Elicitation of User Preferences via Incremental Learning in a Declarative Modelling Environment

Georgios Bardis[1], Vassilios Golfinopoulos[1], Dimitrios Makris[1], Georgios Miaoulis[1,2], and Dimitri Plemenos

[1] Department of Informatics, TEI of Athens, Ag. Spyridonos St., 122 10 Egaleo, Greece
[2] XLIM Laboratory, University of Limoges, 83 rue d'Isle, Limoges, 87000, France
{gbardis,golfinopoulos,demak,gmiaoul}@teiath.gr

Abstract. Declarative Modelling environments exhibit an idiosyncrasy that demands specialised machine learning methodologies. The particular characteristics of the datasets, their irregularity in terms of class representation, volume, availability as well as user induced inconsistency further impede the learning potential of any employed mechanism, thus leading to the need for adaptation and adoption of custom approaches, expected to address these issues. In the current work we present the problems encountered in the effort to acquire and apply user profiles in such an environment, the modified boosting learning algorithm adopted and the corresponding experimental results.

Keywords: Incremental Learning, Irregular Datasets, Declarative Modelling.

1 Introduction

A declarative scene modelling environment offers the designer/user the ability to produce a 3D scene without delving into its explicit geometric properties. User input is a set of declarative objects, associations and properties, practically representing constraints for the geometric properties of the final scene's objects, yet expressed in an intuitive manner ("a large bedroom", "kitchen adjacent to dining room"). The typical declarative scene modelling environment has to map these abstract notions to concrete constraints and, subsequently, resolve these constraints in order to achieve one or more geometric representations of the submitted input. Such a scenario implies numerous possible outcomes, not all of equal interest for the user. Hence, the question of user profiling and intelligent solution evaluation is a considerable concern. The issues that arise in this effort are identified and analysed in the following section. Next, a framework for capturing and applying user preferences in a declarative modelling environment is presented, including the modified version of an incremental learning algorithm. The performance of the proposed solution is exhibited through a number of experimental results which are discussed and analysed for future research directions.

2 Related Work

The ability to capture and apply user preferences in a declarative modelling environment is inherently inhibited by a number of particularities exhibited by the declarative design process, originating, in turn, from the nature of the declarative modelling methodology [3]. A series of approaches have been suggested addressing this issue. The effort presented in [8] offers user preference capture which is isolated per scene, gradually training a dedicated neural network upon user feedback. This approach, while efficient in pinpointing the user interpretation of the scene at hand, does not allow for a universal user profile applicable to newly submitted scenes, since it has to go through training phases corresponding to each new scene. The approach in [4] comprises a hybrid supervised/unsupervised concept acquisition, which constructs similarity classes in an unsupervised manner, presenting class representatives to the user in order to guide the mechanism towards the selection of the target concept of the original description. This approach operates under the assumption that the similarity of solutions grouped together during the unsupervised phase reflects the user's notion of similarity. This, however, may imply contradictions with actual user preferences in the later stages.

The widest scope is covered by [1] which allows for horizontal user profiling, creating for each user a profile that is applicable to all future scenes (s)he may submit. The advantage of this approach becomes evident during the Scene Understanding phase of the Declarative Modelling methodology [8] where the user may be presented with those geometric representatives of the generated solutions that are closer to previously acquired profile. This approach relies on the mapping of the generated solutions to a set of observed attributes, thus relying on the participation of the latter to the formation of actual user preferences. In the following we discuss the corresponding mechanism and present new experimental results based upon an extended attribute set.

3 Machine Learning in a Declarative Modelling Environment

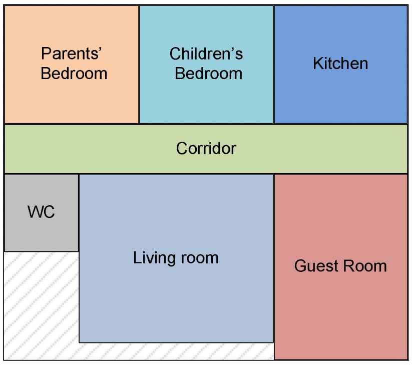
The typical scenario of use for a declarative modelling environment starts with a rough idea of the scene to be designed on behalf of the user. It is often the case that at this stage of design, i.e. early-phase design, the user has a grasp of only an abstraction of the final expected result. Hence, the repertoire of building blocks for this initial input consists of declarative objects, properties and associations, leading to a declarative description similar to the one appearing in Fig. 1, represented by three corresponding sets.

O = {
[*Living Room*],
[*Guest Room*],
[*Parents' Room*],
[*Childrens' Room*],
[*Corridor*],
[*Bathroom*]
}

A = {
[*Living Room* adjacent west *Guest Room*],
[*Living Room* longer than *Guest Room*],
[*Parents' Bedroom* adjacent west *Bathroom*],
[*Children's Bedroom* adjacent east *Bathroom*],
[*Kitchen* adjacent south C*orridor*]
}

P = {
[*Corridor* is very narrow]
}

Fig. 1. Example declarative description of the spatial arrangement of a house



$$C = \frac{\sum_{i=1}^{n} a_i}{a_{bb}}$$

where:
$\boldsymbol{a_i} \rightarrow$ object i area
$\boldsymbol{a_{bb}} \rightarrow$ bounding box area
$\boldsymbol{n} \rightarrow$ number of objects

$$a_i = w_i \cdot l_i \quad , \quad a_{bb} = (max(x_k+w_k)-min(x_j))\cdot(max(y_p+l_p)-min(y_m))$$

Fig. 2. Example observed attribute *Compactness*

Once the user has submitted the declarative description, a mechanism is responsible for translating it into a set of constraints and resolving it, thus yielding – typically numerous – geometric interpretations of the original input, i.e. solutions for the constraint set.

The effort to acquire and apply user preferences to the generated solutions has to consider properties of interest to the user when assessing the generated results. In other words, a minimal set of observed attributes have to be assumed and calculated for each solution as a first step towards the effort to mimic the user's evaluation. Depending on the nature of the scene, these attributes can be, and usually are, suggested by the corresponding domain experts. The only requirement for these attributes, in regard to the automation of the learning process, is their computational interconnection and extraction feasibility with respect to available scene information, namely, its declarative description and its geometric equivalents. In the current work we assume the existence and availability of a well-defined and computationally feasible set of observed attributes, directly related to available scene information. An example attribute and its connection to the solutions' geometric characteristics appears in Fig. 2.

4 Declarative Modelling Machine Learning Requirements

Once the solutions have been mapped to a set of attributes and evaluated by the user we find ourselves in the familiar grounds of classified data vectors which may subsequently be used to train and/or evaluate a machine learning mechanism to be used for automatic classification of future generated solutions. Such a mechanism is expected to have acquired and therefore be able to apply user's preferences on previously unseen samples. A number of reasons, originating from the nature of the declarative modelling process and the corresponding environments prevent us, however, from the straightforward application of traditional classifier approaches. These reasons may be summarised as follows:

Original solutions may require large storage capacities. It may be the case that each solution is a complicated 3D model, a high resolution image or other volume intensive data form [1],[9]. In this case, storage capacity required for maintaining all previously evaluated solutions is prohibited, which, in turn, diminishes the value of preserving the solutions only in the encoded vector form.
Solutions are generated at distinct and usually distant times. The typical session of use of a declarative modelling environment comprises submission of a description on behalf of the user, solution generation on behalf of the environment and global or partial evaluation the outcome population, on behalf of the user. This functionality suggests that samples are available in bursts and in abnormal time intervals.
Solutions populations are of varied size and class representation. A declarative description may range from highly restrictive, thus leading to only a limited number of solutions, to highly abstract, thus yielding hundreds of thousands of alternative interpretations. Depending on the case, the user may or may not be willing to evaluate the entire population of outcomes since user's evaluation may be interrupted after a small number of approvals, thus leading to imbalanced datasets. In addition, depending on input declarative description, some attributes or some classes may not be included in the final outcome and, hence, not be present in the training (sub)set.
Even small data perturbations may imply classification divergence. When user evaluation is involved, small differences in the evaluated samples may cause radically different classification. For example, in terms of the example description of Fig. 1, the misalignment of the edges of two rooms, regardless how insignificant geometrically, may have an important impact on user's preference.

For the aforementioned reasons, many traditional classifiers can not be successfully applied in such an environment. The gradual and limited availability of evaluated samples suggests the need for a mechanism that will be able to maintain knowledge without having to maintain the supporting data for this knowledge. In other words, an incrementally learning mechanism is required to efficiently grow both in terms of size and complexity as well as in terms of knowledge absorption. Even mechanisms able to fulfil this requirement for adaptive learning [5], fail to successfully address the highly unstable nature of the data due to their inherent assumption of data stability, i.e. small perturbations of the input (data sample) imply small perturbation of the output (data classification). Last but not least, the varied and largely unknown distribution of the attribute values restricts the use of formal statistical methods. The requirements for the mechanism to be employed in such an environment may be summarised as follows:

- The mechanism must be able to acquire new knowledge through additional training without exhibiting "catastrophic forgetting".
- The mechanism must be able to capture divergent classifications of samples often exhibiting only subtle differences.
- The mechanism must be able to conform to varied sizes of additional training sets.
- The mechanism must be able to address potential training sets imbalance.

The mechanism employed to fulfil these requirements relies on a committee of weak classifiers, in which new members are gradually trained and added at the rate of newly produced samples. It uses the algorithm presented in [7], which in turn is inspired by [6], applying the boosting learning methodology as its basis, customised to the aforementioned requirements.

5 Algorithm Discussion

The idea of boosting is based on the fact that a large percentage of training time is dedicated in perfecting an already adequately performing mechanism. Hence, it is proposed that instead of one large highly trained mechanism, a committee of under-trained mechanisms, the *weak learners,* are used in order to achieve integral representation of the concept under investigation. This approach is implemented in [6], yet it relies on the existence of the entire training set for its successful application. This notion is carried further in [7], by allowing partial availability of the training set in the form of subsets at the cost of minimal previous knowledge degradation, yet requiring adequate class and attribute representation in each subset. The approach used in [1], moves one step further, accepting the training data subsets without any previous assumption regarding class representation since it is not realistic to expect so in such an environment. This is achieved by modifying the original algorithm to be able to capture, to an adequate degree, new knowledge based on data subsets of varying quality and size, at the cost of higher previous knowledge degradation which, however, is maintained at acceptable levels as demonstrated by experimental results.

The algorithm starts its operation considering all samples of equal weight, the latter representing their classification difficulty. For each new sample subset, a corresponding subcommittee comprising a predefined number of members is constructed. Back-propagation multi-layered neural networks are used as members of the committee, allowing, however, a high error margin during their training due to their expected role as *weak* rather than *strong* learners in the mechanism. During the subcommittee training, samples not successfully classified receive increased weight. Each new member is conclusively accepted to the subcommittee if it does not degrade overall performance below a given threshold. Not all members are created equal however: each bears a degree of importance, reflecting the weights of samples it successfully classified during its subcommittee creation. This allows later, during the prediction phase, for successful classification of hard-to-classify samples based on a minority vote of high-importance members.

One of the points modified to meet the current environment's requirements was subcommittee creation termination criteria. The original algorithm still requires half the remaining population to be used to train each new committee member. In that phase, the increased weight of hard-to-classify samples just increases their probability to be included in the training set due to the weighted random selection of samples. Creating a dataset including the possibly few remaining hard-to-classify samples and a lot of easy samples mislead the training towards the easy ones. In the customised version used herein we focus, instead, on the samples' weights, using *half of the sum of weights* of remaining samples as threshold instead of *half of the number* of remaining samples. This change, combined with the original algorithm's idea of weighted sample selection, has accelerated generation of adequate weak learners especially in advanced stages of the training process where the original threshold lead to weak learners of small contribution to overall performance. The second major difference is that the original algorithm generates new subcommittee members until the predefined number is reached. We have relaxed this restriction using a threshold for overall error instead, in order to accommodate for varied sizes of new training subsets. Hence, if the new version of the entire committee perfectly classifies all current samples, the subcommittee creation is terminated regardless of current number of members. Despite the undecidability of the general neural network loading problem [10], the computational cost of the applied algorithm is kept low due to weak learning requirement for the committee members and low overall error threshold for successful committee extension, further reduced by pre-processed, fine tuned weak learners' parameters. The algorithm used appears in Fig. 3., a modified version of [7] and a notation index is presented in Table 1.

```
for each new population of n samples $s_1$, $s_2$, …, $s_n$     // i.e. for each new subcommittee k
    if k>1 then                            // in case the MLC already contains weak learners
        Evaluate new population according to $MLC_{k-1}$ and update sample weights $w_i$
    else
        Initialise sample weights $w_i$ to 1/n
    p=1
    repeat     // for all weak learners to be created in the subcommittee
        repeat // check new weak learner until overall performance is acceptable
            repeat //check new weak-learner until its performance is acceptable
                Create ts as a set of samples with at least half the total weight.
                Create the es with the remaining samples.
                Train a new weak learner using ts.
                Calculate $e_p$ for the current weak learner for ts and es.
            until $e_p \leq 0.5$
            Add current weak learner to the $C_k$
            Evaluate current population according to $MLC_k$
            Calculate E for the current form of the committee
        until $E \leq 0.5$
        Store $b_p=e_p/(1-e_p)$ for the current weak learner and increment p by 1
        Update sample weights according to recent evaluation and normalize
    until p>m or E=0
end of for                  // subcommittee k creation loop
```

Fig. 3. Algorithm used

Table 1. Algorithm parameters description

Param.	***Description***
m	pre-defined maximum number of members of a subcommittee
p	$p \in \{1..m\}$=the current number of weak learners in the current subcommittee
n	the number of solutions in the current population, $i \in \{1..n\}$
k	number of subcommittees created up to now
C_k	current subcommittee
MLC_k	$\{C_1, C_2, \ldots, C_k\}$= overall committee, including current form of current subcommittee
g_i	grade vector of solution i in the current population
u_i	user evaluation for solution i
s_i	(g_i, u_i) = sample in the current population
w_i	sample weight
ts	current training set – created for every weak learner based on weight distribution.
es	current evaluation set – solutions of current population not belonging to the ts.
e_p	product of weights of all solutions classified incorrectly by the current weak learner
b_p	$=e_p/(1-e_p)$ *importance* of weak learner's p vote during the overall voting
E	product of weights of all solutions classified incorrectly by the current committee

6 Experimental Results

In the following we briefly present some characteristic experimental results, based on actual user data from the MultiCAD environment. Generalisation error is presented with respect to gradual training of the mechanism with bursts of data originating from 5 descriptions. The mechanism was incrementally trained with evaluated solutions from one description at a time simulating actual system use. Generalisation error is measured against user evaluated solutions, previously unseen by the mechanism. An enhanced set of 8 prominent attributes was used for solution representation in vector form, thus demonstrating significant improvement against previous results.

It is interesting to notice in the results the degradation of the mechanism in the effort to acquire new knowledge by incremental training with a new subset at a time, maintaining, nevertheless, an error margin around 0.1. The generalisation error, concentrating on previously unseen solutions, is also maintained at acceptable levels.

The effort to acquire and automatically apply user preferences is bound to exhibit additional anomalies due to user discrepancies. Table 3 summarises the results for an alternative user. Here we notice that the consideration of training subsets from 2 additional scenes do not improve performance, in contrast with the data subset from scene 5. In the next experiment we alter the sequence of training after scene 2, submitting scene 5 immediately afterwards. The result is a steeper improvement in the generalisation error revealing the potentially more complex nature of scene 5.

Table 2. Error Evolution – User 1

	Initial Training Scene 1	***Incr. Training Scene 2***	***Incr. Training Scene 3***	***Incr. Training Scene 4***	***Incr. Training Scene 5***
Error for Scene 1	1.94%	14.93%	7.61%	9.40%	6.87%
Error for Scene 2	23.84%	1.95%	7.30%	16.30%	15.33%
Error for Scene 3	9.34%	14.01%	3.38%	13.04%	10.31%
Error for Scene 4	17.36%	3.96%	10.38%	1.89%	5.66%
Error for Scene 5	8.41%	41.68%	17.22%	13.31%	6.85%
Generalisation	**14.74%**	**19.88%**	**13.80%**	**13.31%**	**-**
Overall	**12.18%**	**15.31%**	**9.18%**	**10.79%**	**9.00%**

Table 3. Error Evolution – User 2

	Initial Training Scene 1	***Incr. Training Scene 2***	***Incr. Training Scene 3***	***Incr. Training Scene 4***	***Incr. Training Scene 5***
Error for Scene 1	1.49%	5.82%	5.82%	5.82%	2.24%
Error for Scene 2	5.35%	0.24%	0.24%	0.24%	0.97%
Error for Scene 3	24.48%	0.16%	0.16%	0.16%	0.00%
Error for Scene 4	12.64%	0.38%	0.38%	0.38%	0.19%
Error for Scene 5	0.98%	4.89%	4.89%	4.89%	0.20%
Generalisation	**10.86%**	**1.81%**	**2.63%**	**4.89%**	**-**
Overall	**8.99%**	**2.30%**	**2.30%**	**2.30%**	**0.72%**

Table 4. Error Evolution – User 2/Alternative Sequence

	Initial Training Scene 1	*Incr. Training Scene 2*	*Incr. Training Scene 5*	*Incr. Training Scene 3*	*Incr. Training Scene 4*
Error for Scene 1	1.49%	5.82%	2.54%	2.24%	5.82%
Error for Scene 2	5.35%	0.24%	0.97%	0.97%	0.24%
Error for Scene 5	0.98%	4.89%	0.20%	0.20%	4.89%
Error for Scene 3	24.48%	0.16%	0.16%	0.00%	0.16%
Error for Scene 4	12.64%	0.38%	2.45%	2.45%	0.38%
Generalisation	**10.86%**	**1.81%**	**1.31%**	**2.45%**	**-**
Overall	**8.99%**	**2.30%**	**1.26%**	**1.17%**	**2.30%**

7 Conclusion

In the current work we have examined and demonstrated the issues arising in the effort to apply machine learning for the acquisition and exploitation of user preferences in declarative modelling. The particular domain requirements, implied by the declarative modelling methodology and the environments implementing it, include datasets of varying sizes, class representation imbalance, user inconsistencies, inherent data instability and irregular availability. We have customised and applied an incremental learning algorithm based on the notion of boosting demonstrating the flexibility and applicability of this neural-network based approach. The results have revealed adequate generalisation performance, given the aforementioned domain characteristics, while largely maintaining previously acquired knowledge. In particular, the adopted mechanism has maintained acceptable levels of generalisation and overall errors, mainly varying with corresponding user consistency. It also appears to be sensitive to the training sets sequence and is, therefore, affected by user choices in system use. Nevertheless, results demonstrate the applicability of such a mechanism in the specific context.

Due to the idiosyncrasy of the domain, several approaches may be followed to improve the presented results. Identifying and utilizing attributes that capture a wider range of potential user evaluation criteria would offer improved mapping of data properties. In the same category of improvements, methods to ensure the absence of user inconsistencies, possibly in the form of user defined similarity metrics, would allow higher confidence in the available datasets. In the context of the machine learning methodology applied, reduction of the generalisation error could be achieved through guided user evaluation of selected scenes and solutions depicting wide variations in the observed attributes and their values. Lastly, the lack of adequate and consistent user feedback can be overcome through the availability of such a platform in a web environment, thus allowing a wider spectrum of evaluation contributions.

References

[1] Bardis, G.: Intelligent Personalization in a Scene Modeling Environment. In: Miaoulis, G., Plemenos, D. (eds.) Intelligent Scene Modelling Information Systems. SCI, vol. 181, pp. 973–978. Springer, Heidelberg (2009) ISBN 978-3-540-92901-7

[2] Bidarra, R., et al.: Integrating semantics and procedural generation: key enabling factors for declarative modeling of virtual worlds. In: Proceedings of the FOCUS K3D Conference on Semantic 3D Media and Content, France (February 2010)

[3] Bonnefoi, P.-F., Plemenos, D., Ruchaud, W.: Declarative Modelling in Computer Graphics: Current Results and Future Issues. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3039, pp. 80–89. Springer, Heidelberg (2004)

[4] Champciaux, L.: Classification: a basis for understanding tools in declarative modelling. Computer Networks and ISDN Systems 30, 1841–1852 (1998)

[5] Doulamis, A., et al.: On Line Retrainable Neural Networks: Improving the Performance of Neural Network in Image Analysis problems. IEEE Trans. on Neural Networks 11(1), 137–155 (2000)

[6] Freund, Y., et al.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)

[7] Lewitt, M., et al.: An Ensemble Approach for Data Fusion with Learn++. In: Windeatt, T., Roli, F. (eds.) MCS 2003. LNCS, vol. 2709. Springer, Heidelberg (2003)

[8] Plemenos, D., et al.: Machine Learning for a General Purpose Declarative Scene Modeller. In: International Conference GraphiCon 2002, Nizhny Novgorod, Russia (2002)

[9] Smelik, R.M., et al.: Declarative Terrain Modeling for Military Training Games. International Journal of Computer Games Technology (2010)

[10] Wiklicky, H.: The Neural Network Loading Problem is Undecidable. In: Proceedings of Euro-COLT 1993, Conference on Computational Learning Theory, pp. 183–192. University Press (1994)