# Course notes, CS664, 9/30/04

- Administrivia: first exam returned.

## METROPOLIS ALGORITHM

- Metropolis Algorithm, parameterized by energy $E$, temperature $T$.

- Start somewhere (i.e., with a labeling $f$ of the image, that has some energy).

  1. Generate a new random state change $f'$ (i.e., change the label of a pixel from $f$). [This step is called sampling]
  2. Let $\Delta E = E(f') - E(f)$. If $\Delta E < 0$ (downhill move), set $f = f'$. Otherwise (uphill move), with probability proportional to $\exp(-\frac{\Delta E}{T})$, set $f = f'$.

- What does $y = \exp(-x/T)$ look like, as a function of $T$? Two asymptotes ($x$ and $y$ axis).

- For historical reasons, $P(x) = C\exp(-x/T)$ is called the Boltzmann (or Gibbs) distribution. ($C$ is a normalization constant.)

- Note that we can do nothing (reject an uphill move), oddly enough

- To understand Metropolis intuitively, consider the extreme values of $T$. At high $T$, any move is accepted, and we do random search. At low $T$, we only accept downhill moves, and do hill-climbing.

- The Metropolis algorithm is *randomized*, in that the output is non-deterministic. The output of a randomized algorithm is a distribution over its possible outputs.

- Question: if you run Metropolis, what is the output distribution, in terms of energy?

- (Amazing) answer: the Boltzman distribution. To understand this fact, we need to talk a bit about Markov chains.

## MARKOV CHAINS

- A Markov chain, also known as a stochastic automaton, is a

directed graph with probabilities on the edges. You can think of this as a generalization of a deterministic automaton (finite state machine).

- The sum of the outgoing probabilities is 1 (note that there can be self-loops). Aka: you have to go somewhere!

- Markov chains are memoryless, in that where you go only depends on where you are, not on how you got there.

- Example: a 2-state markov chain.

$$\begin{pmatrix} 1/3 & 2/5 \\ 2/3 & 3/5 \end{pmatrix}$$

- In a standard automaton, at a given point in time you are in a single state. Here, you could be in different states, so the equivalent is a more general and involves probability.

- Consider a probability distribution over states of the Markov chain, which you can think of as a set of non-negative numbers

that sum to 1. Intuitively, if you had a very large number of possible situations, the number associated with a state is the fraction of these situations in which it is in that state.

- Another useful intuition is to think of dividing up 1 liter of water among all the states. Or, very computationally, think of a state vector of these numbers.

- If you are in a certain distribution (state vector) and take a single step, you move to a new distribution. The update is according to the edge weights. In our little example,

$$P_1^{\text{new}} = \frac{1}{3} \cdot P_1^{\text{old}} + \frac{2}{5} \cdot P_2^{\text{old}}$$
$$P_2^{\text{new}} = \frac{2}{3} \cdot P_1^{\text{old}} + \frac{3}{5} \cdot P_2^{\text{old}}$$

- The water metaphor is helpful; take the water in a particular state at time 1 and ship it out according to the outgoing edges. This is being done simultaneously in parallel at all states, and results in the new state vector at time 2.

- Now consider how to determine how much water a state $s$ has at time 2. It is the sum over the states $s'$ of how much water each

$s'$ sends to $s$. Each one sends the product of how much water $s'$ had at time 1 and the fraction of this water sent to $s$, which is the weight of the edge from $s'$ to $s$.

- This linear sum is easy to write in matrix form. We will define a *stochastic vector* as containing non-negative elements that sum to 1. A stochastic matrix has stochastic columns.

- A state vector $\pi$ for a Markov chain is a stochastic vector. To compute the state vector after another step we multiply it by the stochastic vector which is the transition matrix $K$.

- We write this as $\pi_n = K\pi_{n-1}$. Obviously, $\pi_n = K^n\pi_0$.

- For our example Markov chain,

$$K = \begin{bmatrix} 1/3 & 2/5 \\ 2/3 & 3/5 \end{bmatrix}$$

- Note that

$$K \begin{vmatrix} 3/8 \\ 5/8 \end{vmatrix} = \begin{vmatrix} 3/8 \\ 5/8 \end{vmatrix}.$$

We will define $\pi^*$ to be *stationary* for $K$ if $K\pi^* = \pi^*$. Not all Markov chains have such a $\pi^*$ (also called the fixed point, or attractor).

- We will say that $K$ *converges to* $\pi^*$ if

$$\forall n, \pi \lim_{n \mapsto \infty} K^n \pi = \pi^*$$

- Not all Markov chains converge; consider the "flipping" chain

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

- If the Markov chain is *ergodic*, which means that there is a path from every node to every other node (strongly connected) plus there is at least one self loop, then there is a unique distribution $\pi^*$ that $K$ converges to.

- This is called the Peron-Frobenius theorem, and I know 1 fairly reasonable proof of it and lots of complex ones.

## METROPOLIS MARKOV CHAIN

- OK, back to Metropolis. Each labeling in the Metropolis algorithm can be thought of as a node in a Markov chain. The chain is ergodic (where do the self loops come from?)

- With a little hacking at the boundaries, the chain is actually uniform, in that all nodes have the same out-degree.

- We can now show that the Boltzman distibution is $\pi^*$ for the Markov chain that Metropolis generates. Consider two states $R, S$ and w.l.o.g. assume $E(S) \leq E(R)$.

- We will do this using the "frequency interpretation" of probability, which says that if the probability of an event is $p$, if you run $N$ experiments (outcomes) the number of times the event occurs is $Np$ in the limit.

- Imagine a large number of outcomes of Metropolis (say, $N$ which is very big). Each outcome has an associated energy. We can show that when the number of outcomes with energy $E$ obeys the Boltzman distribution (i.e., is $N \cdot C \exp(-E/T)$), taking a

step does not change this distribution.

- More precisely, we will show that for any two states with moves between them, the number of outcomes that move from $R$ to $S$ is the same as the number that move from $S$ to $R$. We will write these as $\nu_{RS}, \nu_{SR}$.

- We have

$$\nu_{RS} = N \cdot C \exp(-E(R)/T) \cdot P(\text{generate downhill move})$$

$$\nu_{SR} = N \cdot C \exp(-E(S)/T) \cdot P(\text{generate uphill move}) \cdot \exp(-\Delta E/T)$$

where the last factor comes from the fact that we don't always take the uphill moves.

- Since the chain is uniform the chances of move generation are the same (drop them).

- We plug in $\Delta E = E(R) - E(S)$ to get

$$\nu_{SR} = N \cdot C \exp(-E(S)/T) \cdot \exp((E(S) - E(R))/T)$$

Cancel some terms and we are done! (Incidentally, this is pretty much exactly the 1953 proof, minus some terminology from statistical mechanics.)