Skills Embeddings: a Neural Approach to Multicomponent Representations of Students and Tasks

Russell Moore*, Andrew Caines*, Mark Elliott*†, Ahmed Zaidi*, Andrew Rice* and Paula Buttery*
ALTA Institute
*Computer Laboratory †Cambridge Assessment
University of Cambridge
{rjm49 | apc38 | mwe24 | ahz22 | acr31 | pjb48}@cam.ac.uk

ABSTRACT

Educational systems use models of student skill to inform decision-making processes. Defining such a model manually is challenging due to the large number of relevant factors. We introduce an alternative approach by learning multidimensional representations (embeddings) from student activity data. Such embeddings are fixed-length real vectors with three desirable characteristics: co-location of similar students and items in a vector space; magnitude increases with skill, and that absence of a skill can be represented. Based on the Multicomponent Latent Trait Model, we use a neural network with complementary trainable weights to learn these embeddings by backpropagation in an unsupervised manner. We evaluate using synthetic student activity data that provides a ground-truth of student skills in order to understand the impact of number of students, question items and knowledge components in the domain. We find that our data-mined parameter values can recreate the synthetic datasets up to the accuracy of the model that generated them, for domains containing up to 10 simultaneously active knowledge components, which can be effectively mined using relatively small quantities of data (1000 students, 100 items). We describe a procedure to estimate the number of components in a domain, and propose a component-masking logic mechanism that improves performance on high-dimensional datasets.

Keywords

knowledge representation, skills embeddings, multicomponent latent trait model

1. INTRODUCTION

Intelligent tutoring systems (ITS) are required to make decisions about which tasks to present to which students. To this end they should be equipped with comprehensive and accurate student ability models to inform the decision-making process. Histories of student activity on ITS platforms are a rich source of information for such models, which could be constructed from hand-crafted feature extraction from the activity logs or alternatively from unsupervised data mining methods. We explore the latter approach, learning multidimensional representations from simulated student data. We refer to these representations as SKILLS EMBEDDINGS, after *word embeddings* – fixed-width vector representations of words and language constructs that have allowed dramatic advances in the natural language processing field [10, 14].

With word embeddings, the notion of semantic similarity between words is captured by proximity within a latent vector space. Thus 'boat' should be closer to 'car' than 'politics' in this latent space, based on natural language data. We transfer the vector space idea to skills embeddings. As with word embeddings, skills embeddings are fixed-width vectors of real numbers, with some specific characteristics that our representations should exhibit:

- Skills embeddings are **co-proximal** in the vector space if they represent entities comprising similar skills.
- Embedding **magnitude** grows with skill – specifically, a higher skill level should entail a larger value within the embedding.
- It should be possible to represent the special case where a skill is absent. We will refer to this characteristic as **skill masking**.

Additionally, we would ideally like to be able to **detect dimensionality**: that is the number and dependency structure of skills within the domain being learned should not need to be specified in advance.

In this work, we provide the design for an artificial neural network – based on the Multicomponent Latent Trait Model [22] – whose weights take the values of our embeddings. We attempt to demonstrate our three desired characteristics in embeddings built using this network, by training it on synthetic student activity datasets. Synthetic data allows us to systematically vary the number of students, items (questions), and knowledge components in the domain. We demonstrate that embeddings display the three characteristics listed above – co-proximity, magnitude and skills masking – and present a procedure to cater to the fourth (dimensionality detection).

Skills embeddings are designed to be multi-purpose 'profiles', to help provide richer information for decision-making

in educational ITS. Objective multi-dimensional representations of student abilities and the skills required for items, can be used as the foundation for task-selection policy optimisation in technology-supported education scenarios.

2. BACKGROUND

Our work relies on some core principles about the nature of knowledge domains, the way that student ability and item difficulty interact, and the idea that knowledge acquisition can be traced in student activity logs [9].

2.1 Knowledge components

For any given educational domain, such as physics, mathematics or language learning, we can break domain-specific knowledge down into atomistic units known as KNOWLEDGE COMPONENTS (KCs), as defined by Koedinger *et al.* [8]:

> ...an acquired unit of cognitive function or structure that can be inferred from performance on a set of related tasks. [...] As a practical matter, we use 'knowledge component' broadly to generalise across terms for describing pieces of cognition or knowledge, including production rule, schema, misconception, or facet, as well as everyday terms, such as concept, principle, fact, or skill.

For the purposes of this work, we think of 'knowledge component' as being synonymous with 'skill' where the skill is extrinsically irreducible – that is, if skill C comprises subskills A and B, and these cannot be broken down further, we do not represent C directly: we represent A and B and treat their co-occurrence as the pattern for C.

We therefore think of a subject domain as a set K of KCs to be acquired. In our datasets, we vary the number of components from 1 to 100.

2.2 Rasch model

The Rasch item response model [19] is a well-known formulation for the success probability of student s attempting item (question) i, derived by transforming the difference between student ability θ_s and item difficulty β_i through a sigmoid function. That is, the probability of success is given by:

$$Pr(X_{si} = 1 \mid \theta_s, \beta_i) = \sigma(\theta_s, \beta_i) \tag{1}$$

– where σ is the sigmoid (here a logistic curve, see Figure 1):

$$\sigma(\theta_s, \beta_i) = \frac{1}{1 + \exp\left(-(\theta_s - \beta_i)\right)} \tag{2}$$

Note that in this model, an evenly-matched student-item pair implies $\theta = \beta$ and so entails a pass-rate of 0.5. The Rasch model assumes a single dimension of proficiency and

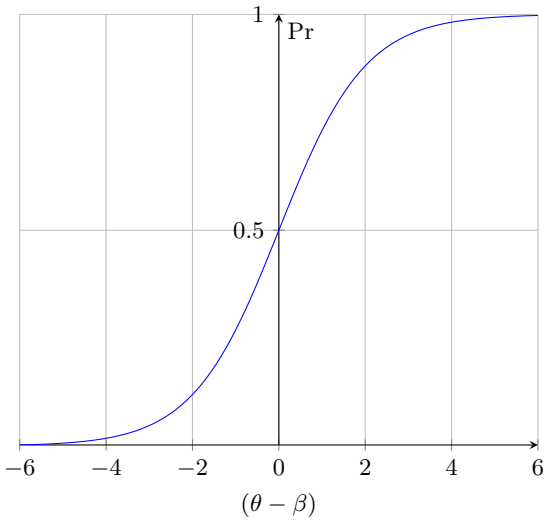

Figure 1: Logistic sigmoid function

embodies *invariant comparison* – this means the student parameter θ can be eliminated algebraically during estimation of the item parameters β, and vice versa [15, 24]. This principle allows Rasch items (and by extension our embeddings) to stand alone as objective representations, independent of the conditions in which they were measured.

2.3 Multicomponent Latent Trait Model

The Rasch model can be extended to the MULTICOMPONENT LATENT TRAIT MODEL (MLTM) of Whitely [22]. Here, the scalars θ and β are replaced by vectors, and the result is a product of sigmoids. The formulation is as follows:

$$Pr(X_{si} = 1 \mid \boldsymbol{\theta_s}, \boldsymbol{\beta_i}) = \prod_{k \in skills(i)} \sigma(\theta_{sk}, \beta_{ik}) \tag{3}$$

Hence the act of student s successfully passing item i is modelled as the conjunction of successes at each of the item's problem-solving steps (denoted k). The probability is given by the product of the probabilities of completing the steps and each step behaves as a Rasch model whose parameters are the corresponding elements of $\boldsymbol{\theta_s}$ and $\boldsymbol{\beta_i}$.

2.4 Item calibration

Traditionally, item calibration with Rasch-type models is carried out using the Birnbaum iteration [3]. However, the Birnbaum algorithm is one-dimensional and to the best of our knowledge has not been extended to multiple dimensions. Moreover, the Rasch approach does not readily allow for the absence of skills: parameters would have to be set to -∞, which is impractical for data mining, particularly in cases where the skill is absent both from a question and student's representations[1]

2.5 Q-Matrix

To explicitly model the presence or absence of skills, we turn to the idea of the Q-MATRIX [18, 2], a binary- or probability-valued matrix that describes which knowledge components (for domain K) are relevant to particular tasks. This has been used previously, for instance, in the Linear Logic Test Model [23]. The matrix is formulated so that each column of $\mathbf{Q}$ represents a task item i, and each row a component k.

The (binary) Q-matrix for a curriculum of items is as follows:

$$q_{ik} = \begin{cases} 1 & \text{if } k \in skills(i), \\ 0 & \text{else;} \end{cases} \tag{4}$$

The Q-matrix for students is represented similarly, with rows indexed by student s and with the row's values representing the student's abilities in the domain at the given time.

3. DATA

The configuration used in this work emulates a *summative* assessment environment, such as a test. In this environment, the item bank represents the test and each student is required to attempt each question in the bank. Each student has one attempt per item, and the result is dichotomous: the response is either right (X=1) or wrong (X=0).

Student activity data is synthesised using a statistical model whose probability mass function (pmf) is just the MLTM model given in equation (3).

Values for the outcome x_{si} (the attempt of student s on item i) are determined by synthesised ground-truth values of the MLTM parameters, indicated by asterisks: $\boldsymbol{\theta}_{\boldsymbol{s}}^{*}$ and $\boldsymbol{\beta}_{\boldsymbol{i}}^{*}$. These are the target parameters we hope to recover from the dichotomous outcome data – that is, we want our embeddings to converge on these values.

The elements of $\boldsymbol{\theta}_{\boldsymbol{s}}^{*}$ and $\boldsymbol{\beta}_{\boldsymbol{i}}^{*}$ themselves are generated uniformly randomly for each student and item – their minimum and maximum values are chosen by an earlier randomised search process, that looks for suitable bounds to generate a balanced dataset given a specific dimensionality $|K|$.

We generated datasets with dimensionalities $|K| \in \{1, 2, 5, 10, 100\}$. We also created datasets where only a subset of components are active – for these datasets, the active components are chosen at random.

[1]Even assuming infinite arithmetic is allowable, if θ and β are both -∞ then $(\theta - \beta) = 0$ and the probability of success is calculated as 0.5; in fact, for an unrequired skill, it should always be 1, since an unrequired step is always 'passed'.

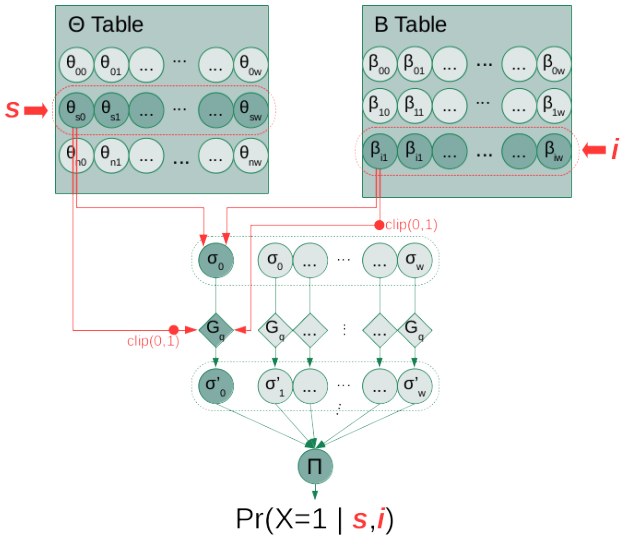

Figure 2: Neural network architecture

4. IMPLEMENTATION

In this section we describe the neural network implementation used, including its design features, software used, and the training regimen applied.

4.1 Neural network architecture

The neural network used in this work is essentially a binary classifier. In a normal supervised learning task, the classifier would take as input some set of features $\Phi(x)$ and a class label C and would learn the probability that a datapoint x is in a given class: $Pr(x \in C \mid \Phi(x))$.

However, in the embedding generation task there is no notion of the feature mapping Φ. Instead we have datapoints of the form $(s, i, pass \in \{T, F\})$, describing whether student s passed item i. Inputs to the neural network then, are s and i, and the class label is $pass$.

The 'features' themselves are learned internally, with two distinct sets of trainable weights. One set of weights ($\Theta \in \mathbb{R}^{|S|\times|K|}$) represents the students, the other ($B \in \mathbb{R}^{|I|\times|K|}$) the items. Whenever s occurs in a datapoint, the weights for s (synonymous with $\boldsymbol{\theta}_{\boldsymbol{s}}$) are selected from the table, and the same happens for $\boldsymbol{\beta}_{\boldsymbol{i}}$ when i occurs.

The weights are fed into a locally connected layer that represents the components of the MLTM. Each unit in the layer performs a sigmoid (logistic) operation to generate a per-component probability. The component probabilities are then multiplied together to get the overall output probability. This is scored against the true $pass$ value using a loss function, and the error is backpropagated to the weights tables.

Since the weights are re-used whenever s or i appear in a datapoint, they converge onto values which best fit the train-

ing data. These trained weights can then be used directly as our embedding values.

The widths of the weights tables (and subsequent layers) determines the fixed width of the embedding that will be generated. Both tables must be the same width, as each student component must have a corresponding item component.

The architecture is illustrated in Figure 2. In this diagram the connections are shown only for a single component, to avoid clutter. Each component functions in parallel in the same way. Note that the diamonds on the diagram represent Q-gates. These are trainable logic components which we will now describe.

4.1.1 Q-gates – logic for absent components

In high-dimensional domains, it is often the case that an item will not possess every skill component. For instance, in both assessment and instruction, questions are usually designed to focus particularly on a subset of skills.

As already described, the Q-matrix is a table of values that represents the presence or absence of components. In our implementation, rather than iterating through $skills(k)$, components are switched on and off using their Q-matrix values per student, q_{sk}, and per item, q_{ik}:

$$Pr(X_{si} = 1 \mid \boldsymbol{\theta_s}, \boldsymbol{\beta_i}) = \prod_{k \in K} G_q(q_{sk}, q_{ik}, \sigma(\theta_{sk}, \beta_{ik})) \quad (5)$$

Where G_q is a *Q-gate*, a ternary logic gate – related to implication – with the following truth table:

q_{ik}	q_{sk}	G_q
1	1	$\sigma(\theta_{sk}, \beta_{ik})$
1	0	0
0	1	1
0	0	1

Q-gates are implemented as part of the neural network, so that they modify the component-level sigmoid outputs:

$$G_q(q_{sk}, q_{ik}, \sigma(\theta_{sk}, \beta_{ik})) = \sigma(\theta_{sk}, \beta_{ik}) q_{ik} q_{sk} + (1 - q_{ik}) \quad (6)$$

The values for q_{ik} and q_{sk} are learned via backpropagation during training. These *q-values* can either be stored in their own set of network weights, or represented by special values in the elements of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. In this work, we use the latter technique – we clip element values to [0,1] to get the q-value. Elements at zero represent 'skill absent'. Elements with values ≥ 1 represent active skills.

4.2 Co-proximity and magnitude

We can force our embedding values into an all-positive vector space by allowing only positive component values: since the Rasch model works on component differences, we simply need to shift all component locations until they are greater than zero. The property of monotonically growing magnitude follows according to the Pythagorean theorem.

We use *weight clipping* in the neural network to do this: weights may only take positive values in $[0, W]$, for some large W. Small weight values in $[0, 1)$ are in the Q-gate activation region and are treated as absent. Larger weights, in $[1, W]$ behave as normal Rasch parameters.

4.3 Training

The network was trained with a categorical cross-entropy loss function using the Adam optimiser [7]. Generally, training is fast and a learning rate α between 0.01 and 0.1 is acceptable.

From all instances in the training set, 10% were randomly chosen for validation and to trigger early-stopping on $loss_{val}$ with $patience = 10$ (*i.e.* we wait for a better value for ten more epochs before quitting, keeping our best weights).

Weights initialisation matters, and a uniformly random initialisation in $[\theta_{min}, \theta_{max}]$ for students and $[\beta_{min}, \beta_{max}]$ for items was found to be most effective.

The software used in this work was implemented in `Python 3.6` using `Keras` [5] with a `TensorFlow` [1] back-end, and `scikit-learn` [13] for the machine learning components.

5. EVALUATION

In this section we describe the steps taken to assess the performance of our approach, in terms of our desired embedding characteristics.

5.1 Prediction agreement

We attempt to recreate the original datasets by using our embeddings $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\beta}}$ to seed our statistical MLTM model. We then score correlation and agreement between the outputs of the original process (seeded with targets $\boldsymbol{\theta}^*$ and $\boldsymbol{\beta}^*$) and the embedding-seeded process.

Since our dataset is synthetic, we can directly observe the outcome probabilities before we stochastically determine the outcome values. Hence we can measure correlation between these and the predicted probabilities from the embedding-seeded version of the dataset. We use Pearson's correlation, which is defined as:

$$\rho_{X,Y} = \frac{cov(X, Y)}{sd(X), sd(Y)} \quad (7)$$

Where X and Y are continuous variables (in this case the real and predicted probabilities of a pass), $cov(\cdot, \cdot)$ is covariance and $sd(\cdot)$ is standard deviation.

In terms of the observed outcomes themselves, because the generator process is stochastic, there will always be some element of chance in the results. We use *Cohen's Kappa* to get a measure of the agreement beyond chance. Where N is the number of datapoints, n_{agreed} is the observed agreement between both runs, and $n_{(k;seed)}$ is the number classed as

category k by the model seeded with *seed*, this gives the following formulation:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \text{ where } \begin{cases} p_o = n_{agreed}/N \\ p_e = \frac{1}{N^2} \sum_{k=\{T,F\}} n_{(k;\boldsymbol{\theta}^*\boldsymbol{\beta}^*)} n_{(k;\hat{\boldsymbol{\theta}}\hat{\boldsymbol{\beta}})} \end{cases} \tag{8}$$

5.2 Co-proximity

To assess our co-proximity requirement, we measure Euclidean distance from the aligned embedding $\hat{\boldsymbol{\theta}}$ (or $\hat{\boldsymbol{\beta}}$) to its target, $\boldsymbol{\theta}^*$ (or $\boldsymbol{\beta}^*$). We compare this to the mean distance from other vectors in the corresponding space, and test them for significance to show that co-proximity to target is not merely by chance.

5.3 Magnitude

To assess magnitude-growth-with-skill, we once again use Pearson correlation (equation 7) at the component level between the elements of $\hat{\boldsymbol{\theta}}$ (or $\hat{\boldsymbol{\beta}}$) and those of target $\boldsymbol{\theta}^*$ (or $\boldsymbol{\beta}^*$). A strong correlation shows that these values grow together as desired.

5.4 Aligning the components

Embeddings are identifiable only up to the ordering of the components due to multiplicative commutivity. Concretely, this means columns in the Θ-Table will align with columns in the B-Table (since they are trained in unison), but these columns may not align with our original target vectors (where the elements may be in a different permutation).

If we wish to visualise the data, or calculate deviations from the real parameters, we must first align the predicted components. A hill-climbing algorithm can be used to do this, minimising the per-column root mean squared error between the predictions and the true values. Although this is not guaranteed to find the optimal alignment of columns, in practice it is a quick and effective means to do so.

The mean absolute parameter errors are given as θ_{MAE} and β_{MAE} in Table 5. Once the components are aligned, the embeddings can be plotted, after a dimensionality reduction step such as PCA or t-SNE if there are more than two dimensions in the domain.

6. RESULTS

6.1 All-active data fit

The main fit statistics for datasets with all-active components are given in Table 1. All datasets were fitted with 1000 students and 100 items.

We give three separate measures of fit – raw accuracy and Cohen's κ to measure agreement between generated datasets, and Pearson's correlation ρ to measure agreement between the underlying probabilities. The model-to-self measures (in brackets) for Acc and κ show us how well the original statistical model (*i.e.* seeded with the true values we are attempting to mine) is able to recreate its own data – this indicates the level of stochastic noise in the target dataset.

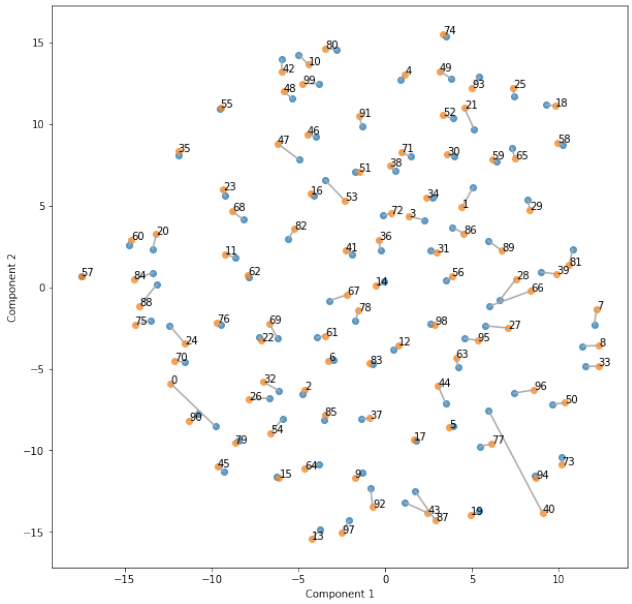

Figure 3: t-SNE visualisation of embeddings in a 100-dimensional space, all active, with 100 students and 1000 items (orange=embedding, blue=target).

We see that model-to-self accuracy drops from 0.844 to 0.651 as the domain dimensionality increases, but that this drop is more pronounced (0.684 to 0.310) when we consider κ, implying that much of the accuracy score is down to chance. This makes sense since the datasets were generated to be well-balanced.

Kappa then, is the more useful score of agreement. We are able to see that at low to moderate dimensions (1a to 10a), κ values closely match, but for the high dimension model (100a) the embeddings do not even achieve half of the model's own agreement, although it is still better than chance ($\kappa = 0$).

Pearson's correlation (on the underlying probabilities, and therefore unaffected by stochastic noise) is very good (>0.9) for the lower dimensionality models (1a to 10a), but drops to 0.44 at high dimensionality (100a). Again this shows a weaker, but present, correlation at high dimensions.

As an illustration, the mined values for a 10-dimension knowledge domain with 100 students and 1000 items ($|K|$=10, $|S|$=100, $|I|$=1000) are shown in Figure 3 using t-SNE visualisation [20]. The correlation between predicted (orange) and target (blue) points is readily visible.

6.2 Q-gated data fit

The main fit statistics for the Q-gated datsets are given in Table 2. The columns and dataset sizes are the same as above, but these datasets are Q-gated, so that components can be switched on and off. The number of active dimensions is given in brackets, so '10 (1-5)' means a domain size |K|=10 but with one to five active components per item.

In the lowest dimensional dataset (2q2), accuracy is lower

Table 1: Accuracy, Cohen's κ agreement and Pearson's correlation (ρ) for all-active datasets. Model-to-self scores show how well the original dataset generator agrees with itself between runs, giving an upper limit to the score.

Model	$\|K\|$-dims (active)	Acc (model-to-self)	κ (model-to-self)	Pearson's ρ
1a	1	0.844 (0.844)	0.680 (0.684)	0.994 (p<0.01)
2a	2	0.776 (0.777)	0.550 (0.556)	0.986 (p<0.01)
5a	5	0.732 (0.742)	0.429 (0.439)	0.961 (p<0.01)
10a	10	0.721 (0.733)	0.425 (0.463)	0.930 (p<0.01)
100a	100	0.572 (0.651)	0.151 (0.310)	0.442 (p<0.01)

Table 2: Accuracy, Cohen's κ agreement and Pearson's correlation (ρ) for Q-gated datasets. Model-to-self scores show how well the original dataset generator agrees with itself between runs, giving an upper limit to the score.

Model	$\|K\|$-dims (active)	Acc (model-to-self)	κ (model-to-self)	Pearson's ρ
2q2	2 (1-2)	0.801 (0.806)	0.576 (0.587)	0.987 (p<0.01)
10q3	10 (1-3)	0.771 (0.802)	0.561 (0.588)	0.943 (p<0.01)
10q5	10 (1-5)	0.822 (0.832)	0.503 (0.522)	0.942 (p<0.01)
100q5	100 (1-5)	0.732 (0.821)	0.509 (0.555)	0.754 (p<0.01)

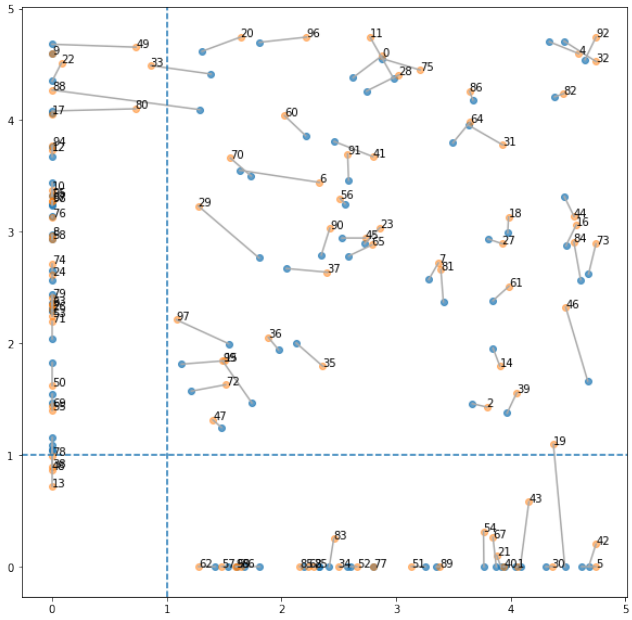

Figure 4: Visualisation of embeddings in a 2-dimensional space, 1000 students and 100 mixed items (50 with two components, 50 with one component). The dotted lines are the Q-gate thresholds below which a respective skill behaves as if inactive.

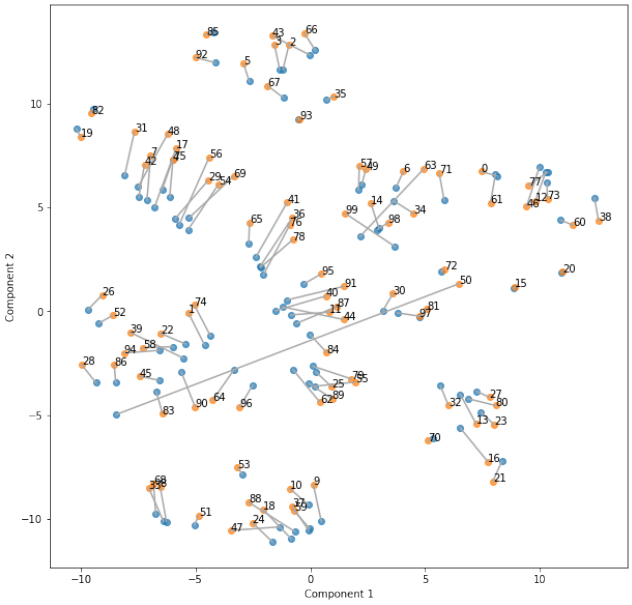

Figure 5: t-SNE visualisation showing 100 items, calibrated from 1000 students. Here, only 1-3 dimensions (from a total 10) are active for any item. The other dimensions are switched off using Q-gate neural logic.

than the non-Q-gated counterpart (2a) for both model-to-self ($0.806 < 0.844$) and embeddings-to-model ($0.801 < 0.844$) and this is also reflected in the corresponding κ values. The higher dimensions (10q3, 10q5, 100q5) show the opposite relationship, with higher accuracy and κ throughout.

Pearson's correlation is comparable to the non-Q-gated data, except for high dimensions (100q5) where the Q-gated dataset achieves a 70% greater correlation score.

Figure 4 shows the mined values for a 2 KC domain, with 1 or 2 active components per item ($|K|$=2, active=[1,2], $|S|$=1000, $|I|$=100). In this case we use Q-gate switching logic, allowing components to be switched on/off. In the plot we add horizontal and vertical dotted lines to indicate the thresholds below which the Q-gate treats skills as inactive.

Figure 5 shows the mined values for a 10 KC domain, with 1 to 3 active components per item ($|K|$=10, active=[1,3], $|S|$=100, $|I|$=1000) and Q-gate switching logic. Here the

Table 3: Mean Euclidean distances from embedding $\hat{v}$ to target v vs other embeddings *(Welch's t-test, df=98, p<0.01)

Dims	$D(\hat{v}, v)$ (σ)	$D(\hat{v}, others)$ (σ)	t^*
1	0.11 (0.09)	3.66 (0.65)	-36,9,
2	0.79 (0.38)	3.74 (0.80)	-33.1
2 (1-2)	0.24 (0.17)	4.76 (0.71)	-61.2
5	2.51 (1.36)	8.63 (1.24)	-33.0
10	13.56 (3.32)	16.47 (1.06)	-8.26
10 (1-3)	0.85 (0.63)	9.18 (1.49)	-51.2
10 (1-5)	1.12 (0.84)	10.53 (1.48)	-54.9
100	79.31 (5.40)	88.30 (2.18)	-15.0
100 (1-5)	1.13 (0.84)	10.53 (1.48)	-54.0

clustering of target points (blue) are more pronounced than in the all-active data (Figure 3) and we can observe how the embeddings form clusters (orange) close to their targets. This clustering is enabled by the Q-gate mechanism.

6.3 Co-proximity and Magnitude

Pertaining to co-proximity of embeddings to target, Table 3 gives for all models the mean Euclidean distance of embedding to target, alongside the equivalent mean distance to other points in the dataset. Mean and standard deviations are given along with Welch's t-test results. For all-active component models, in absolute terms, |K|=100 displays the largest different (8.99, a proportion of 88.30/79.31=1.11), but K=|1| has the largest proportional difference (3.55, a proportion of 3.66/0.11 = 33.3). The Q-gated versions of these models fare better than their un-Q-gated counterparts, with improvements in absolute difference of 1.57 for |K|=2, 0.60 for |K|=10 active=[1,3], 5.42 for |K|=10, active=[1,5] and 6.50 for |K|=100. These correspond to proportional improvements of 4.18, 8.38, 8.89 and 7.74 respectively. All results are significant at p<0.01.

Pertaining to magnitude-growth with skill, Table 4 gives Pearson's correlation for true and mined parameter values across different dimensionalities. High correlation between true and mined parameters indicates that the embedding magnitude grows with skill level. Correlation is high (>0.90) for low to moderate dimensionality (1-10) but low (0.21) although still positive for high dimensionality (100).

Table 4: Component-level Pearson's correlation ρ between true and mined parameters

Dims	Component-level ρ
1	>0.99 (p<0.01)
2	>0.98 (p<0.01)
5	0.93 (p<0.01)
10	0.90 (p<0.01)
100	0.21 (p<0.01)

6.4 Aligned parameter fit

Table 5 gives mean absolute error (MAE) between true and aligned embedding parameter values for all model dimensionalities. The MAE worsens as dimensionality and thus the number of model parameters (Par) increases (from 0.29 and 0.11 for |K|=1, to 7.25 and 6.36 for |K|=100). At higher dimensionality, models with Q-gated components (10 (1-3), 10 (1-5) and 100 (1-5)) show a drop in MAE for both parameters compared to their non-Q-gated counterparts.

Table 5: Mean absolute error in parameter component values (from target) at growing dimensionality

Dims	Par.	θ_{MAE}	β_{MAE}
1	1100	0.29	0.11
2	2200	0.55	0.46
2 (1-2)	2200	0.58	0.41
5	5500	0.94	0.79
10	11000	4.61	3.42
10 (1-3)	11000	1.30	0.13
10 (1-5)	11000	1.33	0.19
100	110000	7.25	6.36
100 (1-5)	110000	3.09	0.27

7. DISCUSSION

Here we discuss the results of these experiments on the unsupervised learning of skills embeddings, addressing in order the characteristics of co-proximity, magnitude, skill masking, and dimensionality detection.

7.1 Co-proximity

The idea of co-proximity can be derived from the definition of a vector. Vectors with equal elements are deemed to be equal: that is, exactly co-proximal in their vector space. More importantly for the embeddings is to show that the mined vectors are closely co-proximal to their targets (*i.e.* the vectors they are supposed to represent), more so than to other vectors. Table 3 shows that in all cases the mined vectors have significantly smaller mean distances to their targets than to other vectors in the space. This holds even at higher dimensions. Thus, at least within the scope of these datasets, we can assert that co-proximity (*i.e.* similarity between mined and real vectors) holds.

7.2 Magnitude

Recall the Rasch probability model, which takes as input the difference between student ability and item difficulty $(\theta - \beta)$. We have specified earlier that components in our embeddings are non-negative. Wright and Linacre prove that Rasch models display specific objectivity [24], and so we can consider the θ and β parameters separately and see that they must both increase commensurately with skill. This is borne out in practice by the results in Table 4, which show a high correlation between the true skill levels and those of the parameters (for $|K| \leq 10$). Thus we can be satisfied that the second of our characteristics has been met.

7.3 Skill masking

It is apparent that our embeddings achieve a high correlation with target outcome probabilities for all but the 100-dimension model. Similar patterns can be seen with other scores. Higher error in the vector space translates to markedly reduced model fit.

There are at least two factors at play here: firstly, for large $|K|$, the 'curse of dimensionality' means that all points within

the space appear metrically equidistant from one another. That is, it becomes difficult to determine the distance (and thus similarity) between them.

The second factor – and perhaps the more important, given that we are still apparently able to identify our target – is informational: for an item with 100 active components, a student must achieve a pass-rate of 99.3% *on each component* to get a 50% pass-rate on the item. Components that are extremely easy carry little information, since our expectation that a student would pass them is almost always met. The sigmoid gradient is very shallow in this region ($x = 4.95$) and this makes the parameter values unstable, being very sensitive to stochastic noise in training data. Furthermore, because dichotomous results give us a paucity of detail, we have no way of knowing which component was responsible for a failed attempt. This combination of factors makes for a very difficulty machine learning task.

Fortunately, for the Q-gated datasets (where our neural logic can learn to mask off components) there is a different pattern. These datasets behave more like low-dimensionality data: for instance, take the case of 100 (1-5 active) dimensions, compared to 100 (all active). Not only is the probability correlation better ($0.75 > 0.44$), but component level error is lower ($3.09 < 7.52$ for $\boldsymbol{\theta}$ and $0.27 < 6.36$ for $\boldsymbol{\beta}$). A similar pattern is seen in the 10-dimensional data.

In practice, the ability to mask off certain components is very useful. Although 100 KCs might be a reasonable estimate for domain size $|K|$, one might expect most questions to feature far fewer active skills. For instance, Pardos *et al.* [11] used a question-set with domain size of 105 skills for high-school mathematics and a maximum of three skills per question. Some kind of masking is vital to represent such a curriculum as fixed-width embeddings.

The advantage of Q-gate masking is that the exact number and composition of skills per question, does not need to be known, because it can be learned during the embedding process. For instance, the dimensionality of the embeddings we use, need not exactly match the domain – if we make the embeddings too wide, the Q-gates will switch off excess components. We give a procedure to estimate domain dimensionality next.

7.4 Dimensionality estimation

Although it is not possible to directly detect the dimensionality $|K|$ of a dataset, there is a simple procedure to estimate $|K|$. Embedding models can be trained across a span of candidate values $|K|_{cand}$ within which the true $|K|$ is believed to lie. The accuracy of fit to the dataset should be observed during training. Typically for $|K|_{cand} > K$ accuracy grows faster but reaches a lower maximum before overfit and decline. For $|K|_{cand} < K$ the growth is considerably slower. Often the true value of $|K|$ can be found by inspection within a few epochs. Alternatively, the training process can be allowed to continue and halted by early-stopping.

Figure 6 shows the result of this process for a dataset with $|K| = 5$, $|S| = 1000$, $|I| = 100$. We plot candidate dimensionality values $|K|_{cand} \in \{1, 3, 5, 7, 10\}$ against the maximum accuracy attained on each attempt to fit a model to the dataset. We applied a neural network 10 times for each value of $|K|_{cand}$, and plot the mean maximum accuracy as a line in Figure 6. A peak in the line at the appropriate value of $|K|_{cand} = |K| = 5$ is clearly visible and therefore we propose this procedure for dimensionality detection.

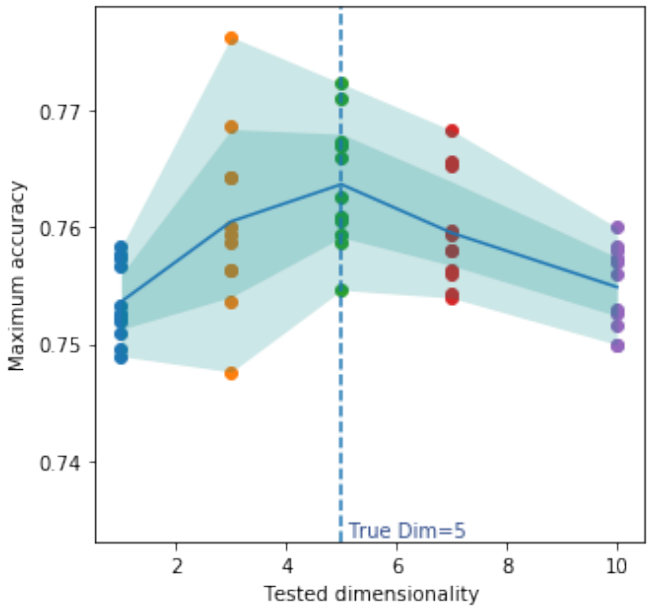

Figure 6: Dimensionality detection: candidate dimensionality values on the x-axis, maximum accuracy for each model fit on the y-axis; ten runs for each dimensionality, with mean maximum accuracy plotted as a line.

8. FUTURE WORK

This work is the beginning of a wider investigation into neural techniques for educational data mining. In this section we outline upcoming work.

8.1 Variant models

We have focussed on the MLTM in this work, but there are several other models for which similar tests can be run. These include the Learning Factors Analysis (LFA) [4] and related Performance Factors Analysis model (PFA) [12] which are multivariate Rasch-type models in which parameters are additive instead of multiplicative.

8.2 Formative assessment

We have worked with static item and student parameters in this work (a summative assessment scenario). To maximise the utility of embeddings, we need to apply them in areas where students are learning, and their embeddings are changing over time. This is a more challenging scenario: it entails modelling learning rates, forgetting, and accounting for interference from outside the system.

The variant models mentioned above support practice-based components, and MLTM can be extended to do so. However, it may be necessary to loosen restrictions on data gathering

– for instance by using polychotomous item responses – allowing us to collect more detailed component-level data in order to mine more complex models.

8.3 Working in higher dimensions

We have seen that masking components (here using Q-gate logic) helps us to deal with high-dimensional vector spaces, that otherwise become problematic. We believe a more thorough analysis of the challenges of training high dimensional models is warranted, since robustness in such models would be useful. Additionally we may be able to use *active learning* – a means of selecting higher-information datapoints [16] – and *attention mechanisms*, in which the neural network has trainable weights that increase the contributions of the most informative datapoints [21]. These two techniques, perhaps combined, could help to increase our information uptake, improving the efficiency of training at higher dimensions.

8.4 Use in reward-based teaching algorithms

One of the key motivations for the embedding approach is its potential for use in reward-based policy generation. In such a setting, a system applies a policy to solve a task, and is rewarded based on some evaluation metric. It then modifies the policy to seek higher reward, a process called *reinforcement learning* [17, 6]. For education, policies might dictate the ordering or pace of delivery of school work. Embeddings have a two-fold use in this setting: because they allow us to objectively compare items and students, we can make policy based on them. We can also link reward from them – for instance by basing reward on magnitude growth, or by rewarding improvements in certain components. This is a rich area for further research.

8.5 Application to real student data

The authors are currently in the process of mining real student activity data, from a major online physics-teaching platform. We believe the embeddings discovered during that process will shed more information on realistic dimensionality, the structure of skills within questions, and help us to study realistic student changes over time. We also intend to report on the interpretability of embeddings by human experts.

9. CONCLUSION

This work introduces a new technique for data mining of multidimensional student and task representations – skills embeddings – using a neural network with complementary sets of weights to perform unsupervised learning of Rasch-type parameters, via a Multidimensional Latent Trait Model formulation.

We applied our data mining model to synthesised student activity datasets in order to learn the appropriate network weights with which to seed the embeddings. This approach is able to extract the underlying parameters in moderately high-dimension data ($|K|$=10) even for small datasets (100 items, 1000 students).

We explained four key desired characteristics for these embeddings: co-proximity of similar objects in vector space, growth of magnitude with skill, applicability in domains of unknown dimension, and ability to model missing skills. Of our desired characteristics, we showed that the embedding technique can directly provide all but the third. We give a procedure to address dimensionality detection, and note that skill masking with Q-gate neural logic provides further mitigation.

We showed that performance, in terms of model agreement with the data, suffers for large numbers of active dimensions (*i.e.* $|K|$=100). We found that skill masking with Q-gates improves this situation, bringing better performance for realistic datasets where only a small subset of skills are active at once. Lastly, we discussed extensions to this idea and outlined our programme of future work.

10. ACKNOWLEDGMENTS

This paper reports on research supported by Cambridge Assessment, University of Cambridge. We thank members of the Isaac Physics team, our colleagues in the ALTA Institute, and the three anonymous reviewers for their valuable feedback.

11. REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] M. Birenbaum, A. Kelly, and K. Tatsuoka. Diagnosing knowledge states in algebra using the rule-space model. *Journal for Research in Mathematics Education*, 24:442–459, 1993.

[3] A. Birnbaum. Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord and M. R. Novick, editors, *Statistical Theories of Mental Test Scores*. Reading, MA: Addison-Wesley, 1968.

[4] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International Conference on Intelligent Tutoring Systems*, pages 164–175. Springer, 2006.

[5] F. Chollet et al. Keras. `https://keras.io`, 2015.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] K. R. Koedinger, A. T. Corbett, and C. Perfetti. The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*, 36(5):757–798, 2012.

[9] K. R. Koedinger, S. D'Mello, E. A. McLaughlin, Z. A. Pardos, and C. P. Rosé. Data mining and education. *Wiley Interdisciplinary Reviews: Cognitive Science*, 6(4):333–353, 2015.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[11] Z. Pardos, N. Heffernan, C. Ruiz, and J. Beck. The composition effect: Conjuntive or compensatory? an analysis of multi-skill math questions in its. In *Educational Data Mining 2008*, 2008.

[12] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger. Performance factors analysis–a new alternative to knowledge tracing. *Online Submission*, 2009.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[15] G. Rasch. On general laws and the meaning of measurement in psychology. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 4, pages 321–333, 1961.

[16] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[17] R. S. Sutton, A. G. Barto, et al. *Introduction to Reinforcement Learning*, volume 135. MIT press Cambridge, 1998.

[18] K. Tatsuoka. Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, 20:345–354, 1983.

[19] W. J. van der Linden and R. K. Hambleton. *Handbook of Modern Item Response Theory*. Springer Science & Business Media, 2013.

[20] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[22] S. E. Whitely. Multicomponent latent trait models for ability tests. *Psychometrika*, 45(4):479–494, 1980.

[23] M. Wilson and P. De Boeck. Descriptive and explanatory item response models. In P. De Boeck and M. Wilson, editors, *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach*. New York: Springer-Verlag, 2004.

[24] B. D. Wright and J. M. Linacre. Dichotomous rasch model derived from specific objectivity. *Rasch Measurement Transactions*, 1:5–6, 1987.