Biostatistics 615/815 Lecture 11: Hidden Markov Models, Standard Template Library, and Boost Library

Hyun Min Kang

October 9th, 2012

DP algorithm for calculating forward probability

- Key idea is to use $(q_t, o_t) \perp \mathbf{o}_t^- | \mathbf{q}_{t-1}$.
- Each of q_{t-1}, q_t, and q_{t+1} is a Markov blanket.

$$
\begin{aligned}
\alpha_t(i) &= \Pr(o_1, \cdots, o_t, q_t = i|\lambda) \\
&= \sum_{j=1}^{n} \Pr(\mathbf{o}_t^-, o_t, q_{t-1} = j, q_t = i|\lambda) \\
&= \sum_{j=1}^{n} \Pr(\mathbf{o}_t^-, q_{t-1} = j|\lambda) \Pr(q_t = i|q_{t-1} = j, \lambda) \Pr(o_t|q_t = i, \lambda) \\
&= \sum_{j=1}^{n} \alpha_{t-1}(j) a_{ji} b_i(o_t) \\
\alpha_1(i) &= \pi_i b_i(o_1)
\end{aligned}
$$

DP algorithm for calculating backward probability

- Key idea is to use $o_{t+1} \perp \mathbf{o}_{t+1}^{+} | \mathbf{q}_{t+1}$.

$$\begin{aligned}
\beta_t(i) &= \Pr(o_{t+1}, \cdots, o_T | q_t = i, \lambda) \\
&= \sum_{j=1}^{n} \Pr(o_{t+1}, \mathbf{o}_{t+1}^{+}, q_{t+1} = j | q_t = i, \lambda) \\
&= \sum_{j=1}^{n} \Pr(o_{t+1} | q_{t+1}, \lambda) \Pr(\mathbf{o}_{t+1}^{+} | q_{t+1} = j, \lambda) \Pr(q_{t+1} = j | q_t = i, \lambda) \\
&= \sum_{j=1}^{n} \beta_{t+1}(j) a_{ij} b_j(o_{t+1}) \\
\beta_T(i) &= 1
\end{aligned}$$

Putting forward and backward probabilities together

- Conditional probability of states given data

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

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

A working example : Occasionally biased coin

A generative HMM

- Observations : $O = \{1(Head), 2(Tail)\}$
- Hidden states : $S = \{1(Fair), 2(Biased)\}$
- Initial states : $\pi = \{0.5, 0.5\}$
- Transition probability : $A(i,j) = a_{ij} = \begin{pmatrix} 0.95 & 0.05 \\ 0.2 & 0.8 \end{pmatrix}$
- Emission probability : $B(i,j) = b_i(j) = \begin{pmatrix} 0.5 & 0.5 \\ 0.9 & 0.1 \end{pmatrix}$

Questions

- Given coin toss observations, estimate the probability of each state
- Given coin toss observations, what is the most likely series of states?

Implementing HMM - `Matrix615.h`

```
#ifndef __MATRIX_615_H  // to avoid multiple inclusion of same headers
#define __MATRIX_615_H
#include <vector>

template <class T>
class Matrix615 {
public:
  std::vector< std::vector<T> > data;
  Matrix615(int nrow, int ncol, T val = 0) {
    data.resize(nrow); // make n rows
    for(int i=0; i < nrow; ++i) {
      data[i].resize(ncol,val); // make n cols with default value val
    }
  }
  int rowNums() { return (int)data.size(); }
  int colNums() { return ( data.size() == 0 ) ? 0 : (int)data[0].size(); }
};

#endif // __MATRIX_615_H
```

HMM Implementations - HMM615.h

```
#ifndef __HMM_615_H
#define __HMM_615_H
#include "Matrix615.h"
class HMM615 {
 public:
  // parameters
  int nStates;  // n : number of possible states
  int nObs;     // m : number of possible output values
  int nTimes;   // t : number of time slots with observations
  std::vector<double> pis; // initial states
  std::vector<int> outs;   // observed outcomes
  Matrix615<double> trans;   // trans[i][j] corresponds to A_{ij}
  Matrix615<double> emis;

  // storages for dynamic programming
  Matrix615<double> alphas, betas, gammas, deltas;
  Matrix615<int> phis;
  std::vector<int> path;
```

HMM Implementations - HMM615.h

```
  HMM615(int states, int obs, int times) : nStates(states), nObs(obs),
    nTimes(times), trans(states, states, 0), emis(states, obs, 0),
    alphas(times, states, 0), betas(times, states, 0),
    gammas(times, states, 0), deltas(times, states, 0),
    phis(times, states, 0)
  {
    pis.resize(nStates);
    path.resize(nTimes);
  }

  void forward();   // given below
  void backward();  //
  void forwardBackward(); // given below
  void viterbi();   //
};
#endif // __HMM_615_H
```

HMM Implementations - HMM615::forward()

```
void HMM615::forward() {
  for(int i=0; i < nStates; ++i) {
    alphas.data[0][i] = pis[i] * emis.data[i][outs[0]];
  }
  for(int t=1; t < nTimes; ++t) {
    for(int i=0; i < nStates; ++i) {
      alphas.data[t][i] = 0;
      for(int j=0; j < nStates; ++j) {
        alphas.data[t][i] += (alphas.data[t-1][j] * trans.data[j][i]
          * emis.data[i][outs[t]]);
      }
    }
  }
}
```

HMM Implementations - HMM615::backward()

```
void HMM615::backward() {
  for(int i=0; i < nStates; ++i) {
    betas.data[nTimes-1][i] = 1;
  }
  for(int t=nTimes-2; t >=0; --t) {
    for(int i=0; i < nStates; ++i) {
      betas.data[t][i] = 0;
      for(int j=0; j < nStates; ++j) {
        betas.data[t][i] += (betas.data[t+1][j] * trans.data[i][j]
                             * emis.data[j][outs[t+1]]);
      }
    }
  }
}
```

HMM Implementations - HMM615::forwardBackward()

```
void HMM615::forwardBackward() {
  forward();
  backward();

  for(int t=0; t < nTimes; ++t) {
    double sum = 0;
    for(int i=0; i < nStates; ++i) {
      sum += (alphas.data[t][i] * betas.data[t][i]);
    }
    for(int i=0; i < nStates; ++i) {
      gammas.data[t][i] = (alphas.data[t][i] * betas.data[t][i])/sum;
    }
  }
}
```

HMM Implementations - HMM615::viterbi()

```
void HMM615::viterbi() {
  for(int i=0; i < nStates; ++i) {
    deltas.data[0][i] = pis[i] * emis.data[i][ outs[0] ];
   }
  for(int t=1; t < nTimes; ++t) {
    for(int i=0; i < nStates; ++i) {
      int maxIdx = 0;
      double maxVal = deltas.data[t-1][0] * trans.data[0][i]
                       * emis.data[i][ outs[t] ];
      for(int j=1; j < nStates; ++j) {
        double val = deltas.data[t-1][j] * trans.data[j][i]
                       * emis.data[i][ outs[t] ];
        if ( val > maxVal ) { maxIdx = j; maxVal = val; }
      }
      deltas.data[t][i] = maxVal;
      phis.data[t][i] = maxIdx;
    }
  }
```

HMM Implementations - HMM615::viterbi() (cont'd)

```
  // backtrack viterbi path
  double maxDelta = deltas.data[nTimes-1][0];
  path[nTimes-1] = 0;
  for(int i=1; i < nStates; ++i) {
    if ( maxDelta < deltas.data[nTimes-1][i] ) {
      maxDelta = deltas.data[nTimes-1][i];
      path[nTimes-1] = i;
    }
  }
  for(int t=nTimes-2; t >= 0; --t) {
    path[t] = phis.data[t+1][ path[t+1] ];
  }
}
```

HMM Implementations - biasedCoin.cpp

```
#include <iostream>
#include <iomanip>
int main(int argc, char** argv) {
  std::vector<int> toss;
  std::string tok;
  while( std::cin >> tok ) {
    if ( tok == "H" ) toss.push_back(0);
    else if ( tok == "T" ) toss.push_back(1);
    else {
      std::cerr << "Cannot recognize input " << tok << std::endl;
      return -1;
    }
  }

  int T = toss.size();
  HMM615 hmm(2, 2, T);

  hmm.trans.data[0][0] = 0.95; hmm.trans.data[0][1] = 0.05;
  hmm.trans.data[1][0] = 0.2;  hmm.trans.data[1][1] = 0.8;
```

HMM Implementations - biasedCoin.cpp

```
  hmm.emis.data[0][0] = 0.5; hmm.emis.data[0][1] = 0.5;
  hmm.emis.data[1][0] = 0.9; hmm.emis.data[1][1] = 0.1;

  hmm.pis[0] = 0.5; hmm.pis[1] = 0.5;

  hmm.outs = toss;

  hmm.forwardBackward();
  hmm.viterbi();

  std::cout << "TIME\tTOSS\tP(FAIR)\tP(BIAS)\tMLSTATE" << std::endl;
  std::cout << std::setiosflags(std::ios::fixed) << std::setprecision(4);
  for(int t=0; t < T; ++t) {
    std::cout << t+1 << "\t" << (toss[t] == 0 ? "H" : "T") << "\t"
        << hmm.gammas.data[t][0] << "\t" << hmm.gammas.data[t][1] << "\t"
        << (hmm.path[t] == 0 ? "FAIR" : "BIASED" ) << std::endl;
  }
  return 0;
}
```

Example runs

```
$ cat ~hmkang/Public/615/data/toss.20.txt | ~hmkang/Public/615/bin/biasedCoin
TIME    TOSS    P(FAIR) P(BIAS) MLSTATE
1       H       0.5950  0.4050  FAIR
2       T       0.8118  0.1882  FAIR
3       H       0.8071  0.1929  FAIR
4       T       0.8584  0.1416  FAIR
5       H       0.7613  0.2387  FAIR
6       H       0.7276  0.2724  FAIR
7       T       0.7495  0.2505  FAIR
8       H       0.5413  0.4587  BIASED
9       H       0.4187  0.5813  BIASED
10      H       0.3533  0.6467  BIASED
11      H       0.3301  0.6699  BIASED
12      H       0.3436  0.6564  BIASED
13      H       0.3971  0.6029  BIASED
14      T       0.5028  0.4972  BIASED
15      H       0.3725  0.6275  BIASED
16      H       0.2985  0.7015  BIASED
17      H       0.2635  0.7365  BIASED
18      H       0.2596  0.7404  BIASED
19      H       0.2858  0.7142  BIASED
20      H       0.3482  0.6518  BIASED
```

STL containers

What are STL containers?

- Data structure for convenient storage and access of multiple elements
- Behaviors are robust for both call-by-value and call-by-reference.
- http://www.cplusplus.com/reference/stl/ serves a great reference to look up.

Three popular STL containers

- `std::vector` - Array. $O(1)$ insert, $O(n)$ search.
- `std::set` - Container of unique elements. $O(\log n)$ insert/search.
- `std:map` - Container for key-value pairs

Using std::vector : Initialization

```
int a1[4];                // OK: initialize an array of size 4
std::vector<int> v1(4);   // OK: make a vector of size 4

int n = atoi(argv[1]);    // OK: run time argument
int a2[n];                // OK: ERROR: n is unknown in compile time
int* a3 = new int[n];     // OK: Must be allocated with new[] operator
delete [] a3;             // OK: And must be deleted after using
std::vector<int> v2(n);   // OK: For vector, n can be determined in run time
v2.resize(2*n);           // OK: And you can resize a vector, but not an array

int a4[4] = {2,0,1,2};    // OK : Multiple element initialization is simple
std::vector<int> v3(4) = {2,0,1,2}; // ERROR : Not allowed for vector
std::vector<int> v4(4);   // OK : Each element has to be assigned separately
v4[0] = 2; v4[1] = 0; v4[2] = 1; v4[3] = 2; // access elements using []

int a5[4] = {-1,-1,-1,-1}; // OK : Need to write redundantly
std::vector<int> v5(4,-1); // OK : Allocate size 4 vector with value -1
```

Using std::vector : Insert, search and remove

Inserting elements

```
int a6[4];           // in an array, the size needs to be specified a priori
for(int i=0; i < 4; ++i) // and assign each value
   a6[i] = (i*i);
std::vector<int> v6; // with vector, initially define an empty vector
for(int i=0; i < 4; ++i)
   v6.push_back(i*i); // using push_back() function, size dynamically changes
```

Search for values

```
for(int i=0; i < 4; ++i)
  if ( a6[i] == 4 ) std::cout << "Found 4" << std::endl;
for(unsigned int i=0; i < v6.size(); ++i) // v6.size() is unsigned
  if ( v6[i] == 4 ) std::cout << "Found 4" << std::endl;
std::vector<int>::iterator it;  // use iterator
for(it = v6.begin(); it != v6.end(); ++it)
  if ( *it == 4 ) std::cout << "Found 4" << std::endl;
v6.clear();         // remove all contents
```

Matrix615.h uses a vector of vector - 2D array

```
template <class T>
class Matrix615 {
public:
  std::vector< std::vector<T> > data; // 2D array
  // initialize 2D array with (nrow) x (ncol) size
  Matrix615(int nrow, int ncol, T val = 0) {
    data.resize(nrow); // make n rows
    for(int i=0; i < nrow; ++i) {
      data[i].resize(ncol,val); // make n cols with default value val
    }
  }
  int rowNums() { return (int)data.size(); } // outer array size is row
  // assuming every column size is the same,
  // retrieve the first column's size if exists
  int colNums() { return ( data.size() == 0 ) ? 0 : (int)data[0].size(); }
};
```

Using std::set for repetitive and fast search

lookup.cpp

```cpp
#include <iostream>
#include <fstream>
#include <set>
#include <string>
int main(int argc, char** argv) {
  std::ifstream ifs(argv[1], std::ifstream::in );
  std::set<std::string> words;
  std::string word;
  while( ifs >> word ) words.insert(word);  // load file to set
  std::cout << "Type any word to lookup: ";
  while( std::cin >> word ) {
    if ( words.find(word) != words.end() )
      std::cout << "Found " << word << std::endl;
    else
      std::cout << "Could not find " << word << std::endl;
    std::cout << std::endl << "Type any word to lookup: ";
  }
  return 0;
}
```

Running lookup.cpp

```
hmkang@galaga:~/Public/615$ ~hmkang/Public/615/bin/lookup ~hmkang/Public/615/data/words
Type any word to lookup: hello
Found hello

Type any word to lookup: world
Found world

Type any word to lookup: biostat615
Could not find biostat615
```

Using std::map as a dictionary

countSubstr.cpp

```
#include <iostream>
#include <fstream>
#include <map>
#include <vector>
#include <string>

using namespace std; // to avoid typing std:: repetitively

int main(int argc, char** argv) {
  if ( argc != 3 ) {
    cerr << "Usage: " << argv[0] << " [input_file.txt] [length]" << endl;
    return -1;
  }

  ifstream ifs(argv[1], ifstream::in );
  int length = atoi(argv[2]);
  map<string,int> mCnt;               // (substr)->(count) map
  map<string, vector<string> > mWord; // (substr)->(list to all words) map
  string word;                        // variable to store a word
  string ss;                          // variable to stotr a substring
```

Using std::map as a dictionary

countSubstr.cpp (cont'd)

```
  while( ifs >> word ) {
    ss = word.substr(0,length);   // make substring
    ++mCnt[ss];                   // update count map (use map like array)
    mWord[ss].push_back(word);    // update word list map
  }
  cout << "Successfully loaded input file " << argv[1] << endl;
  cout << endl << "Type a substring of length " << length << ": ";
  while( cin >> ss ) {
    int cnt = mCnt[ss];
    cout << "There are " << cnt << " words starting with " << ss << endl;
    if ( cnt > 0 ) {                     // print each word in the list
      vector<string>& words = mWord[ss]; // reference type to avoid copy
      for(int i=0; i < cnt; ++i)
         cout << words[i] << endl;
    }
    cout << endl << "Type a substring of length " << length << ": ";
  }
  return 0;
}
```

Running countSubstr.cpp

```
hmkang@galaga:~/Public/615$ bin/countSubstr ~hmkang/Public/615/data/words 4
Successfully loaded input file /afs/umich.edu/user/h/m/hmkang/Public/615/data/words

Type a substring of length 4: bios
There are 4 words starting with bios
bioscience
biosphere
biostatistic
biosynthesize

Type a substring of length 4: kang
There are 1 words starting with kang
kangaroo

Type a substring of length 4: hyun
There are 0 words starting with hyun

Type a substring of length 4:
```

Input/output handling

iomanip for output formatting

```
#include <iomanip>
// write floating point values in fixed point notation
std::cout << std::setiosflags(std::ios::fixed); // see also std::ios::scientific
std::cout << std::setprecision(4); // print up to 4 significant digits
std::cout << 3.14159               // 3.142 will be printed
// managing alignment between output
std::cout << std::setw(10);        // set the minimum width of output to 10
std::cout << std::setiosflags(ios::right) // right align the output
std::cout << 100                   // 100 printed after 7 blanks
```

ifstream for reading files

```
#include <fstream>
std::ifstream ifs("myfile.txt");
std::string s;
while( ifs >> s ) std::cout << "Read " << s << std::endl;
```

More STL examples

std::sort for sorting an array

```
#include <algorithm>
// ...
int myints[] = {32,71,12,45,26,80,53,33};
vector<int> myvector (myints, myints+8);
std::sort(myvector.begin(), myvector.begin()+4); // 12 32 45 71 26 80 53 33
std::sort(myvector.begin(), myvector.end();      // 12 26 32 33 45 53 71 80
```

Define your own comparison function for customized sorting

std::next_permutation for enumerating permutation

```
#include <iostream>
#include <algorithm>
int main(int argc, char** argv) {
  int myints[] = {1,2,3};
  do {
    std::cout << myints[0] << " " << myints[1] << " " << myints[2] << std::endl;
  } while ( next_permutation (myints,myints+3) );
  return 0;
}
```

Using boost C++ libraries

Boost C++ library

- An extensive set of libraries for C++
- Supports many additional classes and functions beyond STL
- Useful for increasing productivity

Using boost C++ libraries

Boost C++ library

- An extensive set of libraries for C++
- Supports many additional classes and functions beyond STL
- Useful for increasing productivity

Examples of useful libraries

- Math/Statistical Distributions
- Tokenizer
- Random Numbers
- Graph Algorithms
- Regular expressions

Test Program for Boost Library : boostTest.cpp

```
#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);  // tokenize string delimited by whitespace
  for(tokenizer<>::iterator i=tok1.begin(); i != tok1.end() ; ++i) {
    cout << *i << endl;  // print each element
  }
  return 0;
}
```

Using boost libraries

At scs.itd.umich.edu

```
$ g++ -o tokenizerTest -I ~hmkang/Public/include/ tokernizerTest.cpp
```

Adding the inclusion path will allow to include boost headers

Installing boost library in your system

- from *http://www.boost.org/*
- `tar xzvf boost_1_51_0.tar.gz`
- `mkdir ~/include` (under my home directory)
- `cp -R boost_1_51_0/boost ~/include/boost`
- `g++ -I ~/include -o tokenizerTest tokenizerTest.cpp` or modify inclusion path in your development environment
- For a complete installation, follow instructions at *http://www.boost.org/users/download/*

boost example 1 : Chi-squared test

```
#include <iostream>
#include <boost/math/distributions/chi_squared.hpp>
using namespace boost::math;
int main(int argc, char** argv) {
  if ( argc != 5 ) {
    std::cerr << "Usage: chisqTest [a] [b] [c] [d]" << std::endl;
    return -1;
  }
  int a = atoi(argv[1]);  // read 2x2 table from command line arguments
  int b = atoi(argv[2]);
  int c = atoi(argv[3]);
  int d = atoi(argv[4]);
  // calculate chi-squared statistic and p-value
  double chisq = (double)(a*d-b*c)*(a*d-b*c)*(a+b+c+d)/(a+b)/(c+d)/(a+c)/(b+d);
  chi_squared chisqDist(1);
  cout << "Chi-square statistic = " << chisq << endl;
  cout << "p-value = " << cdf(complement(chisqDist, chisq)); << endl;
  return 0;
}
```

Running examples of chisqTest

```
user@host~:/$ ./chisqTest 2 7 8 2
Chi-square test statistic = 6.34272
p-value = 0.0117864

user@host~:/$ ./chisqTest 20 70 80 20
Chi-square test statistic = 63.4272
p-value = 1.66408e-15

user@host~:/$ ./chisqTest 200 700 800 200
Chi-square test statistic = 634.272
p-value = 5.88561e-140

user@host~:/$ ./chisqTest 2000 7000 8000 2000
Chi-square statistic = 6342.72
p-value = 0
```

boost Example 2 : Tokenizer

```
#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;
  }

  // advanced use : 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;
}
```

A running example of tokenizerTest

```
user@host~:/$ ./tokenizerTest
Hello
boost
library
Field 1
putting quotes around fields, allows commas
Field 3
```

boost Example : Reading matrix from file

```
#ifndef __MATRIX_615_H
#define __MATRIX_615_H
#include <vector>
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
template <class T>
class Matrix615 {
public:
  std::vector< std::vector<T> > data;

  Matrix615(int nrow, int ncol, T val = 0) {
    data.resize(nrow); // make n rows
    for(int i=0; i < nrow; ++i)
      data[i].resize(ncol,val); // make n cols with default value val
  }
  int rowNums() { return (int)data.size(); }
  int colNums() { return ( data.size() == 0 ) ? 0 : (int)data[0].size(); }
  void readFromFile(const char* fileName);
};
```

Matrix615.h

```cpp
void Matrix615::readFromFile(const char* fileName) {
  // open input file
  std::ifstream ifs(fileName);
  if ( ! ifs.is_open() ) {
    std::cerr << "Cannot open file " << fileName << std::endl;
    abort();
  }

  // set up the tokenizer
  std::string line;
  boost::char_separator<char> sep(" \t");
  // typedef is used to replace long type to a short alias
  typedef boost::tokenizer< boost::char_separator<char> > wsTokenizer;

  // clear the data first
  data.clear();
  int nr = 0, nc = 0;
```

Matrix615.h

```
  // read from file to fill the contents
  while( std::getline(ifs, line) ) {
    if ( line[0] == '#' ) continue; // skip meta-lines starting with #
    wsTokenizer t(line,sep);
    data.resize(nr+1);
    for(wsTokenizer::iterator i=t.begin(); i != t.end(); ++i) {
      data[nr].push_back(boost::lexical_cast<T>(i->c_str()));
      if ( nr == 0 ) ++nc;  // count # of columns at the first row
    }
    if ( nc != (int)data[nr].size() ) {
      std::cerr << "The input file is not rectangle at line " << nr << std::endl;
      abort();
    }
    ++nr;
  }
}
```

Matrix615.h

```
template <class T>
void readFromFile(std::vector<T>& v, const char* fileName) {
  // open input file
  std::ifstream ifs(fileName);
  if ( ! ifs.is_open() ) {
    std::cerr << "Cannot open file " << fileName << std::endl;
    abort();
  }

  v.clear();
  std::string tok;
  while( ifs >> tok ) {
    v.push_back(boost::lexical_cast<T>(tok));
  }
}
```