Rapid #: -19807362

CROSS REF ID:	**1862818**
LENDER:	**SINTU :: NIE Library**
BORROWER:	**AZS :: Main Library**
TYPE:	Article CC:CCL
JOURNAL TITLE:	Journal of applied measurement
USER JOURNAL TITLE:	Journal of applied measurement.
ARTICLE TITLE:	Using the Rasch Model and k-Nearest Neighbors Algorithm for Response Classification
ARTICLE AUTHOR:	Paolino J.-P.
VOLUME:	17
ISSUE:	2
MONTH:	
YEAR:	2016
PAGES:	NA
ISSN:	1529-7713
OCLC #:	43888528

Processed by RapidX: 11/3/2022 8:01:10 PM

This material may be protected by copyright law (Title 17 U.S. Code)

6 **Rapid #: -19807362** **Alma odyssey.rapid.exlibrisgroup.com**

Status	**Rapid Code**	**Branch Name**	**Start Date**
New	AZS	Main Library	11/04/2022 05:37 AM
Pending	SINTU	NIE Library	11/04/2022 05:37 AM
Batch Not Printed	SINTU	NIE Library	11/04/2022 08:28 AM

CALL #:	**QA465 JAM**
LOCATION:	**SINTU :: NIE Library :: NIE NIEBJOURN**
REQUEST TYPE:	Article CC:CCL
JOURNAL TITLE:	Journal of applied measurement
USER JOURNAL TITLE:	Journal of applied measurement.
SINTU CATALOG TITLE:	Journal of applied measurement
ARTICLE TITLE:	Using the Rasch Model and k-Nearest Neighbors Algorithm for Response Classification
ARTICLE AUTHOR:	Paolino J.-P.
VOLUME:	17
ISSUE:	2
MONTH:	
YEAR:	2016
PAGES:	NA
ISSN:	1529-7713
OCLC #:	43888528
CROSS REFERENCE ID:	[TN:1862818][ODYSSEY:34.208.230.79/ILL]
VERIFIED:	

BORROWER: **AZS :: Main Library**

ExLibris RapidILL

This material may be protected by copyright law (Title 17 U.S. Code)
11/4/2022 8:28:06 AM

JOURNAL OF APPLIED MEASUREMENT, *17*(2), 185-192
Copyright© 2016

Using the Rasch Model and *k*-Nearest Neighbors Algorithm for Response Classification

Jon-Paul Paolino
Mercy College

In this paper we propose using the *k*-nearest neighbors (*k*-NN) algorithm (Cover and Hart, 1967) for classifying and predicting the responses to dichotomous items. We show using the percent correct statistic how *k*-NN can be used with Rasch model parameter estimation methods such as joint maximum likelihood (JMLE), conditional maximum likelihood estimation (CMLE), marginal maximum likelihood estimation (MMLE), and marginal Bayes modal estimation (MBME). We further suggest how one can use the algorithm to predict responses on future assessments. The empirical data set that we used to illustrate this procedure was the fraction subtraction data set from Tatsuoka (1984). Using R software we show the accuracy and efficacy of *k*-NN for classifying responses.

Requests for reprints should be sent to Jon-Paul Paolino, 63 Cornwells Beach Road, Port Washington, NY 11050, USA, e-mail: jonpaulpaolino@gmail.com.

Introduction

The Rasch model is a latent trait model used for analyzing educational testing data sets, like those obtained from the SAT and the GRE. Rasch models attempt to estimate latent parameters of persons and items. These models are primarily employed in educational applications, but due to their increasing popularity are used in other academic disciplines such as social sciences (e.g., Spergel and Curry, 2005) and public health (e.g., Shea, Tennant, and Pallant, 2009). In this study we use the *k*-NN algorithm to classify and predict responses to scored dichotomous items using Rasch model parameter estimates.

Overview of dichotomous items and the Rasch model

A scored dichotomous item in an educational assessment takes on a value of zero for an incorrect response and a one for a correct response. In this study we assume an assessment consisting of *I* dichotomously scored items is given to a group of *N* examinees, so that an *N* by *I* response matrix of zeros and ones can be constructed.

The Rasch model can be written as:

$$\Pr(Y_{ni} = 1) = \frac{e^{\beta_n - \delta_i}}{1 + e^{\beta_n - \delta_i}}, \qquad (1)$$

where n is the person index ($n = 1,2,\ldots,N$), i is the item index ($i = 1,2,\ldots,I$), and Y_{ni} is the scored response of subject n to item i. β_n is defined as the person ability parameter and δ_i is defined as the item difficulty parameter.

Common estimation techniques of the Rasch model

Joint maximum likelihood estimation (JMLE) (Wright and Panchapakesan, 1969) treats the N x I item responses as the observational units. The difficulty and ability estimates are fixed effects and the procedure yields estimates for both simultaneously. However, this procedure yields inconsistent estimates due to the nuisance parameter β_n. A solution to the nuisance parameter problem is conditional maximum likelihood estimation (CMLE) Andersen (1970), which is specific only to the Rasch model. CMLE produces unbiased and consistent estimates of the item difficulties by modeling the probabilities of a particular item response pattern conditional on the total score of each individual. Marginal maximum likelihood estimation (MMLE) treats the *N* individuals as the observational units and assumes that they are random effects sampled from a mixing distribution (Bock and Aitkin, 1981). The mixing distribution describes how the ability is distributed in the population. Together the Rasch model and the mixing distribution allow for the calculation of the marginal probability of a particular response pattern. Recently, significant research has been done using Bayesian approaches to Rasch modeling. Swaminathan and Gifford (1982) develop a Bayesian approach to fitting the Rasch model and found estimates to be more accurate than MMLE.

Overview of the k-nearest neighbors algorithm

The *k*-NN algorithm is a non-parametric memory based model used for classification and regression. It is a machine learning algorithm that classifies or predicts unlabeled testing data points based on their similarity with examples from a set of training data points. Every data point consists of a set of independent variables (also known as features) and a dependent outcome.

In *k*-NN classification, the dependent variable output is a category membership. A testing response is classified into a category by a majority vote of its *k*-nearest neighbors. In *k*-NN regression, the dependent variable is a continuous measure. The predicted value is the average of the dependent variable values of its *k*-nearest neighbors.

Visual illustration of the k-NN algorithm for classification

Figure 1 is a simple example illustrating the *k*-NN algorithm for classifying an unlabeled testing point. The triangles and the squares represent different response classes of the data points that are used in the training step and the objective is to classify a test object, shown as a circle, based on a majority vote of its *k*-nearest neighbors. The solid lined black circle indicates $k = 3$, because the three nearest observations to the test object are used in the voting process. The majority vote would lead to the test object being classified as a triangle, because there are two triangles and one square within the solid lined black circle. The broken lined black circle indicates $k = 5$, because the five nearest observations are used in the voting process. The majority vote would lead to the testing object being classified as a square, because there are two triangles and three squares within the broken lined black circle.

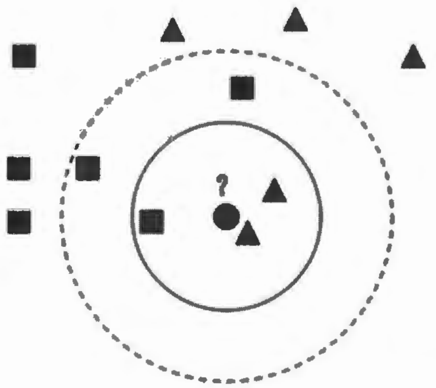

Figure 1. Example of *k*-NN algorithm for classification.

Choosing an appropriate k

Deciding how many neighbors to use for the *k*-NN algorithm determines how well the model will generalize to future data. The balance between over-fitting and under-fitting the training data is a problem known as the variance-bias tradeoff (Lantz, 2013). Choosing a large *k* reduces the variance caused by noisy data, but can bias the algorithm such that it runs the risk of ignoring small but important patterns. Suppose we took the extreme stance of setting a very large *k*, equal to the total number of observations in the training data. The model would always predict the majority class based on the training set, regardless of which neighbors are nearest. On the opposite extreme, using a single nearest neighbor allows mislabeled data or outliers to unduly influence the classification of testing data. For example, suppose that some of the training data points were accidentally mislabeled. Any unlabeled testing point that happens to be nearest to the incorrectly labeled neighbor will be predicted to the incorrect category, even if the other nearest neighbors would have voted differently.

In addition when choosing a value for *k* in classification, one should also consider the number of response categories that are present. For example if there are two categories to be predicted an even number for *k* may result in ties in the voting process, whereas choosing an odd number for *k* would avoid this problem. Tie settling procedures do exist such as breaking at random and using the closest neighbor to the testing point.

Using k-nearest neighbors to classify responses to dichotomous items

Next we describe how to use the *k*-NN algorithm to classify scored dichotomous item reponses based on parameter estimates of the Rasch model. The feature space includes two dimensions, an ability dimension represented by the horizontal axis and a difficulty dimension represented by the vertical axis as illustrated in Figure 2. We included the 45° line which indicates $\beta = \delta$ or $\Pr(Y_{ni} = 1) = .5$. Above this line ($\beta < \delta$) indicates the region of higher level of difficulty than ability and a lower probability of a correct response. Below this line ($\beta < \delta$) indicates the

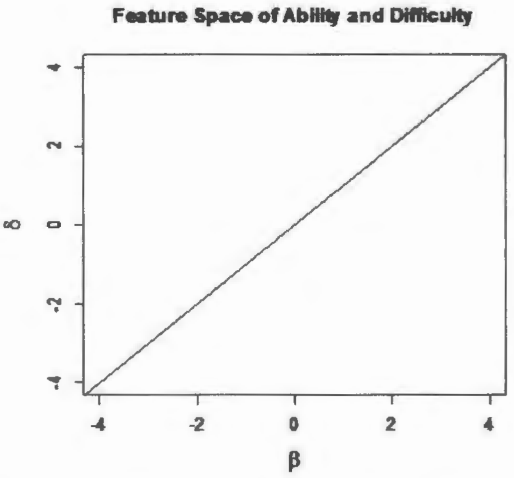

Figure 2. Feature space of the ability and difficulty.

region of lower level of difficulty than ability and a higher probability of a correct response. The 45° line is not typically incorporated in a feature space, but we included it solely for demonstration purposes.

Next to use *k*-NN, the data must be separated into a training set and a testing set. Figure 3 below shows a hypothetical feature space with the training data included. Now the feature space includes zeros indicating incorrect responses and ones indicating a correct response for each observational unit $(\hat{\beta}_n, \hat{\delta}_i)$. One would assume that the majority of the observed zeros would be above the 45° line and the majority of ones would be below the 45° line. Once the feature space of the training set has been obtained, the *k*-NN algorithm can then be used to classify responses from the testing set of parameter estimates based on the same method illustrated in Figure 1.

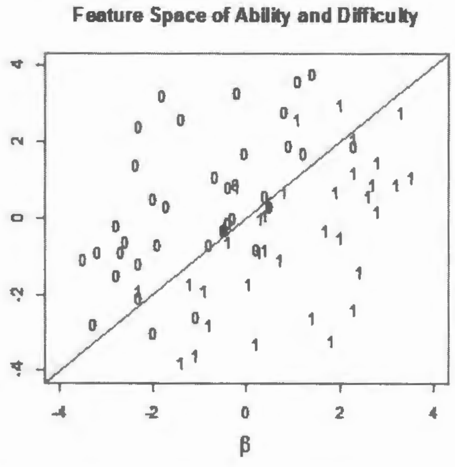

Figure 3. Feature space with training data included.

Organizing the training data and the testing data

Recall in an *N* by *I* scored response matrix for each of the *N* x *I* scored item responses there is both an ability estimate and a difficulty estimate. After the ability estimates and the difficulty estimates have been obtained the scored responses can be stacked into vector form along with each corresponding pair of person ability and item difficulty estimates. Figure 4 shows each scored response Y_{ni} combined with each person ability estimate $\hat{\beta}_n$ and each item difficulty estimate $\hat{\delta}_i$.

Using the k-NN algorithm on a single data set

The first method involves partitioning a single data set into a training set and a testing set. In this scenario $\hat{\delta}_1, \hat{\delta}_2, \ldots, \hat{\delta}_i$ and $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_n$ are all obtained first through estimation (i.e. JMLE, CMLE, or MMLE). Next the data is partitioned

$$\begin{array}{ccc}
Y & X_1 & X_2 \\
Y_{11} & \hat{\beta}_1 & \hat{\delta}_1 \\
Y_{21} & \hat{\beta}_2 & \hat{\delta}_1 \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{N1} & \hat{\beta}_N & \hat{\delta}_1 \\
Y_{12} & \hat{\beta}_1 & \hat{\delta}_2 \\
Y_{22} & \hat{\beta}_2 & \hat{\delta}_2 \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{N2} & \hat{\beta}_N & \hat{\delta}_2 \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{1I} & \hat{\beta}_1 & \hat{\delta}_I \\
Y_{2I} & \hat{\beta}_2 & \hat{\delta}_I \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{NI} & \hat{\beta}_N & \hat{\delta}_I
\end{array}$$

Figure 4. Scored responses matched with corresponding ability and difficulty estimate.

into two smaller data sets of A items used for training and B items used for testing where $A + B = I$. Now there are two smaller response matrices, the N by A training matrix and the N by B testing matrix as shown in Figure 5 and Figure 6 respectively. $\hat{\delta}_A$ represents the A^{th} item difficulty estimate in the training set and $\hat{\delta}_B$ represents the B^{th} item difficulty estimate in the testing set.

The next step is to construct a feature space of the person ability estimates and item difficulty estimates using the training data. Each point $(\hat{\beta}_N, \hat{\delta}_A)$ from the training set is plotted along with the scored response from the examinee as shown in Figure 3. The final step is to use the k-NN algorithm to predict responses to the testing data using the testing set parameter

$$\begin{array}{ccc}
Y & X_1 & X_2 \\
Y_{11} & \hat{\beta}_1 & \hat{\delta}_1 \\
Y_{21} & \hat{\beta}_2 & \hat{\delta}_1 \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{N1} & \hat{\beta}_N & \hat{\delta}_1 \\
Y_{12} & \hat{\beta}_1 & \hat{\delta}_2 \\
Y_{22} & \hat{\beta}_2 & \hat{\delta}_2 \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{N2} & \hat{\beta}_N & \hat{\delta}_2 \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{1A} & \hat{\beta}_1 & \hat{\delta}_A \\
Y_{2A} & \hat{\beta}_2 & \hat{\delta}_A \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{NA} & \hat{\beta}_N & \hat{\delta}_A
\end{array}$$

Figure 5. Data structure used in training set.

$$\begin{array}{ccc}
Y & X_1 & X_2 \\
Y_{1(A+1)} & \hat{\beta}_1 & \hat{\delta}_{A+1} \\
Y_{2(A+1)} & \hat{\beta}_2 & \hat{\delta}_{A+1} \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{N(A+1)} & \hat{\beta}_N & \hat{\delta}_{A+1} \\
Y_{1(A+2)} & \hat{\beta}_1 & \hat{\delta}_{A+2} \\
Y_{2(A+2)} & \hat{\beta}_2 & \hat{\delta}_{A+2} \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{N(A+2)} & \hat{\beta}_N & \hat{\delta}_{A+2} \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{1B} & \hat{\beta}_1 & \hat{\delta}_B \\
Y_{2B} & \hat{\beta}_2 & \hat{\delta}_B \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
Y_{NB} & \hat{\beta}_N & \hat{\delta}_B
\end{array}$$

Figure 6. Data structure used in testing set.

estimates $(\hat{\beta}_N, \hat{\delta}_B)$. Figure 7 shows the data structure comparing the actual scored responses to the predicted scored responses from *k*-NN. As previously stated, is no way to choose *k* optimally. However, one may choose a sequence of different values for *k* in the voting process.

X_1	X_2	Y	$\hat{Y}^{k-NN}$
$\hat{B}_1$	$\hat{\delta}_{A+1}$	$Y_{1(A+1)}$	$\hat{Y}^{k-NN}_{1(A+1)}$
$\hat{B}_2$	$\hat{\delta}_{A+1}$	$Y_{2(A+1)}$	$\hat{Y}^{k-NN}_{2(A+1)}$
.	.	.	.
.	.	.	.
.	.	.	.
$\hat{B}_N$	$\hat{\delta}_{A+1}$	$Y_{N(A+1)}$	$\hat{Y}^{k-NN}_{N(A+1)}$
$\hat{B}_1$	$\hat{\delta}_{A+2}$	$Y_{1(A+2)}$	$\hat{Y}^{k-NN}_{1(A+2)}$
$\hat{B}_2$	$\hat{\delta}_{A+2}$	$Y_{2(A+2)}$	$\hat{Y}^{k-NN}_{2(A+2)}$
.	.	.	.
.	.	.	.
.	.	.	.
$\hat{B}_N$	$\hat{\delta}_{A+2}$	$Y_{N(A+2)}$	$\hat{Y}^{k-NN}_{N(A+2)}$
.	.	.	.
.	.	.	.
.	.	.	.
$\hat{B}_1$	$\hat{\delta}_B$	Y_{1B}	$\hat{Y}^{k-NN}_{1B}$
$\hat{B}_2$	$\hat{\delta}_B$	Y_{2B}	$\hat{Y}^{k-NN}_{2B}$
.	.	.	.
.	.	.	.
.	.	.	.
$\hat{B}_N$	$\hat{\delta}_B$	Y_{NB}	$\hat{Y}^{k-NN}_{NB}$

Figure 7. Data Structure of the testing set along with the predicted responses from *k*-NN

Using k-NN for predicting responses on a post-test.

Another interesting scenario to consider is using the algorithm to predict individual scored responses or scored response patterns from the same set of examinees or subset of examinees on a follow-up test. This is done in a similar way as previously described. Now there are two separate assessments, one that has been completed and one that has not been completed. The assessment that has been completed serves as the training set and we use the Rasch model to obtain estimates $\hat{\delta}_1, \hat{\delta}_2, \ldots, \hat{\delta}_A$ and $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_n$. In addition, the difficulty estimates on the follow-up test $\hat{\delta}_{A+1}, \hat{\delta}_{A+2}, \ldots, \hat{\delta}_B$ must also be known. These difficulties used in the testing set can be obtained from item calibration from previous test administrations. Once both $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_n$ and $\hat{\delta}_{A+1}, \hat{\delta}_{A+2}, \ldots, \hat{\delta}_B$ are known, predictions using the training set and *k*-NN algorithm can then be obtained for the testing set. It should also be noted that *k*-NN would not need the entire sample of examinees used in the training set for the purpose of predicting individual scored responses or entire scored response patterns on the follow-up test, just a subset of the examinees would be sufficient. Once administration of the post-test is complete, one can then compare the actual scored responses to the predicted responses from *k*-NN.

Using a matching matrix to illustrate results and the percent correct statistic for evaluation

In the field of machine learning a matching matrix, also known as a contingency table, is a specific table layout that illustrates the performance of an algorithm. Here we use it to assess the accuracy of *k*-NN classification. Figure 8 below is an example of how a matching matrix can be constructed using the actual scored responses and predicted responses from the *k*-NN algorithm.

Defining our notation we denote f_{11} as the observed frequency of actual correct responses that are predicted as correct by *k*-NN, f_{10} as the observed frequency of actual correct responses that are predicted as incorrect by *k*-NN, f_{01} as the

		Predicted Response from *k*-NN	
		Score = 1	Score = 0
Actual Response	Score = 1	f_{11}	f_{10}
	Score = 0	f_{01}	f_{00}

Figure 8. Example of a matching matrix in the context of predicting dichotomous responses

observed frequency of actual incorrect responses that are predicted as correct by *k*-NN, and f_{00} as the observed frequency of actual incorrect responses that are predicted as incorrect by *k*-NN. Finally, Pr(*C*) is the overall proportion of responses correctly predicted by *k*-NN shown in Equation 2:

$$\Pr(C) = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}. \quad (2)$$

Evaluating the k-NN algorithm using a real data set

We used the fraction subtraction data set from Tatsuoka (1984) for evaluating the *k*-NN algorithm. This data set is from an exam consisting of forty dichotomously scored items administered to five-hundred and thirty-six examinees. First all examinee abilities and item difficulties were estimated using JMLE, MMLE, CMLE, and MBME. Then we formed three different experimental conditions. In Condition 1, 70% of the items (the first twenty-eight items) were used as the training set and 30% of the items (the last twelve items) were used as the testing set. In Condition 2, 80% of the items (the first thirty-two items) were used as the training set and 20% of the items (the last eight items) were used as the testing set. In Condition 3, 90% of the items (the first thirty-six items) were used as the training set and 10% of the items (the last four items) were used as the testing set. We decided to only use an odd number for *k* in the decision process to avoid voting ties. Therefore, we used values of $k = 1$, $k = 3$, $k = 5$, $k = 7$, and $k = 9$.

Using R software for Rasch model parameter estimation and k-NN classification

We chose to use R for Rasch model parameter estimation by JMLE, CMLE, MMLE, and MBME. The TAM package (Kiefer, Robitzsch, and Wu, 2015) estimates the parameters by JMLE. The eRm package (Mair, Hatzinger, and Maier, 2010) fits the Rasch model by CMLE. In the eRm package the abilities are estimated by an ad hoc procedure using weighted maximum likelihood. The ltm package (Rizopolous, 2006) uses MMLE to estimate the Rasch model parameters. An ad hoc step must be used to recover the ability estimates. We chose to use maximum a posteriori for ability estimation. Finally, we used the irtoys (Partchev, 2012) and ICL (Hanson, 2002) packages for fitting the Rasch model by MBME. The irtoys package uses the algorithm from the ltm package to obtain the marginal likelihood function and in conjunction puts a prior distribution on the difficulty parameters. The prior distribution for the difficulty parameters was $N(0, 2^2)$. Similar to the MMLE procedure we chose maximum a posteriori for ability estimation.

After the estimates have been collected we used the DMwR package (Torgo, 2010) in R to predict the testing responses from the training responses. The feature space of the training set consists of each pair of examinee ability estimate and item difficulty estimate $(\hat{\beta}_N, \hat{\delta}_A)$ along with the corresponding scored response. For example, in Condition 1 the training set would include the first twenty-eight item difficulty estimates and all five hundred and thirty-six ability estimates along with the 15008 (536 x 28) scored responses to be plotted onto the feature space. The next step is to use *k*-NN to predict scores for each point $(\hat{\beta}_N, \hat{\delta}_B)$ in the testing set. In Condition 1 the final twelve item difficulty estimates and all five hundred and thirty-six ability estimates are given a predicted score based on a majority vote of the *k* nearest neighbors from the training set. There would be 6432 (536 x 12) values to be predicted using *k*-NN in the testing set.

Results

In this section we discuss and display the results of our empirical study of the *k*-NN algorithm for classifying scored dichotomous responses. We first noticed that increasing the value of *k* improved the percent correct statistics across all four estimation conditions. This is not surprising, as previously noted using just one nearest neighbor does not guarantee best optimal prediction. As shown in the tables below there was a noticeable increase in the percent correct statistics using $k = 9$ compared to $k = 1$. We further surmise since the training set included thousands of data points using a higher value for *k* would lead to a more accurate prediction of testing data points. Finally, we also observed that Condition 2 and Condition 3 provided higher percent correct statistics compared to Condition 1 across all four estimation methods. This also seems plausible as increasing the available testing data should yield better prediction rates.

Discussion

In this paper we suggest using the *k*-NN algorithm to predict or classify scored responses on formal assessments. As we have illustrated it has the capability of producing results with high accuracy. We believe *k*-NN and other machine learning algorithms have the potential to offer novel insight into solving measurement problems and

Table 1

Resulting percent correct in Condition 1 across all estimation methods and options for k

	Number of neighbors used in voting process				
Method	$k = 1$	$k = 3$	$k = 5$	$k = 7$	$k = 9$
JMLE	0.8038	0.8391	0.8453	0.8493	0.8535
CMLE	0.8038	0.8403	0.8492	0.8495	0.8507
MMLE	0.7962	0.8333	0.8441	0.8497	0.8500
MBME	0.8038	0.8383	0.8464	0.8523	0.8524

Table 2

Resulting percent correct in Condition 2 across all estimation methods and options for k

	Number of neighbors used in voting process				
Method	$k = 1$	$k = 3$	$k = 5$	$k = 7$	$k = 9$
JMLE	0.8165	0.8498	0.8559	0.8580	0.8633
CMLE	0.8165	0.8510	0.8559	0.8580	0.8633
MMLE	0.8151	0.8454	0.8608	0.8566	0.8636
MBME	0.8165	0.8477	0.8542	0.8610	0.8622

Table 3

Resulting percent correct in Condition 3 across all estimation methods and options for k

	Number of neighbors used in voting process				
Method	$k = 1$	$k = 3$	$k = 5$	$k = 7$	$k = 9$
JMLE	0.8279	0.8512	0.8615	0.8613	0.8577
CMLE	0.8279	0.8549	0.8587	0.8605	0.8605
MMLE	0.8265	0.8488	0.8563	0.8549	0.8643
MBME	0.8278	0.8493	0.8587	0.8619	0.8615

are worthy of further consideration. For example, in this study we consider the examinee producing responses as the data generation process and use the *k*-NN algorithm as a method to predict these responses. The main hurdle to overcome is the lack of compatibility between machine learning software and standard Rasch model software. We used R for this study because it provides flexibility in data transferability. There is a substantial amount of research that can be done applying machine learning algorithms to the Rasch model. We believe the results of this study are encouraging and illuminating.

References

Andersen, E. B. (1970). Asymptotic Properties of Conditional Maximum Likelihood Estimators. *Journal of the Royal Statistical Society Series B*, *32*, 283-301.

Bock, R. D., and Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika*, *46*, 443-459.

Cover, T., and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *27*, 13-21.

Hanson, B. (2002). ICL: IRT command language. [Computer software]. www.b-a-h.com.

Kiefer, T., Robitzsch, A., and Wu, M., (2015). TAM: Test Analysis Modules R package version 0.15-1. [Computer software] http://CRAN.R-project.org/package=TAM.

Lantz, B. (2013). *Machine learning with R*. Birmingham, UK: Packt Publishing.

Mair, P., Hatzinger R., and Maier, M. (2012). eRm: Extended Rasch modeling. R package version 0.15-1. [Computer software] http://CRAN.R-project.org/package=eRm

Partchev, I. (2012). irtoys: Simple interface to the estimation and plotting of IRT models. R package version 0.1.5. [Computer software] http://CRAN.R-project.org/package=irtoys

R Development Core Team (2011). R: A language and environment for statistical computing [Computer software]. Vienna, Austria: R Foundation for Statistical Computing.

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests*. Copenhagen, Denmark: Danish Institute for Educational Research. (Expanded edition, 1980. Chicago, IL: University of Chicago Press.)

Rizopoulos, D. (2006). ltm: An R package for latent variable modeling and item response theory analyses. *Journal of Statistical Software*, *17*, 1-25.

Shea, T., Tennant, A., and Pallant, J. (2009). Rasch model analysis of the Depression, Anxiety and Stress Scales (DASS). *Journal of Biomedical Psychiatry*, *2*, 9-21.

Spergel, D., and Curry, G. (2005). Studying youth gangs: Alternative methods and conclusions. *Journal of Contemporary Criminal Justice*, *21*, 98-119.

Stehman, S. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, *1*, 77-89.

Swaminathan, H., and Gifford, J. A. (1982). Bayesian estimation in the Rasch model. *Journal of Educational Statistics*, *7*, 175-192.

Tatsuoka, K. (1984). *Analysis of errors in fraction addition and subtraction problems*. Final Report for NIE-G-81-0002, Urbana, IL: University of Illinois, Computer-based Education Research.

Torgo, L. (2010). DMwR: Data Mining with R, learning with case studies. R package version 0.15-1. [Computer software] http://CRAN.R-project.org/package=DMwR

Wright, B. D., and Panchapakesan, N. (1969). A procedure for sample free item analysis. *Educational and Psychological Measurement*, *29*, 23-48.