Efficient Feature Embeddings for Student Classification with Variational Auto-encoders

Severin Klingler
Dept. of Computer Science
ETH Zurich, Switzerland
kseverin@inf.ethz.ch

Rafael Wampfler
Dept. of Computer Science
ETH Zurich, Switzerland
wrafael@inf.ethz.ch

Tanja Käser
Graduate School of Education
Stanford University, USA
tkaeser@stanford.edu

Barbara Solenthaler
Dept. of Computer Science
ETH Zurich, Switzerland
sobarbar@inf.ethz.ch

Markus Gross
Dept. of Computer Science
ETH Zurich, Switzerland
grossm@inf.ethz.ch

ABSTRACT

Gathering labeled data in educational data mining (EDM) is a time and cost intensive task. However, the amount of available training data directly influences the quality of predictive models. Unlabeled data, on the other hand, is readily available in high volumes from intelligent tutoring systems and massive open online courses. In this paper, we present a semi-supervised classification pipeline that makes effective use of this unlabeled data to significantly improve model quality. We employ deep variational auto-encoders to learn efficient feature embeddings that improve the performance for standard classifiers by up to 28% compared to completely supervised training. Further, we demonstrate on two independent data sets that our method outperforms previous methods for finding efficient feature embeddings and generalizes better to imbalanced data sets compared to expert features. Our method is data independent and classifier-agnostic, and hence provides the ability to improve performance on a variety of classification tasks in EDM.

Keywords

semi-supervised classification, variational auto-encoder, deep neural networks, dimensionality reduction

1. INTRODUCTION

Building predictive models of student characteristics such as knowledge level, learning disabilities, personality traits or engagement is one of the big challenges in educational data mining (EDM). Such detailed student profiles allow for a better adaptation of the curriculum to the individual needs and is crucial for fostering optimal learning progress. In order to build such predictive models, smaller-scale and controlled user studies are typically conducted where detailed information about student characteristics are at hand (labeled data). The quality of the predictive models, however, inherently depends on the number of study participants, which is typically on the lower side due to time and budget constraints. In contrast to such controlled user studies, digital learning environments such as intelligent tutoring systems (ITS), educational games, learning simulations, and massive open online courses (MOOCs) produce high volumes of data. These data sets provide rich information about student interactions with the system, but come with no or only little additional information about the user (unlabeled data). Semi-supervised learning bridges this gap by making use of patterns in bigger unlabeled data sets to improve predictions on smaller labeled data sets. This is also the focus of this paper. These techniques are well explored in a variety of domains and it has been shown that classifier performance can be improved for, e.g., image classification [15], natural language processing [28] or acoustic modeling [21]. In the education community, semi-supervised classification has been used employing self-training, multi-view training and problem-specific algorithms. Self-training has e.g. been applied for problem-solving performance [22]. In self-training, a classifier is first trained on labeled data and is then iteratively retrained using its most confident predictions on unlabeled data. Self-training has the disadvantage that incorrect predictions decrease the quality of the classifier. Multi-view training uses different data views and has been explored with co-training [27] and tri-training [18] for predicting prerequisite rules and student performance, respectively. The performance of these methods, however, largely depends on the properties of the different data views, which are not yet fully understood [34]. Problem-specific semi-supervised algorithms have been used to organize learning resources in the web [19], with the disadvantage that they cannot be directly applied for other classification tasks.

Recently, it has been shown (outside of the education context) that variational auto-encoders (VAE) have the potential to outperform the commonly used semi-supervised classification techniques. VAE is a neural network that includes an encoder that transforms a given input into a typically lower-dimensional representation, and a decoder that reconstructs the input based on the latent representation. Hence, VAEs learn an efficient feature embedding (feature representation) using unlabeled data that can be used to improve the performance of any standard supervised learning algorithm [15]. This property greatly reduces the need for problem-specific algorithms. Moreover, VAEs feature the advantage that the trained deep generative models are able to produce realistic samples that allow for accurate data imputation and simulations [23], which makes them an appealing choice for EDM. Inspired by these advantages, and the demonstrated superior classifier performance in other domains as in computer vision [16, 23], this paper explores VAE for student classification in the educational context.

We present a complete semi-supervised classification pipeline that employs deep VAEs to extract efficient feature embeddings from unlabeled student data. We have optimized the architecture of two different networks for educational data - a simple variational auto-encoder and a convolutional variational auto-encoder. While our method is generic and hence widely applicable, we apply the pipeline to the problem of detecting students suffering from developmental dyscalculia (DD), which is a learning disability in arithmetics. The large and unlabeled data set at hand consists of student data of more than 7K students and we evaluate the performance of our pipeline on two independent small and labeled data sets with 83 and 155 students. Our evaluation first compares the performance of the two networks, where our results indicate superiority of the convolutional VAE. We then apply different classifiers to both labeled data sets, and demonstrate not only improvements in classification performance of up to 28% compared to other feature extraction algorithms, but also improved robustness to class imbalance when using our pipeline compared to other feature embeddings. The improved robustness of our VAE is especially important for predicting relatively rare student conditions - a challenge that is often met in EDM applications.

2. BACKGROUND

In the semi-supervised classification setting we have access to a large data set $\mathcal{X}_B$ without labels and a much smaller labeled data set $\mathcal{X}_S$ with labels $\mathcal{Y}_S$. The idea behind semi-supervised classification is to make use of patterns in the unlabeled data set to improve the quality of the classifier beyond what would be possible with the small data set $\mathcal{X}_S$ alone. There are many different approaches to semi-supervised classification including transductive SVMs, graph-based methods, self-training or representation learning [35]. In this work we focus on learning an efficient encoding $\mathbf{z} = E(\mathbf{x})$ for $\mathbf{x} \in \mathcal{X}_B$ of the data domain using the unlabeled data $\mathcal{X}_B$ only. This learnt data transformation $E(\cdot)$ - the encoding - is then applied to the labeled data set $\mathcal{X}_S$. Well-known encoders include principle component analysis (PCA) or Kernel PCA (KPCA). PCA is a dimensionality reduction method that finds the optimal linear transformation from an N-dimensional to a K-dimensional space (given a mean-squared error loss). Kernel PCA [24] extends PCA allowing non-linear transformations into a K-dimensional space and has, among others, been successfully used for novelty detection in non-linear domains [11]. Recently, variational auto-encoders (VAE) have outperformed other semi-supervised classification techniques on several data sets [15]. VAE combine variational inference networks with generative models parametrized by deep neural networks that exploit information in the data density to find efficient lower dimensional representations (feature embeddings) of the data.

Auto-encoder. An auto-encoder or autoassociator [2] is a neural network that encodes a given input into a (typically lower dimensional) representation such that the original input can be reconstructed approximately. The auto-encoder consists of two parts. The encoder part of the network takes the N-dimensional input $\mathbf{x} \in \mathbb{R}^N$ and computes an encoding $\mathbf{z} = E(\mathbf{x})$ while the decoder D reconstructs the input based on the latent representation $\hat{\mathbf{x}} = D(\mathbf{z})$. If we train a network using the mean squared error loss and the network consists of a single linear hidden layer of size K, e.g. $E(\mathbf{x}) = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$ and $D(\mathbf{z}) = \mathbf{W_2}\mathbf{z} + \mathbf{b_2}$ for weights $\mathbf{W}_1 \in \mathbb{R}^{K\times N}$ and $\mathbf{W}_2 \in \mathbb{R}^{N\times K}$ and offsets $\mathbf{b}_1 \in \mathbb{R}^K$ and $\mathbf{b}_2 \in \mathbb{R}^N$, the autoencoder behaves similar to PCA in that the network learns to project the input into the span of the K first principle components [2]. For more complex networks with non-linear layers multi-modal aspects of the data can be learnt. Auto-encoders can be used in semi-supervised classification tasks because the encoder can compute a feature representation $\mathbf{z}$ of the original data $\mathbf{x}$. These features can then be used to train a classifier. The learnt feature embedding facilitates classification by clustering related observations in the computed latent space.

Variational auto-encoder. Variational auto-encoders [15] are generative models that combine Bayesian inference with deep neural networks. They model the input data $\mathbf{x}$ as

$$p_\theta(\mathbf{x}|\mathbf{z}) = f(\mathbf{x}; \mathbf{z}, \theta) \qquad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, I) \tag{1}$$

where f is a likelihood function that performs a non-linear transformation with parameters θ of $\mathbf{z}$ by employing a deep neural network. In this model the exact computation of the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is not computationally tractable. Instead, the true posterior is approximated by a distribution $q_\phi(\mathbf{z}|\mathbf{x})$ [16]. This inference network $q_\phi(\mathbf{z}|\mathbf{x})$ is parametrized as a multivariate normal distribution as

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \mathrm{diag}(\sigma^2_\phi(\mathbf{x}))), \tag{2}$$

where $\mu_\phi(\mathbf{x})$ and $\sigma^2_\phi(\mathbf{x})$ denote vectors of means and variance respectively. Both functions $\mu_\phi(\cdot)$ and $\sigma^2_\phi(\cdot)$ are represented as deep neural networks. Hence, variational autoencoders essentially replace the deterministic encoder $E(\mathbf{x})$ and decoder $D(\mathbf{z})$ by a probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and decoder $p_\theta(\mathbf{x}|\mathbf{z})$. Direct maximization of the likelihood is computationally not tractable, therefore a lower bound on the likelihood has been derived [16]. The learning task then amounts to maximizing this variational lower bound

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right], \tag{3}$$

where KL denotes the Kullback-Leibler divergence. The lower bound consists of two intuitive terms. The first term is the reconstruction quality while the second one regularizes the latent space towards the prior $p(\mathbf{z})$. We perform optimization of this lower bound by applying a stochastic optimization method using gradient back-propagation [14].

3. METHOD

In the following we introduce two networks. First, a simple variational auto-encoder consisting of fully connected layers to learn feature embeddings of student data. These encoders have shown to be powerful for semi-supervised classification [15], and are often applied due to their simplicity. Second, an advanced auto-encoder that combines the advantages of VAE with the superiority of asymmetric encoders. This is motivated by the fact that asymmetric auto-encoders have shown superior performance and more meaningful feature representations compared to simple VAE in other domains such as image synthesis [29].

Student snapshots. There are many applications where we want to predict a label y_n for each student n within an ITS based on behavioral data X_n. These labels typically relate to external variables or properties of a student, such

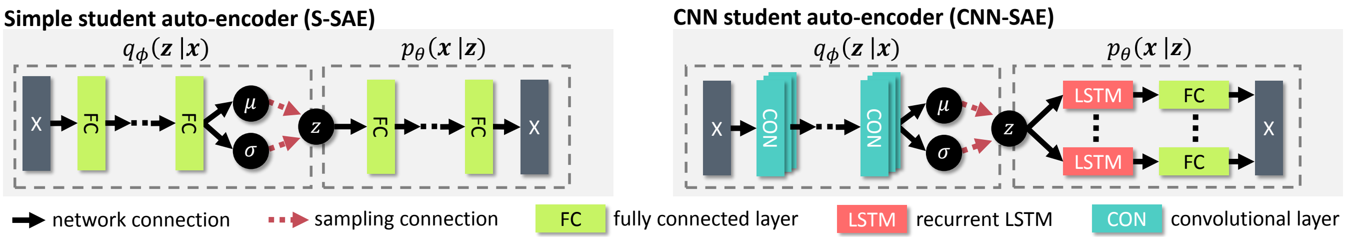


Figure 1: Network layouts for our simple student auto-encoder (left) using only fully connected layers and our improved CNN student auto-encoder (right) using convolutions for the encoder and recurrent LSTM layers for the decoder. In contrast to standard auto-encoders, the connections to the latent space z are sampled (red dashed arrows) from a Gaussian distribution.

as age, learning disabilities, personality traits, learner types, learning outcome etc. Similar to Knowledge Tracing (KT) we propose to model the data $X_n = \{\mathbf{x}_{n1}, \dots, \mathbf{x}_{nT}\}$ as a sequence of T observations. In contrast to KT we store F different feature values $\mathbf{x}_{nt} \in \mathbb{R}^F$ for each element in the sequence, where t denotes the t^{th} opportunity within a task. This allows us to simultaneously store data from multiple tasks in $\mathbf{x}_{nt}$, e.g. $\mathbf{x}_{n1}$ stores all features for student n that were observed during the first task opportunities. For every task in an ITS we can extract various different features that characterize how a student n was approaching the task. These features include performance, answer times, problem solving strategies, etc. We combine this information into a student snapshot $\mathbf{X}_n \in \mathbb{R}^{T\times F}$, where T is the number of task opportunities and F is the number of extracted features.

Simple student auto-encoder (S-SAE). Our simple variational autoencoder is following the general design outlined in Section 2 and is based on the student snapshot representation. For ease of notation we use $\mathbf{x} := \text{vec}(\mathbf{X_n})$, where $\text{vec}(\cdot)$ is the matrix vectorization function to represent the student snapshot of student n. The complete network layout is depicted in Figure 1, left. The encoder and decoder networks consist of L fully connected layers that are implemented as an affine transformation of the input followed by a non-linear activation function $\beta(\cdot)$ as $\mathbf{x}_l = \beta(\mathbf{W}_l\mathbf{x}_{l-1} + \mathbf{b}_l)$, where l is the layer index and $\mathbf{W}_l$ and $\mathbf{b}_l$ are a weight matrix and offset vector of suitable dimensions. Typical choices for $\beta(\cdot)$ include tanh, rectified linear units or sigmoid functions [6]. To produce latent samples $\mathbf{z}$ we sample from the normal distribution (see Equation (2)) using re-parametrization [16]

$$\mathbf{z} = \mu_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x})\epsilon, \tag{4}$$

where $\epsilon \sim \mathcal{N}(0, 1)$, to allow for back-propagation of gradients. For $p_\theta(\mathbf{x}|\mathbf{z})$ (see (1)) any suitable likelihood function can be used. We used a Gaussian distribution for all presented examples. Note that the likelihood function is parametrized by the entire (non-linear) decoder network.

The training of variational auto-encoders can be challenging as stochastic optimization was found to set $q_\phi(\mathbf{z}|\mathbf{x}) = p(\mathbf{z})$ in all but vanishingly rare cases [3], which corresponds to a local maximum that does not use any information from $\mathbf{x}$. We therefore add a warm-up phase that gradually gives the regularization term in the target function more weight:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \alpha\,\text{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right], \tag{5}$$

where $\alpha \in [0, 1]$ is linearly increased with the number of epochs. The warm-up phase has been successfully used for training deep variational auto-encoders [25]. Furthermore, we initialize the weights of the dense layer computing $\log(\sigma^2_\phi(\mathbf{x}))$ to 0 (yielding a variance of 1 at the beginning of the training). This was motivated by our observations that if we employ standard random weight initialization techniques (glorot-norm, he-norm [9]) we can get relatively high initial estimates for the variance $\sigma^2_\phi(\mathbf{x})$, which due to the sampling leads to very unreliable samples $\mathbf{z}$ in the latent space. The large variance in sampled points in the latent space leads to bad convergence properties of the network.

CNN student auto-encoder (CNN-SAE). Following the recent findings in computer vision we present a second, more advanced network that typically outperforms simpler VAE. In [29], for example, these asymmetric auto-encoders resulted in superior reconstruction of images as well as more meaningful feature embeddings. A specific kind of convolutional neural network was combined with an auto-encoder, being able to directly capture low level pixel statistics and hence to extract more high-level feature embeddings.

Inspired by this previous work, we combine an asymmetric auto-encoder (and a decoder that is able to capture low level statistics) with the advantages of variational auto-encoders. Figure 1, right, shows our combined network. We employ multiple layers of one-dimensional convolutions to parametrize the encoder $q_\phi(\mathbf{z}|\mathbf{x})$ (again we assume a Gaussian distribution, see (2)). The distribution is parametrized as follows:

$$\begin{aligned}\mu_\phi(\mathbf{x}) &= \mathbf{W}_\mu\mathbf{h} + \mathbf{b}_\mu\\ \log(\sigma^2_\phi(\mathbf{x})) &= \mathbf{W}_\sigma\mathbf{h} + \mathbf{b}_\sigma\\ \mathbf{h} &= \text{conv}_l(\mathbf{x}) = \beta(\mathbf{W}_l * \text{conv}_{l-1}(\mathbf{x})),\end{aligned}$$

where $*$ is the convolution operator, $\mathbf{W}_l, \mathbf{W}_\mu, \mathbf{W}_\sigma$ are weights of suitable dimensions, $\beta(\cdot)$ is a non-linear activation function and l denotes the layer depth. Further, $\text{conv}_0(\mathbf{x}) = \mathbf{x}$. We keep the standard variational layer (see (4)) while changing the output layer to a recurrent layer using long term short term units (LSTM). Recurrent layers have successfully been used in auto-encoders before, e.g. in [5]. LSTM were very successful for modeling temporal sequences because they can model long and short term dependencies between time steps. Every LSTM unit receives a copy of the sampled points in latent-space, which allows the LSTM network to combine context information (point in the latent

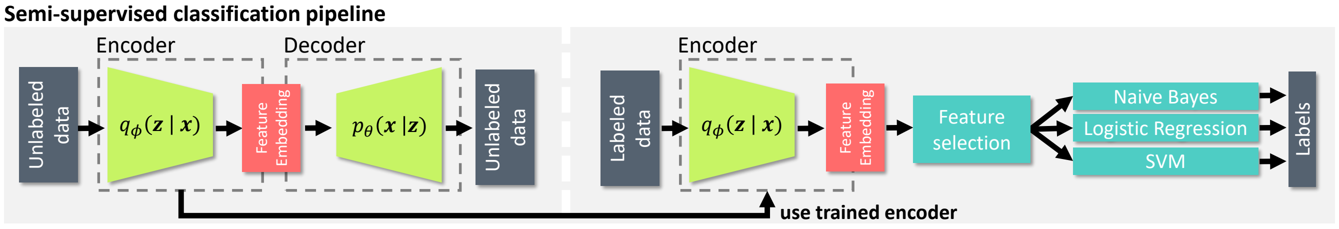


Figure 2: Pipeline overview. We train the variational auto-encoder on a large unlabeled data set. The trained encoder of the auto-encoder can be used to transform other data sets into an expressive feature embedding. Based on this feature embedding we train different classifiers to predict the student labels.

space) with the sequence information (memory unit in the LSTM cell). Using LSTM cells the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ assumes a Gaussian distribution and is parametrized as follows:

$$\mu_{\theta t}(\mathbf{z}) = \mathbf{W}_{\mu z} \cdot \mathrm{lstm}_t(\mathbf{z}) + \mathbf{b}_{\mu z}$$
$$\log(\sigma^2_{\theta t}(\mathbf{z})) = \mathbf{W}_{\sigma z} \cdot \mathrm{lstm}_t(\mathbf{z}) + \mathbf{b}_{\sigma z},$$

where $\mu_{\theta t}(\mathbf{z})$ and $\sigma^2_{\theta t}(\mathbf{z})$ are the t^{th} components of $\mu_\theta(\mathbf{z})$ and $\sigma^2_\theta(\mathbf{z})$, respectively, $\mathrm{lstm}_t(\cdot)$ denotes the t^{th} LSTM cell and $\mathbf{W}_*$ and $\mathbf{b}_*$ denote suitable weight and offset parameters.

Feature selection. VAE provide a natural way for performing feature selection. The inference network $q_\phi(\mathbf{z}|\mathbf{x})$ infers the mean and variance for every dimension z_i. Therefore, the most informative dimension z_i has the highest KL divergence from the prior distribution $p(z_i) = \mathcal{N}(0, 1)$ while uninformative dimensions will have a KL divergence close to 0 [10]. The KL divergence of z_i to $p(z_i)$ is given by

$$KL\left[q_\phi(z_i|\mathbf{x})||p(z_i)\right] = -\log(\sigma_i) + \frac{\sigma_i^2 \mu_i^2}{2} - \frac{1}{2}, \qquad (6)$$

where μ_i and σ_i are the inferred parameter for the Gaussian distribution $q_\phi(z_i|\mathbf{x})$. Feature selection proceeds by keeping the K dimensions z_i with the largest KL divergence.

Semi-supervised classification pipeline. The encoder and the decoder of the variational auto-encoder can be used independently of each other. This independence allows us to take the trained encoder and map new data to the learnt feature embedding. Figure 2 provides an overview of the entire pipeline for semi-supervised classification. In a first unsupervised step we train a VAE on unlabeled data. The learnt encoder $q_\phi(\mathbf{z}|\mathbf{x})$ is then used to transform labeled data sets to the feature embedding. We finally apply our feature selection step that considers the relative importance of the latent dimensions as previously described. We then train standard classifiers (Logistic Regression, Naive Bayes and Support Vector Machine) on the feature embeddings.

4. RESULTS

We evaluated our approach for the specific example of detecting developmental dyscalculia (DD), which is a learning disability affecting the acquisition of arithmetic skills [33]. Based on the learnt feature embedding on a large unlabeled data set the classifier performance was measured on two independent, small and labeled data sets from controlled user studies. We refer to them as *balanced* and *imbalanced* data sets since their distribution of DD and non-DD children differs: the first study has approximately 50% DD, while the second one includes 5% DD (typical prevalence of DD).

4.1 Experimental Setup

All three data sets were collected from *Calcularis*, which is an intelligent tutoring system (ITS) targeted at elementary school children suffering from DD or exhibiting difficulties in learning mathematics [13]. *Calcularis* consists of different games for training number representations and calculation. Previous work identified a set of games that are predictive of DD within *Calcularis* [17]. Since timing features were found to be one of the most relevant indicators for detecting DD [4] and to facilitate comparison to other feature embedding techniques we limited our analysis to log-normalized timing features, for which we can assume normal distribution [30]. Therefore, we evaluated our pipeline on the subset of games from [17] for which meaningful timing features could be extracted and sufficient samples were available in all data sets (we used >7000 samples for training the VAEs). Since our pipeline currently does not handle missing data only students with complete data were included.

Timing features were extracted for the first 5 tasks in 5 different games. The selected games involve addition tasks (adding a 2-digit number to a 1-digit number with ten-crossing; adding two 2-digit numbers with ten-crossing), number conversion (spoken to written numbers in the ranges 0-10 and 0-100) and subtraction tasks (subtracting a 1-digit number from a 2-digit number with ten-crossing). For every task we extracted the total answer time (time between the task prompt until the answer was entered) and the response time (time between the task prompt and the first input by the student). Hence, each student is represented by a 50-dimensional snapshot $\mathbf{x}$ (see Section 3).

Unlabeled data set. The unlabeled data set was extracted using live interaction logs from the ITS *Calcularis*. In total, we collected data from 7229 children. Note that we have no additional information about the children such as DD or grade. We excluded all teacher accounts as well as log files that were < 20KB. Since every new game in *Calcularis* is introduced by a short video during the very first task, we excluded this particular task for all games.

Balanced data set. The first labeled data set is based on log files from 83 participants of a multi-center user study conducted in Germany and Switzerland, where approximately half of the participants were diagnosed with DD (47 DD, 36 control) [31]. During the study, children trained with *Calcularis* at home for five times per week during six weeks and solved on average 1551 tasks. There were 28 participants in 2^{nd} grade (9 DD, 19 control), 40 children in 3^{rd} grade (23 DD, 17 control), 12 children in 4^{th} grade (12 DD) and 3 children in 5^{th} grade (3 DD). The diagnosis of DD was based on standardized neuropsychological tests [31].

Imbalanced data set. The second labeled data set is from a user study conducted in the classroom of ten Swiss elementary school classes. In total, 155 children participated, and a prevalence of DD of 5% could be detected (8 DD, 147 control). There were 97 children in 2^{nd} grade (3 DD, 94 control) and 58 children in 3^{rd} grade (5 DD, 53 control). The DD diagnosis was computed based on standardized tests assessing the mathematical abilities of the children [32, 7]. During the study, children solved 85 tasks directly in the classroom. On average, children needed 26 minutes to complete the tasks.

Implementation. The unlabeled data set was used to train the unsupervised VAE for extracting compact feature embeddings of the data. Based on the learnt data transformations we evaluated two standard classifiers: Logistic Regression (LR) and Naive Bayes (NB). We restricted our evaluation to simple classification models because we wanted to assess the quality of the feature embedding and not the quality of the classifier. More advanced classifiers typically perform a (sometimes implicit) feature transformation as part of their data fitting procedure. To represent at least one model that performs such an embedding we included Support Vector Machine (SVM) in all our results. All classifier parameters were chosen according to the default values in *scikit-learn*. Note that we have additionally performed randomized cross-validated hyper-parameter search for all classifiers, which, however, resulted in marginal improvements only. Because of that, and to keep the model simple and especially easily reproducible, we use the default parameter set in this work. For Logistic Regression we used L2 regularization with $C = 1$, for Naive Bayes we used Gaussian distributions and for the SVM RBF kernels and data point weights have been set inversely proportional to label frequencies. All results are cross-validated using 30 randomized training-test splits on the unlabeled data (test size 5%). The classification part of the pipeline is additionally cross-validated using 300 label-stratified random training-test splits (test size 20%) to ensure highly reproducible classification results.

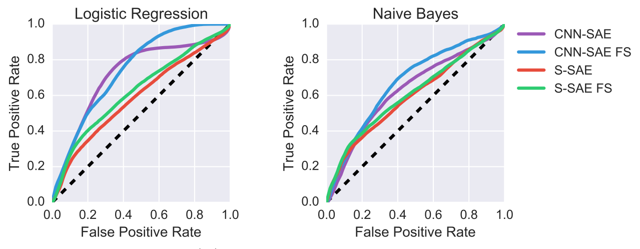
Network hyper-parameters were defined using the approach described in [1]. We increased the number of nodes per layer, the number of layers and the number of epochs until a good fit of the data was achieved. We then regularized the network using dropout [26] with increasing dropout rate until the network was no longer overfitting the data. Activation and weight initialization have been chosen according to common standards: We employ the most common activation function, namely rectified linear activation units (RELU) [20], for all activations. Weight initialization was performed using the method by He et al. [9]. Following this procedure, the following parameters were used for the S-SAE model: encoder and decoders used 3 layers of size 320. The CNN-SAE model was parametrized as follows: 3 convolution layers with 64 convolution kernels and a filter length of 3. We used a single layer of LSTM cells with 80 nodes. We used a batch size of 500 samples and batch normalization and dropout ($r = 0.25$) at every layer. The warm-up phase (see Section 3) was set to 300 epochs. Training was stopped after 1000 (S-SAE) and 500 (CNN-SAE) epochs. The number of latent units was set to 8 in accordance to previous work on detecting students with DD that used 17 features but found that about half of the features were sufficient to detect DD with high accuracy [17]. When feature selection was applied we set the number of features to $K = 4$ and thus we kept exactly half of the latent space features. All networks were implemented using the Keras framework with TensorFlow$^{\text{TM}}$ and optimized using Adam stochastic optimization with standard parameters according to [14].

4.2 Performance comparison

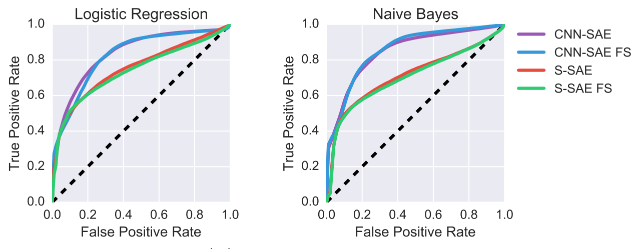
Our VAE models are trained to extract efficient feature embeddings of the data. To assess the quality of these computed feature representations, we compare the classification performance of our method to previous techniques for finding efficient feature embeddings, as well as to feature sets optimized specifically for the task of predicting DD.

Network comparison. In a first experiment we compared the feature embeddings generated by our simple S-SAE and our asymmetric CNN-SAE with and without feature selection. Figure 3 illustrates the average ROC curves of our complete semi-supervised classification pipeline. Our feature embeddings based on asymmetric CNN-SAE clearly outperform the ones from the simple S-SAE on both the imbalanced and the balanced data set for Naive Bayes (NB) and Logistic Regression (LR). For both models, feature selection improves the area under the ROC curve (AUC) for the imbalanced data set (CNN-SAE: LR 4.2%, NB 6.3%; S-SAE: LR 6.8%, NB: 1.6%), but has no effect for the balanced data set. We believe that this is due to the ability of the classifiers to distinguish useful features from noisy ones given enough samples. Since the performance of the classifiers with feature selection (FS) is better or equal to no feature selection in each experiment, we used the CNN-SAE FS model for all further evaluations.

Classification performance. In Figure 4 we compare the classifier performance for different feature embeddings. We compare our method based on VAE to two well-known methods for finding optimal feature embeddings, namely principle component analysis (PCA, green) and Kernel PCA (KPCA, red) [24]. For comparison and as a baseline for the performance of the different methods, we include direct classification results (gray), for which no feature embedding was computed. We used $K = 8$ (dimensionality of feature embedding) for all methods. The features extracted by our pipeline compare favorably to PCA and Kernel PCA showing improvements in terms of AUC of 28% for Logistic Regression and 23% for Naive Bayes on the imbalanced data set and an improvement of 3.75% for Logistic Regression and 7.5% for Naive Bayes on the balanced data set. By using simple classifiers, we demonstrated that our encoder learns an effective feature embedding. More sophisticated classifiers (such as SVM with non-linear kernels) typically proceed by first embedding the input into a specific feature space that is different from the original space.



(a) Imbalanced data set



(b) Balanced data set

Figure 3: ROC curves for the two proposed models with and without feature selection (FS). Our asymmetric CNN-SAE outperforms the simple S-SAE consistently with (blue) and without (purple) feature selection. Feature selection improves performance only on the imbalanced data set.

For the imbalanced data set the overall performance for SVM is significantly lower for all embeddings. This is in line with previous work [12] showing that for imbalanced data sets, the decision boundaries of SVMs are heavily skewed towards the minority class resulting in a preference for the majority class and thus a high miss-classification rate for the minority class. Indeed, we found that SVM predicted only majority labels on the imbalanced data set. For the balanced data set our feature embedding shows improvements of 2.5% over alternative embeddings when using SVM.

Further, Table 1 shows the performance of all feature embeddings using three additional common classification metrics: root mean squared error (RMSE), classification accuracy (Acc.) and area under the precision recall curve (AUPR). We statistically compared the classification metrics of our feature embedding to the best alternative feature embedding using an independent t-test and Bonferroni correction for multiple tests ($\alpha = 0.05$). Our feature embedding significantly outperformed alternative embeddings for all classifiers on both the balanced and imbalanced data sets on most metrics. The main exception was the performance of SVM on the imbalanced data set, which exhibited large variance for all feature embeddings and the worst overall classification performance (compared to the other classifiers).

When comparing classification performance on the imbalanced and the balanced data sets we observed that our pipeline using VAEs showed significant performance improvements compared to other methods for finding feature embeddings. While the unlabeled and the balanced data sets stem from an adaptive version of *Calcularis* the imbalanced data was collected using a fixed task sequence. As our method shows larger improvements on the imbalanced data, we believe CNN-SAE learned an embedding that is robust beyond adaptive ITS. The relative improvements of our feature embeddings is smallest for SVM on the balanced data set. We believe that this is due to ability of the SVM to learn complex decision boundaries given sufficient data. However, the ability for complex decision boundaries renders SVMs more vulnerable to class imbalance, yielding performance at random level on the imbalanced data set.

Comparison to specialized models. Recently, a specialized Naive Bayes classifier (S-NB) for the detection of developmental dyscalculia (DD) was introduced presenting a set of features optimized for the detection of DD [17]. The development of S-NB including the set of features was based on the balanced data set used in this work. In comparison to S-NB, our approach relies on timing data only and the extracted features are independent of the classification task. We compared the performance of S-NB to our CNN-SAE model on both data sets. For the balanced data set we found an AUC of 0.94 for the specialized model (S-NB) compared to an AUC of 0.86 for Naive Bayes using our feature embedding. On the imbalanced data set we found an AUC of 0.67 for S-NB compared to an AUC of 0.77 using Logistic Regression with our feature embedding. These findings demonstrate that while our feature embedding performs slightly worse on the balanced data set (for which the S-NB was developed), we significantly outperform S-NB by 15% on the imbalanced data set, which suggests that our VAE model automatically extracts feature embeddings that are more robust than expert features.

Robustness on sample size. Ideally, a classifier's performance should gracefully decrease as fewer data is provided. A good feature embedding allows a classifier to generalize well based on few labeled examples because similar samples are clustered together in the feature embedding. We therefore investigated the robustness of the different feature representations with respect to the training set size. For this we used the balanced data set where we varied the training set size between 7 (10% of the data) and 62 (90% of the data) by random label-stratified sub-sampling. Figure 5 compares the AUC of the different feature embeddings over different sizes of the training set. In case of Naive Bayes and Logistic Regression our embedding provides superior performance for all training set sizes. For large enough data sets SVM using the raw feature data (Direct, grey) is performing as well as using our embedding (CNN-SAE, blue). However, for smaller data sets starting at 30 samples the performance of SVM based on the raw features declines more rapidly compared to the SVM based on our feature embedding.

5. CONCLUSION

We adapted the recently developed variational auto-encoders to educational data for the task of semi-supervised classification of student characteristics. We presented a complete pipeline for semi-supervised classification that can be used with any standard classifier. We demonstrated that extracted structures from large scale unlabeled data sets can significantly improve classification performance for different labeled data sets. Our findings show that the improvements are especially pronounced for small or imbalanced data sets. Imbalanced data sets typically arise in EDM when detecting relatively rare conditions such as learning disabilities. Im-

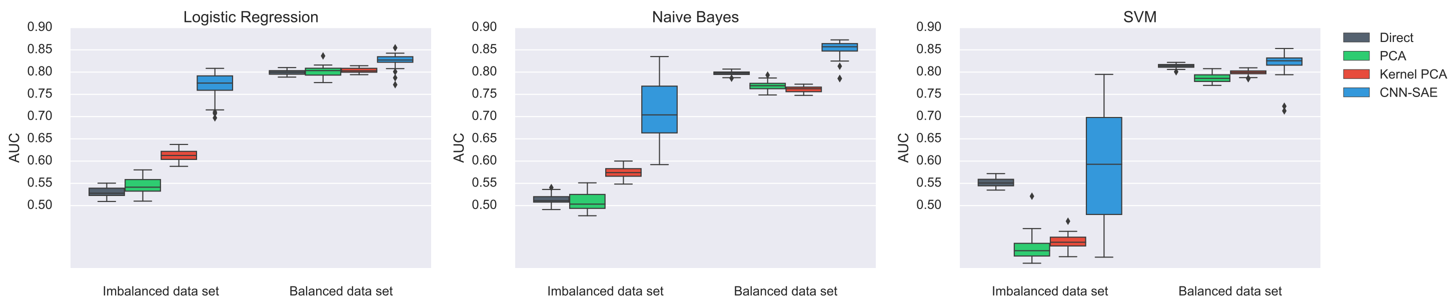


Figure 4: Classification performance for different feature embeddings. Our variational auto-encoder (blue) outperforms other embeddings by up to 28% (imbalanced data set) and by up to 7.5% (balanced data set).

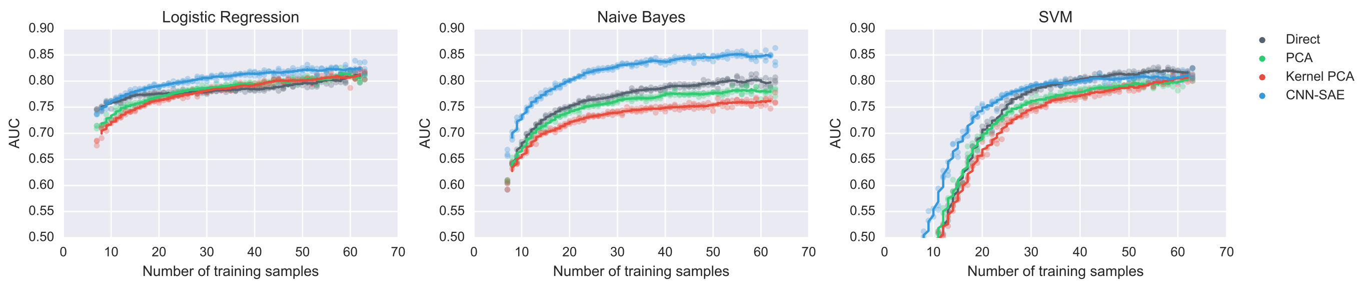


Figure 5: Comparison of classifier performance on the balanced data for different training set sizes (moving average fitted to data points). The features automatically extracted by our variational auto-encoder (blue) maintain a performance advantage even if the training size shrinks to 7 samples (10% of the original size).

Table 1: Comparison of our method to alternative embeddings. Our approach using a variational auto-encoder (CNN-SAE) significantly outperforms other approaches for most cases. The best score for each metric and classifier is shown in bold. *= statistically significant difference (t-test with Bonferroni correction, $\alpha = 0.05$).

	Direct				**PCA**				**Kernel PCA**				**CNN-SAE**			
	AUC	RMSE	AUPR	Acc.	AUC	RMSE	AUPR	Acc.	AUC	RMSE	AUPR	Acc.	AUC	RMSE	AUPR	Acc.
Imbalanced data set																
Logistic Regression	0.53	0.27	0.18	0.91	0.54	0.25	0.17	0.93	0.61	0.25	0.16	0.93	**0.78***	**0.24***	**0.28***	**0.94***
Naive Bayes	0.51	0.29	0.23	0.91	0.50	0.29	0.10	0.90	0.57	0.28	0.20	0.91	**0.70***	**0.25***	**0.24**	**0.93***
SVM	0.55	0.25	**0.22***	0.94	0.40	0.25	0.08	0.94	0.42	0.25	0.09	0.93	**0.59**	0.25	0.16	0.94
Balanced data set																
Logistic Regression	0.80	0.44	0.82	0.73	0.80	0.42	0.84	0.73	0.80	0.42	0.83	0.75	**0.83***	**0.40***	0.84	**0.77**
Naive Bayes	0.80	0.49	0.80	0.73	0.77	0.46	0.77	0.71	0.76	0.46	0.76	0.70	**0.86***	**0.38***	**0.86***	**0.80***
SVM	0.81	0.42	**0.84***	0.75	0.79	0.43	0.81	0.73	0.80	0.43	0.83	0.73	**0.83**	**0.40***	0.81	**0.79***

proved classification results with simple classifiers such as Logistic Regression might indicate that VAEs learn feature embeddings that are interpretable by human experts. In the future we want to explore the learnt representations and compare it to traditional categorizations of students (skills, performance, etc.). Additionally, we want to extend our results to include additional feature types and data reliability indicators to handle missing data. Although we trained our networks on comparatively small sample sizes, the presented method scales (due to mini-batch learning) to much larger data sets (>100K users) allowing the training of more complex VAE. Moreover, the generative model $p_\theta(\mathbf{x}|\mathbf{z})$ that is part of any VAE can be used to produce realistic data samples [29]. Up-sampling of the minority class provides a potential way to improve the decision boundaries for classifiers. In contrast to common up-sampling methods such as ADASYN [8], VAE-based sampling does not require nearest neighbor computations which makes them better applicable to small data sets. Preliminary results for random subsets of the balanced data set showed improvements in AUC by up-sampling based on VAE of 2-3% compared to ADASYN. While we applied our method to the specific case of detecting developmental dyscalculia, the presented pipeline is generic and thus can be applied to any educational data set and used for the detection of any student characteristic.

Acknowledgments. This work was supported by ETH Research Grant ETH-23 13-2.

6. REFERENCES

[1] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. 2012.

[2] Y. Bengio et al. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2009.

[3] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating Sentences from a Continuous Space. In *Proc. CONLL*, pages 10–21, 2016.

[4] B. Butterworth. *Dyscalculia screener*. Nelson Publishing Company Ltd., 2003.

[5] O. Fabius and J. R. van Amersfoort. Variational recurrent auto-encoders. In *Proc. ICLR*, 2015.

[6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[7] J. Haffner, K. Baro, P. Parzer, and F. Resch. Heidelberger Rechentest: Erfassung mathematischer Basiskomptenzen im Grundschulalter, 2005.

[8] H. He, Y. Bai, E. A. Garcia, and S. Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Proc. IJCNN*, pages 1322–1328, 2008.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. ICCV*, pages 1026–1034, 2015.

[10] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*, 2016.

[11] H. Hoffmann. Kernel PCA for novelty detection. *Pattern Recognition*, pages 863–874, 2007.

[12] T. Imam, K. Ting, and J. Kamruzzaman. z-svm: an svm for improved classification of imbalanced data. *AI 2006: Advances in Artificial Intelligence*, pages 264–273, 2006.

[13] T. Käser, G.-M. Baschera, J. Kohn, K. Kucian, V. Richtmann, U. Grond, M. Gross, and M. von Aster. Design and evaluation of the computer-based training program calcularis for enhancing numerical cognition. *Frontiers in Developmental Psychology*, 2013.

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proc. ICLR*, 2015.

[15] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Proc. NIPS*, pages 3581–3589, 2014.

[16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *Proc. ICLR*, 2014.

[17] S. Klingler, T. Käser, A. Busetto, B. Solenthaler, J. Kohn, M. von Aster, and M. Gross. Stealth Assessment in ITS - A Study for Developmental Dyscalculia. In *Proc. ITS*, pages 79–89, 2016.

[18] G. Kostopoulos, S. B. Kotsiantis, and P. B. Pintelas. Predicting Student Performance in Distance Higher Education Using Semi-supervised Techniques. In *Proc. MEDI*, pages 259–270, 2015.

[19] I. Labutov and H. Lipson. Web as a textbook: Curating Targeted Learning Paths through the Heterogeneous Learning Resources on the Web. In *Proc. EDM*, pages 110–118, 2016.

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, pages 436–444, 2015.

[21] H. Liao, E. McDermott, and A. Senior. Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription. In *Proc. ASRU*, pages 368–373, 2013.

[22] W. Min, B. W. Mott, J. P. Rowe, and J. C. Lester. Leveraging semi-supervised learning to predict student problem-solving performance in narrative-centered learning environments. In *Proc. ITS*, pages 664–665, 2014.

[23] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proc. ICML*, pages 1278–1286, 2014.

[24] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *Proc. ICANN*, pages 583–588, 1997.

[25] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Proc. NIPS*, pages 3738–3746, 2016.

[26] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, pages 1929–1958, 2014.

[27] V. Tam, E. Y. Lam, S. Fung, W. Fok, and A. H. Yuen. Enhancing educational data mining techniques on online educational resources with a semi-supervised learning approach. In *Proc. TALE*, pages 203–206, 2015.

[28] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proc. ACL*, pages 384–394, 2010.

[29] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional Image Generation with PixelCNN Decoders. In *Proc. NIPS*, pages 4790–4798, 2016.

[30] W. J. van der Linden. A lognormal model for response times on test items. *Journal of Educational and Behavioral Statistics*, 31(2):181–204, 2006.

[31] M. Von Aster, L. Rauscher, K. Kucian, T. Käser, U. McCaskey, and J. Kohn. Calcularis - Evaluation of a computer-based learning program for enhancing numerical cognition for children with developmental dyscalculia, 2015. 62nd Annual Meeting of the American Academy of Child and Adolescent Psychiatry.

[32] M. von Aster, M. W. Zulauf, and R. Horn. *Neuropsychologische Testbatterie für Zahlenverarbeitung und Rechnen bei Kindern: ZAREKI-R*. Pearson, 2006.

[33] M. G. Von Aster and R. S. Shalev. Number development and developmental dyscalculia. *Developmental Medicine & Child Neurology*, pages 868–873, 2007.

[34] C. Xu, D. Tao, and C. Xu. A survey on multi-view learning. *Neural Comput. Appl.*, pages 2031–2038, 2013.

[35] X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, 2006.