Mapping Problems to Skills Combining Expert Opinion and Student Data

Juraj Nižnan(✉), Radek Pelánek, and Jiří Řihák

Masaryk University, Brno, Czech Republic
{niznan,thran}@mail.muni.cz, pelanek@fi.muni.cz

Abstract. Construction of a mapping between educational content and skills is an important part of development of adaptive educational systems. This task is difficult, requires a domain expert, and any mistakes in the mapping may hinder the potential of an educational system. In this work we study techniques for improving a problem-skill mapping constructed by a domain expert using student data, particularly problem solving times. We describe and compare different techniques for the task – a multidimensional model of problem solving times and supervised classification techniques. In the evaluation we focus on surveying situations where the combination of expert opinion with student data is most useful.

1 Introduction

With the increasing use of technology in education it becomes possible to create educational systems that are personalized and adaptive. One of important aspects of development of such systems is the construction of a mapping between educational content (questions, problems) and latent skills. This mapping is important for student skill estimation, which guides the adaptive behavior of systems, and also for the system interface, e.g., for grouping of problems or feedback to students using open learner models.

Latent skills, also denoted as knowledge components or concepts, may in mathematics correspond for example to "trigonometric functions", "linear functions" or "binary numbers". The suitable granularity and choice of the skills, as well as the construction of the mapping between a system content and skills depends on a particular application. This task is typically done by a human and since it is a difficult process, it requires a domain expert. The labeling of items, particularly for large item pools, may be time-consuming, and consequently the process is rather expensive, which is a drawback particularly for applications tailored for relatively small target groups (specialized domains, languages spoken by relatively small number of speakers). Another approach is to use automatic construction of the mapping from the data. To be reliable, the automatic approach requires large amounts of data. Moreover, the automatically determined skills may be hard to interpret, which may limit their applicability – we want to provide students with feedback on their knowledge of trigonometric functions,

P. Hliněný et al. (Eds.): MEMICS 2014, LNCS 8934, pp. 113–124, 2014.
DOI: 10.1007/978-3-319-14896-0_10

not on their “mathematical skill number 3”. Synergy of these two approaches may bring useful results. We can use a human expert to provide initial labeling of problems and then automatic methods can be used to detect errors that the human might have introduced and to fix them. This approach alleviates the requirements on the human, so it may be possible to use non-expert or to do labeling in quick, coarse way. It also provides interpretability and a good starting point for automatic methods.

Depending on the quality of the provided expert labeling and amount of data, there are three possible scenarios. If the number of expert errors is small or the data are insufficient, it is best to use just the expert opinion. If the expert makes lot of mistakes and large data are available, then it is best to use just the data. We are interested in the region between these two cases, when it is most advantageous to combine both the expert input and available data. Our aim is to explore techniques for such combination and to map the size of this region.

Our particular context are data about problem solving times from Problem Solving Tutor [1,2] (mathematics problems, logic puzzles). We explore two approaches for combining expert input with data. The first one is a multidimensional model of problem solving times. Model parameters are estimated using stochastic gradient descent, where expert input is used for initialization. The second approach is supervised learning approach, where the expert labels are used for learning a classifier and mistakes of the classifier are interpreted as errors of the expert. Results show that the two approaches lead to similar performance results, the supervised learning approach is computationally significantly faster. The results also show that the region, where the combination of expert opinion with data is advantageous, is quite large and that this approach deserves further attention.

2 Related Work

The main area of related research is concerned with the Q-matrix approach [3–5]. This research was done mainly for correctness data, i.e., binary response (correct/incorrect), whereas we use problem solving times. Also, most of the research either uses an expert provided Q-matrix (without any analysis of suitability of the matrix) or tries to find the Q-matrix solely from the data (possibly discussing the differences between mined and expert Q-matrices [6]). In previous work on analysis of skills in context of problem solving times, we [7] used spectral clustering to automatically detect skills from data.

Only few works focused on checking or improving expert provided Q-matrix [8–12]. The closest relevant work [10] used matrix factorization approach to enhance expert provided matrix. The accuracy of this technique was compared with two other techniques on multiple datasets [12]. Our setting is different, as we study only classification and we use problem solving times instead of correctness data.

Other related techniques are matrix factorization in recommender systems [13] and the SPARFA [14] technique. These techniques also automatically detect concepts in data (e.g., in movie rating), but this research has significant differences from our work. At first, the focus of the research in these settings is mainly on improving predictions, not on the concepts themselves. At second, there is usually no way to use expert input and the output usually is not interpretable and thus is not directly usable for adaptive educational systems.

Learning Factors Analysis [15] is a technique for improving a cognitive model provided by an expert. It performs a combinatorial search over many models. While its goal is similar to ours, the approach is different. Other research [16] is based on a Bayesian approach for performing skill discovery where expert labeling is used as a prior. The goal is to better predict the correctness of student responses. Our measure of success is different. We want to find a correct latent mapping of problems to skills. Both of the above mentioned methods work with correctness data.

Our use of supervised learning techniques, which assumes that the provided labeling is not completely correct, is related to learning with noisy labels [17, 18]. This line of research is focused mainly on improvement of classification methods. We assume a symmetrical noise and thus there is no need for these more sophisticated methods. Preliminary version of our work is available [19].

3 Techniques

In this section we describe the proposed techniques for problem classification by combining input from experts and student data.

3.1 Inputs and Outputs

In the following we assume that we have a set of students S, a set of problems P, and data about problem solving times: $t_{s,p}$ is a logarithm of time it took a student $s \in S$ to solve a problem $p \in P$ (reasons for working with a logarithm of time are described for example in [1,20]). The data is not complete, meaning that not every student solved every problem. We denote $\kappa \subseteq S \times P$ the set of student-problem pairs for which we have observed solving times.

We have an expert labeling $l_E : P \rightarrow \Sigma$ where Σ is the set of skills. We assume that the number of skills is rather small when compared to the number of problems. The expert labeling may contain some mistakes when compared to a correct hidden labeling l. The output of our algorithms is some other labeling l_A that may be different from l_E. The goal of our algorithms is to provide a more accurate labeling (according to l) than l_E.

Some of our algorithms do not work with incomplete data. This is why we transform the data using Spearman's correlation coefficient, which gives a metric of similarity between two problems. When computing the correlation coefficient $r(p_i, p_j)$ of problems p_i and p_j we use only the times of students that solved both problems. Correlation can be used to construct a $|P|$-dimensional vector space where each dimension j represents a correlation with problem p_j. A problem p_i is then represented as a vector $\boldsymbol{r_{p_i}} = \{r(p_i, p_j)\}_{1 \leq j \leq |P|}$. We assume that when two problems are similar then they have similar correlations with other problems and therefore their Euclidean distance $d_r(p_i, p_j) = \|\boldsymbol{r_{p_i}} - \boldsymbol{r_{p_j}}\|$ is small.

3.2 Model with Multidimensional Skill

In this section we introduce a model with multidimensional skill for predicting how much time it takes a student to solve a particular problem. The model uses a few latent attributes: problem bias b_p, student bias β_s, problem skill vector $\boldsymbol{q}_p$ and a student skill vector $\boldsymbol{\theta}_s$. It assumes the following relationship between the attributes: $t_{s,p} = b_p + \beta_s + \boldsymbol{q}_p^\intercal \boldsymbol{\theta}_s + \epsilon$ where ϵ is a gaussian error. It always predicts the time $\hat{t}_{s,p} = b_p + \beta_s + \boldsymbol{q}_p^\intercal \boldsymbol{\theta}_s$.

The problem bias b_p can be thought of as being the basic problem difficulty – a term commonly used in item response theory and in our previous research [1]. Similarly, the student bias β_p can be viewed as a basic negative student skill (higher skill should implicate shorter solving time). The vector $\boldsymbol{q}_p$ of length $|\Sigma|$ represents the weight of individual skills in the problem p. We require its values to be approximately in the $[0, 1]$ range to allow for easy interpretation. The vector $\boldsymbol{\theta}_s$ (of length $|\Sigma|$) can be interpreted as the values of skills the student s has.

This model is a multidimensional extension of a basic model of problem solving times that we used in our previous research [1]: $t_{s,p} = b_p + a_p\theta_s + \epsilon$. The main difference between these models, besides the multidimensional skill, is that the basic model is missing the student bias β_s and instead of $\boldsymbol{q}_p$ it uses a discrimination factor a_p. The reason for different notation is that there are different restrictions on a_p than on $\boldsymbol{q}_p$. Mainly, $mean_{p\in P}(a_p) = -1$.

This model can be alternatively viewed as a matrix factorization problem: $\boldsymbol{T} = \boldsymbol{b}\boldsymbol{1}_{|S|}^\intercal + \boldsymbol{1}_{|P|}\boldsymbol{\beta}^\intercal + \boldsymbol{Q}\boldsymbol{\Theta}^\intercal + \boldsymbol{E}$. The matrix of times $\boldsymbol{T}$ has shape $|P| \times |S|$. A column vector of all-ones of length n is denoted as $\boldsymbol{1}_n$.

This model is supervised in a sense that it is learning to predict the student solving times. As a byproduct we get the Q-matrix $\boldsymbol{Q}$ which represents the problem-skill mapping that we are interested in. We can interpret the matrix $\boldsymbol{\Theta}$ as a mapping of individual students to the values of their skills. The objective of the learning of the model is to find the values of the parameters that minimize the squared prediction error $E = \sum_{(s,p)\in\kappa}(t_{s,p} - b_p - \beta_s - \boldsymbol{q}_p^\intercal \boldsymbol{\theta}_s)^2$.

To get the values of the parameters we use stochastic gradient descent, which is a popular method for similar settings, particularly in the area of collaborative filtering and recommender systems [21]. Other approach to get the parameters is to use alternating convex search. If we fix the problem-related parameters then the problem of minimizing the error E is convex. Alternatively if we fix the student-related parameters this problem is also convex. We can alternate these steps, in each using a convex optimization software to estimate the parameters.

Before we start estimating the parameters we need to initialize the values of the parameters. When the expert labeling is not available we can initialize the Q-matrix randomly with values taken uniformly from $[0, 1]$. However, when the expert labeling is available we use that as the initial Q-matrix instead. After

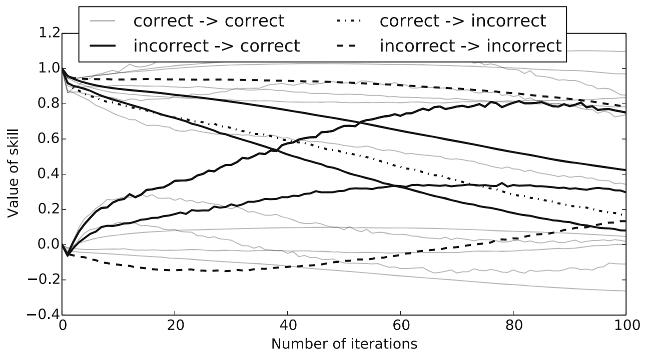

Fig. 1. An example of changing values of one skill during the iterations of gradient descent. Each line represents a single problem.

the algorithm terminates we can check for discrepancies between the expert Q-matrix and the Q-matrix outputted by the parameter estimation algorithm. We will assume that these discrepancies are expert mistakes.

We want to keep the values of the Q-matrix in the $[0, 1]$ range to allow for easy interpretation. When using alternate convex search we can achieve this simply by subjecting it to the appropriate constraints. In stochastic gradient descent we can use L2 regularization by using a slightly modified error function $E' = E + \lambda \sum_{(s,p)\in\kappa} \|\boldsymbol{q}_p\|^2$ where λ is a regularization parameter. Note that using L2 regularization will not keep the values precisely in the $[0, 1]$ range, but choosing an appropriate λ will keep the values from traveling too far.

Figure 1 shows the small gradual changes of values of one skill in the Q-matrix during the passing iterations of stochastic gradient descent. It also illustrates the 4 categories of change that can occur in the Q-matrix: true positive (incorrect classification is fixed), false positive (correct classification changes to incorrect), true negative (correct classification stays correct), false negative (incorrect classification is not fixed). Our experiments have shown that the biconvex optimization approach makes much greater changes to the values of the Q-matrix than gradient descent, resulting in essentially ignoring the expertly initialized Q-matrix. Therefore, in further evaluation we use stochastic gradient descent.

3.3 Supervised Learning

In this section we introduce how supervised classification methods can be used to detect errors in expert labeling.

The main idea can be illustrated by the most straightforward approach which uses k-NN (k-nearest neighbors) algorithm and correlations between problems. We assume that the most correlated problems belong to the same skill and thus have the same labels. So for problem p_i a new label $l_A(p_i)$ will be the most common label among the k most correlated problems from P with problem p_i.

This approach can find some mistakes, however it brings only small improvement of expert labeling l_E.

Similarly, we can perform k-NN with distances d_r defined in Section 3.1. In this case correlation information is used more broadly, because distance of two problems incorporates correlation with all others problems from P. This approach gives more promising results than naive approach above.

Of course it is possible to use different classification methods with metric d_r or vectors $\boldsymbol{r_p}$ as vectors of features. We have chosen logistic regression classifier [22], which is more sophisticated but still computationally fast. Logistic regression classifier finds a linear combination of features $\boldsymbol{w}^\intercal \boldsymbol{r_p}$ of training data which – after passing through logistic function – fits labeling the best. In fact, classifier finds a decision hyperplane which splits the space into two parts corresponding to skills.

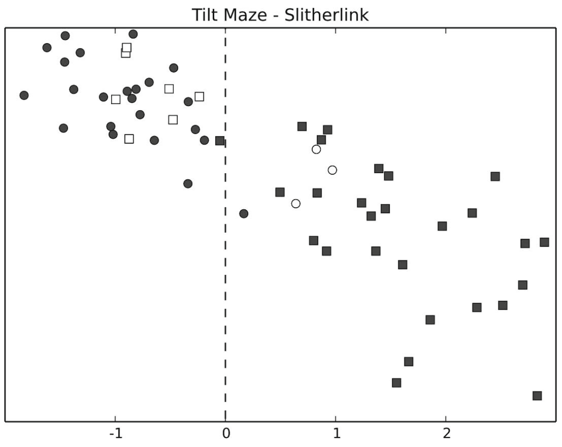

Fig. 2. Projection of vectors $\boldsymbol{r_p}$ to plane, where x-axis represents distance from decision hyperplane. Data points represent problems with labeling provided by expert (squares, circles) and his mistakes (white ones).

Figure 2 gives inside look into how logistic regression works in our setting for 2 skills. The classifier splits the space by decision line to two halfspaces which correspond to the skills. Every point in the figure is a problem and its shape represents labeling provided by expert. White ones mark mistakes of expert. For 3 or more skills one-vs.-all strategy is applied.

Logistic regression classifier (similar as other classifiers) makes two types of mistakes on our training data: (1) mistakes near decision line, which are caused by generalization and not perfect separability of problems; (2) mistakes which correspond to expert mistakes. Our goal is to have maximum of mistakes of type 2 and minimum of mistakes of type 1. The algorithm without regularization ($\lambda = 0$) overfits and does not find any mistakes (it is often possible to split n points in n dimensions into any given two sets). With growing regularization (we used quadratic one) more mistakes of type 2 are found but some mistakes of type 1 occur.

4 Evaluation

Now we turn to evaluation of the proposed techniques on real data.

4.1 Data and Experiment Setup

To evaluate our algorithms we used real data from a Problem Solving Tutor [2]. It is a free web-based tutoring system for practicing problem solving; it is available at `tutor.fi.muni.cz`. The system adapts to an individual student – based on past problem solving attempts the system estimates student's problem solving skill, using this estimated skill it predicts problem solving times for new problems and chooses a suitable problem for a student. The system has more than 12 000 registered students (mainly university and high school students), who have spent more than 18 000 hours solving more than 500 000 problems.

For the evaluation we used problem solving data from 5 problem types: Slitherlink, Tilt Maze, Rushhour, Sokoban, Robotanist. We chose these types of problems because they are some of the most popular ones and therefore a lot of data has been generated from the attempts to solve them. On average we have 80 instances per problem type and 1200 solving attempts per problem.

To simulate multiple skills for the evaluation purposes we mixed data from k problems together ($k \in \{2, 3, 4\}$). Each problem type represents a single skill (or label). An expert is simulated by taking the correct labeling and introducing some random mistakes. Hence, in this situation (as opposed to standard setting), we know the correct "latent" skills and thus we can measure accuracy of a method as the portion of the final labels assigned correctly.

We will denote the general approach described above as ED (expert-data). We compared this approach with the E (expert) approach, where only the expert labeling is used and, the D (data) approach, where only the data is used in an unsupervised fashion.

A baseline approach to unsupervised clustering is the k-means algorithm, in our setting it can be used in straightforward way when the problems are represented by correlation vectors $\boldsymbol{r_p}$ described in Section 3. In previous work [7] we have shown that spectral clustering achieves slightly better results. Here we use for the evaluation of the D approach the spectral clustering.

Expert labeling was obtained as correct labeling (given by problem types) with some additional mistakes. We performed these evaluations with different values of an expert error rate p_e ranging from 0 to 0.5. When the simulated expert was labeling an individual data point, he assigned it the correct label with probability $1 - p_e$, otherwise a random incorrect label was assigned. The expected accuracy of an expert with an error rate p_e is $1 - p_e$.

The expert labeling was then used in the ED approach. When using the model with multidimensional skill, we initialized the algorithm to use the expert labeling. In supervised learning the expert-labeled data was used as a training data set.

4.2 Algorithm Settings

When using the model with multidimensional skill we need to set the following three parameters of stochastic gradient descent: the number of iterations, the learning rate α and the regularization parameter λ. We set all of these parameters empirically. Our experiments have shown that after 100 iterations the values in the Q-matrix are stable. The learning rate α was set to 0.005 which is small enough to allow for smooth changes in the Q-matrix. This is important because our goal is to find a better local optimum near the expertly initialized Q-matrix. The parameter λ was set to 0.0001.

In the case of k-NN algorithm experiments showed that choice of k is not essential; the results are comparable for $20 \leq k \leq \frac{|P|}{2}$. For comparison we used fixed $k = 30$.

For the method based on logistic regression the choice of regularization parameter λ can be used to set the sensitivity of the algorithm to mistakes. With low regularization only the most serious (according to data) mistakes are found (small recall and high precision). With high regularization most of the mistakes of expert are found (rising recall) together with some false positives (dropping precision). Fortunately, the algorithm is not sensitive to regularization too much (the figure uses logarithmic scale of λ) and interval of regularization with good accuracy is wide enough. So it is possible to use one regularization for different skills and gain good results. For comparison we used fixed $\lambda = 2$.

4.3 Results

Figure 3 shows the comparison of the accuracies of the E, ED and D approaches. The left graph shows the results for the mix of 2 problems, the right one for the mix of 3 problems. We can denote three zones within expert error rate based on which approach (E, ED or D) performs the best. The E-zone appears for very small values of p_e, the D-zone appears for very large values of p_e, and the ED-zone appears between them, where the combination of expert labeling and student data is beneficial.

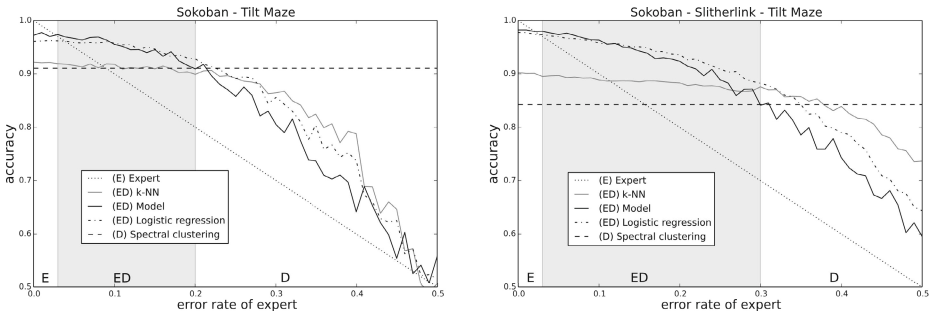

Fig. 3. Comparison of techniques for particular situations (2 and 3 skills). The ED-zone is marked for the model. The accuracy is averaged from 20 repeated runs.

We are interested particularly in the ED-zone, where the newly introduced approaches are the best, specifically in its position and width, which tells us for which values of p_e these approaches are a good choice. Also, maximal benefit of correcting expert labeling (over E and D) can be measured. We will call this feature as *height of ED-zone*.

The figure shows that algorithm based on k-NN brings only small improvement and only for limited range of error rates. The other two approaches based on model and logistic regression are significantly better and to each other comparable. Also, curves of accuracy of these algorithms are often very similar (as illustrated in Figure 3).

Both described algorithms produced similar results, however the algorithm based on logistic regression is significantly faster, because it works with correlation vectors, which substantially reduces the amount of data, and the computation of the decision line is a straightforward process. Algorithm based on the model needs a lot of iterations and repeated passes through all problem solving times. On the other hand this approach gives more information about problem-skill mapping, because it provides Q-matrix and not only labeling.

The ED-zones properties for various numbers of skills and selected combinations of problems sets are covered in Figure 4. The figure illustrates general

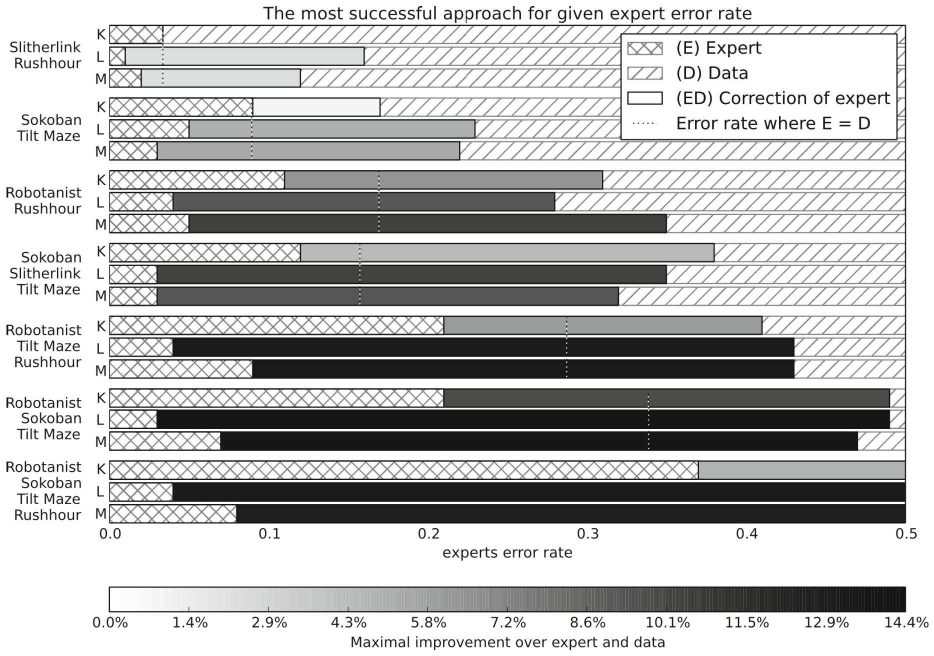

Fig. 4. ED-zones for selected problem combinations. In each case three techniques are compared: k-NN (K), logistic regression (L), and the model with multidimensional skill (M). Degree of shade corresponds to the height of the ED-zone. Dotted lines mark the points where spectral clustering achieved the same results as the simulated expert.

trends that hold also for other combinations of problems. The main fact, which can be observed from the figure, is that with the rising number of skills the ED-zone is larger and higher. This is caused by decreasing accuracy of unsupervised algorithms.

We have further explored properties of the ED-zone using experiments with simulated data. Students times for given skill $i \in \Sigma$ was generated from baseline model (see [1]) as $t_{s,p} = b_p - \theta_{i,s} + \epsilon$ where b is normally distributed difficulty of problems, θ is normally distributed skill of students and ϵ is a gaussian error. The algorithm based on logistic regression was used to compute ED-zone properties.

Figure 5 shows results according to the number of students (30 problems per problem type and 50% completeness of data was used). The combination of expert opinion and student data is most beneficial for middle range of students count. For small number of students there is not enough data for accurate correction of expert. For large amount of students the algorithm which uses only data performs better, thus the ED-zone is smaller, but correction of expert is still the best choice for reasonable error rates of expert. Similar to experiments with real data, the experiments also show that the benefit from combining expert opinion with student data increases with the number of skills.

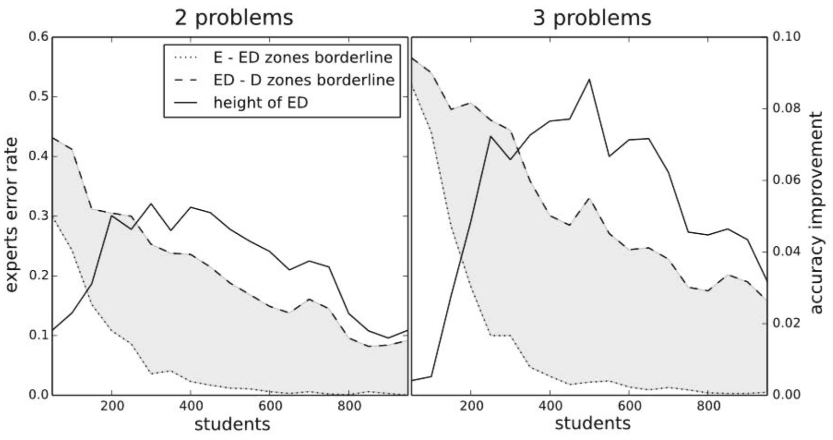

Fig. 5. Properties of the ED-zone on simulated data. Under the dotted line is the E-zone, above the dashed line is the D-zone, the ED-zone is space between them.

5 Discussion

Our experiments address two types of questions: “how” and “when”. The “how” question is concerned with the choice of suitable technique for combining expert opinion and student data. Here the results suggest that on one hand the choice of technique is important – note that two similar supervised approaches (k-NN, logistic regression) achieve quite different results. On the other hand, two significantly different approaches (the multidimensional model of solving times and logistic regression with regularization) achieve very similar results. This suggest that the results of these techniques are close to optimal performance that is possible for the given data.

The "when" question is concerned with mapping when it is useful to use the studied techniques, i.e., when does the combination of expert opinion and student data bring an advantage. The results show that this "zone" is sufficiently large to deserve attention, the size of the zone grows with number of skills and for larger numbers of skills the zone becomes dominant, i.e., for large range of quality of expert input, it is useful to combine the expert opinion with student data.

Another related issue, where combination of expert opinion and student data may be useful, is identification of typical problems and outliers. Based on the student data we can identify candidates for such problems, e.g., using the correlations $r(p_i, p_j)$ (defined in Section 3.1) or the noise in problem solving times (see [1]); our experiments suggest that these different measures lead to similar results. The results may provide valuable information for authors of educational content and teachers, e.g., it may be useful to remove outlier problems from an educational system or use typical problems for exam.

The combination of expert opinion and student data deserves further attention. In the context of correctness modeling, lot of research has been done on the study of Q-matrices, including some works on combining expert opinion with student data. But most of the research studied the "how" question (development of techniques), it would be useful to consider in more detail the "when" question (in what circumstances we should combine expert opinion and data). On the other hand, we have focused mainly on classification (assigning each problem to single skill), whereas the Q-matrix approach can assign items to multiple skills. The above described multidimensional model of problem solving times can be used in the same fashion as the usual Q-matrix approach, but the supervised approach using logistic regression is limited only for classification. Further extensions and evaluations in this direction would be interesting.

References

1. Jarušek, P., Pelánek, R.: Analysis of a simple model of problem solving times. In: Cerri, S.A., Clancey, W.J., Papadourakis, G., Panourgia, K. (eds.) ITS 2012. LNCS, vol. 7315, pp. 379–388. Springer, Heidelberg (2012)
2. Jarušek, P., Pelánek, R.: A web-based problem solving tool for introductory computer science. In: Proc. of Innovation and technology in computer science education, pp. 371–371. ACM (2012)
3. Barnes, T.: The q-matrix method: mining student response data for knowledge. In: American Association for Artificial Intelligence 2005 Educational Data Mining Workshop (2005)
4. Desmarais, M.C., Beheshti, B., Naceur, R.: Item to skills mapping: deriving a conjunctive q-matrix from data. In: Cerri, S.A., Clancey, W.J., Papadourakis, G., Panourgia, K. (eds.) ITS 2012. LNCS, vol. 7315, pp. 454–463. Springer, Heidelberg (2012)

5. Tatsuoka, K.: Rule space: An approach for dealing with misconceptions based on item response theory. Journal of Educational Measurement **20**(4), 345–354 (1983)
6. Barnes, T.: Novel derivation and application of skill matrices: The q-matrix method. In: Handbook on Educational Data Mining (2010)
7. Boroš, P., Nižnan, J., Pelánek, R., Řihák, J.: Automatic detection of concepts from problem solving times. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS, vol. 7926, pp. 595–598. Springer, Heidelberg (2013)
8. De La Torre, J.: An empirically based method of q-matrix validation for the dina model: Development and applications. Journal of Educational Measurement **45**(4), 343–362 (2008)
9. DeCarlo, L.T.: Recognizing uncertainty in the q-matrix via a bayesian extension of the dina model. Applied Psychological Measurement **36**(6), 447–468 (2012)
10. Desmarais, M.C., Naceur, R.: A matrix factorization method for mapping items to skills and for enhancing expert-based Q-matrices. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS, vol. 7926, pp. 441–450. Springer, Heidelberg (2013)
11. Rupp, A., Templin, J.: The effects of q-matrix misspecification on parameter estimates and classification accuracy in the dina model. Educational and Psychological Measurement **68**(1), 78–96 (2008)
12. Desmarais, M.C., Beheshti, B., Xu, P.: The refinement of a q-matrix: assessing methods to validate tasks to skills mapping. In: Proceedings of the 7th International Conference on Educational Data Mining, pp. 308–311 (2014)
13. Kantor, P., Ricci, F., Rokach, L., Shapira, B.: Recommender systems handbook. Springer (2010)
14. Waters, A., Lan, A., Studer, C.: Sparse probit factor analysis for learning analytics. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8776–8780 (2013)
15. Cen, H., Koedinger, K.R., Junker, B.: Learning factors analysis – a general method for cognitive model evaluation and improvement. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 164–175. Springer, Heidelberg (2006)
16. Lindsey, R.V., Khajah, M., Mozer, M.C.: Automatic discovery of cognitive skills to improve the prediction of student learning (2014), submitted for publication
17. Bootkrajang, J., Kabán, A.: Label-noise robust logistic regression and its applications. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part I. LNCS, vol. 7523, pp. 143–158. Springer, Heidelberg (2012)
18. Natarajan, N., Dhillon, I., Ravikumar, P., Tewari, A.: Learning with noisy labels. In: Advances in Neural Information Processing Systems, pp. 1196–1204 (2013)
19. Nižnan, J., Pelánek, R., Řihák, J.: Using problem solving times and expert opinion to detect skills. In: Proceedings of the 7th International Conference on Educational Data Mining, pp. 433–434 (2014)
20. Van der Linden, W.: A lognormal model for response times on test items. Journal of Educational and Behavioral Statistics **31**(2), 181 (2006)
21. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
22. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. The Journal of Machine Learning Research **9**, 1871–1874 (2008)