Neurocomputing 192 (2016) 3–13

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Adaptive structure metrics for automated feedback provision in intelligent tutoring systems

Benjamin Paassen*, Bassam Mokbel, Barbara Hammer

Center of Excellence Cognitive Interaction Technology (CITEC), Bielefeld University, Inspiration 1, 33619 Bielefeld, Germany

ARTICLE INFO

Article history:
Received 11 July 2015
Received in revised form
23 October 2015
Accepted 31 December 2015
Available online 3 March 2016

Keywords:
Metric learning
Intelligent tutoring systems
Sequential data
Learning vector quantization
Algebraic dynamic programming

ABSTRACT

Typical intelligent tutoring systems rely on detailed domain-knowledge which is hard to obtain and difficult to encode. As a data-driven alternative to explicit domain-knowledge, one can present learners with feedback based on similar existing solutions from a set of stored examples. At the heart of such a data-driven approach is the notion of similarity. We present a general-purpose framework to construct structure metrics on sequential data and to adapt those metrics using machine learning techniques. We demonstrate that metric adaptation improves the classification of wrong versus correct learner attempts in a simulated data set from sports training, and the classification of the underlying learner strategy in a real Java programming dataset.

1. Introduction

Intelligent tutoring systems (ITSs) have made great strides in recent years; they offer the promise of individual one-on-one computer based support in the context of scarce human resources, as it is common in massive open online courses (MOOCs), for example [1]. However, researchers have reported 100–1000 h of authoring time for one hour of instructions in ITSs [2]; in addition, ITSs usually require an underlying domain theory such that their applicability is limited in areas where problems and their solution strategies are not easy to formalize [3,4]. In such domains, data-driven approaches are possible, providing feedback based on a set of existing examples for (correct) solutions of the underlying task [4,5]: if the students require a hint on how to change her attempt to get closer to a correct solution, it can be compared to a similar example from the set, and the dissimilarities between her attempt and the example can be contrasted or highlighted in order to help the student to improve her own solution [6–8].

As key ingredients such techniques require data and a suitable metric based on which to compare solutions. More specifically, a suitable metric has to meet at least three requirements in order to be suitable: (A) Solutions are typically non-vectorial. Instead, they are given as *structured data*, that is, as sequences, trees or graphs. Therefore, *structure metrics* have to be used that need to fit the given domain. (B) Feedback should be given based on examples that implement the same underlying strategy. Therefore the metric should emphasize differences in strategy, while being insensitive to differences in style across students. This corresponds to the choice of the metric as well as the choice of parameters for the metric. (C) In order to provide helpful feedback, the metric should be interpretable, in the sense that it should be possible to retrieve the parts of both solutions, which differ from each other.

In this contribution, we focus on alignment metrics for sequential data: recently it has been shown that such metrics can be expressed in terms of a general framework, called algebraic dynamic programming (ADP) [9], which addresses the first requirement (A). Further, we show in this work that all alignment algorithms expressed in that framework can be systematically adapted, as required (B). Finally, all these alignment algorithms allow us to retrieve detailed information which parts of both input solutions are similar and which are not: alignment algorithms match similar parts of both solutions and identify parts which cannot be matched, thereby providing interpretable and actionable knowledge for feedback (requirement C), see e.g. [8].

1.1. Contribution and overview

The main contributions of this paper are the following: First, we show that the general framework of *algebraic dynamic programming* (ADP) enables us to express a broad class of structure metrics, namely alignment distances. Exemplary, we use ADP to express four alignment algorithms: Global sequence alignment, affine sequence alignment, dynamic time warping and the Sakoe–Chiba approximation of

* Corresponding author.
E-mail addresses: bpaassen@techfak.uni-bielefeld.de (B. Paassen), bmokbel@techfak.uni-bielefeld.de (B. Mokbel), bhammer@techfak.uni-bielefeld.de (B. Hammer).

http://dx.doi.org/10.1016/j.neucom.2015.12.108
0925-2312/

dynamic time warping. Second, we demonstrate that gradients on alignment distances can be calculated efficiently using ADP. Third, we use the calculated gradients for *structure metric learning*: We adapt metric parameters to improve the classification accuracy of a *relational generalized learning vector quantization* (RGLVQ) classifier. Finally, we apply the structure metric learning scheme using four different alignment algorithms on two different datasets from the domain of intelligent tutoring systems.

Note that the techniques presented in this work are by no means limited to intelligent tutoring systems but can be applied in all settings, where metrics on sequential data are required and should be adapted to optimize some (differentiable) cost function (see e.g. [10] for an example from the biomedical domain).

The outline of this paper is as follows: In Section 2 we discuss related work, in particular data-driven intelligent tutoring systems (ITSs), similarity-based machine learning, structure metrics and structure metric learning. We also discuss our choice of datasets in the context of existing literature on ITSs. In Section 3, we introduce a simplified version of ADP as generalization of alignment algorithms and use it to express four example metrics, which we evaluate in the experiments later on. We explain metric learning on ADP alignment algorithms using RGLVQ in Section 4. Finally, we report our experiments in Section 5.

2. Related work

This research connects several, seemingly disconnected fields, such as artificial intelligence in education, educational data mining, classic machine learning, structure metrics, metric learning and formal languages. In this section, we provide an overview of these different connections and also embed our own work in the context of the existing literature.

2.1. Data-driven intelligent tutoring systems

Intelligent tutoring systems (ITSs) are systems to enhance student learning via artificial intelligence methods. Most of the time, students proceed through a curriculum of different tasks to obtain skills and knowledge. The systems job is to select the next task depending on the current level of knowledge and individual parameters of the student (*outer loop*) and to support her solving the current task (*inner loop*) [4]. Such systems have been successfully applied in many contexts, especially in learning logic and math concepts, and have been proven to lead to positive learning outcomes for students [11,5].

However, they usually rely on extensive knowledge engineering to formalize domain concepts and explicitly track student knowledge, which is both costly and difficult, especially in domains where explicit and detailed knowledge about the domain cannot be obtained (so-called *ill-defined domains*) [2–4]. To relieve ITS engineers from the burden of knowledge engineering, data-driven approaches have emerged. Such approaches try to replace pre-defined and explicit domain knowledge by inference based on example-data of students interacting with the system [4,5].

Here, we focus on the *inner loop* mentioned before: To support students in solving a task, utilizing only example solution attempts handed in by other students. An intuitive solution to this problem is to base student support on a notion of *similarity* to existing solutions: we can approximate a student model by considering the similarity of her solution to all solutions in the example set. A hint to improve her solution can be based on the difference between her solution and a similar (but better) solution [6–8].

Further, a proper similarity measure enables ITS engineers to apply machine learning techniques for further problems: one can try to detect outliers or buggy solutions, one can estimate the quality of solutions based on the known quality for some examples (regression) and one can cluster or classify solutions into discrete, meaningful sets. In our experiments we focus on the latter and distinguish between correct and wrong executions of a sports exercise (see Section 5.1) and between the underlying algorithms of computer programs (see Section 5.2).

Such an approach requires a proper similarity measure (that is a metric) as key ingredient. Note that most common similarity measures, such as the Euclidean distance or the radial basis function kernel, are based on a vectorial data representation. While first approaches exist to transform student data into a vectorial format, most data is still only available as structured data, such as sequences, trees or graphs [12]. Thus, we face a three-fold challenge: constructing a similarity measure that works on the available data in the first place, adjusting this similarity measure to be apt for the task at hand and utilizing the similarity measure to generate actionable knowledge for an ITS. The latter is the general topic of similarity-based machine learning, the former two refer to structure metrics and (structure) metric learning.

2.2. Similarity-based machine learning

From early on, machine learning methods based on similarity measures have been utilized, starting with simple schemes like k-nearest neighbor classification [13] or k-means clustering [14]. The general rationale is that data which are similar to each other in some respect may be similar in other respects as well. Research on similarity-based machine learning has flourished in recent years, mainly driven by the development of powerful kernel-approaches, and includes such popular methods such as the Support Vector Machine, extended nearest neighbor-schemes and Gaussian process regression [15,16].

Here, we require a method which lends itself to gradient-based optimization. Gradient-based schemes in similarity-based machine learning have been applied successfully in the case of relational learning vector quantization (RGLVQ) [17], which we describe in more detail in Section 4.

Note, however, that the focus of this work is not so much on demonstrating the capabilities of methods based on an existing similarity measure (here, the interested reader is referred to the literature cited above), but rather how to obtain a proper (structure) similarity measure in the first place.

2.3. Structure metrics

Over the years, multiple structure metrics have been suggested, reaching from sequential data over trees to graphs, see e.g. [18] for a recent review. Kernel-approaches have been especially popular, such as the diffusion or convolution kernel approach [19,20]. Unfortunately, most of these approaches cannot directly deal with rich data attached to the graph nodes and/or are runtime-inefficient.

In this contribution, we focus on sequential data, where we can rely on the abundant work on *edit* or *alignment distances*. Such methods extend both input sequences, such that similar elements are *aligned*. They have been successfully applied in diverse domains, such as automatic spell-checking [21,22], bioinformatics [23–25] and speech processing [26]. All of those alignment distances can be efficiently calculated using dynamic programming with a worst-case runtime of $\mathcal{O}(M \cdot N)$, with M and N being the number of sequence elements in the first and the second input sequence respectively.

Given the abundance of alignment algorithms in the literature, we can select a suitable one for our data: for motion data, dynamic time warping is a well established technique [26], accompanied even by techniques to make it a linear-time algorithm [27]. For comparing syntactic building blocks, however, classic edit distance