Lecture 7: GMTK

Lecturer: Mark Hasegawa-Johnson (jhasegaw@uiuc.edu)
TA: Sarah Borys (sborys@uiuc.edu)
Demo Constructed By: Arthur Kantor (akantor@uiuc.edu)

January 15, 2009

1 Dynamic Bayesian Networks

A "Bayesian network" (BN) is a graph in which every node represents a random variable, and edges represent stochastic dependence. Fig. 1, for example, is a graphical representation of the following joint probability:

$$p(a,b,c,d,e,f) = p(a)p(b|a)p(c|a)p(d|b,c)p(e|c)p(f|c) \tag{1}$$

A "dynamic Bayesian network" (DBN) is a network that can dynamically change its size. In a hidden Markov model, for example, in order to recognize an audio file of T frames, we need to compute the values of T hidden state variables, $q_1, \ldots, q_T$. The state variables q_2 and q_3 are usually structurally identical; they depend on similar context variables, only shifted by one frame. Structures of this kind can be represented by allowing the network to dynamically "unroll" in order to create T consecutive state variables. Fig. 2, copied from the GMTK book (http://ssli.ee.washington.edu/ bilmes/gmtk/), shows how a DBN is unrolled in GMTK. Notice two things about this figure:

1. GMTK assumes that the initial and final frames are different from the template, and must be separately specified. Usually, this makes sense: the initial frame can not depend on history, and the final frame can not depend on the future (other frames can).

2. The template can be more than one frame in length—in this case, three frames. The GMTK book includes examples showing how multi-frame templates can be used for multi-rate speech recognition, e.g., recognizing video frames at a rate of 33fps and audio frames at a rate of 100fps.

2 DBNs for Automatic Speech Recognition

In order to perform large vocabulary speech recognition, you need to keep track of the word, phone, and subphone identity at all time. You also need some auxiliary bookkeeping information (where are you within the word, etc.). In a FST decoder, all of these pieces of information are compiled into a single search

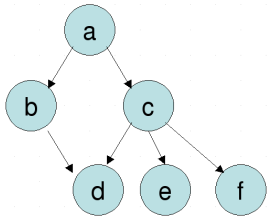

Figure 1: A simple Bayesian network, with six random variables.

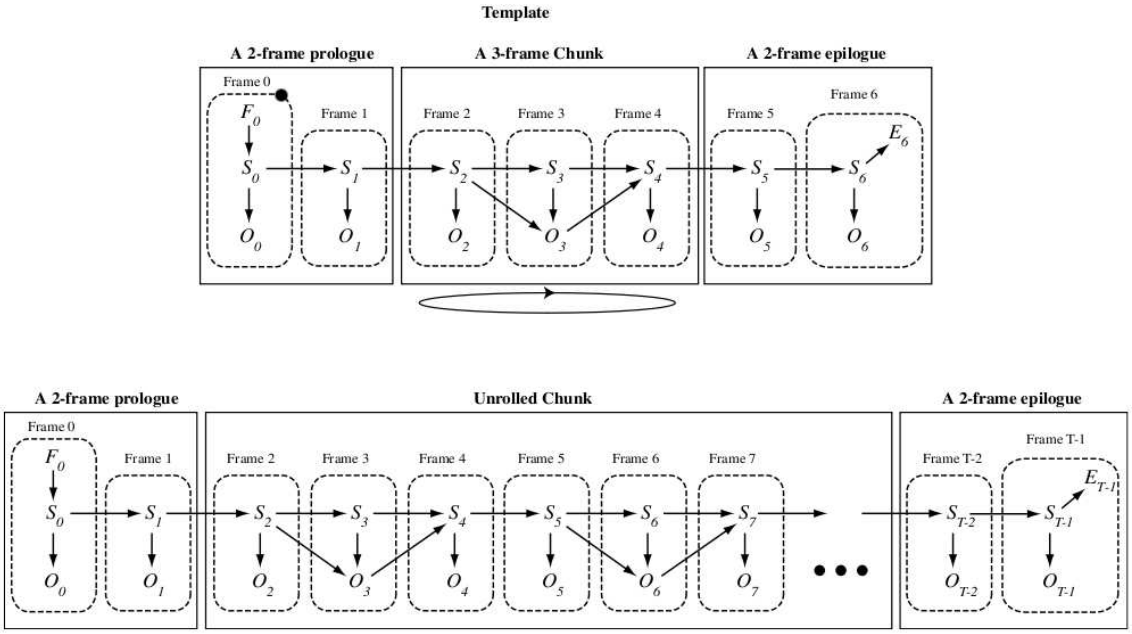

Figure 2: A dynamic Bayesian network can be stretched out, to the length of any particular audio file, by "unrolling" the middle-section template (figure copied from the GMTK book, (http://ssli.ee.washington.edu/ bilmes/gmtk/).

graph [6]. A dynamic Bayesian network keeps track of this information differently: each one of these variables is explicitly represented as a random variable at each time step [7].

The resulting Bayesian nets are HUGE, and look extremely awkward at first glance (e.g., Fig. 3). However, they are VERY powerful. If you want to separately represent the phones being produced by two talkers, for example, it's easy: just add the variables to the graph [3]. If you want to separately represent movements of the lips, tongue, and glottis, it's easy: just split the "phone" variable into separate variables for "lips," "tongue," and "glottis" [5]. If you want to represent audio and video measurements distinctly [2, 4], or if you want to represent wavelet features naturally measured at different time scales [1], those things are also relatively simple.

Fig. 3 shows an example of a dynamic Bayesian network used to train a triphone-based speech recognizer. `wordCounter` specifies which of the words in the current sentence is being heard at time t, `word` specifies the identity of the word, `wordTransition` is a binary random variable set equal to one if and only if a word transition is about to occur. `phoneCounter` specifies which of the phones in the current word is being heard, `phone` is the phone label, and `phoneTransition` indicates a transition. Similar variables are designed for the `subPhone`. `state` specifies the distribution from which the observation (`obs`) has been drawn (according to the hypothesis currently under evaluation). Notice that only `obs` and `subPhoneTransition` are really random variables; in the terminology of hidden Markov models, these two random variables are generated by the observation probability and the state transition probability, respectively. All other variables are deterministic functions of their parents.

3 GMTK

In order to understand GMTK, you really need, primarily, to understand how the information in Fig. 3 is specified to the GMTK re-estimation program (gmtkEMtrain) and to the GMTK decoder program (gmtkViterbi). Four types of files are primarily worth knowing about. They will be listed here, but for descriptions, please consult the GMTK book, or the example code (2009MiniCourseGMTK.tgz) on the course web page.

1. A **structure file** (optionally named with extension `str`, e.g., `train.str`, or `test.str`) specifies the

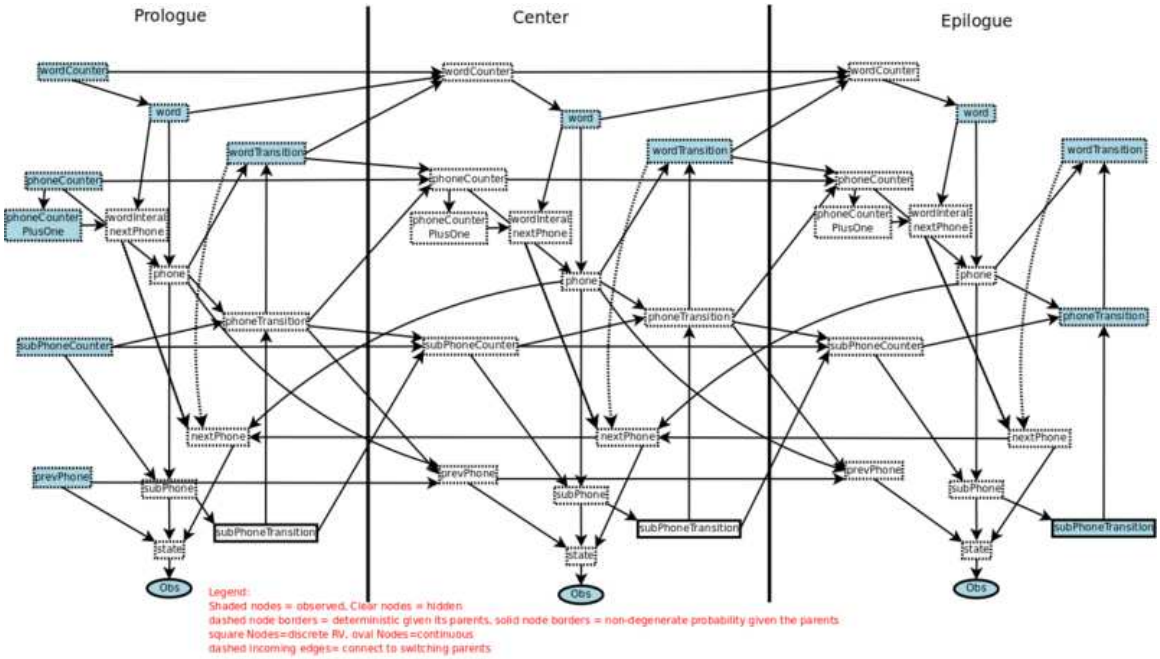

Figure 3: Example of a dynamic Bayesian network used to train a triphone-based speech recognizer. See text for variable definitions.

names of the random variables shown in Fig. 3, the parents of each variable, and parameters as necessary to determine its probability distribution:

(a) Each variable may be hidden or observed. If it is observed, its value must be specified in the structure file, or the structure file must specify a row in the observation file from which its value may be read.

(b) Each variable may be continuous or discrete. If it is discrete, its cardinality must be specified (cardinality=the number of distinct values from which it is chosen).

(c) Each variable may be generated by a `DeterministicCPT` (value of the variable is deterministically specified by its parents), a `DenseCPT` (the dense CPT is a matrix specifying $p(\text{var}|\text{parents})$), or a `mixture collection`.

2. A **master file** (optionally named with extension `mfl`, e.g., `triphones.mfl`) specifies the non-trainable probability densities. These typically are of two types:

   (a) A `DeterministicCPT` specifies how the value of a child variable should be computed from the values of its parents. Usually, this computation is performed using a `DT` (decision tree) object, stored later in the same file. A decision tree object is a tree of up to N levels; at each level, the value of one of the parent variables is examined, and determines which branch should be followed down to the next level. After examining all of the parent variables, one reaches the bottom of the tree, where the resulting value of the child variable is specified.

   (b) A `DenseCPT` object may be fixed, but usually it is trainable.

3. A **learned parameter file** (often laeled with extension gmtk, e.g., `learnedParams.gmtk`) specifies the probability densities that have been (or will be) trained from data. These are of two types.

   (a) A `DenseCPT` is a matrix. The (i, j)th element of the matrix specifies $P(\text{child} = j|\text{parents} = i)$, the probability that the child takes on its jth possible value, given that its parents take on their ith possible combination of values. For example, consider a child with a cardinality of 4, and whose parents have cardinalities of 3, 5, and 7 respectively; the resulting `DenseCPT` is a matrix of size 105×4.

(b) A `mixture collection` is a list of probability densities, each having the following form:

$$p(x|y) = \sum_{k=1}^{K} P(k|y)p(x|\mu_{ky}, \Sigma_{ky}) \tag{2}$$

where x is the child (a continuous random variable), y is the set of parent variables (all of which are discrete random variables), and k is the mixture component. The objects in Eq. 2 are stored in the mixture collection as five different types of sub-object:

i. For each possible value of the parent y, there is a `gm` (Gaussian mixture) object, $p(x|y)$, specifying the name of a `mx` object and the names of K `gc` objects.

ii. The `mx` (mixture CPT) object is a `DenseCPT`, containing K trainable probabilities.

iii. The `gc` (Gaussian component) object specifies the names of a `mean` vector and a `covar` variance.

iv. The `mean` vector is a trainable mean vector.

v. The `covar` matrix usually stores only the diagonal elements of the covariance matrix.

4. An **observations** file may be specified in either HTK format or ICSI format.

References

[1] Özgür Cetin. *Multi-rate Modeling, Model Inference, and Estimation for Statistical Classifiers.* PhD thesis, University of Washington, Seattle, WA, 2004.

[2] Stephen Chu and Thomas S. Huang. Bimodal speech recognition using coupled hidden Markov models. In *Proc. Internat. Conf. Spoken Language Processing*, pages 747–50, Beijing, 2000.

[3] Ameya Nitin Deoras and Mark Hasegawa-Johnson. A factorial HMM approach to simultaneous recognition of isolated digits spoken by multiple talkers on one audio channel. In *Proc. ICASSP*, 2004.

[4] Mark Hasegawa-Johnson, Karen Livescu, Partha Lal, and Kate Saenko. Audiovisual speech recognition with articulator positions as hidden variables. In *International Conference on Phonetic Sciences*, Saarbrucken, 2007.

[5] Karen Livescu. *Feature-Based Pronunciation Modeling for Automatic Speech Recognition.* PhD thesis, MIT, 2005.

[6] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16:69–88, 2002.

[7] Geoffrey Zweig. *Speech Recognition with Dynamic Bayesian Networks.* PhD thesis, U. C. Berkeley, 1998.