Hello, markov chains. Notes include basic definitions, computational classes, stationary states, notes on google page rank, random walk. i do not like probability

Some mushroom photo i took. I ran 18mm with the biggest apperature. a little out of focus. no idea what mushroom tho

1 Notes

A **Markov chain** is a mathematical model that describes a sequence of possible events (states) where the outcome of each event depends only on the state of the previous event. This property is called the *Markov property*, meaning that the future state depends only on the present state, not on the sequence of events that preceded it.

2 Definitions

I will define markov chains more in tune with the definitions found in classical automata theory. (Why? bc why not)

- **State Space**: The set of all possible states in the Markov chain, denoted by S. For the sake of simplicy, let us assume that (for now) $|S| = n < \infty$. Or there is only a finite amount of states.
- **Transition Matrix***: A matrix δ where the element δ_{ij} is the probability of transitioning from state i to state j. Because this is in the space of probabilites, the sum of the columns must be 1.
- **Initial Distribution**: A vector $\underline{\mathbf{x_0}}$ representing the probabilities of starting in each state.

Thus let us represent markov chains with a 3-tuple (S, δ, x_0). We can define a computation up until t to be the lists of the vectors $x_0, x_1, x_2, ..., x_t$.

Note: $*$ The easiest way to verify that the transition matrix is valid is to add up the values of the columns. For example, given this matrix

$$\delta = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n} \\ p_{2,0} & p_{2,1} & \cdots & p_{2,n} \\ p_{3,0} & p_{3,1} & \cdots & p_{3,n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n,0} & p_{n,1} & \cdots & p_{n,n} \end{bmatrix}$$

$$\forall a \mid \sum_{i=0}^{n} p_{i,a} = p_{0,a} + p_{1,a} + ... + p_{n,a} = 1$$

A more algorithmic way to check this is through multiplying the n-dimensional one vector transposed $\underline{1_n}^T \delta = \underline{1_n}^T$. The math is the same, but the latter way is for more computational checking easily implemented in numpy.

We can notice that the following is true:

$$P(X_{t+1} = s \in S \mid \underline{x_t}, \underline{x_{t-1}}, \ldots, \underline{x_0}) = P(X_{t+1} = s \in S \mid \underline{x_t})$$

Or in other words, the **P**, or probability, of state s conditional on all past computation is actually equivalent to just the last computation. This property is called **memoryless**.

2.1 Basic Example

Let us have the following problem. *On a given day in winter, the probability that it will rain depends on the last day. If it rains today we know that there is a 75% it will rain tomorrow. If it doesn't rain today, then there will be a 25%, it rains on the tomorrow. Given that today it rains, what is the chance that it rains a week from today?*

2.1.1 Naive method

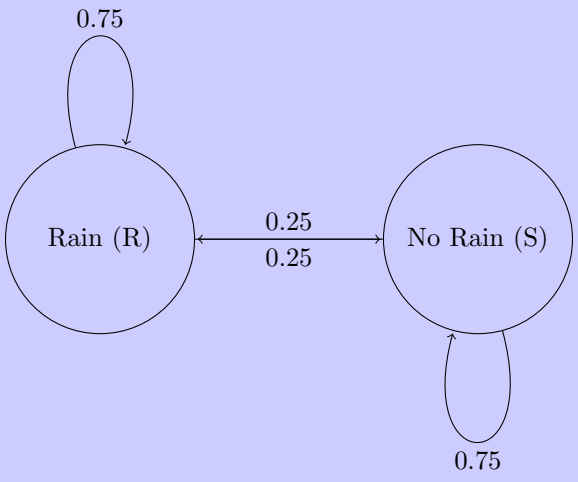
To solve this problem without markov chains, lets try to solve for the first few days:

1. On the day it is $P(\text{rains on day } 0) = 1$. This is the inital state.
2. On the first day after(or tomorrow), we know the chances are equivalent to a $P(\text{rain}) = 0.75$ and $P(\text{no rain}) = 0.25$. Easy
3. On the second day, we know the chances are equivalent to $P(\text{rains} \mid \text{rains on day } 1) = 0.75 * 0.75$ and $P(\text{rains} \mid \text{no rain on day } 1) = 0.25 * 0.25$. Adding the two together we get the following probaility of rain being 0.625
4. Repeating the process for the third day, we have to calculate a total of four combinations: [RRR, RRS, RSR, RSS] out of the total eight combinations of [RRR, RRS, RSR, RSS, SRR, SRS, SSR, SSS] where R means it rains and S means it did not rain. We can sum up the following to get a probability of 0.

$$\begin{aligned}
P(R_3) &= P(R_3|R_2,R_1)P(R_2 \cap R_1) + P(R_3|R_2,S_1)P(R_2 \cap S_1) \\
&\quad + P(R_3|S_2,R_1)P(S_2 \cap R_1) + P(R_3|S_2,S_1)P(S_2 \cap S_1) \\
&\text{by memoryless principal or only the the day before effects the current} \\
&= P(R_3|R_2)P(R_2 \cap R_1) + P(R_3|R_2)P(R_2 \cap S_1) \\
&\quad + P(R_3|S_2)P(S_2 \cap R_1) + P(R_3|S_2)P(S_2 \cap S_1) \\
&= 0.75P(R_2 \cap R_1) + 0.75P(R_2 \cap S_1) + 0.25P(S_2 \cap R_1) + 0.25P(S_2 \cap S_1) \\
&= 0.75(0.75 * 0.75) + 0.75^2 * 0.25 + 0.25 * 0.25 * 0.75 + 0.25 * 0.75 * 0.25 \approx 0.65625
\end{aligned}$$

2.1.2 With markov chains

Note: Following is the state diagram



The three tuple (S, δ, x_0) are as follows:

1. $S = \{R, S\}$
2. $\delta = \begin{bmatrix} \delta(R_t \mid R_{t-1}) = 0.75 & \delta(S_t \mid R_{t-1}) = 0.25 \\ \delta(R_t \mid S_{t-1}) = 0.25 & \delta(S_t \mid S_{t-1}) = 0.75 \end{bmatrix}$
3. $\underline{x_0} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ where $x_0[0]$ means rain and $x_0[1]$ means no rain

Then to calculate the days as follows:

1. **Day 1:** To calculate, we can multiply the probaility matrix.

$$\delta * \underline{x_0} = \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.625 \\ 0.375 \end{bmatrix}$$

   The interpretation of the resulting vector x_1 is there is a 0.75 chance it will rain and a 0.25 chance it will not rain.

2. **Day 2:** We can apply the same idea,

$$\underline{x_2} = \delta * \underline{x_1} = \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix} \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}$$

   The interpretation of the resulting vector x_2 is there is a 0.625 chance it will rain and a 0.375 chance it will not rain.

3. **Any day:** However, on the topic of computation, we can apply subsitution because we know that $\underline{x_1} = \delta \underline{x_0}$ to get

$$\underline{x_2} = \delta * \underline{x_1} = \delta(\delta * \underline{x_0}) = \delta^2 \underline{x_0}$$

   So we can find the t-th day, via jus the formula

$$\underline{x_t} = \delta^t \underline{x_0}$$

   The proof is trivial because it is just applying the subsitution over an over again. But checking if this indeed works for the third day,

$$\begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix}^3 \underline{x_0} = \begin{bmatrix} 0.5625 & 0.4375 \\ 0.4375 & 0.5625 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5625 \\ 0.4375 \end{bmatrix}$$

We can analyze how much easier this is for calculating probalities with more complexity. As the number of possible states increases, say instead of rain or no rain there were levels of precipitation, the naive calculations becomes harder and harder to compute. But for markov chains, the complexity scales with an increase in dimensions. To calculate t-th iteration, in this method it will only take $O(n^3t + n^2) = O(n^3t)$ where n is the size of the matrix or the the number of states($n = |S|$) and t is the desired time from the two steps

1. The first step is multiplying the matrix t times. While speed ups exists through algorithms like strassens, it is trivially useful here. Compute $n-$square matrix multiplication is equivalent to computing the individual positions in the matrix dot producted with the respective row and column. There are n^2 positions to compute and the dot product is done in linear time.

2. The extra n^2 calculation is trivial to the class, but it should be noted. This is the last computation where you multiply the resulting matrix with the start state vector. This multiplication is equivalent to multiplying $(n \times n)(n \times 1)$ or quadratic time.

For asymptotics, this is linear assuming a large enough t as n never changes.

3 Maybe speed up from $O(n^3t)$ to $O(log(t)n)$

On the topic of speeding up computation, we can acutally calculate this even faster with some clever understanding of square matrices. More specifically, it requires the understanding of **diagonalization** and fast ways to exponentiate with integer values.

3.1 Real Symmetric Matrices

We should recall the following fact, that all real square symmetric matrices are diagonalizable. In more simple terms:

Note: A matrix is diagonalizable M implies there exists two matrices P, D such that it satisfies the following

$$M = PDP^{-1}$$

Where

1. P is the change of basis matrix. It is also the matrix of eigenvectors.
2. D are the eigenvectors in diagonal form or it must look as the following

$$\begin{bmatrix} \lambda_1 & 0 & 0 & ... & 0 \\ 0 & \lambda_2 & 0 & ... & 0 \\ 0 & 0 & \lambda_3 & ... & 0 \\ 0 & 0 & 0 & ... & \lambda_n \end{bmatrix}$$

3. P^{-1} exists because eigenvectors span the existing space so it must invertible.

Multiplication is a lot easier now. If we want to find the square of a matrix it is

$$M^2 = (PDP^{-1})^2 = (PDP^{-1})(PDP^{-1}) = PD^2P^{-1}$$

Because the inner $P^{-1}P$ cancels. We can see raising to any power of t is just this method repeated over and over again to get

$$M^t = (PDP^{-1})^t = (PDP^{-1})(PDP^{-1})(PDP^{-1})^{t-2} = PD^2P^{-1}(PDP^{-1})^{t-2} \\ = ... = PD^tP^{-1}$$

In a section, we will see how fast this could be done.

A basic fact about all $n-$dimensional square matrices is that there must exists n eigenvalues or $\lambda_1, \lambda_2, ...\lambda_n$ must all exist. This simple fact can be derived by how determinants are calculated. Recall to calculate determinants, is equivalent to finding when the determinant of $\det(M - \lambda) = 0$ for all λ's. We will get a polynomial(aka characteristic polynomial) and by some fact in algebra, it states that all polynomials of degree n must n complex-roots with multiplicity. **Ok let us assume you have checked that the matrix can be decomposed into the following form then speed ups are immeasurable. Turning a linear problem into a logarithmic problem**

3.2 Speeding up exponentiation

Assume you are given a number and a exponent(integer), to calculate a^n with a the naive method of $a^n = a(a^{n-1}) = a * a * a^{n-2} = ...$ would require $O(n)$ linear time. However, we see that a better way by binary exponentiation exists.

(1) Binary Exponentiation given a^n:
BinEXP(a,n)

1. If $n = 1$ or $n = 0$: return a or 1
2. Return BinEXP(a,n//2) * BinEXP(a,n//2) + residual if n is odd

So for example the calculation of a^{125} are the following functional calls

1. $125 \rightarrow 62$

2. $62 \to 31$
3. $31 \to 15$
4. $15 \to 7$
5. $7 \to 3$
6. $3 \to 1$

A total of $O(log(n))$ time.

3.3 In conclusion

So let us assume that the markov probability matrix is diagonalizable, we only have to run a decomposition of the matrix into it's diagonal form(P, D) and then exponentiate all the eigenvalues(the λ's) through our method to have a total of only $O(log(t))$ time.

Note: We should definitely note that not all matrices are diagonalizable just because they are square. They exist. For example, the following have no diagonalization

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

The reasons are different in every case. For example, the first case is the lack of linear independence and the second is the lack of dimensions.

3.4 Previous Example

In the previous example withe probability matrix

$$\delta = \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix}$$

This matrix is diagonalizable(because it is real symmetric). The following is the decomposition

$$\delta = PDP^{-1} = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

So to calculate the t-th day we just have to do the following

$$x_t = \delta^t x_0 = (PDP^{-1})^t x_0 = PD^t P^{-1} x_0 = P \begin{bmatrix} 0.5^t & 0 \\ 0 & 1^t \end{bmatrix} P^{-1} x_0$$

This takes $log(t)$ time.

4 Stationary Distribution

In probability, there is the basic idea of long runs. For example, when gambling, you always lose because the house has an edge. Some people do not understand: see `https://www.instagram.com/mcmeatrocket/reels/`. The proper definition for stationary distribution is

Note: The **Stationary Distribution** π is $\underline{\pi} = \lim_{n\to\infty} \delta^n \underline{x_0}$ or when $\delta\underline{\pi} = \underline{\pi}$

The idea is that after a long time, one more time evolution from $t \to t+1$, there is no effect on what x_{t+1} is. This is a statement about convergence and not always a statement about exact values about finite times. There can exist situtations where $\delta^n x_0$ never equals said distribution but converges to it. Other names for the following property are **stable state** or **steady state** depending on the domain.

4.1 Solving for stationary distributions

We can solve for the steady state through the following intuition. We know that $\delta\underline{\pi} = \underline{\pi}$ then

$$\delta\underline{\pi} - \underline{\pi} = \underline{0} \leftrightarrow (\delta - I_{n\times n})\,\underline{\pi} = \underline{0}$$

Now there are a lot of ways to calculate this.

1. **Naive method** Just multiply the matrix over and over again until is numerically looks stable. Obviously multipying the matrix once will not work, but like 1000 might work or 10^{10}. At some point it would look stable.

2. **Eigenvector method** Ok so recall that if $\underline{v}$ is an eigenvector then, $M\underline{v} = \lambda\underline{v}$ for a given eigenvalue λ. λ is an eigenvalue if and only if $det(M - \lambda I_{n\times n}) = 0$. So from the above we know that $\underline{\pi}$ must one of the eigenvalues of said space generated by the δ matrix. Then we calculate all the eigenvalues and check each of them one by one to see if the corresponding eigenvalue is 1. If it is 1, then we have found the distribution.

Here is the example from above with the diagonalization decomposition.

$$\delta = PDP^{-1} = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

We can see that 1 is an eigenvalue associated with the corresponding eigenvector $\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$. Thus, we know as we continue to do this matrix multiplication, it will eventually reach the stable state of $\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$. Or after a long time, the chances of rain is only a fair coin flip. However, we should not that winter does not last forever.

Note: If you don't believe it, here are the following calculation of $\delta^{t=i}\underline{x_0}$ with wolfram alpha.

1. **t=10**

$$\begin{bmatrix} 0.500488 & 0.499512 \\ 0.499512 & 0.500488 \end{bmatrix} \underline{x_0} \tag{1}$$

2. **t=100**

$$\begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \underline{x_0} \tag{2}$$

The floating points get too small to even track at t=100.

5 Google PageRank

Google Page Rank is a unique application of markov chains combined with some interpretations of eigenvalues. The basic problem that PageRank is trying to solve is that given a set of pages that the user might want to see, how should they rank it? For example, if the user searches up Markov Chain, should a random blog be shown at the top or the wikipedia page.

5.1 Example

For the stake of this example, let us only have the following pages in our space, [Wikipedia page on Markov Chains(**W**), Blogpost(**B**), University PDF(**P**), Amazon Textbook(**A**), Useless pages(**X,Y,Z**)]. We can represent each of these as a state in out markov chain or $S = \{W, B, P, A, X, Y, Z\}$ and $|S| = 7$.
Each of these pages is hyper linked to a few other links.

6 Finite-State Markov Chains

7 Markov Property

The key feature of a Markov chain is the Markov property:

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = k, \ldots, X_0 = x_0) = P(X_{n+1} = j \mid X_n = i)$$

This states that the probability of transitioning to the next state j depends only on the current state i and not on past states.

8 Transition Matrix

A Markov chain is often described by a **transition matrix** P, where each entry P_{ij} represents the probability of transitioning from state i to state j. The matrix satisfies:

$$P_{ij} \geq 0, \quad \sum_j P_{ij} = 1 \quad \text{for all } i$$

9 Examples

9.1 Example 1: A Simple Weather Model

Consider a Markov chain where the states represent weather conditions: Sunny and Rainy. The transition matrix is given by:

$$P = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix}$$

where:

- The probability of staying sunny is 0.8, and the probability of transitioning from sunny to rainy is 0.2.
- The probability of transitioning from rainy to sunny is 0.4, and the probability of staying rainy is 0.6.

9.2 Example 2: A Random Walk on a Line

Consider a Markov chain modeling a random walk on a line with states $S = \{0, 1, 2, 3\}$. The transition matrix might look like:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

10 Stationary Distribution

A stationary distribution π satisfies:

$$\pi P = \pi$$

This means that if the system starts in the stationary distribution, it will remain in the stationary distribution after one step. To find π, solve the system of equations:

$$\pi_j = \sum_i \pi_i P_{ij} \quad \text{for all } j$$

along with the normalization condition:

$$\sum_j \pi_j = 1$$

11 Applications of Markov Chains

- **PageRank Algorithm**: A Markov chain is used to model the movement of a "random surfer" on the web to rank webpages.
- **Queueing Theory**: Markov chains are used to model systems with queues, such as customer service lines or network traffic.
- **Population Models**: Used to model the growth or decline of populations in biology or economics.
- **Hidden Markov Models (HMMs)**: Used in speech recognition, bioinformatics, and other fields.

12 Conclusion

Markov chains provide a powerful framework for modeling systems that evolve over time. Understanding the transition matrix, stationary distributions, and the Markov property is essential for applying these models to real-world problems.