for some polygons, Example

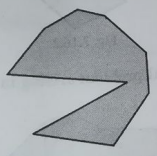


Fig. 3.16.5

Since the triangles formed by a_i and a_{i+1} are not disjoint and do not even fully lie within the polygon:

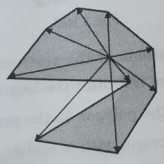


Fig. 3.16.6

For this reason, we consider signed area. The sign of the signed area of the parallelogram formed by the vectors a_i and a_{i+1} depends on how these vectors are arranged about this parallelogram. If a_1 points in the counter clockwise direction about the parallelogram, and a_2 points in the clockwise direction, as in

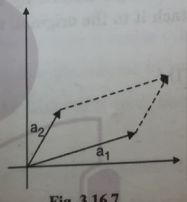


Fig. 3.16.7

Then the area is positive. On the other hand, if a_2 points in the counter clockwise direction and a_1 points in the clockwise direction, as in

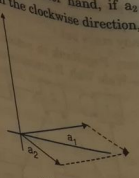


Fig. 3.16.8

Then the area is negative.

Replacing area with signed area in Formula 3.16.3 makes the formula correct for all simple polygons.

$$\frac{1}{2}(\text{signed area } (a_0, a_1) + \text{signed area } (a_1, a_2) + ... + \text{signed area } (a_{n-1}, a_0)) \quad (3.16.4)$$

This formula is convenient because the signed area of the parallelogram defined by a_1 and a_2 has a simple form. Le A be the 2×2 matrix whose columns are a_1, a_2. Then the signed area is A [1, 1] A [2, 2] – A [2, 1] A [1, 2].

3.17 Pagerank

3.17.1 Concepts

- In this lab, we'll be implementing the algorithm that Google originally used to determine the "importance" (or rank) of a web page, which is known as PageRank.
- The idea for PageRank is this : Define a Markov chain that describes the behaviour of a random web-surfer.
- Randy consider the stationary distribution of this Markov chain. Define the weight of a page to be the probability of that page in the stationary distribution.
- First we describe a rudimentary Markov chain, and we discover why it needs to be improved.

- Consider a very simple Markov chain: the surfer jumps from whatever page he's on to a page chosen uniformly at random. Here's the transition matrix for our Thimble Wide Web.

$$A_2 = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 2 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 3 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 4 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 5 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 6 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{array}$$

- This Markov chain has the advantage that it avoids the problems with the previous chain ; the surfer can't get stuck, and there is no fixed period.
- As a consequence this Markov chain does have a unique stationary distribution (it assigns equal probability to every page) and this stationary distribution can be computed using the power method.
- On the other hand, you might point out, this Markov chain does not in any way reflect the structure of the Thimble-Wide Web. Using the stationary distribution to assign weights would be silly.
- Instead, we will use a mixture of these two Markov chains. That is, we will use the Markov chain whose transition matrix is

$$A = 0.85A_1 + 0.15A_2 \quad \ldots(3.17.1)$$

- Since every column of A_1 sums to 1, every column of $0.85A_1$ sums to 0.86 and since every column of A_2 sums to 1, every column of $.15A_2$ sums to 15, so (finally) every columns of $0.85A_1 + 0.15A_2$.

The Markov chain corresponding to the matrix A describes a surfer obeying the following rule.

- With probability 0.85, Randy selects one of the links from his current web page and follows it.
- With probability 0.15, Randy jumps to a web page chosen uniformly at random. (This is called teleporting in the context of PageRank.)

You can think of the second item as modelling the fact that sometimes the surfer gets bored with where he is. However, it plays a mathematically important role.

- The matrix A is a positive matrix (every entry is positive). A theorem ensures that there is a unique stationary distribution, and that the power method will converge to it.
- For an n-page web, A_1 will be the $n \times n$ matrix whose ij entry it.

  1 if $i = j$ and page j has no outgoing links,

  $1/d_j$ if j and d_j outgoing links, and one of them points to gage i, and

  0 otherwise and A_2 will be the $n \times n$ matrix each entry of which is $1/n$.

3.17.2 Working with a Big Dataset

- In this lab we will use a big dataset : articles from Wikipedia. Wikipedia contains a few million articles. Handling all of them would make things run too slowly for a lab. We will therefore work with a subset containing about 825,000 articles chosen by taking all articles that contain the strings mathematic, sport, politic, literate and law.
- This chooses all sorts of articles. For example the article on the impressionist artist Edward Manet is included because his father wanted him to be a lawyer.
- Handling a big dataset presents a few obstacles which we help you to overcome. We will give you specific instructions that will help you to write code that is efficient in terms of both running time and use of memory. Here are some guidelines :
  - Be sure to exploit sparsity. We will use sparse representation of matrices and vectors and exploit sparsity in computations involving them.
  - Do not duplicate data unless you have to. For example, you will have to use the set of titles of Wikipedia entries several times in the code

Task 3.17.3 : Write a procedure power-method with the following spec:

- **Input**

  The matrix A_1, and

  The desired number of iterations of the power method
- **Output :** an approximation to the stationary distribution, or at least a scalar multiple of the stationary distribution.
- Your initial vector can be pretty much anything nonzero. We recommend using an all-ones vector.
- In order to see how well the method converges, at each iteration print the ratio

  (norm of v before the iteration) / (norm of v after the iteration)
- As the approximation for the eigenvector with eigen value 1 gets better, this ratio should get closer to 1.
- Test your code using the matrix A_1 you obtained for the Thimble-Wide Web. The module pagerank_test defines A_2 to allow you to explicitly test whether the vector you get is an approximate eigenvector of A.

  You should obtain as an eigenvector a scalar multiple of the following vector :

  {1 : 0.5222, 2 : 0.6182, 3 : 0.5738, 4 : 0.0705, 5 : 0.0783, 6 : 0.0705}

3.17.4 The Dataset

Importing the pagerank module read into the workspace a few variables and procedures described below. In principle, given enough time you should be able to write perform these tasks yourselves (or actually already did them in previous labs).

1. **read_data :** a function that reads in the relevant data for this lab. This function will take a few minutes to execute, so use it only when needed and only once. It returns a matrix, links, which is the matrix representing the link structure between articles.
2. **find_word :** a procedure that takes a word and returns a list of titles of articles that contain that word. (Some words were omitted since they appear in too many articles or too few; for such a word, find_word returns an empty list or None).

You can view the contents of an article with a given name on http:// en.wikipedia.org. Note that the titles are all in lower case, whereas the Wikipedia article may contain both upper and lower case. Also note that this dataset was generated a while ago, so some of the articles may have changed.

Task 3.17.4

How many documents contain the word Jordan? The first title in the list of articles that contain Jordan is Alabama. Open the Wikipedia page and find out why.

3.17.5 Handling Queries

You next need to write code to support queries.

Task 3.17.5 : Write a procedure wikigoogle with the following spec :

Input

- A single word w.
- The number k of desired results
- The pagerank eigenvector p.

Output : A list of the names of the k highest-pagerank Wikipedia articles containing that word.

First use find_word to obtain the list related of articles that contain w. Then sort the list in descending order with respect to the pagerank vector, using related, sort (key = lambda x : p[x], reverse = True)

(The key keyword lets you specify a function that maps list elements to numbers. lambda x : p[x] is a way to define a procedure that, given x, returns p[x]). Finally, return the first k elements of the list.

Task 3.17.6

Use power-method to compute the pagerank eigenvector for the Wikipedia corpus and try some queries to see the titles of the top few pages: "Jordan", "obama" "tiger" and of course "matrix". What do you get for your top few articles? Can you explain why? Are the top ranked results more relevant or important in some sense than, say, the first few articles returned by find_word without ranking ?

labels of matrices and vectors. Make sure you do not create new copies of this set (assignment of a set does not copy the set, just creates an additional reference to it).

- Test your code on a small test case (we provide one) before running it with the big dataset.
- Remember to use the imp module to reload your file after a change, so that you do not need to re-import the pagerank module. (Use from imp import reload when you start python, then use reload (myfile) to reload your file without re-importing pagerank.)
- Don't use other programs such as a web browser while computing with a big dataset.
- Leave enough time for computation. The power-method computation should take between five and ten minutes.

- Your mat module should be okay if you implemented matrix-vector multiplication in the way suggested in lecture. If you get into trouble, use our implementation of mat.

3.17.3 Implementing PageRank using the Power Method

- The power method is a very useful method in linear algebra for approximating the eigenvector corresponding to the eigenvalue of largest absolute value.
- In this case, the matrix we are interested in is A given above. The key observation is that for a random vector v, $A^k v$ (A^k is A multiplied by itself k times) is very likely to be a good approximation for the eigenvector corresponding to A's largest eigenvalue.
- We will compute $A^k v$ iteratively. We maintain a vector v, and update it using the rule v : = Av.
- After just a few iterations (say 5), we stop. Sounds trivial, right? The problem is that A is 825372×825372, and v is a 825372-vector. Representing A or A_2 explicitly will take too much space, and multiplying either matrix by a vector explicitly will take too much time.
- We will exploit the structure of A to compute each power method iteration of the more efficiently. Recall that $A = 0.85A_1 + 0.15A_2$. We will treat each of these terms separately. By distributivity, $Av = 0.85A_1 v + 0.15A_2 v$.

Handling A_2

Suppose you've computed a vector w = 0.85 A_1 v. What's involved in adding 0.15 A_2 v to w? In particular, can you do that without explicitly constructing A_2?

Computing A_1

The input data will consist of a square matrix L whose nonzero entries are all 1. This matrix represents the link structure between articles. In particular, the rc entry of L is 1 if article c links to article r.

For testing purposes, we have provided the module pagerank_test, which defines the matrix small_links representing the link structure of the Thimble-Wide Web. It also defines the corresponding matrix A_2.

Task 3.17.1 : Write a procedure find_num_links with the following spec :

Input : A square matrix L representing a link structure as described above.

Output : A vector num _ links whose label set is the column-label set of L, such that, for each column-label c, entry c of num_links is the number of nonzero entries in column c of L.

Try to write the procedure without using loops or comprehensions on matrix L. Or so we are led to believe by the original article. At this point, the details of the algorithm used are closely guarded secret but we suspect that the ideas of PageRank still play a major role.

Task 3.17.2 : Write a procedure make_Markov with the following spec :

- **Input :** A square matrix L representing a link structure as described above.
- **Output :** This procedure does not produce new output, but instead mutates L (changing its entries) so that it plays the role of A_1.
- The description of A_1 is given earlier in this writeup. Using mutation instead of returning a new matrix saves space. Your procedure should make use of find_num_links.
- Test your procedure on small_links. Make sure the matrix you obtain is correct. You will be given such a matrix links that describes the link structure among wikipedia entries. Its row and column-labels are titles of Wikipedia entries.

.6 Biasing the Pagerank

Suppose you are particularly interested in sports. You would like to use PageRank but biased towards sports interpretations of words.

Let A_{sport} be the n × n transition matrix in which every page transitions to the page whose title is sport. That is, row sport is all ones, and all other rows are all zeroes.

Then $0.55\ A_1 + 0.14\ A_2 + 0.3\ A_{sport}$ is the transition matrix of a Markov chain in which Randy occasionally jumps to the sport article.

Exercise

Q. 1 For each of the following matrix vector equations find the solutions

(a) $\begin{bmatrix} 10 & 2 & -3 & 5 & 3 \\ 0 & 0 & 1 & 20 & 13 \end{bmatrix} * [x_1, x_2, x_3, x_4] = [1, 3]$

(b) $\begin{bmatrix} 2 & 0 & 1 & 3 \\ 0 & 0 & 5 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} * [x_1, x_2, x_3, x_4] = [1, -1, 3]$

Q.2 For each of the matrix – vector equations check whether the solution exist or not? If it exists then solve –

(a) $\begin{bmatrix} 1 & 3 & -2 & 1 & 0 \\ 0 & 0 & 2 & -3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * [x_1, x_2, x_3, x_4, x_5] = [5, 3, 2, 1]$

(b) $\begin{bmatrix} 1 & 2 & -8 & -4 & 0 \\ 0 & 0 & 2 & 12 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * [x_1, x_2, x_3, x_4, x_5] = [5, 4, 0, 0]$

Q. 3 Solve the following system by Gaussion elimination method

(a)
$$\begin{aligned} x + 2y + 3z - t &= 10 \\ 3x + 2y - 4z + 3t &= 2 \\ 2x + 3y - 3z - t &= 1 \\ 2x - y + 2z + 3t &= 7 \end{aligned}$$

(b)
$$\begin{aligned} x + 4y - z &= -5 \\ x + y - 6z &= -12 \\ 3x - y - z &= 4 \end{aligned}$$

(c)
$$\begin{aligned} 2x + 3y - z &= 7 \\ x + 2y + z &= 6 \\ x + y - z &= 2 \end{aligned}$$

(d)
$$\begin{aligned} 6x - y - z &= 19 \\ 3x + 4y + z &= 26 \\ x + 2y + 6z &= 22 \end{aligned}$$

Q. 4 Find the orthonormal basis for subspace of $\mathbb{R}^4$ generated by the following

(a) (1, 2, 1, 0) and (1, 2, 3, 1)

(b) (1, 1, 0, 0) (1, −1, 1, 1) and (−1, 0, 2, 1)

Q. 5 Suppose v = (1, 3, 5, 7). Find the projection v on two in other words find $w \in W$ that minimizes $\| v - w \|$

Where W is subspace of $\mathbb{R}^4$ spanned by

(1) $u_1 = (1, 1, 1, 1)$, and $u_2 = (1, -3, 4, -2)$

(2) $u_1 = (1, 1, 1, 1)$ and $u_2 = (1, 2, 3, 2)$

Q. 6 Suppose v = (1, 2, 3, 4, 6). Find the projection v onto W in other word find $w \in W$ that minimizes $\|v - w\|$ where W is subspace of $\mathbb{R}^5$. Spanned by

(a) $u_1 = (1, 2, 1, 2, 1)$ and $u_2 = (1, -1, 2, -1, 1)$

(b) $v_1 = (1, 2, 1, 2, 1)$ and $v_2 = (1, 0, 1, 5, -1)$

Inner product and Orthoganality

Q.1 Let $u = (1, 3, -4, 2)$

$v = (4, -2, 2, 1)$

$w = (5, -1, -2, 6)$ in $\mathbb{R}^4$

(a) Show $\langle 3u, -2v, w \rangle = 3 \langle u, w \rangle - 2 \langle v, w \rangle$

(b) Also normalize u and v.

Q.2 Find the angle between the two vector u, v using the formula

$$\cos(\theta) = \frac{\langle u, v \rangle}{\| u \| \cdot \| v \|}$$

(a) $u = (2, 3, 5)$, $v = (1, -4, 3)$

(b) $u(t) = 3t - 5$, $v(t) = t^2$

Here, $\langle u(t), v(t) \rangle := \int_0^1 u(t)\, v(t)\, dt$

Q. 3 Expand

(a) $< 5u_1 + 8u_2, 6v_1 - 7v_2 >$

(b) $< 3u + 5v, 4u - 6v >$

(c) $\| 2u - 3v \|^2$

Q. 4 Find θ if $A = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

Where, $< A, B > : =$ trace $(B^t A)$

Q. 5 Find k so that u = (1, 2, k, 3)

V = (3, k, 7, –5) in $\mathbb{R}^4$ are orthogonal.

Q. 6 Let W be the subspace of $\mathbb{R}^5$ sparned by

u = (1, 2, 3, –1, 2) and v = (2, 4, 7, 2, –1)

Find a basis of the orthogonal complement $W^{\perp}$ of W.

Q. 7 Find C and the projection of v = (1, – 2, 3, 4) along W = (1, 2, 1, 2) in $\mathbb{R}^4$.

$\left(\textbf{Ans.} = C = \frac{-4}{5}, \text{Proj } (v, w) = cw = \left(\frac{-4}{5}, \frac{-8}{5}, \frac{-4}{5}, \frac{-8}{5} \right) \right)$

Q. 8 Consider the subspace U of $\mathbb{R}^4$ spanned by the vectors.

$v_1 = (1, 1, 1, 1)$, $v_2 = (1, 1, 2, 4)$, $v_1 = (1, 2, -4, -3)$

Find (a) an orthogonal basis of U.

(b) an orthogonal basis of U.

(**Hint :** Gram Schmidt orthogonalization process)

Q. 9 For each of the following a, b find $b^{\perp a}$ and $b^{\perp a}$

(a) a = (3, 0) ; b (2, 1)

(b) a = (1, 2, –1) , b = (1, 1, 4)

(c) a = [3, 3, 12] , b = (1, 1, 4)

Q. 10 For each of the vectors a, b find the vector in span {a} that is closest to b.

(i) a = (1, 2), b(2, 3)

(ii) a = (0, 1, 0), $b = (\sqrt{2}, 1, \sqrt{3})$

(iii) a = (–3, –2, –1, 4) , b = (7, 2, 5, 0)

Q. 11 Find generators for the orthogonal complement of U with respect to W.

Where,

(i) U = span {(–4, 3, 1, –2), (–2, 2, 3, –1) } and $W = \mathbb{R}^4$

(ii) U = span { (3, 0, 1) } , W = span {(1, 0, 0), (1, 0, 1)}

Q. 12 Let, $A = \begin{bmatrix} -4 & -1 & -3 & -2 \\ 0 & 4 & 0 & -1 \end{bmatrix}$

use orthogonal complement to find a basis for the null space of A.

Q. 13 Find eigen values, eigen vectors of the the following matrices. Also check for diagonalization the matrices. If the matrix is diagonalizable then find the transforming matrix P.

(a) $A = \begin{bmatrix} 2 & -1 & 1 \\ 1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix}$ (b) $A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$

(c) $A = \begin{bmatrix} 2 & 1 & 1 \\ 2 & 3 & 2 \\ 3 & 3 & 4 \end{bmatrix}$ (d) $A = \begin{bmatrix} 8 & -6 & 2 \\ -6 & 7 & -4 \\ 2 & -4 & 3 \end{bmatrix}$

(e) $A = \begin{bmatrix} -2 & 5 & 4 \\ 5 & 7 & 5 \\ 4 & 5 & -2 \end{bmatrix}$ (f) $A = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$

(g) $A = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{bmatrix}$ (h) $A = \begin{bmatrix} 4 & 6 & 6 \\ 1 & 3 & 2 \\ -1 & -5 & -2 \end{bmatrix}$

(i) $A = \begin{bmatrix} -9 & 4 & 4 \\ -8 & 3 & 4 \\ -16 & 8 & 7 \end{bmatrix}$ (k) $A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$

Q. 14 Show that following matrices are similar to diagonal matrices. Find the diagonal matrix and the transforming matrix.

(a) $\begin{bmatrix} -2 & 2 & -3 \\ 2 & 1 & -6 \\ -1 & -2 & 0 \end{bmatrix}$ (b) $\begin{bmatrix} -17 & 18 & -6 \\ -18 & 19 & -6 \\ -9 & 9 & 2 \end{bmatrix}$

Q. 15 Show that the following are not diagonalisable.

(a) $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & -2 & 0 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{bmatrix}$ (c) $\begin{bmatrix} 3 & 10 & 5 \\ -2 & -3 & -4 \\ 3 & 5 & 7 \end{bmatrix}$

Q. 16 Find eigen values and eigen vectors for

(a) $\begin{bmatrix} 2 & 1 & 1 \\ 2 & 3 & 2 \\ 3 & 3 & 4 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$ (c) $\begin{bmatrix} 4 & 6 & 6 \\ 1 & 3 & 2 \\ -1 & -5 & -2 \end{bmatrix}$ (d) $\begin{bmatrix} 3 & 17 \\ 0 & 24 \end{bmatrix}$

Markor chain

Q.1 A salesman's territory consists of three cities A, B and C. He never sells in the same city on successive days. If he sells in city A, then the next day he sells in B. However if he sells in either B or C, then the next day he is twice likely to sell in city A as in the other city. In long run, how often does he sell in each of the cities ?

Q.2 Two boys b_1 and b_2 and two girls g_1 and g_2 are throwing a ball from one to the other each boy throws the ball to the other boy with probability 1/2 and to each girl with probability 1/4.

On the other hand, each girl throws the ball to each boy with probability 1/2 and never to the other girl. In the long run, how often does each receive the ball ?

Q. 3 There are two marbles in urn A and 3 red marbles in urn B. At each step of the process a marble is selected from each urn and the two marbles selected are intercharged. Let the state ai of the system be the number i of red marble in urn A.

(i) Find the transition matrix P

(ii) What is the probability that there are 2 red marbles in urn A after 3 step ?

(iii) In the long run, what is the probability that there are 2 red marbles in urn A ?

❑❑❑

List of Practicals

Program 1 : Write a program which demonstrates the following.

(a) Addition of two complex numbers

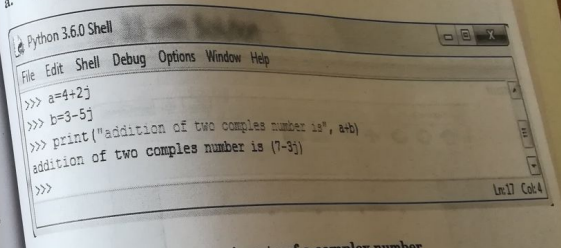
(b) Displaying the conjugate of a complex number

(c) Plotting a set of complex numbers

(d) Creating a new plot by rotating the given number by a degree90,180, 270 degrees and also by scaling by a number a=1/2, a=1/3, a=2 etc.

Solution :

a. **Addition of two complex numbers**



b. **Displaying the conjugate of a complex number**