week08-03–recap-markov

March 22, 2024

1 George McNinch Math 87 - Spring 2024

2 Week 08

2.1 # Markov discussion - recaps

3 Introduction

Last week, we introduced some finite-state machines whose transition behavior could be described by a matrix. This week, we investigate properties of such matrices, by studying their eigenvalues and eigenvectors.

4 Eigen-stuff

Recall that a number $\lambda \in \mathbb{R}$ is an *eigenvalue* of $A \in \mathbb{R}^{n \times n}$ if there is a non-zero vector $\mathbf{v} \in \mathbb{R}^n$ for which

$$A\mathbf{v} = \lambda \mathbf{v};$$

$\mathbf{v}$ is then called an *eigenvector*.

If A is diagonal – e.g. if

$$A = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix} = \operatorname{diag}(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$$

– it is easy to see that each standard basis vector $\mathbf{e}_i$ is an eigenvector, with corresponding eigenvalue λ_i (the (i, i)-the entry of A).

5 Diagonalizable matrices

Now suppose that A is an $n \times n$ matrix, that $\mathbf{v}_1, \dots, \mathbf{v}_n$ are eigenvectors for A, and that $\lambda_1, \dots, \lambda_n$ are the corresponding eigenvalues. Write

$$P = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix}$$

for the matrix whose columns are the $\mathbf{v}_i$.

Theorem: If the eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ are linearly independent – equivalently, if the matrix P is invertible – then

$$P^{-1}AP = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_n \end{bmatrix} = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$$

i.e. $P^{-1}AP$ is the diagonal matrix $n \times n$ matrix whose diagonal entries are $\lambda_1, \dots, \lambda_n$.

---

6 Diagonalizable matrices & power iteration

Theorem: Let A be a diagonalizable $n \times n$, with n linearly independent eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ with corresponding eigenvalues $\lambda_1, \dots, \lambda_n$. As before, write

$$P = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix}.$$

a) Suppose $|\lambda_i| < 1$ for all i. Then $A^m \to \mathbf{0}$ as $m \to \infty$.

b) Suppose that $\lambda_1 = 1$, and $|\lambda_i| < 1$ for $2 \le i \le n$. Any vector $\mathbf{v} \in \mathbb{R}^n$ may be written

$$\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{v}_i.$$

If $c_1 \neq 0$, then

$$A^m \mathbf{v} = c_1 \mathbf{v}_1 \quad \text{as} \quad m \to \infty.$$

If $c_1 = 0$ then

$$A^m \mathbf{v} = \mathbf{0} \quad \text{as} \quad m \to \infty.$$

6.1 Corollary

Suppose that A is diagonalizable with eigenvalues $\lambda_1, \dots, \lambda_n$, that $\lambda_1 = 1$, and that $|\lambda_i| < 1$ for $i = 2, ..., n$. Let $\mathbf{v_1}$ be a 1-eigenvector for A.

Then

$$A^m \to B \quad \text{as } m \to \infty$$

for a matrix B with the property that each column of B is either $\mathbf{0}$ or some multiple of $\mathbf{v_1}$.

---

7 Stochastic Matrices

A vector $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}^T \in \mathbb{R}^n$ will be said to be a *probability vector* if all of its entries v_i satisfy $v_i \geq 0$ and if

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \cdot \mathbf{v} = \sum_{i=1}^{n} v_i = 1.$$

Let $A = \begin{bmatrix} a_{ij} \end{bmatrix} \in \mathbb{R}^{n \times n}$. We say that A is a *stochastic matrix* if $a_{ij} \geq 0$ for all i, j and if

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \cdot A = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix};$$

in words, A is a stochastic matrix if each column of A is a probability vector.

Proposition: Let A be a stochastic matrix.
a) A has an eigenvector with eigenvalue 1.
b) Let λ be any eigenvalue of a A. Then $|\lambda| \leq 1$.
c) If $\mathbf{w}$ is an eigenvector of A with eigenvalue λ satisfying $\lambda \neq 1$ then $\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \mathbf{w} = 0$.

Corollary: Suppose that the stochastic matrix A is diagonalizable, and that the *1-eigenspace* of A has dimension 1. Let $\mathbf{v}$ be an eigenvector for A with eigenvalue 1, and set $c = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \mathbf{v}$. Then $\mathbf{w} = \dfrac{\mathbf{v}}{c}$ is a probability vector, and

$$A^m \to B \quad \text{as } m \to \infty$$

for a stochastic matrix B. Each column of B is equal to $\mathbf{w}$.

Theorem: (Perron-Frobenius) Let G be a transition diagram for a Markov chain, and suppose that G is strongly connected and aperiodic. Let P be the corresponding stochastic matrix. The multiplicity of the eigenvalue $\lambda = 1$ for P is 1 – i.e.

$$\dim \operatorname{Null}(P - I_n) = 1.$$

All other eigenvalues λ satisfy $|\lambda| < 1$.

There is a 1-eigenvector $\mathbf{v}$ which is a probability vector.

[ ]:
```python
import numpy as np

from numpy.random import default_rng

from numpy.linalg import eig, matrix_power

rng = default_rng()

def rand_stoch(n):
    v=np.array([rng.random(n) for i in range(n)])
```

```
    f=np.ones(n)@v
    return (1/f)*v
```

[ ]:
```
T=rand_stoch(8)
np.ones(8)@T
```

[ ]:
```
e_vals,e_vecs = eig(T)
```

[ ]:
```
e_vals
```

[ ]:
```
w=e_vecs[:,0]
v = (1/(np.ones(8)@w))*w
v
```

[ ]:
```
vv=(matrix_power(T,200)[:,0])
```

[ ]:
```
vv
```

[ ]:
```
v-vv < 1e-7*np.ones(8)
```

[ ]:
```
(v-vv < 1e-7*np.ones(8)).all()
```

7.1 How to rank-order the entries in a vector??

Python can sort – but sometimes you don't just want the sorted values.

[ ]:
```
vv
```

[ ]:
```
ll = [(i,vv[i]) for i in range(8)]
ll
```

[ ]:
```
ll.sort(key=lambda x:(-1)*x[1])
```

[ ]:
```
ll
```

[ ]:
```
def top_ten(n,it):
    T=rand_stoch(n)
    vv=matrix_power(T,it)[:,0]
    ll=[(i,vv[i]) for i in range(n)]
    ll.sort(key=lambda x:(-1)*x[1])
    iter_string = "\n".join([f"{l[0]:3d} - {l[1]:.5f}" for l in ll[0:10]])

    e_vals,e_vecs = eig(T)
    w=e_vecs[:,0]
    ww = (1/(np.ones(n)@w))*w
    kl=[(i,ww[i]) for i in range(n)]
    kl.sort(key=lambda x:(-1)*x[1])
    eig_string = "\n".join([f"{l[0]:3d} - {l[1]:.5f}" for l in kl[0:10]])
```

```
    return iter_string + "\n\n" + eig_string
```

[ ]:
```
print(top_ten(50,2))
```