Noname manuscript No.
(will be inserted by the editor)

Clustering Educational Items from Response Data using Penalized Pearson coefficient and deep autoencoders

Khadidja Harbouche · Nassima Smaani · Imene Zenbout

Received: date / Accepted: date

Abstract Educational data mining techniques are very useful to analyze learner performance in purpose to optimize the approach of item-to-skill mapping. Therefore computing a degree of similarity between items using different measures based on the performance of the learner toward items, enhance the clustering of different items into knowledge components. This paper proposes a computational framework to group the elements of the corresponding knowledge component. The first phase of the framework represents a variation of Pearson coefficient to measure item similarity by applying a penalty score that is calculated from the number of hints taken by the learner during solving two items. The second phase applies a dimensionality reduction using deep auto encoders to improve the clustering accuracy. The experimental results show that clustering based on the penalized Pearson coefficient and the deep dimensionality reduction (PPC+DDR) outperforms basic clustering based on different similarity methods , with approximately +0.2 in Mean silhouette coefficient.

Keywords Educational Data mining · Learner model · Machine learning · Deep learning · Item-to-skill mapping · Clustering

K.Harbouche
LRSD laboratory, University Ferhat Abbes Setif 1, El Bez, Setif, Algeria
E-mail: khadidja.harbouche@univ-setif.dz

N.Smaani
University Ferhat Abbes Setif 1, El Bez, Setif, Algeria
E-mail: nassima.smaani@univ-setif.dz

I. Zenbout
Fundamental Informatics and its Applications Department, misc laboratory, Faculty NTIC, University Abdelhamid Mehri - Constantine 2, Constantine, Algeria
E-mail: imene.zenbout@constantine2.dz

1 Introduction

Course curricula are usually organized in a meaningful sequence that evolves from relatively simple lessons to more complex ones. Incorporating prerequisite skill structures into education systems helps identify the order in which concepts should be presented to learners to maximize their success. Many skills have a strong causal relationship in which one skill must be presented and mastered by the learner before another (hierarchy of skills according to prerequisites). To sequence learning in an intelligent tutoring system (ITS), we refer to prerequisite structure as the relationships among skills that place strict constraints on the order in which skills can be acquired. Recent interest in computer assisted education promises large amounts of data from students solving items: questions, problems, parts of questions ... That data are used to create student models. These models represent an estimate of skill proficiency at a given point in time [8]. Student models are often used to personalize instruction in tutoring systems or to predict future student performance. Prior work has investigated how to discover prerequisites among items without considering their mapping into skills[6][3]. Item-to-skill mappings (also called Q-matrices) are desirable because they allow more interpretable diagnostic information. They are standard representation used to specify the relationships between individual test items and target skills. There are two approaches to item-to-skills mapping: Model-based approach, and Similarity-based ones that are based on the assumption that learners will tend to perform similarly on items that require the same skill, so we need to identify similarity between pairs of items. Our work falls into the second approach (Similarity-based). In this paper, we present a PPC-DDR architecture based on: Firstly, a proposed measure, we call penalized Pearson coefficient PPC, to calculate similarity score between two items; and secondly, a deep auto encoder to reduce the dimensionality of the Item to Item similarity matrix (Deep Dimensionality Reduction DDR). A set of experimental results were conducted to evaluate the proposed approach on the corresponding data set.
The rest of the paper is structerd as follow: Section 2, review some related works in educational datamining , Then we descriped, and explained our proposal in section3. We conducted a set of experiments and comparison in section4 to evaluate our proposal. Finally, we concluded our work in section 5 with a general overview and perspectives.

2 Related works to educational items clustering

As mentioned above, item-item similarity is a crucial phase to perform a relevant educational item clustering. Therefore, many researches have been introduced in the literature. where:
Jirí Rihák and Radek Pelánek applied an automatic computation of item similarities based on learners' performance data using different measures such as pearson yule ,jaccad with different setting and different levels .The results

showed that The Pearson correlation coefficient is a good default choice. [12]. Dharaneeshwaran et al Calculated the User-item Similarity using Pearson's and Cosine Correlation by using three different techniques for computing similarities for obtaining a recommendation for them[7]. Pelánek et al proposed a systematic approach to study similarity measures, they made an evaluation of similarity measures for several introductory programming environments[2]. Nazaretsky et al proposed a new item similarity measure named 'Kappa learning' which compute the similarity between items based on the response data giving by the learner , they took in consideration improving the sequence of the knowledge skills [11]. Bjørn et al developed a framework using an artificial neural network model to show the main differences between various types of similarity measures[10]

3 Proposed Architecture

Regrouping items into KC based only on similarity measuring through learner performance correct /incorrect answers, may leads to assigning items to the wrong cluster .Taking the assumption where for two items the answer of the learners were correct yet after asking for a lot of hints.Here if we only based on the correctness ratio we may assign the two items under one cluster yet looking to high hint asking frequency , we can't ignore that the learner was not able to answer the items without taking a considerable set of hints , which open the question about the similarity between the two items , if it is really high as it was computed or not. Illustrating that assumption, lets take the two items $i1$ and $i2$ where $i1 : (7 - 5)$ and $i2 : (5 - 7)$, the two items can be yield in the subtraction knowledge but to achieve the answer of the second item the learner need to have a prerequisites in negative number skill. So, he will ask for aid in order to answer the second item correctly , which reduce the similarity between i1 and i2 , here we propose to take in consideration the number of hints asked by the learner to answer the two items since we don't have a certain order on knowledge retrieval, So , the learner may answer $i1$ than $i2$ or the reverse in order to build a certain skill. Therefor we applied a penalty score on the coefficient similarity between $i1$ and $i2$.

3.1 Penalized pearson

Lets denote $L_{u,i}$ the learner-item matrix , where u represents the learner and i the item. From this matrix we compute the contingency matrix (table 1) of each item i to item j as well as counting the following combination:

- $Nb_{correct}$**:** is the number of the passage of the items i and j with the outcome=correct given by the learner.
- $Nb_{incorrect}$**:** is the number of the passage of the items i and j with the outcome=incorrect given by the learner .

		$item_i$	
		Incorrect	Correct
$item_j$	Incorrect	a	b
	Correct	c	d

Table 1 item-item contingency matrix

- Nb_{hint}**:** number of hint asked by the learner during solving the items i and j
- N**:** total passage on the two items $\sum(Nb_{correct}, Nb_{incorrect}, Nb_{hint})$. Next we define a coefficient λ_{ij} that is calculated following equation1. λ value is between 0 and 1.

$$\lambda = \frac{Nb}{N} \tag{1}$$

The final step is to calculate the item similarity score between i and j. As we mentioned above we proposed a Pearson variation that is based on a penalty on the score between the two items. Taking The Pearson formula as it is defined in (equation 2), after we apply a penalty on it based on λ.

$$Ps = \frac{(a \times d) - (c \times b)}{\sqrt{(a+b) \times (a+c) \times (b+d) \times (c+d)}} \tag{2}$$

When calculating the number of hints the more the learner asks for hints the bigger λ is i.e.λ is closer to one, so multiplying the Pearson score directly to λ will apply a higher penalty(λ closer to zero) on items with a few or no hints. To over come this we multiply the similarity score by $(1 - \lambda)$ to reverse the penalty score. Therefore we obtain the formula in equation3

$$Ps = \frac{(a \times d) - (c \times b)}{\sqrt{(a+b) \times (a+c) \times (b+d) \times (c+d)}} \times (1 - \lambda) \tag{3}$$

The algorithm 1 , explains the steps that we followed in order to calculate the item-item similarity matrix based on the proposed PPC.

3.2 Dimensionality reduction

After constructing the item similarity matrix , it may hold lot of noise due to its high dimensionality , that leads to poor results in clustering items, therefor ye apply a dimensionality reduction using deep auto encoder to learn a new features' representation. Though before proceed to the dimensionality reduction phase we passes through a data preprocessing as follows:
Eliminating irrelevant items: We eliminated items that have a number of missing values that is equal to or more than 40% missing values.
Imputation of missing values: Missing values is a well-known problem in data science that need to be handled because they reduce the quality for any of

Algorithm 1: Penalized Pearson Coefficient to construct item-item similarity

Data: learner-item, a, b,c,d, N, Nb_{hint}, λ
Result: item-item
for *each* $item_i$ *in leaner-item* **do**
  for *each* $item_j$ *in learner-item* **do**
    Initialize to zero (a,b,c,d,N, Nb_{hint});
    for *each learner in learner-item* **do**
      Get $Infoitem_{l,i}$;
      Get $Infoitem_{l,j}$;
      Update(a, c , b , d);
      Update(Nb_{hint});
      Update(N);
    end
    if Nb_{hint}=*0* **then**
      λ=0
    end
    else
      Update lambda (equation1)
    end
    Calculate PPC (equation 3);
    Assign $item - item_{i,j}$
  end
end

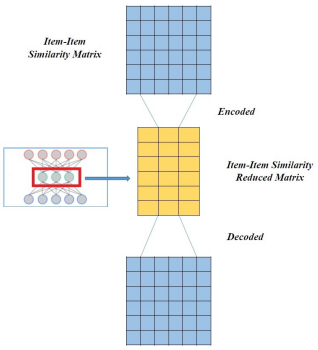


Fig. 1 Dimensionality reduction auto encoder

our performance metrics. We Imputed our data for completing missing values using k-Nearest Neighbors imputer, where Each sample's missing values are imputed using the mean value from k-neighbors found in the dataset. Then we used Z score to normalize it.

3.2.1 Deep Auto encoder

Our architecture is based on Unsupervised learning , where we used deep autoencoders to reduce the dimensionality[1] of item-item matrix for improving

Number of Students/Learners	36
Number of Unique Steps	3,735
Total Number of Steps	24,890
Total Number of Transactions	71,553
Total Student Hours:	334.24
Knowledge Component Model(s):	• Default (7 knowledge components) • Single-KC (1 knowledge component) • Unique-step (2192 knowledge components)

Table 2 Data set statistics

the clustering phase by eliminating any noises data and to learn better descriminative reduced features space.
Let's denote the item-item matrix I of range $M \times N$, each i represents an item that to be further clustered and each j is the item feature used for training a clustering model. The proposed auto encoder is a symmetric deep model where the encoder takes the input X and encoded it into a latent space of range K where $K < M$. The decoder serves to reconstruct the input X as the output X', where $X \approx X'$. The consistency of the autoencoder was evaluated using the mean-square-error (mse) (equation 4), that calculates the difference between X and X'.

$$mse = 1/n \sum_{i=1}^{n} (y_i - y_i')^2 \tag{4}$$

4 Experimental results

In order to evaluate the correctness of our proposal we conducted an experimental scenario on an educational dataset and draw a set of comparison to evaluate the performance of the proposed topology. The deep learning architecture was implemented using keras package [4]

4.1 Data Collection and preparation

From the DataShop, (https://pslcdatashop.web.cmu.edu/) educational data repository we collected the. *'FrenchLanguage2'*, which is a dataset (table 2) corresponds to an e-learning French course. One of the steps in the problem solving pipeline that is of huge importance is data preparation. Missing this step would cause the results of the analysis to be misleading. In this step we have 3 phases :
1.Choice of features that we use during our work : In our work , we focus on learner performance , which it is related to the item,and the outcome provided by the learner : (correct /incorrect) or taking a hint. The description of features is described in table 3 . *2.Elimination of items passed by just one student:*

Feature	Description	Total Number
Anon Student Id	ID of the student	36
Problem name	items	749
Outcome	The response giving by the student (correct/incorrect) Or taking a Hint	—

Table 3 Final dataset description

Parameter	Value
Activation function	tanh
Epochs	400
batch-size	100
Optimizer	sgd
loss	mse (0.5)
encoder	(200,100,50,30)
decoder	(50,100,200)

Table 4 Parameters of deep auto encoder

Before , the construction of the learner-item matrix we have to keep only those items that have been taken at least by one learner(equation5):

$$number_item = M_{item,learner}. \qquad |M_{item_learner} > 0 \tag{5}$$

3.Construction of the Learner-Item matrix : The learner-item matrix contains the counts of how many times a student passed through the same item and the outcome of each transaction whether its correct, incorrect, or asking hint.

4.2 Dimestionality reduction autoencoder training

We construct a deep symmetric autoencoder with an input layer of 737 nodes. Three middle layers with 200, 100 , and 50 nodes respectively, besides a bottleneck layer with 30 nodes that represents the latent space used in clustering. An decoder with three middle layer with 50, 100, 200 nodes respectively and an output layer composed of 737 nodes. The model was trained using a stochastic gradient descent optimizer[14] and mini-batch training. To train our model and select the best parameters settings we used hold-out cross validation and split our data set into 80% training and 20% testing, where we trained the autoencoder on the training set and test it's performance on unseen dataset through thousands of executions and we found that our model converge to it's performance (figure2) using the parameters illustrated in table 4.

Although, the objective of using an autoencoder here was not the reconstruction of the input X,yet evaluating the similarity between X and X' still the best factor to observe and decide whether the learned latent space was able to map the input towards the output with a little amount of loss. Here our trained model achieved an error score of 0.5, without a noticeable overfitting,

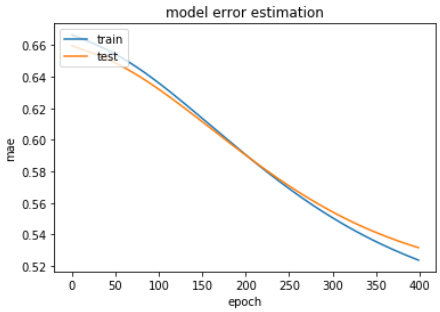


Fig. 2 Training loss of Deep auto encoder

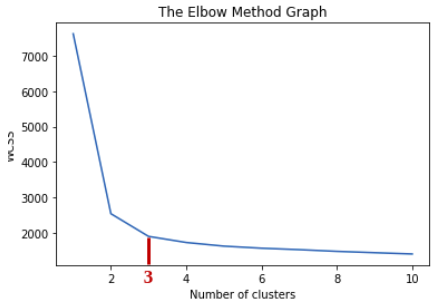


Fig. 3 WCSS Elbow graph to define clusters

which is considered a very promising score that allows us to use the learned latent space matrix I' of range $M \times K$.

4.3 Clustering

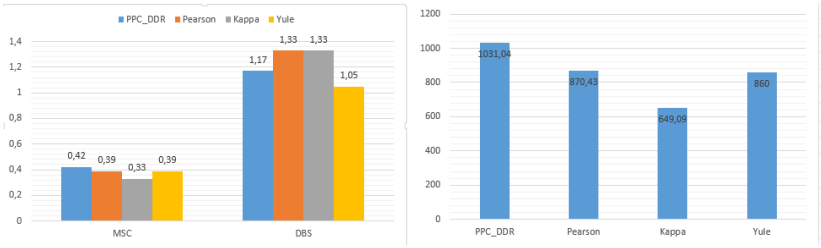
The last step of our experimentation is the clustering using the kmeans model as well as the implementation of WCSS to define the optimal number of clusters. As shown in figure3 that plots the elbow graph of the WCSS algorithm among 15 cluster , the edge falls between four and two clusters so the best option of cluster number is three.

4.4 Results and discussion

We performed a comparison of our proposal with state of the art methods by evaluating the following three metrics: Mean Silhouette Coefficient(MSC)[13] , Calinski-Harabasz Index (CHI)[9], and Davies-Bouldin score (BDS)[5]. After the choice of the metrics we selected three state of the arts similarity measurements methods, namely Pearson, Yule and Kappa. Table 5 exhibit the overall

	MCS	CHs	DBS
PPC-DDR	**0,42**	**1031.04**	**1.17**
Pearson	0,20	146.22	1.36
Yule	0.21	191.3	1.61
Kappa	0,14	160.19	2.01

Table 5 Performance of PPC-DDR and other similarity models



(a) (b)

Fig. 4 Performance of the methods with dimensionality reduction (a): MSC and DBS metrics, (b): CHs metric

performance of our penalised dimensionality reduction approach compared to the other methods across all metrics, where we can visualize the huge superiority between the four models. The PPC-DDR was able to score a MCS of 0.42 compared to 0.20 by Pearson, and 0.1, 0.14 using Yule and Kappa respectively. In terms of CHs and DBS, PPc-DDR performance was also higher than the performance of the other models. It is really surprising the fact that the performance' differences between our proposal and the other models, and may open a perplex about whether this is real or imaginary. To address this question , we assumed that this scoring gap can be due to not applying dimensionality reduction after using pearson, yule and kappa before clustering. To check this assumption, we draw another test for two reason, the first to check the difference of performance between the models with and without dimensionality reduction. Besides to test if our model can still perform better in same conditions as the other algorithms. To perform this test used the before mentioned similarity measuring methods in addition to dimensionality reduction on the outputs of those methods. The results shown in figure4, visualize the performance of our PPC-DDR and the other methods plus dimensionality reduction, where we can notice the huge improvement in pearson, yule and kappa performance, when applying dimensionality reduction compared to the previous results as well as, it is clear that our model still able to compete with the three methods, in which we notice that the proposed approach outperformed the other methods in terms of MSC ,and CHs . Yet we also notice that yule plus deep dimensionality reduction was able to achieve better score in terms of DBS. This second comparison allowed us to understand the deep impact of dimensionality reduction on the clustering phase. As well as we as-

sume that superiority of the yule algorithm maybe due to the tested data set, therefore using other dataset may help to visualize the performance of our proposal.

5 Conclusion

In this work we explored the integration of data mining in learning systems ,which have as an object : optimizing the learning methods, managing the large pool of items(questions, problems). Collecting data about learners' performance can be used to get insight into item properties. So,it is useful to analyze item similarities, which can be used as input to clustering or visualization techniques. Where we proposed Penalized Pearson coefficient to measure item-similarity and integrated deep learning by reducing the representation space. The obtained results were very promising , which inspired us to explore further in this field, and work on the order in which concepts should be presented to learners to optimize their success .

References

1. A survey on deep learning and its applications. Computer Science Review **40**, 100379 (2021)
2. et al, R.P.: Measuring item similarity in introductory programming. In: In Proceedings of the Fifth Annual ACM Conference on Learning at Scale (June 2018)
3. Annalies Vuong Tristan Nixon, a.T.: A method for finding prerequisites within a curriculum. in educational data mining 2011. Jiawei Han and Micheline Kamber (2010)
4. Chollet, F.: keras. https://github.com/fchollet/keras (2015)
5. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-1**(2), 224–227 (1979)
6. Desmarais, M.C., Meshkinfam, P., Gagnon, M.: Learned student models with item to item knowledge structures. User Modeling and User-Adapted Interaction **16**(5), 403–434 (2006)
7. Dharaneeshwaran, Nithya, S., Srinivasan, A., Senthilkumar, M.: Calculating the user-item similarity using pearson's and cosine correlation. In: 2017 International Conference on Trends in Electronics and Informatics (ICEI), pp. 1000–1004 (2017)
8. Kass, R.: Student modeling in intelligent tutoring systems—implications for user modeling. In: User models in dialog systems, pp. 386–410. Springer (1989)
9. Kozak, M.: "a dendrite method for cluster analysis" by calinski and harabasz: A classical work that is far too often incorrectly cited. Communication in Statistics- Theory and Methods **41**, 2279–2280 (2011)
10. Mathisen, B.M., Aamodt, A., Bach, K., Langseth, H.: Learning similarity measures from data. CoRR **abs/2001.05312** (2020)
11. Nazaretsky, T., Hershkovitz, S., Alexandron, G.: Kappa Learning: A New Item-Similarity Method for Clustering Educational Items from Response Data. In: Proceedings of the 12th International Conference on Educational Data Min-ing. International Educational Data Mining Society (2019)
12. Rihák, J., Pelánek, R.: Measuring similarity of educational items using data on learners' performance. In: 10th International Conference on Educational Data Mining.Wuhan, China:International Educational Data Mining Society, pp. 16–23 (2017)
13. Wang, F., Franco-Penya, H.H., Kelleher, J., Pugh, J., Ross, R.: An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity (2017)
14. Yang, J., Yang, G.: Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer. Algorithms **11**(3) (2018)