Sparse modeling of test scores for estimating skills acquired by students

Shohei KIKUCHI

Graduate School of Library, Information and Media Studies

University of Tsukuba

March 2019

Contents

List of Figures

Chapter 1

Introduction

One important goal of education is to help students acquire new "skills" that are useful for achieving their goals in life. In educational science, a skill refers to an elementary constituent of knowledge obtained through learning [1]. Skill refers to any knowledge, not limited to procedural ones, representing a general state of understanding often characterized by a distinguishable moment of "getting it". For example, knowing how to factor a polynomial is a skill. Understanding the laws of motion is a skill too.

To be a good instructor, one must monitor how well each student is doing in acquiring skills. Based on such monitoring, the instructor can adjust how fast they explains the material or whether they should go back to what they has just presented.

In practice, however, it is not easy to know how well each student understood, especially when there are many students in the classroom. It becomes even harder in online education, which is becoming increasingly popular in recent years. It is therefore natural that instructors try to know how well students understood through requiring them to take a test. The total score of a test reflects how well the student has understood about the whole subject. However, if the aim is to know the level of understanding at a finer scale, that is, for each skill rather than the whole subject, the instructor must explicitly assign to each problem what skills are necessary to answer it correctly. The process would require much manual work, so instructors often will not do it. Then they will not know what skills each student has acquired and what skills he has not. M oreover, it is likely to lead to lessons where students are left behind because students can not analysis for mistake of tests myself.

To confront this issue, we propose a method of automatic skill extraction from test scores. The method defines skills by grouping problems and at the same time estimating how well each student is doing in acquiring these skills. It frees instructors from tedious work of specifying what skills are needed to solve each problem.

The approach we take is similar to clustering. Problems do not need to be assigned to predefined skills manually. Instead, we define skills automatically as vectors consisting of relevant problems. This is accomplished by sparse modeling, which is a mathematical procedure that decomposes an observed matrix into a product of two matrices, one being sparse. In our case, matrix representing test scores would be decomposed into matrix that relates students to skills and two matrix that relates skills to problems. One merit of this approach is that implicit and subtle skills that were previously unknown to the instructor may be obtained in addition to obvious ones.

Chapter 2

Related work

2.1 Test-result matrix, skill-acquisition matrix, Q-matrix

The result of a test can be represented using matrix that denote it by $\boldsymbol{R}$. Each row represents a student, and each column represents a problem. An element R_{ij} represents the score student i obtained for problem j. We will call such matrix a **test-result matrix**. A **skill-acquisition matrix** $\boldsymbol{D}$ is matrix whose element D_{ih} represents how well student i has acquired skill h. For procedural knowledge, acquiring a skill can mean that one become able to conduct a certain procedure. For conceptual knowledge, acquiring a skill means grasping a specific concept.

A **Q-matrix** $\boldsymbol{Q}$ represents what skills are necessary to solve each of the problems in a test. Each row represents a skill, and each column represents a problem. In other words, Q_{hj} represents how much skill h is required to solve problem j. Some researchers assume that the elements of the Q-matrix should be binary, that is, be either 0 or 1. The value represents whether the corresponding skill is necessary to solve the problem or not. In this thesis, we define the Q-matrix to be real-valued. Each element of the Q-matrix then represents how much each skill is required for solving each problem [2, 3, 4].

2.2 Matrix factorization for skill estimation

Let $\boldsymbol{R} \in \mathbb{R}^{m \times n}$, $\boldsymbol{D} \in \mathbb{R}^{m \times k}$, and $\boldsymbol{Q} \in \mathbb{R}^{k \times n}$. An element R_{ij} of a test-result matrix $\boldsymbol{R}$ is the score that student i obtained for problem j. Our model assumes that the score R_{ij} results from the weighted sum of certain skills necessary to solve problem i.

An element D_{ih} of skill-acquisition matrix $\boldsymbol{D}$ represents how well student i did in acquiring skill h. An element Q_{hj} of a Q-matrix $\boldsymbol{Q}$ represents how important skill h is to solve problem j. The product $D_{ih}Q_{hj}$ would contribute when student i tackles problem j. Using h to represent the number of skills, the sum $\sum_{h=1}^{k} D_{ih}Q_{hj}$ is the resulting score, and it should be close to R_{ij}. In other words, $R_{ij} \approx \sum_{h=1}^{k} D_{ih}Q_{hj}$. Since this applies to all i and j, three matrices $\boldsymbol{R}$, $\boldsymbol{D}$, and $\boldsymbol{Q}$ should fulfill $\boldsymbol{R} \approx \boldsymbol{DQ}$. Therefore, skill estimation can be formalized as matrix factorization problem that finds matrices $\boldsymbol{D}$ and $\boldsymbol{Q}$ that fulfill $\boldsymbol{R} \approx \boldsymbol{DQ}$. Since $\boldsymbol{R}$ can be approximated by $\boldsymbol{D}$ and $\boldsymbol{Q}$ in multiple ways, more restrictions are added to obtain a solution that is most suitable the purpose of modeling skills. Depending on what kind of restriction is added to the elements of the factor matrices $\boldsymbol{D}$ and $\boldsymbol{Q}$, there are several ways to decompose matrix for skill estimation, as described in the

following subsections.

2.2.1 Boolean matrix factorization (BMF)

A binary matrix is matrix where all of its elements are either 0 or 1. Boolean matrix factorization (BMF) factors a binary matrix into a product of two binary matrices [5]. In other words, it decomposes $\boldsymbol{R} \in \{0,1\}^{m \times n}$ into a product of $\boldsymbol{D} \in \{0,1\}^{m \times k}$ and $\boldsymbol{Q} \in \{0,1\}^{k \times n}$. If a score given to any of the problems are neither zero nor one, BMF cannot be used. Further, BMF cannot express a state where a student has partially acquired a certain skill since the elements of the skill-acquisition matrix cannot take a fractional value, i.e. 0.1, 0.01, etc.

2.2.2 Non-negative matrix factorization (NMF)

A non-negative matrix is matrix whose elements are all non-negative. Non-negative matrix factorization (NMF) decomposes a non-negative matrix into a product of two non-negative matrices [6]. Educational data mining researchers have been using it for obtaining skills [1, 7]. However, one limitation of applying NMF for factorizing a test-result matrix is that it adds the non-negatively constraint. For example, observed data (test scores) cannot be statistically standardized, since it would result in negative values.

2.3 Sparse modeling

Sparse modeling is an approach where observed data is broken down into a superposition of a few templates.When templates are given in advance, it is called sparse coding. In the case of an image, it is thought that various images are generated by overlapping templates called base images. Sparse modeling also covers the case where templates are obtained from data, that is, in an unsupervised manner. It can be formalized as factorization into matrix of templates and a sparse matrix of weights. Its alternative name is **dictionary learning** since the matrix consisting of templates is often called a dictionary [8].

The observed datmatrix $\boldsymbol{R}$ can be considered as a series of samples, where each column of $\boldsymbol{R}$ is a sample. Sparse modeling tries to reconstruct each sample $\boldsymbol{R}_{:j}$ using a weighted sum of templates. Templates are selected from dictionary represented by matrix $\boldsymbol{D}$. For each h, the hth column $\boldsymbol{D}_{:h}$ of $\boldsymbol{D}$ is a template, and Q_{hj} is a weight given to template $\boldsymbol{D}_{:h}$ for obtaining an observed sample j.

$$\boldsymbol{R}_{:j} \approx \sum_{h=1}^{k} \boldsymbol{D}_{:h} Q_{hj} \tag{2.1}$$

The restriction is that each observed sample $\boldsymbol{R}_{:j}$ should be reconstructed using as few templates as possible. The maximum number of templates used for reconstructing each sample is indicated by s. It corresponds to the maximum number of non-zero elements allowed to appear in $\boldsymbol{Q}_{:j}$, and is a parameter that must be set in advance. In other words, $\boldsymbol{Q}$ should be a sparse matrix. Hence the name, sparse modeling.

In sparse modeling, templates are constructed solely from observed data, and no templates need to be defined in advance. It is similar to the process of clustering. Samples are grouped into clusters that naturally form based on how samples are distributed. There

is no need to define clusters in advance. Like clusters, templates are constructed based on how samples distribute.

Sparse modeling has attracted much attention in many fields including signal processing, machine learning, and data science, due to its versatile nature of modeling a wide range of phenomena [9]. In this thesis, we propose to apply sparse modeling for finding skills and their relations to test problems and students from the test-result matrix. In this application, the dictionary corresponds to the skill-acquisition matrix, and the sparse matrix corresponds to the Q-matrix.

Chapter 3

Feature analysis on students' behavior

In this section, we describe our analysis on how students behave on an e-learning system. When using e-learning system, one can predict the student's performance by directly observing the learning method that mean "skills". In addition, by analyzing the model predicting performance, you can discover learning method that contributed to your grades.

Also, predicting the final grade scores of students from how they interacted with an e-learning system is important for developing and improving system for online learning. If some functions in the system affect students' final grade scores significantly, it will be worthwhile to put more effort in extending those functions. Also, teachers can guide students to use the e-learning system in ways that would increase their expected final grade scores.

In this work, logs recorded on an e-learning system that provides online teaching material, were used to predict the final grade scores of students. The system used for logging is BookRoll [10, 11, 12]. BookRoll is a developed systems that allows to view digital materials used in lectures. It is an online environment which allows teachers to upload content as pdf file which the student can browse anytime and anywhere from web browser in their personal devices (computer or smartphone). There are features like bookmarks, markers, memo functions, which the students can use for learning. The data was provided for the 5th ICCE workshop on Learning Analytics (LA) Joint Activity on Predicting Student Performance.

3.1 Datasets

Datasets colleceted by BookRoll consists of dataset 1, which is of one intensive lecture, and dataset 2, which spans three lectures. For each of these courses, there are two main types of features, namely clickstream and scores.

The details of the file columns are 3.1 and 3.2.

Dataset 1 includes 53 students' scores and 28,826 logs operated by students. Dataset 2 includes 56 students' scores and 36,929 logs operated by students.

Table 3.1: Features related to clickstream

feature	description	example
userid	Anonymized student userid	ds1001,ds1002
action	API verb for the action	/xapi/adb/verbs/read
operationname	The action that was done	OPEN,CLOSE,NEXT,PREV
markercolor	Color of the marker added to a page	rgb(255,0,0)
processcode	Grouping of event logs by event type	
devicecode	Type of device used to view BookRoll	mobile
markerposition	The position (x,y,w,h,onscreen w,onscreen h)	367,34,28,20,714,504
description	The page the user moved to	
pageno	The current page	
contentsid	The id of the e-book that is being read	
memotext	The text contained in the memo	
eventtime	The timestamp of when the event occurred	

Table 3.2: Features related to scores

feature	description	example
userid	Anonymized student userid	ds1001,ds1002
score	The final score that received for the course	80,100

3.2 Designed features

Before training machine learning models, we carefully designed features (attributes) that would contribute to making good predictions. In addition to features already provided in the data sets, some extra features were constructed heuristically. Event time that student was just using BookRoll system was split into parts representing year, month, day, day of the week, and hour within the day. This is to see time in different time scales. By counting the number of events happened within each time scale, we obtained new attributes representing how often events happened in a certain time frame. For example, a feature named "hour_8" represents how many times the student accessed the e-learning system between 8 am till 9 am of any day. A feature named "day_5" represents how many times the student accessed in day 5 of any month. These features were added based on a hypothesis that times that students access the system correlates with how motivated or involved they are to the course. Similarly, the number of times the student accessed a certain page was turned into a feature. For example, "pageno_10" represents how many times the student accessed page 10 of the teaching material. This is based on a hypothesis that well-performing students would be checking out more relevant or difficult part of the teaching material, whereas less-performing students might be looking at less relevant or easier part of the material. The number of times the student conducted a specific process is turned into a feature too. For example, "processcode_3" represents how many times the student performed the process labeled by 3. After adding these features, we trained machine learning models. We evaluated models with different levels of complexity. Specifically, we compared Lasso, Elastic Net, multilayer perceptron regressor (MLP), kernel ridge regressor, random forest regressor, AutoSklearn (Efficient and Robust AutoML for Scikit-learn, abbreviated as AutoML)[13], and gradient boosting regressor.

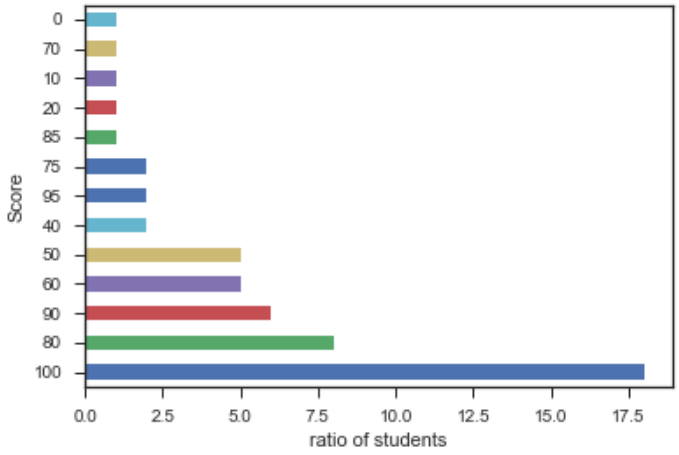

Figure 3.1: Ratio of students having specific scores, sorted in ascending order (for dataset 1)

3.3 Results

3.3.1 Dataset 1

Figures 3.1 and 3.2 illustrate how final grade scores are distributed. Most students scored between 75 and 100, indicating much deviation from a fitted Gaussian distribution. This suggests that methods that assume samples follow a Gaussian distribution may not perform well. For example, since linear regressor is derived from maximum likelihood estimation where error follows a Gaussian distribution, it may not be an appropriate method in this case.

Learning curves for different machine learning models are illustrated in Figure 3.3 - 3.9. A learning curve is a graphical representation of how an learning score (measured on the vertical axis) comes from iteration (the horizontal axis); or how the more someone (or something) performs a task, the better they get at it. Also, learning curve include "training scores" and "validation scores". "Training score" and "validation score" are calculated by machine learning model learning training data and validation data. Validation scores significantly lower than training scores indicate overfitting occurring in some methods. Since validation scores approach training scores as the number of training sample increases, it suggests that more complex models such as AutoML may perform even better as the number of samples is increased.

Distributions of RMSE (root mean squared error) for dataset 1 is indicated in Table 3.3 and Figure 3.10. They are obtained by 30 validations, resulting from performing 3-fold cross-validation 10 times.

After training the random forest, we show the top 20 features that contributed more were ranked using Gini-importance values. The result is indicated in Figure 3.11. It shows that "hour_17" feature contributed significantly, which indicates that the number of accesses that a student makes at 5 pm greatly affects how well the student performance in terms of the final grade score. The random sequence of characters ranked third in Gini-importance is an ID for a book that students could access from the e-learning system.

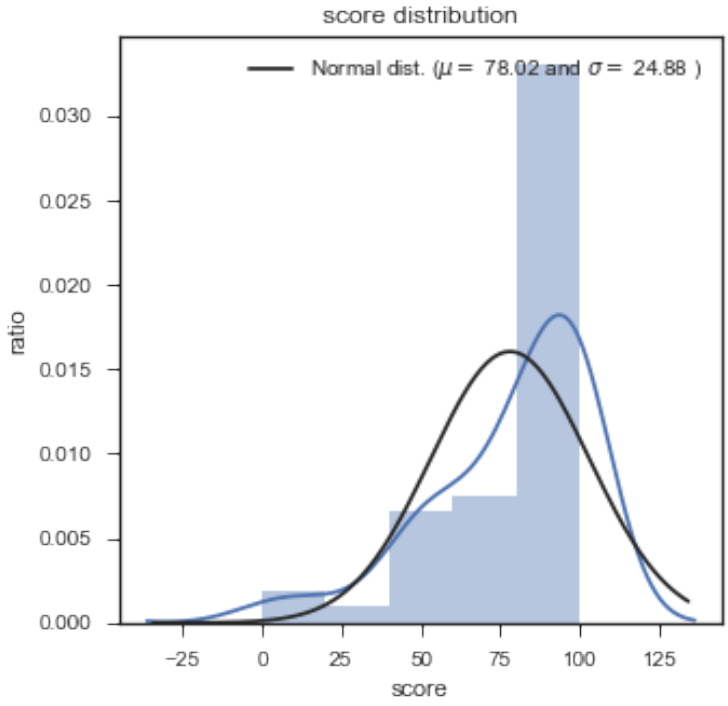

Figure 3.2: Score distribution (histogram), fitting by a Gaussian distribution (black line), and Gaussian kernel density estimate (blue line). (for dataset 1)

Table 3.3: RMSE's mean and standard deviation for methods compared by dataset 1

method	**mean**	**std**
LASSO	141.7914	200.6841
Elastic Net	137.4087	192.9349
MLP regressor	70.7624	50.5348
kernel ridge regressor	54.7972	26.6537
random forest regressor	23.4546	15.7610
AutoML	23.4572	15.4890
gradient boosting	25.7972	16.3673

3.3.2 Dataset 2

Figure 3.12 illustrates the distribution of final grade scores for dataset 2. It is less concentrated around score 100 when compared to dataset 1. However, it is still not well fitted to a Gaussian distribution, indicating simple models like linear regressor would not be appropriate.

Figure 3.13 - 3.19 illustrates learning curves for dataset 2. Like in dataset 1, validation scores are sometimes much lower than training scores in some models. It suggests that with more data, complex models may have validation scores closer to training scores, resulting in better predictions.

Distributions of RMSE (root mean squared error) for dataset 2 is indicated in Table 3.4 and Figure 3.20. They are obtained by 30 validations, resulting from performing 3-fold cross-validation 10 times.

We show the top 20 features that contributed according to random forest are indicated

Table 3.4: RMSE's mean and standard deviation for methods compared by dataset 2

method	**mean**	**std**
LASSO	1238.1671	5716.8297
Elastic Net	995.1362	4514.8347
MLP regressor	95.9370	130.1382
kernel ridge regressor	50.0973	26.7959
random forest regressor	14.9165	10.0090
AutoML	14.3660	9.4390
gradient boosting	13.3342	9.9282

in Figure 3.21. This time, the ID of a book contributed to most, suggesting referring more or less to a specific book in the e-learning system greatly affected how well the student performs in terms of the final grade score. A time feature "hour_5" was also important, suggesting accessing at a certain time of the day also affects the student's performance.

3.4 Discussion

Possibly due to the size of datasets, AutoML didn't perform any better than random forest or gradient boosting. Using more data might make these complex models more competitive. Random forest performed best for dataset 1, and gradient boosting did so for dataset 2. These are ensemble methods, showing their effectiveness with datasets of this size. The difference in the performance of two models between datasets may result from distributions of data. In order to explain it, we need to conduct further analysis on the distributions. Features that contributed most to making predictions included ones regarding times, representing at which time of the students were accessing the e-learning system. Features representing which part of the teaching material were also important. RMSE obtained for dataset 2 is as low as 13.34 where the grade ranges between 0 and 100, indicating using log data from an e-learning system can be an effective way of predicting students' performance. One must note, however, that in the datasets used for analysis, scores are mostly above 70, predictors at this moment may not be useful for predicting how well a moderately performing student would perform among others. On the other hand, it could be useful for detecting students that will perform significantly worse than others.

One interesting observation obtained from this analysis is that at what time (hour) of the day students access the e-learning system greatly contributes to predicting the final grade score of students. It may suggest that this feature represents a student's attitude and motivation toward the course. The results suggest that analyzing log data can contribute to making e-learning better.

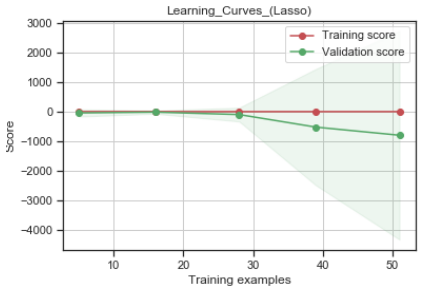

Figure 3.3: Dataset 1, Lasso

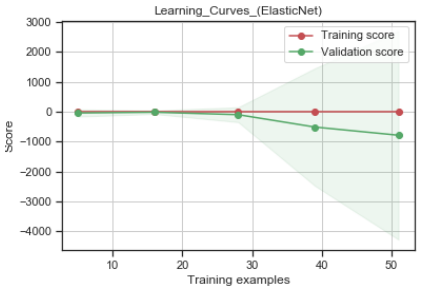

Figure 3.4: Dataset 1, ElasticNet

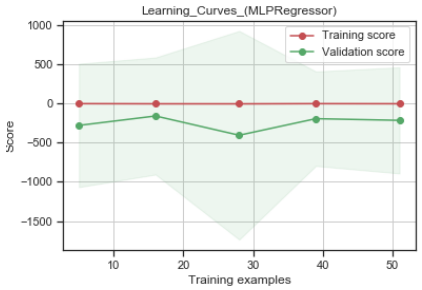

Figure 3.5: Dataset 1, MLP regressor

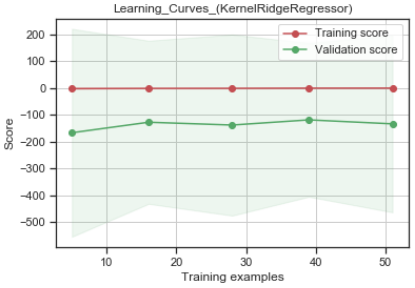

Figure 3.6: Dataset 1, kernel ridge regressor

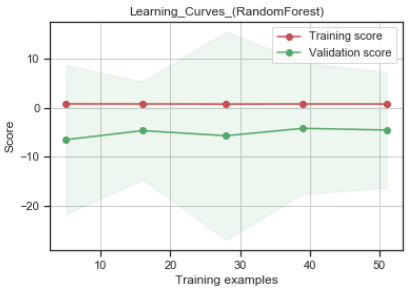

Figure 3.7: Dataset 1, random forest

Figure 3.8: Dataset 1, AutoSklearn regressor

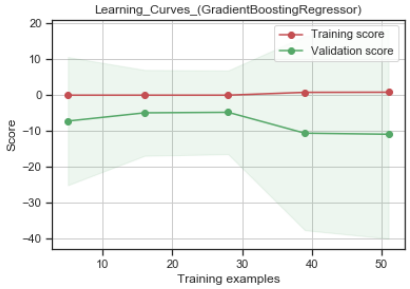

Figure 3.9: Dataset 1, gradient boosting regressor

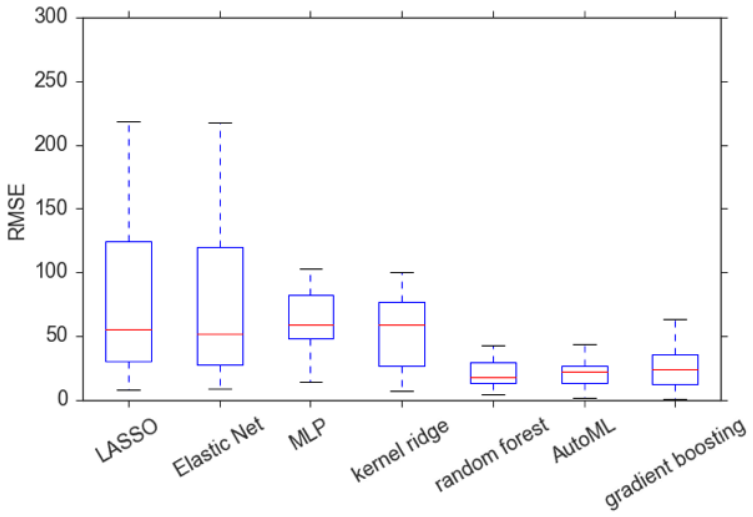

Figure 3.10: Distributions of RMSE over 3-fold cross-validation using 10 different partitions. Boxes represents the lower to upper quartile values of the data, with a red line at the median. The whiskers show the range of data (for dataset 1)

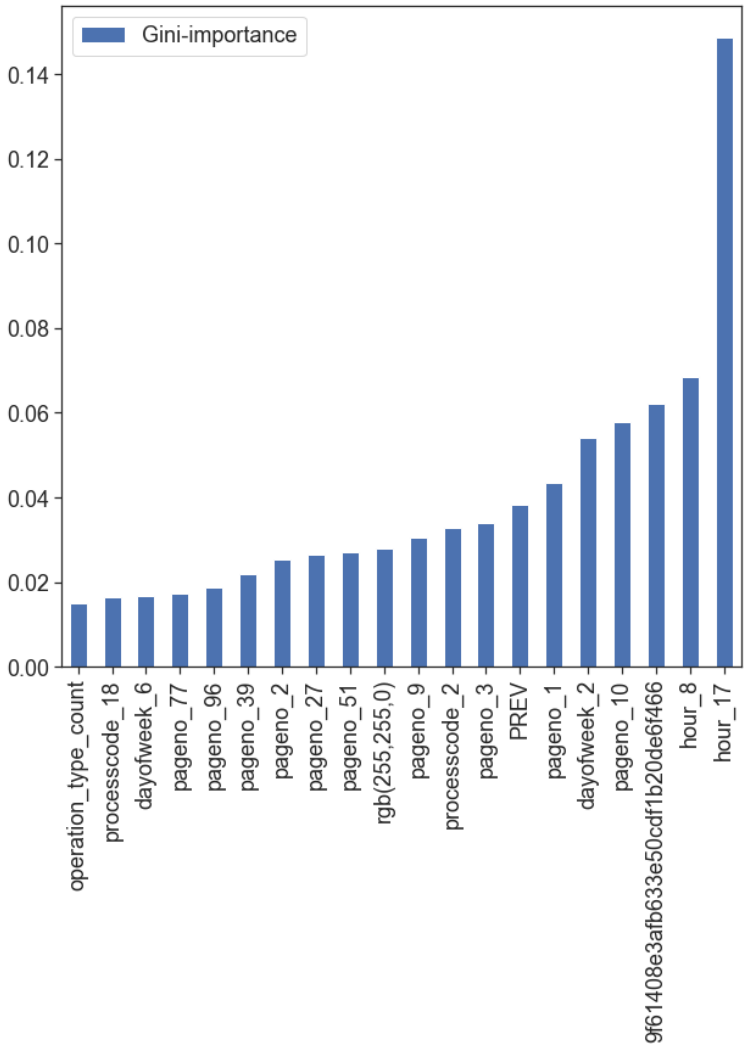

Figure 3.11: Features sorted by Gini-importance, indicating how much each feature contributed to making predictions in random forest (for dataset 1)

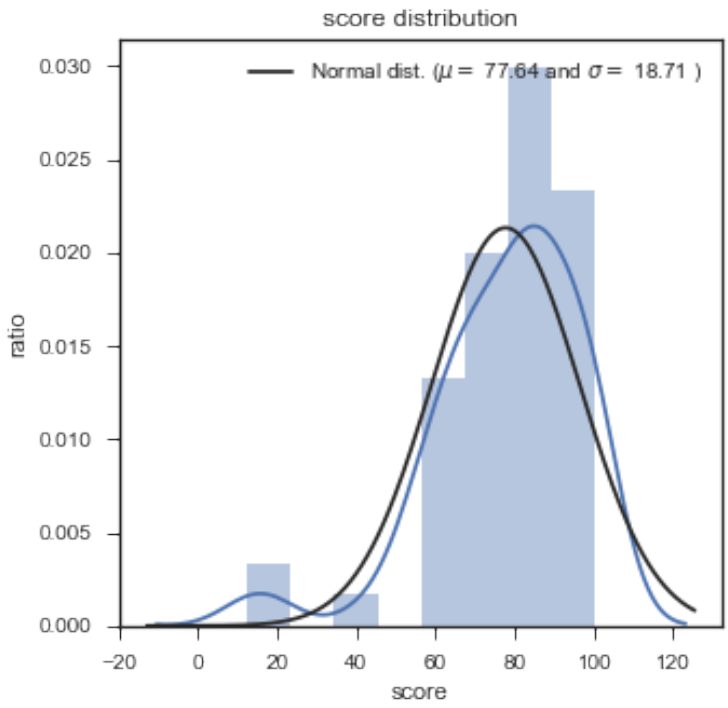

Figure 3.12: Score distribution (histogram), fitting by a Gaussian distribution (black line), and Gaussian kernel density estimate (blue line) (for dataset 2)

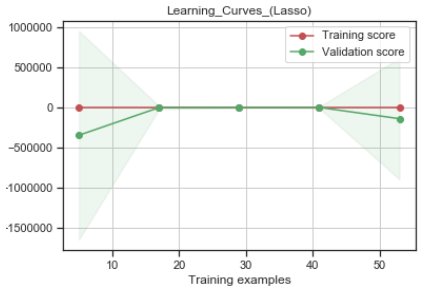

Figure 3.13: Dataset 2, Lasso

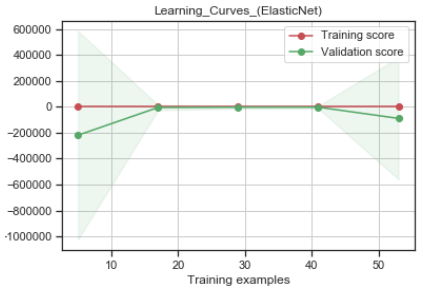

Figure 3.14: Dataset 2, ElasticNet

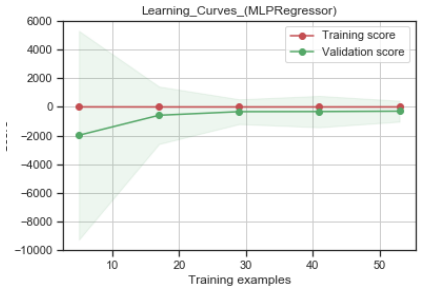

Figure 3.15: Dataset 2, MLP regressor

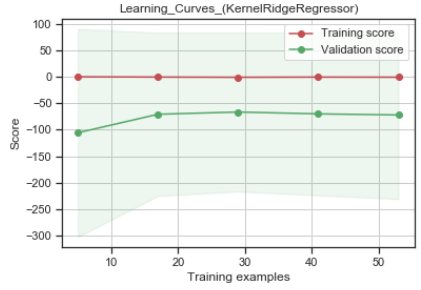

Figure 3.16: Dataset 2, kernel ridge regressor

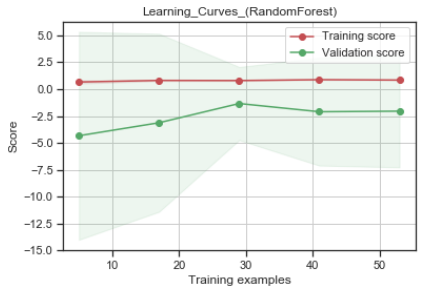

Figure 3.17: Dataset 2, random forest

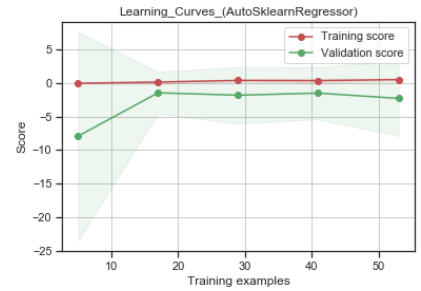

Figure 3.18: Dataset 2, AutoSklearn regressor

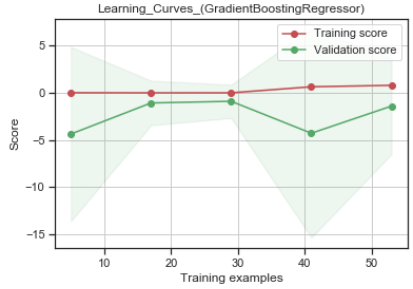

Figure 3.19: Dataset 2, gradient boosting regressor

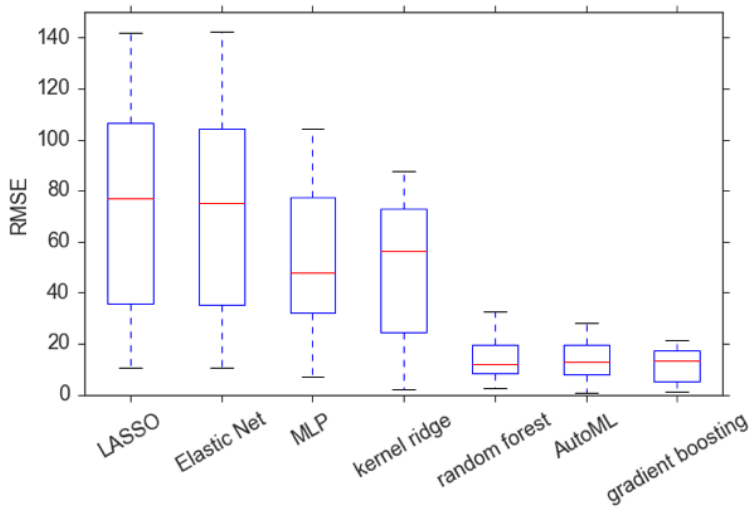

Figure 3.20: Distributions of RMSE over 3-fold cross-validation using 10 different partitions. Boxes represents the lower to upper quartile values of the data, with a red line at the median. The whiskers show the range of data (for dataset 2)

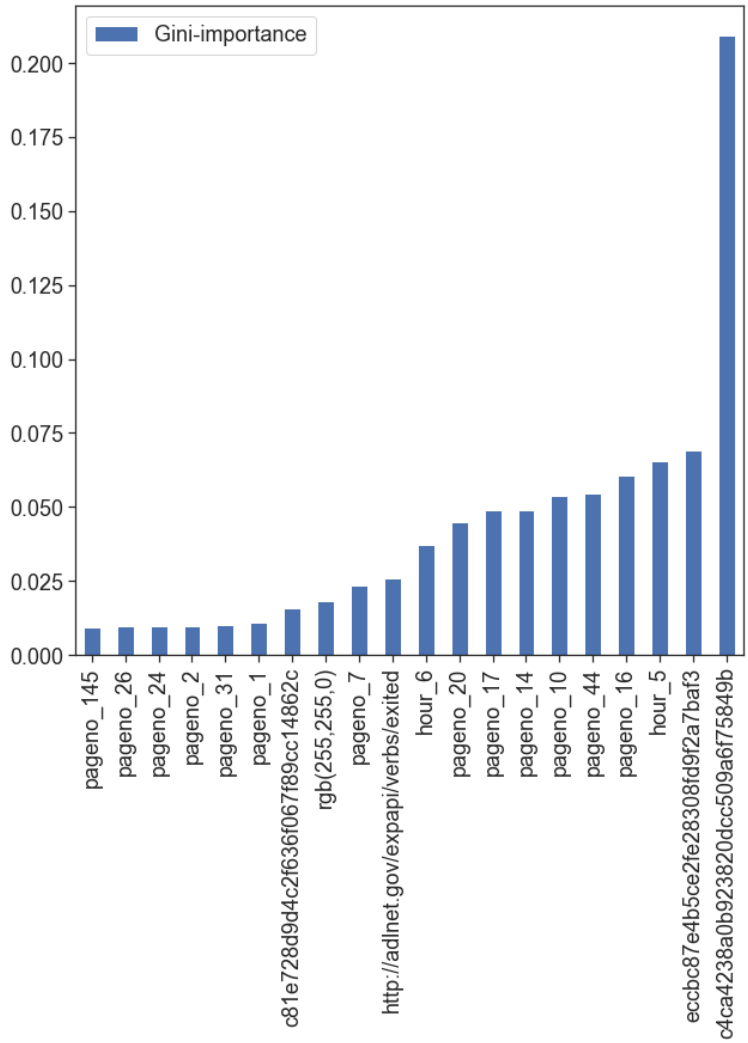

Figure 3.21: Features sorted by Gini-importance, indicating how much each feature contributed to making predictions in random forest. (for dataset 2)

Chapter 4

Factorization of the score matrix

This section describes the method that we will use to extract the skills-acquisition matrix $\boldsymbol{D}$ and the Q-matrix $\boldsymbol{Q}$ from the test-result matrix $\boldsymbol{R}$ using sparse modeling.

Our proposed method has two main features. One is that it uses K-SVD, an existing method of sparse modeling that uses an iterative procedure to factorize the observed data matrix into the product of a dictionary $\boldsymbol{D}$ and a sparse matrix $\boldsymbol{Q}$. Another feature is to employ bidictionary learning, which we newly proposed. The purpose of the latter is to evaluate the number of columns in $\boldsymbol{D}$, which corresponds to the number of templates. It is a hyper-parameter usually required to set manually in K-SVD.

4.1 Sparse modeling by K-SVD

Let $\boldsymbol{A}_{i:}$ denotes the i-th row vector of matrix $\boldsymbol{A}$ and $\boldsymbol{A}_{:j}$ denotes the j-th column vector of matrix $\boldsymbol{A}$. Given observed datmatrix $\boldsymbol{R} \in \mathbb{R}^{m \times n}$, dictionary learning seeks a dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times k}$ and a sparse matrix $\boldsymbol{Q} \in \mathbb{R}^{k \times n}$ which fulfills $\boldsymbol{R} \approx \boldsymbol{D}\boldsymbol{Q}$. The columns of $\boldsymbol{D}$ are called *atoms*. They are usually normalized using $_2$-norm, so that $\|\boldsymbol{D}_{:h}\| = 1$ for all h.

One of the most widely used dictionary learning methods, K-SVD [8], is outlined in Code 1. It starts with a randomly generated dictionary, and iteratively applies two stages, i.e., (1) sparse coding and (2) a dictionary update, for a set number of iterations. Dictionary $\boldsymbol{D}$ stays fixed in the sparse coding stage, and the optimal $\boldsymbol{Q}$ is sought. Any method of sparse coding can be used, e.g., orthogonal matching pursuit (OMP) or basis pursuit (BP). Both $\boldsymbol{D}$ and $\boldsymbol{Q}$ are revised in the dictionary update stage. The main goal is to minimize the norm of the error matrix, $\boldsymbol{E} = \boldsymbol{R} - \boldsymbol{D}\boldsymbol{Q}$. To measure the overall value of the elements of $\boldsymbol{E}$, its Frobenius norm, defined by $\|\boldsymbol{E}\|_2 = \sqrt{\sum_{i,j} E_{ij}^2}$, is used. In other words, it is the square root of the sum of squares of all elements in $\boldsymbol{E}$. It can be expanded as follows.

$$\begin{aligned} \|\boldsymbol{E}\|_F &= \|\boldsymbol{R} - \boldsymbol{D}\boldsymbol{Q}\|_F = \left\| \boldsymbol{R} - \sum_{j=1}^{k} \boldsymbol{D}_{:j}\boldsymbol{Q}_{j:} \right\|_F = \left\| \left(\boldsymbol{R} - \sum_{j \neq h} \boldsymbol{D}_{:j}\boldsymbol{Q}_{j:} \right) - \boldsymbol{D}_{:h}\boldsymbol{Q}_{h:} \right\|_F \\ &= \|\boldsymbol{E}_h - \boldsymbol{D}_{:h}\boldsymbol{Q}_{h:}\|_F \end{aligned} \tag{4.1}$$

In the final term of Equation 4.1, $\boldsymbol{E}_h$ is defined as $\boldsymbol{R} - \sum_{j \neq h} \boldsymbol{D}_{:j}\boldsymbol{Q}_{j:}$. This reduces the problem to find the rank-1 approximation to $\boldsymbol{E}_h$ for each h. When this approximation is conducted, $\boldsymbol{Q}_{h:}$ must stay sparse. In order to do this, a smaller matrix $\tilde{\boldsymbol{E}}_h$ defined below

is introduced, so that the zero entries of $\boldsymbol{Q}_{h:}$ are not revised in the iterative algorithm described below. An ascending sequence of integers defined by $\omega_h = (q|1 \leq q \leq n, \boldsymbol{Q}_{hq} \neq 0)$ is introduced for this purpose. ω_h is often called *support*. ω_h consists of the indices of the non-zero entries of $\boldsymbol{Q}_{h:}$. Using this, projection matrix Ω_h is defined as follows. Let $|\omega_h|$ indicate the number of elements in $|\omega_h|$. For $i = 1, ..., |\omega_h|$, the $(w_h(i), i)$-th entries of Ω_h are 1, and all other entries are 0. Finally, define $\tilde{\boldsymbol{E}}_h = \boldsymbol{E}_h\Omega_h$, and $\tilde{\boldsymbol{E}}_h$ is then approximated by a product of a column vector and a row vector. In other words, $\tilde{\boldsymbol{E}}_h$ is approximated by a rank 1 matrix. In K-SVD, singular value decomposition (SVD) is used for this low rank approximation. Let $\tilde{\boldsymbol{E}}_h = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T$ be SVD. Then we can usse an approximation $\boldsymbol{E}_h \simeq \Lambda_{11}\boldsymbol{U}_{:1}(\boldsymbol{V}_{:1})^T$. Finally, $\boldsymbol{U}_{:1}$ is substituted into $\boldsymbol{D}_{:h}$, while $\Lambda_{11}(\boldsymbol{V}_{:1})^T$ is substituted into the non-zero entries of $\boldsymbol{Q}_{h:}$. The SVD part can also be replaced by a more computationally efficient low-rank approximation method [14].

Code 1 K-SVD

Input: Observed data matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$
Output: Estimated dictionary $\boldsymbol{D} \in \mathbb{R}^{m\times k}$ and estimated source matrix $\boldsymbol{Q} \in \mathbb{R}^{k\times n}$

```
Set D randomly
for t = 1 to τ
  Sparse coding stage:
    Get sparse Q that fulfills R ≈ DQ
  Dictionary update stage:
    for h = 1 to k
      E_h ⟵ R − Σ_{j≠h} D_{:j} Q_{j:}
      Obtain Ω_h using Q_{h:}
      Ẽ_h ⟵ E_h Ω_h
      By applying SVD to Ẽ_h, get U_{:1}, V_{:1}, and Λ_11
      D_{:h} ⟵ U_{:1}
      Replace the non-zero entries of Q_{h:} by Λ_11 (V_{:1})^T
    end
end
```

4.2 Model selection by bidictionary learning

The number of columns k for $\boldsymbol{D}$ represents the number of templates. We call this value the *rank*. More templates there are, their sum can fit observed data more. This is because using more templates (which corresponds to the number of columns in $\boldsymbol{D}$), it is easier to reconstruct $\boldsymbol{R}$ using a weighted sum of templates. The rank k, therefore, represents how complex the model is. If k is too large, the model tends to overfit to training data. More templates there are, less difference there will be between $\boldsymbol{R}$ and $\boldsymbol{DQ}$, because using more templates (which corresponds to the number of columns in $\boldsymbol{D}$), it is easier to reconstruct $\boldsymbol{R}$ using a weighted sum of templates. However, that would result in overfitting. For example, if $k \geq n$, where n is the number of columns in $\boldsymbol{R}$, then there is a trivial solution where there is a template for each sample since there are more templates than there are samples. It means that k should not be too big to make the dictionary applicable to newly observed data. In our case, it is necessary to obtain skills that are general enough.

The rank is a hyperparameter that represents the number of templates used to construct observed data. The value must be optimized to obtain skills that represents the underlining structure of the test. Since it determines the complexity of the model, optimization of the

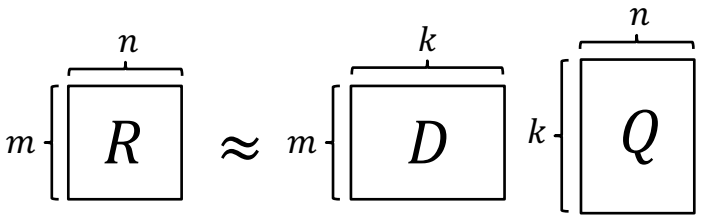

Figure 4.1: Factorization of a test-result matrix $\boldsymbol{R}$ to $\boldsymbol{D}$ and $\boldsymbol{Q}$ by dictionary learning.

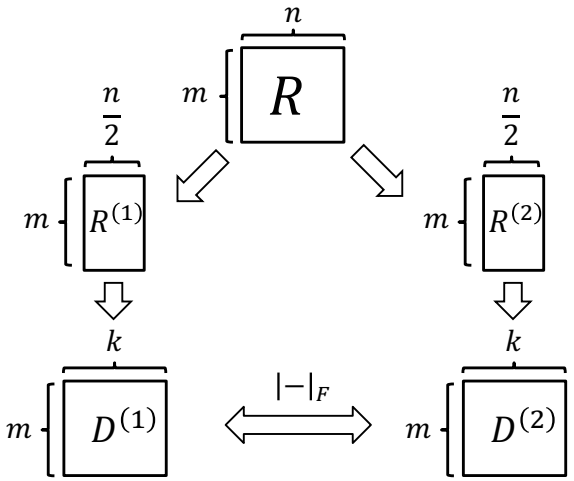

Figure 4.2: In bidictionary learning, $\boldsymbol{R}$ is split into two parts, then dictionary learning is conducted for each part.

column number in matrix factorization is a typical model selection problem. In this thesis, we propose a way to evaluate k by seeing how stable templates are concerning variation in observed data. We will call this method **bidictionary learning**, since two dictionaries are constructed and compared. By checking how stable templates are when different parts of observed data were used, we see the reliability of the obtained templates, meaning they are more likely to correspond to implicit features (in this case, skills), rather than mere noise and artifacts.

Figure 4.2 illustrates the process of bidictionary learning. First, separate the matrix into two parts, $\boldsymbol{R}^{(1)}$ and $\boldsymbol{R}^{(2)}$. Second, dictionary learning is carried out for each of the two parts, so that two dictionaries $\boldsymbol{D}^{(1)}$ and $\boldsymbol{D}^{(2)}$ are obtained. Finally, two learned dictionaries are compared using the Frobenius norm of the difference.

If templates differ significantly between $\boldsymbol{D}^{(1)}$ and $\boldsymbol{D}^{(2)}$, they are not stable concerning statistical variation in data. They do not represent the underlying skill structure. We should choose the model that gives the most stable set of skills. In other words, with the minimum difference between $\boldsymbol{D}^{(1)}$ and $\boldsymbol{D}^{(2)}$. The best model — or the number of templates — is the one that makes the difference smallest.

One thing that should be taken care of when measuring the difference between $\boldsymbol{D}^{(1)}$ and $\boldsymbol{D}^{(2)}$ is that the set of skills does not change by permuting columns in the dictionary and

Code 2 bidictionary learning

Input: Observed data matrix $\boldsymbol{R} \in \mathbb{R}^{m \times n}$ and rank k
Output: Estimated difference $\hat{d}_{\boldsymbol{D}}$
  Split $\boldsymbol{R}$ into two parts, $\boldsymbol{R}^{(1)}$ and $\boldsymbol{R}^{(2)}$
  Set $\hat{d}_{\boldsymbol{D}} = 0$
    For h in $[1, 2]$
      Do dictionary learning on $\boldsymbol{R}^{(h)}$ and get $\boldsymbol{D}^{(h)} \in \mathbb{R}^{m \times k}$
    For i in $[1, k]$
      Find j that minimizes $\|\boldsymbol{D}^{(1)}_{:i} - \boldsymbol{D}^{(2)}_{:j}\|$ and put it into $\hat{j}$
      $\hat{d}_{\boldsymbol{D}} \longleftarrow \|\boldsymbol{D}^{(1)}_{:i} - \boldsymbol{D}^{(2)}_{:\hat{j}}\| + \hat{d}_{\boldsymbol{D}}$
      Remove column $\boldsymbol{D}^{(1)}_{:i}$ from $\boldsymbol{D}^{(1)}$
      Remove column $\boldsymbol{D}^{(2)}_{:\hat{j}}$ from $\boldsymbol{D}^{(2)}$

rows in the Q-matrix. Equation 4.2 shows that replacing columns of $\boldsymbol{A}$ and rows of $\boldsymbol{B}$ at the same time results in the same product.

$$[\,\boldsymbol{A}_{:1}\boldsymbol{A}_{:2}]\begin{bmatrix}\boldsymbol{B}_{1:}\\ \boldsymbol{B}_{2:}\end{bmatrix} = \boldsymbol{A}_{:1}\boldsymbol{B}_{1:} + \boldsymbol{A}_{:2}\boldsymbol{B}_{2:} = \boldsymbol{A}_{:2}\boldsymbol{B}_{2:} + \boldsymbol{A}_{:1}\boldsymbol{B}_{1:} = [\,\boldsymbol{A}_{:2}\boldsymbol{A}_{:1}]\begin{bmatrix}\boldsymbol{B}_{1:}\\ \boldsymbol{B}_{2:}\end{bmatrix} \tag{4.2}$$

When measuring the similarity between two dictionaries $\boldsymbol{D}^{(1)}$ and $\boldsymbol{D}^{(2)}$, a column of the former must be matched to the one that is most similar to it among all columns of the latter. In other words, we must find the minimum dissimilarity using different transforms that permute columns of $\boldsymbol{D}^{(2)}$, as expressed in Equation 4.3, where ρ represents a transform (permutation) that changes the order of columns. m is the number of rows, and k is the number of columns for $\boldsymbol{D}^{(1)}$.

$$\check{d}_{\boldsymbol{D}} = \frac{1}{mk}\min_{\rho}\|\boldsymbol{D}^{(1)} - \rho(\boldsymbol{D}^{(2)})\| \tag{4.3}$$

Since there are $k!$ permutations of columns in $\boldsymbol{D}^{(2)}$, computing $\check{d}_{\boldsymbol{D}}$ is not feasible when $\boldsymbol{D}^{(2)}$ has many columns. Instead, we use a greedy algorithm to approximate $\check{d}_{\boldsymbol{D}}$. The algorithm is described as Code 2. In each step, it finds the column $\boldsymbol{D}^{(2)}_{:j}$ that has the minimum norm with $\boldsymbol{D}^{(1)}_{:i}$. In other words, it finds $\boldsymbol{D}^{(2)}_{:j}$ that minimizes $\|\boldsymbol{D}^{(1)}_{:i} - \boldsymbol{D}^{(2)}_{:j}\|$. Then columns $\boldsymbol{D}^{(1)}_{:i}$ is removed from $\boldsymbol{D}^{(1)}$, and the column $\boldsymbol{D}^{(2)}_{:j}$ is removed from $\boldsymbol{D}^{(2)}$. The sum of the differences obtained in such a greedy manner will be denoted by $\hat{d}_{\boldsymbol{D}}$. It is an estimate to $\check{d}_{\boldsymbol{D}}$.

4.3 Model selection by information theoretic criteria

This section explains how to select models based on information theory. Akaike's Information Criterion (AIC) [15] is a typical example of model selection method based on information theory. It is defined as

$$\mathrm{AIC} = -2\ln(L) + 2\kappa, \tag{4.4}$$

where L is the likelihood function, κ is the number of parameters.
For dictionary learning, κ is the number of elements in the dictionary $\boldsymbol{D}$.

Bayesian Information Criterion (BIC) [16] tries to improve AIC by penalizing it more when the number of samples are large. It is defined as

$$\mathrm{BIC} = -2\ln(L) + \kappa \ln(n), \tag{4.5}$$

where n is the number of samples.

The left side of the Equations 4.4 and 4.5 represents the goodness of fit of the model, and the right side has the effect of regularization. In other words, these criteria are selecting the most applicable parameters while keeping the model from becoming excessively complicated. When selecting a model based on AIC or BIC, it is good to select a parameter that minimizes the score. The likelihood function in dictionary learning is defined as

$$P(\mathbf{q}|\mathbf{D},\mathbf{r}) = \frac{1}{Z(\mathbf{r},\mathbf{D})} \prod_{i=1}^{N} \left[(1-\rho)\delta\left(q_i\right) + \rho\phi\left(q_i\right)\right] \prod_{\mu=1}^{M} \frac{1}{\sqrt{2\pi\Delta_\mu}} e^{-\frac{1}{2\Delta_\mu}\left(r_\mu - \sum_{i=1}^{N} D_{\mu i} q_i\right)^2} \tag{4.6}$$

where q is the sparse vector, r is the test-result vector, δ is the Gaussian distribution, ϕ is the Laplace distribution, Δ is the noise parameter [17].

4.4 Dataset

By imposing students to take a final exam for a course provided in a university, we requested that dataset of created by expert for evaluating our proposed method. The course was titled an *Introduction to Databases* and targeted on undergraduate students.

In the exam, there were 45 multiple-choice questions. Most questions had four or five options to choose from, as indicated in the following examples, translated from Japanese.

Sample questions from the exam used in evaluation

The set of attributes that a relation has is called **Q1** . Two commonly ways to design **Q1** are **Q2** and **Q3** . When there is a relation having to many attributes after conducting **Q2** , the relation is often further decomposed using **Q3** .

Q1. **a.** dependency. **b.** transaction. **c.** SQL. **d.** candidate key. **e.** schema.

Q2. **a.** ER diagram. **b.** normalization. **c.** isolation level. **d.** index. **e.** relational algebra.

Q3. **a.** primary key. **b.** ACID properties. **c.** index. **d.** inner join. **e.** normalization.

Among those taking the exam, 100 students agreed to join the experiment and have their data processed for this work. The test-result matrix R has 45 rows and 100 columns, that is, $m = 45$ and $n = 100$.

The average score of the students was 33.14 correct out of 45 questions. The highest score was 45, and the lowest score was 16. The standard deviation of the score was 7.61. The easiest question was the one that 96 students answered correctly. The most difficult question was the one that only 7 students answered correctly.

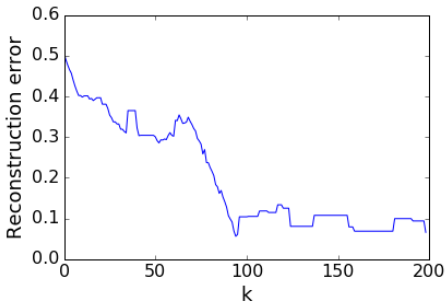

Figure 4.3: Reconstruction error $\|\boldsymbol{E}\|_F = \|\boldsymbol{R} - \boldsymbol{DQ}\|_F$ for different values of k.

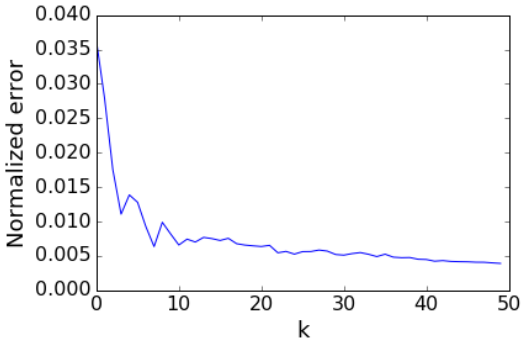

Figure 4.4: The unstability $\check{d}_{\boldsymbol{D}}$ of the dictionary as the rank k is changed from 1 to 50.

4.5 Results

4.5.1 Reconstruction error

Throughout the experiments, the number of iteration for dictionary learning is set to 500. The number of non-zero elements s is set to 3.

The metric for how well the product of the skill-acquisition matrix $\boldsymbol{D}$ and the Q-matrix $\boldsymbol{Q}$ represents the test-result matrix $\boldsymbol{R}$ is measured using $\|\boldsymbol{E}\|_F = \|\boldsymbol{R} - \boldsymbol{DQ}\|_F$ defined by Equation 4.1. In this section we will call $\|\boldsymbol{E}\|_F$ the *reconstruction error*. Figure 4.3 illustrates the reconstruction error $\|\boldsymbol{E}\|_F = \|\boldsymbol{R} - \boldsymbol{DQ}\|_F$ for different values of k. As expected, the error decreases as k increases, since $\boldsymbol{R}$ can be reconstructed more accurately using more templates. However, when k is large, we expect that overfitting is happening.

4.5.2 Bidictionary learning

For evaluating bidictionary learning, we use $\check{d}_{\boldsymbol{D}} = \frac{1}{mk} \min_\rho \|\boldsymbol{D}^{(1)} - \rho(\boldsymbol{D}^{(2)})\|_2$ introduced in Section 4.2. We will call $\check{d}_{\boldsymbol{D}}$ the *unstability* of the dictionary. Figure 4.4 illustrates how the difference between two dictionaries in bidictionary learning, represented by unstability $\check{d}_{\boldsymbol{D}}$, changes as rank k is increased.

The result shows that the instability drops significantly as k increases until $k = 8 \sim 10$, but then shifts to the phase of gradual decrease. It suggests that setting k to around 8 to 10 is suitable since it would give a small enough error without using an over-complex model.

It also happened to be that when the problems in the exam were manually categorized, they grouped into ten categories. Since it is interesting that the numbers roughly matched, we conducted a further experiment to see how well categorization based on estimated skills correspond to manual categorization. The result is presented in the following subsection.

4.5.3 Information theoretic criteria

In order to evaluate the results based on information theory, we used two commonly used information criteria, AIC and BIC, as described in Section 4.3. Figure 4.5 is the result of computing AIC and BIC to the dataset. The smaller the score, the better the information criterion. However, the result shows that the simplest model parameter has the best value. This result implies that these criteria were not working properly for this model class and

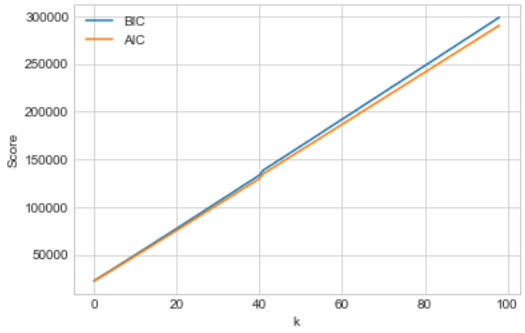

Figure 4.5: Applying the information criterion of AIC, BIC to experimental data.

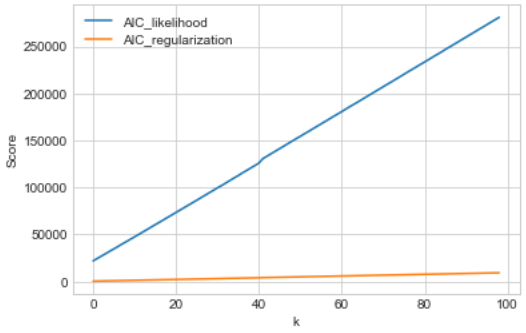

Figure 4.6: Comparison of the likelihood term with the regularization term using AIC.

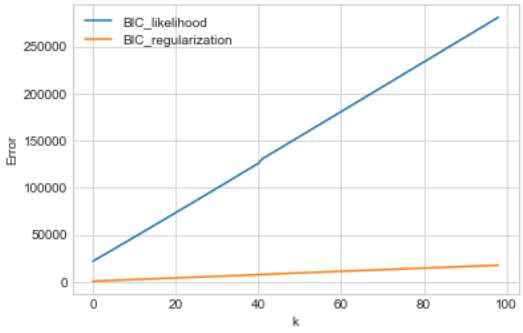

Figure 4.7: Comparison of the likelihood term with the regularization term using BIC.

the dataset. Figure 4.7 shows the composition of the loss function. The large discrepancy between the likelihood term and the regularization term was the reason for assigning the best value for the simplest model. The value of the likelihood term was too large compared to the penalty imposed by the regularization term.

4.5.4 Comparison with manual categorization

The Q-matrix obtained by sparse modeling was further evaluated based on how well it classifies problems. First, manual categorization was carried out, grouping problem based on their similarity and knowledge required to solve them. It resulted in ten categories. Since there were 45 problems, it means that each contained on average 4.5 problems. This categorization can be represented by matrix $\boldsymbol{G}$ where $G_{hj} = 1$ if problem j is assigned to category h, and $G_{hj} = 0$ otherwise. Next, for each skill h, the elements of the hth row of $\boldsymbol{Q}$ were sorted, and the four largest elements were assigned to skill h. Since a column of $\boldsymbol{Q}$ corresponds to a problem, it means four problems were assigned to skill h. This categorization is represented by $\tilde{\boldsymbol{Q}}$ where $\tilde{Q}_{hj} = 1$ if problem j is assigned to skill h, and $\tilde{Q}_{hj} = 0$ otherwise.

We would like to compare how two ways of categorization differ. The difference can be measured using the absolute error defined by $\breve{d}_{\boldsymbol{G}} = \min_{\rho} \|\boldsymbol{G} - \rho(\tilde{\boldsymbol{Q}})\|_1$, where ρ represents an arbitrary permutation of rows. However, since checking all permutation of columns would computationally intense, so $\breve{d}_{\boldsymbol{G}}$ was estimated greedily. For each row $\boldsymbol{G}_{h:}$, all rows of $\tilde{\boldsymbol{Q}}$

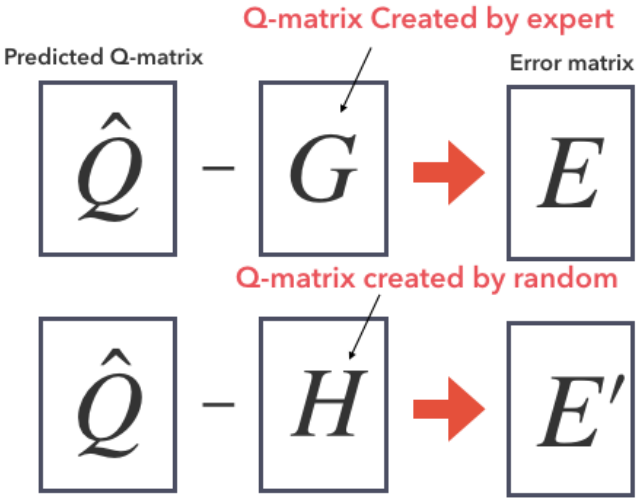

Figure 4.8: Comparison of error matrices created by an expert and created randomly.

were compared in terms of the absolute error $\|\boldsymbol{G}_{h:} - \tilde{\boldsymbol{Q}}_{\ell:}\|_1$, and the ℓth row having the least value was selected, and removed from $\tilde{\boldsymbol{Q}}$. In this way, we obtain a sequence of absolute errors between matching pairs.

To evaluate if these errors are small or not, we generated a random assignment of test problems into categories. For each skill h, four problems were randomly selected. It can be represented by matrix $\boldsymbol{H}$ in the same way as $\boldsymbol{G}$. The difference between $\boldsymbol{H}$ and $\tilde{\boldsymbol{Q}}$ were measured in the same way as between $\boldsymbol{G}$ and $\tilde{\boldsymbol{Q}}$. Figure 4.8 compares error matrices by expert and random. Figure 4.9 illustrates the result of comparison. In the figure, the skills were sorted in ascending order of the absolute error. The random assignment had a larger absolute error than categories assigned by estimated skills. For the random assignment, the average absolute error was 6.4. For assignment based on estimated skills, the average absolute error was 5.7.

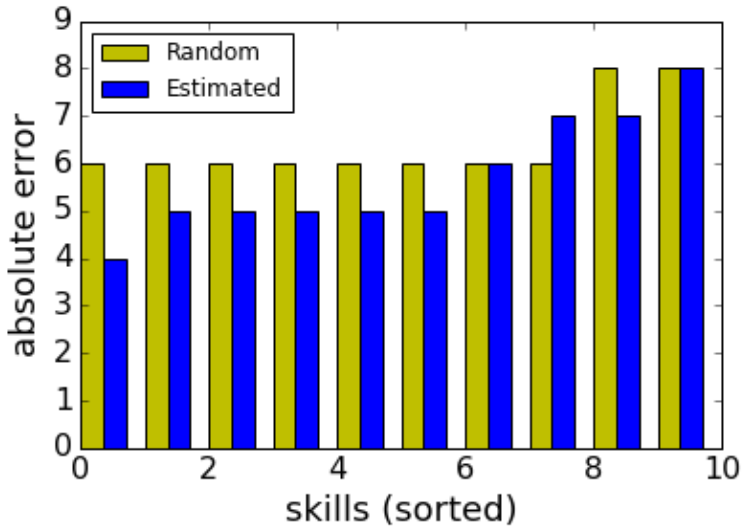

Figure 4.9: Absolute errors of categorization of problems based on estimated skills with respect to manual categorization, compared with random categorization.

Chapter 5

Conclusion

Sparse modeling of test results provides a way of knowing what problems constitute skills and how well each student acquired them. In order to use it in practice, one needs to set the model complexity. In this case, it corresponds to the rank of matrix factorization. We proposed and compared various model selection methods, including our original method, bidictionary learning.

We tested the method using an exam taken by college students. Information theoretic criteria for model selection turned out to be unsuccessful to this task. We assumed it is due to the complex nature of the dictionary learning task, compared to more well-behaved statistical models that AIC and BIC has been applied to. On the other hand, bidictionary learning suggested that the optimal rank could be obtained by measuring the unstability of dictionaries.

The result of dictionary learning using the selected model complexity showed that the estimated skills resembles manual categorization of test problems, suggesting the method can uncover skills implicit in each problem.

Not only is the proposed method practically useful for helping students, but it can also be used to find elementary units of knowledge that constitute problems. It will provide a model on what it means to understand the subject taught in a course. This is of much interest in the field of educational psychology.

Feature analysis for predicting students' performance on an e-learning system revealed that time related features play an important role in making good predictions. Students accessing or not accessing the e-learning system at specific time of the day performed better than students who didn't. Among various machine learning methods being compared, random forest, AutoML, and gradient boosting performed better than other methods. It suggests that more recently proposed methods improve predictions in educational data mining as well.

In this work, we only focused on optimizing the rank of matrix factorization, which partially determines the model complexity. The rank corresponds to the number of templates used for reconstructing observed data. In sparse modeling, another important hyperparameter that determines the model complexity is the number of non-zero elements. In the experiments, we have used a constant value for this. Optimizing the number of non-zero elements together with the rank is a part of future work.

Acknowledgement

I would like to thank T. Tezuka and K. Nozawa for useful discussions. “Good Game Well Played.”

This work was supported by JSPS KAKENHI Grant Numbers JP16K00228 and JP16H02904.

References

[1] M.C. Desmarais, “Mapping question items to skills with non-negative matrix factorization”, ACM SIGKDD Explorations Newsletter 13.2, pp.30-36, 2012.

[2] T. Barnes, “The Q-matrix method: Mining student response data for knowledge”, American Association for Artificial Intelligence 2005 Educational Data Mining Workshop, 2005.

[3] T. Winters, C. Shelton, T. Payne, and G. Mei, “Topic extraction from item-level grades”, American Association for Artificial Intelligence 2005, Workshop on Educational Datamining, Pittsburgh, PA. Vol. 1. No. 2. 2005.

[4] K. Tatsuoka, Cognitive Assessment: An Introduction to the Rule Space Method, Routledge Academic, 2009.

[5] P. Miettinen and J. Vreeken. “mdl4bmf: Minimum description length for Boolean matrix factorization”, ACM Transactions on Knowledge Discovery from Data (TKDD) 8.4, 2014.

[6] D.D. Lee and H.S. Seung, “Algorithms for Non-negative Matrix Factorization”, Advances in Neural Information Processing Systems 13, 2000.

[7] S. Oeda, Y. Ito, and K. Yamanishi, “Extracting Latent Skills from Time Series of Asynchronous and Incomplete Examinations”, Proceedings of the 7th International Conference on Educational Data Mining (EDM2014), pp. 367-368, 2014.

[8] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation”, IEEE Transactions on Signal Processing, vol.54, no.11, pp.4311-4322, 2006.

[9] A. Bruckstein, D. L. Donoho, and M. Elad, “From sparse solutions of systems of equations to sparse modeling of signals and images”, SIAM Review, vol.51, Issue 1, pp.34-81, 2009.

[10] H. Ogata, C. Yin, M. Oi, F. Okubo, A. Shimada, K. Kojima, and M. Yamada, E-Book-based learning analytics in university education, Proceedings of the 23rd International Conference on Computers in Education (ICCE 2015) pp.401-406, 2015.

[11] B. Flanagan, H. Ogata, Integration of Learning Analytics Research and Production Systems While Protecting Privacy, Proceedings of the 25th International Conference on Computers in Education (ICCE2017), pp.333-338, 2017.

[12] H. Ogata, M. Oi, K. Mohri, F. Okubo, A. Shimada, M. Yamada, J. Wang, and S. Hirokawa, Learning Analytics for E-Book-Based Educational Big Data in Higher Education, In Smart Sensors at the IoT Frontier, pp.327-350, 2017.

[13] F. Matthias, K. Aaron, E. Katharina, S. Jost, B. Manuel, H. Frank, Efficient and robust automated machine learning, Advances in Neural Information Processing Systems, pp.2962-2970, 2015.

[14] T. Tezuka, “Dictionary learning by normalized bilateral projection”, Journal of Information Processing, Vol.24, No.3, pp.565-572, 2016.

[15] H.Akaike, "Information theory and an extension of the maximum likelihood principle." Selected papers of hirotugu akaike, pp.199-213, 1998.

[16] G.Schwarz, Estimating the dimension of a model, The annals of statistics, 6(2), pp461-464, 1978.

[17] F.Krzakala, M.Mézard, F.Sausset, Y.Sun, L.Zdeborová, Probabilistic reconstruction in compressed sensing: algorithms, phase diagrams, and threshold achieving matrices, Journal of Statistical Mechanics: Theory and Experiment, P08009, 2012.