MATH 242: Friday May 1, 2020

TODAY: Markov Chain Monte Carlo (MCMC)

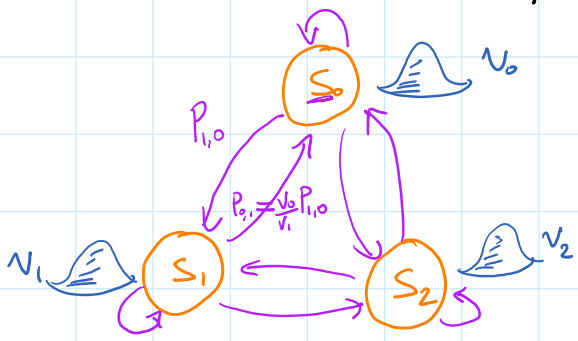
First: How to print fewer decimal digits in Numpy matrices?

```
numpy.set_printoptions(precision = 3)
```

RECALL: Given a steady-state vector $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$

we can compute a transition matrix P

for a Markov chain with steady-state vector $\mathbf{v}$.



NOW: Suppose we have a <u>huge</u> number of states.

Maybe $n = \underbrace{10^9}_{\text{1 Billion}}$. The matrix P would need 10^9 rows and 10^9 columns, so 10^{18} entries, far too large for our computer's memory!

In general: we will have a function that gives the relative frequencies for each state.

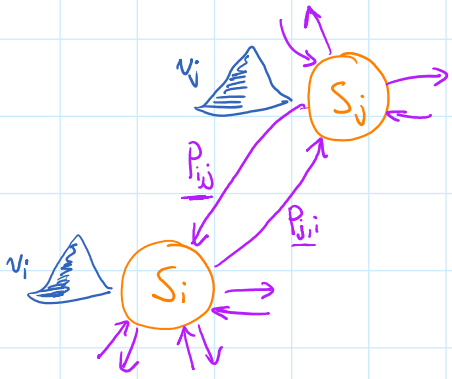
We will compute probabilities, but not the entire transition matrix.

EXAMPLE:

S_i has frequency ν_i

S_j has frequency ν_j

Assume $\nu_j \leq \nu_i$



Transition probability $S_j \to S_i$ is $0 < P_{i,j} < 1$.

So the amount of sand moving from S_j to S_i is $P_{i,j}\,\nu_j$.

Likewise, the amount of sand moving from S_i to S_j is $P_{j,i}\,\nu_i$.

So, to keep things balanced: $P_{i,j}\,\nu_j = P_{j,i}\,\nu_i$

$$P_{i,j}\left[\frac{\nu_j}{\nu_i}\right] = P_{j,i}$$

key

$$1 \geq \frac{\nu_j}{\nu_i}$$

ν_j ← smaller

ν_i ← larger

Markov Chain Monte Carlo: Algorithm for simulating a random walk on a Markov Chain given only states, allowable transitions, and the relative frequencies.

Let S_j be the current state.

Choose a neighboring state S_i at random.

↳ uniformly? or by some probability?

- If $\nu_j \leq \nu_i$, then move to S_i.

  ⇦ The random walk wants to move to states with greater relative frequency.

- If $\nu_j > \nu_i$, then move to S_i with probability $\rho = \frac{\nu_i}{\nu_j}$.

  Flip a coin that lands heads with probability ρ.
  If heads, then move to S_i. If tails, stay at S_j.

  ⇖ The random walk will consider moving to states with lower relative frequency.