Boosted Decision Tree for Q-matrix Refinement

Peng Xu
Polytechnique Montreal
peng.xu@polymtl.ca

Michel C. Desmarais
Polytechnique Montreal
michel.desmarais@polymtl.ca

ABSTRACT

In recent years, substantial improvements were obtained in the effectiveness of data driven algorithms to validate the mapping of items to skills, or the Q-matrix. In the current study we use ensemble algorithms on top of existing Q-matrix refinement algorithms to improve their performance. We combine the boosting technique with a decision tree. The results show that the improvements from both the decision tree and Adaboost combined are better than the decision tree alone and yield substantial gains over the best performance of individual Q-matrix refinement algorithm.

1. INTRODUCTION

A Q-matrix, as proposed by Tatsuoka (Tatsuoka, 1983), is a term commonly used in the literature of psychometrics and cognitive modeling that refers to a binary matrix which shows a correspondence between items and their latent attributes. Items can be questions or exercises proposed to students, and latent attributes are skills needed to succeed these items. Usually, a Q-matrix is defined by a domain expert. However, this task is non trivial and there might be errors, which in turn will result in erroneous diagnosis of students knowledge states (Rupp & Templin, 2008; Madison & Bradshaw, 2015). Therefore, better means to validate a Q-matrix is a highly desirable goal.

A fair number of algorithms have emerged in the last decade to validate an expert given Q-matrix based on empirical data (see for eg. recent work from Chen, Liu, Xu, & Ying, 2015; de la Torre & Chiu, 2015; Durand, Belacel, & Goutte, 2015). Desmarais, Xu, and Beheshti (2015) showed that Q-matrix refinement algorithms can be combined using an ensemble learning technique. They used a decision tree and the results show a substantial and systematic performance gain over the best algorithm, in the range of 50% error reduction for real data, even though the best algorithm is not always the same for different Q-matrices.

The encouraging the results obtained by combining the output of Q-matrix refinement algorithms leads us to pursue further along the line of using ensemble learning, or meta-learning techniques. In particular, a common approach is to use boosting with a decision tree algorithm. This is the approach explored in the current study.

2. THREE TECHNIQUES TO Q-MATRIX VALIDATION

Our approach relies on meta-learning algorithms whose principle in a general way is to combine the output of existing algorithms to improve upon the individual or average results. In our case, the approach combines a decision tree trained on the output of Q-matrix validation algorithms with boosting, a weighted sampling process in the training of the decision tree to improve its accuracy. In this section, we first describe the Q-matrix validation techniques before describing the decision tree and boosting algorithms.

2.1 minRSS

The first Q-matrix refinement technique that serves as input to the decision tree is from Chiu and Douglas (2013). We name this technique minRSS. The underlying cognitive model behind minRSS is the DINA model(see De La Torre, 2009).

For a given Q-matrix, there is a unique and ideal response pattern for a given a student skills mastery profile. That is, if there are no slip and guess factors, then the response pattern for every category of student profile is fixed. The difference between the real response pattern and the ideal response pattern represents a measure of fit for the Q-matrix, typically the Hamming distance. Chiu and Douglas (2013) considered a more refined metric. The idea is if an item has a smaller variance (or entropy), then it should be given a higher weight in measure of fit. A first step is to compute the ideal response matrix for all possible student profile, and then to find the corresponding student profiles matrix A given observed data. First, a squared sum of errors for each item k can be computed by

$$RSS_k = \sum_{i=1}^{N} (r_{ik} - \eta_{ik})^2$$

where r is the real response vector while η is the ideal response vector, and N is the number of respondents. Then, the worst fitted item (highest RSS) is chosen to update its correspondent q-vector. Given all permutations of the skills for a q-vector, the q-vector with the lowest RSS is chosen to

replace the original one. The Q-matrix is changed and the whole process repeated, but the previously changed q-vector is eliminated from the next iteration. The whole procedure terminates when the *RSS* for each item no longer changes. This method was shown by Wang and Douglas (2015) to yield good performance under different underlying conjunctive models.

2.2 maxDiff

Akin to minRSS, the maxDiff algorithm relies on the DINA model. De La Torre (2008) proposed that a correctly specified q-vector for item j should maximize the difference of probabilities of correct response between examinees who have all the required attributes and those who do not. A natural idea is to test all q-vectors to find that maximum, but that is computationally expensive. De La Torre (2008) proposed a greedy algorithm that adds skills into a q-vector sequentially. Assuming δ_{jl} represents the difference to maximize, the first step is to calculate δ_{jl} for all q-vectors which contains only one skill and the one with biggest δ_{jl} is chosen. Then, δ_{jl} is calculated for all q-vectors which contains two skills including the previously chosen one. Again the q-vector with the biggest δ_{jl} is chosen. This whole process is repeated until no addition of skills increases δ_{jl}. However, this algorithm requires knowing slip and guess parameters of the DINA model in advance. For real data, they are calculated by EM (Expectation Maximization) algorithm (De La Torre, 2009).

2.3 ALSC

ALSC (Conjunctive Alternating Least Square Factorization) is a common matrix Factorization (MF) algorithm. Desmarais and Naceur (2013) proposed to factorize student test results into a Q-matrix and a skills-student matrix with ALSC.

ALSC decomposes the results matrix $\mathbf{R}_{m\times n}$ of m items by n students as the inner product two smaller matrices:

$$\neg\mathbf{R} = \mathbf{Q}\,\neg\mathbf{S} \tag{1}$$

where $\neg\mathbf{R}$ is the negation of the results matrix (m items by n students), $\mathbf{Q}$ is the m items by k skills Q-matrix, and $\neg\mathbf{S}$ is negation of the the mastery matrix of k skills by n students (normalized for rows columns to sum to 1). By negation, we mean the 0-values are transformed to 1, and non-0-values to 0. Negation is necessary for a conjunctive Q-matrix. As such, the model of equation (1) is analogous to the DINA model without a slip and guess parameter.

The factorization consists of alternating between estimates of $\mathbf{S}$ and $\mathbf{Q}$ until convergence. Starting with the initial expert defined Q-matrix, $\mathbf{Q}_0$, a least-squares estimate of $\mathbf{S}$ is obtained:

$$\neg\hat{\mathbf{S}}_0 = \left(\mathbf{Q}_0^{\mathrm{T}}\,\mathbf{Q}_0\right)^{-1}\mathbf{Q}_0^{\mathrm{T}}\,\neg\mathbf{R} \tag{2}$$

Then, a new estimate of the Q-matrix, $\hat{\mathbf{Q}}_1$, is again obtained by the least-squares estimate:

$$\hat{\mathbf{Q}}_1 = \neg\mathbf{R}\,\neg\hat{\mathbf{S}}_0^{\mathrm{T}}\left(\neg\hat{\mathbf{S}}_0\,\neg\hat{\mathbf{S}}_0^{\mathrm{T}}\right)^{-1} \tag{3}$$

And so on until convergence. Alternating between equations (2) and (3) yields progressive refinements of the matrices $\hat{\mathbf{Q}}_i$ and $\hat{\mathbf{S}}_i$ that more closely approximate $\mathbf{R}$ in equation (1). The final $\hat{\mathbf{Q}}_\mathbf{i}$ is rounded to yield a binary matrix.

3. DECISION TREE

The three algorithms for Q-matrix refinement described in the last section are to be combined to yield with a decision tree to obtain an improved refinement recommendation, and further improved by boosting. We describe the decision tree before moving on to the boosting method.

Decision tree is a well-know technique in machine learning and it often serves as an ensemble learning algorithm to combine individual models into a more powerful model. It uses a set of feature variables (individual model predictions) to predict a single target variable (output variable). There are several decision tree algorithms, such as ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), CART (Breiman, Friedman, Stone, & Olshen, 1984). We used `rpart` function from the R package of the same name (Therneau, Atkinson, & Ripley, 2015). It implements the CART algorithm. This algorithm divides the learning process into two phases. The first phase is for feature selection, or tree growing, during which the feature variables are chosen sequentially according to *Gini impurity* (Murphy, 2012). Then in the second phase, the pruning phase, deep branches are split into wider ones to avoid overfitting.

A decision tree is a supervised learning technique and therefore requires training data. To obtain training data of sufficient size, Desmarais et al. (2015) use synthetic data from Q-matrices generated by random permutations of the perturbated Q-matrix. Since the ground-truth Q-matrix of synthetic data is known, it becomes possible to generate training data containing the class label. The training set for decision tree can take this form:

	Algorithm target prediction			Other factors
Target	minRSS	maxDiff	ALSC	...
1	1	0	1	...
0	0	1	0	...
...	...	...	...	...

The other factors considered to help the decision tree to improve prediction are the number of skills per row (SR), number of skills per column (SC). Moreover, a feature named *stickiness* is introduced and makes a critical difference. It measures the rigidity of cells under each validation methods. Stickiness represents the rate of a given algorithm's false positives for a given cell of a Q-matrix. The rate is measured by "perturbating" in turn each and every cell of the Q-matrix, and by counting the number of times the cell is a false positive. The decision tree can use the stickiness factor as an indicator of the reliability of a given Q-matrix refinement algorithm suggested value for a cell. Obviously, if a cell's stickiness value is high, the reliability of the corresponding algorithm's refinement will be lower.

4. BOOSTING

The current work extends the idea of using a decision tree with another meta-learning technique named boosting.

Boosting (Schapire & Freund, 2012) serves as a meta-learning technique for lifting a base learner. It operates on weights of the loss function terms. For a training set of N samples

and a given loss function L, the global loss is

$$Loss = \sum_{i=1}^{N} L(y_i, f(x_i))$$

Different ways of choosing loss function yield different boosting algorithm. The most famous algorithm for boosting is Adaboost (Freund & Schapire, 1997), which is especially set for binary classification problem and uses exponential loss.

In our case, the base learner is the decision tree. Adaboost trains the decision tree for several iterations, but with a different weighted training data for each iteration. That is, each time a decision tree is trained, the wrongly predicted data records in the current iteration will be assigned higher weights in the computation of the loss function for the next training iteration of the decision. The final output of Adaboost is a `sgn` function (*sign function*) of a weighted sum of all "learners" trained in the whole procedure (the decision tree with different weights vectors).

For a training set of N samples, the whole procedure for Adaboost is shown below (Murphy, 2012):

Initialize $\omega_i = 1/N$
for $i = 1$ to M **do**
  Fit the classifier $\phi_m(x)$ to the training set using weights w
  Compute $err_m = \frac{\sum_{i=1}^{N} \omega_i I(\tilde{y_i} \neq \phi_m(x_i))}{\sum_{i=1}^{N} \omega_i}$
  Compute $\alpha_m = \log[(1 - err_m)/err_m]$
  set $\omega_i \leftarrow \omega_i \exp[\alpha_m I(\tilde{y_i} \neq \phi_m(x_i)]$
end for
return $f(x) = \text{sgn}(\sum_{m=1}^{M} \alpha_m \phi_m(x))$

In which M is the number of iterations (10 in our experiment), ω_i is the weight for i-th data, $I(\cdot)$ is the indicator function, $\tilde{y_i} \in \{1, -1\}$ is the class label of training data, and $\phi_m(x)$ is the decision tree model in our case.

Boosting has had stunning empirical success (Caruana & Niculescu-Mizil, 2006). More detailed explanation and analysis of boosting can be found in Bühlmann and Hothorn (2007) and Hastie, Tibshirani, and Friedman (2009). The Adaboost algorithm was implemented in this experiment to improve the results obtained by Desmarais et al. (2015). The results are reported in section 7.

5. METHODOLOGY AND PERFORMANCE CRITERION

To estimate the ability of an algorithm to validate a Q-matrix, we perturbate a "correct" Q-matrix and verify if the algorithm is able to recover this correct matrix by identifying the cells that were perturbated while avoiding to classify unperturbated cells as perturbated. In this experiment, only one perturbation is introduced. For synthetic data, the "correct" matrix is known and is the one used in the generation of the data. For real data, we assume the expert's is the correct one, albeit it may contain errors.

Table 1: Q-matrix for validation

Name	Number of			Description
	Skills	Items	Cases	
QM1	3	11	536	Expert driven from (Henson, Templin, & Willse, 2009)
QM2	3	11	536	Expert driven from (De La Torre, 2008)
QM3	5	11	536	Expert driven from (Robitzsch, Kiefer, George, & Uenlue, 2015)
QM4	3	11	536	Data driven, SVD based

In order to use a standard performance measure, we define the following categories of correct and incorrect classifications as the number of:

- **True Positives (TP)**: perturbed cell correctly recovered
- **True Negatives (TN)**: non perturbed cell left unchanged
- **False Positives (FP)**: non perturbed cell incorrectly recovered
- **False Negatives (FN)**: perturbed cell left unchanged

We give equal weight to perturbed and unperturbed cells and use the F1-score, or F-score for short. The F-score is defined as

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

In which *precision* is calculated by the model accuracy on non-perturbated cell while *recall* is calculated by the model accuracy on perturbated cell.

6. DATASET

For the sake of comparison, we use the same datasets as the ones used in Desmarais et al. (2015). Table 1 provides the basic information and source of each dataset.

7. RESULT

The results of applying Adaboost over the decision tree (DT) are reported in table 2 for synthetic data and Table 3 for real data. The individual results of each algorithm are reported (minRSS, maxDiff, and ALSC), along with the decision tree (DT) and the boosted decision tree (BDT). Different improvement over baselines are reported as:

- **DT %Gain**: the Decision Tree (DT) improvement over the **best** of the three individual algorithm (minRSS, maxDiff, ALSC)
- **BDT %Gain**: Boosted Decision Tree improvement over the DT performance, which corresponds to the gain we get from boosting.

Let us focus on the F-Score which is the most informative since it combines results of the perturbed and non perturbed

Table 2: Results for synthetic data

QM	Individual			Ensemble	
	minRSS	maxDiff	ALSC	DT (%Gain)	BDT (%Gain)
Accuracy of perturbated cells					
1	0.809	0.465	0.825	0.946 (69.4%)	**0.951** (9.2%)
2	0.069	0.259	0.359	0.828 (73.2%)	**0.903** (43.5%)
3	0.961	0.488	0.953	**1.000** (99.7%)	**1.000** (0.0%)
4	0.903	0.489	0.853	0.956 (54.3%)	**0.971** (33.9%)
$\overline{\mathbf{X}}$	0.685	0.425	0.747	0.933 (74.2%)	**0.956** (21.7%)
Accuracy of non perturbated cells					
1	0.970	0.558	0.387	**0.990** (65.1%)	**0.990** (0.0%)
2	0.987	0.529	0.431	0.989 (20.5%)	**0.996** (59.1%)
3	0.950	0.258	0.736	0.994 (88.9%)	**1.000** (100.0%)
4	0.966	0.559	0.391	0.997 (92.2%)	**0.998** (19.2%)
$\overline{\mathbf{X}}$	0.968	0.476	0.486	0.993 (65.3%)	**0.996** (49.4%)
F-score					
1	0.882	0.507	0.527	0.968 (72.4%)	**0.970** (7.4%)
2	0.128	0.348	0.392	0.902 (83.8%)	**0.947** (46.1%)
3	0.955	0.337	0.831	0.997 (93.5%)	**1.000** (100.0%)
4	0.934	0.522	0.536	0.976 (64.0%)	**0.984** (33.6%)
$\overline{\mathbf{X}}$	0.725	0.429	0.571	0.961 (78.4%)	**0.975** (46.4%)

Table 3: Results for real data

QM	Individual			Ensemble	
	minRSS	maxDiff	ALSC	DT (%Gain)	BDT (%Gain)
Accuracy of perturbated cells					
1	0.485	0.167	0.515	**0.758** (50.0%)	**0.758** (0.0%)
2	0.345	0.093	0.564	0.618 (12.5%)	**0.764** (38.1%)
3	0.212	0.091	0.364	**0.818** (71.4%)	**0.818** (0.0%)
4	0.394	0.111	0.576	0.576 (0.0%)	**0.818** (57.1%)
$\overline{\mathbf{X}}$	0.359	0.115	0.505	0.692 (33.5%)	**0.789** (23.8%)
Accuracy of non perturbated cells					
1	0.435	**0.670**	0.418	0.606 (−19.4%)	0.606 (0.0%)
2	0.875	0.929	0.110	0.956 (37.9%)	**0.966** (21.4%)
3	0.661	**0.830**	0.219	0.785 (−26.2%)	0.752 (−15.1%)
4	0.520	**0.889**	0.148	0.546 (−308.7%)	0.658 (24.7%)
$\overline{\mathbf{X}}$	0.623	**0.829**	0.224	0.723 (−79.1%)	0.746 (8.0%)
F-score					
1	0.459	0.267	0.461	**0.673** (39.4%)	**0.673** (0.0%)
2	0.495	0.168	0.184	0.751 (50.6%)	**0.853** (40.9%)
3	0.321	0.164	0.273	**0.801** (70.7%)	0.784 (−8.7%)
4	0.448	0.198	0.235	0.560 (20.3%)	**0.730** (38.5%)
$\overline{\mathbf{X}}$	0.431	0.199	0.288	0.696 (45.25%)	**0.760** (17.8%)

cells of the Q-matrix. For synthetic data, the error reduction of boosting over the gain from the decision tree is substantially improved for all Q-matrices. The range of improvement is from 7% to 100%. For real data, two of the four Q-matrices show substantial improvements of around 40%, whereas the other two show no improvements, even a decrease of 8.7% for Q-matrix 3 which is characterized by a single skill per item. However, let us recall that we assume the expert Q-matrices are correct, which may be over optimistic. Violation of this assumption could negatively affect some of the Q-matrices scores for real data.

Note that QM3 has an inconsistent 100% gain from boosting with synthetic data compared to a small loss is obtained with real data. The value of 100% should be taken cautiously because the F-score difference is measured close to the boundary of 1 and therefore the result of only a few cases in our sample. Nevertheless, the fact that a very high F-score is obtained for synthetic data compared to real data does raise attention and might be related to the fact that it is the only single skill per item matrix.

8. DISCUSSION

This study shows that the gain obtained from combining the output of multiple Q-matrix refinement algorithms with a decision tree can be further improved with boosting. The results for synthetic data show an F-score error reduction from boosting over the DT score of close to 50% on average for all four Q-matrices, and a 18% reduction for real data. Compared with the score of the three individual refinement algorithms, minRSS, maxDiff, and ALSC, the combined ensemble learning of decision tree is very effective.

However, we find strong differences between the Q-matrices. For eg., QM2 benefits of improvements close to 50% (QM2), while QM1 has a null improvement for real data and only 7.4% for synthetic data. In that respect, the boosting does not provide a gain that is as systematic as the one obtained from the DT which is positive for all matrices.

An important advantage of the meta-learning approach outlined here is that it can apply to any combination of algorithms to validate Q-matrices. Future work could look into combining more than the three algorithms of this study, and add new algorithms that potentially outperform them. And if the current results generalize, we would expect to make supplementary gains over any of them.

Moreover, the Q-matrices used in this research are quite small in size. The performance of boosted decision tree on larger Q-matrix and larger dataset would also be of interest.

However, besides the Q-matrix-based algorithms mentioned above, there are other frameworks for knowledge tracing or domain modeling, especially when dealing with dynamic data. For example, there are Learning Factor Analysis (Cen, Koedinger, & Junker, 2006), Weighted CRP (Lindsey, Khajah, & Mozer, 2014), HMM-based Bayesian Knowledge Tracing (Corbett & Anderson, 1994; Lindsey et al., 2014) and other HMM-based models (González-Brenes, 2015). Comparison with these frameworks are also left to future work.

References

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Bühlmann, P., & Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, 477–505.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on machine learning* (pp. 161–168).

Cen, H., Koedinger, K., & Junker, B. (2006). Learning factors analysis–a general method for cognitive model evaluation and improvement. In *Intelligent tutoring systems* (pp. 164–175).

Chen, Y., Liu, J., Xu, G., & Ying, Z. (2015). Statistical analysis of q-matrix based diagnostic classification models. *Journal of the American Statistical Association*, *110*(510), 850–866.

Chiu, C.-Y., & Douglas, J. (2013). A nonparametric approach to cognitive diagnosis by proximity to ideal response patterns. *Journal of Classification*, *30*(2), 225–250.

Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, *4*(4), 253–278.

De La Torre, J. (2008). An empirically based method of q-matrix validation for the dina model: Development and applications. *Journal of educational measurement*, *45*(4), 343–362.

De La Torre, J. (2009). Dina model and parameter estimation: A didactic. *Journal of Educational and Behavioral Statistics*, *34*(1), 115–130.

de la Torre, J., & Chiu, C.-Y. (2015). A general method of empirical q-matrix validation. *Psychometrika*, 1–21.

Desmarais, M. C., & Naceur, R. (2013). A matrix factorization method for mapping items to skills and for enhancing expert-based q-matrices. In *Artificial intelligence in education* (pp. 441–450).

Desmarais, M. C., Xu, P., & Beheshti, B. (2015). Combining techniques to refine item to skills q-matrices with a partition tree. In *Educational data mining 2015*.

Durand, G., Belacel, N., & Goutte, C. (2015). Evaluation of expert-based q-matrices predictive quality in matrix factorization models. In *Design for teaching and learning in a networked world* (pp. 56–69). Springer.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119–139.

González-Brenes, J. P. (2015). Modeling skill acquisition over time with sequence and topic modeling. In *Aistats*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. Springer.

Henson, R. A., Templin, J. L., & Willse, J. T. (2009). Defining a family of cognitive diagnosis models using log-linear models with latent variables. *Psychometrika*, *74*(2), 191–210.

Lindsey, R. V., Khajah, M., & Mozer, M. C. (2014). Automatic discovery of cognitive skills to improve the prediction of student learning. In *Advances in neural information processing systems* (pp. 1386–1394).

Madison, M. J., & Bradshaw, L. P. (2015). The effects of q-matrix design on classification accuracy in the log-linear cognitive diagnosis model. *Educational and Psychological Measurement*, *75*(3), 491–511.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Quinlan, J. R. (1993). *C4. 5: Programs for machine learning* (Vol. 1). Morgan Kaufmann.

Robitzsch, A., Kiefer, T., George, A. C., & Uenlue, A. (2015). Cdm: Cognitive diagnosis modeling [Computer software manual]. Retrieved from `http://CRAN.R-project.org/package=CDM` (R package version 4.5-0)

Rupp, A. A., & Templin, J. (2008). The effects of q-matrix misspecification on parameter estimates and classification accuracy in the dina model. *Educational and Psychological Measurement*, *68*(1), 78–96.

Schapire, R. E., & Freund, Y. (2012). *Boosting: Foundations and algorithms*. MIT press.

Tatsuoka, K. K. (1983). Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of educational measurement*, *20*(4), 345–354.

Therneau, T. M., Atkinson, B., & Ripley, B. (2015). rpart: Recursive partitioning. r package version 4.1-10. *Computer software program retrieved from http://CRAN. R-project. org/package= rpart*.

Wang, S., & Douglas, J. (2015). Consistency of nonparametric classification in cognitive diagnosis. *Psychometrika*, *80*(1), 85–100.