GMTK Tutorial

2012/09/18 Ting-Wei, Chung-Yi

Outline

- GMTK Tutorial
  - What is GMTK.
  - Installation of GMTK
  - gmtkTriangulate and gmtkEMtrainNew
  - The GMTK Structure Language and master the file.
  - Example program
- Reference

What is GMTK

- Graphical Models Toolkit
  - developed and maintained by University of Washington
- For developing dynamic Bayesian Network (DBN) based speech recognition and general time series systems.
- Open-source
  - But it hasn't released its source code yet.

Why do we need to learn this tool

- Learning to use a tool is sometimes time-consuming.
  - Especially for those tools that are not well-documented.
  - But sometimes it's worth it.

- Each time we design a graphical model, we always want to know whether it can explain the data well.
  - But we have to implement it first.

- If we have tools on hand, we can spend much less time on building the model and know the performance of our model without lots of effort.

Installation

- Download from
  - http://ssli.ee.washington.edu/~bilmes/gmtk/linux/2009/
- We use the Linux binaries. (Jan 2009)
  - If you want to work on the 217 workstations, you should download the binaries in m64 directory.

- For hw1, you need two binary files:
  - gmtkTriangulate and gmtkEMtrainNew

gmtkTriangulate

- Create a triangulate file, which will be used for the main training program (gmtkEMtrainNew).

- For hw1, you only need to download it and set the right path in the provided Makefile.
  - or put it in “pgm_hw1/bin/”

gmtkEMtrainNew

- gmtkEMtrainNew is the main EM training program for GMTK.
  - EM is an algorithm which use iterative method to update and estimate parameters.

- gmtkEMtrainNew requires three input files
  - structure file (*.str)
  - master file (*.master)
  - data file (*.pfile)

GMTKL

- The GMTK structure language.
  - Specify the structure of a graphical model
  - Used in structure file. (*.str)

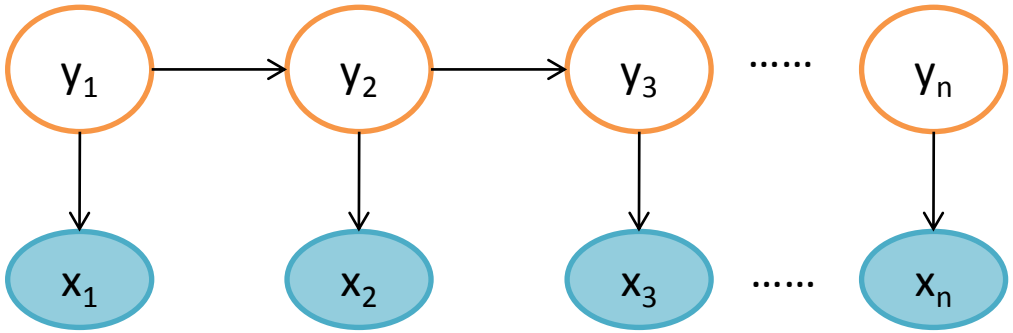
- Example HMM structure file
  - You can find it in “pgm_hw1/example/hmm.str”

HMM: structure file (hmm.str)

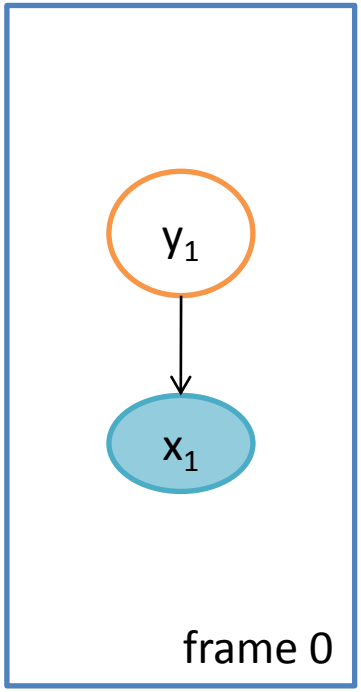
```
GRAPHICAL_MODEL hmm

frame : 0 {
  variable : state {
    type: discrete hidden cardinality 2;
    switchingparents: nil;
    conditionalparents: nil
          using DenseCPT("InitialState");
  }
  variable : outcome{
    type: discrete observed 0:0 cardinality 2;
    switchingparents: nil;
    conditionalparents: state(0)
          using DenseCPT("Emission");
  }
}

frame : 1 {
  variable : state{
    type: discrete hidden cardinality 2;
    switchingparents: nil;
    conditionalparents: state(-1)
          using DenseCPT("Transition");
  }
  variable : outcome{
    type: discrete observed 0:0 cardinality 2;
    switchingparents: nil;
    conditionalparents: state(0)
          using DenseCPT("Emission");
  }
}
chunk 1:1
```

HMM: graphical structure

GRAPHICAL_MODEL hmm



HMM: graphical structure

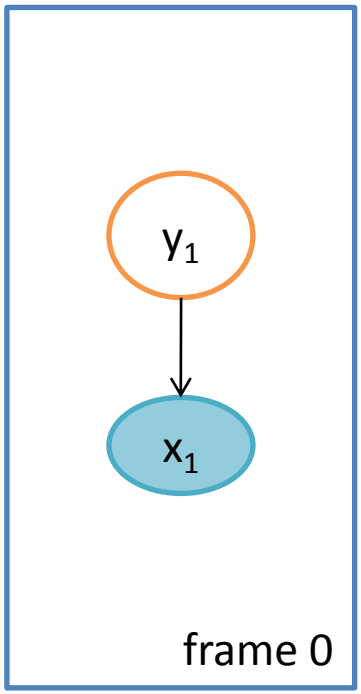



```
frame : 0 {
  variable : state {
    type: discrete hidden cardinality 2;
    switchingparents: nil;
    conditionalparents: nil using DenseCPT("InitialState");
  }
  variable : outcome{
    type: discrete observed 0:0 cardinality 2;
    switchingparents: nil;
    conditionalparents: state(0) using DenseCPT("Emission");
  }
}
```

y_1

x_1

HMM: graphical structure



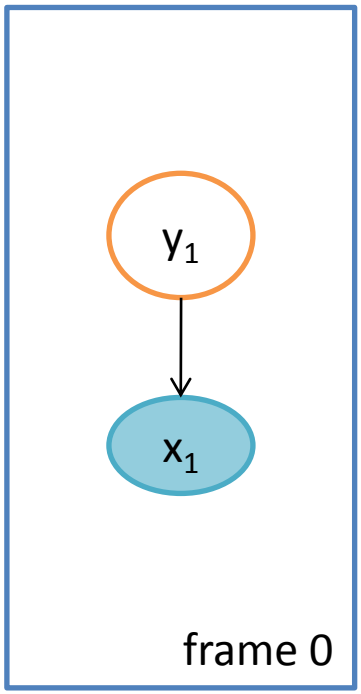
y_1

x_1

```
frame : 0 {
  variable : state {
    type: discrete hidden cardinality 2;
    switchingparents: nil;
    conditionalparents: nil using DenseCPT("InitialState");
  }
  variable : outcome{
    type: discrete observed 0:0 cardinality 2;
    switchingparents: nil;
    conditionalparents: state(0) using DenseCPT("Emission");
  }
}
```

$y_1=\{0,1\}$

HMM: graphical structure



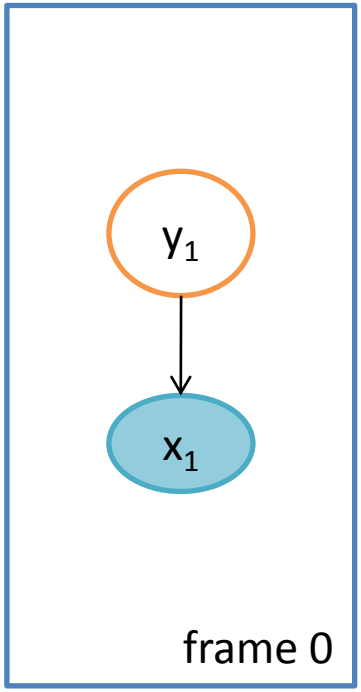
y_1

x_1

```
frame : 0 {
  variable : state {
    type: discrete hidden cardinality 2;
    switchingparents: nil;
    conditionalparents: nil using DenseCPT("InitialState");
  }
  variable : outcome{
    type: discrete observed 0:0 cardinality 2;
    switchingparents: nil;
    conditionalparents: state(0) using DenseCPT("Emission");
  }
}
```

y_1 has no parent

HMM: graphical structure



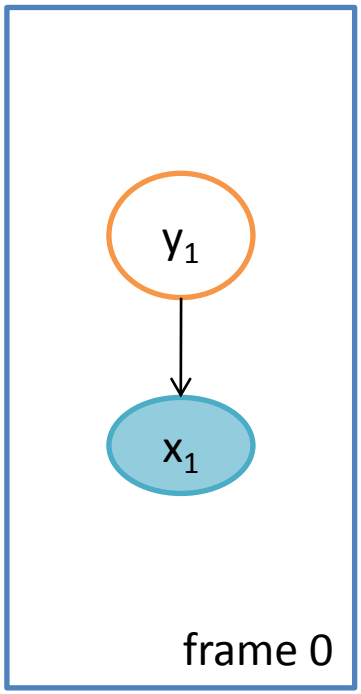

```
frame : 0 {
 variable : state {
  type: discrete hidden cardinality 2;
  switchingparents: nil;
  conditionalparents: nil using DenseCPT("InitialState");
 }
 variable : outcome{
  type: discrete observed 0:0 cardinality 2;
  switchingparents: nil;
  conditionalparents: state(0) using DenseCPT("Emission");
 }
}
```

y_1

x_1

CPT of y_1

HMM: graphical structure



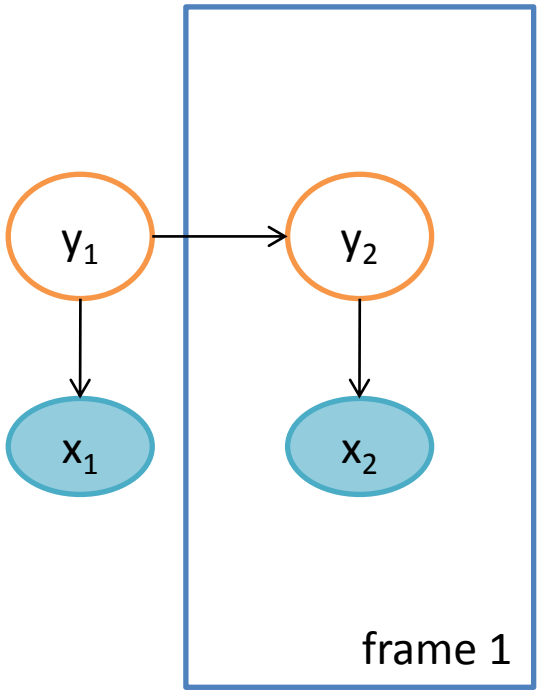

```
frame : 0 {
 variable : state {
   type: discrete hidden cardinality 2;
   switchingparents: nil;
   conditionalparents: nil using DenseCPT("InitialState");
 }
 variable : outcome{
   type: discrete observed 0:0 cardinality 2;
   switchingparents: nil;
   conditionalparents: state(0) using DenseCPT("Emission");
 }
}
```

y_1

x_1

The dimension of this variable

HMM: graphical structure

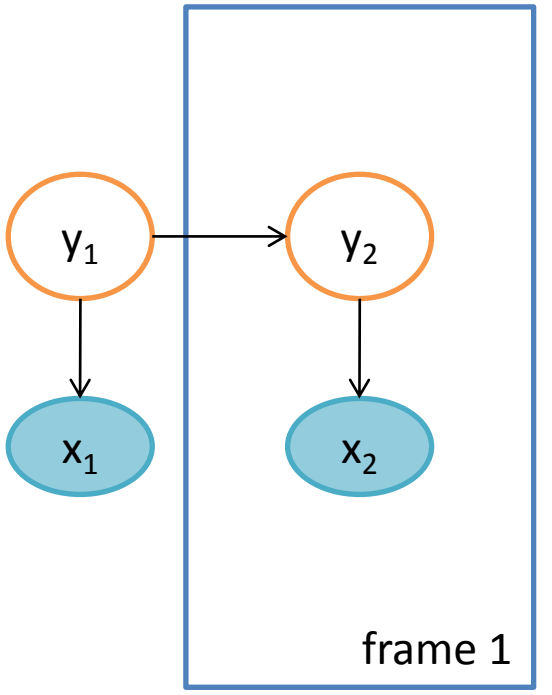


y_2

x_2

```
frame : 1 {
  variable : state{
    type: discrete hidden cardinality 2;
    switchingparents: nil;
    conditionalparents: state(-1) using DenseCPT("Transition");
  }
  variable : outcome{
    type: discrete observed 0:0 cardinality 2;
    switchingparents: nil;
    conditionalparents: state(0) using DenseCPT("Emission");
  }
}
```

HMM: graphical structure



y_2

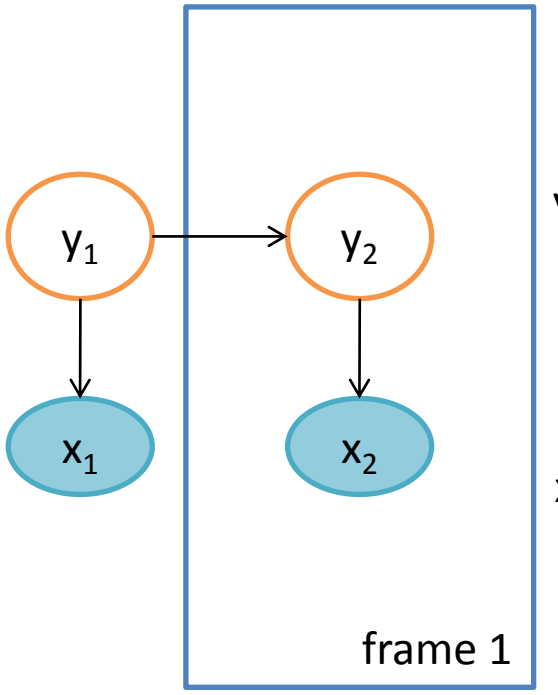
```
frame : 1 {
  variable : state{
    type: discrete hidden cardinality 2;
    switchingparents: nil;
    conditionalparents: state(-1) using DenseCPT("Transition");
  }
```

y_1

x_2

```
  variable : outcome{
    type: discrete observed 0:0 cardinality 2;
    switchingparents: nil;
    conditionalparents: state(0) using DenseCPT("Emission");
  }
}
```

HMM: graphical structure



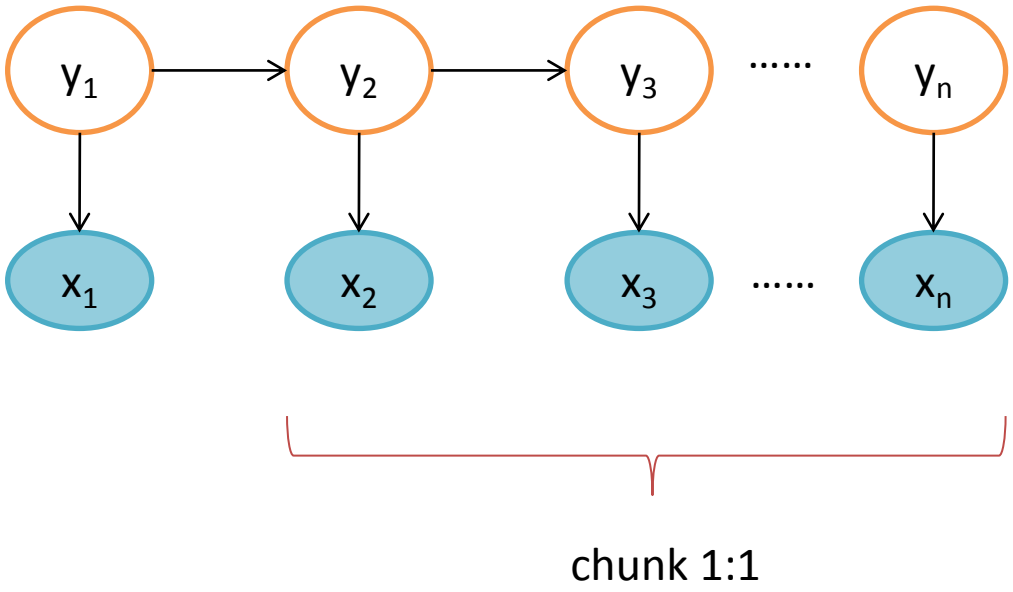
y_2

x_2

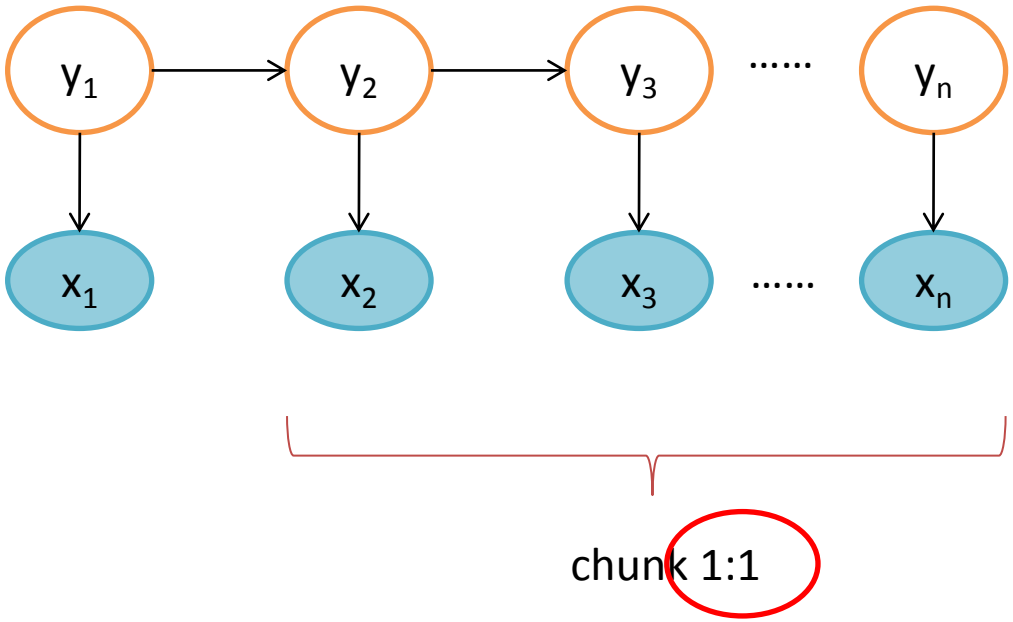
```
frame : 1 {
  variable : state{
    type: discrete hidden cardinality 2;
    switchingparents: nil;
    conditionalparents: state(-1) using DenseCPT("Transition");
  }
  variable : outcome{
    type: discrete observed 0:0 cardinality 2;
    switchingparents: nil;
    conditionalparents: state(0) using DenseCPT("Emission");
  }
}
```

CPT of y_2

HMM: graphical structure



HMM: graphical structure

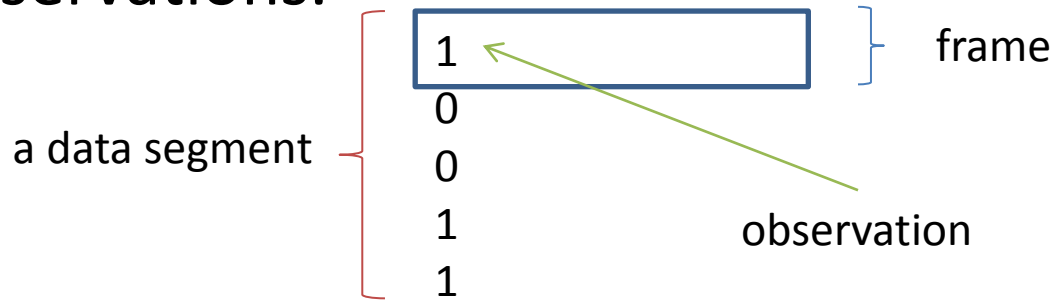


means frame 1 to frame 1 is the repeating unit.

HMM: data (hmm.pfile)

- This data file contains the observation of the graphical model.
  - There're two sections in pfile format
    - header and data

- There are multiple frames in a segment (data may have many segments), and each frame is a vector of observations.



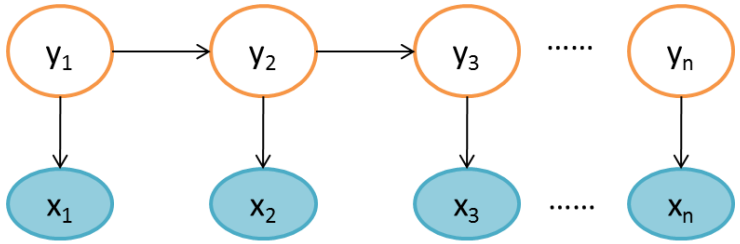
HMM: master file

- The example master file (hmm.master) is empty.
  - because the CPTs are unknown, we want to learn them with EM.

- But what if we want to specify the CPTs?
  - Dense CPT

Dense CPT: HMM master file

DENSE_CPT_IN_FILE inline
3 % number of dense CPTs
0 % the first one
InitialState % name of CPT
0 % number parents
2 % self cardinality = 2
2.25058877030877800e-59 1.00000000000000000e+00
1 % the second one
Emission % name of CPT
1 % number parents
2 2 % parent cardinality = 2, self cardinality = 2;
9.99999940233369022e-01 5.97666310031556278e-08
3.48122867653281989e-07 9.99999651877132356e-01
2 % the third one
Transition % name of CPT
1 % number parents
2 2 % parent cardinality = 2, self cardinality = 2;
9.09090862189374960e-01 9.09091378106250125e-02
1.33333318275715290e-01 8.66666681724284738e-01



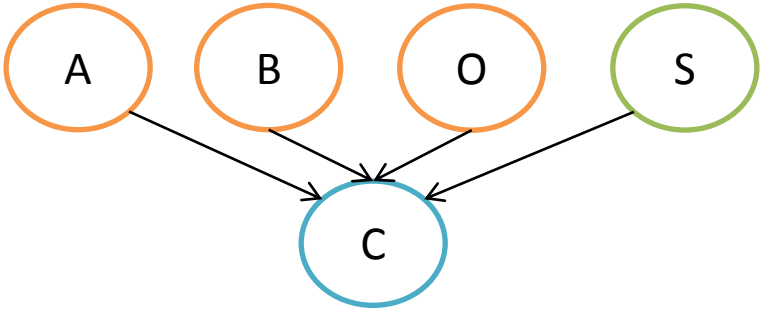
Master file

- If you want to fix some parameters when you run gmtkEMtrainNew,
  - add [–objsNotToTrain filename] argument
  - otherwise they will be updated during the training.

- For example, if you add one line “DENSECPT InitialState” to the not-to-train file, the CPT of InitialSate will be fixed during the training.

Generalized CPT using Decision Tree

- Assume there's a Bayesian Network with five nodes

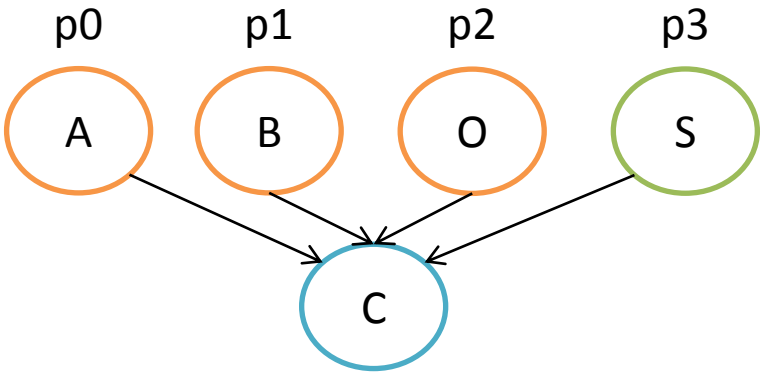


- And we want to express an OR relationship $\begin{cases} C=A, \text{ if } S=0 \\ C=B, \text{ if } S=1 \\ C=O, \text{ others} \end{cases}$

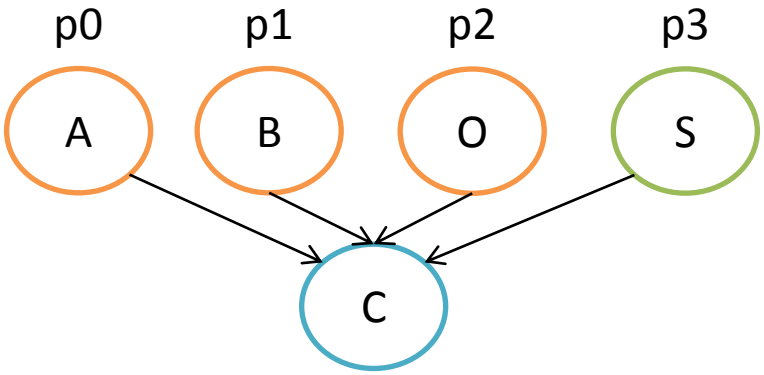
- We can use generalized CPT to represent it.
  - That is, we use a decision tree to define a CPT.
  - It may be helpful for hw1.

str File for the BN

```
GRAPHICAL_MODEL OrModel

frame : 0{
  ...
  variable : C{
    type: discrete observed 0:0 cardinality 2;
    switchingparents: nil;
    conditionalparents: A(0), B(0), O(0), S(0)
          using DeterministicCPT(“orCPT");
  }
}

chunk 0:0
```



master file for the BN



DETERMINISTIC_CPT_IN_FILE inline
1 % number of deterministic CPT
0 % the first one
orCPT % name of CPT
4 % number of parent
3 3 3 2 3 % cardinality of A, cardinality of B, cardinality of O, cardinality of S, cardinality of C
orDT % name of DT used for this CPT

DT_IN_FILE inline
1 % number of decision tree
0 % the first one
orDT % name of DT
4 % number of parent
3 3 0 1 default % (query parent, number of splits, integer range, integer range, “default”)
-1 {{p0}} % first split, p0: the value of the first parent
-1 {{p1}} % second split, p1: the value of the second parent
-1 {{p2}} % default split, p2: the value of the third parent

Example program

- The example program is in “pgm_hw1/example”
  - Type the command “make train” to start training.
  - The learned CPTs can be found in hmm.cpt.
  - The structure file is hmm.str.

- You can see the Makefile for the detailed arguments.

Reference

- GMTK documentation files
  - http://ssli.ee.washington.edu/~bilmes/gmtk/doc.pdf

PGM Hw1 part1

- Two hand-written questions and two programming problems.

- Deadline:
  - Hand-written section: 10/9 (Tue) before class
  - Programming section: 10/14 (Sun) 23:59