Biostatistics 615/815 Lecture 11:
More Graph Algorithms
Hidden Markov Models

Hyun Min Kang

February 10th, 2011

Annoucement

- Homework 3 is announced, due next Tuesday.
- Try to install **boost** library by today and ask technical questions tomorrow if there is any

Recap : boost library

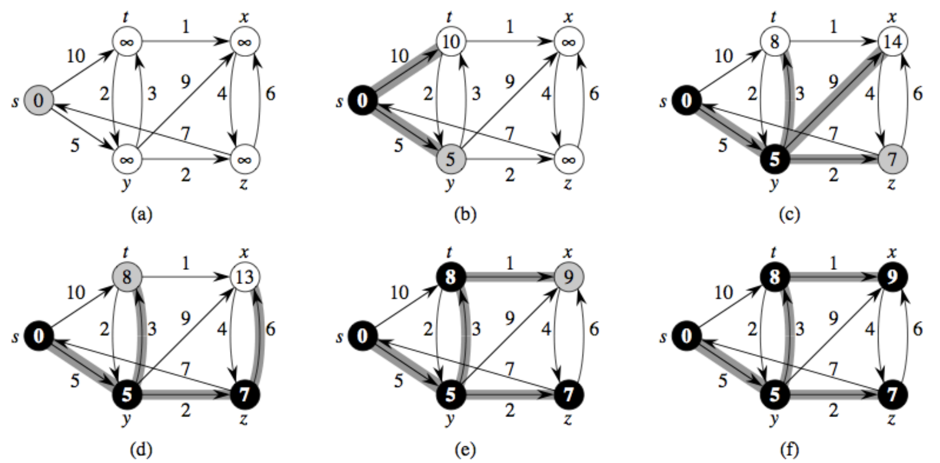
```
#include <iostream>
#include <boost/tokenizer.hpp>
#include <string>
using namespace std;
using namespace boost;

int main(int argc, char** argv) {
  // default delimiters are spaces and punctuations
  string s1 = "Hello, boost  library";
  tokenizer<> tok1(s1);
  for(tokenizer<>::iterator i=tok1.begin(); i != tok1.end() ; ++i) {
    cout << *i << endl;
  }
  // you can parse csv-like format
  string s2 = "Field 1,\"putting quotes around fields, allows commas\",Field 3";
  tokenizer<escaped_list_separator<char> > tok2(s2);
  for(tokenizer<escaped_list_separator<char> >::iterator i=tok2.begin();
       i != tok2.end(); ++i) {
    cout << *i << endl;
  }
  return 0;
}
```


Recap : Dijkstra's algorithm

Algorithm $\textsc{Dijkstra}$

Data: G : graph, w : weight, s : source
Result: Each vertex contains the optimal weight from s
$\textsc{InitializeSingleSource}$(G,s);
$S = \emptyset$;
$Q = G.V$;
while $Q \neq \emptyset$ **do**
  $u = \textsc{ExtractMin}(Q)$;
  $S = S \cup \{u\}$;
  for $v \in G.Adj[u]$ **do**
    $\textsc{Relax}(u, v, w)$;
  end
end

Recap : Illustration of DIJKSTRA's algorithm



Calculating all-pair shortest-path weights

A dynamic programming formulation

Let $d_{ij}^{(k)}$ be the weight of shortest path from vertex i to j, for which intermediate vertices are in the set $\{1, 2, \cdots, k\}$.

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & k = 0 \\ \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & k > 0 \end{cases}$$

Floyd-Warshall Algorithm

Algorithm FLOYDWARSHALL

Data: W : $n \times n$ weight matrix
$D^{(0)} = W$;
for $k = 1$ **to** n **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**
      $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$;
    end
  end
end
return $D^{(n)}$;

Summary: shortest path finding algorithms

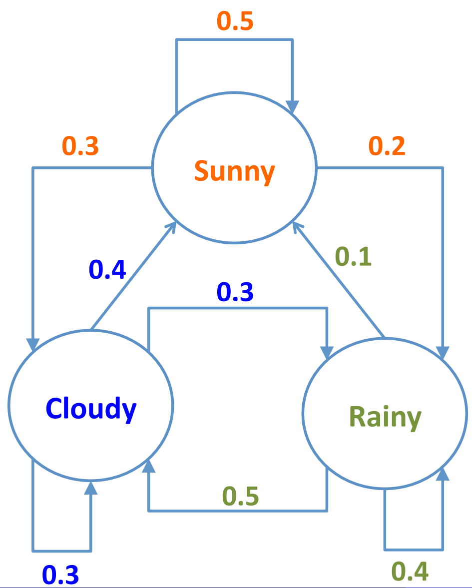
Dijkstra's algorithm

- $\Theta(|V| \log |V| + |E|)$ dynamic programming algorithm.
- Compute optimal path from a single source to each node
- Track optimal path from the closest node from the source, and expand to adjacent node

Floyd-Warshall algorithm

- $\Theta(|V|^3)$ All-pair shortest path finding algorithms with non-negative weights
- Use the fact that the maximum length of each possible optimal path is $|V|$.
- For each possible pairs of sources and destinations, iteratively update optimal distance matrix $|V|$ times.

Markov Process : An example



Mathematical representation of a Markov Process

$$\pi = \begin{pmatrix} \Pr(q_1 = S_1 = \text{Sunny}) \\ \Pr(q_1 = S_2 = \text{Cloudy}) \\ \Pr(q_1 = S_3 = \text{Rainy}) \end{pmatrix} = \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix}$$

$$A_{ij} = \Pr(q_{t+1} = S_i | q_t = S_j)$$

$$A = \begin{pmatrix} 0.5 & 0.4 & 0.1 \\ 0.3 & 0.3 & 0.5 \\ 0.2 & 0.3 & 0.4 \end{pmatrix}$$

Example questions in Markov Process

What is the chance of rain in the day 2?

$$\Pr(q_2 = S_3) = (A\pi)_3 = 0.24$$

If it rains today, what is the chance of rain on the day after tomorrow?

$$\Pr(q_3 = S_3 | q_1 = S_3) = \left[A^2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right]_3 = 0.33$$

Stationary distribution

$$\begin{aligned} \mathbf{p} &= A\mathbf{p} \\ p &= (0.346, 0.359, 0.295)^T \end{aligned}$$

Markov process is only dependent on the previous state

If it rains today, what is the chance of rain on the day after tomorrow?

$$\Pr(q_3 = S_3 | q_1 = S_3) = \left[A^2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right]_3 = 0.33$$

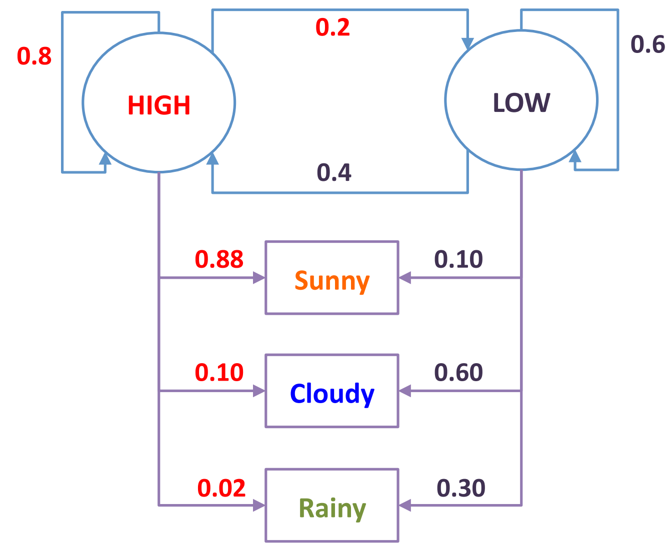
If it has rained for the past three days, what is the chance of rain on the day after tomorrow?

$$\Pr(q_5 = S_3 | q_1 = q_2 = q_3 = S_3) = \Pr(q_5 = S_3 | q_3 = S_3) = 0.33$$

Hidden Markov Models (HMMs)

- A Markov model where actual state is unobserved
  - Transition between states are probablistically modeled just like the Markov process
- Typically there are observable outputs associated with hidden states
  - The probability distribution of observable outputs given an hidden states can be obtained.

An example of HMM



Mathematical representation of the HMM example

States $S = \{S_1, S_2\}$ = (HIGH, LOW)

Outcomes $O = \{O_1, O_2, O_3\}$ = (SUNNY, CLOUDY, RAINY)

Initial States $\pi_i = \Pr(q_1 = S_i)$, $\pi = \{0.7, 0.3\}$

Transition $A_{ij} = \Pr(q_{t+1} = S_i | q_t = S_j)$

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

Emission $B_{ij} = b_{q_t}(o_t) = b_{S_j}(O_i) = \Pr(o_t = O_i | q_t = S_j)$

$$B = \begin{pmatrix} 0.88 & 0.10 \\ 0.10 & 0.60 \\ 0.02 & 0.30 \end{pmatrix}$$

Interesting Questions

- What is the chance of rain in the day 3?
- What is the chance of rain in the day 3, if it rained in the day 2?
- What is the chance of rain in the day 3, if it rained in the day 1 and day 2?
- If the observation was (SUNNY,SUNNY,CLOUDY,RAINY,RAINY) from day 1 through day 5, what would be the mostly likely sequence of states?
- Can we infer the HMM paremeters if we have a large number of observations?

Unconditional marginal probabilities

What is the chance of rain in the day 4?

$$\mathbf{f}(\mathbf{q}_3) = \begin{pmatrix} \Pr(q_4 = S_1) \\ \Pr(q_4 = S_2) \end{pmatrix} = A^3\pi = \begin{pmatrix} 0.669 \\ 0.331 \end{pmatrix}$$

$$\mathbf{g}(o_4) = \begin{pmatrix} \Pr(o_4 = O_1) \\ \Pr(o_4 = O_2) \\ \Pr(o_4 = O_3) \end{pmatrix} = B\mathbf{f}(\mathbf{q}_4) = \begin{pmatrix} 0.621 \\ 0.266 \\ 0.233 \end{pmatrix}$$

The chance of rain in day 3 is 23.3%

Calculating conditional probabilities

What is the chance of rain in the day 2 if it rains in the day 1?

$$\begin{aligned} \Pr(o_2 = O_3 | o_1 = O_3) &= \Pr(o_2 = O_3 | q_2 = S_1) \Pr(q_2 = S_1 | o_2 = O_3) \\ &\quad + \Pr(o_2 = O_3 | q_2 = S_2) \Pr(q_2 = S_2 | o_2 = O_3) \\ &\quad \cdots \end{aligned}$$

This is already quite complicated!

Organizing the likelihood

- Let $\lambda = (A, B, \pi)$
- For a sequence of observation $\mathbf{o} = \{o_1, \cdots, o_t\}$,

$$\begin{aligned} \Pr(\mathbf{o}|\lambda) &= \sum_{\mathbf{q}} \Pr(\mathbf{o}|\mathbf{q}, \lambda) \Pr(\mathbf{q}|\lambda) \\ \Pr(\mathbf{o}|\mathbf{q}, \lambda) &= \prod_{i=1}^{t} \Pr(o_i|q_i, \lambda) = \prod_{i=1}^{t} b_{q_i}(o_i) \\ \Pr(\mathbf{q}|\lambda) &= \pi_{q_1} \sum_{i=2}^{t} a_{q_i q_{i-1}} \\ \Pr(\mathbf{o}|\lambda) &= \sum_{\mathbf{q}} \pi_{q_1} b_{q_1}(o_{q_1}) \prod_{i=2}^{t} a_{q_i q_{i-1}} b_{q_i}(o_{q_i}) \end{aligned}$$

Naive computation of the likelihood

$$\Pr(\mathbf{o}|\lambda) = \sum_{\mathbf{q}} \pi_{q_1} b_{q_1}(o_{q_1}) \prod_{i=2}^{t} a_{q_i q_{i-1}} b_{q_i}(o_{q_i})$$

- Number of possible $q = 2^t$ are exponentially growing with the number of observations
- Computational would be infeasible for large number of observations
- Algorithmic solution required for efficient computation.

Dynammic Programing approach for HMMs

- If each possible q_t is represented as a vertex of graph and $a_{t(t-1)}$ represents edges, then the problem becomes a graph algorithm
- Finding the most likely path given a series of observations is very similar to the Dijkstra's algorithm
- $\Pr(\mathbf{o}|\lambda)$ can also be efficiently calculated using dynamic programming called "forward-backward algorihm"

Forward-backward algorithm

- Define forward probability $\alpha_t(i)$ as

$$\alpha_t(i) = \Pr(o_1, \cdots, o_t, q_t = S_i|\lambda)$$

- $\alpha_t(i)$ can be efficiently computed using dynamic programming
  - $\alpha_1(i) = \pi_i b_i(o_1)$
  - $\alpha_t(i) = \sum_{j=1}^n \alpha_{t-1}(j) a_{ij} b_i(o_t)$
  - $\Pr(\mathbf{o}|\lambda) = \sum_{i=1}^n \alpha_t(i)$
- Time complexity is $\Theta(n^2 t)$.

Forward-backward algorithm (cont'd)

- Backward probability $\beta_t(i)$ as

$$\beta_t(i) = \Pr(o_{t+1}, \cdots, o_T | q_t = S_i, \lambda)$$

- $\beta_t(i)$ can also be efficiently computed using dynamic programming
  - $\beta_T(i) = 1$
  - $\beta_t(i) = \sum_{j=1}^n a_{ji} b_j(o_{t+1}) \beta_{i+1}(j)$
- Time complexity is $\Theta(n^2(T-t))$.

Forward-backward algorithm (cont'd)

- We can infer the conditional probability of each state given observations by

$$\begin{aligned}\gamma_t(i) &= \Pr(q_t = S_i|\mathbf{o}, \lambda) \\ &= \frac{\Pr(\mathbf{o}, q_t = S_i|\lambda)}{\sum_{j=1}^n \Pr(\mathbf{o}, q_t = S_j|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^n \alpha_t(j)\beta_t(j)}\end{aligned}$$

- Time complexity is $\Theta(n^2 T)$.

Viterbi algorithm

- Finding the most likely trajectory of states given a series of observations
- Want to compute
$$\arg\max_{\mathbf{q}} \Pr(\mathbf{q}|\mathbf{o}, \lambda)$$
- Define $\delta_t(i)$ as
$$\delta_t(i) = \max_{\mathbf{q}} \Pr(\mathbf{q}, \mathbf{o}|\lambda)$$
- Use dynamic programming algorithm to find the 'shortest' path

Viterbi algorithm (cont'd)

Initialization $\delta_1(i) = \pi b_i(o_1)$ for $1 \leq i \leq n$.

Maintenance $\delta_t(i) = \max_i \delta_{t-1}(i) a_{ji} b_j(o_t)$

Termination Max likelihood is $\max_i \delta_T(i)$

Reconstruction How to reconstruct the optimal path?