Biostatistics 615/815 Lecture 9: Dynamic Programming and Hidden Markov Models

Hyun Min Kang

October 2nd, 2012

Minimum edit distance problem

Edit distance

Minimum number of letter insertions, deletions, substitutions required to transform one word into another

An example

FOOD → MOOD → MON^D → MONED → MONEY

Edit distance is 4 in the example above

More examples of edit distance

F	O	O		D
M	O	N	E	Y

A	L	G	O	R		I		T	H	M
A	L		T	R	U	I	S	T	I	C

- Similar representation to DNA sequence alignment
- Does the above alignment provides an optimal edit distance?

A dynamic programming solution

		A	L	G	O	R	I	T	H	M
	0	1	2	3	4	5	6	7	8	9
A	1	0	1	2	3	4	5	6	7	8
L	2	1	0	1	2	3	4	5	6	7
T	3	2	1	1	2	3	4	4	5	6
R	4	3	2	2	2	2	3	4	5	6
U	5	4	3	3	3	3	3	4	5	6
I	6	5	4	4	4	4	3	4	5	6
S	7	6	5	5	5	5	4	4	5	6
T	8	7	6	6	6	6	5	4	5	6
I	9	8	7	7	7	7	6	5	5	6
C	10	9	8	8	8	8	7	6	6	6

Recursively formulating the problem

- Input strings are $x[1, \cdots, m]$ and $y[1, \cdots, n]$.
- Let $x_i = x[1, \cdots, i]$ and $y_j = y[1, \cdots, j]$ be substrings of x and y.
- Edit distance $d(x, y)$ can be recursively defined as follows

$$d(x_i, y_j) = \begin{cases} i & j = 0 \\ j & i = 0 \\ \min \left\{ \begin{array}{l} d(x_{i-1}, y_j) + 1 \\ d(x_i, y_{j-1}) + 1 \\ d(x_{i-1}, y_{i-1}) + I(x[i] \neq y[j]) \end{array} \right\} & otherwise \end{cases}$$

- Similar to the Manhattan tourist problem, but with 3-way choice.
- Time complexity is $\Theta(mn)$.

Edit Distance Implementation

editDistance.cpp

```
#include <iostream>
#include <climits>
#include <string>
#include <vector>
#include "Matrix615.h"
int main(int argc, char** argv) {
  if ( argc != 3 ) {
    std::cerr << "Usage: editDistance [str1] [str2]" << std::endl;
    return -1;
  }
  std::string s1(argv[1]);
  std::string s2(argv[2]);
  Matrix615<int> cost(s1.size()+1, s2.size()+1, INT_MAX);
  Matrix615<int> move(s1.size()+1, s2.size()+1, -1);
  int optDist = editDistance(s1, s2, cost,move, cost.rowNums()-1, cost.colNums()-1);
  std::cout << "EditDistance is " << optDist << std::endl;
  printEdits(s1, s2, move);
  return 0;
}
```


editDistance() algorithm

editDistance.cpp

```
// note to declare the function before main()
int editDistance(std::string& s1, std::string& s2, Matrix615<int>& cost,
                 Matrix615<int>& move, int r, int c) {
  int iCost = 1, dCost = 1, mCost = 1; // insertion, deletion, mismatch cost

  if ( cost.data[r][c] == INT_MAX ) {
    if ( r == 0 && c == 0 ) { cost.data[r][c] = 0; }
    else if ( r == 0 ) {
      move.data[r][c] = 0;  // only insertion is possible
      cost.data[r][c] = editDistance(s1,s2,cost,move,r,c-1) + iCost;
    }
    else if ( c == 0 ) {
      move.data[r][c] = 1;  // only deletion is possible
      cost.data[r][c] = editDistance(s1,s2,cost,move,r-1,c) + dCost;
    }
```


editDistance() algorithm

editDistance.cpp

```
    else {  // compare 3 different possible moves and take the optimal one
      int iDist = editDistance(s1,s2,cost,move,r,c-1) + iCost;
      int dDist = editDistance(s1,s2,cost,move,r-1,c) + dCost;
      int mDist = editDistance(s1,s2,cost,move,r-1,c-1) +
                  (s1[r-1] == s2[c-1] ? 0 : mCost);
      if ( iDist < dDist ) {
        if ( iDist < mDist ) { // insertion is optima
          move.data[r][c] = 0;
          cost.data[r][c] = iDist;
        }
        else {
          move.data[r][c] = 2; // match is optimal
          cost.data[r][c] = mDist;
        }
      }
```

editDistance() algorithm

editDistance.cpp

```
      else {
        if ( dDist < mDist ) {
          move.data[r][c] = 1; // deletion is optimal
          cost.data[r][c] = dDist;
        }
        else {
          move.data[r][c] = 2; // match is optimal
          cost.data[r][c] = mDist;
        }
      }
    }
  }
  return cost.data[r][c];
}
```


editDistance.cpp: printEdits()

editDistance.cpp

```
int printEdits(std::string& s1, std::string& s2, Matrix615<int>& move) {
  std::string o1, o2, m;    // output string and alignments
  int r = move.rowNums()-1;
  int c = move.colNums()-1;
  while( r >= 0 && c >= 0 && move.data[r][c] >= 0) { // back from the last character
    if ( move.data[r][c] == 0 ) {  // insertion
      o1 = "-" + o1;  o2 = s2[c-1] + o2;  m = "I" + m; --c;
    }
    else if ( move.data[r][c] == 1 ) { // delettion
      o1 = s1[r-1] + o1;  o2 = "-" + o2;  m = "D" + m; --r;
    }
    else if ( move.data[r][c] == 2 ) { // match or mismatch
      o1 = s1[r-1] + o1; o2 = s2[c-1] + o2;
      m = (s1[r-1] == s2[c-1] ? "-" : "*") + m;
      --r; --c;
    }
    else std::cout << r << " " << c << " " << move.data[r][c] << std::endl;
  }
  std::cout << m << std::endl << o1 << std::endl << o2 << std::endl;
}
```

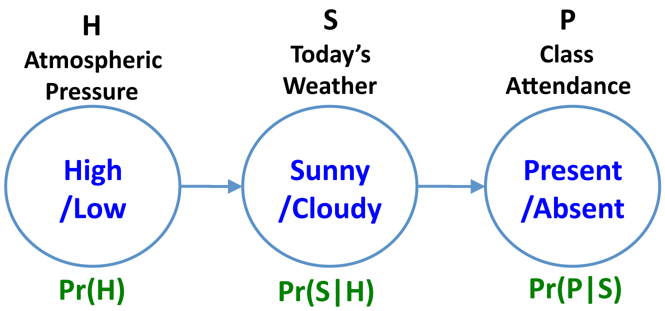

Running example

```
$ ./editDistance FOOD MONEY
EditDistance is 4
*-I**
FO-OD
MONEY
```


Graphical Model 101

- Graphical model is marriage between probability theory and graph theory (Michiael I. Jordan)
- Each random variable is represented as vertex
- Dependency between random variables is modeled as edge
  - Directed edge : conditional distribution
  - Undirected edge : joint distribution
- Unconnected pair of vertices (without path from one to another) is independent
- An effective tool to represent complex structure of dependence / independence between random variables.

An example graphical model



- Are H and P independent?
- Are H and P independent given S?

Example probability distribution

$\Pr(H)$

Value (H)	Description (H)	$\Pr(H)$
0	Low	0.3
1	High	0.7

$\Pr(S\|H)$

S	Description (S)	H	Description (H)	$\Pr(S\|H)$
0	Cloudy	0	Low	0.7
1	Sunny	0	Low	0.3
0	Cloudy	1	High	0.1
1	Sunny	1	High	0.9

Probability distribution (cont'd)

$\Pr(P\|S)$

P	Description (P)	S	Description (S)	$\Pr(P\|S)$
0	Absent	0	Cloudy	0.5
1	Present	0	Cloudy	0.5
0	Absent	1	Sunny	0.1
1	Present	1	Sunny	0.9

Full joint distribution

$\Pr(H, S, P)$

H	S	P	$\Pr(H, S, P)$
0	0	0	0.105
0	0	1	0.105
0	1	0	0.009
0	1	1	0.081
1	0	0	0.035
1	0	1	0.035
1	1	0	0.063
1	1	1	0.567

- With a full join distribution, any type of inference is possible
- As the number of variables grows, the size of full distribution table increases exponentially

$\Pr(H, P|S) = \Pr(H|S)\Pr(P|S)$

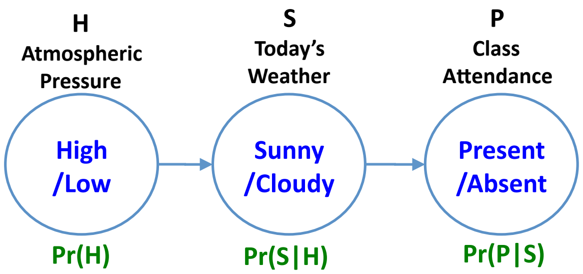
$\Pr(H, P|S)$

H	P	S	$\Pr(H, P\|S)$
0	0	0	0.3750
0	1	0	0.3750
1	0	0	0.1250
1	1	0	0.1250
0	0	1	0.0125
0	1	1	0.1125
1	0	1	0.0875
1	1	1	0.7875

$\Pr(H|S)$, $\Pr(P|S)$

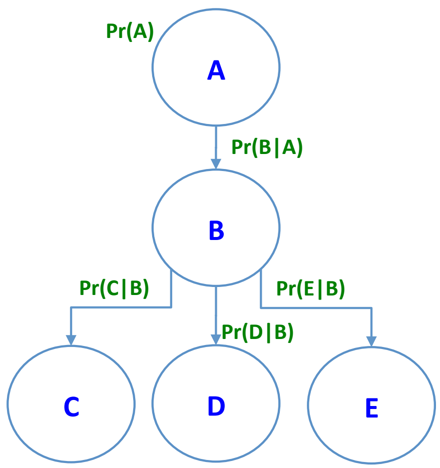
H	S	$\Pr(H\|S)$	P	S	$\Pr(P\|S)$
0	0	0.750	0	0	0.500
1	0	0.250	1	0	0.500
0	1	0.125	0	1	0.100
1	1	0.875	1	1	0.900

H and P are conditionally independent given S



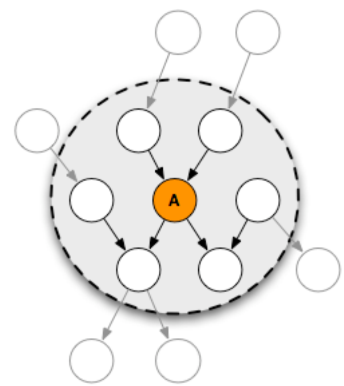
- H and P do not have direct path one from another
- All path from H to P is connected thru S.
- Conditioning on S separates H and P

Conditional independence in graphical models



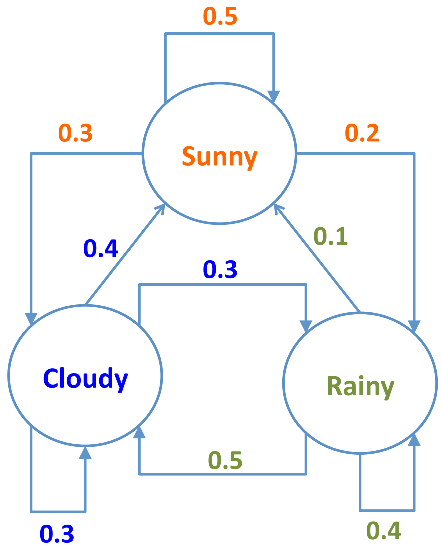
- $\Pr(A, C, D, E|B) = \Pr(A|B)\Pr(C|B)\Pr(D|B)\Pr(E|B)$

Markov Blanket



- If conditioned on the variables in the gray area (variables with direct dependency), A is independent of all the other nodes.
- $A \perp (U - A - \pi_A) | \pi_A$

Markov Process : An example



Mathematical representation of a Markov Process

$$
\begin{aligned}
\pi &= \begin{pmatrix} \Pr(q_1 = S_1 = \text{Sunny}) \\ \Pr(q_1 = S_2 = \text{Cloudy}) \\ \Pr(q_1 = S_3 = \text{Rainy}) \end{pmatrix} = \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix} \\
A_{ij} &= \Pr(q_{t+1} = S_j | q_t = S_i) \\
A &= \begin{pmatrix} 0.5 & 0.3 & 0.2 \\ 0.4 & 0.3 & 0.3 \\ 0.1 & 0.5 & 0.4 \end{pmatrix}
\end{aligned}
$$

Example questions in Markov Process

What is the chance of rain in the day 2?

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

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

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

Stationary distribution

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

Markov process is only dependent on the previous state

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

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

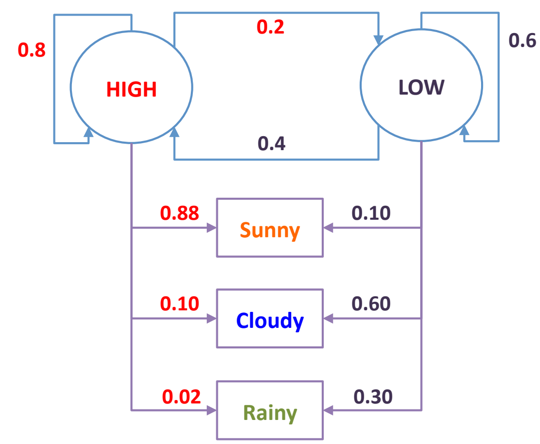
If it has rained for the past three days, what is the chance of rain on the day after tomorrow?

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

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



- Direct Observation : (SUNNY, CLOUDY, RAINY)
- Hidden States : (HIGH, LOW)

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

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

Emission $B_{ij} = b_{q_t}(o_t) = b_{S_i}(O_j) = \Pr(o_t = O_j | q_t = S_i)$

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

Unconditional marginal probabilities

What is the chance of rain in the day 4?

$$\mathbf{f}(\mathbf{q}_4) = \begin{pmatrix} \Pr(q_4 = S_1) \\ \Pr(q_4 = S_2) \end{pmatrix} = (A^T)^3 \pi = \begin{pmatrix} 0.669 \\ 0.331 \end{pmatrix}$$

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

The chance of rain in day 4 is 23.3%

Summary

Edit Distance

- Alignment between two strings
- Can be converted to a problem similar to MTP

Hidden Markov Models

- Graphical models
- Conditional independence and Markov blankets
- Markov process
- Introduction to hidden Markov models

Next lectures

- More hidden Markov Models