Math 4570 Matrix Methods for DA and ML

Dynamical System Examples

Instructor: He Wang
Department of Mathematics
Northeastern University

- **Markov decision processes**

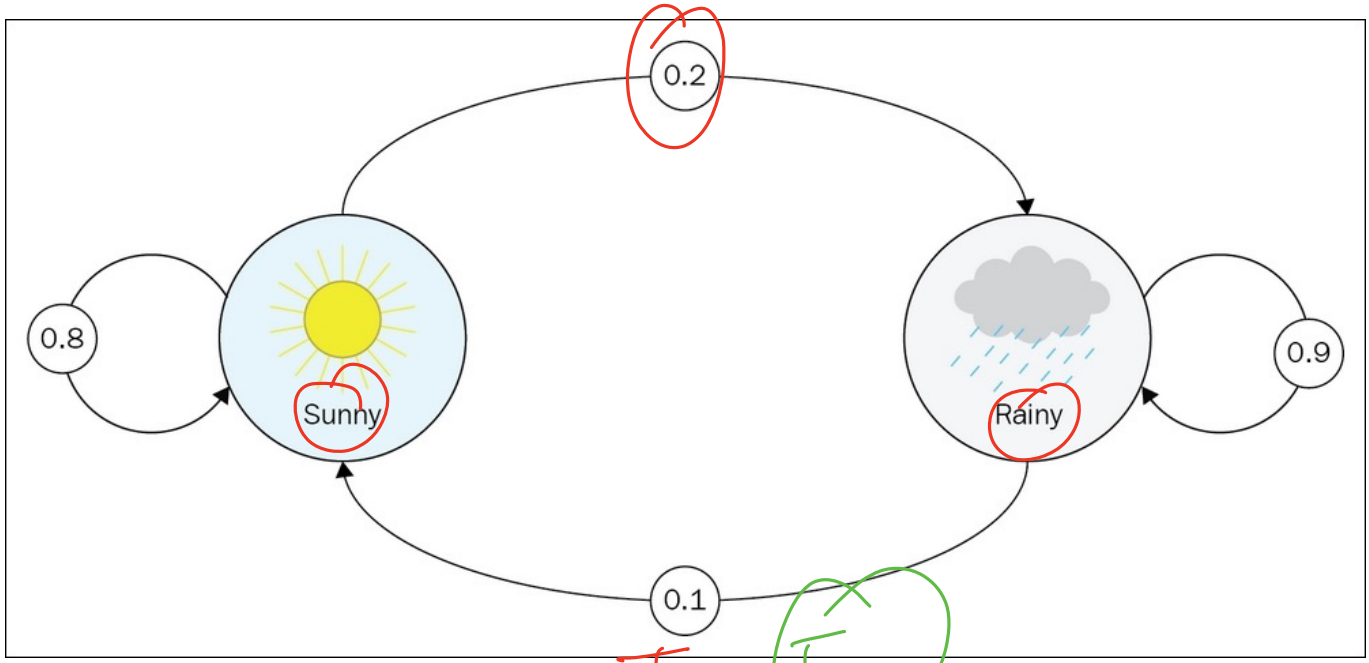
  Theoretical foundation of Reinforcement Learning, Google Search's Page Rank algorithm, population models, economic models, etc.

Example 1:

Simplest model of the weather in some city, we can observe the current day as **sunny** or **rainy**, which is our **state space**.

A sequence of observations over time forms a chain of states, such as [sunny, sunny, rainy, sunny, …]. For example, suppose we know

- If we have a sunny day, then there is an 80% chance that the next day will be sunny and a 20% chance that the next day will be rainy.
- If we have a rainy day, then there is a 10% probability that the weather will be sunny and a 90% probability of the next day being rainy.



To

From

	sunny	rainy
sunny	0.8	0.2
rainy	0.1	0.9

From

To

$$[\cdot \quad \cdot] = [P_1 \quad P_2] \begin{bmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{bmatrix}$$

$$[\quad] = \begin{bmatrix} 0.8 & 0.1 \\ 0.2 & 0.9 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$$

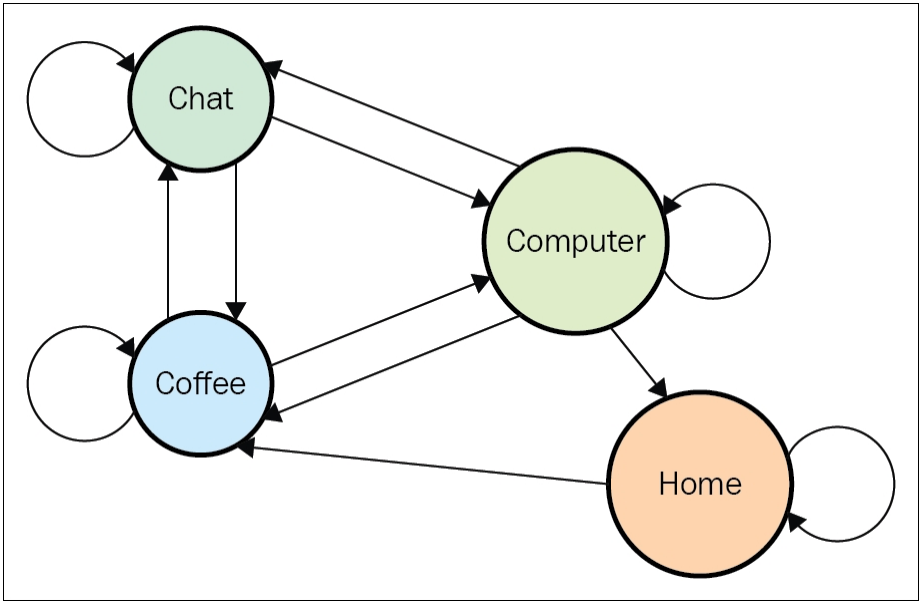
Our Markov model to represent only the cases when a sunny day can be followed by a rainy one, with the same probability, regardless of the amount of sunny days we've seen in the past. It's not a very realistic model, as from common sense we know that the chance of rain tomorrow depends not only on the current condition, but on a large number of other factors, such as the season, our latitude, and the presence of mountains and sea nearby. It was recently proven that even solar activity has a major influence on weather. So, our example is really naïve, but it's important to understand the limitations and make conscious decisions about them.

Of course, if we want to make our model more complex, we can always do this by extending our state space, which will allow us to capture more dependencies in the model at the cost of a larger state space. For example, if you want to capture separately the probability of rainy days during summer and winter, then you can include the season in your state. In this case, your state space will be
[sunny+summer, sunny+winter, rainy+summer, rainy+winter] and so on.

Example 2: Office Worker

State Space:

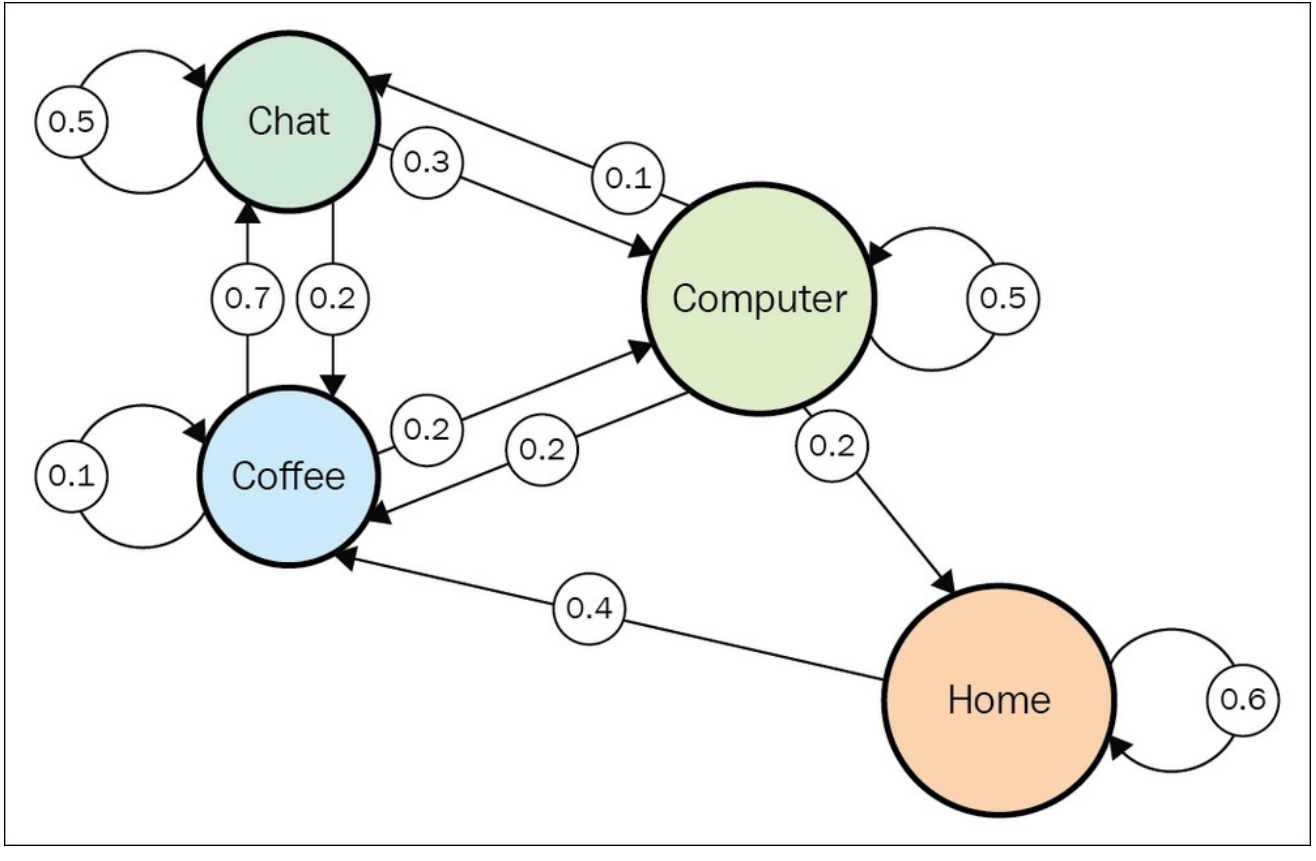
Home: He's not at the office
Computer: He's working on his computer at the office
Coffee: He's drinking coffee at the office
Chatting: He's discussing something with colleagues at the office



We expect that his work day usually starts from the **Home** state and that he always starts his work day with **Coffee**, without exception (no **Home** → **Computer** edge and no **Home** → **Chatting** edge).
The preceding diagram also shows that work days always end (that is, the going to the **Home** state) from the **Computer** state. The transition matrix for the preceding diagram is as follows:

	Home	Coffee	Chat	Computer
Home	60%	40%	0%	0%
Coffee	0%	10%	70%	20%
Chat	0%	20%	50%	30%
Computer	20%	20%	10%	50%

The transition probabilities could be placed directly on the state transition graph, as shown here:



In practice, we rarely have the luxury of knowing the exact transition matrix. A much more real-world situation is when we have only observations of our systems' states, which are also called episodes:

- home → coffee → coffee → chat → chat → coffee → computer → computer → home
- computer → computer → chat → chat → coffee → computer → computer → computer
- home → home → coffee → chat → computer → coffee → coffee

It's not complicated to estimate the transition matrix by our observation; we just **count** all the transitions from every state and normalize them to a sum of 1. The more observation data we have, the closer our estimation will be to the true underlying model.

References:

Deep Reinforcement Learning Hands-On published by Packt.

https://github.com/PacktPublishing/Deep-Reinforcement-Learning-Hands-On