

## 1. Overview of Estimation Approach

In the AHM, the artificial neural network (ANN) approach is employed to estimate the probability that examinees possess specific attributes measured on tests (Cui & Gierl, 2009; Gierl, Cui, & Hunka, in press). The ANN is a highly flexible nonlinear statistical technique for modeling complex relationships between a set of independent and dependent variables. In our application to diagnostic testing, ANNs may be used to evaluate student performance by establishing a relation between student response data and student attribute mastery levels. With ANNs, the probability of a student's mastery of an attribute is estimated based on his or her responses to test items. To implement the ANN for the AHM, a program is written using *Mathematica* code. The *Wolfram Mathematica* program was first released in 1988. Currently, *Mathematica* is widely used across the world. Although individual packages had been developed for specific numeric, algebraic, graphical or other tasks, *Mathematica* is the only fully integrated environment for technical computing. It can handle all the various aspects of technical computing in a coherent and unified way<sup>1</sup>.

## 2. Estimated Model

ANNs comprise a wide range of models, but they are all composed of interconnected artificial neurons or nodes. The strength of the connections between nodes is controlled by weights, which are analogous to beta coefficients in a regression model. These connection weights are the unknown parameters of ANN models. The goal of using ANNs is to find estimates of connection weights so that the functional relationship among nodes can be approximated with a desired level of precision. To date, the most frequently used ANN model is the so-called *multi-layer perceptron* (Rumelhart & McClelland, 1986) where the nodes are arranged in a layered architecture. Typically, this type of ANNs consists of three or more layers: one input layer that contains nodes representing independent variables, one output layer that contains nodes representing dependent variables, and at least one hidden layer. Data are processed from the input layer, through the hidden layer, to the output layer. The idea is to predict the values of output nodes given the values of input nodes. The user enters a set of input nodes for a given examinee—in the context of diagnostic testing, the inputs would be the scored item response string for that examinee. The output nodes are the probabilities that the student possesses each attribute.

Consider a three-layer ANN with  $I$  input nodes,  $J$  hidden nodes, and  $K$  output nodes. Given that the values of  $I$  input nodes are known, the value of the  $j$ th hidden nodes can be calculated in two steps. In the first step, a weighted sum of all input nodes is calculated as

$$a_j = \sum_{i=1}^I W_{ji} X_{iV}$$

---

<sup>1</sup> SAS macros to perform the estimation tasks required for a AHM diagnostic analysis is currently under development.

where  $a_j$  is the weighted sum for hidden node  $j$ ;  $W_{ji}$  is the connection weight from input node  $i$  to hidden node  $j$ ;  $X_i$  is the value of input node  $i$ . In the second step, the calculated sum is transformed by an activation function  $f(\cdot)$  to give the value of the hidden node as

$$h_j = f(a_j) = f\left(\sum_{i=1}^I W_{ji} X_i\right).$$

A commonly used continuous activation function is the sigmoid or logistic function, specified as

$$f(x) = \frac{1}{1 + \exp^{-x}}.$$

A benefit of using the logistic function is that its values range from 0 to 1, which allows for a probabilistic interpretation of the values taken by the nodes (McClelland, 1998). Once the values of the hidden nodes are obtained, they can be used to calculate the values of output nodes in a similar manner but with a new set of connection weights representing strength of connections between hidden and output nodes. That is,

$$Y_k = f(b_k) = f\left(\sum_{j=1}^J V_{kj} h_j\right),$$

where  $Y_k$  is the estimated value of the  $k$ th output node;  $b_k$  is the weighted sum of hidden nodes for the  $k$ th output node;  $f(\cdot)$  is the activation function;  $V_{kj}$  is the connection weight from hidden node  $j$  to output node  $k$ ;  $h_j$  is the value of hidden node  $j$ . The independent variables represented by the input nodes are related to the dependent variables by two successive transformations: one from input layer to hidden layer and one from hidden layer to output layer. To predict the values of the output nodes for the ANN, the unknown parameters that must be estimated are two sets of connections weights, one linking input nodes to hidden nodes and the other linking hidden nodes to output nodes. To achieve the best prediction, it is natural to minimize the errors between the actual and predicted values of the dependent variables. The error function that summarizes the root mean square of errors over all dependent variables and over all data points is given by

$$E = \sqrt{\frac{\sum_N \sum_K (Y_{kn} - \hat{Y}_{kn})^2}{NK}},$$

where  $\hat{Y}_{kn}$  and  $Y_{kn}$  are the predicted and actual value of dependent variable  $k$  ( $k = 1, 2, \dots, K$ ) for the  $n$ th data point ( $n = 1, 2, \dots, N$ ), respectively. Values of connection weights that minimize this error function will be chosen as the estimates of model parameters.

Typically, the process of error function minimization begins with random assignment of the initial connection weights. The output values can be computed using these initial weights together with the values of input nodes from the independent variables. The calculated output values are then compared to the observed values from the dependent variables, and the connection weights are modified accordingly to make the value of the error function as low as possible. The most studied and used method for adjusting connection weights in ANNs is the *back propagation algorithm* (Rumelhart, Hinton, & Williams, 1986) in which each connection weight is modified by an increment that is proportional to the negative gradient of the error function. The *gradient* of a function  $f(\cdot)$  with respect to  $\vec{x} = (x_1, x_2, \dots, x_n)$ , denoted by  $\nabla(f)$ , is defined as a vector whose components are the partial derivatives of  $f$ . That is,

$$\nabla(f) = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$

$\nabla(f)$  has this important feature: If  $\vec{x}$  goes in the direction of the negative  $\nabla(f)$ , then the function  $f$  decreases most rapidly. This feature forms the basis of the back propagation algorithm. The back propagation algorithm is an iterative procedure designed to find optimal connection weights that best describe the relationship between input and output nodes. With each iteration, a data point consisting of values of independent and dependent variables is randomly selected and presented to the ANN. The result for the output nodes are then compared with the observed values from the corresponding dependent variables. Each connection weight is modified using the increment

$$\Delta w_{jk} = -\gamma \frac{\partial E}{\partial w_{jk}}$$

where  $\gamma$  represents a *learning rate* parameter that defines the step length of each iteration in the negative gradient direction, and  $\frac{\partial E}{\partial w_{jk}}$  is the partial derivative of the error function with respect to the connection weight  $w_{jk}$ . Once the weights have been adjusted, another data point is selected and presented to the ANN, and a new iteration starts. To update the connection weights, the error function is evaluated for each data point. The process is repeated until the error term has converged to an acceptable level around zero or the maximum number of iterations is reached.

### 3. Technical Features

Class of Features	Feature	Implementation
Basic Characteristics	Required primary software	Mathematica <sup>1</sup>
	Fee for software	\$ 2495.00
	Programming language used	Mathematica
	GUI for input	Yes
	If not GUI, specify input format	---
	GUI for output	Yes
	If not GUI, specify output format	---
	Estimation approach	Least square estimation
	Ability to modify estimation parameters	Yes
Input Characteristics	Number of Response Variables	Unlimited <sup>2</sup>
	Scale Types	Dichotomous
Model Structures	File formats	Various
	Compensatory DCMs	No
	Non-compensatory DCMs	Yes
	Different DCMs for different items	No
	Number of Latent Variables	Unlimited <sup>2</sup>
	Scale Types	Continouous
	Maximum Number of Scale Points	Unlimited <sup>2</sup>
	Structural modeling of attribute space	Yes
Estimates	Models for attribute space	Hierarchical
	Item parameters	No
	SEs for item parameters	No

	Other item parameter estimates	No
	Person parameters	Yes
	SEs for person parameters	No
	Other person parameter estimates	Yes
	Classification reliability	Yes
	Information indices	No
	Traditional CTT statistics for subscales	Yes
Fit Indices	Item fit	No
	Person fit	Yes
	Absolute model fit	Yes
	Relative model fit	No
	Q-matrix misspecification	No
Complex Sampling Designs	Data missing by design / Multiple forms	No
	Data missing at random	No
	Multi-group estimation	No
	Inclusion of sampling weights	Yes
	Inclusion of stratification variables / Covariates	No
	Hierarchical modelling	No

**Note.** 1. Mathematica is required for execute the AHM functions. But the AHM functions are available for free on our website. 2. Unlimited means theoretically unlimited. Practically, the estimation is limited by memory capacities of the machine on which the software is installed as well as realistic running times that are acceptable to practitioners.

#### 4. Interface Characteristics

##### *a. Sample Input*

```

BPN V5 Mod date: Jan 14, 2008
Run ID code= four hidden nodes
Run date-time stamp= 20080619092004
Input cells= 15
Hidden cells= 4
Output cells= 7
Exemplar order=random
Starting exemplar span= 16
Learning Parameter 1 = 1.
Learning Parameter 2 = 1.
Starting epochs= 500
Max error data to save in RAM= 500
Max response data to save in RAM= 500
Epoch data to files= n
Random seed source= 0
Random seed: new
Random number range= {-0.1,0.1}

```

## Sample Output

Starting Hidden Cell Weights(i,h)

	HidC1	HidC2	HidC3	HidC4
I1	-0.0249493	0.0386664	0.0483568	0.0767485
I2	-0.0973465	0.00892967	-0.014257	-0.0229659
I3	-0.0130159	-0.0326831	0.0071017	0.0400872
I4	0.0917066	0.0588454	-0.096039	-0.0974866
I5	0.0041111	0.0959758	0.0517164	-0.0795325
I6	-0.0620346	-0.0550339	-0.047688	0.0421316
I7	0.0865457	0.0589345	0.0263245	0.0542112
I8	0.000132952	-0.0781133	-0.0970001	0.0367092
I9	0.0587025	0.0775826	-0.0422338	0.00215189
I10	0.0143546	0.0699153	0.0136456	0.0937748
I11	0.0966404	0.056281	-0.0826052	-0.0315319
I12	-0.066569	0.0349025	0.035683	-0.0703925
I13	0.0807772	0.085876	-0.00107923	-0.00180911
I14	-0.0990389	-0.0341958	0.0176698	-0.0579417
I15	-0.00604977	0.0183156	0.0724288	-0.0838439

Starting Output Cell Weights (h,o)

	OutC1	OutC2	OutC3	OutC4
H1	0.00328298	-0.0141119	0.0134594	0.0109459
H2	-0.00480901	0.059508	0.0827596	0.0622921
H3	0.0842921	-0.0146286	-0.0916627	-0.0337943
H4	0.00251052	0.0848408	-0.0552464	-0.00945758

  

	OutC5	OutC6	OutC7
H1	0.0742367	0.0446292	-0.0586538
H2	-0.0398598	0.0619493	-0.0332417
H3	-0.0275691	-0.0696274	-0.0539195
H4	-0.0779257	-0.0940818	0.00340765

Exemplar replications: {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}

Starting Simulation at: {2008,6,19,9,20,53.7656250}

Exemplar Presentation Frequency=

{518,488,498,549,469,534,465,507,497,506,506,471,495,493,496,508}

Last Response to each Exemplar

	Out1	Trgt1	Out2	Trgt2	Out3	Trgt3
Ex1	0.243	0.	0.111	0.	0.001	0.
Ex2	0.867	1.	0.112	0.	0.	0.
Ex3	0.993	1.	0.889	1.	0.073	0.
Ex4	0.993	1.	0.96	1.	0.92	1.
Ex5	0.969	1.	0.941	1.	0.079	0.
Ex6	0.999	1.	0.997	1.	0.964	1.
Ex7	0.899	1.	0.943	1.	0.115	0.
Ex8	0.977	1.	0.993	1.	0.91	1.
Ex9	0.985	1.	0.966	1.	0.038	0.
Ex10	1.	1.	0.999	1.	0.971	1.
Ex11	0.909	1.	0.972	1.	0.068	0.
Ex12	0.987	1.	0.998	1.	0.904	1.
Ex13	0.993	1.	0.978	1.	0.018	0.
Ex14	1.	1.	1.	1.	0.97	1.
Ex15	0.927	1.	0.985	1.	0.047	0.
Ex16	0.997	1.	0.999	1.	0.928	1.

	Out4	Trgt4	Out5	Trgt5	Out6	Trgt6
Ex1	0.156	0.	0.112	0.	0.035	0.
Ex2	0.002	0.	0.	0.	0.001	0.
Ex3	0.058	0.	0.	0.	0.003	0.
Ex4	0.089	0.	0.003	0.	0.001	0.
Ex5	0.877	1.	0.076	0.	0.087	0.
Ex6	0.933	1.	0.039	0.	0.045	0.
Ex7	0.979	1.	0.872	1.	0.064	0.
Ex8	0.993	1.	0.971	1.	0.019	0.
Ex9	0.969	1.	0.022	0.	0.9	1.
Ex10	0.993	1.	0.016	0.	0.914	1.
Ex11	0.999	1.	0.961	1.	0.926	1.
Ex12	1.	1.	0.993	1.	0.932	1.
Ex13	0.989	1.	0.004	0.	0.998	1.
Ex14	1.	1.	0.04	0.	0.999	1.
Ex15	1.	1.	0.981	1.	0.999	1.
Ex16	1.	1.	0.965	1.	0.999	1.

	Out7	Trgt7
Ex1	0.003	0.
Ex2	0.	0.
Ex3	0.	0.
Ex4	0.	0.
Ex5	0.	0.
Ex6	0.	0.
Ex7	0.	0.
Ex8	0.	0.
Ex9	0.088	0.
Ex10	0.066	0.
Ex11	0.075	0.
Ex12	0.051	0.
Ex13	0.923	1.
Ex14	0.932	1.
Ex15	0.92	1.
Ex16	0.927	1.

Total epochs= 500

Last epoch RMS error=

{0.00640314,0.00302358,0.00407852,0.00356659,0.00265661,0.00265732,0.00266588}

Mean RMS error= 0.00357881

End of simulation at: {2008,6,19,9,20,59.8750000}

Hidden Cell weights (i,h)

	HidC1	HidC2	HidC3	HidC4
I1	-3.271	1.891	2.871	1.221
I2	0.146	1.802	0.409	-1.616
I3	-1.699	-2.889	1.158	-3.657
I4	3.457	2.871	-2.308	-0.239
I5	0.464	1.218	-0.65	-1.97
I6	3.902	-0.097	0.256	-0.301
I7	0.92	0.26	0.529	-1.034
I8	-0.874	1.462	-1.611	0.267
I9	-1.25	0.607	-0.382	-0.7
I10	0.881	0.4	-0.232	-0.126
I11	0.127	0.184	-0.518	-0.449
I12	-1.336	0.545	-4.34	0.261
I13	-0.961	0.206	-1.058	-0.662
I14	0.098	0.106	-1.376	-0.16
I15	-0.015	0.042	-0.351	-0.335

Hidden Cell response (ex,h)

	HidC1	HidC2	HidC3	HidC4
Ex1	0.5	0.5	0.5	0.5
Ex2	0.037	0.869	0.946	0.772
Ex3	0.042	0.976	0.964	0.402
Ex4	0.008	0.691	0.988	0.017
Ex5	0.582	0.999	0.726	0.347
Ex6	0.288	0.993	0.815	0.002
Ex7	0.986	0.998	0.773	0.282
Ex8	0.981	0.994	0.906	0.001
Ex9	0.368	1.	0.346	0.409
Ex10	0.046	0.999	0.375	0.001
Ex11	0.986	1.	0.351	0.311
Ex12	0.943	1.	0.383	0.
Ex13	0.133	1.	0.007	0.474
Ex14	0.005	1.	0.003	0.001
Ex15	0.953	1.	0.002	0.333
Ex16	0.643	1.	0.	0.

## 5. Current Research Directions

SAS macros for estimating the attribute hierarchy method and implementing the ANN approach are currently under development.

## 6. Contact Information

The AHM functions for diagnostic modelling can be downloaded free of charge from the CRAME website at <http://www.education.ualberta.ca/educ/psych/crame/research.html>.

Please also see the following links for more information about the AHM and its application:  
[http://en.wikipedia.org/wiki/Attribute\\_Hierarchy\\_Method\\_\(AHM\)](http://en.wikipedia.org/wiki/Attribute_Hierarchy_Method_(AHM))  
<http://escholarship.bc.edu/cgi/viewcontent.cgi?article=1109&context=jtla>.

## 7. References

- Cui, Y., & Gierl, M. J. (2009). Pattern recognition techniques for cognitive diagnostic assessment scoring and reporting: An artificial neural network approach. Manuscript submitted for publication.
- Gierl, M. J., Cui, Y., & Hunka, S. (in press). Using connectionist models to evaluate examinees' response patterns on tests. *Journal of Modern Applied Statistical Methods*.
- McClelland, J. L. (1998). Connectionist models and Bayesian inference. In M. Oaksford & N. Chater (Eds.), *Rational Models of Cognition*. Oxford: Oxford University Press. 21-53.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533-536.
- Rumelhart, D. E., & McClelland, J. L. (Eds.). (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Vol. 1. Foundations*. Cambridge, MA: MIT Press.