Comparison of Classifiers for use in a Learning by Demonstration System for a Situated Agent

Michael W. Floyd and Babak Esfandiari

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, Ontario

Abstract. In *learning by demonstration* systems, learning is performed by an agent observing how an expert behaves in response to a given input. The learning task is more difficult when the agent is situated in a dynamically-changing environment, of which only a partial view is available at any time. In this paper we propose a comparison of different learning by demonstration techniques, namely case-based reasoning, decision trees, support vector machines and naive bayes classifiers, in the dynamic environment constituted by the RoboCup robotic soccer simulation. An initial look at the results indicate that case-based reasoning behaves well across all experiments and outperforms the other classifiers. Case-based reasoning even outperforms a decision tree classifier when learning from an expert whose internal reasoning is represented as a decision tree.

1 Introduction

Case-based reasoning has been successfully applied to a variety of gaming applications [1–5] and, more recently, has been used to learn to play a game by watching an expert play. In these *learning by demonstration* systems, cases are generated by observing how an expert behaves in response to the current state of the game and then behaving similarly when presented with similar game states. Using case-based reasoning for learning by demonstration has been used in a variety of games including robotic soccer [6, 7], real-time strategy [8] and Tetris [9].

In some games, like chess or Tetris, the game playing agent will have a complete world view at all times and will be fully aware of the state of the environment. However, in many real-world domains the agent will only have a partial world view. This is common when the agent is able to move around a large, dynamic environment. If the agent is only able to view a portion of the environment at a given time, it will never be fully aware of the state of the environment at all times.

This partial world view can pose a challenge since the environmental stimuli may not be consistent over time. For example, as the agent moves around the environment objects may move in and out of its field of vision. Also, there is no guarantee that any specific object will be visible to the agent at a particular moment in time. In fact, the agent may never know how many objects of any kind are present in total in the environment. One approach that has been used to overcome this is to have other agents provide information about unseen areas of the environment [3] although this requires multiple cooperating agents that are adequately distributed over the environment.

This difficulty in dealing with data from a partial world view is not limited to case-based reasoning, but exists in most learning by demonstration work. Typical approaches to deal with a partial world view include only using omnipresent objects as inputs [10], ignoring external stimuli [11], or only reasoning using commonly visible object [12, 13]. Finding a reasoning technique that does not require such limitations or restrictions would be a preferable solution as it would allow for learning by demonstration even if the agent only had a partial world view.

The goal of this paper will to be compare case-based reasoning to several other classifiers to determine which is most applicable to a learning by demonstration system that receives the sensory information of an agent with a partial world view as inputs. In Section 2 we describe the raw data available when observing an expert perform a task and Section 3 presents a method to transform the raw data so it can be used by common classifiers. Section 4 provides the experimental results and, finally, Section 5 discusses the conclusions we can draw from those experiments.

2 Sensory Stimuli

When a software agent or a robot is situated in a large, dynamic environment it will likely have a partial, and always changing, world view. For example, the field of vision of a soccer player is shown at three points in time in Figure 1. At different points in time, as shown in Table 1, the player observes different objects. Additionally, the player may not have the ability to uniquely identify objects. For the soccer player, this would mean it would be unable to differentiate between the various *players* or know if it is observing the same *ball* at the different points in time.

Each sensory stimulus, S, of an agent can then be modelled as a collection of multi-valued attributes, V_i. If the agent is able to differentiate between n types of objects, there will be n multi-valued attributes. Going back to the soccer example, there would be two types of objects: balls and players.

$$S = \{V_1, \ldots, V_n\} \tag{1}$$

The multi-valued attribute for each type of object contains the objects, of that type, that are currently visible to the agent. Each of these multi-valued attributes is an unordered set, with the size of that set being variable. This variability in set size can cause different types of objects to have different sized sets but can also cause the same object type to have differently sized sets at different points in time if objects move in or out of the agent's field of vision.

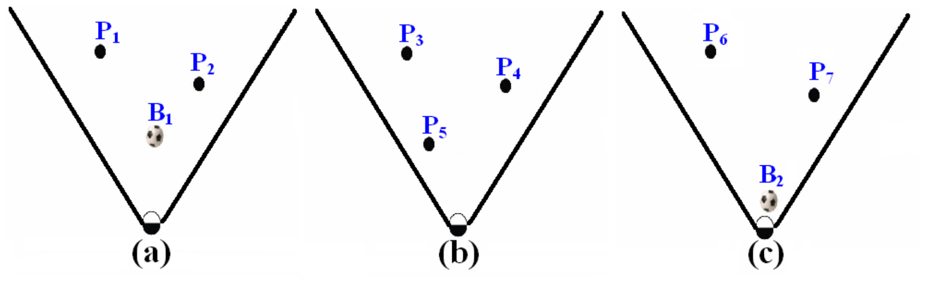

Fig. 1. A graphical representation of a soccer player's field of vision at three different points in time.

Field of Vision	Balls	Players
(a)	B_1	P_1, P_2
(b)		P_3, P_4, P_5
(c)	B_2	P_6, P_7

Table 1. The attributes of each field of vision.

3 Data Transformation

Most classification algorithms expect each feature to be single-valued. We therefore need to break down our multi-valued features into sets of single-valued ones, while ensuring a systematic way for each value of a stimulus to be mapped to an appropriate feature. This can be defined as a problem of matching the visible objects in a stimulus, $S = \{o_1, \ldots, o_m\}$, to a set of single-valued features, $F = \{f_1, \ldots, f_n\}$ (where $m \leq n$).

If, in the soccer example, there were always exactly one *ball* and one *player* visible in every stimulus then a feature vector could be constructed that contained two objects. The first object in the feature vector, representing the *ball*, could always be compared to the first object in other feature vectors. However, if there were multiple instances of each type of object then a correspondence problem arises. If a pair stimuli each contain two *player* objects, it then becomes necessary to determine how those objects will be matched when comparing the stimuli. For example, we would need to determine if the first player in the first stimulus would be compared to the first player or the second player in the second stimulus. This correspondence problem is further complicated by the fact that

Ball	Distance	Direction
B_1	3	$0°$
B_2	1	$10°$

Table 2. The attributes of each ball.

different stimuli could have different numbers of objects of a given type. When comparing two stimuli this can result in objects that do not have a matching object to be compared to. An example where this correspondence issue would be a problem is in a decision tree. If a node of the tree makes a decision based on a single object of a specific type, there is no way to know which object of that type should be used.

There has been some work examining how data with multi-valued attributes can be used in a decision tree classifier [14–16]. In these approaches, when traversing the tree all possible valid paths are considered. When a test input contains a multi-valued attribute, any decisions based on that attribute are made using all of the values of the attribute so that several paths through the tree are taken simultaneously. This results in examining a series of paths through the tree that only involve a single value of each attribute and then combining the results. Since these approaches only examine a single object of each type during each traversal of the tree, they are not applicable in applications where information from multiple values of a multi-value attribute are needed. In the soccer example, this implies that only a single *player* object would be considered during each traversal of the decision tree, so no decisions involving multiple players would be possible.

Another possible approach would be to convert Table 1 into first normal form. This would involve replacing the *Balls* and *Players* columns with separate tables that contain a single row for each object. Using normalized relational tables, inductive logic programming (ILP) [17] could be used for multi-relational classification [18, 19]. However, similar to the decision tree techniques described above these, ILP approaches treat each row containing a multi-valued attribute as multiple rows containing single-valued attributes, thereby not allowing all of the multiple values of an attribute to be used simultaneously.

In order to make this data usable by most classifiers, we will transform the data into a form that mirrors the biases imposed by our existing case-based reasoning system [6, 7]. The case-based reasoning system adds two primary biases:

1. **Set Ordering:** In order to provide an ordering to the multi-valued attributes, objects are matched with similar objects when comparing two sensory stimuli. If there are an unequal number of objects then some objects may not have a match. This does not provide a fixed ordering of the objects, since the matching can be different depending on the other sensory stimulus that is being compared to.
2. **Extra Objects:** When calculating the distance between two sensory stimuli, an object that does not have an appropriate matching object in the other sensory stimulus results in a penalty value being added. This penalizes stimuli containing a different number of objects.

We will attempt to keep similar biases in the transformed data so as to avoid giving an advantage to any of the classifiers during the experimental comparison. The data transformation involves the following steps:

1. **Fixed Sized Feature Vector:** Each sensory stimulus will be represented as a feature vector of a fixed length. Initially, a set of training stimuli will be examined to determine the maximum size, M_i, of each of the multi-valued attributes. The feature vectors will then be created so that each attribute is able to contain the maximum number of values. If there are N attributes, the vector will be able to hold $\sum_{i=1}^{N} M_i$ objects. Since each object is complex and has both a distance and a direction, relative to the agent, the feature vector will actually be twice that size ($Vectorlength = 2\sum_{i=1}^{N} M_i$). For example, the first $2 \times M_1$ values of the vector will contain data related to the 1st multi-valued attribute.
2. **Object Ordering:** Objects are ordered by sorting them based on their distance from the agent. Objects that are closer to the agent will be placed in the feature vector before more distant objects of the same type (when two feature vectors are compared the objects closest to the agent will be compared to each other, second closest objects compared, etc.). This does not allow for more precise object matching, since the ordering is performed in advance, but is similar to the matching performed by our case-based reasoning system since it is computationally inexpensive and can be used in real-time.
3. **Normalization:** Similar to how a training set was mined to find the maximum size of each multi-valued attribute, it will also be mined to find the minimum and maximum values of the object distances and directions. These maximum and minimums will be used to normalize the distances and directions, between *0 and 1*, so that all elements of the feature vector are of a similar scale.
4. **Padding:** If a sensory stimulus contained fewer than the maximum number of values for any of the multi-valued attributes, the remaining entries in the vector will be padded with values that represent *unseen objects.* These values are used to mirror the penalty values used by the case-based reasoning system, since a visible object will have values than are dissimilar to the unseen object values.

4 Experimental Results

Our experiments will look to compare the performance of our existing case-based reasoning system [6, 7] with three popular classification methods: decision trees, support vector machines and naive bayes. For these three algorithms the Weka [20] implementations will be used[1].

The data will be generated by observing simulated robotic soccer players in the RoboCup Simulation league [21]. Two teams of players will be observed: Krislet agents [22] and CMUnited agents [23]. Krislet agents behave in a simple manner. They turn until they can see the soccer ball and then run toward the ball. When they get to the ball they attempt to kick it toward their opponent's

[1] The J48 algorithm for the decision tree, SMO algorithm for support vector machines and NaiveBayes algorithm for naive bayes.

goal. Krislet was selected because it is a simple, reactive team that should be easily represented as a decision tree. CMUnited is far more complex and was the former champions of the RoboCup Simulation League. They use a layered learning architecture and a number of strategies including formation strategies and agent communication. CMUnited players can have multiple states of behaviour and maintain internal models of the world, so their behaviour is likely more similar to that of a human expert.

The agents were observed while playing games of simulated soccer. During the games, each team was comprised of *11* players per team and the opposing team was always made of Krislet agents. Both Krislet and CMUnited agents were observed playing *25* complete games of soccer resulting in approximately *100000* observations being collected per team. Each classifier was trained using *5000* randomly selected observations[2] and tested using 10-fold cross validation. This testing was performed *25* times, for both the Krislet and CMUnited data, using each classifier.

The complete set of collected observations was mined in order to determine the maximum number of objects of each type visible during an observation and the maximum and minimum distance and direction values to use for normalization. The maximum and mean occurrences of each type of object[3], in the Krislet data, are shown in Table 3. Therefore, each stimulus for the Krislet data will be represented by a vector of length *114* (57 object each with a distance and direction value). The CMUnited data is similar to the Krislet data except the maximum number of flags is slightly higher. This is because the CMUnited agents do not use the standard sized field of vision, but instead use a wider field of vision (but this wider field of vision also leads to noisier estimates of the position of objects).

The feature vectors were padded with distance and direction values of *-1* when fewer than the maximum number of objects, of a specific type, were visible. This value was chosen since it will never occur in any visible objects since they are normalized between *0 and 1*.

Lastly, each of the classifiers had its performance optimized using feature selection. Using a methodology similar to that used in previous work [7], the most important types of objects were found for each classifier. Data from any objects that did not positively affect the classification performance was removed. It should be noted that, with only a few exceptions, each classifier selected the same features to use and those features were consistent with our previous studies[4].

[2] Approximately the number of observations collected during a single game.

[3] The objects in simulated RoboCup soccer are: soccer ball, flags, boundary lines, goal nets, teammates, opponents and unknown players (their team is unknown due to noise).

[4] Most classifiers found the ball to be important for Krislet and the ball, goal and flags to be important for CMUnited. However, naive bayes also found teammates to be important for both Krislet and CMUnited and did not find the goal important for CMUnited.

	Ball	**Flag**	**Line**	**Goal**	**Team**	**Opp.**	**Unk.**	**Total**
Max	1	16	1	2	10	11	16	57
Mean	0.6	5.8	1.0	0.5	4.4	3.2	1.8	17.3

Table 3. The maximum and mean occurrences of each type of object in the Krislet data.

4.1 Results

Our results measure the ability of each classifier to predict the action the expert would have performed given a similar stimulus. The possible actions are *kicking*, *dashing* and *turning*. Each action also has associated parameters, like the dashing power, but we ignore those and only look to get the action correct. For each classifier we measure the performance using the *f-measure*. The f-measure, which is a function of the precision and recall of each action, was selected because it is an acceptable metric to use when data is extremely imbalanced (only around *0.2%* of the data is for the kick action).

Table 4 shows the average f-measure values over the *25* tests. Examining the results from Krislet, we can see that both case-based reasoning (CBR) and the decision tree approach perform best. It was expected that the decision tree approach would work well, since the reasoning logic of Krislet can be represented as a decision tree, but it is interesting to note that the CBR approach actually performs slightly better (although not a statistically significant difference).

One other area to consider when comparing the decision tree and the CBR approach is that the CBR approach makes use of a k-nearest neighbour search. During each classification, the CBR approach will need to compare the input to the training instances in order to find similar neighbours. Operating in a real-time environment, like robotic soccer, means that there will be a constraint placed on the execution time of the search which may limit the number of training instances that can be used [7]. The decision tree, which does not employ such an expensive search, might then have a benefit in being able to use more training data than the CBR approach. Figure 2 shows the change in performance of CBR and the decision tree as the size of the training set is increased. From this we can see that the decision tree does not show any noticeable improvement with more training instances whereas the CBR approach does. This shows that the decision tree is unable to surpass the performance of the CBR approach, even with access to more training data.

	CBR	**Decision Tree**	**SVM**	**Naive Bayes**
Krislet	0.83 +/- 0.002	0.82 +/- 0.005	0.70 +/- 0.003	0.66 +/- 0.017
CMUnited	0.61 +/- 0.002	0.42 +/- 0.007	0.47 +/- 0.011	0.16 +/- 0.024

Table 4. The f-measure results using each classifier.

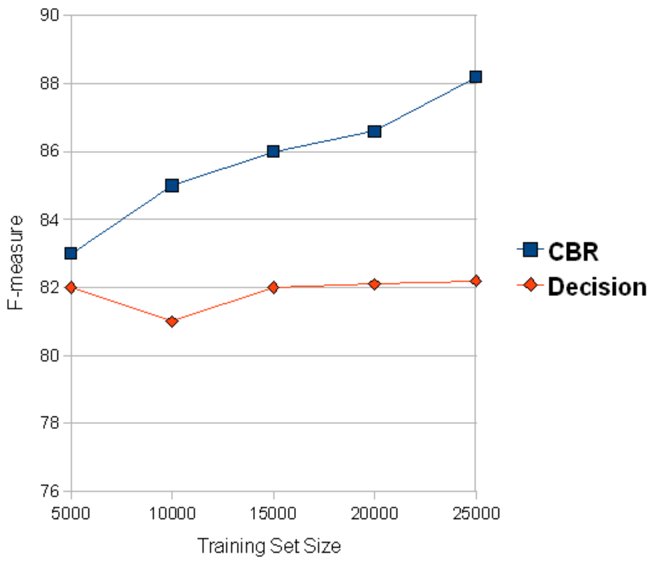

Fig. 2. Change in f-measure as number of training instances is increased.

An initial look at the results in Table 4 indicate that the CBR approach clearly outperforms the other classifiers on the CMUnited data. While we have not thoroughly looked at optimizing the parameters of any of the algorithms, it is promising to see that CBR outperforms the others using default parameters. This is likely due to the complexity of the reasoning used by CMUnited. Whereas the Krislet agent only reasons using the soccer ball and opponent's goal net, the CMUnited agents use a variety of objects. For the balls and goals, there are only ever one or two instances of those objects. Whereas with flag objects there are many of them, so more objects influence the classification process. Additionally, the data does not contain all of the information the CMUnited agents use during reasoning. Things like internal states and inter-agent communication are not included in the data so the stimuli may become more difficult to separate using rules or generalizations. The benefit of CBR is that it keeps the data stimuli unchanged, so important information is not discarded as noise.

Further examination of the results show that the decision trees generated on the Krislet data only contain a few decision nodes. These decisions are quite close to the decisions that the Krislet agent actually uses during reasoning, however it often underestimates the decision bounds resulting in some misclassification during testing. On the other hand, with the CMUnited data the decision tree uses hundreds of decision nodes. The decisions are often not generalizations of the data, but instead make rules that completely describe individual training instances. Looking at the support vector machines and naive bayes, they also do fairly well on the Krislet data but have significant trouble on the CMUnited data. Similar to the decision trees, these approaches tended to over train the class boundaries based on the training instances.

While the results we have presented are not from an exhaustive comparison of all possible classifiers, or all combinations of algorithm parameters, they do show a case-based reasoning approach performs well when learning from agents of various complexities. More importantly, CBR is competitive with a decision tree classifier when learning from an agent who reasons using a decision tree.

5 Conclusions

In this paper we examined the data available when an agent only has a partial view of the world and how the objects in its field of vision can change as the agent moves around the environment. This only gives the agent limited information about what it can see and causes a correspondence problem when attempting to use this information in learning by demonstration systems. The primary reason for this correspondence problem is an inability to uniquely identify objects, so the agent only knowns what *type* of object it sees, rather than *which* object of that type it sees.

Our results show that using case-based reasoning on such agent sensory data outperforms a variety of other classification algorithms when learning by observing a simulated soccer agent. Our experiments did not attempt to perform an exhaustive comparison of all possible classifiers or use all possible algorithm parameters, but instead looked to show that our case-based reasoning system performs well on both simple and complex data and requires no transformation of the data. Even if the agent being learnt from can easily be represented by a set of rules or a decision tree, the CBR approach still performs as well as a classifier more suited to that form of learning.

An additional benefit of an instance based learning approach, like case-based reasoning, is that the input data can be compared with actual stimulus rather than generalizations of the stimulus. This is important when ordering the multi-valued attributes since the objects can be matched using more sophisticated matching algorithms rather than being ordered using a fixed ordering rule (like being ordered by distance to the agent). These findings lead us to believe case-based reasoning is an appropriate technique to use when learning by demonstration as it can be used regardless of the complexity of teacher and allows for the use of all knowledge contained in the training data since no generalization occurs.

References

1. Molineaux, M., Aha, D.W., Moore, P.: Learning continuous action models in a real-time strategy environment. In: 21st International Florida Artificial Intelligence Research Society Conference. (2008) 257–262
2. Auslander, B., Lee-Urban, S., Hogg, C., Muñoz-Avila, H.: Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. In: 9th European Conference on Case-Based Reasoning. (2008) 59–73

3. Ros, R., de Mántaras, R.L., Arcos, J.L., Veloso, M.M.: Team playing behavior in robot soccer: A case-based reasoning approach. In: 7th International Conference on Case-Based Reasoning. (2007) 46–60
4. Watson, I., Rubin, J.: Casper: A case-based poker-bot. In: 21st Australasian Joint Conference on Artificial Intelligence. (2008) 594–600
5. Powell, J.H., Hauff, B.M., Hastings, J.D.: Evaluating the effectiveness of exploration and accumulated experience in automatic case elicitation. In: 6th International Conference on Case-Based Reasoning. (2005) 397–407
6. Floyd, M.W., Esfandiari, B., Lam, K.: A case-based reasoning approach to imitating robocup players. In: 21st International Florida Artificial Intelligence Research Society Conference. (2008) 251–256
7. Floyd, M.W., Davoust, A., Esfandiari, B.: Considerations for real-time spatially-aware case-based reasoning: A case study in robotic soccer imitation. In: 9th European Conference on Case-Based Reasoning. (2008) 195–209
8. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-based planning and execution for real-time strategy games. In: 7th International Conference on Case-Based Reasoning. (2007) 164–178
9. Romdhane, H., Lamontagne, L.: Reinforcement of local pattern cases for playing tetris. In: 21st International Florida Artificial Intelligence Research Society Conference. (2008) 263–268
10. Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: Fourteenth International Conference on Machine Learning. (1997) 12–20
11. Coates, A., Abbeel, P., Ng, A.Y.: Learning for control from multiple demonstrations. In: 25th International Conference on Machine Learning. (2008) 144–151
12. Thurau, C., Bauckhage, C.: Combining self organizing maps and multilayer perceptrons to learn bot-behavior for a commercial game. In: Proceedings of the GAME-ON Conference. (2003)
13. Grollman, D.H., Jenkins, O.C.: Learning robot soccer skills from demonstration. In: IEEE International Conference on Development and Learning. (2007)
14. Chen, Y.L., Hsu, C.L., Chou, S.: Constructing a multi-valued and multi-labeled decision tree. Expert Systems with Applications **25**(2) (2003) 199–209
15. Chou, S., Hsu, C.L.: Mmdt: a multi-valued and multi-labeled decision tree classifier for data mining. Expert Systems with Applications **28**(4) (2005) 799–812
16. Li, H., Zhao, R., Chen, J., Xiang, Y.: Research on multi-valued and multi-labeled decision trees. In: Second International Conference on Advanced Data Mining and Applications. (2006) 247–254
17. Muggleton, S.: Inductive logic programming. Morgan Kaufmann (1992)
18. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artificial Intelligence **101**(1-2) (1998) 285–297
19. Yin, X., Han, J., Yang, J., Yu, P.S.: Efficient classification across multiple database relations: A crossmine approach. IEEE Transactions on Knowledge and Data Engineering **18**(6) (2006) 770–783
20. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2nd edition edn. Morgan Kaufmann (2005)
21. RoboCup: Robocup official site. http://www.robocup.org (2009)
22. Langner, K.: The Krislet Java Client. http://www.ida.liu.se/ frehe/RoboCup/Libs (1999)
23. Stone, P., Riley, P., Veloso, M.M.: The CMUnited-99 champion simulator team. In: RoboCup. (1999) 35–48