An IRT-based Parameterization for Conditional Probability Tables

Russell G. Almond
Educational Psychology and Learning Systems
College of Education
Florida State Univeristy
Tallahassee, FL 32312

Abstract

In educational assessment, as in many other areas of application for Bayesian networks, most variables are ordinal. Additionally conditional probability tables need to express monotonic relationships; e.g., increasing skill should mean increasing chance of a better performances on an assessment task. This paper describes a flexible parameterization for conditional probability tables based on item response theory (IRT) that preserves monotonicity. The parameterization is extensible because it rests on three auxiliary function: a *mapping* function which maps discrete parent states to real values, a *combination* function which combines the parent values into a sequence of real numbers corresponding to the child variable states, and a *link* function which maps that vector of numbers to conditional probabilities. The paper also describes an EM-algorithm for estimating the parameters, and describes a hybrid implementation using both R and Netica, available for free download.

1 INTRODUCTION

The most commonly used parameterization for learning conditional probability tables (CPTs) in discrete Bayesian networks with known structure is the hyper-Dirichlet model (Spiegelhalter & Lauritzen, 1990). Spiegelhalter and Lauritzen show that under two assumptions, global parameter independence and local parameter independence, the hyper-Dirichlet distribution is a natural conjugate of the conditional multinomial distribution, that is, a Bayesian network. In the complete data case, parameter learning is accomplished by generating contingency tables corresponding to the CPT and simply counting the number of events corresponding to each combination in the training data. This counting algorithm is easily extended to an EM algorithm when some of the variable are missing at random (e.g., some are latent), and this EM algorithm is implemented in many common Bayesian network software packages (e.g., Netica; Norsys, 2012).

In many applications, all of the variables in the Bayesian network are ordinal and the network is expected to be monotonic (van der Gaag, Bodlaender, & Feelders, 2004), higher values of parent variables are associated with higher values of child variables. For example, in education, increasing ability should result in a better performance (say a higher partial credit score on short answer assessment item). This monotonicity condition is a violation of the local parameter independence assumption, which assumes that the distribution of the parameters for the rows of the CPT are independent.

Even ignoring this assumption violation, the hyper-Dirichlet results may not provide stable estimates of the CPT. If the parent variables of a particular node are moderately to strongly correlated, then certain row configurations may be rare in the training set. For example if Skill 1 and Skill 2 are correlated, few individuals for whom Skill 1 is high and Skill 2 is low may be sampled. The problem gets worse as the number of parents increases, as number of parameters of the hyper-Dirichlet distribution grows exponentially with the number of parents.

The solution is to build parametric models for the CPTs. A common parametric family is based on noisy-min and noisy-max models (Díez, 1993; Srinivas, 1993). Almond et al. (2001) proposed a method based adapting models from item response theory (IRT). This new model had three parts: (1) a *mapping* from the discrete parent variables to an *effective theta*, a value on an equal interval real scale, (2) a *combination* function that combined the effective thetas for each parent variable into a single effective theta for the item, and (3) a *link* function, based on the graded response model (Samejima, 1969). Almond (2010) and Almond, Mislevy, Steinberg, Yan, and Williamson (2015) extend and develop this model adding a new mapping function and new link function based on the probit function. Almond, Kim, Shute, and Ventura (2013) provides an additional extension based on the generalized partial credit

model (Muraki, 1992).

This paper organizes these IRT-based parametric models into an extensible framework that can be implemented in the R language (R Core Team, 2015). It introduces a Parameterized Node object which has two functional attributes: `rules`—which specifies the combination rules,—and `link`—which specifies the link function. It also has a `parent tvals` method which specifies the mapping from discrete states to effective thetas, which can be overridden to extend the model Because R is a functional language (functions can be stored as data), this creates an open implementation protocol (Maeda, Lee, Murphy, & Kiczales, 1997) which can be easily extended by an analyst familiar with R.

2 ITEM RESPONSE THEORY FOR CONDITIONAL PROBABILITY TABLES

Item response theory (IRT; Hambleton, Swaminathan, & Rogers, 1991) describes a family of models for the performance of an examinee on a assessment. Let X_{ij} be a binary variable representing whether Examinee i got Item j correct or incorrect and let θ_i be the (latent) ability of the examinee. The probability of a correct response is modeled as a function of the examinee ability and item parameters. A typical parameterization is the 2-parameter logistic (2PL) model:

$$\Pr(X_{ij} = 1|pa_i(X_j)) = \frac{exp\left(1.7\alpha_j(\theta_i - \beta_j)\right)}{1 + exp\left(1.7\alpha_j(\theta_i - \beta_j)\right)}\ , \quad (1)$$

also written as $\text{logit}^{-1}\left(1.7\alpha_j(\theta_i - \beta_j)\right)$. The scale parameter α_j is called the *discrimination* and the location parameter β_j is called the *difficulty*. The constant 1.7 a scaling factor used to make the inverse logistic function approximately the same as the normal ogive (probit) function. In order to identify the latent scale, the mean and variance of the latent ability variables θ_i are set to 0 and 1 in the target population. There are many variations on this basic model; two variations for polytomous options are the graded response model (GRM; Samejima, 1969) and the generalized partial credit model (GPC; Muraki, 1992).

A key insight of Lou DiBello (Almond et al., 2001) is that if each configuration of parent variables can be mapped to an *effective theta*, a point on a latent normal scale, then standard IRT models can be used to calculate conditional probability tables. This section reviews the following prior work in IRT-based parameterizations of CPTs. The general framework can be described in three steps:

Mapping Each configuration, i', of the K parent variables is mapped into a real value vector of effective thetas, $\boldsymbol{\theta}(i') = (\tilde{\theta}(i'), \ldots, \tilde{\theta}_K(i'))$.

Combination Rule A combination function, $Z_j(\tilde{\boldsymbol{\theta}}(i'))$, is applied to the effective thetas to yield a combined effective theta for each row of the CPT for Variable j.

Link The link function, $g_j(z)$, is evaluated at $Z_j(\boldsymbol{\theta}(i'))$ to produce the conditional probabilities for Row i' of the CPT for Variable j.

This is an extension of the generalized linear model (McCullagh & Nelder, 1989) with a linear predictor $Z_j(\tilde{\boldsymbol{\theta}})$, and a link function, $g_i(z)$.

Section 2.1 reviews the hyper-Dirichlet model of Spiegelhalter and Lauritzen (1990). Section 2.2 describes how to map discrete parent variables on the theta scale. Section 2.3 describes rules for combining multiple parent variables. Section 2.4 describes the generalized partial credit model, graded response and probit link functions.

2.1 HYPER-DIRICHLET MODEL

The work of constructing a Bayesian network consists of two parts: constructing the model graph, and constructing the conditional probabilities $\Pr(X|\,\text{pa}(X))$, the *conditional probability tables* (CPTs). A CPT is usually expressed as a matrix in which each row corresponds to a configuration of the parent variables and each column corresponds to a state of the child variable. A *configuration* is a mapping of each of the parent variables into a possible state, and the corresponding row of the CPT is the conditional probability distribution over the possible values of the child variable given that the parent variables are in the corresponding configuration.

A commonly used parameterization for Bayesian networks is the *hyper-Dirichlet* model (Spiegelhalter & Lauritzen, 1990). Note that in a discrete Bayesian network, a node which has no parents follows a categorical or multinomial distribution. The natural conjugate prior for the multinomial distribution is the Dirichlet distribution. If the node has parents, then each row of the CPT is a multinomial distribution, and the natural conjugate for that row is also a Dirichlet distribution.

Spiegelhalter and Lauritzen (1990) introduce two additional assumptions. The *global parameter independence assumption* states that the probabilities for the conditional probability tables for any pair of variables are independent. (In the psychometric context, this is equivalent to the assumption that the parameters for different items are independent.) The *local parameter independence assumption* states that the probabilities for any two rows in a CPT are independent. Under these two assumptions, the *hyper-Dirichlet distribution*, the distribution where every row of every CPT is given an independent Dirichlet distribution, is the natural conjugate of the Bayesian network.

Let the matrix A_X be the parameter for the CPT of Variable X. The rows of A_X correspond to the possible configurations of $pa(X)$ and the columns of A_X correspond to the possible states of X, indexed by the numbers 0 to S.[1] Thus, $\mathbf{a}_{i'} = (a_{i'0}, \ldots, a_{i'S})$ (the i'th row of A_X) is the parameters of the Dirichlet distribution for $\Pr(X|pa(X) = i')$.

Let $\hat{Y}_X$ be the matrix of observed counts associated with Variable X. In other words, let $y_{i'v}$ be then number of times when $pa(V) = i'$ that $V = v$. Under the global independence assumptions, this is the sufficient statistic for the CPT. Under the hyper-Dirichlet distribution, the posterior parameter for the CPT for X is $\hat{Y}_X + A_X$. The posterior probability for Row i' is a Dirichlet distribution with parameter, $(\hat{y}_{i'0} + a_{i'0}, \ldots, \hat{y}_{i'S} + a_{i'S})$. The weight given to the data, the observed sample size for the row, is $y_{i'+} = \sum_s y_{i's}$ and the weight given the prior, or *effective sample size* of the prior, is $a_{i'+} = \sum_s a_{i's}$.

The number of elements of A_X grows exponentially with the number of parents. Furthermore, the observed sample of certain rows will be higher, sometimes much higher than others. In particular, when the parent variables are moderately correlated, rows corresponding to configurations where one variable has a high value and the other a low variable will be less common than cases where both parents have similar values. In extreme cases, or with small training samples, $\hat{Y}_{i's}$ will be very close to $a_{i's}$ for those rows.

A second problem is that in many applications the CPTs should be monotonically non-decreasing. When the parent variables take on higher values, the probability that the child variable should take on higher values should not decrease. It is possible, especially with a small training sample, for the estimated CPT to have an unexpected non-monotonic pattern.

The IRT-based CPT parameterizations described below mitigate both of these problems. First, the number of parameters grows linearly in the number of parents. Second, the parameterizations force the CPTs to be monotonically increasing.

2.2 MAPPING DISCRETE VARIABLES ONTO CONTINUOUS SCALES

IRT models assume that the parent variables are continuous. Basing CPTs for discrete Bayesian networks on IRT models requires first assigning a real value $\tilde{\theta}_{mk}$ to each possible state, m, of each parent variable V_k. Following the IRT convention, these numbers should be on the same scale as a unit normal distribution with 0.0 representing the median individual in the population and 1.0 representing an individual one standard deviation from the median.

Almond (2010) suggested using equally spaced quantiles of the normal distribution as effective theta values. For a parent variable V_k that takes one of M_k possible values, the effective theta values for state m is $\Phi^{-1}((2m+1)/2M_k)$, where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. The quantiles need not be equally spaced, but can be matched to any marginal distribution (Almond et al., 2015). However, equally spaced quantiles form a uniform marginal distribution over the parent variable, and thus are ready to be combined with information about the distribution of the parent variable from elsewhere in the network.

The function which assigns a configuration of the parent variables, i' to a vector of effective theta values, $\tilde{\boldsymbol{\theta}}(i')$, is called the *mapping function*. In most cases, this is a composition function consisting of separate mapping functions for each parent variable. The function based on normal quantiles given above is one example of a mapping function; however, any function that maps the states of the parent variables to the values on the real line is a potential mapping function. To preserve monotonicity, the mapping should also be monotonic.

2.3 COMBINATION RULES

In the case of a binary child variable, the IRT 2PL function $g_j(z) = \text{logit}^{-1}(1.7z)$ is a natural link function. The combination rule $Z_j(\tilde{\theta}_1(i'), \ldots, \tilde{\theta}_K(i'))$ must map the effective thetas for the parent variables onto a single dimension representing the item. Also, in order to preserve monotonicity, $Z_j(\cdot)$ must by monotonic.

Almond et al. (2001) noted that changing the functional form of $Z_j(\cdot)$ changed the design pattern associate with the item. They suggested the following functions:

Compensatory This structure function is a weighted average of relevant skill variables. $Z_j(\tilde{\boldsymbol{\theta}}(i')) = \frac{1}{\sqrt{K}} \sum_k \alpha_{jk} \tilde{\theta}_k(i') - \beta_j$. Here $1/\sqrt{K}$ is a variance stabilization term which help keep the variance of $Z_j(\cdot)$ from growing as more parents are added.

Conjunctive This structure function is based on the minimum of the relevant skill variables (weakest skill dominates performance). $Z_j(\tilde{\boldsymbol{\theta}}(i')) = \min_k \alpha_{jk} \tilde{\theta}_k(i') - \beta_j$.

Disjunctive This structure function is based on the maximum of the relevant skill variables (strongest skill dominates performance). $Z_j(\tilde{\boldsymbol{\theta}}(i')) = \max_k \alpha_{jk} \tilde{\theta}_k(i') - \beta_j$.

Inhibitor A conditional function where the if the first skill is lower than a threshold, θ^*, then the value is at a

[1] In the equations, the states run from lowest to highest, but in the implementation the states run from highest to lowest.

nominal low value (usually based on the lowest possible value of the second skill, $\tilde{\theta}_2(0)$), but if the first skill exceeds that threshold, the other skill dominates the value of the function. This models tasks where a certain minimal level of the first skill is necessary, but after that threshold is achieved more of the first skill is not relevant.

$$Z_j(\tilde{\theta}_{1(i')}, \tilde{\theta}_2(i')) = \begin{cases} \beta_j + \alpha_j \tilde{\theta}_2(0) & \text{if } \tilde{\theta}_1(i') < \theta^*, \\ \beta_j + \alpha_j \tilde{\theta}_2(i') & \text{if } \tilde{\theta}_1(i') \geq \theta^*, \end{cases} \tag{2}$$

Note that the difficulty and discrimination parameters are built into the structure functions here. When there are multiple parent variables, there are multiple slope parameters, α_{jk}, giving the relative importance of the parent variables. This makes a lot of sense in the case of the compensatory rule (which is just a linear model). In the case of the conjunctive and disjunctive rules, it often makes more sense to have multiple intercepts (indicating different minimum levels of skill are required for successful performance).

Offset Conjunctive Structure function is based on the minimum of the relevant skill variables with different thresholds. $Z_j(\tilde{\boldsymbol{\theta}}(i')) = \alpha_j \min_k(\tilde{\theta}_k(i') - \beta_{jk})$.

Offset Disjunctive Structure function is based on the maximum of the relevant skill variables with different thresholds. $Z_j(\tilde{\boldsymbol{\theta}}(i')) = \alpha_j \max_k(\tilde{\theta}_k(i') - \beta_{jk})$.

The link function for polytomous items (variables that take on more than two states) typically require a different value $Z_{js}(\tilde{\boldsymbol{\theta}}(i'))$ for each state s of the child variable. Often this is achieved by simply using different values of the difficultly parameter for each s. In the more general case, each level of the dependent variable would have a different set of discriminations, $\boldsymbol{\alpha}_{js}$, and difficulties, $\boldsymbol{\beta}_{js}$, or even potentially a different functional form for $Z_{js}(\tilde{\boldsymbol{\theta}}(i'))$.

Von Davier (2008) proposed a similar class of generalized diagnostic models based on an IRT framework. Von Davier expresses the parent-child relationship proficiency variables and observable outcome variables through a Q-matrix (Tatsuoka, 1984). This is a matrix where the columns represent proficiency variables and the rows represent observable outcomes. When Variable k is thought to be relevant for Item j, then $q_{jk} = 1$; otherwise, $q_{jk} = 0$. Note that the Q-matrix defines a large part of the graphical structure of the Bayesian network (Almond, 2010): there is a directed edge between the Proficiency Variable k and the Observable Outcome Variable j if and only if $q_{jk} = 1$.

In a Bayesian network, the structure of the graph is used in place of the Q-matrix (the Q-matrix approach is really only useful for bipartite graphs where the parent variables and child variables each form distinct sets). However, this notation is useful for situations in which certain parent variables have relevance for certain state transitions. Let $q_{jsk} = 1$ if Parent k of Item j is relevant for the transition from state $s-1$ to s and zero otherwise, so that Q_j is an item specific Q-matrix with columns corresponding to the parent variables. Let the combination rule for state s be $Z_{js}(\tilde{\boldsymbol{\theta}}(i')[\mathbf{q}_{js}]$, where the square bracket represents a selection operator (it is styled after the R selection operator) which selects those values of $\tilde{\boldsymbol{\theta}}(i')$ for which $q_{jsk} = 1$. Note that in this case, the parameters for different states of the child variable (e.g., $\boldsymbol{\alpha}_{js}$ and $\boldsymbol{\alpha}_{js'}$) could have different dimensions.

2.4 LINK FUNCTIONS

The expressions $Z_{js}(\boldsymbol{\theta}(i')[\mathbf{q}_{js}])$ associates a real value with each configuration of parent variables, i', and each state of the child variable, s (although usually $Z_{j0}(\cdot)$ is set to a constant such as zero or infinity). The goal of the link function $g_j(\cdot)$ is to change these values into the conditional probability function. In the case of the binary variable, the inverse logit function is a possible link function. This section describes three more possibilities for cases where there the child variable has more than one state.

Graded Response Link. Almond et al. (2001) suggested using the graded response function (Samejima, 1969) for the link function. The graded response is based off of a series of curves representing $\Pr(X_j \geq s|\theta)$, and the probability that $X_j = s$ is given by subtracting two adjacent curves. In this case, let $P^*_{js}(i') = \text{logit}^{-1}\left(1.7 Z_{js}(\tilde{\boldsymbol{\theta}}(i')[\mathbf{q}_j s])\right)$ and set $P^*_{j0} = 1$ and $P_{j,S+1} = 0$, where the states are represented by integers going from 0 to S. Then the entries for Row i' of the conditional probability table will be $p_{js}(i') = P^*_{js}(i') - P^*_{j,s+1}(i')$.

Note that in order for the graded response function to not produce negative probabilities, it must be the case that $Z_{j1}(\tilde{\boldsymbol{\theta}}(i')[\mathbf{q}_{j1}]) < Z_{j2}(\tilde{\boldsymbol{\theta}}(i')[\mathbf{q}_{j2}]) < \cdots < Z_{jS}(\tilde{\boldsymbol{\theta}}(i')[\mathbf{q}_{jS}])$ for all i'. Note that the easiest way to achieve this is to have all of the $Z_{js}(\cdot)$ with the same functional form and slopes, differing only in the intercepts.

Probit Link. Almond et al. (2015) presents an alternative way of thinking about the link function based on a regression model. The child variable like the parent variables in mapped onto an effective theta scale. In particular a series of cut points $c_0 = -\infty < c_1 < \cdots < c_S < c_{S+1} = \infty$ are established so that $\Phi(c_{s+1}) - \Phi(c_s)$ provides a desired marginal probability (Almond, 2010). The values $Z_j(\tilde{\boldsymbol{\theta}}(i'))$ are a series of linear predictors for each row, i', of the CPT, and the conditional probabilities $\Pr(X_j = s|i')$ is calculated by finding the probability that a normal random variable with mean $Z_j(\tilde{\boldsymbol{\theta}}(i'))$ and standard deviation σ_j falls between c_s and c_{s+1}.

Note that this link function uses a single combination rule

for all states of the child variable. It also introduces a *link scale parameter*, σ_j. Guo, Levina, Michailidis, and Zhu (2015) introduce a similar model with $\sigma_j = 1$.

Partial Credit Link. Muraki (1992) introduces an IRT model appropriate for a constructed response item where the solution to the problem requires several steps. Assume that the examinee is assigned a score on that item based on how many of the steps were completed. If Item j has S_j steps, then the possible scores are $0, \ldots, S_j$. For $s > 0$, let $P_{js|s-1}(\tilde{\boldsymbol{\theta}}(i')) = \Pr(X_j \geq s | X_j \geq s-1, \tilde{\boldsymbol{\theta}}(i')) = \text{logit}^{-1}(1.7 Z_{js}(\tilde{\boldsymbol{\theta}}(i')[\mathbf{q}_j s]))$; that is, let $P_{js|s-1}(\tilde{\boldsymbol{\theta}}(i'))$ be the probability that the examinee completes Step s, given that the examinee has completed steps $1, \ldots, s-1$. Note that $\Pr(X_j \geq 0) = 1$, and so let $P_{j0|-1} = 1$, and $Z_{j0}(\tilde{\boldsymbol{\theta}}(i')) = 0$. The probability that Examinee i will achieve Score s on Item j is then:

$$\Pr(X_j = s | \tilde{\boldsymbol{\theta}}(i')) = \frac{\prod_{r=0}^{s} P_{jr|r-1}(\tilde{\boldsymbol{\theta}}(i'))}{C},$$

where C is a normalization constant. Following Muraki (1992), note that this collapses to:

$$\Pr(X_j = s | \tilde{\theta}(i')) = \frac{\exp\left(1.7 \sum_{r=0}^{s} Z_{jr}(\tilde{\boldsymbol{\theta}}(i')[\mathbf{q}_j r])\right)}{\sum_{R=0}^{S_j} \exp\left(1.7 \sum_{r=0}^{R} Z_{jr}(\tilde{\boldsymbol{\theta}}(i')[\mathbf{q}_j r])\right)}. \quad (3)$$

The partial credit link function is the most flexible of the three. Note that each state of the child variable can have a different combination rule $Z_{js}(\cdot)$ as well as potentially different slope and intercept parameters.

3 THE PARAMETERIZED NETWORK AND NODE OBJECT MODELS

The mapping, combination rule and link functions described above represent a non-exhaustive survey of those known to the author at the time this paper was written. Software implementing this framework should be extensible to cover additional possibilities. The implementation in R (R Core Team, 2015) described in this paper uses an open implementation framework (Maeda et al., 1997) to ensure extensibility. The design is opened in two ways. First, because R is a functional language, the combination rule and link functions can be stored as fields of a parameterized node object. While library functions are available for the combination rule and link functions described above, adding variants is straightforward. Second, it uses standard object oriented conventions to allow functions such the mapping function and the function that computes the CPT from the parameters to be overridden by subclasses.

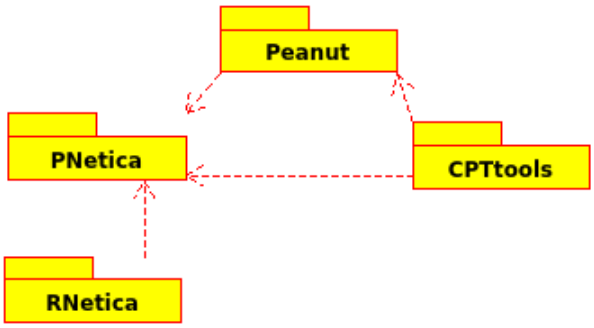

Figure 1: Package Structure

3.1 PACKAGE STRUCTURE

The implementation in R uses four R packages to take advantage of previous work (Figure 1) and to minimize dependencies on the specific Bayes net engine used for the initial development (Netica; Norsys, 2012). The packages are as follows:

CPTtools CPTtools (Almond, 2015) is an existing collection of R code that implements many of the mapping, combination rule and link functions described above. In particular, it creates CPTs as R data frames (tables that can store both factor and numeric data). The first several factor-valued columns of the data frame describe the configurations of the parent variables and the last few numeric columns the probabilities for each of the states.

RNetica RNetica (Almond, 2015) is basically a binding in R of the C API of Netica. It binds Netica network and node objects into R objects so that they can be accessed from R. It is able to set the CPT of a Netica node from the data frame representation calculated in the CPTtools package.

Peanut Peanut[2] is a new package providing abstract classes and generic functions. The goal is similar to the DBI package (R Special Interest Group on Databases, 2013); that is, to isolate the code that depends a specific implementation to allow for easy extensibility.

PNetica PNetica is a new package that provides an implementation of the Peanut generic functions using RNetica objects. In particular, fields of Peanut objects are serialized and stored as node user data in Netica nodes and networks. Collections of nodes are represented as node sets in Netica.

Peanut uses the S3 class system (Becker, Chambers, & Wilks, 1988) which is quite loose. In particular, any object can register itself as a parameterized node or network,

[2]The name is a corruption of parameterized net.

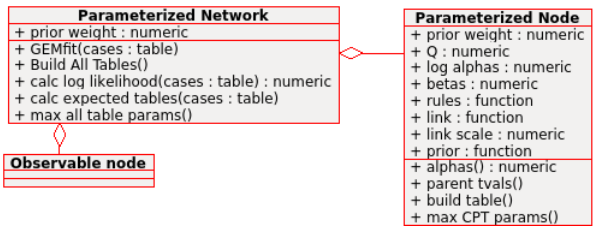

Figure 2: Parameterized Network (Peanut) Class diagram

it just needs to implement the methods described in the object model below. This should make it straightforward to replace the RNetica (which requires a license from Norsys for substantial problems) and PNetica packages with similar packages which adapt the algorithm to a different Bayes net engine.

3.2 OBJECT MODEL

Figure 2 shows the object model for Peanut. There are two important classes, Parameterized Network and Parameterized Node, both of which extend ordinary Bayesian network and node classes. The Parameterized network is mostly a container for Parameterized Node and Observable Node (nodes which are referenced in the case data). Most of its methods iterate over the the parameterized nodes.

The attributes[3] can be divided into two groups: numeric attributes representing parameters and functional attributes which take functions (or names of functions) as values. In the key methods, `build tables()` and `max CPT params()` the functional attributes are applied to numeric values to construct the tables.

The three functional attributes, `rules`, `link` and `prior` all must have specific signatures as they perform specific functions.

rule(theta,alpha,beta) A combination rule must take table of effective thetas, corresponding to the configuration of parents, and return a vector of values, one for each row. Note that in R all values can be either scalars or vectors, so that there can be either a different alpha or different beta for each parent (depending on the specific rule.

link(et,link scale=NULL) When the rule is applied repeatedly, once for each state except for the last one (CPTtools assumes the states are ordered highest to lowest), the result is a matrix with one fewer columns than the child variable has states. The link function takes this matrix and produces the conditional probability tables. Note that there is an optional link scale parameter which is used for the probit method, but not for the partial credit or graded response method.

prior(log alpha,beta,link scale) This takes a set of parameters and returns the log prior probability for that parameter set. Note that any of the parameters may be vectors or lists.

Figure 3 describes the `calcDPCTable` function (part of the `CPTtools` package) which shows how these functional arguments are used. Key to understanding this code is that the R function `do.call` takes two arguments, a function and a list of arguments and returns the value of evaluating the call. Consequently, the `rules` and `link` argument are functions or names of functions. Also, R is vectorized. So if the `rules` argument is a list of rules, then a different function will be applied for each state of the child variable. Similarly if the `lnAlphas`[4] and `betas` are lists, then different values are used for each state. The `build table()` method of the Parameterized Node class is just a wrapper which calls `calcDPCTable` with arguments taken from its attributes and uses it to set the CPT of the node.

The use of functional arguments makes `calcDPCtable` (and hence the Parameterized Node) class easy to extend. Any function that fits the generic model can be used in this application, so the set of functions provided by `CPTtools` is easily extended. The mapping function is slightly different, as the mapping is usually associate with the parent variable. As a consequence, a generic function `parent tvals()` is used to calculated the effective values. The current implementation for Netica nodes retrieves them from the parent nodes, but this could be overridden using a subclass of the parameterized node.

4 AN EM ALGORITHM FOR ESTIMATING PARAMETERIZED NODE PARAMETERS

A large challenge in learning educational models from data is that the variables representing the proficiencies are seldom directly observed. The EM algorithm (Dempster, Laird, & Rubin, 1977) provides a framework for MAP or MLE estimation in problems with latent or missing variables. The EM algorithm alternates between two operations:

E-Step Calculate the expected value of the likelihood or posterior by integrating out over the missing values using the current set of parameter values. When

[3]These are expressed attributes in the object diagrams, but implemented through accessor methods.

[4]Because in educational settings, the discrimination parameters are restricted to be strictly positive, a log transformation of the alphas is used in parameter learning instead of the original scale.

```
calcDPCTable <-
  function (skillLevels, obsLevels,
    lnAlphas, betas,
    rules="Compensatory",
    link="partialCredit",
    linkScale=NULL,Q=TRUE,
    tvals=lapply(skillLevels,
      function (sl)
        effectiveThetas(length(sl)))
    ) {
  ## Error checking and argument
  ## processing code omitted

  ## Create table of effective thetas
  thetas <- do.call("expand.grid",tvals)

  ## Apply rules for each state (except
  ## last) to build per state table of
  ## effective thetas.
  et <- matrix(0,nrow(thetas),k-1)
  for (kk in 1:(k-1)) {
    et[,kk] <-
      do.call(rules[[kk]],
              list(thetas[,Q[s,]],
                   exp(lnAlphas[[kk]]),
                   betas[[kk]]))
  }

 ## Apply Link function to build the CPT
 do.call(link,list(et,linkScale,
                   obsLevels))
}
```

Figure 3: Function to build CPT

the likelihood comes from an exponential family, as the conditional multinomial distribution does, this is equivalent to setting the sufficient statistics for the distribution to their expected values.

M-Step Find values for the parameters which maximize the complete data log posterior distribution.

Dempster, Laird and Rubin show that this algorithm will converge to a local mode of the posterior distribution.

For Bayesian network models, the E-Step can take advantage of the conditional independence assumptions encoded in the model graph. Adopting the *global parameter independence* assumption of Spiegelhalter and Lauritzen (1990) allows the M-Step to take advantage of the graphical structure as well. The global parameter independence assumption says that conditioned on the data being completely observed, the parameters of the CPTs for different variables in the network are independent. This means that the structural EM algorithm (Meng & van Dyk, 1997) can be used, allowing the M-Step to be performed separately for each CPT.

Another simplification can be found through the use of sufficient statistics. Under the global global parameter independence assumption, the contingency table formed by observing X_j and $pa(X_j)$, $\hat{Y}_{X_j}$, is a sufficient statistic for $\Pr(X_j|pa(X_j)$ (Spiegelhalter & Lauritzen, 1990). Thus in the E-Step it is sufficient to work out the expected value for this contingency table. If $\mathbf{O}_i$ is the observed data for Participant i, then this can be found by $\tilde{Y}_{X_j} = \sum_i \Pr(X_{ij}, pa(X_{ij})|\mathbf{O}_i)$; the inner term of the sum can be calculated by normal Bayesian network operations. In this case the M-Step consists of finding the values of the parameters for $\Pr(X_j|pa(X_j))$, $\boldsymbol{\theta}_j$, that maximize

$$\sum_{i'} \Pr(\tilde{y}_{i'X_j}|pa(X_j) = i', \boldsymbol{\theta}_j) \tag{4}$$

It is quite likely that sparseness in the data will cause cells in $\tilde{Y}_{X_j}$ to have zero or near-zero values. This can come about for two reasons. One is that certain responses may be rare under certain configurations of the parent variables. The second is that often Bayesian networks are used to score complex constructed response tasks, and some variables may not be observed unless the examinee employs a certain approach to problem, producing a low effective sample size for that variable. In either case, the zero values are a problem for maximum likelihood estimation. Equally problematic are values that are small integers, because the parameter values may be determined by only a handful of observations.

A common approach used in the analysis of contingency tables is to add a value less than one (usually 0.5) to each cell of the table. (Note that the Dirichlet distribution with all

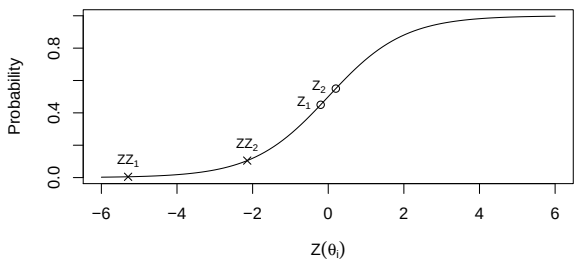

Figure 4: Two possible solutions (labeled Z and ZZ) for a probability difference of .1

parameters equal to 0.5 is the non-informative prior produced by applying Jeffrey's rule.) Applying any proper prior distribution mitigates the problem with low cell count, and often the Bayesian network construction process elicits a proper prior parameters from the subject matter experts. In this case, suppose the experts supply parameter values $\boldsymbol{\psi}_j^{(0)}$, then $\Pr(X_j|pa(X_j), \boldsymbol{\psi}_j^{(0)})$ is a matrix containing the expected value of a Dirichlet prior for the CPT for X_j. To get the Dirichlet prior, choose a vector of weights $\mathbf{w}_j$ where $w_{i'j}$ is the weight in terms of number of observations to be given to Parent Configuration i'. Then set the Dirichlet prior $A_j = \mathbf{w}_j^t \Pr(X_j|pa(X_j), \boldsymbol{\psi}_j^{(0)})$. The the expression to maximize in the M-step becomes:

$$\sum_{i'} \Pr(\tilde{y}_{i'X_j} + a_{i'j}|pa(X_j) = i', \boldsymbol{\psi}_j) \qquad (5)$$

This provides a semi-Bayesian approach to estimating the CPT parameters.

If the discrete partial credit model is used for the conditional probability table, Equation 5 finds the points on the logistic curve that will maximize the likelihood of the observed data. However, the full logistic curve does not enter the equation, just the points corresponding to possible configurations of the parent variables. Consider a very simple model where the target variable has one parent with two states. Assume that the conditional probabilities for one outcome level given those states differ by .1. There are multiple points on the logistic curve that provide that probability difference, Figure 4 illustrates this. The two points marked with circles (Z_1 and Z_2) differ by .1 on the y-axis and correspond to a solution with a difficulty of zero and a discrimination of around 0.3. The two points marked with crosses (ZZ_1 and ZZ_2) also differ by .1 on the y-axis and correspond to a solution with a difficulty of 2.2 and a discrimination of 2.3.

A fully Bayesian estimation would not have this identifiability problem: even if the solutions shown in Figure 4 have identical likelihoods, the posterior distribution would differ, and provide a preference for the solution which is closer to the normal range of the parameters. Assuming that the experts have provided initial values for the parameters, $\boldsymbol{\psi}_j^{(0)}$, a weakly informative prior can be created by using a normal distribution with a mean at $\boldsymbol{\psi}_j^{(0)}$ and the variance chosen so that the 95% interval includes all of the reasonable values for the parameters. Let the resulting distribution be $\pi(\boldsymbol{\psi}_j)$. The M-Step for the fully Bayesian solution then maximizes:

$$\sum_{i'} \Pr(\tilde{y}_{i'X_j} + a_{i'j}|pa(X_j) = i', \boldsymbol{\psi}_j)\pi(\boldsymbol{\psi}_j) \; . \qquad (6)$$

4.1 IMPLEMENTATION OF THE GENERALIZED EM ALGORITHM IN PEANUT

The `GEMfit` method takes advantage of the fact that the E-step of the EM algorithm is already implemented in Netica! Netica, like many other Bayes net packages, offers a version of the EM algorithm for hyper-Dirichlet distributions described in Spiegelhalter and Lauritzen (1990). In Netica, running the learning algorithm on the case file (basically, a table of values of observable nodes for various subjects) produces both a new CPT for each node as well as a set of posterior weights.[5] Multiplying the new table by the prior weights produces the table of expected counts which is the sufficient statistic required for the E-step.

Figure 5 shows the EM algorithm implementation in Peanut. Note the most of the work is done by four generic functions `BuildAllTables()`, `calcPnetLLike()`, `calcExpTables()`, and `maxAllTableParameters()`, allowing the EM algorithm to be customized by overriding those methods. The functions `BuildAllTables()` and `maxAllTableParameters()` iterate over the Parameterized Nodes, allowing for further customization at the node rather than the net level.

In particular, these functions perform the following roles:

PnetBuildTables() This calculates CPTs for all Parameterized Nodes and sets the CPT of the node as well as its prior weight.

calcPnetLLike(cases) This calculates the log likelihood of the case files using the current parameters. In the Netica implementation, it loops over the rows in the case file and calculates the probability of each set of findings. The log probabilities are added together to produce the log-likelihood for the current parameter values.

calcExpTables(cases) This runs the E-step. In the Netica implementation it calls Netica's EM learning algorithm.

[5]These are called *node experience* in Netica.

```
GEMfit <-
  function (net, cases, tol=1e-6,
            maxit=100, Estepit=1,
            Mstepit=3) {

  ## Base case
  converged <- FALSE
  llike <- rep(NA, maxit+1)
  iter <- 1

  ## This next function sets both the
  ## prior CPTs and the prior weights
  ## for each node.
  BuildAllTables(net)

  ## Initial value of likelihood for
  ## convergence test
  llike[iter] <-
    calcPnetLLike(net,cases)

  while(!converged && iter <= maxit) {
    ## E-step
    calcExpTables(net, cases,
        Estepit=Estepit, tol=tol)

    ## M-step
    maxAllTableParams(net,
      Mstepit=Mstepit, tol=tol)

    ## Update parameters and
    ## do  convergence test
    iter <- iter + 1
    BuildAllTables(net)
    llike[iter] <-
      calcPnetLLike(net,cases)
    converged <-
      (abs(llike[iter]-
           llike[iter-1]) < tol)
  }

  list(converged=converged,iter=iter,
       llikes=llikes[1:iter])
}
```

Figure 5: Generalized EM algorithm for Parameterized Networks

maxAllTableParams() This runs the M-step. It iterates over the `max CPT params()` methods for the Parameterized Node objects. The Netica implementation calculates the expected table from the current CPT and posterior weights. It then calls the `CPTtools` function `mapDPC()`. Like `calcDPCTable()` this function takes most of the attributes of the Parameterized Node as arguments. It then finds a new set of parameters (of the same shape as its input) to fit the expected table using a gradient decent algorithm.

Note that neither the EM algorithm embedded in the E-step (the one run natively in Netica) nor the gradient decent algorithm in the M-step need to be run to convergence. The convergence test is done at the level of the whole algorithm (this makes it a Generalized EM algorithm).

The `prior weight` is a tuning parameter for the algorithm. High values of `prior weight` will cause the estimates to shrink towards the prior (expert) values. Low values of `prior weight` will cause the estimates to be unstable when the data are sparse. I have found that values around 10 provide a good compromise.

5 SOFTWARE AVAILABILITY

The software is available for download from `http://pluto.coe.fsu.edu/RNetica`.[6] All of the software is open source, which also should assist anyone trying to extend it either to cover more distribution types or different Bayes net engines; however, RNetica links to the Netica API which requires a license from Norsys.

The framework described here is very flexible, perhaps too flexible. First, it allows an arbitrary number of parameters for each CPT. There is no guarantee that those parameters are identifiable from the data. In many cases, the posterior distribution may look like the prior distribution. Second, R has a very weak type system. The framework assumes that the combination rules and link functions are paired with appropriate parameters (alphas, betas and link scale parameters). This is not checked until the `calcDPCTable` function is called to build the table meaning that errors could be not caught until the analyst tries to update the network with data.

Despite these weaknesses, the framework is a flexible one that supports the most common design patterns used in educational testing applications (Mislevy et al., 2003; Almond, 2010; Almond et al., 2013, 2015). Hopefully, these design patterns will be useful in other application areas as well. If not, the open implementation protocol used in the framework design makes it easy to extend.

[6] The code is still be tested, check back for updates.

References

Almond, R. G. (2010). 'I can name that Bayesian network in two matrixes'. *International Journal of Approximate Reasoning*, *51*, 167–178. Retrieved from http://dx.doi.org/10.1016/j.ijar.2009.04.005 doi: 10.1016/j.ijar.2009.04.005

Almond, R. G. (2015, May). RNetica: Binding the Netica API in R (-3.4 ed.) [Computer software manual]. Retrieved from http://pluto.coe.fsu.edu/RNetica/RNetica.html (Open source software package)

Almond, R. G., DiBello, L., Jenkins, F., Mislevy, R. J., Senturk, D., Steinberg, L. S., & Yan, D. (2001). Models for conditional probability tables in educational assessment. In T. Jaakkola & T. Richardson (Eds.), *Artificial intelligence and statistics 2001* (p. 137-143). Morgan Kaufmann.

Almond, R. G., Kim, Y. J., Shute, V. J., & Ventura, M. (2013). Debugging the evidence chain. In R. G. Almond & O. Mengshoel (Eds.), *Proceedings of the 2013 uai application workshops: Big data meet complex models and models for spatial, temporal and network data (UAI2013AW)* (pp. 1–10). Aachen. Retrieved from http://ceur-ws.org/Vol-1024/paper-01.pdf

Almond, R. G., Mislevy, R. J., Steinberg, L. S., Yan, D., & Williamson, D. M. (2015). *Bayesian networks in educational assessment*. Springer.

Becker, R. A., Chambers, J. M., & Wilks, A. R. (1988). *The new S language: a programming environment for data analysis and graphics*. Wadworth & Brooks/Cole.

Dempster, A. P., Laird, N., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *JRSS B*, *39*, 1-38.

Díez, F. J. (1993). Parameter adjustment in Bayes networks. the generalized noisy or-gate. In D. Heckerman & A. Mamdani (Eds.), *In uncertainty in artificial intelligence 93* (pp. 99–105). Morgan-Kaufmann.

Guo, J., Levina, E., Michailidis, G., & Zhu, J. (2015). Graphical models for ordinal data. *Journal of Computational and Graphical Statistics*, *24*(1), 183-204. Retrieved from http://dx.doi.org/10.1080/10618600.2014.889023 doi: 10.1080/10618600.2014.889023

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. Sage.

Maeda, C., Lee, A., Murphy, G., & Kiczales, G. (1997). Open implementation analysis and design. In *Ssr '97: Proceedings of the 1997 symposium on software reusability* (pp. 44–52). New York, NY, USA: ACM. doi: http://doi.acm.org/10.1145/258366.258383

McCullagh, P., & Nelder, J. A. (1989). *Generalized linear models. (2nd edition)*. Chapman and Hall.

Meng, X.-L., & van Dyk, D. (1997). The EM algorithm — an old folk-song sung to a fast new tune. *Journal of the Royal Statsitical Society, Series B*, *59*(3), 511-567.

Mislevy, R. J., Hamel, L., Fried, R. G., Gaffney, T., Haertel, G., Hafter, A., ... Wenk, A. (2003). *Design patterns for assessing science inquiry* (PADI Technical Report No. 1). SRI International. Retrieved from http://padi.sri.com/publications.html

Muraki, E. (1992). A generalized partial credit model: Application of an em algorithm. *Applied Psychological Measurement*, *16*(2), 159–176. doi: 10.1177/014662169201600206

Norsys. (2012). Netica-c programmer's module (5.04 ed.) [Computer software manual]. Retrieved from http://norsys.com/netica_c_api.htm (Bayesian Network Software)

R Core Team. (2015). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from http://www.R-project.org/

R Special Interest Group on Databases. (2013). DBI: R database interface [Computer software manual]. Retrieved from http://CRAN.R-project.org/package=DBI (R package version 0.2-7)

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph No. 17*, *34*(4), (Part 2).

Spiegelhalter, D. J., & Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, *20*, 579–605.

Srinivas, S. (1993). A generalization of the noisy-or model, the generalized noisy or-gate. In D. Heckerman & A. Mamdani (Eds.), *Uncertainty in artificial intelligence '93* (pp. 208–215). Morgan-Kaufmann.

Tatsuoka, K. K. (1984). *Analysis of errors in fraction addition and subtraction problems* (Vol. 20; NIE Final report No. NIE-G-81-002). University of Illinois, Computer-based Education Research.

van der Gaag, L. C., Bodlaender, H. L., & Feelders, A. (2004). Monotonicity in Bayesian networks. In M. Chickering & J. Halpern (Eds.), *Proceedings of the twentieth conference on uncertainty in artificial intelligence* (pp. 569–576). AUAI.

von Davier, M. (2008). A general diagnostic model applied to language testing data. *British Journal of Mathematical and Statistical Psychology*, *61*, 287–307.